

PROJECT REPORT ON:

Dynamic Pricing Prediction for Cabs Using IBM Watson

-

TABLE OF CONTENTS:

1. Introduction	2
2. Literature survey	2
3. Theoretical Analysis	2
4. Experimental Investigations	4
5. Flow Chart	4
6. Results	5
7. Advantages and Disadvantages	6
8. Conclusion	7
9. Future Scope	7
10. Bibliography	7
11. Appendix	7-15

1. INTRODUCTION:

Many organizations do not have a direct role in travel and tourism but offer related products and services. Some examples would be offering travel insurance, parking facilities at airports, theatre and event tickets, car hire, and travel by rail or coach to airports, etc. at competitive rates. There are various different forms of

dynamic pricing:

1. Peak Pricing – This is a strategy that is common in transportation businesses. Airlines are a good example. Airlines often charge a higher price to travel during rush hour mostly on weekdays and sometimes on weekends.
2. Surge Pricing – Companies such as Uber respond dynamically to changes in supply and demand in order to price their services differently. Like most of us have noticed, this frequently happens on stormy evenings and nights when more people request for cabs. Taxify also not so long ago introduced dynamic pricing to ensure the drivers are encouraged to go online and offer services when the demand is high.

2. LITERATURE REVIEW

We know the popularity of uber in the recent years and about the urban citizens who are benefited by the uber. Later the author compares the difference between the competitive taxis and uber and defines new way of calling and also the new way of paying for cabs, the author also tells us about the importance of data produced by the cabs daily and also about the visualization and analysis of data. After that the author tells how the different time and different environments will have an effect on passengers to make different choices.

3. THEORITICAL ANALYSIS:

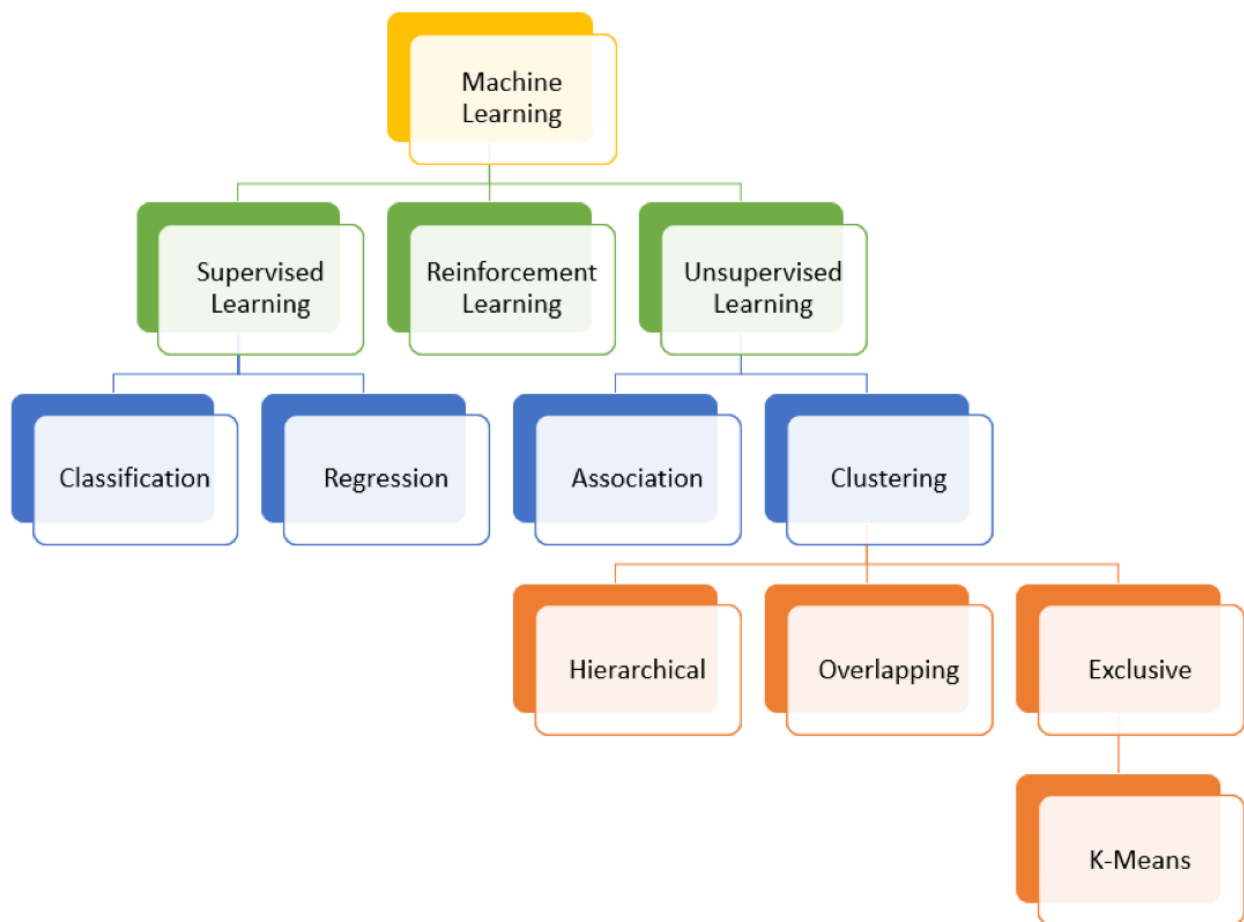
After analyzing the different parameters, here are a few pointers that we can conclude. If you were a Business Analyst or a data scientist working for either Uber or Lyft, you could draw the following conclusions:

