Project Report On Cab Fare Prediction

By Yamini Peddireddi

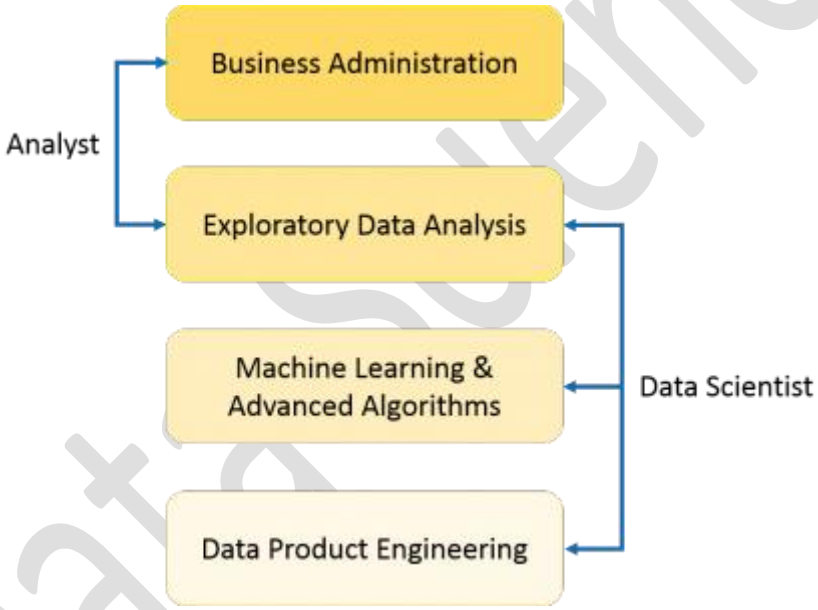
Index

Chapter 1: Introduction

1.1. What is Data Science

Data Science is a blend of various tools, algorithms, and machine learning principles with the goal to discover hidden patterns from the raw data.

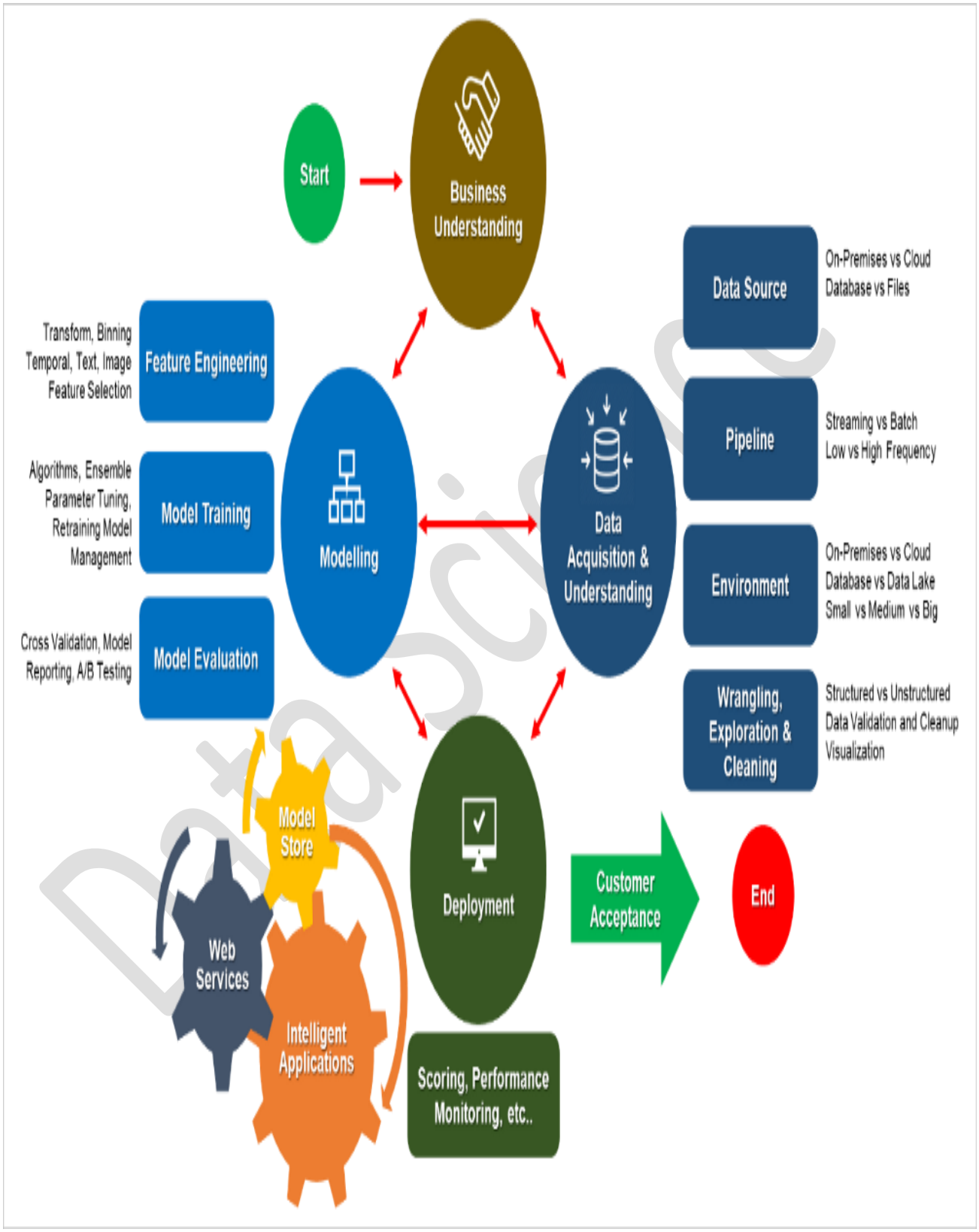
This is different from what used to be normally done. Data Science is the future of Artificial Intelligence. Therefore, it is very important to understand what is Data Science and how can it add value to your business.



Data science is the study of where information comes from, what it represents and how it can be turned into a valuable resource in the creation of business and IT Strategies Mining large amounts of structured and unstructured data to identify patterns can help an organization rein in costs, increase efficiencies, recognize new market opportunities and increase the organization's competitive advantage.

The data science field employs mathematics, statistics and computer science disciplines, and incorporates techniques like machine learning, cluster analysis, data mining and visualization.

1.2. Problem solving in Data Science



From the above diagrams it is clear about how to proceed to solve a problem in data science. Data science is a revolution to the fields and gives a lot of insights and transformative approach in problem solving for many problems. As the data is getting increased day by day in every field, organization and industry.

Chapter 2. Methodology

2.1. Problem statement

For example, I have cab rental start-up company. I have successfully run the pilot project and now want to launch your cab service across the country. I have collected the historical data from your pilot project and now have a requirement to apply analytics for fare prediction. I need to design a system that predicts the fare amount for a cab ride in the city.

2.2. Steps involved in problem solving

Here the problem technique is to design a system through which cab fare prediction is done for the given data or for next few weeks or days or seasons as per the requirement. So here requires the optimal solution for cab fare prediction in the city.

