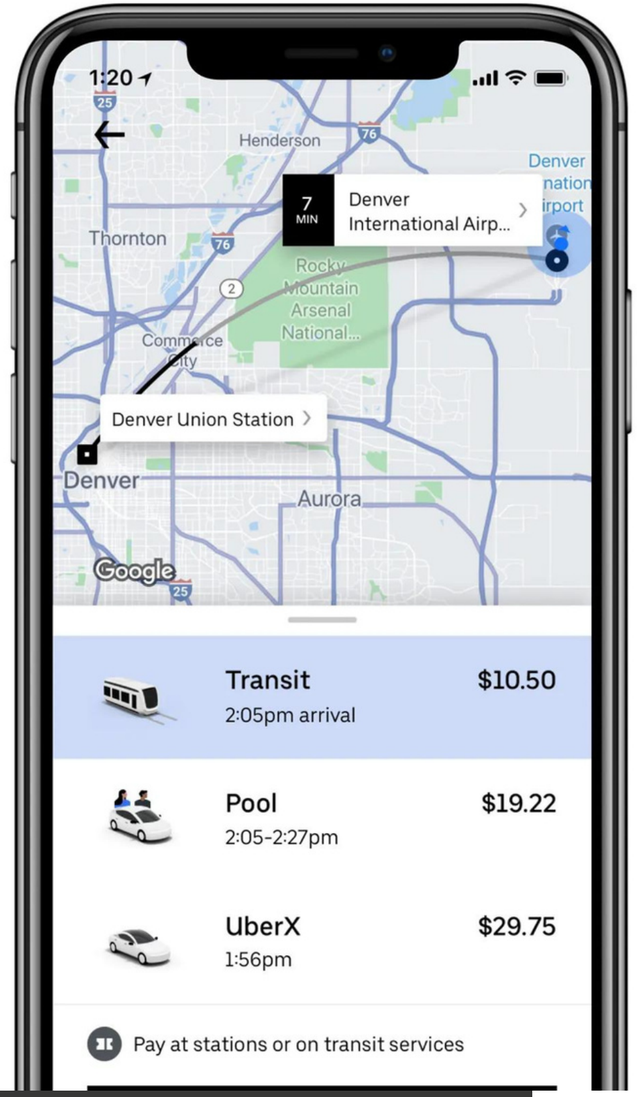
UBER & LYFT PRICE ESTIMATOR

TEAM: JAMES LEE | COLE KINCAID | ZACHARY ASKINS

RedID: 820655947 822602112 822719840

BACKGROUND

Living in a society where Ubers and Lyfts have taken over the taxi industry, we are always confronted with the problem of how much it would cost. Starting from this initial thought, we began to work on creating a model that could provide an accurate solution.



PROJECT OBJECTIVE

Our product aims to give users accurate estimate prices on their Uber or Lyft ride.

TASKS

- select data set
- transform/massage data set
- run basic linear regression
- analyze outputs and metrics
- determine possible optimizations for model
- run optimized linear regression
- analyze optimized model outputs and metrics
- compare results with other model types

DATASET

Obtaining data:

We found our dataset on Kaggle which was created by Ravi Munde and Karan Barai. Because Uber and Lyft do not provide public data on rides, all information was collected (every 5 minutes) through api-queries in real time. The data is was collected during a week span in November of 2018 within the city of Boston, MA.

Formatting Data:

While importing our data from the .csv file, we noticed that some price values were empty. Using panda's dropna dataframe function, we were able to remove all missing values within the "prices" column.

