Predicting Flight Delays with Probabilistic Model for Know Delay

Team D: Jin Haeng Lee[1], Ashwin Malgaonkar[1], Manas Chatterjee[1]
Georgia Institute of Technology, Atlanta, GA

Abstract
Predicting flight delays from weather conditions of related itinerary has been researched and feasible. To access and analyze the probabilities of flight delays, we discuss three multiclass-multioutput probabilistic models (Random Forest, XGBoosting, K-Nearest Neighbors). Probabilistic modeling approach and its approach is yet to be solidified and this project attempts to tackle data analytics, feature selection, parameter tuning and performance comparison to understand better about the feasibility of existing models with METAR and ASPM data.

Our code and supplementary documentation can be downloaded at Google Drive

1. Introduction and Background

Nearly every airline passenger has experienced the uncertainty and stress associated with flight delays. Delayed flights can cause a passenger to miss business meetings, connecting flights, appointments, or important events. Almost half of all flight delays and cancellations are caused by adverse weather conditions. While weather may only minimally impact other modes of transportation, it is the single most prevalent cause of flight delays. Researchers have been tackling to predict flight delays via machine learning [1], including binary classification [2] with a certain threshold on delayed time in minutes to obtain whether flight will be delayed or not as an output. Regression also is a common approach to pin down exact minutes of predicted delays [3].

2. Problem Formulation

This paper discusses the feasibility of multiclass-multioutput probabilistic flight delay predictions on a given METAR and ASPM data for a given airport. The contribution of this report is that it is a multi-output modeling, indicating we are not only classifying which interval (class) the prediction might belong to, but also provides the probabilities for each interval with a given predicted value.
We demonstrate our methodology via a METAR-ASPM-merged dataset from 35 major airports in the United States, averaged at a scale of every hour. Each airport needs to be treated uniquely when building a model, hence there exists a model for each airport.

In summary, the objective of this report aims to Integrate hourly METAR-ASPM data with the planned itinerary to evaluate whether the itinerary may be subject to delay and also provide the likelihood of the delay.

[1]jlee3693@gatech.edu, mchatterjee30@gatech.edu, amalgaonkar3@gatech.edu

3. Approach and Implementation

Figure 1. Workflow of Modeling Building

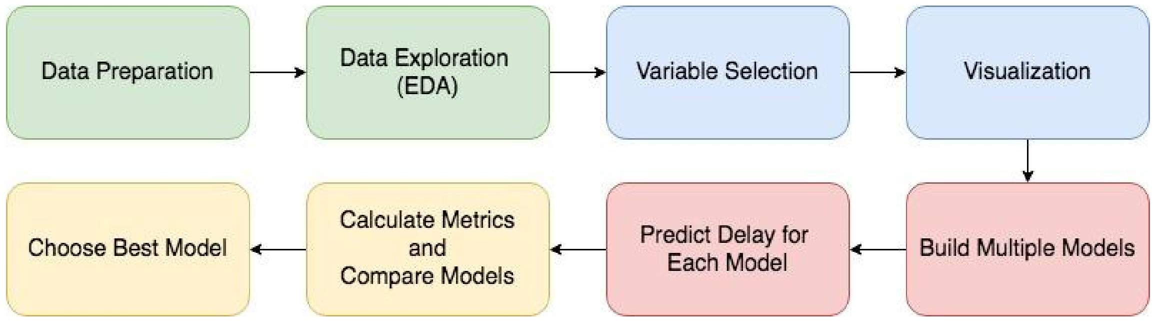


Figure 1 depicts the workflow of our methodology. Raw input is a merged dataset between METAR and ASPM for a specific airport. Data preparation and data exploration is performed to clean up and reformat given data into desired input. Variable selection or feature selection is tied together with model building as they are subjected to recursive procedures. Once a model is built, metrics such as Mean Absolute Percentage Error (MAPE) or accuracy is calculated for model comparison and to observe how well it performed. Based on metrics, the best model will be chosen and the probability for each class is calculated. In the upcoming section, each procedure will be discussed in more detail.

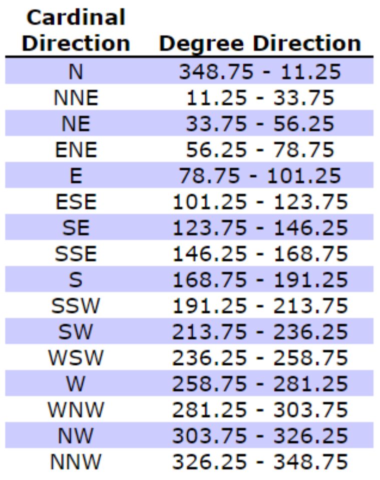
3.1 Data Preparation and Exploration:

i. ***Null Values:*** During our exploratory data analysis and preliminary review, we identified a handful of null values depending on the attribute. The null entries are pre-processed by either removing or replacing them and such decision relies on the characteristic of the attribute.

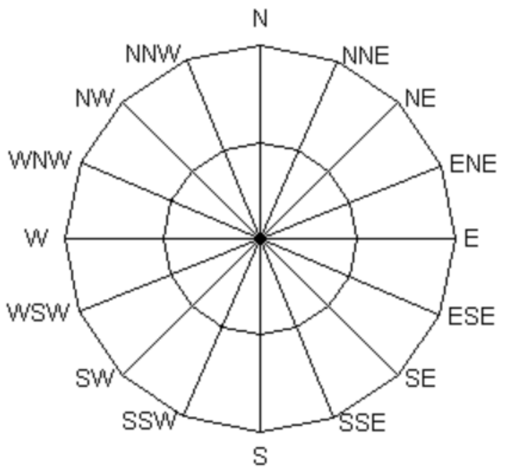
   For example, null values for Wind Gust and Peak Gust Speed are replaced with zero as an indication of no excessive wind speed but steady state. Moreover, for few rows that were recorded after 24 hour period, all attributes commonly share null entries hence are subject for deletion. For Layer 1, which indicates the cloud altitude, null values equate to no visible cloud up to a certain level in feet. Therefore, we replaced them with the maximum value of Layer 1. Null entries for categorical features are replaced with another categorical entry for further analyses.

ii. ***Categorization of continuous variable:*** Upon bivariate analysis of each predictor with response variable, a more pragmatic way for analysis was discovered to be bucketing continuous attributes into categories. Figure 2 demonstrates how Wind Direction is bucketed.

Figure 2. Wind and Peak Gust Direction Buckets.



Cardinal Direction	Degree Direction
N	348.75 - 11.25
NNE	11.25 - 33.75
NE	33.75 - 56.25
ENE	56.25 - 78.75
E	78.75 - 101.25
ESE	101.25 - 123.75
SE	123.75 - 146.25
SSE	146.25 - 168.75
S	168.75 - 191.25
SSW	191.25 - 213.75
SW	213.75 - 236.25
WSW	236.25 - 258.75
W	258.75 - 281.25
WNW	281.25 - 303.75
NW	303.75 - 326.25
NNW	326.25 - 348.75



Categorization was processed on our response variable as well. Our initial approach of performing regression with continuous response variables did not yield ample accuracy. Therefore, the response variable is bucketed into intervals of 10 minutes as shown in table 1, for better performance.

Table 1. Example of continuous response variable into multi-class bucket.

Avg Gate Arrival Delay (min)	Bucketed Result
20.29	2
6.33	0
13.35	1

iii. ***Class Imbalance:*** Class imbalance can cause severe challenges when building predictive models. It refers to when the class distribution is skewed and can be considered as severe when the ratio between minority to majority class is larger than 1:100 [4]. We discovered that the number of occurrences of delay greater than 30 minutes is a much smaller proportion as compared to that of overall observations for any airport. Such phenomenon is exemplified in Atlanta airport where only 3% of observations have delays greater than 30 minutes. In order to avoid overfitting of the model, we implemented a simple but useful technique known as random oversampling, which allows the generation of new samples for minority class by replacement. Figure 3 depicts class imbalance for JFK airport whereas figure 4 shows the efficacy of oversampling.

