A Machine Learning Approach to Wind Estimation and Uncertainty Using Remote Dropsondes and Deterministic Forecasts

by

Mitchell Plyler

Submitted to the Department of Aerospace and Astronautics
in partial fulfillment of the requirements for the degree of
Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

Author .. Signature redacted .....
Department of Aeronautics and Astronautics
May 25, 2017

Certified by ..... Signature redacted .....
Kerri Cahoy
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by . Signature redacted .....
Kai Angermueller
Principal Member of the Technical Staff
Draper Fellowship Advisor

Accepted by ..... Signature redacted .....
Youssef M. Marzouk
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

A Machine Learning Approach to Wind Estimation and Uncertainty Using Remote Dropsondes and Deterministic Forecasts

By

Mitchell L. Plyler

Submitted to the Department of Aeronautics and Astronautics
On May 25th, 2017 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Aeronautics and Astronautics

Abstract

The U.S. Air force currently has a need for high altitude, unguided airdrops without making two passes over a drop zone (DZ). During conventional high altitude drops, aircrews fly over a DZ, release a dropsonde, compute a payload release point, loop back to the DZ, and release a payload. This work proposes a machine learning method that enables a single pass airdrop mission where a dropsonde is released en route to a DZ, the dropsonde measurement is merged with a weather forecast using machine learning methods, and the aircrew releases the payload when they reach the drop zone. Machine learning models are trained to use a deterministic weather forecast and a dropsonde measurement to predict the winds over a DZ. The uncertainty in the DZ wind prediction is inferred using quantile regression. The uncertainty estimate is non-static meaning it is unique for each airdrop mission, and the uncertainty estimate is derived from data that is already available to aircrews. The quantile regression uncertainty estimate replaces the single pass mission's potential need for ensemble forecasts. The developed models are evaluated using data near Yuma, AZ, with later evaluation of several other locations in the US. The machine learning models are shown to improve the accuracy of the wind prediction at the DZ from a remote location up to 117 km away by up to 43% over other methods. To generalize findings, we develop models at several US locations and demonstrate the machine learning methodology is successful at other geographic locations. Models trained on data from a set of DZs are then shown to be transferrable to DZs unseen by models during training. This moves the wind prediction methodology closer to a global solution. The inferred prediction uncertainty is found to reliably reflect the accuracy in the wind prediction. The dynamic wind uncertainty estimate allows for the assessment of mission risks as a function of day-of-drop conditions. For nominal drop parameters, single pass airdrop missions were simulated around the Yuma DZ, and the machine learning methodology is shown to be approximately 20% more accurate than other methods.

Thesis Supervisor: Kerri Cahoy
Title: Associate Professor of Aeronautics and Astronautics

Draper Advisors: Kai Angermueller, Natasha Markuzon

Acknowledgments

This work was supported by an internal investment at Draper. I would like to thank Draper for taking me as a Draper Fellow. In particular, I would like to thank Kai Angermueller for finding a place for me and teaching me about airdrop, Natasha Markuzon for teaching how to conduct a machine learning project, and Dan Chen for keeping the project relevant to the airdrop community's needs. I would like to thank Professor Kerri Cahoy at MIT for her patience and willingness to take on different projects. Thank-you to my family and Maddie for pushing me through.

Contents

List of Figures

List of Tables

List of Procedures

List of terms

DZ Drop Zone
RL Remote Location
CARP Computed Area Release Point
OWAM Operational Wind Assimilation Model
PCA Principal Component Analysis
MSL Mean Sea Level
AGL Above Ground Level
RDSPM Remote Dropsonde Single Pass Mission
RAP Rapid Refresh
ML Machine Learning
RMSE Root Mean Squared Error

1 Introduction

1.1 Single Pass Airdrop Motivation

Airdrop is a process developed to deliver cargo from aircraft to recipients on the ground. Historically, unguided parachutes were attached to cargo and allowed to fall towards the ground from low altitudes. The airdrop community refers to this as a ballistic drop. Payloads hit targets with precision when dropped from these relatively low altitudes, 500-800 feet. Dangers on the ground, e.g. enemies firing at the aircraft, created a demand for high-altitude airdrops, 3,000 to 25,000 feet above ground level (AGL), and commanders still required precision. The Joint Precision Airdrop System (JPADS) is a system created to answer the demand for high-altitude airdrops. The JPADS system utilizes precise wind estimates from a dropsonde instrument, an advanced guidance system, a steerable parafoil, and mission planning software (Benney, et al., 2005). During a guided drop, the guidance system tracks the payloads location and controls its trajectory using the steerable parafoil. The guided JPADS system increases accuracy, but it is expensive. The guided systems are not typically used for low risk or large-scale missions (Gerlach & Doman, 2017). In 2010, more than 99% of payloads by mass were unguided (Bowman, 2011).

When a payload is released from an aircraft, the payload's initial, lateral velocity is equal to the aircraft's, and this causes the payload to drift away from its release point. While the payload falls, the winds pushing on the payload cause it to drift even further from the release point. The mission planning software calculates a release point, accounting for predicted payload drift, that will ideally result in the payload landing on the DZ. This release point is referred to as a Computed Air Release Point (CARP), and its quality is strongly dependent on the quality of the

prior wind estimate. This wind estimate comes from a dropsonde which is a small wind sensor, released from the aircraft, that collects and transmits data as it falls.

During descent, the JPADS guidance and control system can correct for modeling errors, and perhaps more significantly, it can correct for wind estimation errors. Because an unguided system is unable to correct for these modeling and estimation errors, the wind estimate and CARP become more important for ballistic drops. Above 3000 feet AGL, aircrews are required to release a dropsonde before a drop. Below 3000 feet, aircrews can perform the drop based on aircraft wind estimates, or they can release a prior dropsonde when the necessary crewmember is present.

For a better wind estimate, dropsondes are typically released over or very near a target DZ fifteen to twenty minutes before a payload drop (Staine-Pyne, 2009). If the aircraft is flying at approximately 240 knots when the dropsonde is released, the aircraft would travel approximately 70 miles by the time the dropsonde fully transmits the measured wind data. The aircraft then must loiter over the DZ, calculate a CARP based on the collected wind information, prepare the aircraft, and finally release the payload over the DZ. A dropsonde wind measurement over the DZ requires the aircraft to make two passes over the DZ. This pattern could alert enemies on the ground as to when and where the payload will be delivered which creates significant risk (Staine-Pyne, 2009). Precision airdrop is in the position where the two passes demanded by the dropsonde are necessary for accuracy, but in some situations, greatly increase mission risk.

1.2 Currently Proposed Solutions for Single Pass Airdrop

The Air Force has a current need for precise, high altitude airdrops without the dropsonde DZ pass (Staine-Pyne, 2009). Bowman (2011) proposed a mission strategy where a Predator drone would fly over a DZ, launch a dropsonde, collect wind data, and transmit the data to a tailing aircraft. The tailing aircraft would calculate a CARP and release the payload according to the dropsonde wind estimate. The mission would be completed in a single pass of two aircraft; however, the second aircraft would greatly increase the mission cost. The first aircraft could also be a signal to nearby enemies about an incoming airdrop.

Separately, Jones (2016) proposed an aircraft mounted Lidar systems that could remotely sense the winds over the DZ. His work draws from earlier AC130 mounted Lidar systems that were shown to provide quality wind estimates. Work is still being done to show how these Lidar systems can be integrated into the airdrop mission.

There is also work being done with radar-based wind estimation schemes. QinetQ, Welham, MA, has developed a ground based wind sensing radar system for use with airdrop. The system would be operated by personnel on the ground, and the data would be transmitted to aircraft before the payload is released (PADS WiPPR: Wind Profiling Portable RADAR, 2015). This system is stationed on a trailer, and its use is therefore limited to areas the trailer can access. The radar system is also potentially detectable to enemies around the DZ.

Meier (2010) developed a method for estimating winds over a DZ using IR satellite soundings. The wind estimation is derived from thermal measurements and are shown to consistently estimate winds above the boundary layer. The planetary boundary layer is the portion of the atmosphere closest to and most influenced by the ground. It can be 1500 meters above the

ground, and for many drops, the wind errors in this space cause the most problems. Meir (2010) left wind estimation within the boundary layer for future work. Both of these radar-based system require instruments that present significant monetary investments, development, and testing.

A better single pass solution would make use of equipment that is already integrated into the JPADS system and data that is currently available to aircrews. This thesis proposes a single pass mission where a dropsonde is released en route of the DZ, and the dropsonde measurement is merged with a prior weather forecast to predict the winds over the DZ. Chapter details the requirements of the single pass mission. Chapter 3 discusses the methodology for training machine learning models that can accurately predict winds over the DZ by learning on a history of forecasts and dropsondes. Parallel to the DZ wind prediction, quantile regression models quantify the uncertainty in the prediction. The wind uncertainty can be used to estimate the probability of a drop missing a DZ or the probability of hitting a protected object. The results in Chapter 4 show this method is an improvement over other possible methods during simulated missions around Yuma, AZ and other DZs. Chapter 5 discusses some necessary future work to verify the methodology and take the solution further.

2 Single Pass Airdrop through Remote Dropsonde Assimilation

This thesis proposes a method for single pass airdrop that uses only equipment and data currently available to aircrews. This chapter discusses the flow of the proposed single pass mission, some standard data assimilation methods drawn from the meteorology community, and the data set used to train the machine learning models.

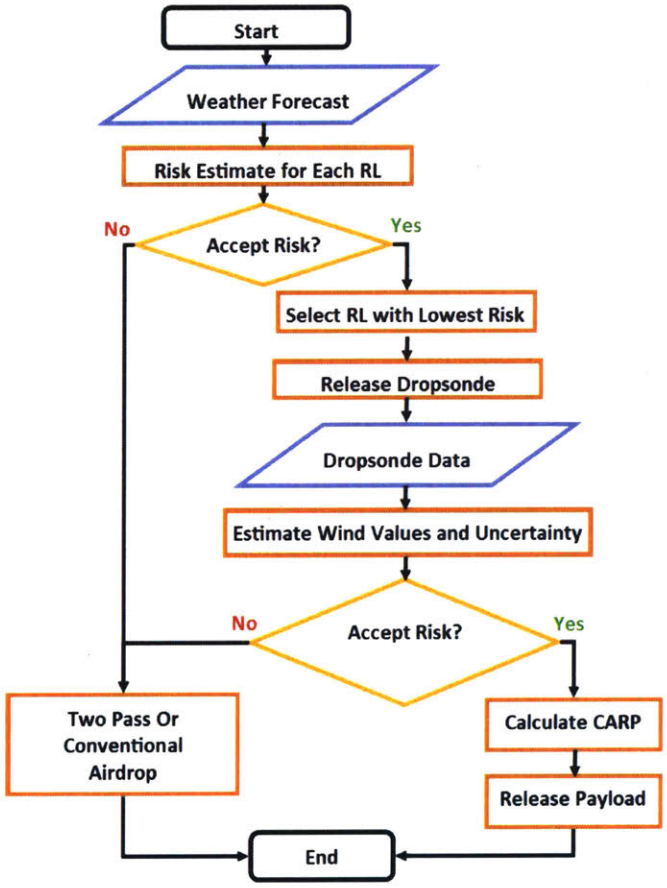

Figure 1:Remote Dropsonde Single Pass Mission.
Mission assessment begins with an estimate, based on a deterministic forecast, of the quality of reach potential remote location for releasing the dropsonde. If the uncertainty estimate is acceptable, a dropsonde is released en route of the DZ at the RL. The dropsonde is assimilated for an updated DZ wind estimate with an associated uncertainty. Based on the new uncertainty estimate, a mission could continue with a single-pass, switch to a conventional two-pass mission, or be aborted.

Figure 1 outlines a proposed single-pass mission where a dropsonde is released en route to the DZ referred to as the Remote Dropsonde Single Pass Mission (RDSPM). The mission would begin with a recommendation for the remote dropsonde location. This recommendation, for each potential dropsonde location, would be a predicted miss distance distribution for a payload released at the DZ. If one accepted the uncertainty estimate for a particular remote location, the dropsonde would be released at that remote location and assimilated. The result would be in an estimate of the winds over the DZ and an updated miss distance distribution. Based on the new uncertainty estimate, an aircrew could decide if they should continue the single-pass mission, switch to a conventional two-pass drop, or completely abort the mission

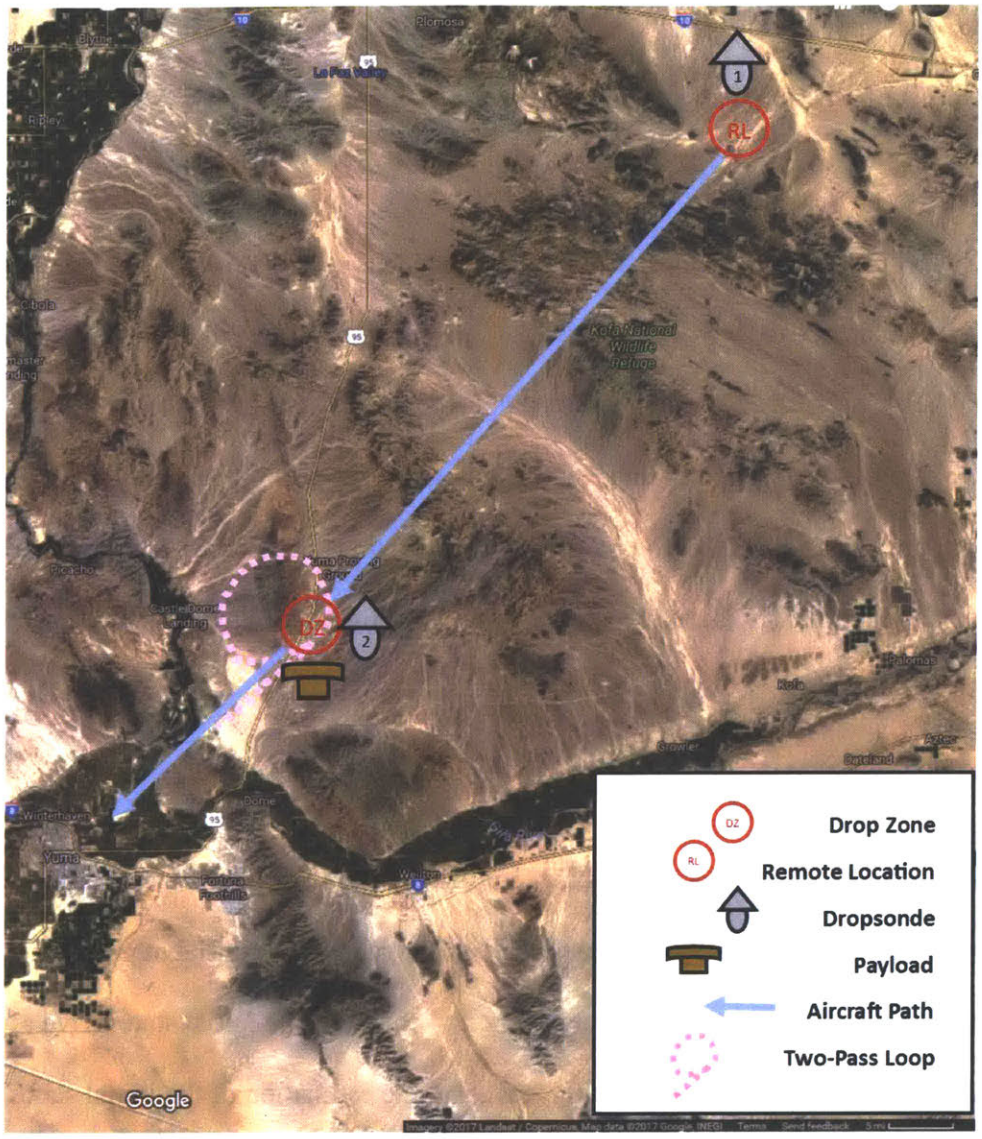

Figure 2: Airdrop Flight Paths
Compares the flight path for a single pass and double pass airdrop. This shows an example DZ and RL. In the single pass mission, the dropsonde would be released at the remote location (1), and in the two pass mission, the dropsonde would be released at the DZ (2). The two-pass loop in pink shows the extra flight path for a two-pass drop. (Google Maps, 2017)

Figure 2: Airdrop Flight Paths illustrates the aircraft flight paths for single and two pass missions. During the single pass case, the dropsonde would be released only at a remote location

(dropsonde 1) and not over the DZ (dropsonde 2). This map shows an example remote location 92 km northeast of a DZ on Yuma Proving Ground in Arizona. The 'Two-Pass Loop' in pink highlights the added risk of the two pass mission.

The RDSPM relies on two estimates. The first estimate, based only on a weather forecast, is an uncertainty assessment for each potential dropsonde location before a dropsonde is released. The winds over the DZ, after a dropsonde is released, is the second estimate necessary for a RDSPM. This thesis addresses the second estimate and leaves the assessment of each remote location for future work. The assessment of each potential dropsonde location is dependent on the method used for wind prediction, so its development naturally follows after a wind prediction method is established.

2.1.1 Weather Data Assimilation Methods

Numerical weather prediction, or forecasting, predicts future atmospheric states based on initial atmospheric conditions. The problem is essentially a propagation of initial atmospheric states through differential equations modeling atmospheric behavior. The initial states have a significant impact on the accuracy of the forecast, and the estimation of the initial states is an active research area (Navon, 2009). Typically, meteorologists blend prior state estimates, e.g. forecasts, with atmospheric measurements to create updated estimates of the atmospheric state. The updated estimates are used as initial conditions for future forecasts. The processes of blending priors with measurements is referred to as data assimilation, and the posterior state estimate is called an analysis (Barker, et al., 2012).

In the past, all analyses were created by hand and were subjective to a person's experience and judgment. Some situations still call upon these 'subjective' analysis (United Surface Analysis

Manual). Researchers created methods to assimilate observations automatically. These methods are termed objective analysis and have been around for many years. Some of the earliest objective analysis schemes are successive correction methods (Navon, 2009). These methods, (Bergthorsson & Döös, 1955) and (Cressman, 1959), iteratively updated a blend of forecasts and climatology with observations. The Barnes (1964) scheme is a successive correction method were the weighting of the observations is defined by the distance between the observation and a location on the background grid. Procedure 1 shows the method from (Barnes, 1964) as it is adapted for airdrop mission planning to assimilate dropsonde measurements with wind forecasts. It is an iterative method that updates a three dimensional grid of forecasts with a dropsonde measurement with weights that vary with horizontal distance h, vertical distance v, and difference in time, t, between the forecast grid and the dropsonde measurement.

Procedure 1: Barnes Interpolation

This algorithm is iterative. m is the iteration number

1. *In the first iteration, initialize a three dimensional grid, over i, j, and w, of prior estimates for iteration $m = 0$, from the forecast grid, $f^{p}_{i,j,w}$:*

$$f_{i,j,w}{}^{m=0} = f^{p}_{i,j,w}$$

2. *In each iteration, m, Calculate an error, e_k, between the k^{th} observation, a_k, and, f^{m}_{k}, the a priori estimate of the kth observation interpolated from this iteration's prior grid, $f_{i,j}{}^{m}$, to the kth observation's location:*

$$e_k = a_k - f^{m}_{k}$$

3. *Make a correction only to each prior grid point at (i,j) with at least two measurements within a radius, D, using:*

$$w_{k,h} = exp\left(-\frac{R_{k,h}^2}{D_h^2}\right)$$

$$w_{k,v} = exp\left(-\frac{R_{k,v}^2}{D_v^2}\right)$$

$$w_{k,t} = exp\left(-\frac{R_{k,t}^2}{D_t^2}\right)$$

$$f_{i,j,w}^{m+1} = f_{i,j,w}^{m} + \frac{\sum_{k=1}^{N} C\, w_{k,h}\, w_{k,v} w_{k,t} e_k}{\sum_{k=1}^{N} w_{k,h}\, w_{k,v} w_{k,t}}$$

If two measurements are not within a radius D, no update is made to that grid point. R_k is the distance from the observation to the grid point. D is a specified scan radius constant centered on the grid point. The subscript h denotes horizontal position, v denotes vertical position in log pressure, and t denotes position in time. C is a measurement weighting constant.

4. *Repeat steps 2 and 3, until five iterations have completed or the difference in all successive updates is less than 1e-3 m/s.*

The influence of the measurement is dispersed to grid points according to the weight function, w_k, which is specified by the parameter D. The weight function approaches zero asymptotically as the distance from the measurement to a grid point increases which agrees with intuition that measurements very far from a grid point should have no influence on the prior (Barnes, 1964). This method can be used for any weather variable, and it could be used to estimate DZ winds using a forecast and remote dropsonde. But for this study, it is only relevant for updating wind measurements.

The Barnes scheme from Procedure 1 is used as a baseline comparison to the methods developed in this thesis. The constants used in the baseline procedure are shown in Table 1:

Table 1: Barnes Interpolation Constants

D_h	240,000 m
D_v	.015 ln(Pa)
D_t	1 second
C	53.16

The current state of the art in objective weather analysis has moved from successive correction methods into 4-D VAR. The 4-D VAR system uses an optimal control approach to finding a blend of past forecasts and observations that minimizes the squared error between the analysis field and the observations. These schemes typically use an adjoint atmospheric model to estimate the gradient of the squared error, and this gradient enables a variety of optimization techniques (Cacuci, Navon, & Ionescu-Bujor, 2014). These methods are typically the most computationally expensive, and they are used in weather forecasting and assimilation systems that receive continuous streams of observational data (Barker, et al., 2012).

Weather ensemble based methods are tangential to the 4-D VAR systems. Evensen (1994) introduced the ensemble Kalman filter as a Monte Carlo based data assimilation method. The ensemble Kalman filter is an adaptation of the classic Kalman filter that makes Bayesian updates to vectors of Gaussian random variables (Kalman, 1960). With Kalman filters, the state update relies on a background error covariance matrix and an observation error covariance matrix. The ensemble Kalman filter quantifies the background error covariance as the sample distribution of an ensemble set of weather forecasts. The ensemble forecast is typically created by propagating a set of initial states through an atmospheric model. Evensen (2003) details a methods for creating the set of initial states and outlines an analysis procedure for the ensemble Kalman filter. There are some memory and computational burdens associated with the ensemble Kalman filter. The

ensembles or the covariance must be stored on disk, and an inverse in the analysis update must be performed.

The proposed single pass mission uses a dropsonde measurement far away from the DZ to make a prediction of the winds over the DZ. When forecasts are available prior to the dropsonde, this can be framed as an analysis, and one of the many data assimilation methods, like Barnes, 4D-VAR, or the ensemble Kalman filter, could be used to estimate the winds over the DZ. Clearly, each of the methods discussed thus far has different data and computational requirements. The Barnes method is relatively cheap and only requires a deterministic forecast and a dropsonde measurement. The 4D-VAR method requires a deterministic forecast, a dropsonde measurement, and an adjoint atmospheric model. The ensemble Kalman filter requires an ensemble of forecasts and the dropsonde measurement.

The Barnes scheme is the simplest method and requires the least amount of information. In fact, it is the basis for the assimilation methods used in airdrop mission planning today. The downside of the Barnes scheme is that it has no intrinsic method for choosing a remote dropsonde location. Based on the Barnes distance weighting method, one would have no method for choosing one dropsonde location that is 50 km from the DZ over different potential dropsonde locations that are also 50 km away from the DZ. The Barnes scheme also does not include a method for determining the uncertainty of the analysis. One could assume a forecast and measurement uncertainty and propagate them through the Barnes scheme, but the uncertainty would be static for these assumptions. The uncertainty in the final wind estimate would be predetermined by the distance between the DZ and RL, the assumed forecast error, and the assumed measurement error. The uncertainty would not take into account any of the information in the forecast or

dropsonde measurement, and it is to be expected that an uncertainty estimate using this information should be more meaningful.

4DVAR requires an execution of the derivative atmospheric model which typically has upwards of 10^7 model states (Navon, 2009). It is unlikely the onboard laptops used for airdrop data assimilation and mission planning could make a 4DVAR analysis in the time between dropsonde release and arrival at the DZ.

The ensemble Kalman filter is potentially well suited for the remote dropsonde mission. One could choose the optimal remote location, for some mission constraints, that would minimize the final analysis uncertainty. This estimate could be derived directly from the ensemble forecast error covariance, and would solve the problem of choosing the remote dropsonde location. After the dropsonde measurement, the ensemble Kalman filter can be used to estimate of the winds over the DZ, and it provides an estimate uncertainty. The issue with the ensemble Kalman filter is the large data requirement. To make the posterior analysis, the filter requires either the forecast covariance matrix or the complete set of forecast ensemble members on the mission planning laptop. In theater, download bandwidth is limited, and for the foreseeable future, it is unlikely mission planners will be able to download and use ensemble forecasts. This eliminates the ensemble Kalman filter from consideration for the single pass mission.

2.1.2 Machine Learning as Weather Data Assimilation Method

For most precision airdrops, planners are able to download a deterministic weather forecast which provides a single estimate of the atmospheric state. Because of the deterministic forecast, it follows that the Barnes method is the current assimilation method used for airdrop, and this thesis presents the potential accuracy of using this method as a baseline. The Barnes scheme

requires some hard assumptions like static forecast and measurement errors and a distance defined weighting. The Barnes weighting scheme states that every dropsonde released 50 kilometers from a DZ should be weighted exactly the same regardless of the terrain, weather patterns, and observed data. Because of variations in terrain and atmospheric conditions, it is unlikely this methodology will work for distant dropsondes at all places and times. This motivated us to think about different methods for making an assimilation for the very specialized remote dropsonde mission. This methodology would use a deterministic forecast and a remote dropsonde measurement to predict the winds over the DZ.

One can view the wind assimilation task as a function F that takes an input vector x and predicts a response variable y. Where the function F is the assimilation method, the input vector x is made up of forecast and measurement data, and the response variable y is the true wind over the DZ. If we propose some true assimilation method, F^*, that makes a perfect estimate every time, we can try to approximate this function with F.

Often, the best platform for function approximation tasks is found in machine learning. In the weather literature, Radhika & Shash (2009) used Support Vector Machines to forecast future max temperature values. Markuzon (2012) used Random Forests to predict landslides from weather and terrain information. Krasnopolsky & Fox-Rabinovitz (2010) used neural networks to parameterize cloud coverage in the atmosphere. Each of these studies solved problems that were not being solved by traditional statistical or meteorological methods alone. Machine learning methods were able to recognize patterns in a dataset and make useful estimations. The remote dropsonde mission requires an estimate of the DZ winds with 'day-of-drop' uncertainty using only a deterministic forecast and a remote dropsonde.

