Prediction of bike rental count based on environmental conditions

Shafeeq Ahmed

CONTENTS

CHAPTER 1 INTRODUCTION

1.1 Problem Statement 1

1.2 Dataset 1

CHAPTER 2 EXPLORATORY DATA ANALYSIS

2.1 Significance 3

2.2 Important Visualisations 3

CHAPTER 3 DATA PRE-PROCESSING

3.1 Need for Pre-Processing 6

3.2 Outlier detection and removal 6

3.3 Feature Selection 7

CHAPTER 4 MODEL PHASE

4.1 Model Selection 9

4.2 Multiple Linear Regression 9

4.3 Decision Tree 11

4.4 Random Forest 12

4.5 K – Nearest Neighbours 13

CHAPTER 5 MODEL EVALUATION & SELECTION

5.1 Performance Metrics 14

5.2 Mape comparison of different models 14

5.3 Anomaly in testing data 15

5.4 Conclusion 15

APPENDIX – A 16

APPENDIX – B 19

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

The objective of this project is to predict the count of bikes rented by both casual as well as registered users on a daily basis. The rental count is affected by a number of environmental factors as will be depicted by exploratory analysis of the acquired dataset. Our aim, therefore, is to build a model that can predict the count of rental bikes by exploring and taking in to account, the various features of the dataset.

1.2 DATASET

A sample of the entire dataset which we will use to build our regression model is given below in two parts: dependent and independent variables

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed
1	01-01-2011	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446
2	02-01-2011	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539
3	03-01-2011	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309
4	04-01-2011	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296
5	05-01-2011	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900

TABLE 1.1 PREDICTOR VARIABLES

The predictor variables given in the above table 1.1 is essentially a precise statistical collection of environmental and work related circumstances that can, in one way or other, necessitate renting bikes for commute.

casual	registered	cnt
331	654	985
131	670	801
120	1229	1349
108	1454	1562
82	1518	1600

TABLE 2.2 RESPONSE VARIABLES

Our task is to predict the value of response variables (sample given in table 1.2) by using these predictor variables. The number of casual users and the number of registered users make up the total count of customers who are likely to rent a bike on any given day.

CHAPTER 2

EXPLORATORY DATA ANALYSIS

2.1 SIGNIFICANCE

For any given problem statement, exploring the given dataset and acquiring a deeper knowledge of the relationship between the variables is vital for producing more precise end results. Exploratory data analysis comprises calculating statistical parameters like mean, median, correlation coefficient etc., and also more powerful method of visualisations. Visualization has the added advantage of bringing forth hidden patterns in the dataset and this helps in procuring a better business knowledge.

2.2 IMPORTANT VISUALISATIONS

Here we highlight some of the predictor variables and their influence on the number of bikes rented by the customers. The corresponding python and R codes for the figures are given in appendix.

A Scatterplot displaying the spread of rental count throughout the given duration 2011-2012 is given below in figure 2.1 with the data points colour coded with corresponding seasons.

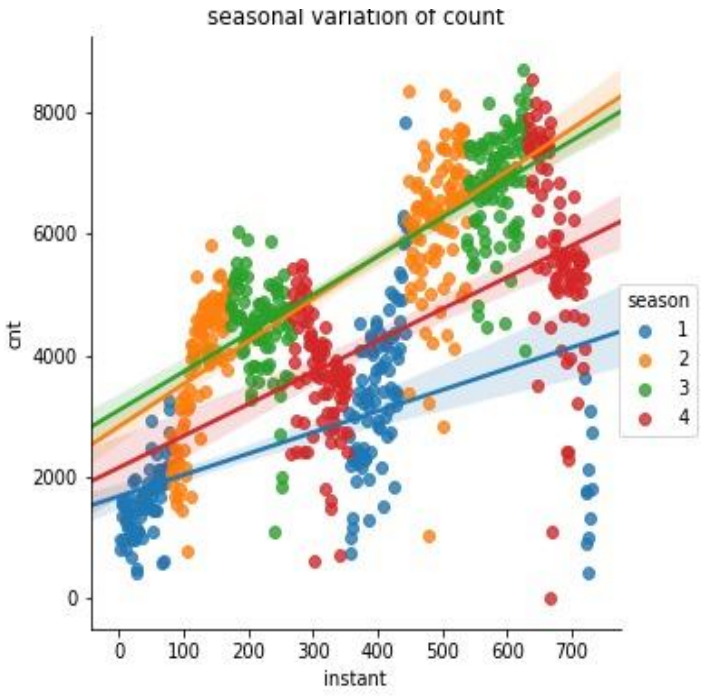


FIGURE 2.1

A lot of information is revealed in this plot including the fact that the rental count is more elevated in 2012 than 2011 as indicated by the regression lines. We could clearly see that the rental count is at its minimum during spring, starts rising throughout summer and reaches its maximum during autumn after which it begins to fall during winter. This overall inverse parabolic trend holds true for both the years

A scatter plot to depict the effect of humidity on rental count is given in figure 2.2

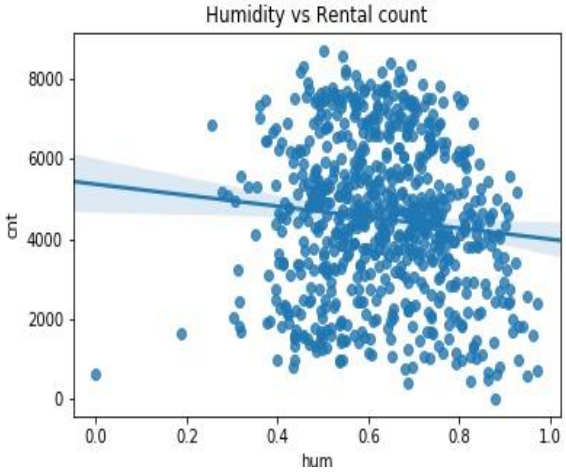


FIGURE 2.2

When we look at the humid conditions (represented by 'hum' variable) prevalent throughout 2011-2012 its influence on the rental count is not so apparent at first. But when we consider the corresponding season, there is a declining trend in rental count with increase in humidity as clearly shown in figure 2.3.

