UNIVERISTY OF MICHIGAN

SI 618 PROJECT 2

US Domestic Airline Consumer Airfare: an Inter-City Analysis

Author:
Haoquan Zhou

December 3, 2022

Contents

List of Figures

1 Introduction and Motivation

Traveling by airplane is one of the most common modes of transportation for the US dwellers. Compared with countries like Japan and China, which have a high population density, the distribution of the population in the US is quite sparse and thus not suitable for developing a high speed railway network [1]. However, the US also enjoys a huge land area, spanning from the Pacific Ocean to the Atlantic Ocean, which makes long range transportation a necessity for the US citizens. Throughout the history of the US, airplanes have thus been playing the essential role of long distance transportation.

Due to the popularity of air transportation, airfare has become one of the hottest problems discussed by the public. Understanding the trend of airfare and estimating reasonable prices are thus become important for money-saving. In this project, the US domestic airline consumer airfare will be analyzed closely based on an inter-city view. The prices of flights between US major cities will be collected and inspected. In particular, there are three overarching questions covered by this project:

1. What is the overall trend of US domestic airfare after 1990s?
2. What factors determine the airfare between two cities?
3. Can we predict the airfare based on given information?

Answering these questions will help us understand the inner logic of airfare determination, which may further lead to better decision makings when purchasing flights.

2 Data Source

This project will all based on the official data provided by the U.S. Department of Transportation [2]. It is a csv data set covers the 1,000 largest city-pair markets in the 48 contiguous states, recording the commercial aviation market status between each of these city pairs. To be more specific, it contains 100K records indicating the market situation for each city pair quarterly from 1995 to present. Among all 26 data fields, there are several fields that are particularly related to this project's topic:

1. `Year, quarter (int)`: These columns show the time index of each record, which is essential to analyze the trend of the airfare.
2. `city1, city2 (str)`: These columns indicate the city pair the record describes.
3. `nsmiles, passengers, large_ms, lf_ms (float)`: These fields indicates the distance between the cities, the number of passengers per day, the largest market share percentage, and the market share percentage of the lower price stakeholder, respectively. These are potential factors that can affect the airfare.
4. `fare (float)`: This field contains the average airfare.

Based on the information above, we are going to conduct our project and try to discover some useful relationships.