The steps taken in achieving or designing or solving this data science problem are:

- Data cleaning and exploration
- Outlier analysis
- Data understanding
- Feature engineering
- Data visualization
- Feature selection
- Establishing train and validation data
- Model Building
- Hyperparameter tuning
- Model Evaluation
- Prediction on test data

2.3. Challenges

The challenges in solving the data science problem:

- Data collection
- Data limitation
- Manipulated data
- Security

Chapter 3. Data Analysis

3.1. Data

The data collected part contains 7 features and the dataset contains completely 16067 rows of data and this data consists of data from 2009-2015 cab fare data through which test data or any other data can be predicted.

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_cour
0	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1.
1	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1.
2	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2.
3	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	1.
4	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1.

3.2. Features in the dataset

Number of attributes:

- pickup_datetime - timestamp value indicating when the cab ride started.
- pickup_longitude - float for longitude coordinate of where the cab ride started.
- pickup_latitude - float for latitude coordinate of where the cab ride started.
- dropoff_longitude - float for longitude coordinate of where the cab ride ended.
- dropoff_latitude - float for latitude coordinate of where the cab ride ended.
- passenger_count - an integer indicating the number of passengers in the cab ride.

3.3. Data exploration and cleaning

	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	16067.000000	16067.000000	16067.000000	16067.000000	16012.000000
mean	-72.462787	39.914725	-72.462328	39.897906	2.625070
std	10.578384	6.826587	10.575062	6.187087	60.844122
min	-74.438233	-74.006893	-74.429332	-74.006377	0.000000
25%	-73.992156	40.734927	-73.991182	40.734651	1.000000
50%	-73.981698	40.752603	-73.980172	40.753567	1.000000
75%	-73.966838	40.767381	-73.963643	40.768013	2.000000
max	40.766125	401.083332	40.802437	41.366138	5345.000000

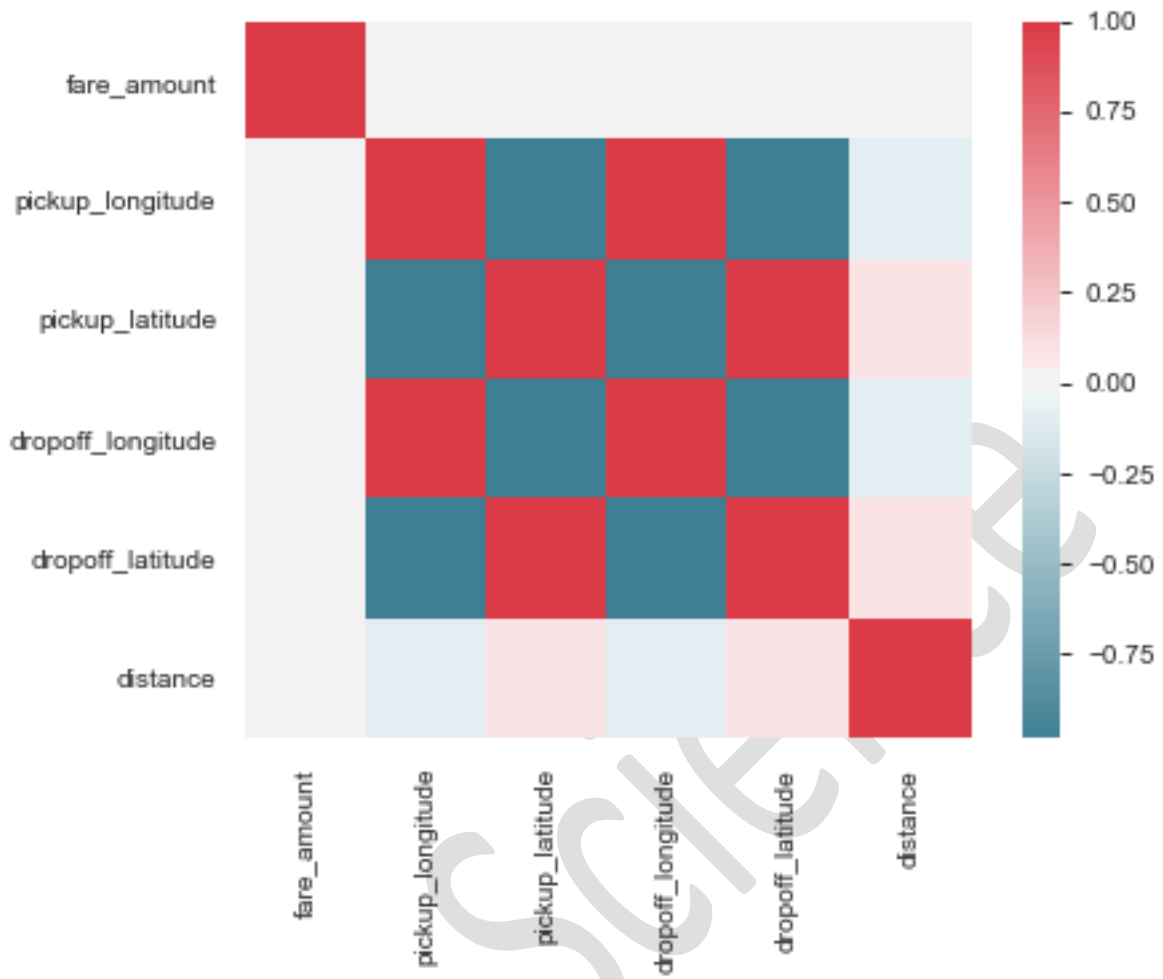
There are missing values in the passenger count and fare amount. I just removed that rows.

3.4. Data and new columns

After exploring the data, we find that there are 7 features in the dataset which impacts the dependent variable fare amount. Fare amount can be anything and is of numerical type so it is a regression problem.

There is a pickup date time column which can be further divided to many columns as it has lot of details in one column. So, I split that data into the columns Year, month, date, Hour, minute, Day of week and Day of year. Further I dropped the column pick up date time as I have taken the required data from that feature and that is no longer useful now. If you allow that data for further solving it show multicollinearity in the data which is not desired for solving the problem and increases the error rate. So that is removed.

Some correlation between the variables can be seen from the below diagram,



3.5. Feature engineering

The Earth is round but big, so we can consider it flat for short distances. But, even though the circumference of the Earth is about 40,000 kilometres, flat-Earth formulas for calculating the distance between two points start showing noticeable errors when the distance is more than about 20 kilometres. Therefore, calculating distances on a sphere needs to consider spherical geometry, the study of shapes on the surface of a sphere.

Spherical geometry considers spherical trigonometry which deals with relationships between trigonometric functions to calculate the sides and angles of spherical polygons. These spherical polygons are defined by a number of intersecting great circles on a sphere. Some rules found in spherical geometry include:

- There are no parallel lines.
- Straight lines are great circles, so any two lines meet in two points.
- The angle between two lines is the angle between the planes of the corresponding great circles.