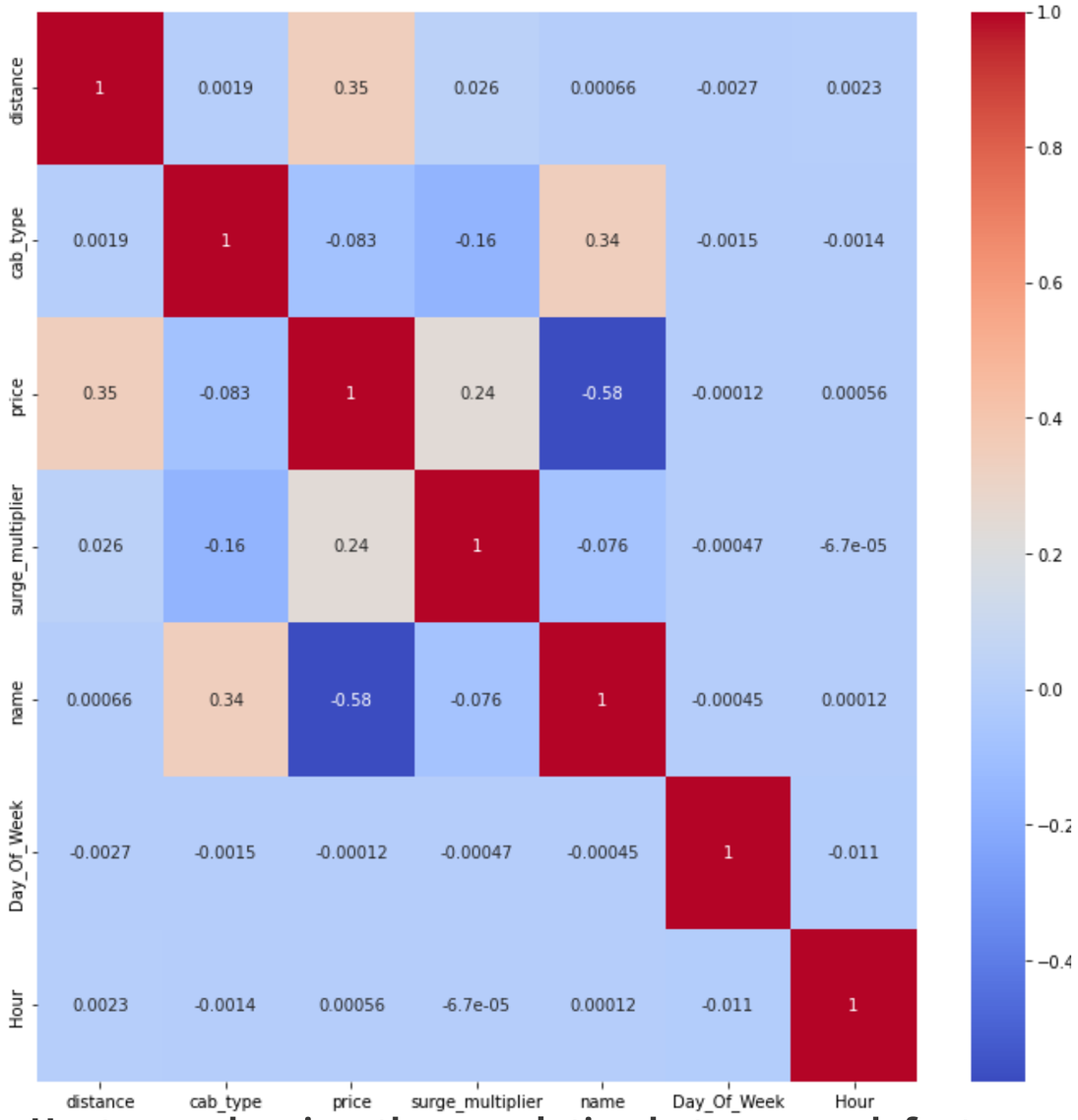
DATASET

Features (X):

- Distance
- Cab Type (Uber or Lyft)
- Day of the Week
- Hour
- Surge Multiplier
- Name (UberX, LyftXL, etc.)

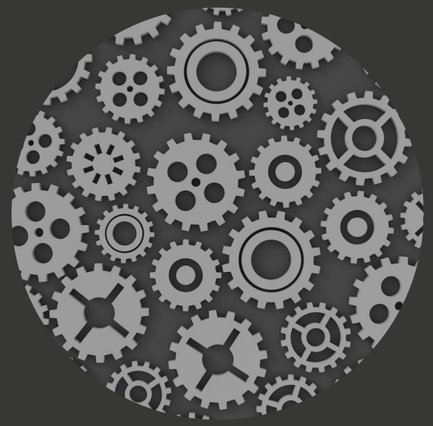
Output (Y):

