Bike Renting Predictions

Juhi Sahu

2 October 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

The objective is to forecast bike rental demand of Bike sharing program in Washington, D.C based on historical usage patterns in relation with weather, environment and other data. We would be interested in predicting the rentals on various factors including season, temperature, weather and building a model that can successfully predict the number of rentals on relevant factors.

1.2 Data

This dataset contains the seasonal and weekly count of rental bikes between years 2011 and 2012 in Capital bikeshare system with the corresponding temperature and humidity information. Bike sharing systems are a new way of traditional bike rentals. The whole process from membership to rental and return back has become automatic. The data was generated by 500 bike-sharing programs and was collected by the Laboratory of Artificial Intelligence and Decision Support (LIAAD), University of Porto. Given below is the description of the data which is a (731, 16) shaped data, The variables are:

S.No	Feature Name	Description
1	Instant	Daily customer index
2	Dteday	Date index for both the years
3	Season	Season type: 1-Spring, 2-Summer, 3-Fall, 4-Winter
4	Yr	The year 0-2011, 1-2012
5	Mnth	Month 1-12
6	Holiday	Whether day is holiday or not
7	Weekday	Day of the week 0-Monday, 6-Sunday
8	Workingday	Whether day is working day or not
9	Weathersit	Weather 1-Clear, 2-Misty, 3-Light rain, 4-Heavy rain
10	Temp	Normalized value of temperature at every instant
11	Atemp	Normalized value of absolute temperature
12	Humidity	Contains the normalized value for the humidity
13	Windspeed	Contains the normalized value for the windspeed
14	Casual	The number of unregistered users at a given day
15	Registered	The number of registered users at a given day
16	Count (Target)	Total Rentals with both casual and registered users

Chapter 2

Methodology

2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**. For our data we apply preprocessing techniques that we necessary.
Our preprocessing involves following steps:

1. Missing value analysis
2. Visualizations
3. Outlier Analysis
4. Feature selection
   4.1 Correlation Analysis
   4.2 ANOVA
   4.3 VIF Test
5. Normalization

2.1.1 Missing value analysis

Missing values occur when no data value is stored for the variable in an observation. Missing values are a common occurrence, and you need to have a strategy for treating them. A missing value can signify a number of different things in your data. Perhaps the data was not available or not applicable or the event did not happen. It could be that the person who entered the data did not know the right value, or missed filling in. Typically, ignore the missing values, or exclude any records containing missing values, or replace missing values with the mean, or infer missing values from existing values. We check for missing values in our data and came to know that there are no missing values.

2.1.2 Visualizations

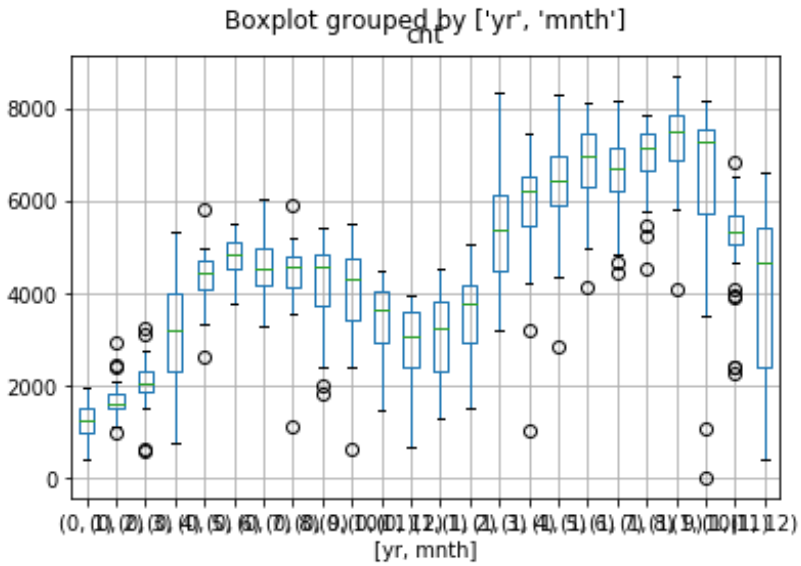


Fig 2.1 – Boxplot of count of bike rents grouped by year and month

From fig 2.1, we can infer that count of rental bikes have increased over the year 2012. In each particular year, month 5(May), 6(June), 7(July) have highest rentals.

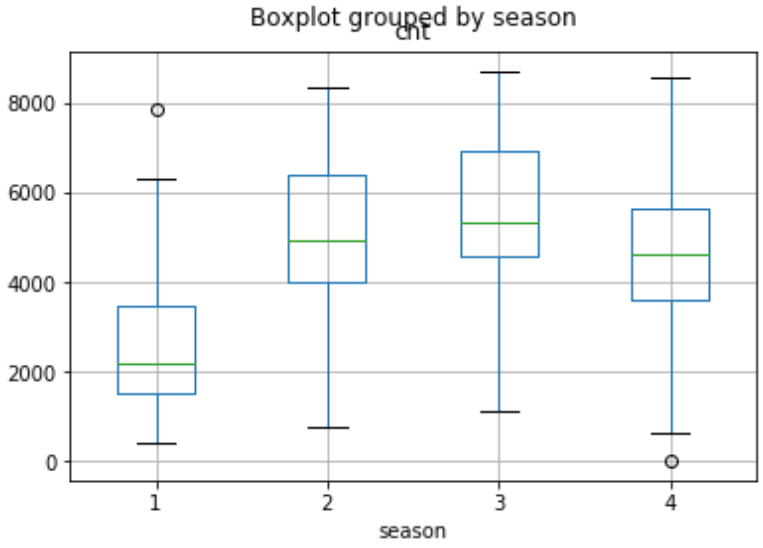


Fig 2.2 – Boxplot grouped by Season

From fig 2.2, we can infer that rentals are higher in Season3 (fall) and least in Season1 (Spring)

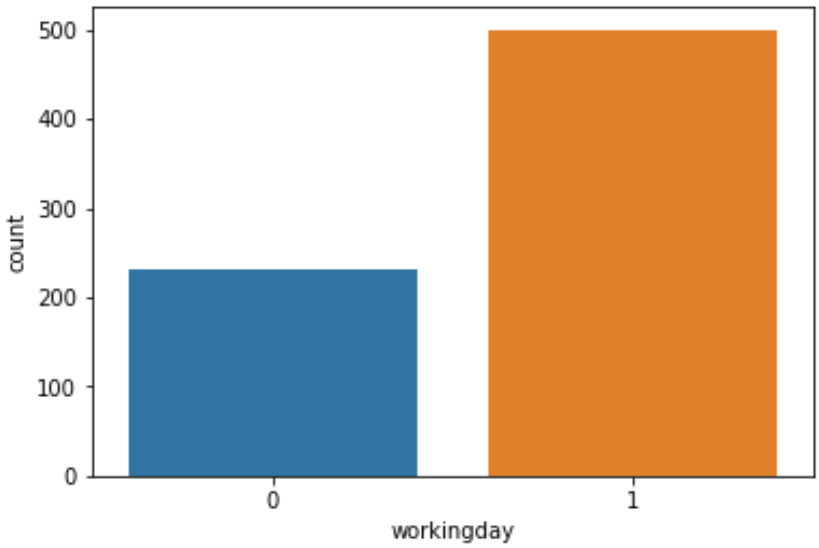


Fig 2.3 – Bar graph of working day

From fig 2.3 **,** we can infer that people rent bikes more on working days as there may be chances of rental requirements in the office commute rather than on non-working days.

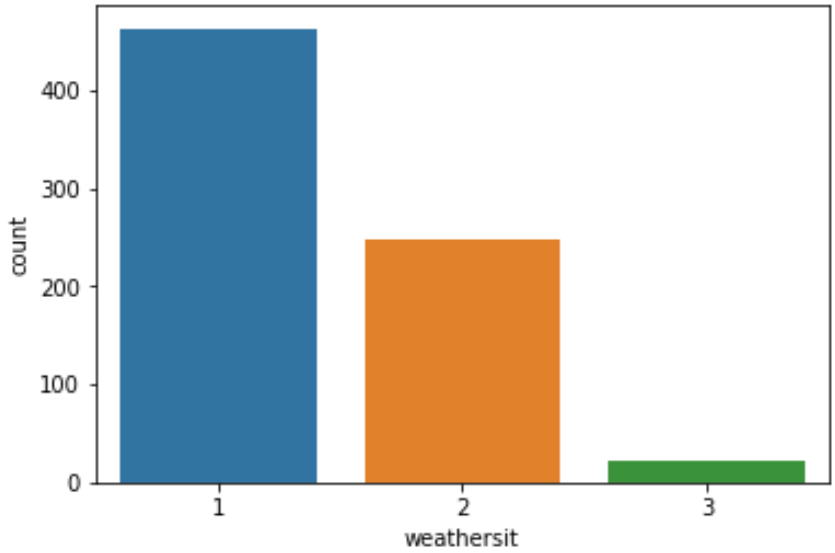


Fig 2.4 – Bar graph of weather sit

From fig 2.4, we can infer that on weatersit1 (clear weather), people prefer more to rent bikes rather on weathersit3(heavy rain).