1. Uber is more economical; however, Lyft also provides fair competition.
2. People prefer to have shared rides during the nighttime.

3. People avoid taking rides when it rains.
4. When traveling long distances, the price does not increase linearly. However, based on the time and demand, a surge can affect the cost.
5. Uber can be the first choice for long distances.

However, acquiring and analyzing similar data is only a tipping point for several companies. There are several enterprises in the market that can help bring data from multiple sources and in different formats into the data warehouse of your choice.

Block Diagram:



b. **Hardware / Software designing:**

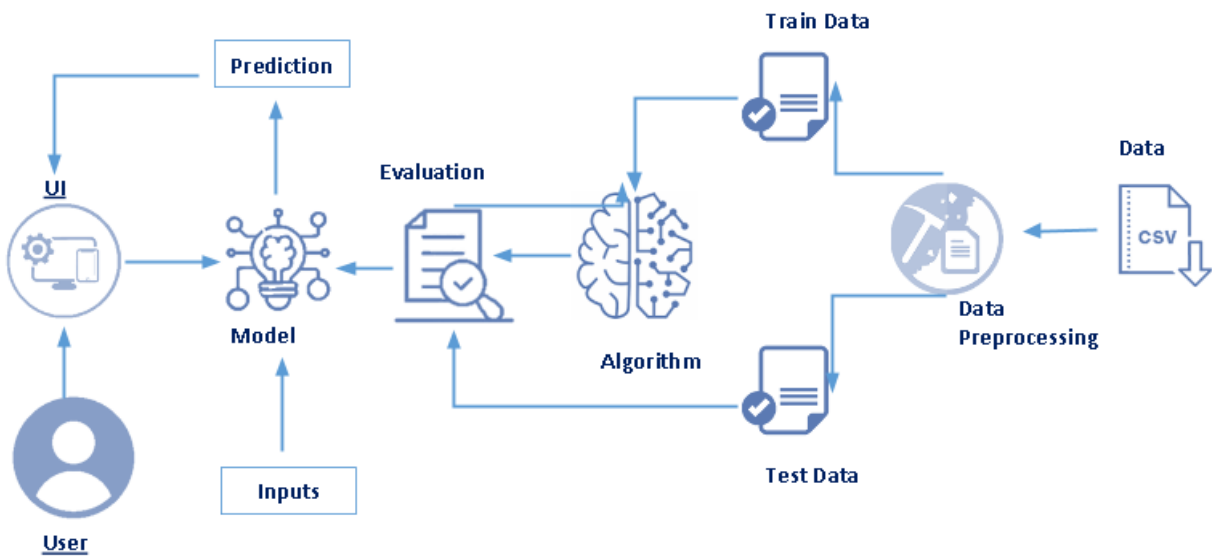
- python
- HTML
- Google Chrome
- IBM Cloud
- Microsoft Excel