The data assimilation methods discussed thus far fall short for different reasons. The 4D-VAR system has too much computational cost. The Barnes scheme cannot provide a 'day-of-drop' uncertainty. The ensemble Kalman filter relies on ensemble forecasts, and aircrews are not able to download that much data. For these reasons, the methodologies discussed in Chapter 3 were developed to make the DZ wind prediction and infer its uncertainty.

Machine learning methods use a training dataset, X, and learn to predict a response, Y. This thesis details a method for building assimilation models where the input vector, x, is made up of features derived from a deterministic forecast and data collected by a remote dropsonde, and the target variable is the wind over the DZ. The machine learning models are trained on a dataset of past forecast and dropsonde data.

2.2 The Rapid Refresh (RAP) Dataset

This study aims to learn models on a dataset of past forecasts and dropsondes to predict the winds far away from the dropsonde measurement. The forecast dataset that was used, the Rapid Refresh (RAP) model, is available from the National Oceanic & Atmospheric Administration (NOAA). The Rapid Refresh model is a continental-scale forecast and analysis system that is updated hourly. Every hour, observations are merged with a prior forecast to create an analysis, and the analysis is used as the initial condition to create hourly forecasts up to eighteen hours in the future. The RAP system began operation on May 1st of 2012 and is still in operation. It operates on a thirteen kilometer grid that spans North America and covers thirty-seven atmospheric pressure levels (NOAA, 2017).

To simulate operational forecasts, forecasts were subsampled to a 20$x20 grid point region centered on a DZ. This is a 260x260 km region that captures the local weather patterns around a

DZ. This size is also similar to forecasts used operationally, and it is large enough to contain potential remote dropsonde locations. The RAP forecast grid resolution, 13 km, is similar to those available from the Air Force Weather Agency and the UK MET Office which are actually used. The RAP forecasts have many weather variables. Because not all of these variables are available in operational weather forecasts, only the east wind component, the north wind component, the vertical winds, temperature, relative humidity, and geopotential height are used. More on this is shown in Table 4. Because there is no publicly available dropsonde database, the remote dropsonde and the true winds over the DZ are simulated with the analysis from the RAP dataset.

3 Methodology for wind estimation and uncertainty quantification

3.1 Approach to training learned models

Chapter 2 shows the procedure for the remote dropsonde single pass mission. The mission begins by predicting, for each remote location, the accuracy of a drop if a dropsonde were to be released at that remote location. If the estimated accuracy associated with a RL is acceptable, a dropsonde is released at that RL, assimilated, and an uncertainty estimate is used to decide whether or not to continue the mission. This chapter presents a machine learning method for assimilating the dropsonde data to predict the winds over the DZ.

Machine Learning algorithms typically use a data set with many samples of input and output to fit a model that can be used on future data. The models developed in this chapter take an input feature set derived from the data in a deterministic forecast and a dropsonde released away from the DZ. They provide a mean estimate of the winds and a set of discrete quantile estimates that are used to produce an error distribution function. Because there is no large enough dataset of dropsondes, the dataset used in this study is made up of deterministic forecasts and matching analysis. The analysis are generally a good proxy for truth, so they are used to simulate dropsondes at the remote location and the winds over the DZ.

Grim et. all (2015) conducted a study investigating if remote measurements were correlated with measurements over a DZ. They chose one balloon sounding site as their DZ. They found the correlation between the winds measured by the balloon at the DZ and winds measured by balloons at remote locations. They concluded the winds at a RL were sufficiently correlated with winds at the DZ to justify research into using remote dropsondes to predict winds at a DZ. They noted the balloon sounding dataset to not be representative of all RLs, DZs, and times, so they

studied if weather analyses could be used as proxies for the balloon soundings. They found correlations between the winds at the DZ and RL balloon sites using data from a set of weather analyses. They concluded that the analyses and balloon soundings produced similar correlations, and that they could build predictive models that target data derived from the analysis dataset. Their conclusion relating analyses to balloon soundings is used to justify training models on pseudo-dropsondes derived from a weather analysis.

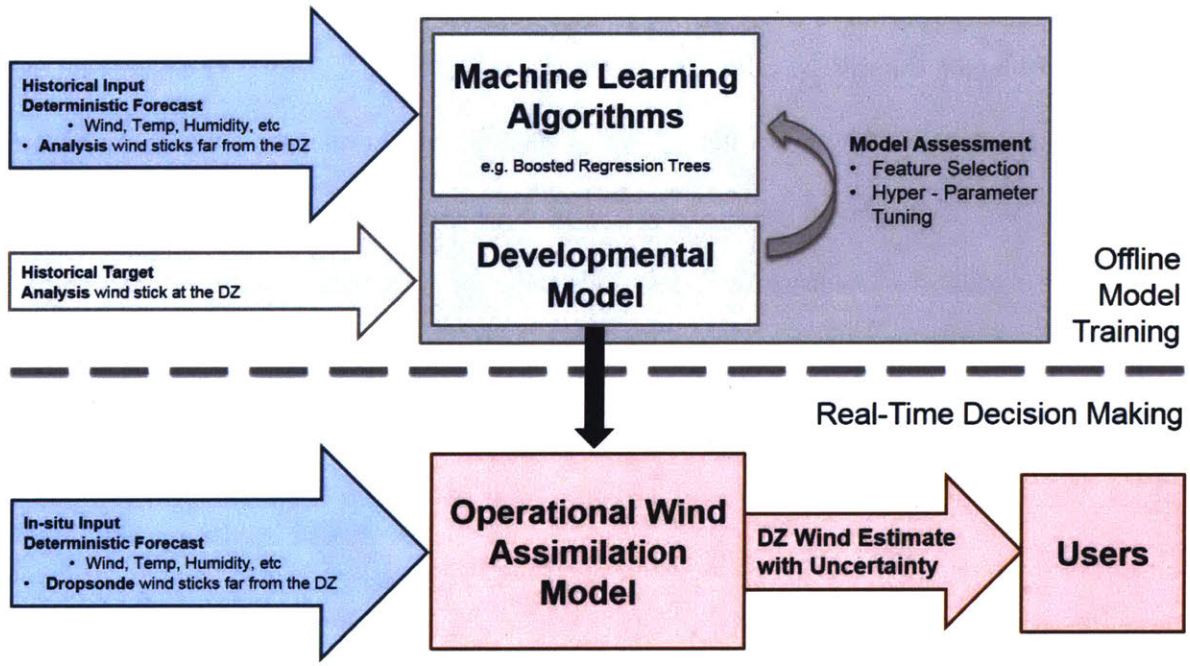

Figure 3: Model Development Block Diagram.
Machine learning models are trained using historical forecasts and analysis. Features are extracted from the entire forecast weather cube and are added to features taken from a pseudo-dropsonde at the RL. Operationally, the model would take the same forecast features used during training and features from an actual dropsonde released at the DZ.

Figure 3 shows the general approach for model training and operational use. Samples of forecast and dropsondes are used as input into preliminary models. The models are trained to predict the analysis winds at the DZ. Model training is an iterative procedure where different feature combinations, parameters, and learning algorithms are tried. Once a model's performance converges and is verified, they could be implemented operationally. The operational model would then take forecasts, as it did before, but would now use actual dropsonde data instead of the analysis wind stick at the RL. The output of the operational model would be an estimate and associated uncertainty of the winds at the DZ. Users could then use this information to update the plan for the airdrop mission. In the Operational Wind Assimilation Model (OWAM), regression models are used to predict the mean winds over the DZ. To quantify the uncertainty of this estimate, quantile regression is used to estimate quantiles of the winds over the DZ. The mean wind estimate is used to find the best Calculated Area Release Point (CARP) for an accurate drop. The uncertainty estimates can help, with other tools, mission planners understand the probability of various drop scenarios.

3.2 Breaking the atmosphere into separate models

The OWAM from Figure 3 uses a forecast and dropsonde measurement to estimate the winds over the DZ. The winds over the DZ are quantified by a north wind component, u, and an east wind component, v, at a discrete set of isobaric pressure levels. The OWAM consists of a set of regression models. Each regression model in the set predicts either the u or v wind component for one isobaric pressure level based on one RL

For a particular weather variable, forecasts are presented as three dimensional arrays spanning an east direction, a north direction, and vertical pressure levels. They can contain many variables including but not limited to: u winds, v winds, vertical winds, relative humidity, temperature,

and geopotential height. The three dimensional arrays for each weather variable are regularly spaced in the east and north directions and irregularly spaced in the vertical direction.

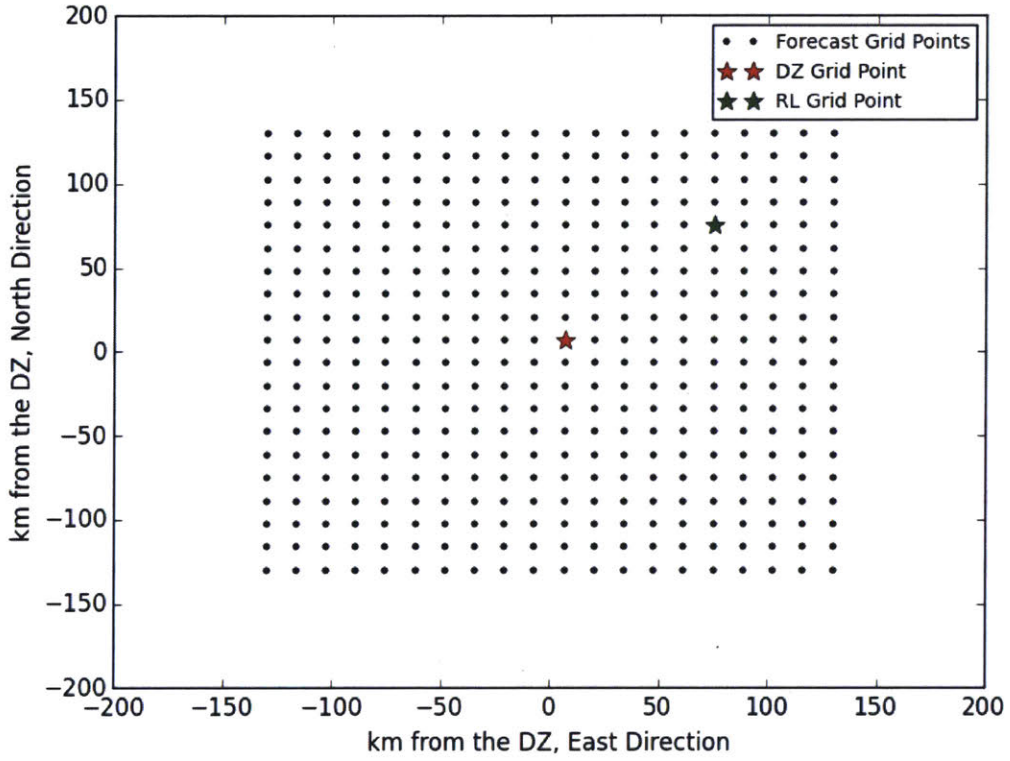

Figure 4: 2-Dimensional Weather Forecast Grid Projection for One Pressure Level.

Figure 4 shows an example of a weather forecast grid for one isobaric pressure level, l. Each dot represents a grid point where the forecast stores data for each weather variable. This forecast grid is centered on a DZ which is shown by the red star. The green star shows a potential remote dropsonde location, (RL_i, RL_j). Using forecast data points from (RL_i, RL_j) and l, the dropsonde measurements at (RL_j, RL_i) and l, and features derived from the entire forecast grid, one regression model, F, predicts a wind component, either u or v, at the DZ location on pressure level l.

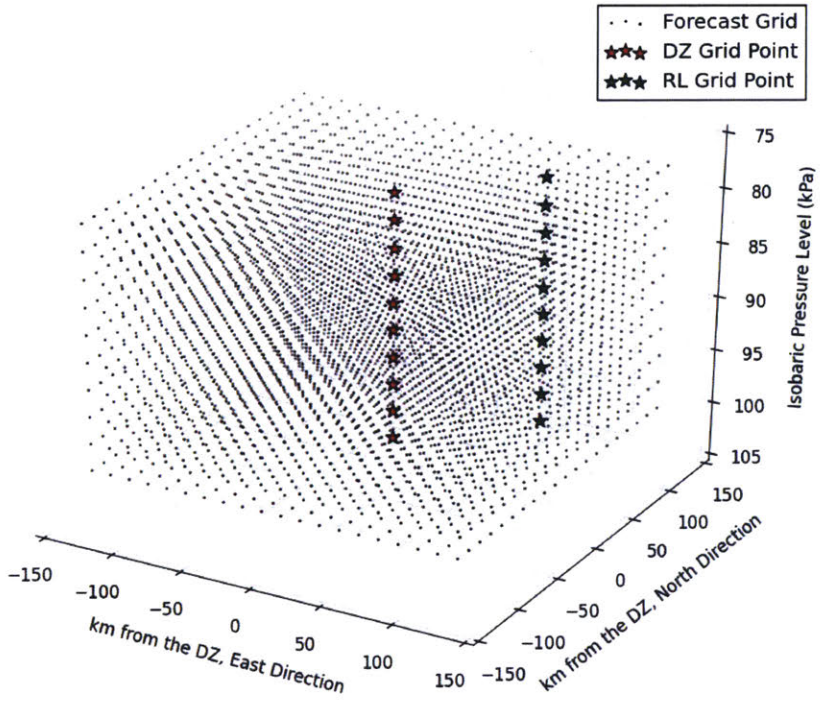

Figure 5: 3-Dimensional Forecast Weather Grid

Figure 5 shows the structure of a forecast weather cube. Figure 4 showed the forecast grid for one pressure level, and Figure 5 shows how the two-dimensional grid structure is repeated vertically at discrete pressure levels. The OWAM predicts the u and v winds at each pressure level over the DZ. In this example, a dropsonde would be released at the upper most green star and allowed to collect wind data at each lower green star. The OWAM would have two regression models for each green star, and each regression model would estimate either the true u or the true v wind component at the red star on the same pressure level.

The weather cube was broken into many different regression models for a variety of reasons. It was hypothesized that the relationships between the winds at one remote location and DZ would

be unique to that remote location. The similarity patterns between winds at the DZ and winds at a RL northeast of the DZ might be very different than the similarity patterns between winds at the DZ and winds at a RL southeast of the DZ. One could train a model using dropsonde data from both the northeast and southeast RLs and include information in the input features that prescribe the dropsonde location. These models would need to not only learn assimilation patterns, but they would have to learn how the dropsonde location affected these assimilation patterns. For any one regression model, the dropsonde location was fixed so the models could better learn patterns that are unique to that location. Section 4.3.2 explores this assumption by evaluating a set of regression models trained at one RL on data from other RLs. The set of models trained on one location are shown to perform similarly on data from other nearby RLs.

Separating the models by pressure level and RL reduces the number of features that used by any one model. For most regression modeling methods, large feature vectors can lead to overfitting (Hastie, Tibshirani, & Friedman, 2009). Input vectors are mostly composed of features derived from local pressure levels. To further limit the length of the input feature vector, a method for generating feature vectors, evaluating their performance, and trimming weakly performing inputs is used. Before feature trimming, a large set of features is derived from the forecast one hour prior to the drop, the forecast at the drop time, the forecast one hour after the drop time, and the dropsonde released before the drop. Section 3.6 details the trimming procedure that is used to reduce the large feature set.

3.3 Training Tree Boosting Models

Predictive models are fit using the Tree Boosting algorithm introduced by (Friedman, 2001). Tree Boosting is a popular, off-the-shelf data mining tool that has several advantages over other algorithms. It is able to optimize over any differentiable loss function, L, and in this study, both

the least squares loss function and the check loss function are used. Tree Boosting is based on regression trees and retains many of their useful characteristics: Tree Boosting is robust to univariate feature transformations, able to handle features of mixed type, and provides an intrinsic method for quantifying the importance of individual features. Tree Boosting is an additive model that seeks to iteratively add basis regression trees.

Regression and decision trees, which form the basis functions of Tree Boosting, are data mining models that take some input feature vector, x, and learn to predict a response y. For decision trees, the response is some categorical random variable, and for regression trees, the response is a continuous random variable. Most tree building algorithms iteratively split on tree nodes according to some minimizing split criterion. One of the most popular, and what is used in Tree Boosting, is the CART algorithm. In the CART algorithm, a regression tree, f, can be seen to partition the response in to M regions $R_1, \ldots, R_M$ (Hastie, Tibshirani, & Friedman, 2009). Where the function response would follow as:

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m) \quad (1)$$

Where I is an indicator random variable returning 1 if x is in region R_m and 0 if not, and c_m is the response for the m^{th} region. CART uses a greedy algorithm to find a splitting variable j and a split point s that minimizes the sum of squares and defines a pair of half-planes, R_1 and R_2.

$$R_1(j,s) = \{X|X_j \leq s\} \; and \; R_2(j,s) = \{X|X_j > s\} \quad (2)$$

The splitting variable j and split point s solve:

$$\min_{j,s}\left(\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right) \quad (3)$$

The inner minimization is solved by the average of the response in each region:

$$\hat{c_1} = ave(y_i|x_i \in R_1(j,s)) \text{ and } \hat{c_2} = ave(y_i|x_i \in R_2(j,s)) \quad (4)$$

A regression tree, h, is denoted by its split points, c, and disjoint regions, R, in the set a.

$$a = \{c, R\} \quad (5)$$

The response, y, of a regression tree is specified by the parameter a and the input feature vector, x.

$$y_i = h(x_i, a) \quad (6)$$

This splitting process is done recursively on tree nodes to grow the depth of the tree. Trees can add more splits until some convergence criterion is met, or more often, some hard stopping point is reached. The stopping point could be a fixed tree depth, a fixed number of terminal nodes, or a maximum number of splits. The stopping point for the regression trees is an important tuning parameter in the Tree Boosting algorithm.

Tree Boosting seeks to approximates some optimal function, $F^*(x) = y$, where y is a response variable and x is an input feature vector. It is a stage-wise algorithm where basis functions, in this case regression trees, $h(x; a)$ are iteratively added to model. The algorithm does this by finding a constraint-based gradient direction, a, through line search that brings the function approximation, F, closer to the optimal function F^*. This gradient direction is made to run parallel to the true steepest descent gradient, g_m. The result is a minimization over a set of pseudo responses, $\{\tilde{y}_i = -g_m(x_i)\}_{i=1}^N$, instead of raw response values y. The general procedure (Friedman, 2001):

Freidman Gradient Boosting Algorithm

$$F_0(x) = argmin_\rho \sum_{i=1}^{N} L(y_i, \rho)$$

For m = 1 to M do:

$$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}, i = 1, N$$

$$a_m = argmin_{a,B} \sum_{i=1}^{N} [\tilde{y}_i - Bh(x_i; a)]^2$$

$$\rho_m = argmin_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i; a_m))$$

$$F_m(x) = F_{m-1} + \rho_m h(x; a_m)$$

end For

M is the number of basis funcitons included in the model, N is the number of training samples, and ρ is the gradient distance taken in the direction a. This algorithm is used with least squares loss function to predict a mean of the winds over the DZ and, separately, with the check loss function to predict quantiles of the winds over the DZ.

The Tree Boosting algorithm implemented in the python package Scikit-Learn was used to fit wind prediction models (Pedregosa, et al., 2011). For this Tree Boosting implementation, there are a number of parameters that must be specified including: the loss function, the learning rate, the number of estimators, the max depth of the basis trees, the number of features used to fit a basis regression tree, and the subsample percentage. Most parameters were chosen during an exploration phase of the study by using cross-validation grid search.

The loss function dictates the final use of the model. A model fit using the least squares loss function predicts a conditional mean of the response variable. A model fit with the check loss function predicts a conditional quantile of the response variable. The learning rate controls the

influence of each individual regression tree. A lower learning rate typically increases the necessary training time but tends to increase generalizability. The necessary training time is directly related the number of basis functions that are used. When using Tree Boosting, the number of regression trees, estimators, was allowed to be dynamically determined. The number of features used to fit a basis tree, 'Feature %', is another tunable parameter where a value less than the total number of features, n, generally increases bias while decreasing variance.

Friedman (2002) details a modification of the original Tree Boosting algorithm called Stochastic Gradient Boosting. In the original procedure, each basis regression tree has access to the entire training data set when fitting. In the stochastic variant, random training samples are drawn without replacement for each individual regression tree, $h_j(x, a_j)$, fit procedure. Adding this randomization tends to produce more generalizable models. The percentage of available samples is a tunable parameter referred to as 'Subsample %' that can affect the quality of the models.

After a period of cross-validated trial and error, models were fit using the parameters:

Table 2 Scikit-Learn Gradient Boosting Regressor Parameters

Loss Function	Least Squares and Check
Learning Rate	.01
# Estimators	Algorithmically Determined
Max Depth	5
Split Criterion	Least Squares
Subsample %	70
Feature %	60

The Tree Boosting algorithm adds one basis function at each iteration. The fitting process is monitored by evaluating the loss on the validation set after every 250 iterations and saving a copy of the model. The model stops adding basis functions after three consecutive increases in

the validation loss score. The model version with the lowest validation score is kept. This method of monitoring the performance on the validation set is used to control against over fitting. When models continue to minimize a loss function, either least squares or the check loss function, on the training set, eventually the model will fit too closely to the data it has observed. The validation set gives an idea of how a model will generalize to unseen data, so it is useful to stop fitting the model when the performance on the validation set is decreasing. This method detects a decrease in performance on the validation set as being a decrease in performance over three sets of 250 iterations. The three sets of 250 iterations was chosen because it was observed to almost surely take the validation error out of a local minimum.

3.4 Estimators for winds at DZ

The Friedman's Tree Boosting algorithm shown in Section 3.3 is used to fit models that predict the winds over the DZ. These models take input features from the forecast and remote dropsonde and predict a conditional mean of the winds over the DZ. The algorithm is fit using the least squares loss function.

The least squares loss function is often the standard choice is regression analysis. This is because of both mathematical convenience and the usefulness of a mean estimate (Koenker, 2000). Many predictive models are interested in the mean value of a response variable. Because of this, they rely on the least squares loss function to produce of conditional mean estimate. This can be seen in that a population's mean, μ, minimizes the least squares cost function.

$$\min_{\mu \in R} \sum_{i=1}^{n} (y_i - \mu)^2 \quad (7)$$

Linear regression seeks to estimate a response variable, y, with some inputs, X, with a vector of parameters β.

$$\hat{y} = \beta X \quad (8)$$

For standard linear regression, least squares can be viewed as producing a conditional mean estimator. The β parameters are estimated using least squares.

$$\min_{\mu \in R} \sum_{i=1}^{n} (y_i - x_i'\beta)^2 \quad (9)$$

The Tree Boosting algorithm builds a nonlinear model that takes an input feature vector and predicts a conditional mean of the response function. This estimate will have some accompanied uncertainty, and the next section discusses separate Tree Boosting models that estimate conditional quantiles of the prediction error.

3.5 Estimators for uncertainty of winds at DZ

For airdrop, mission planners are just as interested in the uncertainty of an estimate as they are in the estimate itself. Most plans are formulated based on the risk of an unsuccessful drop. Wind estimation errors, that may seem intuitively small, can push a payload far away from a drop zone. A useful prediction of the winds over the DZ should be accompanied by an uncertainty in that prediction. Ideally, this uncertainty will be 'day-of-drop' meaning that it will be unique for every planned mission, and will be dependent on all the available information. Some atmospheric conditions will be easier to predict than others, so a useful model should be able to recognize these conditions. The meteorological community solves this issue with ensemble forecasts. These are sets of atmospheric states, and the sample variance from this set can be used to quantify a forecast's uncertainty. The ensemble forecasts are too large to download, so

quantile regression is used to infer the uncertainty in the wind prediction. Quantile regression provides a day-of-drop error CDF that is both tight and reliable.

Quantile regression uses the check loss function to produce a conditional quantile estimate of a response variable. (Koenker, 2000) has a detailed explanation of the procedure.

I is an indicator random variable taking value 1 when a condition 1 is met and value 0 when the condition is not met.

$$I = \begin{cases} 1 \; if \; u < 0 \\ 0 \; if \; u > 0 \end{cases} \quad (10)$$

For a random variable, X, the CDF is defined as:

$$F(x) = P(X \leq x) \quad (11)$$

The inverse CDF is defined as:

$$F^{-1}(\tau) = \inf\{x : F(x) \geq \tau\} \quad (12)$$

This shows a quantile of X, at τ, is defined as the value of X at which $\tau * 100\%$ of the probability space is less than $F^{-1}(\tau)$. If one were to draw random samples from X, they should expect $\tau * 100\%$ to be less than the value at $F^{-1}(\tau)$. Koenneker shows that the estimation of quantiles for X can be framed as an optimization problem where the check loss function, ρ_τ, is minimized.

$$\rho_\tau(u) = u(\tau - I(u < 0)) \quad (13)$$

For this exercise, one seeks the value, $\hat{x}$, of X at the quantile τ. The check loss function uses:

$$u = (X - \hat{x}) \quad (14)$$

Like most optimization procedures, the expected loss is minimized.

$$E[\rho_\tau(X-\hat{x})] = (\tau-1)\int_{-\infty}^{\hat{x}}(x-\hat{x})dF(x) + \tau\int_{\hat{x}}^{\infty}(x-\hat{x})dF(x) \quad (15)$$

Differentiated with respect to $\hat{x}$,

$$0 = (1-\tau)\int_{-\infty}^{\hat{x}} dF(x) - \tau\int_{\hat{x}}^{\infty} dF(x) = F(\hat{x}) - \tau \quad (16)$$

Any element of $\{x: F(x) = \tau\}$ minimizes the expected loss because F is monotone. A proof of this is shown in (Manski, 1988). To find the τ^{th} sample quantile through optimization, one seeks a parameter ξ from the set of reals, R, that minimizes:

$$\underset{\xi \in R}{min} \sum_{i=1}^{n} \rho_\tau(y_i - \xi) \quad (17)$$

through a linear program with $2n$ slack variables v. The new problem is

$$\underset{(\xi,u,v) \in RxR_+^{2n}}{min} \{\tau 1_n' u + (1-\tau)1_n' v \mid 1_n'\xi + u - v = y\} \quad (18)$$

where 1_n is a vector of ones with length n.