Figure 3. Demonstration of Class Imbalance in JFK airport for 'Average Delayed Time in Minutes'

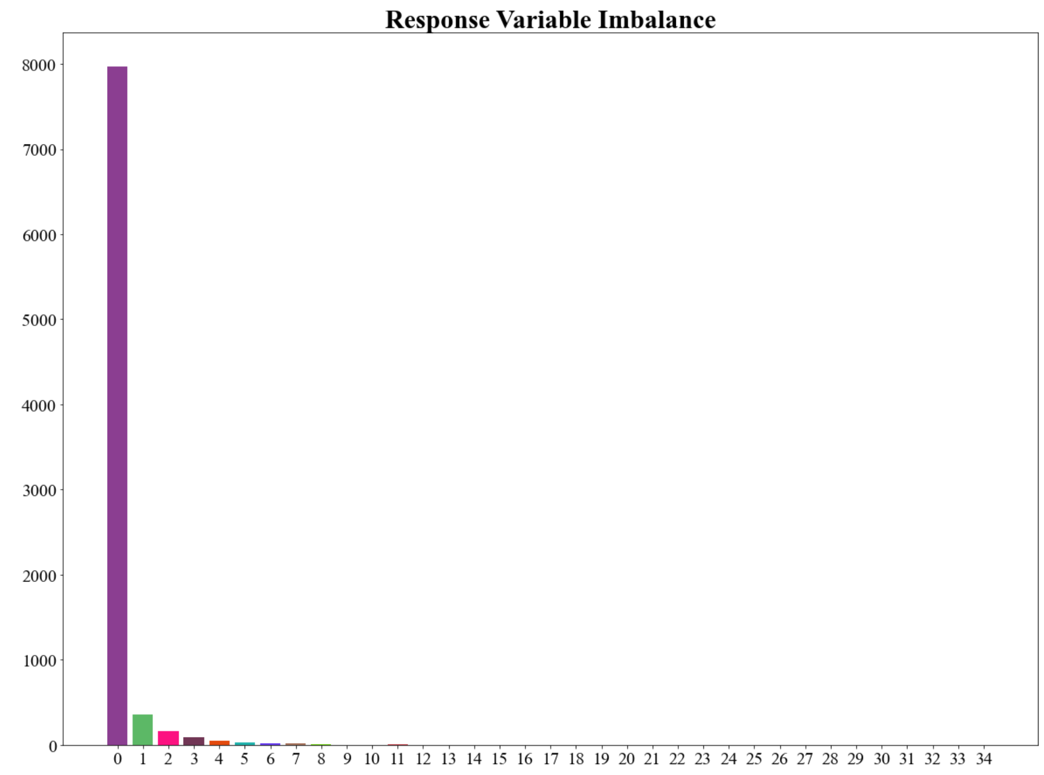
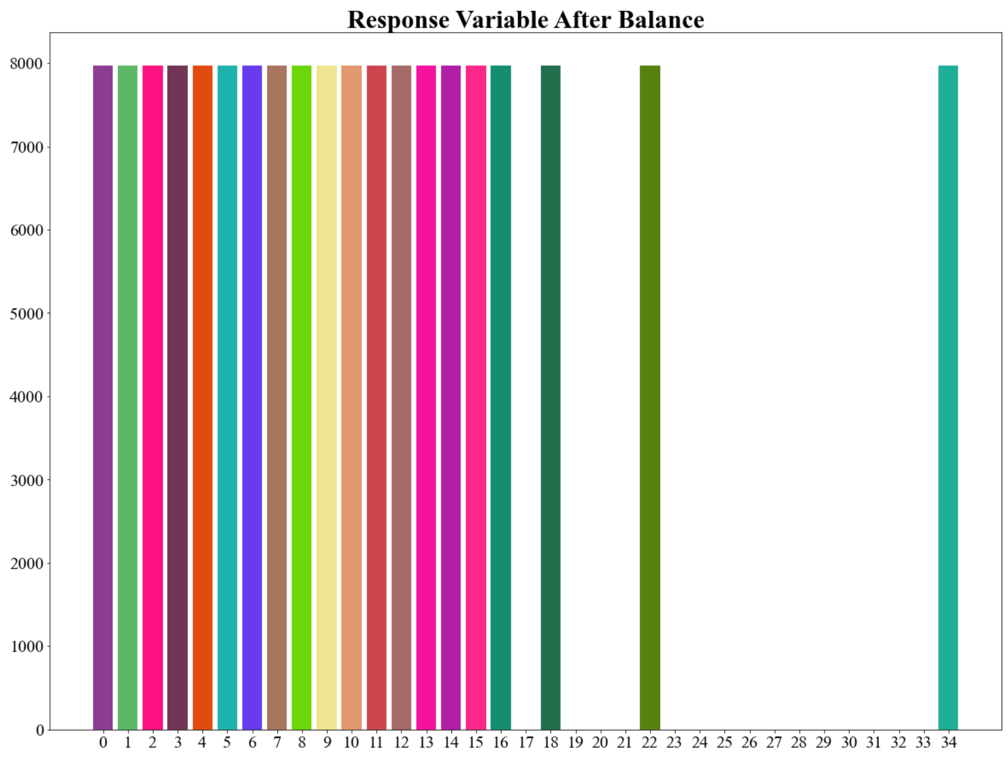


Figure 4. JFK airport after Random Oversampling.



iv. ***Feature Encoding:*** Feature encoding is a necessary step for model building as machine learning algorithms can only process numerical entries. Our initial approach takes predictor WX Codes, which explains the weather condition in codes, and applies integer encoding, where each code is converted into numerical class based on the number of unique code combinations.

Additionally, one-hot encoding is executed for the predictor, Sky 1. This method creates separate columns for each categorical option and marks 1 for entry that contains the categorical value, 0 otherwise. Table 1 illustrates different techniques mentioned above.

Table 1. Illustration of Different Encoding Techniques

Before Encoding		**After Encoding**				
WX Codes	**Sky 1**	**WX Codes**	**BKN**	**CLR**	**OVC**	**VV**
RA BR	BKN	0	1	0	0	0
FG	CLR	1	0	1	0	0
BR	OVC	2	0	0	1	0
HZ	VV	3	0	0	0	1

