Bike Rental Count Prediction

By

Trayan Das

02/08/2020

Table of Contents

Chapter 1

Introduction

1.1. Problem Statement

This business problem belongs to a bike ride-hailing company. Predicting the no. of ride rentals on a day is very important for any ride-hailing company, because it helps the company to keep a balance between no. of rides (in this case, bikes) employed on a day & no. of rides required on that day (supply-demand ratio); which in turn helps the company to employ right business strategies. My objective of this project is to accurately predict the count of bike rentals on a specific day based on certain environmental and seasonal settings.

1.2. Data

There are a total of 731 observations & 16 variables in the dataset. A sample of the dataset is given below.

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

Figure 1. Dataset Sample

The observations denote different days; and the variables represent different environmental and seasonal settings as described below-

1) instant: Record index
2) dteday: Date
3) season: Season (1:springer, 2:summer, 3:fall, 4:winter)
4) yr: Year (0: 2011, 1:2012)
5) mnth: Month (1 to 12)
6) holiday: weather day is holiday or not (extracted from Holiday Schedule)
7) weekday: Day of the week
8) workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
9) weathersit: (extracted from Freemeteo)
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
10) 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)
11) atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt_
min), t_min=-16, t_max=+50 (only in hourly scale)
12) hum: Normalized humidity. The values are divided to 100 (max)
13) windspeed: Normalized wind speed. The values are divided to 67 (max)
14) casual: count of casual users
15) registered: count of registered users
16) cnt: count of total rental bikes including both casual and registered

1.3. Softwares Used

1. Python 3.7.1 for 64 bit.
2. R 3.6.3 for 64 bit.

Chapter 2

Methodology for Python

2.1. Data Pre- processing

All the required packages are loaded. After setting the working directory, the given ‘day’ dataset in CSV format is loaded into the ‘df_day’ object. I can see that out of 16 variables 'instant','dteday' variables simply represent the ‘record index’ & ‘date’; hence are statistically insignificant. So, I dropped these 2 variables. Also, I can see that even though 'season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday' & 'weathersit' are in ‘int64’ format, they’re actually categorical variables consisting of 2 or more categories. So, I converted these variables to ‘category’ format.

2.2. Univariate Analysis

I plotted the target variable ‘cnt’ using the ‘distplot’ function in ‘seaborn’ package to check normality of this variable.

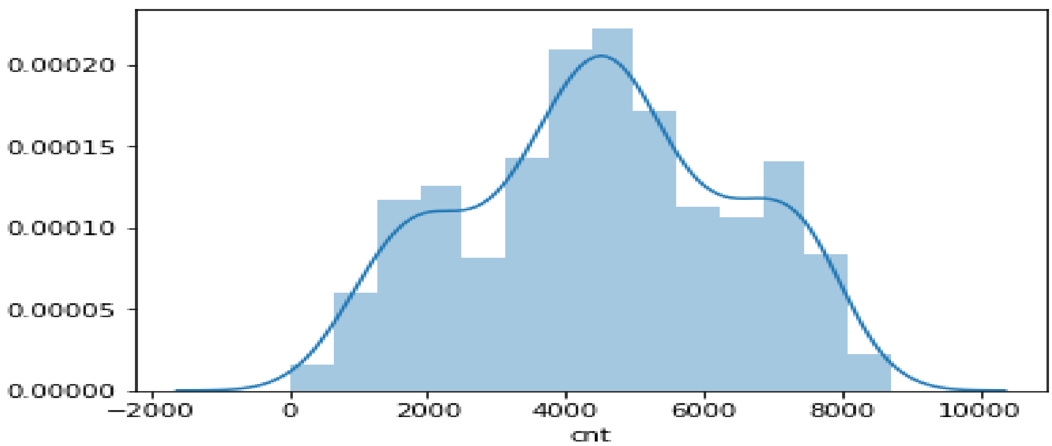


Figure 2. Distribution of target Variable

Then I plotted all 6 independent numeric variables using the same function in order to check normality of them.

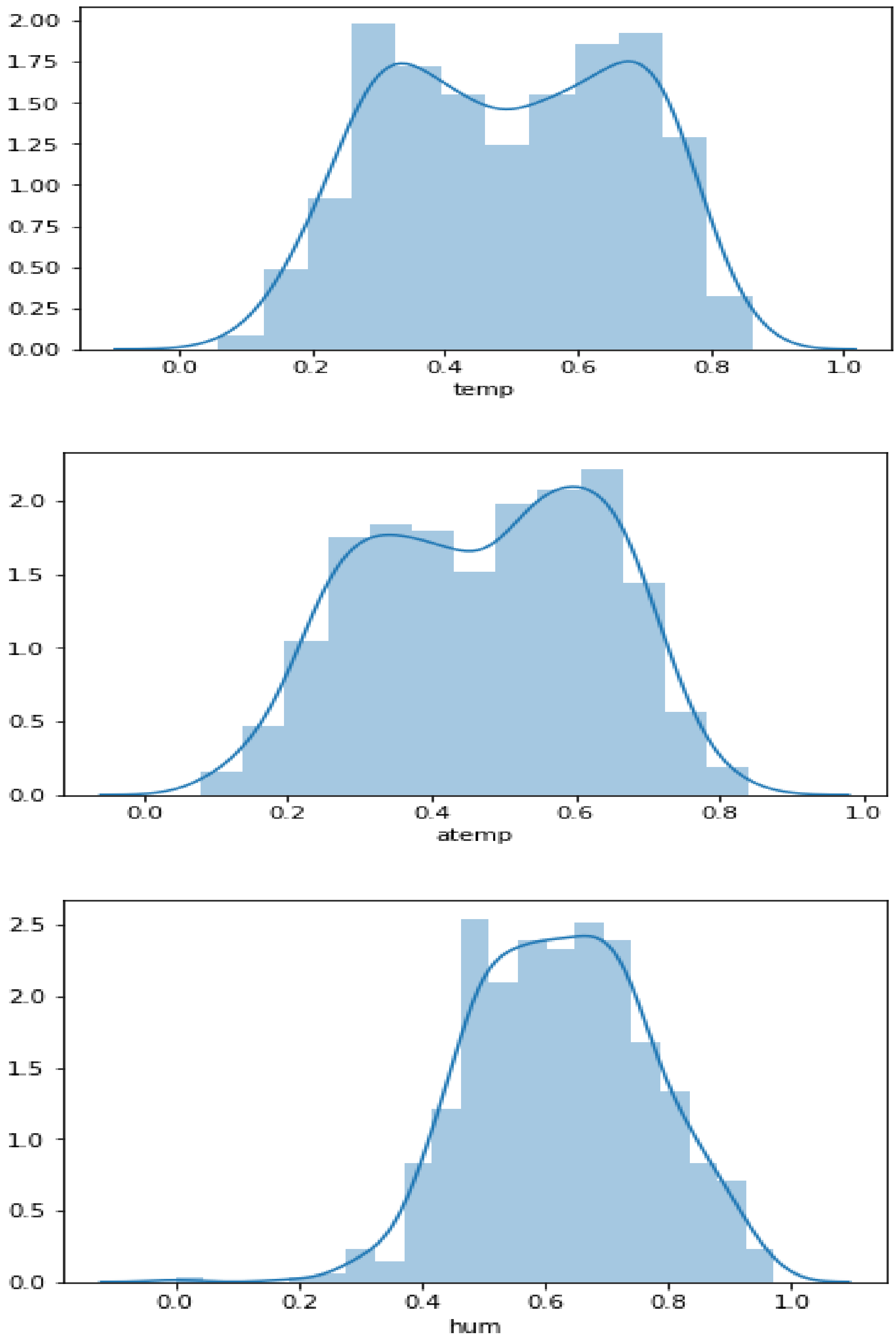


Figure 3A. Distribution of 'temp', 'atemp' & 'hum' independent variables

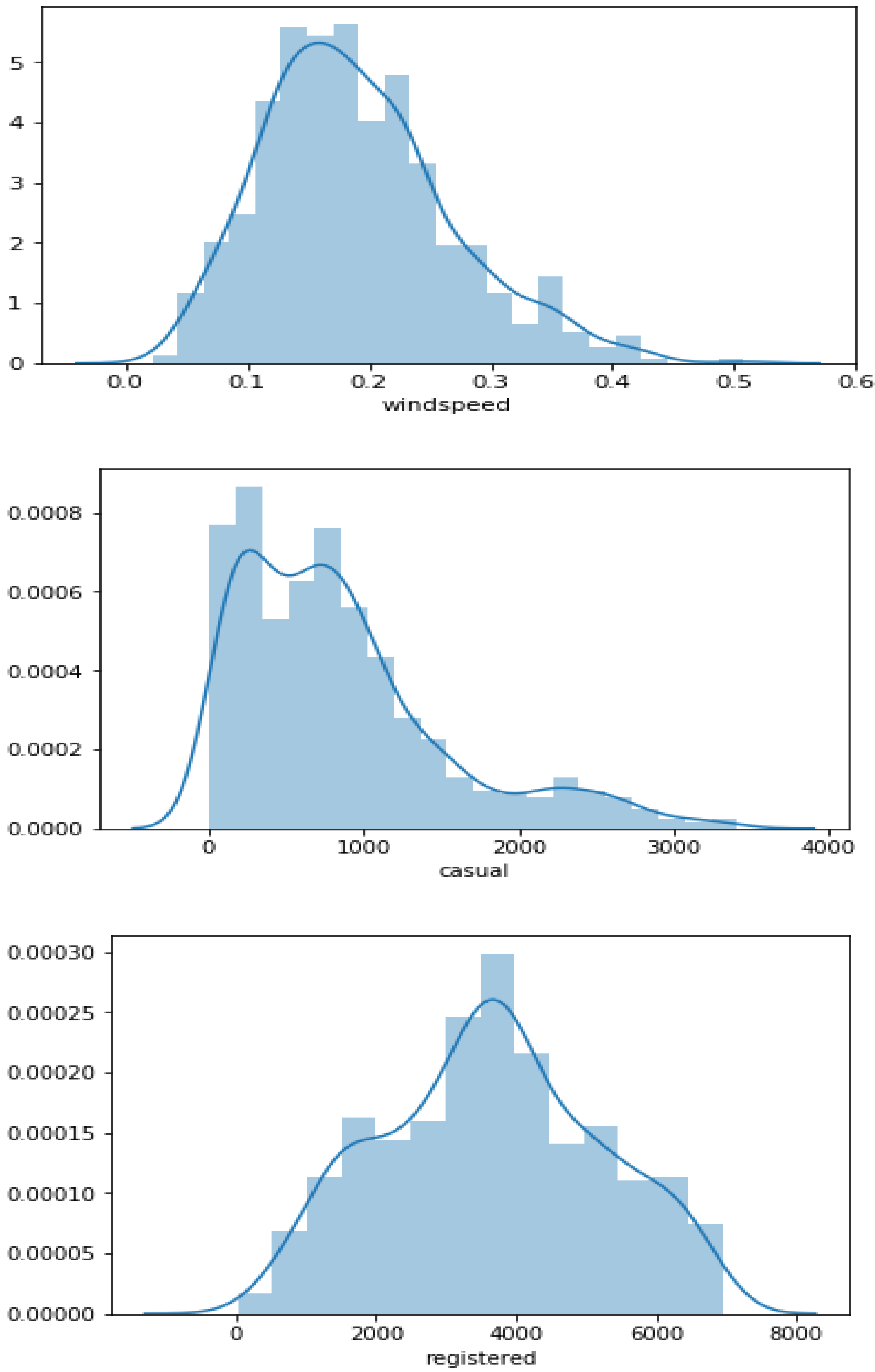


Figure 3B. Distribution of ‘windspeed, ‘casual & ‘registered independent variables

2.3. Bivariate Analysis

Next, I derived the distinct value counts for all 6 independent numeric variables & respectively plotted them using scatter plot in the x-axis with the target variable ‘cnt’ in the y-axis. Thus, I got an idea of the relation between different independent numeric variables & target variables. The generated scatter plots are-

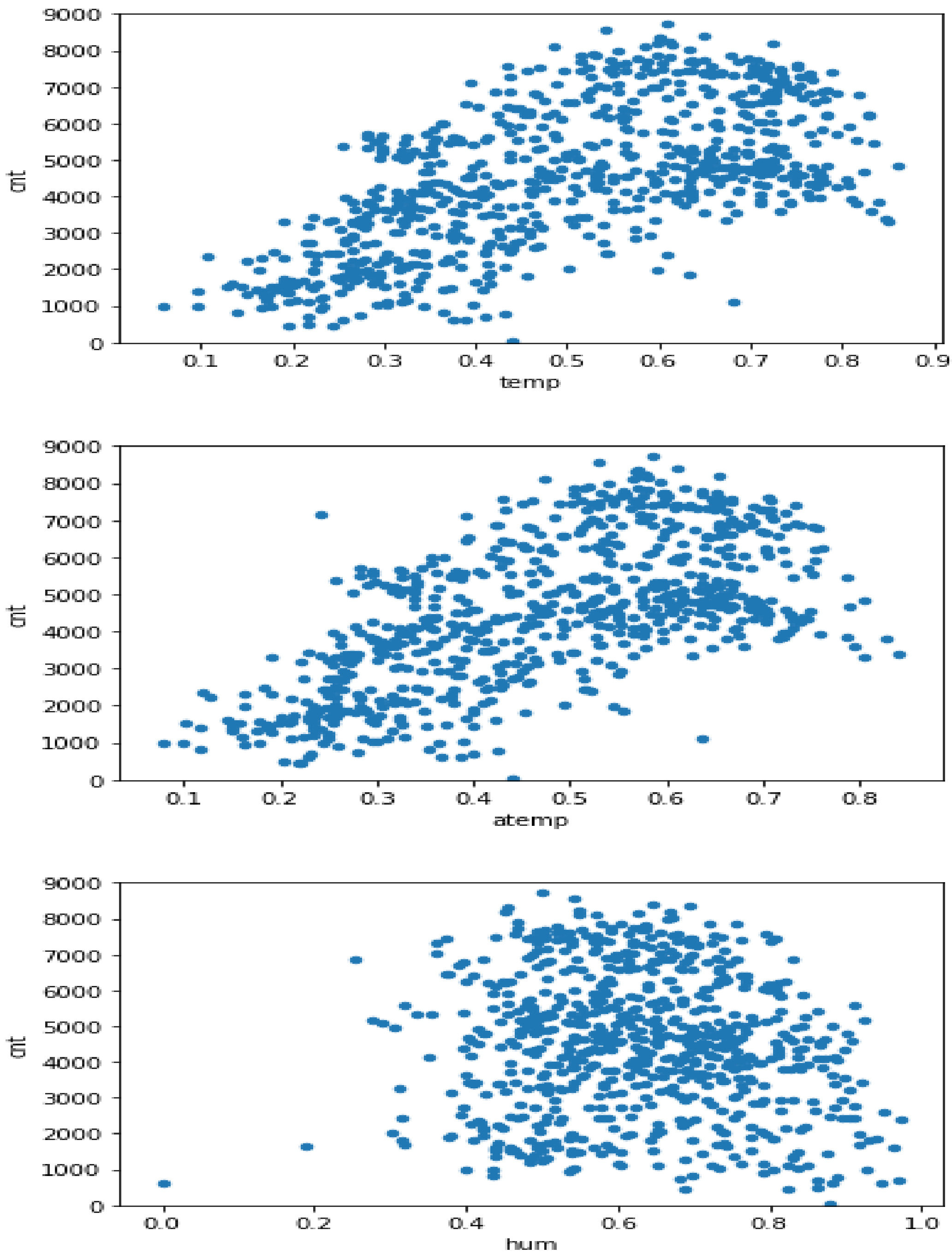


Figure 4A. Relation between ‘temp’, ‘atemp’ & ‘hum’ continuous independent variables & target variable

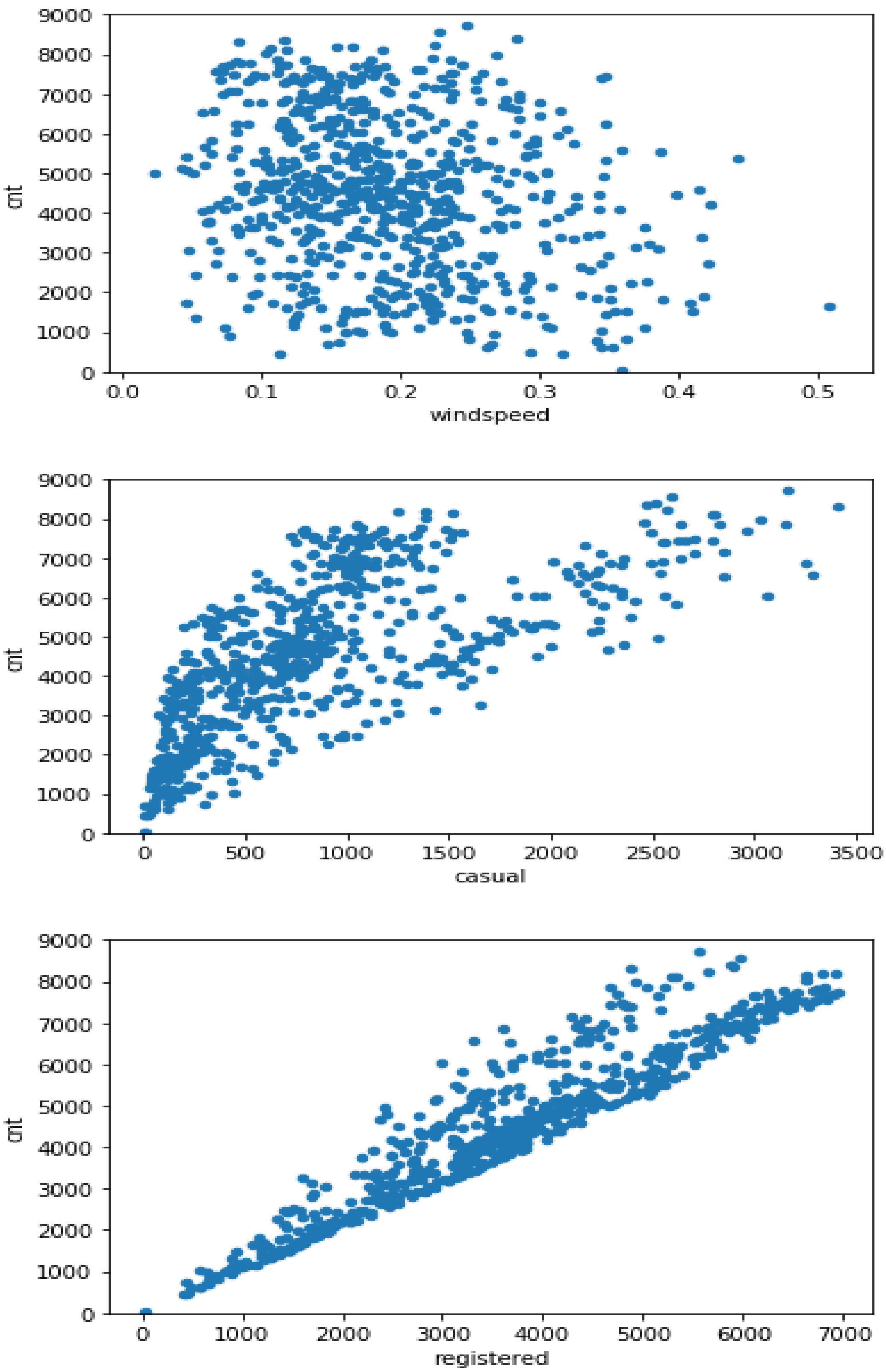
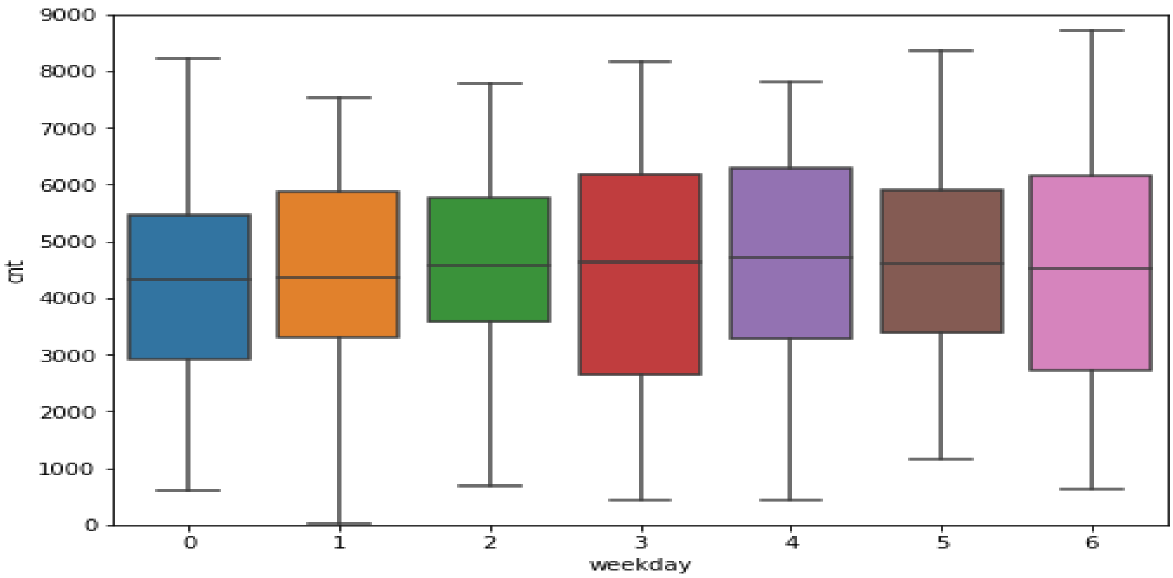