4. **EXPERIMENTAL INVESTIGATIONS:**

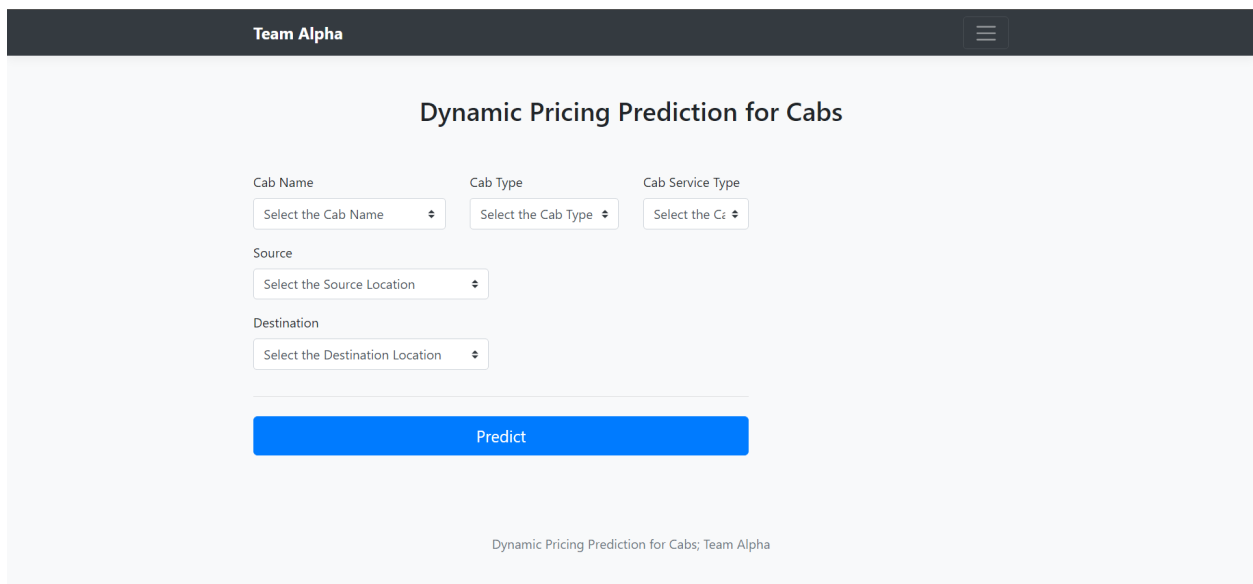
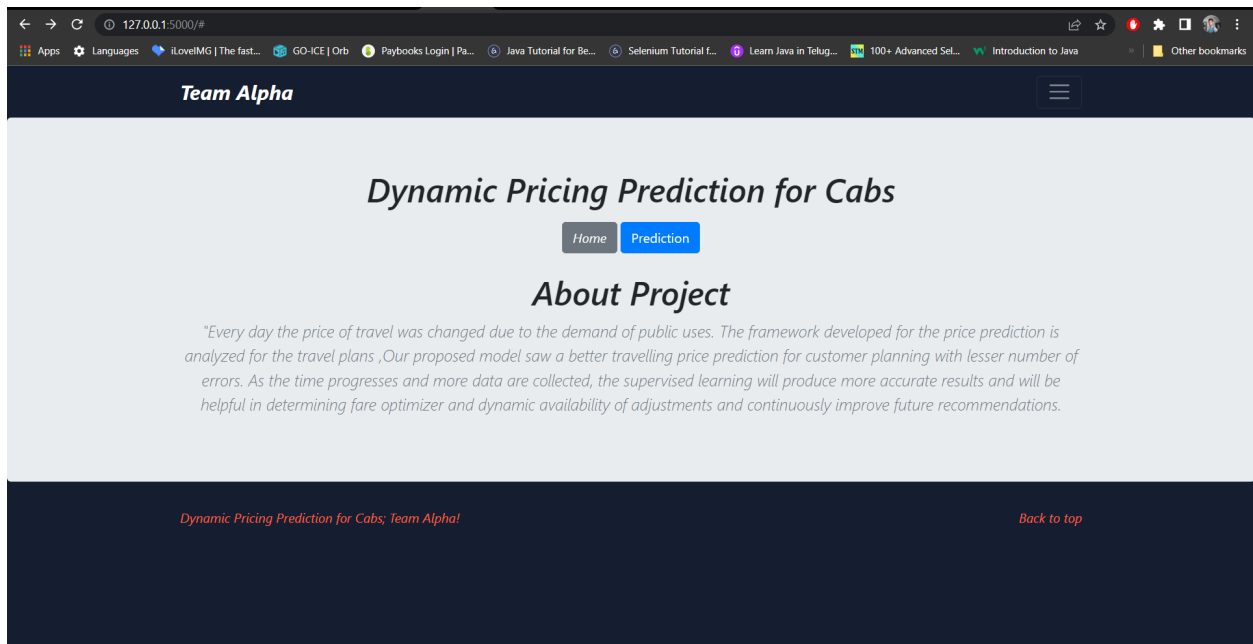
After collecting the segmentation data set, required data preprocessing and data visualizations are done to our data set and later on we checked the accuracy of our data set by applying different machine learning algorithms like H-clustering, k-means clustering Decision tree, Random Forest, KNN, and xgboost. As we used here different machine learning techniques like Supervised and Unsupervised Machine learning algorithms, we got different accuracies accordingly and from that we got highest accuracy for xgboost Supervised machine learning algorithm. We stored that model in pickle file and by running that file in flask with required python code with extracting the saved pickle file and by coding in HTML for the web pages i.e., for user interface we got the required output for our project. Later on we deployed our model in IBM Watson cloud and after deploying we got the python code according to that deployment and after merging with our python code we got the same output as before i.e., by giving the required values the model predicts whether the customer is highly potential or potential or not potential.

