**BIKE RENTING**

*done by* ***SAMUEL JOHN***

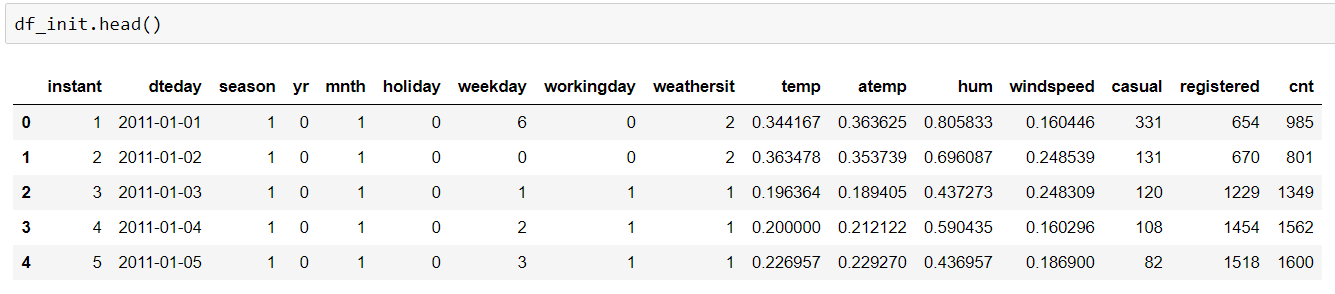
**CONTENTS**

1. **Introduction**
   1. Problem Statement
   2. Data
2. **Methodology**
   1. Pre-Processing
   2. Modeling
3. **Conclusion**
   1. Model Evaluation
   2. Model Selection
4. **Appendix A – Python Code**
5. **Appendix B – R Code**
6. **INTRODUCTION**
   1. **Problem Statement**

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings. We are provided with a public dataset that contains the counts of Bikes that were rented per day along with the climatic, seasonal and other usage patterns. We are expected to develop a regression algorithm to predict the number of Bikes that can be rented based on this pattern.

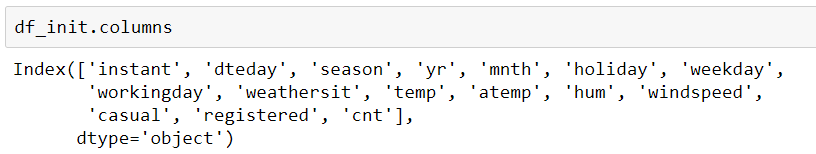
* 1. **Data**

Our task is to create model that can predict daily bike rental count based on the available data. Below is a sample of the data along with the variables.



The data was given as a single dataset and it has 731 rows and 16 columns.

The columns are as mentioned below.



The target variable is cnt. We have to predict this count using the remaining variables.

1. **METHODOLOGY**
   1. **Pre-Processing**

The initial step in any project is to clean the data that is handed over to us. There are various parts to it. This phase is also known as Exploratory Data Analysis. During this phase, we clean the data in such a way that it is suitable for giving as an input to the model.

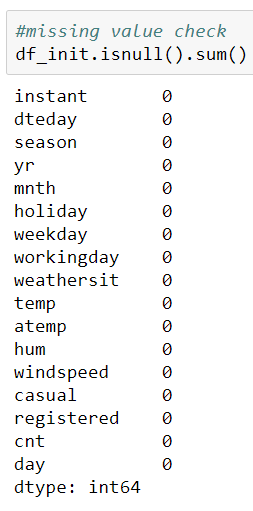
For this project, the first pre-processing step that we tried to do was a bit of feature engineering. We could see that a new feature called ‘day’ of a month can be created from an existing variable ‘dteday’. We used the below code to create this new feature.



The next pre-processing step that we tried to do was Missing Value Analysis.

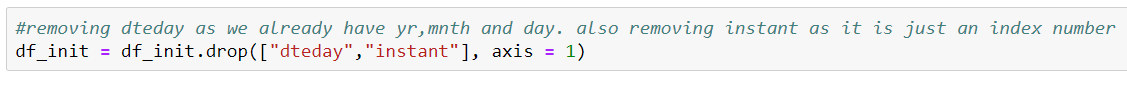
Usually we check whether there are any missing values and treat them accordingly.

Using the below code, we checked whether there any missing values.

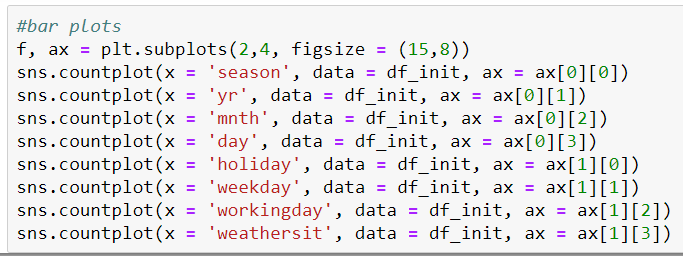


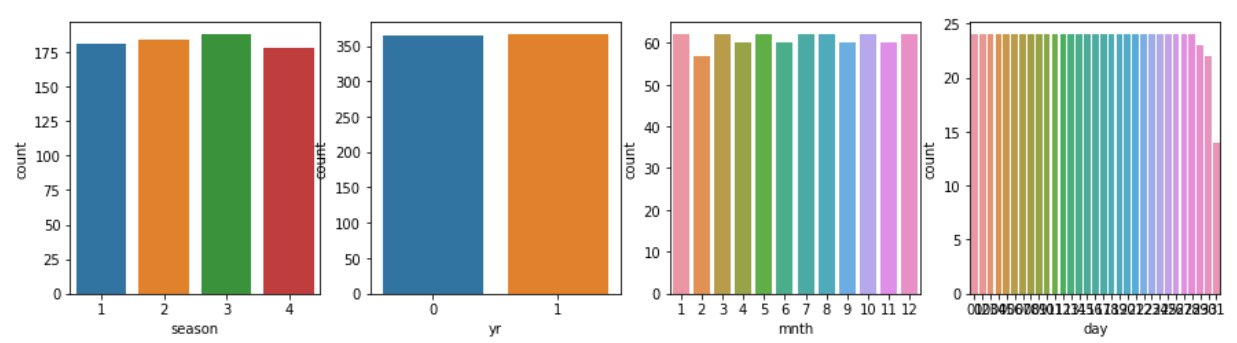
From the above snippet, we can see that number of missing values is 0.

