CIS 520 Fall 2020, Project Report

US Flight Delay Prediction

Team Name: Flight Delay Prediction
Team Members:

- Xinyuan Zhao; Email: `zxy9815@seas.upenn.edu`
- Tinghao Lu; Email: `tinghao@seas.upenn.edu`
- Songhao Liu; Email: `liuso@seas.upenn.edu`
- Chenyuan Wu; Email: `wucy@seas.upenn.edu`

Abstract

This project aims to predict the flight arrival delay at Los Angeles Airport(LAX) based on information that are known to the travelers before flights. To explore the relationship between features and the arrival delay time, we performed feature selections and then apply various supervised machine learning approaches to learn from the US Flight Delay data, such as kernel regression, SVM, gradient tree boosting, etc. We set KNN as our baseline model and compare every other model with the KNN model to find the better one. By comparing mean squared error from each model, the best model is Gradient Tree Boosting with a test MSE of 66.34. The Random Forest model has the second lowest MSE of 68.50.

1 Motivation

Nowadays, air traffic has become more convenient and popular. According to the statistics, there are 252 million people traveling to or from the United States in 2019. Los Angeles Airport (LAX) is one of the busies airport in 2019, hosting 88.06 million passengers in that year according to Wikipedia. However, with more and more passengers and airline companies, delay becomes more and more frequent, which is always an annoying thing to passengers. Moreover, as airlines may adjust their route and flight speed, the delay at departure is often different from the actual delay when arriving at the destinations. Therefore, before people go onto their flights, they would like to know what will be their actual arrival delay based on the current known information. With these interests, our project is aimed to predict actual arrival delay time in terms of the flight information, such as Scheduled departure time, Weekday, Airline and Distance, etc. Thus people can have more knowledge about the actual arrival time before boarding, which will allow them to plan ahead and make their trip more efficient.

2 Data Set

We used the data set provided by the U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics, which tracks the on-time performance of domestic flights operated by large air carriers during 2015: https://www.kaggle.com/usdot/flight-delays. This dataset includes $n = 5,819,079$ observations and $p = 31$ features. Each observation is described by the following variables: dates of the flight, origin and destination airports, airline, flight distance, scheduled departure time and arrival time, and departure delay and arrival delay. Among these variables, we think the Arrival Delay is of our major concern because travelers often care about their actual arrival time and thus should be predicted given the selected remaining features.

In order to select the best features correlated with the Arrival Delay, we could first visualize the correlations between features using heatmap. (Figure 1)

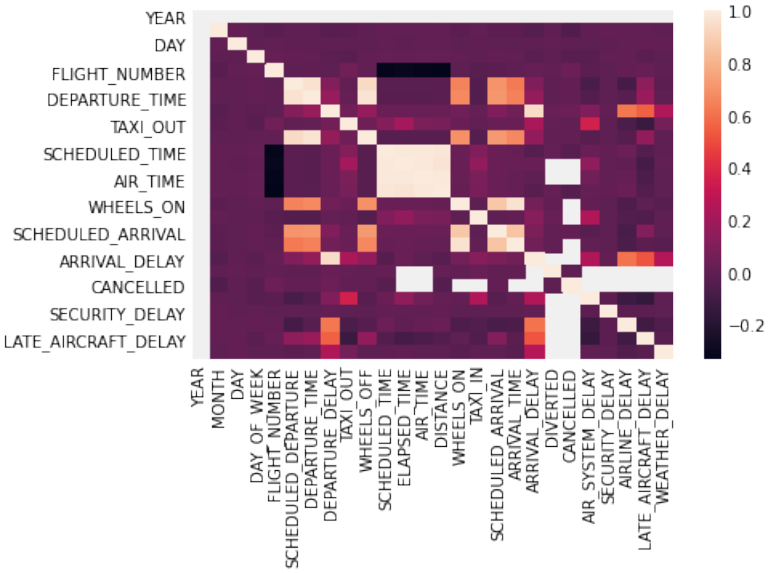


Figure 1: Correlation between Features

From Figure 1 we can observe that Air time, Distance and Scheduled time are clearly correlated. Scheduled Departure, Departure Delay and Arrival time and delay are also correlated. We could also visualize relationships between selected features and arrival delay using scatter and bar plots.

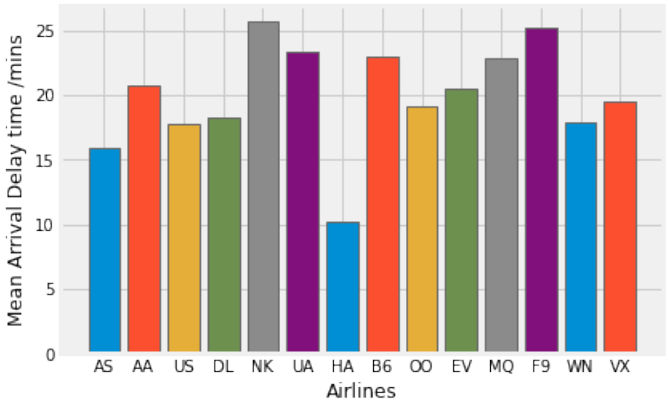


Figure 2: Relationship between Airlines and Mean Arrival Delay

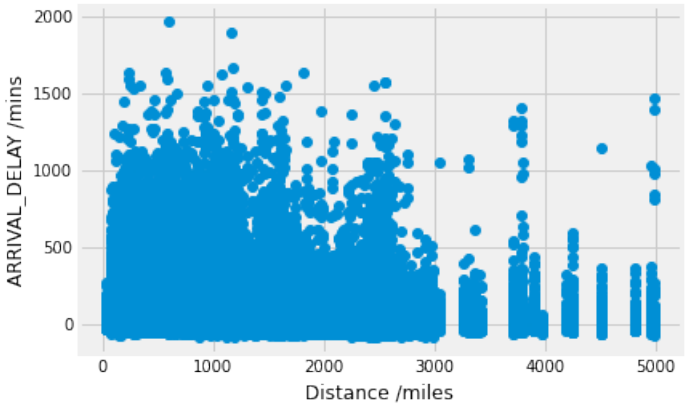


Figure 3: Relationship between Travel Distance and Mean Arrival Delay

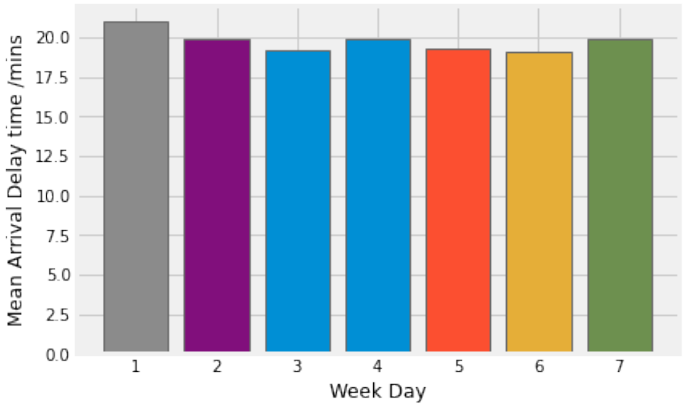


Figure 4: Relationship between Day of Week and Mean Arrival Delay

From Figure 2 we can observe that there is a correlation between airlines and delay times. It is also interesting that when travel distance increase, the arrival delay tends to decrease (Figure 3). This is because airlines often increase the flight speed to compensate for the departure delay and this will have larger effects on long distance flights. There is no obvious relationships between Day of the Week and arrival delays, but mean delay times are slightly higher on Mondays and Tuesdays as we can see from Figure 4.

We will select features as input based on the level of correlation with label and which information is known to the travelers before boarding. As the number of samples is very large in this data set, selecting samples only from LAX as the origin airport will significantly shrink down the size and so computation time is reasonable for colab. From Table 1, we can see the most frequent origin airports in the data set. There are over 190k samples from LAX and we will use them to train and test our model.

Origin Airport	**Number of Samples**
ATL	346836
ORD	285884
DFW	239551
DEN	196055
LAX	194673

Table 1: Top 5 Origin Airports with Most Number of Samples

The dataset also contains a reasonable amount of missing values. Among all the features, the highest

percentage of missing values is 1.8%. For pre-processing, we decided to drop all samples with missing values since they consist of very small portion of the data. Then we drop all samples with extreme arrival delay times (Arrival Delay greater than 60 mins for example) because these samples are rare (mainly caused by unexpected reasons) and will produce large bias to our trained model. Then we extract samples only from LAX as origin airport to shrink the sample size and exclude the unrelated features. Sample and feature selections are described in detail below.

3 Related Work

On kaggle, there are many teams doing various kind of analysis using the same dataset. Some teams use polynomial regressions with L2 regularization to predict departure delay[1]. Some other teams are aiming at predicting arrival delays by using random forest regression and boosted regression, concluding that departure delay is the main problem which is creating arrival delay in the aviation industry[2].

In research literature, apart from what had been adopted on Kaggle, researchers use more powerful and complicated machine learning models to explore this problem. Manna et al.[3] investigated the effectiveness of Gradient Tree Boosting in the air traffic delay prediction tasks. With the help of this model, day-to-day sequences of the departure and arrival flight delays of an individual airport can be predicted efficiently. Kim et al.[4] also modeled day-to-day sequences of the departure and arrival flight delays of an individual airport by using LSTM, and studied four different ways of building deep RNN architecture. Gui et al.[5] compared LSTM with random forest-based model on a dataset that covers a broader scope of factors than ours, including automatic dependent surveillance-broadcast (ADS-B) messages and other information such as weather condition and flight schedule. Yu et al.[6] studied a novel deep belief network method that is employed to mine the inner patterns of flight delays on high-dimensional data from Beijing International Airport. Their proposed method ultimately enables connected airports to collectively alleviate delay propagation within their network through collaborative efforts (e.g., delay prediction synchronization).

4 Problem Formulation

We can use this data set to train a Delay Time Prediction System. The original data set contains 31 features (flight information) and over 5 million samples. We are only interested in a small portion of the samples and features:

- Choice of Samples:
  We decided to choose all samples from only `LAX` as the origin airport. As we can see from Table 1 that LAX has reasonable amount of sample flights for us to train and test our models and it is also one of the most popular airport for US travelers.

- Choice of Features:
  First we exclude the features that are not known to the travelers before boarding so that travelers could use our models to predict their arrival delay based on information known. Therefore, features like Taxi time, Wheels On/OFF etc. are excluded from input. We also excluded features we do not care about in this problem such as Weather Delay, Cancellation, Air System Delay etc. For the remaining features, it contains real values like Departure Delay, Scheduled times and Dates as well as categorical features like Airline and Destination Airport. For real valued features we can observe from the heatmap (Figure 5) that departure delay and arrival delay are closely correlated, so we are going to keep departure delay as one of the input features. Furthermore, we can see from Figure 6 that Departure and Arrival delays have a linear relationship with a non-trivial amount of variance. For categorical features, we use one-hot encoding to check their correlations with Arrival Delay and Departure Delay. From Figure 2, we already saw that there are correlations between airlines and delay times. Surprisingly, the Destination Airports have almost no correlation with delay times (See Figure 7). As Destination Airports have 322 unique categories, we decided to exclude them from the input features because they would cause significant feature expansion and have low correlation with delay times.

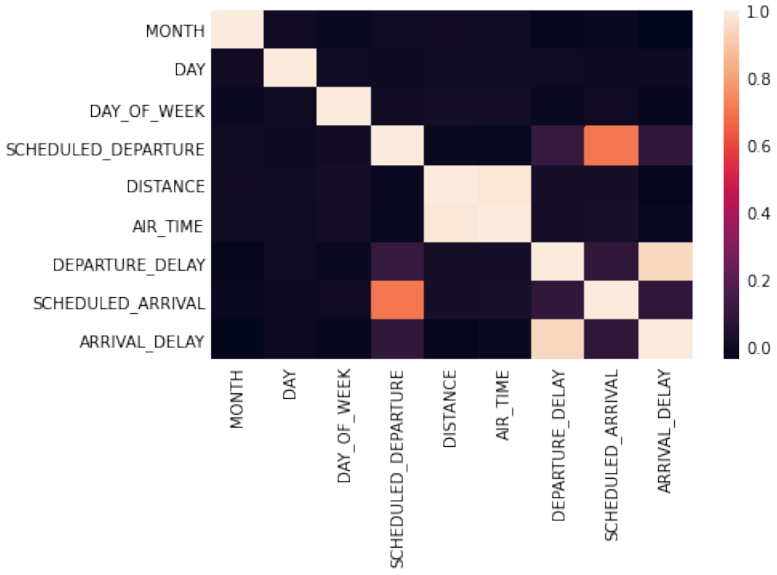


Figure 5: Correlation between Selected Real Value Features

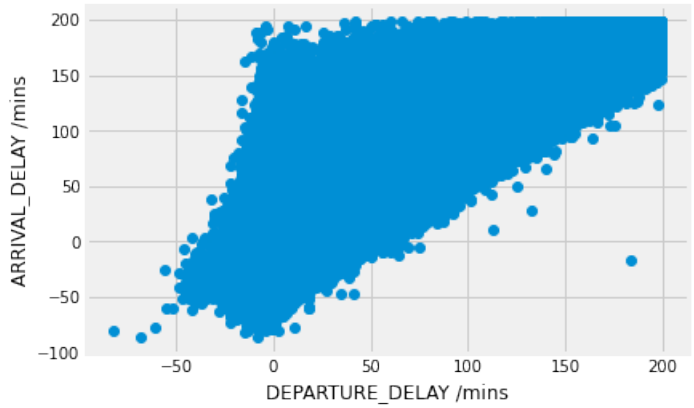


Figure 6: Relationship Between Departure and Arrival Delay times

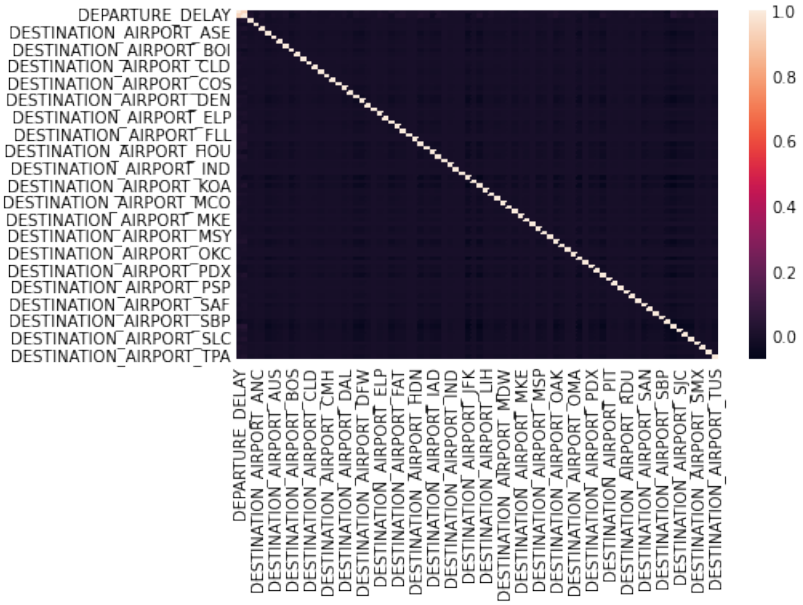


Figure 7: Correlation Between Destination Airport and Delay times

After selecting features, we can formulate a supervised learning problem: predict the Arrival Delay time given selected flight information. So we have a real valued $\mathbf{Y}$ which is `Arrival Delay time`. For input $\mathbf{X}$, we decided to keep `Departure Delay, Dates, Scheduled Departure and Arrival times, Air time, Distance and Airlines` as features.

For pre-processing, we need to remove excluded features, extract only samples from LAX as Origin Airport, and Encode `Airline` using one-hot encoding since there is only 14 distinct airlines. The resulting dimension of the data is: $n = 179,346$, $p = 21$. Then we split the data into training, validation and test sets by ratio of (0.8, 0.1, 0.1). So we have $143,476$ training samples and $17,935$ for both testing and validation samples.

For loss function, we decided to use Mean Squared Error (MSE) to optimize because it will punish more on the large errors. As MSE squares the errors and the units of data is also squared up, we also check the mean absolute error to directly evaluate our models.

5 Methods

5.1 Heuristic: Predict Directly by Departure Delay

As we decided to include Departure Delay into our input features, the first question we would naturally ask ourselves is can we know arrival delay directly from departure delay? The answer is we do know to some extent depending how accurate one wants to predict. We can see from Figure 6 that departure and arrival delays have a linear relationship, but they are not exactly linear. Based on our MSE and MAE evaluations, the MSE between Departure and Arrival Delays in our test set is 170.38, and MAE is 10.04 mins. This means that on average when predicting arrival delay purely by departure delay you will get a error of 10 mins, which is not bad, but we want to show that by including more features and apply Machine Learning Methods to learn the data, people can get a much better prediction.

5.2 Baseline: KNN

Since the non-linearity of the data and the continuity of target value, we decided to use K-nearest neighbors regression as our baseline model, which is also a very intuitive way to explore the data. We then used Skicit-learn KNeighborsRegressor. The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set. Then we experimented with the parameter k by using GridSearchCV to find the optimal patameter K.

5.3 Linear Regression

A linear regression model is a linear approach which models the relationship between target and one or more variables. To prevent overfitting of the model, penalties should also be applied, such as L1, L2, and Elastic Net(combination of L1 and L2). L1 penalty adds absolute value of magnitude of coefficient as penalty term to the loss function and L2 adds squared magnitude of coefficient as penalty term.
Even though the data is non-linear, we first try Linear Regression with l1 l2 and elastic net regularization to prevent over-fitting. Because of the non-linearity, we choose polynomial regression with a degree of 3 to best catch the relationship between target and variables by transforming the features.

5.4 Kernel Regression

In Kernel regression, a kernel is a weighting function which measures the similarity between two points, like distance. These kernel values are used to derive weights at each xi to determine the weight to predict from given input values.

$$\hat{y}(x) = \frac{\sum_{i=1}^{N} K(x, x_i) y_i}{\sum_{i=1}^{N} K(x, x_i)} \tag{1}$$

The choices of kernels and bandwidth can be different. Here, I tried two kernels introduced in the class: linear kernel and Gaussian kernel.

Linear kernel:

$$K(x, y) = x^T y \tag{2}$$

Gaussian Kernel:

$$K(x, y) = exp(-\frac{||x - y||^2}{\sigma^2}) \tag{3}$$

Kernel ridge regression

Ridge regression adds a regularization penalty to the cost term. Kernel ridge regression combines ridge regression with the kernel trick. KRR learns a linear function in the space induced by the respective kernel.

$$argmin \frac{1}{n} \sum_{i=1}^{N} ||f_i - y_i||_2^2 + \lambda ||f||^2 \tag{4}$$

$$f_i = \sum_{i=1}^{N} \alpha_j K_{i,j} \tag{5}$$

The solution can be written in a closed form as:

$$\alpha = (K + \lambda I)^{-1} y \tag{6}$$

where K is the kernel matrix and α is the vector of weights induced by the kernel.

5.5 Linear and Kernel SVM

Support Vector Machine tries to minimize the L_2 norm of the weights (w) and the sum of slack variables (ξ). It has a loss function of

$$loss = argmin_{w,b,\xi_i} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i \tag{7}$$

Unlike Linear Regressions, the SVM Regression allows us to choose error margin (ϵ) and tolerance to the slack variables (C). So we decided to try using Linear, Polynomial and RBF Kernel SVM with tuning ϵ and C to see the performance on our data set compared to other methods.

5.6 Ensemble Methods

As many real world ML challenges show, ensemble methods have superior performance on medium sized non-linear datasets over any individual method, which is exactly our case. Thus, We decided to use three ensemble methods for our problem: random forest regression, Adaboost, gradient tree boosting.

In random forests, each tree in the ensemble is built from a sample drawn with replacement from the training set. Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of the features. The purpose of these two sources of randomness is to decrease the variance of the forest estimator, reducing the chance of overfit. Note that though decision tree is widely used in classification problems, it can also be used in regression, e.g. it can learn from data to approximate a sine curve with a set of if-then-else decision rules.

An Adaboost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases. We use an exponential loss when updating the weights after each boosting iteration, together with three different base regressors, i.e. ridge regression, lasso regression and elastic nets.

Grdient tree boosting is current state-of-the-art for moderate-sized datasets, which performs on average slightly bettet than random forests. It is very similar to Adaboost, except that it only uses a decision tree of specified depth as a base learner under L2 loss, and that it does stagewise estimation of $h(x)$.

5.7 Implementation

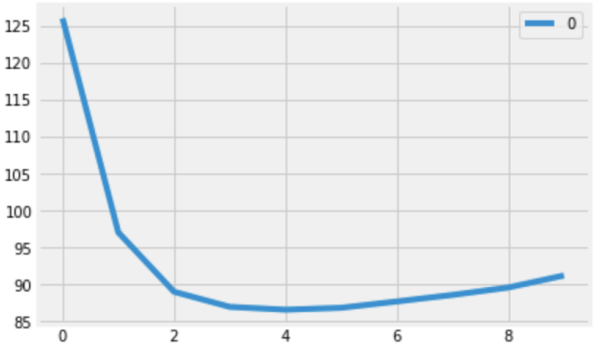
To implement the above models, we use Sklearn to train the model. Then we will choose the hyper paramenters on the validation set. Details are reported in Section 6.

6 Evaluation

Given the nature of this regression problem, we decide to use Mean Squared Error as the basic metrics for evaluation. We train our models on a training set using the same input features as described in section 4 (consists of 143,476 observations), and then pick the best hyper parameters for each model on a validation set (consists of 17,935 observations). Our test set also consists of 17,935 observations. Below we report for each method the best hyper parameters we have chosen and the testing MSE.

1. **Baseline: KNN**
   We first try the most intuitive method, KNN regression. In order to choose the most optimal K, we try K ranges from 1 to 10.We evaluate model performance for each K by using the validation set. The results and plot are listed below.

```
MSE for k=  1 is: 126.02732088095902
MSE for k=  2 is: 97.07152216336772
MSE for k=  3 is: 89.03743766068828
MSE for k=  4 is: 86.99207903540564
MSE for k=  5 is: 86.60618678561472
MSE for k=  6 is: 86.86538580677137
MSE for k=  7 is: 87.72228512257983
MSE for k=  8 is: 88.60640768748257
MSE for k=  9 is: 89.60582969364683
MSE for k=  10 is: 91.236676331196
```



As shown in the above result and graph. We can see that the optimal K is **5** which generates the lowest MSE on the test set, **85.104**.

2. **Linear Regression with L1, L2, ElasticNet Regularizations, Polynomial Regression**
We tried linear regression with L1 L2 and Elastic Net(L1 and L2) regularization, and find the best hyper paramter by using GridSearchCV. the MSE on the test set are listed below

Linear Regression Types	**Test Set MSE**
Linear Regression	97.7216
Ridge Regression, $\alpha = 1$	97.7216
Lasso Regression, $\alpha = 0.0001$	97.7214
Elastic Net Regression, $\alpha = 0.0001$, L1 ratio = 0.9	97.7213

As we can see here, The MSE are almost the same in different regularization, MSE = **97.72**. This may imply that the the linear regression with no regularization does not have any problem of overfitting. However, the MSE is higher than KNN's MSE, because of the non-linearity of the data. Then we try polynomial regression to catch the non-linearity nature of the data.
The **Polynomial Regression** is performed by transform the variables to degree of 3 by using PolynomialFeatures in sklearn. Then fit the linear regerssion model on the transformed polinomial feature set. The results are shown below.

Polynomial Regression Types	**Test Set MSE**
No regularization	79.2716
L1 regulatization	84.5725

By applying penalties to the linear model, the result shows that the best case is without penalty, which means the model is not overfitting at all. Thus the model with penalty will just add bias to the model, and reduce the accuracy. The model without penalty is already good to fit the highly non-linear nature of the data. Compared with the MSE in KNN regression, the MSE improved a lot in the polynomial regression, since it catches the nonlinear relationship in the data.

3. **Kernel Ridge Regression**

There are several parameters that we can choose (Alpha, kernel types). According to the method we mentioned, there is an close form solution. However, people often use grid search to find the best parameter since it is faster. Here is the Test MSE table:

Kernel Types	**Alpha=1**	**Alpha=0.1**	**Alpha=0.01**	**Alpha=0.001**	**Alpha=0.0001**
Linear	100.4117	100.4259	100.4311	100.4282	100.4484
Polynomial	101.9342	114.6744	103.1740	103.1740	103.1740
RBF	294.3849	293.5467	293.3997	293.3839	293.3823

Table 2: MSE based on Different Kernel types and Alpha

We find the best parameter is $\alpha = 1$, linear kernel, which gives the best performance. And MSE on the test set is 100.4117.

4. **Kernel Regression**

Compared to KNN, kernel regression selects all the neighbors with different weights instead of all neighbors with equal weight and the number of neighbors customized globally.

We built two kernels learned in the lecture, Linear and Gaussian and calculated the MSE on test set:

Kernel Types	**Test Set MSE**
Linear	291.5109
Gaussian	227.0081

Table 3: MSE based on Different kernel types

And about Gaussian kernel, we calculate MSE based on different σ and select the $\sigma = 0.05$ since the MSE is lowest. When the sigma value gets smaller, the kernel regression model gets more complex and easier to overfit data. When the sigma value gets larger, the kernel regression model tends to be similar to a linear regression model and easier to underfit data.

σ	**Test Set MSE**
0.05	227.0081
0.1	251.7842
0.15	259.8088
0.2	262.3617
0.5	266.1899

Table 4: MSE based on Different σ

5. **Linear and Kernel SVM**

We first try to apply `Linear, Polynomial and RBF` kernels to SVM with default hyperparameters and evaluate the performance of the models using test set. The results are shown below.

Kernel Types	**Test Set MSE**
Linear	100.737
Polynomial	316.436
RBF	319.984

As shown in the above table, the Linear SVM clearly has the best performance in this data set. As some features show linear relationships with Y we are predicting, this result makes sense. Then we use the linear SVM model and test for different regularization parameters. There are many parameters we could choose: we can choose either **L1** or **L2** as loss function, either solve for **Dual** or **Primal** Optimization and choose the best slack penalty term **C**.

After training models with either loss functions and optimizations and test using validation sets, we found that the combination of **L2** loss and **Primal** optimization has the best performance. This is because the number of samples in our dataset is much larger than the number of features ($n >> p$), and our dataset is non-sparse. Then we continue to test for different values of **C**. From Figure 8 we can see that the validation MSE is minimum at $C = 0.05$. Using these hyperparameters, we finally train the model and get a MSE of the test set as **97.889**, and the corresponding MAE is 7.4 mins.

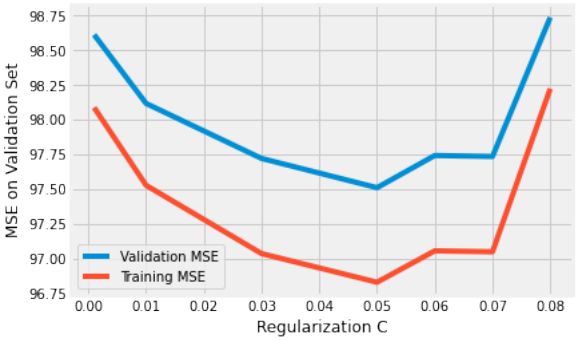


Figure 8: Training and Validation MSE vs. Regularization C

6. Ensemble Methods

We first implemented Gradient Tree Boosting with L2 loss function. Then we tuned the maximum depth of the base learners, which are fixed depth regression trees. As Figure 9 shows, when we increase the maximum tree depth, the training MSE always decrease and even reach zero when depth is around 30, while the validation MSE is lowest when depth is 16 and increase again when depth is around 30. This shows if the base learner is too strong, we have a problem of overfit. To avoid overfit, we choose depth to be 16 and measure GTB's performance on test set. Finally we get a MSE of the test set as **66.34**, and the corresponding MAE as **5.99** mins. We repeat the same process for Random Forest Regression, and finally get a MSE of the test set as **68.50**.

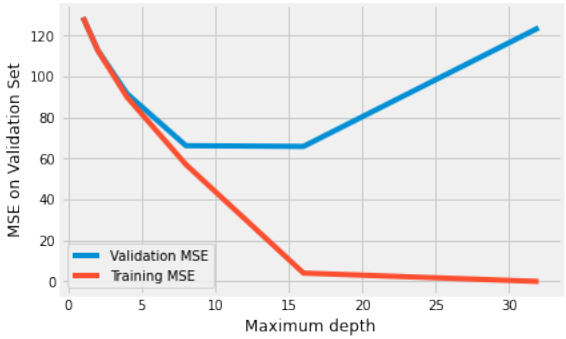


Figure 9: Training and Validation MSE vs. Tree Depth

For Adaboost, we tried different models as base estimators, including KNN, Ridge Regression, Lasso Regression, Elastic Net and Decision Tree: we fix the depth of decision tree to be 10, and adopt the best hyper parameters as described in above subsections for all other models. We report the testing MSEs in Table 5.

Base Estimator	**Testing MSE**
KNN	95.70
Ridge Regression	138.41
Lasso Regression	157.15
Elastic Net	142.29
Decision Tree	107.84

Table 5: Adaboost with Different Base Estiators

7. Summary

Method	Testing MSE
Heuristic	170.38
KNN	85.1043
Ridge Regression	97.7216
Lasso Regression	97.7214
Elastic Net	97.7213
Polynomial Regression	79.2716
Kernel Regression with linear kernel	276.1641
Kernel Regression with Gaussian kernel	227.0081
Kernel ridge Regression	100.4117
Linear SVM	97.889
Random Forest	68.50
Adaboost with Ridge Regression	138.41
Adaboost with Lasso Regression	157.15
Adaboost with Elastic Net	142.29
Adaboost with KNN	95.70
Adaboost with Decision Tree	107.84
Gradient Tree Boosting	66.34

Table 6: Summarized Evaluation Results

Here the heuristic approach means instead of using any ML models, we directly use the departure delay in the input features to approximate the arrival delay, and calculate its deviation from the real arrival delay. We can see that most of our methods outperform the heuristics, with our lowest MSE as 66.34, which is almost three times less than the heuristic's MSE. This implies well-designed ML models can obtain more accurate prediction than human heuristics, while unsuitable models do even perform worse than that.

7 Conclusion and Discussion

Overall, we have shown that using feature engineering and appropriate machine learning methods, we can achieve better predictions on Arrival Delay compared to heuristic approach and baseline method. We achieved a minimum MSE of 66.34 using Gradient Tree Boosting, which is much less than the MSE using heuristic approach (170.38) as well as the baseline KNN method (85.10). To be more intuitive, we can assess the improvement in prediction accuracy using the mean absolute error: we have improved the average error of prediction from 10.04 mins (heuristics) to 5.99 mins (Gradient Tree Boosting) for the test set. This is quite impressive given that we have limited the samples in our dataset to only include delays within 60 mins. Comparing the results from different methods, we found that Linear Methods generally have a better performance than Kernel Methods in solving our problem. This is because there is a linear relationship between departure delay and arrival delay, and the feature of departure delay is clearly the most correlated one with the output label. However, other non-linear features in our dataset are also informative and so the Tree-based Ensemble Methods turned out to be the best models in learning our dataset. Using the Random Forest model, we could also obtain a visualization of feature importance.

We used Random Forest model as the estimator and permute on every feature 5 times and take the mean score to obtain Figure 10. From the figure, we can see that Departure Delay, as expected, is the most important feature and gives the highest Information Gain. Besides, it is interesting that Distance and Air Time are also very informative in predicting Arrival Delay. As we observed the correlations between Distance and Arrival Delay in Figure 3, the Arrival Delay decreases exponentially as travel distance increases. The Airlines turned out to be the least informative features. When we visualized the data, the airlines indeed have low correlations with the label, but intuitively they definitely have influence on the Arrival Delay, which can be shown from Figure 2. This means that Random Forest models (In fact, all of our models) did a good

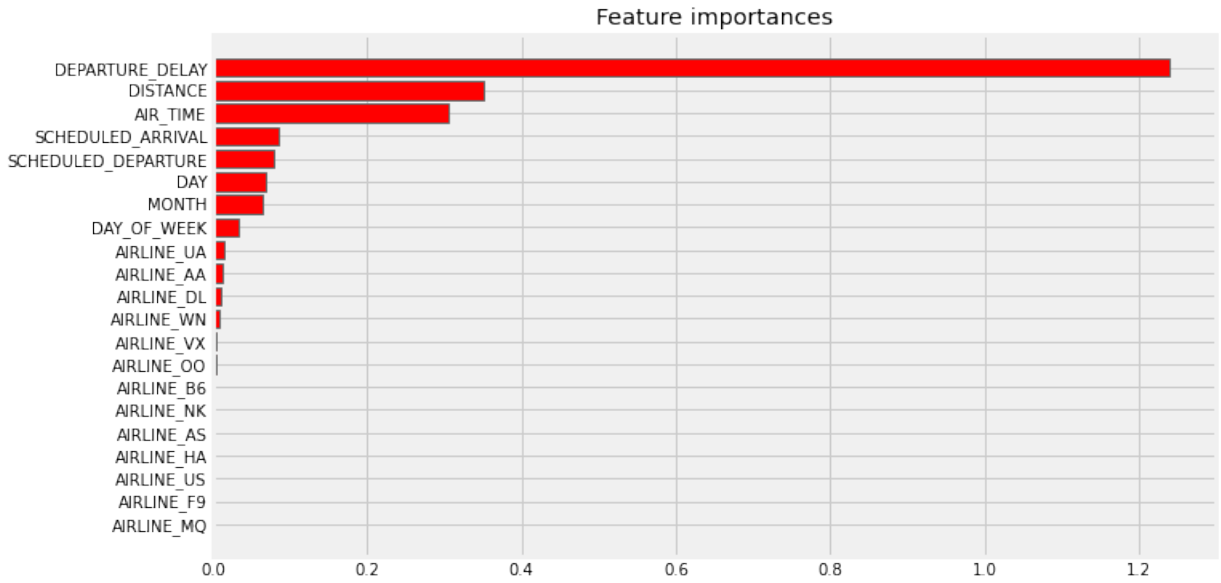


Figure 10: Permutation Importance by Random Forest

job in finding correlations but cannot find causality.

7.1 Conclusion Based On Our Models:

- KNN regression: This regression model has a optimal MSE of 85.1043, which is much better than the Heuristic MSE of 170.38. Even though this model is the baseline model, it still performs quite well. For our dataset, KNN performs better than all of the linear models, which means that KNN is good at learning non-linear data. However the accuracy is not good enough, which might be caused by high dimensionality of the features.

- Linear Regression: Linear regression method has an optimal MSE of 97.72, which is the same even with regularization. This result is caused by the non-linearity. Since most of the features are non-linear, they cannot provide useful information in prediction. The Polynomial Regression performs quite well with an optimal MSE of 79.27, better than KNN regression. By transforming the features, it reduces the affect of non-linearity. Therefore, more features become informative in predicting. However, with higher degree, the prediction could be better, because degree of 3 could not full represent the actual non-linearity.

- Kernel Regression: The regression model gives an optimal MSE of 100.4117 in the kernel ridge regression model. Adding kernel can generate a non-linear model to better fit the data and reduce the prediction error in the test set. It learns a linear function in the space induced by the respective kernel and the data. The reason why the MSE is higher than the Ridge regression maybe we only train parts of the data which might not explained the data well. When doing predictions, SVR is faster since it learns a sparse model. And KRR needs to perform a grid search to get the best hyperparameters, including alpha, types of kernel(linear, poly, rbf), etc, which can consume a lot of time.

- Linear SVM: Similar to Linear Regression Methods, the Linear SVM Regressor model has an optimal MSE of 97.889. With regularizations on slack variables and primal optimization, the Linear SVM model still cannot perform as good as KNN. Although Departure Delay is linear with the output label, most of the other features are non-linear and are also informative. So linear methods indeed give some good predictions, but their accuracies are still limited compared to ensemble methods.

- Ensemble methods: Ensemble methods have an optimal MSE of 66.34 by using Gradient Tree Boosting. We learn from our experiments that Decision Trees are strong estimators for non-linear data, but they can also overfit even with boosting. Also, we learn that if the base estimators are too weak or unsuitable for this problem, than using Adaboost doesn't give us any performance gain. For example, Adaboost with linear models perform badly, because an ensemble of linear models are still linear, while our problem is shown to be non-linear.

7.2 Future Extensions:

Our dataset contains 31 features and over 5 million samples, so there are many possible extensions based on our research in this dataset. Our research only focused on a small part of this dataset: Given Flight Information from the origin airport of LAX, can we accurately predict the Arrival Delay Time? We have tested several methods and the current best model is Gradient Tree Boosting, however, we believe that this accuracy can be improved further. If given more time, we would definitely try Auto-sklearn and Neural Networks. For future research, this problem can also be reformulated by re-selecting the input features. For example, what will be the accuracy of predicting arrival delay if considering all airports? What if Departure Delay is unknown? Flight Delay Predictions remain to be big concerns to travellers and airlines, so we believe that more machine learning researches on this dataset can lead to many more interesting and helpful results.

References

[1] Fabien Daniel. Predicting flight delays [Tutorial]. https://www.kaggle.com/fabiendaniel/predicting-flight-delays-tutorial.

[2] Abhishek Banerjee. 2015 Flight Delays and Cancellation Prediction. https://www.kaggle.com/abhishek211119/2015-flight-delays-and-cancellation-prediction.

[3] S. Manna, S. Biswas, R. Kundu, S. Rakshit, P. Gupta, and S. Barman. A statistical approach to predict flight delay using gradient boosted decision tree. In *2017 International Conference on Computational Intelligence in Data Science(ICCIDS)*, pages 1–5, 2017.

[4] Y. J. Kim, S. Choi, S. Briceno, and D. Mavris. A deep learning approach to flight delay prediction. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–6, 2016.

[5] G. Gui, F. Liu, J. Sun, J. Yang, Z. Zhou, and D. Zhao. Flight delay prediction based on aviation big data and machine learning. *IEEE Transactions on Vehicular Technology*, 69(1):140–150, 2020.

[6] Bin Yu, Zhen Guo, Sobhan Asian, Huaizhu Wang, and Gang Chen. Flight delay prediction for commercial air transport: A deep learning approach. *Transportation Research Part E: Logistics and Transportation Review*, 125:203 – 221, 2019.