- Price of Ride



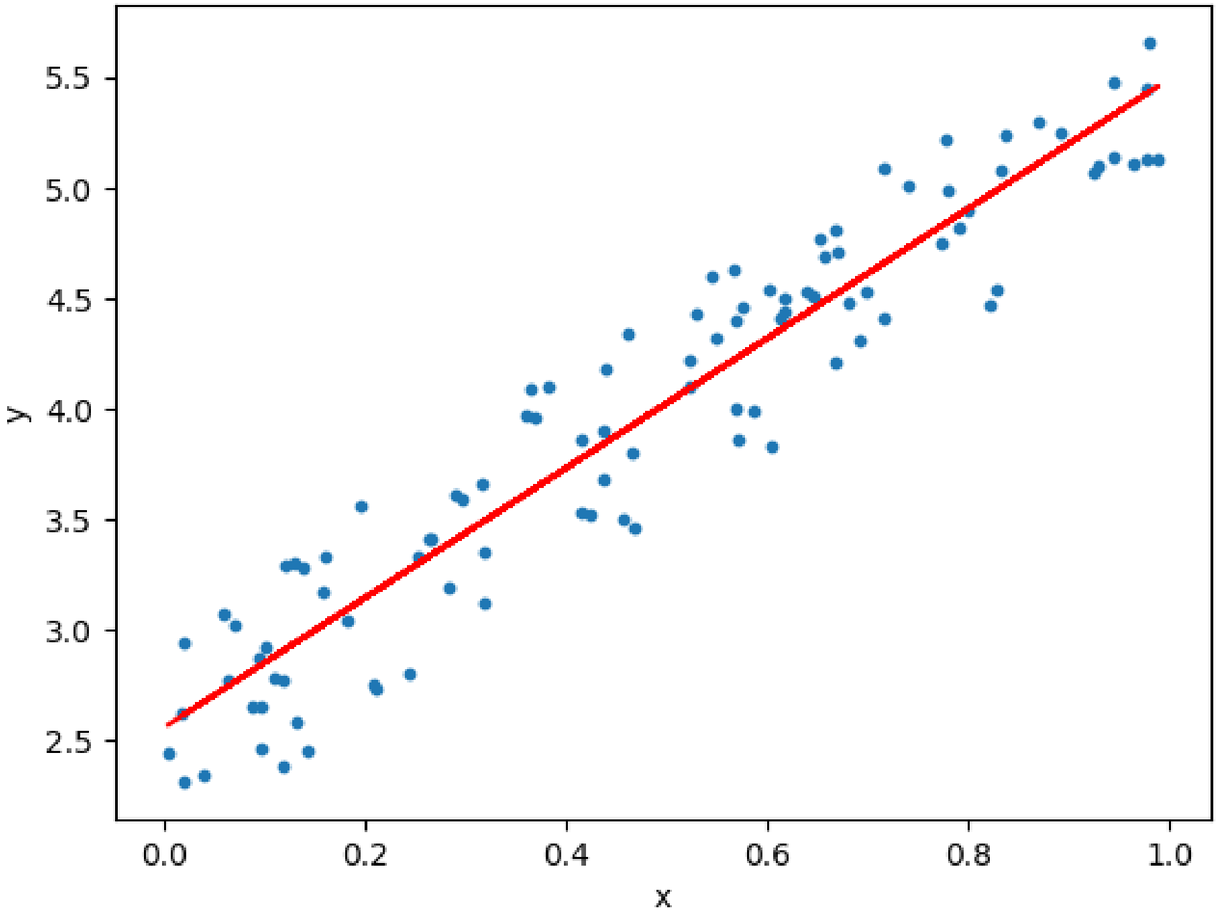
Heat map showing the correlation between each feature

METHODOLOGY



- Training Algorithm Choice
- Model Customization
- Other algorithms

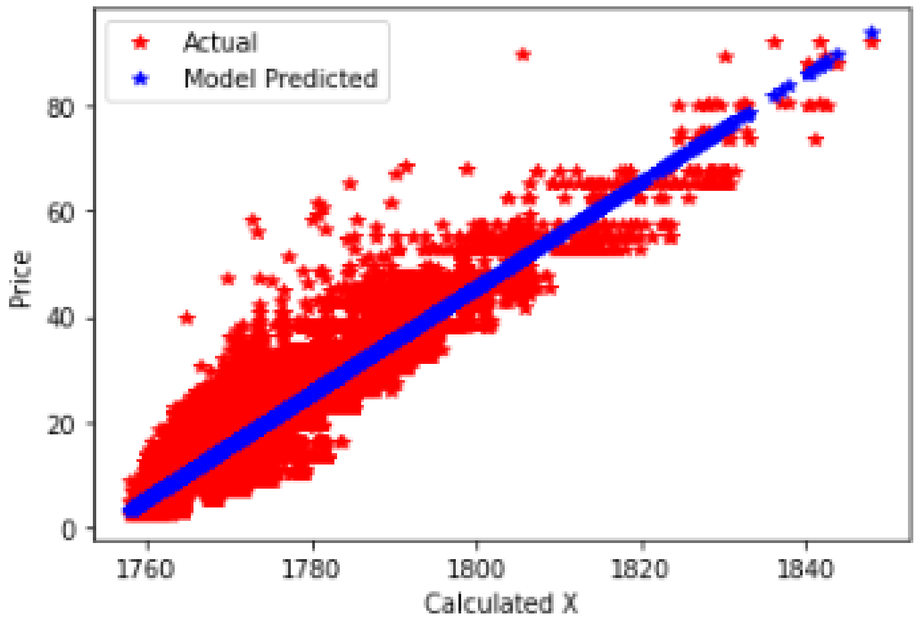
TRAINING ALGORITHM CHOICE



- The algorithm we chose to use was linear regression.
- Is effective and efficient for computing the model.
- Was most known by the group members due to time spent in class.
- Programming was easier to implement due to the support from the SKlearn libraries.

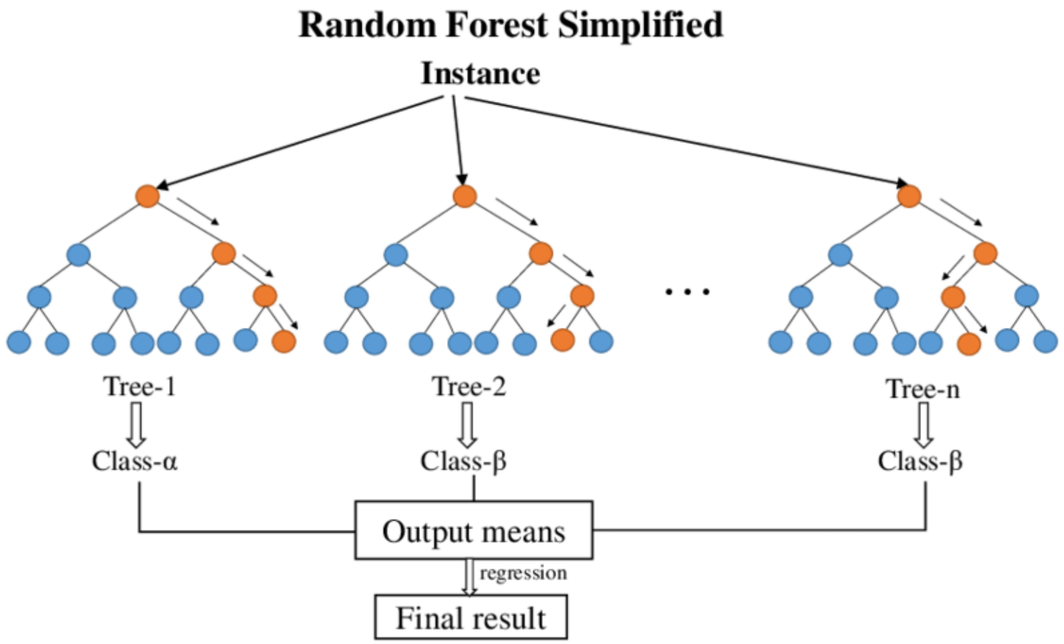
MODEL CUSTOMIZATION

With Polynomial Features



- Customization was needed as the default linear regression produced a low score
- SKLearn library for linear regression does not have methods to change hyper parameters
- We used Polynomial Features which is an SKlearn preprocessing function
- The addition of more the larger input size allowed for a more accurate prediction model
- Programming was easier to implement due to the support from the SKlearn libraries.

OTHER ALGORITHMS



We explored other algorithms that could have been used: Random Forest Classifier and Stochastic Gradient Descent

Random Forest Classifier uses decision tree classifiers for fitting while Stochastic Gradient descent uses smaller subsets of training data

Both of these algorithms produced worse results overall compared to linear regression

COMBINING FEATURES

```
df['combined'] = df['name'] + "_" + df['surge_multiplier'].astype(str)
df['total'] = df['cab_type'] + "_" + df['distance'].astype(str)

leCombined = preprocessing.LabelEncoder()
leCombined.fit(pd.unique(df['combined']))

letotal = preprocessing.LabelEncoder()
letotal.fit(pd.unique(df['total']))

df['combined'] = leName.transform(df['combined'])
df['total'] = letotal.transform(df['total'])
```

```
Mean squared error: 5.151414
Root Mean Squared Error: 0.008985860703171146
Training Score: 0.94
Test set score: 0.94
```

PROCESS

- We start off by looking at the heat map created by our dataset.
- I found that while combining two strong features are good, it was even better with one strong and one weaker feature.
- We then treat these combined features like normal features and fit/transform them into the dataframe.

EXPERIMENT SETTING

Training/Testing Dataset

Evaluation Metrics

Baseline Methods

Dataset Settings

Our dataset contained 693071 different entries, which were mainly found in 2018 in the greater Boston area.

For our training and testing dataset we used train_test_split with a default split of .25 for the testing set. We tested the model using a .1 and a .3 split for the testing set, however, we received the same scores regardless of which split was used. Since we saw similar scores within the different splits, we decided to use the default settings.

TRAINING/TESTING DATASET

Evaluation Metrics

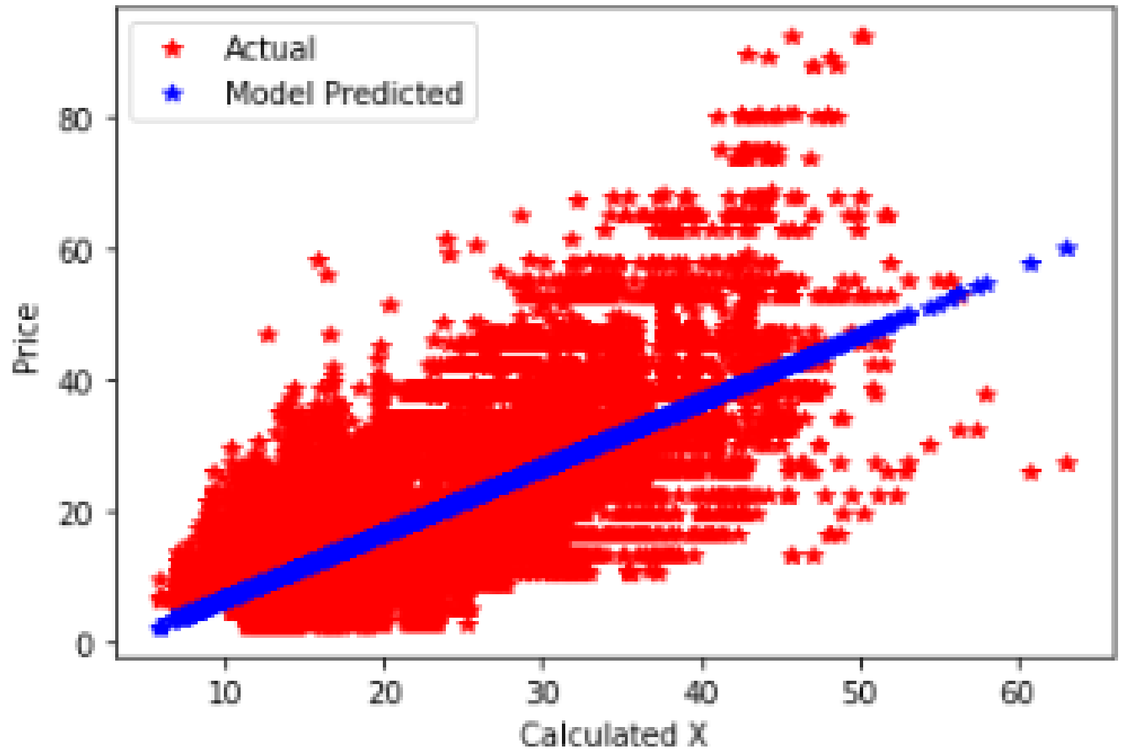
We used:

- Mean Squared Error
- Root Mean Squared Error
- Linear Regression Score (Accuracy)
- K-Fold

EVALUATION METRICS

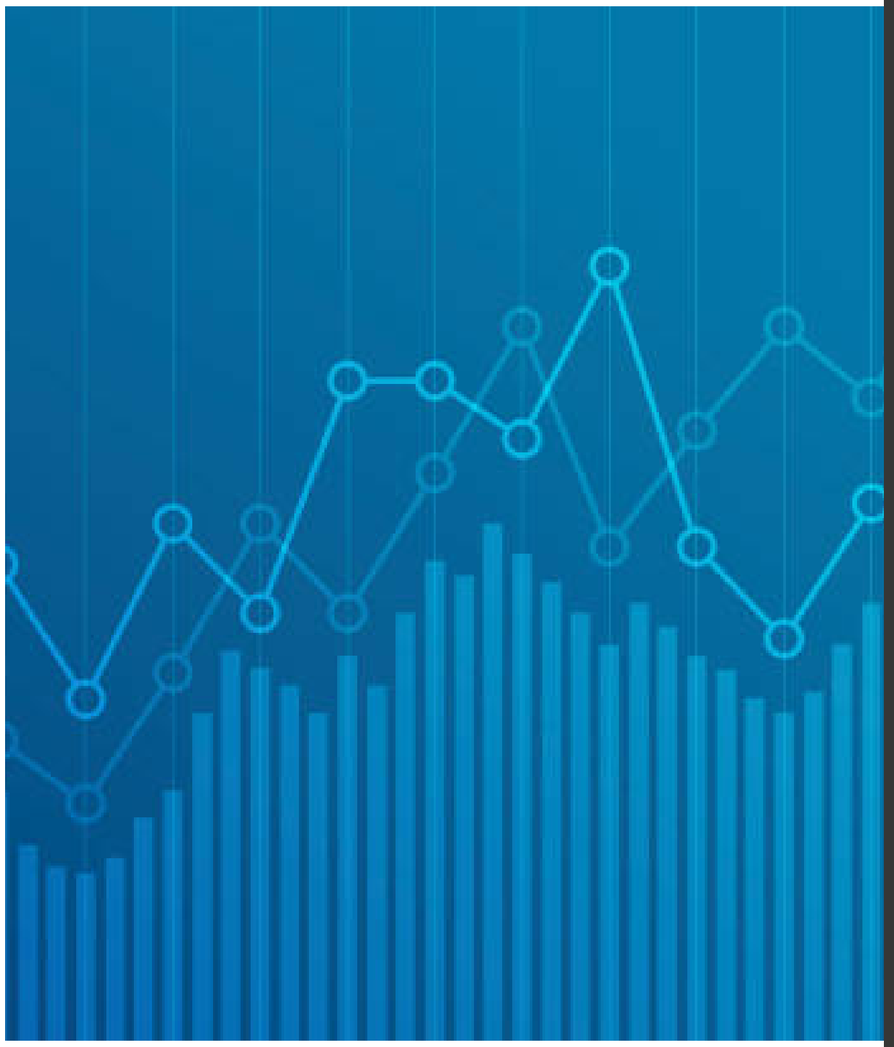
Baseline Methods

- Not massaging the data and just using Linear Regression.



BASELINE METHODS

RESULTS

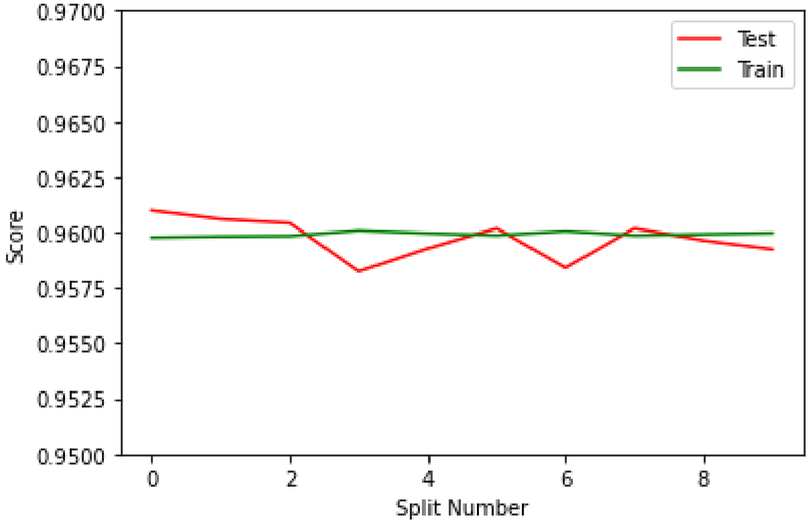


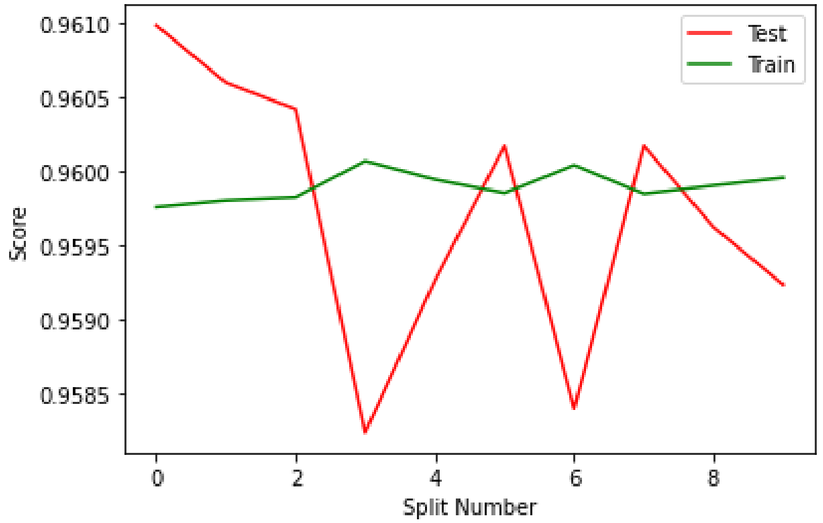
The numerical and qualitative results from our model

NUMERICAL RESULTS

```
Mean squared error: 3.490690
Root mean squared error: 0.004678
Training Score: 0.96
Test set score: 0.96
```

Score running Linear Regression with polynomial features





Kfold running Linear Regression with polynomial features

QUALITATIVE RESULTS

Successes

- Discovered and implemented the polynomial features into the model which produced better accuracy overall, increased the degree gradually until accuracy was sufficient
- Realized that the combination of features lead to a more direct impact on the output. This was done due to two of the features not having a large influence. Therefore, we tried combining them together.
- Used Kfold for the validation and achieved the optimal results. Provided with a robust validation due to nature of the Kfold.

Failures

- We had a lot of initial issues trying to determine which features impacted the linear regression model. Our solution was creating a heat map that allowed us to see which features had a bigger impact on price.
- Using a combination of features we had a short increase in performance, however, the results were minuscule compared to the Polynomial Features.
- We tried combining the features and then using Polynomial Features which had higher performance scores for the first 5 degrees, but on the 6th degree the performance started to decrease.

CONCLUSION

MAJOR FINDINGS

- We found that although combining features can increase performance, it is heavily dependent on the combinations of the features. Many performed the same as our original model, and some were even worse. The select few that were good were based off of a heat map that displayed correlation figures.
- The most accurate model was running a Linear Regression by fitting our Polynomial Features data along with our price labels.
- While our accuracy was high for predicting the price during this specific week of 2018 it may not be accurate looking at future timelines where the prices may have changed drastically.

THANK YOU