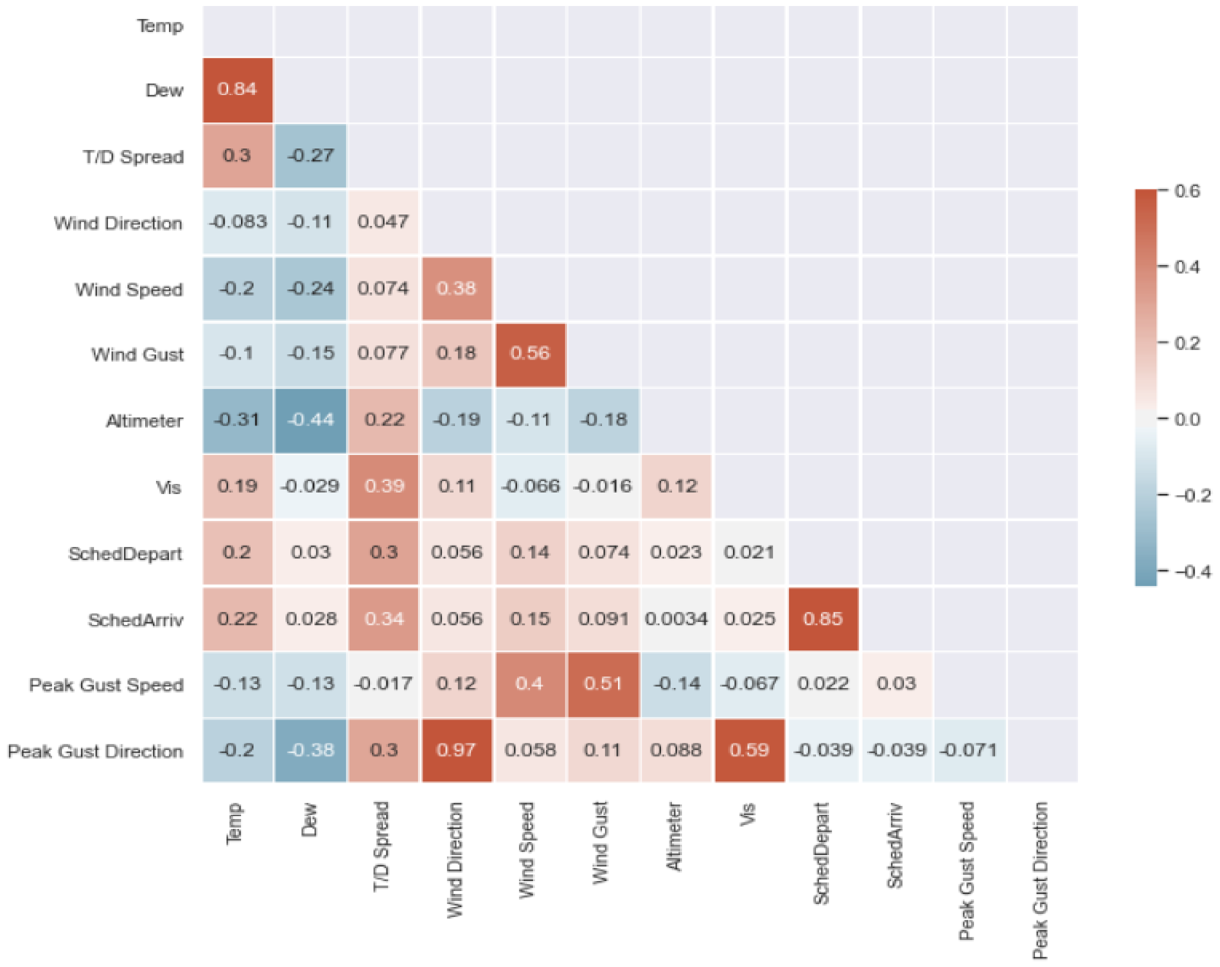
For the purpose of integration and possibilities of having input data with categories that are not part of training procedures, more effective methodology was sought. As a result, binary encoding was applied, where it is a combination of hash encoding and one-hot encoding. The categorical features are first converted to numerical via ordinal encoder, followed by conversion to binary.

3.2 Feature Selection:

Feature selection is considered to be one of the most important steps in model building to reduce overfitting, improve accuracy, and lessen computational expense. Among diverse feature selection techniques, we have implemented the following: 1. Pearson correlation matrix, 2. Elastic Net Regression, 3. Lasso Regression, 4. Ridge Regression.

- ***Pearson Correlation Matrix****:* Pearson correlation matrix is a multivariate methodology, which calculates how closely each predictors are correlated to each other. In other words, the purpose of performing this analysis is to reduce the multicollinearity that could be detrimental. Figure 5 shows a sample heat map generated for Atlanta airport. In this project, we have set a threshold of 0.8 and selectively remove highly correlated predictors.

Figure 5. Correlation Heatmap for Atlanta Airport



- ***Lasso and Ridge Regression****:* In this section, we discuss Lasso and Ridge together as they are mathematically similar in how they operate. In Ridge regression, predictors are scaled to z-scores, ensuring that the penalty term penalizes different coefficients. Lasso is very similar to Ridge but it penalizes coefficients as L1 penalty, whereas Ridge is scaled with L2 penalty. Lasso selects features by zeroing coefficients for less important predictors, while Ridge shrinks the coefficients nearly to zero but never zero.

  To execute and control shrinking or zeroing degree (regularization penalty) in Python's Scikit, we first need to define appropriate alpha for optimized results. This can be accomplished by grid searching with cross-validation for various alpha values, calculating mean test score for each alpha to finalize which performs the highest score. For both methodology, we selected alphas to be 0.00001, 0.001, 0.01, 1 and 10 to have sufficient coverage. Figure 6 illustrates how selected alphas performed and figure 7 & 8 depict variable importance for selected features after Lasso and Ridge have been performed for JFK airport.