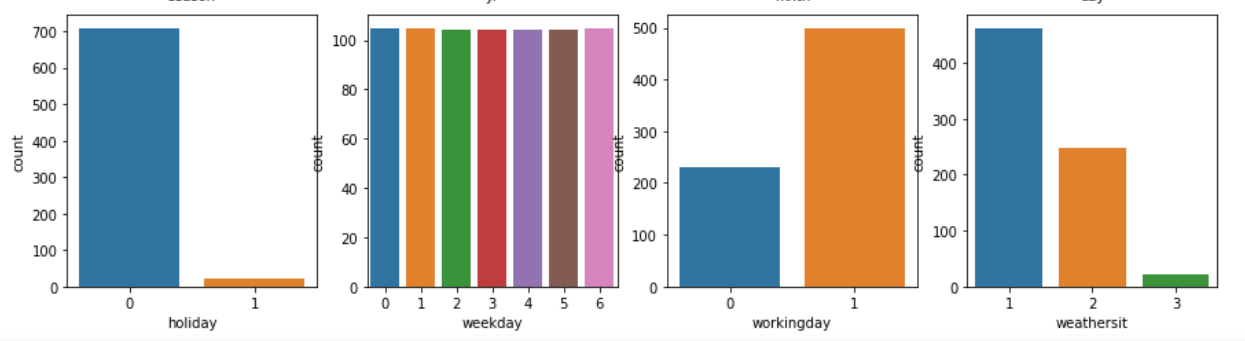
We can see that the variable ‘dteday’ can be removed as the variables day, mnth and yr are already derived from it. The variable ‘instant’can also be removed as it is just and index number.



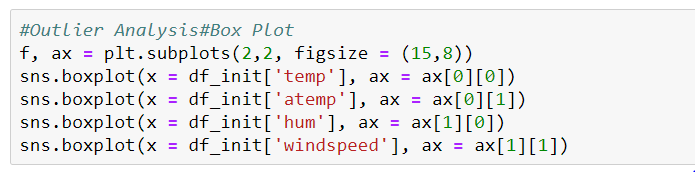
Next, we tried to do some visualisation of the categorical variables. We plotted a bar plot/count plot to check this.

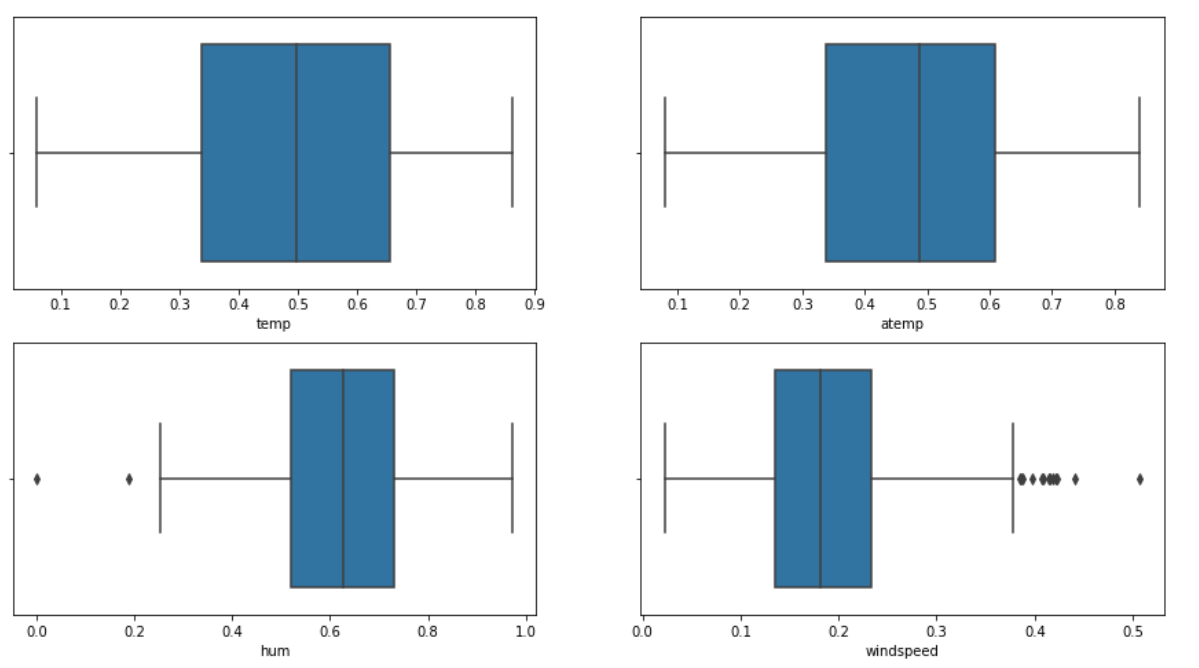




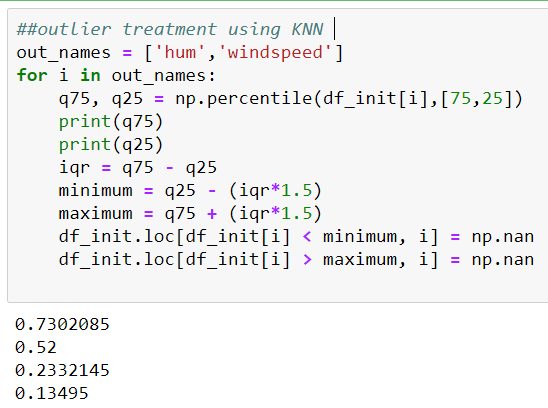


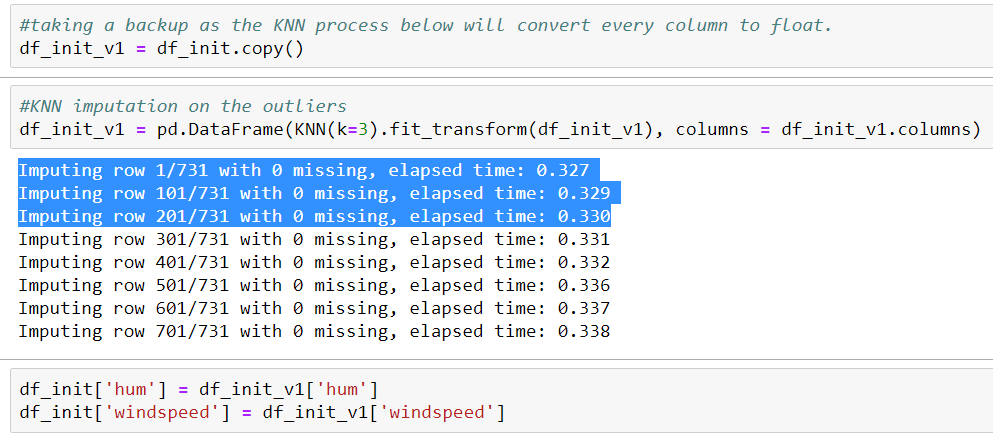
The next step which we did for data pre-processing was outlier analysis. For that first we need to plot the continuous variables in a box plot.