5. FLOWCHART:

5. FlowChart



6. Results:



Team Alpha

Dynamic Pricing Prediction for Cabs

Cab Name

Uber

Cab Type

Uber WAV

Cab Service Type

Uber WAV

Source

Back Bay

Destination

North Station

Predict

Dynamic Pricing Prediction for Cabs; Team Alpha

Team Alpha

Final Fare Is :- "\$ 9.547530349537121"

7. ADVANTAGES & DISADVANTAGES:

As for the relative pros and cons, it's vital to comprehend the notion that the seller usually wins: dynamic pricing algorithms ideally allow for maximizing the profits out of every single client. The seller can establish some discriminatory policies that are going to reflect his or her subjective overview of the market, even though the customers may realize that they are overpaying for the same solution, factually speaking. Such an approach may

turn away some of the clients, yet still, allow the seller to receive the utmost benefit. Otherwise, if the demand for the product is relatively low, it's still possible to generate some revenue by offering promotions for appreciating the general supply/demand chart. In such a scenario, it will be the customer who “gains” the most. Even though the real-life models may contain slightly more variables that influence the pricing, it always comes down to these rooting rules and principles.

8. **CONCLUSION**

This project gives us basic understanding of how we can use machine learning in order to predict the cab fare from given source to destination before starting the cab ride. The model created is able to give us the predictions which are not exactly equal to the actual the price fluctuation is around the difference of ten to twenty rupees compared to the actual price. Since the model is good but not the best, we can improve the predictions of the model by using the Fine-tuning technique. If fine tuning is applied to the existing model, we are able to get higher accuracy than the proposed model.

9. **Future Scope**

From a long cycle perspective, the ride demand may first increase and then decrease, or first decrease and then increase, so the benchmark model may not be intuitive. In this section, we construct a general model to extend our benchmark model in Section 4, which can provide reference for the ride-hailing platform enterprises when the situation in which there is increasing or decreasing fluctuation of market demand (e.g., first increasing and then decreasing). In addition, we find the characteristics of the optimal dynamic price trajectory in the general model are consistent with those of the benchmark model, that is, the other forms of requirement functions will not change the correctness of the conclusion.

10. **BIBILORAPHY**

[1] J. Guo, "Analysis and comparison of Uber, Taxi and Uber request via Transit,," IIJRD, vol. 4, no. 2, pp. 60- 62, 2015.

[2] N. G. G. K. Uyanik, "A study on multiple linear regression analysis," Procedia- Social and Behavioral Sciences, vol. 106, pp. 234-240, 2013.

[3] Y. J. Y. Zhang, "A data-driven quantitative assessment model for taxi industry: the scope of business ecosystem's health," Eur. Transp. Res., vol. 9, pp. 1-23, 2017.

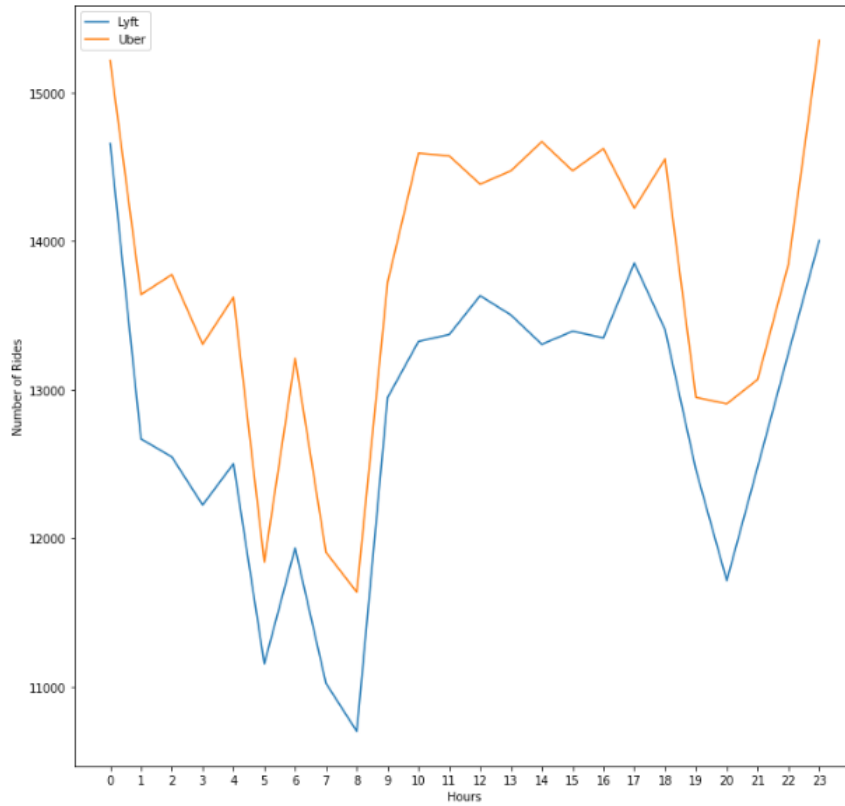
[4] U. Patel, "NYC Taxi Trip and Fare Data Analytics using BigData," Department of Computer Science and Engineering University of Bridgeport, USA, 2018.