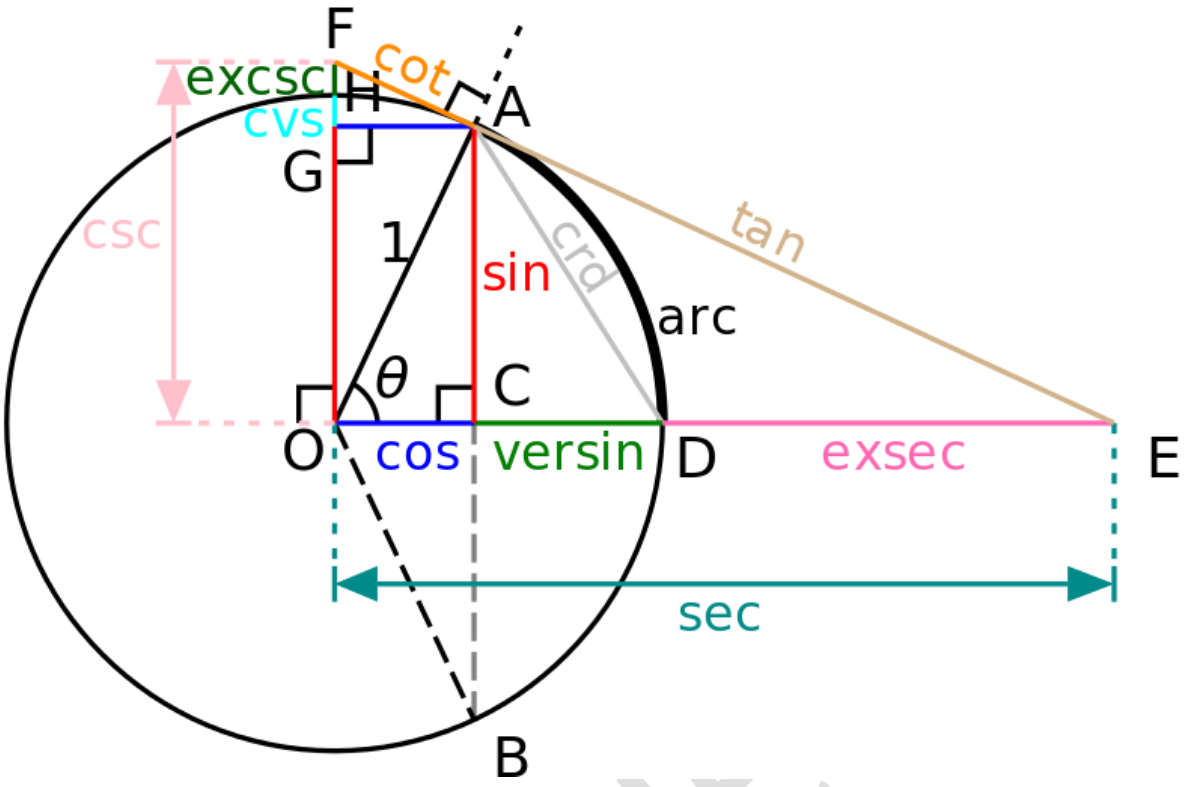
The Haversine

The additional trigonometric functions
are *versine*, *haversine*, *coversine*, *hacoversine*, *exsecant*, and *excosecant*. All of these can be expressed simply in terms of the more familiar trigonometric functions. For example, haversine(θ) = $\sin^2(\theta/2)$.

The haversine formula is a very accurate way of computing distances between two points on the surface of a sphere using the latitude and longitude of the two points. The haversine formula is a re-formulation of the spherical law of cosines, but the formulation in terms of haversines is more useful for small angles and distances.

One of the primary applications of trigonometry was navigation, and certain commonly used navigational formulas are stated most simply in terms of these archaic function names. But you might ask, why not just simplify everything down to sines and cosines? The functions listed above were from a time without calculators, or efficient computer processors, when the user calculated angles and direction by hand using log tables, every named function took appreciable effort to evaluate. The point of these functions is if a table simply combines two common operations into one function, it probably made navigational calculations on a rocking ship more efficient.

These function names have a simple naming pattern, in this example, the "Ha" in "Haversine" stands for "half versed sine" where haversin(θ) = versin(θ)/2.



The following equation where ϕ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km) is how we translate the above formula to include latitude and longitude coordinates. Note that angles need to be in radians to pass to trig functions:

$a = \sin^2(\phi B - \phi A/2) + \cos \phi A * \cos \phi B * \sin^2(\lambda B - \lambda A/2)$

$c = 2 * \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$

$d = R \cdot c$

- Using the above formula new feature distance is introduced which plays a key role in cab fare prediction the city.
- This is applied on both training and test datasets.
- Next the features Month, date, Hour, minute, Day of week and Day of year have been further used for making new column which gives much more favourable features for predicting the cab fare.
- Month is divided into seasons, Hour is divided into Time of day (morning, noon, night), Day of week divided into Week type (Week day and week end), Date is divided and created new column Week in month, Month Days.
- One hot encoded the categorical variables