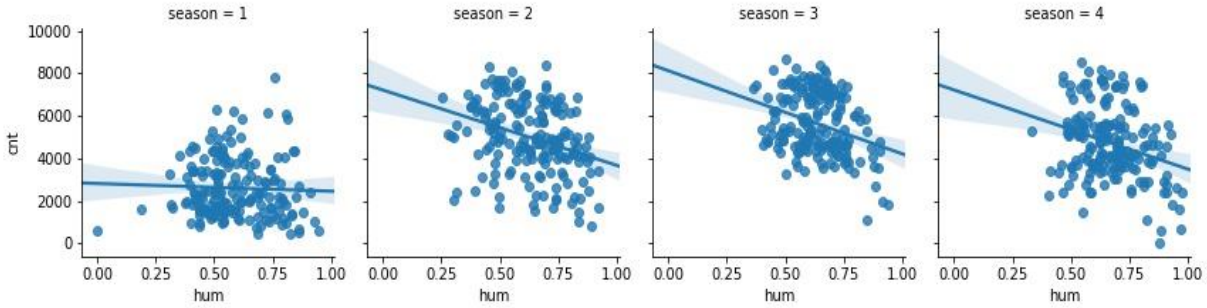


FIGURE 2.3

Hence we can safely conclude that as the season progresses from spring to winter, humidity inversely affects the rental count which also makes sense as riding a bike out in the open is less favourable in more humid conditions. This relationship also holds true for wind speed.

The relationship between the generic weather and rental count also reveals interesting information. The bar graph indicating the average number of bikes rented during various weather conditions is given on figure 2.4.

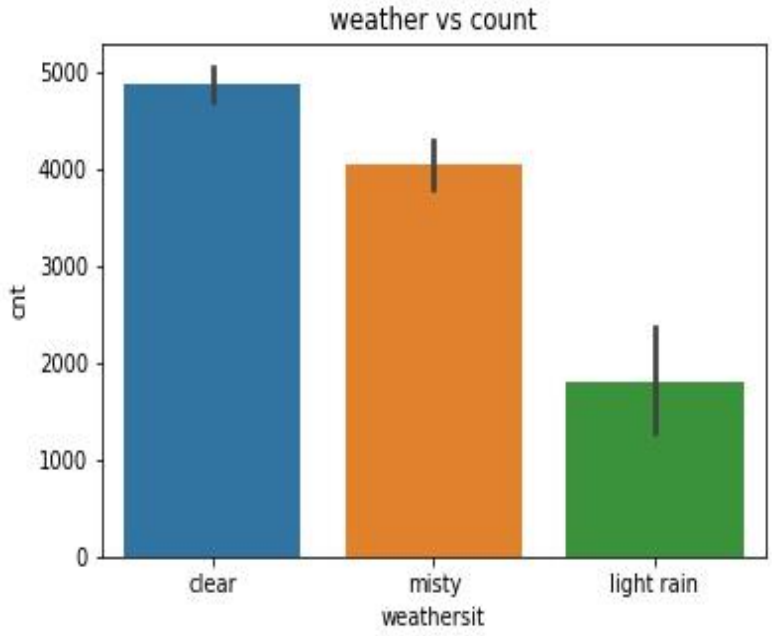


FIGURE 2.4

The average number of bikes rented when there were clear weather conditions differs from the average number of bikes rented when there were little or a decent chance for rain by a significant margin. From this, we can derive the conclusion that there is a better chance for customers renting more bikes when there is little or no chance for rain.

CHAPTER 3

DATA PRE-PROCESSING

3.1 NEED FOR PRE-PROCESSING

The raw data that we are considering for our problem statement may not be in the ideal shape that meet the standards of training our model. To name a few, it may contain irregularities like missing values, redundant variables, outliers etc., so it is a preliminary requirement for building our model that we clean our data. The subsequent topics elaborately describe the different steps involved in data pre-processing.

3.2 OUTLIER DETECTION AND REMOVAL

Outliers are data points that differ significantly from the overall spread of the data. The presence of outliers in the variables degrade the quality of learning of the model as it distorts the weightage that is assigned to the predictor variables. Removal of outliers from the dataset is mandatory before using it for further analysis. The below boxplot demonstrates the presence of outliers in all of the variables of the entire dataset.