3 Data Manipulation Methods

3.1 Overall Description

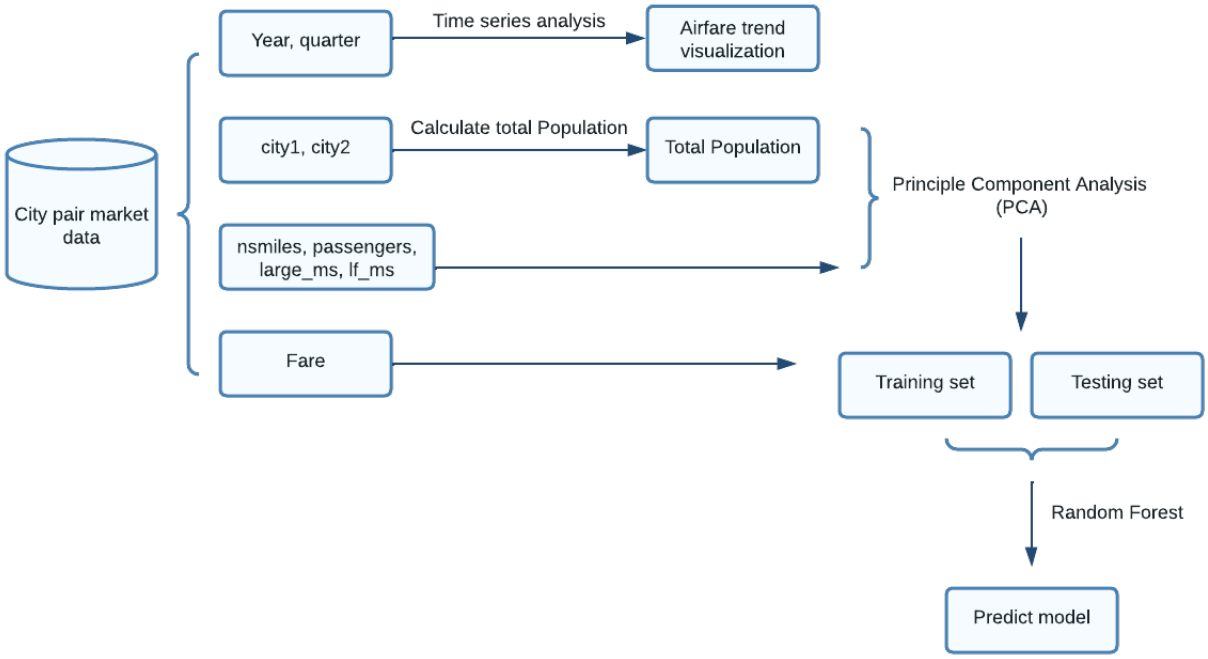

Figure 1: Data Manipulation Flow

The overall data manipulation flow is shown in Figure. 1. The first question will require the usage of `Year` and `quarter` to perform a time series analysis from 1990 to 2022 in order to show the airfare trend visualization. Meanwhile, to figure out what factors can influence the airfare, a Principle Component Analysis (PCA) is performed to potential dominating factors. This step is also meaningful for later part, which is predicting the airfare based on all the variables above. Note that since city pair is a categorical data, we need to transform it to numerical data type. And total population of two cities are selected in this project. As for the prediction part, a random forest model is selected to pursue a better performance.

3.2 Question 1: Airfare Trend

After load the data set, we group the records according to the `year` and `quarter`. The grouped data is then aggregated by calculating the average air fare across all the city pairs. This part of the data is quite clean and well-maintained, which save me the time to clean the missing data.

```
quarter_trend = df.groupby(by = ['Year',
   'quarter'])[['fare']].agg(np.average)
quarter_trend = quarter_trend.reset_index()
```

To construct a column more suitable for plotting, we create a column called `time` and convert the `year` and `quarter` to `pd.DatetimeIndex` data type. This data type is easy to manage when it comes to axis constructing.

```
quarter_trend['time'] = quarter_trend.apply(lambda x:
   pd.to_datetime(str(int(x[0])) + '-' + str(int(x[1]))), axis = 1)
```

During the manipulation, I suddenly realize that the data may show a some degree of seasonality. Thus, it becomes a little more difficult to sketch the actual annual trend of airfare. I apply the de-seasonality method, which requires an Ordinary Least Square fit on the categorical data. A new column named `deseasonalized` is then created for later plots.

```
model = smf.ols('fare ~ C(quarter)', data = quarter_trend).fit()
quarter_trend['deseasonalized'] = model.resid +
   np.mean(quarter_trend['fare'])
```

The data is now able to purely reflect the overall trend of the airfare.

3.3 Question 2: Factors Analysis

As introduced before, we need first ensure that all the factors are numerical, which will be required for later analysis. In particular, we need to change the city pair to a related value which can reflect the relationship between cities and airfare.

Under most cases, changing categorical data to numerical data can be implemented using one hot encoding. However, considering that the number of cities involved in this data set is more than 1K. One hot encoding all these cities will add a huge amount of columns compared to the original data set, which is definitely not desirable.

Thus, a value between city pair and fair should be discovered and replace the city. By intelligent guess, I tried to consider the sum of populations of city pairs. It is natural to assume that cities with larger population will have more passengers, which may further lead to an influence in the airfare. To verify this, a correlation analysis is conducted.

