Uber and Lyft Dataset Boston, MA

Introduction

Sometimes it's safer or more practical to let someone else deal with the traffic and driving hassles. The two iconic transportation service firms, Uber and Lyft, eventually developed from this concept.

Lyft operates in the U.S. and Canada. The business has a number of different service categories and levels, and it has certain requirements for the drivers of the cars used. The Lyft app for cellphones alerts users when the driver will arrive and provides them with an estimated fare in advance.

Uber offers services to customers in numerous cities throughout the world, including those in the European Union, Central and South America, Africa, Asia, Australia, and New Zealand. It also provides services to customers in the United States and Canada.Uber specifies the kind of vehicle that must be used as well as a number of service types. The Uber app makes it easier for drivers and customers to connect and provides an upfront estimate of the fare.

Problem statement

In this project, we have a dataset consisting of 6,93,071 data points with 57 features of Boston city in Massachusetts . The goal of this project is to compare the taxi services Uber and Lyft in terms of price, source and destination, time, weather, etc. Additionally, a predictive model will be created to estimate the price for a given source and destination in terms of time, weather, surge multiplier, etc.

Dataset and its attributes

Dataset link - https://www.kaggle.com/datasets/brllrb/uber-and-lyft-dataset-boston-ma

The dataset consists of the following features -

['timestamp', 'hour', 'day', 'month', 'datetime', 'timezone', 'source', 'destination', 'cab_type', 'name', 'price', 'distance', 'surge_multiplier', 'latitude', 'longitude', 'temperature', 'apparentTemperature', 'short_summary', 'precipIntensity', 'precipProbability', 'humidity', 'windSpeed', 'windGust', 'windGustTime', 'visibility', 'temperatureHigh', 'temperatureHighTime', 'temperatureLow', 'temperatureLowTime', 'apparentTemperatureHigh', 'apparentTemperatureHighTime', 'apparentTemperatureLow', 'apparentTemperatureLowTime', 'icon', 'dewPoint', 'pressure', 'windBearing', 'cloudCover', 'uvIndex', 'visibility.1', 'ozone', 'sunriseTime', 'sunsetTime', 'moonPhase', 'precipIntensityMax', 'uvIndexTime', 'temperatureMin', 'temperatureMinTime', 'temperatureMax', 'temperatureMaxTime', 'apparentTemperatureMin', 'apparentTemperatureMinTime', 'apparentTemperatureMax', 'apparentTemperatureMaxTime']

Many of these features can therefore be removed, which brings us to data cleaning and preprocessing.

Data Cleaning and Preprocessing

Data cleaning is preparing data for analysis by removing or modifying incorrect, incomplete, irrelevant, duplicated, or improperly formatted data.

This data is usually unnecessary or helpful when analyzing data because it may hinder the process or provide inaccurate results. There are several

methods for cleaning data depending on how it is stored along with th,e answers being sought.

- ***Dropping features that are of no use or have been repeated:***

df.drop(['id','long_summary'],axis = 1,inplace = True)

df.drop('product_id',axis = 1,inplace = True)

- ***Converting timestamps to date time format:***

from datetime import datetime

def to_DateTime(timestamp):

return datetime.fromtimestamp(timestamp)

pd.DataFrame(df['timestamp'].apply(to_DateTime))

	timestamp
0	2018-12-16 09:30:07.890
1	2018-11-27 02:00:23.677
2	2018-11-28 01:00:22.198
3	2018-11-30 04:53:02.749
4	2018-11-29 03:49:20.223
...	...
693066	2018-12-01 23:53:06.000
693067	2018-12-01 23:53:06.000
693068	2018-12-01 23:53:06.000
693069	2018-12-01 23:53:06.000
693070	2018-12-01 23:53:06.000

693071 rows × 1 columns

- ***Dropping datetime as it is the same as timestamp:***

df.drop('datetime',axis = 1,inplace = True)

- ***Extracting minutes from timestamp and adding a new column minute in the data frame:***

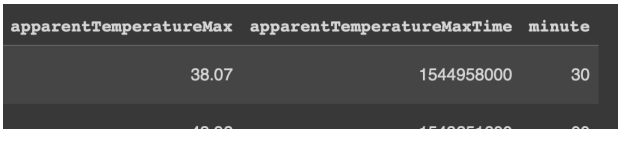
from datetime import datetime

def extractMinutes(date):

Minute = date.strftime("%M")

return Minute

df['minute'] = df['timestamp'].apply(extractMinutes)



apparentTemperatureMax	apparentTemperatureMaxTime	minute
38.07	1544958000	30

43.92	1543251600	00
44.12	1543320000	00
38.53	1543510800	53
35.75	1543420800	49
...	...	...
44.09	1543690800	53
44.09	1543690800	53
44.09	1543690800	53
44.09	1543690800	53
44.09	1543690800	53

- ***Handling missing values dropping one cabtype of uber as we dont have enough data about it:***

```
df[df['price'].isna()]['name'].value_counts()

df.dropna(inplace=True)

df
```

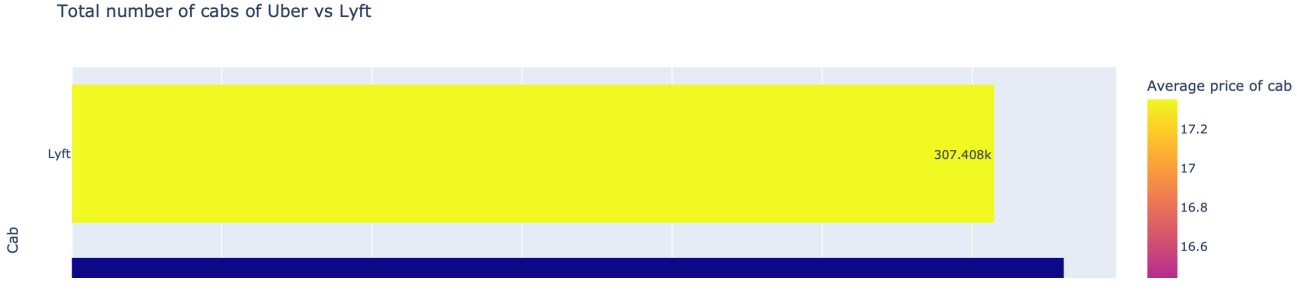
- ***Dropping timezones as there is no variation:***

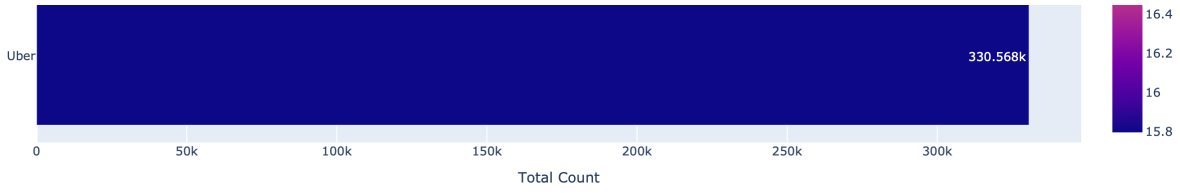
```
df['timezone'].value_counts()

df.drop('timezone',axis=1,inplace=True)
```

EDA(Exploratory Data Analysis)

Exploratory data analysis (EDA) is a technique for analysing data sets in order to highlight their key properties. It often makes use of statistical graphics and other types of data visualisation. The following graphs were plotted using EDA:

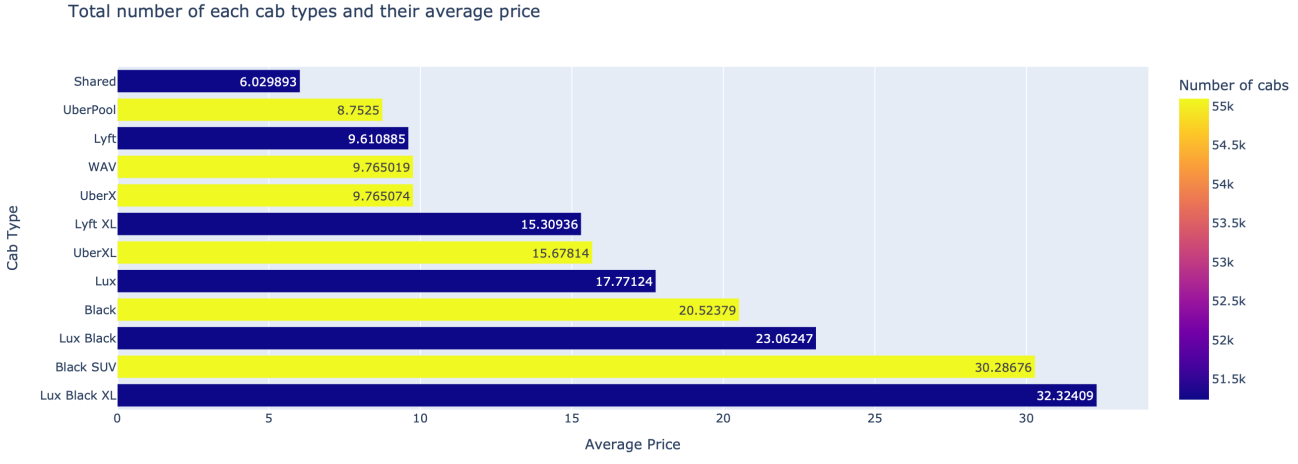
- **Total number of cabs of Uber vs Lyft and their average prices**





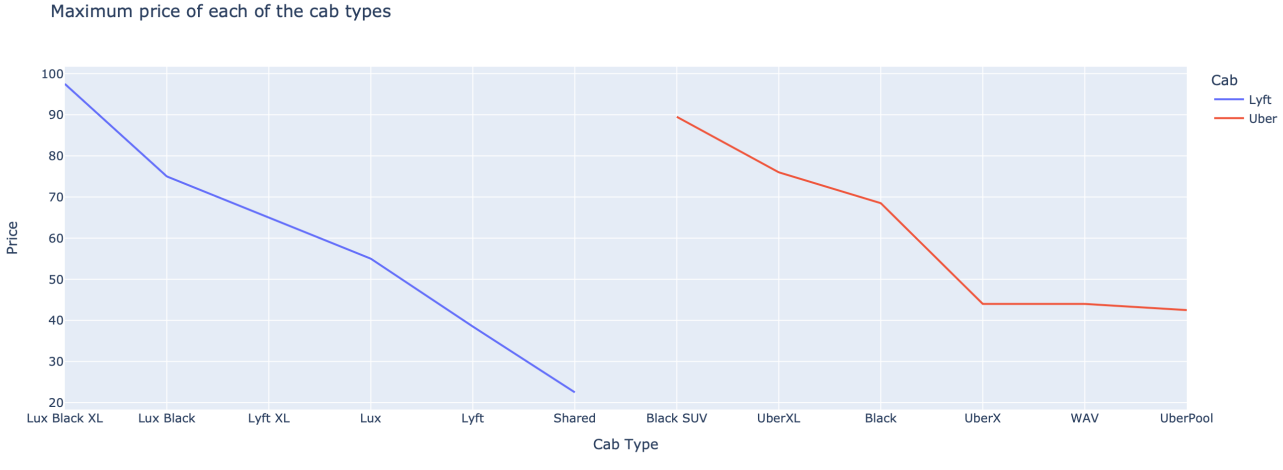
From the graph above, we can see that Uber has more cabs than Lyft does, however Uber charges a lesser price than Lyft.

- ***Total number of each cab type of Uber vs Lyft and their average prices***



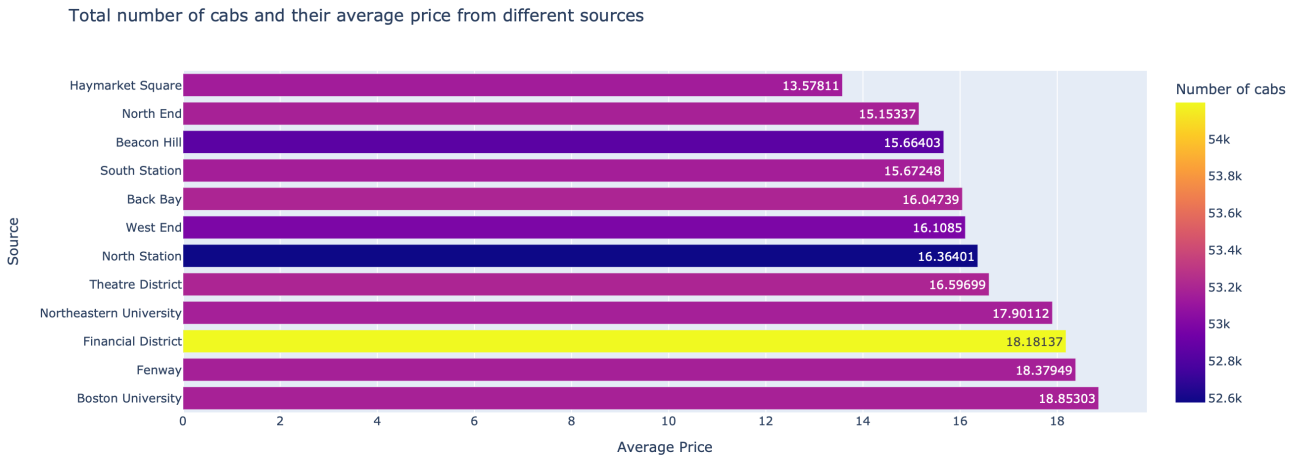
According to the graph above, Uber's Black SUV and Lux Black XL charge their customers the highest prices, while UberPool and Lyft Shared provide the lowest prices. Additionally, Uber has more cabs than Lyft.

- ***Maximum prices of each cab type of Uber vs Lyft***



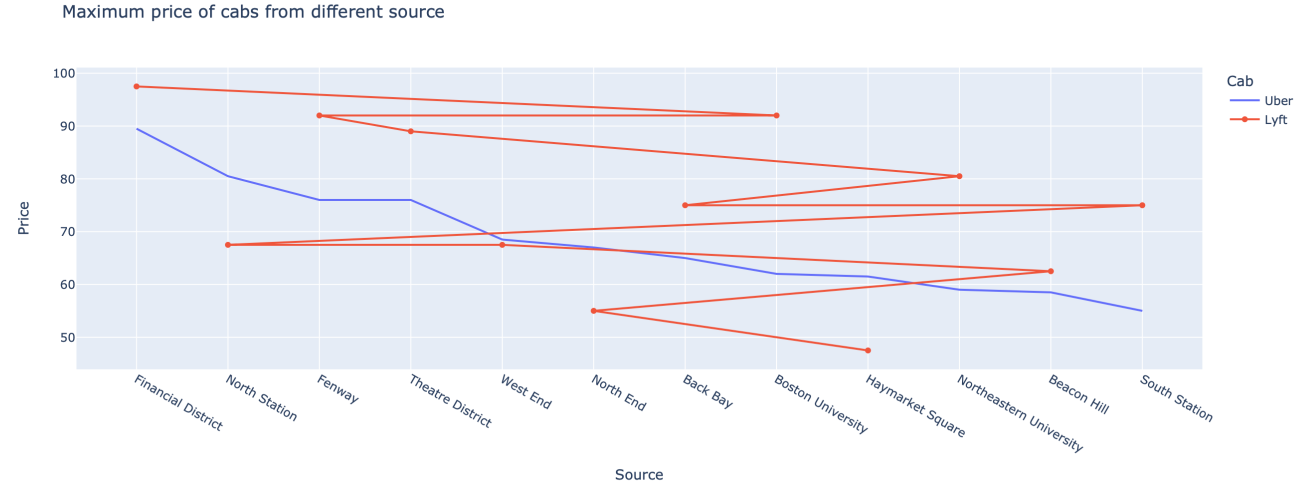
The graph above displays the highest prices that Lyft and Uber have ever charged for various types of cabs. The most costly ride is Lux Black XL, which has a maximum price of about $100.

- ***Total number of cabs and their average prices from different sources***



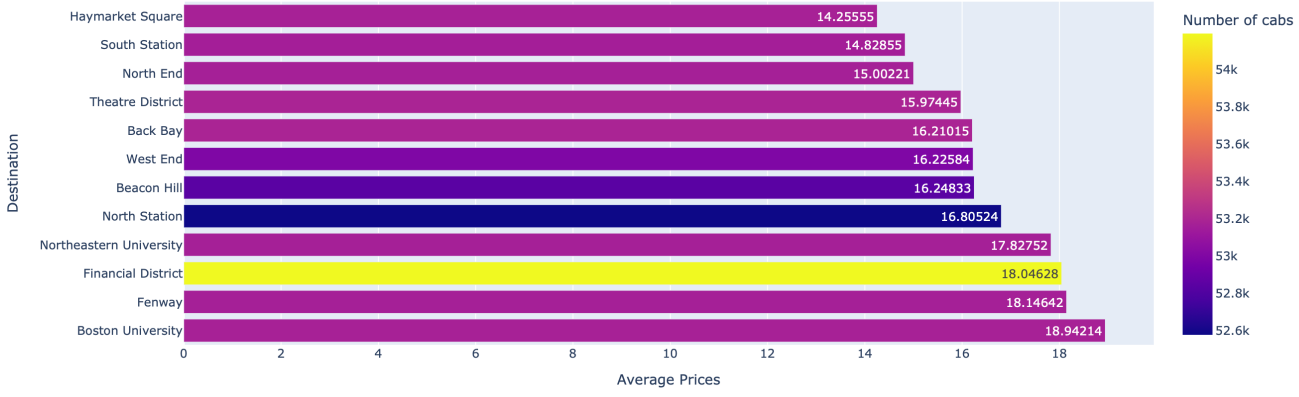
The graph at the top shows the average prices for different types of cabs from various sources. Cab costs from Boston University and Fenway are typically higher than those from Beacon Hill, the North End, Haymarket Square, etc. The Financial district is where the most number of taxis are operated.

- ***Maximum prices charged by Uber and Lyft from different sources***



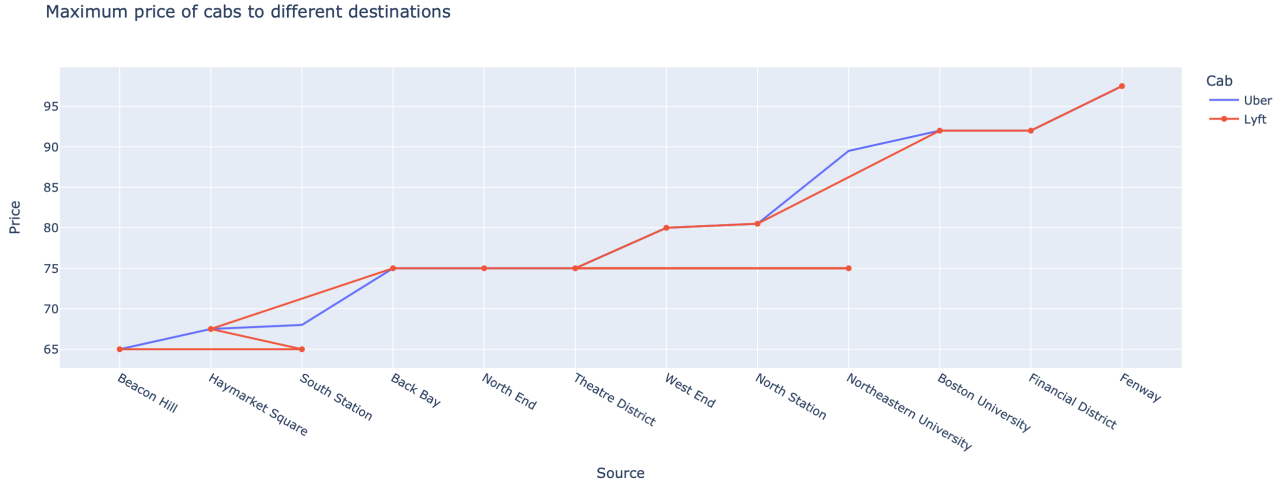
The graph above displays the highest prices ever incurred by Uber and Lyft from a number of sources. Lyft has so far charged the maximum prices from locations like the Financial District, Fenway, and many others, while Uber has done so from locations like North Station and Haymarket Square.

- ***Total number of cabs and their average prices to different destinations***



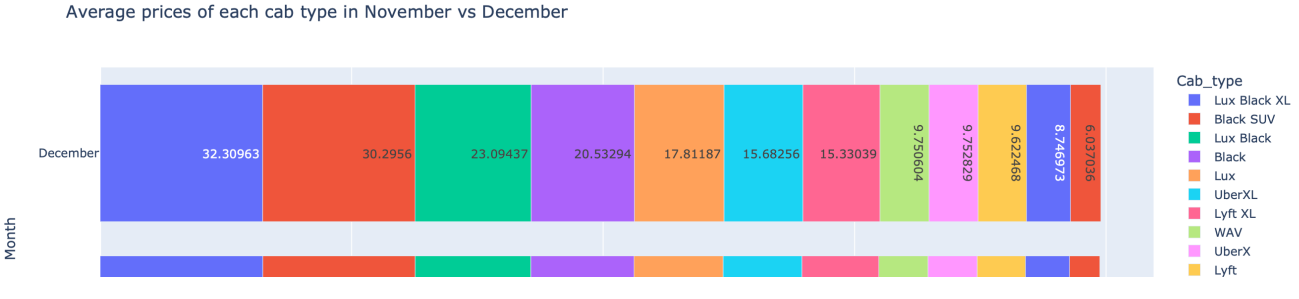
The graph at the top shows the average prices for different types of cabs to various destinations. Cab costs to Boston University and Fenway are typically higher than those till South Station, the North End, Haymarket Square, etc. The Financial district is where the most number of taxis are operated.

- ***Maximum prices charged by Uber and Lyft to different destinations***



The graph above displays the highest prices ever incurred by Uber and Lyft to a number of destinations. The highest price ever charged is same by Uber and Lyft for most of the destinations with exceptions being Northeastern University, South Station.

- ***Average prices of each cab type in November vs December***

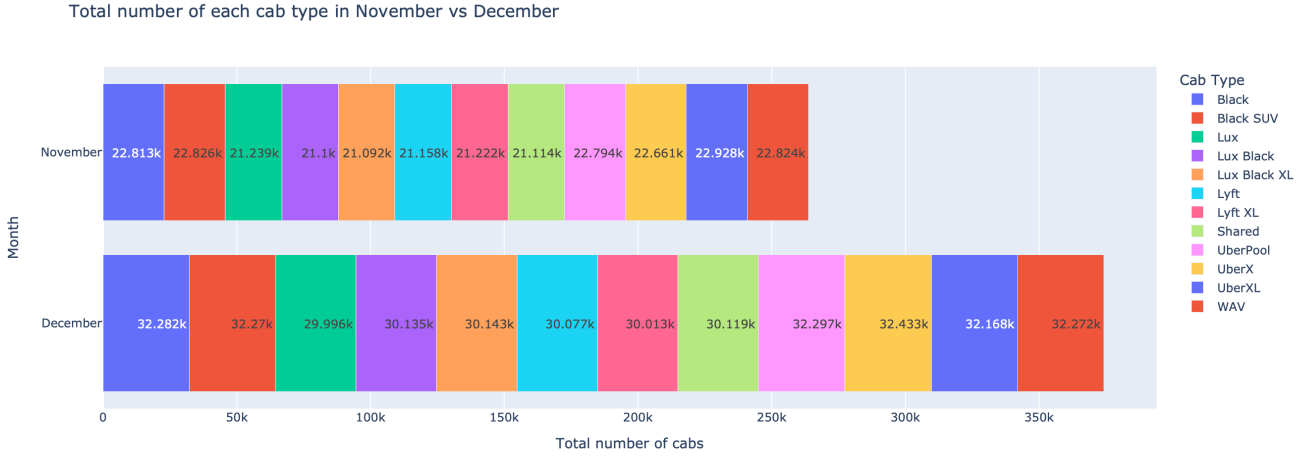


The data shows that, with a very slight difference, the cab fares were cheaper for the month of December than for the month of November for cab types such Lux Black XL, WAV, UberX, and UberPool. Fares for other types of cabs slightly increased in December. In general, cab fares are about the same in both months.

- ***Maximum prices of each cab type in November vs December***

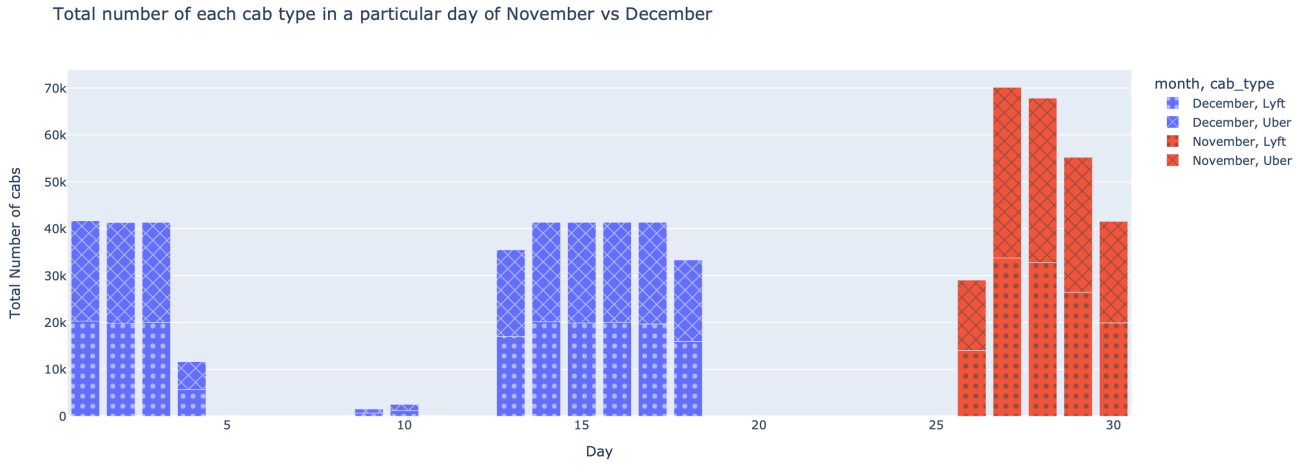
The data reveals that for cab types like Lux Black XL, Uber Black, and WAV, the highest price ever charged is in December, however for cab types like Lux Black, UberX, and Shared, the highest price ever charged is the same for both months.

- ***Total number of each cab type in November vs December***



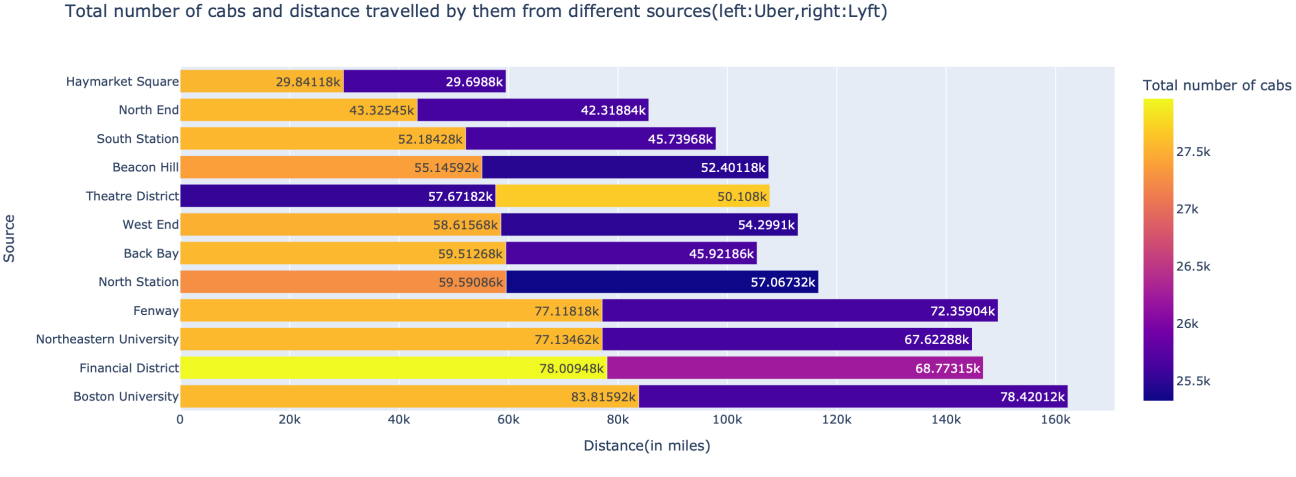
The data shows that there are more number of cabs of each cab type in December than in November, which can be attributed in part to the fact that fewer days in November were included in the dataset.

- ***Total number of each cab type in a particular day of November vs December***



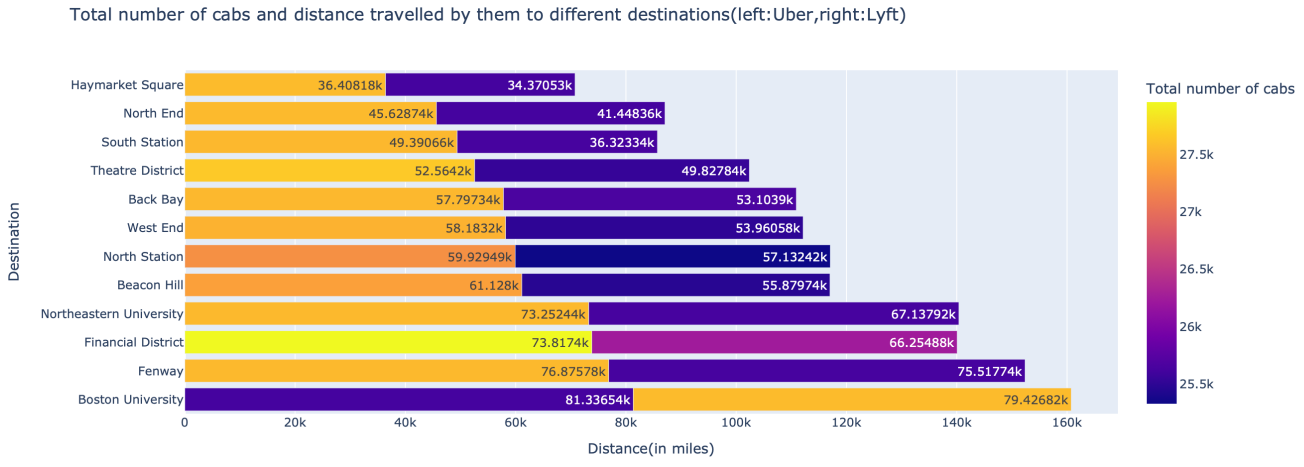
According to the data, there are the greatest number of cabs of each type on dates like November 26 and 27, with roughly equal numbers of Uber and Lyft cabs on different dates.

- ***Total number of cabs and distance travelled by them from different sources***



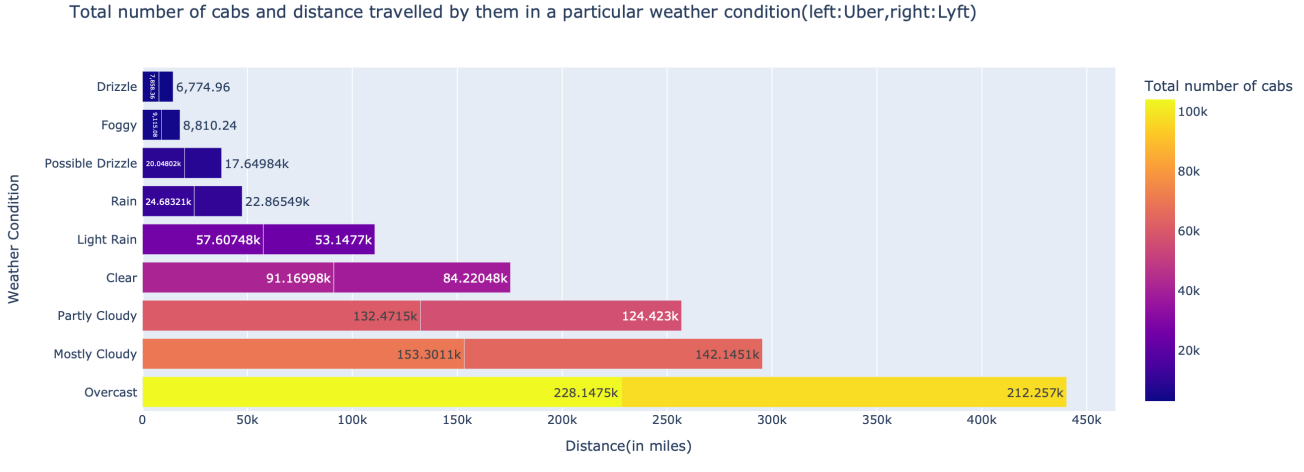
The data shows that, with the exception of Theatre District, there are more Uber than Lyft cabs available at any given time at a given source. Additionally, Uber and Lyft drive a greater distance overall from locations like Boston University and the Financial District, with Uber having the upper hand.

- ***Total number of cabs and distance travelled by them to different destinations***



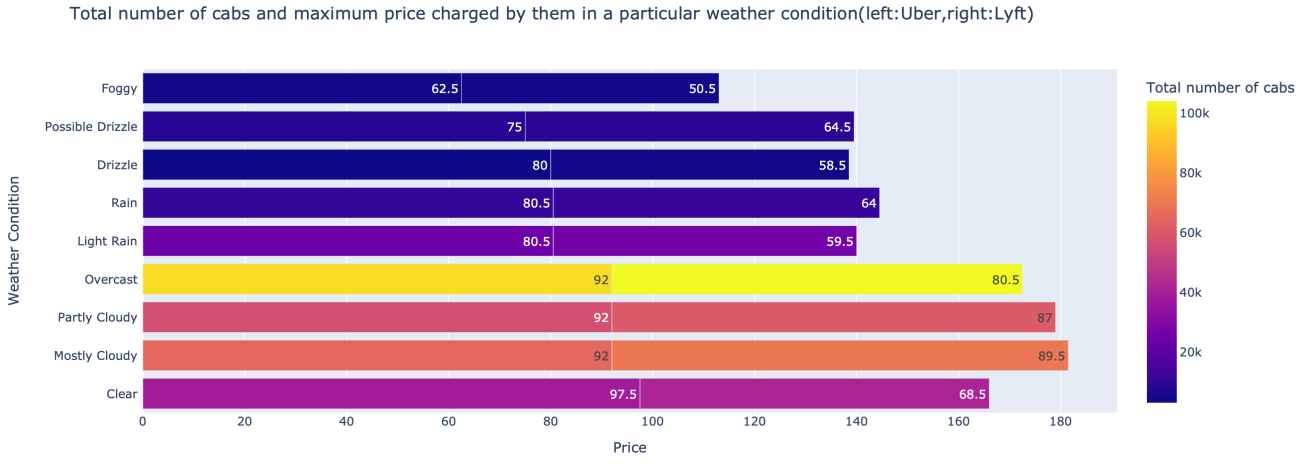
The data shows that, with the exception of Boston University, there are more Uber than Lyft cabs available at any given time for different destinations. Additionally, Uber and Lyft drive a greater distance overall to locations like Boston University and Fenway, with Uber having the upper hand.

- ***Total number of cabs and distance travelled by them in a particular weather condition***



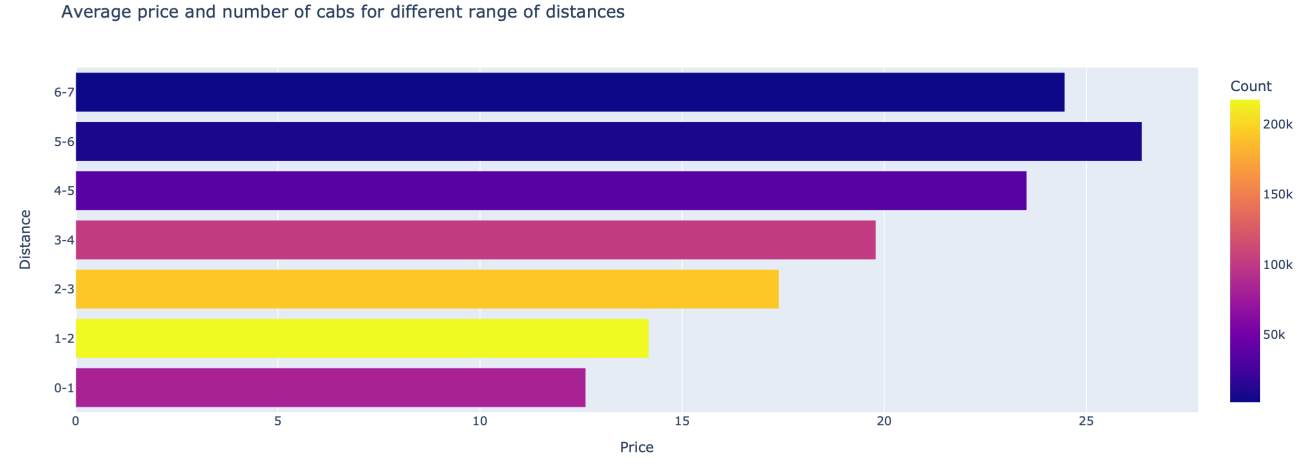
According to the data, under all weather conditions there are more Uber than Lyft cabs accessible at any given moment. Additionally, Uber and Lyft cover a greater distance overall when it's cloudy or overcast outside, with Uber having the advantage. It's challenging to find a cab in severe conditions like mist and fog.

- ***Total number of cabs and maximum price charged by them in a particular weather condition***



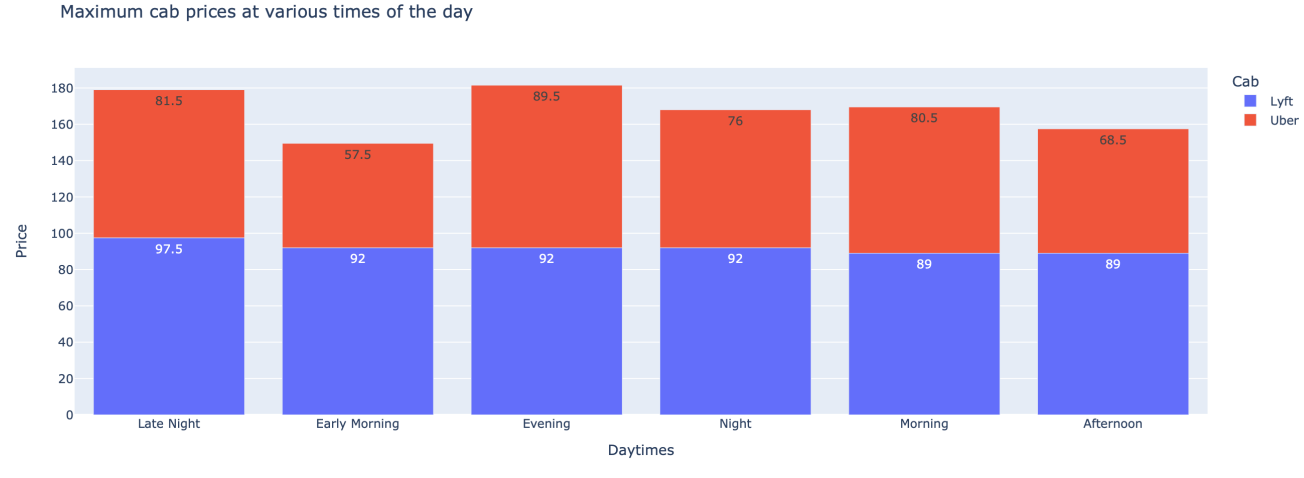
Uber has typically charged the highest cost for various weather situations, according the statistics.

- ***Average price and number of cabs for different range of distances***



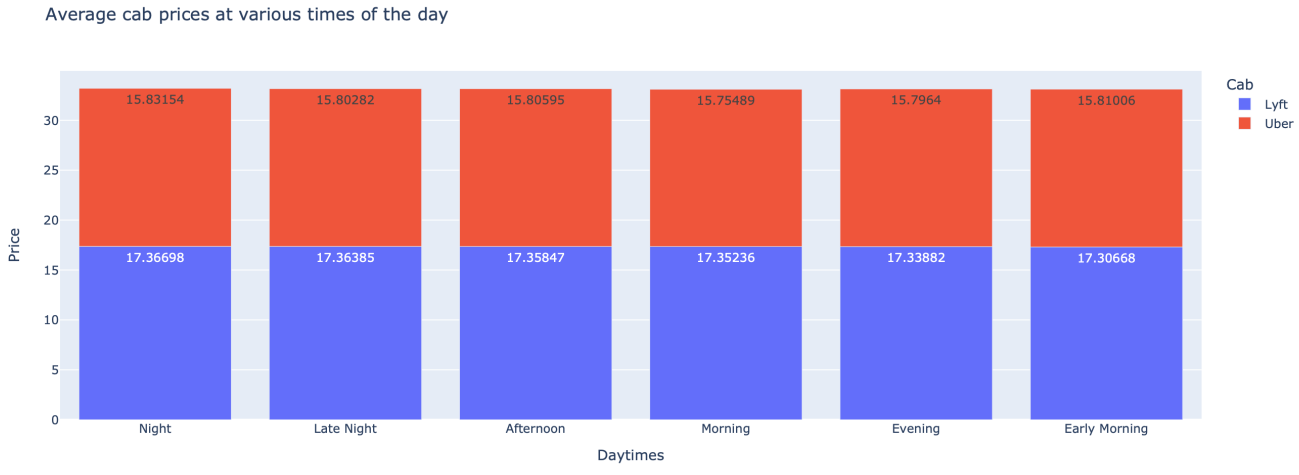
The graph shows that trips of 1-2 and 2-3 miles are the ones for which cabs are used the most frequently. Additionally, the prices are typically the highest for trips lasting 5 to 6 miles.

- ***Maximum cab prices at various times of the day***



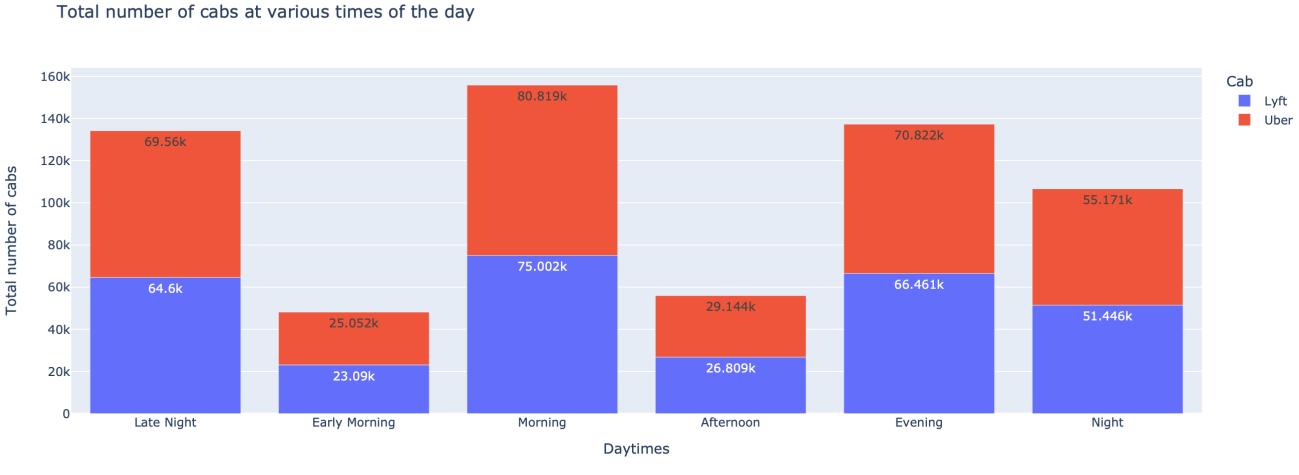
According to the graph, the highest priced charged is by Lyft at various times of the day.

- ***Average cab prices at various times of the day***



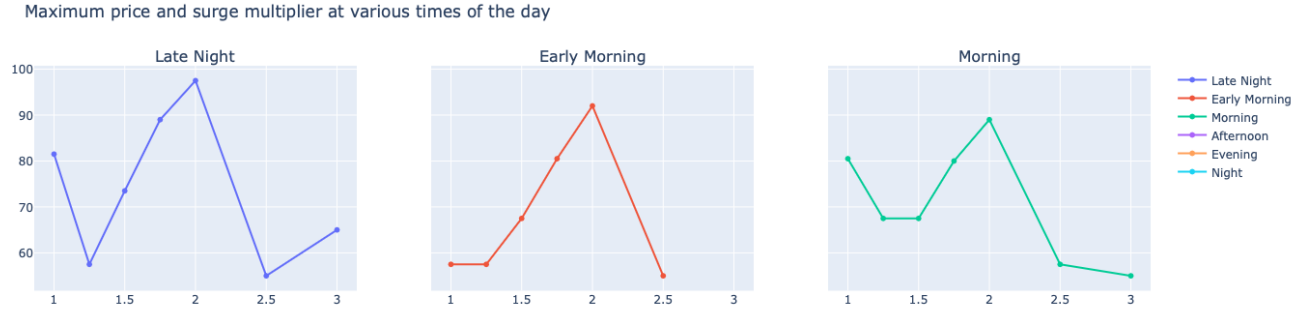
According to the graph, Lyft costs more than Uber at various times of the day.

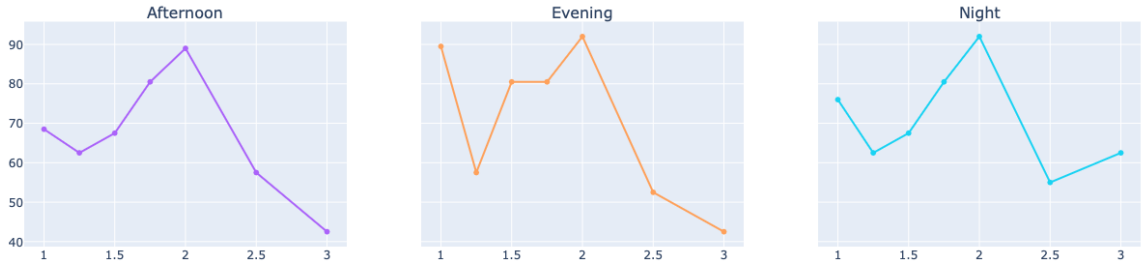
- ***Total number of cabs at various times of the day***



The graph shows that Uber has more cabs available than Lyft at various times of the day. Additionally, early in the morning and late in the afternoon are the hardest times to find a cab.

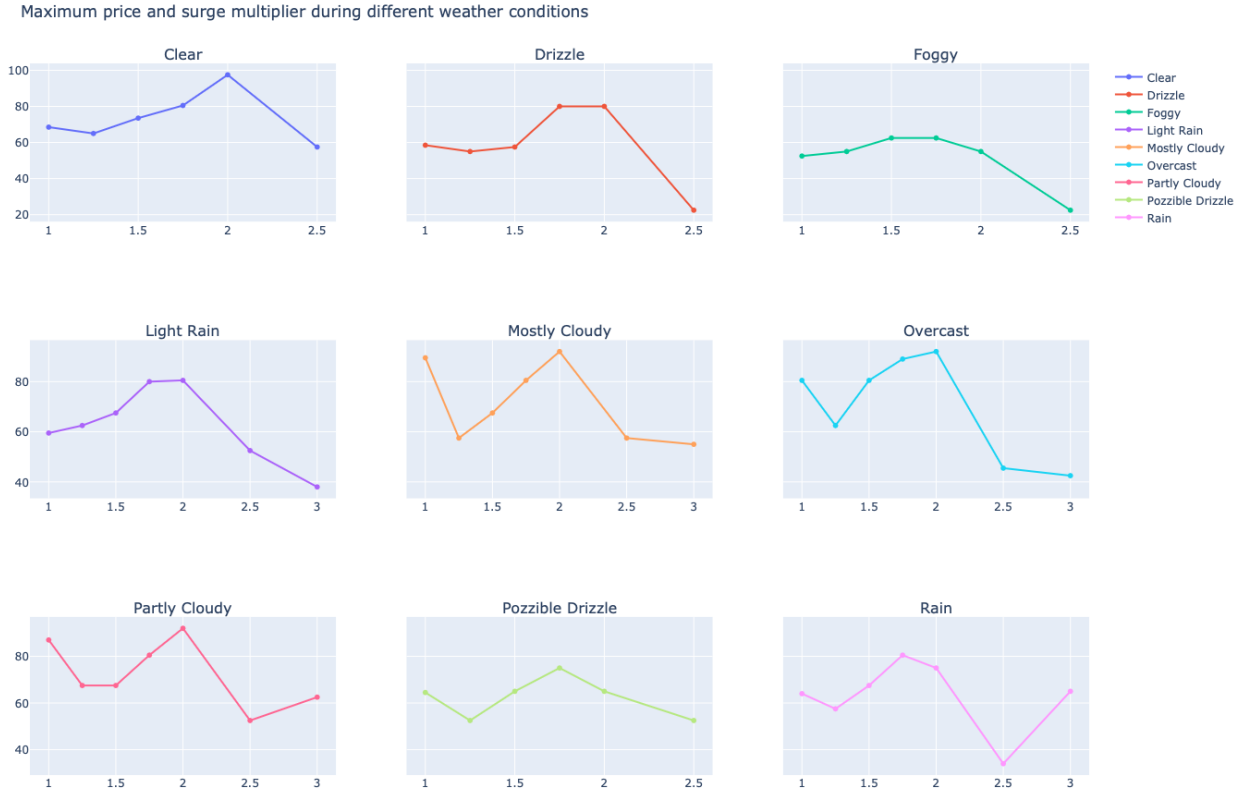
- ***Maximum prices and surge multiplier at various times of the day***





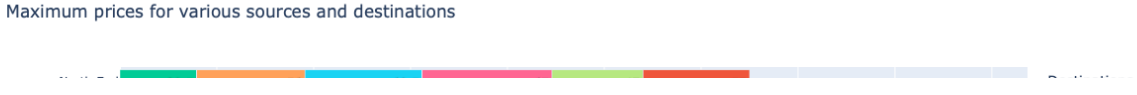
The graph displays the highest price and how it relates to the surge multiplier throughout the day. Around $100 is the biggest amount that is charged during the night.

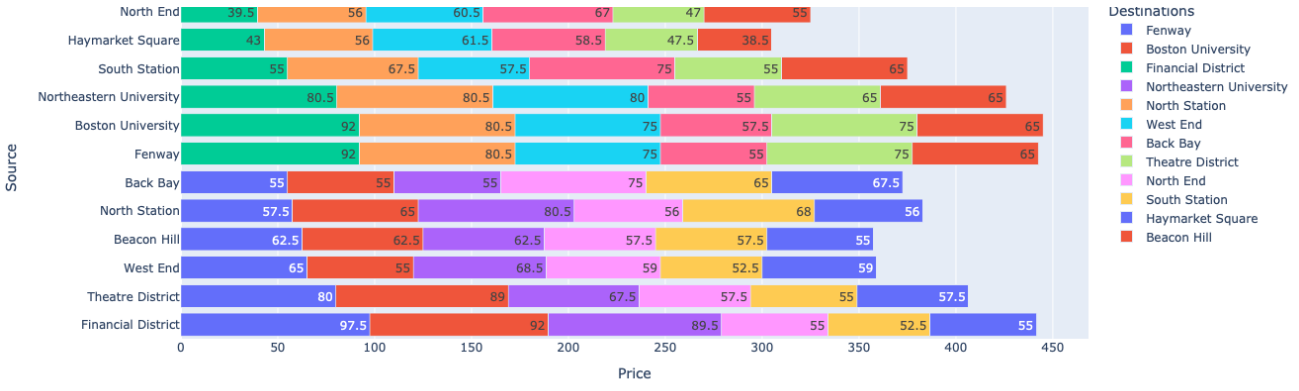
- ***Maximum price and surge multiplier during different weather conditions***



The graph displays the highest price and how it relates to the surge multiplier during different weather conditions.

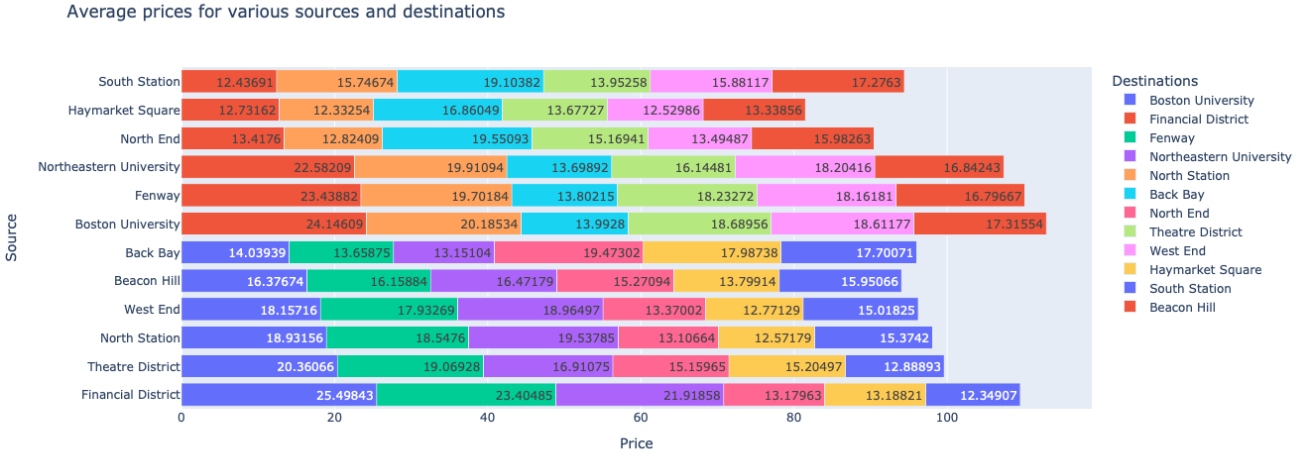
- ***Maximum prices for various sources and destinations***





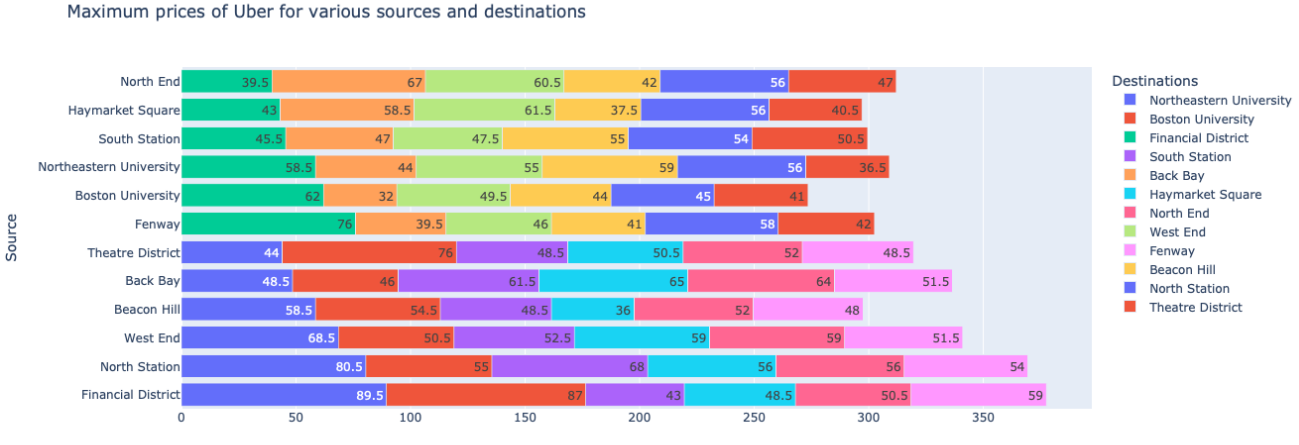
The graph displays the highest rates that Uber and Lyft has ever charged between various sources and destinations. The trip from Financial District to Fenway has the highest maximum fare.

- ***Average prices for various sources and destinations***



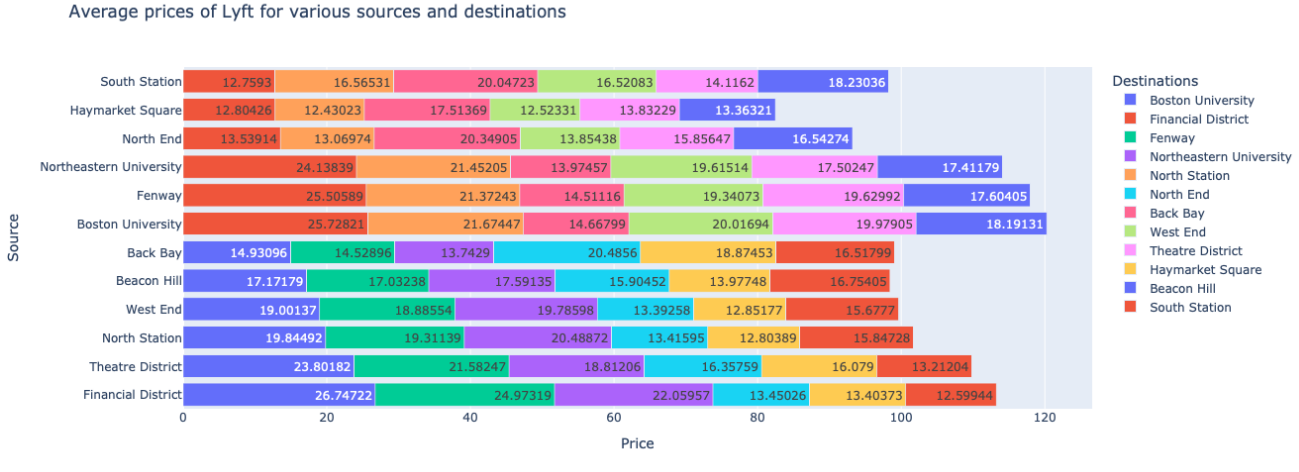
Uber and Lyft's typical rates between different origins and destinations are shown on the graph. The most expensive route is typically from the Financial District to Boston University, while the least expensive is from Haymarket Square to North Station.

- ***Maximum prices of Uber vs Lyft for various sources and destinations***



Uber and Lyft's maximum rates ever charged between different origins and destinations are shown on the graphs. Uber charged a maximum price of $89.5 from Financial District to the Northeastern University whereas Lyft charged a maximum price of $97.5 from Financial District to Fenway.

- ***Average prices of Uber vs Lyft for various sources and destinations***



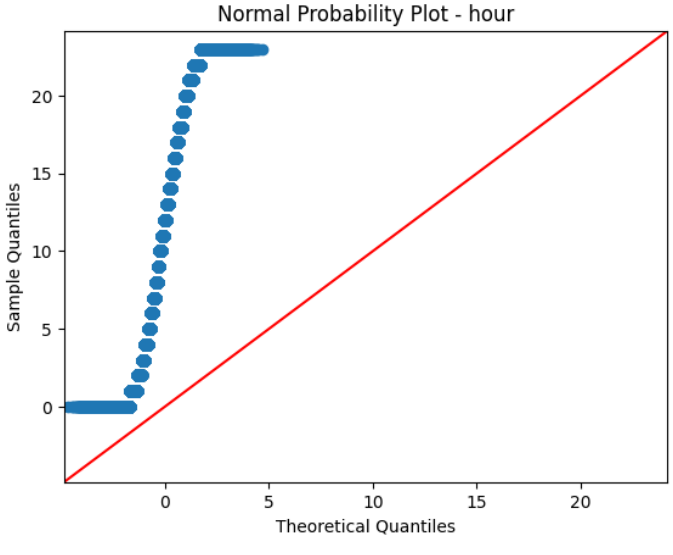
The graphs display the average rates for Uber and Lyft between various origins and destinations. For the same trips, Lyft typically charges more than Uber. For both Uber and Lyft, the most expensive route is from Financial District to Boston University. Additionally, for Uber and Lyft, the cheapest rides are from Haymarket Square to North Station and from Financial District to South Station, respectively.

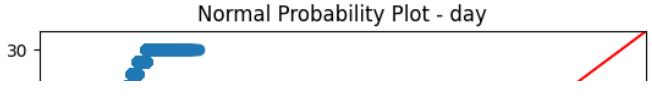
Q-Q plot

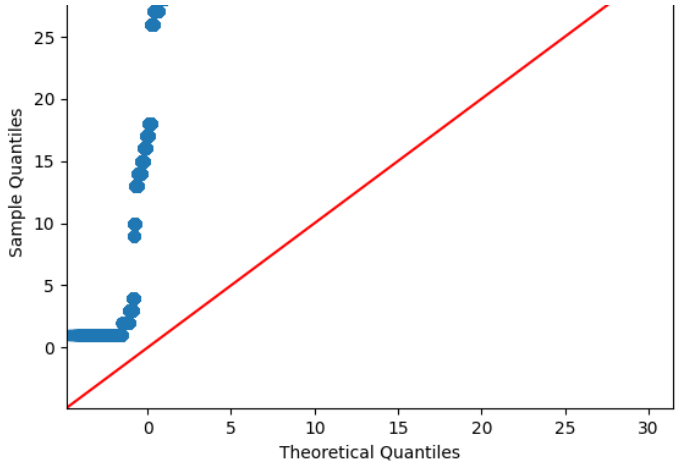
A statistical graphical tool called a Q-Q plot (Quantile-Quantile plot) is used to determine if a dataset conforms to a theoretical distribution. It is being used in this particular case to verify the normal distribution. It contrasts the observed data's quantiles with the quantiles of a typical normal distribution. The data is assumed to have a normal distribution if the data points roughly follow a straight line.

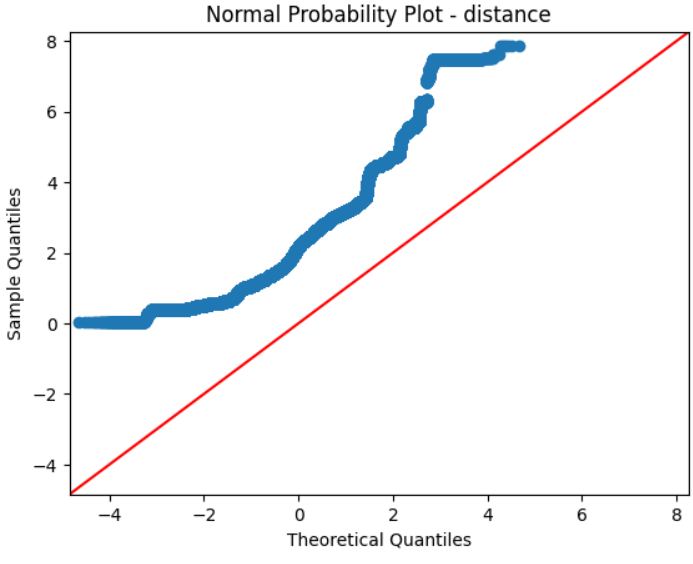
Q-Q plots are an important tool for exploratory data analysis because they provide insight into a dataset's distributional properties and assist in determining whether a given theoretical distribution is suitable for modelling the data.

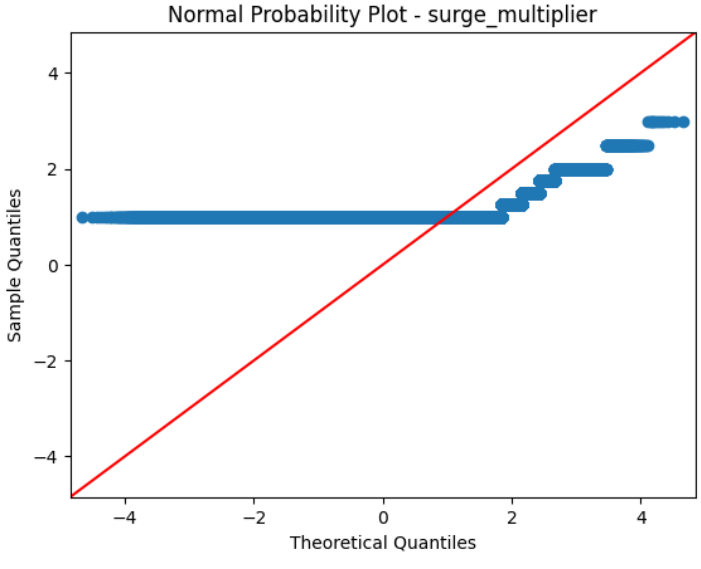
Since the data points do not fall on a straight line, it can be shown from the multiple Q-Q plots displayed against different features that none of the features follow normal distribution.The following graphs represent a few of them.

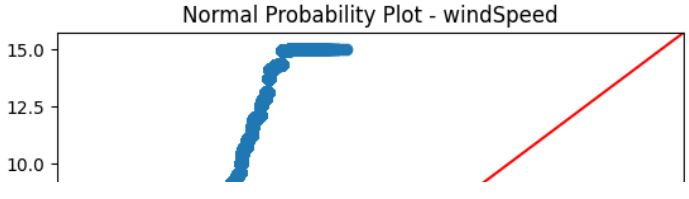


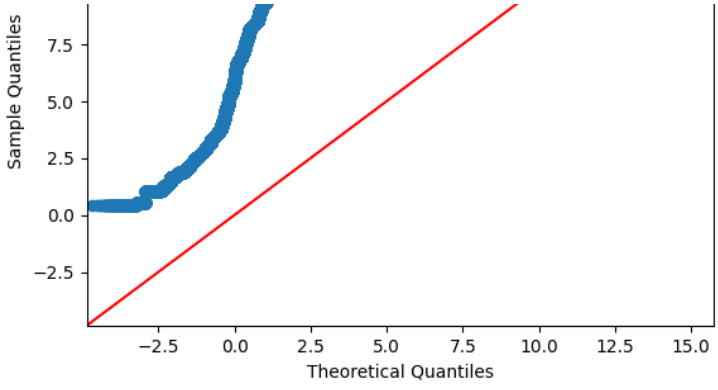



Sample Quantiles
Theoretical Quantiles


Normal Probability Plot - distance
Sample Quantiles
Theoretical Quantiles


Normal Probability Plot - surge_multiplier
Sample Quantiles
Theoretical Quantiles


Normal Probability Plot - windSpeed



Non-parametric tests must be used to identify the best features because the data is not normally distributed, as seen in the graphs above.

Feature Selection

Feature selection is a way of reducing the input variable for the model by using only relevant data in order to reduce overfitting in the model.

We used chi-squared test to obtain relevant categorical features for our project and spearman's rank correlation to obtain relevant numerical features.

- ***SPEARMAN'S RANK CORRELATION for numerical variables***

Spearman's rank correlation coefficient is a statistical measure used to assess the strength and direction of the monotonic relationship between two variables. Unlike Pearson correlation, Spearman's correlation is based on the ranks of the data rather than the actual values.

Calculation of Spearman's Rank Correlation Coefficient (ρ):

The Spearman's rank correlation coefficient (ρ) is calculated using the ranked data. It measures the degree to which the ranks of the two variables are related. The coefficient ranges from -1 to +1, where:

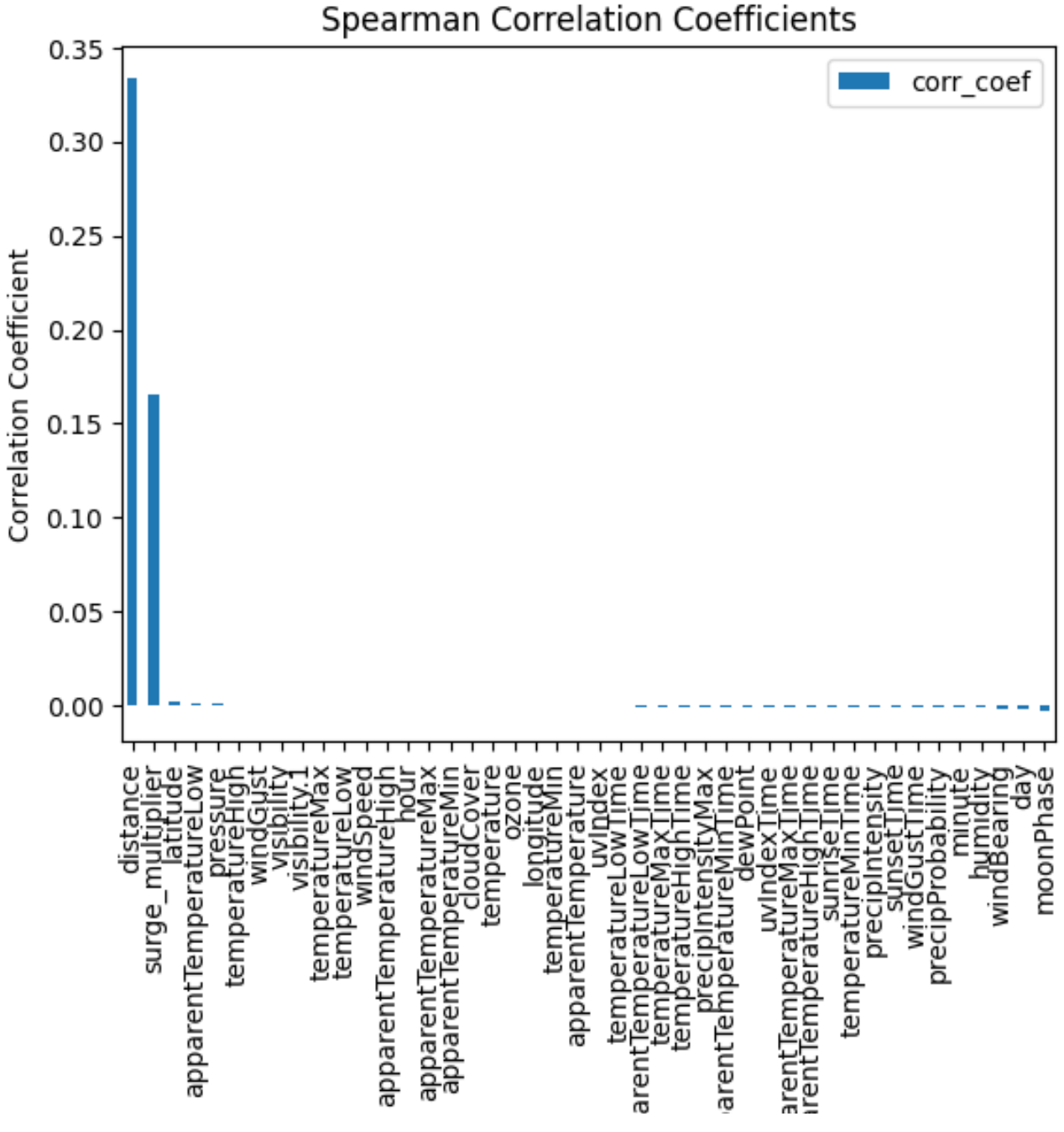
- ρ = +1 indicates a perfect increasing monotonic relationship.

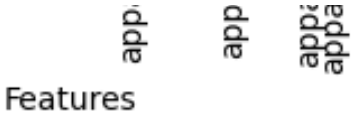
- ρ = -1 indicates a perfect decreasing monotonic relationship.

- ρ = 0 indicates no monotonic relationship.

Below are the selected features for modelling purpose based on the value of their correlation coefficient value with the target feature (i.e the price feature).

	corr_coef
distance	0.333871
surge_multiplier	0.165611
latitude	0.002037
apparentTemperatureLow	0.001378
pressure	0.001282
temperatureHigh	0.000861





- ***CHI Square test for categorical variables***

Since the chi square test is used to ascertain the correlation between categorical features, we must first turn our pricing feature into a categorical before performing the chi-squared test. Here, the chi square statistic and the p value have been calculated as two test statistics.

The chi square statistic measures the discrepancy between the observed frequencies and the expected frequencies under the assumption of independence between the features and the target variable. A higher chi square statistic indicates a stronger association between the variables.

The p-value represents the probability of obtaining a chi-square statistic as extreme as the observed value, assuming the null hypothesis of independence is true. We compared the p-values of different features and used them as a criterion for feature selection. Lower p-values suggest a stronger association between the feature and the price hence more important for our model.

	chi_square
name	2.331125e+06
cab_type	3.890842e+05
source	9.246860e+04
destination	9.066977e+04
minute	8.408397e+03
short_summary	1.179543e+03
icon	8.142800e+02
month	1.589691e+02

	p_value
minute	0.942255
icon	0.932497
short_summary	0.400588
month	0.218778
source	0.000000
destination	0.000000
cab_type	0.000000
name	0.000000

From the above tests, the features for the selection are-

['distance', 'surge_multiplier', 'latitude', 'apparentTemperatueLow', 'pressure', 'temperatueHigh', 'month', 'source', 'destination', 'cab_type', 'short_summary', 'name']

- ***Encoding Categorical variables***

Before performing feature selection we encode categorical data. For this purpose we will be using target guided encoding which basically means that the numerical value given to a feature will depend on the target(example mean or median(if outliers exist). Since a lot of outliers exist in our case therefore we will be using median i.e. for every categorical feature we will replace the categories with median of price feature.

Modelling

In order to start modelling we first normalize and standardize our independent variables in the dataset so that it becomes easy for the model to run predictions and saves calculation space.

After that, we divide our data into x and y where x consists of independent features and y consists of dependent (target) variable. We further use train_test_split to split the data into train and test the data for training the model and testing its metrics respectively.

- ***Standardizing***

We will be using scale function from sklearn.preprocessing to standardize data. Standardizing means data preprocessing such that it has a mean of '0' and standard deviation of '1'.

Now our dataset looks something like this -

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	-1.54064	-0.157905	-2.577771	1.632431	-0.440597	0.132242	0.588364	-1.646155	-1.140607	0.0	-2.039995	-0.502447	1.036986	0.0	0.0	-0.345180	-1.157672
1	-1.54064	-0.157905	-2.577771	1.632431	1.435644	0.663565	-1.415614	-1.313115	0.249000	0.0	-2.039995	-0.502447	1.036986	0.0	0.0	-0.345180	0.102702
2	-1.54064	-0.157905	-2.577771	1.632431	0.064545	-0.215671	0.588364	0.200705	0.412483	0.0	-2.039995	-0.502447	1.036986	0.0	0.0	-0.345180	-0.892330
3	-1.54064	-0.157905	-2.577771	1.632431	-0.079782	-0.603292	0.588364	0.907154	0.698578	0.0	-2.039995	-0.502447	1.036986	0.0	0.0	2.711688	1.893760
4	-1.54064	-0.157905	-2.577771	1.632431	-0.296271	0.126569	0.588364	0.836509	0.575966	0.0	-2.039995	-0.502447	1.036986	0.0	0.0	-0.345180	-0.295311

1) SGD Regressor

The SGDRegressor is a machine learning algorithm that stands for Stochastic Gradient Descent Regressor. It is used for solving regression problems, where the goal is to predict a continuous numerical value based on a set of input features.

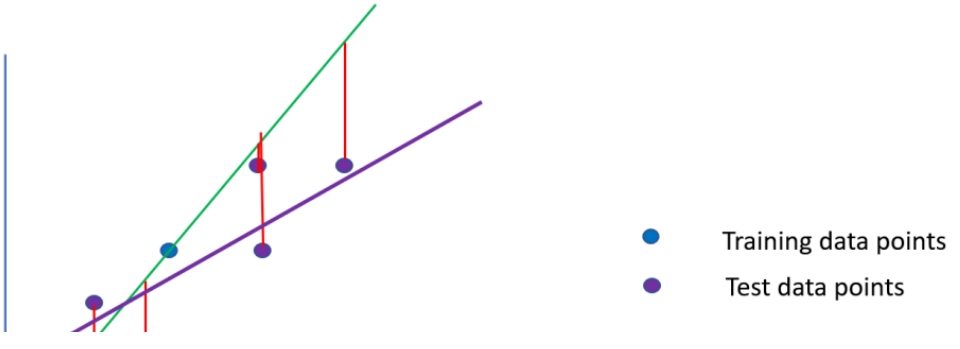
In simple terms, the SGDRegressor works by finding the best-fitting line or curve that can predict the output variable based on the given input features. It iteratively updates the model's coefficients by randomly selecting one data point at a time and adjusting the coefficients based on the prediction error. The process repeats until convergence or a maximum number of iterations is reached. The model is then used to make predictions on new data points. SGDRegressor is memory-efficient and suitable for large datasets .

The following metrics were received which show that our model showed a R2 score of 0.92298.

```
Mean squared error: 0.077446562808395
R2 score -  0.9229875328883878
```

- *R-squared (R2) score also known as the coefficient of determination, is a statistical measure that indicates the proportion of the variance in the dependent variable that can be explained by the independent variables in a regression model.*
- *In simple terms, the R2 score tells us how much of the variation in the target variable (the variable we are trying to predict) can be explained by the predictor variables (the independent variables used in the regression model). It is expressed as a value between 0 and 1, where:*
- *- R2 = 0 indicates that the regression model does not explain any of the variability in the target variable.*
- *- R2 = 1 indicates that the regression model perfectly explains all the variability in the target variable.*

2) Lasso Regression



The Lasso regressor is a machine learning algorithm used for regression tasks, which aims to predict a continuous numerical value based on a set of input features. Lasso stands for *Least Absolute Shrinkage and Selection Operator.*

In simple terms, the Lasso regressor works by finding the best-fitting line or curve that can predict the output variable based on the given input features. It does this by adjusting the coefficients assigned to each feature while minimizing the prediction error. The unique aspect of the Lasso regressor is its ability to perform feature selection and regularization.

The Lasso-Regressor is a linear regression model that uses L1 regularization to perform feature selection and control model complexity. During training, it minimizes the sum of squared residuals while penalizing the absolute values of the coefficients. This encourages some coefficients to become exactly zero, effectively performing automatic feature selection. LassoRegressor is useful for high-dimensional datasets with many irrelevant features like the given dataset.

We will also be comparing lasso regression vs linear regression. Lasso regression with different values of alpha will also be compared.

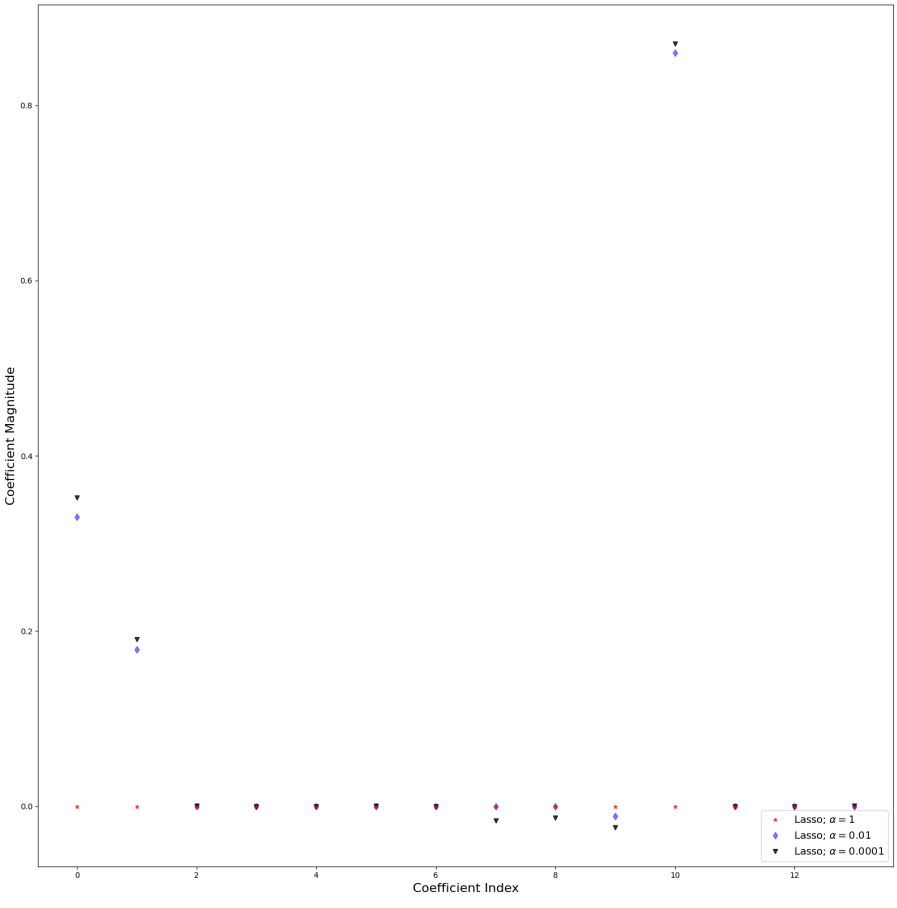
Below are the evaluation metrics for lasso regression model for alpha values 1 , 0.01 and 0.001 respectively and also the evaluation metrics for linear regression for a clear comparison between them -

```
training score: 0.0
test score:  -4.653126062148516e-05
number of features used:  0
training score for alpha=0.01: 0.9245552840496631
test score for alpha =0.01:  0.9251256222499437
number of features used: for alpha =0.01: 4
training score for alpha=0.0001: 0.925263210325928
test score for alpha =0.0001:  0.9258424897526181
number of features used: for alpha =0.0001: 11
LR training score: 0.9252633983032151
LR test score:  0.9258426294107442
```

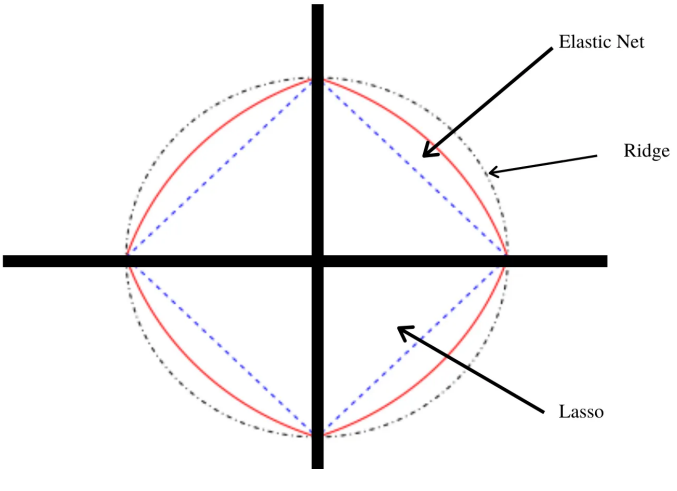
The following metrics were received which show that our model showed a LR test score of 0.9258.

Clearly alpha = 1 performs a lot of regularization to prevent overfitting that it doesn't even use one of the features , hence we have to use lesser values of alpha in order for this model to work appropriately

Now we will be visualizing our model with coefficient index and its magnitude respectively to see which features are given how much importance and also to see the difference in coefficient importance for changing alpha values.



3) Elastic Net Regressor



The ElasticNetRegressor is a linear regression model that combines both L1 (Lasso) and L2 (Ridge) regularization techniques to balance feature selection and model complexity. It works by minimizing the sum of squared residuals with a combination of L1 and L2 penalties.

It is also particularly useful when dealing with datasets that have a large number of features (like given dataset) and potential multicollinearity. By combining L1 and L2 regularization, it provides a flexible approach for controlling model complexity and performing feature selection.

First we performed evaluation on the training dataset of our model and then perform predictions on the test dataset.

Below MAE stands for mean absolute error. It is calculated by taking the absolute difference between each prediction and its corresponding true value, averaging them, and providing a single value that represents the average error. MAE provides a straightforward measure of model accuracy, where lower values indicate better performance.

```
Mean MAE: 0.629 (0.002)
```

The following metrics were received which show that our model showed a R2 score of 0.38122.

```
Root mean squared error -  0.7831711185139474
R2 score -  0.3812222716175493
```

Hyperparameter Tuning:

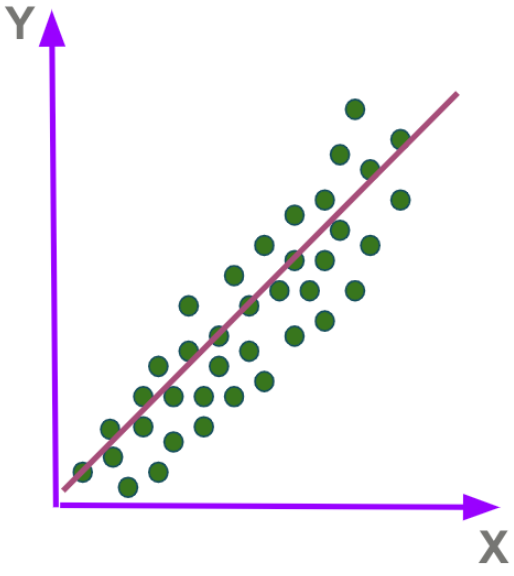
Since the R2 score is very less we will perform hyperparameter tuning to get the best value of alpha and lambda respectively. The scikit-learn library also provides a built-in version of the algorithm that automatically finds good hyperparameters via the ElasticNetCV class.

Using the tuned values of hyperparameters alpha and lambda which came out to be 0.0001 and 0.88 respectively, in the actual model boosts up the accuracy.

The following metrics were received again which show that our model showed an improved R2 score of 0.92439.

```
Root mean squared error -  0.27375197962746767
R2 score -  0.9243975475017498
```

4) Linear Regression



Linear regression is a statistical modeling technique used to understand the relationship between a dependent variable (the variable we want to predict) and one or more independent variables (the variables we use to make predictions). It assumes that this relationship can be represented by a straight line.

The process of fitting a linear regression model involves the following steps:

1. Data: Collecting data that includes the values of the dependent variable and one or more independent variables for each observation.

2. Scatterplot: Visualizing the data by creating a scatterplot, with the independent variable(s) on the x-axis and the dependent variable on the y-axis. This helps to understand the general relationship between the variables.

3. Line fitting: Finding the line that best represents the relationship between the variables. The line is defined by the equation: y = mx + b, where y is the dependent variable, x is the independent variable, m is the slope of the line, and b is the y-intercept (the value of y when x is zero).

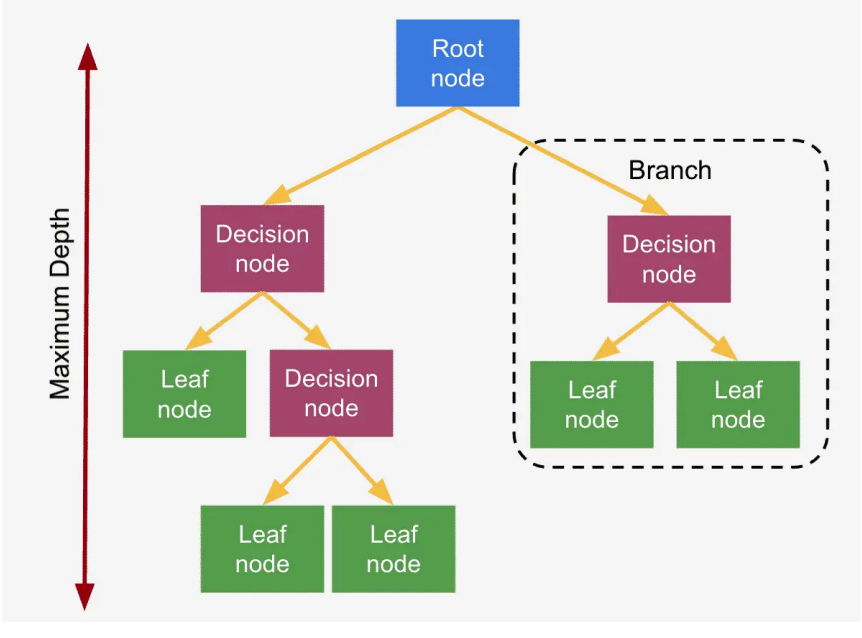
4. Coefficient estimation: Estimating the values of the coefficients (m and b) that minimize the difference between the predicted values and the actual values of the dependent variable. This is typically done using a method called least squares, which calculates the line that minimizes the sum of squared differences between the predicted and actual values.

5. Prediction: Once the coefficients are estimated, the linear regression model can be used to make predictions. Given the values of the independent variable(s), the model can calculate the predicted value of the dependent variable.

After following the above steps, the necessary metrics were received which show that our model showed a R2 score of 0.92551.

```
[122] X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

      lr = LinearRegression()
      lr.fit(X_train, y_train)

      y_pred = lr.predict(X_test)

      r2 = r2_score(y_test, y_pred)
      print("R-squared score: ", r2)

      R-squared score:  0.9255118789662787

[123] mse=mean_squared_error(y_test,y_pred)
      print("Mean squared error: ",mse )

      Mean squared error:  0.07469107958549008
```

5) Decision Tree Regression



Decision tree regression is a machine learning algorithm used for regression tasks, which involves predicting a continuous numerical value based on a set of input features. It utilizes a decision tree, a flowchart-like structure, to make predictions by splitting the data into smaller and more manageable subsets.

In simple terms, decision tree regression works by creating a tree-like model of decisions based on the input features. The tree starts with a single node,

which represents the entire dataset. It then splits the data at each node based on a chosen feature and threshold value, creating branches and new nodes. This splitting process continues recursively until a stopping criterion is met, such as reaching a maximum depth or a minimum number of data points in a node.

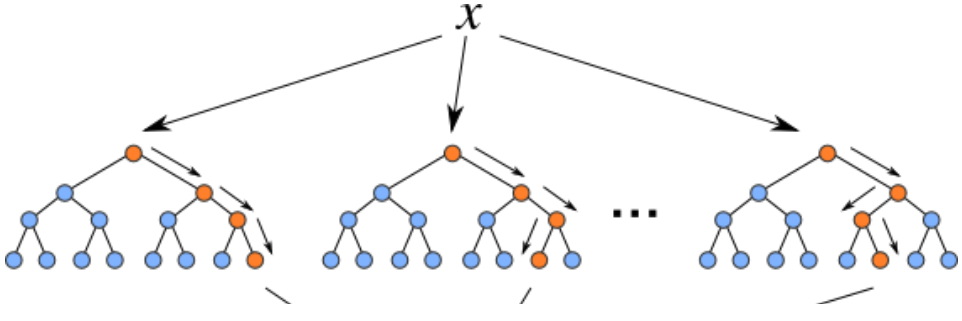
One of the advantages of decision tree regression is its interpretability. The resulting tree structure can be visualized and understood, as it provides a clear set of rules for making predictions. However, decision trees can sometimes overfit the training data, meaning they become overly complex and fail to generalize well to unseen data. Techniques such as pruning, setting maximum depths, or using ensemble methods like random forests can help alleviate this issue.

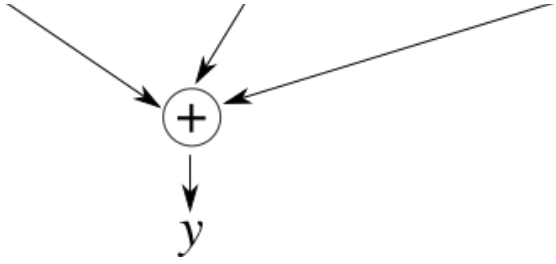
Overall, decision tree regression is a powerful and intuitive algorithm for regression tasks. It can handle various types of features, capture non-linear relationships, and provide interpretable models.

The following metrics were received again which show that our model showed an R2 score of 0.96516.

```
[124] reg = DecisionTreeRegressor(criterion="squared_error",max_depth=10,min_samples_split= 10,
      random_state=0)

      reg.fit(X_train,y_train)

      y_pred = reg.predict (X_test)
      y_pred

      array([-0.89170142, -0.64203908, -0.99431546, ..., -0.01425853,
             -0.80599039, -0.5195177 ])

[125] mse=mean_squared_error(y_test,y_pred)
      print("Mean squared error: ",mse )

      r2 = r2_score(y_test, y_pred)
      print("R-squared score: ", r2)

      Mean squared error:  0.03493449516034717
      R-squared score:  0.965160432568424
```

6) Random Forest





It combines multiple decision trees to make predictions. Each tree in the random forest is trained on a random subset of the training data, and the final prediction is determined by aggregating the predictions of all individual trees.

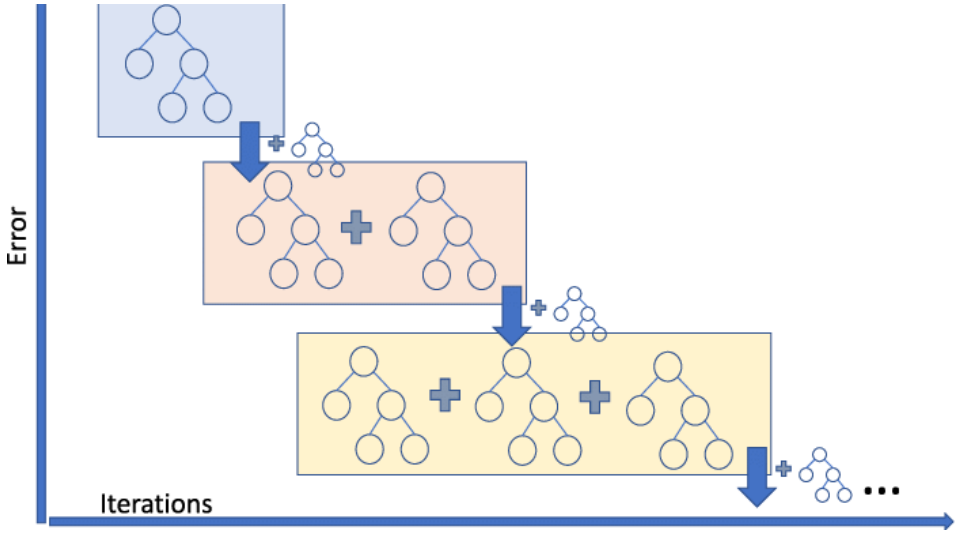
It has high accuracy, robustness to outliers, feature importance and can handle large datasets.

The following metrics were received again which show that our model showed an R2 score of 0.96238.

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

rf = RandomForestRegressor(n_estimators=100, random_state=42)

rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("Mean squared error: ", mse)

r2 = r2_score(y_test, y_pred)
print('R2 score - ', r2)
```

```
<ipython-input-126-754827d96ce9>:5: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

Mean squared error:  0.03771668633481727
R2 score -  0.9623858014599571
```

7) Gradient Boosting Regressor



Gradient boosting is a technique that combines multiple weak predictive models, typically decision trees, to create a stronger and more accurate model. The basic idea behind gradient boosting is to iteratively build a sequence of models, where each model tries to correct the mistakes made by the previous model.

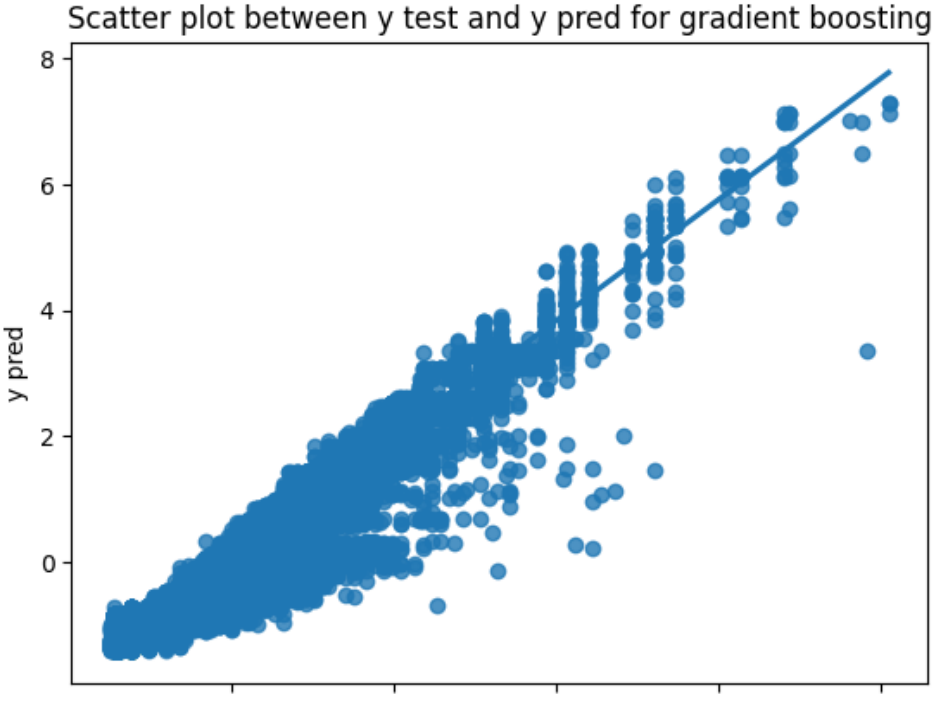
Gradient boosting is capable of capturing nonlinear relationships between features and the target variable and avoids overfitting.

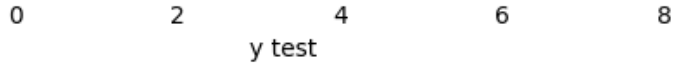
The following metrics were received again which show that our model showed an R2 score of 0.96400.

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)


gb_reg = GradientBoostingRegressor( learning_rate=0.1, n_estimators=100, max_depth=3)
gb_reg.fit(X_train, y_train)


y_pred = gb_reg.predict(X_test)


rmse = mean_squared_error(y_test, y_pred, squared=False)
print("Root Mean Squared Error:", rmse)

r2 = r2_score(y_test, y_pred)
print("R-squared score: ", r2)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_gb.py:437: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().

Root Mean Squared Error: 0.18997219400950172
R-squared score:  0.9640086315560873
```

The model is more accurate the more points there are on the line or extremely close to it.





Summary

The EDA and price prediction project helped in comparing Uber and Lyft in the Boston city. Through exploratory data analysis, several key insights were covered, shedding light on various factors influencing the pricing strategies of Uber and Lyft. Additionally, various predictive models were developed to estimate the fare prices based on relevant features.

During the EDA phase, it was observed that both Uber and Lyft exhibited similar pricing patterns. Further analysis indicated that the type of ride (such as UberX, Uber Black, Lux Black XL, etc.) significantly impacted the fare prices. Moreover, factors like distance, weather conditions etc. were found to influence the pricing structure.

Using the insights gained from the EDA, various predictive models for fare price estimation were developed with R2 score ranging from 0.92-0.965 which showed our models were successful in predicting prices to a greater extent. The models incorporated features such as distance, source, destination etc. to predict the fare accurately.

In conclusion, the EDA and price prediction project comparing Uber and Lyft provided valuable insights into the pricing strategies and dynamics of these ride-sharing platforms. The analysis highlighted the influence of various factors on fare prices and revealed differences in surge pricing and geographic variations between the two services. The developed predictive model enables users to estimate ride fares, contributing to better planning and decision-making for ride-sharing services.