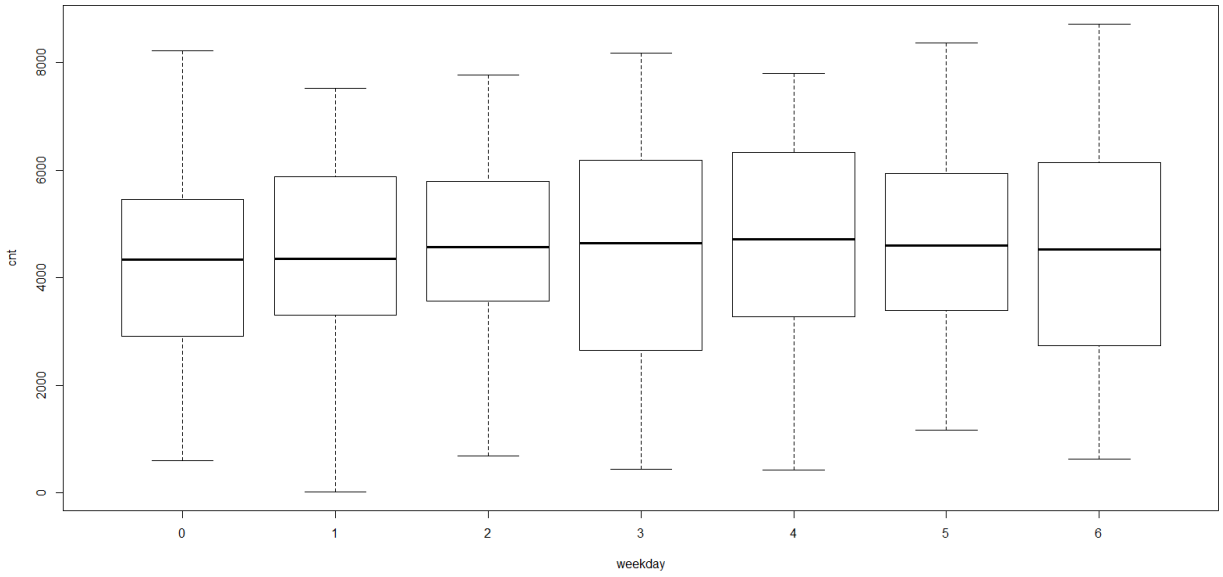


Fig 2.5 – Boxplot of weekdays

From fig 2.5 we can infer, that all the days have almost same rental counts.

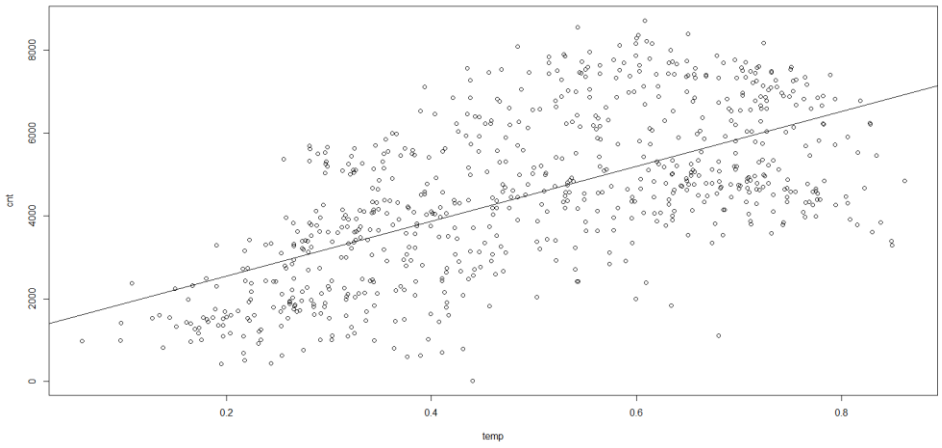


Fig 2.6 – Scatter plot of Temp vs Count

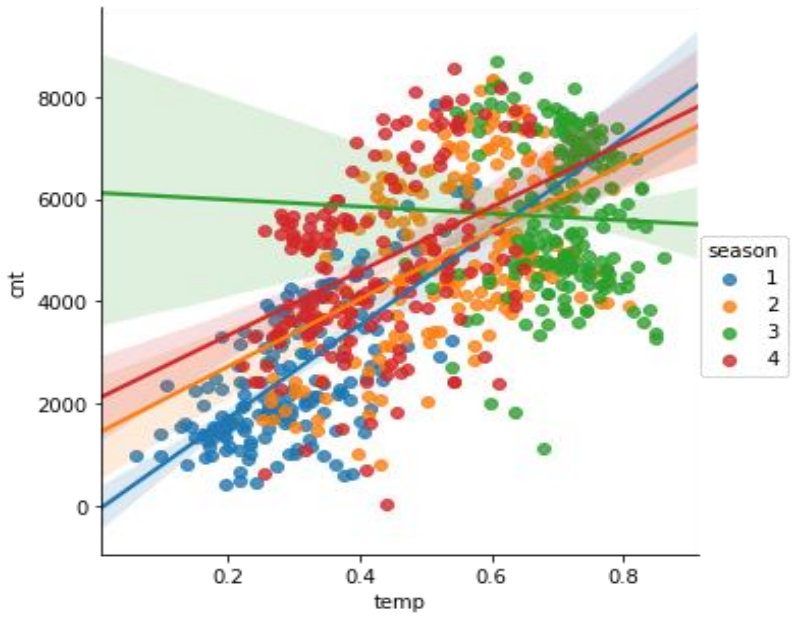


Fig 2.7 – Linear plot of temp Vs count (season wise)

Both the figures (2.6 & 2.7) shows that high temperatures are the main reason of people to rent out bikes more. Also plot 2.7 shows the variation of counts over temperature in each season.

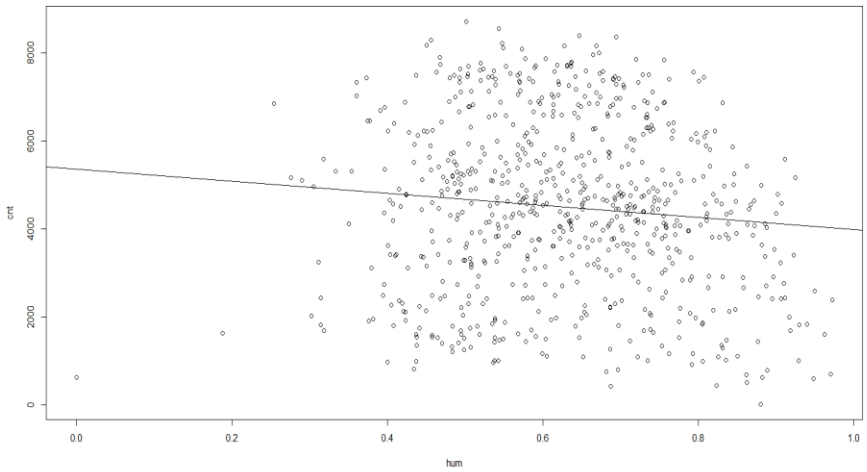


Fig 2.8 - Scatter plot of humidity vs count

From fig 2.8, it is clear that humidity is slightly inversely proportion with count, i.e. with increase in humidity, the rental counts decrease.

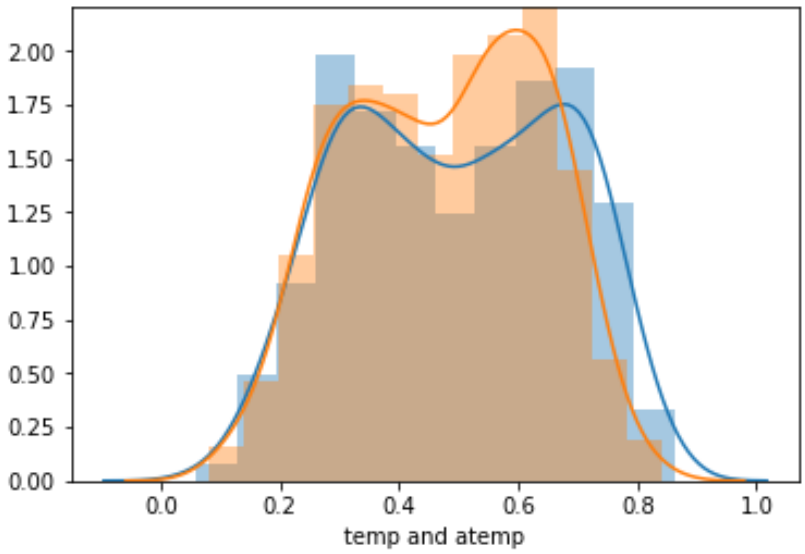


Fig 2.9 – Distribution of temp & atemp

From the above figure we can infer that temp and atemp have almost similar distribution. We will confirm this in our next Feature selection segment from correlation test

2.1.3 Outlier Analysis

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. Outliers can drastically change the results of the data analysis and statistical modeling. There are numerous unfavorable impacts of outliers in the data set. It increases the error variance and reduces the power of statistical tests. If the outliers are non-randomly distributed, they can decrease normality. They can also impact the basic assumption of Regression, ANOVA and other statistical model assumptions. The boxplot for our data could be seen as follows:

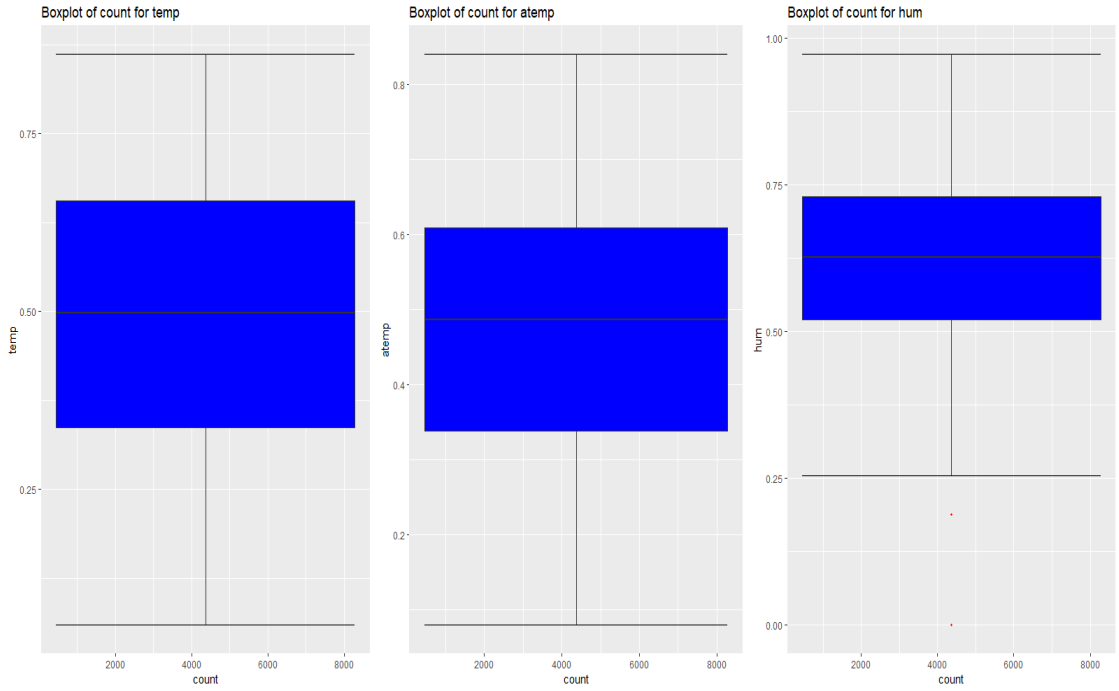


Fig 2.10

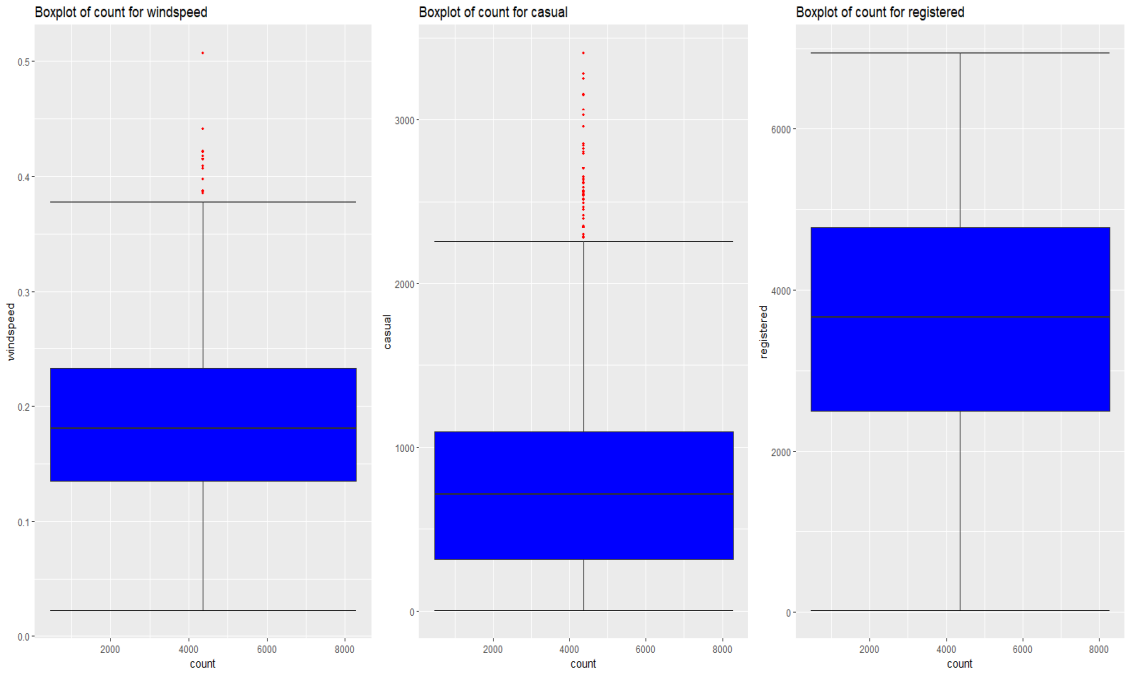


FIG 2.11

The boxplot helps us to identify outliers in each column. In our data, outliers are found in humidity, windspeed and casual columns.

- Outliers in humidity were imputed with mean of that column. (only 2 outliers)
- Outliers in windspeed were imputed with season wise mean of windspeed.

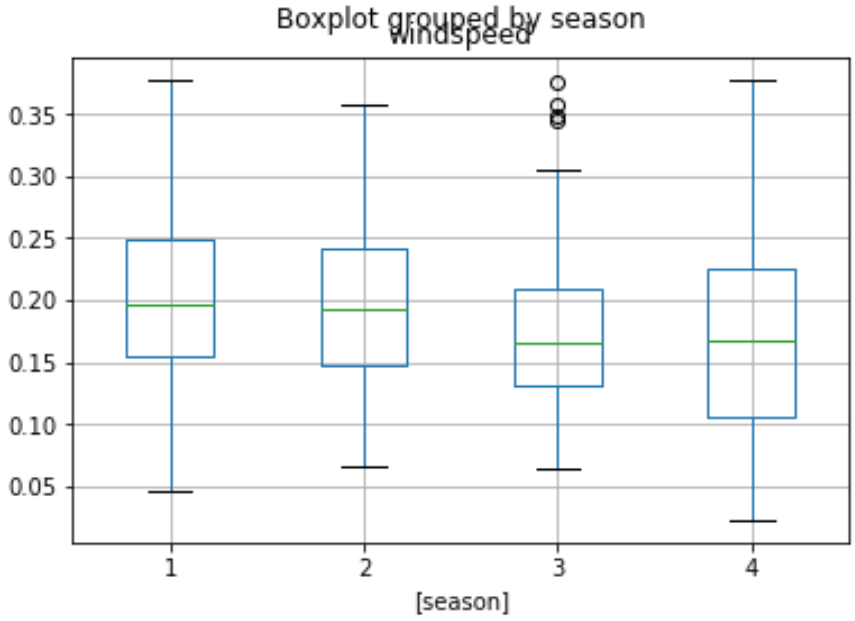


Fig 2.12 – Boxplot of windspeed grouped by season

- Outliers in casual column were calculated as :

  Casual = count – registered

2.1.4 Feature Selection

Variable selection is an important aspect of model building. It helps in building predictive models free from correlated variables, biases and unwanted noise. It helps in selecting a subset of relevant features (variables, predictors) for use in model construction and subset of a learning algorithm's input variables upon which it should focus attention, while ignoring the rest.

Correlation Analysis

A heatmap is a graphical representation of data where the individual values contained in a matrix are represented as colors. Here each numerical variable's correlation is mapped with each other's in a matrix which has been plotted in the following heatmap.

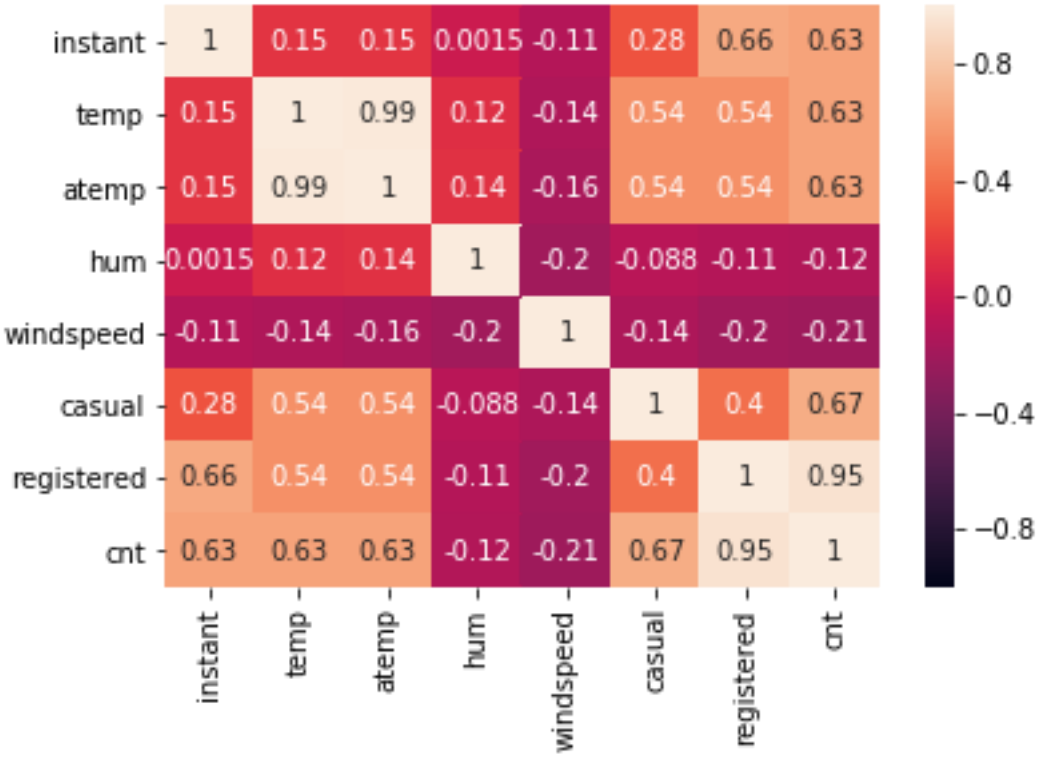


FIG 2.13

Our correlation matrix shows some interesting results as follows

1. Temp and atemp are highly correlated
2. Registered and count are highly correlated (since count is a sum of registered and casual)

We can now remove one of the highly correlated variable so that our model can perform well with much accuracy.