Figure 6. Selected alphas and corresponding Mean Test Score

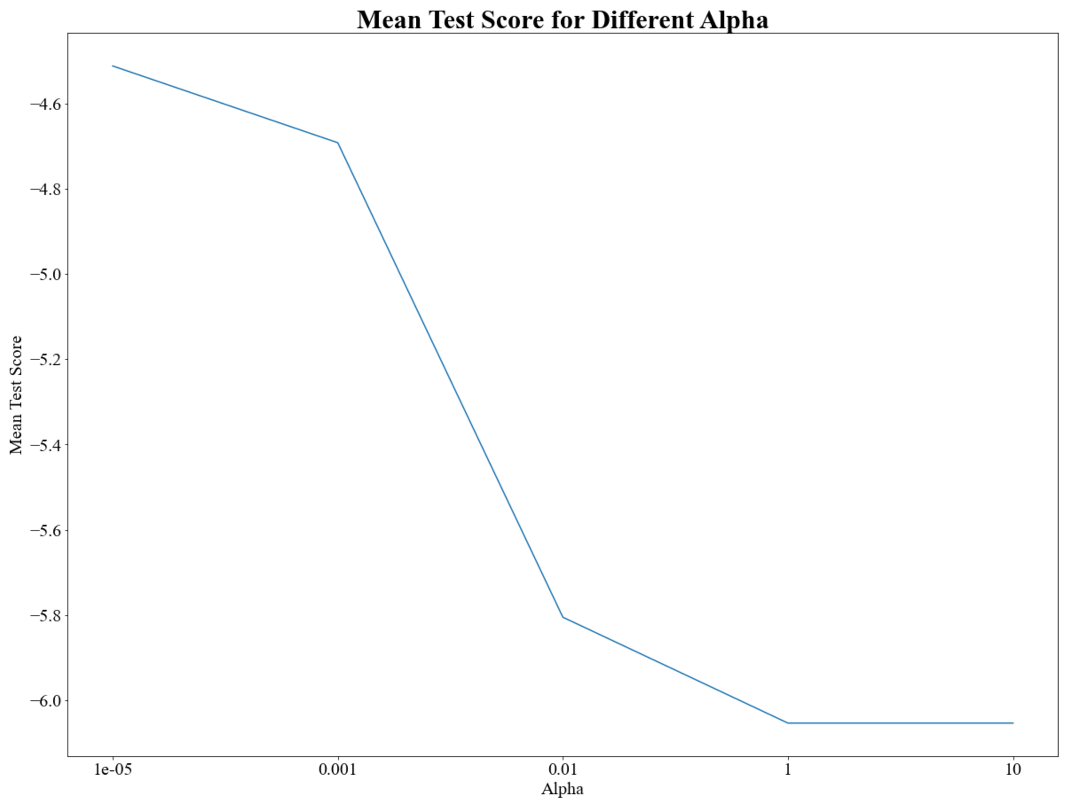


Figure 7. Lasso Feature Selection Result

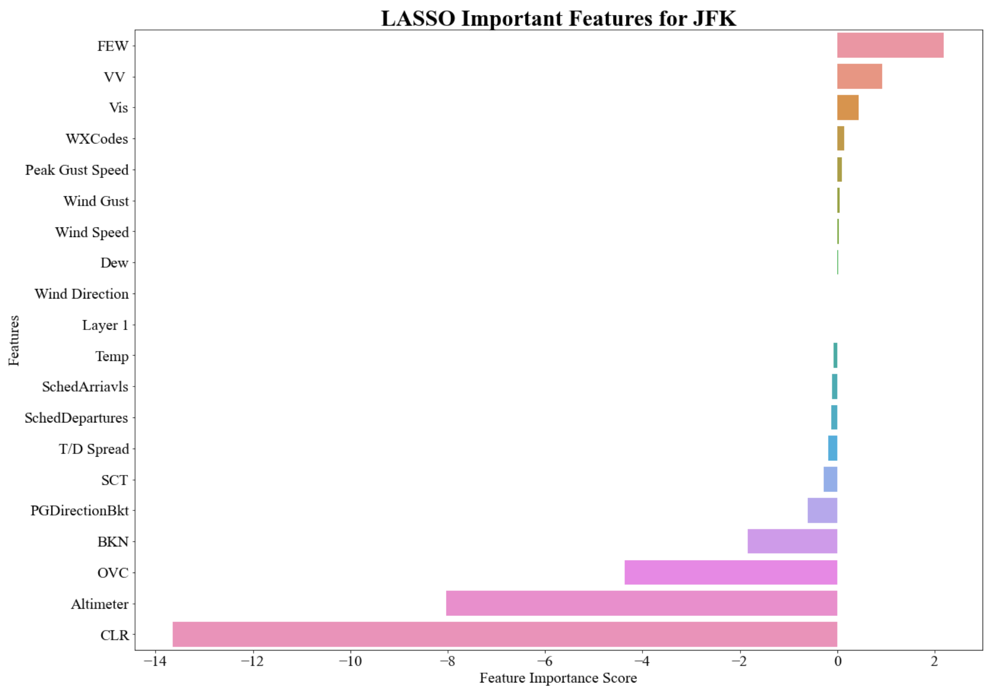
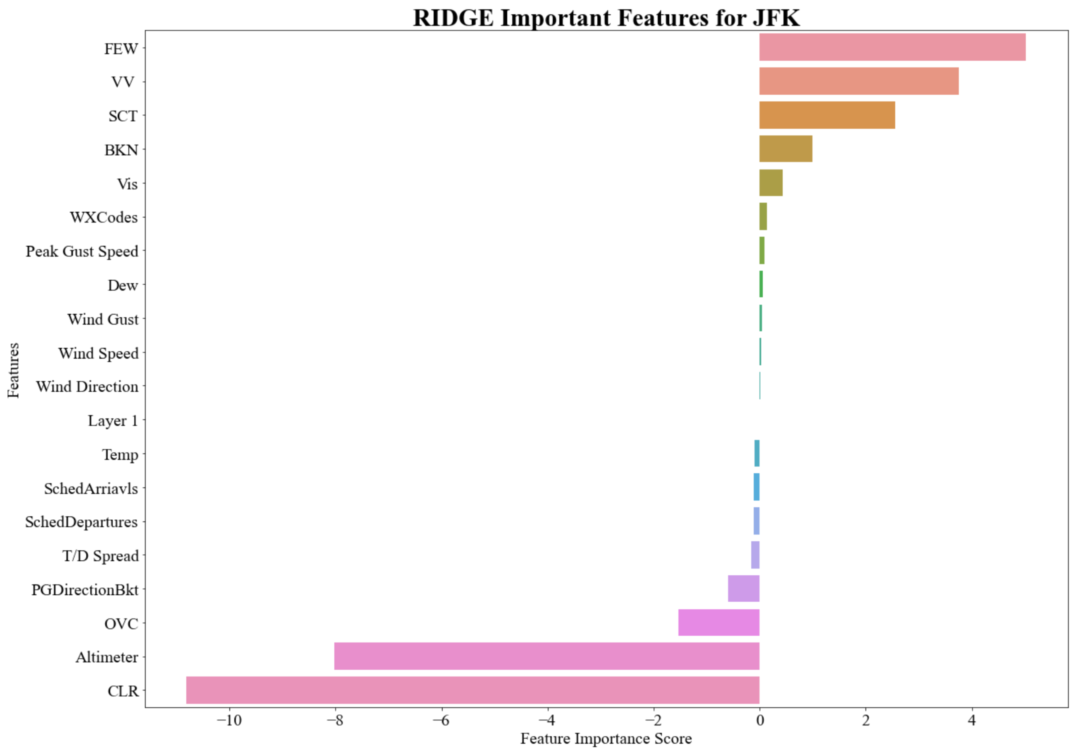


Figure 8. Ridge Feature Selection Result

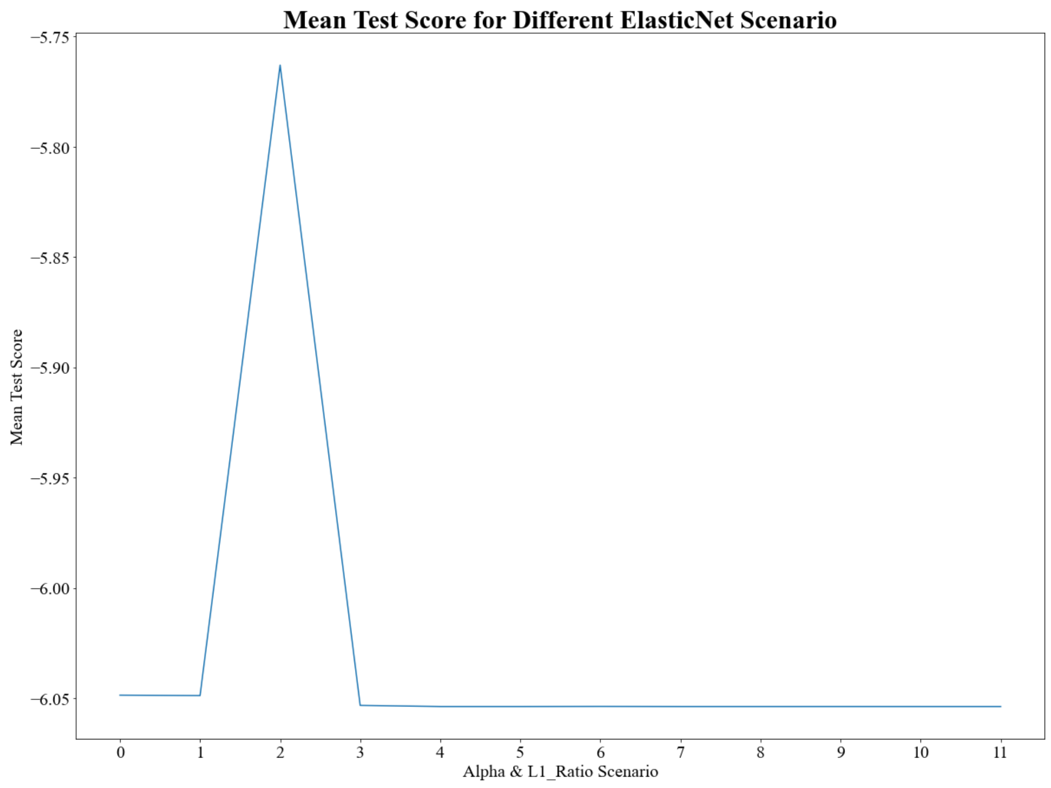


- ***Elastic Net:*** Elastic Net is another regularization process, where it resembles a combination of L1 (Lasso) and L2 (Ridge) approaches, attempting to get the best of both. Unlike Lasso or Ridge, Elastic Net has two main parameters, lambda and alpha. Grid search with cross-validation is performed in search for the best lambda and alpha combination that would result in the highest score. Table 2 showcases different parameter combinations tested and figure 9 illustrates the test score from table 2's scenario (see below)

Table 2. Combinations of parameters tested for Elastic Net (Best parameters are highlighted in red).