Up till now, Konneker has shown us that one can find the quantiles of a random variable through a linear program with the check loss function. It was shown that the conditional mean is a solution to the least squares loss function. Quantile regression can now be shown to produce a conditional quantile estimate in a similar way. Define $\hat{\alpha}$ as the τ^{th} sample quantile. This value solves the minimization:

$$\underset{\alpha \in R}{min} \sum_{i=1}^{n} \rho_\tau(y_i - \alpha) \quad (19).$$

This leads to quantile regression which allows one to estimate a conditional quantile function $Q_y(\tau|x)$ for a given input variable x and model parameter β.

$$Q_y(\tau|x) = x'\beta(\tau) \quad (20)$$

This is the solution to:

$$\min_{\beta \in R^{\rho}} \sum_{i=1}^{n} \rho_\tau(y_i - x_i'\beta) \quad (21).$$

To go back to the linear program formulation:

$$\min_{(\beta, u, v) \in R^{\rho} x R_+^{2n}} \{\tau 1_n' u + (1-\tau) 1_n' v \,|X\beta + u - v = y\} \quad (22)$$

This quantile regression linear program can be used during the Tree Boosting Algorithm presented in 3.3. Using the same feature vector, x, as was used when predicting the mean response, check loss function is used to predict conditional quantiles of the winds at the DZ.

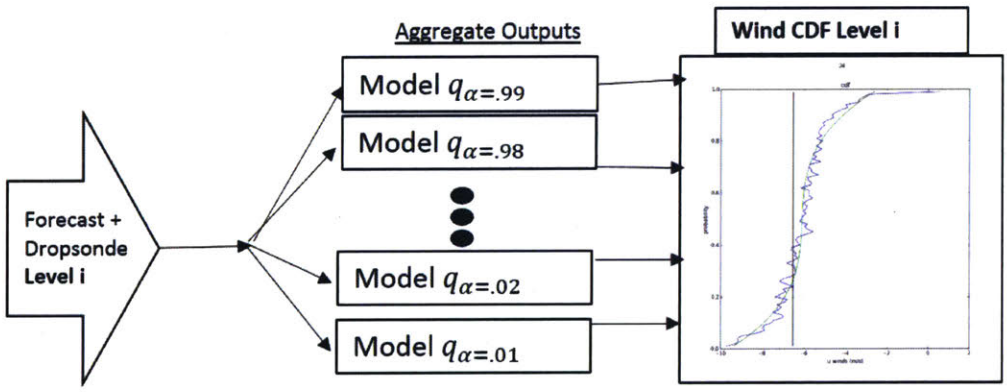

Figure 6 Wind CDF Structure for Level i.
Forecast and RL dropsonde features are used as inputs to a set of quantile regression models that each estimate a discrete quantile of the response variable. The aggregate of the quantile function responses is a CDF of the winds over the DZ that represents the prediction uncertainty. This procedure is repeated for each pressure level of interest.

Figure 6 shows the general structure for how quantile regression is used to quantify the prediction error. For one pressure level, quantile regression models are fit at a set of discretized quantiles. These models each take the same input feature vector but predict different conditional quantiles of the possible values for the winds over the DZ. The aggregate of the predictions for these models is a conditional CDF for the winds on that pressure level. A wind error CDF could be produced by subtracting the mean response estimate from the CDF produced in Figure 6.

The procedure outlined in Figure 6 is repeated for each pressure level. The output of all these quantile regression models is the CDF for the winds on each pressure level over the DZ.

3.6 **Feature Generation and Selection**

Feature selection was done through a generate and test procedure. When predicting the winds over the DZ, the system has access to forecasts and dropsonde data. After the forecast is subsampled to the 260 x 260 kilometer region, there are roughly 10^5 data points within the forecast. There is too much available data to use the forecast without preprocessing. Intuitively, the information in the forecast is of varying degrees of relevancy. This study uses a combination of guessing and automatic feature selection. Features are extracted, which may or may not be helpful, from both the available forecast and dropsonde. This section shows how the feature vector is generated, evaluated, and trimmed to find a set of features that best predict the winds over the DZ. Section 3.2 discussed how regression models are created for each pressure level, remote dropsonde location, and target wind component . For a drop at the maximum altitude, thirty pressure levels of models would be used.

3.6.1 Feature Generation

3.6.1.1 Forecast variables

From the forecasts, this feature generation methodology takes raw variables that are most likely to be in most available numerical weather forecasts. This study included:

Table 3 Forecast Variables

Relative humidity
U winds
V winds
Vertical winds
Geopotential height
Temperature

For each of these variables, a feature is extracted using the following method:

Table 4 Features Generated from Forecast Variables

Value at the sonde location and on the pressure level
Value at the DZ location and on the pressure level
Mean value of the stick at the sonde location
Mean value of the stick at the DZ location
Mean value on the pressure level
Variance of the stick at the sonde location
Variance of the stick at the DZ location
Variance of the values on the pressure level

At this point, a feature vector of length 48 is created by taking the cross-product of Table 2 and Table 4.

3.6.1.2 PCA Features

PCA vectors based on the U and V wind grids are concatenated with the prior feature vector. This is an important dimensionality reduction step that allows the model to capture some of the variability in the forecast grid. The PCA vectors are created by starting with a grid for one wind component of shape $(F,\ L,\ J,\ I)$, where F is the number of forecasts in the data set, L is the number of pressure levels in a forecast, J is the number of forecast grid points in the northern direction, and I is the number of grid points in the eastern direction. Note that both J and I are centered on the DZ grid point. The wind grid of shape $(F,\ L,\ J,\ I)$ is reshaped into two dimensions to become $(FL,\ JI)$. At this point, a PCA function can be fit using the methods discussed in Hastie, Tibshirani, & Friedman (2009). A PCA function for each wind component is fit using winds strictly in the training set. These PCA functions can then reduce the wind grid into a smaller vector length which makes training easier. At this point, a PCA vector for both U and V each of length P. Concatenating this with the previous feature vector, this brings the total feature count to $48 + 2P$.

3.6.1.3 Finite Difference Time Features

This methodology assumes and requires forecast information one hour before and one hour after a drop time. Based on the way drops are normally planned and conducted, it is almost always true that when forecasts are available, there will be forecast data one hour before and one hour after a drop time. The feature vector of length $48 + 2P$ is now used generate a discrete time derivate for each feature. This is done using a central difference.

$$
\begin{aligned}
v_t &= \ feature\ vector\ at\ drop\ time\ t \\
v_{t+1} &= \ feature\ vector\ at\ drop\ time\ t + 1\ hour \\
v_{t-1} &= \ feature\ vector\ at\ drop\ time\ t - 1\ hour \\
z_t &= discrete\ time\ feature\ vector\ at\ time\ t
\end{aligned}
$$

$$z_t = v_{t+1} - v_{t-1} \quad (23)$$

The discrete time derivative features are then concatenated with the original feature vector. The total feature vector is now of length $2(48 + 2P)$.

3.6.1.4 Dropsonde Features

Features derived from the dropsonde data which, not surprisingly, are the most powerful features, are now concatenated with the feature vector. For this method, dropsonde data must be available for each pressure level for both above and below a model for a particular pressure level. This data is leveraged to create features according to Table 5. Note that subscript s denotes data from the dropsonde, subscript f denotes data from the forecast, subscript RL denotes data from the dropsonde location, and subscript DZ denotes data from the DZ location.

Table 5 Dropsonde Features

Dropsonde u on pressure level	$u_{s,RL}$
Dropsonde v on pressure level	$v_{s,RL}$
Forecast error u	$u_{s,RL} - u_{f,RL}$
Forecast error v	$v_{s,RL} - v_{f,RL}$
DZ forecast error corrected u	$u_{s,RL} - u_{f,RL} + u_{f,DZ}$
DZ forecast error corrected v	$v_{s,RL} - v_{f,RL} + v_{f,DZ}$

Notice that 'DZ forecast error corrected u' and 'DZ forecast error corrected v' are created by adding the forecast error at the remote location to the forecast estimate at the DZ. These features use the assumption that if the forecast is wrong by some amount at one location then it must be wrong by the same amount everywhere else. This is obviously not true, but it is a good starting estimate to go from. It is also interesting that these features are simply the addition of two other included features. A learned model could ideally extract these type of feature interactions on its

own, but in some cases, it is useful to supply the interaction at the start if it is clearly useful. This results in six features from the dropsonde data, and after concatenating it with the prior features, brings the feature vector to $2(48 + 2P) + 6$ total features.

3.6.1.5 Pressure Level Finite Difference Features

A feature vector is generated and a model is trained for each pressure level and each remote location in the forecast cube. So far, features have only been generated from a model's pressure level or from the wind stick. Because there is vertical movement in the atmosphere, it is likely a model for one pressure level could benefit from features extracted from other levels in the weather cube. This information is provided to the model through discrete derivatives with respect to pressure level, z_l.

$$v_l = feature\ vector\ for\ the\ pressure\ level\ of\ interest$$
$$v_{l+1} = \ feature\ vector\ from\ one\ pressure\ level\ above$$
$$v_{l-1} = feature\ vector\ from\ one\ pressure\ level\ below$$
$$z_l = v_{l+1} + v_{l-1}$$

The above method for z_l only works for pressure levels that are not at the top or bottom of the weather cube. If the pressure level of interest, l, is that the top or bottom z_l is found using:

$$l = l_{max}: \quad z_l = v_{l+1} - v_l \quad (24)$$
$$l = l_{min}: \quad z_l = v_l - v_{l-1} \quad (25)$$

A new feature vector is formed by concatenating v_l and z_l. The new feature vector is of length $2(2(48 + 2P) + 6)$ or $(204 + 4P)$.

3.6.2 Feature Selection

After generating a feature vector of length $(204 + 4P)$, the feature vector probably contains many redundant or worthless features. One the advantages of using tree methods is that they

provide an estimate of relative feature importance I_j on the variation of $\hat{F}(x)$. From (Friedman, 2001), generally this measure is taken as:

$$I_j = \left(E_x \left[\frac{\partial \hat{F}(x)}{\partial x_j} \right]^2 var_x[x_j] \right)^{\frac{1}{2}} \quad (26)$$

Where E_x is the expected value and var_x is the variance. This is approximated for regression trees by:

$$\hat{I}_j^2(T) = \sum_{t=1}^{J-1} \hat{\imath}_t^2 1(v_t = j) \quad (27)$$

Where t are the non-terminal nodes of the J-terminal node tree T, v_t is the splitting variable at node t , and $\hat{\imath}_t^2$ is the corresponding empirical improvement in squared-error. For M trees, the influence is measured using:

$$\hat{I}_j^2 = \frac{1}{M} \sum_{m=1}^{M} \hat{I}_j^2(T_m) \quad (28)$$

These relative feature measures can be used to select the most useful features. Removing less useful features will decrease computational costs while potentially affecting performance. This importance metric is used to iteratively trim and select features.

3.6.2.1 Iterative Feature Trimming

An iterative feature trimming method is used to gradually remove noise from the model by removing weakly predictive features. The procedure is as follows:

Procedure 2: Iterative Feature Trimming

1. *Generate a full feature set*
2. *Build and evaluate models through cross validation*
3. *Remove the lowest p% of features from the feature vector*
4. *Generate a new feature vector from the remaining features*
5. *Repeat steps 2-4 for a predetermined number of iterations*
6. *Select the feature set corresponding to the iteration with the lowest cross validation score*

Models are evaluated at each iteration by the mean, cross-validated RMSE. The RMSE is expected to start at higher value when there are many noisy features. After some iterations, the weakly predictive features will be removed and the RMSE is expected to gradually improve. Towards the end of the procedure, the RMSE is expected to increase which signifies useful features are being removed from the model. The feature set that corresponds to the lowest RMSE value is taken as the final model.

The computational cost for fitting a model with the Tree Boosting algorithm scales linearly with the number of included features. Decreasing the amount of features included in the model will become important as models are trained on larger histories of weather and data or train models on a variety of DZ.

3.6.2.2 PCA Vector Length Selection

The method shown in Procedure 2 can also be used to select an appropriate PCA vector length, P. One can generate starting feature sets for different values of P, repeat the trimming procedure, and select the feature set with the best performance from all iterations of each P value.

3.7 Adapting to the RAP Data set

Ideally, models could train on a data set where there is a measurement of the wind stick at the DZ and a measurement of the wind stick at the RL at the same time. Models could have one stick, at the RL, taken as input to predict the other wind stick at the DZ. Unfortunately, this data set does not exist, and it would be very expensive to create. Instead, this methodology uses pseudo-dropsondes in place of actual measurements. In this study, dropsondes at the DZ and the RL are simulated with wind sticks taken from the weather analysis. Models are trained on dataset of simulated drops where the input is made up of data from the forecasts and remote pseudo-dropsondes. During training, models learn to predict the pseudo-dropsonde at the DZ. The forecast is taken as a RAP weather forecast with a six hour lead time, and the pseudo-dropsondes are taken from the analysis at the forecast time.

As discussed in Section 2.2, the RAP dataset is used because it has matching forecasts and analysis. In this study, models use forecast data that what would be available to aircrews during actual airdrop missions. Aircrews typically receive a block of forecasts that are spaced hourly and encompass the drop time within a few hours. These forecasts are typically downloaded on the ground before a drop, so the lead times of the forecasts can range from 2 to 24 hours. This study makes the assumptions that forecasts are always available for one hour prior to a forecast and one hour after a forecast, forecasts cover the DZ 130 km in the East direction and 130 km in the North direction, and the block of forecasts downloaded shortly before a drop are the only information available. This study also only uses forecasts with a lead time of six hours. Because the RAP dataset has records for many places and times, machine learning models can be trained for a variety of conditions. In this study, the forecast is segmented to a 260 km x 260 km grid centered on the DZ. This means each forecast weather cube is shaped (37 ,20 ,20) for each

weather variable of interest. This forecast size was chosen because it approximately represents the minimum forecast area aircrews would receive operationally.

The RAP dataset provides forecasts and analysis for about 3.5 years of data and covers geographic North America. The training dataset consists of samples with unique forecast times. The samples are divided into a fit, validation, and test set. The fit data set consists of the forecasts and pseudo-dropsondes used while the model is training. The validation set is only evaluated periodically during the fitting process to monitor how the model generalizes. The test set is completely unseen during the fitting process and is used to evaluate its performance. All plots in this document are shown from a test data set unless otherwise noted.

Table 6: RAP Dataset Months and Samples

Month	# Samples	Month	# Samples	Month	# Samples	Month	# Samples
Jan-12	0	Apr-13	719	Jul-14	729	Oct-15	744
Feb-12	0	May-13	744	Aug-14	738	Nov-15	720
Mar-12	0	Jun-13	720	Sep-14	689	Dec-15	744
Apr-12	0	Jul-13	0	Oct-14	744	Jan-16	744
May-12	552	Aug-13	702	Nov-14	720	Feb-16	696
Jun-12	711	Sep-13	719	Dec-14	743	Mar-16	744
Jul-12	676	Oct-13	744	Jan-15	729	Apr-16	676
Aug-12	0	Nov-13	720	Feb-15	488	May-16	709
Sep-12	679	Dec-13	743	Mar-15	728	Jun-16	653
Oct-12	0	Jan-14	738	Apr-15	647	Jul-16	744
Nov-12	0	Feb-14	615	May-15	744	Aug-16	744
Dec-12	743	Mar-14	549	Jun-15	695	Sep-16	0
Jan-13	744	Apr-14	720	Jul-15	741	Oct-16	0
Feb-13	672	May-14	744	Aug-15	730	Nov-16	0
Mar-13	743	Jun-14	720	Sep-15	720	Dec-16	0
						TOTAL	33920

Table 6 shows the months from the RAP dataset. The '# Samples' columns shows the number of forecasts used in each month. Each sample comes from a matching pair of 6 hour lead time

forecasts and analyses that are hourly spaced. Some months have zero data either because the data was not produced, missing, or not downloaded for this study.

Data was placed into the fit, validation, and test sets at random. For all experiments in this study, 75% of the data was placed in the fit set, 15% was in the validation set, and 10% was used in the test set. The random dispersion of the data allowed for more varied weather conditions to be in each set. For the data shown in Table 6, 25440 random samples were in the training set, 5088 random samples were in the validation set, and random 3392 samples were in the test set.

3.8 Ballistic Wind Estimates

While a payload is falling toward the ground, the wind acting on it causes it to drift. For simulation purposes, it is useful to only model the aggregate wind that acts on the payload. This value is wrapped up in what is called a ballistic wind, and it summarizes the degree to which the winds cause an object to drift. The ballistic wind, B, can be calculated using the total drift, D, and time of fall, T, as $B = D/T$. This metric summarizes the quality of a wind estimate for a column of air, and this metric is used to verify the predictive models discussed thus far.

The RAP dataset contains 37 pressure levels. The maximum drop altitude for operation airdrop is approximately 25,000 feet AGL. In this study, the lowest 30 pressure levels are used to create predictive models. The top most pressure level has an isobaric pressure of 25 kPa which corresponds to approximately 34,000 feet above MSL.

3.8.1 Ballistic Wind Procedure

Wind information for one wind component, u or v, is given as a vector of wind values over the DZ. The values in the wind vector match specific altitudes, corresponding to the pressure levels discussed earlier, and are quantified by a geopotential height, a. The ballistic wind calculation

uses a function $coesa()$ which is a lookup table for the 1976 Coesa Standard Atmosphere model and returns a density, ρ, for a given MSL altitude (U.S. Standard Atmosphere 1976).The procedure for calculating the ballistic wind is as follows:

Procedure 3: Ballistic Wind Calculation

$drift = 0$
$time = 0$
$For\ i\text{:}\quad 0..len(u) - 1$
$\rho_1 = coesa(a[i])$
$\rho_2 = coesa(a[i+1])$
$fr_1 = \left(\frac{2g}{\rho_1}\right)^{\frac{1}{2}}$
$fr_2 = \left(\frac{2g}{\rho_2}\right)^{\frac{1}{2}}$
$fr = \frac{fr_1 + fr_2}{2}$
$dz = a[i+1] - a[i]$
$w = \frac{u[i] + u[i+1]}{2}$
$drift \mathrel{+}= w\frac{dz}{fr}$
$time \mathrel{+}= dz/fr$
$ballistic = drift/time$

Where g is a gravity constant taken to be 9.81 m/s, u is a wind component,w is the average wind in a section of a column of air, $drift$ is the lateral movement of the payload, dz is the change in altitude, and fr is a fall rate They payload's fall rate is approximated by its terminal velocity, $v_t = \sqrt{\frac{2mg}{\rho S C_d}}$. The Ballistic Wind calculation procedure normalizes by the mission specific parameters, $\sqrt{\frac{m}{SC_d}}$. m is the mass of the payload, S is the projected area, and C_d is the drag coefficient. Because these are constant for each altitude, they can be accounted for at the end and make a ballistic wind calculation reflect a mission's parameters.

3.8.2 Prediction of the Mean Winds over the DZ

Models fit using least squares were shown predict mean values of the winds over the DZ (Section 3.4). To quantify the ballistic winds over the DZ, the output of the mean wind models are used as input to the ballistic wind calculation. This ballistic wind value could be used to calculate a CARP.

3.8.3 Ballistic Wind Uncertainty

Mission planners need the uncertainty of the ballistic wind estimate to derive probabilities for different mission failure modes. Ideally, this uncertainty would be presented in the form of an error CDF. This CDF can be used to estimate the probability of a drop missing the DZ or the probability of hitting a protected object on the ground. The quality of an error CDF is often quantified by its tightness and reliability. Tightness meaning that the widths of the distribution are narrow and reliability meaning that for a given quantile, α, the expected percentage of estimation errors are actually below that threshold. As seen in 3.8, the ballistic wind is a weighted integration over the wind stick. By treating the prediction error on each pressure level as a random variable, the problem becomes one of integrating over random variables.

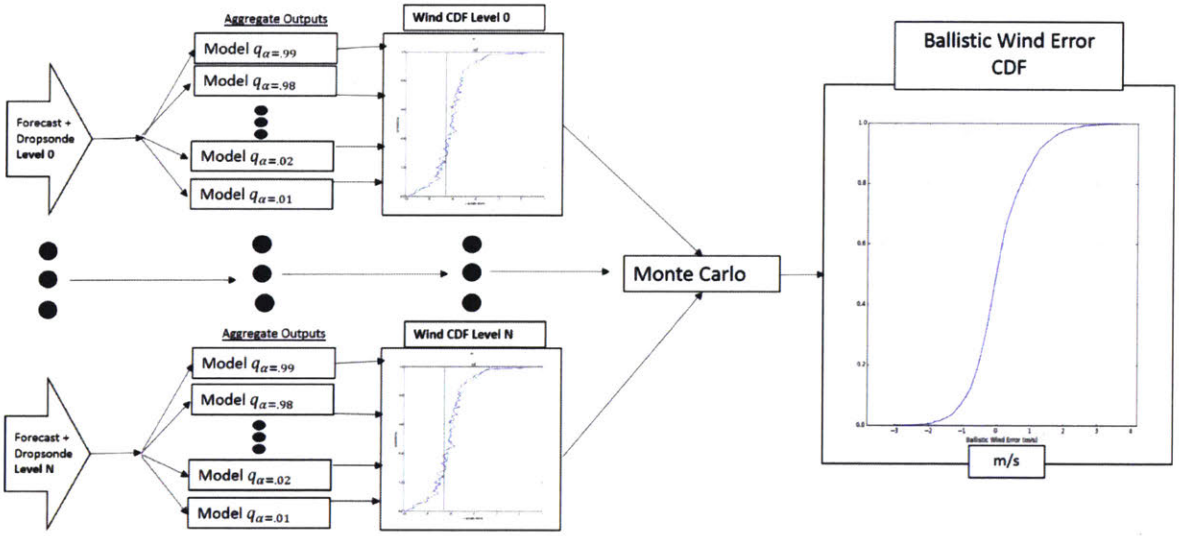

Figure 7: Ballistic Wind Monte Carlo

Figure 7 shows the general outline for how the ballistic wind error CDFs are inferred. For each pressure level, a set of quantile regressors are fit that together predict wind CDFs for each pressure level. The wind errors on each pressure level are random variables that are treated as correlated. The integration of the winds on all the pressure levels is solved through Monte Carlo.

When the random variables are independent, the summation of n random variables is solved through recursive convolution. For two random variables, where $F(x_i)$ specifies a marginal distribution function and Z is the sum, the procedure is as follows:

$$F(x_i) = P(X_i \leq x_i) \quad (29)$$

$$Z = X_1 + X_2 \quad (30)$$

$$F(z) = \sum_{x_2 \subset R_{x_2}} F(z - x_2)F(x_2) \quad (31).$$

For n random variables, the distribution of the summation, Z, is found using:

$$Z = X_1 + \cdots + X_n \quad (32)$$

$$Y_2 = X1 + X_2 \quad (33)$$

$$Z = Y_n = \ Y_{n-1} + X_n. \quad (34)$$

Unfortunately, the prediction errors on each pressure level are not independent. Quantile regression models are fit and used for each pressure level in the ballistic wind calculation. The pressure levels that are near each other use many of the same input features during training and operation. The result is that predicted error CDFs are correlated, so one cannot use recursive convolution to sum the random variables. This section details a Monte Carlo based integration method for correlated prediction CDFs.

As discussed above, the Monte Carlo needs samples from a vector of correlated, arbitrary prediction distributions. This is accomplished through the use of a Gaussian Copula. A copula, C, is the CDF of a random vector with uniform marginal distributions.

$$C(u) = P[U_1 \leq u_1, \dots \ , U_n \leq u_n] \quad (35)$$

Copula theory relies on Sklar's theorem which states that any multivariate cumulative distribution function can be specified by a copula and its marginal distributions, and the copula is unique if the marginal distributions are continuous (R¨uschendorf, 2013).

A copula takes advantage of the fact that the probability integral transform, $F_X(X)$, of a random variable is a random variable with a uniform distribution. X is a random variable with an arbitrary probability distribution. If $Y = F_X(X)$ then Y has a uniform probability distribution $U(0,1)$.

For a vector of random variables, the probability integral transform is applied to each component to form the vector of uniformly distributed random variables needed for the copula.

$$(U_1, \dots \ , U_n) = \left(F_{X_1}(X_1), \dots \ , F_{X_n}(X_n)\right) \quad (36)$$

If the inverse of each of the CDFs exists, the original margins can be recovered.

$$X_1 = F_{X_1}^{-1}(U_1) \quad (37)$$

$$(X_1, \dots, X_n) = (F_{X_1}^{-1}(U_1), \dots, F_{X_n}^{-1}(U_n)) \quad (38)$$

This method can be used to generate a vector of uniformly distributed random variables with prescribed correlation. Consider the multivariate standard normal distribution, $N(0, \Sigma)$, where each component is a standard normal, $N(0,1)$, and the correlation matrix is specified by Σ. Applying the inverse probability integral transform produces a vector of uniformly distributed random variables, U(0,1), with a ranked correlation matrix Σ_R. The linear correlation between two random variables is found through:

$$\rho_{X_1,X_2} = \frac{E[(X_1 - \mu x_1)(X_2 - \mu x_2)]}{\sigma_x \sigma_y}. \quad (39)$$

The ranked correlation matrix, Σ_R, is calculated using the ranked correlation coefficient ρ_R.

$$\rho_R = \rho(F_{X_1}(X_1), F_{X_2}(X_2)) \quad (40)$$

(Papaefthymiou & Kurowicka, 2009) shows the two correlations are related by:

$$\rho(X_1, X_2) = 2\sin(\frac{\pi}{6}\rho_r(X_1, X_2)) \quad (41)$$

Figure 8 shows this for the two dimensional case with correlation 0.5.

