Project Report

Bike Renting

July 25, 2019

Contents

Chapter 1

Introduction

1.1 Problem Statement

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

1.2 Data

Data Set – consists of one csv file i.e. day.csv

Number of Instances: 731

Number of Attributes: 16

Missing Values: N/A

Variables Information:

The details of data attributes in the dataset are as follows -

instant: Record index

dteday: Date

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

yr: Year (0: 2011, 1:2012)

mnth: Month (1 to 12)

hr: Hour (0 to 23)

holiday: weather day is holiday or not (extracted fromHoliday Schedule)

weekday: Day of the week

workingday: If day is neither weekend nor holiday is 1, otherwise is 0.

weathersit: (extracted fromFreemeteo)

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

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

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

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

temp: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)

atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-16, t_max=+50 (only in hourly scale)

hum: Normalized humidity. The values are divided to 100 (max)

windspeed: Normalized wind speed. The values are divided to 67 (max)

Dependent variables:

casual: count of casual users

registered: count of registered users

cnt: count of total rental bikes including both casual and registered

Sample of the dataset:

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

Chapter 2

Methodology

2.1 Pre-Processing

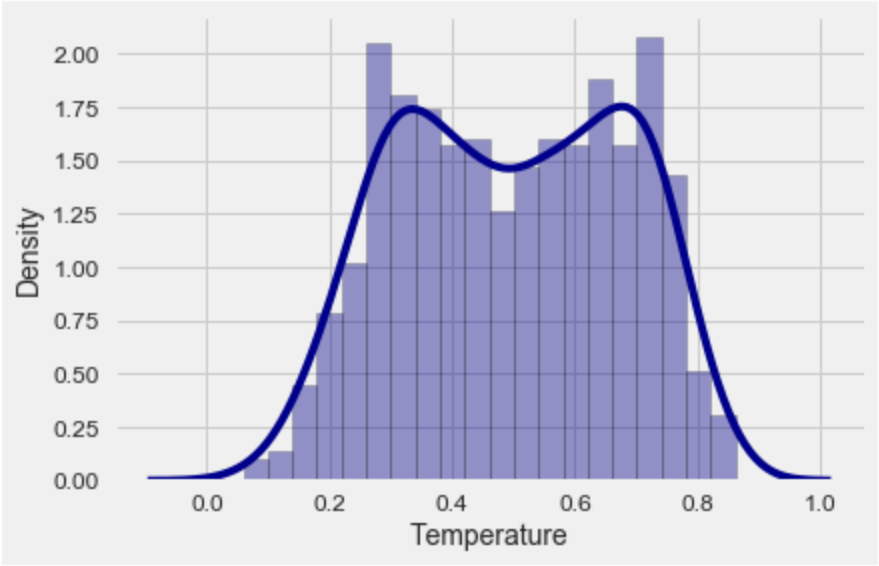
Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This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

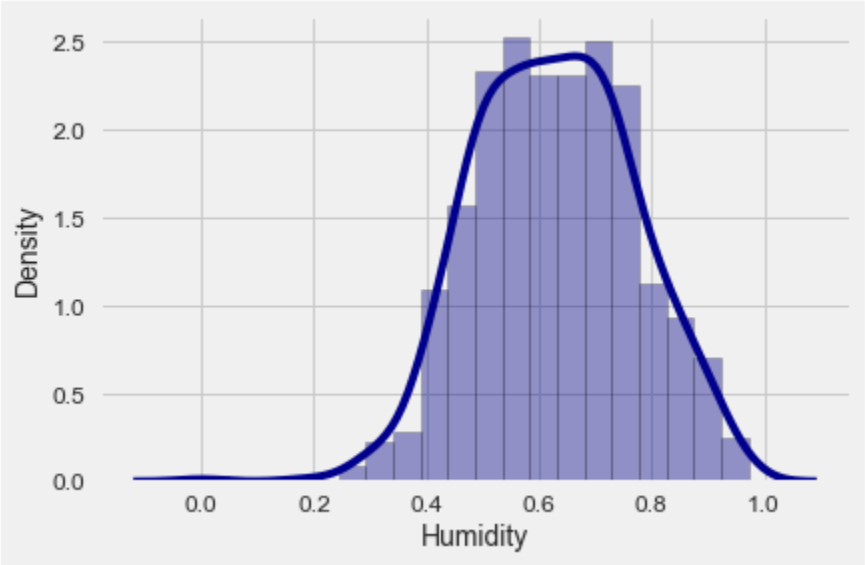
Here are some of the hypothesis which I thought could influence the demand of bikes:

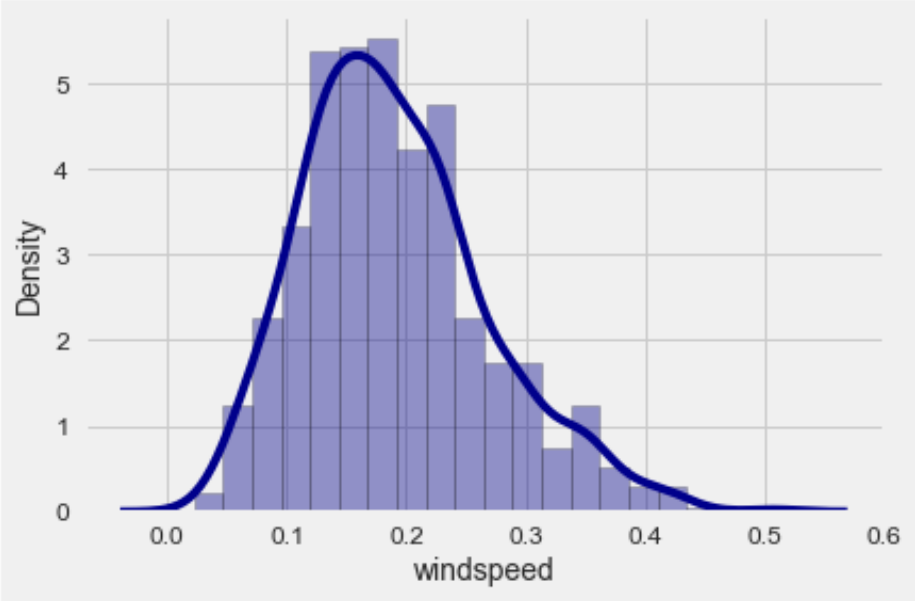
- **Weekends**: casual users count will increase over weekend.
- **Humidity**: higher humidity will cause slower demand and vice versa.
- **Temperature**: Temperature will have positive correlation with bike demand in cold countries.
- **Weather**: Clear weather will bring more users as compared to rainy/misty weather.
- **Time** (**year**): Total demand will have higher contribution of registered user as compared to casual because registered user base would increase over time.

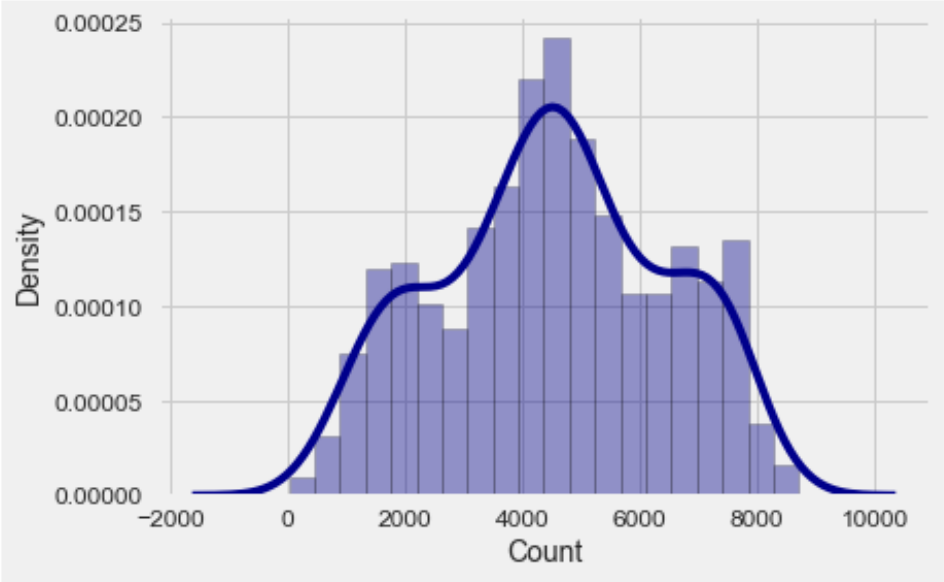
To start this process, we will first try and look at all the probability distributions of the variables. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

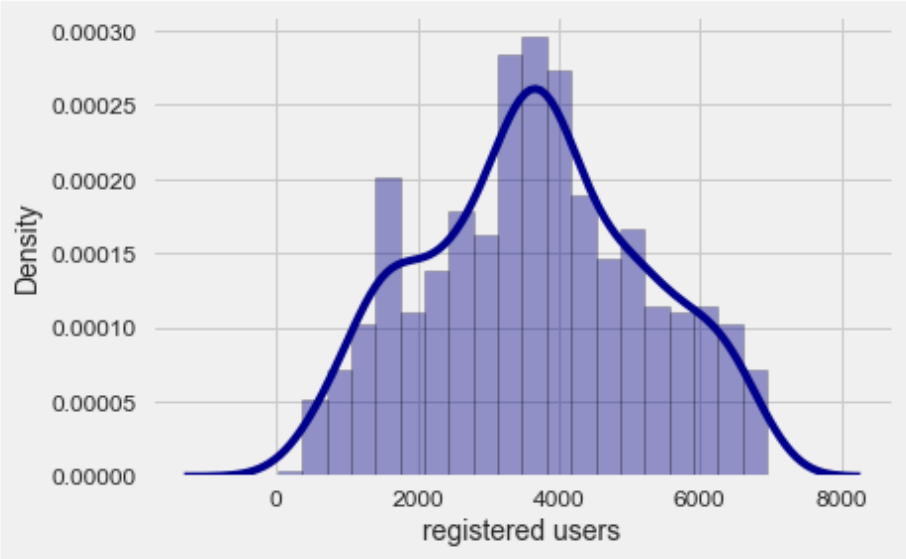
We have plotted the probability density functions of all the independent continuous variables in the data as well as the dependent variable. The blue lines indicate Kernel Density Estimations (KDE) of the variable.

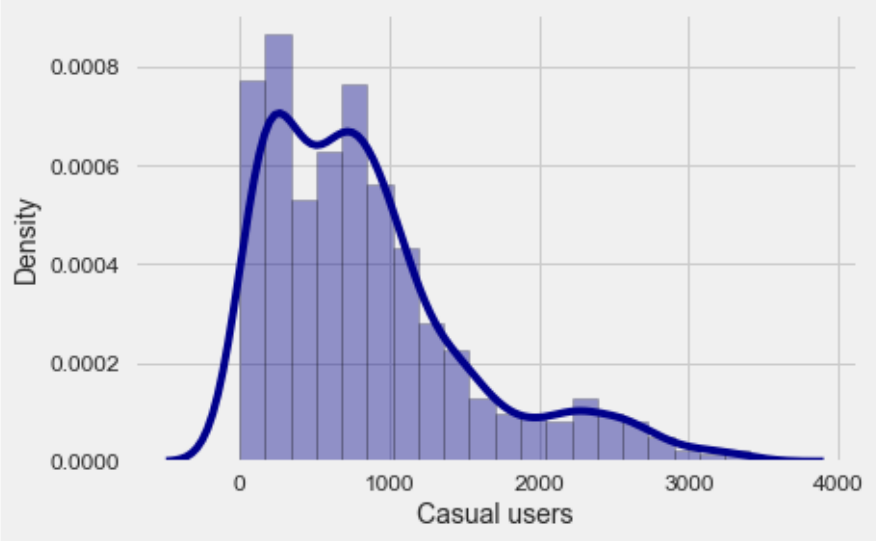

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Density
0.0
0.2
0.4
0.6
0.8
1.0
Temperature