ANOVA

Analysis of variance (ANOVA) is a statistical technique that is used to check if the means of two or more groups are significantly different from each other. ANOVA checks the impact of one or more factors by comparing the means of different samples. As our target variable is numerical we will use ANOVA for feature selection technique to see whether any categorical variable is related to target variable. The result of anova is as follows:

1. <u>For Seasons:</u>

	df	sum_sq	mean_sq	F	PR(>F)
season	3.0	9.505959e+08	3.168653e+08	128.769622	6.720391e-67

Residual	727.0	1.788940e+09	2.460715e+06	NaN	NaN

2. For Year:

	df	sum_sq	mean_sq	F	PR(>F)
yr	1.0	8.798289e+08	8.798289e+08	344.890586	2.483540e-63
Residual	729.0	1.859706e+09	2.551038e+06	NaN	NaN

3. For Month

	df	sum_sq	mean_sq	F	PR(>F)
mnth	11.0	1.070192e+09	9.729021e+07	41.903703	4.251077e-70
Residual	719.0	1.669343e+09	2.321757e+06	NaN	NaN

4. For Holiday

	df	sum_sq	mean_sq	F	PR(>F)
holiday	1.0	1.279749e+07	1.279749e+07	3.421441	0.064759
Residual	729.0	2.726738e+09	3.740381e+06	NaN	NaN

5. For Weekday

	df	sum_sq	mean_sq	F	PR(>F)
weekday	6.0	1.765902e+07	2.943170e+06	0.782862	0.583494
Residual	724.0	2.721876e+09	3.759498e+06	NaN	NaN

6. For Workingday

	df	sum_sq	mean_sq	F	PR(>F)
workingday	1.0	1.024604e+07	1.024604e+07	2.736742	0.098495
Residual	729.0	2.729289e+09	3.743881e+06	NaN	NaN

7. For Weathersit

	df	sum_sq	mean_sq	F	PR(>F)
weathersit	2.0	2.716446e+08	1.358223e+08	40.066045	3.106317e-1
Residual	728.0	2.467891e+09	3.389960e+06	NaN	NaN

H_o = Categorical variable is Independent from the Target variable
H_a = Categorical variable is Dependent on the Target variable

If the p value of the categorical variable is less than 0.05 then we will consider that the target variable is dependent on the categorical variable for which we reject the null hypothesis.

From the above result we can see that only five variables are very much related to target variable hence we delete all the other variables.

Therefore from both the correlation analysis and ANOVA we got some variable which we shouldn't consider for further processing. The variables that could be deleted are as follows

Numerical: Instant, atemp, casual, registered
Categorical: dteday, holiday, workingday

VIF TEST

The variance inflation factor (VIF) is the ratio of variance in a model with multiple terms, divided by the variance of a model with one term alone. It quantifies the severity of multicollinearity in an ordinary least squares regression analysis. It provides an index that measures how much the variance (the square of the estimate's standard deviation) of an estimated regression coefficient is increased because of collinearity. VIF for our data can be seen as follows.

	VIF Factor	features
0	61.575840	Intercept
1	7.485484	season[T.2]
2	10.700284	season[T.3]
3	7.395067	season[T.4]
4	1.045391	yr[T.1]
5	1.832116	mnth[T.2]
6	2.608054	mnth[T.3]
7	5.677686	mnth[T.4]
8	6.822593	mnth[T.5]
9	7.309326	mnth[T.6]
10	9.316998	mnth[T.7]
11	8.634651	mnth[T.8]
12	6.473108	mnth[T.9]
13	5.572091	mnth[T.10]
14	4.924150	mnth[T.11]
15	3.172803	mnth[T.12]
16	1.619663	weathersit[T.2]
17	1.312625	weathersit[T.3]
18	6.883740	temp
19	2.101808	hum

Fig 2.14

2.1.4 Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing steps. Normalization also called Min-Max scaling. It is the process of reducing unwanted variation either within or between variables. Normalization brings all of the variables into proportion with one another. It transforms data into a range between 0 and 1.
All our continuous variables are already normalized except the target variable which we prefer not to scale because its variation are spread quite widely and after scaling, the difference between the number is diminishing.

2.2 Modeling

2.2.1 Cross validation

Cross Validation is a technique which involves reserving a particular sample of a dataset on which you do not train the model. Later, you test your model on this sample before finalizing it.

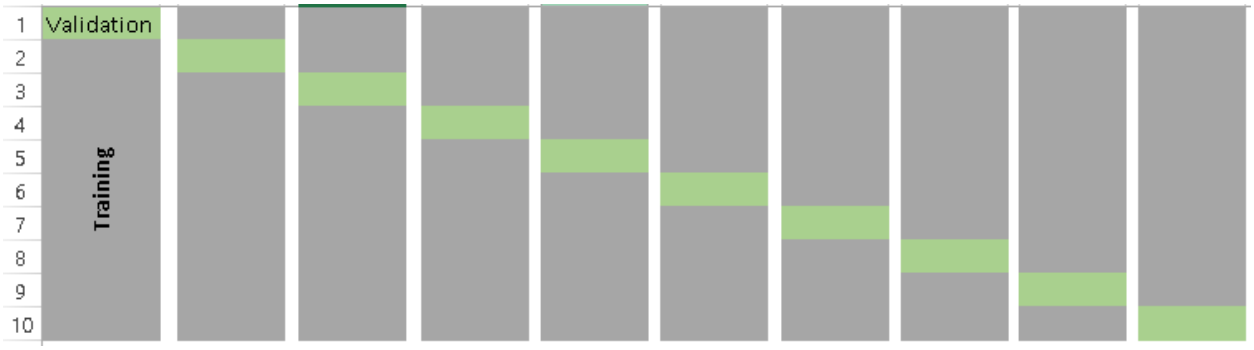
Here are the steps involved in cross validation:

1. You reserve a sample data set
2. Train the model using the remaining part of the dataset
3. Use the reserve sample of the test (validation) set. This will help you in gauging the effectiveness of your model's performance. If your model delivers a positive result on validation data, go ahead with the current model

We need to remember that:

1. We should train the model on a large portion of the dataset. Otherwise we'll fail to read and recognize the underlying trend in the data. This will eventually result in a higher bias
2. We also need a good ratio of testing data points. As we have seen above, less amount of data points can lead to a variance error while testing the effectiveness of the model
3. We should iterate on the training and testing process multiple times. We should change the train and test dataset distribution. This helps in validating the model effectiveness properly

"**K-fold cross validation**" method considers all these points. It's easy to follow and implement.



2.2.2 Model Selection

After a thorough preprocessing we will be using some regression models on our processed data to predict the target variable.

Decision Tree: Decision tree is a rule. Each branch connects nodes with “and” and multiple branches are connected by “or”. It can be used for classification and regression. It is a supervised machine learning algorithm. Accept continuous and categorical variables as independent variables. Extremely easy to understand by the business users. Split of decision tree is seen in the below tree. Decision tree regression is as follows

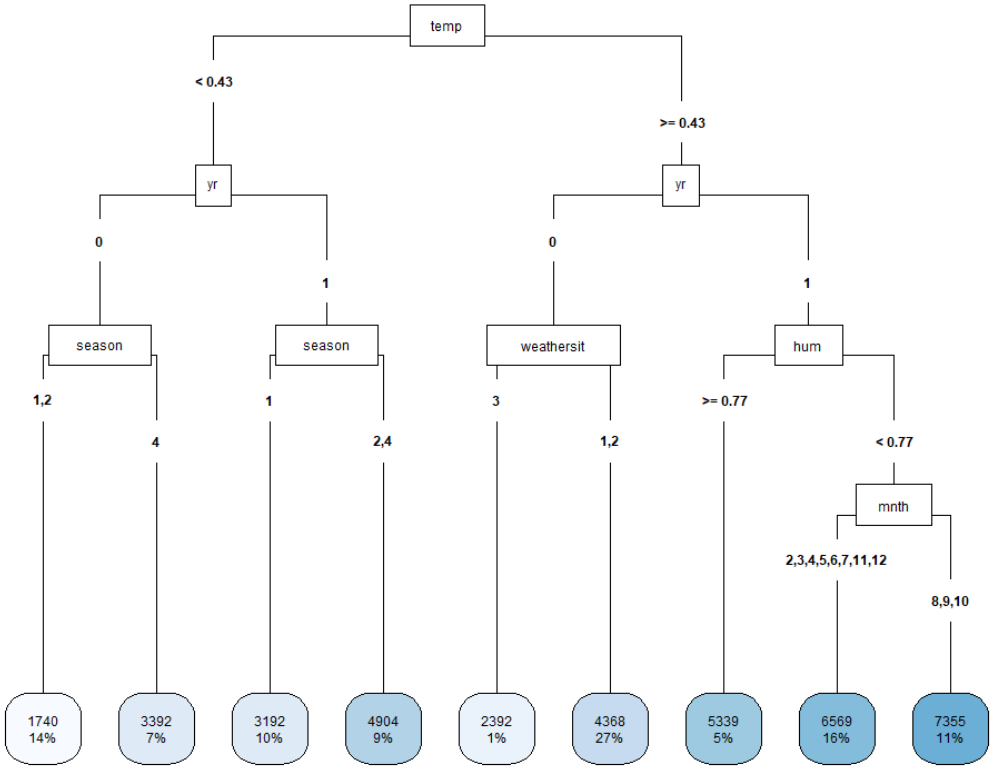


FIG 2.15

Random Forest: Random Forest or decision tree forests are an ensemble learning method for classification, regression and other tasks. It consists of an arbitrary number of simple trees, which are used to determine the final outcome. In the regression problem, their responses are averaged to obtain an estimate of the dependent variable. Using tree ensembles can lead to significant improvement in prediction accuracy (i.e., better ability to predict new data cases). The goal of using a large number of trees is to train enough that each feature has a chance to appear in several models. We can see certain rules of random forest in the R code.

```
Call:
 randomForest(formula = cnt ~ ., data = data_train, importance = TRUE, ntree
= 500)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 471352
                    % Var explained: 87.54
```