Figure 4B. Relation between 'windspeed, 'casual' & 'registered' continuous independent variables & target variable

The insights that I can obtain from these scatter plots are-

(i) there is good positive relation between 'temp' and 'cnt'
(ii) there is good positive relation between 'atemp' and 'cnt'
(iii) there is poor relation between 'hum' and 'cnt'
(iv) there is negative relation between 'windspeed' and 'cnt'
(v) there is somewhat good positive relation between 'casual' and 'cnt'
(vi) there is good positive relation between 'registered' and 'cnt'

After that, I generated boxplots of the target variable for each of the categories in all the categorical variables. These are the generated boxplots-



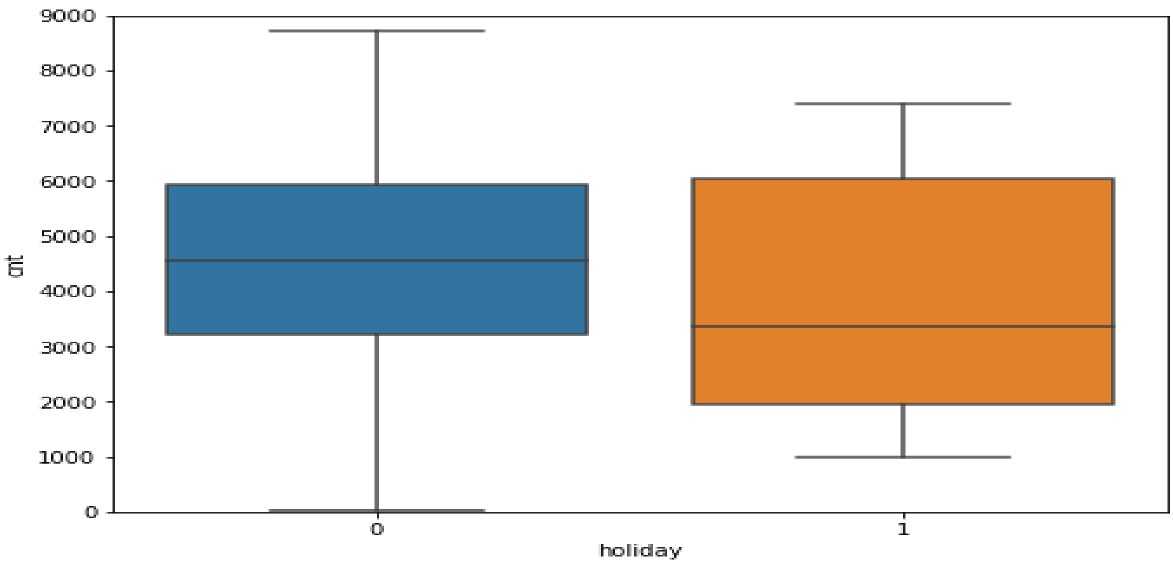


Figure 5A. Relation between ‘weekday’, ‘holiday’ categorical independent variables & target variable

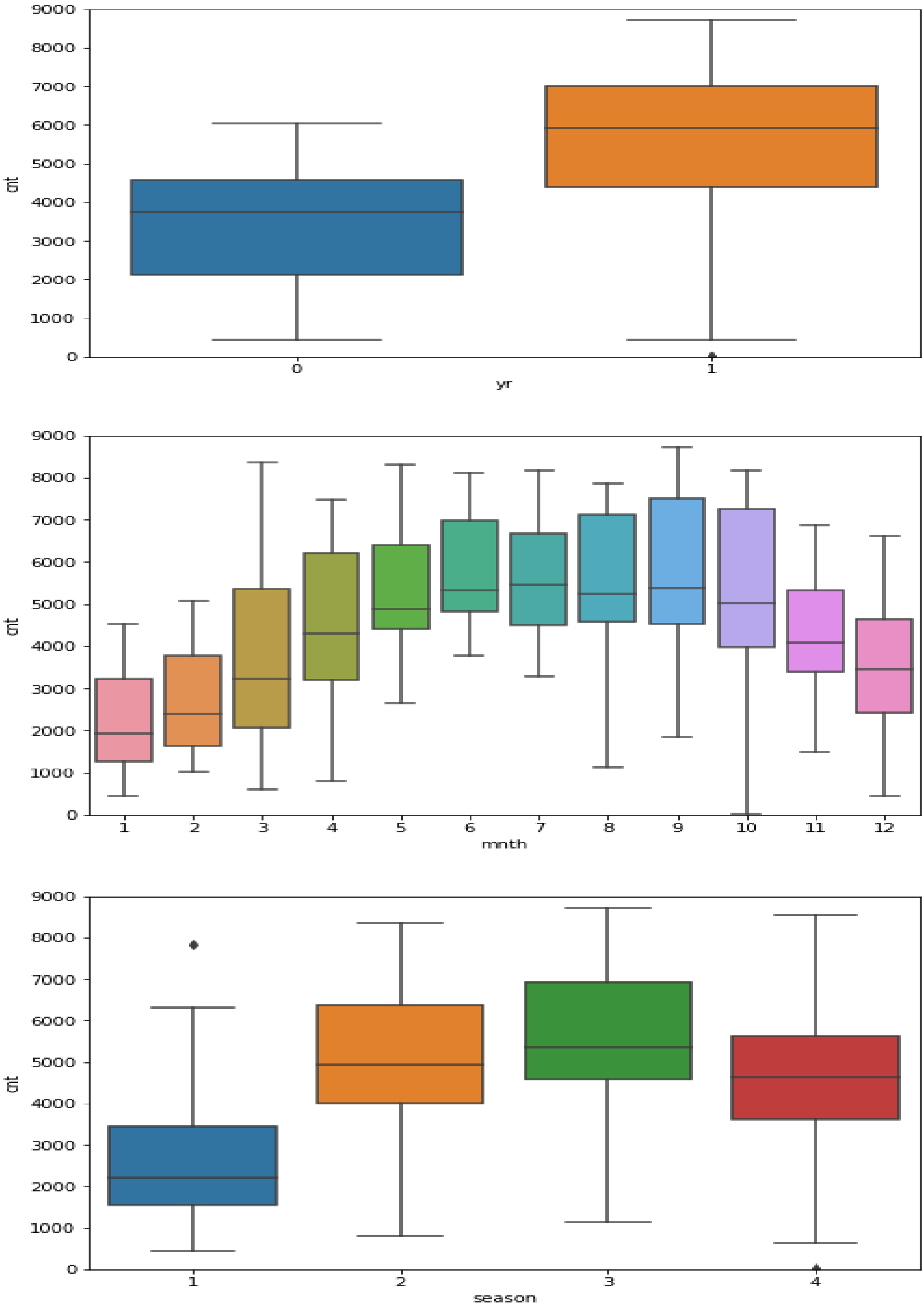


Figure 5B. Relation between ‘yr’, ‘mnth’, ‘season’ categorical independent variables & target variable

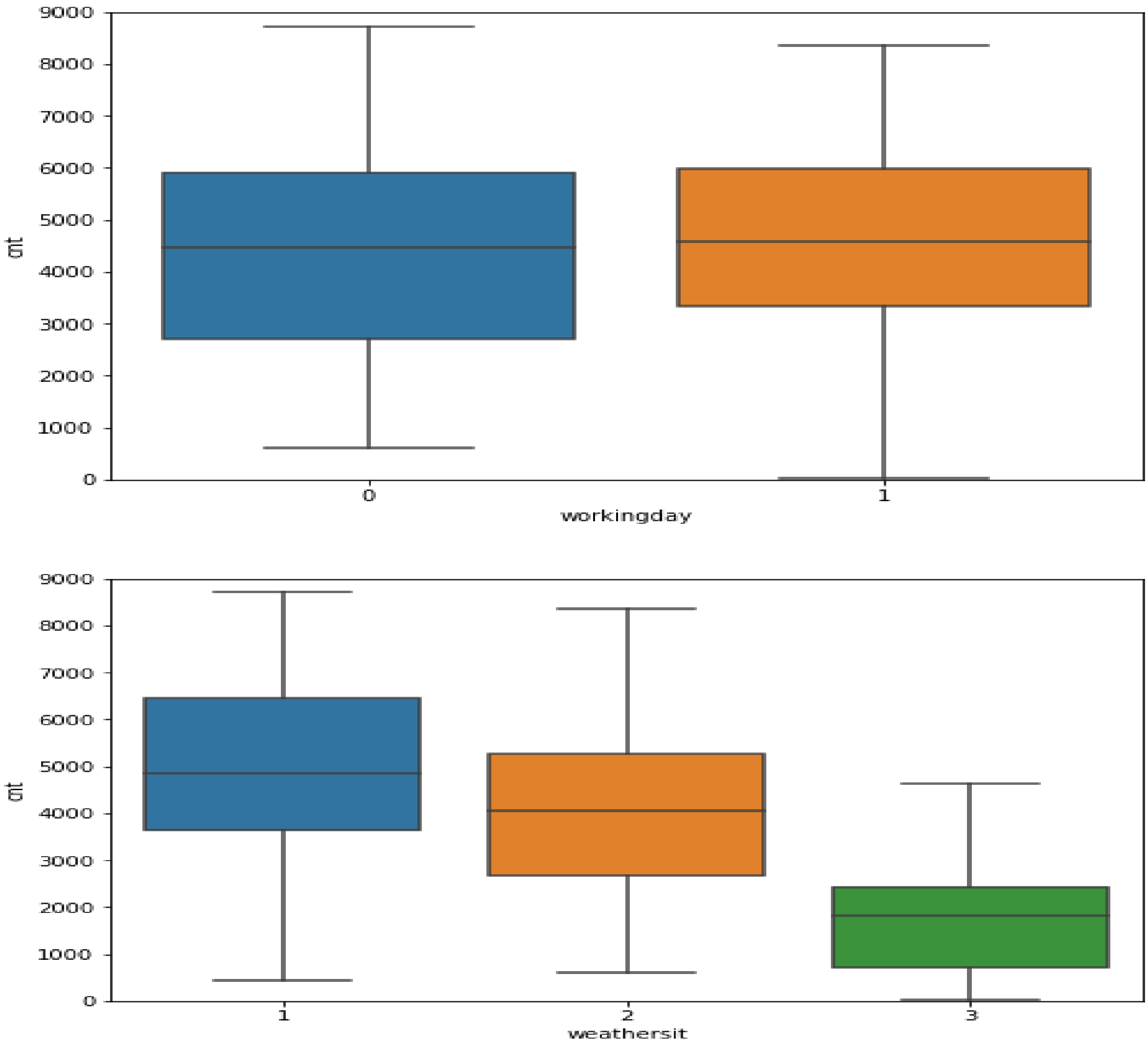


Figure 5C. Relation between 'workingday', 'weathersit' categorical independent variables & target variable

The insights that I can obtain from these boxplots are-

(i) for all the weekdays median count is in between 4000- 5000
(ii) for holiday- It is showing that median is high on 0 compared to 1
(iii) median count is higher on 2012 than 2011
(iv) there's high variability in median counts from different months, with July & September having highest median
(v) median count is higher for season 2 & season 3 compared to other seasons
(vi) median count is approximately same whether the day is working day or not
(vii) median count follows this pattern in the weathersit variable : 1>2>3

2.4. Missing Value Analysis

Using the ‘isnull’ function, I derived the sum of missing values in each variable in ‘df_day’ dataset & saved the missing values into ‘missing_val’ object. From ‘missing_val’, I can see that there aren’t any missing values present in the datset.

missing_val

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

Figure 6. Missing Value Distribution

2.5. Outlier Analysis

Then I generated boxplots for all the continuous variables in order to detect outliers.

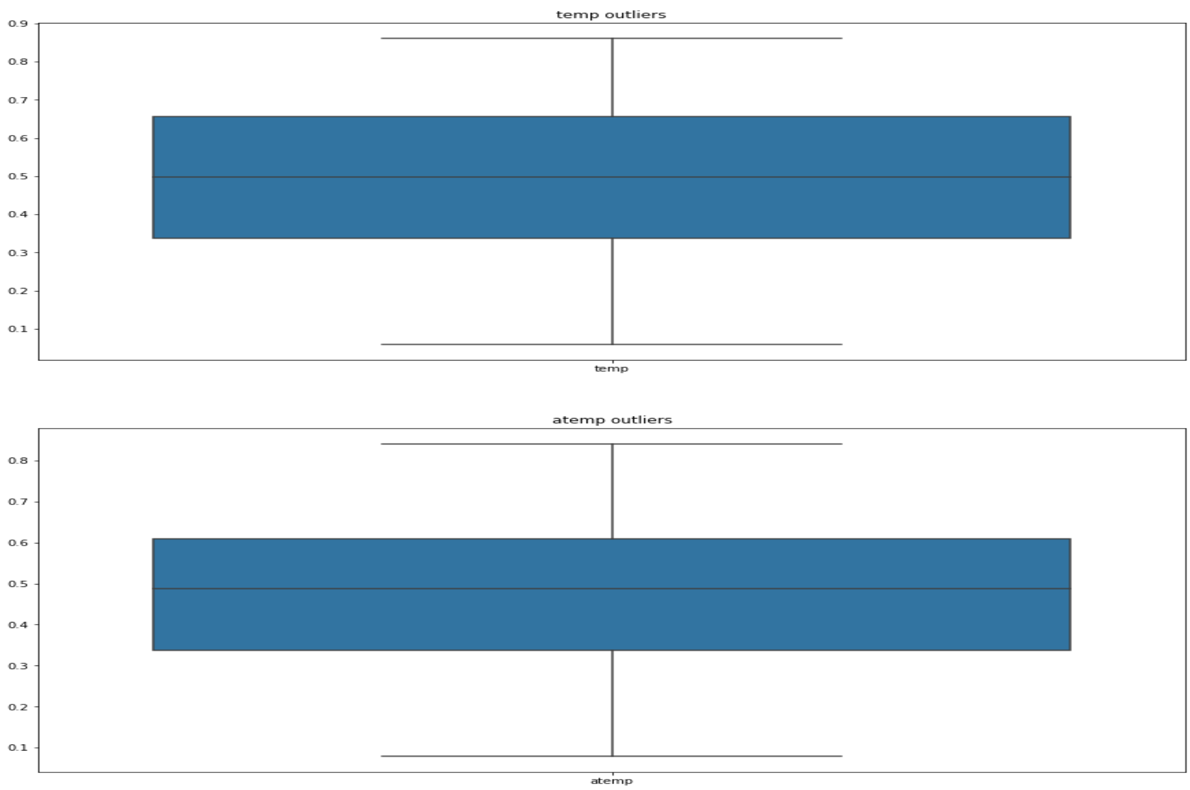


Figure 7A. Boxplots of ‘temp’ & ‘atemp’