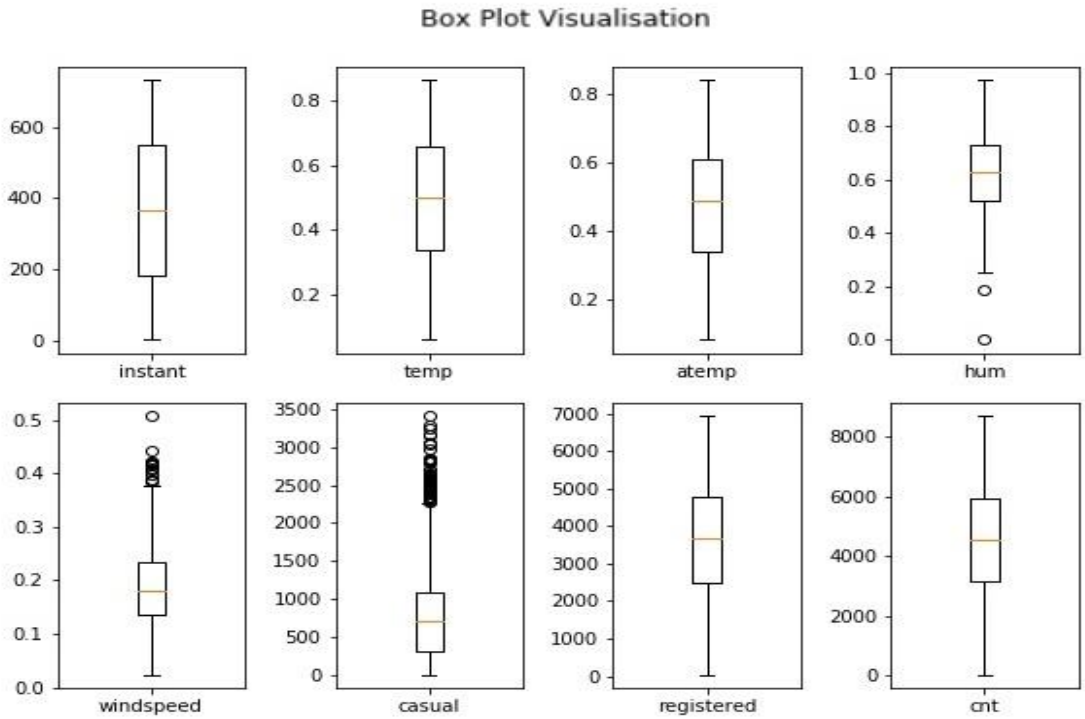


FIGURE 3.1

Figure 3.1 shows that the variables hum, windspeed and casual contains outliers which must be removed before proceeding further. The below piece of python code calculates the inter-

quantile range to find out the upper and lower extremes of variables. Then the data points that lie beyond these extremes are classified as outliers and removed.

```
for i in bk_numeric:
    q25,q75=np.percentile(bk_data.loc[:,i],[25,75])
    iqr=q75-q25
    max=q75+(1.5*iqr)
    min=q25-(1.5*iqr)
    bk_data=bk_data.drop(bk_data.loc[bk_data.loc[:,i]>max,:].index)
    bk_data=bk_data.drop(bk_data.loc[bk_data.loc[:,i]<min,:].index)
```

3.3 Feature Selection

The data acquired for resolving a particular problem statement may not always be fully relevant to the case in point. Since there may be multiple sources from where data is extracted, there is a good chance for irrelevant features to find their way in to the dataset. If a predictor variable that has no useful information to predict the outcome of a response variable is included in training set of our model, it can cause severe performance degradation of the model.

Moreover, if two or more predictor variables included in the dataset contains the same information or in technical terms, highly correlated, then the redundant information can also impact the performance of our model. Hence it is a usual practice to subject the features to statistical tests like correlation analysis to determine the level of contribution they make to the prediction of response variables. The following table illustrates the pairwise correlation between numerical variables of the dataset and the heat map visualises the same.

	instant	temp	atemp	hum	windspeed	casual	registered	cnt
instant	1.000000	0.150580	0.152638	0.016375	-0.112620	0.275255	0.659623	0.628830
temp	0.150580	1.000000	0.991702	0.126963	-0.157944	0.543285	0.540012	0.627494
atemp	0.152638	0.991702	1.000000	0.139988	-0.183643	0.543864	0.544192	0.631066
hum	0.016375	0.126963	0.139988	1.000000	-0.248489	-0.077008	-0.091089	-0.100659
windspeed	-0.112620	-0.157944	-0.183643	-0.248489	1.000000	-0.167613	-0.217449	-0.234545
casual	0.275255	0.543285	0.543864	-0.077008	-0.167613	1.000000	0.395282	0.672804
registered	0.659623	0.540012	0.544192	-0.091089	-0.217449	0.395282	1.000000	0.945517
cnt	0.628830	0.627494	0.631066	-0.100659	-0.234545	0.672804	0.945517	1.000000

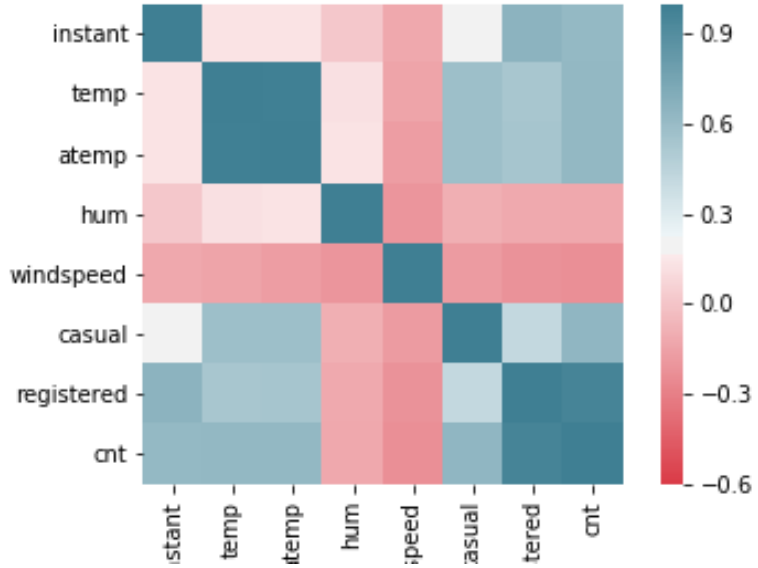


FIGURE 3.2

The value of correlation coefficient ranges between +1 to -1, +1 being highly positively correlated and -1 being highly negatively correlated. As we can see from the above heatmap, the diagonal is densely coloured as each variable is highly correlated with itself, but the variables temp and atemp are very highly correlated (correlation coefficient = 0.99). Hence as part of feature selection, we can remove the variable atemp and thus the redundant information carried by it.