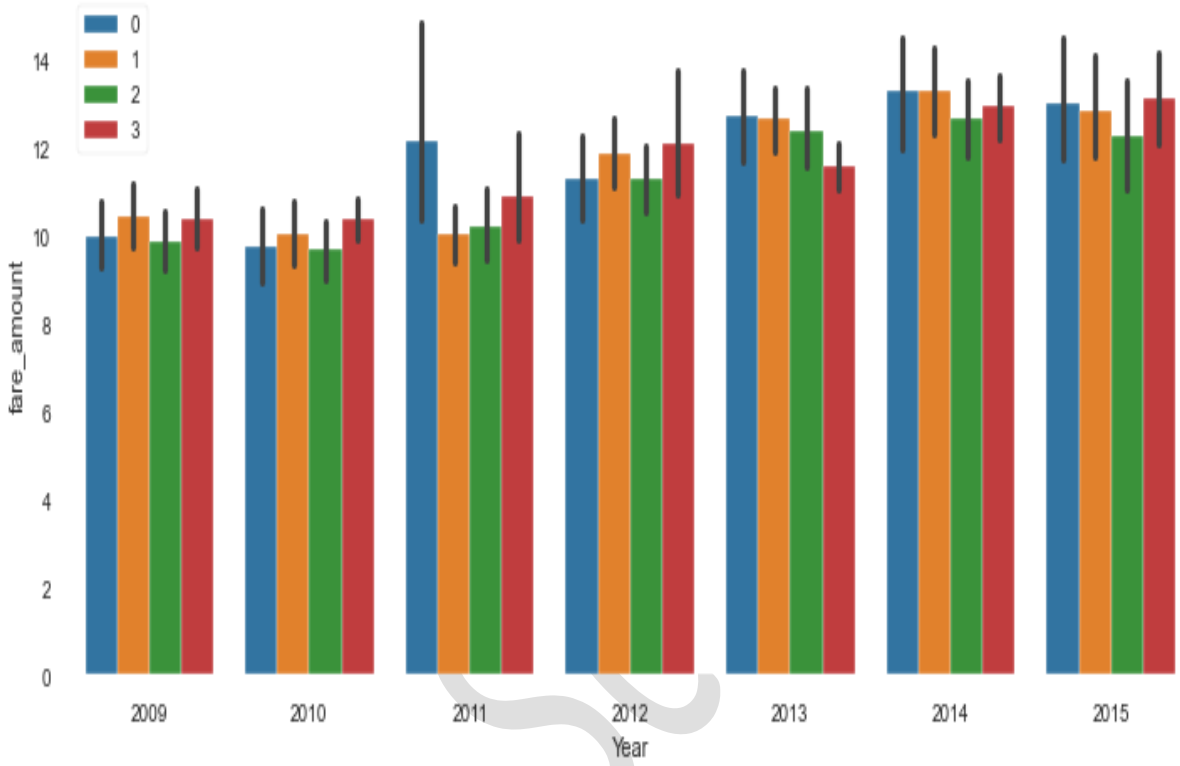
3.6. Feature selection

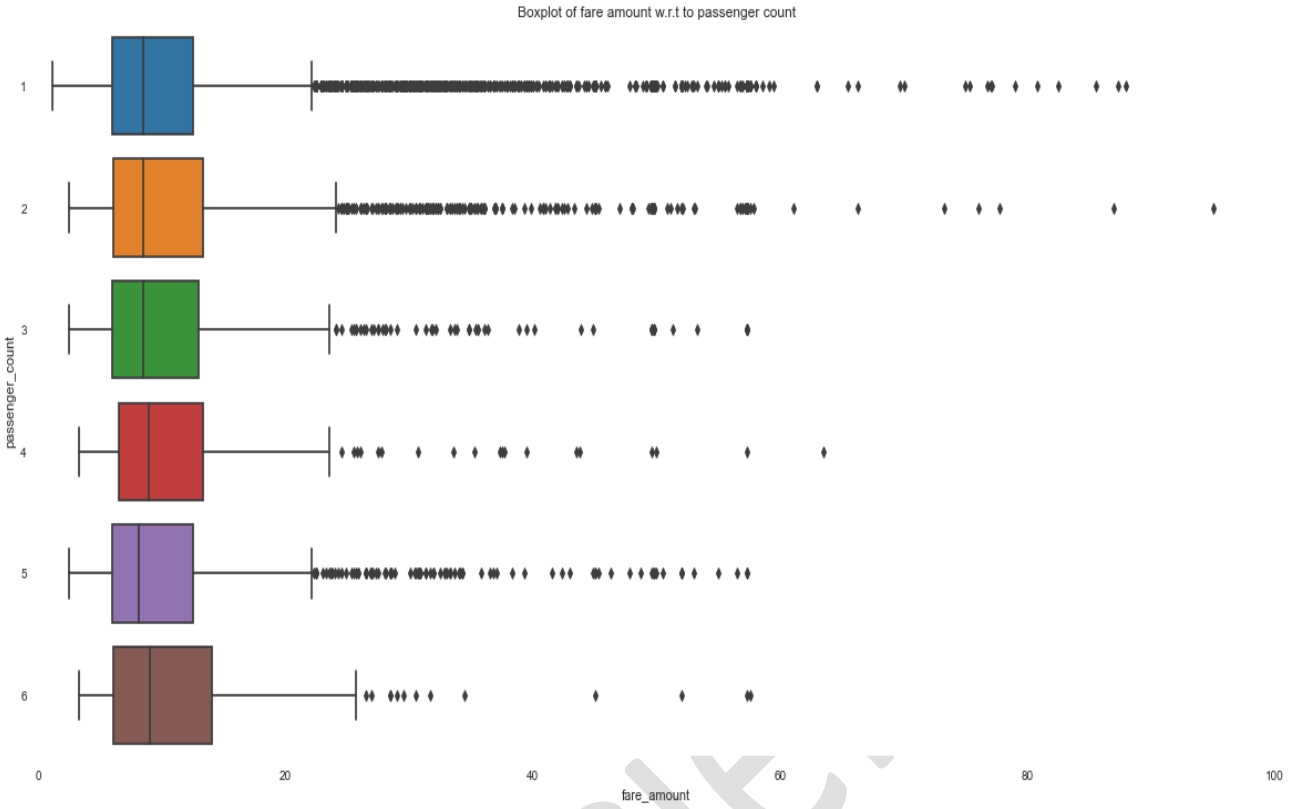
In Feature selection the important attributes from the dataset are extracted removing the redundant columns.

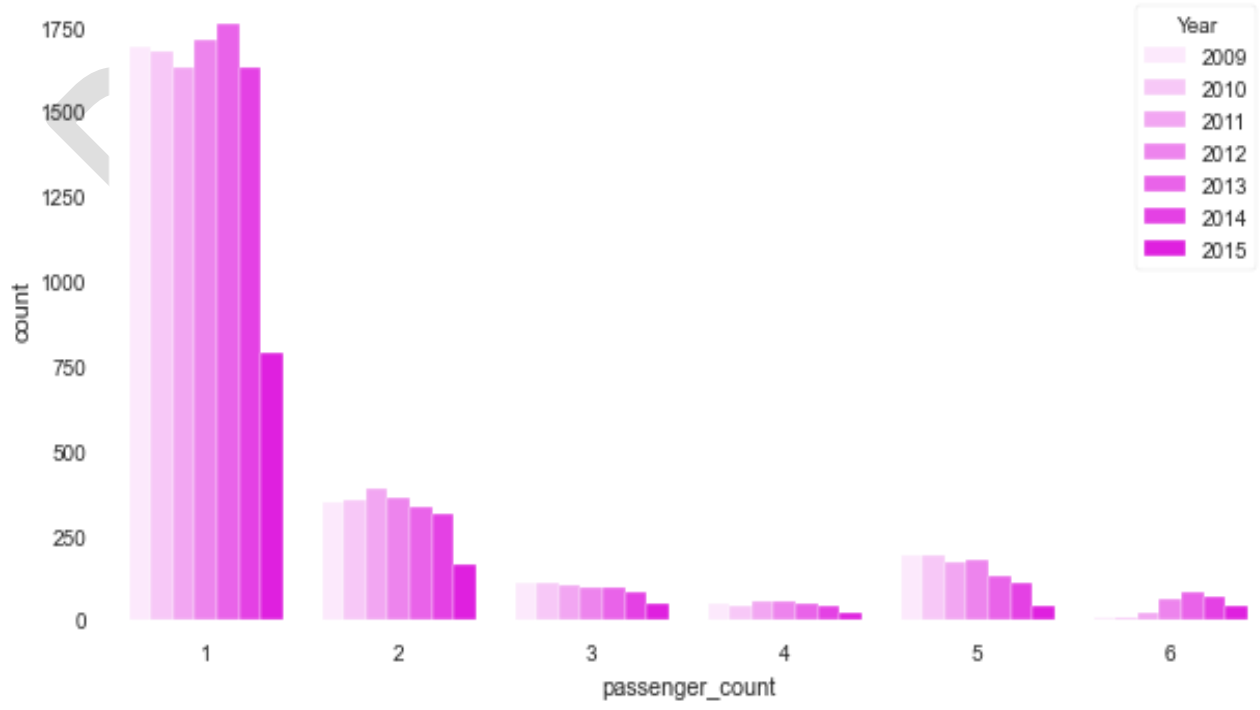
- Here for the model building latitude and longitude columns are dropped as distance gives the same information for predicting the fare amount and is much more useful feature than any other
- The final columns after feature selection and applying all other techniques are