Error vs number of trees to be used graph is as follows:

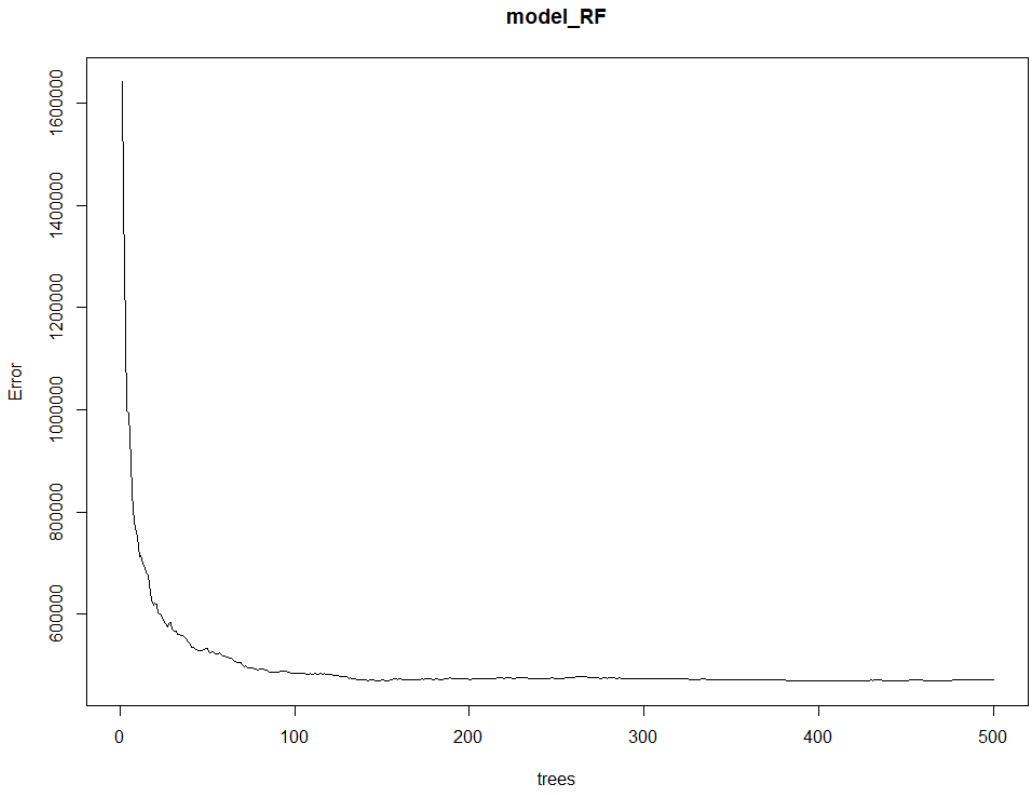


FIG 2.16

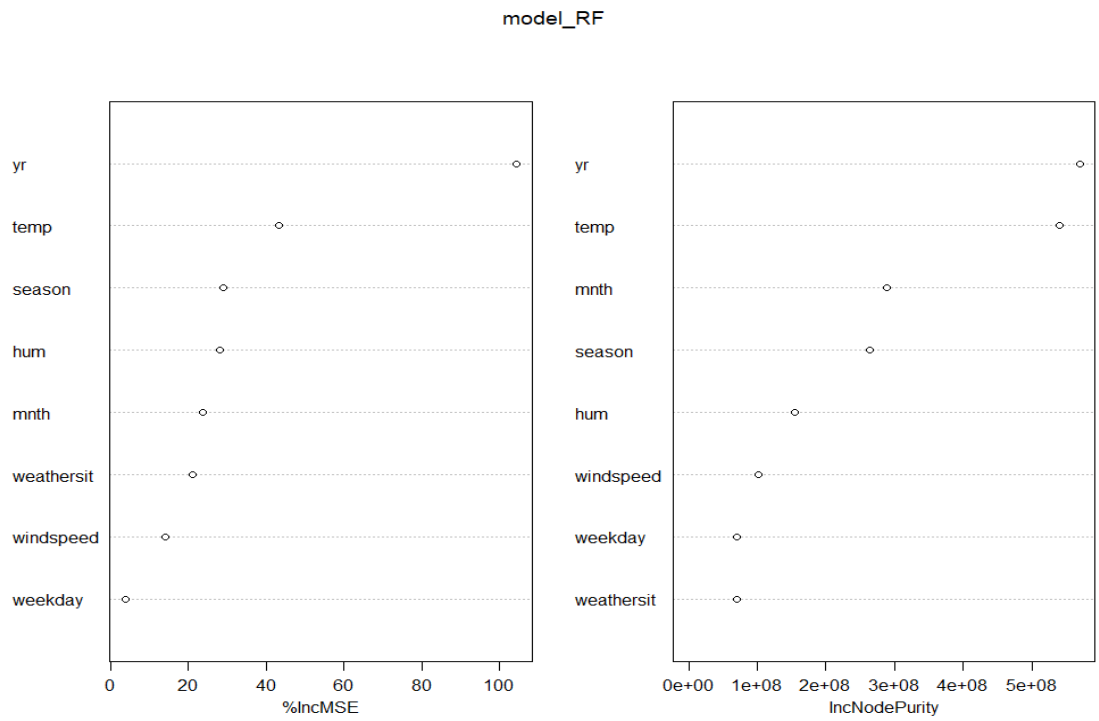


Fig 2.17 – Variable importance

The first graph shows that if a variable is assigned values by random permutation by how much will the MSE increase. Higher the value, higher the importance.
On the other hand, node purity is measured by the Gini index which is the difference between before and after split on that variable.

Linear Regression: Linear regression is the most basic type of regression and commonly used predictive analysis. Linear regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.

We have to convert all the categorical variable into Dummies because Machine learning algorit hms require numbers as input. In regression analysis, a dummy variable (also known as an indic ator variable, design variable, Boolean indicator, binary variable, or qualitative variable) is one t hat takes the value 0 or 1 to indicate the absence or presence of some categorical effect that m ay be expected to shift the outcome

Following is the summary of the Linear model:

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

Residuals:
    Min      1Q  Median      3Q     Max
-3856.8  -336.4    88.4   469.3  2830.5