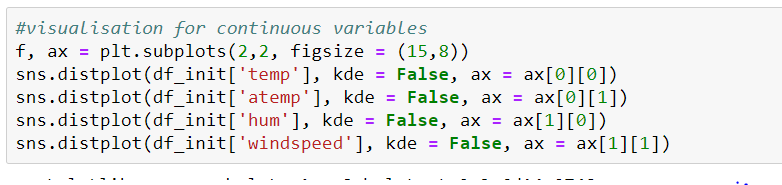


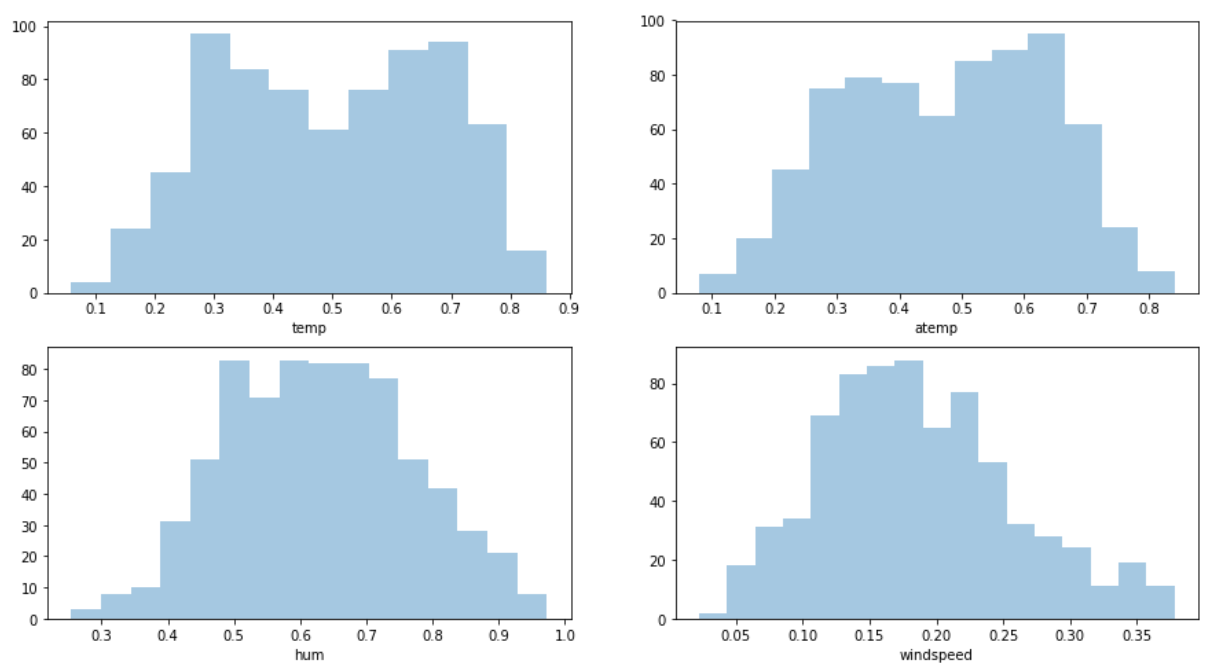
As we can see, there are outliers in the variables ‘hum’ and ‘windspeed’. We are going to remove those values and convert them to nan. Then we are going to impute values to them using KNN method.





Now, we are going to visualize the continuous variables using histograms.

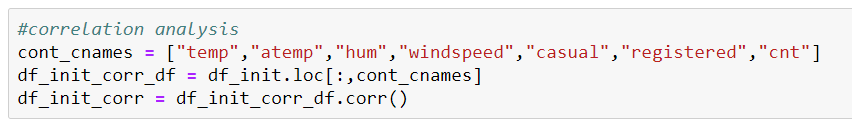




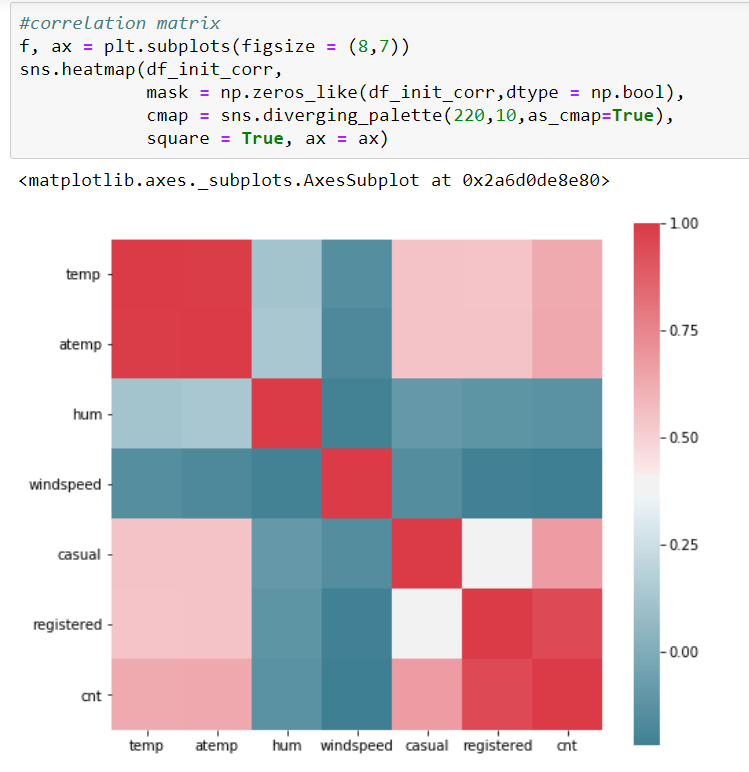
The next part of the pre-processing journey is the Feature Selection.

We are going to this for numeric variables. This can be done through Correlation Analysis.

This was done as per the below code snippets.



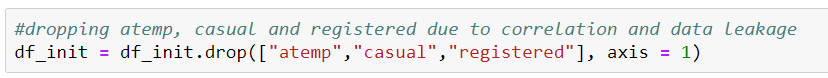
The Correlation plot is below.



From the diagram, we can see that ‘atemp’ and ‘temp’ are highly correlated. So we have to drop any one of the variables from these pairs of correlated variables. I decided to drop the variable ‘atemp’.

Also, if you see the variables casual and registered can add to data leakage in our project as they are also basically the counts, which in turn is our target variable.

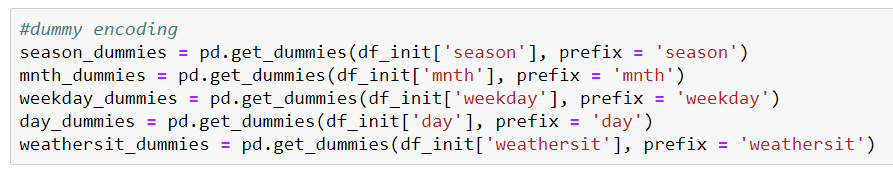
I used the below code snippet to remove them.