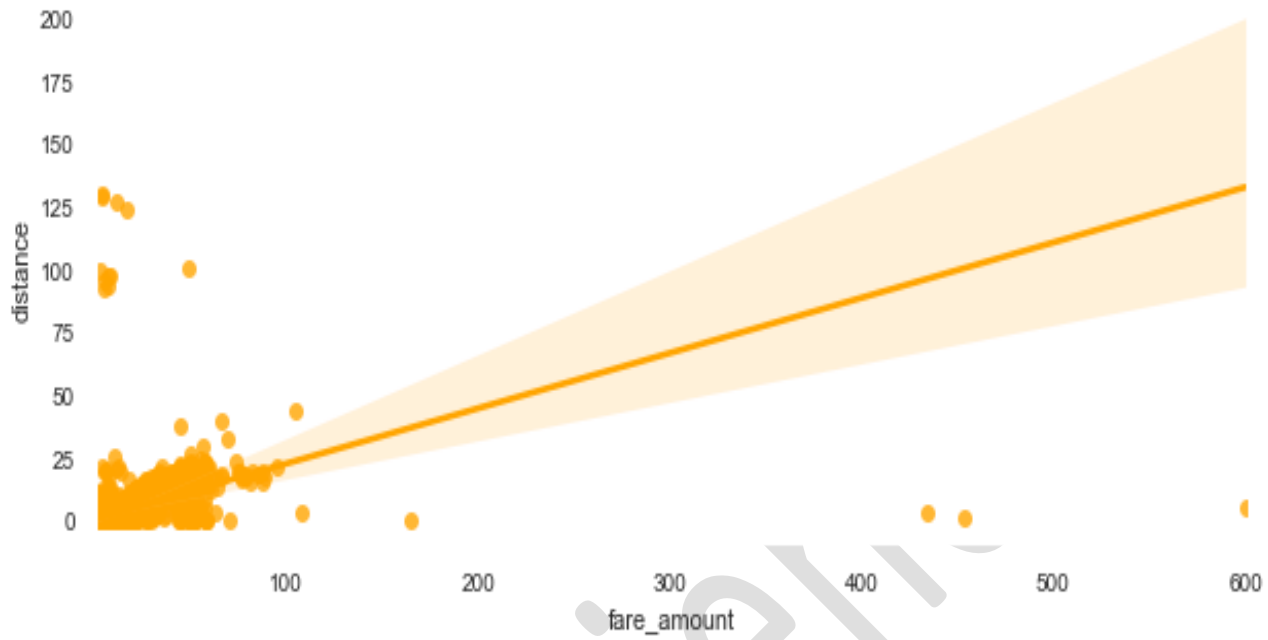
a. distance float64
b. passenger_count_2 uint8
c. passenger_count_3 uint8
d. passenger_count_4 uint8
e. passenger_count_5 uint8
f. passenger_count_6 uint8
g. Year_2010 uint8
h. Year_2011 uint8
i. Year_2012 uint8
j. Year_2013 uint8
k. Year_2014 uint8
l. Year_2015 uint8
m. Time_of_day_1 uint8
n. Time_of_day_2 uint8
o. Time_of_day_3 uint8
p. Season_1 uint8
q. Season_2 uint8
r. Season_3 uint8
s. Weektype_1 uint8
t. WeekinMonth_1 uint8
u. WeekinMonth_2 uint8
v. WeekinMonth_3 uint8
w. MonDays_1 uint8

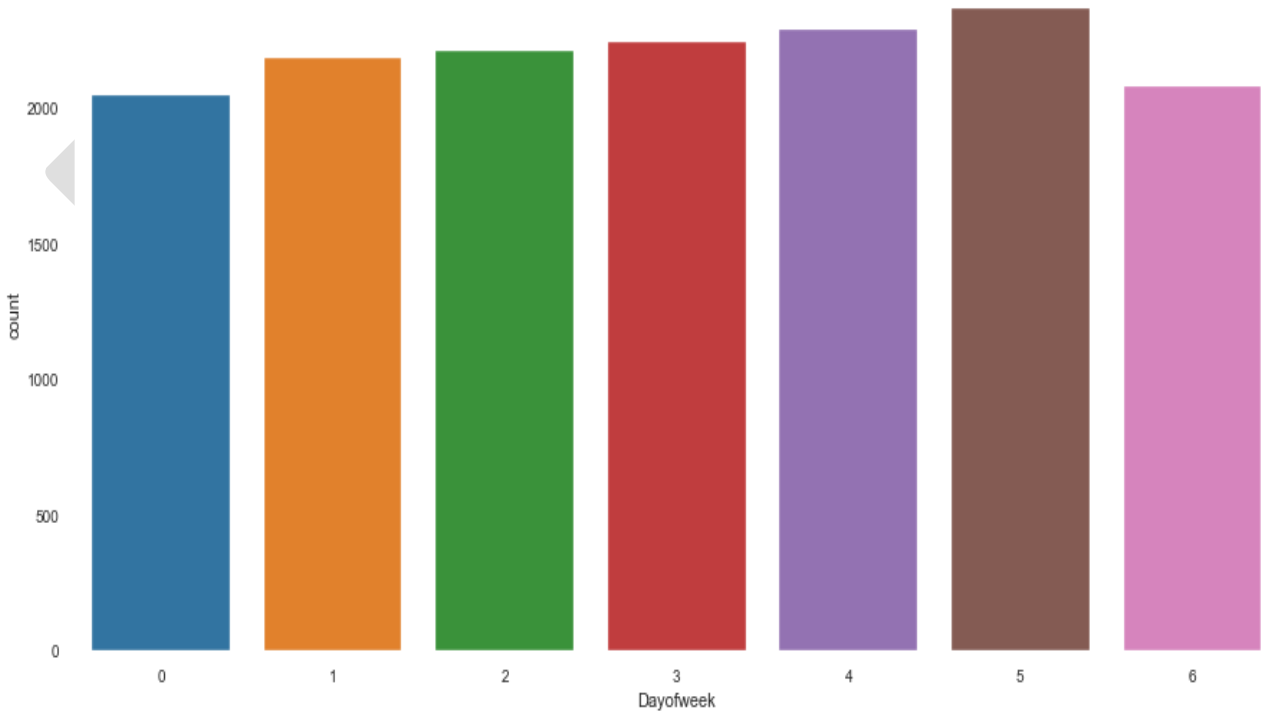
Chapter 4: Data visualization




Boxplot of fare amount w.r.t to passenger count
passenger_count
1
2
3
4
5
6
0
20
40
60
80
100
fare_amount


Year
2009
2010
2011
2012
2013
2014
2015
count
1750
1500
1250
1000
750
500
250
0
1
2
3
4
5
6
passenger_count


200
175
150
125
100
75
50
25
0
distance
100
200
300
400
500
600
fare_amount


2000
1500
1000
500
0
count
0
1
2
3
4
5
6
Dayofweek

Chapter 5: Modelling

5.1. Train and validation split

Now the complete data is explored, filtered, engineered, selected and extracted all the required variables and made the data more useful for prediction of fare amount. Now this data is further split into train and validation split for knowing the performance of the model after training so that we can use on any data also for the test set prediction.

For that sklearn package train_test_split library is used and divided the data into train and validation set.

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)

Here x contains only independent variables and y contains only dependent variable.

After this model building is made with the following models for the regression dataset

5.2. Linear regression

In general, regression is a statistical method that estimates relationships between variables. Classification also attempts to find relationships between variables, with the main difference between classification and regression being the output of the model.

