Bike Rental Count Prediction

Author: Mayank Juneja

INDEX

1. INTRODUCTION

1.1 Problem Statement

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings.

1.2 Data Understanding

Understanding of data is the very first and important step in the process of finding solution of any business problem. Here in our case we are provided with a data set with following features, we need to go through each and every variable of it to understand and for better functioning.

Size of Dataset Provided: - 731 rows, 16 Columns (including dependent variable)

Missing Values: No
Outliers Presented: Yes

2. DATA PRE-PROCESSING

When we required to build a predictive model, we require to look and manipulate the data before we start modelling which includes multiple preprocessing steps such as exploring the data, cleaning the data as well as visualizing the data through graph and plots, all these steps is combined under one shed which is **Exploratory Data Analysis**, which includes following steps:

- Data Exploration And Cleaning
- Missing Value Analysis
- Outlier Analysis
- Feature Engineering
- Feature Selection

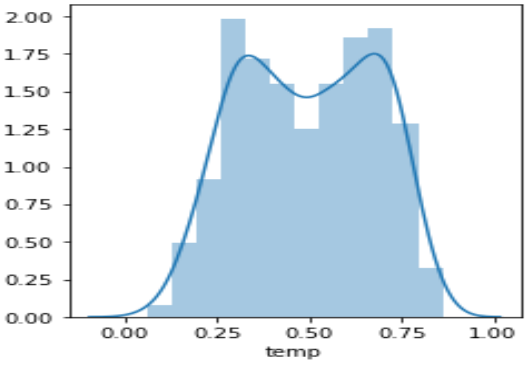
Missing Value Analysis

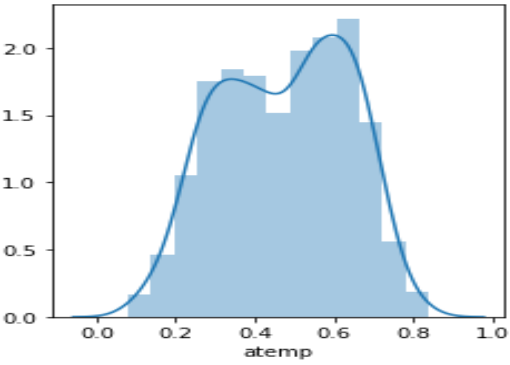
There are no missing values in train dataset.

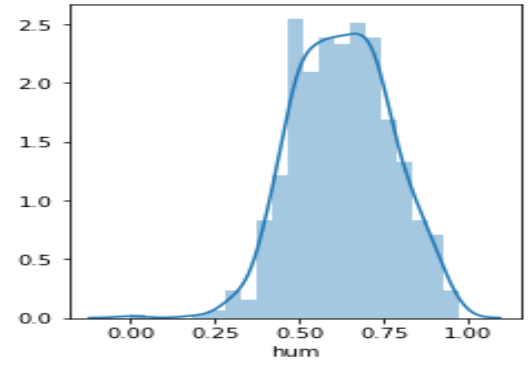
Data Exploration

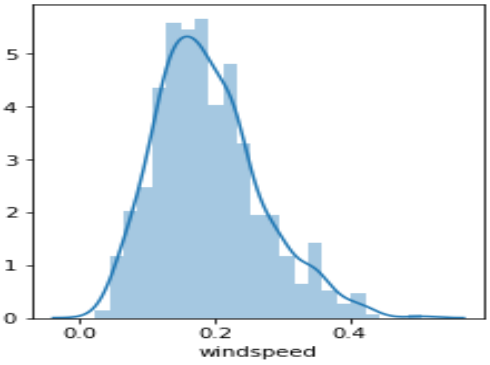
Exploring the data, which means looking at all the features and knowing their characters. According to our problem statement, we are analysing the data of bike rental count, whose features are comprised of cnt, casual, registered, temp, humidity, windspeed and datetime data.