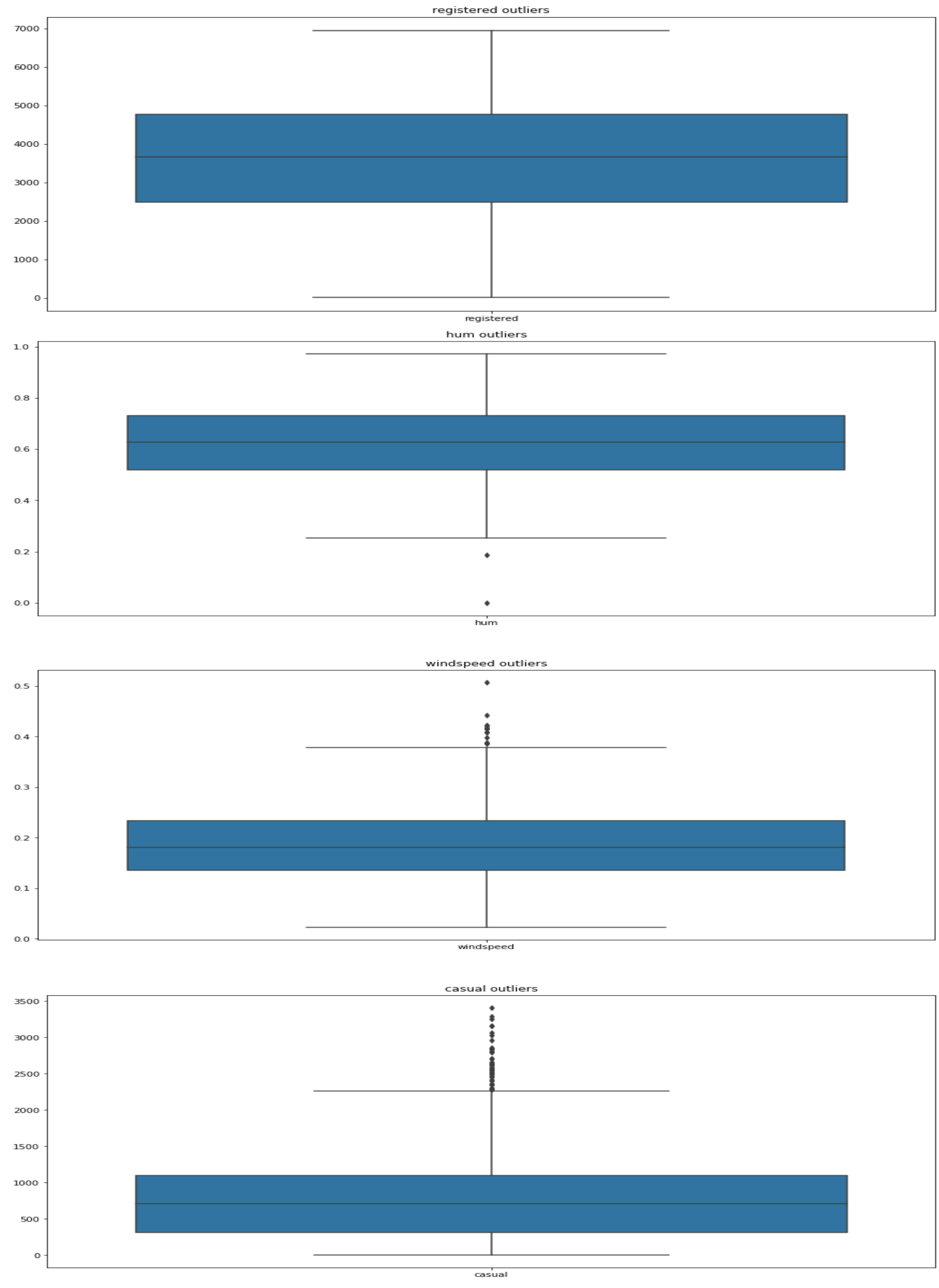


Figure 7B. Boxplots of ‘registered’, ‘hum’, ‘windspeed’ & ‘casual’

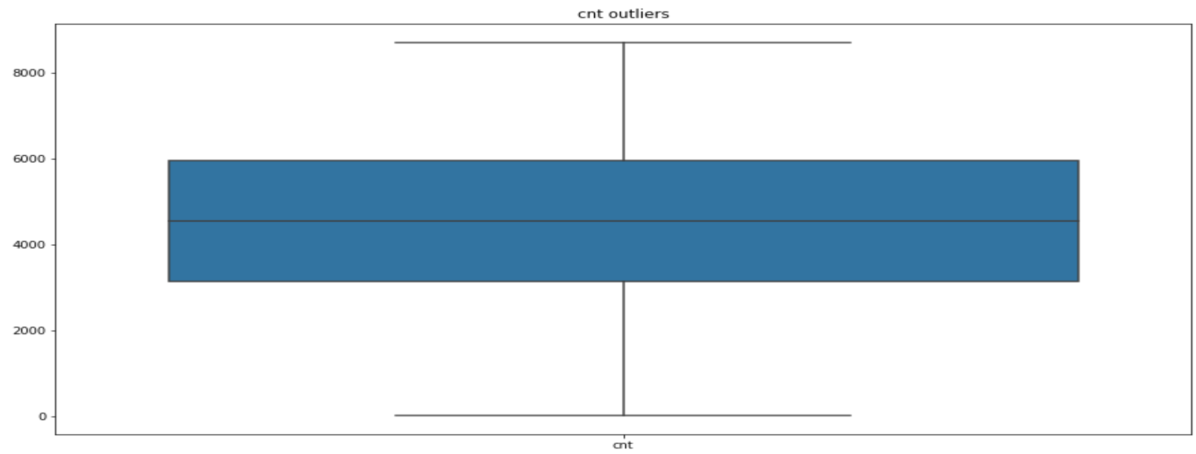


Figure 7C. Boxplots of 'cnt'

From these boxplots, I can see 'hum', 'windspeed', 'casual' variables contain outliers.

So, at first, I saved the 'hum' variable from 'df_day' dataset into 'df_1' dataframe & the 'hum' variable itself into 'c1' list. Then I derived the 25^{th} & 75^{th} percentile from the 'hum' variable in df_1 dataset. Then I derived the inter-quartile range from the same & assigned the upper (UL1) & lower fence (LL1) for 'hum'. Then I extracted the observations in 'hum' whose values were either greater than UL1 or less than LL1. Then I assigned 'nan' to them. Then using the 'isnull' function, I extracted those values in 'df_1' dataset which contains 'nan' & saved them in the 'missing_val_1' dataset. Undoubtedly, these are the outliers,

Then, using the same method for 'windspeed' & 'casual', I got the outliers for these variables as well, assigned 'nan' to them & saved these 'nan' values (outliers) into 'missing_val_2' & 'missing_val_3' dataset repectively. Then I merged 'missing_val_1', 'missing_val_2' & 'missing_val_3' into 'missing_val_ol' dataset. I saw that 'hum', 'windspeed', 'casual' variables contain 2, 13 & 44 outliers respectively. Now, I chose not to drop the observations containing outliers as I wanted to save the information, so I imputed the 'nan' values.

In order to select the best performing imputation method, at first, I took a known value from the 'df_1' dataset (5^{th} row in 'hum' variable) & assigned 'nan' to it, effectively making it a missing value. Then I imputed its value using mean, median & knn imputation method & found out that the predicted value I got using the median method is closest to the original value. So, I chose 'median' as the best performing method for imputation.
But, we've to remember, that as the mean & median are constants for any given column, & knn will vary with different data points. I can get different best methods if I take another known value for method selection.
Then I reloaded the df_1 dataset & imputed the 'nan's in 'missing_val_1', 'missing_val_2' & 'missing_val_3' dataset using median & again checked for 'nan' in these three datasets. Now I found zero 'nan' values. That means all the outliers have been successfully imputed in these 3 datasets. So, I replaced the columns containing outliers in my original 'df_day' dataset with the imputed columns from 'missing_val_1', 'missing_val_2' & 'missing_val_3' respectively.

2.6. Feature Selection

I saw that there are 7 independent categorical variables, 6 independent numeric variables & 1 dependent numeric variable. I've used correlation analysis for numerical variables & I've used different methods to check dependencies between them.

2.6.1. Correlation Analysis

At 1st, I saved all the numeric variables into 'day_numeric' dataframe. Then, to understand which features have multicollinearity in this dataset, I generated a heatmap, a correlation chart based on Pearson correlation & a scatter plot depicting the relation between each 2 variables in 'day_numeric' dataframe.

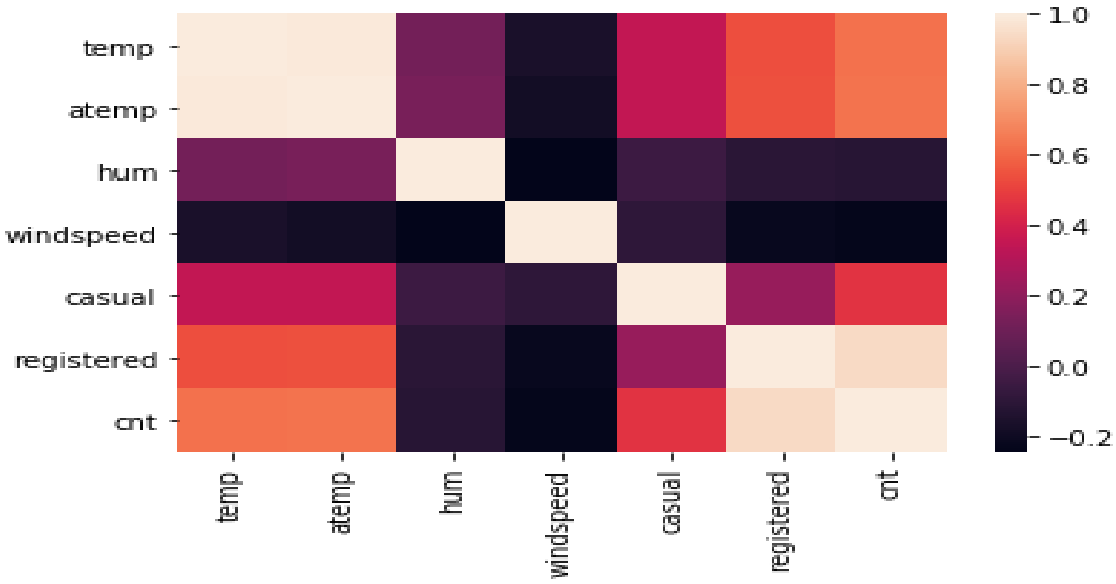


Figure 8A. Heatmap depicting correlation

	temp	atemp	hum	windspeed	casual	registered	cnt
temp	1.0	0.99	0.12	-0.16	0.35	0.54	0.63
atemp	0.99	1.0	0.14	-0.18	0.35	0.54	0.63
hum	0.12	0.14	1.0	-0.24	-0.05	-0.11	-0.12
windspeed	-0.16	-0.18	-0.24	1.0	-0.093	-0.22	-0.23
casual	0.35	0.35	-0.05	-0.093	1.0	0.22	0.47
registered	0.54	0.54	-0.11	-0.22	0.22	1.0	0.95
cnt	0.63	0.63	-0.12	-0.23	0.47	0.95	1.0

Figure 8B. Pearson Correlation Chart

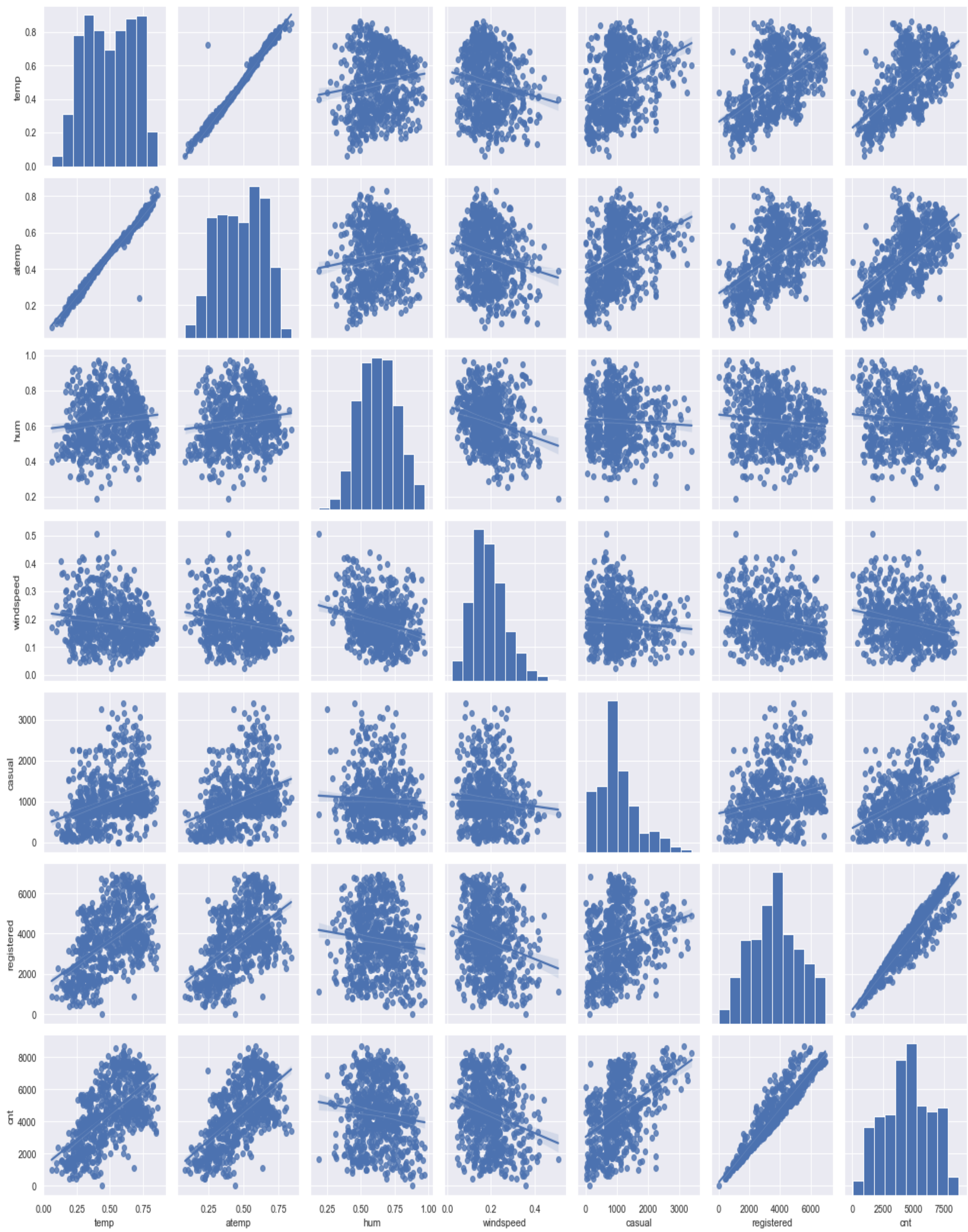


Figure 8C. Scatter plot depicting correlation

From these, I can see high positive correlation between Independent variables 'temp' and 'atemp' so, I'll drop atemp.

2.6.2. Analysis of Variance (ANOVA) Test

Then I saved all the categorical Variables in 'df_day' into 'cat_var' vector. After that I did ANOVA Test for the target variable with respect to the categorical Variables in 'cat_var'. Few prerequisites about ANOVA are-

- It is carried out to compare between each groups in a categorical variable.
- ANOVA only lets us know the means for different groups are same or not. It doesn't help us identify which mean is different.
- Hypothesis testing :
    - Null Hypothesis: mean of all categories in a variable are same.
    - Alternate Hypothesis: mean of at least one category in a variable is different.
- If p-value is less than 0.05 then we cannot accept the null hypothesis.
- And if p-value is greater than 0.05 then we accept the null hypothesis.

From the ANOVA table, I can see that the p-value is

- less than 0.05 for season
- less than 0.05 for weathersit
- less than 0.05 for yr
- less than 0.05 for mnth
- greater than 0.05 for weekday
- greater than 0.05 for holiday
- greater than 0.05 for workingday

So, I accepted the null hypothesis for weekday, holiday & workingday, saying that the means of all categories in these variables are same. &, I couldn't accept the null hypothesis for season, weathersit, yr & mnth, saying that the means of all categories in these variables are not same.

However, as ANOVA doesn't specify which group means are different, we can't conclude from the test results about which categorical variables I should remove.

2.6.3. Chi squared Test of Independence

So, I did Chi squared test of independence to select relevant features out of all the categorical features in 'cat_var'.

Few prerequisites about Chi squared test are-

- Hypothesis testing :
  - Null Hypothesis: 2 variables are independent
  - Alternate Hypothesis: 2 variables are not independent
- If p-value is less than 0.05 then we cannot accept the null hypothesis.
- And if p-value is greater than 0.05 then we accept the null hypothesis.

variables which are highly dependent on each other based on p-values are:

- season and month-0

- season and weathersit-0.0211

- mnth and weathersit-0.014

- holiday and weekday-8.56e-11

- hoilday and workingday-4.033e-11

- weekday and workingday-6.77e-136

So I will remove season,holiday.

Finally, I dropped the 'atemp', 'season' & 'holiday' variables from the 'df_day' dataset as these features were dependent on other features.

2.7. Model Development

2.7.1. Model Selection

As the dataset 'df_day' contains a target variable (cnt), so I'll have to use supervised machine learning algorithms. Now, the dependent variable can fall in any of the four categories:

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

As the dependent variable, in this case (cnt) is interval-scaled numeric variable; I'll have to do regression analysis to generate a model. I'm choosing these machine learning algorithms to solve this problem-

a) Decision tree
b) Random Forest
c) Linear Regression

As feature scaling does not have any impact on these algorithms, I chose not to do feature scaling.

2.7.2. Different Machine Learning Algorithms

a) Decision Tree Regressor

At first, I divided the ‘df_day’ dataset into ‘train’ and ‘test’ dataset using train_test_split function from scikit learn package (train containing 80% of the data). Here ‘train’ contains 584 observations & ‘test’ contains 147 observations. Then, I saved all the independent variables in ‘train’ into ‘train_features_one’ & the target variable in ‘train’ into ‘train_target_feature’. Next, I saved all the independent variables in ‘test into ‘test_features_one’ & the target variable in ‘test’ into ‘test_target_feature’.

After that, I generated the 1st decision tree model (fit_dt) by fitting ‘train_features_one’ & ‘train_target_feature’ with max_depth set at 2 & applying ‘DecisionTreeRegressor’ function. Then, I applied the ‘fit_dt’ model on the ‘test_features_one’ dataset to predict the target variable in ‘test’ dataset & saved the output (predictions) into ‘predictions_DT’.

Then, I derived 3 error metric functions, namely MAE, MAPE & RMSE & calculated the error rate of ‘ ’fit_dt’ using these metrics. I found out that for this model, MAE= 549.2014964998775, MAPE= 17.03381757772561% & RMSE= 736.8110420750387.

Then, in order to control the overfitting, I set "max_depth" to 14 and "min_samples_split" to 7 & generated the 2nd decision tree model (fit_dt_2). For this model, I got MAE= 206.4498866213152, MAPE= 5.725581445090314% & RMSE= 344.3223945256568.

Then, I set "max_depth" to 16 and "min_samples_split" to 8 & generated the 3rd decision tree model (fit_dt_3). For this model, I got, MAE= 211.36469063816003, MAPE= 5.896082901559185% & RMSE= 350.95749871340985.

As the error rate increased, I chose ‘fit_dt_2’ as the final decision tree model (with max_depth = 14 & min_samples_split =7).