2.5
2.0
1.5
1.0
0.5
0.0
Density
0.0
0.2
0.4
0.6
0.8
1.0
Humidity

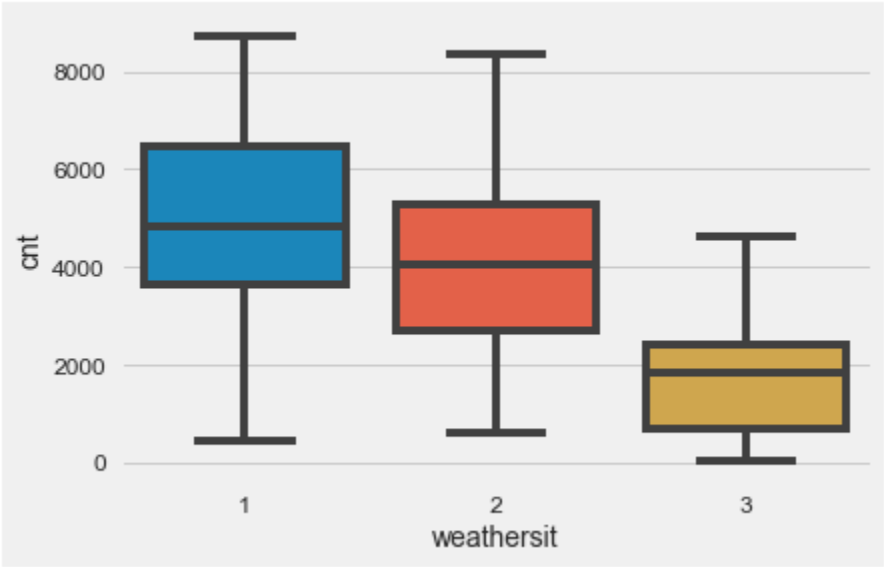

Density
windspeed
0
1
2
3
4
5
0.0
0.1
0.2
0.3
0.4
0.5
0.6

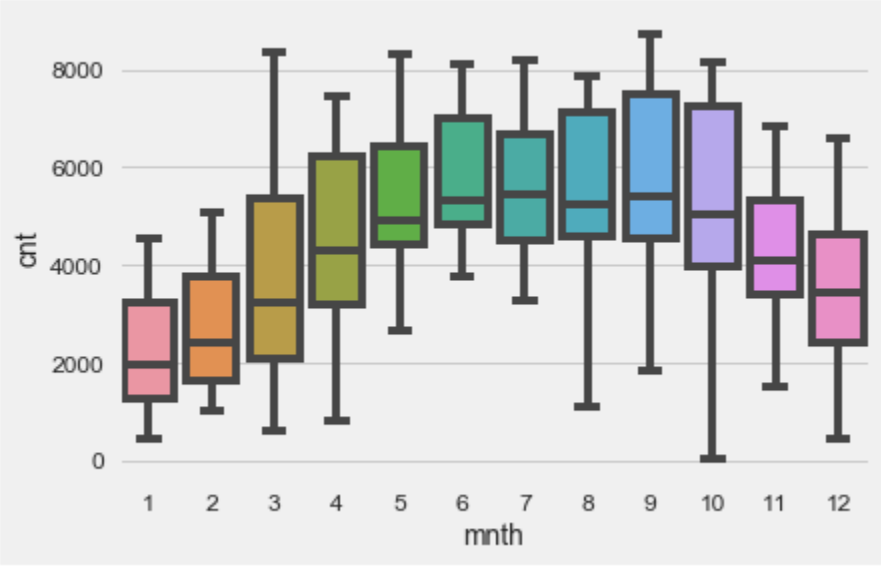

Density
Count
0.00000
0.00005
0.00010
0.00015
0.00020
0.00025
−2000
0
2000
4000
6000
8000
10000

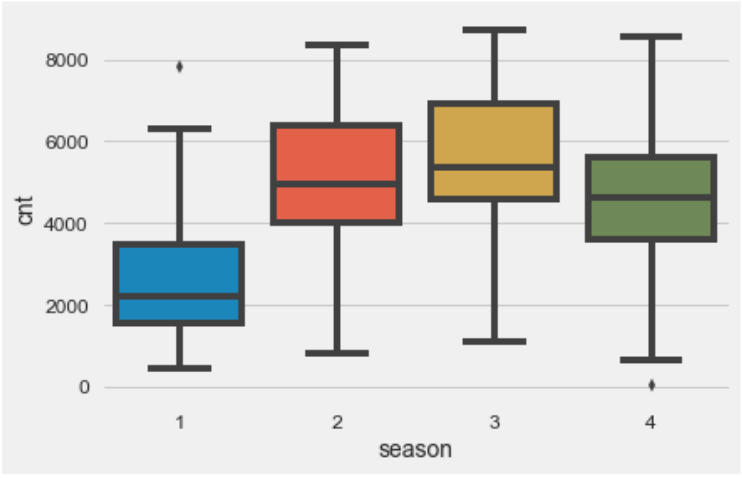

0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
Density
0
2000
4000
6000
8000
registered users

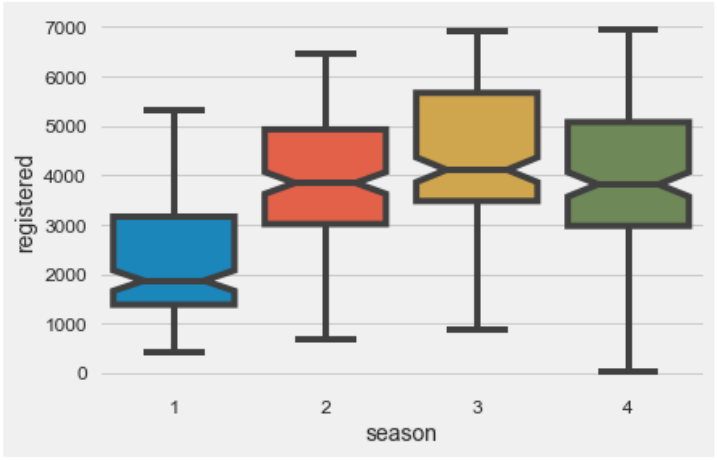

0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
1000
2000
3000
4000
Casual users

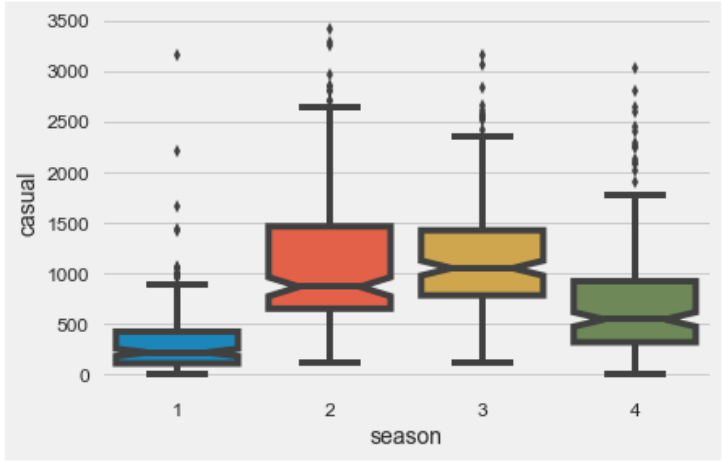
Let’s plot the box plot to visualize the relationship of continuous variables w.r.t categorical variables.