However, when calculating the sum of populations, I found that the city names are not consistent. To be more specific, since the are cities sharing the same airport (like Dallas/Fort Worth), the population is hard to esitimate. In order to solve this problem, regular expression is used to retrieve the key. Here we just consider the last city in the city list for simplicity. After all, airport-sharing situation is not common, which should not cause large difference in the relation analysis.

```
df['city1'] = df['city1'].apply(lambda x: re.findall(pattern = r'[a-zA-Z
   ]+, [A-Z]{2}', string = x)[0])
df['city2'] = df['city2'].apply(lambda x: re.findall(pattern = r'[a-zA-Z
   ]+, [A-Z]{2}', string = x)[0])
```

We then merge the population table

```
pop1 = pd.merge(left = df, right = pop_table, left_on = 'city1',
   right_on = 'City', how = "left")
pop2 = pd.merge(left = pop1, right = pop_table, left_on = 'city2',
   right_on = 'City', how = "left")
data = pop2[['quarter', 'city1', 'city2', 'nsmiles', 'passengers',
   'fare', 'large_ms', 'lf_ms', 'Population_x',
   'Population_y']].dropna(axis = 0)
```

Note that the merge process may yield some null value, just drop them. All the manipulation above results in a 10% loss in data and gives around 90K valid records.

Now, a plot can be used to verify the relationship between population and passengers.

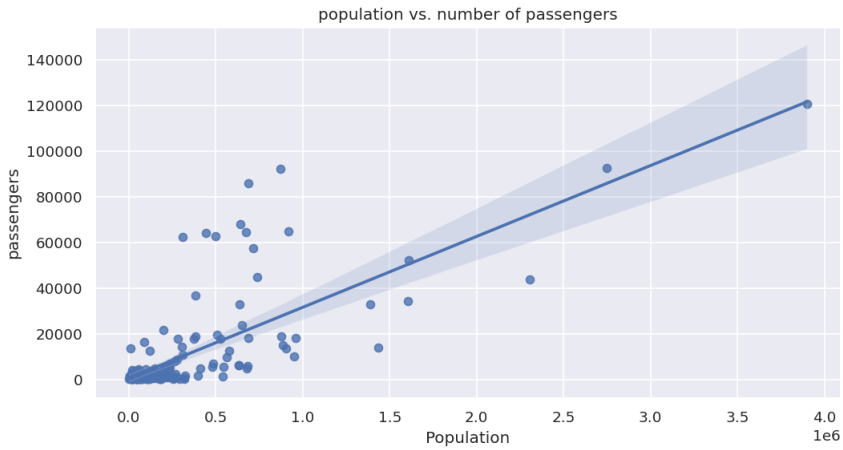

Figure 2: Relation between population and passenger

```
plot = sns.regplot(data = passenger_pop, y = 'passengers', x =
    'Population').set(title = 'population vs. number of passengers')
```

As shown in the Figure. 2, there is a clear positive relationship between city population and number of passengers. Also, the correlation coefficient is around 0.70, which proves the relation as well. By now, we have successfully transform the city pair to a numerical value. Although the number of passengers do not necessary lead to a significant relation with airfare, we create a useful variable for further prediction process.

With respect to other factors, the data are clean enough for immediate use. We then apply the Principle Component Analysis.

```
import sklearn.decomposition as skd
pca_model = skd.PCA().fit(data.iloc[:, [2, 3, 5, 6, 9]])
```

3.4 Question 3: Predict Airfare

The data manipulation in this part just follow the classic procedures of machine learning. We first select the features including `nsmiles`, `passengers`, `large_ms`, `lf_ms`, `sum_population`. Then divide the whole data set into training set and testing set, with a size percentage of 4 to 1.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, [2, 3,
    5, 6, 9]], data['fare'], test_size = 0.2, random_state = 0)
```

We then choose a random forest regressor with 5-fold validation on the forest depth. The potential candidates ranges from 20 to 50, with step length 5.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
rf = RandomForestRegressor(n_estimators = 100, random_state = 0, n_jobs
    = -1)
param_grid = {'max_depth': [ 20, 25, 30, 35, 40, 45, 50]}
grid_search = GridSearchCV(rf, param_grid, cv = 5, verbose = 2)
```

It then comes to actually train the model and test it on the test set.

```
grid_search.fit(X_train, y_train)
y_pred = grid_search.predict(X_test)
```

There is no challenge for this part due to its simplicity.