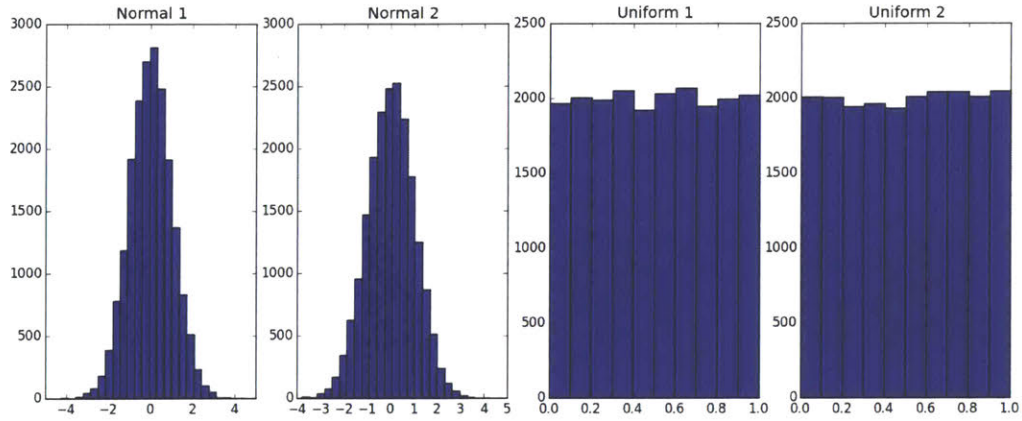

Figure 8: Inverse Probability Transform for Two Dimensional Standard Normal. This is an example showing that the correlation between two Gaussian random variables is maintained after the inverse probability transform is applied to each.

Figure 8 shows two correlated normal random variables that were transformed into two correlated uniform random variables.

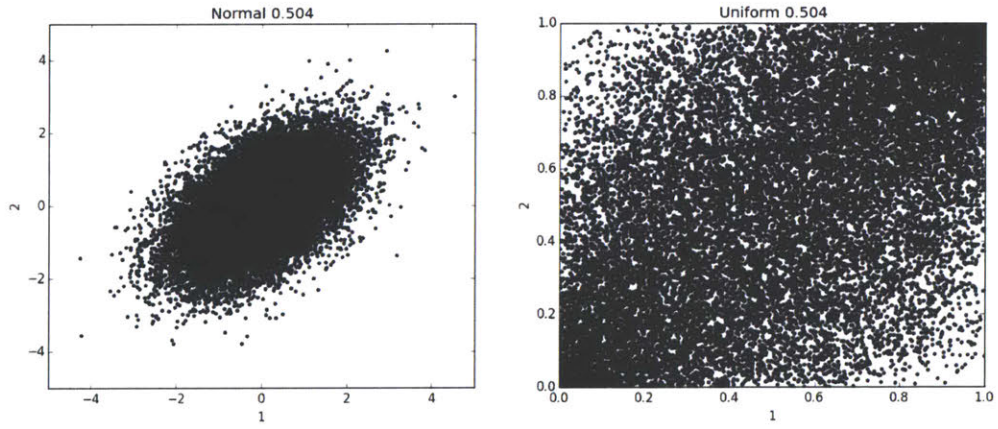

Figure 9: Scatter Plots Showing the Correlation Between the Generated Random Variables. The left shows a scatter plot for the two Gaussain random variables. The right shows the samples from the uniform random variables obtained after the inverse probability transform.

Figure 9 shows that the uniform random variables maintain the correlation after the probability transform function is applied to the normal random variables.

The vector of correlated, uniformly distributed random variables can now be used to generate samples from the wind distributions.

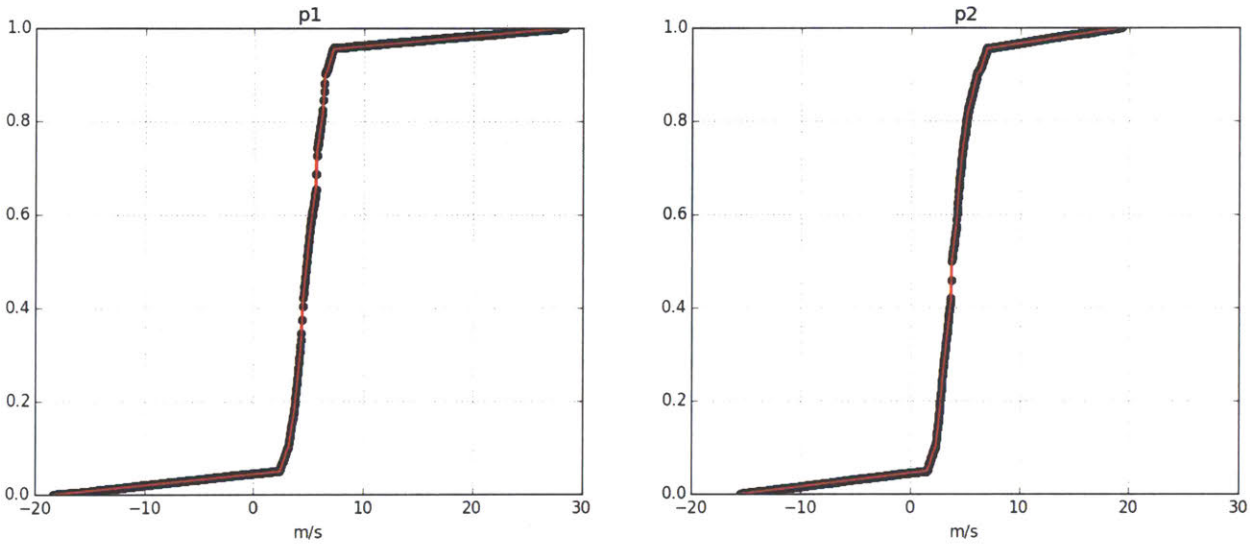

Figure 10: Example Prediction CDFs for the Winds on Two Pressure. The red line show the output of the quantile regression models. The block dots show the distribution of generated samples for these CDFs.

Figure 10 shows, in red, an example of two CDFs prescribed by the output of the quantile regression functions on two pressure levels for the east winds. 10,000 samples were generated from these CDFs using the method described thus far and a ranked correlation of 0.5. The black dots show the distribution of the samples generated from the CDFs in red, and they appear to match well.

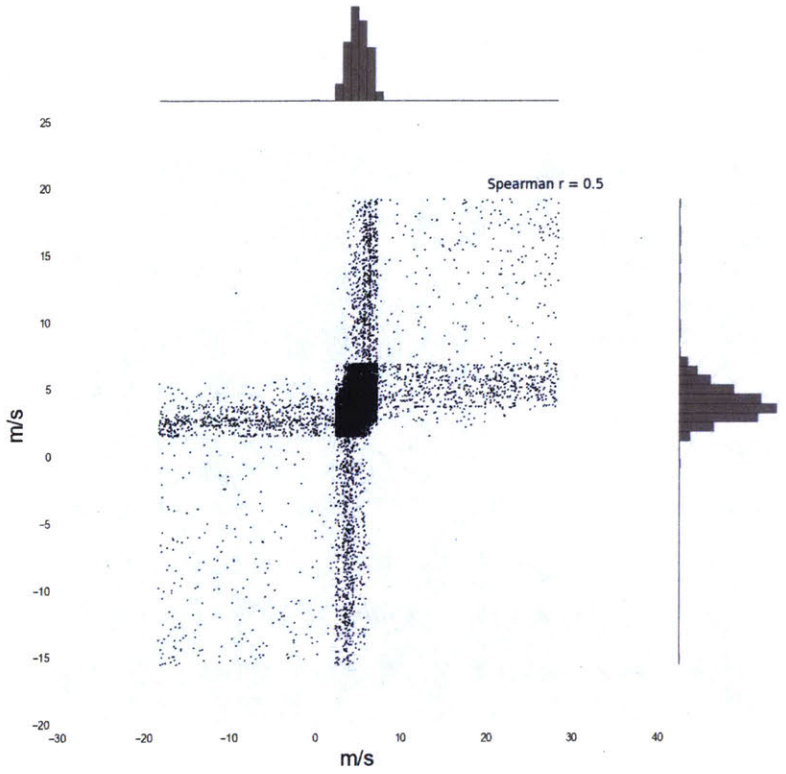

Figure 11 Samples Generated from Example Uncertainty Distributions.
Two example uncertainty distributions for the winds on separate pressure levels are shown on the axes. These are the PDFs of the CDFs shown in Figure 10. The scatter plot shows correlated samples drawn from both distributions.

Figure 11 shows the density functions of the samples generated from the CDFs in Figure 10 on the axes, and the samples are plotted in the center. The prescribed correlation was 0.5 and the samples recover that value according to the spearman ρ coefficient.

The same procedure generalizes from the two dimensional case to the n dimensional case. The result is many generated wind stick samples that together represent the total uncertainty. For each wind stick sample, the wind values are integrated through pressure altitudes to get a ballistic wind. Doing this for each sample wind stick, there are now many samples for what the ballistic wind value could be according to the prediction CDFs. An outline of the procedure:

Procedure 4: Ballistic Wind Monte Carlo

1. *From the quantile regression models, obtain marginal distributions for the winds on each pressure levels.*
2. *Define a covariance matrix, Σ_R, that specifies the dependency you wish to model between the random variables. This covariance is based on the Spearman rank correlation coefficient.*
3. *Generate samples from a vector of Gaussian random variables with the same covariance Σ. This covariance is calculated from Σ_R using the relation between linear correlation, ρ, and rank correlation, ρ_R.* $\rho(X_1, X_2) = 2\sin(\frac{\pi}{6}\rho_r(X_1, X_2))$
4. *Apply the Gaussian probability integral transform to the generated Gaussian samples to get a vector of correlated, uniformly distributed random variables*
5. *Apply the inverse cumulative distribution function of each marginal defined in (1) to each component of the vector produced in (4). The result is a vector of samples from the margins specified in (1) with covariance, Σ_R, specified in (2).*
6. *Estimate a ballistic wind for each sample from the random vector. The accumulation of these samples specify a discrete probability density function for the ballistic winds.*

This procedure is based the work in (Bohdalová & Šlahor, 2014), (Embrechts, Lindskog, & McNeil, 2001), and (Papaefthymiou & Kurowicka, 2009) . One of the interesting extensions here is that the marginal error CDFs are produced using quantile regression.

The correlation matrix, Σ_R, used in the Ballistic Wind Monte Carlo has a large effect the resultant ballistic wind uncertainty distribution. The correlation matrix specifies the correlation between predicted wind errors on each pressure level used in the ballistic wind calculation. This study began by modeling these errors as completely uncorrelated or correlated. The uncorrelated model simulates that prediction errors on one level do not indicate prediction errors on any other level. The perfectly correlated model simulates that the wind error on one pressure level fully specifies the wind errors on the other pressure levels. The uncorrelated model tended to underestimate the ballistic wind errors, and the perfectly correlated model tended to overestimate

the ballistic wind errors. Using the knowledge that best answer is somewhere in between, a method was developed for optimizing the correlation based on the performance in the validation data set. Note that the validation set is different than the final test set used to qualify performance.

Intuitively, pressure levels that are closer to each other should have errors that are more correlated than pressure levels further away from each other. The correlation, ρ, between pressure levels is modeled as exponentially decreasing with distance. The correlation between to pressure levels, i and j, is found using

$$\rho_{i,j} = \rho_{j,i} = e^{\frac{-r}{|i-j|}}. \quad (42)$$

The parameter r controls the amount of correlation where a larger value decreases the amount a correlation and a smaller value decreases the amount of correlation.

The amount of correlation between pressure levels is now controlled by the single parameter, r, and the distance. r is selected based on the performance in the validation set. The validation set is set of samples not directly used by the models for fitting, so it gives an idea of how the model generalizes. Optimizing a parameter using the validation set could limit one's ability to regularize a model, but it was found to be effective in this case. The procedure for evaluating a particular r value was as follows:

Procedure 5: Evaluating The Performance of Parameter r

1. *Generate CDFs for the errors on each pressure using the quantile regression models for the data in the validation set.*
2. *Calculate a correlation matrix Σ_R using r.*

3. *Run the Ballistic Wind Monte Carlo Procedure for each sample from step (1) and the correlation matrix from step (2).*
4. *For each validation sample, find the quantile, q, of the true ballistic value in the uncertainty distribution produced in step (3).*
5. *Sort the observed quantiles, q, into b bins. The normalized number of observations is each bin, b_i, is denoted as n_i.*
6. *Evaluate the loss,*

$$L_r = \max_i (n_i - \frac{1}{b}). \quad (43)$$

Procedure 5 is used in a golden interval search (Press, Teukolsky, Vetterling, & Flannery, 2007) to find an optimal r value. The starting bounds for an interval search is set at 0 and 4. r values less than 0 result in correlations greater than 1, and r values greater than 4 were observed to create too much correlation between pressure levels.

The method of selecting an r value is analogous to building a regression model for the ballistic wind from one drop altitude. The methodology in Section 3.2 built regression models for the winds on one pressure level at a time. This was done to reduce the dimensionality for any one regression model and to keep the solution modular. It is possible for an aircraft to release the dropsonde from a different altitude than the final payload. If a model tried to predict the ballistic winds directly, one would require a feature set, and model, for each combination of dropsonde and payload altitudes, or some way of including a dropsonde's position in the feature set. These models would find relationships between the winds on each pressure level and produce a dynamic ballistic wind value. The r value and the ballistic wind Monte Carlo are how the methodology presented thus far balance a modular solution and what might be an optimum solution found by ballistic wind regression models.

4 Results and Analysis

4.1 Summary of Findings

The overall goal of this thesis is to produce a method for estimating the winds at a DZ using deterministic forecasts and a dropsonde released at a distant, remote location. The available forecasts are formatted as large arrays of multiple variables. Section 3.6.2.1 shows a method for generating features from the forecasts and iteratively trimming poorly performing features. Section 4.2 shows how this method successfully reduced the feature space by 69% and decreased training time by approximately 75%. This methodology could be used to select features for future models. After Section 3.6.2.1, models were trained using a feature set without the time derivative features, without the pressure level derivative features, with PCA vectors of length 50, and without trimming poorly predictive features. The exact features used in this set are shown in Table 8 in Appendix A. Finding and studying which features are most predictive at different positions in the weather cube using the iterative trimming method is left for future work.

Section 3.3 details how to fit Tree Boosting models that predict DZ winds using forecasts and a remote dropsonde. Section 3.8 shows a method for integrating over a set of DZ wind predictions to produce a ballistic wind estimate and uncertainty. To verify this wind estimation system, it is first shown that the method is successful when fit using pseudo-dropsondes released 92 km northeast of a DZ near Yuma, AZ. A variety of mission restrictions might make releasing a dropsonde exactly 92 km northeast of a DZ impossible, so it is then established that the method can be repeated for other RLs around the same DZ. If one was interested in performing remote dropsonde missions at this DZ today, they could use the developed system successfully. Airdrop missions are conducted all over the globe, so it is next established that the method developed for the DZ at Yuma could be extended to other geographic locations. Models are trained and

evaluated using data from DZs near Dyess TX, Riverton WY, and Minot ND. These models have similar performance when tested on data from the same geographic location as the training site.

Ideally, one would have a history of forecasts and analyses for every location around the globe. One could then pull the relevant data for a target DZ and fit a model before an airdrop mission. Unfortunately, this data does not exist with necessary resolution, so for this method to be useful, it must perform on locations unseen during training. Some preliminary results are shown for models trained using a few DZs and later tested on data from a different DZ (Section 4.3.4).

In the airdrop community, a wind estimate is only as good as its uncertainty. Section 3.8.3 detailed a quantile regression and Monte Carlo based method for inferring the uncertainty of a ballistic wind estimate. Section 4.4.3 shows that the dynamically produced uncertainty CDFs are useful.

The results in this section are taken from a test dataset unless denoted otherwise. Section 3.7 details where the data samples come from and how the complete dataset is split into a training, validation, and test set. For all experiments in this chapter, 75% of the data was placed in the fit set, 15% was in the validation set, and 10% was used in the test set. For the data shown in Table 6, 25440 random samples were in the training set, 5088 random samples were in the validation set, and random 3392 samples were in the test set. The features used to fit these models are shown in Table 8 in Appendix A.

4.2 Feature Selection

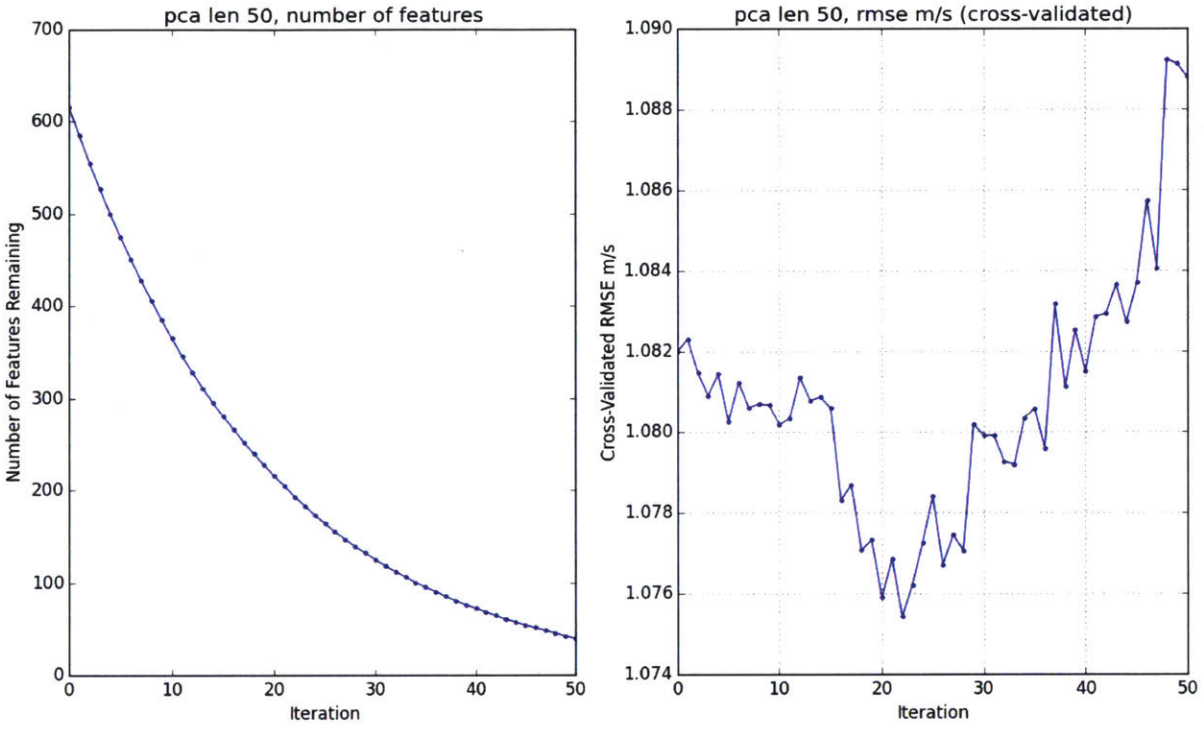

Figure 12: Iteration vs Cross-Validated RMSE for P=50.
Results for the application of Procedure 2: Iterative Feature Trimming. The worst 5% performing features were removed from the model after each iteration. The figure on the left shows the number of features included in the model after each iteration. The figure on the right show the 3-fold RMSE for each iteration. The Tree Boosting algorithm effectively chewed through noisy features from iteration 0 to 21. After iteration 21, the trimming procedure removed predictive features.

Figure 12 shows the results of the iterative trimming method, Procedure 2, for PCA vectors of length 50 ($P = 50$) and data from the 92.5 kPa pressure level, the 92 km northeast RL, a u model, and a Yuma DZ. The trimming rate, p, was set to 95%. The left plot shows the length of the feature vector evaluated at each iteration. With this P value, the procedure started at 616 features at iteration 0 and ended with 39 features at iteration 50. The right subplot shows the 3-fold cross-validation RMSE for each iteration. It was expected that the RMSE plot would show a

noisy valley where the performance gradually improved to a global minimum then started to get worse as the trimming began removing useful features. Interestingly, this experiment showed that the estimation performance improved very gradually, and the global RMSE minimum only increased the performance by 0.6% from the starting value. This shows the Tree Boosting algorithm was effective in weeding through the insignificant features at early iterations. If one chose the iteration with the lowest RMSE, the feature set is reduced by 69%.

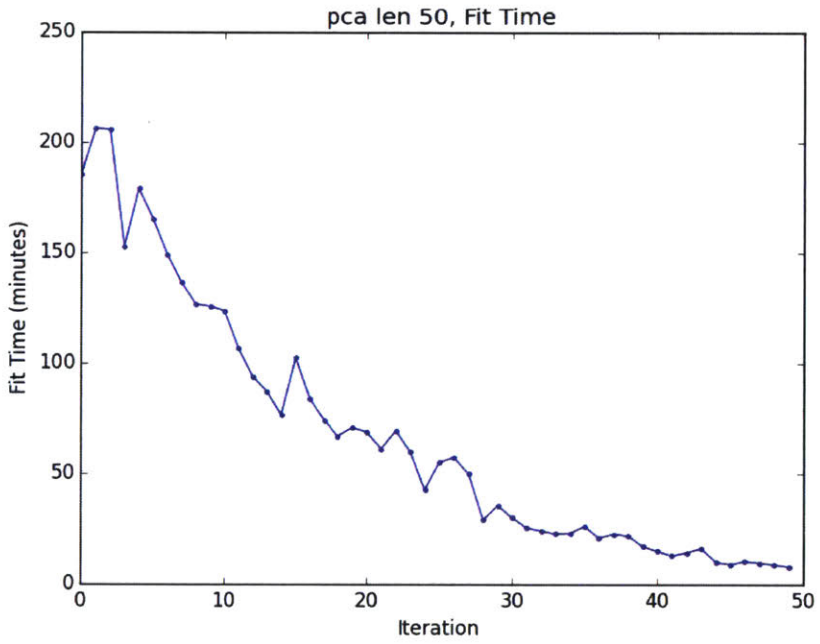

Figure 13 Training Time vs Iteration Number.
This plot show average training time for a model in the 3-fold cross validation. The number of features decreased at each iteration and the training time decreased accordingly. From (Hastie, Tibshirani, & Friedman, 2009), the Tree Boosting algorithm's required training time scales linearly with the number of included features.

As seen in Figure 14, the reduced feature set allowed models to train more quickly. This gain will become more important as regression models are trained on larger histories of weather and data or train models on a variety of DZ.

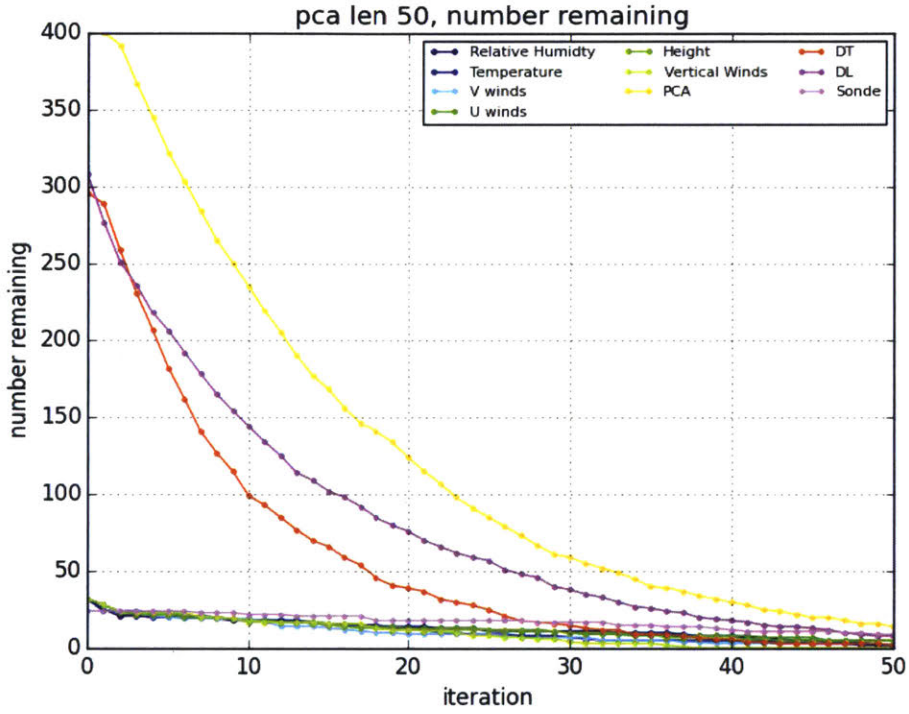

Figure 14: Number of Features in Categories After Each Trimming Iteration. The number of features remaining in a number of categories is plotted against iteration. The categories are not mutually exclusive, so the sum of the size of all categories for an iteration does not equal the feature vector length.

For the same experiment, Figure 14 shows the number of features remaining in different feature categories after each trimming iteration. Notice that for each iteration the sum of all the categories does not equal the length of the feature set. The *DT* and *DL* categories representing the discrete time and pressure level derivatives share features with some or all of the other categories. This plot shows that the *DT*, *DL*, and *PCA* features make up a significant portion of

the feature space. Perhaps unsurprisingly, these categories quickly shrink. This is better illustrated in Figure 15.

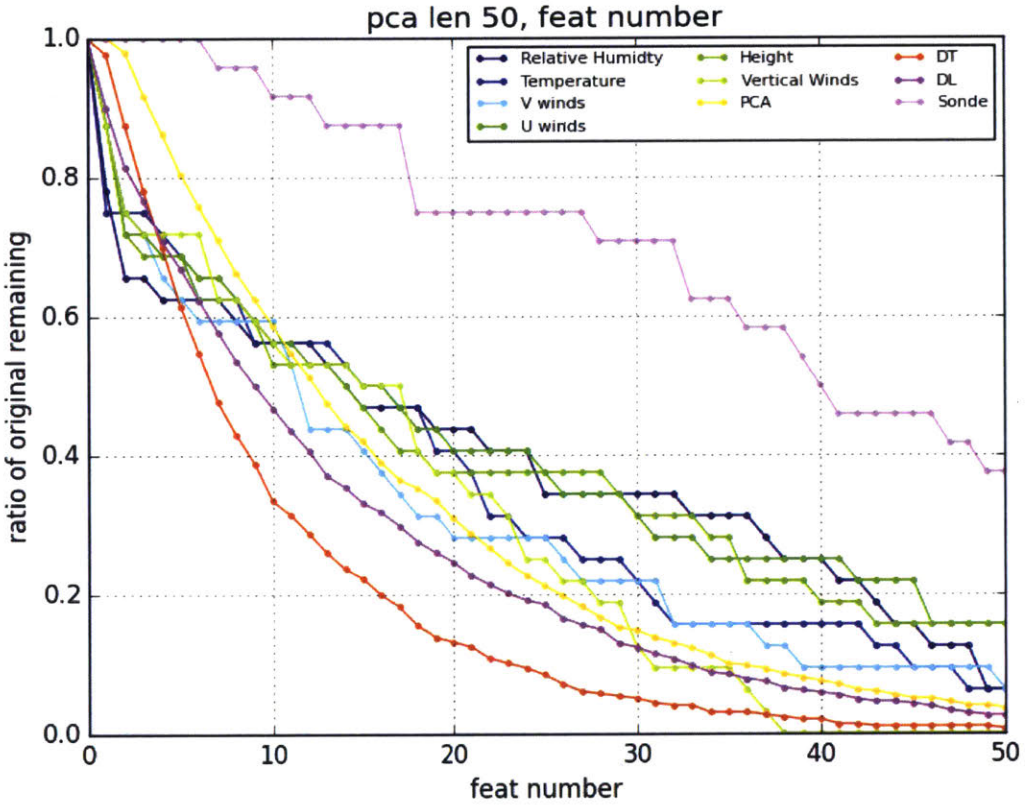

Figure 15: Ratio of Features Remaining in Each Category After Each Trimming Iteration. The ratio of remaining features in a category is the number of features in a category at iteration i divided by the number of features in that category at iteration 0. This figure illustrates the rate at which different feature categories were trimmed. It shows that the dropsonde features, 'Sonde', are generally the most predictive category and the features in the 'Vertical Winds' category are generally the least predictive category. The 'DT' and 'DL' features are also trimmed very quickly, but this is likely because they make up a large percentage of the total feature vector length.

