BIKE RENTING

- Priyamvada

Contents

Introduction

1.1 Problem Statement

The aim of this project is prediction of bike rental count based on the environmental and seasonal settings. Bike-sharing rental process is highly correlated to the environmental and seasonal settings. For instance, weather conditions, precipitation, day of week, season etc. can affect the rental behaviours. Our task is basically to build regression models which will predict the count of bike rented depending on various environmental and seasonal conditions.

This assignment will help us learn the application of machine learning algorithms to data sets. This involves learning what data means, how to handle data, training, cross validation, prediction, testing the model, etc.

1.2 Data

A dataset has been provided which has 16 attributes. The details of data attributes in the dataset are as follows

- instant: Record index
- dteday: Date
- season: Season (1:springer, 2:summer, 3:fall, 4:winter)
- yr: Year (0: 2011, 1:2012)
- mnth: Month (1 to 12)
- hr: Hour (0 to 23)
- holiday: weather day is holiday or not (extracted from Holiday Schedule)
- weekday: Day of the week
- workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
- weathersit: (extracted fromFreemeteo)
  1: Clear, Few clouds, Partly cloudy, Partly cloudy
  2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
  3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
  4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)
- atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_max- t_min), t_min=-16, t_max=+50 (only in hourly scale)

- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

A sample dataset is shown below:

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

Fig 1: Sample dataset

1.3 Software Requirement

- R 3.6.1 for 64 bit
- Anaconda3 2019.07 for 64bit

Methodology

2.1 Data Pre Processing

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real world data are generally incomplete (lacking attribute values, lacking certain attributes of interest, or containing only aggregate data), Noisy (containing errors or outliers), inconsistent (containing discrepancies in codes or names) and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues.

There are many steps involved in pre-processing like exploratory data analysis, missing value analysis, outlier analysis, feature selection, feature scaling etc.

2.1.1 Exploratory Data Analysis

Exploratory data analysis (EDA) is an approach to analysing data sets to summarize their main characteristics, often with visual methods. EDA is done for seeing what the data can tell us beyond the formal modelling or hypothesis testing task.

All the required libraries are installed and loaded in both the environments. The working directory is set. The data set given in the csv format, is loaded.

In Figure 2 and 3, we have plotted the probability density functions of numeric variables present in the data including target variable cnt..

i. Target variable cnt is normally distributed

ii. Independent variables like 'temp','atemp', and 'regestered' data is distributed normally.

iii. Independent variable 'casual' and 'hum' data is slightly skewed. There is chances of getting outliers.

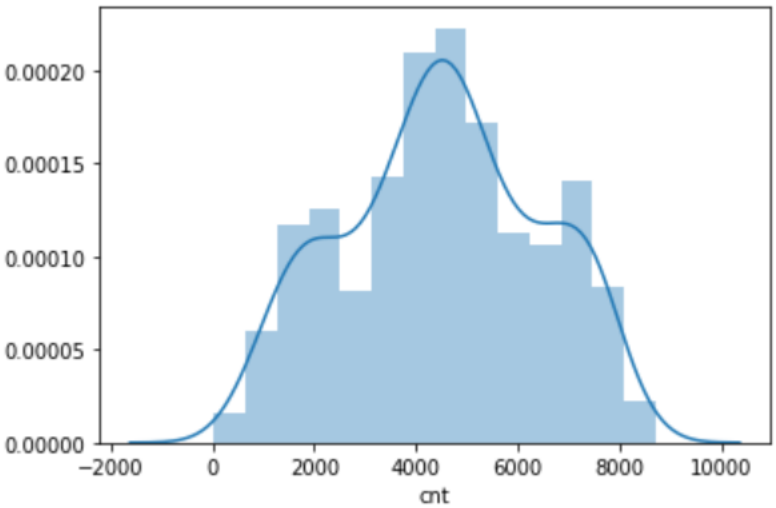


Fig 2 : Distribution of count variable (target variable)