4 Analysis and Visualization

4.1 Airfare Trend

To conduct a time series analysis, we first use the `plotnine` library to create a line plot. Usually this plot will give us a basic idea about the trend.

```
(ggplot(quarter_trend, aes('time', 'fare'))) + geom_point() +
   geom_line() + ggtitle("Airfare Price") + theme(figure_size = (10,5))
```

The result is shown in Figure. 3 After observing the plot, we discover that the airfare trend

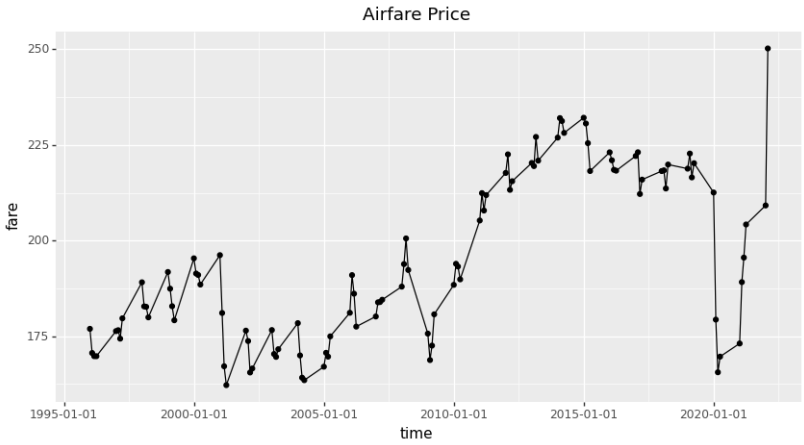

Figure 3: Airfare Trend

moves up and down regularly. However, we want to focus more on the pure trend of the airfare over the years, instead of the seasonality. Thus, a quarter trend plot is helpful to reflect the existence of seasonality.

```
ggplot(quarter_trend, aes('quarter', 'fare', group = 'Year')) +
   ggtitle("Average airfair prices by quarter") + geom_line(aes(color =
   'Year')) + geom_point(aes(color = 'Year')) + theme(figure_size =
   (10,5))
```

The result is shown in Figure. 4 From the figure, we can spot that in most of the years, the airfare shows a slightly decreasing trend, which means the airfare tend to be higher in the first two quarters and lower in the third and fourth quarter. Although the airfare seasonality is not significant, getting rid of it may still be meaningful and necessary.

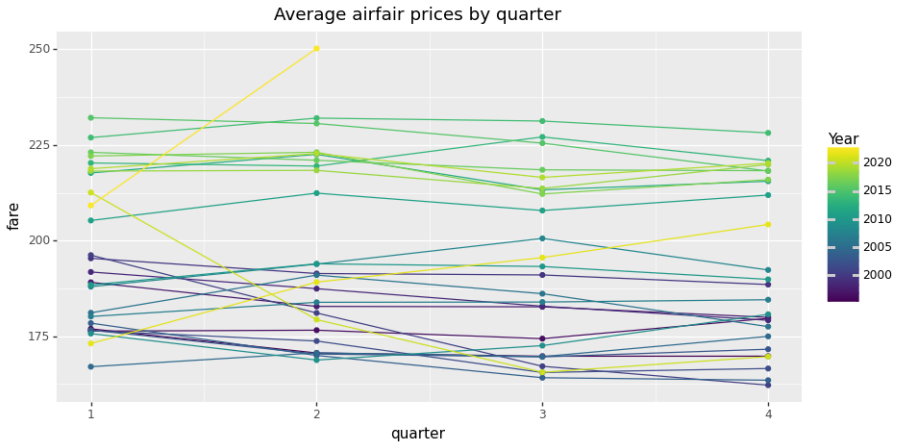

Figure 4: Seasonality Plot

```
model = smf.ols('fare ~ C(quarter)', data = quarter_trend).fit()
quarter_trend['deseasonalized'] = model.resid +
   np.mean(quarter_trend['fare'])
ggplot(quarter_trend, aes('time', 'deseasonalized')) + geom_point() +
   geom_line() + ggtitle("Deseasonified Airfare Price") +
   theme(figure_size = (10, 5))
```