As for as irrelevant variables are concerned, we can safely remove the below two variables for the cited reasons.

1. *instant → we can remove this variable as this is a mere index of the observations*
2. *dteday → this variable can be removed as the mnth and yr variables sufficiently cover the information provided by it.*

CHAPTER 4

MODEL PHASE

4.1 MODEL SELECTION

As we have already defined in our problem statement, the main goal of this project is to predict the count of bikes rented on any given day and hence, our problem statement falls under the category "regression". We can choose a variety of machine learning algorithms for regression type. For this project, the following 4 algorithms are chosen and their performance is evaluated on the same train and test data to select the best one of them.

1. *Multiple Linear Regression*
2. *Decision Tree*
3. *Random Forest*
4. *KNN*

4.2 MULTIPLE LINEAR REGRESSION

Separate linear regression models are developed for each of the response variables 'casual' and 'registered' and the summary of the linear models are given below:

```
> summary(lm_model_1)

Call:
lm(formula = casual ~ season + yr + mnth + holiday + weekday +
    workingday + weathersit + temp + hum + windspeed, data = train_sample)

Residuals:
    Min      1Q  Median      3Q     Max
-970.74 -163.65   -7.69  147.49 1068.42

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   609.18      98.91   6.159 1.48e-09 ***
season2       177.35      73.56   2.411 0.016269 *
season3        82.29      87.40   0.942 0.346831
season4        31.41      73.48   0.428 0.669185
yr1           229.47      24.50   9.367  < 2e-16 ***
mnth2          41.14      57.63   0.714 0.475639
mnth3         302.36      68.08   4.441 1.10e-05 ***
mnth4         304.81     102.44   2.976 0.003063 **
mnth5         330.44     110.26   2.997 0.002861 **
mnth6         207.07     118.17   1.752 0.080303 .
mnth7         220.90     131.34   1.682 0.093207 .
mnth8         329.69     125.15   2.634 0.008686 **
mnth9         338.70     108.81   3.113 0.001957 **
mnth10        402.91      99.67   4.043 6.10e-05 ***
mnth11        242.46      92.91   2.610 0.009325 **
mnth12         88.43      73.85   1.197 0.231712
holiday1      303.95      84.61   3.592 0.000359 ***
weekday1     -557.70      46.18 -12.076  < 2e-16 ***
weekday2     -629.27      45.54 -13.817  < 2e-16 ***
```

```
weekday3     -651.63      45.27 -14.393  < 2e-16 ***
weekday4     -609.32      44.61 -13.658  < 2e-16 ***
weekday5     -455.64      45.46 -10.022  < 2e-16 ***
weekday6      112.83      48.78   2.313 0.021125 *
workingday1       NA         NA      NA       NA
weathersit2  -103.33      32.78  -3.152 0.001716 **
weathersit3  -364.74      83.50  -4.368 1.52e-05 ***
temp         1233.91     174.70   7.063 5.34e-12 ***
hum          -423.54     127.81  -3.314 0.000985 ***
windspeed    -833.23     180.95  -4.605 5.22e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 274.3 on 512 degrees of freedom
Multiple R-squared:  0.7425, Adjusted R-squared:  0.7289
F-statistic: 54.67 on 27 and 512 DF,  p-value: < 2.2e-16
```

```
> summary(lm_model_2)

Call:
lm(formula = registered ~ season + yr + mnth + holiday + weekday +
    workingday + weathersit + temp + hum + windspeed, data = train_sample)

Residuals:
    Min      1Q  Median      3Q     Max
-3481.3  -279.0    69.6   362.8  1524.9


Coefficients: (1 not defined because of singularities)

             Estimate Std. Error t value Pr(>|t|)
(Intercept)   834.12     224.65   3.713 0.000227 ***
season2       644.72     167.08   3.859 0.000129 ***
season3       815.75     198.50   4.110 4.62e-05 ***
season4      1487.30     166.88   8.912  < 2e-16 ***
yr1          1732.24      55.64  31.133  < 2e-16 ***
mnth2         117.00     130.89   0.894 0.371791
mnth3         182.57     154.62   1.181 0.238254
mnth4         159.14     232.66   0.684 0.494298
mnth5         401.47     250.44   1.603 0.109532
mnth6         181.97     268.38   0.678 0.498059
mnth7        -336.94     298.31  -1.129 0.259220
mnth8          16.00     284.26   0.056 0.955137
mnth9         540.03     247.14   2.185 0.029334 *
mnth10        -22.64     226.37  -0.100 0.920390
mnth11       -305.99     211.01  -1.450 0.147637
mnth12        -88.65     167.74  -0.528 0.597402
holiday1    -1511.52     192.17  -7.866 2.20e-14 ***
weekday1      981.11     104.89   9.354  < 2e-16 ***
weekday2     1103.68     103.44  10.670  < 2e-16 ***
weekday3     1178.79     102.83  11.463  < 2e-16 ***
weekday4     1132.13     101.33  11.173  < 2e-16 ***
weekday5     1027.74     103.26   9.953  < 2e-16 ***
weekday6      334.58     110.80   3.020 0.002656 **
workingday1       NA         NA      NA       NA
weathersit2  -401.37      74.46  -5.391 1.07e-07 ***
weathersit3 -1626.92     189.66  -8.578  < 2e-16 ***
temp         3090.73     396.78   7.790 3.77e-14 ***
hum         -1012.81     290.28  -3.489 0.000526 ***
windspeed   -1832.64     410.99  -4.459 1.01e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 623.1 on 512 degrees of freedom
Multiple R-squared:  0.853,  Adjusted R-squared:  0.8453
F-statistic:   110 on 27 and 512 DF,  p-value: < 2.2e-16
```

Each predictor variable is assigned a weightage and star-coded to indicate their significance. The more stars they have, the more important they are, to the prediction.