.

b) Random Forest

At first, I generated the 1st random forest model (RF_model_one) using 'RandomForestRegressor' & fitting 'train_features_one' & 'train_target_feature' with n_estimators set at 500. Then, I applied the 'RF_model_one' on the 'test_features_one' dataset to predict the target variable in test dataset & saved the output (predictions) into 'RF_predict_one'. For this model, I got,
MAE= 133.55099319727896, MAPE= 4.0145657045380245% & RMSE= 187.8525706744702.

Then I applied the 'mutual_info_regression' function for feature ordering on 'train_features_one' & 'train_target_feature'; & saved the resultant file into 'mir_result'. Then I saved the feature importance's of 'RF_model_one' into 'importances' list. Then I sorted the feature importances by 'most important first' rule & saved into 'feature_importances' element. Then I saved all the Independent variables in 'train' into 'train_variables_one_1' dataset & printed the importances of these features. These are-

yr = 0.27706467431401305
mnth = 0.3801051158708031
weekday = 0.0805232101495541
workingday = 0.025237911320865836
weathersit = 0.060003221622345615
temp = 0.4044496896198808
hum = 0.04820394053561383
windspeed = 0.025118295631000542
casual = 0.3183048321850168
registered = 1.6768948531675667

Then, using 'matplotlib' package, I plotted a bar-chart with the variables in the x-axis & their importances in the y-axis.

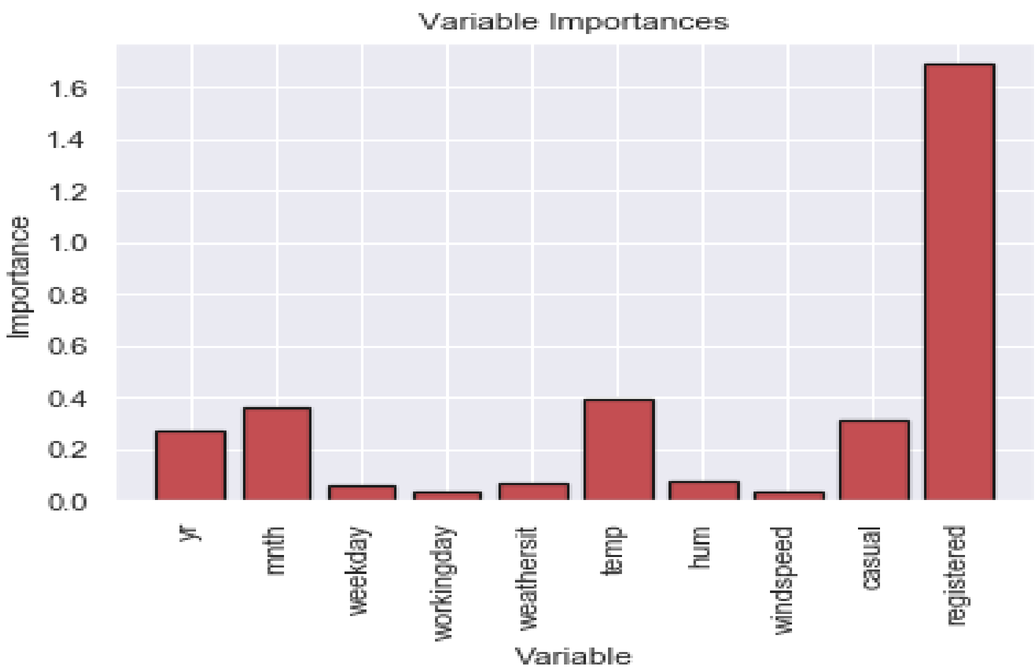


Figure 9. Variable Importance

The previous graph is stating that not all features are important to decide the accuracy of the model. So, I saved all the independent variables apart from the least important variable ‘workingday’ from the ‘train’ & ‘test’ into ‘train_feature_two’ & ‘test_feature_two’ respectively. I generated the 2nd random forest model (RF_model_two) by fitting ‘train_feature_two’ & ‘train_target_feature’ with n_estimators set at 500. Then, I applied the ‘RF_model_two’ on the ‘test_features_two’ dataset to predict the target variable in test dataset & saved the output (predictions) into ‘RF_predict_one’. For this model, I got, MAE= 156.0843537414966, MAPE= 4.658882779427421% & RMSE= 213.19551173856684.

So, I can see the error rate has increased. So, I'm not reducing the no. of variables (RF_model_one is the final Random forest model).

c) Linear Regression

As I have categorical variables in both the train & test dataset having more than 2 categories, I'll have to convert them to numeric type in order to use the 'linear regression' method optimally. At first, I saved 'yr', 'mnth', 'weekday', 'workingday' & 'weathersit' variables into 'cat_var1' list. Then, by using the 'get_dummies' function from 'pandas' package, I created dummy variables for all the categorical variables of the 'df_day' dataset & joined them to the same data. Then I dropped the original categorical variables as they contained redundant information. Now, I got a total of 32 variables. Then, by using the train_test_split function, I split the data into 'train_lr' & 'test_lr' (train_lr containing 80% of the data).

Then I saved all the independent features of 'train_lr' into 'train_features_lr' & the dependent variable into 'train_target_feature_lr'.

Then I saved all the independent features of 'test_lr' into 'test_features_lr' & the dependent variable into 'test_target_feature_lr'.

After that I developed, 'linear_regression_model' by fitting 'train_target_feature_lr', 'train_features_lr' using 'ols' function from 'statsmodels' library. Then I obtained the summary of the model.

Then, I applied the 'linear_regression_model' on the 'test_features_lr' dataset to predict the target variable in 'test_lr' dataset & saved the output (predictions) into 'predict_LR'. For this model, I got, MAE= 203.02773706676462, MAPE= 18.17626269527007% & RMSE= 270.9225450246571.

2.8. Conclusion

2.8.1. 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 of Bike rental prediction, Computation Efficiency, isn't that significant. Therefore I will use Interpretability and Predictive performance as the criteria to compare and evaluate models. Predictive performance can be measured by comparing some average error measure. I've used a total of 3 types of error metrics for the evaluation of different models. Those are-

(i) Mean Absolute Error (MAE)
(ii) Mean Absolute Percentage Error (MAPE)
(iii) Root Mean Square Error (RMSE)

However, as we're dealing with a non-time series data, RMSE is not recommended in this case. And, out of MAE & MAPE, MAPE is more Interpretable; so, I chose MAPE as the most significant error metric in this case.

2.8.2. Most Preferable Model Selection

Based on the Mean Absolute Percentage Error (MAPE), it can be observed that the Random Forest model, 'RF_model_one' is giving least amount of error.

That's why I chose Random Forest as the best model for this dataset.

2.9. Employing Model to predict new cases

I've selected a few observations from the given dataset & slightly altered them to create a new dataset in order to get features with realistic values; I shall use this dataset as a new sample input & predict the output & at last I'll see how well the model is performing (by checking the error rate).

Instructions for using this model on a dataset

1. Drop 'instant','dteday','season','holiday' & 'atemp' variables as these are statistically insignificant
2. Convert 'yr','mnth','weekday','workingday' & 'weathersit' variables to category type
3. Apply the model on the indepenent variables

First of all, the new dataset (day_new) in CSV format is loaded into the ‘df_eval’ dataset. Then, I dropped the 'instant', 'dteday', 'season', 'holiday' & 'atemp' variables as I previously found out that these variables are statistically insignificant. Then I converted the data types of 'yr', 'mnth', 'weekday', 'workingday' & 'weathersit' variables into category. Thereafter, I saved all the independent variables into ‘eval_features’; & the dependent variable into ‘eval_target_feature’.

Then I applied the ‘RF_model_one’ on the ‘eval_features’ to predict the target variable in ‘df_eval’ dataset & saved the output (predictions) into ‘RF_predict_eval’. For this model, I got,
MAPE= 7.13317658934405%

By getting such a low error rate, we can see that the generated model is performing well with a new dataset.

2.10. Complete Python Code

```python
import pandas as pd #for dataframe
import os #To Interact with local system directories
import numpy as np # linear algebra
import matplotlib.pyplot as plt # for visualizations
import seaborn as sns # for visualizations
from scipy import stats #import chi2_contigency #  for Chi square Test
from scipy.stats import chi2_contingency
from fancyimpute import KNN #for missing value analysis
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor # for decission tree
from sklearn.ensemble import RandomForestRegressor # for random forest
import sklearn.feature_selection as fs # feature selection library in scikit-learn
import statsmodels.api as sm # for linear regression
get_ipython().run_line_magic('matplotlib', 'inline')
os.getcwd()
os.chdir("C:\Python")
os.getcwd()
df_day = pd.read_csv('day.csv')
df_day.shape
df_day.describe()
df_day.info()
df_day.head()
df_day=df_day.drop(['instant','dteday'],axis=1)
df_day.shape
df_day.head()
cat_var=['season','yr','mnth','holiday','weekday','workingday','weathersit']
df_day[cat_var]=df_day[cat_var].apply(lambda x: x.astype('category') )
df_day.describe()
df_day.info()
sns.distplot(df_day['cnt']);
print("Skewness: %f" % df_day['cnt'].skew())
print("Kurtosis: %f" % df_day['cnt'].kurt())
sns.distplot(df_day['temp']);
sns.distplot(df_day['atemp']);
sns.distplot(df_day['hum']);
sns.distplot(df_day['windspeed']);
sns.distplot(df_day['casual']);
sns.distplot(df_day['registered']);
df_day['temp'].value_counts()
var = 'temp'
data = pd.concat([df_day['cnt'], df_day[var]], axis=1)
```

```python
data.plot.scatter(x=var, y='cnt', ylim=(0,9000));
df_day['atemp'].value_counts()
var = 'atemp'
data = pd.concat([df_day['cnt'], df_day[var]], axis=1)
data.plot.scatter(x=var, y='cnt', ylim=(0,9000));
df_day['hum'].value_counts()
var = 'hum'
data = pd.concat([df_day['cnt'], df_day[var]], axis=1)
data.plot.scatter(x=var, y='cnt', ylim=(0,9000));
df_day['windspeed'].value_counts()
var = 'windspeed'
data = pd.concat([df_day['cnt'], df_day[var]], axis=1)
data.plot.scatter(x=var, y='cnt', ylim=(0,9000));
df_day['casual'].value_counts()
var = 'casual'
data = pd.concat([df_day['cnt'], df_day[var]], axis=1)
data.plot.scatter(x=var, y='cnt', ylim=(0,9000));
df_day['registered'].value_counts()
var = 'registered'
data = pd.concat([df_day['cnt'], df_day[var]], axis=1)
data.plot.scatter(x=var, y='cnt', ylim=(0,9000));
var_weekdays = 'weekday'
data = pd.concat([df_day['cnt'], df_day[var_weekdays]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var_weekdays, y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
var_holiday = 'holiday'
data = pd.concat([df_day['cnt'], df_day[var_holiday]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var_holiday, y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
var_yr = 'yr'
data = pd.concat([df_day['cnt'], df_day[var_yr]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var_yr, y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
var_mnth = 'mnth'
data = pd.concat([df_day['cnt'], df_day[var_mnth]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var_mnth, y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
var_season = 'season'
data = pd.concat([df_day['cnt'], df_day[var_season]], axis=1)
```

```
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var_season, y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
var_wd = 'workingday'
data = pd.concat([df_day['cnt'], df_day[var_wd]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var_wd, y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
var_ws = 'weathersit'
data = pd.concat([df_day['cnt'], df_day[var_ws]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var_ws, y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
missing_val = pd.DataFrame(df_day.isnull().sum())
missing_val
fig,ax=plt.subplots(figsize=(15,8))
sns.boxplot(data=df_day[['temp']])
ax.set_title('temp outliers')
plt.show()
fig,ax=plt.subplots(figsize=(15,8))
sns.boxplot(data=df_day[['atemp']])
ax.set_title('atemp outliers')
plt.show()
fig,ax=plt.subplots(figsize=(15,8))
sns.boxplot(data=df_day[['hum']])
ax.set_title('hum outliers')
plt.show()
fig,ax=plt.subplots(figsize=(15,8))
sns.boxplot(data=df_day[['windspeed']])
ax.set_title('windspeed outliers')
plt.show()
fig,ax=plt.subplots(figsize=(15,8))
sns.boxplot(data=df_day[['casual']])
ax.set_title('casual outliers')
plt.show()
fig,ax=plt.subplots(figsize=(15,8))
sns.boxplot(data=df_day[['registered']])
ax.set_title('registered outliers')
plt.show()
fig,ax=plt.subplots(figsize=(15,8))
sns.boxplot(data=df_day[['cnt']])
ax.set_title('cnt outliers')
plt.show()
df_1=pd.DataFrame(df_day,columns=['hum'])
c1 = ['hum']
for i in c1:
    print(i)
    q75,q25=np.percentile(df_1.loc[:,i],[75,25]) # Divide data into 75%quantile and 25%quantile.
    iqr=q75-q25 #Inter quantile range
    LL1=q25-(iqr*1.5) #inner fence
    UL1=q75+(iqr*1.5) #outer fence
    print(LL1)
    print(UL1)
df_1.loc[df_1['hum']<LL1,:i]=np.nan  #Replace with NA
df_1.loc[df_1['hum']>UL1,:i]=np.nan  #Replace with NA
missing_val_1 = pd.DataFrame(df_1.isnull().sum())
missing_val_1
df_2=pd.DataFrame(df_day,columns=['windspeed'])
c2 = ['windspeed']
for i in c2:
    print(i)
    q75,q25=np.percentile(df_2.loc[:,i],[75,25]) # Divide data into 75%quantile and 25%quantile.
    iqr=q75-q25 #Inter quantile range
    LL2=q25-(iqr*1.5) #inner fence
    UL2=q75+(iqr*1.5) #outer fence
    print(LL2)
    print(UL2)
df_2.loc[df_2['windspeed']<LL2,:i]=np.nan  #Replace with NA
df_2.loc[df_2['windspeed']>UL2,:i]=np.nan  #Replace with NA
missing_val_2 = pd.DataFrame(df_2.isnull().sum())
missing_val_2
df_3 = pd.DataFrame(df_day,columns=['casual'])
c3 = ['casual']
for i in c3:
    print(i)
    q75,q25=np.percentile(df_3.loc[:,i],[75,25]) # Divide data into 75%quantile and 25%quantile.
    iqr=q75-q25 #Inter quantile range
    LL3=q25-(iqr*1.5) #inner fence
    UL3=q75+(iqr*1.5) #outer fence
    print(LL3)
    print(UL3)
df_3.loc[df_3['casual']<LL3,:i]=np.nan  #Replace with NA
df_3.loc[df_3['casual']>UL3,:i]=np.nan  #Replace with NA
missing_val_3 = pd.DataFrame(df_3.isnull().sum())
```

```python
missing_val_3
missing_val_ol = missing_val_1.append(missing_val_2).append(missing_val_3)
missing_val_ol
df_1['hum']=df_1['hum'].fillna(df_1['hum'].median())
df_2['windspeed']=df_2['windspeed'].fillna(df_2['windspeed'].median())
df_3['casual']=df_3['casual'].fillna(df_3['casual'].median())
df_1.isnull().sum()
df_2.isnull().sum()
df_3.isnull().sum()
df_day['hum']=df_day['hum'].replace(df_1['hum'])
df_day['windspeed']=df_day['windspeed'].replace(df_2['windspeed'])
df_day['casual']=df_day['casual'].replace(df_3['casual'])
df_day.head()
df_day.describe()
df_day.info()
day_numeric = df_day.loc[:,['temp','atemp','hum','windspeed','casual','registered','cnt']]
day_numeric.shape
sns.heatmap(day_numeric.corr())
day_numeric.corr(method='pearson').style.format("{:.2}").background_gradient(cmap=plt.get_cmap('coolwarm'), axis=1)
sns.set()
cols = ['temp', 'atemp', 'hum', 'windspeed', 'casual','registered','cnt']
sns.pairplot(day_numeric[cols], height = 2.5,kind="reg")
plt.show();
cat_var
def anova_test(df_day,target):
    for i in cat_var:
        formula=('{} ~ {}').format(target, i)
        df_day.lm = ols(formula,data=df_day).fit()
        table = sm.stats.anova_lm(df_day.lm, typ=1)
        print('Anova table between',target,'and',i,'is\n',table)
from statsmodels.formula.api import ols
print('\n For target var = cnt--')
anova_test(df_day,'cnt')
for i in cat_var:
    for j in cat_var:
        if(i != j):
            chi2, p, dof, ex = chi2_contingency(pd.crosstab(df_day[i], df_day[j]))
            if(p < 0.05):
                print(i,"and",j,"are dependent on each other with",p,'----Remove')
            else:
                print(i,"and",j,"are independent on each other with",p,'----Keep')
df_day = df_day.drop(['atemp','season','holiday'],axis = 1)
df_day.info()
df_day.shape
train, test = train_test_split(df_day, test_size=0.2)
train.shape
train.head()
test.shape
test.head()
train_features_one = train[['yr','mnth','weekday','workingday','weathersit','temp','hum','windspeed','casual','registered']].values
train_target_feature = train['cnt'].values
test_features_one = test[['yr','mnth','weekday','workingday','weathersit','temp','hum','windspeed','casual','registered']].values
test_target_feature= test['cnt'].values
train_features_one
fit_dt = DecisionTreeRegressor(max_depth=2).fit(train_features_one, train_target_feature)
print(fit_dt)
predictions_DT = fit_dt.predict(test_features_one)
print(predictions_DT)
def MAE(y_true, y_pred):
    mae = np.mean(np.abs(y_true - y_pred))
    return mae
MAE(test_target_feature, predictions_DT)
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape
MAPE(test_target_feature, predictions_DT)
def RMSE(y_test,y_predict):
    mse = np.mean((y_test-y_predict)**2)
    rmse=np.sqrt(mse)
    return rmse
RMSE(test_target_feature, predictions_DT)
max_depth = 14
min_samples_split =7
fit_dt_2 = DecisionTreeRegressor(max_depth =max_depth , min_samples_split =min_samples_split, random_state = 1)
fit_dt_2 = fit_dt_2.fit(train_features_one, train_target_feature)
print(fit_dt_2)
predictions_DT_two = fit_dt_2.predict(test_features_one)
print(predictions_DT_two)
MAE(test_target_feature,predictions_DT_two)
MAPE(test_target_feature,predictions_DT_two)
RMSE(test_target_feature,predictions_DT_two)
max_depth = 16
min_samples_split =8
fit_dt_3 = DecisionTreeRegressor(max_depth =max_depth , min_samples_split =min_samples_split, random_state = 1)
fit_dt_3 = fit_dt_3.fit(train_features_one, train_target_feature)
```

```
print(fit_dt_3)
predictions_DT_three = fit_dt_3.predict(test_features_one)
print(predictions_DT_three)
MAE(test_target_feature,predictions_DT_three)
MAPE(test_target_feature,predictions_DT_three)
RMSE(test_target_feature,predictions_DT_three)
RF_model_one = RandomForestRegressor(n_estimators= 500, random_state=100).fit(train_features_one,train_target_feature)
RF_predict_one= RF_model_one.predict(test_features_one)
MAE(test_target_feature,RF_predict_one)
MAPE(test_target_feature,RF_predict_one)
RMSE(test_target_feature,RF_predict_one)
mir_result = fs.mutual_info_regression(train_features_one, train_target_feature) # mutual information regression for feature ordering
mir_result
importances = list(RF_model_one.feature_importances_)
print(importances)
feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(train_features_one, importances)]
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)
train_variables_one_1= train[['yr','mnth','weekday','workingday','weathersit','temp','hum','windspeed','casual','registered']]
train_variables_one_1
for name, importance in zip(train_variables_one_1, mir_result):
    print(name, "=", importance)
x_values = list(range(len(mir_result)))
plt.bar(x_values, mir_result, orientation = 'vertical', color = 'r', edgecolor = 'k', linewidth = 1.2)
plt.xticks(x_values, train_variables_one_1, rotation='vertical')
plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importances');
train_feature_two = train[["yr" ,"mnth","weekday","weathersit","temp","hum","windspeed","casual","registered"]].values
test_feature_two= test[["yr" ,"mnth","weekday","weathersit","temp","hum","windspeed","casual","registered"]].values
Rf_model_two = RandomForestRegressor(n_estimators= 500, random_state=100).fit(train_feature_two,train_target_feature)
RF_predict_two= Rf_model_two.predict(test_feature_two)
print(RF_predict_two)
MAE(test_target_feature,RF_predict_two)
MAPE(test_target_feature,RF_predict_two)
RMSE(test_target_feature,RF_predict_two)
df_day.head()
cat_var1 = ['yr','mnth','weekday','workingday','weathersit']
for i in cat_var1:
    ''' Creating dummies for each variable in cat_var and merging dummies dataframe to our original dataframe '''
    temp = pd.get_dummies(df_day[i], prefix = i)
    df_day = df_day.join(temp)
df_day.head()
df_day.columns
df_day = df_day.drop(['yr','mnth','weekday','workingday','weathersit'],axis = 1)
df_day.shape
```

```
train_lr, test_lr = train_test_split(df_day, test_size=0.2)
train_lr.shape
train_lr.head()
test_lr.shape
test_lr.head()
train_features_lr = train_lr[['temp', 'hum', 'windspeed', 'casual', 'registered',
        'weathersit_1', 'weathersit_2', 'weathersit_3', 'yr_0', 'yr_1',
        'mnth_1', 'mnth_2', 'mnth_3', 'mnth_4', 'mnth_5', 'mnth_6', 'mnth_7',
        'mnth_8', 'mnth_9', 'mnth_10', 'mnth_11', 'mnth_12', 'weekday_0',
        'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4', 'weekday_5',
        'weekday_6', 'workingday_0', 'workingday_1']].values
train_target_feature_lr = train_lr['cnt'].values
test_features_lr = test_lr[['temp', 'hum', 'windspeed', 'casual', 'registered',
        'weathersit_1', 'weathersit_2', 'weathersit_3', 'yr_0', 'yr_1',
        'mnth_1', 'mnth_2', 'mnth_3', 'mnth_4', 'mnth_5', 'mnth_6', 'mnth_7',
        'mnth_8', 'mnth_9', 'mnth_10', 'mnth_11', 'mnth_12', 'weekday_0',
        'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4', 'weekday_5',
        'weekday_6', 'workingday_0', 'workingday_1']].values
test_target_feature_lr= test_lr['cnt'].values
train_features_lr
linear_regression_model = sm.OLS(train_target_feature_lr, train_features_lr).fit()
linear_regression_model.summary()
predict_LR = linear_regression_model.predict(test_features_lr)
print(predict_LR)
MAE(test_target_feature_lr,predict_LR)
MAPE(test_target_feature_lr,predict_LR)
RMSE(test_target_feature_lr,predict_LR)
df_eval = pd.read_csv('day_new.csv')
df_eval.shape
df_eval
df_eval.info()
df_eval=df_eval.drop(['instant','dteday','season','holiday','atemp'],axis=1)
df_eval
cat_var_new=['yr','mnth','weekday','workingday','weathersit']
df_eval[cat_var_new]=df_eval[cat_var_new].apply(lambda x: x.astype('category') )
df_eval['cnt'] = df_eval['cnt'].astype('float64')
df_eval['casual'] = df_eval['casual'].astype('float64')
df_eval['registered'] = df_eval['registered'].astype('float64')
df_eval.columns
eval_features = df_eval[['yr', 'mnth', 'weekday', 'workingday',
        'weathersit', 'temp', 'hum', 'windspeed', 'casual',
        'registered']].values
eval_target_feature = df_eval['cnt'].values
RF_predict_eval= RF_model_one.predict(eval_features)
MAPE(eval_target_feature,RF_predict_eval)
```