Figure 15 shows the ratio of features remaining in each category compared to the starting set. The slope of each line demonstrates the attrition rate for each category. The sonde features are trimmed at the slowest rate which reflects the intuition that they would be most predictive of a

sonde at the DZ. The *DL* and *DT* features are trimmed at the fastest rate which implies that they are generally less predictive than other categories.

The method shown in 3.6.2.1 can also be used to select an appropriate PCA vector length, *P*. One can generate starting feature sets for different values of *P*, repeat the trimming method, and select the feature set with the best performance from all iterations of each *P* value.

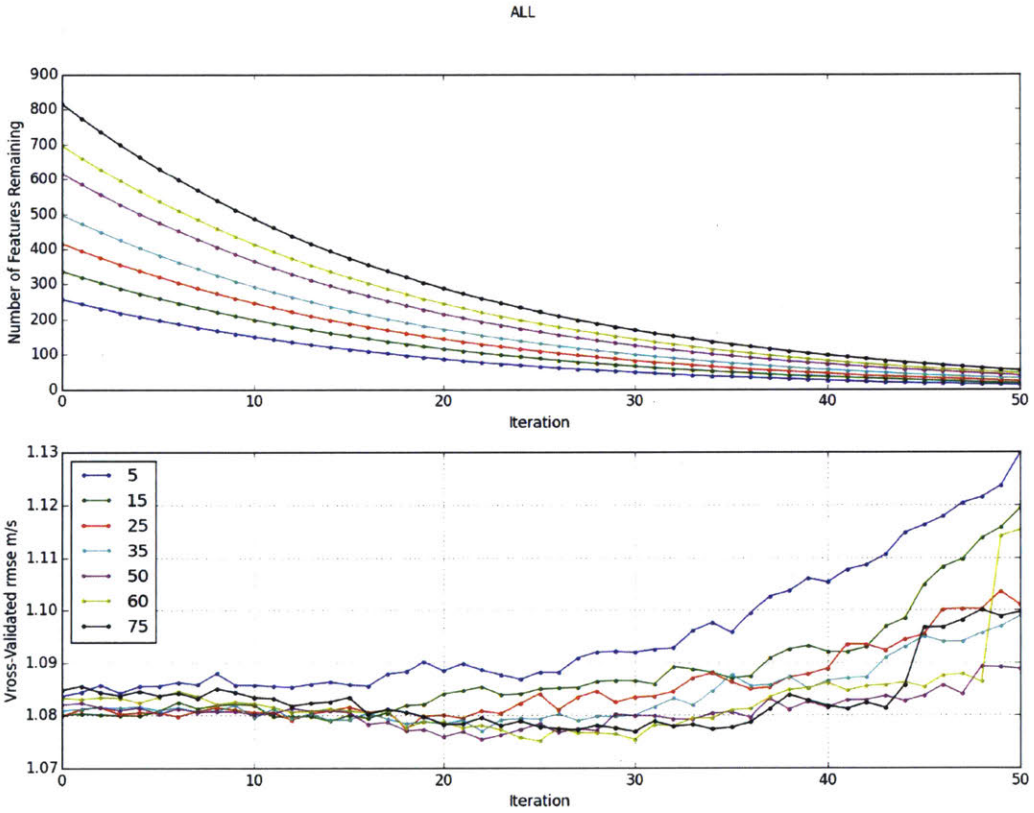

Figure 16 PCA Selection through Iterative Feature Trimming.
The trimming procedure was repeated for vectors with different PCA lengths. All showed a similar pattern of consistent or improved performance until a minimum and a decrease in performance after. Future models were trained using the features at the iteration and P value with the lowest RMSE.

Figure 16 shows the results for experiments with P values of 5, 15, 25, 35, 50, 60, and 75. For most P values, the trend from Figure 12 is repeated where performance generally improved until the point where useful features are trimmed. The best performance was observed at iteration 25 for a P value of 60. At this point the feature vector is of length 187. This P value and reduced feature set is the recommended value for future models. This procedure can be repeated for models fit on data from each pressure level and for each wind component.

4.3 Predicting the Winds Over the DZ using Machine Learning

4.3.1 Using a model for a fixed RL to predict the winds at the DZ

The weather cube used in this study is a three dimensional array spanning an east direction, a north direction, and vertical pressure levels. The first goal was to build a regression model that could predict the winds at the DZ using features derived from a remote dropsonde and a deterministic forecast. A regression model for each wind component and for each pressure level in the weather cube and, eventually, every remote grid point was fit and evaluated separately. A model was fit for both the u and v wind components seperatly. The first grid point tested was approximately 92 km northeast of the DZ at gridpoint (15,15). Figure 17 shows how the RMSE, when predicting DZ winds, for each pressure level's model. For this experiment, the models were fit on the RAP dataset discussed in Section 3.7. The set of models' performance on the test data set is shown in this section.

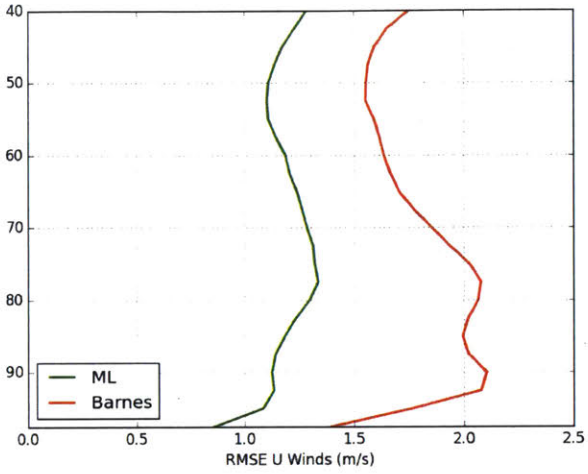

Figure 17 RMSE for U wind estimate on each pressure level.
The performance of the models for each pressure level and each wind component are shown. The ML models are shown to consistent outperform Barnes in a test dataset.

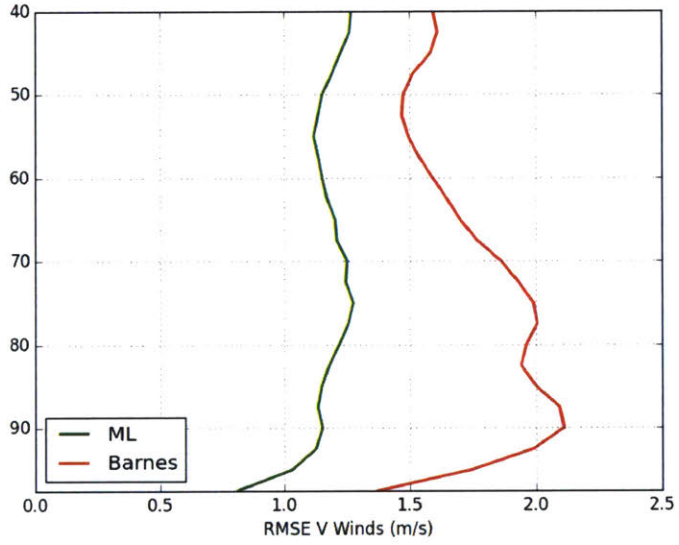

Figure 18: RMSE for V wind estimate on each pressure level.
The performance of the models for each pressure level and each wind component are shown. The ML models are shown to consistent outperform Barnes in a test dataset.

Figure 17 shows RMSE for the machine learning model (ML) and for the Barnes method. For both the u and v wind components, the ML model outperforms the Barnes scheme. The wind

predictions over the DZ for each pressure level are not particularly useful by themselves, but they are they later used in the ballistic wind estimates which are used to calculate a CARP.

Section 3.8 discusses why the ballistic wind is the most useful wind metric when evaluating wind assimilation methods for airdrop. The ballistic wind is a density weighted integration of the winds experienced by a payload falling through the atmosphere. The experiments in this section calculated a ballistic wind for payloads with normalized parameters, $\frac{m}{sC_D} = 1$. For drops of various mass (m), drag area (s), or drag coefficient (C_D), the ballistic wind can be converted through a post multiply.

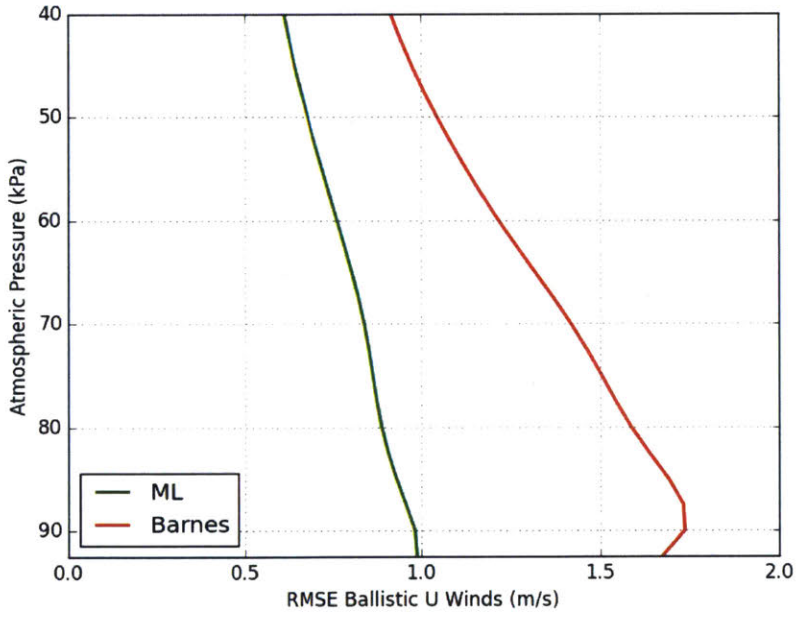

Figure 19: U Ballistic Wind RMSE. Yuma
The Ballistic wind is calculated according to Procedure 3. This plot shows the RMSE ballistic wind for a drop from each pressure level. Again, the ML models are shown to outperform the Barnes scheme.

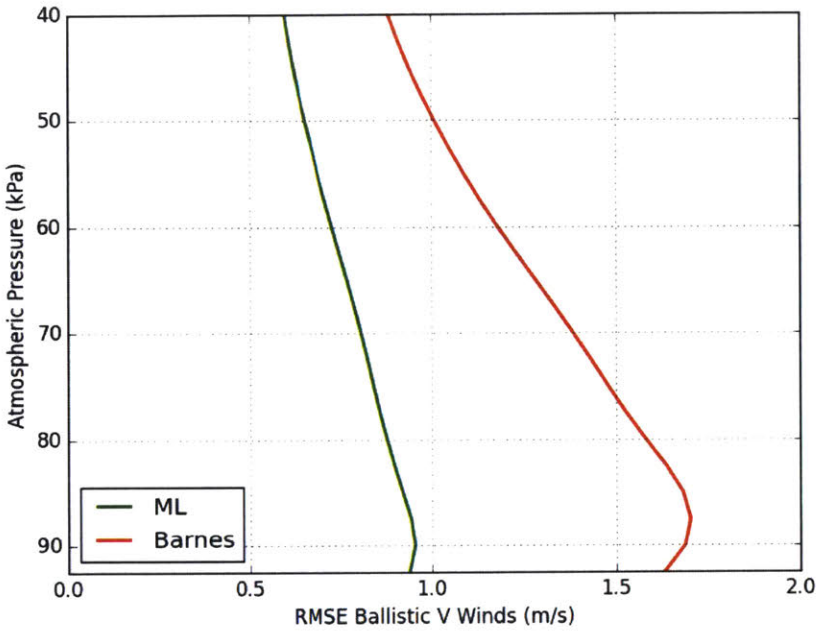

Figure 20: V Ballistic Wind RMSE. Yuma
The Ballistic wind is calculated according to **Procedure 3**. *This plot shows the RMSE ballistic wind for a drop from each pressure level. Again, the ML models are shown to outperform the Barnes scheme.*

Figure 19 show the ballistic wind RMSE for both the ML Models and the Barnes scheme when dropping from each pressure level. Figure 17 showed that the ML Models outperformed the Barnes scheme when estimating winds on each pressure level, and Figure 19 shows that these gains translate to the ballistic wind errors. Figure 19 was produced from the test data set for Yuma with the remote location 92 km northeast of the DZ (grid point (15,15)).

4.3.2 Results extended to other RLs around the Yuma DZ

Sections 4.3.1 showed the regression models outperform the Barnes scheme when fit and tested on dropsonde data from the grid point 92 km northeast of the DZ. Ideally, a remote dropsonde mission would succeed when the aircraft approaches the DZ from any direction. Figure 21 shows that similar performance is achieved at each remote location within the wind grid that is greater

than 50 km away from the DZ. The RLs are taken to be the grid points of the forecasts and analyses.

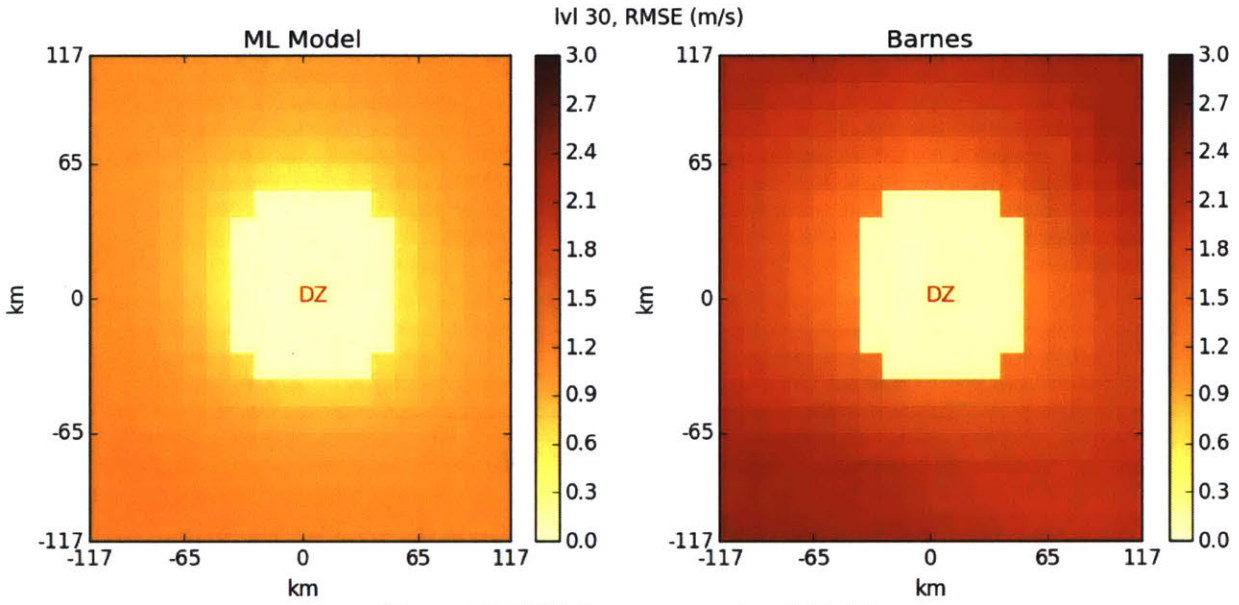

Figure 21 RME for pressure level 85 kPa.
Each RL has its own model. All models try to predict the winds at the DZ with a dropsonde at its location. ML models outperform the Barnes scheme at each remote location. The variance maps show that he ML models are also more reliable than the Barnes scheme.

For drops from the 85 kPa pressure level, the ML models had a ballistic wind RMSE between 0.8 and 1.1 m/s for each remote location. From the map, it is apparent that the ML models' performance had some dependence on the distance from the RL to the DZ, but it is a relatively weak. The Barnes scheme had ballistic wind RMSE values between 1.2 and 2.3 m/s showing its performance was much worse on average. The Barnes scheme's performance also shows a much stronger dependence on the distance from the DZ to the RL.

For the 20 x 20 weather grids used in this study, this methodology would require 400 sets of ML models if one wanted to release a dropsonde at any remote location. This is computationally expensive, and it might not be necessary. The input feature set essentially masks the location of

the dropsonde. This was done because, in these initial cases, models are trained using data from one droposnde location and from one DZ. Any geographic information based on terrain or the distance from the DZ to the RL is not included in the input features because they are static in the data set. This limits the models to finding patterns in weather values at the RL and DZ and patterns in the wind fields. Perhaps it should be expected that the models trained based on one RL should work reasonably well on data from a different RL.

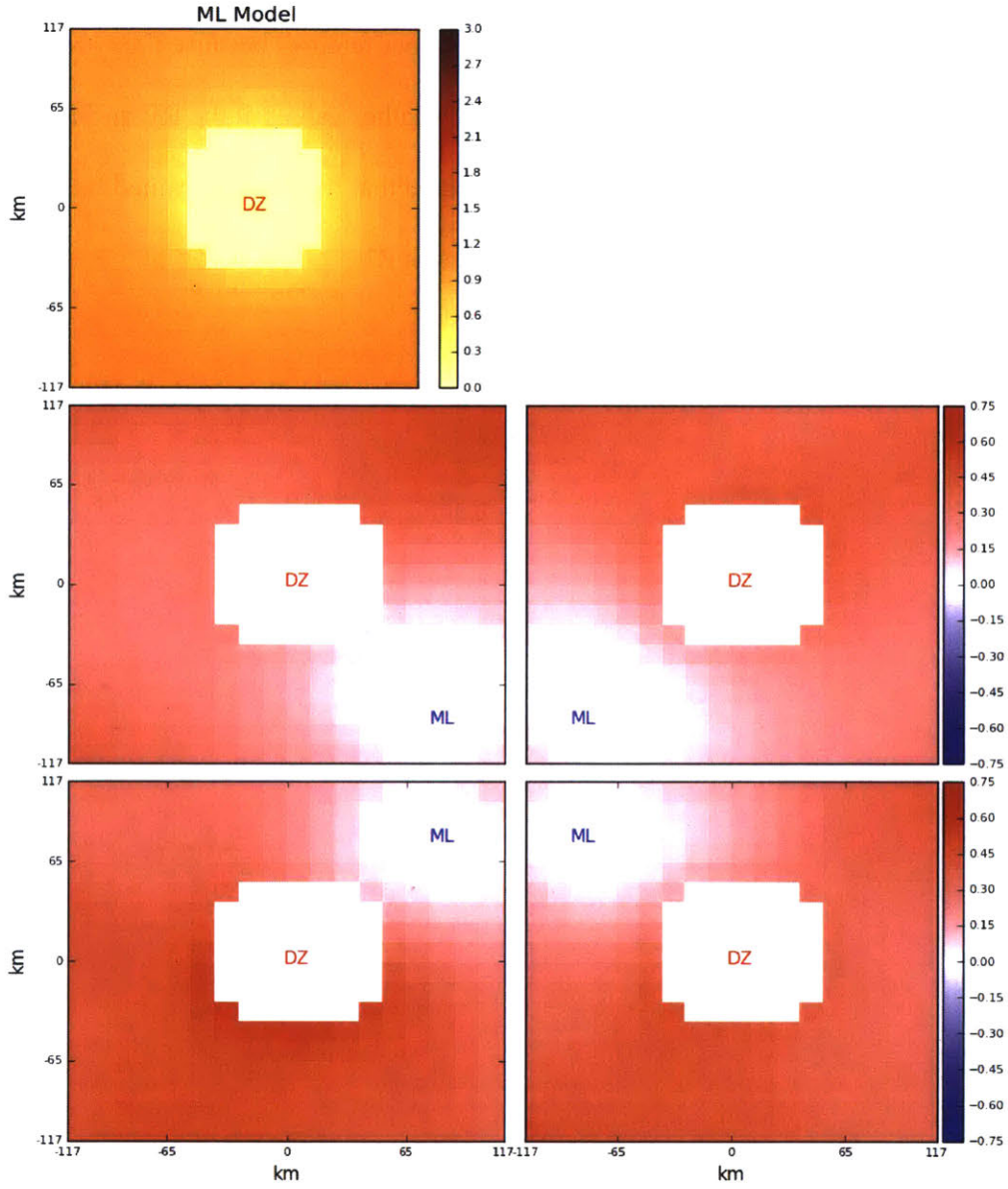

Figure 22: ML Models Fit at One RL and Tested at Other RLs, U Ballistic Wind. The orange heat map shows the RMSE for models trained on data from each pixel. On the lower 4 heat maps, the ML Model that was fit at the grid point marked in blue was tested on data from each of the other grid points. The red heat maps show the additional RMSE over the orange heat map. The model trained on data from the marked grid point performs well on data from nearby grid points. The performance decreases as the distance from the training grid point increases.

Figure 22 shows how a regression model fit at one RL performs based on dropsonde data at other RLs. The pixels in Figure 22 show the additional error for models trained on data from one grid point, marked with blue text, and tested on data from each other RL. The performance for grid points adjacent to the training point are effectively the same. The performance degrades with the distance from the training grid point.

This is a useful result for a number of reasons. First, it shows the ML Models are generally robust to positioning errors created by the dropsonde or the aircraft. Second, it shows that regression models for each RL might not be necessary. One might be able to save computer disk space and development time by building one set of regression models for a geographic region.

4.3.3 Extending the Methodology to other DZ Locations: Dyess, Minot, and Riverton

Up till now, models have been trained and tested on data from Yuma, AZ. Airdrop missions could be performed almost anywhere and at any time, so it is pertinent that these results were not unique for Yuma.

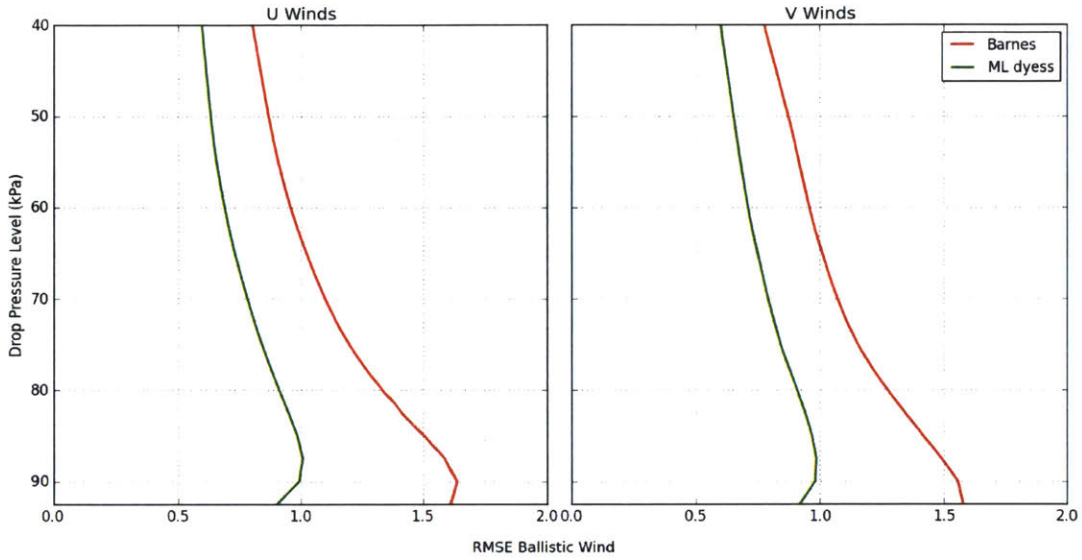

Figure 23 U and V Ballistic Wind RMSE. Dyess
This figure repeats the procedure shown in Figure 19 and 16, but the data is from Dyess, TX instead of Yuma. The performance is similar to the model from Yuma, and shows ML models can work at other geographic locations when they are trained on data from that location.

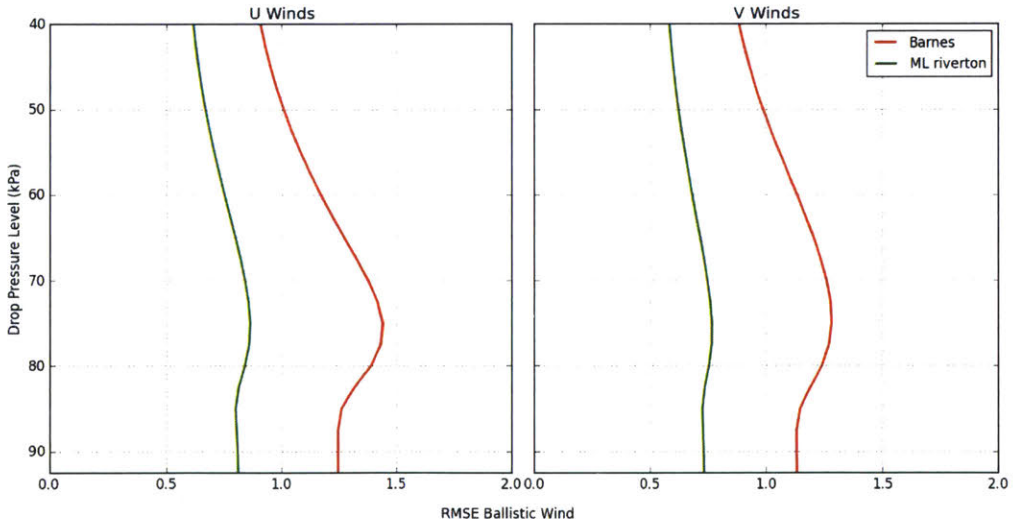

Figure 24 U and V Ballistic Wind RMSE. Riverton
This figure repeats the procedure shown in Figure 19 and 16, but the data is from Riverton, UT instead of Yuma. The performance is similar to the model from Yuma, and shows ML models can work at other geographic locations when they are trained on data from that location.