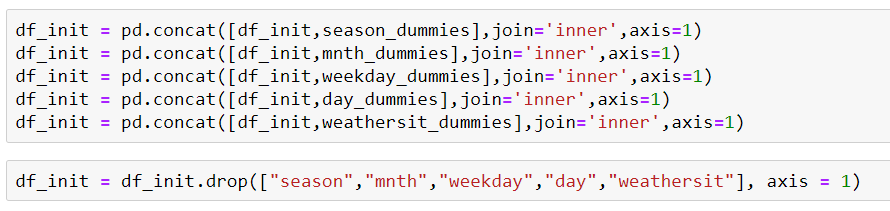
The next pre-processing technique which we did was encoding. This involves converting the character values to numerics. For variables with just two values, such as True or False, Yes or No, we can use Label Encoding. For variables which has more than 2 values, we can use dummy encoding.

For this project we won’t have to do label encoding as all the categorical variables with 2 values were given as ‘0’s and ‘1’s.

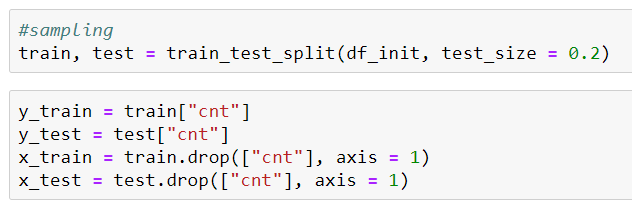
There are a few categorical variables with multiple values in it. We have to perform dummy encoding for those variables.



Then we removed the parent variables and then concatenated the dummy-variables to the master data.



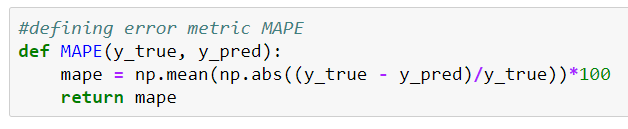
The last pre-processing technique we used was sampling. Before feeding the data to machine learning algorithms, we always have to split the data into training and test data. To divide the data into training and test datasets we used the below code. I used stratified sampling and moved 20% of the data to test dataset.



* 1. **Modeling**

Before starting the modelling process, we need to decide and define the error metric that we are going to use for this project. For this project, we are going to use the Error Metric MAPE, as it is a regression problem. MAPE stands for Mean Absolute Percentage Error.

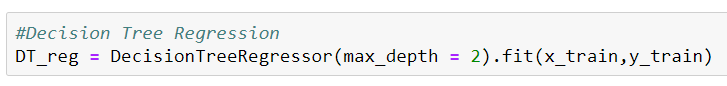
We are going to use the below snippet of code for defining MAPE.



1. **Decision Tree Model**

The first machine learning algorithm that we tried is Decision tree. It is an ML algorithm that follows tree-based learning. It’s a predictive model based on a branching series of Boolean tests.

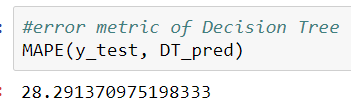
The model was trained by using the below code snippet.



The test dataset was predicted by using the below code snippet.



We are going to use the error metric that we defined earlier to verify our model.

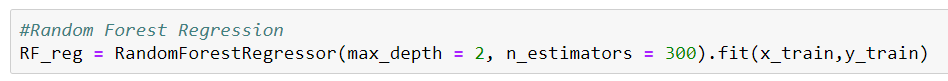


As per the above snippet, we can see that we have received an MAPE of 28.2%. Let’s try few more models and see the error metric of them as well.

1. **Random Forest Model**

Random Forest is an ensemble technique. It consists of many decision trees. This method combines Breiman’s bagging idea and random selection of features.

Using the below code snippet, we train the Random Forest model.

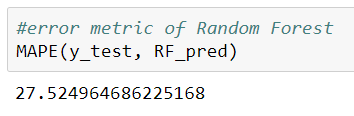


We should give the number of trees as quite low in the beginning. We should keep increasing the number of trees to see if the error is reducing and select a suitable number of trees according to get the better error metric.

Once the model is trained, we should predict the test data. Here, we did this using the below code snippet.



Next, we should compare our predictions with the actual output using the MAPE error metric.

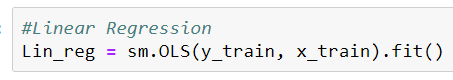


This model has given us an MAPE of 27.52%. Let’s check a few more algorithms.

1. **Linear Regression model**

Linear Regression is a statistical model which can be used to find linear relationship between the target variable and one or more predictors.

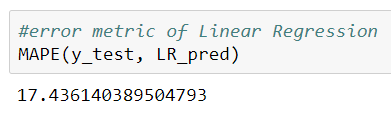
Using the below code snippet, we can train the linear regression model.



Once the model is trained, we can use the below code snippet to test the test dataset.



Next, we should compare our predictions with the actual output using the MAPE error metric.



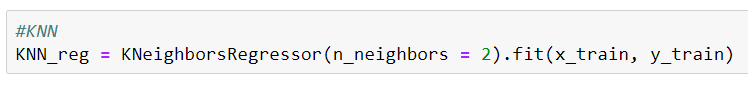
On checking the MAPE, we get 17.43%. It is lower than that of the tree models, which is a good sign.

1. **KNN Model**

KNN is a simple algorithm that stores all available cases and predicts new cases based on a similarity measure.

Before using the KNN model, we have to do feature scaling to bring all the variables in a similar scale. But in our case, all our variables are already in a similar scale.

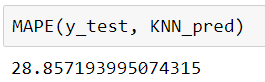
We can train the model using the below code snippet.



After the model is trained, we can test the test dataset using the below code snippet.



Next, we have to compare the actual test results with our predictions.



As per the above results, we have received an MAPE of 28.85%.

1. **CONCLUSION**

We have used the error metric MAPE to assess the performance of the various models. Metrics of the various models we used are as follows.