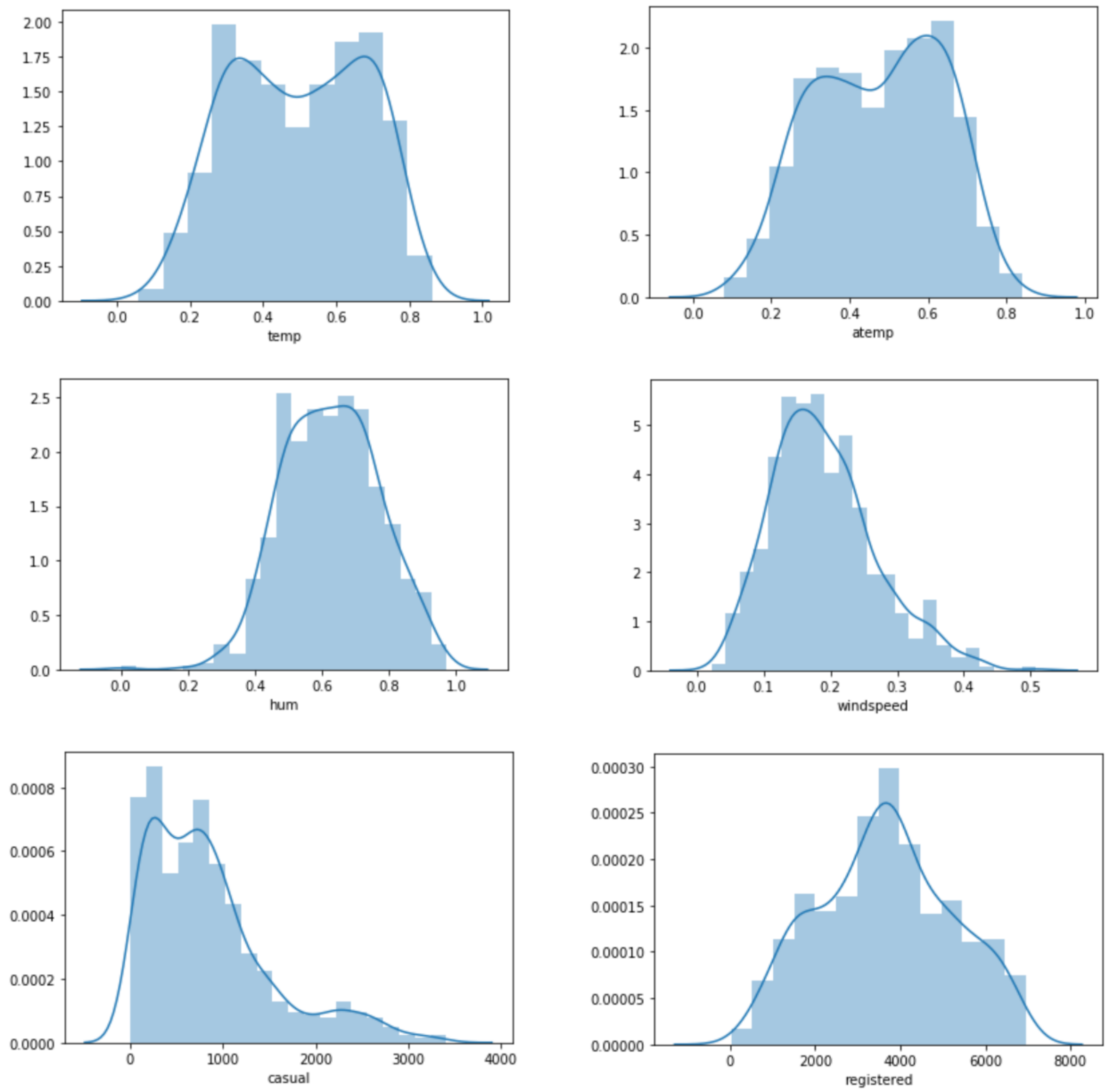


Fig 3: Plot showing distribution of different dependent variables

Steps taken :

- season, mnth, workingday, weathersit were converted into categorical variables.
- Feature Engineering: dteday variables's date value was changed to day of date and converted to categorical variable having 31 levels as a month has 31 days.
- Deleted the instant variable because as it is just index.
- Removed registered and casual variable as sum of registered and casual is the total count that is what we have to predict.

2.1.2 Missing Value Analysis

Missing data (or missing values) is defined as the data value that is not stored for a variable in the observation of interest. The problem of missing data is relatively common in almost all research and can have a significant effect on the conclusions that can be drawn from the data. It is created due to human error, refusal to answer while surveying or faulty survey questions etc. The missing value can be either imputed or the observations containing the missing values can be ignored depending upon the percentage of missing value present in the data. We can use the central tendencies to fill the missing values like mean or median, or we can use KNN imputation. KNN imputation finds the nearest neighbours based on existing attributes using Euclidean or Manhattan distance.

Missing value analysis is done to check if there is any missing value present in given dataset.

After doing missing value analysis, it was found that there are no missing values present in the given dataset. Fig 4 shows the same.

```
# Missing value analysis
missing_val = bike_data.isnull().sum()
missing_val
```

```
dteday        0
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
cnt           0
dtype: int64
```

Fig 4: Python code for checking missing value present in the data.

In R, “function(x){sum(is.na(x))} ” is the function used to check the sum of missing values while in python “ bike_data.isnull().sum()” is used to detect the missing values.

2.1.3 Outlier Analysis

An outlier is a data point that differs significantly from other observations. An outlier can cause serious problems in statistical analysis. The analysis of outlier data is referred to as outlier analysis or outlier mining. It is done to handle all inconsistent observations present in given dataset. It can only be done on continuous variable. One of the best method to detect outliers is Boxplot. Boxplot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles.

Figure 5 and 6 are visualization of numeric variable present in our dataset to detect outliers using boxplot.

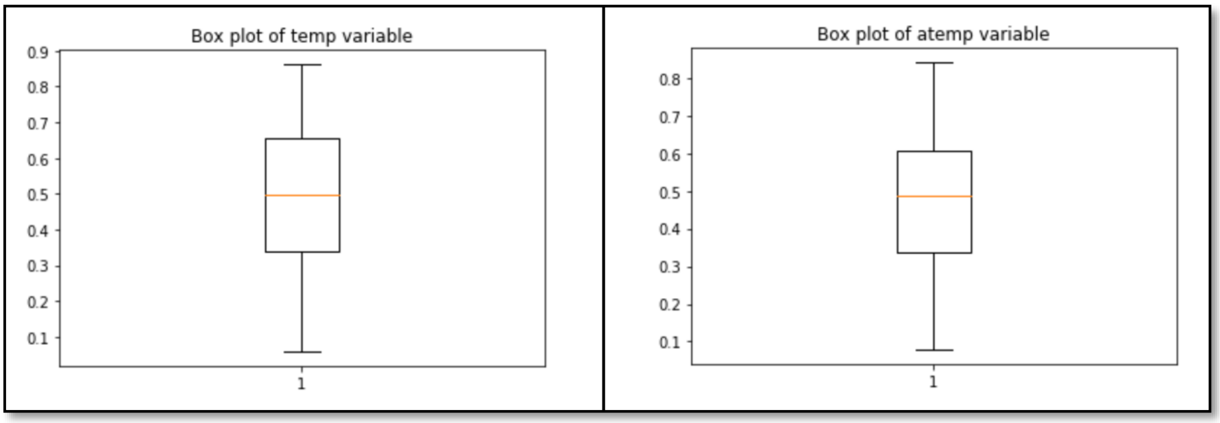


Fig 5 : Boxplot of temp variable (left) and atemp variable (right)

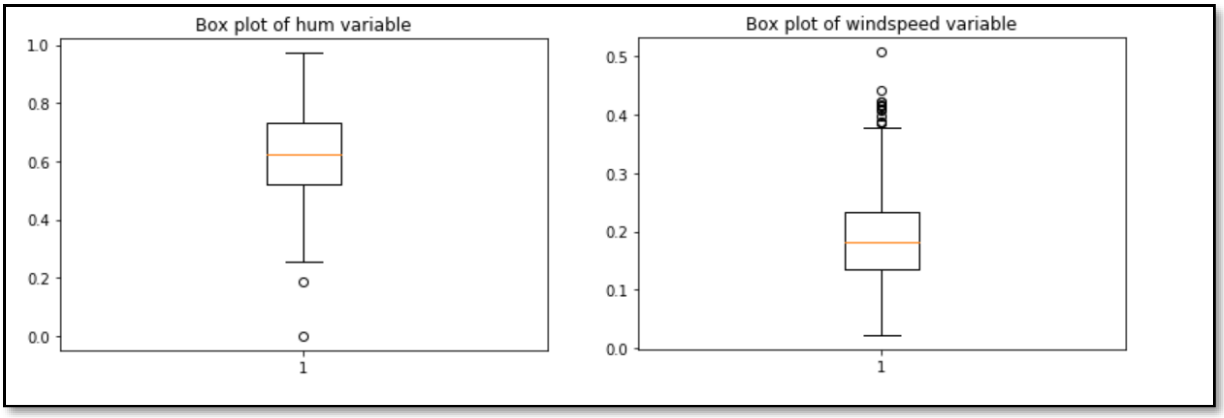


Fig 6: Boxplot of humidity variable 'hum' (left) and windspeed variable (right)

As seen above, temp and atemp has no outlier but hum and windspeed variables do have a few outliers. We can ignore these outliers.

2.1.4 Feature Selection

Feature selection means selecting a subset of relevant feature(Variables, predictors) for use in model construction.

Correlation Analysis - Correlation tells you the association between two continuous variables. It ranges from -1 to 1. Measures the direction and strength of the linear relationship between two quantitative variables.

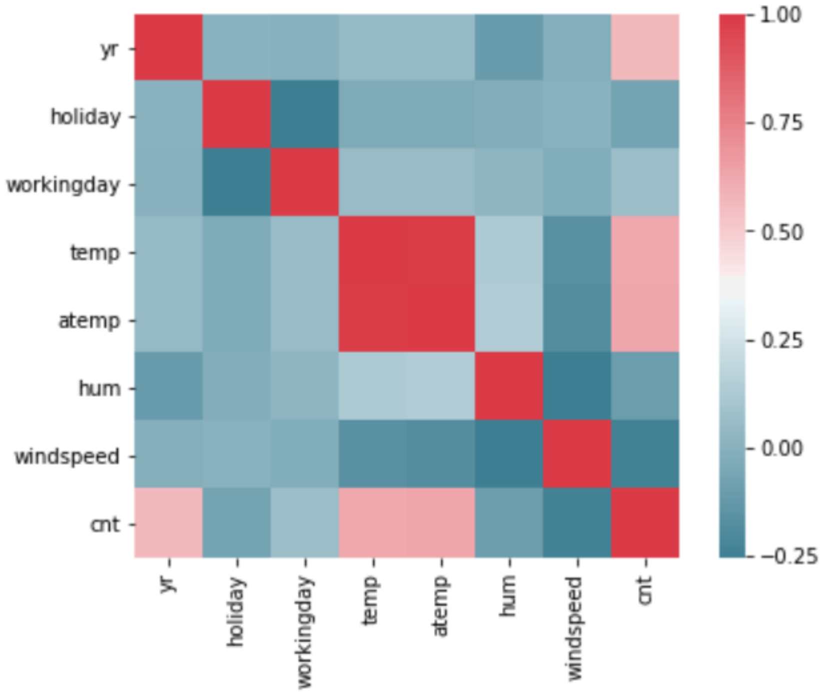


Fig 7: Correlation plot

It is clear from the heatmap above that temp and atemp are highly correlated. Since temp and atemp highly correlated, we are dropping atemp.

2.1.4 Feature Scaling

Feature Scaling is a technique which includes standardizing and normalizing the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. In given dataset all numeric values are already present in normalized form.

Models

3.1 Models

In this case we have to predict the count of bike renting according to environmental and seasonal condition. So the target variable here is a continuous variable. For Continuous variables we can use various Regression models. We will build three models here :-

(i) Linear Regression

(ii) Random Forest

(iii) c50 (Decision tree for regression target variable)

Model having less error rate and more accuracy will be our final model.

3.2 Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. c50 algorithm fits classification tree models or rule-based models using Quinlan’s C5.0 algorithm

For this model we have divided the dataset into train and test part using random sampling. Train part contains 80% data of data set and test contains 20% data and contains 12 variable where 12th variable is the target variable.

In R

```
# Decision tree
fit = rpart(cnt ~ ., data = train, method = "anova")
predictions_DT = predict(fit, test[,-12])
```

In python

```
#dividing data into train and test
train, test = train_test_split(bike_data, test_size=0.2)
```

```
# Decision Tree (c50)
fit_DT = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0:11], train.iloc[:,11])
predictions_DT = fit_DT.predict(test.iloc[:,0:11])
predictions_DT
```

3.3 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

In Random forest we have divided the dataset into train and test part using random sampling. For this model we have divided the dataset into train and test part using random sampling Where train contains 80% data of data set and test contains 20% data and contains 12 variable where 12th variable is the target variable.

In R

```
# Random Forest Model
library(randomForest)
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)
predictions_RF = predict(RF_model, test[,-12])
plot(RF_model)
```

In Python

```
#random forest
RFmodel = RandomForestRegressor(n_estimators = 200).fit(train.iloc[:,0:11], train.iloc[:,11])
RF_Predictions = RFmodel.predict(test.iloc[:,0:11])
#RF_Predictions
```

Below Figure 8 represents the curve of error rate as the number of trees increases. After 200 trees the error rate reaches to be constant. In this model we are using 200 trees to predict the target variable.

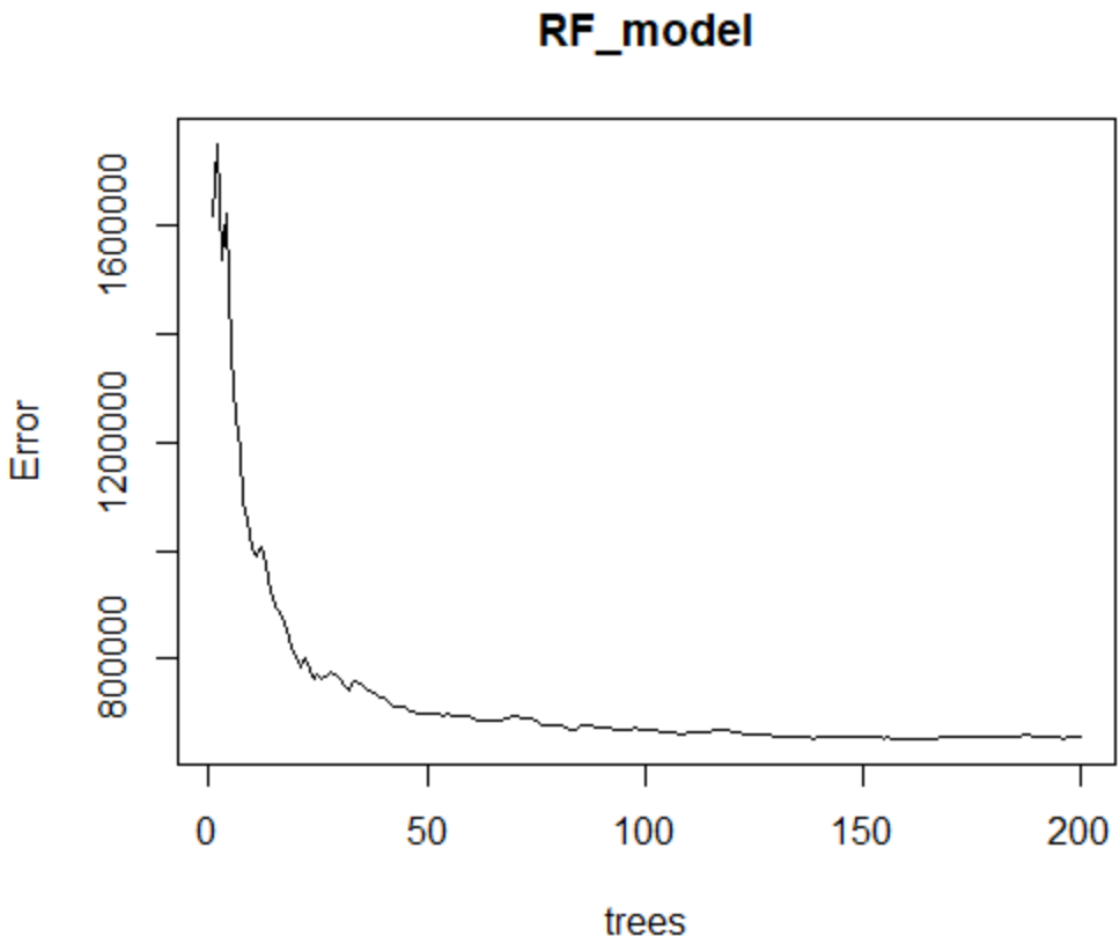


Fig 8: curve of error rate as the number of trees increases.

3.4 Linear Regression

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression.

For linear regression model we have divided the categorical containing more than 2 classes into dummy variable. So that all categorical variable should be in binary classes form. On creating dummy variable there are 64 variable in both R and Python. Where 64th is the target variable.

Further the data is again divided into train and test with 80 % train data and 20 % test data using random sampling.

In R

```
#Linear regression model making
lm_model = lm(cnt ~., data = train_lr)
predictions_LR = predict(lm_model,test_lr[,-64])
plot(lm_model)

summary(lm_model)
```

In Python

```
trainlr, testlr = train_test_split(data_lr, test_size=0.2)
model = sm.OLS(trainlr.iloc[:,63], trainlr.iloc[:,0:63]).fit()
predictions_LR = model.predict(testlr.iloc[:,0:63])
predictions_LR
```

Model Summary :

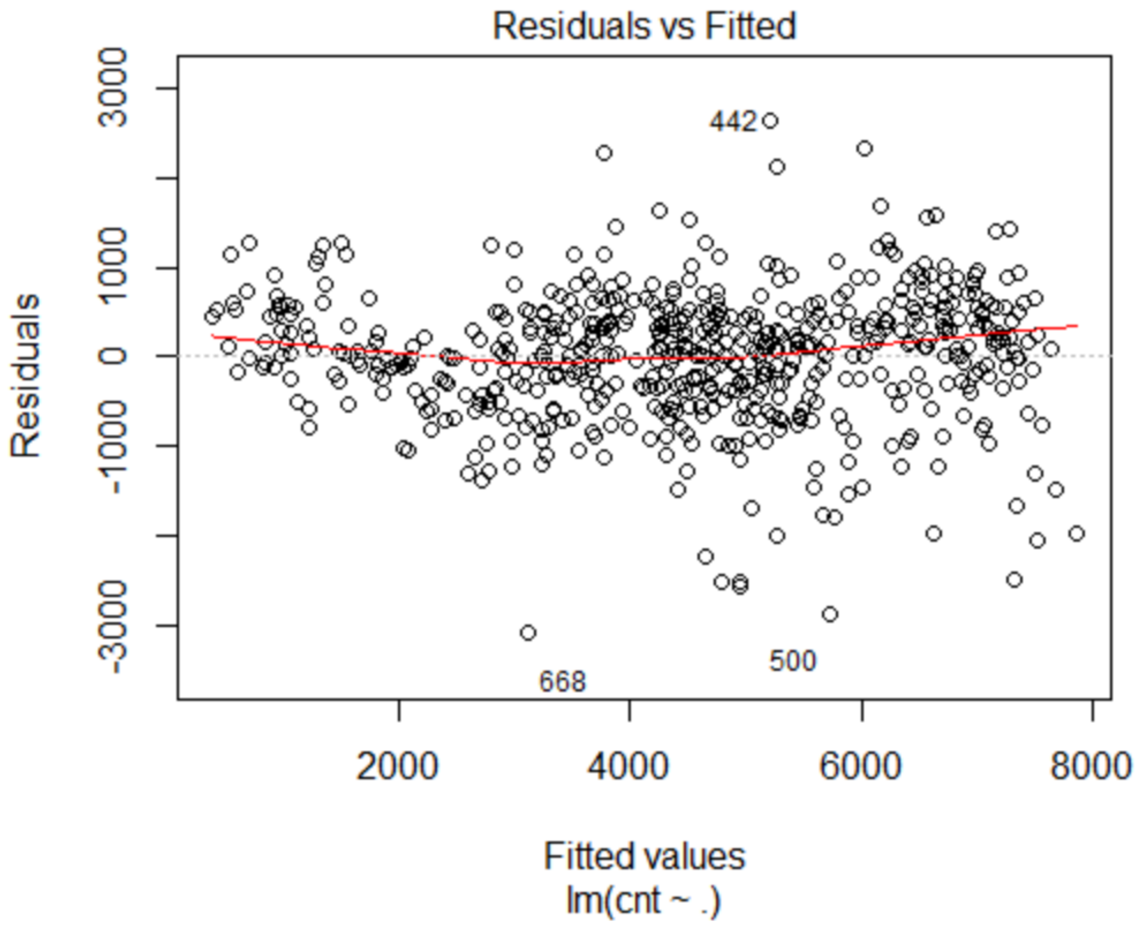
```
Call:
lm(formula = cnt ~ ., data = train_lr)

Residuals:
     Min       1Q   Median       3Q      Max 
-3092.90  -375.12    66.17   440.75  2627.24 

Coefficients: (6 not defined because of singularities)
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1402.063    456.244   3.073 0.002229 ** 
dteday_01     -506.473    301.212  -1.681 0.093269 .  
dteday_02     -133.357    300.336  -0.444 0.657206    
dteday_03     -102.125    303.570  -0.336 0.736693    
dteday_04       98.869    300.823   0.329 0.742543    
dteday_05     -272.252    297.957  -0.914 0.361279    
dteday_06      -45.780    299.802  -0.153 0.878694    
dteday_07     -282.707    305.392  -0.926 0.355018    
dteday_08     -367.659    297.857  -1.234 0.217625    
dteday_09     -224.046    300.179  -0.746 0.455775    
dteday_10     -122.468    299.467  -0.409 0.682741    
dteday_11       58.483    291.094   0.201 0.840849    
dteday_12      -62.428    299.469  -0.208 0.834949    
dteday_13     -223.049    304.038  -0.734 0.463506    
dteday_14     -140.062    294.892  -0.475 0.635012    
dteday_15        5.849    300.051   0.019 0.984455    
dteday_16       71.741    302.381   0.237 0.812553    
dteday_17      110.364    303.210   0.364 0.716016    
dteday_18     -123.119    303.477  -0.406 0.685131    
dteday_19      -96.955    305.353  -0.318 0.750978    
dteday_20       32.372    293.048   0.110 0.912082    
dteday_21       14.492    301.649   0.048 0.961699    
dteday_22     -425.605    305.660  -1.392 0.164386    
dteday_23     -318.018    308.440  -1.031 0.302990    
dteday_24     -429.848    295.879  -1.453 0.146881    
dteday_25     -523.356    299.991  -1.745 0.081644 .  
dteday_26     -228.430    300.072  -0.761 0.446848    
dteday_27     -306.609    301.801  -1.016 0.310129    
dteday_28     -434.950    297.851  -1.460 0.144805    
dteday_29     -512.956    294.801  -1.740 0.082442 .  
dteday_30     -272.686    298.010  -0.915 0.360599    
dteday_31           NA         NA      NA       NA    
season_1     -1648.066    207.053  -7.960 1.07e-14 ***
season_2      -817.733    233.245  -3.506 0.000494 ***
season_3      -674.365    203.563  -3.313 0.000987 ***
season_4            NA         NA      NA       NA    
mnth_1          70.156    211.838   0.331 0.740643    
mnth_2         322.883    216.192   1.493 0.135907    
mnth_3         720.464    211.221   3.411 0.000697 ***
mnth_4         677.213    266.883   2.537 0.011453 *  
mnth_5         939.000    290.436   3.233 0.001301 ** 
mnth_6         648.479    294.855   2.199 0.028290 *  
mnth_7          83.366    308.870   0.270 0.787338    
mnth_8         542.150    294.465   1.841 0.066164 .  
mnth_9        1091.110    233.480   4.673 3.77e-06 ***
mnth_10        683.698    180.831   3.781 0.000174 ***
mnth_11        -26.449    166.955  -0.158 0.874187    
mnth_12             NA         NA      NA       NA    
weekday_0     -423.073    118.506  -3.570 0.000390 ***
weekday_1     -215.325    121.130  -1.778 0.076040 .  
weekday_2      -79.045    118.528  -0.667 0.505135    
weekday_3      -27.138    118.985  -0.228 0.819677    
weekday_4      -91.771    117.141  -0.783 0.433733    
weekday_5      -15.245    115.779  -0.132 0.895293    
weekday_6           NA         NA      NA       NA    
weathersit_1  2181.238    218.296   9.992  < 2e-16 ***
weathersit_2  1709.359    206.655   8.272 1.09e-15 ***
weathersit_3        NA         NA      NA       NA    
yr1           2034.550     64.501  31.543  < 2e-16 ***
holiday1      -656.683    194.868  -3.370 0.000807 ***
workingday1         NA         NA      NA       NA    
temp          4268.834    462.742   9.225  < 2e-16 ***
hum          -1340.711    325.565  -4.118 4.44e-05 ***
windspeed    -2723.585    467.388  -5.827 9.83e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 755.8 on 526 degrees of freedom
Multiple R-squared:  0.8617,	Adjusted R-squared:  0.8467 
F-statistic: 57.51 on 57 and 526 DF,  p-value: < 2.2e-16
```

Visualization of Linear regression model. In above figure red line represent the predicted values and small circle are actual values.



Conclusion

4.1 Model Evaluation

Our goal in this model is to predict the bike rental count based on seasonal and environmental settings. To achieve this goal we applied many data pre-processing techniques and then built a few models based on the data. Each model predicted the value of target variable. Now we need to determine which model is best for our use here i.e. which model has the least error. We need to evaluate and compare each model. There are several criteria to be taken into consideration while evaluating a model like (i) Predictive Performance (ii) Interpretability (iii) Computational Efficiency

In our case of Bike Renting, the latter two, Interpretability and Computation Efficiency, do not hold much significance. Therefore we will use Predictive performance as the criteria to compare and evaluate models. Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

4.1.1 MAPE

- MAPE :- Mean Absolute Percentage Error (MAPE) is a simple average of absolute percentage errors. It is a measure of prediction accuracy of a forecasting method in statistics. It can be calculated as

$$\left(\frac{1}{n}\sum \frac{|Actual - Forecast|}{|Actual|}\right) * 100$$

Defining mape function. Here y_true is the actual value and y_pred is the predicted value. It will provide the error percentage of model.

```
#defining MAPE function
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape
```

Mape value for different models are shown below:

In Python

```
#MAPE for decision tree regression
MAPE(test.iloc[:,11], predictions_DT)

29.299607655928256

#MAPE for random forest regression
MAPE(test.iloc[:,11],RF_Predictions)

13.40560971690254

#MAPE for linear regression
MAPE(testlr.iloc[:,63], predictions_LR)

15.23723008980353
```

In R

```
> MAPE(test[,12], predictions_DT)
[1] 27.25373
> MAPE(test[,12], predictions_RF)
[1] 25.43628
> MAPE(test_lr[,64],  predictions_LR)
[1] 122.2972
```

Where predictions_DT are predicted values from C50 model. predictions_RF are predicted values from random forest model. predictions_LR are predicted values from linear regression model.

4.2 Model Selection

Using MAPE, we can clearly see that, out of the three models, Random Forest performs best as it has the least error (In R as well as in python). So the selected model is Random forest with 87% accuracy in python and 75% accuracy in R.

Extracted predicted value of random forest model are saved with .csv file format.

Visualizations

5.1 Visualization of result based on season

The bar graphs shown below shows the predicted count and actual count based on seasonal conditions.

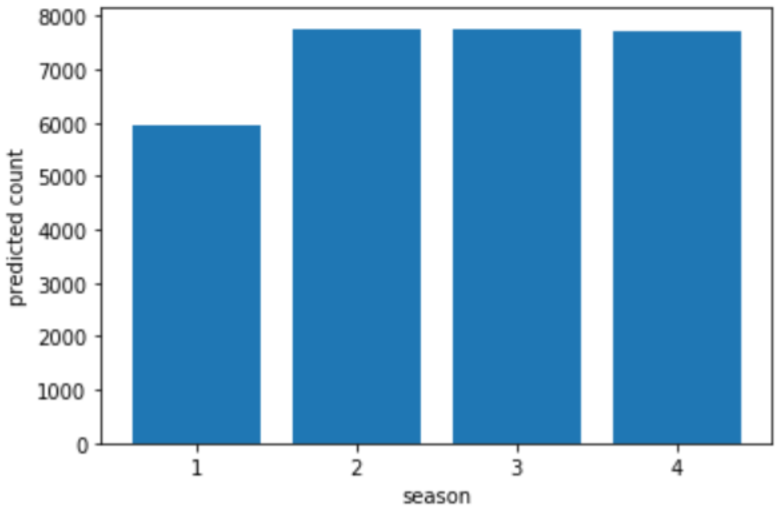


Fig 9: Bike Renting analysis using predicted count, based on season

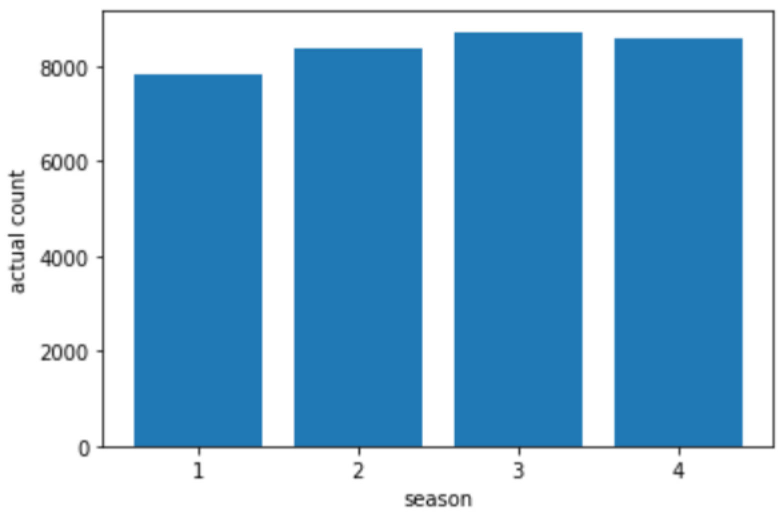


Fig 10: Bike Renting analysis using the actual count, based on season

[Season (1:springer, 2:summer, 3:fall, 4:winter)]

5.2 Visualization of result based on weather

The bar graphs shown below shows the predicted count and actual count based on weather conditions.

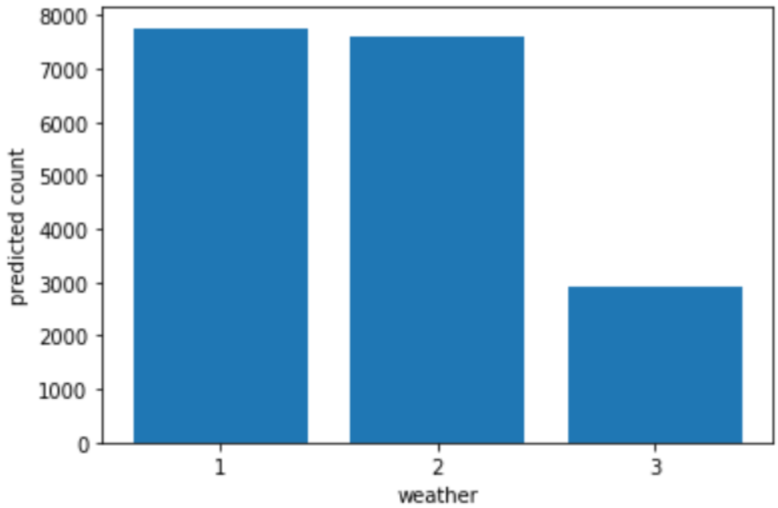


Fig 11: Bike Renting analysis using predicted count, based on weather

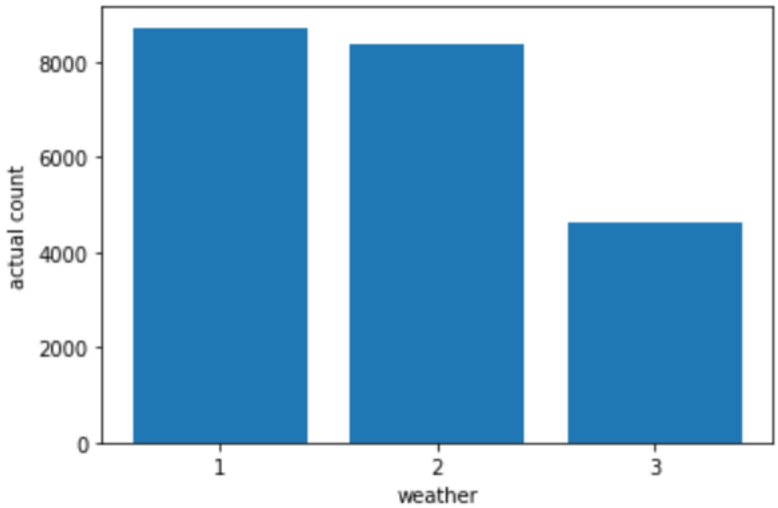


Fig 12: Bike Renting analysis using actual count, based on weather

- ✓ weathersit:

1: Clear, Few clouds, Partly cloudy, Partly cloudy

2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

Above bar graph shows predicted count and actual count based on weather conditions

According to Seasonal and weather condition bar graph we can notice that fall season and where weather conditions are clear, few or partly cloudy on these conditions bike rent count is quite high than any other condition.