This gives a deseasonalized plot shown in Figure. 5 Now, we can conduct analysis on the

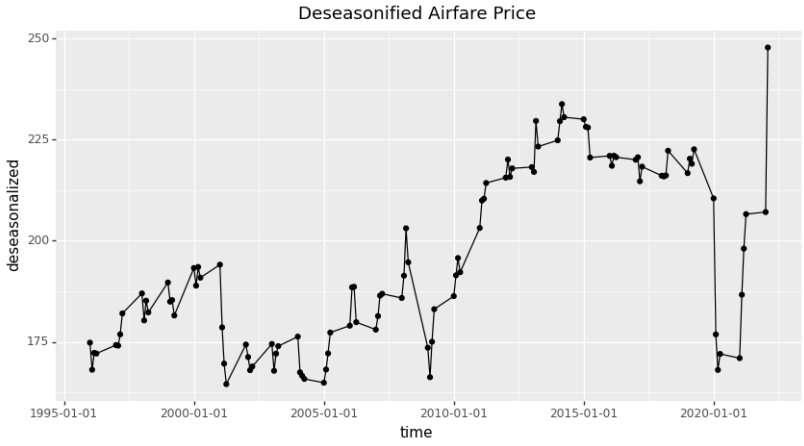

Figure 5: Deseasonalized Plot

pure trend of airfare. By carefully analyzing the plot, we can discover that:

1. The overall trend of airfare is increasing since 1990s, the average airfare increases by around 30% in the 2010s compared to that of 1995.

2. Apart from the overall increasing trend. There are three periods when the airfare drops significantly, which are around 2000, 2008 and 2020. The drop in 2008 and 2020 are due to the weak aviation market caused by financial crisis and COVID pandemic respectively. The drop around 2000-2001 might be caused by the terror attacks on New York and Washington, D.C..

3. It is foreseeable that the airfare will at least maintain the high level in the next couple of years.

4.2 Factors Analysis

In this part, the Principle Component Analysis (PCA) will be used to evaluate the influence of each factor on the airfare. Ideally, the PCA will return 5 eigenvectors indicating the relationship and relative strength of the factors. The PCA can be done by

```
import sklearn.decomposition as skd
pca_model = skd.PCA().fit(data.iloc[:, [2, 3, 5, 6, 9]])
```

We then query the eigenvectors

```
pca_model.components_
```

The eigenvectors are listed in Table. 1 From the table, we discover that each eigenvector

	PC1	PC2	PC3	PC4	PC5
Distance	2.60e-05	-3.79e-02	**9.99e-01**	1.56e-04	1.16e-04
Number of Passengers	3.19e-04	**9.99e-01**	3.79e-02	1.83e-05	1.07e-05
Largest Market Share	-3.27e-08	-9.23e-06	-1.52e-04	2.53e-01	**9.67e-01**
Lowest Price Market Share	-1.59e-08	-1.03e-05	-1.21e-04	**9.67e-01**	-2.53e-01
Sum of Population	**9.99e-01**	-3.17e-04	-3.81e-05	1.37e-08	2.11e-08

Table 1: PCA Table

is significantly dominated by one of the five factors, which are marked in bold. This result shows that each of the components have equal influence on the airfare, and the relation between each factor pair is not significant.

We then plot the variance explained by each eigenvector. Note that we use the logarithm transformation on the y-axis to keep it readable.

```
import matplotlib.pyplot as plt
plt.plot(range(1,6),np.log(pca_model.explained_variance_),'b-o')
plt.xlabel('Principal Component')
plt.xticks(range(1,6))
plt.ylabel('Log of Explained Variance')
plt.title('Scree Plot')
```

The plot result is shown in Figure. 6 We can see that the former three eigenvectors explain