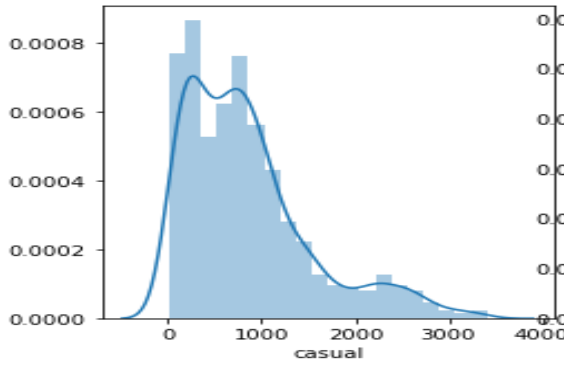
We plotted the distributions of all the numerical features in the dataset to see the skewness in the distribution of features.

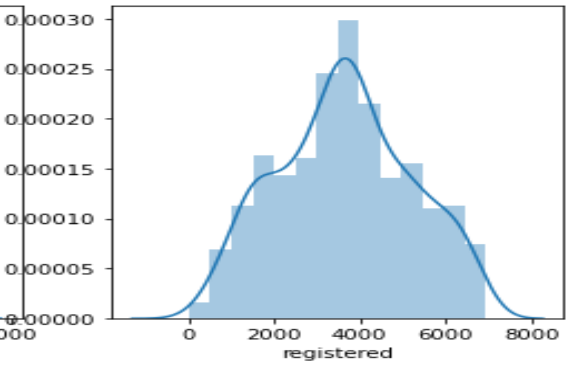

temp

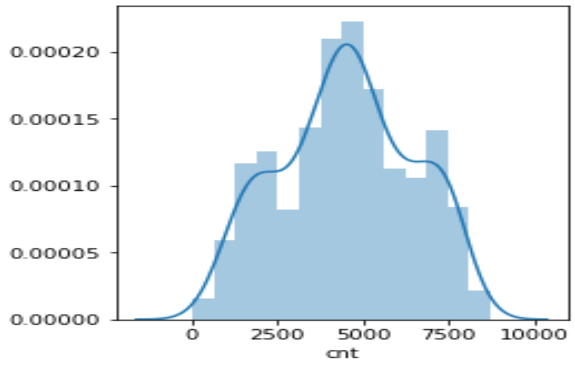

atemp


hum


windspeed


casual

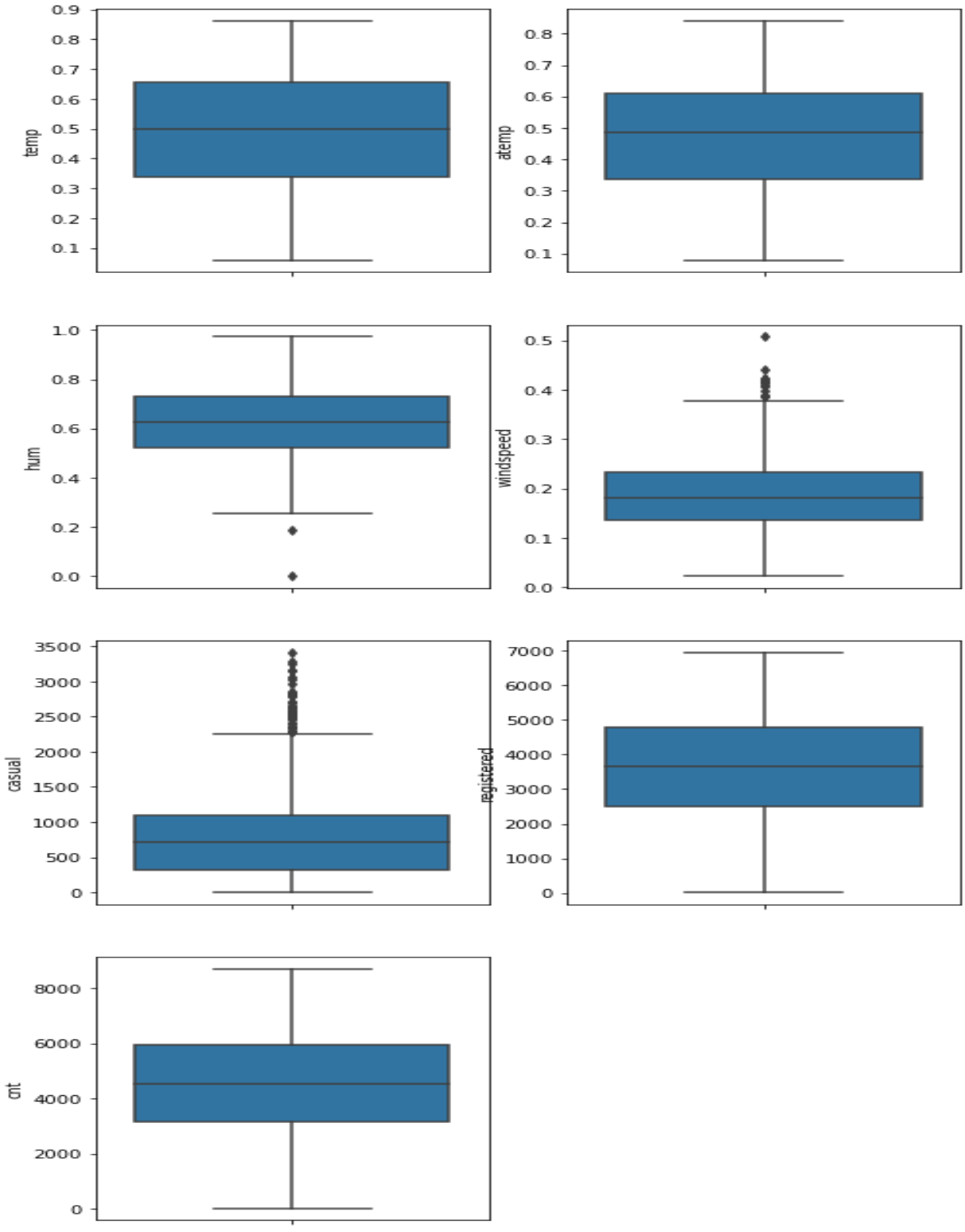

registered


cnt

Outlier Analysis

In Python