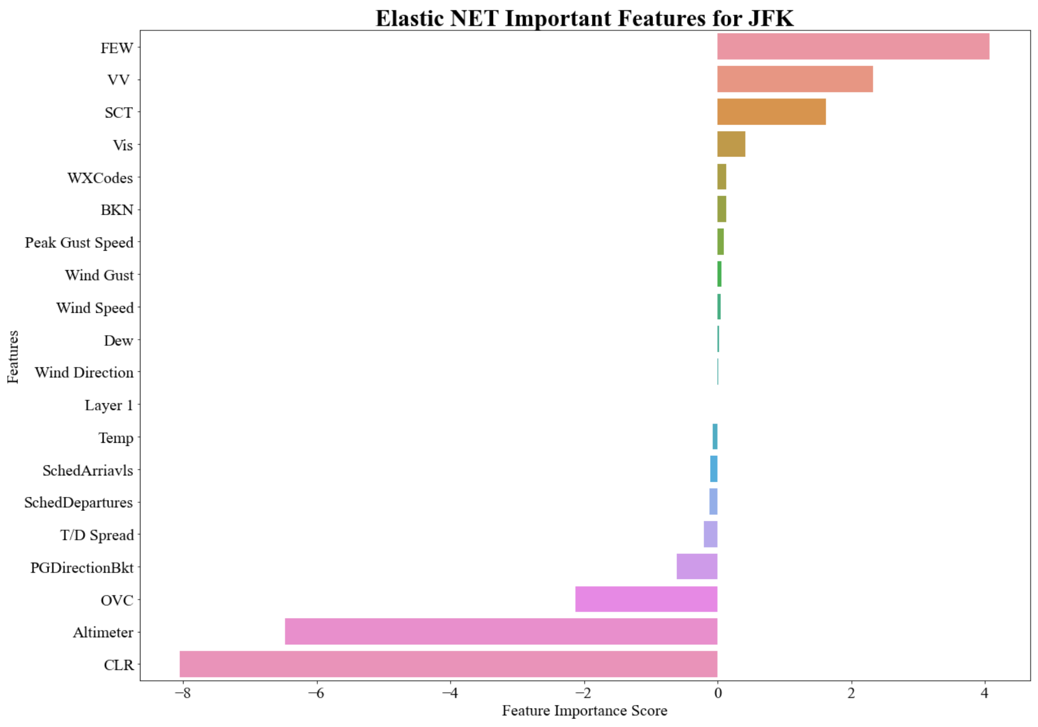
Scenario	Lambda	Alpha (L1 ratio)
0	0.01	0
1	0.01	0.5
2	**0.01**	**1.0**
3	0.1	0
4	0.1	0.5
5	0.1	1.0
6	1.0	0
7	1.0	0.5
8	1.0	1.0
9	10	0
10	10	0.5
11	10	1.0

Figure 9. Mean Test Score results from a combination of parameters listed in Table 2.



With chosen parameters Elastic Net feature selection is performed to generate finalized features as shown in figure 10 below.

Figure 10. Features selected from Elastic Net Regression.



As previously mentioned, each airport will be treated uniquely and this logic applies to feature selection as well. This project is designed for each airport to choose the best performing option from three feature selection methods above. The decision is made once a model is built for each feature selection method and metric is calculated and compared.

3.3 Model Building and Predicting Delay:

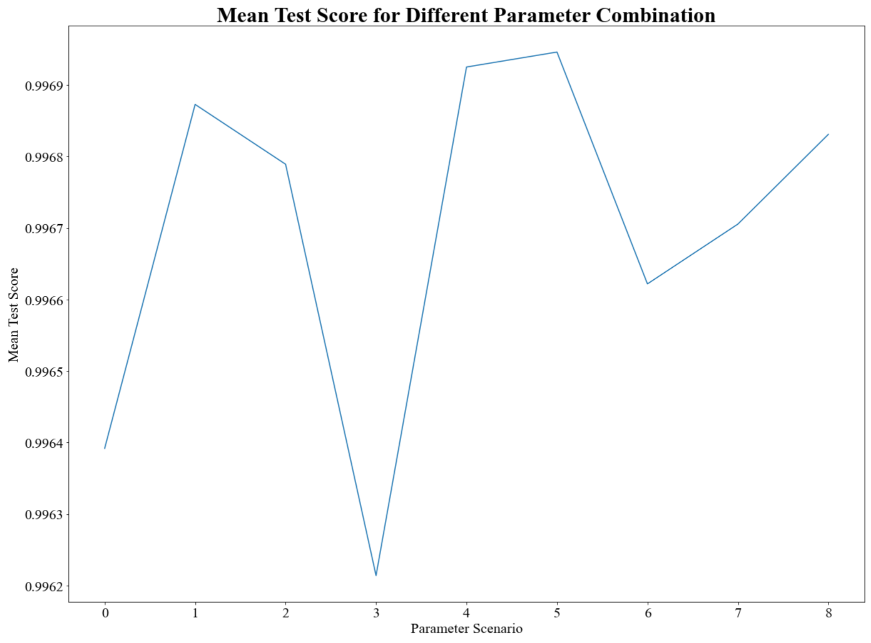
As previously mentioned, for higher model accuracy and performance, the response variable is bucketed into intervals of 10 minutes, which makes modeling a multi-class problem. In this section, three multi-class modeling approaches (Random Forest, XGBoosting, K-Nearest Neighbor) are presented and discussed.

- ***Random Forest Classification:*** Random Forest is first properly introduced by Leo Breiman [5], which is an ensemble technique of multitudes of decision trees. It excels to fulfill multiclass-multi output objective, by pooling all decision trees and taking the mode of predicted classes. However, executing Random forest to full potential requires fine tuning of hyper-parameters from Python's Scikit's default values. Grid search with cross-validation is utilized in search for best parameters while fixating few. Table 3 shows various combinations of parameters tested and figure 11 demonstrates the performance for each combination. As a result, parameters with best mean test score are selected to proceed into training the model.

Table 3. Combinations of parameters tested for Random Forest (Best parameters are highlighted in red).

Scenario	Bootstrap	Max Depth	Max Features	Min Samples Leaf	Min Samples Split	N Estimators
0	True	50	5	3	8	50
1	True	50	5	3	8	100
2	True	50	5	3	8	200
3	True	70	5	3	8	50
4	True	70	5	3	8	100
5	**True**	**70**	**5**	**3**	**8**	**200**
6	True	100	5	3	8	50
7	True	100	5	3	8	100
8	True	100	5	3	8	200

Figure 11. Mean Test Score Result for various parameter combinations for JFK airport.



- ***XGBoosting:*** XGBoosting is a relatively new machine learning algorithm, which is based on an ensemble of decision trees with a gradient boosting framework. First created by Tianqi Chen [6], it is designed for computational speed and performance. To evaluate XGBoosting performance, objective function needs to be defined.

  Numerous objective options exist in XGBoosting integration with Python such as logistic regression, binary classification, Poisson regression, etc. For the purpose of this project, multiclass classification using Softmax objective is chosen.

- ***K-Nearest Neighbor:*** The infamous KNN algorithm can handle both classification and regression. It is simple and efficient when dealing with supervised learning. Principle behind this algorithm is to find the closest distance for a new data entry to the training dataset and predict labels based on classes of neighboring data points.

  When it comes to KNN classification one of the most vital parameters to set is the user-defined number of neighbors points for prediction, hence k-nearest neighbors. In this project, as we bucketed the response variable into intervals of 10 minutes, it makes sense to match the number of neighbors. Therefore, depending on which airport k would vary.

3.3 Metric Calculation and Model Comparison:

Metric calculation is an essential part of evaluating model performance and choosing appropriate metric is deemed to be crucial when finalizing model selection. Various metrics are available in machine learning applications and a single metric alone cannot provide concrete evaluation. We present # of metrics specifically used for classification purposes that are: weighted accuracy, f1-score, precision, recall and mean absolute percentage error.