4.3 DECISION TREE

The decision trees that are developed for predicting the casual and registered users count are given in figure 4.1 and 4.2

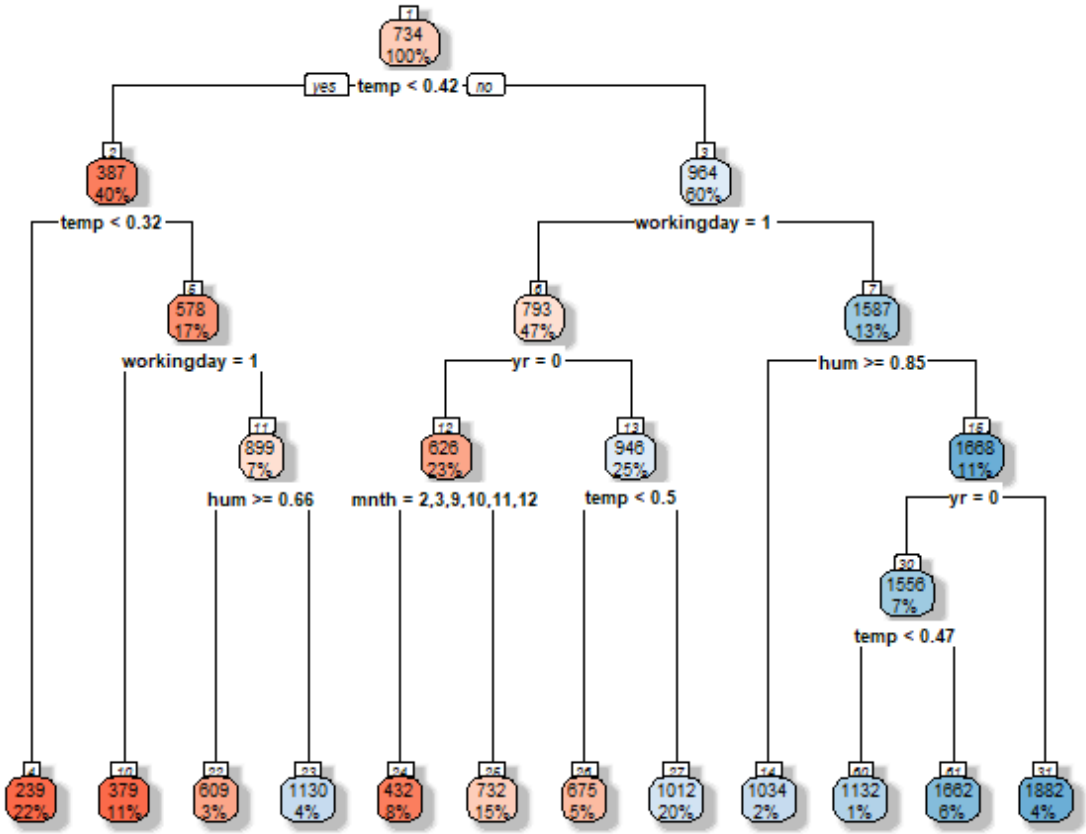


FIGURE 4.1 DECISION TREE FOR PREDICTING CASUAL USERS

For casual users, temperature is chosen as the root node and the predicted values are being passed down to other sub nodes based on their preceding node values. This process continues till the end result is achieved.

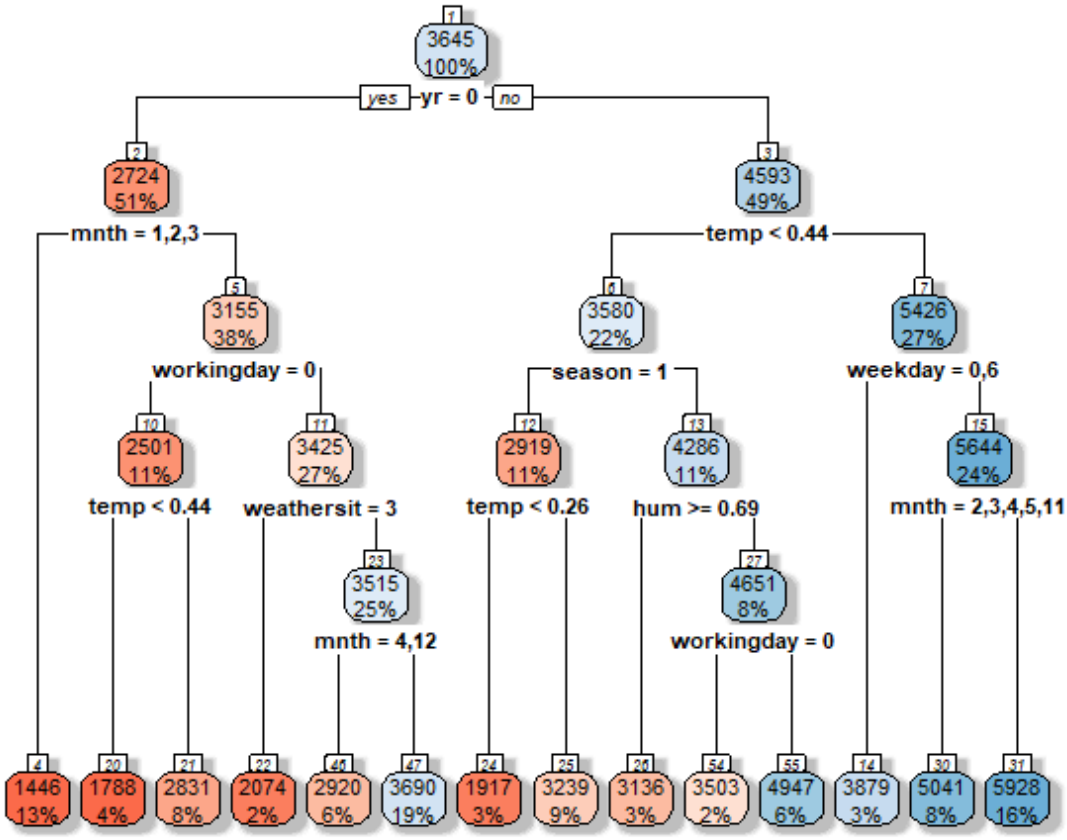


FIGURE 4.2 DECISION TREE FOR PREDICTING REGISTERED USERS

For registered users, we can see that the prediction flow starts from year as root node and bifurcates to go down the predictor variables.

4.4 RANDOM FOREST

Random forest is an ensemble technique which uses multiple decision trees to further enhance the accuracy of prediction. A suitable number of decision trees is chosen and their result are iterated to produce a collective result. The prediction, as far as this project is concerned has improved significantly in terms of performance by introducing Random Forest ensemble technique. The details regarding the performance statistics is illustrated in a detailed manner in the following chapter.

R-code

```
# RANDOM FOREST
rf_model_1=randomForest(casual~season+yr+mnth+holiday+weekday+workingday+weathersit+temp+hum+windspeed,train_sample,importance = TRUE, ntree = 500)
rf_model_2=randomForest(registered~season+yr+mnth+holiday+weekday+workingday+weathersit+temp+hum+windspeed,train_sample,importance = TRUE, ntree = 500)
RF_predictions=final_model(rf_model_1,rf_model_2)
```

4.5 K – NEAREST NEIGHBOURS

K - Nearest Neighbours (KNN) is a distance based algorithm which predicts the values of the test cases by comparing the Euclidean distance between other observations and calculating the mean of k values whose distances are minimum and assigning it for the value of the test cases. The peculiar feature of this algorithm is that the model remains lazy during the training phase and uses the training set only during the predicting phase. This algorithm is also considered for our project and the performance metrics are given in the next chapter.

R-code

```
#KNN
bk_data_KNN=bk_data
bk_data_KNN[-train_index,c("casual","registered","cnt")]=NA
bk_data_KNN=knnImputation(bk_data_KNN,k=3)
```

CHAPTER 5

MODEL EVALUATION & SELECTION

5.1 PERFORMANCE METRICS

We have developed four models so far to predict the number of bikes rented by casual and registered users on any given day. In order to select the best suitable model, we have to use some sort of performance metrics which can be used as a touchstone to compare the performance of different model. We are particularly interested in reducing the difference between the actual and predicted values and thus Mean Absolute Percentage Error (MAPE) is much suitable for our analysis. We have created a simple function as shown below to calculate the mape of different models.

```
mape= function(actual,predicted) {(mean(abs(actual-predicted)/actual)*100)
}
```

5.2 MAPE COMPARISON OF DIFFERENT MODELS

The mape value of different models are given below. For the sake of exact comparison, the same set of test and training samples are used for calculating the MAPE of different models. This way, we can get a clear picture of how efficient the individual models learn and predict the output given that the samples they are dealing with, are exactly the same. The MAPE values for casual users, registered users and the total count for different models are illustrated below

1. *Simple Linear Regression:*

```
> mape(test_sample$casual,reg_predictions$casual)
[1] 54.92737
> mape(test_sample$registered,reg_predictions$registered)
[1] 15.9978
> mape(test_sample$cnt,reg_predictions$cnt)
[1] 14.14265
```

2. *Decision Tree:*

```
> mape(test_sample$casual,dec_predictions$casual)
[1] 59.38854
> mape(test_sample$registered,dec_predictions$registered)
[1] 19.63741
> mape(test_sample$cnt,dec_predictions$cnt)
[1] 20.04064
```

3. *Random Forest:*

```
> mape(test_sample$casual,RF_predictions$casual)
[1] 25.33787
> mape(test_sample$registered,RF_predictions$registered)
[1] 9.468739
> mape(test_sample$cnt,RF_predictions$cnt)
[1] 9.633818
```

4. *Knn:*

```
> mape(test_sample$casual,bk_data_KNN[-train_index,"casual"])
[1] 47.7188
> mape(test_sample$registered,bk_data_KNN[-train_index,"registered"])
[1] 21.5998
> mape(test_sample$cnt,bk_data_KNN[-train_index,"cnt"])
[1] 20.39411
```

5.3 ANOMALY IN TESTING DATA

Before proceeding with model selection, it is necessary to point out one anomaly in the dataset which, if included in testing data significantly affects MAPE value.

```
> bk_mdata[614,]
    season yr mnth holiday weekday workingday weathersit temp  hum windspeed casual registered cnt
614      4  1   10       0       1          1          3 0.44 0.88    0.3582      2         20  22
```

The value of the response variables vary so significantly from the rest of the data that the particular observation becomes an anomaly and degrades the performance metric. The observation does not fall under the category of outliers. However, removing this particular anomaly brings the performance metrics of the models back on track as indicated below.

```
> test_sample=filter(test_sample,cnt!=22)
> mape(test_sample$casual,RF_predictions$casual)
[1] 21.78166
> mape(test_sample$registered,RF_predictions$registered)
[1] 8.537699
> mape(test_sample$cnt,RF_predictions$cnt)
[1] 7.97806
```

5.4 CONCLUSION

From the above detailed portrayal of the performance statistics of individual models, it is evident that Random Forest performs significantly well and the prediction prevails over other models by a clear margin. Hence Random forest can be selected for predicting the bike rental count for future test cases and the same can be used for deployment.

APPENDIX - A

Python code for Fig 2.1 (seasonal variation of count)

```
sns.lmplot('instant','cnt',data=bk_data,hue='season')
plt.title("seasonal variation of count")
```

Python code for Fig 2.2 & Fig 2.3 (humidity vs rental count with respect to season)

```
sns.regplot('windspeed','cnt',data=bk_data)
plt.title("Humidity vs Rental count")
```

```
fct=sns.FacetGrid(bk_data,col='season')
fct=fct.map(sns.regplot,'hum','cnt')
```

Python code for Fig 2.4 (weather vs count)

```
sns.barplot('weathersit','cnt',data=bk_data)
plt.xticks([0,1,2],['clear','misty','light rain'])
plt.title('weather vs count')
```

Python code for Fig 3.1 (Box Plot visualisation of numerical variables)

```
fig,axes=plt.subplots(2,4,figsize=(8,6))
axes=axes.reshape(1,8)
axes=axes.flatten()
for i,j in enumerate(bk_numeric):
    axes[i]=axes[i].boxplot(bk_data.loc[:,j],labels=[j])
fig.suptitle("Box Plot Visualisation",y=1.05)
fig.tight_layout()
```

Python code for Fig 3.2 (Correlation Heat map)

```
bk_corr=bk_data.loc[:,bk_numeric]
bk_corr=bk_corr.corr()
pl=sns.diverging_palette(10,220,as_cmap=True)
sns.heatmap(bk_corr,cmap=pl,square=True,vmin=-0.6)
```

R - Code for Fig 4.1 & Fig 4.2 (decision tree for predicting casual and registered users)

```
library(rpart.plot)
rpart.plot(dec_model_1, box.palette="RdBu", shadow.col="gray", nn=TRUE)
rpart.plot(dec_model_2, box.palette="RdBu", shadow.col="gray", nn=TRUE)
```

Modules used in Python

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import statsmodels.api as sr
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn import tree
from fancyimpute import KNN
```

Libraries used in R

```
library(corrgram)
library(caret)
library(randomForest)
library(rpart)
library(rpart.plot)
library(usdm)
library(DMwR)
library(dplyr)
```

Python Code for Models:

```
#Linear regression
bk_model=sr.OLS(train.iloc[:,10:13],train.iloc[:,0:10]).fit()
predict_lin=bk_model.predict(test.iloc[:,0:10])
predict_lin.columns=['casual','registered','count']
predict_lin['actual']=test.iloc[:,12]
```

```
#Decision Tree
bk_model_dt=DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0:10],train.iloc[:,10:13])
predict_dt=pd.DataFrame(bk_model_dt.predict(test.iloc[:,0:10]),columns=['casual','registered','cnt'])
test1=test.reset_index(drop=True)
predict_dt['actual']=test1.iloc[:,12]
```

```
#Random Forest
bk_model_rf=RandomForestRegressor(n_estimators=10).fit(train.iloc[:,0:10],train.iloc[:,10:13])
predict_rf=pd.DataFrame(bk_model_rf.predict(test.iloc[:,0:10]),columns=['casual','registered','cnt'])
test1=test.reset_index(drop=True)
predict_rf['actual']=test1.iloc[:,12]
```

```
#KNN
bk_KNN=bk_mdata.copy()
bk_KNN.loc[test.index,['casual','registered','cnt']]=np.nan
bk_KNN=pd.DataFrame(KNN(k=3).complete(bk_KNN),columns=bk_KNN.columns)
predict_KNN=bk_KNN.loc[test.index,['casual','registered','cnt']]
predict_KNN['actual']=test['cnt']
```

Python Code for MAPE of different Models:

```
#linear regression
print("casual:",MAPE(test['casual'],predict_lin['casual']))
print("registered:",MAPE(test['registered'],predict_lin['registered']))
print("cnt:",MAPE(test['cnt'],predict_lin['count']))
```

```
casual: 62.47306132744929
registered: 17.22664505281096
cnt: 17.986065466586496
```

```
#Decision Tree
print("casual:",MAPE(test1['casual'],predict_dt['casual']))
print("registered:",MAPE(test1['registered'],predict_dt['registered']))
print("cnt:",MAPE(test1['cnt'],predict_dt['cnt']))
```

```
casual: 67.711694619494
registered: 24.295246625096027
cnt: 22.91470312948361
```

```
#Random Forest
print("casual:",MAPE(test1['casual'],predict_rf['casual']))
print("registered:",MAPE(test1['registered'],predict_rf['registered']))
print("cnt:",MAPE(test1['cnt'],predict_rf['cnt']))
```

```
casual: 30.46961092032938
registered: 13.585314124053458
cnt: 13.62352188915055
```

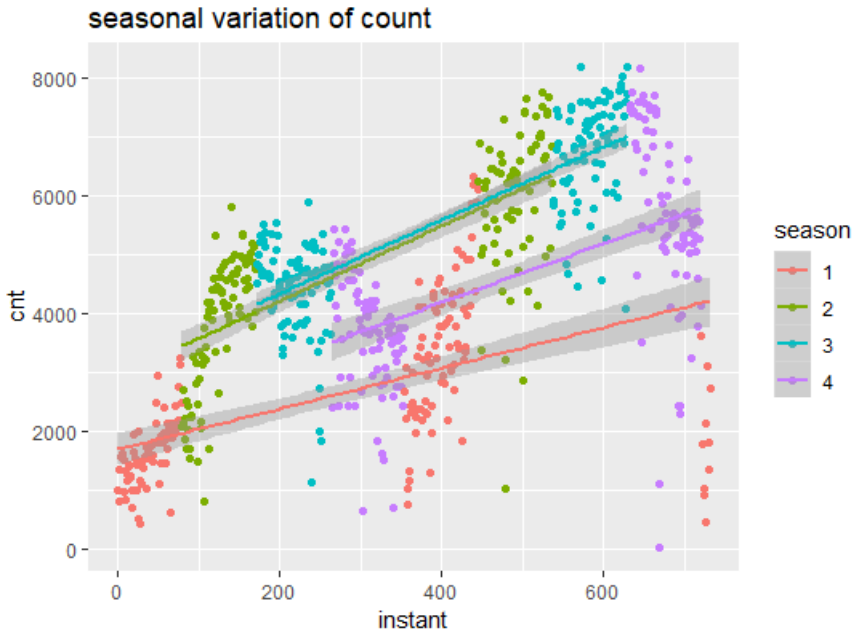
```
#KNN
print("casual:",MAPE(test['casual'],predict_KNN['casual']))
print("registered:",MAPE(test['registered'],predict_KNN['registered']))
print("cnt:",MAPE(test['cnt'],predict_KNN['cnt']))
```

```
casual: 45.873896434153636
registered: 21.618458564163777
cnt: 23.61904304290119
```

APPENDIX – B

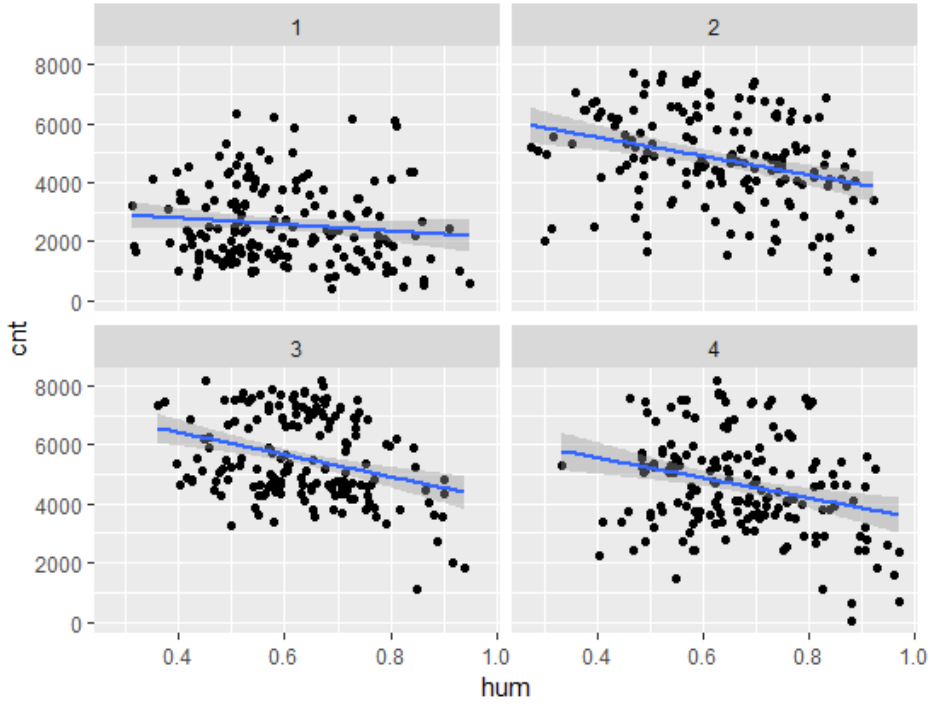
Exploratory Data Analysis –R:

Seasonal variation of count (analogous to fig 2.1)



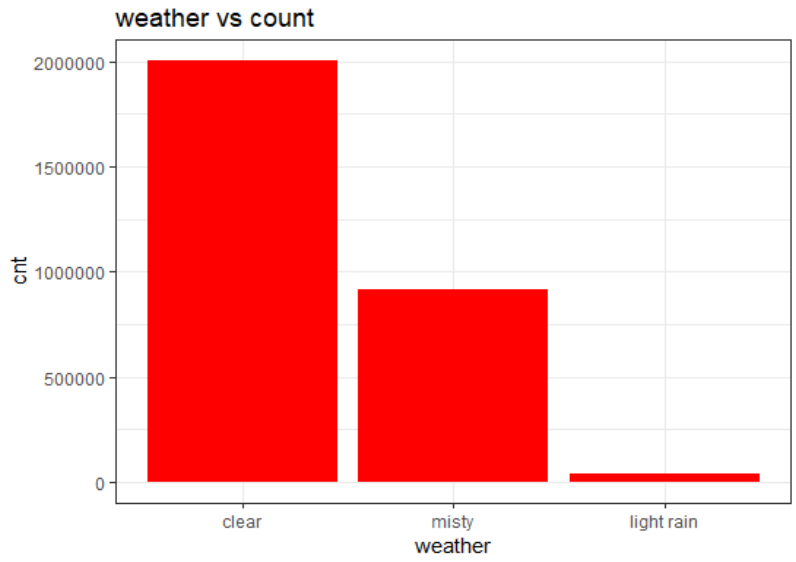

```
ggplot(bk_data,aes(x=instant,y=cnt,colour=season))+
  geom_point()+geom_smooth(method='lm')+
  labs(title="seasonal variation of count")
```

Humidity vs count for various seasons (analogous to fig 2.3)



```
ggplot(bk_data,aes(x=hum,y=cnt))+
  geom_point()+geom_smooth(method='lm')+
  facet_wrap(~season)
```

Weather vs count (analogous to fig 2.4)




```
ggplot(bk_data,aes(x=weathersit,y=cnt))+
  geom_bar(fill='red',stat='identity')+theme_bw()+
  scale_x_discrete(breaks=c(1,2,3),
                   labels=c("clear","misty","light rain"))+
  labs(x='weather',title="weather vs count")
```