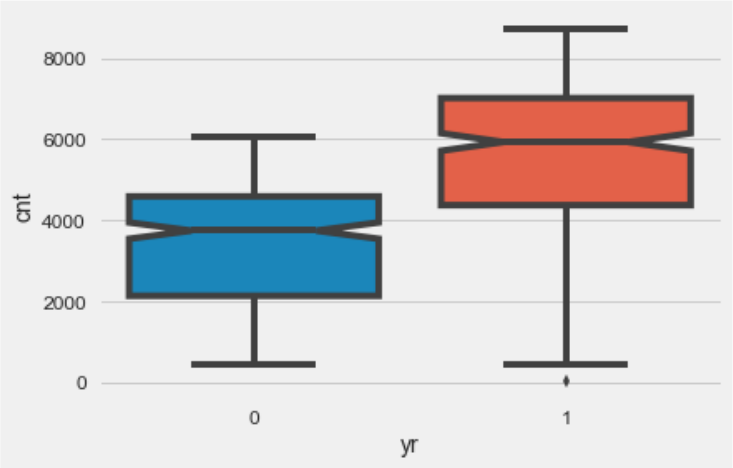


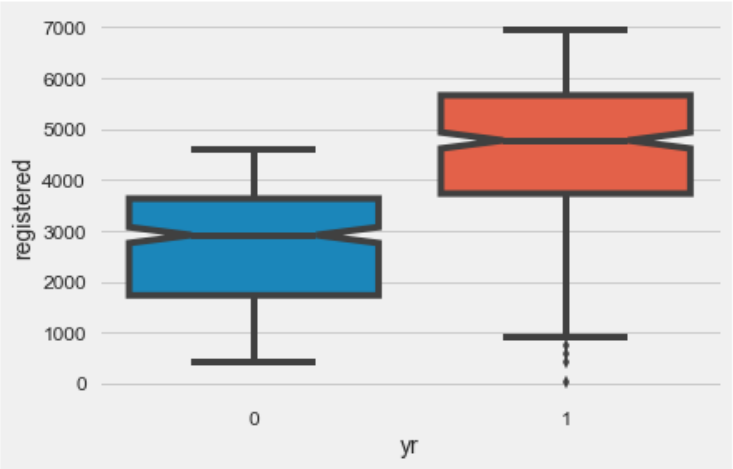


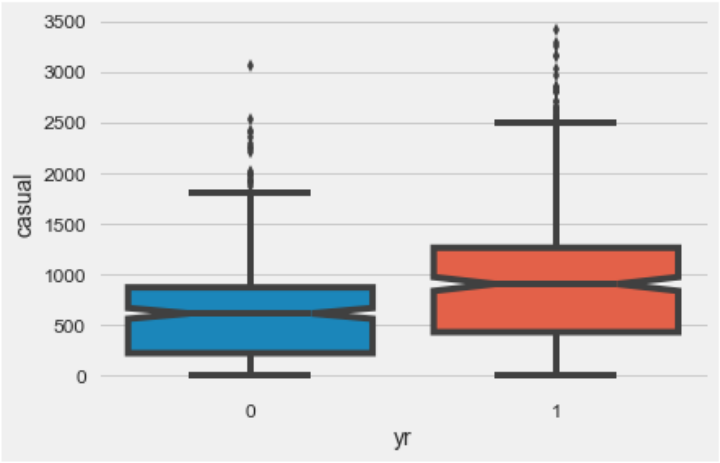

8000
6000
4000
2000
0
cnt
1
2
3
4
season

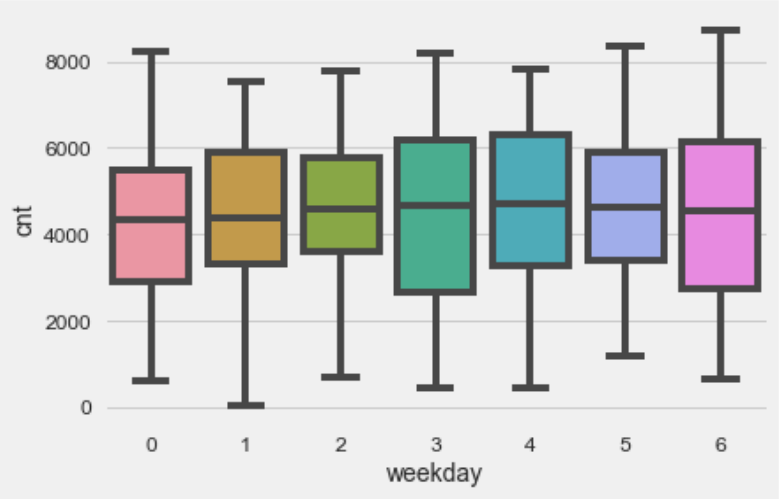

7000
6000
5000
4000
3000
2000
1000
0
registered
1
2
3
4
season

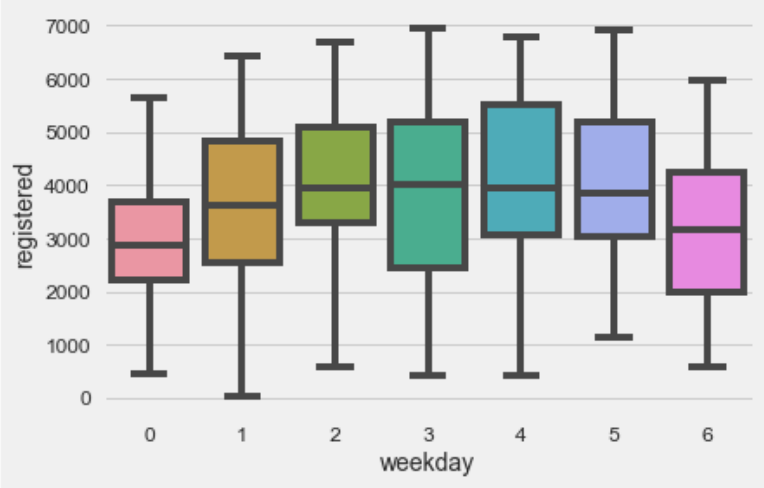

3500
3000
2500
2000
1500
1000
500
0
casual
1
2
3
4
season

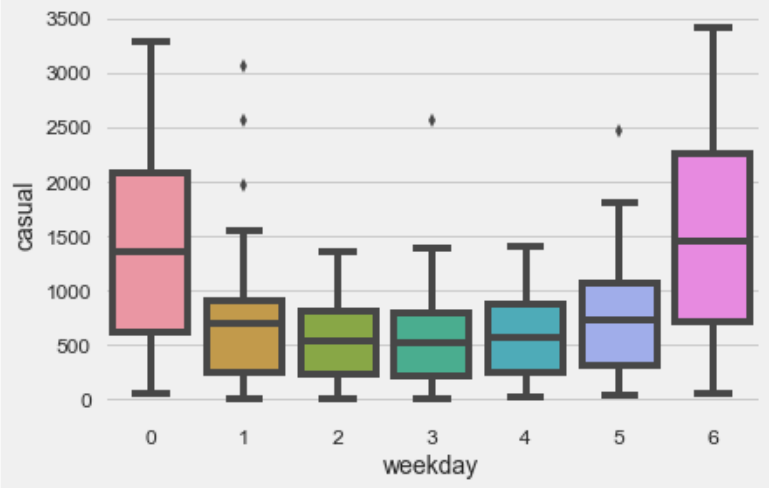

8000
6000
4000
2000
0
cnt
0
1
yr


7000
6000
5000
4000
3000
2000
1000
0
registered
0
1
yr


3500
3000
2500
2000
1500
1000
500
0
casual
0
1
yr


cnt
8000
6000
4000
2000
0
0
1
2
3
4
5
6
weekday


registered
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
6
weekday


casual
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
weekday

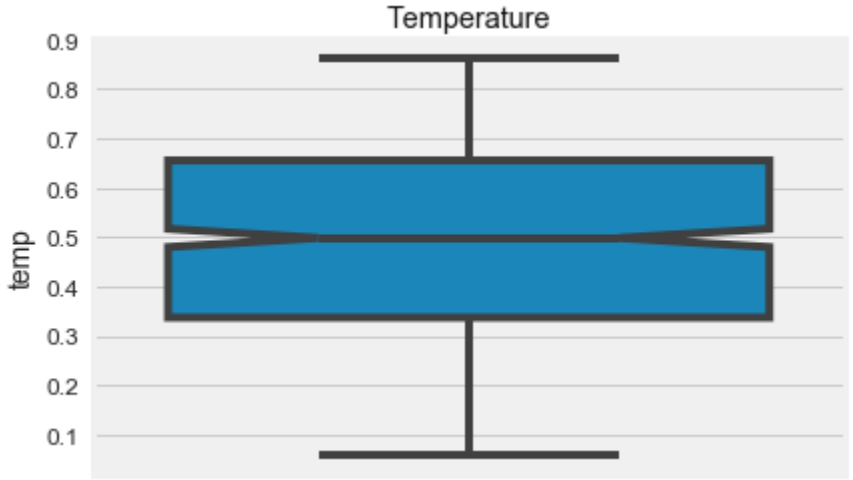
Missing value Analysis:

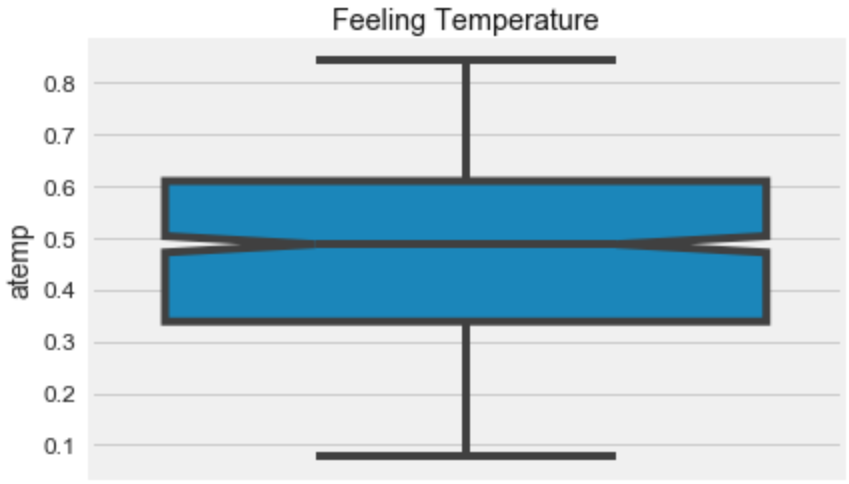
Data set does not contain any missing values.

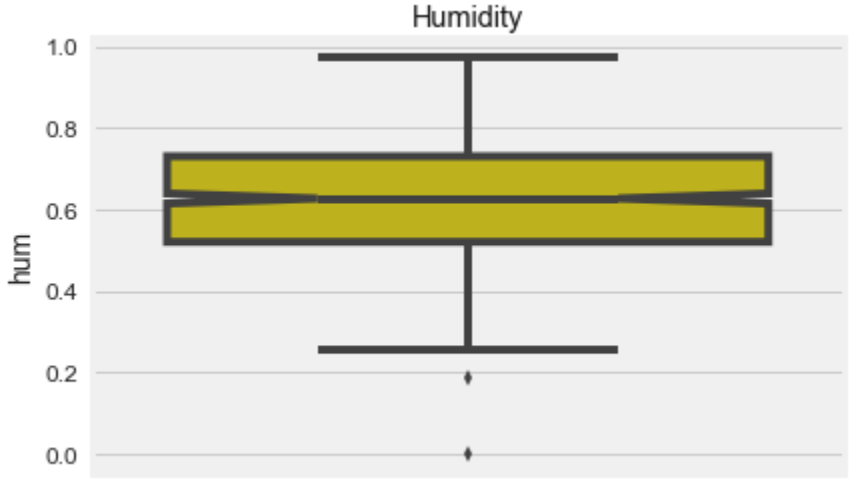
cnt	0
registered	0
casual	0
windspeed	0
hum	0
atemp	0
temp	0
weathersit	0
workingday	0
weekday	0
holiday	0
mnth	0
yr	0
season	0
dteday	0
instant	0

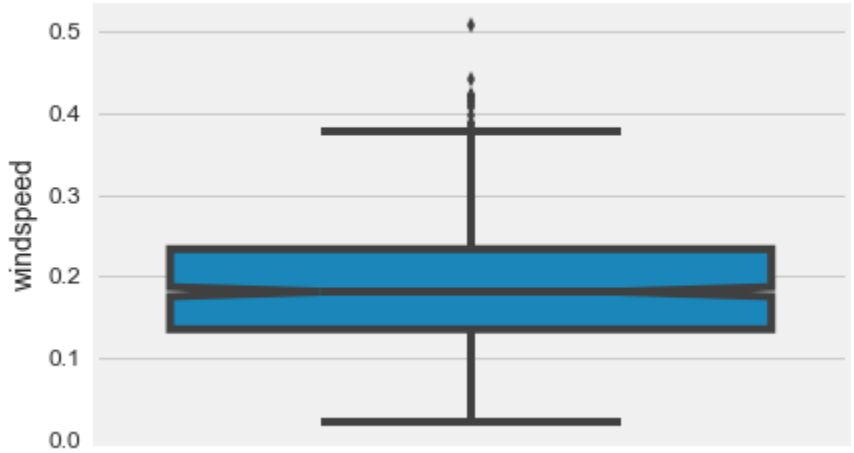
2.1.1 Outlier Analysis

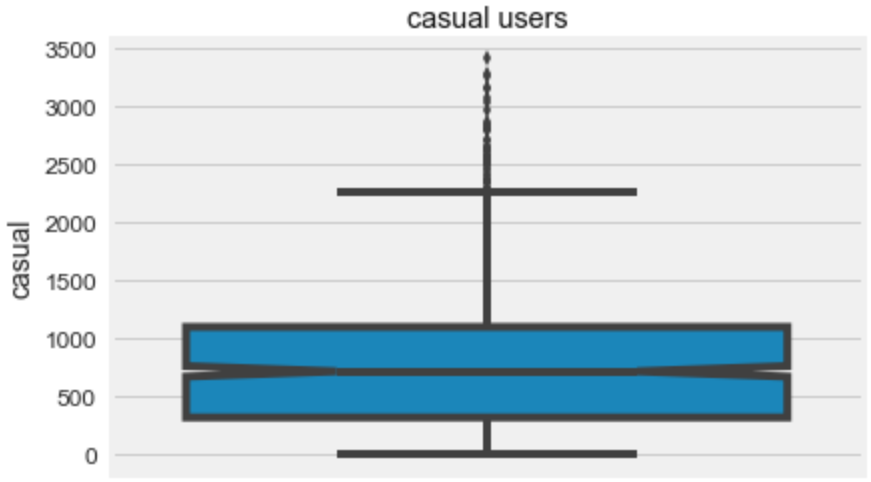
Casual, windspeed and Humidity features have natural outliers. So, no outlier handling analysis required.



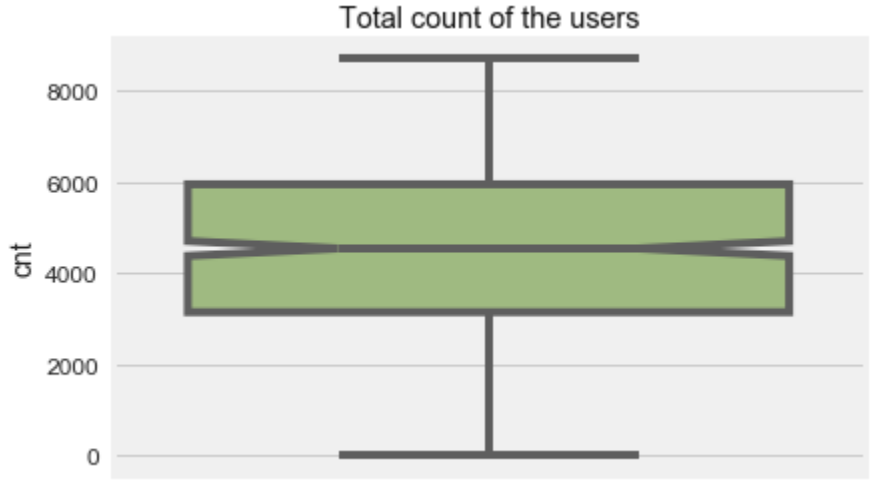

Feeling Temperature
atemp
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1


Humidity
hum
1.0
0.8
0.6
0.4
0.2
0.0


Wind speed
windspeed
0.5
0.4
0.3
0.2
0.1
0.0


casual users
casual
3500
3000
2500
2000
1500
1000
500
0

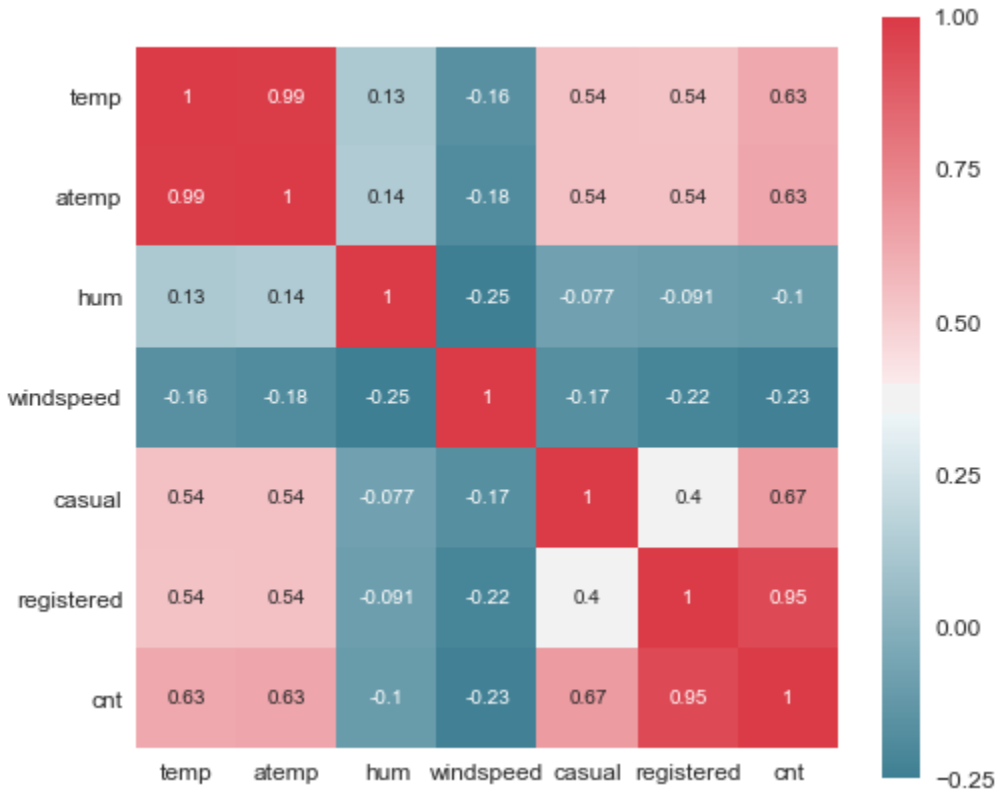

Registeres users
registered
7000
6000
5000
4000
3000
2000
1000
0


Total count of the users
cnt
8000
6000
4000
2000
0

2.1.2 Feature Selection

Before performing any type of modeling, we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that.

A very simple way of looking at correlations in the data is shown below through the correlation matrix:



The 'casual' and 'registered' columns are simply subcategories of the 'cnt' column.
These columns leak information on the target column, so we'll have to drop them.

Also, “team” and “atemp” have high value of correlation. So, we will drop one of them (here atemp).

ANOVA test for categorical variables:

ANOVA test is used only when you want to find the relationship between a categorical predictor variable and a quantitative target variable.

But how does the ANOVA work? It simply compares the means and variations of different groups (or levels) of the categorical variable and tests if there is any significant difference in their values. It yes,

then we can say that there is an association (relationship) between the categorical predictor variable and quantitative target variable, otherwise not.

Null Hypothesis: There is nothing going on between the variables, there is no relationship between the two variables X and Y.