[5] J. Chao, "Modeling and Analysis of Uber's Rider Pricing," Advances in Economics, Business and Management Research, vol. 109, pp. 639-711, 2019.

11. APPENDIX:

- a. After Importing necessary libraries and loading the data set.
- b. We can also see count,standard deviation,mean,min,max,25%,50%,75% with data.describe() command.
- c. We can see about our data by checking the info, here we can also see that there are no null values mentioned in our columns.
- d. Next we perform Data visualization.

```
In [25]: # The ride distribution in one day
fig, ax = plt.subplots(figsize=(12,12))
ax.plot(df_rides_weather[df_rides_weather['cab_type'] == 'Lyft'].groupby('Hour').Hour.count().index, df_rides_weather[df_rides_w
ax.plot(df_rides_weather[df_rides_weather['cab_type'] == 'Uber'].groupby('Hour').Hour.count().index, df_rides_weather[df_rides_w
ax.legend()
ax.set(xlabel = 'Hours', ylabel = 'Number of Rides')
plt.xticks(range(0,24,1))
plt.show()
```

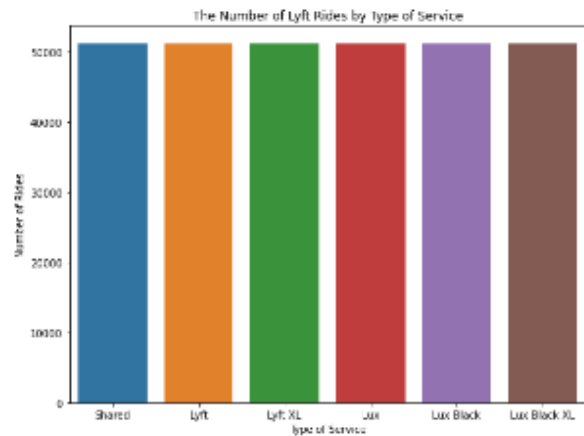
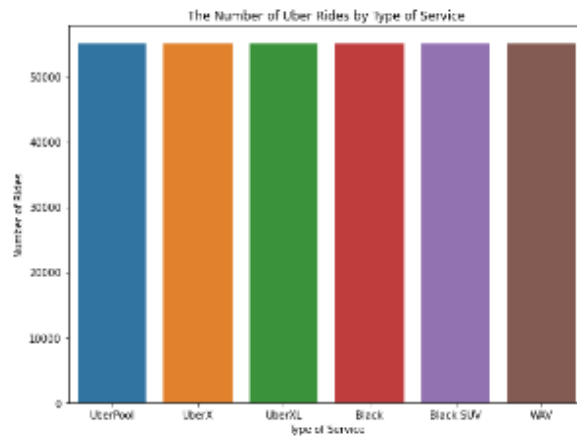
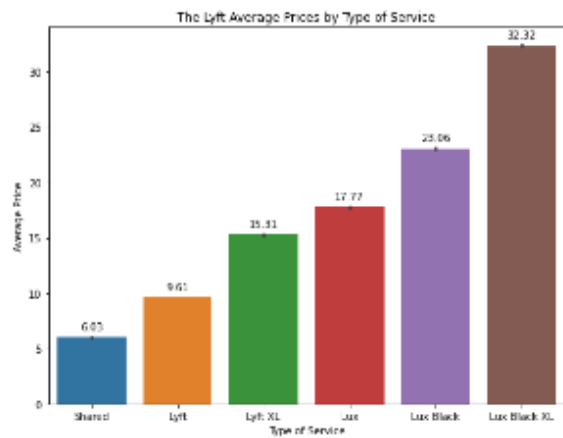
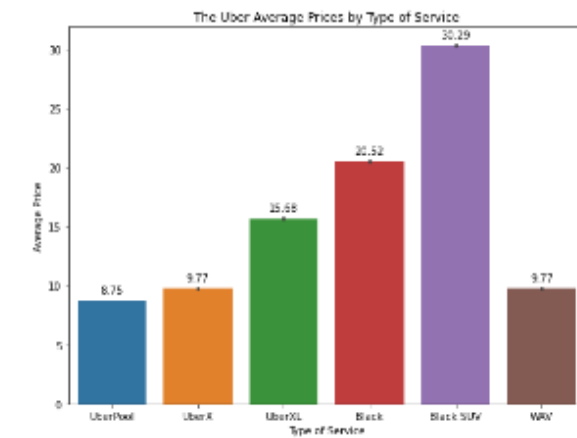


```

: # The Average price of rides by type of service
import seaborn as sns

uber_order = [ 'UberPool', 'UberX', 'UberXL', 'Black', 'Black SUV', 'WAV' ]
lyft_order = [ 'Shared', 'Lyft', 'Lyft XL', 'Lux', 'Lux Black', 'Lux Black XL' ]
fig, ax = plt.subplots(2,2, figsize = (20,15))