In a regression task, the output variable is numerical or continuous in nature, while for classification tasks the output variable is categorical or discrete in nature. If a variable is categorical it means that there is a finite/discrete number of groups or categories the variable can fit into.

continuous variables will have an infinite number of values between any two variables. The difference between two given numbers can be represented as an infinite number of ways, writing out ever longer decimals. This means that even things like date and time measurements can be considered continuous variables if the measurements are not put into discrete categories.

While regression tasks are concerned with estimating the relationship between some input variable with a continuous output variable, there are different types of regression algorithms:

- Linear regression

- Polynomial regression
- Stepwise regression
- Ridge regression
- Lasso regression
- ElasticNet regression

These different types of regression are suitable for different tasks. RIDGE REGRESSION is best used when there are high degrees of collinearity or nearly linear relationships in the set of features. Meanwhile, POLYNOMIAL REGRESSION is best used when there is a non-linear relationship between features, as it is capable of drawing curved prediction lines.

Linear regression is one of the most commonly used regression types, suited for drawing a straight line across a graph that shows a linear relationship between variables.

- Here I used Multiple Linear Regression and I observe underfitting as the r2 score is very low and mape (mean absolute percentage error) is high.

5.3. Decision tree

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.

The core algorithm for building decision trees called **ID3** by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with *Standard Deviation Reduction*.

- Decision Tree is drawn and graph is also presented in python code in which r2 score is 77% and it gets decreased to 68.2% after some tuning so these model is not good to fit model for this dataset

5.4. Random Forest

A random forest is an ensemble model that consists of many decision trees. Predictions are made by averaging the predictions of each decision tree. Or, to extend the analogy—much like a forest is a collection of trees, the random forest model is also a collection of decision

tree models. This makes random forests a strong modeling technique that's much more powerful than a single decision tree.

Each tree in a random forest is trained on the subset of data provided. The subset is obtained both with respect to rows and columns. This means each random forest tree is trained on a random data point sample, while at each decision node, a random set of features is considered for splitting.

In the realm of machine learning, the random forest regression algorithm can be more suitable for regression problems than other common and popular algorithms. Below are a few cases where you'd likely prefer a random forest algorithm over other regression algorithms:

There are non-linear or complex relationships between features and labels.

You need a model that's robust, meaning its dependence on the noise in the training set is limited. The random forest algorithm is more robust than a single decision tree, as it uses a set of uncorrelated decision trees.

If your other linear model implementations are suffering from overfitting, you may want to use a random forest.

- Next when the random forest model has been used r2_score seems to be 74.68% and error rmse is 4.71, mape is 23.44.
- From the above results random forest is good as it is giving >70% of good predictions then error rate is still high.

5.5. Gradient Boosting

"Boosting" in machine learning is a way of combining multiple simple models into a single composite model. This is also why boosting is known as an additive model, since simple models (also known as weak learners) are added one at a time, while keeping existing trees in the model unchanged. As we combine more and more simple models, the complete final model becomes a stronger predictor. The term "gradient" in "gradient boosting" comes from the fact that the algorithm uses gradient descent to minimize the loss.

When gradient boost is used to predict a continuous value – like age, weight, or cost – we're using gradient boost for regression.

Decision trees are used as the weak learners in gradient boosting. Decision Tree solves the problem of machine learning by transforming the data into tree representation. Each

internal node of the tree representation denotes an attribute and each leaf node denotes a class label. The loss function is generally the squared error (particularly for regression problems). The loss function needs to be differentiable.
Also like linear regression we have concepts of **residuals** in Gradient Boosting Regression as well. Gradient boosting Regression calculates the difference between the current prediction and the known correct target value.
This difference is called residual. After that Gradient boosting Regression trains a weak model that maps features to that residual. This residual predicted by a weak model is added to the existing model input and thus this process nudges the model towards the correct target. Repeating this step again and again improves the overall model prediction.
Also it should be noted that Gradient boosting regression is used to predict continuous values like house price , while Gradient Boosting Classification is used for predicting classes like whether a patient has a particular disease or not.

The high-level steps that we follow to implement Gradient Boosting Regression is as below:

a. Select a weak learner
b. Use an additive model
c. Define a loss function
d. Minimize the loss function

- When working with regression problem one of the efficient algorithms is Gradient Boosting Regressor
- This uses the same procedure to build, predict and evaluate the model.
- To fit the model GradientBoostingRegressor.fit() method is used and some of the hyperparameters are specified in the algorithm for better prediction.
- Next the predict method is used for prediction on validation set as model.predict(x_test)
- Next evaluation is made by comparing the true values with predicted values. If it seems good the model is used for prediction on any other unknown fare prediction data.
- Here the r2_score is 79.18% and rmse is 4.71, mape is 22.32.
- These results are much better compared to previous model.

5.6. Extra Trees Regressor

The Extra-Trees algorithm builds an ensemble of unpruned decision or regression trees according to the classical top-down procedure. Its two main differences with other tree-

based ensemble methods are that it splits nodes by choosing cut-points fully at random and that it uses the whole learning sample (rather than a bootstrap replica) to grow the trees.

The predictions of the trees are aggregated to yield the final prediction, by majority vote in classification problems and arithmetic average in regression problems.

The difference between Random forest and Extra Trees Regressor:

- Random forest uses bootstrap replicas, that is to say, it subsamples the input data with replacement, whereas Extra Trees use the whole original sample. In the Extra Trees sklearn implementation there is an optional parameter that allows users to bootstrap replicas, but by default, it uses the entire input sample. This may increase variance because bootstrapping makes it more diversified.
- Another difference is the selection of cut points in order to split nodes. Random Forest chooses the optimum split while Extra Trees chooses it randomly. However, once the split points are selected, the two algorithms choose the best one between all the subset of features. Therefore, Extra Trees adds randomization but still has optimization.

These differences motivate the reduction of both bias and variance. On one hand, using the whole original sample instead of a bootstrap replica will reduce bias. On the other hand, choosing randomly the split point of each node will reduce variance.

In terms of computational cost, and therefore execution time, the Extra Trees algorithm is faster. This algorithm saves time because the whole procedure is the same, but it randomly chooses the split point and does not calculate the optimal one.