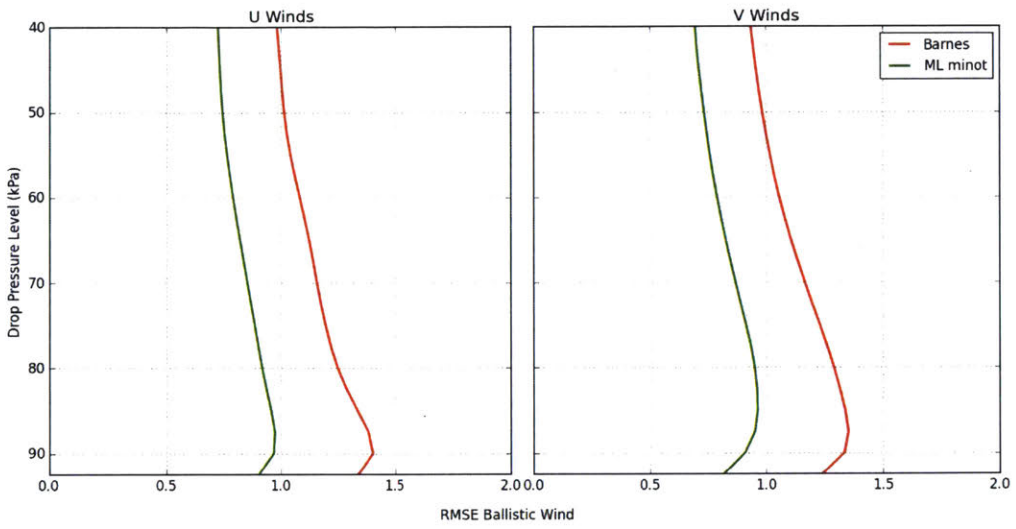

Figure 25 U and V Ballistic Wind RMSE. Minot
This figure repeats the procedure shown in Figure 19 and 16, but the data is from Minot,ND instead of Yuma. The performance is similar to the model from Yuma, and shows ML models can work at other geographic locations when they are trained on data from that location.

Figure 23-24 show the Ballistic wind errors for the RL 92 km northeast of the various DZ. This shows that models trained on different geographic areas can have similar results when testing at the training DZ.

4.3.4 Transferring Models from one DZ to a Different DZ

Aircrews could perform airdrops missions almost anywhere in the world. Unfortunately, there is no set of forecasts and analyses, with a fine enough resolution, that span the globe, and it is unlikely that one exists. For an attempt at a global solution, models were trained on some set of DZs and then tested on totally separate DZs.

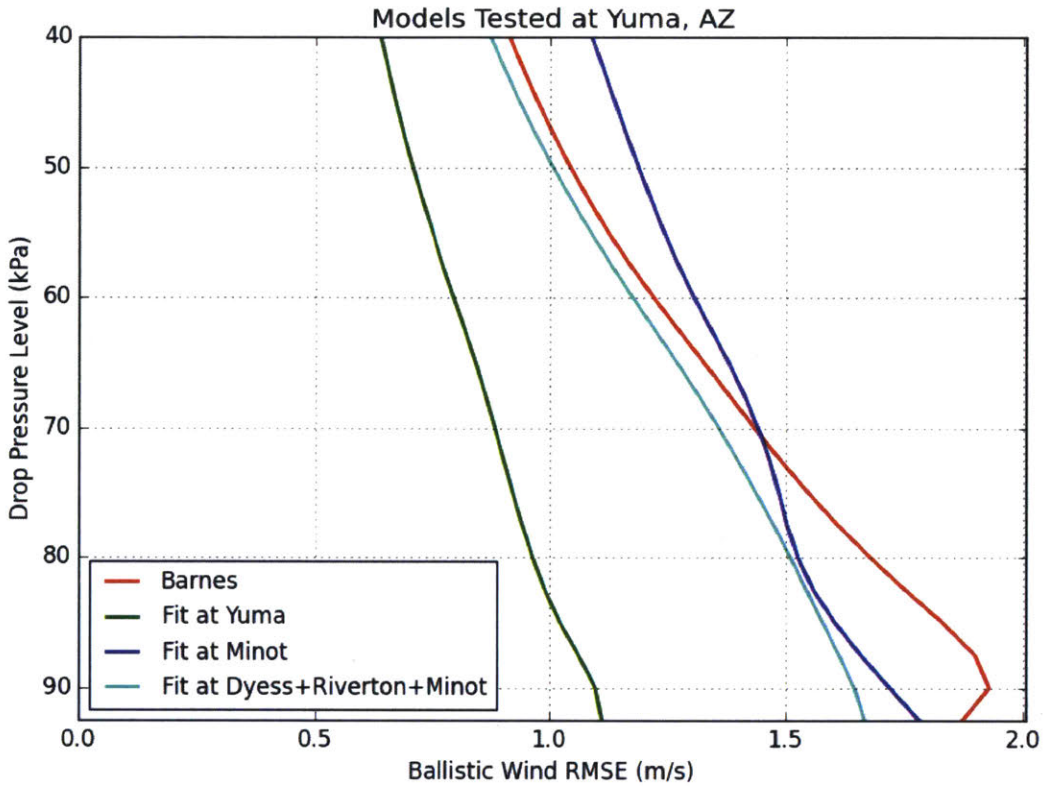

Figure 26: Training and testing on separate sets of DZs, U Ballistic Winds. Previously, it was established that models at different geographic locations could successfully be fit and tested on data from that location. This figure establishes that models trained on one DZ can be successfully used on data from another DZ. The model trained on one DZ, Minot, and tested on another DZ, Yuma, still outperforms the Barnes scheme at lower altitudes. The Model trained on multiple DZs, Dyess Riverton Minot, outperforms the model trained on one other DZ and the Barnes scheme.

Figure 26 shows results for models trained and tested on separate sets of DZs. The errors shown in these plots were all based on test data from Yuma. The models were trained on a single or multiple DZs. The legend shows the training location for the different models. Unsurprisingly, the model with the lowest error was the one fit and tested on the same location, Yuma. The model trained on only one location, Minot, ND, that was different than the test location showed significantly worse performance, but still outperformed the Barnes scheme at lower altitudes.

This is an encouraging result that shows, in at least some instances, models fit on one location can be used at other locations.

Figure 26 also shows errors for a model trained on Dyess, Riverton, and Minot then tested on Yuma. While this model also did not perform as well as the model trained and tested on the same location, it outperformed Barnes and the model trained on only one DZ. This increase in performance, while marginal, over the model trained on only one location shows that training on multiple locations can improve a model's performance when testing on other DZs.

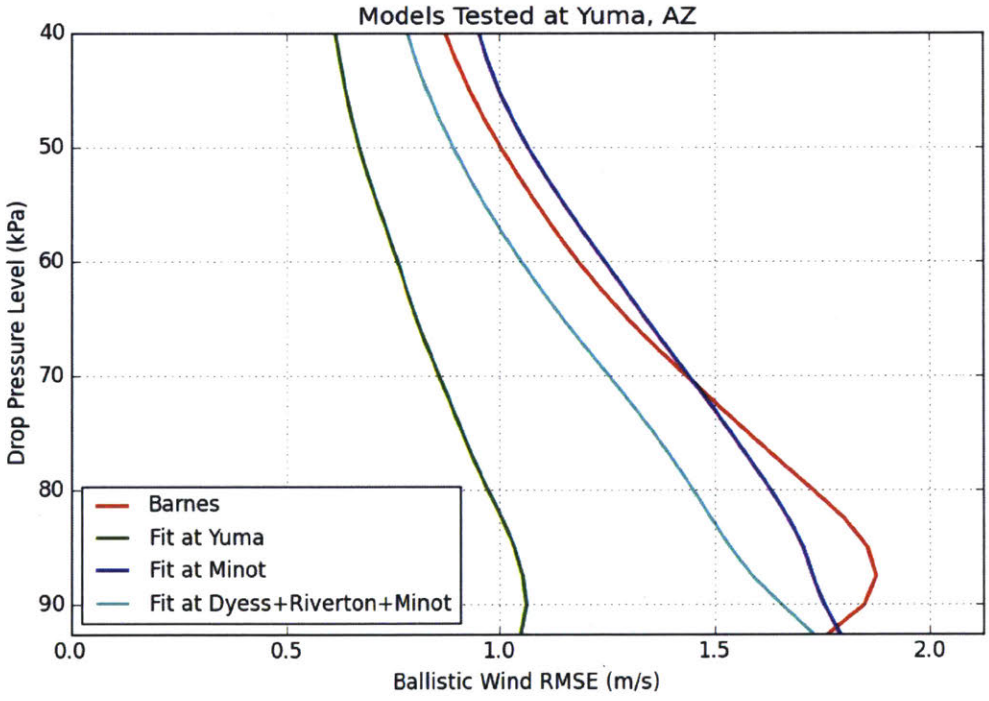

Figure 27: Training and testing on separate sets of DZs, V Ballistic Winds. This figure repeats the results from Figure 26 for the V ballistic winds. This figure establishes that models trained on one DZ can be successfully used on data from another DZ. The model trained on one DZ, Minot, and tested on another DZ, Yuma, still outperforms the Barnes scheme at lower drop altitudes. The Model trained on multiple DZs, Dyess Riverton Minot, outperforms the model trained on one other DZ and the Barnes scheme at all altitudes.

Figure 27 repeats the results from Figure 26 for the v ballistic winds. The model trained on data from only the Minot DZ outperforms the Barnes scheme at lower altitudes, but it is worse at lower altitudes. The model trained on the data from three DZs, Dyess + Riverton + Minot, outperforms both the Minot only model and the Barnes scheme at all drop levels. Like the case for the u ballistic winds, the models trained on other DZs did not recover the performance of the model trained and tested on the Yuma DZ dataset. This shows a trend where training on multiple DZs improves performance, and models can be used DZs. Determining how much data is beneficial is to be determined.

4.4 Quantifying a Dynamic Uncertainty of the Prediction

4.4.1 Estimating the correlation between error CDFs

For each pressure level and for each wind component, a set of quantile regression models produce a wind CDF for each prediction. A Monte Carlo procedure, outlined in Section 3.5, integrates through random variables, wind CDFs, associated with each pressure level. One of the interesting aspects of this Monte Carlo is that the random variables are modeled as correlated through a Gaussian copula.

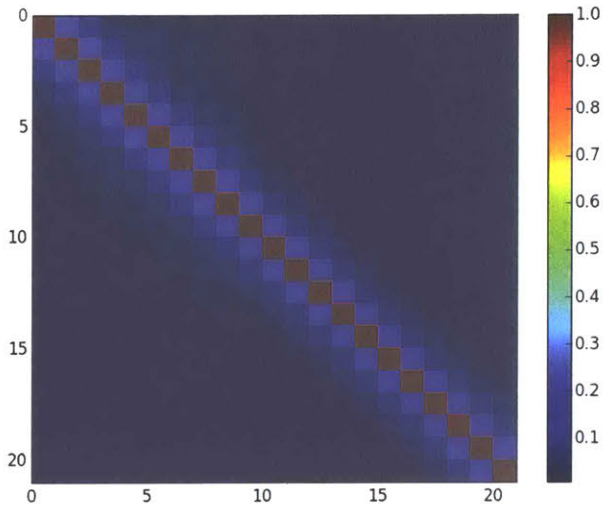

Figure 28 Correlation Matrix for Monte Carlo Procedure 4.
This correlation matrix is used in the ballistic wind Monte Carlo to produce uncertainty CDFs for the ballistic wind estimate. The correlation matrix was calculated using Procedure 5: Evaluating The Performance of Parameter r.

Figure 28 shows the correlation matrix for the model at Yuma RL (15,15). Pressure levels that are close to each other are more correlated than pressure levels that are further away.

4.4.2 Sample Output for Ballistic Wind Error

For each ballistic wind estimate, the Monte Carlo procedure from Section 3.5 produces a prediction error CDF. This specifies the estimated possible values for the ballistic wind error and their associated likelihood.

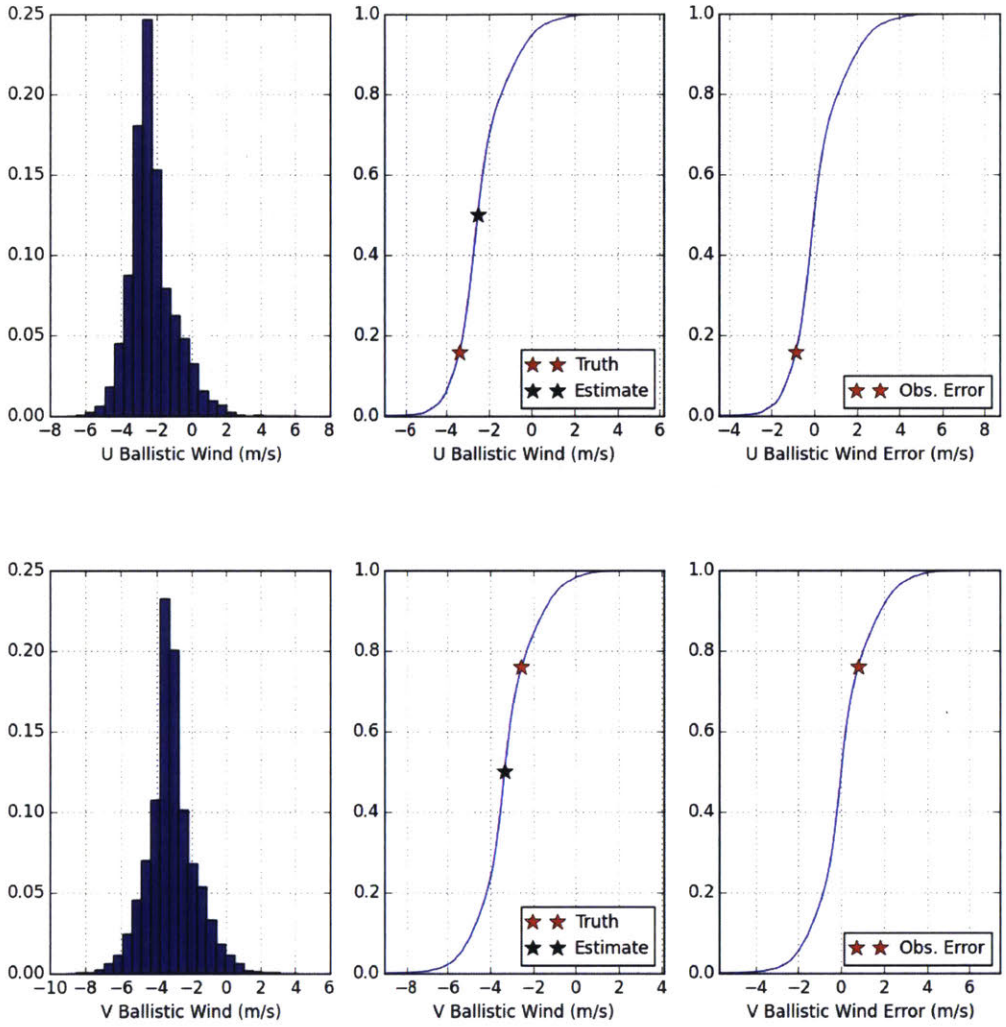

Figure 29: Sample Monte Carlo Output.
For each drop, the ballistic wind Monte Carlo produces a distribution of possible winds over the DZ. This is an example output for one drop from 85 kPa Pressure Altitude.

Figure 29 shows a sample output for the Monte Carlo for at the Yuma Model with RL (15,15). The leftmost subplot shows the density for the possible wind and the right plot shows the error CDF.

4.4.3 Dynamic Uncertainty Performance

For each drop time and for each pressure level, a new uncertainty CDF is produced. To evaluate these CDFs, the average observed quantile is observed. The observed quantile can be found for any sample where a prediction is made and the truth is later observed. The procedure is in Section 3.5. On average, the observed quantiles should follow a standard uniform distribution $U(0,1)$.

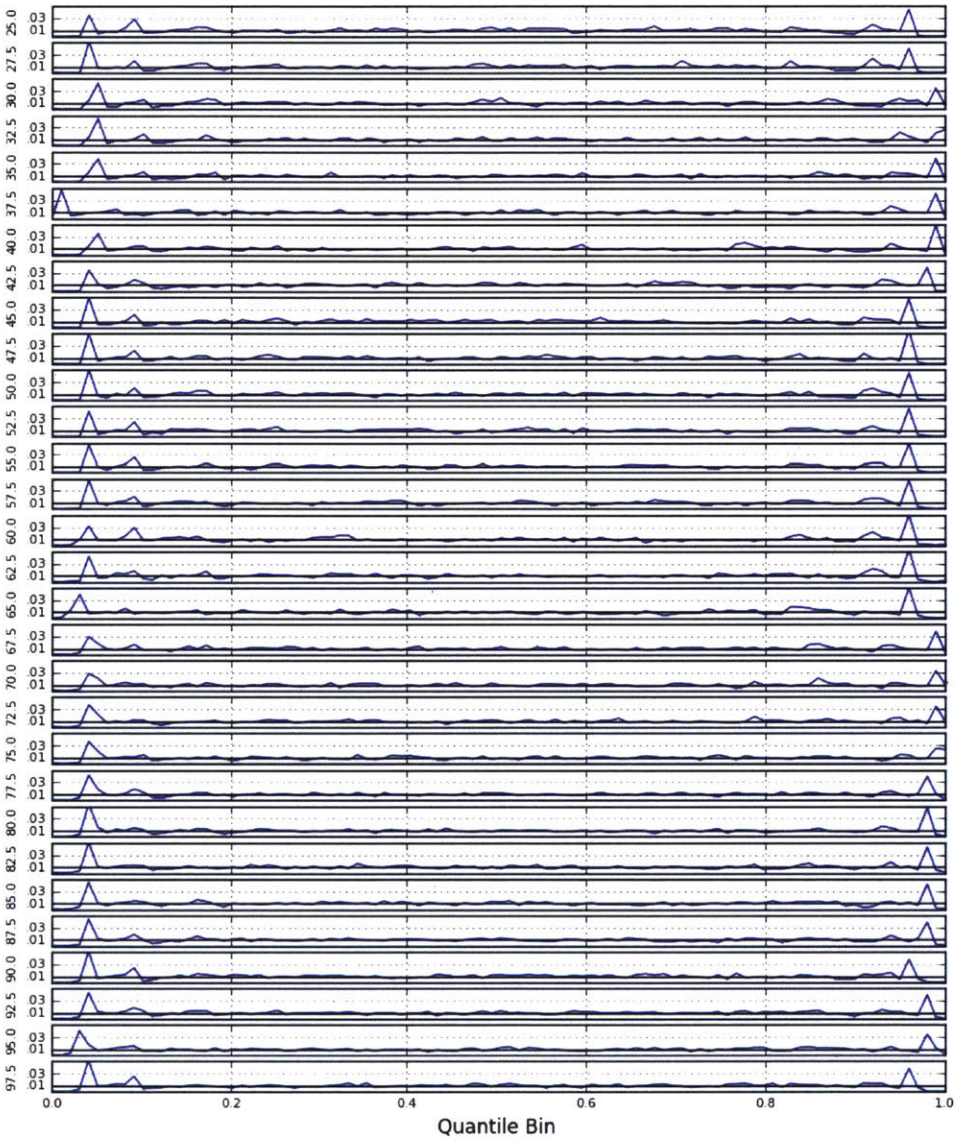

Figure 30 U Wind Distribution of Observed Quantiles for Each Pressure Level. For each pressure level there are a set of quantile regression models that, together, produce an uncertainty CDF for the winds on that level. This shows the normalized number of observations that fall into quantile bins of width 0.01. Ideally, the observations would be uniformally distributed between each quantile bin. This demonstrates an 'average error' for the quantile regression models. They are well behaved at most quantiles but perform relatively poorly near the tails. The atmospheric pressure (kPa) is displayed by the y-axis for each pressure level.

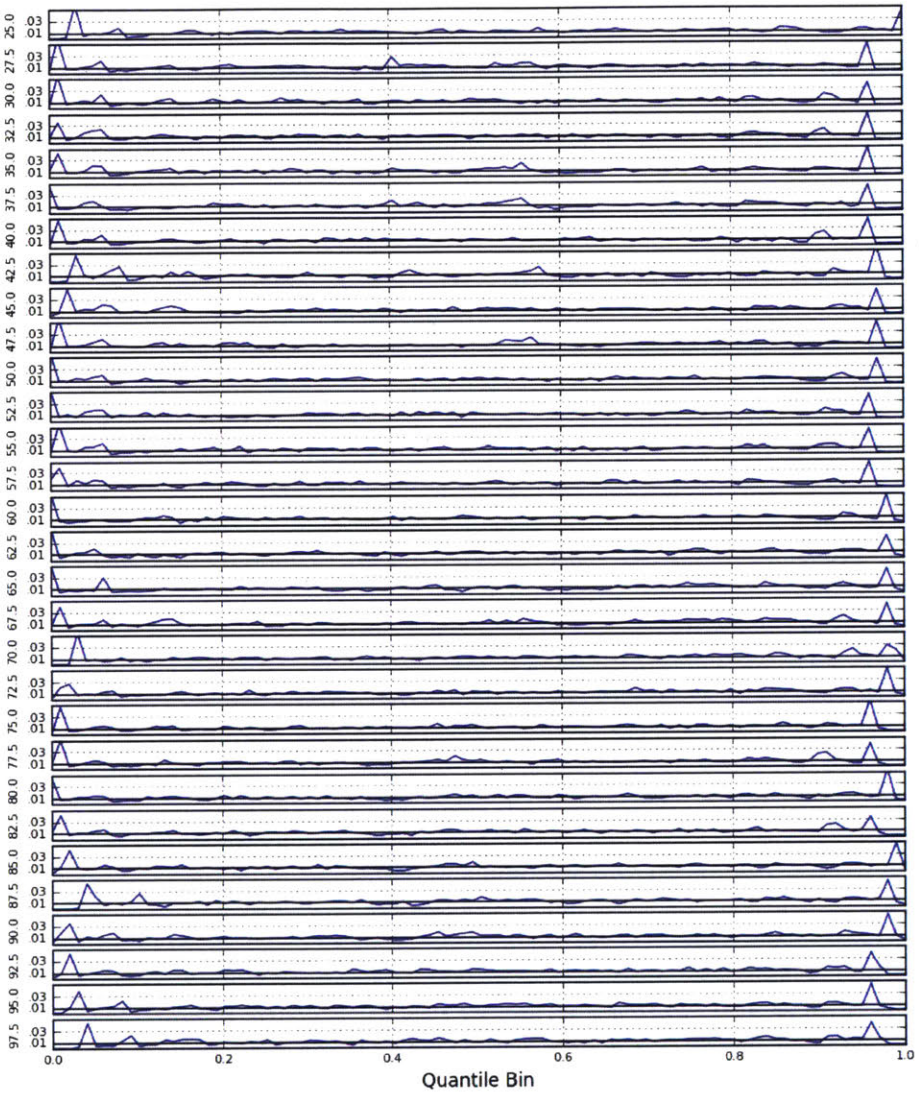

Figure 31: V Wind Distribution of Observed Quantiles for Each Pressure Level. For each pressure level there are a set of quantile regression models that, together, produce an uncertainty CDF for the winds on that level. This shows the normalized number of observations that fall into quantile bins of width 0.01. Ideally, the observations would be uniformly distributed between each quantile bin. This demonstrates an 'average error' for the quantile regression models. They are well behaved at most quantiles but perform relatively poorly near the tails. The atmospheric pressure (kPa) is displayed by the y-axis for each pressure level.

Figure 30 shows the distribution of observed quantiles in the test data set for each the prediction CDFs for each pressure level.

For each sample, the wind prediction system finds an error CDF for the ballistic wind. Ideally, the true ballistic wind would be less than or equal to the winds at the τ^{th}quantile $\tau * 100\%$ of the samples.

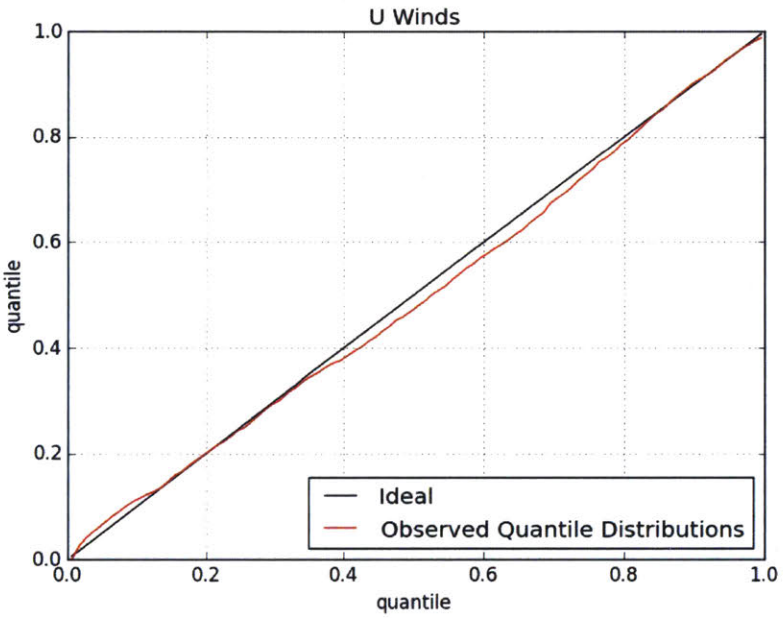

Figure 32 U Ballistic Wind QQ Plot for Ballistic Wind CDFs.
This plot shows the performance of the ballistic wind CDFs produced for a drop from the 65 kPa pressure level. The QQ plot shows actual number of observations below a quantile vs the ideal number of observations below a quantile. Data from the Yuma RL (15,15) model.