ax1 = sns.barplot(x = df_rides_weather[df_rides_weather['cab_type'] == 'Uber'].name, y = df_rides_weather[df_rides_weather['cab_type'] == 'Uber'].price, order = uber_order)
ax2 = sns.barplot(x = df_rides_weather[df_rides_weather['cab_type'] == 'Lyft'].name, y = df_rides_weather[df_rides_weather['cab_type'] == 'Lyft'].price, order = lyft_order)
ax3 = sns.barplot(x = df_rides_weather[df_rides_weather['cab_type'] == 'Uber'].groupby('name').name.count().index, y = df_rides_weather[df_rides_weather['cab_type'] == 'Uber'].price, order = uber_order)
ax4 = sns.barplot(x = df_rides_weather[df_rides_weather['cab_type'] == 'Lyft'].groupby('name').name.count().index, y = df_rides_weather[df_rides_weather['cab_type'] == 'Lyft'].price, order = lyft_order)
for p in ax1.patches:
    ax1.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.get_height()), ha = 'center', va = 'center',
    for p in ax2.patches:
    ax2.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.get_height()), ha = 'center', va = 'center',
ax1.set(xlabel = 'Type of Service', ylabel = 'Average Price')
ax2.set(xlabel = 'Type of Service', ylabel = 'Average Price')
ax3.set(xlabel = 'Type of Service', ylabel = 'Number of Rides')
ax4.set(xlabel = 'Type of Service', ylabel = 'Number of Rides')
ax1.set_title('The Uber Average Prices by Type of Service')
ax2.set_title('The Lyft Average Prices by Type of Service')
ax3.set_title('The Number of Uber Rides by Type of Service')
ax4.set_title('The Number of Lyft Rides by Type of Service')
plt.show()

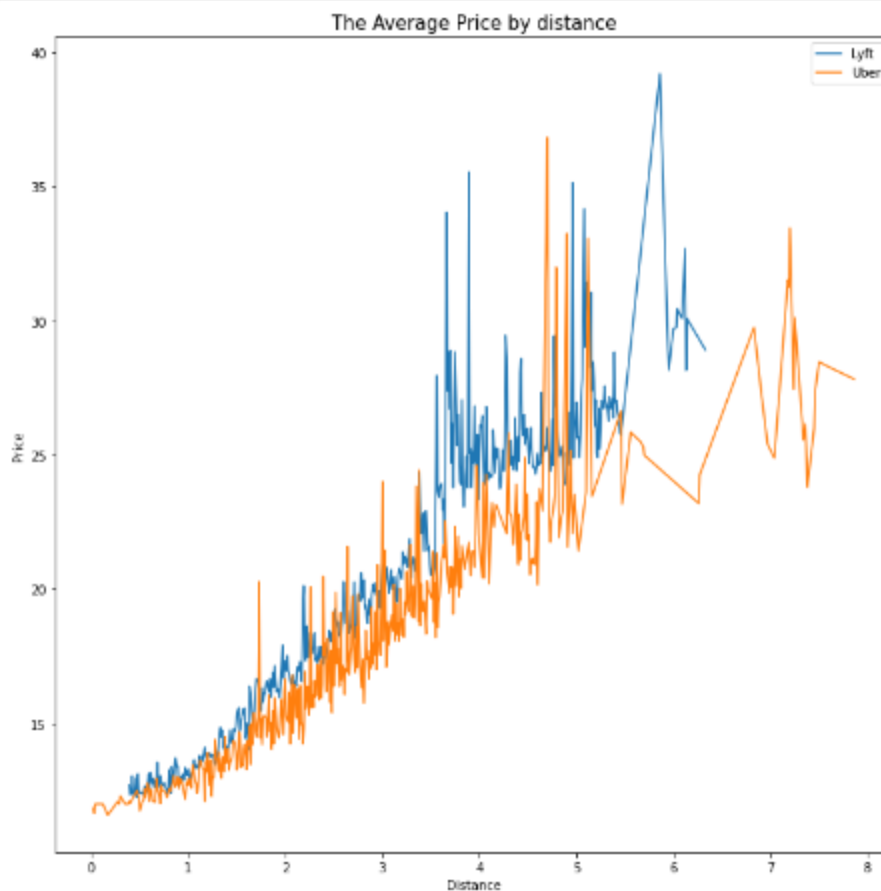
```