Coefficients: (5 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   3605.04     437.49   8.240 1.23e-15 ***
season_1     -1696.23     200.94  -8.441 2.72e-16 ***
season_2      -808.88     245.83  -3.290 0.001064 **
season_3      -766.10     215.71  -3.551 0.000416 ***
season_4           NA         NA      NA       NA
yr_0         -1977.81      69.05 -28.641  < 2e-16 ***
yr_1               NA         NA      NA       NA
mnth_1         133.03     204.87   0.649 0.516394
mnth_2         234.16     203.12   1.153 0.249471
mnth_3         709.74     208.15   3.410 0.000697 ***
mnth_4         510.65     281.82   1.812 0.070525 .
mnth_5         821.93     300.34   2.737 0.006404 **
mnth_6         416.90     307.49   1.356 0.175709
mnth_7        -109.04     326.66  -0.334 0.738663
mnth_8         402.15     308.23   1.305 0.192532
mnth_9         936.02     249.21   3.756 0.000191 ***
mnth_10        490.16     187.34   2.616 0.009128 **
mnth_11       -131.98     178.50  -0.739 0.459984
mnth_12            NA         NA      NA       NA
weekday_0     -462.98     124.34  -3.724 0.000216 ***
weekday_1     -335.56     123.97  -2.707 0.007001 **
weekday_2     -177.45     125.93  -1.409 0.159368
weekday_3      -93.40     126.15  -0.740 0.459373
weekday_4     -128.51     124.77  -1.030 0.303480
weekday_5      -38.63     127.89  -0.302 0.762701
weekday_6          NA         NA      NA       NA
weathersit_1  1930.98     237.87   8.118 3.06e-15 ***
weathersit_2  1544.39     219.86   7.024 6.29e-12 ***
weathersit_3       NA         NA      NA       NA
temp          4759.17     496.96   9.576  < 2e-16 ***
hum          -1691.12     366.28  -4.617 4.84e-06 ***
windspeed    -2946.50     524.66  -5.616 3.09e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 811.3 on 557 degrees of freedom
Multiple R-squared:  0.831,    Adjusted R-squared:  0.8231
F-statistic: 105.3 on 26 and 557 DF,  p-value: < 2.2e-16
```

Chapter 3

Conclusion

3.1 Model Evaluation

Model evaluation is done on basis of evaluation metrics or error metrics. Evaluation metrics explain the performance of a model. An important aspect of evaluation metrics is their capability to discriminate among model results. Simply, building a predictive model is not our motive. But, creating and selecting a model which gives high accuracy on out of sample data. Hence, it is crucial to check accuracy or other metric of the model prior to computing predicted values. In our data as we applied regression models we have error metrics like Mean square error(MSE), MAPE, Root mean square error (RMSE), Mean absolute error (MAE).

	PYTHON				R			
MODELS	**MSE**	**RMSE**	**MAPE**	**R-SQ**	**MAE**	**RMSE**	**MAPE**	**R-SQ**
DECISION TREE	923641	961	20	0.73	989	1247	34	0.62
RANDOM FOREST	623811	789	19	0.82	500	721	14	0.88
LINEAR REGRESSION	863884	929	21	0.75	514	697	18	0.85

Table- 3.1

3.2 Model Selection

We can see that all models perform comparatively on average and therefore we select random forest classifier models for better prediction.

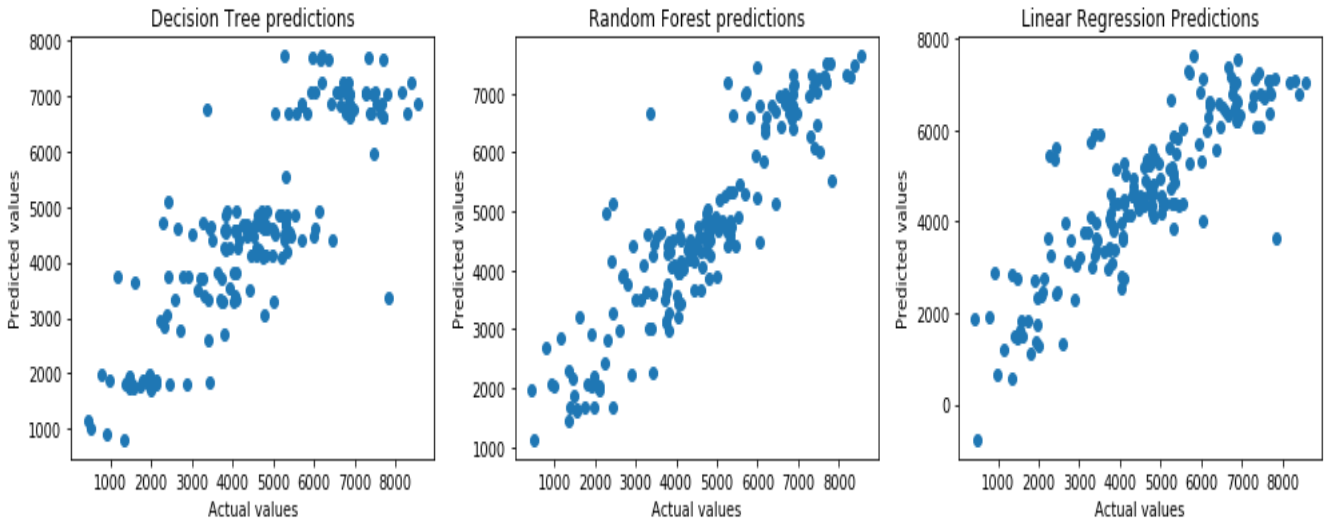


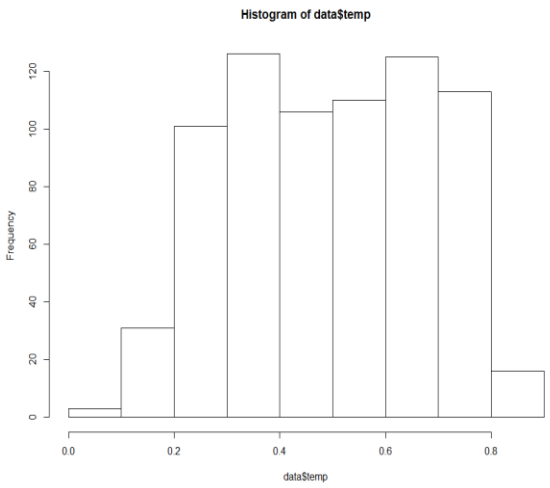
Fig 3.1

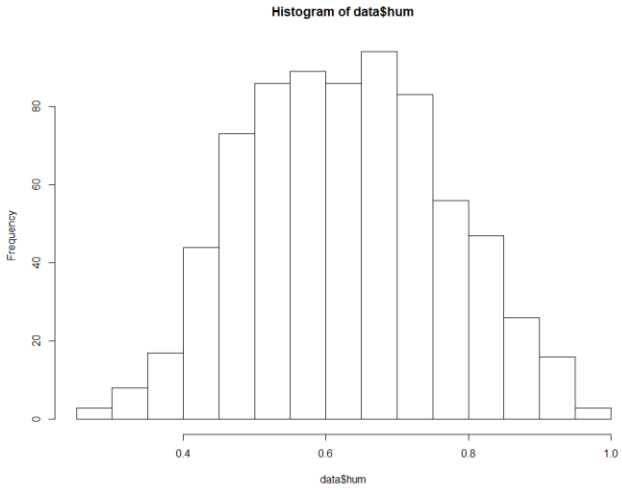
From the above plots of Actual Vs Predicted values, we can infer that values of Random forest falls on straight line indicating random forest fits better than the other three models.

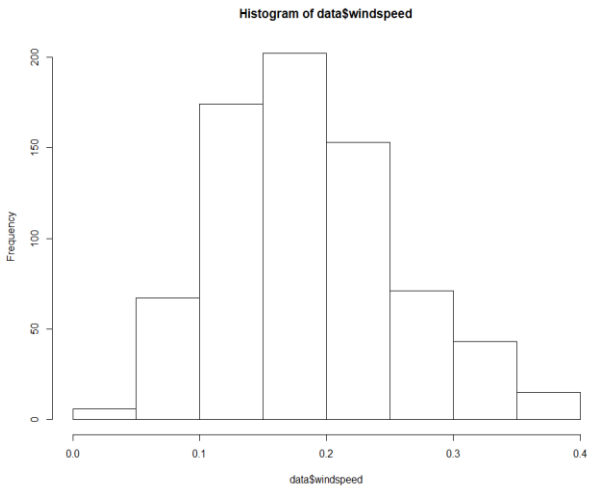
Also amongst the three models, Random forest has best R-sq. (Coef. of determination). Hence we'll fix Random Forest as our model.

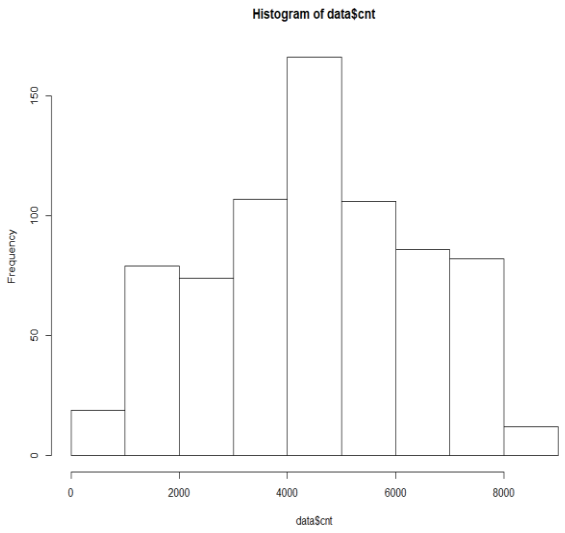
Appendix A – Extra Figures

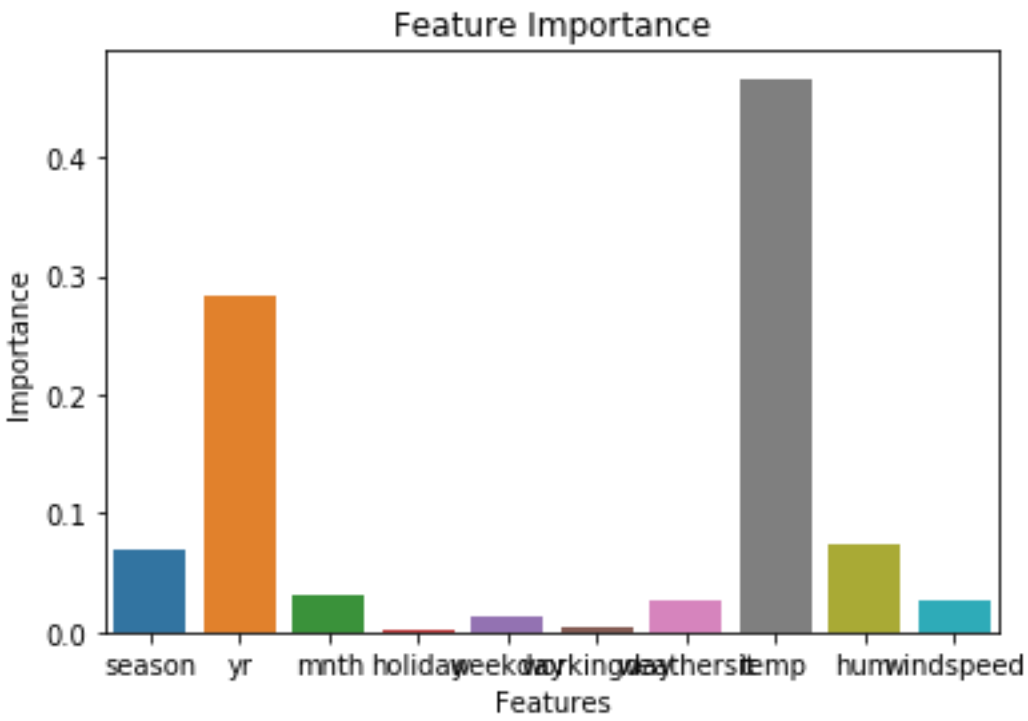
Normality check plots of various numerical variables:










Feature Importance
Importance
0.0
0.1
0.2
0.3
0.4
season
yr
mnth
holiday
Features

Appendix B - Code

R code

```
rm(list = ls())
setwd("E:/Study/edWisor Stuff/Project2")
getwd()
data = read.csv("day.csv", header = TRUE)
str(data)
sapply(data, function(x) sum(is.na(x))) #no missing values
data$season= factor(data$season)
data$yr = factor(data$yr)
data$mnth = factor(data$mnth)
data$holiday = factor(data$holiday)
data$weekday = factor(data$weekday)
data$workingday = factor(data$workingday)
data$weathersit = factor(data$weathersit)

#----------------------------------------------VISUALIZATION----------------------------------------
plot(cnt ~ season , data = data) #least rentals are in season 1 and maximum are in season 3
plot(cnt ~ weathersit , data = data) #weather 3 have least of rentals
plot(cnt ~ weekday , data = data)
install.packages("ggplot2")
install.packages("scales")
library(ggplot2)
library(scales)
#library(psych)
ggplot(data , aes_string(x=data$workingday)) + geom_bar(stat = "count", fill = "DarkslateBlue")
+
xlab("Working day") + ylab("Count") + ggtitle("Working day distribution") + theme(text =
element_text(size = 15))
#bikes are rented more on working days
reg1 = lm(cnt ~ temp, data = data)
summary(reg1)
with(data ,plot(temp, cnt))
abline(reg1) #rental counts increase with increase in temperature
reg2 = lm(cnt ~ hum, data = data)
summary(reg2)
with(data ,plot(hum, cnt))
abline(reg2) #rental count decrease with increase in humidity
```

```
#----------------------------------------------OUTLIER ANALYSIS-------------------------------------
#Boxplotting
numeric_index = sapply(data, is.numeric)
numeric_data = data[,numeric_index]
cnames = colnames(numeric_data)
for (i in 1:length(cnames)){
assign(paste0("gn", i), ggplot(aes_string(y = cnames[i], x = "cnt"), data = subset(data)) +
stat_boxplot(geom = "errorbar", width = 0.5) +
geom_boxplot(outlier.colour = "red", fill = "blue", outlier.shape = 18, outlier.size = 1, notch =
FALSE) +
theme(legend.position = "bottom") + labs(y = cnames[i], x="count") + ggtitle(paste("Boxplot of
count for", cnames[i])))
}
#plotting plots together
gridExtra::grid.arrange(gn2, gn3, gn4, ncol = 3)
gridExtra::grid.arrange(gn5, gn6, gn7, ncol = 3)
#replace outliers with NA and impute
for(i in cnames) {
print(i)
val = data[,i][data[,i] %in% boxplot.stats(data[,i]) $out]
print(length(val))
data[,i][data[,i] %in% val] = NA
}
#imputing NA values
data$casual[is.na(data$casual)] = data$cnt - data$registered
data$hum[is.na(data$hum)] = mean(data$hum,na.rm = T)
data$windspeed[is.na(data$windspeed)] = mean(data$windspeed, na.rm = T)
# creating copy
copy1 = data

#---------------------------------------FEATURE SELECTION-----------------------------------------
install.packages("corrgram")
library(corrgram)
round(cor(numeric_data),2)
corrgram(data[, numeric_index], order = F, upper.panel = panel.pie, text.panel = panel.txt, main
= "correlation plot")
#temp and atemp are strongle correlated
data = subset(data, select=-c(instant, atemp, casual, registered, dteday))
#anova test
AnovaModel_season =(lm(cnt ~ season, data = data))
summary(AnovaModel_season) #keep
AnovaModel_year =(lm(cnt ~ yr, data = data))
summary(AnovaModel_year) #keep
AnovaModel_month =(lm(cnt ~ mnth, data = data))
```

```
summary(AnovaModel_month) #keep
AnovaModel_holiday =(lm(cnt ~ holiday, data = data))
summary(AnovaModel_holiday) #remove
AnovaModel_weekday =(lm(cnt ~ weekday, data = data))
summary(AnovaModel_weekday) # keep
AnovaModel_workingday =(lm(cnt ~ workingday, data = data))
summary(AnovaModel_workingday) #remove
AnovaModel_weathersit =(lm(cnt ~ weathersit, data = data))
summary(AnovaModel_weathersit) #keep
data = subset(data, select=-c(holiday, workingday))
#Multicollinearity test
install.packages("usdm")
library(usdm)
vifcor(data[,c(6,7,8)])

#-----------------------------------------FEATURE SCALING---------------------------------------
hist(data$temp)
hist(data$hum)
hist(data$windspeed)
hist(data$cnt)
#data['cnt'] = (data$cnt-min(data$cnt))/(max(data$cnt)-min(data$cnt))

#-----------------------------------------MODELING----------------------------------------
#sampling
set.seed(101)
train_index = sample(1:nrow(data), 0.8*nrow(data))
data_train = data[train_index,]
data_test = data[-train_index,]
#mape
mape = function(actual, predict){
mean(abs((actual-predict)/actual))*100
}

##Random forest
install.packages("randomForest")
library(randomForest)
library(inTrees)
#model
set.seed(101)
model_RF = randomForest(cnt ~. , data_train, importance = TRUE, ntree = 500)
model_RF
#error plotting
plot(model_RF)
#predict test data using RF model
```

```
RF_predict = predict(model_RF, data_test[,-9])
plot(data_test$cnt, RF_predict, xlab = 'Actual values', ylab = 'Predicted values', main = 'RF
model')
install.packages("caret")
library(caret)
postResample(RF_predict, data_test$cnt)#R-sq = 0.88
mape(data_test$cnt, RF_predict)
varImpPlot(model_RF)

##Linear regression
install.packages("dummies")
library(dummies)
#?dummy.data.frame()
data_new = dummy.data.frame(data, sep = '_')
set.seed(101)
train_index1 = sample(1:nrow(data_new), 0.8*nrow(data_new))
data_train_new = data_new[train_index1,]
data_test_new = data_new[-train_index1,]
#model
set.seed(101)
model_LR = lm(cnt ~. , data = data_train_new)
summary(model_LR)
#predictions
LR_predict = predict(model_LR, data_test_new[,-32])
plot(data_test_new$cnt, LR_predict, xlab = 'Actual values', ylab = 'Predicted values', main = 'LR
model')
#evaluation statistics
postResample(LR_predict, data_test_new$cnt)#R-sq = 0.83
mape(data_test_new$cnt, LR_predict)

##Decison tree
install.packages("rpart.plot")
library(rpart.plot)
library(rpart)
#model
set.seed(101)
model_DT = rpart(cnt~. , data = data_train, method = "anova")
summary(model_DT)
plt = rpart.plot(model_DT, type = 5, digits = 2, fallen.leaves = TRUE)
#?rpart.plot
#predictions
DT_Predict = predict(model_DT, data_test[,-9])
plot(data_test$cnt, DT_Predict, xlab = 'Actual values', ylab = 'Predicted values', main = 'DT
model')
```

```
#evaluation statistics
postResample(DT_Predict, data_test$cnt)#R-sq = 0.76
mape(data_test$cnt, DT_Predict)

##SVM
install.packages("e1071")
library(e1071)
#model
set.seed(101)
SVM_model = svm(cnt ~., data = data_train)
#predictions
SVM_predict = predict(SVM_model, data_test[,-9])
plot(data_test$cnt, SVM_predict, xlab = 'Actual values', ylab = 'Predicted values', main = 'SVM
model')
#evaluation statistics
postResample(SVM_predict, data_test$cnt)
#RMSE = 0.08, R-sq = 0.88, MAE = 0.05

#------------------------------------------------------------------------------
#CROSS VALIDATION RESULTS
set.seed(101)
train_control = trainControl(method="cv", number=10)
model1 = train(cnt~., data=data_train, trControl=train_control, method="rf")
print(model1)
pred1 = predict(model1, data_test[,-9])
postResample(pred1, data_test$cnt) #Rsq = 0.84
mape(data_test$cnt, pred1)
plot(data_test$cnt, pred1)

#model2 = train(cnt~., data=data_train, trControl=train_control, method="rpart")
#print(model2)
#pred2 = predict(model2, data_test[,-9])
#postResample(pred2, data_test$cnt) #Rsq = 0.62
#mape(data_test$cnt, pred2)

#model3 = train(cnt~., data=data_train_new, trControl=train_control, method="lm")
#print(model3)
#pred3 = predict(model3, data_test_new[,-32])
#postResample(pred3, data_test_new$cnt) #Rsq = 0.85
#mape(data_test_new$cnt, pred3)

#model4 = train(cnt~., data=data_train, trControl=train_control, method="svmPoly")
#print(model4)
#pred4 = predict(model4, data_test[,-9])
```

```
#postResample(pred4, data_test$cnt) #Rsq = 0.87
#mape(data_test$cnt, pred4)
```

Python code

```
# coding: utf-8

# In[ ]:


import os
os.chdir("E:/Study/edWisor Stuff/Project2")
os.getcwd()


# In[ ]:


import pandas as pd
import numpy as np


# In[ ]:


data = pd.read_csv("day.csv", sep = ',')


# In[ ]:


data.shape


# In[ ]:


data.head()


# In[ ]:


data.dtypes


# In[ ]:


#changing data types
for i in ['season' , 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
'weathersit']:
```

```
    data[i] = data[i].astype(object)


# In[ ]:


copy1 = data.copy()
#data = copy1


# In[ ]:


#missing value analysis
pd.isnull(data).sum() #no missing value


# # Visualization

# In[ ]:


sns.pairplot(data)


# In[ ]:


import matplotlib.pyplot as plt
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')


# In[ ]:


data.boxplot(column='cnt', by=['yr','mnth']) #over the years count of
rental bikes have increased


# In[ ]:


#season count
data.boxplot(column='cnt', by='season') #season 3 have maximum count
while season 1 have least


# In[ ]:
```

```
sns.countplot(x='workingday', data= data) #working days have more
count


# In[ ]:


sns.barplot(x='weekday', y='cnt', data= data) #every day of the week
have same count


# In[ ]:


sns.countplot(x='holiday', data= data)


# In[ ]:


sns.countplot(x='weathersit', data= data) #weathersit 1 have highest
count


# In[ ]:


plt.scatter(x='season', y='mnth', data= data)


# In[ ]:


sns.lmplot(x='temp',y='cnt',data=data, hue='season')


# In[ ]:


sns.distplot(data['temp'])
sns.distplot(data['atemp']) # seems correlated
plt.xlabel('temp and atemp')