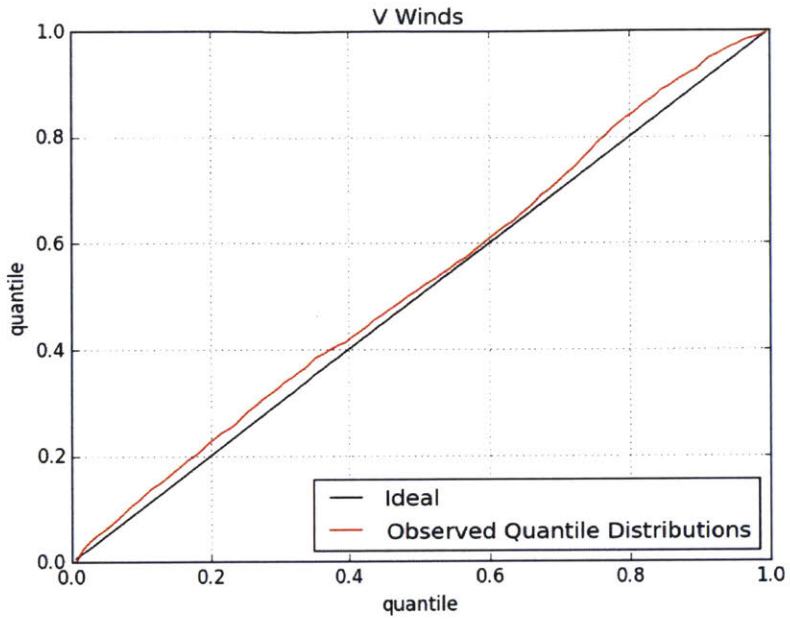

Figure 33 V Ballistic Wind QQ Plot for Ballistic Wind CDFs.
This plot shows the performance of the ballistic wind CDFs produced for a drop from the 65 kPa pressure level. The QQ plot shows actual number of observations below a quantile vs the ideal number of observations below a quantile. Data from the Yuma RL (15,15) model.

Figure 323 show the QQ plots for the u and v winds for the Yuma RL (15,15) model. These plots show that the dynamic uncertainty estimate accurately reflects the error distribution on average. These error distributions can be used to derive important mission risks like the probability of a drop landing off of a DZ or the probability of hitting a protected object on the DZ.

4.5 Accuracy in Estimating the Payload Drift

Thus far, Sections 4.3 and 4.4 showed that the machine learning (ML) models outperform the currently used Barnes scheme. Figure 17 RMSE for U wind estimate on each pressure level. showed that the ML models outperform Barnes when estimating DZ winds for each pressure level. Procedure 4 shows a method for integrating the wind stick over the DZ to estimate a

ballistic wind. This ballistic wind represents the total wind a payload would experience during a drop. Figure 19 and 16 showed that the ML models also outperform Barnes in predicting ballistic winds over the DZ. The ballistic wind is found by the drift divided by the time of fall. Section 4.3.1 showed ballistic wind values, plotted against drop pressure levels, for a normalized set of payload parameters.

$$\sqrt{\frac{m}{sC_d}} = 1 \quad (44)$$

Where m is the total payload mass, s is the cross-sectional area, and C_d is the drag coefficient. Answering whether the estimations are accurate enough is dependent on these payload parameters. If one substitutes typical payload parameters, one can get a better idea of how the system will perform.

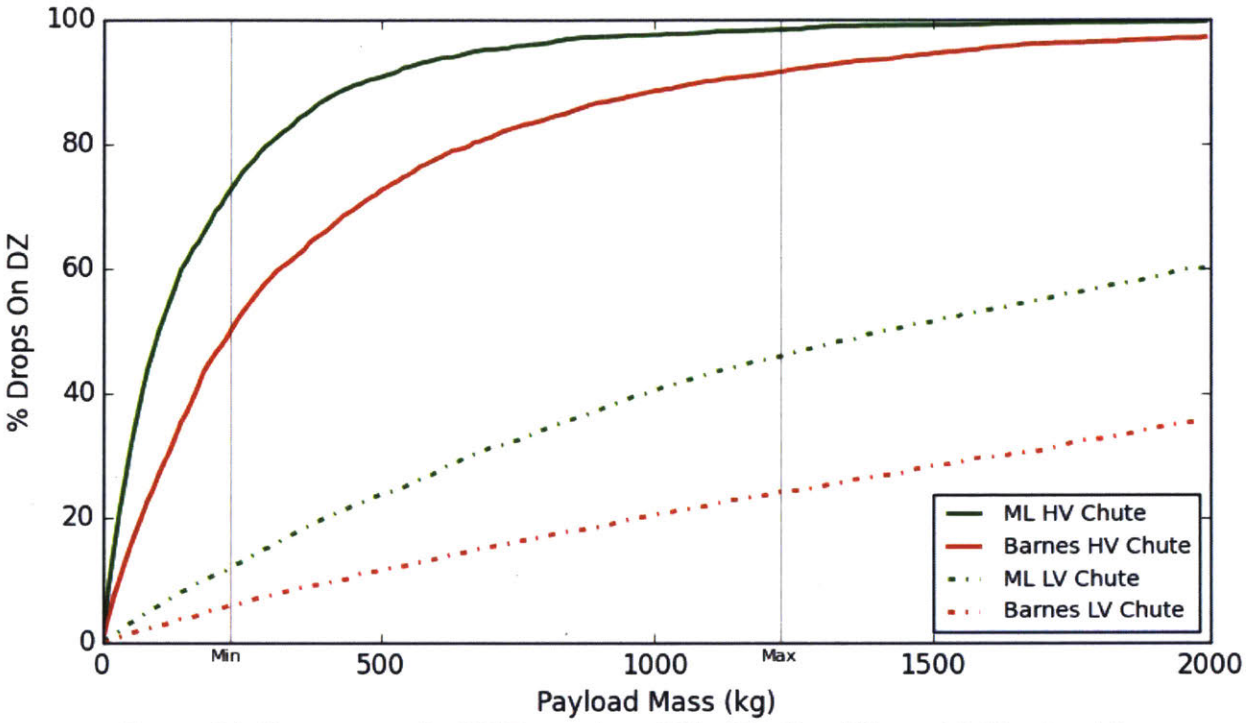

Figure 34: Percentage On DZ Drops for a DZ of Radius 400 m with Varying Mass. Drops are from 40 kPa pressure altitude. Data is from the test set for the DZ at Yuma and the dropsonde location 92 km northeast of the DZ. 'HV' denotes a high-velocity chute, and 'LV' denotes a low-velocity chute. This plot also shows the ML models outperform the Barnes scheme. Note that drops with a higher fall rate (high-velocity chute and high payload mass) tend to be much more accurate. This is simply because wind estimation errors have less time to blow the payload off target.

Figure 34 shows the percentage of drops that would hit the DZ from the test set at Yuma, AZ and the RL 92 km northeast of the DZ. A missed drop is defined as landing more than 400 meters from the DZ. The model was fit and tested based on dropsondes released 92 km northeast of the DZ. The payload was simulated as released form 40 kPa which is approximately 7,100 meters altitude MSL. These results are based on using a both high-velocity and low-velocity parachutes with $sC_d = 219\ m^2, 283.5\ m^2$. Clearly payloads with more mass have a higher fall rate than payloads with less mass. A lower fall rate causes more wind dependent drift, so wind estimation

errors become more pronounced and cause more off DZ drops when using low-velocity chutes. For the best accuracy, one should always choose high mass payload with high velocity chutes. Some payload demand a low impact velocity to prevent damage, so low velocity chutes are used in some instances. Based on the average results in Figure 34, low-velocity chutes greatly decrease mission accuracy.

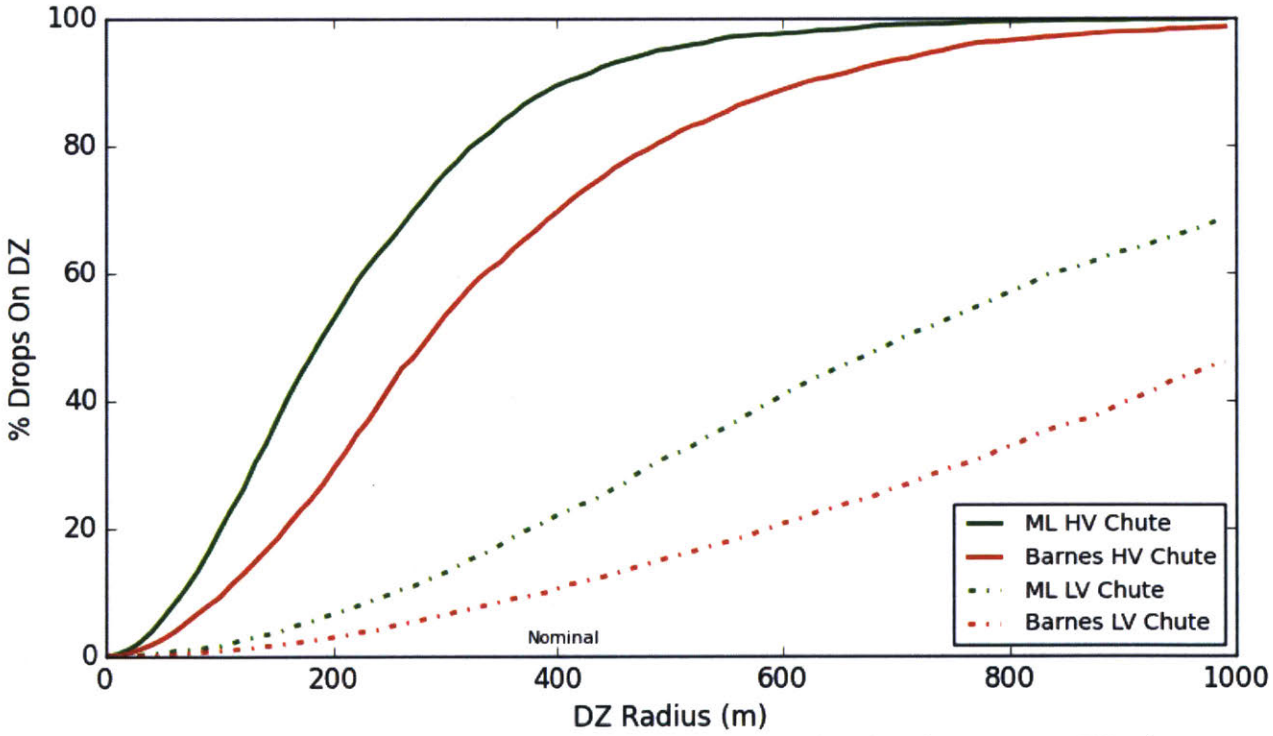

Figure 35: Percentage of on DZ Drops for a 750 kg payload with Varying DZ Radius. Drops are from 40 kPa pressure altitude. Data is from the test set for the DZ at Yuma and the dropsonde location 92 km northeast of the DZ. 'HV' denotes a high-velocity chute, and 'LV' denotes a low-velocity chute. This plot also shows the ML models outperform the Barnes scheme. Note that the gains over the Barnes scheme increase with DZ radius for the LV chutes while the gains decrease with radius for the HV chutes.

Figure 34 showed the miss percentage based on a DZ with a 400 meter radius. DZs, just like people, come in all shapes and sizes. Figure 35 shows how the miss percentage varies for a nominal payload mass, 750 kg, released from 40 kPa. As was shown previously, the high-velocity chute greatly outperforms the low-velocity chute. Unsurprisingly, the hit percentage increases as the size of the target increases. For very large DZs, the ML models offer very little advantage over the Barnes scheme when using high-velocity chutes. The gains over Barnes are more pronounced with the low-velocity cases, but based on the average performance, it is a risky mission for either estimation method.

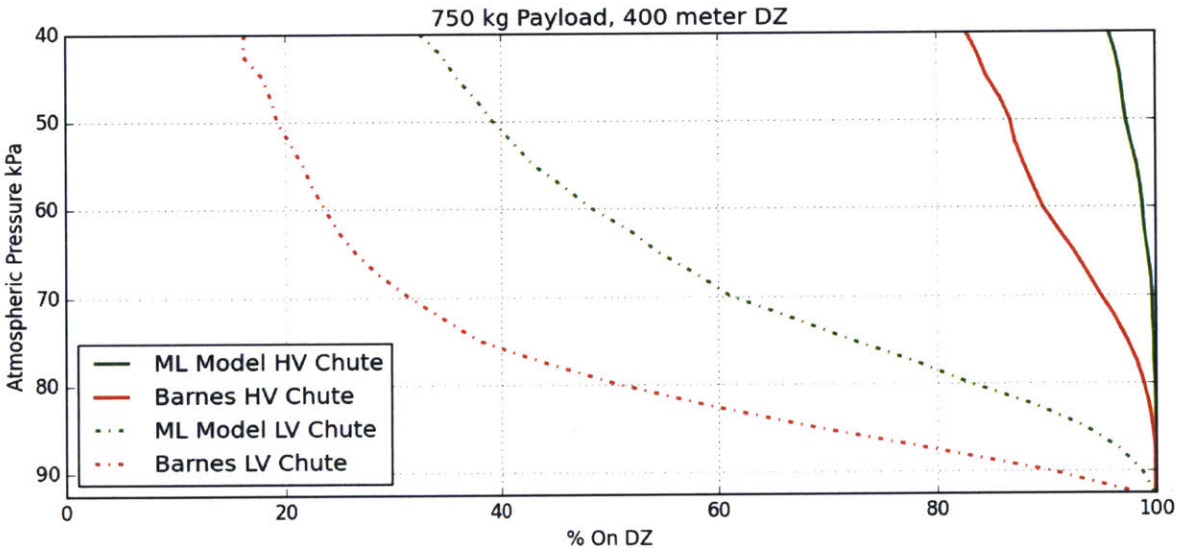

Figure 36: Miss Percentage for a Various Drop Altitudes for a Nominal Drop. These results are based on a 750 kg payload and a 400 meter radius DZ. These are both typical parameter for real drops. The ML models outperform Barnes again in all cases. The high-velocity chutes (HV) again are much high performing than the low-velocity chutes (LV). The hit percentage also increases as the drop altitude decreases. Lower altitude drops have less time for wind estimation errors to propagate and cause off DZ drops.

Figure 36 shows the miss percentage for typical payload released from various drop altitudes. Again, the ML models outperform the Barnes scheme in all cases. The high-velocity chutes also

prove to much more consistent than the low-velocity chutes. Drops are also much more accurate from lower drop altitudes. At lower altitudes, there is less time for wind estimation errors to propagate and push the payload off target.

5 Conclusions

Aircrews demand a method for accurately estimating the winds over a drop zone without making a direct measurement. Discussed thus far is a method for assimilating remote dropsondes with deterministic forecasts using machine learning. There are a variety of weather measurement assimilation schemes available, but the method based on machine learning addresses the specific single pass airdrop problem.

Section 3.6 discussed a methodology for pulling predictive features from the weather forecast and remote dropsonde measurement. Using unprocessed weather forecasts is unfeasible because of the large number of states. This study used a generate and test procedure that balanced intuition with predictive results. Section 4.2 shows the feature selection methodology was successful in improving cross-validated accuracy by 0.6% and reducing the size of the feature set by 69%.

Section 4.3 summarizes the predictive accuracy when estimating the winds over of a DZ near Yuma, AZ. This study first established, that on a per pressure level basis, the machine learning models outperformed the Barnes scheme by 0.5 to 1 m/s RMSE. A ballistic wind is a density weighted integration of the winds in a column of air, and it was shown the machine learning models also outperform the Barnes scheme in predicting this metric by 0.2 to 0.6 m/s RMSE. The results were repeated for a map of RLs surrounding the Yuma DZ. The Barnes scheme showed a strong dependence on the distance between the RL and the DZ with RMSE values between 1.2 and 2.3 m/s, and the machine learning models showed a much weaker dependence with RMSE values between .8 and 1.1 m/s.

Airdrop missions must be conducted on locations without a large historical dataset of weather forecasts. Section 4.3.4 showed that models trained on a set of DZs outperform the Barnes scheme when tested on a DZ unseen during training by up to 0.3 m/s RMSE. This shows it is useful to learn on multiple DZs, and it may be possible to produce a global solution with this methodology.

Section 3.5 detailed a methodology for inferring the uncertainty of a wind prediction using quantile regression. Section 4.4 shows that this methodology produces a dynamic uncertainty estimate unique to each airdrop mission, and the uncertainty estimate accurately reflect the error in simulated in the drops. This uncertainty estimate is useful for inferring the probability of various failure scenarios like a payload missing a DZ or a payload hitting a protected object.

Section 4.5 shows the accuracy for various simulated drop scenarios. It is shown that the machine learning models result in approximately 20% more drops hitting the DZ for nominal drop weights and altitudes. These simulations are based on the assumption that the winds in the weather analysis would be the true winds experienced by the payload, and there would be no other errors during a drop that would push the payload off the DZ. This is a strong assumption, but the on DZ percentages presented in Section 4.5 are still useful for estimating the effect of wind estimation errors.

5.1 Comparison to other methods

Thus far, most results have compared the developed ML method with the Barnes scheme. The Barnes scheme was not designed specifically for this task, and perhaps unsurprisingly, it is not the best suited. It is treated as a baseline because it is the method currently used to assimilate dropsondes with deterministic forecasts in the airdrop community. There are a variety of other

methods that might work for this dropsonde assimilation task, and some might be worth pursuing in the future.

Section 2.1.1 briefly touched on how the ensemble Kalman filter could be used for the remote dropsonde mission. It checks most of the mission requirement boxes. One could pick the optimal dropsonde location based on the ensemble forecast covariance matrix. After releasing a sonde, the filter could provide an optimal estimate, in the information theory sense, of the winds anywhere in the weather cube. The final answer would also come with an updated uncertainty that is dependent on the dropsonde and forecast uncertainty. Current data limitations make the EKF unusable. In the purest form, a forecast ensemble is typically 12-24 times larger than a standard deterministic forecast. In some cases this is 12-24 times more data than an aircrew is able to download before a drop. It might be possible, and worth pursing, to develop a method for compressing the information in the forecast ensemble to a size small enough to download. There are many open ended questions when it comes to the ensemble Kalman filter. What is the accuracy when applied to the remote dropsonde mission? How much data can aircrews download before a mission? If the data needed to be compressed, how could one compress the data while maintaining acceptable performance? All of these questions are equally important and require significant development time before they can be answered.

The results in this study suggest the best method might be another machine learning algorithm. The Boosted Regression Tree algorithm, that was used in this study, operates on a single processor core, and therefore scales poorly. Section 4.3.4 showed some encouraging results where models are trained on sets of multiple of geographic locations (DZs). Training on multiple DZs greatly increases the training data, and how many DZs should be included in a training set is still an open question. GPU trained neural networks have been used to solve the greatest

challenges in the function approximation domain, but they can demand large amounts of data to be successful. Training on multiple DZs opens up much more data for training, and it is likely that neural networks could better learn from the increased data set.

5.2 Data Dependency

The results in Chapter 4 were for models fit on data from one or a few geographic locations and a time period from mid-2012 until late-2016. The size of the dataset is a function of both development time and data availability. We found that models started to outperform Barnes when approximately 15,000 samples were available for training. These models were fit using approximately half of the total available data and tested on data from the same geographic location, and it was observed that the models generally improved as more training data was used. This study used the RAP dataset available from NOAA. NOAA is still producing forecasts and analyses using the RAP system, so it is likely the models could continue to improve as more data becomes available. There is likely a point where this methodology reaches diminishing returns for the data set size, but that point has not been reached in this study.

Section 4.3.4 showed some results for models trained on a few DZs and tested on data from a different DZ. There the performance was shown to increase as more DZs were added, but the performance did not reach that of the model trained and tested on the same location. It has been discussed that airdrop missions could be planned for locations where histories of forecasts and analyses are not available. Even if this data was available for the entire globe, it is computationally infeasible to train models for every potential DZ. This fosters the demand for models that can perform on geographic locations different than the fit location. The question of how much data and how many geographic locations is unanswered in this study.

5.3 Run Time

(Friedman, 2001) states that the training time for the Tree Boosting algorithm scales linearly with both the number of features and the number of samples in the model. Figure 13 shows an example of how the training time scales with the number of features included in the model. If one was lucky enough to have historical data for a DZ, these figures would give an idea about the amount of time needed fit a model for the drop. Typically, preliminary planning for a drop begins days or weeks ahead of time, so there would clearly be enough time to train a model for a specific location in the event that the data is available. Ideally, a set of models will be trained offline and either a combination of the set or a selected member from the set will be used for a drop. This makes the complexity for a producing an estimate from the model the key metric.

Producing an estimate from the models falls into two phases: data preprocessing and execution. Preprocessing includes: subsampling the forecast area to the DZ, generating the features from Section 3.6.1.1, producing the PCA features according to Section 3.6.1.2, and preparing the dropsonde features according to the method in Section 3.6.1.4. Execution includes: obtaining an estimate for each wind component for the mean winds on each pressure levels, getting quantile estimates for discrete quantile of each wind component on each pressure level, estimating a ballistic wind for the mean estimates, and finally running the ballistic wind Monte Carlo outlined in Section 3.5. Example run times for each phase is shown in Table 7. This is based on a drop with 30 pressure levels and 10,000 Monte Carlo runs for the both the u and v winds.

Table 7: Runtime for Subprocesses

Data Preprocessing	.040 s
Mean Execution	.057 s
Quantile Execution	.009 s
Monte Carlo Execution	16.9 s

5.4 Future Work

5.4.1 Scaling to a global solution

The greatest limitation of this study is that it only shows the method works on four DZ locations. The method also works best for data from geographic locations included in the training set. Section 4.3.4 shows that there are performance gains over the Barnes scheme when testing on geographic locations not included in the training set. While this is an encouraging result, it is still an open question of if or when a model's performance will reach that of model trained on the test location. The ultimate solution will be a system that can be used at any location.

One solution to the global solution might be training on as many geographic locations as possible. If a model is exposed to data from every location in North America, would that model be useful at places outside North America? 4.3.4 shows moderate success for training on three DZs and testing on a fourth, but learning on hundreds or thousands would likely be a much different problem.

A different approach to the global solution might be training a representative set of models. One could define a set of terrain descriptions like: mountainous, coastal, and flat. One could define another set of climates: tropical, desert, mild, and tundra. One could build a model based on the cross-product of the terrain set and climate set to form a set of representative DZs. Ideally, there would be training set of forecasts and analyses matching the characteristic of each of the representative DZs, and one could train the models for the representative sets. To perform a drop for any new DZ, one could try to classify the DZ into one of the representative sets or use a blend of the representative sets. Finding the locations for the representative set, identifying the best representative set member for a new DZ, and answering how many members of the representative set is sufficient are all difficult problems.

5.4.2 Finding the best dropsonde location

Before one could use the methods outlined so far, a dropsonde must be released en route of the DZ. We have trained models for each potential remote location grid point and evaluated their average performance. Based on the average performance through time, one would want to release the dropsonde as close to the DZ as possible when using both ML models and the Barnes scheme. For any one drop time, the dropsonde location that would give the best final answer could be anywhere on the map, and again, this true for both the ML models and the Barnes scheme. A system is needed to pick out the best dropsonde location before the dropsonde is released. The system would ideally take deterministic forecast as input, and return as output, an estimate of the miss distance for a remote dropsonde mission for each potential location. This would be a miss distance estimate for each remote location and each potential release altitude. The system would allow a mission planner to select the best dropsonde location based on predicted miss distance.

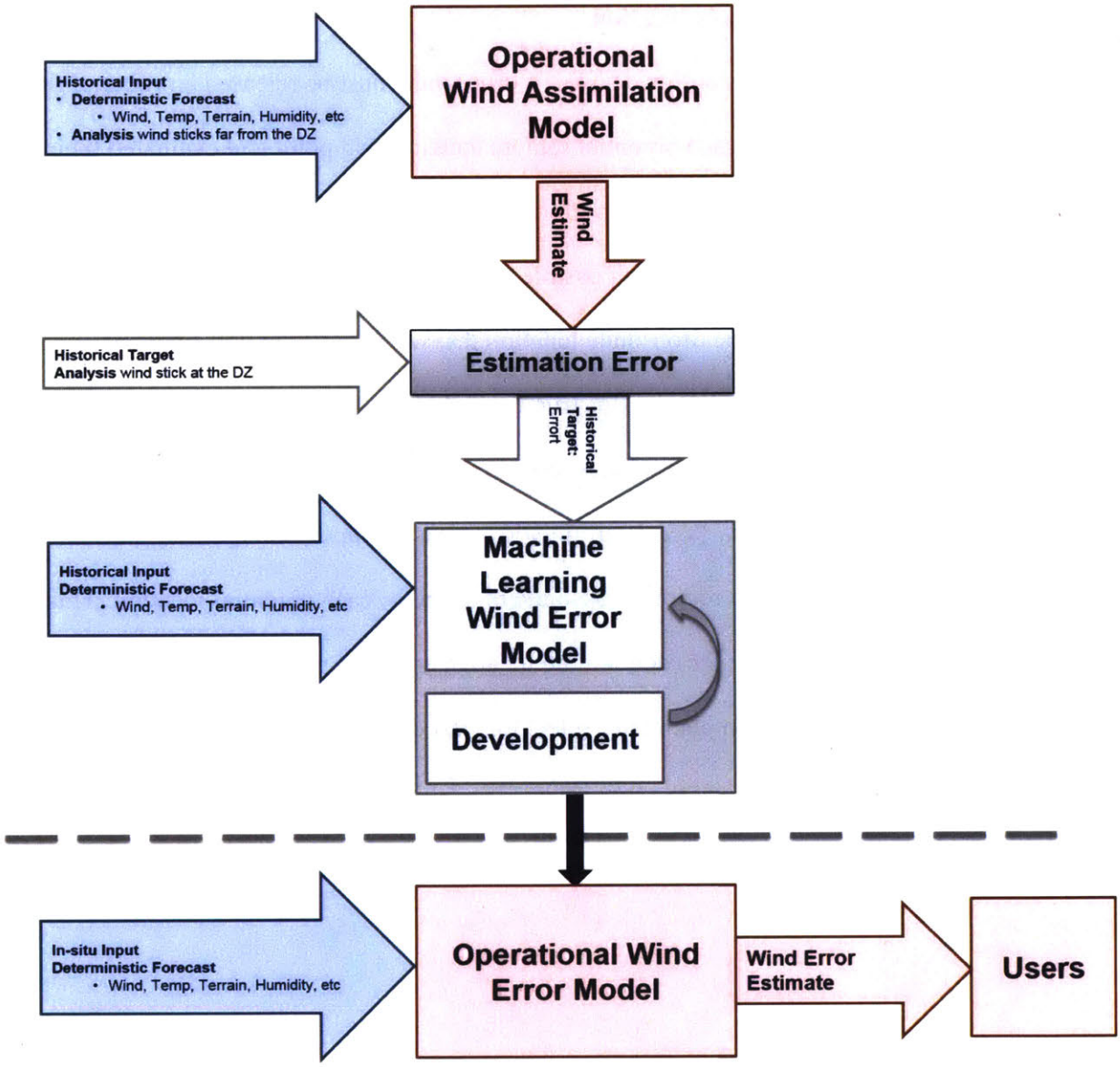

Figure 37 Flowchart for Wind Error Estimation Model development. This model was take forecast features as input and try to predict the final wind error of the wind estimation models (Model A). The training set would come from Model A's estimations errors in its test set.