**Decision Tree:**

MAPE = 28.29%

**Random Forest:**

MAPE = 27.52%

**Linear Regression:**

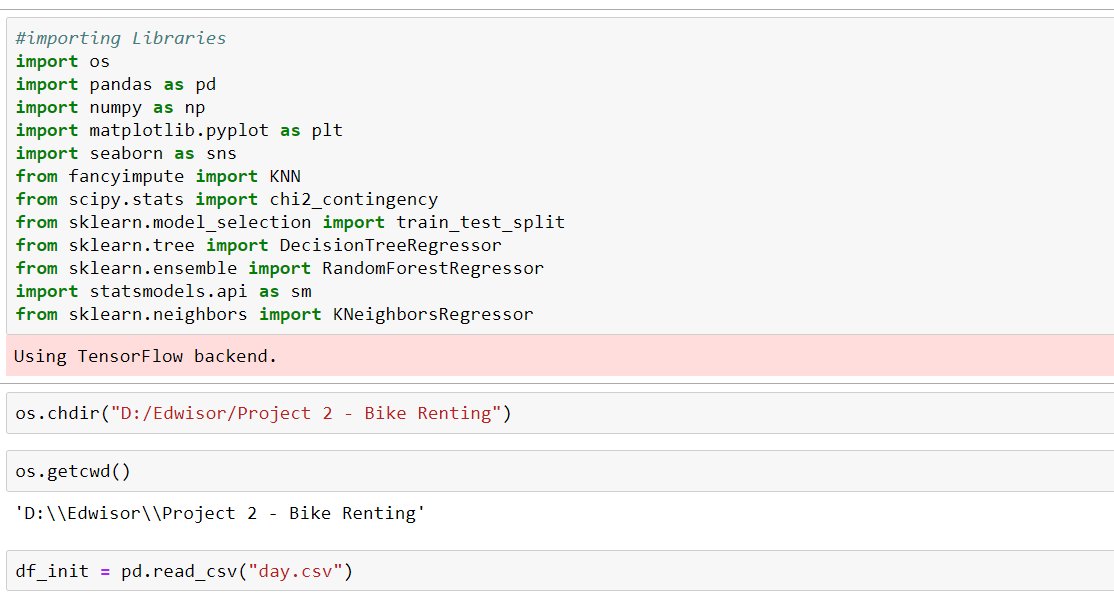
MAPE = 17.43%

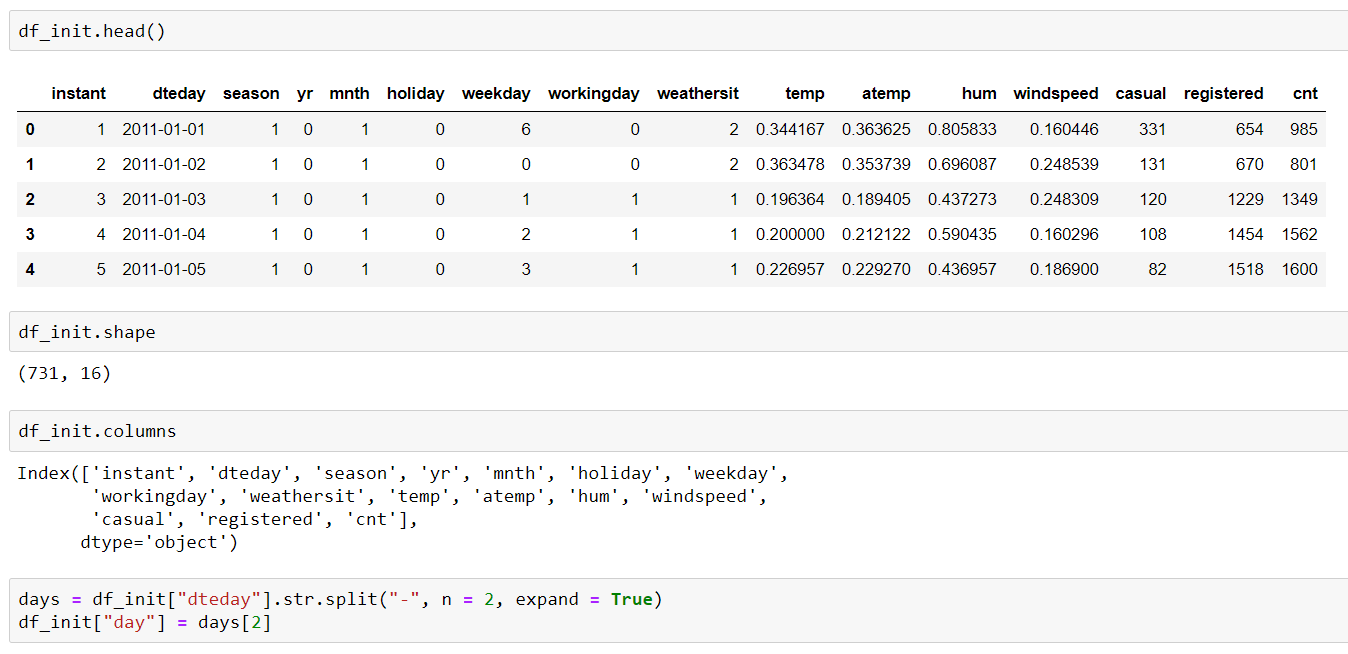
**KNN Model:**

MAPE = 28.85

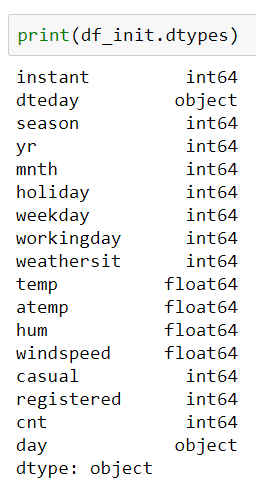
As per the above metrics, we can conclude that **Linear Regression** is the better model as it has relatively very less MAPE.