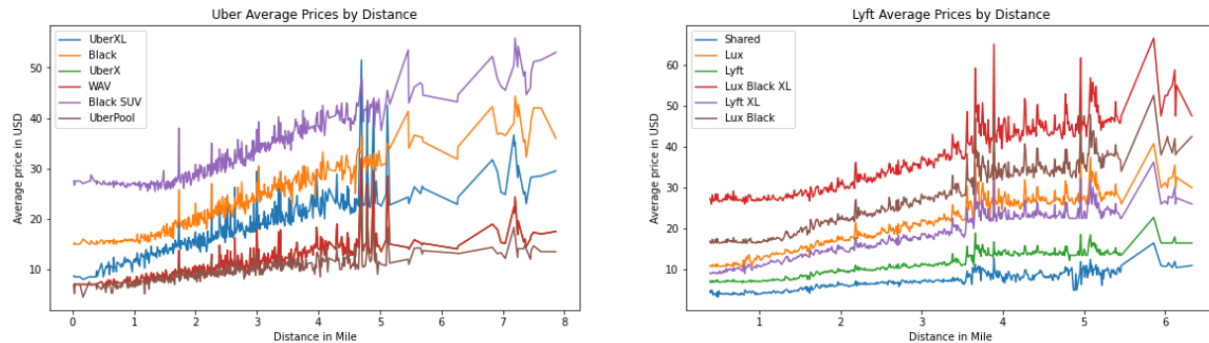
```

# The average price by distance
fig, ax = plt.subplots(figsize = (12,12))
ax.plot(df_rides_weather[df_rides_weather['cab_type'] == 'Lyft'].groupby('distance').price.mean().index, df_rides_weather[df_rides_weather['cab_type'] == 'Lyft'].groupby('distance').price.mean().index, df_rides_weather[df_rides_weather['cab_type'] == 'Uber'].groupby('distance').price.mean().index, df_rides_weather[df_rides_weather['cab_type'] == 'Uber'].groupby('distance').price.mean().index)
ax.set_title('The Average Price by distance', fontsize= 15)
ax.set_xlabel = 'Distance', ylabel = 'Price' )
ax.legend()
plt.show()

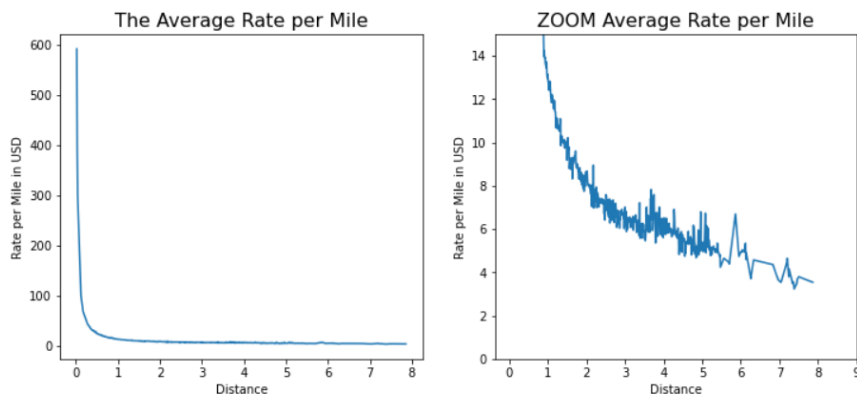
```

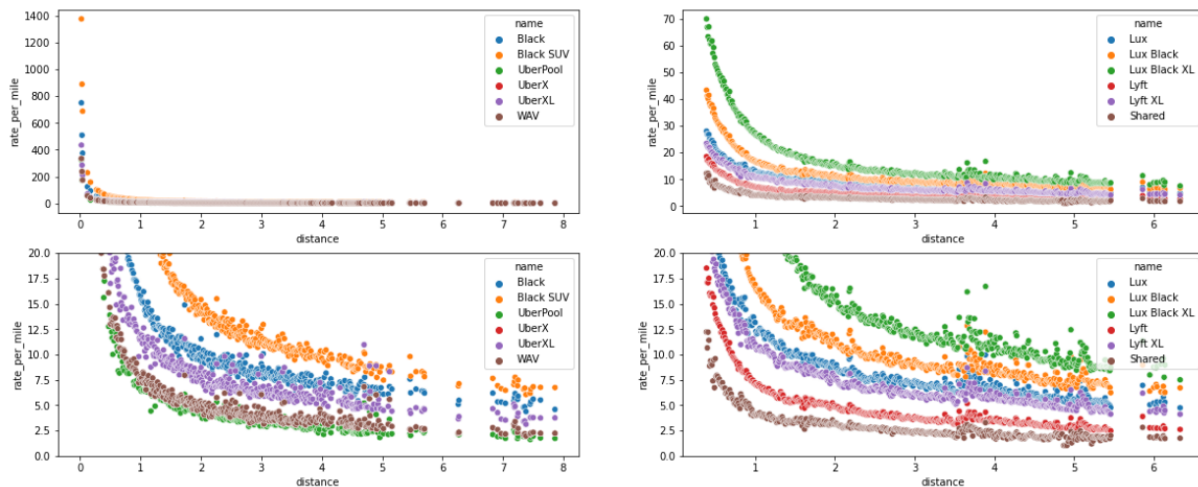


```
# The average price by distance
fig, ax = plt.subplots(1,2 , figsize = (20,5))
for i,col in enumerate(df_rides_weather[df_rides_weather['cab_type'] == 'Uber']['name'].unique()):
    ax[0].plot(df_rides_weather[ df_rides_weather['name'] == col].groupby('distance').price.mean().index, df_rides_weather[ df_rides_weather['name'] == col].groupby('distance').price.mean().values)
ax[0].set_title('Uber Average Prices by Distance')
ax[0].set_xlabel = 'Distance in Mile', ylabel = 'Average price in USD')
ax[0].legend()
for i,col in enumerate(df_rides_weather[df_rides_weather['cab_type'] == 'Lyft']['name'].unique()):
    ax[1].plot(df_rides_weather[ df_rides_weather['name'] == col].groupby('distance').price.mean().index, df_rides_weather[ df_rides_weather['name'] == col].groupby('distance').price.mean().values)
ax[1].set_xlabel = 'Distance in Mile', ylabel = 'Average price in USD')
ax[1].set_title('Lyft Average Prices by Distance')
ax[1].legend()
plt.show()
```



```
# the average rate per mile
df_rides_weather['rate_per_mile'] = round((df_rides_weather['price'] / df_rides_weather['distance'] ),2)
# The average rate per mile plot
fig, ax = plt.subplots(1,2,figsize = (12,5))
ax1 = sns.lineplot(x = df_rides_weather.groupby(['distance'])['rate_per_mile'].mean().index, y = df_rides_weather.groupby('distance')['rate_per_mile'].mean().values)
ax2 = sns.lineplot(x = df_rides_weather.groupby(['distance'])['rate_per_mile'].mean().index, y = df_rides_weather.groupby('distance')['rate_per_mile'].mean().values)
plt.xticks(range(0, 10,1))
ax1.set_xlabel = 'Distance', ylabel = 'Rate per Mile in USD')
ax2.set_xlabel = 'Distance', ylabel = 'Rate per Mile in USD', ylim = (0,15))
ax1.set_title('The Average Rate per Mile', fontsize = 16)
ax2.set_title('ZOOM Average Rate per Mile', fontsize = 16)
plt.show()
```





Using Random forest Algorithm

```
]: from sklearn.ensemble import RandomForestRegressor
rand=RandomForestRegressor(n_estimators=20,random_state=52,n_jobs=-1,max_depth=4)
rand.fit(x_train,y_train)
```

<ipython-input-64-57509395cdf7>:3: DataConversionWarning: A column-vector y was passed instead of the shape of y to (n_samples,), for example using ravel().

```
rand.fit(x_train,y_train)
```

```
]: RandomForestRegressor(max_depth=4, n_estimators=20, n_jobs=-1, random_state=52)
```

Predicting the Result

```
: ypred=rand.predict(x_test)
print(ypred)
```

```
[33.44544798 19.16381383  9.54753035 ...  6.02421004 26.79738243
17.55244465]
```

Score of the model

```
: rand.score(x_train,y_train)
```

```
: 0.7575275520145969
```