A possible method, shown in Figure 37, for producing this system would be to fit ML models, wind error models, with forecast features as input and the error from an estimation method, wind assimilation models, as the response. It would be best to use examples from the test set of the

wind estimation models, so the total dataset for the wind error model would be severely limited in comparison.

6 Bibliography

Barker, D., Huang, X.-Y., Liu, Z., Auligné, T., Zhang, X., Rugg, S., & Ajjaji, R. (2012). The Weather Research and Forecasting Model's Community Variational/Ensemble Data Assimilation System WRFDA. *American Meteorological Society*, 831-843.

Barnes, S. L. (1964). A Technique for Maximizing Details in Numerical Weather Map Analysis. *Journal of Applied Meteorology Volume 3*, 396-409.

Benney, R., Barber, J., McGrath, J., McHugh, J., Noetscher, G., & Tavan, S. (2005). The Joint Precision Airdrop System Advanced Concept Technology Demonstration. *18th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar.* Munich, Germany: AIAA.

Benney, R., Barber, J., McGrath, J., McHugh, J., Noetscher, G., & Tavan, S. (2005). The New Military Applications of Precision Airdrop Systems. *Infotech@Aerospace.* Arlington.

Benney, R., Henry, M., Lafond, K., & Meloni, A. (2009). DODNEWJPADS PROGRAMS & NATO ACTIVITIES . *20th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar.* Seattle, WA: AIAA.

Bergthorsson, P., & Döös, B. (1955). Numerical weather map analysis. *Tellus 7 (3)*, 329-340.

Bohdalová, M., & Šlahor, Ľ. (2014). *Monte Carlo Simulations of the multivariate distributions with different marginals.*

Bouttier, F. (1994). A Dynamical Estimation of Forecast Error Covariances in an Assimilation System. *Monthly Weather Review*, 2376-2390.

Bowman, K. B. (2011). AFRL Precision Air Drop. *12th Annual Science & Engineering Technology Conference/DoD Tech Exposition*, (pp. 2,4,5).

Bremnes, J. B. (2004). Probabilistic Wind Power Forecasts Using Local Quantile Regression. *Wind Energy*, 47-54.

Cacuci, D. G., Navon, I. M., & Ionescu-Bujor, M. (2014). *Computational Methods For Data Evaluation and Assimilation.* Boca Raton, FL: CRC Press.

Cressman, G. (1959). An operational objective analysis system. *Monthly Weather Review 87*, 367-374.

Eaton, J. A. (2012). *Point of Impact: Delivering Mission Essential Supplies to the Warfighter through the Joint Precision Airdrop System (JPADS).* Cambridge, MA: Massachusetts Institute of Technology.

Embrechts, P., Lindskog, F., & McNeil, A. (2001). *Management, Modelling Dependence with Copulas and Applications to Risk.* Zürich: Department of Mathematics ETHZ CH-8092.

Evensen, G. (1994). Sequential data assimilation with a nonlinear quasi-geotrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research, VOL. 99, No. C5*, 10143-10162.

Evensen, G. (2003). The Ensemble Kalman Filter: theoretical formulation and pratical implementation. *Ocean Dynamics 53*, 343-367.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics Volume 29, Number 5*, 1189-1232.

Friedman, J. H. (2002). Stochastic Gradient Boosting. *Computational Statistics & Data Analysis - Nonlinear methods and data mining*, 367 - 378.

Gerlach, A. R., & Doman, D. B. (2017). Analytical Solution for Optimal Drogue-to-Main Parachute Transition Altitude for Precision Ballistic Airdrops. *Journal of Guidance, Control, and Dynamics, Vol. 40, Special Issue on Computational Guidance and Control*, 439-452.

Google Maps. (2017, 5 25). Retrieved from Yuma Proving Grounds 33°00'55.9"N, 114°15'09.9"W.

Grim, J. J. (2015). Wind profiles from remote dropsondes as proxies for wind profiles at a DZ: inferences from simulations and observations. . *DOD Airdrop Weather Working Group meeting.* Cambridge, MA.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York: Springer Science +Business Media, LLC 2009.

Jones, W. (2016). Lidar Wind Sensing for Improved Precision AirDrop and Gunship Wind Sensing. *18th Coherent Laser Radar Conference* , (pp. 1-5).

JR., W. I., & Bryson, R. A. (1966). An Investigation of the Potential of Component Analysis for Weather Classification. *Monthly Weather Review*, 697-709.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME - Journal of Basic Engineering (82)*, 35-45.

Kelly, K., & Pena, B. (2001). Wind Study and GPS Dropsonde Applicability to Airdrop Testing. *16th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar.* Boston, MA: AIAA.

Koenker, R. (2000). *Quantile Regression.* Champaign, IL: Department of Economics, Urbana-Champagn.

Krasnopolsky, V. M., & Fox-Rabinovitz, M. S. (2010). Development of Neural Network Convection Parameterizations for. *Neural Networks (IJCNN), The 2010 International Joint Conference on.* Barcelona, Spain: IEEE.

Manski, C. F. (1988). *p. 55.* New York, NY: Chapman & Hall.

Manual, U. S. (2013). *Unified Surface Analysis Manual.* Weather Prediction Center; Ocean Prediction Center; National Hurricane Center; Honolulu Forecast Office .

Markuzon, N., Regan, J., & Slesnick, C. (2012). Using a Weather-based Data Driven Approach For. *Infotech@Aerospace.* Garden Grove, CA: AIAA.

Meier, D. C. (2010). *Application of Satellite-Derived Wind Profiles to Joint Precision Airdrop System (JPADS) Operations.* Wright-Patterson Air Force Base, Ohio: Department of Engineering Physics Graduate School of Engineering and Management Air Force Institute of Technology .

Navon, I. M. (2009). *Data Assimilation for Numerical Weather: A Review.* Springer-Verlag Berlin Heidelberg 2009.

NOAA. (2017, May 19). *Rapid Refresh (RAP)*. Retrieved from ncdc.noaa.gov: https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/rapid-refresh-rap

PADS WiPPR: Wind Profiling Portable RADAR. (2015). Retrieved from qinetiq-na.com.

Papaefthymiou, G., & Kurowicka, D. (2009). Using Copulas for Modeling Stochastic Dependence in Power System Uncertainty Analysis. *IEEE TRANSACTIONS ON POWER SYSTEMS, VOL. 24*, 40-48.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Perrot, M. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research (12)*, 2825--2830.

Press, W., Teukolsky, S., Vetterling, W., & Flannery. (2007). *Numerical Recipes: The Art of Scientific Computing (3rd ed.).* New York, NY: Cambridge University Press.

R¨uschendorf, L. (2013). *Mathematical Risk Analysis.* Heidelberg: Springer Series in Operations Research.

Radhika, Y., & Shashi, M. (2009). Atmospheric Temperature Prediction using. *International Journal of Computer Theory and Engineering*, 55.

Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. *Publ. Inst. Statist. Univ. Paris (8)*, 229-231.

Staine-Pyne, E. (2009). Precision Airdrop Improvements . *AMC Industry Days* , (pp. 15, 20, 21).

(1976). *U.S. Standard Atmosphere 1976.* Washington, D.C.: National Oceanic and Atmospheric Administration.

Xydas, E., Qadrdan, M., Marmaras, C., Cipcigan, L., Jenkins, N., & Ameli, H. (2017). Probabilistic wind power forecasting and its application in the scheduling of gas-fired generators. *Applied Energy 192*, 382-394.

Yu, K., Lu, Z., & Stander, J. (2003). Quantile regression: applications and current research areas. *The Statistician 52 Part 3*, 331-350.

Appendix A

Table 8: Features Used to fit Models in Chapter 4.
These features are used to fit models and make predictions for the DZ winds. These are referenced in Section 4.2. 'rl' refers data from the remote location. 'dz' refers to data from the drop zone (forecast data only). 'mean stick' is the mean of the values in the column of air. 'mean lvl' is the mean value of a variable from a isobaric pressure level. 'var stick' is the variance of value from a column of air. 'var lvl' is the variance of a variable on a pressure level. 'PCA' refers to the features generated in Section 3.6.2. The number in the PCA feature strings refers to their position in the PCA vector.

1	Temperature_rl	99	PCA_U_Winds_40_0	197	PCA_U_Winds_38_1
2	Temperature_dz	100	PCA_U_Winds_41_0	198	PCA_U_Winds_39_1
3	Temperature_rl_mean_stick	101	PCA_U_Winds_42_0	199	PCA_U_Winds_40_1
4	Temperature_dz_mean_stick	102	PCA_U_Winds_43_0	200	PCA_U_Winds_41_1
5	Temperature_mean_lvl	103	PCA_U_Winds_44_0	201	PCA_U_Winds_42_1
6	Temperature_rl_var_stick	104	PCA_U_Winds_45_0	202	PCA_U_Winds_43_1
7	Temperature_dz_var_stick	105	PCA_U_Winds_46_0	203	PCA_U_Winds_44_1
8	Temperature_var_lvl	106	PCA_U_Winds_47_0	204	PCA_U_Winds_45_1
9	Relative humidity_rl	107	PCA_U_Winds_48_0	205	PCA_U_Winds_46_1
10	Relative humidity_dz	108	PCA_U_Winds_49_0	206	PCA_U_Winds_47_1
11	Relative humidity_rl_mean_stick	109	PCA_V_Winds_0_0	207	PCA_U_Winds_48_1
12	Relative humidity_dz_mean_stick	110	PCA_V_Winds_1_0	208	PCA_U_Winds_49_1
13	Relative humidity_mean_lvl	111	PCA_V_Winds_2_0	209	PCA_V_Winds_0_1
14	Relative humidity_rl_var_stick	112	PCA_V_Winds_3_0	210	PCA_V_Winds_1_1
15	Relative humidity_dz_var_stick	113	PCA_V_Winds_4_0	211	PCA_V_Winds_2_1
16	Relative humidity_var_lvl	114	PCA_V_Winds_5_0	212	PCA_V_Winds_3_1
17	U_Winds_rl	115	PCA_V_Winds_6_0	213	PCA_V_Winds_4_1
18	U_Winds_dz	116	PCA_V_Winds_7_0	214	PCA_V_Winds_5_1
19	U_Winds_rl_mean_stick	117	PCA_V_Winds_8_0	215	PCA_V_Winds_6_1
20	U_Winds_dz_mean_stick	118	PCA_V_Winds_9_0	216	PCA_V_Winds_7_1
21	U_Winds_mean_lvl	119	PCA_V_Winds_10_0	217	PCA_V_Winds_8_1
22	U_Winds_rl_var_stick	120	PCA_V_Winds_11_0	218	PCA_V_Winds_9_1
23	U_Winds_dz_var_stick	121	PCA_V_Winds_12_0	219	PCA_V_Winds_10_1
24	U_Winds_var_lvl	122	PCA_V_Winds_13_0	220	PCA_V_Winds_11_1
25	V_Winds_rl	123	PCA_V_Winds_14_0	221	PCA_V_Winds_12_1
26	V_Winds_dz	124	PCA_V_Winds_15_0	222	PCA_V_Winds_13_1
27	V_Winds_rl_mean_stick	125	PCA_V_Winds_16_0	223	PCA_V_Winds_14_1
28	V_Winds_dz_mean_stick	126	PCA_V_Winds_17_0	224	PCA_V_Winds_15_1
29	V_Winds_mean_lvl	127	PCA_V_Winds_18_0	225	PCA_V_Winds_16_1
30	V_Winds_rl_var_stick	128	PCA_V_Winds_19_0	226	PCA_V_Winds_17_1
31	V_Winds_dz_var_stick	129	PCA_V_Winds_20_0	227	PCA_V_Winds_18_1
32	V_Winds_var_lvl	130	PCA_V_Winds_21_0	228	PCA_V_Winds_19_1
33	Vertical_Winds_rl	131	PCA_V_Winds_22_0	229	PCA_V_Winds_20_1
34	Vertical_Winds_dz	132	PCA_V_Winds_23_0	230	PCA_V_Winds_21_1
35	Vertical_Winds_rl_mean_stick	133	PCA_V_Winds_24_0	231	PCA_V_Winds_22_1
36	Vertical_Winds_dz_mean_stick	134	PCA_V_Winds_25_0	232	PCA_V_Winds_23_1
37	Vertical_Winds_mean_lvl	135	PCA_V_Winds_26_0	233	PCA_V_Winds_24_1
38	Vertical_Winds_rl_var_stick	136	PCA_V_Winds_27_0	234	PCA_V_Winds_25_1

39	Vertical_Winds_dz_var_stick	137	PCA_V_Winds_28_0	235	PCA_V_Winds_26_1
40	Vertical_Winds_var_lvl	138	PCA_V_Winds_29_0	236	PCA_V_Winds_27_1
41	Geopotentail_Height_rl	139	PCA_V_Winds_30_0	237	PCA_V_Winds_28_1
42	Geopotentail_Height_dz	140	PCA_V_Winds_31_0	238	PCA_V_Winds_29_1
43	Geopotentail_Height_rl_mean_stick	141	PCA_V_Winds_32_0	239	PCA_V_Winds_30_1
44	Geopotentail_Height_dz_mean_stick	142	PCA_V_Winds_33_0	240	PCA_V_Winds_31_1
45	Geopotentail_Height_mean_lvl	143	PCA_V_Winds_34_0	241	PCA_V_Winds_32_1
46	Geopotentail_Height_var_lvl	144	PCA_V_Winds_35_0	242	PCA_V_Winds_33_1
47	sonde_u_on_lvl	145	PCA_V_Winds_36_0	243	PCA_V_Winds_34_1
48	sonde_v_on_lvl	146	PCA_V_Winds_37_0	244	PCA_V_Winds_35_1
49	Forecast_Error__u_on_lvl	147	PCA_V_Winds_38_0	245	PCA_V_Winds_36_1
50	Forecast_Error__v_on_lvl	148	PCA_V_Winds_39_0	246	PCA_V_Winds_37_1
51	rlerr_u_on_lvl	149	PCA_V_Winds_40_0	247	PCA_V_Winds_38_1
52	rlerr_v_on_lvl	150	PCA_V_Winds_41_0	248	PCA_V_Winds_39_1
53	DerivLvl_sonde_u_on_lvl	151	PCA_V_Winds_42_0	249	PCA_V_Winds_40_1
54	DerivLvl_sonde_v_on_lvl	152	PCA_V_Winds_43_0	250	PCA_V_Winds_41_1
55	DerivLvl_Forecast_Error__u_on_lvl	153	PCA_V_Winds_44_0	251	PCA_V_Winds_42_1
56	DerivLvl_Forecast_Error__v_on_lvl	154	PCA_V_Winds_45_0	252	PCA_V_Winds_43_1
57	DerivLvl_rlerr_u_on_lvl	155	PCA_V_Winds_46_0	253	PCA_V_Winds_44_1
58	DerivLvl_rlerr_v_on_lvl	156	PCA_V_Winds_47_0	254	PCA_V_Winds_45_1
59	PCA_U_Winds_0_0	157	PCA_V_Winds_48_0	255	PCA_V_Winds_46_1
60	PCA_U_Winds_1_0	158	PCA_V_Winds_49_0	256	PCA_V_Winds_47_1
61	PCA_U_Winds_2_0	159	PCA_U_Winds_0_1	257	PCA_V_Winds_48_1
62	PCA_U_Winds_3_0	160	PCA_U_Winds_1_1	258	PCA_V_Winds_49_1
63	PCA_U_Winds_4_0	161	PCA_U_Winds_2_1		
64	PCA_U_Winds_5_0	162	PCA_U_Winds_3_1		
65	PCA_U_Winds_6_0	163	PCA_U_Winds_4_1		
66	PCA_U_Winds_7_0	164	PCA_U_Winds_5_1		
67	PCA_U_Winds_8_0	165	PCA_U_Winds_6_1		
68	PCA_U_Winds_9_0	166	PCA_U_Winds_7_1		
69	PCA_U_Winds_10_0	167	PCA_U_Winds_8_1		
70	PCA_U_Winds_11_0	168	PCA_U_Winds_9_1		
71	PCA_U_Winds_12_0	169	PCA_U_Winds_10_1		
72	PCA_U_Winds_13_0	170	PCA_U_Winds_11_1		
73	PCA_U_Winds_14_0	171	PCA_U_Winds_12_1		
74	PCA_U_Winds_15_0	172	PCA_U_Winds_13_1		
75	PCA_U_Winds_16_0	173	PCA_U_Winds_14_1		
76	PCA_U_Winds_17_0	174	PCA_U_Winds_15_1		
77	PCA_U_Winds_18_0	175	PCA_U_Winds_16_1		
78	PCA_U_Winds_19_0	176	PCA_U_Winds_17_1		
79	PCA_U_Winds_20_0	177	PCA_U_Winds_18_1		
80	PCA_U_Winds_21_0	178	PCA_U_Winds_19_1		
81	PCA_U_Winds_22_0	179	PCA_U_Winds_20_1		
82	PCA_U_Winds_23_0	180	PCA_U_Winds_21_1		
83	PCA_U_Winds_24_0	181	PCA_U_Winds_22_1		
84	PCA_U_Winds_25_0	182	PCA_U_Winds_23_1		
85	PCA_U_Winds_26_0	183	PCA_U_Winds_24_1		
86	PCA_U_Winds_27_0	184	PCA_U_Winds_25_1		
87	PCA_U_Winds_28_0	185	PCA_U_Winds_26_1		
88	PCA_U_Winds_29_0	186	PCA_U_Winds_27_1		
89	PCA_U_Winds_30_0	187	PCA_U_Winds_28_1		

90	PCA_U_Winds_31_0	188	PCA_U_Winds_29_1		
91	PCA_U_Winds_32_0	189	PCA_U_Winds_30_1		
92	PCA_U_Winds_33_0	190	PCA_U_Winds_31_1		
93	PCA_U_Winds_34_0	191	PCA_U_Winds_32_1		
94	PCA_U_Winds_35_0	192	PCA_U_Winds_33_1		
95	PCA_U_Winds_36_0	193	PCA_U_Winds_34_1		
96	PCA_U_Winds_37_0	194	PCA_U_Winds_35_1		
97	PCA_U_Winds_38_0	195	PCA_U_Winds_36_1		
98	PCA_U_Winds_39_0	196	PCA_U_Winds_37_1		

Table 9: Features From the Best Scoring Trimming Iteration.
These features are the result of the feature trimming procedure Procedure 2. 'rl' refers data from the remote location. 'dz' refers to data from the drop zone (forecast data only). 'mean stick' is the mean of the values in the column of air. 'mean lvl' is the mean value of a variable from a isobaric pressure level. 'var stick' is the variance of value from a column of air. 'var lvl' is the variance of a variable on a pressure level. 'PCA' refers to the features generated in Section 3.6.2. The number in the PCA feature strings refers to their position in the PCA vector. Features are listed in order of importance as returned by the Tree Boosting algorithm.

1	rlerr_u_on_lvl	96	DT_PCA_ U_4
2	U_Winds_dz	97	DT_V_Winds_dz_mean_stick
3	sonde_u_on_lvl	98	PCA_ V_0
4	DL_dl_ferr_u_on_lvl	99	DL_DT_TMP_P0_L100_GLC0_mean_lvl
5	U_Winds_rl_var_stick	100	DT_V_Winds_rl
6	Temperature_	101	PCA_ U_22
7	V_Winds_rl_mean_stick	102	Vertical_Velocity_rl_var_stick
8	Geopotential_Heightdz_var_stick	103	DL_PCA_ U_38
9	U_Winds_dz_var_stick	104	PCA_ U_16
10	U_Winds_dz_mean_stick	105	U_Winds_rl
11	V_Winds_dz_mean_stick	106	Geopotential_Heightdz
12	Relative Humiditydz_mean_stick	107	DL_PCA_ V_54
13	sonde_v_on_lvl	108	PCA_ U_43
14	PCA_U_4	109	DL_PCA_ U_40
15	U_Winds_var_lvl	110	DL_PCA_ U_13
16	Geopotential_Heightrl_var_stick	111	Vertical_Velocity_dz_var_stick
17	V_Winds_dz_var_stick	112	PCA_ U_13
18	PCA_U_5	113	DL_PCA_13
19	DL_dl_sonde_v_on_lvl	114	DL_Vertical_Velocity_mean_lvl
20	Geopotential_Heightvar_lvl	115	DT_Vertical_Velocity_rl
21	V_Winds_var_lvl	116	Relative Humiditydz
22	Relative Humidityrl_var_stick	117	DL_Vertical_Velocity_var_lvl

23	U_Winds_mean_lvl	118	DT_PCA_V_3
24	Relative Humiditydz_var_stick	119	DL_PCA_U_8
25	DL_sonde_u_on_lvl	120	DT_TMP_P0_L100_GLC0_dz_var_stick
26	DL_Relative Humiditymean_lvl	121	DL_rlerr_u_on_lvl
27	PCA_V_4	122	DL_DT_TMP_P0_L100_GLC0_rl
28	dl_sonde_u_on_lvl	123	DL_PCA_U_30
29	U_Winds_rl_mean_stick	124	PCA_ U_28
30	PCA_U_0	125	DT_Relative Humidityrl_var_stick
31	Relative Humidityvar_lvl	126	TMP_P0_L100_GLC0_dz_mean_stick
32	PCA_ U_1	127	PCA_ U_30
33	DL_V_Winds_mean_lvl	128	DL_PCA_ U_16
34	Relative Humiditymean_lvl	129	DL_rlerr_v_on_lvl
35	Relative Humidityrl_mean_stick	130	DL_PCA_ U_6
36	PCA_V_1	131	PCA_ U_9
37	Geopotential_Heightdz_mean_stick	132	PCA_ V_14
38	PCA_ U_6	133	DL_PCA_U_27
39	PCA_ U_3	134	DT_PCA_V__3
40	DL_Geopotential_Heightvar_lvl	135	DL_PCA_U_32
41	V_Winds_rl_var_stick	136	DL_PCA_U_4
42	DL_PCA_V_1	137	PCA_V_39
43	rlerr_v_on_lvl	138	DL_DT_U_Winds_dz
44	PCA_ U_8	139	DT_PCA_U_14
45	PCA_ U_11	140	DL_V_Winds_var_lvl
46	PCA_9	141	DL_PCA_11
47	Geopotential_Heightrl	142	DT_V_Winds_dz_var_stick
48	DL_Geopotential_Heightdz	143	DL_PCA_ U_33
49	PCA_11	144	DT_PCA_ U_17
50	DL_dl_ferr_v_on_lvl	145	PCA_ V_21
51	PCA_6	146	DL_PCA_22
52	ferr_v_on_lvl	147	DL_PCA_ U_24
53	ferr_u_on_lvl	148	PCA_ V_31
54	DL_ferr_u_on_lvl	149	PCA_ V_5
55	DT_U_Winds_dz	150	PCA_ V_47
56	dl_sonde_v_on_lvl	151	DT_TMP_P0_L100_GLC0_rl_var_stick
57	DL_Geopotential_Heightrl	152	Relative Humidity_rl
58	PCA_2	153	DL_PCA_33
59	PCA_3	154	DL_DT_Temperature_
60	DL_TMP_P0_L100_GLC0_rl	155	PCA_ V_18
61	Geopotential_Heightrl_mean_stick	156	PCA_ V_10
62	DL_U_Winds_dz	157	DL_PCA_ V_27
63	DL_PCA_5	158	DL_DT_ V_PCA_2

64	DT_PCA_ U _4	159	DL_PCA_ V_39
65	TMP_P0_L100_GLC0_dz_var_stick	160	DL_PCA_ U_1
66	dl_ferr_u_on_lvl	161	DT_PCA_ V_11
67	PCA_ U _7	162	DL_PCA_4
68	PCA_ U _2	163	PCA_ V_41
69	DT_Vertical_Velocity_mean_lvl	164	PCA_ U_39
70	DT_PCA_ U _6	165	DL_DT_PCA_9
71	PCA_ U _25	166	PCA_ U_44
72	Vertical_Velocity_var_lvl	167	PCA_ U_28
73	PCA_ U _17	168	DL_PCA_ U_52
74	DL_TMP_P0_L100_GLC0_mean_lvl	169	Vertical_Velocity_rl
75	DL_dl_rlerr_u_on_lvl	170	PCA_ U_45
76	DL_PCA_ U _5	171	DL_PCA_ V_9
77	DL_dl_sonde_u_on_lvl	172	DL_PCA_ V_30
78	DL_PCA_47	173	DL_PCA_ U_36
79	DT_Relative Humidityvar_lvl	174	PCA_ V_38
80	DT_Geopotential_Heightrl_mean_stick	175	DL_DT_Relative Humiditymean_lvl
81	DL_PCA_9	176	DL_PCA_ V_12
82	PCA_ U _29	177	DT_PCA_ U_5
83	PCA__ U 10	178	PCA_4 V_7
84	DT_Relative Humidityrl_mean_stick	179	DT_PCA_ U_18
85	DL_PCA_ U _16	180	DL_PCA_ V_26
86	DL_ferr_v_on_lvl	181	DL_PCA_ U_56
87	PCA_ V_18	182	PCA_ V_34
88	PCA_ V_33	183	PCA_U_22
89	Vertical_Velocity_dz_mean_stick	184	DL_PCA_ U_48
90	DL_PCA_ V_2	185	PCA_ U_21
91	DL_PCA_ V_24	186	PCA_ U_23
92	DL_Relative Humidityvar_lvl	187	PCA_ U_17
93	PCA_ V_8		
94	DL_Temperature_		
95	DT_U_Winds_dz_var_stick		