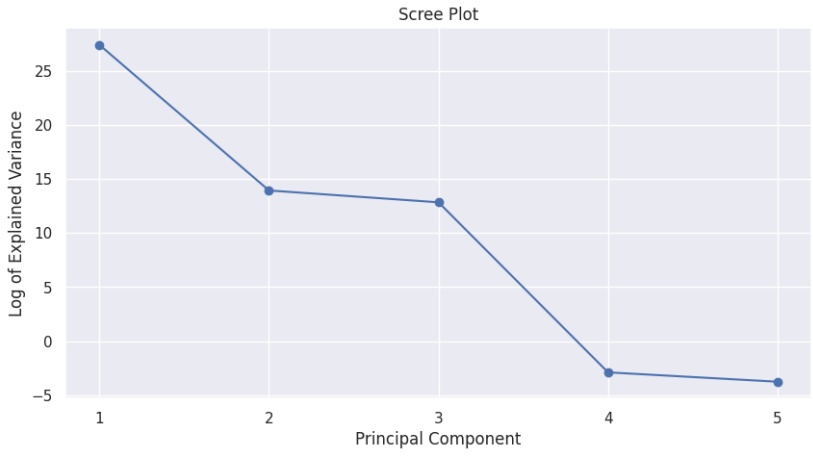

Figure 6: Variance Explained

most of the variance, which indicates that the Distance, Number of Passengers, and Sum of Population play relatively important role in deciding the airfare. This information will not only explain the inner logic of how the airlines come up with their airfare, but also give us a chance to predict the airfare based on the given information.

4.3 Predict Airfare

Random Forest Model is a powerful model in both classification and regression tasks. It is a decision tree based algorithm which utilize multiple decision trees with different features as discriminators. The result is derived by voting within trees and take the majority vote. However, this model also has shortcomings. Since the number of features cannot be controlled, it is time consuming for Random Forest to calculate the result if the number of features are too large.

In our project, we will compare two Random Forest Models. The first one will use all five features listed in Table. 1. While the second one will only use the significant features indicated by the PCA analysis, which are Distance, Number of Passengers, and Sum of Population. We will also use a 5-fold validation technique to find a suitable tree depth for models above.

4.3.1 Model 1: Full Features Utilized

To construct, train, as well as test the model, we use `SkLearn` library

```
# divide the data into training and testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, [2, 3,
    5, 6, 9]], data['fare'], test_size = 0.2, random_state = 0)

# build a random forest regressor with 5 fold cross validation on depth
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
rf = RandomForestRegressor(n_estimators = 100, random_state = 0, n_jobs
    = -1)
param_grid = {'max_depth': [ 20, 25, 30, 35, 40, 45, 50]}
grid_search = GridSearchCV(rf, param_grid, cv = 5, verbose = 2)
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)

# predict the test data
y_pred = grid_search.predict(X_test)
```

We then consider the error and performance of the model

```
# calculate the mean squared error
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)

# for each test point, calculate the percertage loss
y_test = np.array(y_test)
y_pred = np.array(y_pred)
loss = np.abs(y_test - y_pred) / y_test
np.median(loss)
```

4.3.2 Model 2: Dominant Features Utilized

Similar to the model above, we construct, train, as well as test the model

```
# divide the data into training and testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, [2, 3,
    9]], data['fare'], test_size = 0.2, random_state = 0)

# build a random forest regressor with 5 fold cross validation on depth
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
rf = RandomForestRegressor(n_estimators = 100, random_state = 0, n_jobs
    = -1)
param_grid = {'max_depth': [ 20, 25, 30, 35, 40, 45, 50]}
grid_search = GridSearchCV(rf, param_grid, cv = 5, verbose = 2)
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)

# predict the test data
y_pred = grid_search.predict(X_test)

# calculate the mean squared error
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)
```

We then calculate the mean square error and the performance

```
# for each test point, calculate the percertage loss
y_test = np.array(y_test)
y_pred = np.array(y_pred)
loss = np.abs(y_test - y_pred) / y_test
np.median(loss)
```

4.3.3 Overall Analysis on Both Model

The training results for both models are listed in Table. 2 From the performance table, we

Model Name	Training Time	Best Forest Depth	MSE	Median Prediction Accuracy
Model 1	3 m 59 s	30	**803.6**	**91.6%**
Model 2	**2 m 38 s**	**20**	897.6	90.9%

Table 2: Model Performance

find that although the model with all features utilized have better performance compared with the model with only dominant features, it is significantly more complex and consumes more time to train. Also, utilizing all the features do not make the model outperform obviously in the prediction accuracy. Thus, we prefer to use the dominant features provided by the PCA analysis to construct the Random Forest Model. This enables us to predict the airfare more efficiently and economically.