For checking the outliers in our numerical features we used the boxplot for visualization.



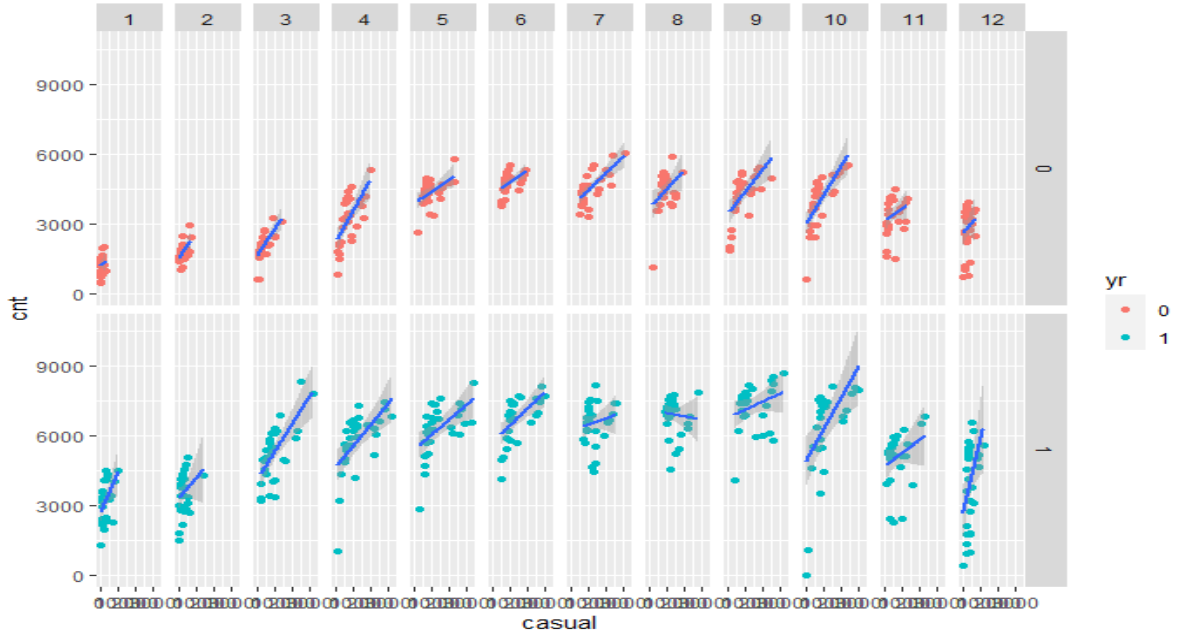
Count of outliers in Numerical Data

```
instant         0
dteday          0
season          0
yr              0
mnth            0
holiday         0
weekday         0
workingday      0
weathersit      0
temp            0
atemp           0
hum             2
windspeed      13
casual         44
registered      0
cnt            44
```