# In[ ]:


cname = ['temp','atemp',
'hum','windspeed','casual','registered','cnt']
numeric_data = data[cname]
numeric_data.head()
```

```
# # Outlier Analysis

# In[ ]:


plt.subplot(2,4,1)
plt.boxplot(data['temp'])
plt.title('temp')

plt.subplot(2,4,2)
plt.boxplot(data['atemp'])
plt.title('atemp')

plt.subplot(2,4,3)
plt.boxplot(data['hum'])
plt.title('hum')

plt.subplot(2,4,4)
plt.boxplot(data['windspeed'])
plt.title('windspeed')

plt.subplot(2,4,5)
plt.boxplot(data['casual'])
plt.title('casual')

plt.subplot(2,4,6)
plt.boxplot(data['registered'])
plt.title('registered')

plt.subplot(2,4,7)
plt.boxplot(data['cnt'])
plt.title('cnt')

plt.tight_layout()


# In[ ]:


#outlier analysis
for i in cname:
    print(i)
    q75, q25 = np.percentile(data.loc[:,i],[75,25])
    iqr = q75- q25
    min = q25 - (1.5*iqr)
    max = q75 + (1.5*iqr)

    data.loc[data.loc[:,i] < min , i] = np.nan
    data.loc[data.loc[:,i] > max , i] = np.nan
```

```
# In[ ]:


#pd.isnull(data).sum() #59outliers


# In[ ]:


#Windspeed Vs Season
data.boxplot(column='windspeed', by=['season'])


# In[ ]:


#imputing missing windspeed
def impute_windspeed(cols):
    Season = cols[0]
    Windspeed = cols[1]

    if pd.isnull(Windspeed):
        if(Season==1 | Season==2):
            return 0.19
        else:
            return 0.17
    else:
        return Windspeed


data['windspeed'] =
data[['season','windspeed']].apply(impute_windspeed, axis=1)


# In[ ]:


#imputing missing humidity
data['hum'] = data['hum'].fillna(data['hum'].mean())


# In[ ]:


#imputing missing casual
data['casual'] = data['casual'].fillna(data['cnt']-data['registered'])


# # Feature Selection

# In[ ]:
```

```
data.corr()


# In[ ]:


sns.heatmap(data.corr(), vmin=-1.00, vmax=1.00, annot=True)


# In[ ]:


#Dropping variables
data = data.drop(['instant','atemp','casual','registered','dteday'],
axis=1)
#instant is unique for all observations hence has no significance
#atemp is strongly correlated with temp
#cnt = casual + registered
#data = data.drop(['instant','dteday'], axis=1)


# In[ ]:


#feature importance from random forest
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 100, random_state =
123).fit(data.iloc[:,0:10],data.iloc[:,10])
print(rf.feature_importances_)

#Feature importance plotting
names=list(data)
names = names[0:10]

sns.barplot(x=names ,y=rf.feature_importances_)  #remove holiday,
working day

plt.title('Feature Importance')
plt.xlabel('Features')
plt.ylabel('Importance')


# In[ ]:


#anova test
import statsmodels.api as sm
from statsmodels.formula.api import ols


# In[ ]:
```

```
mod1 = ols('cnt ~ season', data = data).fit()
aov_table1 = sm.stats.anova_lm(mod1, type=2)
print(aov_table1) #keep


# In[ ]:


mod2 = ols('cnt ~ yr', data = data).fit()
aov_table2 = sm.stats.anova_lm(mod2, type=2)
print(aov_table2) #keep


# In[ ]:


mod3 = ols('cnt ~ mnth', data = data).fit()
aov_table3 = sm.stats.anova_lm(mod3, type=2)
print(aov_table3) #keep


# In[ ]:


mod4 = ols('cnt ~ holiday', data = data).fit()
aov_table4 = sm.stats.anova_lm(mod4, type=2)
print(aov_table4) #remove


# In[ ]:


mod5 = ols('cnt ~ weekday', data = data).fit()
aov_table5 = sm.stats.anova_lm(mod5, type=2)
print(aov_table5) #remove


# In[ ]:


mod6 = ols('cnt ~ workingday', data = data).fit()
aov_table6 = sm.stats.anova_lm(mod6, type=2)
print(aov_table6) #remove


# In[ ]:


mod7 = ols('cnt ~ weathersit', data = data).fit()
aov_table7 = sm.stats.anova_lm(mod7, type=2)
```

```
print(aov_table7) #keep


# In[ ]:


#from anova and feature importance plotting of random forest, we
decided to drop holiday and workingday
data = data.drop(['holiday', 'workingday'], axis=1)


# In[ ]:


data.head()


# In[ ]:


#VIF test
from patsy import dmatrices
from statsmodels.stats.outliers_influence import
variance_inflation_factor

#get x and y dataframe
y, x = dmatrices('cnt~ + season + yr + mnth + weathersit + weekday +
temp + hum + windspeed', data, return_type = 'dataframe')

vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(x.values, i) for i in
range(x.shape[1])]
vif["features"] = x.columns
vif


# In[ ]:


#FEATURE SCALING
#distribution of continuous features
plt.figure(figsize=(14,4))

plt.subplot(1,4,1)
sns.distplot(data['temp'])
plt.title('temperature distribution')

plt.subplot(1,4,2)
sns.distplot(data['hum'])
plt.title('humidity distribution')

plt.subplot(1,4,3)
```

```
sns.distplot(data['windspeed'])
plt.title('windspeed distribution')

plt.subplot(1,4,4)
sns.distplot(data['cnt'])
plt.title('count distribution')

plt.tight_layout()


# In[ ]:


#Feature Scaling
#data['cnt'] = (data['cnt'] - data['cnt'].min())/(data['cnt'].max() -
data['cnt'].min())
#we are not scaling the target variable


# In[ ]:


#MODELING
#sampling
from sklearn.cross_validation import train_test_split
from sklearn import metrics
from sklearn.model_selection import KFold, cross_val_score,
cross_val_predict #cross validation


# In[ ]:


train, test = train_test_split(data, test_size = 0.25, random_state =
100)
data.shape, test.shape , train.shape


# In[ ]:


#calculate evaluation statistics
def evaluation_stats(actual, predict):
    print('MSE:', metrics.mean_squared_error(actual, predict))
    print('RMSE:', np.sqrt(metrics.mean_squared_error(actual,
predict)))
    print('MAPE:',np.mean(np.abs((actual-predict)/actual))*100)
    print('R-Sq:', metrics.r2_score(actual, predict))


# In[ ]:
```

```
#Decision Tree
from sklearn.tree import DecisionTreeRegressor

k_fold = KFold(n_splits = 10, shuffle=True,
random_state=101).get_n_splits(train.iloc[:,0:8])

dt1 = DecisionTreeRegressor(max_depth = 4,
random_state=123).fit(train.iloc[:,0:8],train.iloc[:,8])
rmse = np.sqrt(cross_val_score(dt1, train.iloc[:,0:8],
train.iloc[:,8], cv= k_fold))
#print('RMSE training : ', rmse)

prediction_dt1 = cross_val_predict(dt1, test.iloc[:,0:8],
test.iloc[:,8] , cv=k_fold)

#error matrix
evaluation_stats(test.iloc[:,8],prediction_dt1)


# In[ ]:


#Random Forest
from sklearn.ensemble import RandomForestRegressor

k_fold = KFold(n_splits = 10, shuffle=True,
random_state=101).get_n_splits(train.iloc[:,0:8])

rf1 = RandomForestRegressor(n_estimators = 100, random_state =
123).fit(train.iloc[:,0:8],train.iloc[:,8])
rmse = np.sqrt(cross_val_score(rf1, train.iloc[:,0:8],
train.iloc[:,8], cv= k_fold))
#print('RMSE training : ', rmse)

#predictions
prediction_rf1 = cross_val_predict(rf1, test.iloc[:,0:8],
test.iloc[:,8] , cv=k_fold)

#error matrix
evaluation_stats(test.iloc[:,8],prediction_rf1)


# In[ ]:


#Linear Regression
from sklearn.linear_model import LinearRegression

#creating dummies
data_LR = pd.get_dummies(data, columns = ['season', 'yr',
'mnth','weekday','weathersit'], drop_first = True)
```

```
data_LR

col = list(data_LR)
col = col[0:3] + col[4:27] + [col[3]]
data_LR = data_LR[col]

#lr sampling
train_LR, test_LR = train_test_split(data_LR, test_size = 0.25,
random_state = 100)
k_fold_LR = KFold(n_splits = 10, shuffle=True,
random_state=101).get_n_splits(train_LR.iloc[:,0:26])

#model
lin_model = LinearRegression().fit(train_LR.iloc[:,0:26],
train_LR.iloc[:,26])

#pred_LR = lin_model.predict(test_LR.iloc[:,0:26])
pred_LR = cross_val_predict(lin_model, test_LR.iloc[:,0:26],
test_LR.iloc[:,26] , cv=k_fold_LR, verbose= True)

#error matrix
evaluation_stats(test_LR.iloc[:,26],pred_LR)


# In[ ]:


train_LR.head()


# In[ ]:


plt.figure(figsize=(14,4))

plt.subplot(1,3,1)
plt.title('Decision Tree predictions')
plt.scatter(test.iloc[:,8] , prediction_dt1)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')

plt.subplot(1,3,2)
plt.title('Random Forest predictions')
plt.scatter(test.iloc[:,8] , prediction_rf1)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')

plt.subplot(1,3,3)
plt.title('Linear Regression Predictions')
plt.scatter(test_LR.iloc[:,26] , pred_LR)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
```

```
plt.tight_layout()

# In[ ]:

#finalise random forest
```