Chapter 3

Methodology for R

3.1. Data Pre- processing

After setting the working directory, the given ‘day’ dataset in CSV format is loaded into the ‘day’ dataset. I can see that out of 16 variables 'instant','dteday' variables simply represent the ‘record index’ & ‘date’; hence are statistically insignificant. So, I dropped these 2 variables. Also, I can see that even though 'season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday' & 'weathersit' are in ‘int’ format, they’re actually categorical variables consisting of 2 or more categories. So, I converted these variables to ‘category’ format.

3.2. Univariate Analysis

I plotted the target variable ‘cnt’ using the ‘ggplot2’ library to check normality of this variable.

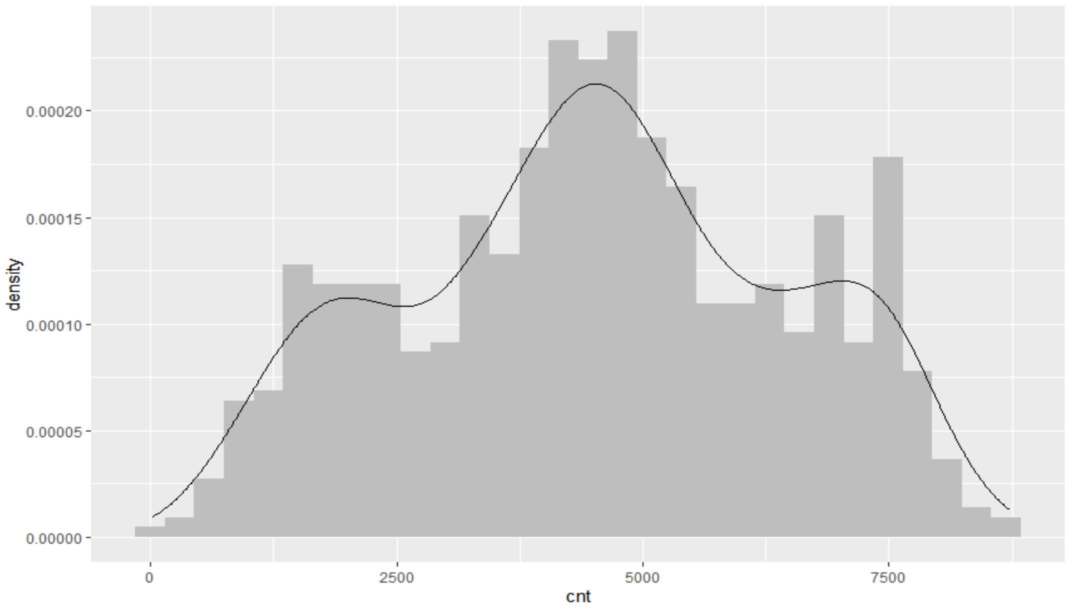


Figure 10. Distribution of target Variable

Then I plotted all 6 independent numeric variables using the same process in order to check normality of them.

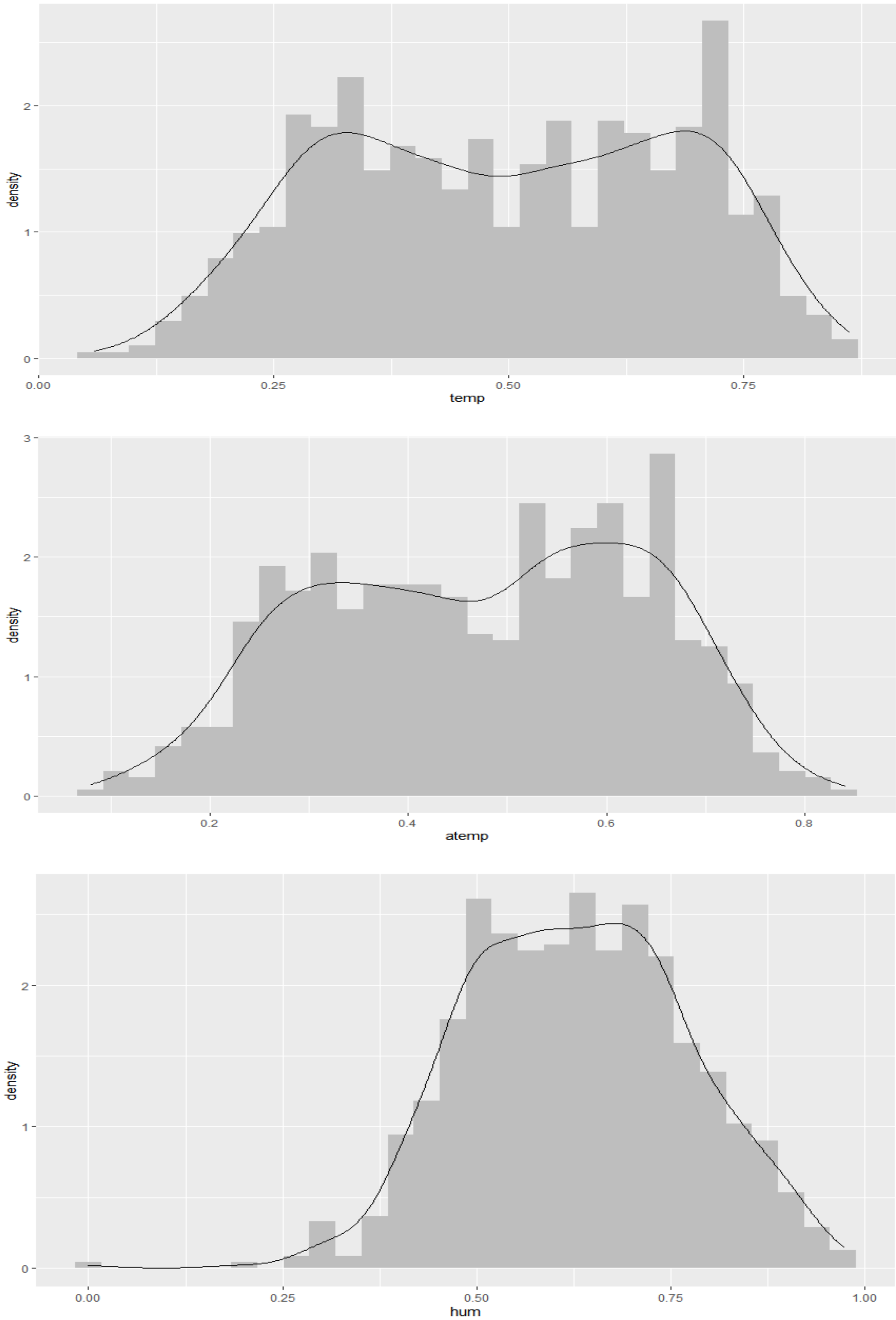


Figure 11A. Distribution of 'temp', 'atemp' & 'hum' independent variables

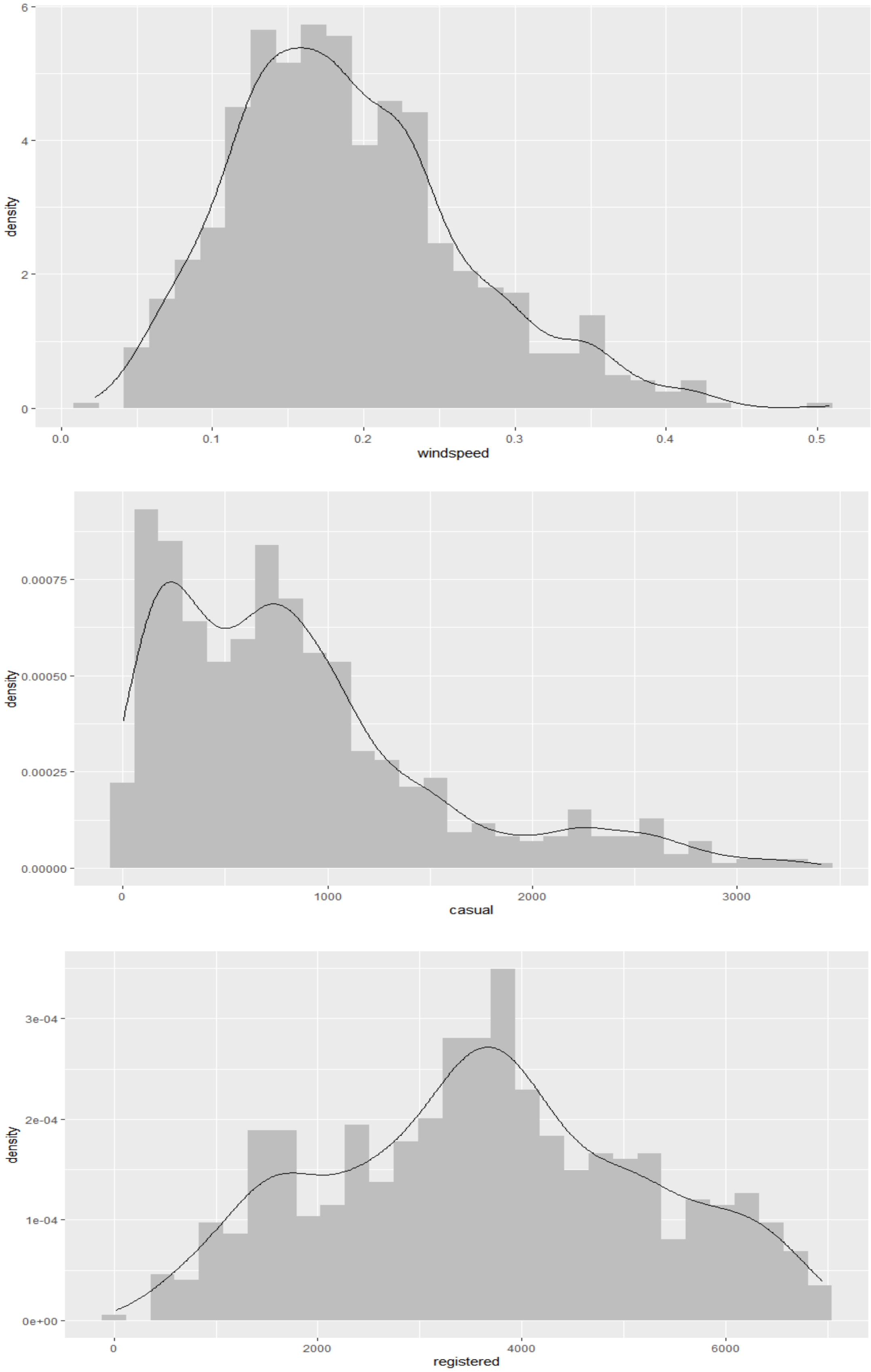


Figure 11B. Distribution of ‘windspeed, ‘casual & ‘registered independent variables

3.3. Bivariate Analysis

Next, I plotted the relation between target variable & all the continuous independent variables using ggplot2 library with the continuous independent variables in the x-axis & the target variable 'cnt' in the y-axis. Thus, I got an idea of the relation between different independent numeric variables & target variables. The generated scatter plots are-

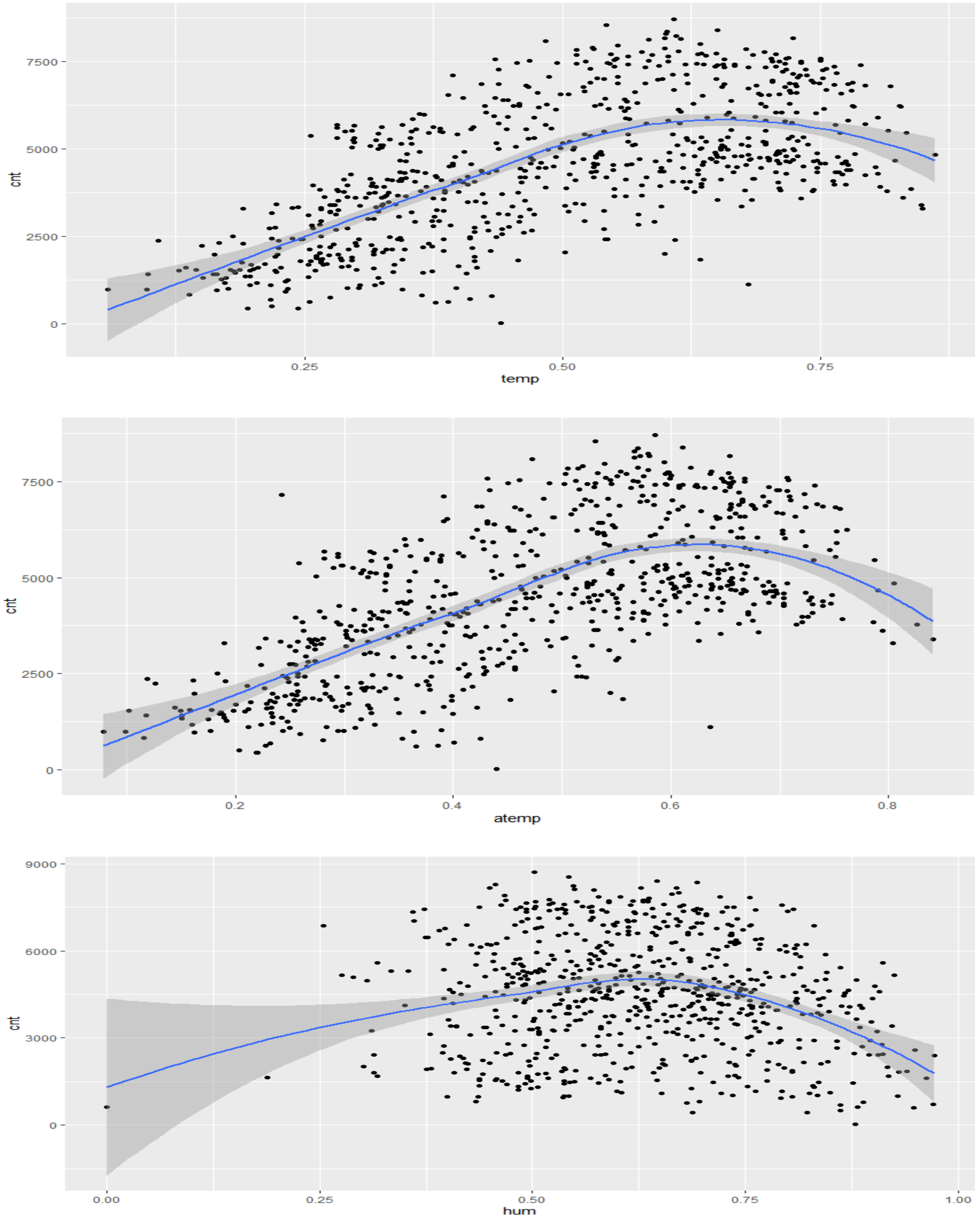


Figure 12A. Relation between 'temp', 'atemp' & 'hum' continuous independent variables & target variable