Alternate Hypothesis: There is something going on between the predictor and target variable, or there is a relationship between the two.

$$F = \frac{variation\ among\ group\ means}{variation\ within\ group}$$

P- value is defined as the probability of getting that observed F-statistic (after the ANOVA test) or more extreme value of F-statistic provided the null hypothesis is true. So, if we get very less p-value, it means it is extremely rare to get that F-statistic when the null hypothesis is true (sometimes it may occur which is also known as Type I error) and we can reject the null hypothesis. Generally, we take the cutoff for p-value as 0.05 (which is 95% significance level).

ANOVA test for cnt ~ weathersit

As the p-value < 0.05, we reject the Null hypothesis.

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.099
Model:	OLS	Adj. R-squared:	0.097
Method:	Least Squares	F-statistic:	40.07
Date:	Sun, 21 Jul 2019	Prob (F-statistic):	3.11e-17
Time:	09:19:15	Log-Likelihood:	-6531.5
No. Observations:	731	AIC:	1.307e+04
Df Residuals:	728	BIC:	1.308e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4876.7862	85.567	56.994	0.000	4708.798	5044.774
weathersit[T.2]	-840.9238	145.073	-5.797	0.000	-1125.736	-556.112
weathersit[T.3]	-3073.5005	410.790	-7.482	0.000	-3879.975	-2267.026

Omnibus:	38.064	Durbin-Watson:	0.260
Prob(Omnibus):	0.000	Jarque-Bera (JB):	15.665
Skew:	-0.061	Prob(JB):	0.000397
Kurtosis:	2.293	Cond. No.	6.46

ANOVA test for cnt ~ season

As p < 0.05, we reject the Null hypothesis.

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.347
Model:	OLS	Adj. R-squared:	0.344
Method:	Least Squares	F-statistic:	128.8
Date:	Sun, 21 Jul 2019	Prob (F-statistic):	6.72e-67
Time:	09:19:15	Log-Likelihood:	-6413.9
No. Observations:	731	AIC:	1.284e+04
Df Residuals:	727	BIC:	1.285e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2604.1326	116.598	22.334	0.000	2375.223	2833.042
season[T.2]	2388.1989	164.221	14.543	0.000	2065.795	2710.603
season[T.3]	3040.1706	163.352	18.611	0.000	2719.472	3360.869
season[T.4]	2124.0303	165.588	12.827	0.000	1798.943	2449.118

Omnibus:	3.050	Durbin-Watson:	0.469
Prob(Omnibus):	0.218	Jarque-Bera (JB):	2.765
Skew:	0.080	Prob(JB):	0.251
Kurtosis:	2.745	Cond. No.	4.81

ANOVA test for cnt ~ yr

As p < 0.05, we reject the Null hypothesis.

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.321
Model:	OLS	Adj. R-squared:	0.320
Method:	Least Squares	F-statistic:	344.9
Date:	Sun, 21 Jul 2019	Prob (F-statistic):	2.48e-63
Time:	09:19:15	Log-Likelihood:	-6428.1
No. Observations:	731	AIC:	1.286e+04
Df Residuals:	729	BIC:	1.287e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3405.7616	83.601	40.738	0.000	3241.634	3569.889
yr[T.1]	2194.1728	118.149	18.571	0.000	1962.220	2426.126

Omnibus:	42.410	Durbin-Watson:	0.448
Prob(Omnibus):	0.000	Jarque-Bera (JB):	47.537
Skew:	-0.609	Prob(JB):	4.76e-11
Kurtosis:	2.720	Cond. No.	2.62

ANOVA test for cnt ~ mnth

As p < 0.05, we reject the Null hypothesis.

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.391
Model:	OLS	Adj. R-squared:	0.381
Method:	Least Squares	F-statistic:	41.90
Date:	Sun, 21 Jul 2019	Prob (F-statistic):	4.25e-70
Time:	09:19:15	Log-Likelihood:	-6388.6
No. Observations:	731	AIC:	1.280e+04
Df Residuals:	719	BIC:	1.286e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2176.3387	193.514	11.246	0.000	1796.419	2556.259
mnth[T.2]	478.9595	279.607	1.713	0.087	-69.985	1027.904
mnth[T.3]	1515.9194	273.670	5.539	0.000	978.631	2053.208
mnth[T.4]	2308.5613	275.941	8.366	0.000	1766.814	2850.308
mnth[T.5]	3173.4355	273.670	11.596	0.000	2636.147	3710.724
mnth[T.6]	3596.0280	275.941	13.032	0.000	3054.281	4137.775

ANOVA test for cnt ~ holiday

As p > .0.05, we accept the Null hypothesis.

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.005
Model:	OLS	Adj. R-squared:	0.003
Method:	Least Squares	F-statistic:	3.421
Date:	Sun, 21 Jul 2019	Prob (F-statistic):	0.0648
Time:	09:19:15	Log-Likelihood:	-6568.0
No. Observations:	731	AIC:	1.314e+04
Df Residuals:	729	BIC:	1.315e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4527.1042	72.582	62.372	0.000	4384.610	4669.599
holiday[T.1]	-792.1042	428.231	-1.850	0.065	-1632.817	48.608

Omnibus:	62.533	Durbin-Watson:	0.312
Prob(Omnibus):	0.000	Jarque-Bera (JB):	20.398
Skew:	-0.041	Prob(JB):	3.72e-05
Kurtosis:	2.186	Cond. No.	5.99

ANOVA test for cnt ~ weekday

As p > .0.05, we accept the Null hypothesis.

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.006
Model:	OLS	Adj. R-squared:	-0.002
Method:	Least Squares	F-statistic:	0.7829
Date:	Sun, 21 Jul 2019	Prob (F-statistic):	0.583
Time:	09:19:15	Log-Likelihood:	-6567.3
No. Observations:	731	AIC:	1.315e+04
Df Residuals:	724	BIC:	1.318e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4228.8286	189.221	22.349	0.000	3857.340	4600.317
weekday[T.1]	109.2952	267.599	0.408	0.683	-416.068	634.659
weekday[T.2]	281.8349	268.242	1.051	0.294	-244.790	808.460
weekday[T.3]	319.7099	268.242	1.192	0.234	-206.915	846.335
weekday[T.4]	438.4310	268.242	1.634	0.103	-88.194	965.056
weekday[T.5]	461.4599	268.242	1.720	0.086	-65.165	988.085
weekday[T.6]	321.7143	267.599	1.202	0.230	-203.649	847.078

Omnibus:	67.596	Durbin-Watson:	0.298
Prob(Omnibus):	0.000	Jarque-Bera (JB):	21.407
Skew:	-0.058	Prob(JB):	2.25e-05
Kurtosis:	2.170	Cond. No.	7.85

ANOVA test for cnt ~ workingday

As p > .0.05, we accept the Null hypothesis.

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.004
Model:	OLS	Adj. R-squared:	0.002
Method:	Least Squares	F-statistic:	2.737
Date:	Sun, 21 Jul 2019	Prob (F-statistic):	0.0985
Time:	09:19:15	Log-Likelihood:	-6568.3
No. Observations:	731	AIC:	1.314e+04
Df Residuals:	729	BIC:	1.315e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4330.1688	127.308	34.013	0.000	4080.235	4580.103
workingday[T.1]	254.6512	153.932	1.654	0.098	-47.552	556.854

Omnibus:	61.015	Durbin-Watson:	0.304
Prob(Omnibus):	0.000	Jarque-Bera (JB):	20.070
Skew:	-0.032	Prob(JB):	4.38e-05
Kurtosis:	2.191	Cond. No.	3.32

Data columns chosen for model development:

```
Data columns (total 8 columns):
season        731 non-null category
yr            731 non-null category
mnth          731 non-null category
weathersit    731 non-null category
temp          731 non-null float64
hum           731 non-null float64
windspeed     731 non-null float64
cnt           731 non-null int64
dtypes: category(4), float64(3), int64(1)
```

Feature Scaling

Feature scaling is a method used to normalize 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 step.
Also known as min-max scaling or min-max normalization, is the simplest method and consists in rescaling the range of features to scale the range in [0, 1] or [−1, 1]. Selecting the target range depends on the nature of the data. The general formula for a min-max of [0, 1] is given as:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

where is an original value, is the normalized value.

Independent variables temp, hum, windspeed are already normalized in the dataset.

2.2 Modeling
2.2.1 Model Selection

The dependent variable can fall in either of the four categories:

1. Nominal
2. Ordinal
3. Interval
4. Ratio

If the dependent variable is Nominal the only predictive analysis that we can perform is **Classification**, and if the dependent variable is Interval or Ratio the normal method is to do a **Regression** analysis, or classification after binning.

As our target variable (cnt) is Numerical, we will use **regression** algorithm.

We always start your model building from the simplest to more complex.

Base Line model:

For the base line model, we have considered a **Simple Linear regression** taking **temp** as explanatory variable and **cnt** as outcome variable.

Reason for choosing temp: A linear relationship between the **cnt** and **temp** variables and good positive correlation (0.63) value.

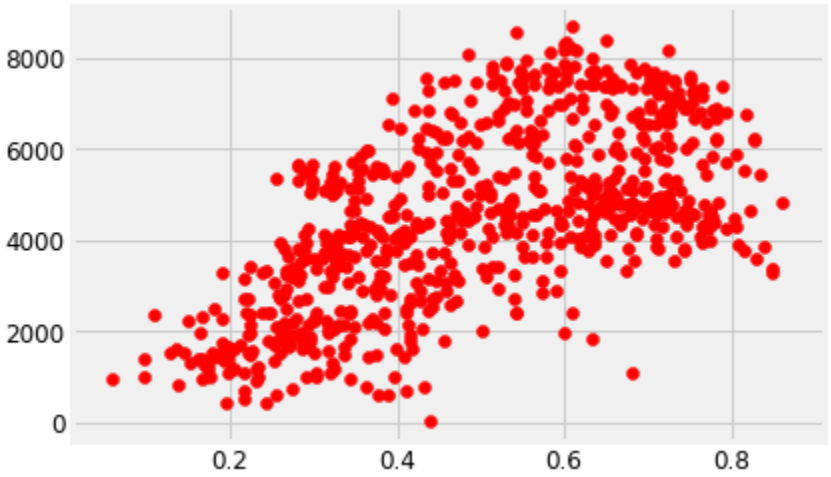


Fig. showing scatterplot of cnt vs temp

Error metrics obtained from Simple linear regression model:

Simple Linear Regression	TRAIN	TEST
RMSE	1513.809	1484.061
R2 score	0.377	0.445

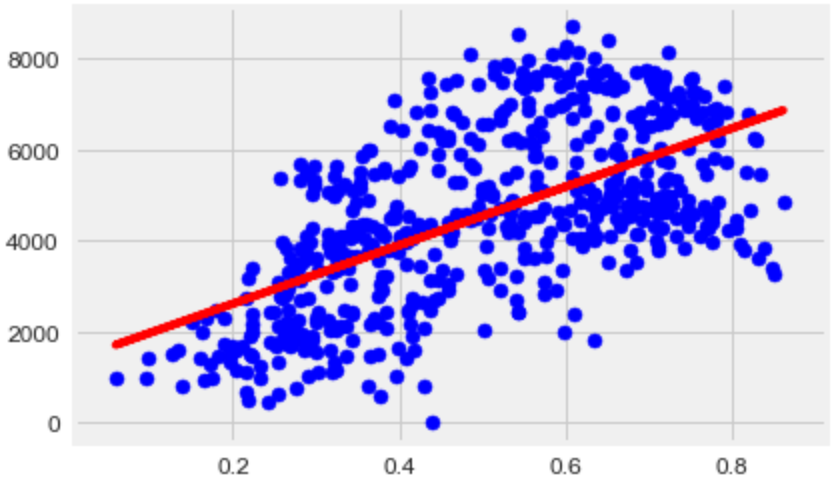


Fig. red line shows the best-fit regression line for the training data

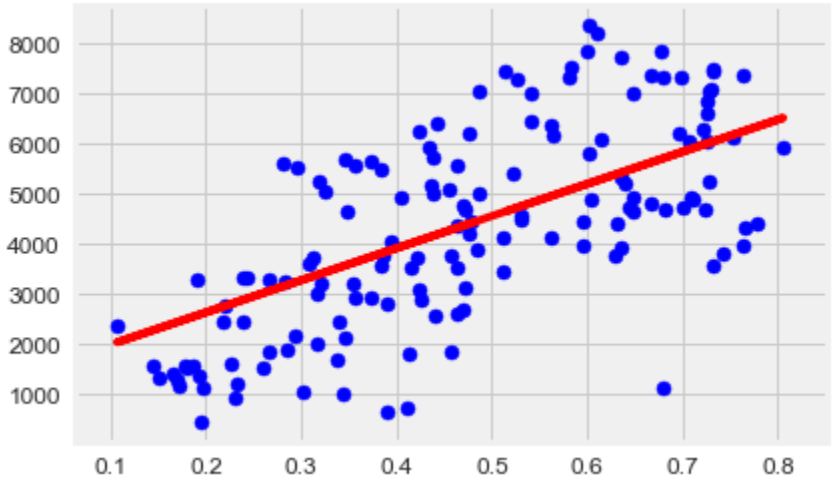


Fig. red line shows the best-fit regression line for the test data

RANdom SAmple Consensus (RANSAC) algorithm

It fits a regression model to a subset of the data, the so-called inliers. Linear regression models can be heavily impacted by the presence of outliers. Using RANSAC, we reduced the potential effect of the outliers in this dataset.