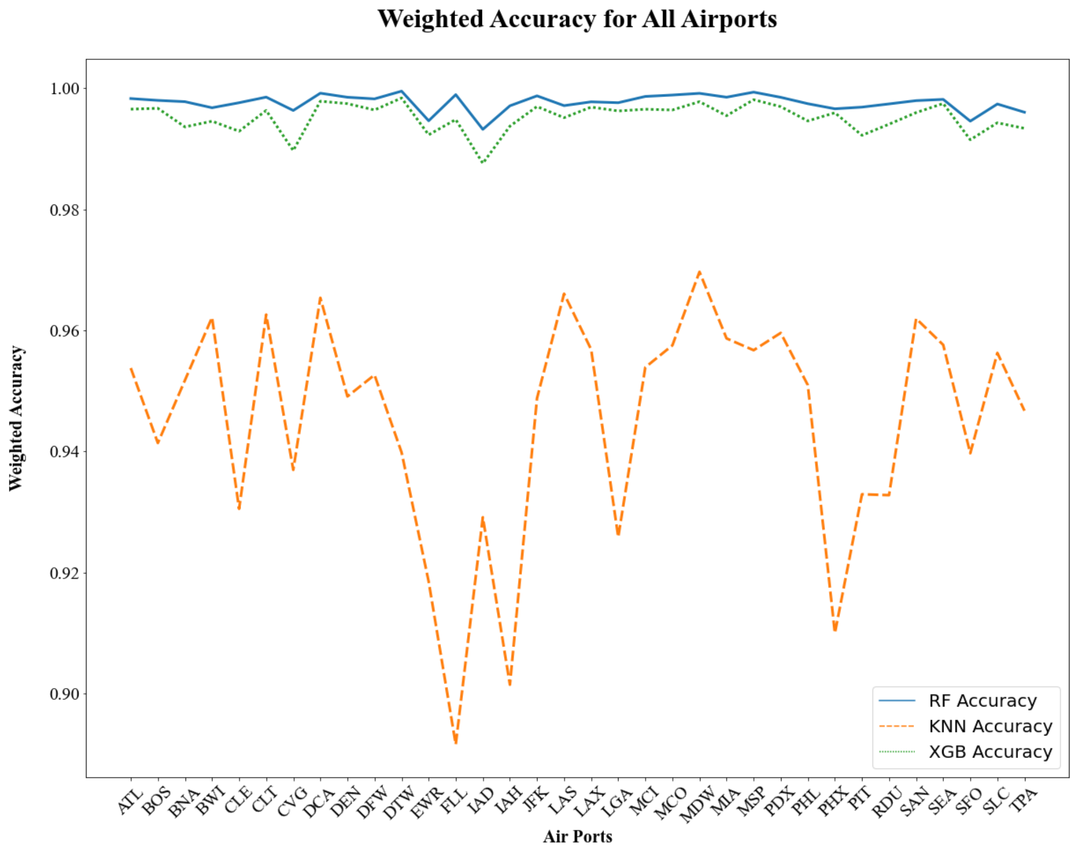
1. ***Weighted Accuracy:*** Weighted accuracy is the number of correctly predicted instances in a specific class, divided by the total number of instances in that class [7]. Due to the nature of the response variable's distribution being heavily skewed and biased, weighted accuracy is chosen over regular accuracy.
2. ***Recall:*** Recall metric is defined by the number of true positives divided by total actual positives. This metric is considered to be useful when high cost is associated with the false negatives [8]. In this project, falsely predicting a flight as non-delay is costly for both airlines and passengers, hence the recall score is suitable.
3. ***Precision:*** Precision is defined by the number of true positives divided by total predicted positives [7]. Similarly to recall, precision is useful when high cost is associated with the false positives [8]. In this project, false positives would be predicting that a flight will be delayed when it is actually not delayed. Cost of this would also be high similarly to that of false negatives. Therefore, precision is deemed to be suitable.
4. ***F1-score:*** F1-score is a function of both precision and recall. Thus, f1-score is another efficient tool to seek balance between precision and recall. Also, f1-score is considered to be especially efficient under presence of uneven class distribution [8], which suits well for given type of data in this project.

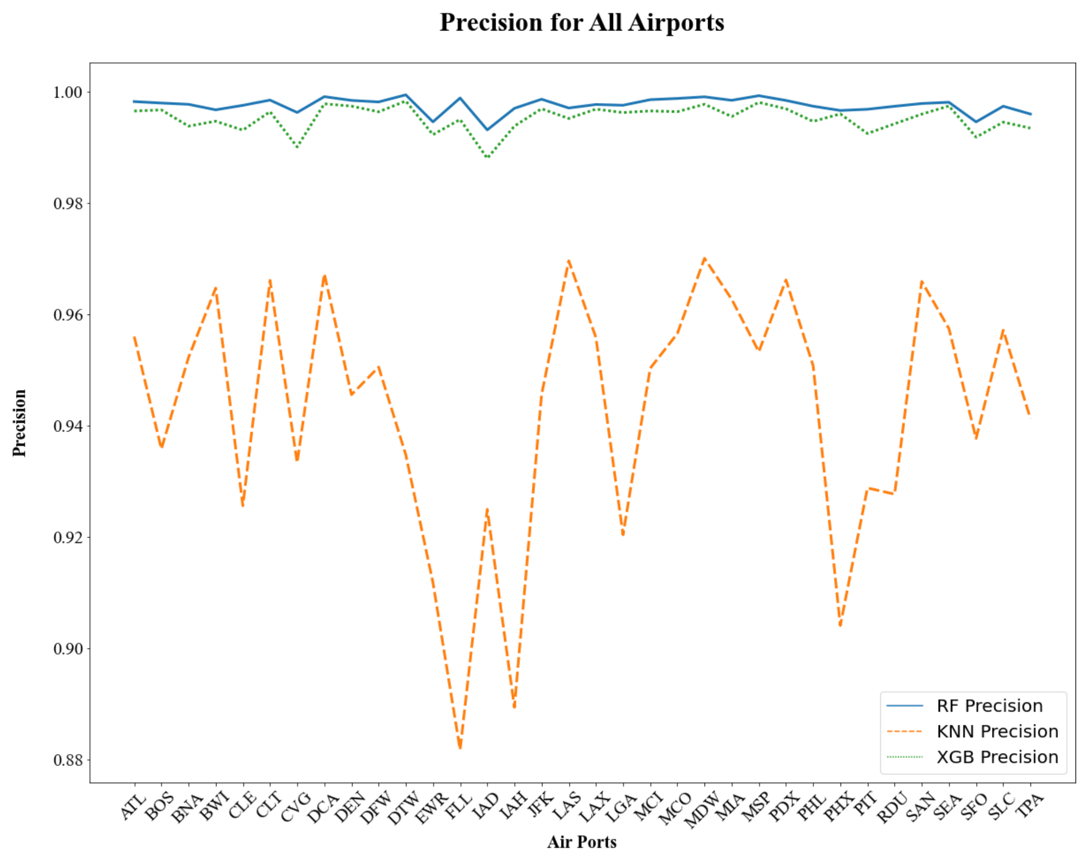
Consequently, the best model can be decided based on metric calculations and overall results. Main deciding factor will be weighted accuracy as we believe other metric measures are supplementary.