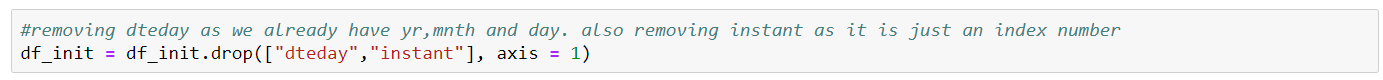
1. **APPENDIX A – PYTHON CODE**

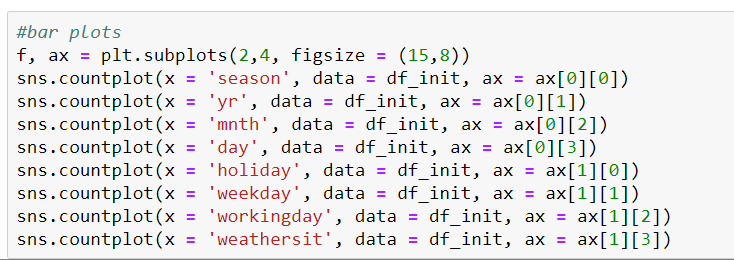


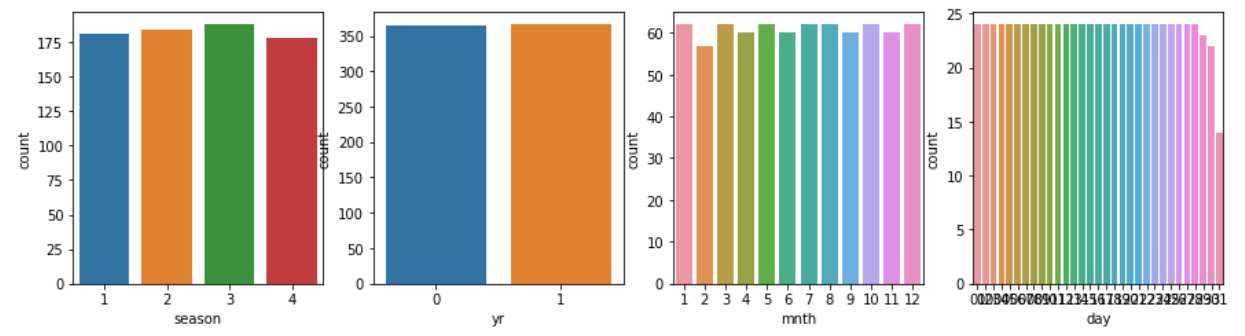


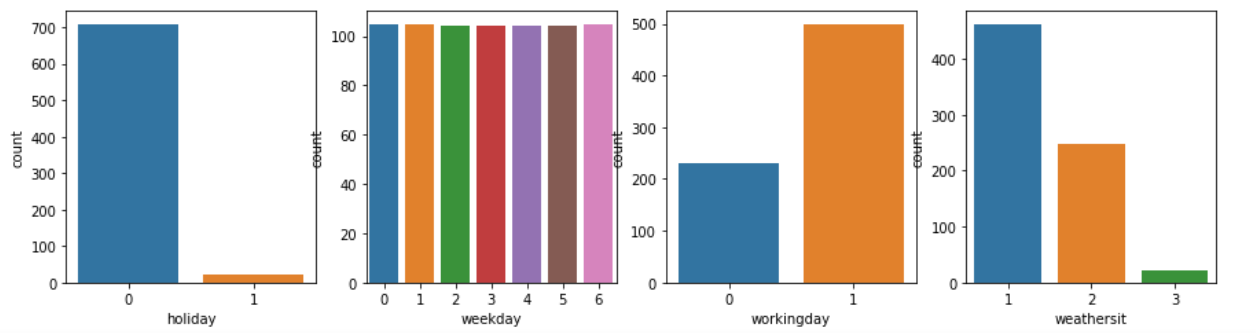


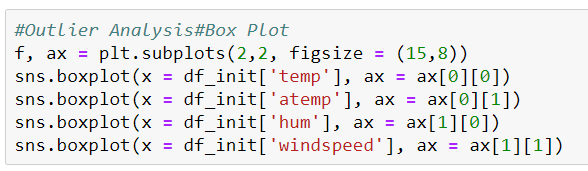


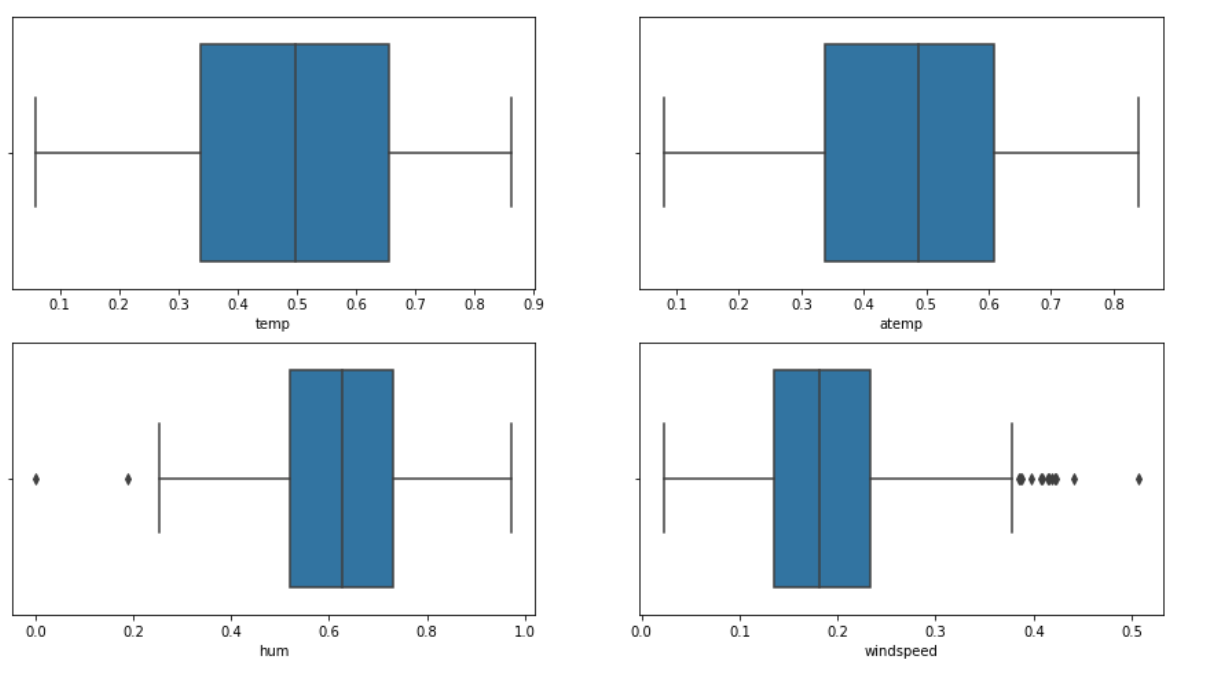


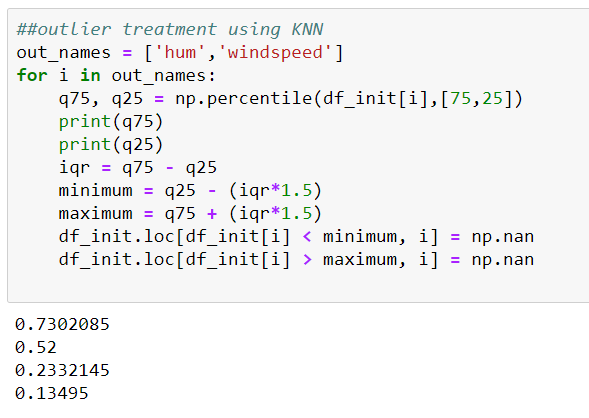


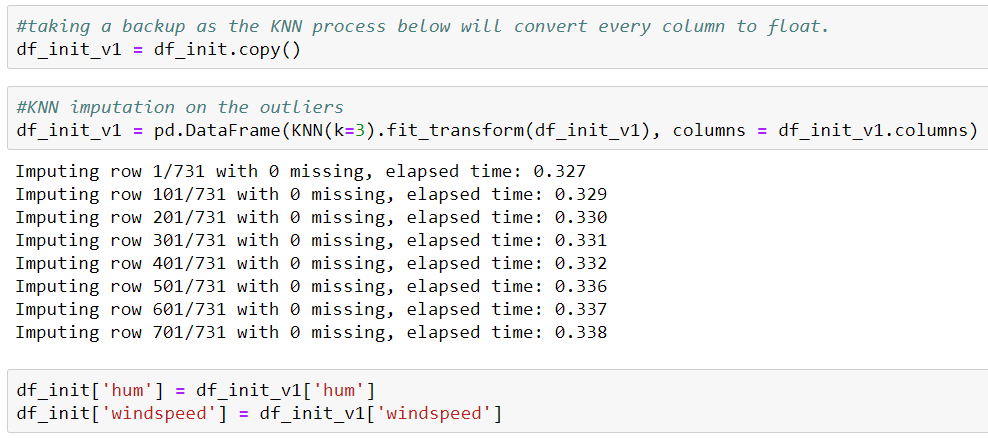


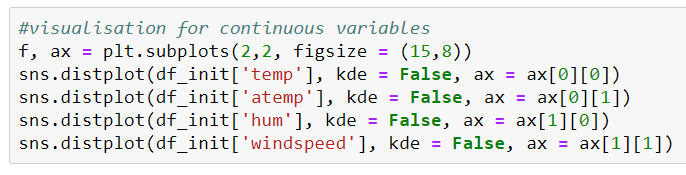


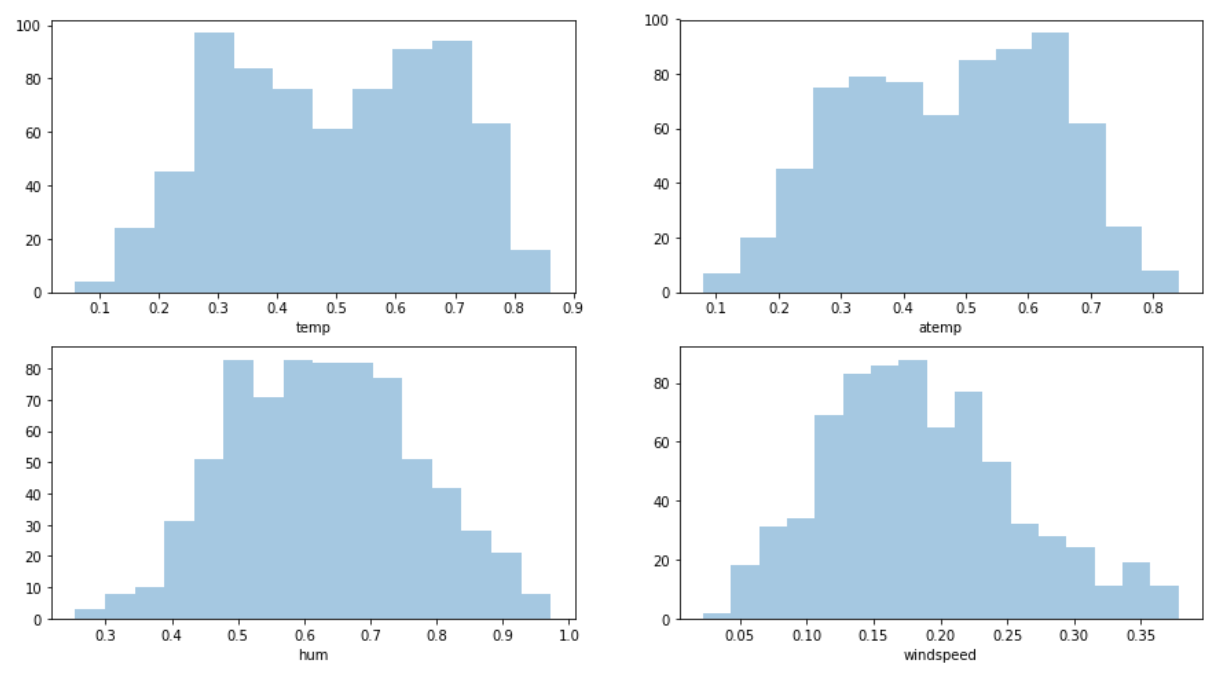


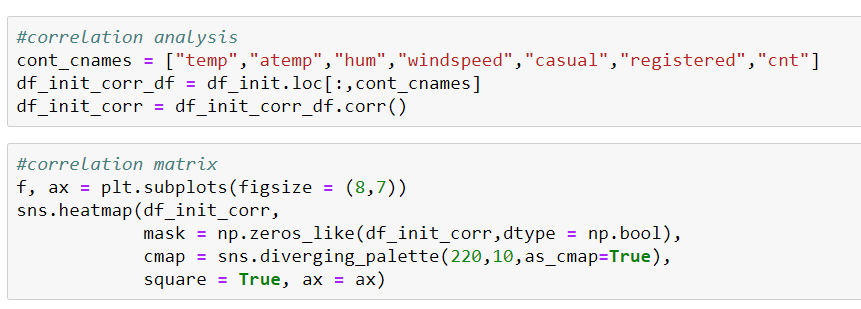


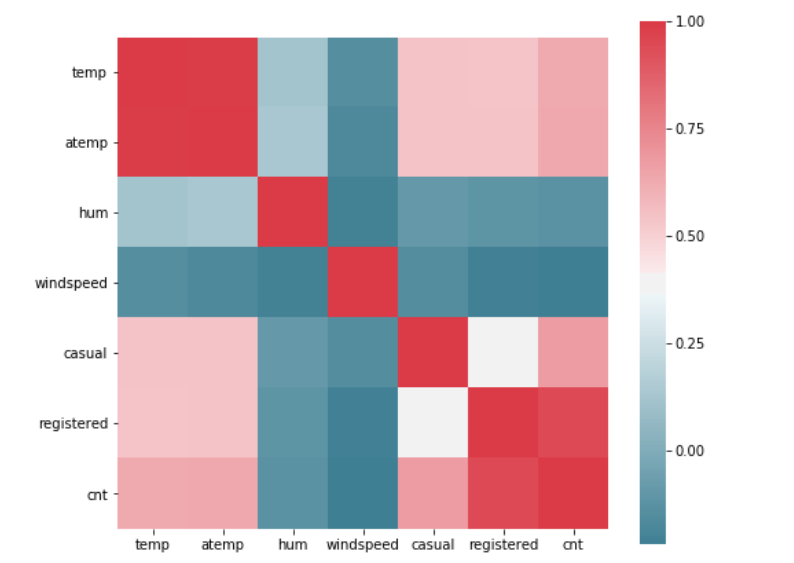






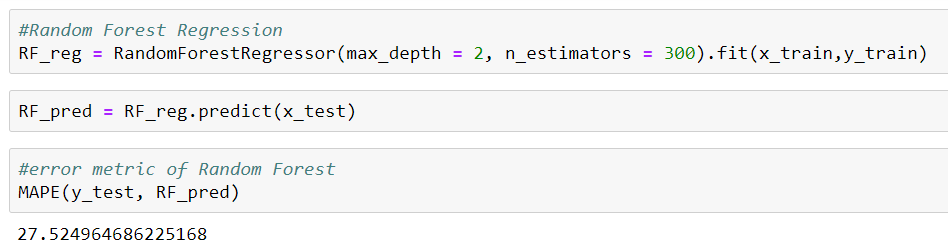


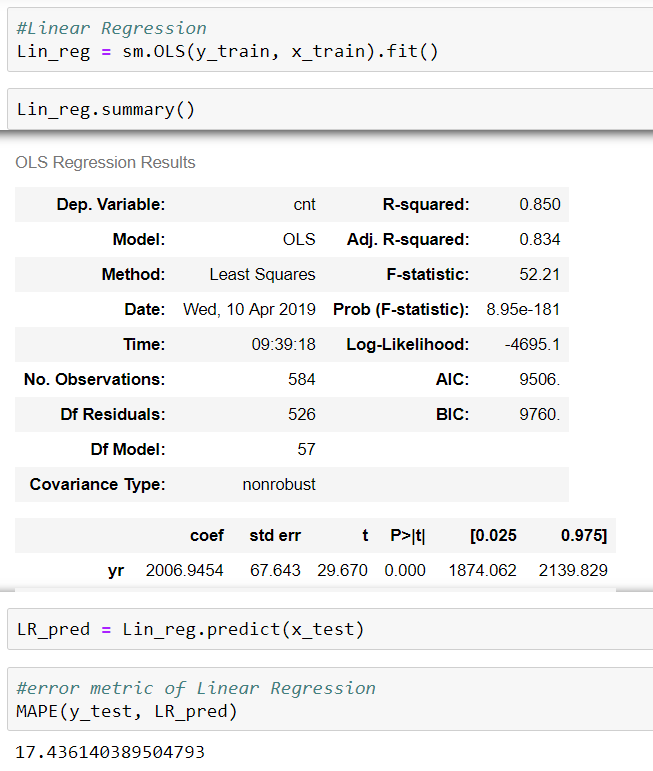


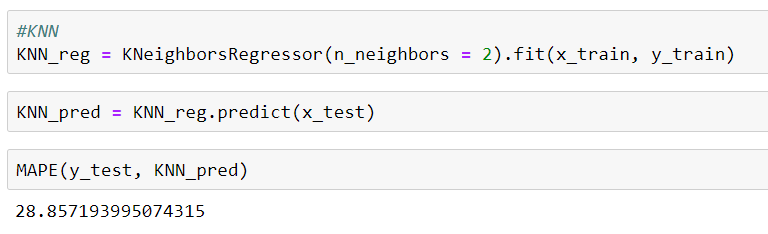












1. **APPENDIX B – R CODE**

rm(list=ls())

setwd('D:/Edwisor/Project 2 - Bike Renting')

getwd()

#importing libraries

library('corrgram')

#install.packages('dummyVars')

library('ggplot2')

library('DMwR')

library('caret')

library('rpart')

library('randomForest')

#loading the data

df\_init = read.csv("day.csv")

#checking summary

summary(df\_init)

#checking the datatype of each column

sapply(df\_init, function(x) class(x))

#creating day variable from dteday

day = read.table(text = as.character(df\_init$dteday), sep = "-", stringsAsFactors=FALSE)

df\_init$day = day$V3

#checking count of missing values in each column

sapply(df\_init, function(x) sum(is.na(x)))

#removing dteday as we already have yr,mnth and day. also removing instant as it is just an index number

df\_init$dteday = NULL

df\_init$instant = NULL

#barplots of categorical variables

barplot(table(df\_init$season))

barplot(table(df\_init$yr))

barplot(table(df\_init$mnth))

barplot(table(df\_init$holiday))

barplot(table(df\_init$weekday))

barplot(table(df\_init$workingday))

barplot(table(df\_init$weathersit))

barplot(table(df\_init$day))

#boxplot for outlier analysis

boxplot(df\_init$temp)

boxplot(df\_init$atemp)

boxplot(df\_init$hum)

boxplot(df\_init$windspeed)

#outlier treatment using KNN

out\_cnames = c('hum','windspeed')

for (i in out\_cnames){

val = df\_init[,i][df\_init[,i] %in% boxplot.stats(df\_init[,i])$out]

df\_init[,i][df\_init[,i] %in% val] = NA

}

df\_init = knnImputation(df\_init, k = 3)

#histogram for continuous variables

hist(df\_init$temp)

hist(df\_init$atemp)

hist(df\_init$hum)

hist(df\_init$windspeed)

#correlation plot

cont\_cnames = c('temp','atemp','hum','windspeed','registered','casual','cnt')

corrgram(df\_init[,cont\_cnames], order = F, upper.panel = panel.pie, text.panel = panel.txt, main = "Correlation Plot")

#dropping atemp, casual and registered due to correlation and data leakage

df\_init$atemp = NULL

df\_init$casual = NULL

df\_init$registered = NULL

df\_init = df\_init1

#dummy encoding

df\_init$season = as.factor(df\_init$season)

dmy = dummyVars("~ season", data = df\_init)

df\_append = data.frame(predict(dmy,newdata = df\_init))

df\_init = cbind(df\_init,df\_append)

df\_init$mnth = as.factor(df\_init$mnth)

dmy = dummyVars("~ mnth", data = df\_init)

df\_append = data.frame(predict(dmy,newdata = df\_init))

df\_init = cbind(df\_init,df\_append)

df\_init$weekday = as.factor(df\_init$weekday)

dmy = dummyVars("~ weekday", data = df\_init)