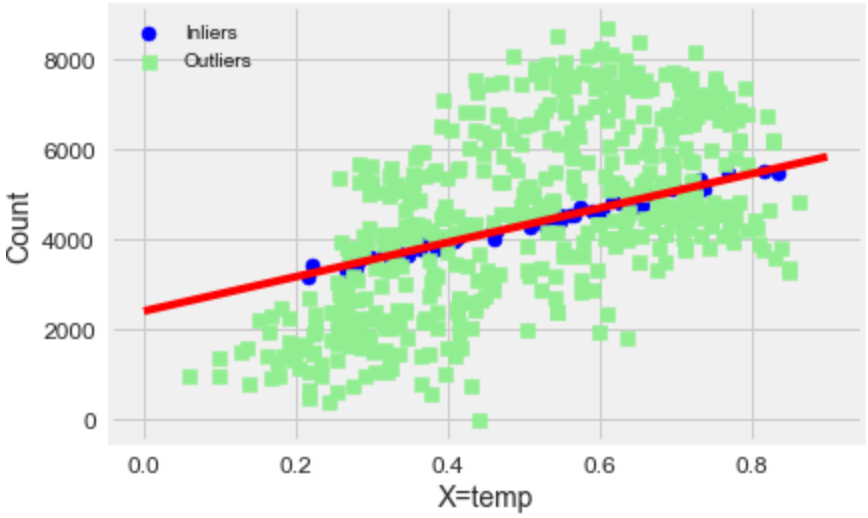


Fig red line showing best fit regression line using RANSAC (without outliers)

Simple Linear Regression using RANSAC	TRAIN	TEST
RMSE	1604.83	1613.702
R2 score	0.300	0.344

Observation: RMSE is high and R2 score is close to 0. Hence the model is not good for prediction. Any other model should have lower RMSE as compared to Base Model.

Linear Regression:

Linear regression is used for finding linear relationship between target and one or more predictors. There are two types of linear regression- Simple and Multiple.

Assumptions in the Linear Regression:

1. Linear Relationship between the features and target
2. Little or no Multicollinearity between the features
3. Homoscedasticity Assumption.
4. Normal distribution of residuals
5. Little or No autocorrelation in the residuals

Detecting Multicollinearity:

Multicollinearity can be detected by calculating VIF for each independent variable. VIFs start at 1 and have no upper limit.

VIF =1 => No correlation between the independent variables

1<VIF<5 => suggest that there is a moderate correlation, but it is not severe enough to warrant corrective measures.

VIF>5 => greater than 5 represent critical levels of multicollinearity where the coefficients are poorly estimated, and the p-values are questionable.

	features	VIF Factor
0	const	45.579683
1	season	3.759572
2	yr	1.034658
3	mnth	3.561089
4	weathersit	1.836585
5	temp	1.198144
6	hum	2.023005
7	windspeed	1.172051

Result of the Linear Regression:

Multiple Linear Regression	TRAIN	TEST
RMSE	875.115	945.411
R2 score	0.788	0.79

Interpreting the result:

Since our model uses multiple explanatory variables, we can't visualize the linear regression line (or hyperplane to be precise) in a two-dimensional plot, but we can plot the residuals (the differences or vertical distances between the actual and predicted values) versus the predicted values to diagnose our regression model.

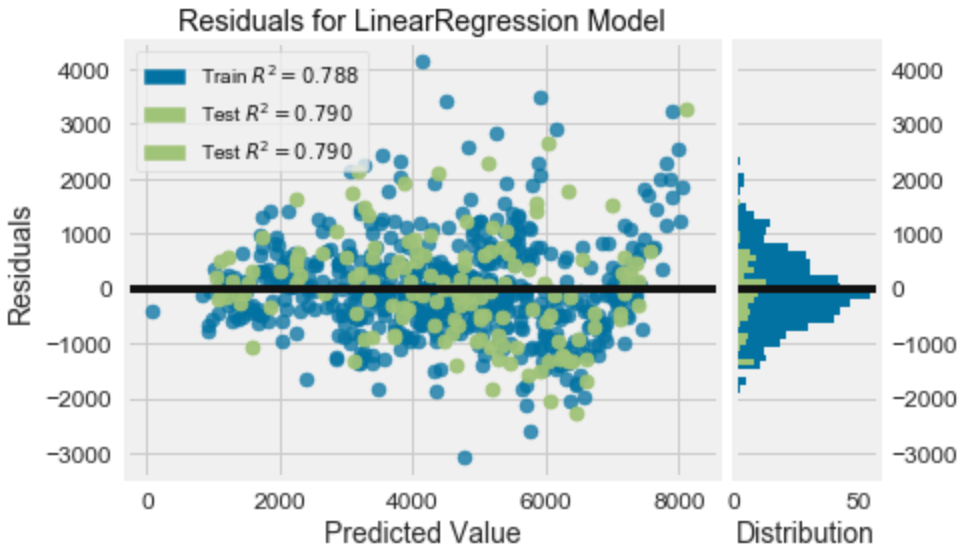


Fig showing the residual plot

For a good regression model, we would expect that the errors are randomly distributed and the residuals should be randomly scattered around the centerline. If we see patterns in a residual plot, it means that our model is unable to capture some explanatory information, which is leaked into the residuals. It is also one of the assumptions of the Linear Regression model, Homoscedasticity Assumption. There should be no clear pattern in the distribution and if there is a specific pattern, the data is heteroscedastic.

A random pattern of residuals supports a linear model; a non-random pattern supports a non-linear model. The sum of the residuals is always zero, whether the data set is linear or nonlinear.

Here, we see that the residual plot shows a fairly random pattern and the residuals are normally distributed.

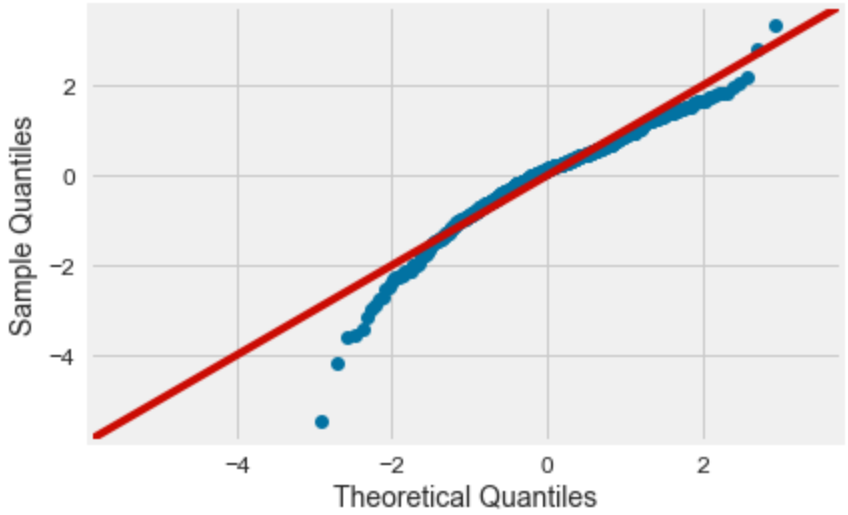


Fig showing Q-Q plot

The q-q plot of the bike rental data set shows that the errors(residuals) are fairly normally distributed. It satisfies the Linear Regression assumption.

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.964
Model:	OLS	Adj. R-squared:	0.964
Method:	Least Squares	F-statistic:	2208.
Date:	Sun, 21 Jul 2019	Prob (F-statistic):	0.00
Time:	11:04:08	Log-Likelihood:	-4818.0
No. Observations:	584	AIC:	9650.
Df Residuals:	577	BIC:	9681.
Df Model:	7		
Covariance Type:	nonrobust		

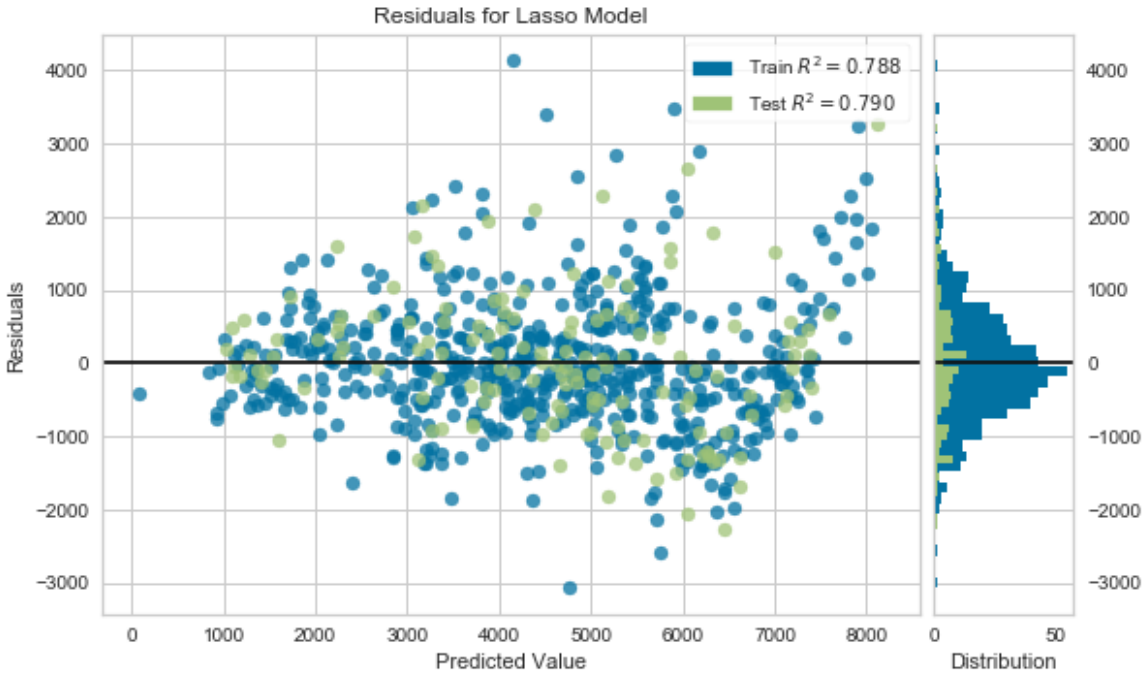
	coef	std err	t	P>|t|	[0.025	0.975]
season	522.5577	67.049	7.794	0.000	390.868	654.248
yr	2118.6115	76.072	27.850	0.000	1969.200	2268.023
mnth	-28.7347	21.040	-1.366	0.173	-70.059	12.590
weathersit	-576.7371	96.576	-5.972	0.000	-766.420	-387.054
temp	5777.7207	221.458	26.089	0.000	5342.758	6212.683
hum	483.5419	310.640	1.557	0.120	-126.581	1093.664
windspeed	-351.0421	425.583	-0.825	0.410	-1186.923	484.839

Omnibus:	101.074	Durbin-Watson:	1.954
Prob(Omnibus):	0.000	Jarque-Bera (JB):	236.235
Skew:	-0.911	Prob(JB):	5.04e-52
Kurtosis:	5.527	Cond. No.	89.5

Here value of Durbin-Watson test is 1.968 quite close to 2 as said before when the value of Durbin-Watson is equal to 2, r takes the value 0 from the equation 2*(1-r), which in turn tells us that the residuals are not correlated. This satisfies the last assumption of the Linear Regression I.e. Little or No autocorrelation in the residuals.

As the model is prone to overfitting (test RMSE is higher than train RMSE), we tried **LASSO regularization** to counter overfitting.

Lasso Regularization /Linear Regression	TRAIN	TEST
RMSE	875.115	945.284
R2 score	0.788	0.79



Error metrics is same as before. The linear regression method is great for datasets with lots of continuous data, but a lot of the columns in this dataset is not continuous, but rather categorical.

Also, the error metric RMSE is high in case of Linear Regression, we can use the decision tree to see if we can improve our predictions.

Decision Tree:

It is one of the most widely used and practical methods for supervised learning used for both classification and regression tasks. An advantage of the decision tree algorithm is that it does not require any transformation of the features if we are dealing with nonlinear data.

It uses MSE as Impurity measure for Regression.

In the context of decision tree regression, the MSE is often also referred to as within-node variance, which is why the splitting criterion is also better known as variance reduction.

DT is prone to overfitting which can be taken care by pruning the tree.