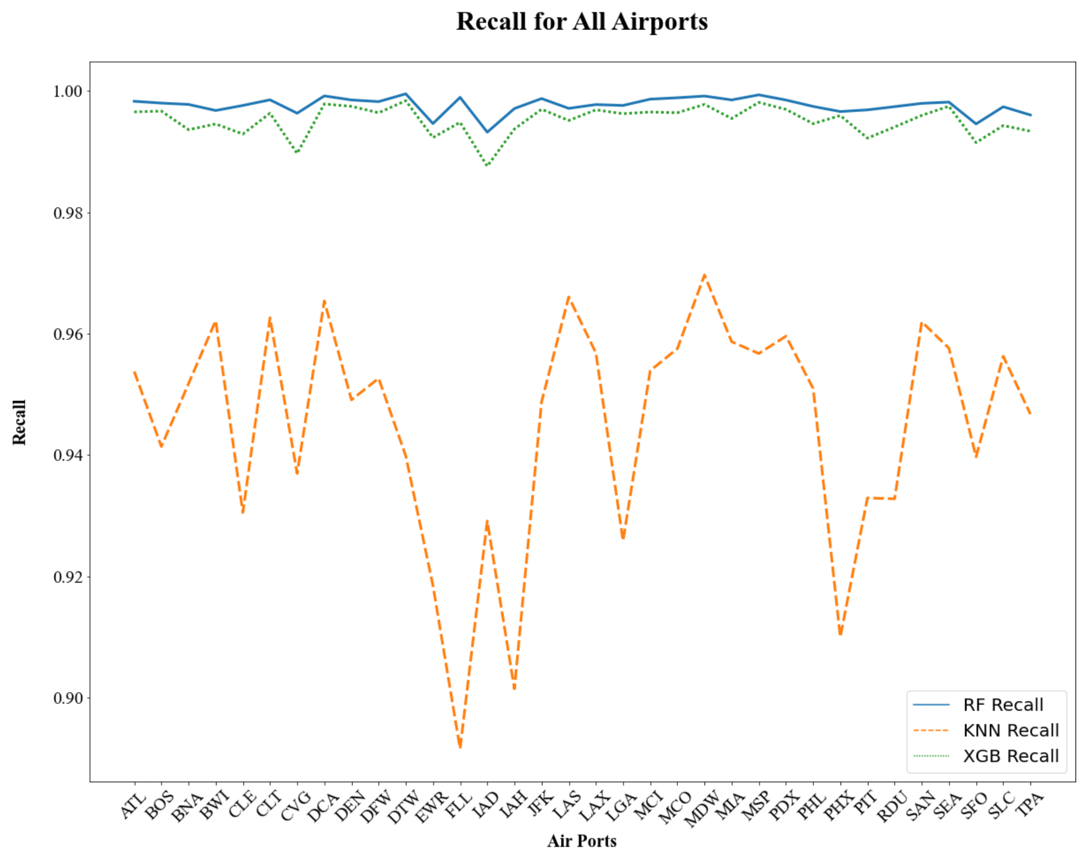
5. Model Performance Evaluation

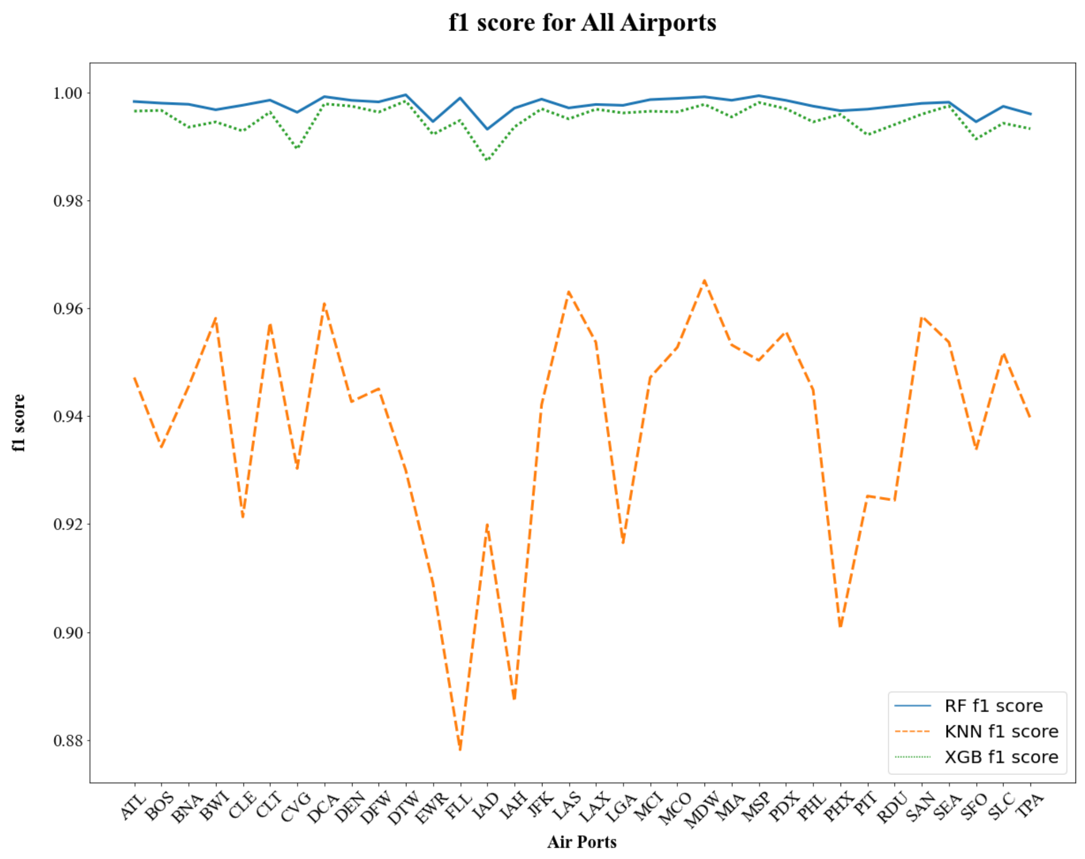
After raw data for each airport goes through pre-processing, cleaning and random oversampling, populated data entries are split into training and testing subsets with 60% and 40% ratio. Various ratios have been tested with 35%, 30%, 25% for testing subset, yet we recognized the difference is negligible and hence decided 40% is a reasonable value for modeling to minimize training bias, yet enough to train the model effectively. Figure 12 demonstrates how each model performed based on metrics mentioned above for all 34 major airports in the United States with 10 folds of cross-validation.

Figure 12. Model performance for 34 major airports in the United States.




Precision for All Airports
Precision
1.00
0.98
0.96
0.94
0.92
0.90
0.88
RF Precision
KNN Precision
XGB Precision
ATL BOS BNA BWI CLE CLT CVG DCA DEN DFW DTW EWR FLL IAD IAH JFK LAS LAX LGA MCI MCO MDW MIA MSP PDX PHL PHX PIT RDU SAN SEA SFO SLC TPA
Air Ports


Recall for All Airports
Recall
1.00
0.98
0.96
0.94
0.92
0.90
RF Recall
KNN Recall
XGB Recall
ATL BOS BNA BWI CLE CLT CVG DCA DEN DFW DTW EWR FLL IAD IAH JFK LAS LAX LGA MCI MCO MDW MIA MSP PDX PHL PHX PIT RDU SAN SEA SFO SLC TPA
Air Ports



As can be seen from above, the plots show almost the same trend across all metrics. For all metrics, Random Forest resulted to be superior over K-Nearest Neighbors yet fairly close to XGBoosting. From such matching trends, we observed that models concurrently performed slightly worse or better for certain airports. Moreover, we can safely conclude that any of the metrics can be a measure to choose the best model among three.

In addition to 3 modeling approaches, we have initially tested Adaboosting and Logistic Regression, which neither of them were on par with the above three. As a result, for delivering and integrating our output purposes, only Random Forest will be provided if in need for retraining.

6. Integration with application and Deliverables

Our team provides four different types of deliverables. Please refer to our User Guide for detailed instructions.

1. ***Codes:*** Two different functional codes will be provided, one for retraining and saving encoders, feature selection method and Random Forest model, another for loading encoders, feature selecting method and Random Forest model to test and predict.
2. ***Tables:*** There are two tables generated as final outputs. 'KnowDelay - All Models' table consists of metrics for all three models with 10 folds of cross-validation. It also contains the likelihood of a flight being delayed or not delayed for each airport. 'KnowDelay RandomForest Results' table consists of metrics for Random Forest with 5 fold cross-validation. It is partitioned into two sub-tables; one with all entries, another with entries that are delayed only.
3. ***KnowDelay App:*** As our historical training demonstrated, Random Forest proved to be superior among all models. Hence, Random Forest with each feature selection method was trained and exported. Moreover, encoders are exported for each airport to utilize consistent features that were used for training. All three feature selection methods are also saved to extract selected features for each airport. Utilizing exported files above, API based application was prepared into a package with required Python library packages to be called.
4. ***Json Files:*** Finally, json files are saved for each airport and also a combined version for all airports, that consists of the best feature selection method, features, accuracy, and likelihood, in order to integrate and convert into API application.

Figure 13. API Integration workflow.