df\_append = data.frame(predict(dmy,newdata = df\_init))

df\_init = cbind(df\_init,df\_append)

df\_init$weathersit = as.factor(df\_init$weathersit)

dmy = dummyVars("~ weathersit", data = df\_init)

df\_append = data.frame(predict(dmy,newdata = df\_init))

df\_init = cbind(df\_init,df\_append)

df\_init$day = as.factor(df\_init$day)

dmy = dummyVars("~ day", data = df\_init)

df\_append = data.frame(predict(dmy,newdata = df\_init))

df\_init = cbind(df\_init,df\_append)

df\_init$season = NULL

df\_init$mnth = NULL

df\_init$weathersit = NULL

df\_init$day = NULL

df\_init$weekday = NULL

df\_init1 = df\_init

#sampling

train.index = createDataPartition(df\_init$cnt, p=0.8, list = FALSE)

train\_df = df\_init[train.index,]

test\_df = df\_init[-train.index,]

y\_train = train\_df$cnt

y\_test = test\_df$cnt

train\_df$cnt = NULL

test\_df$cnt = NULL

#defining error metric MAPE

MAPE = function(y,yhat){

mean(abs((y-yhat)/y))\*100

}

#Decision Tree Regression(MAPE = 22.40)

DT\_reg = rpart(y\_train ~ ., data = train\_df, method = 'anova')

DT\_pred = predict(DT\_reg, test\_df)

MAPE(y\_test, DT\_pred)

#Random Forest Regression(MAPE = 14.086)

RF\_reg = randomForest(y\_train ~ ., data = train\_df, ntree = 300)

RF\_pred = predict(RF\_reg, test\_df)

MAPE(y\_test, RF\_pred)

#Linear Regression(MAPE = 17.58)

Lin\_reg = lm(y\_train ~ ., data = train\_df)

summary(Lin\_reg)

LR\_pred = predict(Lin\_reg, test\_df)

MAPE(y\_test, LR\_pred)

#KNN Regression(MAPE = 30.75)

KNN\_reg = knnreg(train\_df, y\_train, k = 2)

KNN\_pred = predict(KNN\_reg, test\_df)

MAPE(y\_test, KNN\_pred)