We see there are 44 outliers in our target variable ‘casual’, we set them to NaN and also in corresponding we set NaN for those values in ‘cnt’ where ‘casual’ was NaN as we know ‘cnt’ is the sum of ‘casual’ and ‘registered’.

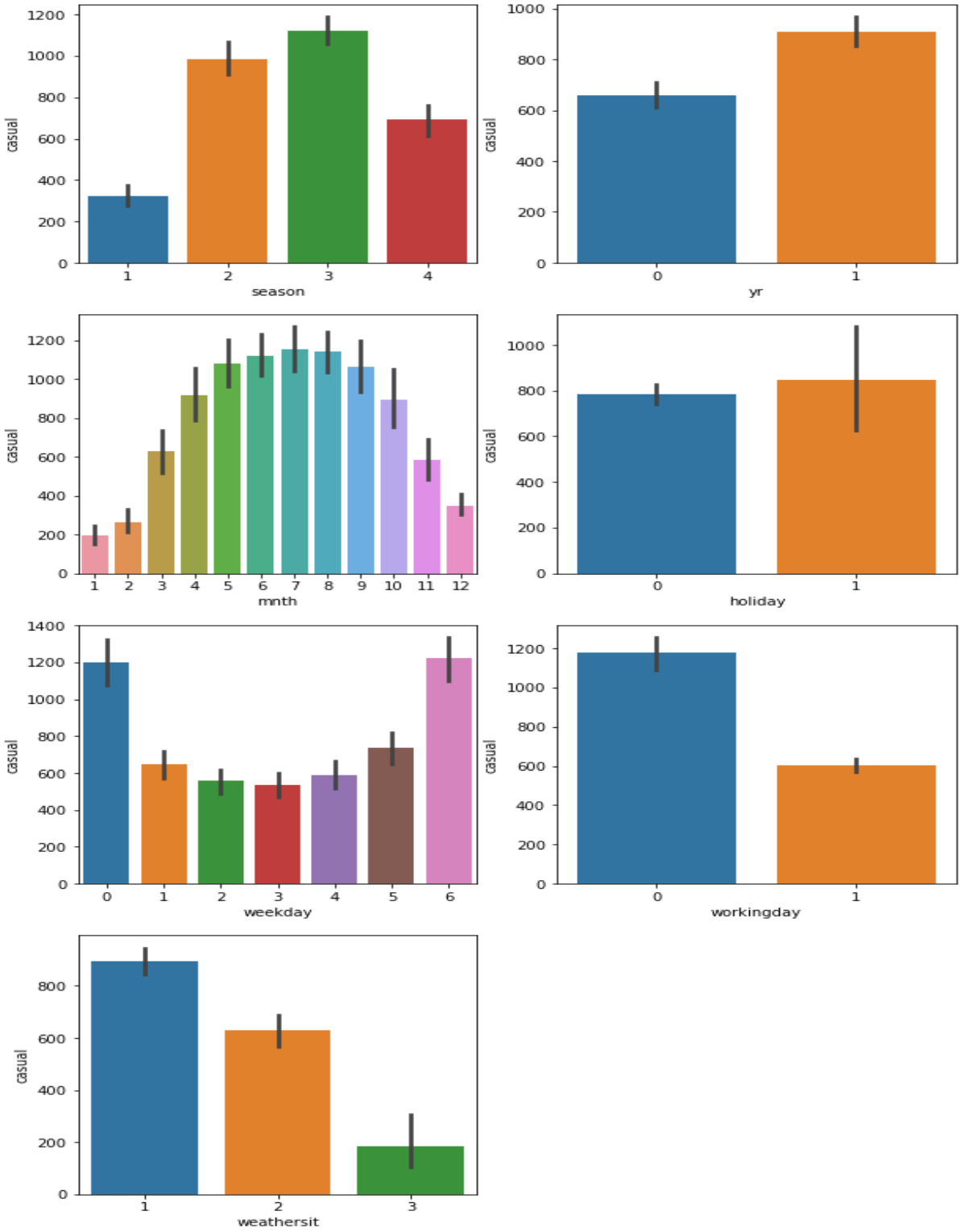
IN R code:

Different approach was followed in R that is to not to remove outliers from ‘casual’ variable as if we see the ‘cnt-casual’ scatterplot for different years and months, we can infer that those outliers were natural and also our target variable ‘cnt’ has no outliers so doing nothing with them in ‘casual’ variable would be better.



As for variables like windspeed and hum we impute them using KNN Imputation.

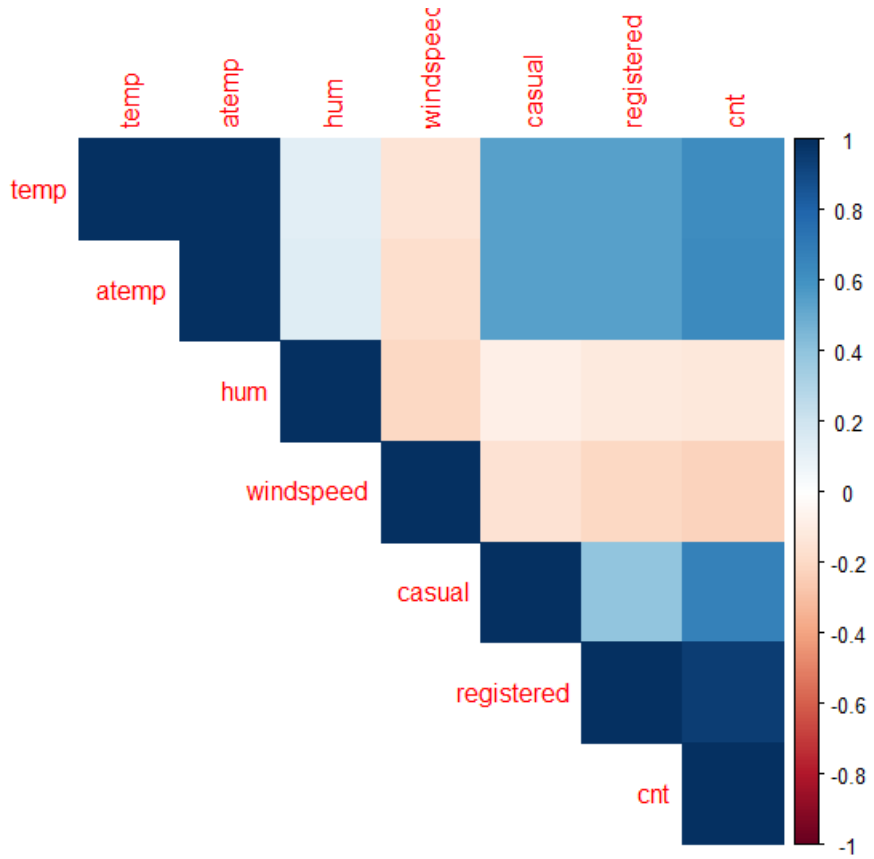
For Categorical Data, we use barplot visualizations to see for any relations.



Feature Selection

NUMERICAL FEATURES

Correlation Analysis using Pearson's coefficient was used for feature selection.



In Python

We see there are 3 dependent variables "cnt", "casual" and "registered" where cnt is highly correlated to both casual and registered, so we dropped it and on checking vif for predictors we dropped "hum" and "atemp" to avoid multicollinearity.

IN R

We dropped casual and registered in R as they were highly correlated to cnt and our ultimate target was cnt, so we dropped them and also dropped atemp on checking vif.

CATEGORICAL FEATURES

ANOVA was used for feature selction for categorical features.

In Python

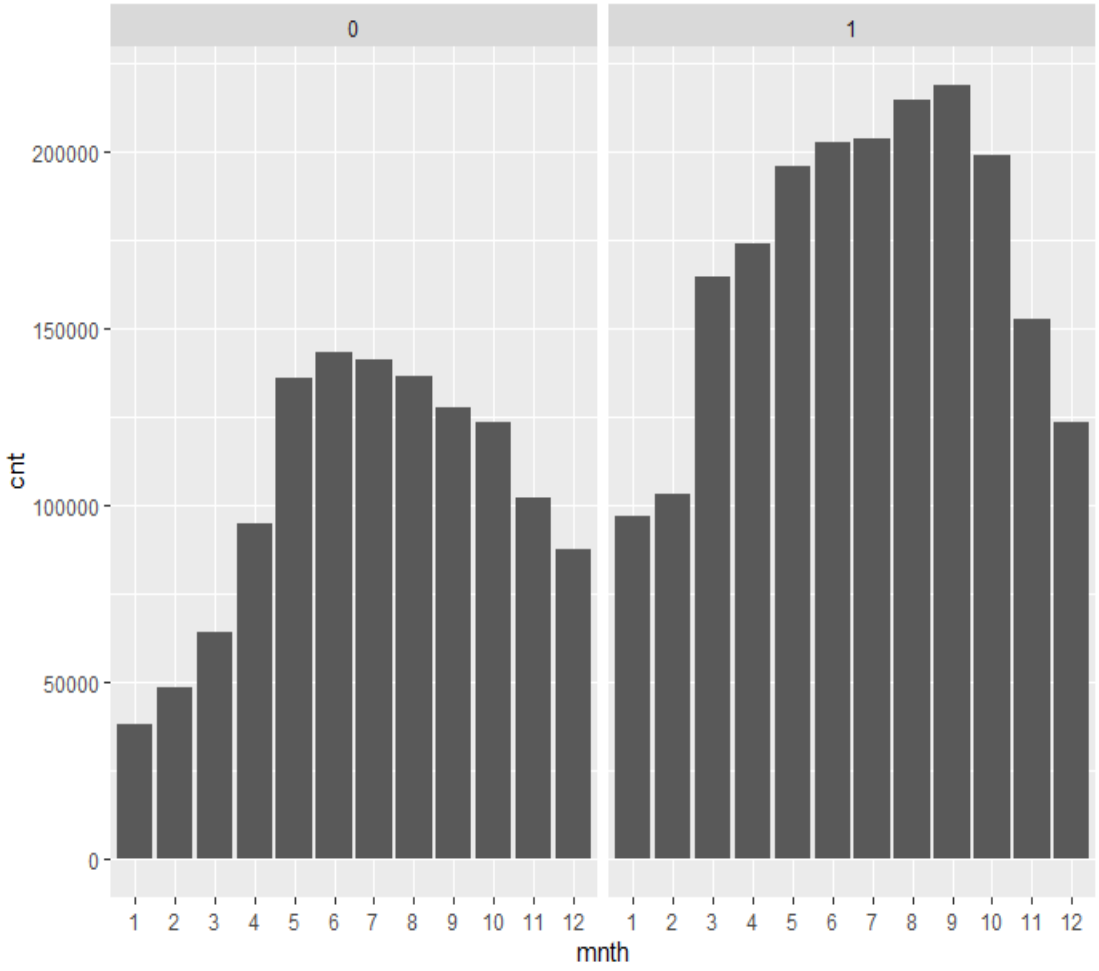
No variable was dropped.

In R

Holiday, weekday and working day were dropped as they had p-value less than 0.05.

Feature Engineering

A new variable named “month_fe” was created from “mnth” variable with fewer categories based on demand in that interval of months. Also the trend was same in both years, so we derived a new variable.



Target Variable Transformation

We used Logarithmic method for transforming target variables in both R and Python to make our computation easier and also large range of values can decrease the overall accuracy in model.

For categorical features we created dummy variables for each category.

MODELING

The next step is to differentiate the train data into 2 parts i.e. train and valid. The splitting of train data into 2 parts is very important factor to verify the model performance and to understand the problem of over-fitting and under-fitting. Over-fitting is the term where training error is low and testing error is high and under-fitting is the term where both training and testing error is high. Those are the common problem of complex model. In this analysis, since we are predicting fare amount which is the numeric variable. So, we come to know that, our problem statement is predicting type. So, what we can do is we will apply supervise machine learning algorithms to predict our target variable. As we know our target variable is continuous in nature so, here we will build regression model. **Root Mean Square Error** (RMSE) to measures how much **error** there is between two data sets. In other words, it compares a predicted value and an observed or known value. The RMSE is directly interpretable in terms of measurement units, and so is a better measure of goodness of fit. So, in our case any model we build should have lower value of an **RMSE** and higher value of variance i.e**. R square.** Other metrics which are used for evaluation of model are **Mean Absolute Error(MAE)** and **Mean Absolute Percentage Error(MAPE).**

LINEAR REGRESSION

Linear Regression is one of the statistical methods of prediction. It is used to find a linear relationship between the target and one or more predictors. It means the target variables should be continuous in nature. The main idea is to identify a line that best fits the data. To build any model we have some assumptions to

put on data and model. This algorithm is not very flexible, and has a very high bias. Below we calculated RMSE values using linear regression.

	Python	R
RMSE Train	918.64	950.216
RMSE Valid(test)	1052.23	846.044

RANDOM FOREST

Random forest is the collection of multiple decision trees. In Random Forest, output is the average prediction by each of these trees. For this method to work, the baseline models must have a lower bias. The idea behind Random Forest is to build n number of trees to have more accuracy in dataset. Random Forest uses bagging method for predictions. It can handle large no of independent variables without variable deletion and it will give the estimates that what variables are important. The RMSE values are shown below.

	Python	R
RMSE Train	279.35	392.431
RMSE Valid(test)	850.6	701.384

Hyperparameter tuning for Random Forest Model

Here we have used Random Search CV for hyper parameters tuning.

Random Search CV: This algorithm set up a grid of hyperparameter values and select random combinations to train the model and score. The number of search iterations is set based on time/resources.

After tuning the model we get following results:

	Python	R
RMSE Train	513.51	333.921
RMSE Valid(test)	840.31	710.123

EXTREME GRADIENT BOOSTING

Gradient boosting is currently one of the most popular machine learning techniques for efficient modeling of tabular datasets of all sizes. It is very fast, takes quite less RAM to run, and focuses on the accuracy of the result. It is a technique which applicable for regression and classification type of problems. In this method, multiple weak learners are ensemble to create strong learners. Gradient boosting uses all data to train each learner. But instances that were misclassified by the previous learners are given more weight, so that subsequent learners can give more focus to them during training. Here we select XGBoost for model development.

	Python	R
RMSE Train	52.66	9.51
RMSE Valid(test)	755.06	715.184

Hyperparameter tuning for Extreme Gradient Boosting

Using Grid Search CV and Randomized Search CV to find the best parameters

	Python	R
RMSE Train	235.51	574.631
RMSE Valid(test)	737.15	644.469

Note: The RMSE values are high as while predictiong error the predictions were transformed to their original scale by taking antilog and they were compared with actual ones.

CONCLUSION

Model Evaluation:

Above model help us to calculate the **Root Mean Square Error (RMSE)** and **R-Squared** Values. RMSE is the standard deviation of the prediction errors. Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. RMSE is an absolute measure of fit. R-squared is a relative measure of fit. R-squared is basically explains the degree to which input variable explain the variation of the output. In simple words R-squared tells how much variance of dependent variable

explained by the independent variable. It is a measure if goodness of fit in regression line. Value of R-squared is between 0-1, where 0 means independent variable unable to explain the target variable and 1 means target variable is completely explained by the independent variable. So, Lower values of RMSE and higher value of R-Squared Value indicate better fit of model.

Model Selection

On the basis RMSE and R Squared results a good model should have least RMSE and max R Squared value. So, from tables for various algorithms we can see:

- From the observation of all RMSE Value we have concluded that,
- Gradient Boosting perform comparatively well while comparing their RMSE.
- So finally, we can say that Extreme Gradient Boosting(XGBoost) model is the best method to make prediction for this project with highest explained variance of the target variables and lowest error chances with parameter tuning.