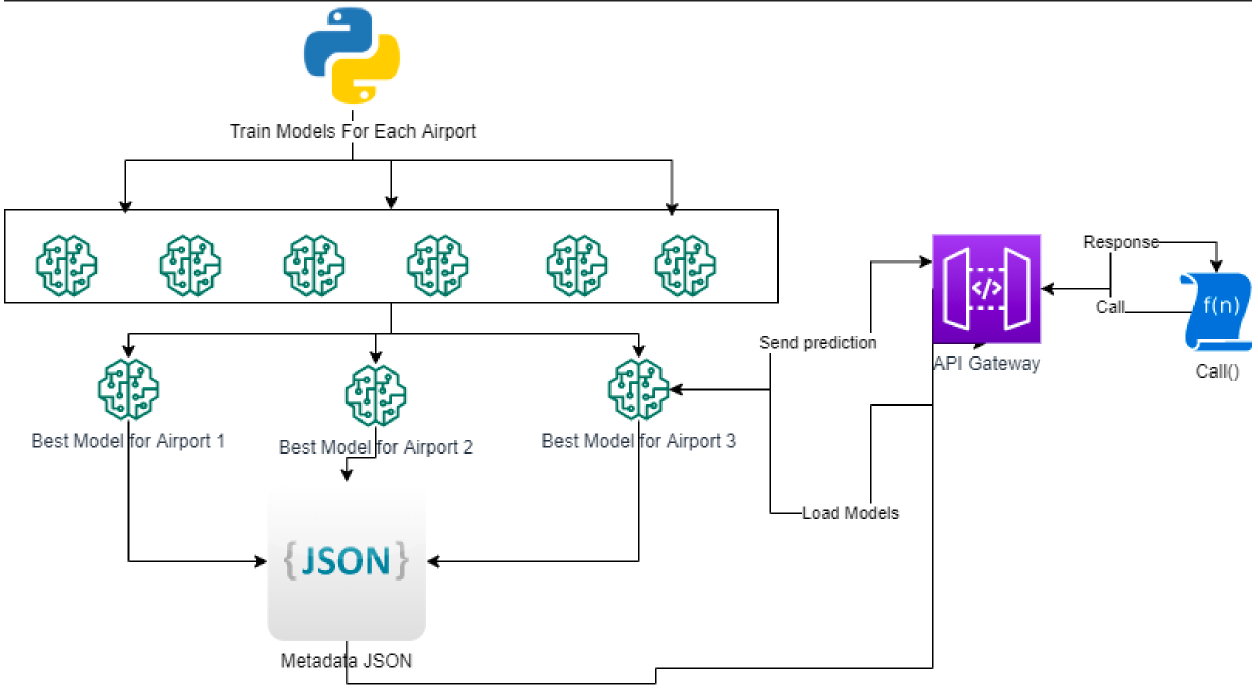
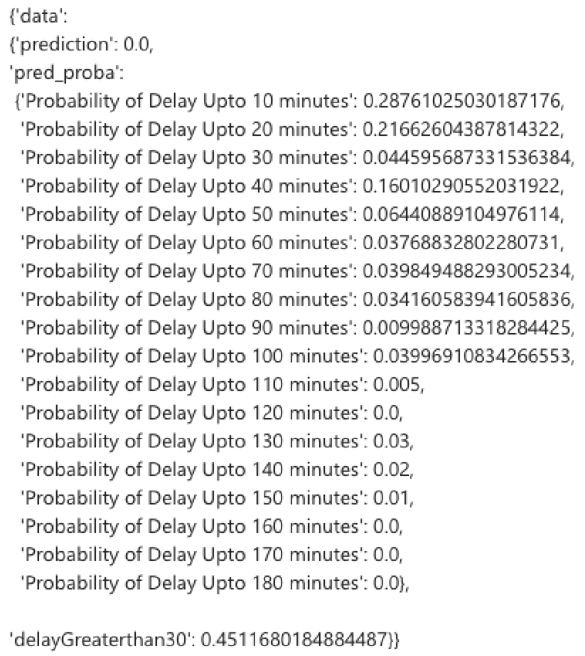


Figure 13 illustrates our overall integration procedures.

1) Accept user input and identify airport
2) Lookup in JSON file and find out the encoder to be used
3) Encode data and lookup for best model for this airport
4) Predict using the best model for the airport.

As previously mentioned, a combined version of the json file (Metadata json) will be generated at the end of our modeling, which acts as an input to the API gateway. Once converted to API, a call can be made for API to find the best performing model for each airport to make a predictions and sends the response formatted as follows:



```
{'data':
{'prediction': 0.0,
'pred_proba':
 {'Probability of Delay Upto 10 minutes': 0.28761025030187176,
  'Probability of Delay Upto 20 minutes': 0.21662604387814322,
  'Probability of Delay Upto 30 minutes': 0.044595687331536384,
  'Probability of Delay Upto 40 minutes': 0.16010290552031922,
  'Probability of Delay Upto 50 minutes': 0.06440889104976114,
  'Probability of Delay Upto 60 minutes': 0.03768832802280731,
  'Probability of Delay Upto 70 minutes': 0.039849488293005234,
  'Probability of Delay Upto 80 minutes': 0.034160583941605836,
  'Probability of Delay Upto 90 minutes': 0.009988713318284425,
  'Probability of Delay Upto 100 minutes': 0.03996910834266553,
  'Probability of Delay Upto 110 minutes': 0.005,
  'Probability of Delay Upto 120 minutes': 0.0,
  'Probability of Delay Upto 130 minutes': 0.03,
  'Probability of Delay Upto 140 minutes': 0.02,
  'Probability of Delay Upto 150 minutes': 0.01,
  'Probability of Delay Upto 160 minutes': 0.0,
  'Probability of Delay Upto 170 minutes': 0.0,
  'Probability of Delay Upto 180 minutes': 0.0},

'delayGreaterthan30': 0.4511680184884487}}
```

As can be seen above, prediction indicates which bucket or 10 minute-interval that the model predicted, in this case 0 means 0 - 10 minute interval. Predict_proba objective indicates the probability of predicted class being in each bucket of 10 minute interval. Therefore, for an example above, the sum of probabilities for 30 minutes and greater equate to 45%, which aligns with predicted class, 0-10 minute.

7. Conclusion

Previous research has been mainly involved with building a regression or binary classification model with a specific threshold for delay. In this report, we presented a multiclass-multioutput approach with three different classification models. With each airport treated uniquely, it is given a choice of selecting a feature selection method among three in conjunction with three models. As a result, we discovered that for each airport, a different feature selection method was chosen, yet Random Forest Classification seems to come out on top for all airports.

Random Forest multiclass classification proved its' efficacy through high weighted accuracy along with supplementary metrics such as recall, precision and f1 score. Furthermore, with highly accurate predictions, we were able to generate probabilities of a response variable being in each 10 minute interval. Adding these probabilities with a delay threshold of 30 minutes, we were able to calculate likelihood of a flight being delayed and likewise for non-delayed.

8. Possible Future Extension

Although with every step of modeling, it is attempted to optimize parameter tuning, challenges still remain. Future work will be involved with the followings:

1. Dive deeper into hyper-parameter tuning for XGBoosting and K-nearest neighbor
2. Try different intervals other than 10 minutes.
3. Input data used for training the model are hourly averaged entries. There might have been pitfalls utilizing averaged values for training, over-simplifying or overfitting the model. In future, differently scaled time series data need to be tested.
4. Although METAR and ASPM data are merged for input, we believe trained predictors are limited and some vital ones might have been excluded. For instance airline type, or destination airports.
5. Delay defining threshold 30 minutes was tested for the purpose of this project. Various delay thresholds need to be tested.
6. As mentioned, for this project 34 major airports are involved. More airports' inputs need to be tested.

9. References

1. Sternberg A, Soares J, Carvalho D, Ogasawara E. A review on flight delay prediction. arXiv preprint arXiv:1703.06118. 2017 Mar 15.

2. Horiguchi Y, Baba Y, Kashima H, Suzuki M, Kayahara H, Maeno J. Predicting fuel consumption and flight delays for low-cost airlines. InTwenty-Ninth IAAI Conference 2017 Feb 8.

3. Rebollo JJ, Balakrishnan H. Characterization and prediction of air traffic delays. Transportation research part C: Emerging technologies. 2014 Jul 1;44:231-41.

4. Krawczyk B. Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence. 2016 Nov;5(4):221-32.

5. Breiman L. Random forests. Machine learning. 2001 Oct;45(1):5-32.

6. Chen T, Guestrin C. Xgboost: A scalable tree boosting system. InProceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining 2016 Aug 13 (pp. 785-794).

7. Powers DM. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. arXiv preprint arXiv:2010.16061. 2020 Oct 11.

8. Accuracy, Precision, Recall or F1? Towards Data Science Web Article. 2018 Mar 15., can be retrieved from: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9