Result:

Decision Tree (max_depth=3)	TRAIN	TEST
RMSE	885.153	1029.278
R2 score	0.784	0.751

Decision Tree (max_depth=5)	TRAIN	TEST
RMSE	650.392	902.199
R2 score	0.883	0.809

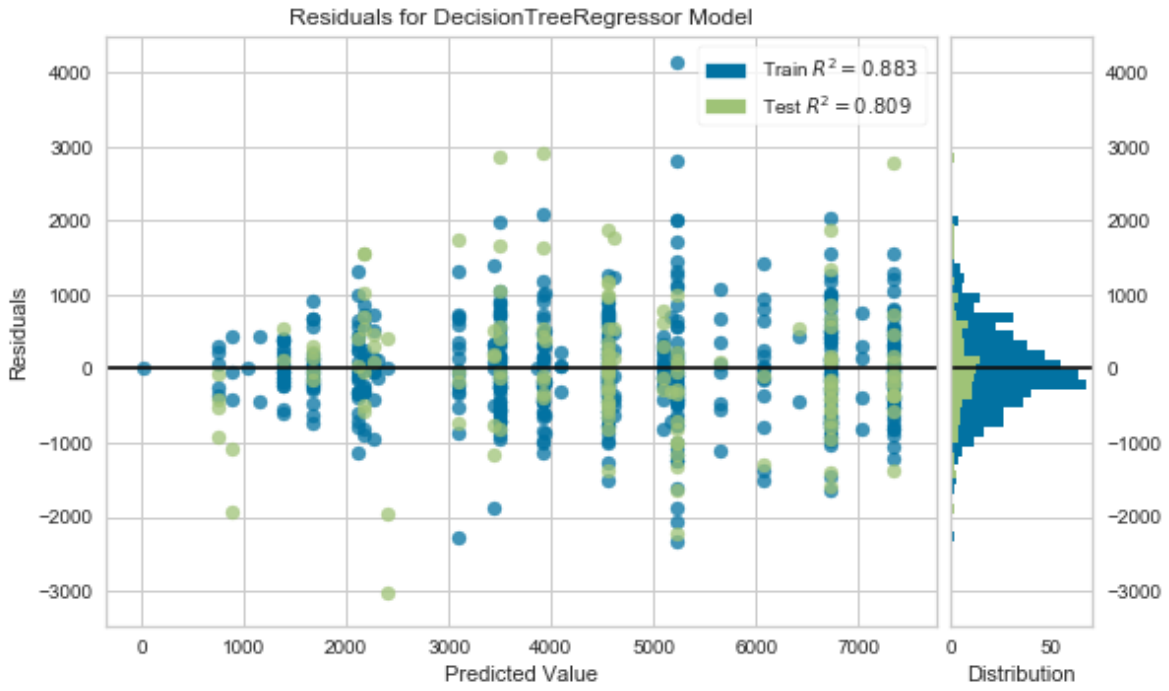


Fig showing residual plot above

We see the points are randomly dispersed around the horizontal axis, a linear regression model is usually appropriate for the data. We can also see from the histogram that our error is normally distributed around zero, which also generally indicates a well fitted model.

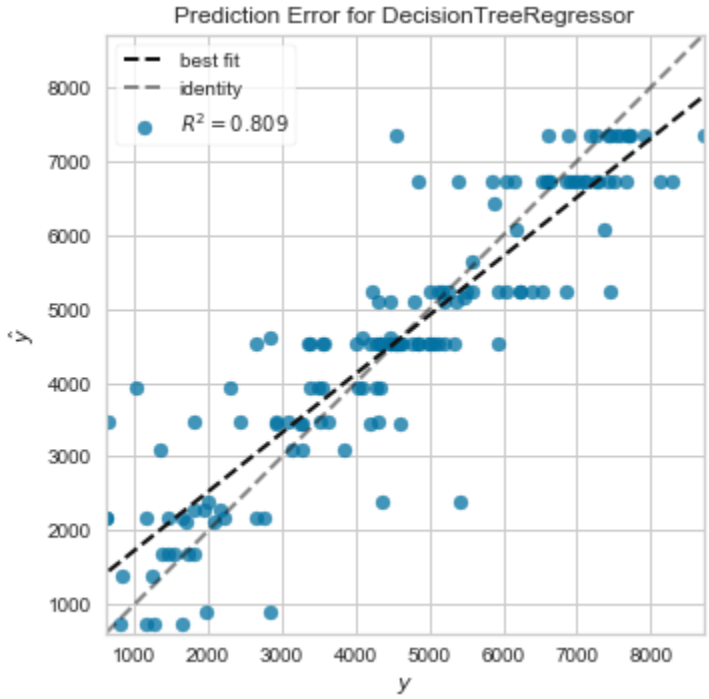


Fig. Prediction Error plot for Decision Tree

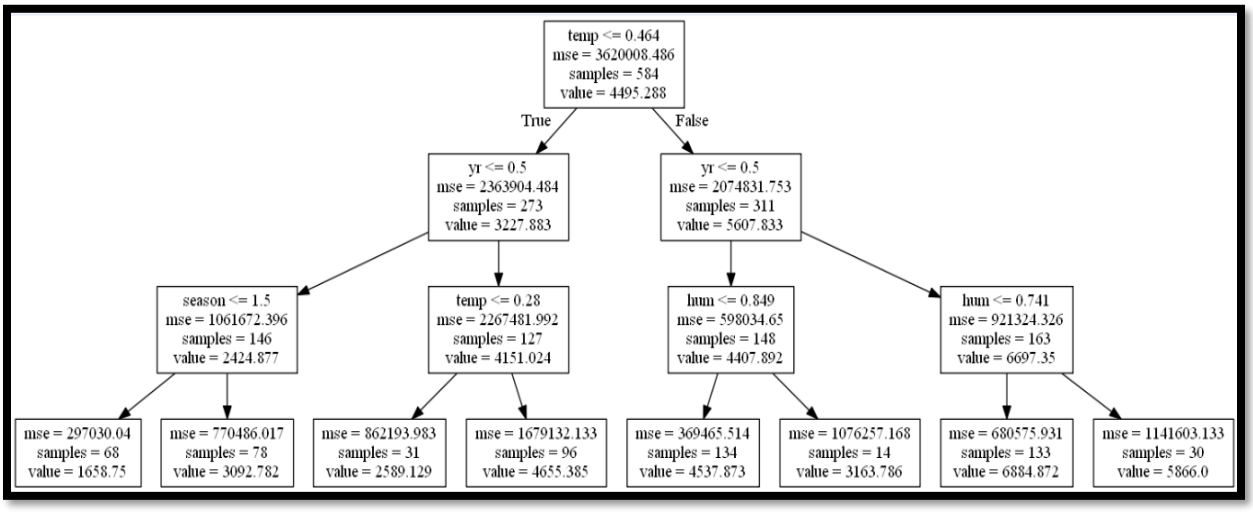


Fig. showing decision tree plot for max_depth=3

Interpretation of the result:

RMSE is lower as compared to Linear Regression but the test RMSE is still higher as compared from the train dataset. As the max_depth parameter increases, overfitting also increases i.e. the model predicts poorly on the unseen data.

Temp is the most important split.

Hence, this model is prone to overfitting. Let's explore random forest model as it handles overfitting better.

Random Forest:

Random forest is one of the popular machine learning algorithm used for both classification and regression problems.

- ensemble of DTs.
- combine weak learners to build a strong learner.
- less prone to overfitting.

Steps in RF:

1. Draw a random bootstrap sample of size n (randomly choose 'n' samples from the training set with replacement).
2. Grow a decision tree from the bootstrap sample. At each node:
    1. Randomly select 'd' features without replacement.
    2. Split the node using the feature that provides the best split according to the objective function, for instance minimizing the variance in regression.
3. Repeat the steps 1 to 2 'k' times.
4. Aggregate the prediction by each tree using averaging to improve the predictive accuracy and control over-fitting.

Result:

Random Forest	TRAIN	TEST
RMSE	258.620	679.580
R2 score	0.982	0.891

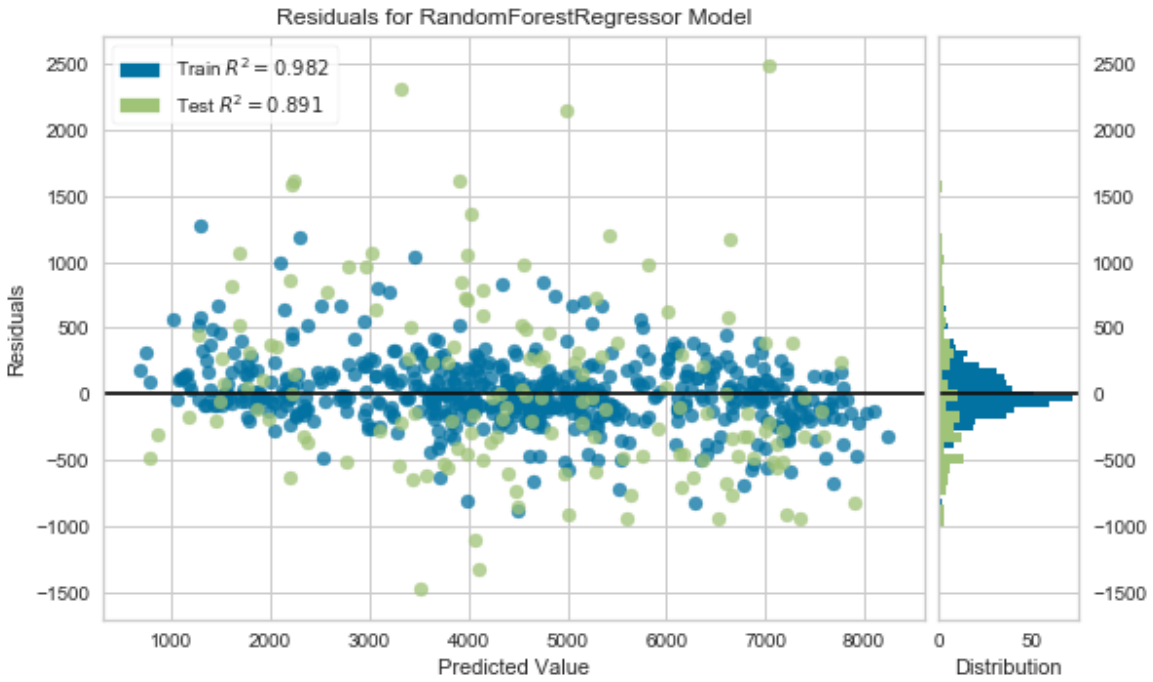


Fig. Residual plot for Random Forest

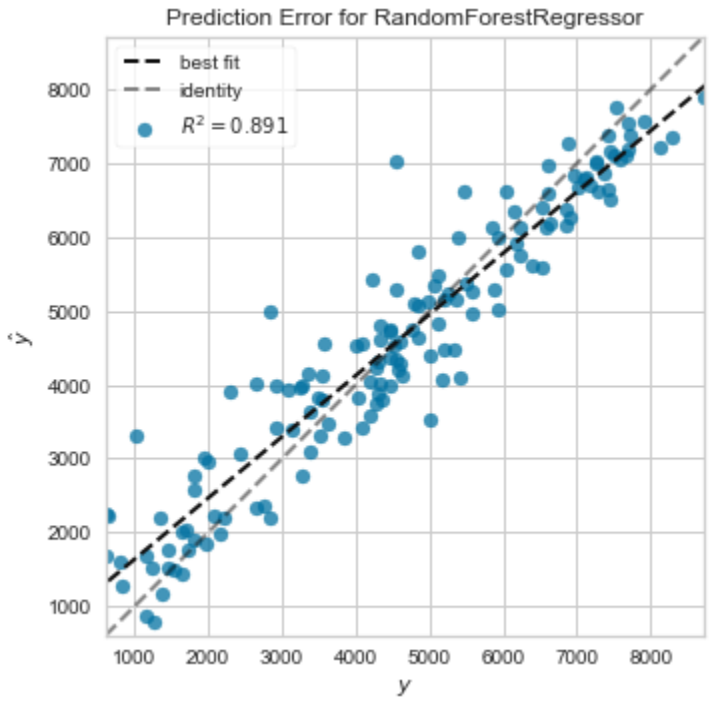


Fig. Prediction Error plot for Random Forest

RF using Hyperparameter tuning

Hyperparameter tuning using GridSearchCV is performed to get a set of optimal hyperparameters.

```
Fitting 5 folds for each of 480 candidates, totalling 2400 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   13.3s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:   54.5s
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done 640 tasks      | elapsed:  4.0min
[Parallel(n_jobs=-1)]: Done 1005 tasks      | elapsed:  6.6min
[Parallel(n_jobs=-1)]: Done 1450 tasks      | elapsed:  9.5min
[Parallel(n_jobs=-1)]: Done 1977 tasks      | elapsed: 13.5min
[Parallel(n_jobs=-1)]: Done 2400 out of 2400 | elapsed: 18.7min finished

{'max_depth': None, 'max_features': 'sqrt', 'n_estimators': 850}
```

Post Hyperparameter tuning, we get the error metric as below:

Random Forest (Hyperparameter tuning)	TRAIN	TEST
RMSE	249.778	697.249
R2 score	0.983	0.886

