

# Flight Delays: Can They be Predicted?

By Charlie Evert

Data: <https://www.kaggle.com/datasets/jimschacko/airlines-dataset-to-predict-a-delay>

## Introduction

Flight delays are a constant problem in the modern world. The amount of time and money this problem wastes are immense, and certain airlines have worse reputations than others in causing delays. Moreover, delays take place more often in the middle of the week on Tuesday, Wednesday and Thursday. The time that each flight takes is also a major factor in predicting delays since longer flights have more room for error (in terms of the time it takes for planes to arrive).

Delays are the problem that will be explored in this paper. By predicting delays, we can observe which airlines to avoid for a speedy flight, and assess the probability of having a flight delayed based on a combination of factors. My approach will be to clean airline data gathered from Kaggle, develop several classification models trained using this data, and compare the predicted results to the actual results in the test set. The main goal of this paper is to identify the ideal model in

Several sci-kit learn classification models will be used in this project, including:

- Logistic Regression
- Decision Tree Classification
- Support Vector Machine Classification
- K Nearest Neighbors Classification

Data will undergo cross-validation, and flow through several pipelines to automate a few key processes. This paper will examine the problem (including the models that can solve such a problem and expectations), the resultant model (its results, methodology and other key points) and future research topics. The source data is from a Kaggle competition, so there are many other predictive models that have been made to explore this obscure topic.

I hypothesized that there would be an accurate model, and in that airline would have the largest influence on delays followed by the length of flights and then the days of the week.

Of all the models, the KNN model was the most effective by a few key metrics. This is because the data itself is quite simple, and is therefore simply classified. My contributions here are simply another way to look at a popular dataset and apply modern ML algorithms to correctly classify whether a flight will be delayed or not. If anything, this work is most useful if you would like to figure out if a future flight will be delayed based on a combination of its flight time, airline and day of the week.

## Problem Definition and Algorithms

### Task Definition & Data Exploration/Cleaning

Flight delays are affected by a few key factors, and this project aims to explore the degree to which these factors influence delays. To effectively do so, we must dummy code the relevant string variables, minimize the amount of processing power needed to run predictive models and make sure that the data makes sense.

There are 539,383 rows of data and 9 variables in this dataset. Variable distributions can be viewed in Appendix 1. Most notably, the length of delays is skewed leftward, there are less instances of delays than the alternative (although they are fairly evenly split), there are more flights in the middle of the week, and Atlanta Airport is the most represented airport in this dataset. The following describes these variables:

Variable	Description	Used?	Input/Output?	Data Type
id	The Primary Key	No	N/A	String
Airline	The Airline of the Flight	Yes	Input	String
Flight	The Flight Number	No	N/A	String
AirportFrom	The Origin Airport	No	N/A	String
AirportTo	The Destination Airport	No	N/A	String
DayOfWeek	The Day of the Week (Monday thru Sunday)	Yes	Input	String
Time	The Length of the Flight	Yes	Input	Integer
Length	The Length of the Delay	No	N/A	Integer
Delay	Did a Delay Occur?	Yes	Output	Boolean

Important to note, Origin and Destination Airports were excluded as variables due to the large number of distinct values these attributes possess; there were unique values for 293 destinations and 80 origins, and feeding these to a classification model is not feasible without significant amounts of computing power. I took a random sample of 10,000 observations for the same reason.

I then dummy coded DayOfWeek after mapping its originally numerical values (digits 1-7 to represent the days of the week) to the names of each day in English (Sunday as 1, through Saturday as 7). This was an assumption, and the names of the days is not specified on Kaggle which is a minor limitation and may skew my interpretation of the results if the digits represent Monday through Sunday instead.

I also dummy coded the Airline variable, which was designated by 2-3 letter codes.

Overall, there were 26 variables (25 inputs and 1 target) once these variables were dummy coded.

## Model Definitions

Classification tasks are meant to predict how an incoming observation should be classified amongst one or many alternatives. In this case, our model must classify flights by whether they will (should) be delayed.

A Logistic Regression Classification Model was used to predict delays (as 0 being no delay and 1 being a delay). This works similarly to other forms of regression with the key distinction being that it is being used to classify instead of predicting a number. These models are useful in predicting the likelihood of an event, which is what we are after.

A Random Forest Classification model was used to predict delays by splitting the characteristics of delays into several branches in a tree and then merging several trees. Decision trees are useful in communicating complex relationships, but in general serve to create several groupings of classes by their likelihood of being present in a delay; a random forest serves to aggregate several decision trees into a more accurate model.

A K Nearest Neighbors Classification (KNN) model was used to predict delays by classifying based on grouping the nearest observations (in terms of location) on a graph. If K was 2, then if 2 of the nearest neighbors on a graph for a given observation are delayed flights, then the given observation is predicted/classified to be a delayed flight.

A Support Vector Machine (SVM) Classification model was used to predict delays by creating a hyperplane between dimensions and grouping similar to KNN. The hyperplane finds the optimal way to divide the data in to 2 classes on a 3-dimensional plane.

## Expectations

Given the large number of variables, I expect that there will be a highly accurate model that will be outputted. I am wary of overfitting since there are so many variables, which is why I set the training set to be only 20% of the full data frame. I don't believe that the data would be underfit given the large number of observations, and I can always expand the random sample to include more observations if this is the case.

I expect certain airlines to sway classification far more than others; for instance, I have never flown on an American Airlines flight that wasn't delayed and I expect to see this skew reflected in the predictive model. I also expect to see more delays on days with more flights, since with more flights there is a higher potential for air-traffic-jams. Finally, I expect the length of flights to be highly predictive of delays. If flights take longer, there is more potential for that flight to take longer to arrive since many flights fly back and forth between 2 cities.

## Experimental Evaluation

### Methodology

I am using a combination of metrics to evaluate my models. Not only will the accuracy of individual models be examined, but the net accuracy of models will be examined through cross validation. Means are very similar to one another as seen in cross validation, so models will be evaluated based on their risk as well. I present this in several graphs and in screenshots in the appendices.

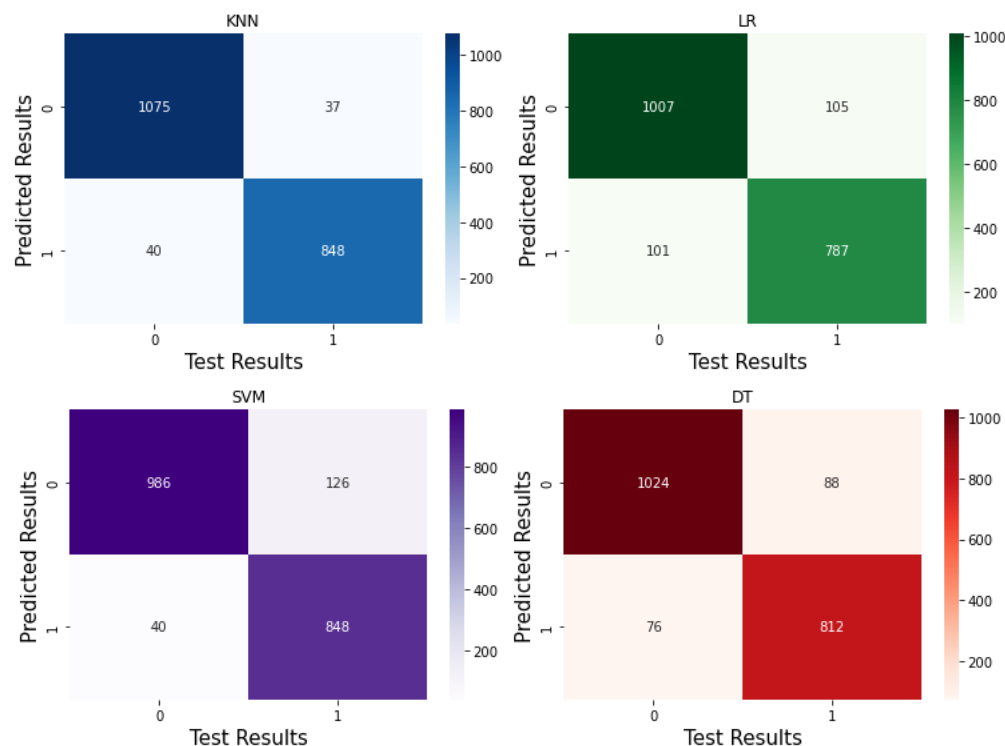
Please see Appendix 2 for the grid search. The grid search showed that the best estimator was an SVC model, which we used for the SVM model. In cross validation, it was 100% accurate in predicting the test set. A perfect score in this situation is ideal, but could mean that the model is overfit. We will see later that this is not the case.

As an ensemble method, I used a random forest model that was not as accurate as the others. Overall, the models performed as expected and each had its own strengths and weaknesses.

### Results

The accuracy of the models was very high. The models were not overfit nor were they underfit since they all performed very well with around 90% accuracy according to cross validation. Despite this, the KNN model was slightly more accurate than the others (95% accuracy), and Logistic Regression model was the least accurate of the models (89%) by a small margin. The ensemble-based random forest model had 94% accuracy and the SVM model had 92% accuracy. Considering the SVM model performed highest in the grid search, it is clear that the model was slightly overfit to the training set for this model. Standard deviations, however, were least dispersed for the logistic regression model, followed by the KNN, then SCM, then DT models. Lower standard deviations mean less risk for predictions, so in this way the LR model is the least risky of the bunch. See Appendix 3 for the accuracy scores as determined by cross validation.

Not factoring in cross validation, the models that are graphed below had different accuracy scores. In these models, the KNN model is the most accurate at 96.2%, followed by DT (91.8%), SVM (91.7%) and finally LR (89.7%). This means that the models I generated perform slightly better or worse than the average scores listed above (as generated by K-fold cross-validation). See Appendix 4 for those scores. The confusion matrices below demonstrate how each model performed, and the specifics of what was classified correctly and incorrectly.



The confusion matrices above show that KNN is clearly the most accurate model, with the least number of false delays and false no-delays (40 and 37, respectively). All models predicted more no-delays than delays, which was consistent with the distributions in the dataset. SVM had the largest amount of false no-delay predictions, and LR had the largest amount of false delay predictions.

The classification reports in Appendix 5 show more specifics about the confusion matrices above. The f1-score of each model was very similar. Like the accuracy scores above, KNN has the highest (96%), followed by DT & SVM (92%) and finally LR (90%). Interestingly, the SVM model was least accurate in predicting delays but was tied for the most accurate in predicting no-delays (with KNN). KNN was significantly better at predicting Delays than any other model.

## Discussion

The models are all had their strengths and weaknesses. Of all of them, I would most rely on the KNN model since it had the highest accuracy in both cross validation and the final model (as pictured in the confusion matrix). This makes sense, given that KNN models are very simple and so were the variables that I used to train this model. Most were simple Booleans, and a few were numerical; the trends here are not that complex to model, and therefore the simplest model best captures these trends. However, the downside of this model is that it is not human-readable; it is difficult to understand/explain why the KNN model classifies each observation as a delay or not.

## Related Work

This model was taken from Kaggle, and many other Kaggle users have used this data to create prediction models. The data quality is questionable in some respects (the time and length variables are not explained thoroughly, and the key/value pairs offered to explain which airlines are meant by abbreviated values are not all accurate). Despite this, the model was clearly designed with classification in mind, and my KNN model was as accurate as many other models on the platform.

## Future Work

The greatest shortcoming of this work was my inability to dummy code airports, as this data would undoubtedly aid the prediction models. I could not do so because there were hundreds of these values, and my computer could not handle creating models with hundreds of attributes and 10k rows. Future work will make use of cloud services to circumnavigate this issue of insufficient computing power.

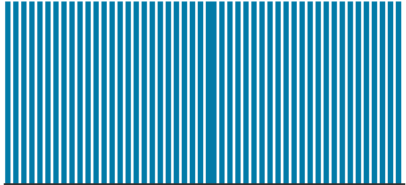


Other shortcomings include the need to only use 10k rows and not visualizing the results more. These two shortcomings can be attributed to the same lack of computing power as above.

## Conclusion

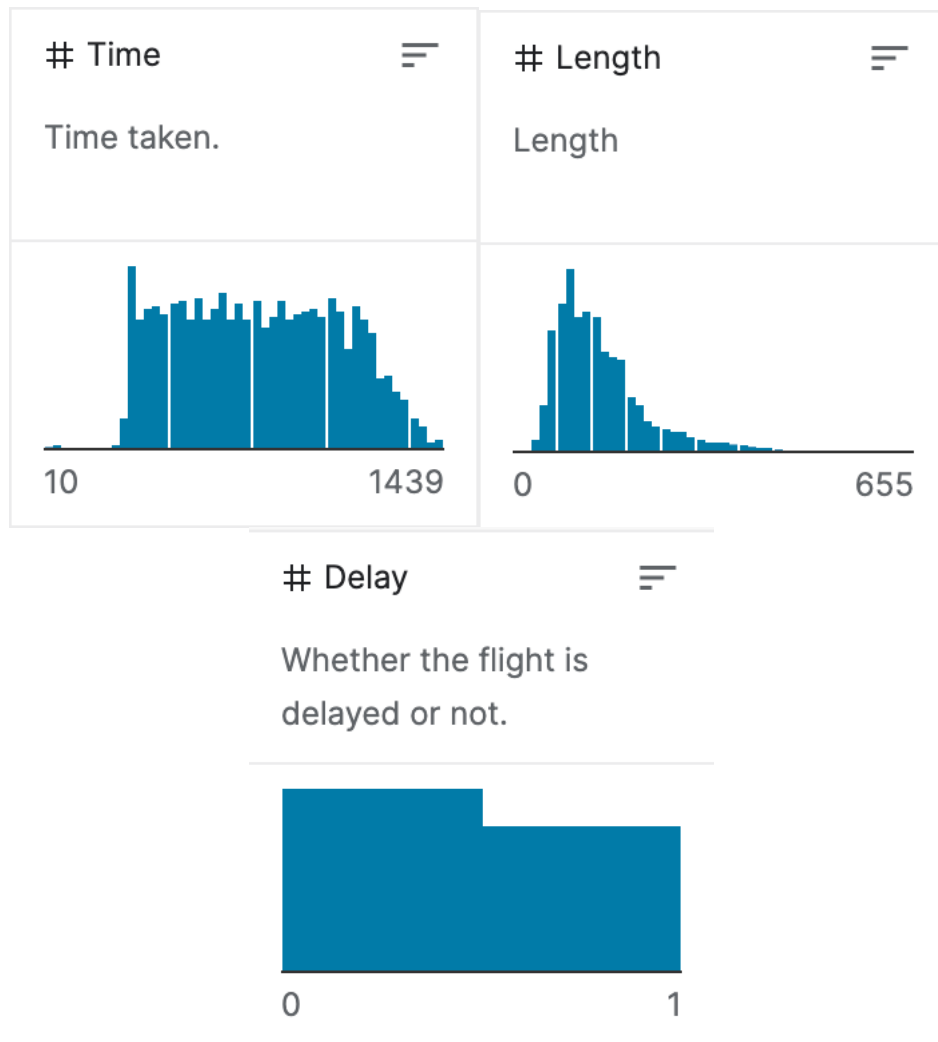
The most important finding of this work is that it is quite simple to classify flights into whether or not they will be delayed. This is great, as predictable results mean that people can plan for the future. I would love to use the KNN model to see if a flight I am taking in a few weeks is likely to be delayed- if it is, then I should probably change flights because I need to be at an event shortly after landing.

It is also important to note that the grid search used in this project pointed to the SVM model being the most accurate, when in fact it was overfitted to the training data. The SVM model did perform well, but by the nature of the algorithm it is highly probable that it would be overfitted, which I did not fully expect and turned out to be the case. A model that is 100% accurate on the training set is indicative of an inaccurate model in many cases.

## Appendix 1: Variable Distributions

<div>🔑 id</div> <div>Serial No</div> <div>  </div>	<div>✈️ Airline</div> <div>Different types of commercial airlines</div> <div> <table> <tr> <td>WN</td><td>17%</td></tr> <tr> <td>DL</td><td>11%</td></tr> <tr> <td>Other (384346)</td><td>71%</td></tr> </table> </div>	WN	17%	DL	11%	Other (384346)	71%
WN	17%						
DL	11%						
Other (384346)	71%						
<div># Flight</div> <div>Types of Aircraft</div> <div>  </div>	<div>✈️ AirportFrom</div> <div>Source Airport</div> <div> <table> <tr> <td>ATL</td><td>6%</td></tr> <tr> <td>ORD</td><td>5%</td></tr> <tr> <td>Other (480112)</td><td>89%</td></tr> </table> </div>	ATL	6%	ORD	5%	Other (480112)	89%
ATL	6%						
ORD	5%						
Other (480112)	89%						
<div>✈️ AirportTo</div> <div>Destination Airport</div> <div> <table> <tr> <td>ATL</td><td>6%</td></tr> <tr> <td>ORD</td><td>5%</td></tr> <tr> <td>Other (480072)</td><td>89%</td></tr> </table> </div>	ATL	6%	ORD	5%	Other (480072)	89%	<div># DayOfWeek</div> <div>Tells you about the day of week</div> <div>  </div>
ATL	6%						
ORD	5%						
Other (480072)	89%						

## Appendix 1: Variable Distributions (Cont.)





## Appendix 2: Grid Search

```
In [466]: param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
...:                  'gamma': [0.001, 0.01, 0.1, 1, 10, 100]}
...: from sklearn.model_selection import GridSearchCV
...: grid_search = GridSearchCV(SVC(), param_grid, cv=5)
...: grid_search.fit(X_train, y_train)
...: test_score = grid_search.score(X_test, y_test)
...: print("Test set score {:.2f}".format(test_score))
...: print("Best parameters: {}".format(grid_search.best_params_))
...: print("Best cross-validation score: {:.2f}".format(grid_search.best_score_))
...: print("Best estimator:\n{}".format(grid_search.best_estimator_))
Test set score 1.00
Best parameters: {'C': 100, 'gamma': 0.001}
Best cross-validation score: 1.00
Best estimator:
SVC(C=100, gamma=0.001)
```

## Appendix 3: Cross Validation Model Accuracy Scores

```
In [504]: accuracies_lr = cross_val_score(estimator = pipeline_lr, X = X_train, y = y_train, cv = 10)
...: print(accuracies_lr.mean())
...: print(accuracies_lr.std())
0.8928750000000001
0.013440261344185231
```

```
In [505]: accuracies_dt = cross_val_score(estimator = pipeline_dt, X = X_train, y = y_train, cv = 10)
...: print(accuracies_dt.mean())
...: print(accuracies_dt.std())
0.9369999999999999
0.01700367607313195
```

```
In [506]: accuracies_svm = cross_val_score(estimator = pipeline_svm, X = X_train, y = y_train, cv = 10)
...: print(accuracies_svm.mean())
...: print(accuracies_svm.std())
0.9176249999999999
0.016830496873235797
```

```
In [507]: accuracies_knn = cross_val_score(estimator = pipeline_knn, X = X_train, y = y_train, cv = 10)
...: print(accuracies_knn.mean())
...: print(accuracies_knn.std())
0.946875
0.015724682031761413
```

## Appendix 4: Actual Model Accuracy Scores

```
In [508]:  
...: for i,model in enumerate(pipelines):  
...:     print("{} Test Accuracy:{}".format(pipe_dict[i],model.score(X_test,y_test)))  
Logistic Regression Test Accuracy:0.897  
Decision Tree Test Accuracy:0.918  
Support Vector Machine Test Accuracy:0.917  
K Nearest Neighbor Test Accuracy:0.9615
```

## Appendix 5: Classification Reports

```
In [491]: print(classification_report(y_test, y_preds_knn, target_names=target_names))
```

	precision	recall	f1-score	support
No Delay	0.96	0.97	0.97	1112
Delay	0.96	0.95	0.96	888
accuracy			0.96	2000
macro avg	0.96	0.96	0.96	2000
weighted avg	0.96	0.96	0.96	2000

```
In [492]: print(classification_report(y_test, y_preds_lr, target_names=target_names))
```

	precision	recall	f1-score	support
No Delay	0.91	0.91	0.91	1112
Delay	0.88	0.89	0.88	888
accuracy			0.90	2000
macro avg	0.90	0.90	0.90	2000
weighted avg	0.90	0.90	0.90	2000

```
In [493]: print(classification_report(y_test, y_preds_svm, target_names=target_names))
```

	precision	recall	f1-score	support
No Delay	0.96	0.89	0.92	1112
Delay	0.87	0.95	0.91	888
accuracy			0.92	2000
macro avg	0.92	0.92	0.92	2000
weighted avg	0.92	0.92	0.92	2000

```
In [494]: print(classification_report(y_test, y_preds_dt, target_names=target_names))
```

	precision	recall	f1-score	support
No Delay	0.93	0.92	0.93	1112
Delay	0.90	0.91	0.91	888
accuracy			0.92	2000
macro avg	0.92	0.92	0.92	2000
weighted avg	0.92	0.92	0.92	2000