We also plot the predicted airfare versus the original airfare in test set, the result is shown in Figure. 7 From the figure, we can conclude that almost all the points lie near the 100% accuracy line. This means that the model fits well for our data.

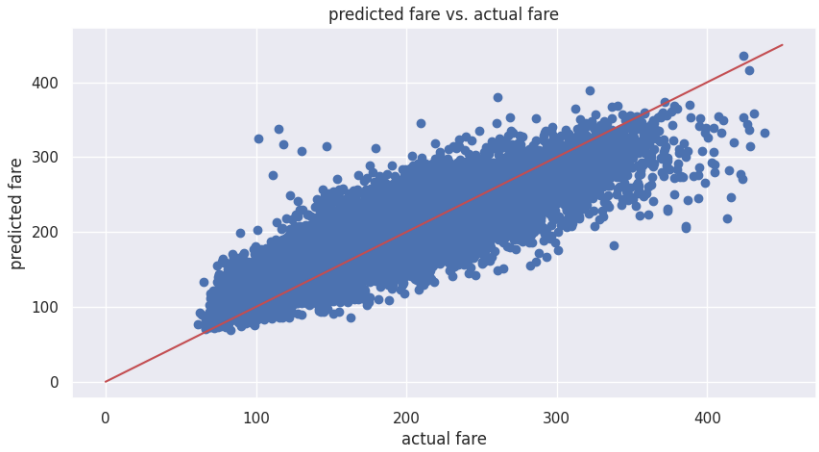

Figure 7: Model 2 Performance Visualization

Beside the inter model comparison, we also look at existing airfare prediction models. In particular, we compare our result with the model in [3]. Our model is a little weak than their random forest model in terms of accuracy. However, considering that their project is based on the real time data scrapped from website, which is far more informative, our static data based model is also persuasive.

5 Challenges and Limitations

5.1 Challenges

The major challenge for this project is that the data fields are still to little to figure out some strong relationships between certain factors and the airfare. Although we dig out some potential factors that can affect the airfare, they are not significant enough. This also result in a deep random forest which consumes calculation resource.

5.2 Limitations

Although I am an aviation lover, my personal background is not related to aviation or business. Experienced airfare analyst may make use of this data set far more better than me.

Also, due to the time limitation, other dimension reduction techniques like exploratory factor analysis are not used in the project. They might be able to find more useful clues and factors related to airfare.

6 Conclusion

Understand and predict airfare is never an easy task. The airfare is related to a fast changing market depends not only on inner factors like distance, but also outer factors like the financial condition worldwide. This project only explains a little bit about how airfare comes into being. Lots of further works are still needed to understand this problem better.

7 References

[1] *Why hasn't the U.S. developed a high-speed rail network?* URL: https://www.quora.com/Why-hasnt-the-U-S-developed-a-high-speed-rail-network.

[2] Randall Keizer. *Consumer Airfare Report: Table 1 - top 1,000 contiguous state city-pair markets: Department of Transportation - Data Portal*. Oct. 2022. URL: https://data.transportation.gov/Aviation/Consumer-Airfare-Report-Table-1-Top-1-000-Contiguo/4f3n-jbg2.

[3] MeshalAlamr. *MESHALALAMR/flight-price-prediction: Predicting flight ticket prices using a random forest regression model based on scraped data from kayak. A kayak scraper is also provided.* URL: https://github.com/MeshalAlamr/flight-price-prediction.

8 Appendix

All the source code in this appendix can be found in https://github.com/TaikiShuttle/SI618/tree/main/Project2