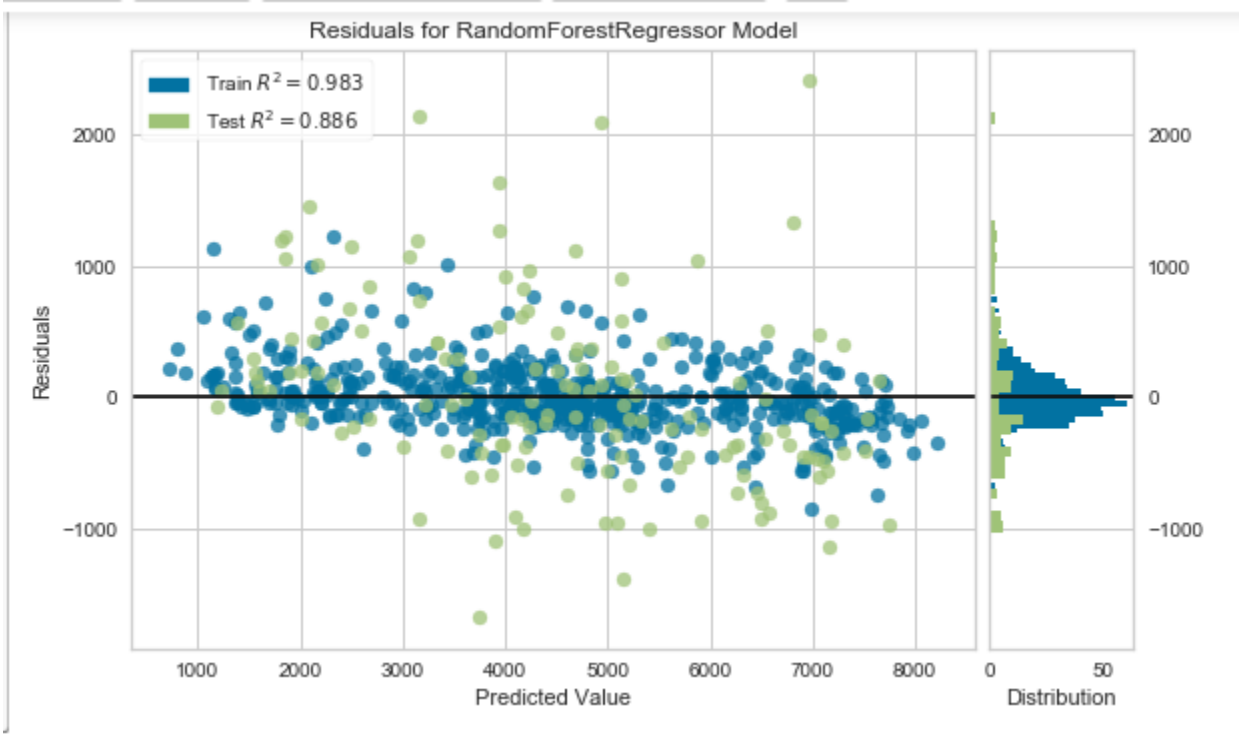


Fig. Residual plot for Random Forest

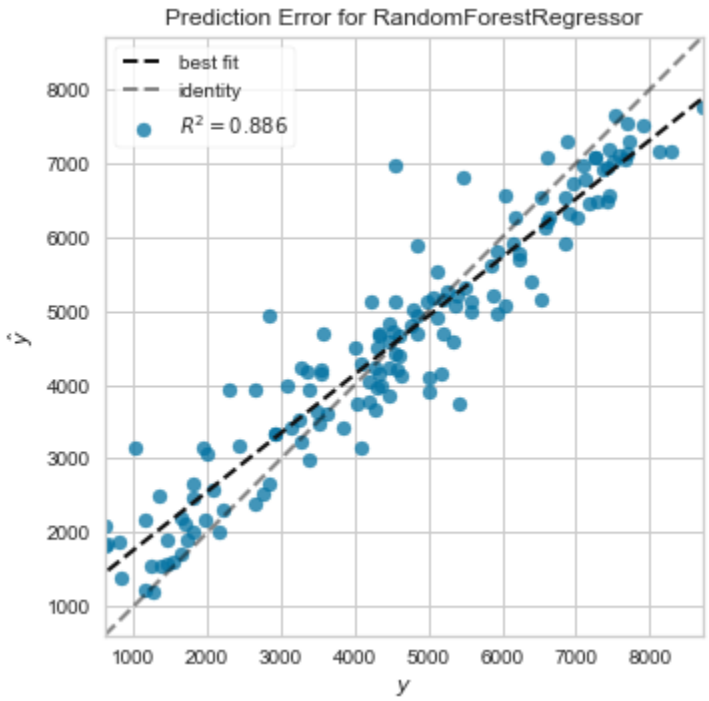


Fig. Prediction Error plot for Random Forest (after hyperparameter tuning)

XGBoost:

XGBoost (Extreme Gradient Boosting) belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework at its core.

But what makes XGBoost so popular?

- **Speed and performance**: Originally written in C++, it is comparatively faster than other ensemble classifiers.
- **Core algorithm is parallelizable**: Because the core XGBoost algorithm is parallelizable it can harness the power of multi-core computers. It is also parallelizable onto GPU's and across networks of computers making it feasible to train on very large datasets as well.
- **Consistently outperforms other algorithm methods**: It has shown better performance on a variety of machine learning benchmark datasets.

- **Wide variety of tuning parameters**: XGBoost internally has parameters for cross-validation, regularization, user-defined objective functions, missing values, tree parameters, scikit-learn compatible API etc.

Boosting is a sequential technique which works on the principle of an ensemble.

It combines a set of weak learners and delivers improved prediction accuracy.

At any instant t, the model outcomes are weighed based on the outcomes of previous instant t-1. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher. Note that a weak learner is one which is slightly better than random guessing.

Results:

XGBoost using hyperparameter tuning snapshot:

```
{'alpha': [10],
 'colsample_bytree': [0.3, 0.5, 0.7, 0.9],
 'learning_rate': [0.01, 0.05, 0.1, 0.2, 0.5],
 'max_depth': [1, 2, 3, 4, 5],
 'n_estimators': [50,
                  100,
                  150,
                  200,
                  250,
                  300,
                  350,
                  400,
                  450,
                  500,
                  550,
                  600,
                  650,
                  700,
                  750,
                  800,
                  850,
                  900,
                  950,
                  1000]}
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0,
       importance_type='gain', learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=None, subsample=1, verbosity=1)
Fitting 3 folds for each of 2000 candidates, totalling 6000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   12.7s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:   20.7s
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:   32.3s
[Parallel(n_jobs=-1)]: Done 905 tasks      | elapsed:   58.3s
[Parallel(n_jobs=-1)]: Done 1635 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 2273 tasks      | elapsed:  2.2min
[Parallel(n_jobs=-1)]: Done 3317 tasks      | elapsed:  3.2min
[Parallel(n_jobs=-1)]: Done 3924 tasks      | elapsed:  3.9min
[Parallel(n_jobs=-1)]: Done 4870 tasks      | elapsed:  4.9min
[Parallel(n_jobs=-1)]: Done 5797 tasks      | elapsed:  5.9min
[Parallel(n_jobs=-1)]: Done 5993 out of 6000 | elapsed:  6.1min remaining:    0.3s
[Parallel(n_jobs=-1)]: Done 6000 out of 6000 | elapsed:  6.2min finished

[00:08:11] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objective/regression_obj.cu:152: reg:linear i
s now deprecated in favor of reg:squarederror.
{'alpha': 10, 'colsample_bytree': 0.7, 'learning_rate': 0.05, 'max_depth': 4, 'n_estimators': 150}
```

XGBoost	TRAIN	TEST
RMSE	365.651	665.499
R2 score	0.961	0.896

Interpretation of the results:

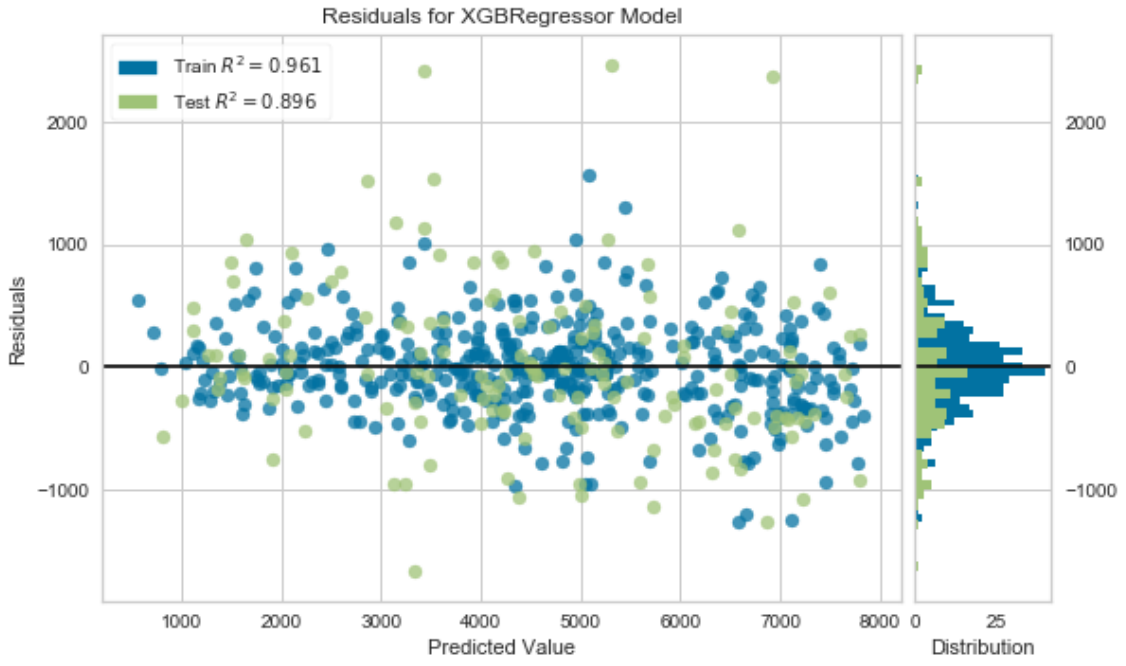


Fig showing residual plot

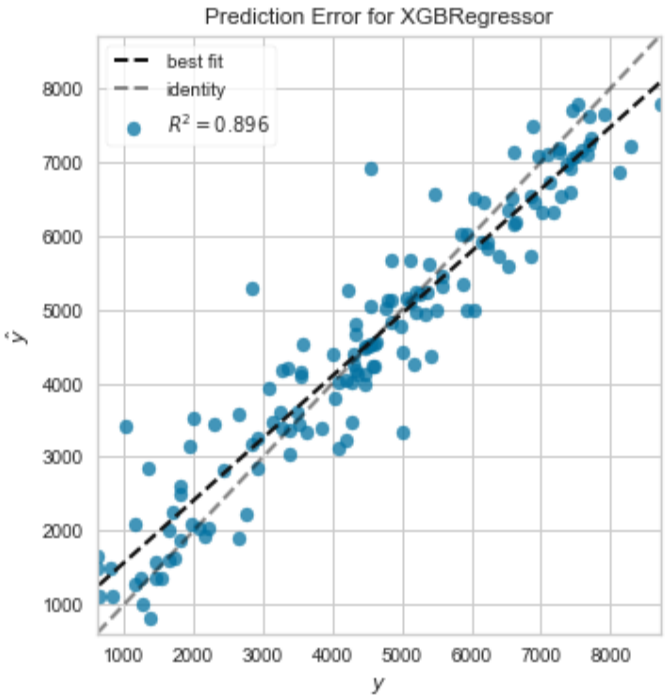


Fig showing prediction error plot

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case, *Interpretability* and *Computation Efficiency*, do not hold much significance. Therefore, we will use *Predictive performance* as the criteria to compare and evaluate models.

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

Error metrics chosen for model evaluation:

We have used **RMSE and R Square (R^2)** error metrics for our model.

RMSE: unit is same as target. It's hard to realize if our model is good or not by just looking at the absolute values of RMSE. We would probably want to measure how much our model is better than the constant baseline. This is given by the R square metric.

MSE:

The `mean_squared_error` function computes mean square error, a risk metric corresponding to the expected value of the squared (quadratic) error or loss.

If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the mean squared error (MSE) estimated over n_{samples} is defined as

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

RMSE is simply the square root of MSE.

R^2 score, the coefficient of determination

It suggests the proportion of variation in Y which can be explained with the independent variables.

The `r2_score` function computes the coefficient of determination, usually denoted as R^2.

It represents the proportion of variance (of y) that has been explained by the independent variables in the model. It provides an indication of goodness of fit and therefore a measure of how well unseen samples are likely to be predicted by the model, through the proportion of explained variance.

As such variance is dataset dependent, R^2 may not be meaningfully comparable across different datasets. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.

If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value for total n samples, the estimated R^2 is defined as:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$.

Note that `r2_score` calculates unadjusted R^2 without correcting for bias in sample variance of y.

When $R^2 < 0$ i.e. negative. it means that the model is worse than predicting the mean.
A value close to 1 indicates a model with close to zero error,
and a value close to zero indicates a model very close to the baseline.

Regression visualizers:

1. **Prediction error plot** shows the actual targets from the dataset against the predicted values generated by our model. This allows us to see how much variance is in the model. Data scientists can diagnose regression models using this plot by comparing against the 45-degree line, where the prediction exactly matches the model.
2. **Residual plot:** for a good regression model, we would expect that the errors are randomly distributed, and the residuals should be randomly scattered around the centerline. If we see patterns in a residual plot, it means that our model is unable to capture some explanatory information, which is leaked into the residuals. It is also one of the assumptions of the Linear Regression model, Homoscedasticity Assumption. There should be no clear pattern in the distribution.