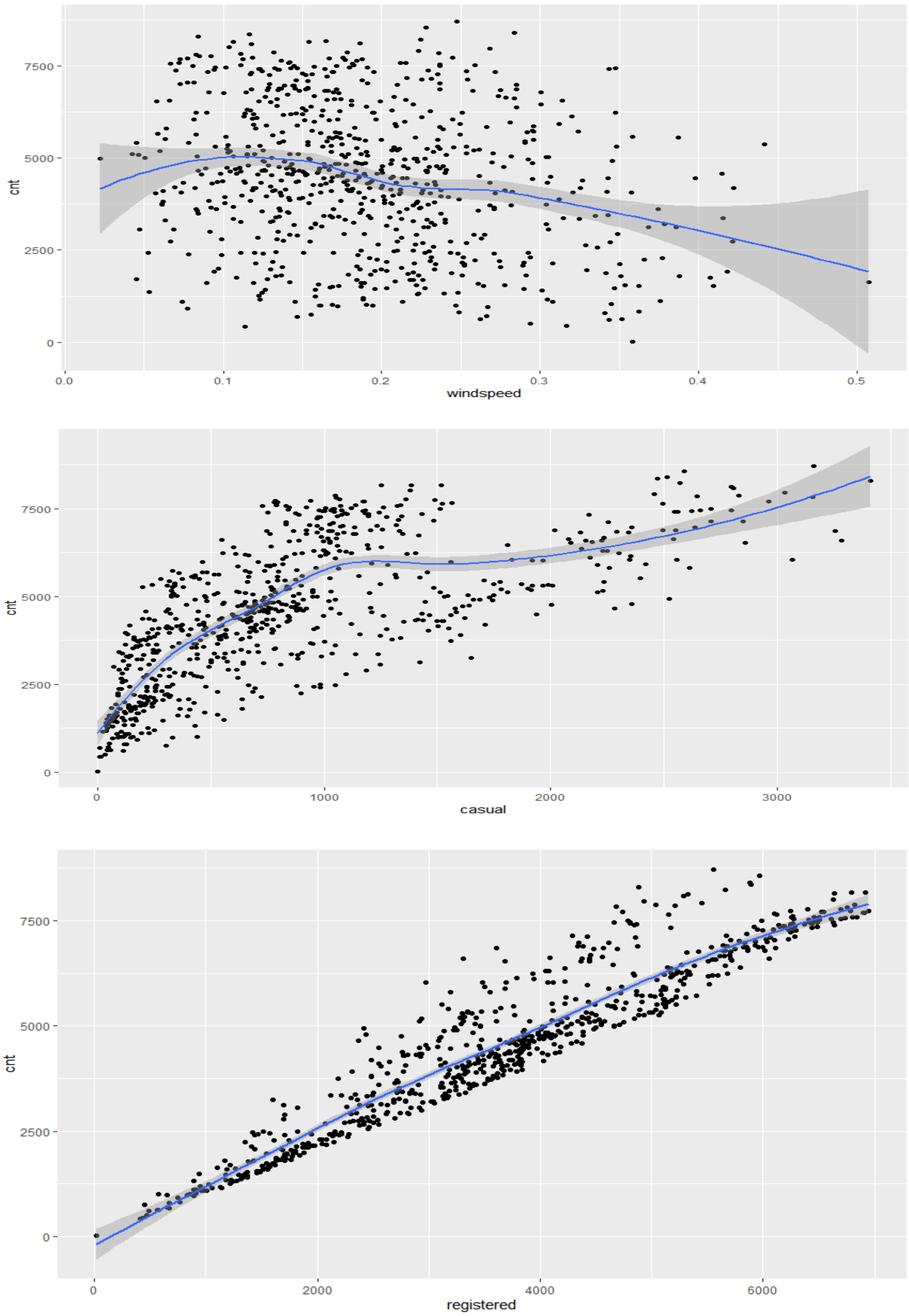


Figure 12B. Relation between 'windspeed, 'casual' & 'registered' continuous independent variables & target variable

The insights that I can obtain from these scatter plots are-

(i) there is good positive relation between 'temp' and 'cnt'
(ii) there is good positive relation between 'atemp' and 'cnt'
(iii) there is poor relation between 'hum' and 'cnt'
(iv) there is negative relation between 'windspeed' and 'cnt'
(v) there is somewhat good positive relation between 'casual' and 'cnt'
(vi) there is good relation between 'registered' and 'cnt'

After that, I generated boxplots of the target variable for each of the categories in all the categorical variables. These are the generated boxplots-

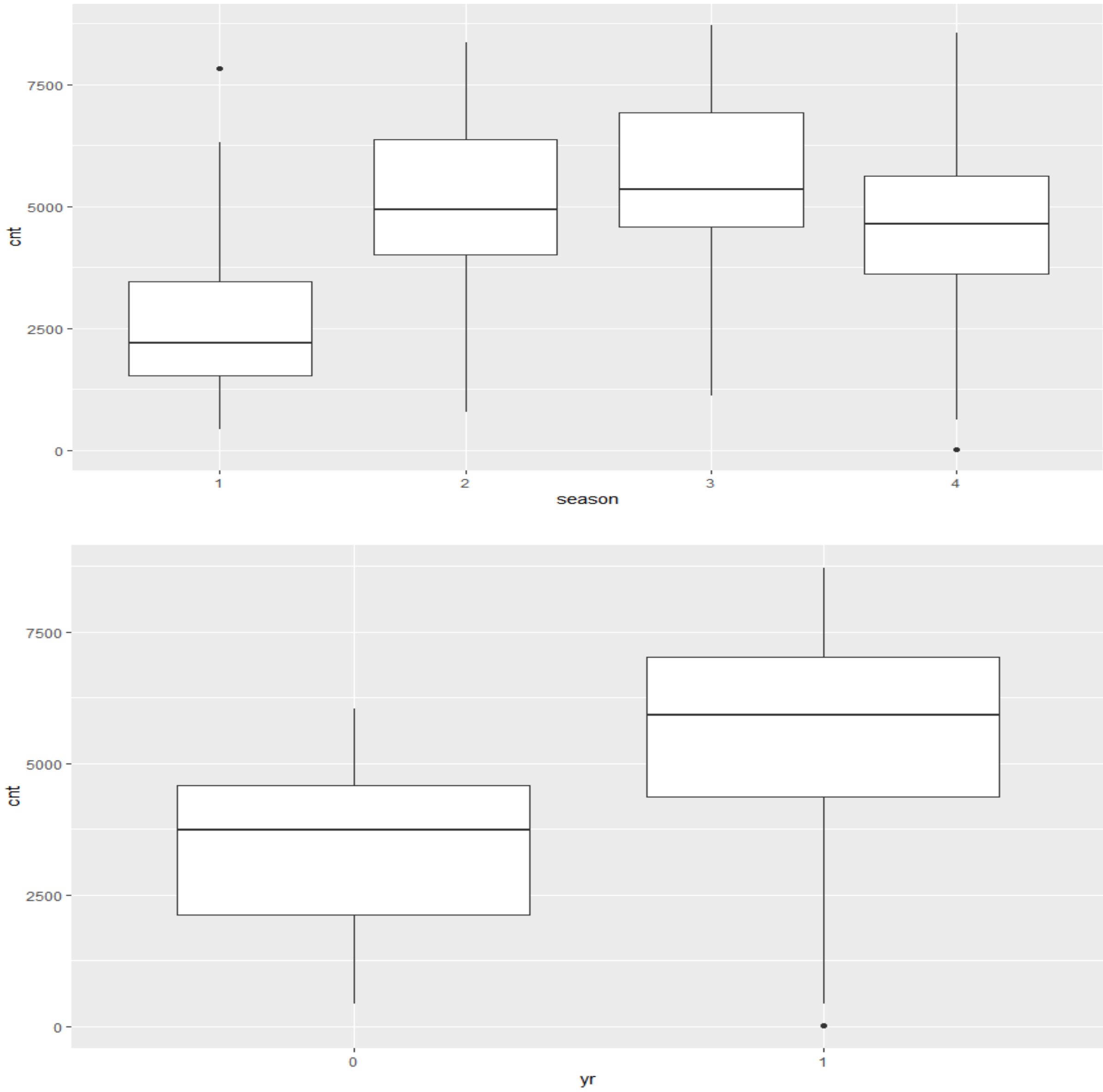


Figure 13A. Relation between 'season', 'yr' categorical independent variables & target variable

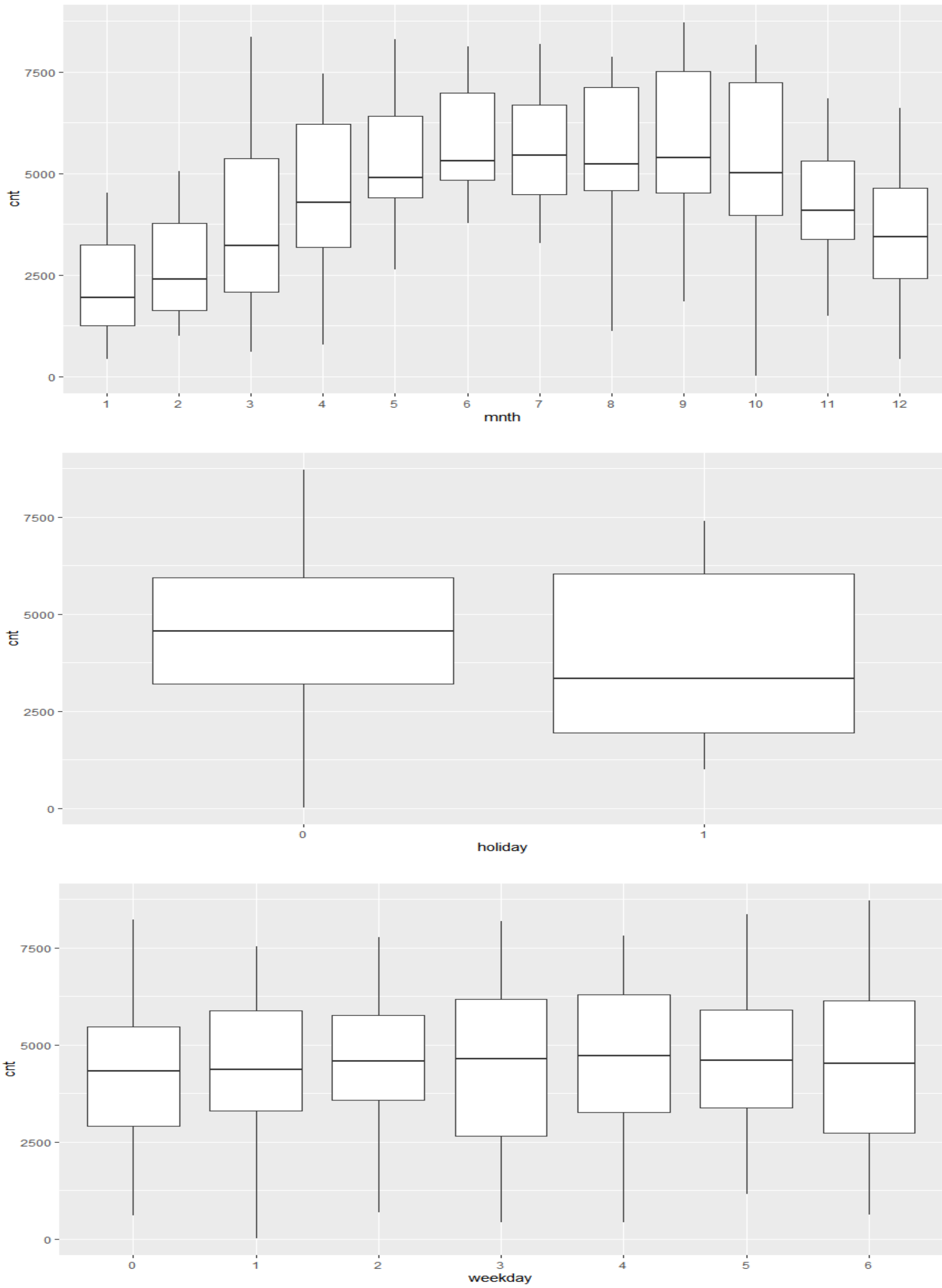


Figure 13B. Relation between ‘mnth’, ‘holiday’, ‘weekday’ categorical independent variables & target variable

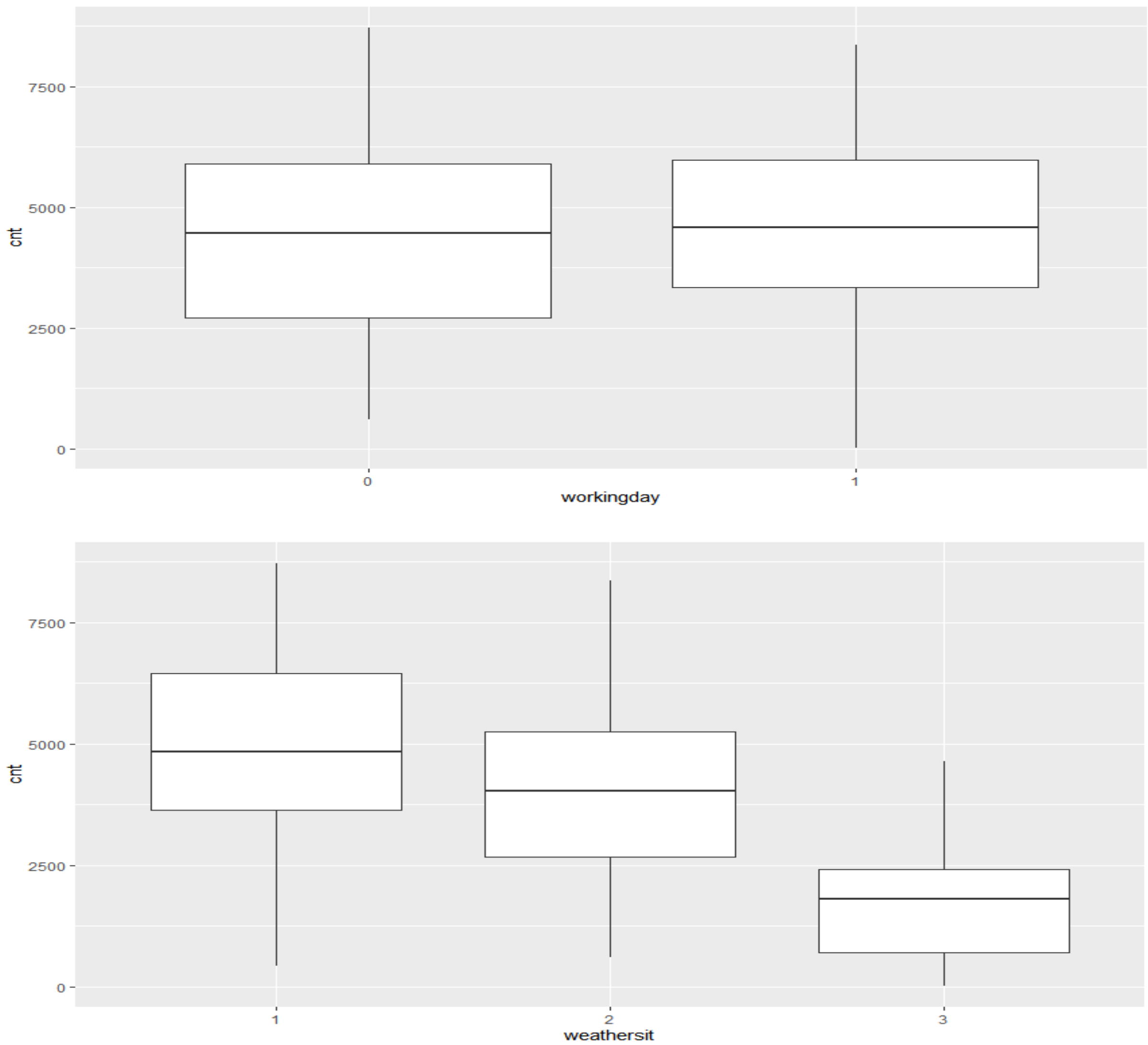


Figure 13C. Relation between 'workingday', 'weathersit' categorical independent variables & target variable

The insights that I can obtain from these boxplots are-

(i) for all the weekdays median count is in between 4000- 5000
(ii) for holiday- It is showing that median is high on 0 compared to 1
(iii) median count is higher on 2012 than 2011
(iv) there's high variability in median counts from different months, with July & September having highest median
(v) median count is higher for season 2 & season 3 compared to other seasons
(vi) median count is approximately same whether the day is working day or not
(vii) median count follows this pattern in the weathersit variable : 1>2>3

3.4. Missing Value Analysis

At first, I extracted all the missing values (na) in the df_day dataset into 'missing_val' dataframe; then I set proper column names & ordered the 'missing_val' dataset.

	Columns	Missing_Value
1	season	0
2	yr	0
3	mnth	0
4	holiday	0
5	weekday	0
6	workingday	0
7	weathersit	0
8	temp	0
9	atemp	0
10	hum	0
11	windspeed	0
12	casual	0
13	registered	0
14	cnt	0

Figure 14. Missing Value Distribution

From this, I can see that there are no missing values.

3.5. Outlier Analysis

Then I generated boxplots for all the continuous variables in order to detect outliers.

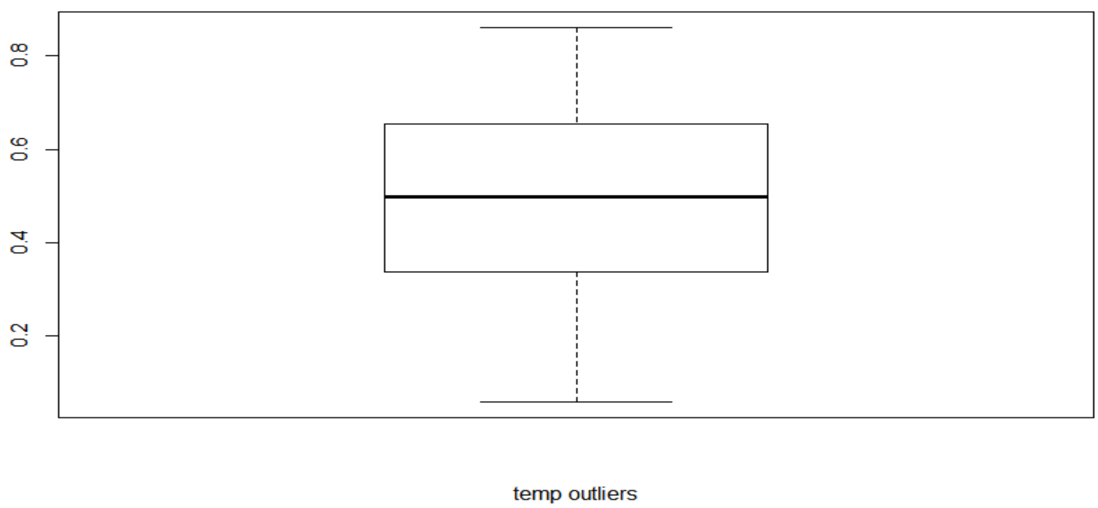


Figure 15A. Boxplots of 'temp'

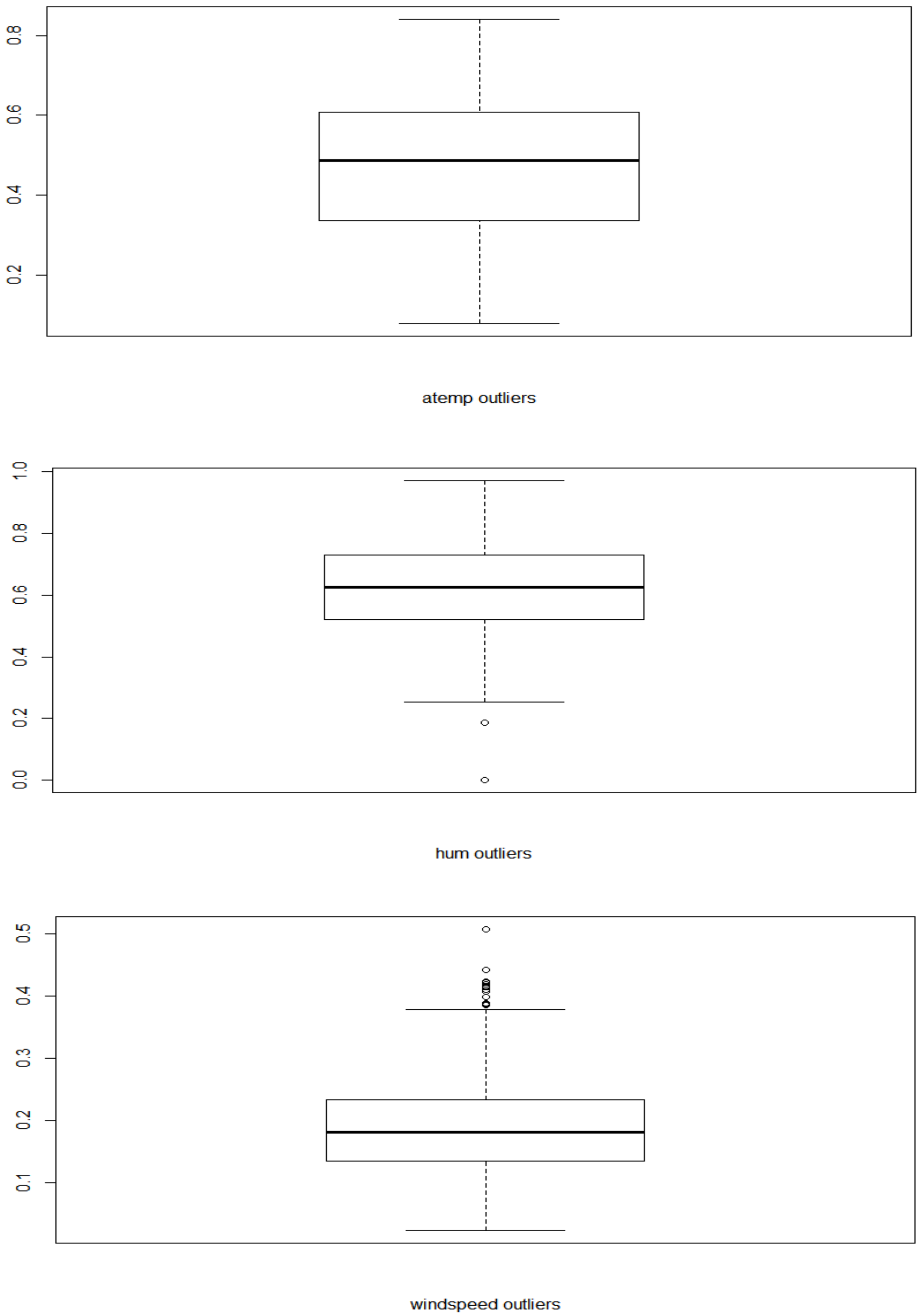
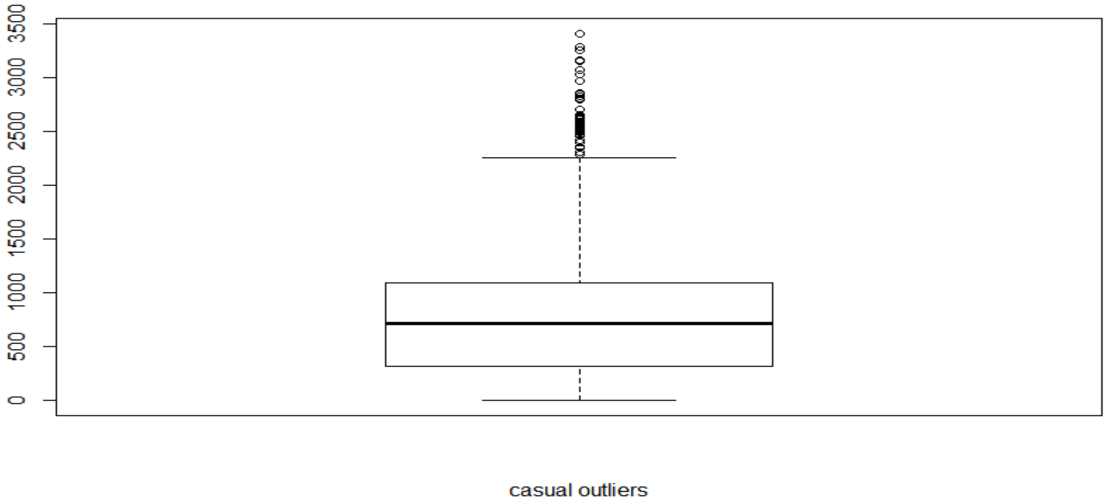
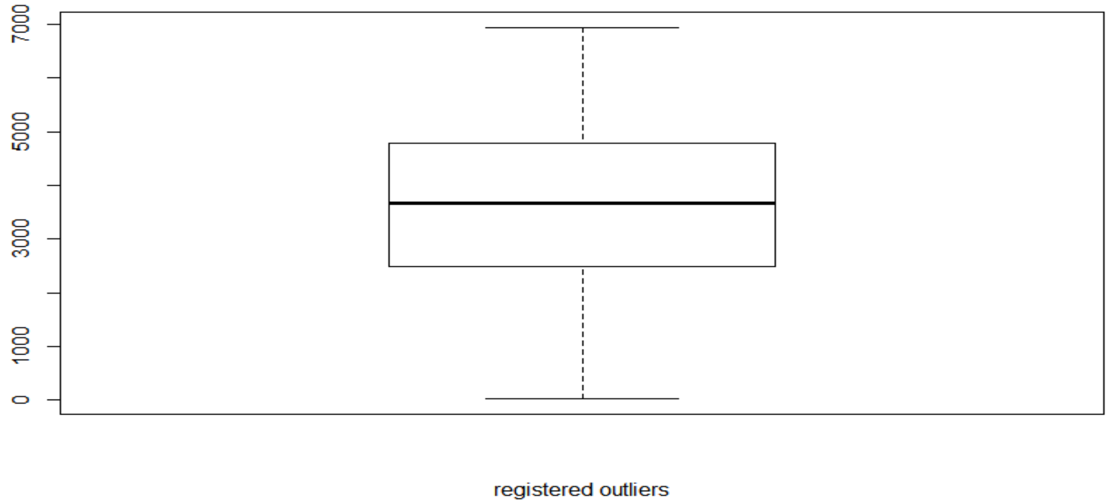


Figure 15B. Boxplots of ‘atemp’, ‘hum’, ‘windspeed’





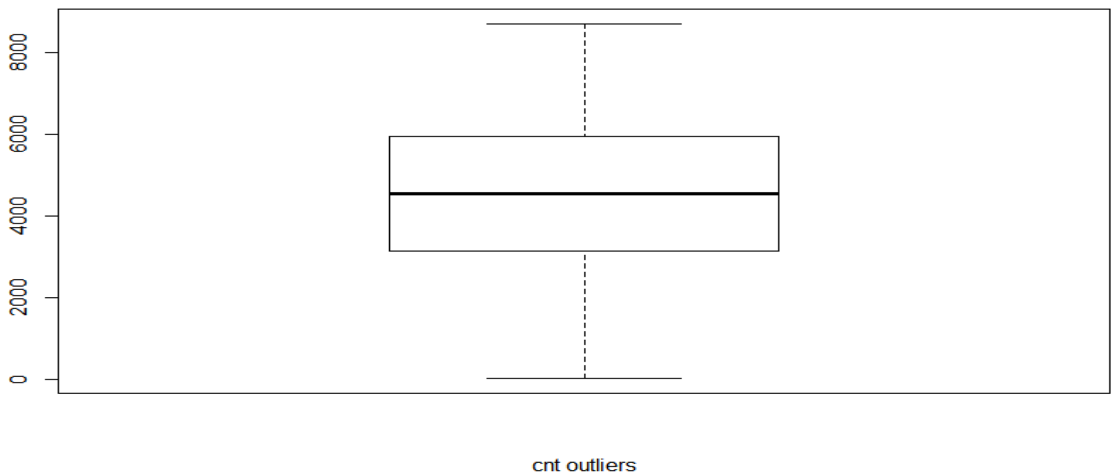


Figure 15C. Boxplots of 'casual', 'registered', 'cnt'

From these boxplots, I can see 'hum', 'windspeed', 'casual' variables contain outliers.
So, I created a dataframe ‘out_data’ containing these 3 variables & saved the feature names of ‘out_data’ into ‘cnames’ list. Then I assigned ‘na’ to all the outliers in the ‘day’ dataset in their respective variables. Then, I extracted all the ’na’ values in the ‘day’ dataset into ‘missing_val_new’ dataframe. From here, we can see ‘hum’, ‘windspeed’ & ‘casual’ contain 2, 13 & 44 ‘na’ values.

Now, I chose not to drop the observations containing outliers as I wanted to save the information, so I imputed these ‘na’ values. In order to select the best performing imputation method, at first, I took a known value from the ‘day’ dataset (2^{nd} row in ‘windspeed variable) & assigned ‘na’ to it, effectively making it a missing value. Then I imputed its value using mean, median & knn imputation method & found out that the predicted value I got using the mean method is closest to the original value. So, I chose ‘mean’ as the best performing method for imputation.

But, we've to remember, that as the mean & median are constants for any given column, & knn will vary with different data points. I can get different best methods if I take another known value for method selection.

Then I reloaded the ‘day’ dataset & imputed the ‘na’s using mean & again checked for ‘na’ in the dataset.

Now I found zero ‘na’s. That means all the outliers have been successfully imputed in the ‘day’ dataset.

3.6. Feature Selection

I saw that there are 7 independent categorical variables, 6 independent numeric variables & 1 dependent numeric variable. I've used correlation analysis for numerical variables & I've used ANOVA & Chi squared test for categorical variables in order to check dependencies between them.

3.6.1. Correlation Analysis

At 1st, I saved all the numeric variables into 'num_var' list. Then, to understand which features from 'num_var' have multicollinearity in 'day' dataset, I generated a correlation plot using the 'corrgram' function depicting the relation between each 2 variables in the dataset.

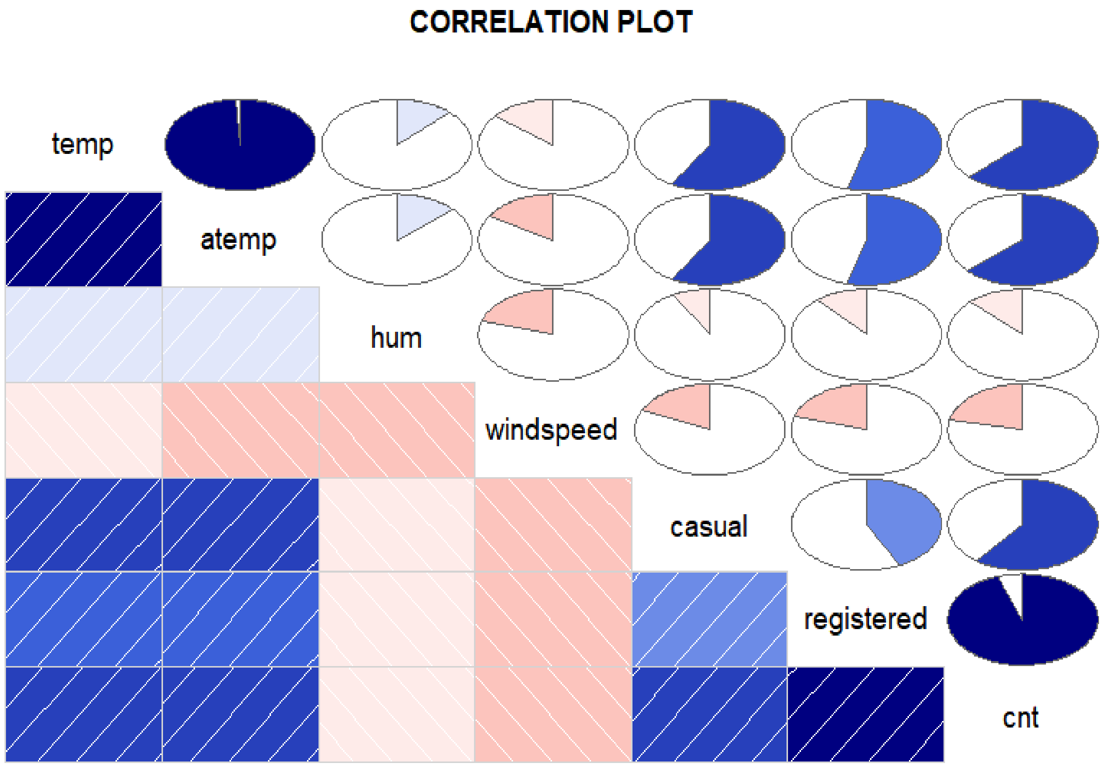


Figure 16. Correlation plot

From the plot, we can see there's high correlation between 'temp' & 'atemp'; so I'll remove 'atemp'. & we can see there's high correlation between 'registered' & 'cnt'; but as 'cnt' is target variable, I'll keep 'registered' variable.

3.6.2. Analysis of Variance (ANOVA) Test

I did ANOVA Test for the target variable with respect to all the categorical Variables in 'day'. Few prerequisites about ANOVA are-

- It is carried out to compare between each groups in a categorical variable.
- ANOVA only lets us know the means for different groups are same or not. It doesn't help us identify which mean is different.
- Hypothesis testing :
    - Null Hypothesis: mean of all categories in a variable are same.
    - Alternate Hypothesis: mean of at least one category in a variable is different.
- If p-value is less than 0.05 then we cannot accept the null hypothesis.
- And if p-value is greater than 0.05 then we accept the null hypothesis.

From the ANOVA results, I can see that the p-value is

- less than 0.05 for season
- less than 0.05 for weathersit
- less than 0.05 for yr
- less than 0.05 for mnth
- greater than 0.05 for weekday
- greater than 0.05 for holiday
- greater than 0.05 for workingday

So, I accepted the null hypothesis for weekday, holiday & workingday, saying that the means of all categories in these variables are same. &, I couldn't accept the null hypothesis for season, weathersit, yr & mnth, saying that the means of all categories in these variables are not same.

However, as ANOVA doesn't specify which group means are different, we can't conclude from the test results about which categorical variables I should remove.

3.6.3. Chi squared Test of Independence

At first, I saved all the categorical features in ‘cat_var’ dataframe. Then, I did Chi squared test of independence to select relevant features out of all the features in ‘cat_var’.

Few prerequisites about Chi squared test are-

- Hypothesis testing :
  - Null Hypothesis: 2 variables are independent
  - Alternate Hypothesis: 2 variables are not independent
- If p-value is less than 0.05 then we cannot accept the null hypothesis.
- And if p-value is greater than 0.05 then we accept the null hypothesis.

variables which are highly dependent on each other based on p-values are:

- "season" & "weathersit" with a p-value of 0.0211793

- "mnth" & "season" with a p-value of 0

- "mnth" & "weathersit" with a p-value of 0.01463711

- "holiday" & "weekday" with a p-value of 8.567055e-11

- "holiday" & "workingday" with a p-value of 4.033371e-11

- "weekday" & "workingday" with a p-value of 6.775031e-136

So I will remove season,holiday.

Finally, I dropped the 'atemp', 'season' &'holiday' variables from the ‘day’ dataset as these features were dependent on other features.

3.7. Model Development

3.7.1. Model Selection

As the dataset ‘day’ contains a target variable (cnt), so I’ll have to use supervised machine learning algorithms. Now, the dependent variable can fall in any of the four categories:

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

As the dependent variable, in this case (cnt) is interval-scaled numeric variable; I’ll have to do regression analysis to generate a model. I’m choosing these machine learning algorithms to solve this problem-

d) Decision tree
e) Random Forest
f) Linear Regression

As feature scaling does not have any impact on these algorithms, I chose not to do feature scaling.

3.7.2. Different Machine Learning Algorithms

a) Decision Tree Regressor

At first, I divided the 'day' dataset into 'train' and 'test' dataset using 'train_test_split' function from scikit learn library (train containing 80% of the data). Here 'train' contains 584 observations & 'test' contains 147 observations. Then, using 'rpart' function (from rpart library) on the 'train' dataset, I generated the decision tree model, named 'dt_model'. Then I applied this model on the independent variables from the 'test' dataset & generated the predictions, which I stored into the 'dt_predictions'. Then I plotted the 'dt_predictions (predicted values)' & 'cnt' variables from 'test' dataset (actual values) together.

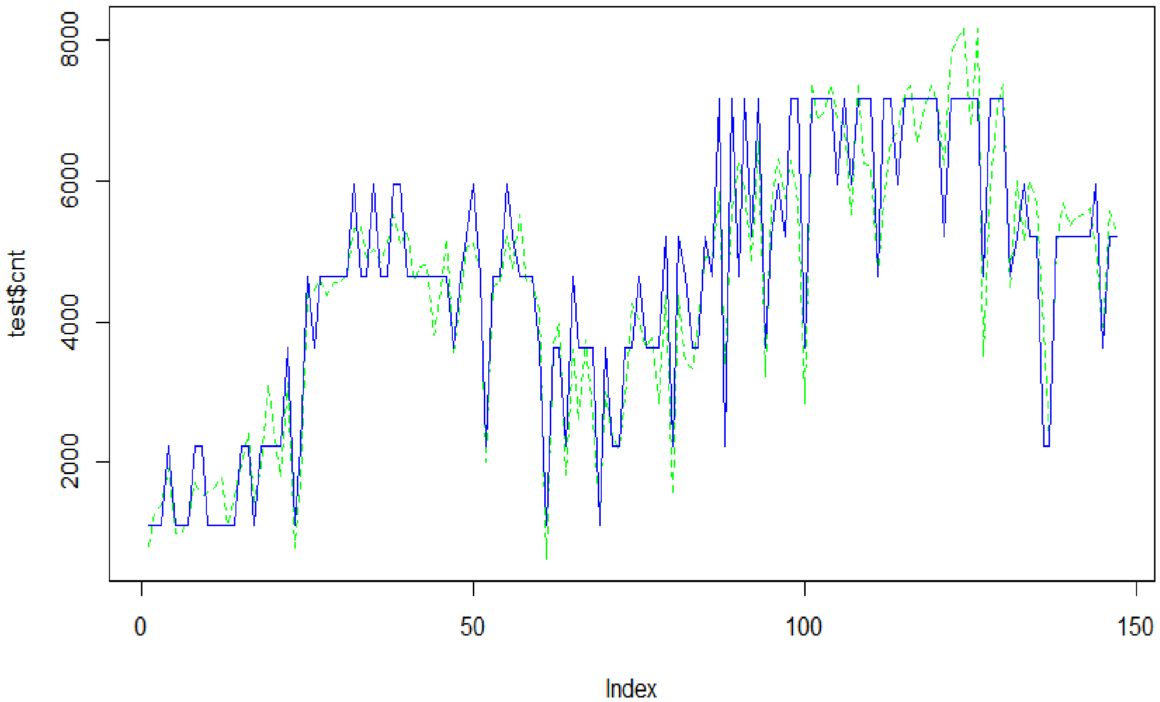


Figure 17. predicted values vs. actual values by decision tree

Here, green represents the actual values & blue represents the predicted values.

Then I derived 3 error metric functions, namely MAE, MAPE & RMSE & calculated the error rate of the decision tree model using these metrics. I found out that, MAE= 492.0515, MAPE= 13.496% & RMSE= 622.733.

b) Random Forest

First of all, I loaded the 'randomForest' library. Then, by using the 'randomForest' function from that library, I created the random forest model, 'rf_model'. Then I applied this model on the independent variables from the 'test' dataset & generated the predictions, which I stored into the 'rf_predictions'. Then I plotted the 'rf_predictions (predicted values)' & 'cnt' variables from 'test' dataset (actual values) together.

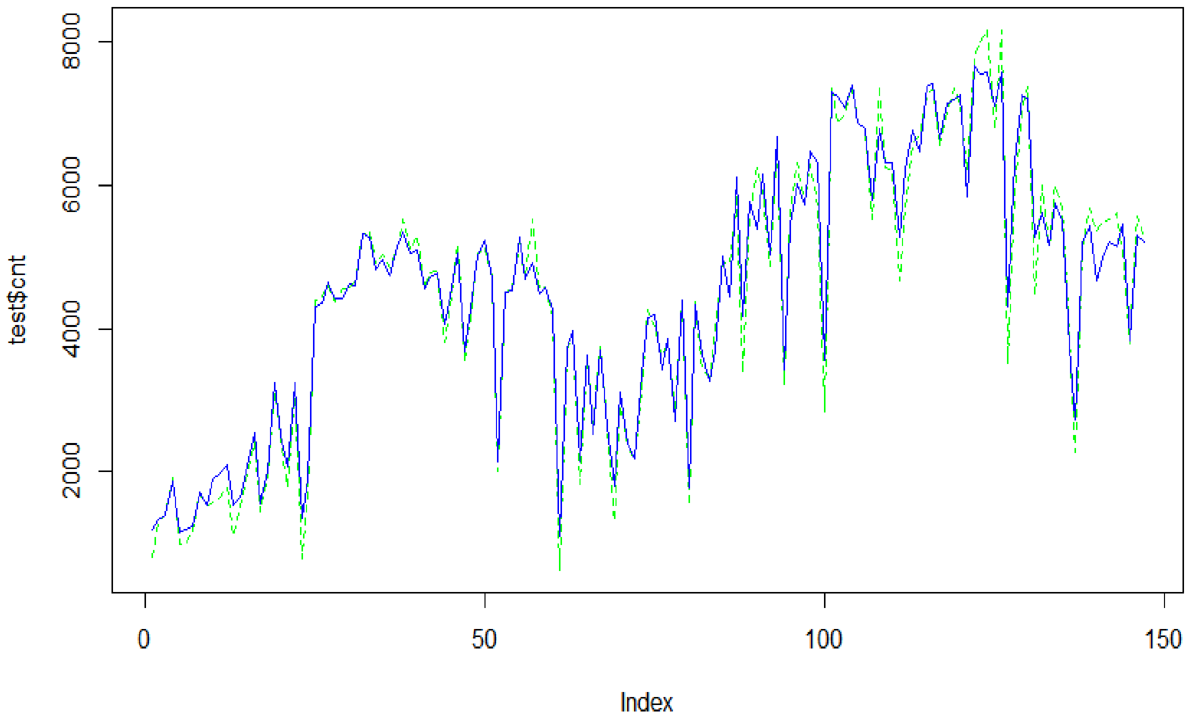


Figure 18. predicted values vs. actual values by random forest

Here, green represents the actual values & blue represents the predicted values.

Then I calculated the error rate of the random forest model using these metrics. I found out that, MAE= 202.4917, MAPE= 6.697316% & RMSE= 282.5122.

c) Linear Regression

Thereafter, I generated the linear regression model (lr_model) using the 'lm' function from the train dataset. Then I checked the summary of the model; then I applied this model on the independent variables from the 'test' dataset & generated the predictions, which I stored into the 'lr_predictions'. Then I plotted the 'lr_predictions (predicted values)' & 'cnt' variables from 'test' dataset (actual values) together.

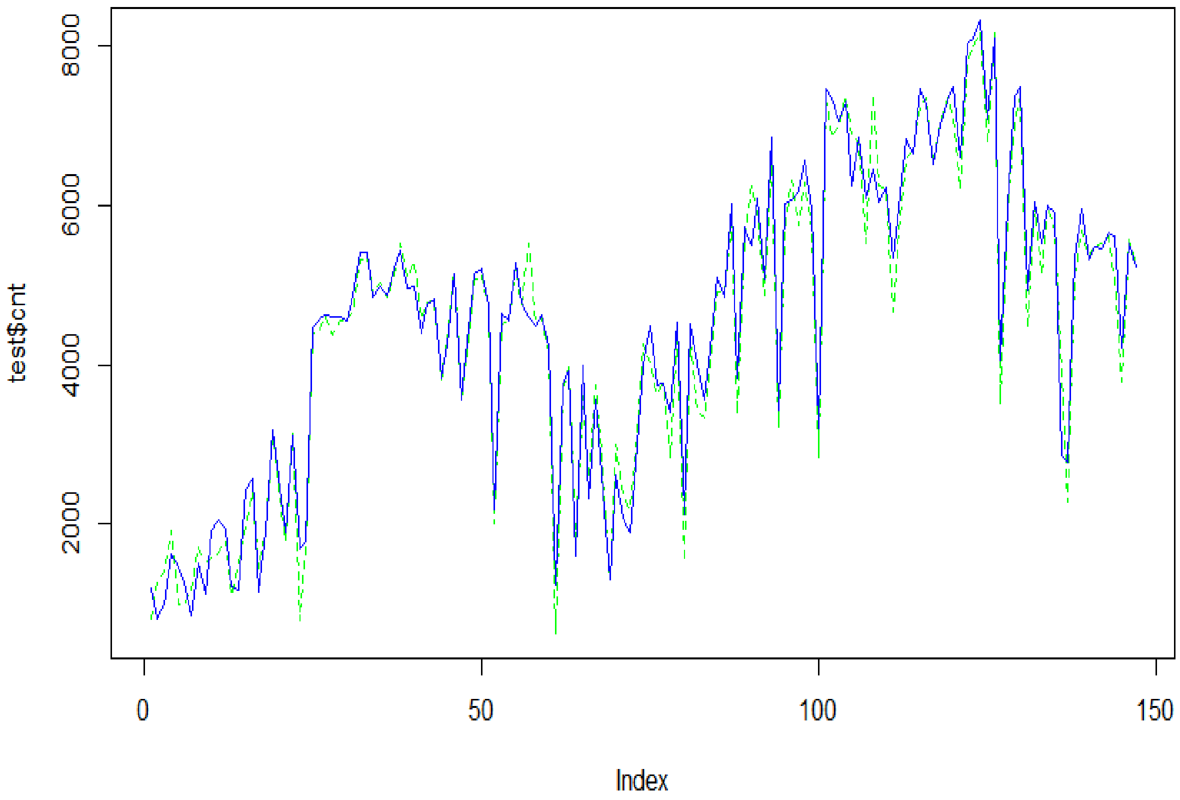


Figure 19. predicted values vs. actual values by linear regression

Here, green represents the actual values & blue represents the predicted values.

Then I calculated the error rate of the linear regression model using these metrics. I found out that, MAE= 228.7196, MAPE= 8.585146% & RMSE= 311.651.

3.8. Conclusion

3.8.1. 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 of Bike rental prediction, Computation Efficiency, isn't that significant. Therefore I will use Interpretability and 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. I've used a total of 3 types of error metrics for the evaluation of different models. Those are-

(i) Mean Absolute Error (MAE)
(ii) Mean Absolute Percentage Error (MAPE)
(iii) Root Mean Square Error (RMSE)

However, as we're dealing with a non-time series data, RMSE is not recommended in this case. And, out of MAE & MAPE, MAPE is more Interpretable; so, I chose MAPE as the most significant error metric in this case.

3.8.2. Most Preferable Model Selection

Based on the Mean Absolute Percentage Error (MAPE), it can be observed that the Random Forest model, 'rf_model' is giving least amount of error.

That's why I chose Random Forest as the best model for this dataset.

3.9. Employing Model to predict new cases

I've selected one observation (4th row) from the test data, which I shall use as a new sample input & predict the output & at last I'll see how well the model is performing (by checking the error rate).

Few instructions to use this model on a new dataset containing all the variables present in the original dataset-

(i) Drop 'instant','dteday','season','holiday' & 'atemp' variables as these are statistically insignificant
(ii) Convert 'yr','mnth','weekday','workingday' & 'weathersit' variables to category type

At first, I checked the actual value of the 'cnt' variable in the selected observation & found it to be 1927 & took a note of it. Then, applied my 'rf_model' on the independent variables of that observation, thus predicted the value of the target variable, which was found to be 1861.331. Then I calculated the 'MAPE' of the prediction, which was 3.407836%.

By getting such a low error rate, we can see the generated model is performing well with a new dataset.

3.10. Complete R Code

```r
rm(list = ls())
setwd("C:/R programs")
getwd()
day = read.csv(file = "day.csv", header = T, sep = ",", na.strings = c(" ", "", "NA"))
head(day)
str(day)
dim(day)
day$instant = NULL
day$dteday = NULL
dim(day)
day$season = as.factor(day$season)
day$yr = as.factor(day$yr)
day$mnth = as.factor(day$mnth)
day$holiday = as.factor(day$holiday)
day$weekday = as.factor(as.character(day$weekday))
day$workingday = as.factor(as.character(day$workingday))
day$weathersit = as.factor(day$weathersit)
str(day)
library(ggplot2)
ggplot(day)+
  geom_histogram(aes(x=cnt,y=..density..),
                 fill= "grey")+
  geom_density(aes(x=cnt,y=..density..))
ggplot(day)+
  geom_histogram(aes(x=temp,y=..density..),
                 fill= "grey")+
  geom_density(aes(x=temp,y=..density..))
ggplot(day)+
  geom_histogram(aes(x=atemp,y=..density..),
                 fill= "grey")+
  geom_density(aes(x=atemp,y=..density..))
ggplot(day)+
  geom_histogram(aes(x=hum,y=..density..),
                 fill= "grey")+
  geom_density(aes(x=hum,y=..density..))
ggplot(day)+
  geom_histogram(aes(x=windspeed,y=..density..),
                 fill= "grey")+
  geom_density(aes(x=windspeed,y=..density..))
ggplot(day)+
  geom_histogram(aes(x=casual,y=..density..),
                 fill= "grey")+
  geom_density(aes(x=casual,y=..density..))
ggplot(day)+
  geom_histogram(aes(x=registered,y=..density..),
                 fill= "grey")+
  geom_density(aes(x=registered,y=..density..))
ggplot(day, aes(x= temp,y=cnt)) +
  geom_point()+
  geom_smooth()
ggplot(day, aes(x= atemp,y=cnt)) +
  geom_point()+
  geom_smooth()
ggplot(day, aes(x= hum,y=cnt)) +
  geom_point()+
  geom_smooth()
```

```
ggplot(day, aes(x= windspeed,y=cnt)) +
  geom_point()+
  geom_smooth()
ggplot(day, aes(x= casual,y=cnt)) +
  geom_point()+
  geom_smooth()
ggplot(day, aes(x= registered,y=cnt)) +
  geom_point()+
  geom_smooth()
ggplot(day, aes(x=season, y=cnt)) +
  geom_boxplot()
ggplot(day, aes(x=yr, y=cnt)) +
  geom_boxplot()
ggplot(day, aes(x=mnth, y=cnt)) +
  geom_boxplot()
ggplot(day, aes(x=holiday, y=cnt)) +
  geom_boxplot()
ggplot(day, aes(x=weekday, y=cnt)) +
  geom_boxplot()
ggplot(day, aes(x=workingday, y=cnt)) +
  geom_boxplot()
ggplot(day, aes(x=weathersit, y=cnt)) +
  geom_boxplot()
missing_val = data.frame(apply(day,2,function(x){sum(is.na(x))}))
missing_val$Columns = row.names(missing_val)
names(missing_val)[1] =  "Missing_Value"
missing_val = missing_val[order(-missing_val$Missing_Value),]
row.names(missing_val) = NULL
missing_val = missing_val[,c(2,1)]
write.csv(missing_val, "Miising_val.csv", row.names = F)
boxplot(day$temp, xlab="temp outliers")
boxplot(day$atemp,xlab = "atemp outliers")
boxplot(day$hum,xlab ="hum outliers")
boxplot(day$windspeed,xlab ="windspeed outliers")
boxplot(day$casual,xlab ="casual outliers")
boxplot(day$registered,xlab ="registered outliers")
boxplot(day$cnt,xlab ="cnt outliers")
out_data = day[c('hum','windspeed','casual')]
cnames = colnames(out_data)
for(i in cnames){
  #print(i)
  val = day[,i][day[,i] %in% boxplot.stats(day[,i])$out]
  print(length(val))
  day[,i][day[,i] %in% val] = NA
}
missing_val_new = data.frame(apply(day,2,function(x){sum(is.na(x))}))
sum(is.na(day$hum))
day$windspeed
day$hum[is.na(day$hum)] = mean(day$hum, na.rm = T)
day$windspeed[is.na(day$windspeed)] = mean(day$windspeed, na.rm = T)
day$casual[is.na(day$casual)] = mean(day$casual, na.rm = T)
sum(is.na(day))
num_var = c('temp', 'atemp', 'hum', 'windspeed','casual','registered','cnt')
library(corrgram)
corrgram(day[,num_var],
         order = F,  #we don't want to reorder
```

```
          upper.panel=panel.pie,
          lower.panel=panel.shade,
          text.panel=panel.txt,
          main = 'CORRELATION PLOT')
anova_season =(lm(cnt ~ season, data = day))
summary(anova_season)
anova_year =(lm(cnt ~ yr, data = day))
summary(anova_year)
anova_month =(lm(cnt ~ mnth, data = day))
summary(anova_month)
anova_holiday =(lm(cnt ~ holiday, data = day))
summary(anova_holiday)
anova_weekday =(lm(cnt ~ weekday, data = day))
summary(anova_weekday)
anova_workingday =(lm(cnt ~ workingday, data = day))
summary(anova_workingday)
anova_weathersit =(lm(cnt ~ weathersit, data = day))
summary(anova_weathersit)
cat_var = c('season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday','weathersit')
cat_day = day[,cat_var]
for (i in cat_var){
  for (j in cat_var){
    print(i)
    print(j)
    print(chisq.test(table(cat_day[,i], cat_day[,j]))$p.value)
  }
}
day$atemp = NULL
day$season = NULL
day$holiday = NULL
head(day)
set.seed(123)
train_index = sample(1:nrow(day), 0.8 * nrow(day))
train = day[train_index,]
test = day[-train_index,]
dim(train)
dim(test)
library(rpart)
dt_model = rpart(cnt ~ ., data = train, method = "anova")
dt_predictions = predict(dt_model, test[,-11])
plot(test$cnt,type="l",lty=2,col="green")
lines(dt_predictions,col="blue")
MAE = function(actual, pred){
  print(mean(abs(actual - pred)))
}
MAE(test[,11], dt_predictions)
MAPE = function(actual, pred){
  print(mean(abs((actual - pred)/actual)) * 100)
}
MAPE(test[,11], dt_predictions)
RMSE = function(actual, pred){
  print(sqrt(mean((actual - pred)^2)))
}
RMSE(test[,11], dt_predictions)
library(randomForest)
rf_model = randomForest(cnt~., data = train, ntree = 500)
```

```
rf_predictions = predict(rf_model, test[,-11])
plot(test$cnt,type="l",lty=2,col="green")
lines(rf_predictions,col="blue")
MAE(test[,11], rf_predictions)
MAPE(test[,11], rf_predictions)
RMSE(test[,11], rf_predictions)
lr_model = lm(formula = cnt~., data = train)
summary(lr_model)
lr_predictions = predict(lr_model, test[,-11])
plot(test$cnt,type="l",lty=2,col="green")
lines(lr_predictions,col="blue")
MAE(test[,11], lr_predictions)
MAPE(test[,11], lr_predictions)
RMSE(test[,11], lr_predictions)
predict(rf_model,test[4,-11])
test[4,11]
MAPE(1927, 1861.331)
```