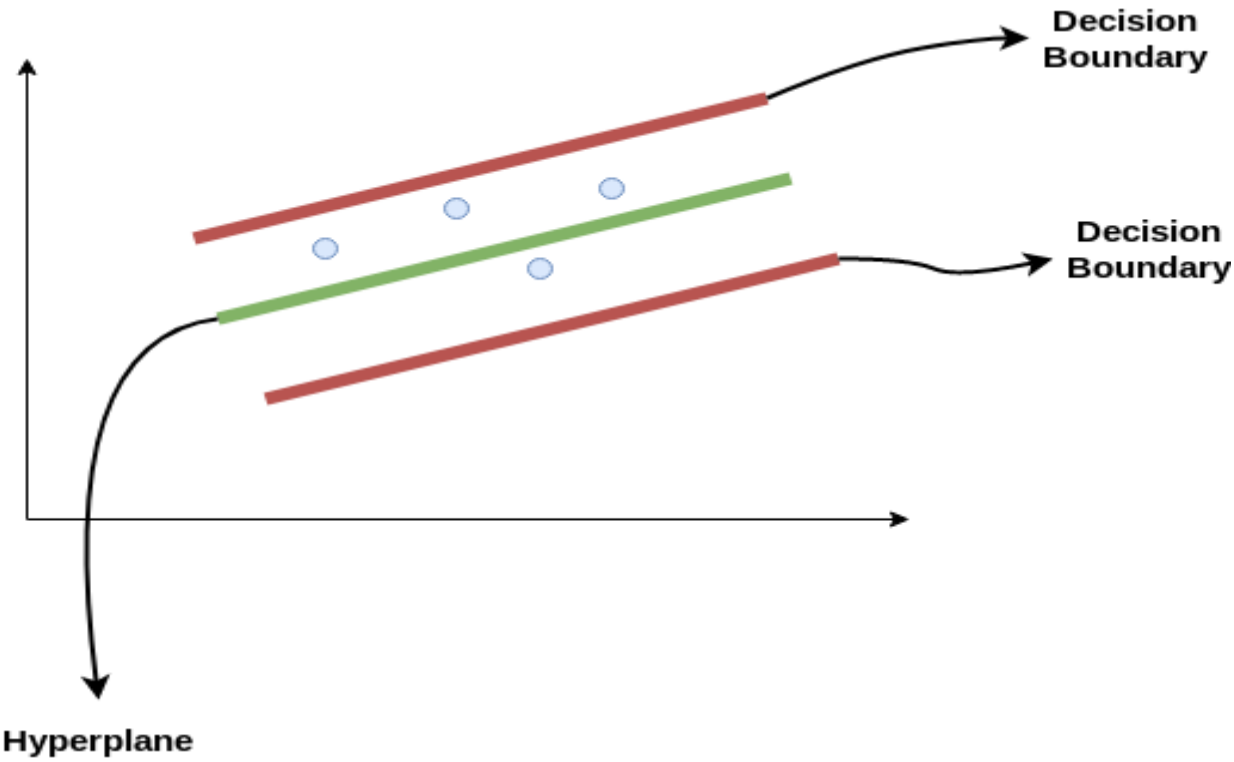
- Extra Trees Regressor performs faster than Random Forest then this model seems not better in terms of its r2 score in comparison with Random Forest.
- The Extra Trees Regressor r2 score is 70.03% and the error rate mape is 34.33, rmse is 4.71.
- This model did not improve after tuning so this model is not useful for predictions for this dataset instead performs poor so this not useful for cab fare prediction for this data.

5.7. SVR

Consider these two red lines as the decision boundary and the green line as the hyperplane. **Our objective, when we are moving on with SVR, is to basically consider the points that are within the decision boundary line.** Our best fit line is the hyperplane that has a maximum number of points.

The first thing that we'll understand is what is the decision boundary (the danger red line above!). Consider these lines as being at any distance, say 'a', from the hyperplane. So, these are the lines that we draw at distance '+a' and '-a' from the hyperplane. This 'a' in the text is basically referred to as epsilon.

Our main aim here is to decide a decision boundary at 'a' distance from the original hyperplane such that data points closest to the hyperplane or the support vectors are within that boundary line.



- The r2_score with SVR for this cab fare prediction dataset is 79.26%
- The error rate mape and rmse is 17.45 and 4.71 respectively.
- This model seems to perform much better than any other model for the fare prediction.

5.8. Stacking

Stacking is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The base level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features.

Applying stacked models to real-world big data problems can produce greater prediction accuracy and robustness than do individual models. The model stacking approach is

powerful and compelling enough to alter your initial data mining mindset from finding the single best model to finding a collection of really good complementary models.

Of course, this method does involve additional cost both because you need to train a large number of models and because you need to use cross validation to avoid overfitting.

- Stacking has got good r2 score of 74.75% and after tuning it improves a lot and reaches to 79.36% making it best to consider for model predictions as it is the combination of many models.

Chapter 6: Model Evaluation and selection

6.1. Model evaluation metrics

Evaluating your developed model helps you refine the model. You keep developing and evaluating your model until you reach an optimum model performance level. (Optimum model performance doesn't mean 100 percent accuracy; 100 percent accuracy is a myth).

Regression Model Performance Parameters

Let's talk about the regression model evaluation metrics. We usually check these parameters while developing linear regression models or some other regression models where the dependent variable is continuous (non-binary or categorical) in nature.

1. MAPE:

Mean absolute percentage error (MAPE) is the simplest evaluation metric to calculate in regression. It uses actual and predictive numbers directly without any treatment, hence highly affected by outlier values in data. If MAPE decreases, model performance will improve.

$$\frac{\sum(abs(Actual - Predicted)/abs(Actual))}{n}$$

MAPE metric is given by:

Where n is the number of observations.

2. RMSE:

Root mean square error (RMSE) is the most used evaluation metric in regression problems. It follows an assumption that error is unbiased and follows a normal distribution. It avoids the use of absolute error and uses the square of the difference of actual and predicted, as an absolute value is highly undesirable in mathematical calculations. RMSE is highly affected by outlier values. Hence, make sure you've removed/treated the outliers from your data set before using this metric. If RMSE decreases, model performance will improve.

RMSE metric is given by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Where, N is Total Number of Observations.

3. R Square:

If we talk about MAPE and RMSE, we do not do any benchmark comparison. Hence, we use the R square statistic for that. R square limit is 0 to 1. Value more towards 1, tells us that the developed model is high on accuracy. R square metric is given by:

$$R^2 = 1 - \frac{\text{MSE(model)}}{\text{MSE(baseline)}}$$

$$\frac{\text{MSE(model)}}{\text{MSE(baseline)}} \qquad \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (\overline{y_i} - \hat{y}_i)^2}$$

MSE (model): Mean Square Error of the predictions against actual.

MSE (baseline): Mean Square Error of mean prediction against actual.

4. Adjusted R Square:

Adjusted R-Square metric is a more advanced version of R-Square. On adding new variables to the model, the R-Square value either increases or remains the same. R-Square does not penalize for adding variables that add no value to the model. But on the other hand, adjusted R-Square increases only if a significant variable is added into the model. Adjusted R- Square metric is given by-

$$\bar{R}^2 = 1-\left(1-R^2\right)\left[\frac{n-1}{n-(k+1)}\right]$$

k: number of variables

n: number of observations

Adjusted R-Square takes the number of variables into account. When we add more variables in the model, the denominator n-(k +1) decreases, so the whole expression increases.

If Adjusted R-Square does not increase, that means the variable added isn't valuable for the model. So overall we subtract a greater value from 1 and Adjusted R-Square will decrease.

Apart from these four above parameters, we have many other performance parameters, but these are the most commonly used.

- Yes, these are metrics used in this cab fare prediction evaluation.

6.2. Hyperparameter tuning

When creating a machine learning model, you'll be presented with design choices as to how to define your model architecture. Often times, we don't immediately know what the optimal model architecture should be for a given model, and thus we'd like to be able to explore a range of possibilities. In true machine learning fashion, we'll ideally ask the machine to perform this exploration and select the optimal model architecture automatically. Parameters which define the model architecture are referred to

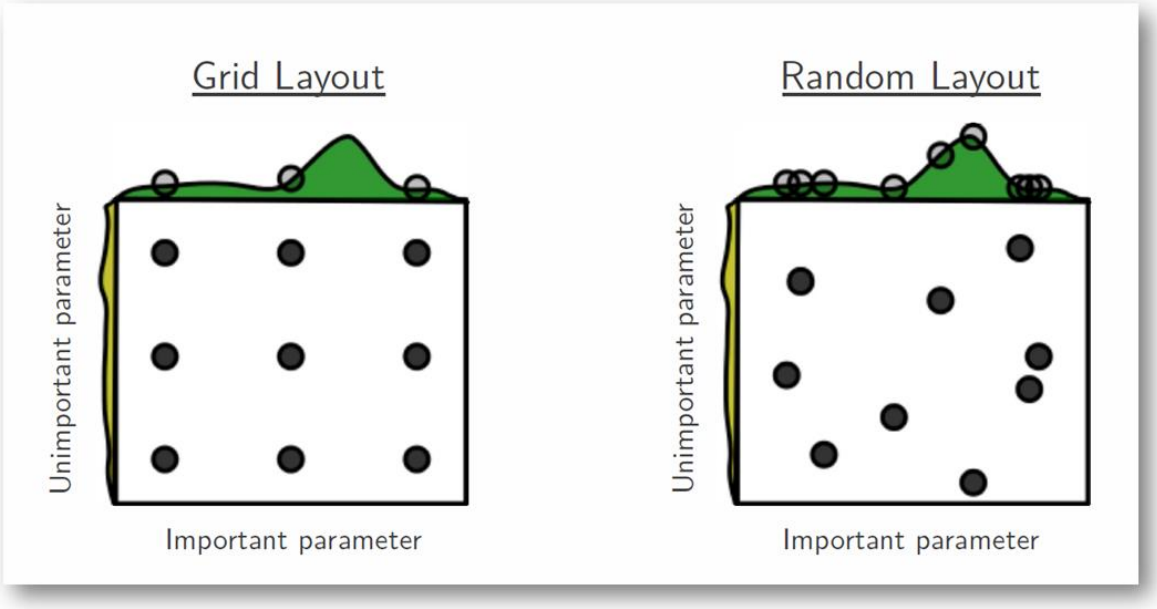
as **hyperparameters** and thus this process of searching for the ideal model architecture is referred to as HYPERPARAMETER TUNING.

I want to be absolutely clear, ***hyperparameters are not model parameters*** and they cannot be directly trained from the data. MODEL PARAMETERS are learned during training when we optimize a loss function using something like gradient descent.The process for learning parameter values is shown generally below.

Whereas the model parameters specify how to transform the input data into the desired output, the hyperparameters define how our model is actually structured. Unfortunately, there's no way to calculate "which way should I update my hyperparameter to reduce the loss?" (ie. gradients) in order to find the optimal model architecture; thus, we generally resort to experimentation to figure out what works best.

In general, this process includes:

1. Define a model
2. Define the range of possible values for all hyperparameters
3. Define a method for sampling hyperparameter values
4. Define an evaluative criterion to judge the model
5. Define a cross-validation method



- The hyperparameter tuning is done using the above methods Grid Search and Random Search CV approaches.

6.3. Model selection

- After performing all the analysis, model building and evaluation after hyperparameter tuning it is found that Stacking and SVR models perform better with almost 80% of correct predictions
- The Stacking and SVR models r2 scores are 79.35% and 79.26% respectively.
- The rmse and mape scores of Stacking and SVR models are 4.71,22.93 and 4.71,17.45 respectively.
- From this I can say Stacking and SVR are good performing models with just a slight difference. Now I choose SVR for fare prediction on required test set as r2_scores say Stacking is superior then the error rate seems high in Stacking so I choose SVR be the better model than Stacking.
- Stacking also can be used as it is the combination of many models including SVR and many other models can be used. Stacking seems a very good option for many datasets even here it performs almost the best and for this dataset SVR also seems as good as Stacking so I choose Stacking for predictions on the required cab fare prediction test data.

Chapter 7: Conclusion

7.1. Predictions on test set

```
In [358]: test_df.head()
```

Out[358]:

2	Year_2013	...	Time_of_day_3	Season_1	Season_2	Season_3	Weektype_1	WeekinMonth_1	WeekinMonth_2	WeekinMonth_3	MonDays_1	fare_amount
0	0	...	0	1	0	0	0	0	0	1	1	10.203263
0	0	...	0	1	0	0	0	0	0	1	1	10.524567
0	0	...	0	0	0	0	1	1	0	0	0	4.424247
1	0	...	1	0	0	0	1	0	0	0	0	8.026364
1	0	...	1	0	0	0	1	0	0	0	0	14.910192

- Here are the final predictions made on cab fare data with SVR model
- This comes with almost 80% of correct predictions on test data so this is with any other unknown data.
- Data science project ends with deployment and feedback. If it goes well this is high point and if it is not performing well again it comes to data scientists for further modification and enhancement.

7.2. Deployment and running the code

- After successful predictions it is required to sent it to the required place for deployment where the model gets used.
- For that it is not possible to send just the python or R file it is required in some form of webpage or API through which clients or users can use the prepared model for their company or individual purpose.
- The deployment can be done through many ways like from AWS, Google, Heroku, Flask and many more cloud environments.
- From this cloud sources we create a project in that environments for making webpage with our model and other required resources.
- Some of the files made through this are app.yaml, main.py(model file), requirements.txt file, python.py file , train.py file etc.
- Before preparing these files in python we save our model through joblib package and save the model in pickle format and load this model in another python file and make necessary use of it.
- After making all the files we open the prompt or through Flask(Pycharm env) we give run the model and write the command
- 'pip requirements.txt'
- Now you will see the https:localhostxxxx or some IP address through which it successfully creates the webpage in which the model is implemented which can be used by client or users whoever requires this information. In this case Cab owners or Transportation providers use this for their Cab Fare prediction.
- I have added some links for this deployment or example for deployment from some sources in references section.
- This the end to end project made through Machine Learning algorithms for solving the real time problems or cab fare prediction.

References:

https://searchenterpriseai.techtarget.com/definition/data-science

https://stackabuse.com/multiple-linear-regression-with-python/

https://www.saedsayad.com/decision_tree_reg.htm

https://heartbeat.fritz.ai/random-forest-regression-in-python-using-scikit-learn-9e9b147e2153

https://blog.paperspace.com/implementing-gradient-boosting-regression-python/

https://machinelearningmastery.com/extra-trees-ensemble-with-python/

https://www.thekerneltrip.com/statistics/random-forest-vs-extra-tree/

https://scikit-learn.org/stable/modules/generated/sklearn.tree.ExtraTreeRegressor.html

https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/

https://www.aionlinecourse.com/tutorial/machine-learning/support-vector-regression

https://mlfromscratch.com/model-stacking-explained/#/

https://blogs.sas.com/content/subconsciousmusings/2017/05/18/stacked-ensemble-models-win-data-science-competitions/

https://www.mygreatlearning.com/blog/model-evaluation-metrics-for-machine-learning/

https://www.jeremyjordan.me/hyperparameter-tuning/

https://www.statsmodels.org/stable/generated/statsmodels.stats.anova.anova_lm.html

https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/

https://www.deploymachinelearning.com/

https://www.analyticsvidhya.com/blog/2020/04/how-to-deploy-machine-learning-model-flask/

https://medium.com/datadriveninvestor/deploy-your-machine-learning-model-using-flask-made-easy-now-635d2f12c50c