3.2 Model Selection

XGBoost using hyperparameter tuning yields the lowest Test RMSE as compared to all the models and hence the model is chosen. Hence, it will predict better for the unseen data. Though all the models suffer from the varying degree of overfitting problem.

Error metric in python:

Simple Linear Regression (Base Model)	TRAIN	TEST
RMSE	1513.809	1484.061
R2 score	0.377	0.445

Multiple Linear Regression	TRAIN	TEST
RMSE	875.115	945.411
R2 score	0.788	0.79

Multiple Linear Regression using Lasso Regularization	TRAIN	TEST
RMSE	875.115	945.284
R2 score	0.788	0.79

Decision Tree	TRAIN	TEST
RMSE	650.392	902.199
R2 score	0.883	0.809

Random Forest	TRAIN	TEST
RMSE	257.083	682.023
R2 score	0.982	0.891

Random Forest (Hyperparameter tuning)	TRAIN	TEST
RMSE	249.778	697.249
R2 score	0.983	0.886

XGBoost	TRAIN	TEST
RMSE	171.56	729.637
R2 score	0.991	0.875

XGBoost (Hyperparameter Tuning)	TRAIN	TEST
RMSE	365.651	665.499
R2 score	0.961	0.896

Error metric in R

Multiple Linear Regression	TRAIN	TEST
RMSE	904.918	948.4809
R2 score	0.798272	0.7855706

Decision Tree	TRAIN	TEST
RMSE	860.7013	924.24
R2 score	0.805097	0.761857

Random Forest	TRAIN	TEST
RMSE	377.8026	674.4018
R2 score	0.967435	0.8732322

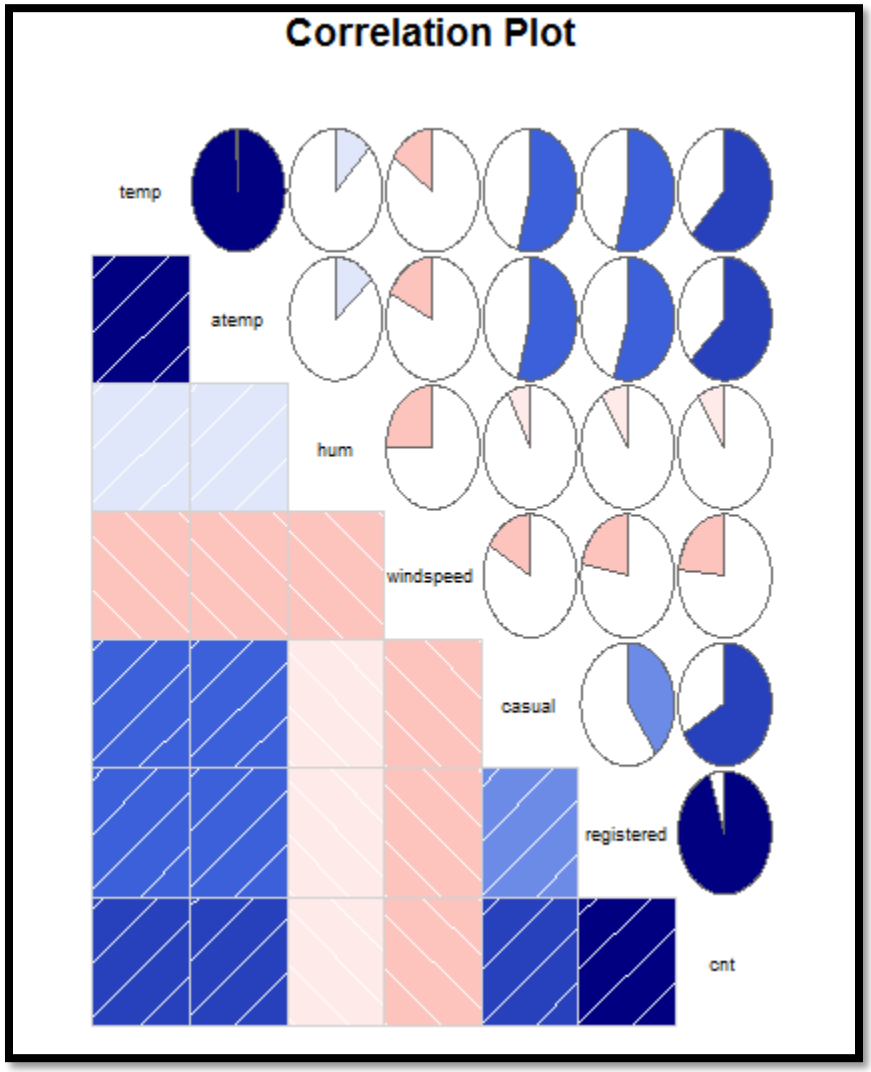
Random Forest (Hyperparameter tuning)	TRAIN	TEST
RMSE	249.778	697.249
R2 score	0.983	0.886

XGBoost	TRAIN	TEST
RMSE	228.99	707.343
R2 score	0.986	0.8613

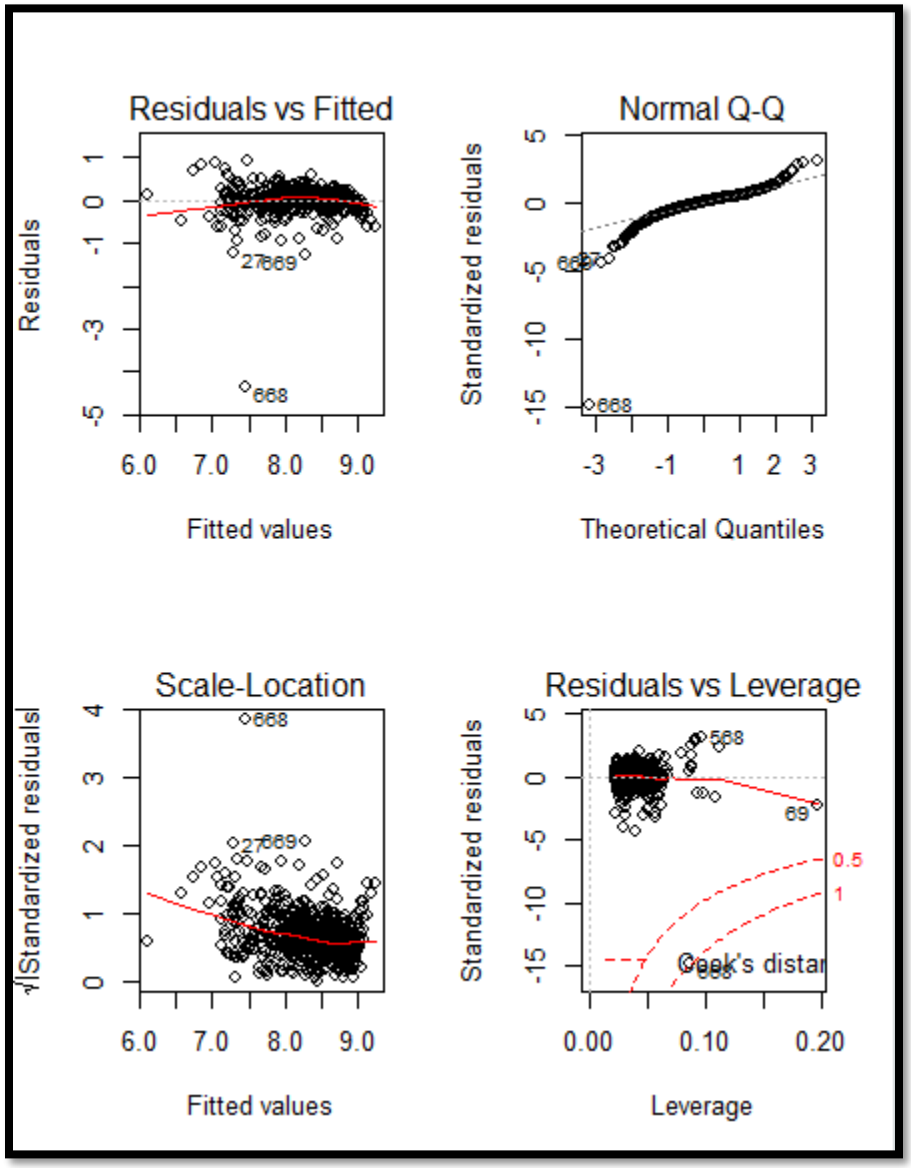
XGBoost (Hyperparameter tuning)	TRAIN	TEST
RMSE	489.3982	615.2403
R2 score	0.937344	0.8934894

Additional diagrams:

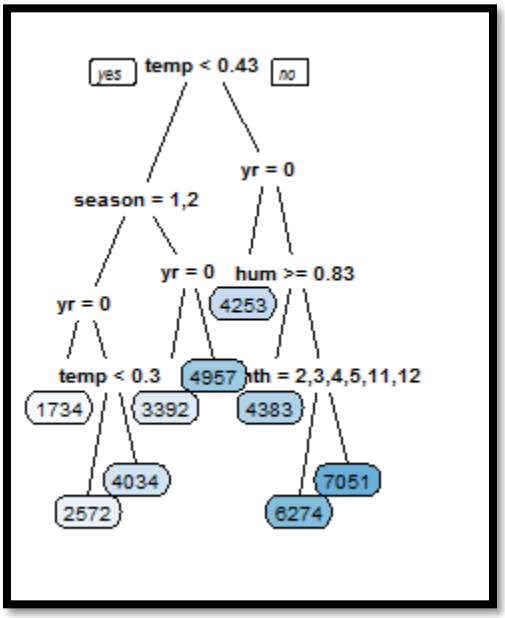
Correlation matrix in R



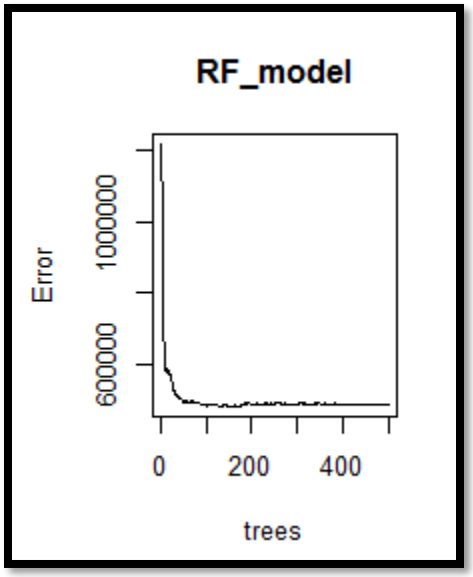
Linear Regression plot in R



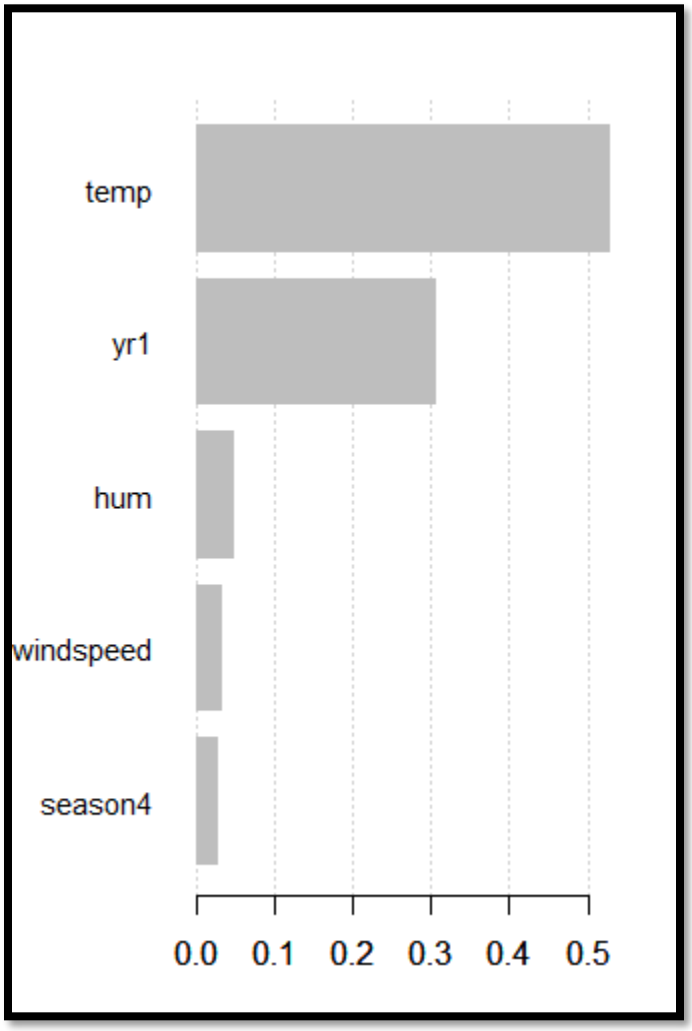
Decision tree in R



Random Forest plot in R



Feature Importance in XGBoost in R



References:

https://lightgbm.readthedocs.io/en/latest/Parameters.html

https://xgboost.readthedocs.io/en/latest/python/python_api.html

https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics