STAT 510 Fall 2023

Final Report

Bonnie Tran and Ian Lee

Prediction of Aircraft Runway Time

Motivation

Our goal is to predict the amount of time commercial aircrafts at JFK (New York Airport) take on the runway. This will increase customer satisfaction through saving time, and prevent profit loss through better air traffic management with planes arriving or departing.

Data Summary

Input Features*: Total of 22 input features (16 numerical, 6 categorical)*

- **MONTH, DAY_OF_MONTH, DAY_OF_WEEK** - Time Variables
- **OP_UNIQUE_CARRIER** - Carrier Company (Delta, American Airlines)
- **TAIL_NUM** - Airflight number
- **DEST** - Destination
- **DEP_DELAY** - Departure delay of the flight
- **CRS_ELAPSED_TIME** - Scheduled journey time of the flight
- **DISTANCE** - Distance of the flight
- **CRS_DEP_M** - Scheduled departure time
- **DEP_TIME_M** - Actual departure time
- **CRS_ARR_M** - Scheduled arrival time
- **Temperature, Dew Point, Humidity, Wind, Wind Speed, Wind Gust, Pressure, Condition** - Regarding the climate/weather
- **sch_dep** - Number of flights scheduled for arrival
- **sch_arr** - Number of flights scheduled for departure
    - *Categorical features:* OP_UNIQUE_CARRIER, TAIL_NUM, DEST, Wind, Condition, Dew Point

Output Feature:

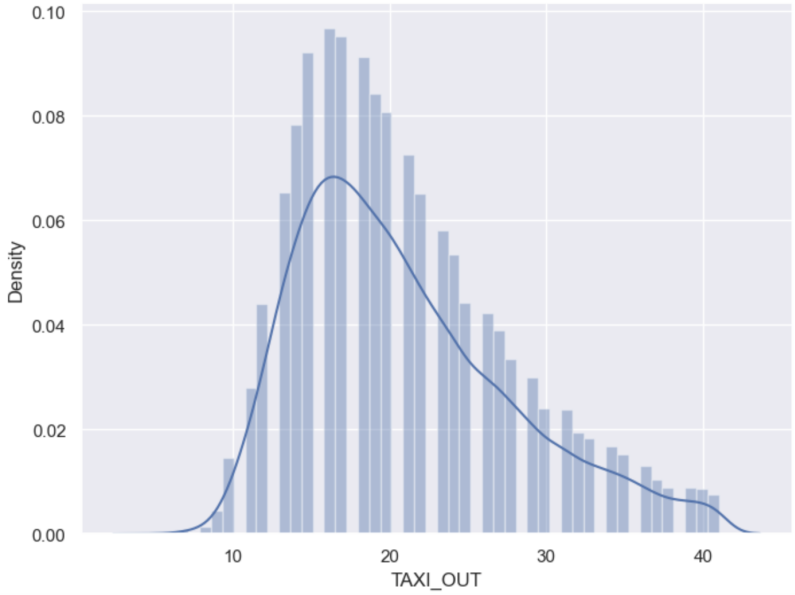
- **TAXI_OUT** - Runway time

Shape: 28820 rows, and 23 columns

Missing Data: 2 missing values in the **Wind** column

Descriptive Statistics

TAXI_OUT Distribution:



- The distribution of the TAXI_OUT variable is slightly right skewed.
- Takes around 10-40 minutes for an aircraft to leave the runway.
- Majority of the distribution is between 15-18 minutes.

Default Model : Multiple Linear Regression

In order to create a full model that includes all features, we must first get rid of all of the categorical variables.

```
Call:
lm(formula = TAXI_OUT ~ ., data = fullModelData)

Residuals:
    Min      1Q  Median      3Q     Max
-18.761  -4.774  -1.137   3.875  23.777

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       4.320e+01  5.238e+00   8.247  < 2e-16 ***
MONTH             4.760e-02  7.944e-03   5.992 2.10e-09 ***
DAY_OF_MONTH     -2.704e-02  4.525e-03  -5.975 2.33e-09 ***
DAY_OF_WEEK       1.082e-01  1.994e-02   5.429 5.73e-08 ***
DEP_DELAY         4.452e-03  1.018e-03   4.371 1.24e-05 ***
CRS_ELAPSED_TIME  2.914e-02  3.340e-03   8.726  < 2e-16 ***
DISTANCE         -3.485e-03  4.481e-04  -7.778 7.61e-15 ***
CRS_DEP_M         2.055e-05  4.080e-04   0.050  0.95983
DEP_TIME_M        8.231e-04  4.114e-04   2.000  0.04546 *
CRS_ARR_M         3.817e-04  1.356e-04   2.815  0.00488 **
Temperature      -2.145e-01  8.496e-03 -25.244  < 2e-16 ***
Dew.Point         1.181e-01  6.263e-03  18.860  < 2e-16 ***
Humidity         -1.289e-02  1.733e-03  -7.437 1.06e-13 ***
Wind.Speed        6.011e-03  9.317e-03   0.645  0.51881
Wind.Gust         6.085e-02  4.588e-03  13.264  < 2e-16 ***
Pressure         -8.334e-01  1.717e-01  -4.855 1.21e-06 ***
sch_dep           1.209e-01  4.289e-03  28.194  < 2e-16 ***
sch_arr           3.584e-02  5.820e-03   6.158 7.48e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.56 on 28800 degrees of freedom
Multiple R-squared:  0.08339,   Adjusted R-squared:  0.08285
F-statistic: 154.1 on 17 and 28800 DF,  p-value: < 2.2e-16
```

This is a very poor prediction model since the R2 score is only about 0.08, and the F-statistic is 154.1. Additionally, the p-values for the features CRS_DEP_M at 0.95983 and for Wind Speed at 0.51881 much greater than the desired 0.05 cut-off in a 95% confidence interval.

Modification

We attempted a linear model with only numerical features on python; however, it returned an extremely small F-statistic (i.e., e-30) with an R2 score of 1.0. This does not make sense since the larger the R2 score, the larger the F-statistic should be too.

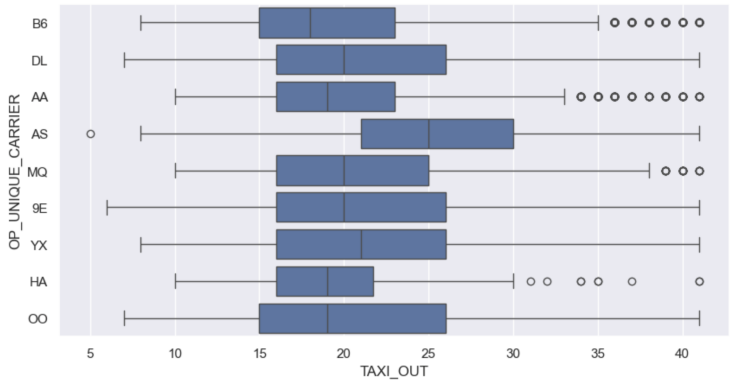
PYTHON

Adding Categorical Features to the Model:

The boxplots for categorical data displayed which unique values had common or unusual averages, and excessive outliers. These details were used to reduce the number of unique values by grouping the insignificant values together. Afterward, our goal was to apply the appropriate encoding methods to low-cardinality and high-cardinality features.

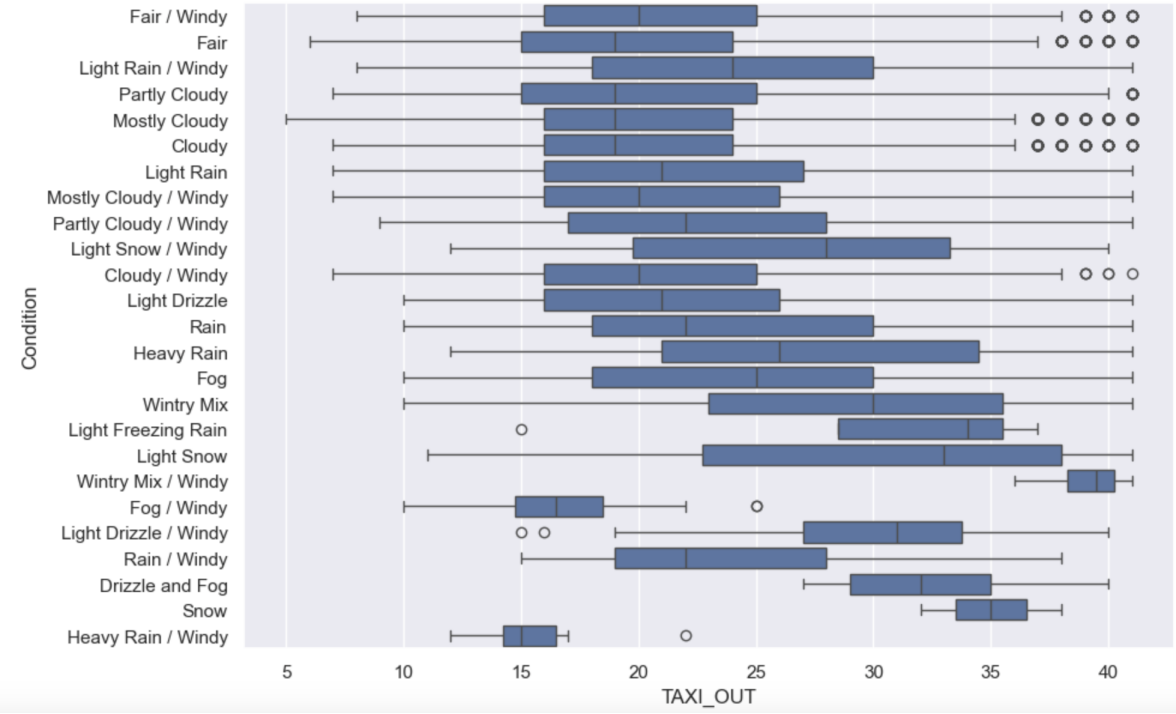
Using python, we avoided one-hot encoding or dummy variables since it could add excessive columns to our dataset, totalling over 2,000 columns. Instead, we ordinally encoded based on ascending mean and target encoded low-cardinality features such as OP_UNIQUE_CARRIER, Wind, and Condition. For Wind, we considered the amount of wind to be more important than the direction (i.e., NW ordinally encoded as 2 since wind blows in two directions, north and west). Since TAIL_NUM and DEST both have high-cardinality (large number of unique values) and no particular order, only target encoding seemed the best option to identify most frequent flight numbers or destinations without increasing columns.

OP_UNIQUE_CARRIER vs TAXI_OUT Boxplot:



We determined that AS, B6, and AA were significant enough to have their own dummy variables since AS had the highest average, and B6 and AA had the most outliers outside of the upper limit. This was also a method we used to reduce the number of unique values per feature.

Condition vs TAXI_OUT Boxplot:



Based on this boxplot, we picked the categories with extremely high or low averages as their own dummy variables (Heavy Rain / Windy, Fog / Windy, Drizzle and Fog, Light Snow, Light Drizzle / Windy, Light Freezing Rain, Snow, Wintry Mix / Windy), and everything else we considered as unremarkable.

Scaling the Data:

After obtaining the distribution plots for the features, now all numerical after encoding, we were able to separate distributions into three main categories: (1) normal or approximately normal, (2) skewed, and (3) other such as bimodal or multimodal.

Based on the distribution each feature had, we standardized or normalized, and also removed outliers as a precaution. Before applying other models to our data, we split the data into training and test data.

Feature Selection Methods:

We based our feature selection on two methods to increase model accuracy:

1. ***Heatmap*** to identify which encoded features were most correlated to the target TAXI_OUT, and chose features that were greater than or equal to 0.1.
2. ***Backward selection*** using the p-value as the metric, dropping the highest p-value one feature at a time, and stopping when all p-values are lower than 0.05.

Alternative Encoding Methods and VIF:

In a separate attempt to improve the model accuracy in python, we only ordinally encoded for low-cardinality categorical features and target encoded for high-cardinality before reducing the model based on the VIF method. We confirmed, once again, that all our models scored very low in accuracy despite deleting features with high correlation to each other.

RESORTING TO R

At this point, we repeatedly witnessed that no matter what encoding methods we used, whether we scaled or dealt with outliers, or applied VIF, we still ended up with very poor results regarding the accuracy score for any of the models we tried. We were limited to multiple lines of code in python and complicated methods of feature selection to reduce our model. As a result, we decided to transform our categorical variables in R.

When we created a model based solely on numerical data in python, the results differed from in R, so our goal when switching to R was to both create a better model with more aptly built-in functions for regression modeling and double checking that results were consistent.

Encoding Methods:

Dummy Variables and Step Function Based on AIC:

In R, applying only dummy variables for OP_UNIQUE_CARRIER and Condition based on the same boxplot method in python improved our model by increasing the R-squared to 0.1146 and the F-statistic to 119.4. We omitted the rest of the categorical variables for this model since they showed no significant changes in mean from the boxplot method.

The step function minimizes the AIC of the model, which in turn minimizes the SSE, based on the features that are added to or eliminated from the model. This decreased the R-squared to 0.1145, but improved our model from the dummy variable encoding by increasing our F-statistic to 139.3.

Building off Previous Models:

At this point, we only applied target encoding from the start similar to a version we did in python, and from there we added several methods of feature selection to improve our model accuracy. Each new model was built off the last one.

Only ***target encoding*** for all the categorical features improved the R-squared score to 0.2017 and the F-statistics to 296.9.

We got significant changes when employing the ***VIF method*** here with the target encoding model as opposed to with the dummy variable model above. We deleted CRS_ELAPSED_TIME and DEP_TIME_M and got a slightly lower R-squared of 0.2016 and F-statistic of 326.4.

We applied the ***step function based on the AIC*** again from the newly built VIF model and got an R-squared of 0.2016 and an F-statistic of 343.6.

We reduced the model with the ***backward selection*** method by dropping features with the highest p-value one at a time from the AIC model, and got an R-squared of 0.1995 and an F-statistic of 402.8.

Results

Python Comparison Table

R2 Score Based on Target Encoding Only:

	OLS	Ridge	Lasso	RidgeCV	LassoCV	Quadratic
Full Model	0.196251	0.196251	0.161812	0.196251	0.186389	0.240449
Heatmap Correlation	0.175587	0.175587	0.143411	0.175587	0.175572	0.179989
Backward Selection	0.196085	NaN	NaN	0.196085	0.194171	0.228661

```
1 scores.max(skipna =True)
```

```
OLS          0.196251
Ridge        0.196251
Lasso        0.161812
RidgeCV      0.196251
LassoCV      0.194171
Quadratic    0.240449
dtype: float64
```

Alternative Model Building Processes:

Scores from either of these processes *ordinal and/or target encoding, and VIF* or *scaling and removing outliers* have lower accuracy lower than if we only applied target encoding.

R Comparison Table

	Dummy Variables	Step Function (AIC)	Target Encoding	VIF	Step Function (AIC)	Backward Selection
OLS	**F:** 119.4 **R2:** 0.1146	**F:** 139.3 **R2:** 0.1145	**F:** 296.9 **R2:** 0.2017	**F:** 326.4 **R2:** 0.2016	**F:** 343.6 **R2:** 0.2015	**F:** 402.8 **R2:** 0.1995

Backward Selection Linear Model Better than Default Linear Model

Despite the higher R-squared scores obtained from only target encoding in python, most of it might be due to overfitting. As a result, the best model we were able to obtain is a ***linear model from R*** that was target encoded, in addition to feature selection methods such as the VIF method, the step function based on the AIC metric, and the backward selection based on the p-value metric.

Analysis

We transitioned from using dummy variables to target encoding by calculating the mean TAXI_OUT based on grouping each categorical feature, which yielded better results.

The majority of our categorical features have a large number of unique values and are nominal data. Since there is no real order to any of the features, except for Condition, ordinally encoding would create a false sense of order and using dummy variables would add unnecessary columns to the dataframe. This could contribute error to the model through overfitting by considering insignificant categorical values, adding penalty through the Lasso or Ridge models, or complicating a model. Deciding how to encode the categorical variables was the biggest turning point for our model, as our R-squared and F-statistic nearly doubled after making this change.

However, it was still a full model after transforming the categorical variables. Thus, feature selection was necessary in order to remove multicollinearity between highly correlated features, and large values of AIC and p-values through the step function and backward selection respectively. That is, we eliminated features where the coefficients of the model failed to be significant with an alpha of less than 0.05. These methods considered which features affected the TAXI_OUT similarly, resulting in only keeping features that were both unique and significant in predicting the target.

Our final model had a total of 16 predictors, and the predictors that contributed the most to the prediction were TAIL_NUM (airflight number) with a coefficient of 0.857 and Pressure with a coefficient of -0.832. This tells us that the plane used contributes the most to how long it takes to depart the runway, taking into account some planes take longer than others to get off the runway. Pressure has a negative slope, meaning the higher the pressure becomes the lower the time it takes to get off the runway. This makes sense intuitively because low pressure causes wind and bad weather.

```
Call:
lm(formula = TAXI_OUT ~ ., data = pValueTrain3)

Residuals:
     Min       1Q   Median       3Q      Max
-19.4968  -4.3476  -0.8754   3.5890  23.7890

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.383e+01  5.333e+00  -2.593 0.009508 **
MONTH         3.532e-02  7.791e-03   4.533 5.83e-06 ***
DAY_OF_MONTH -2.139e-02  4.475e-03  -4.780 1.76e-06 ***
DAY_OF_WEEK   1.027e-01  1.976e-02   5.197 2.04e-07 ***
TAIL_NUM      8.567e-01  1.681e-02  50.976  < 2e-16 ***
DEST          6.233e-01  3.477e-02  17.929  < 2e-16 ***
DEP_DELAY     3.334e-03  9.925e-04   3.359 0.000783 ***
DISTANCE     -1.399e-04  4.599e-05  -3.043 0.002346 **
CRS_DEP_M     1.097e-03  1.496e-04   7.338 2.24e-13 ***
Temperature  -1.476e-01  8.467e-03 -17.430  < 2e-16 ***
Dew.Point     5.890e-02  6.140e-03   9.592  < 2e-16 ***
Wind          6.125e-01  4.997e-02  12.259  < 2e-16 ***
Wind.Speed    2.874e-02  7.432e-03   3.867 0.000110 ***
Pressure     -8.321e-01  1.690e-01  -4.923 8.56e-07 ***
Condition     6.989e-01  3.434e-02  20.354  < 2e-16 ***
sch_dep       1.002e-01  4.099e-03  24.442  < 2e-16 ***
sch_arr       4.306e-02  5.398e-03   7.977 1.56e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.154 on 25863 degrees of freedom
Multiple R-squared:  0.1995,    Adjusted R-squared:  0.199
F-statistic: 402.8 on 16 and 25863 DF,  p-value: < 2.2e-16
```

Our final model is still very far from accurately predicting runway time, with only an R-squared of 0.1995. Alternatively, we could have created a better linear model, or used other types of regression models. It is also plausible the amount of time spent before departing a runway is truly impossible to predict, as there are too many variables affecting the amount of time on the runway.