2021

Project Report

PREDICATION OF BIKE RENTAL COUNT BASED ON THE ENVIRONMENTAL AND SEASONAL SETTINGS.

SREEJITH SUKUMARAN

Contents

INTRODUCTION:

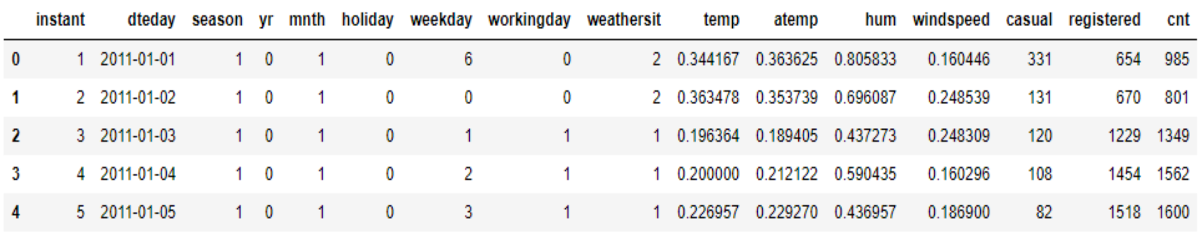
Problem Statement:

The project is to predict the bike rental count on daily basis, from the historical data, based on the environmental and seasonal settings.

These predicted values will help the business to meet the demand on those particular days by maintaining the amount of bike supply.

Data Set:

1) day.csv
   We will load this data as df. The Head of the data is shown 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

Exploratory data analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, spot anomalies, test hypotheses, and check assumptions with the help of summary statistics and graphical representations.

The given dataset contains 16 variables and 731 observations. The "cnt" is the target variable and remaining all other variables are the independent variables. Following are the findings:

Data description:

Size of day.csv: - 731 rows,16 Columns (including dependent variable) Below mentioned is a list of all the variable names with their meanings:

Variables	Description
instant	Serial Number/ index number
dteday	Date
season	Season (1:spring, 2:summer, 3:fall, 4:winter)
yr	Year (0: 2011, 1:2012)
mnth	Month (1 to 12)
holiday	Holiday (0:not holiday, 1:holiday)

weekday	Day of the week (0 to 6: Sunday to Saturday)
workingday	If the day is neither weekend nor holiday is 1, otherwise is 0.
weathersit	Weather situation 1: Clear, Few clouds, 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 from 100 (max)
windspeed	Normalized wind speed. The values are divided from 67(max)
casual	Count of casual users
registered	Count of registered users
cnt	Total count of users

Pre-processing:

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues.

1. Data type errors

We see that 'season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit' are factors/conditions and therefore we will convert the datatype to category. 'dteday' is datetime..

2. Missing values

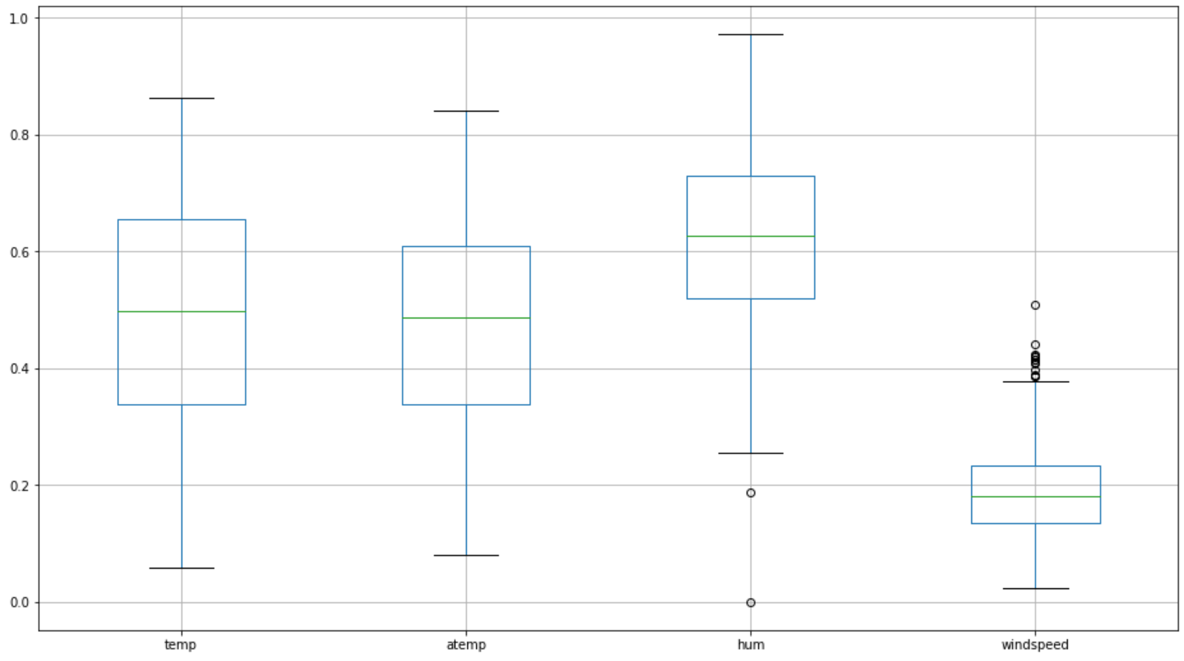
There are no missing values in the data.

Variable	No.of missing values
instant	0
dteday	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

3. Outliers

For outlier analysis, we used the boxplot technique:

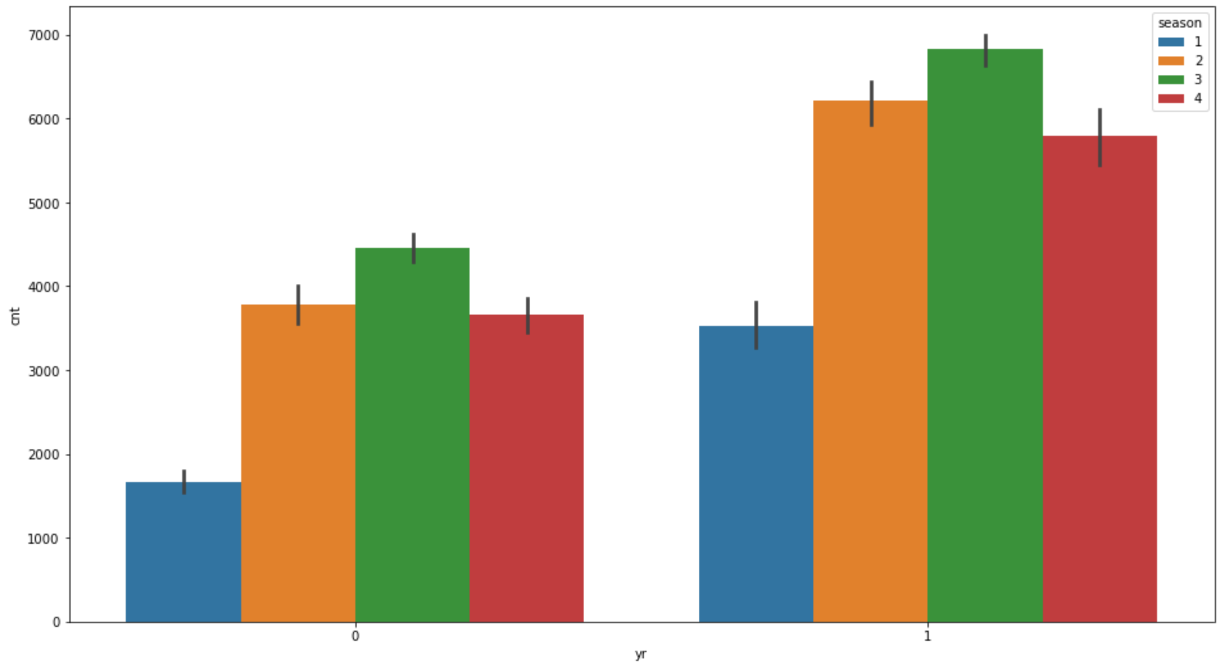


We see outliers in humidity, windspeed. These might be naturally occurring outliers or errors. We used the median to impute the outliers.

Relationship plots:

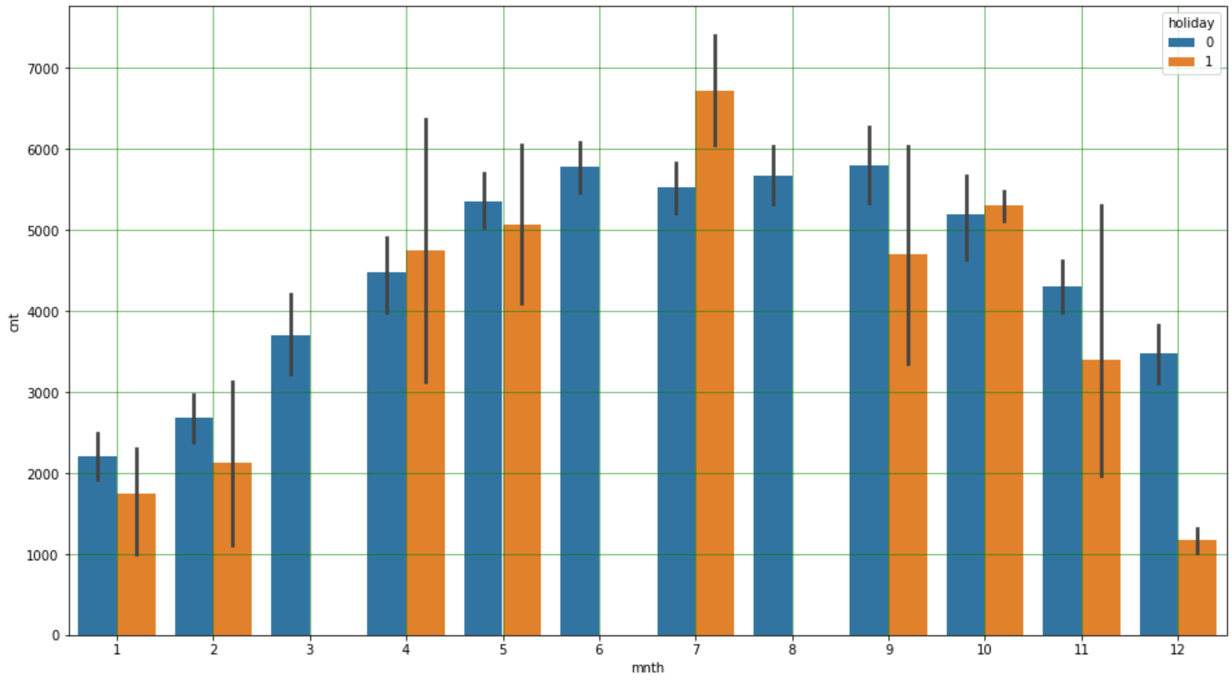
In this section, we looked into the relationship of the variables with the help of bar charts, scatterplots and correlation plots. The plots are shown below:

1. Seasonal change in cnt in 2011 and 2012

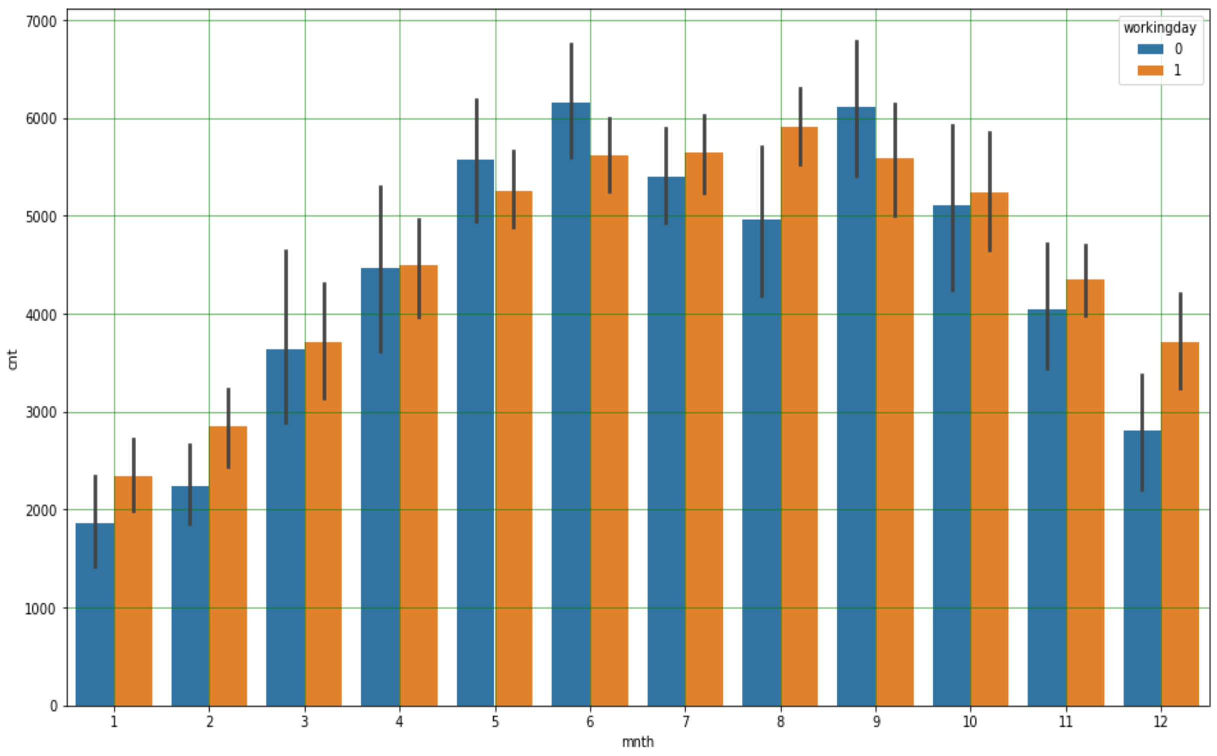


From the chart, it’s clear that the seasonal effect on the count in 2011 and 2012 is the same although the count increased in 2012. Most rides are in the fall and the least is in spring.

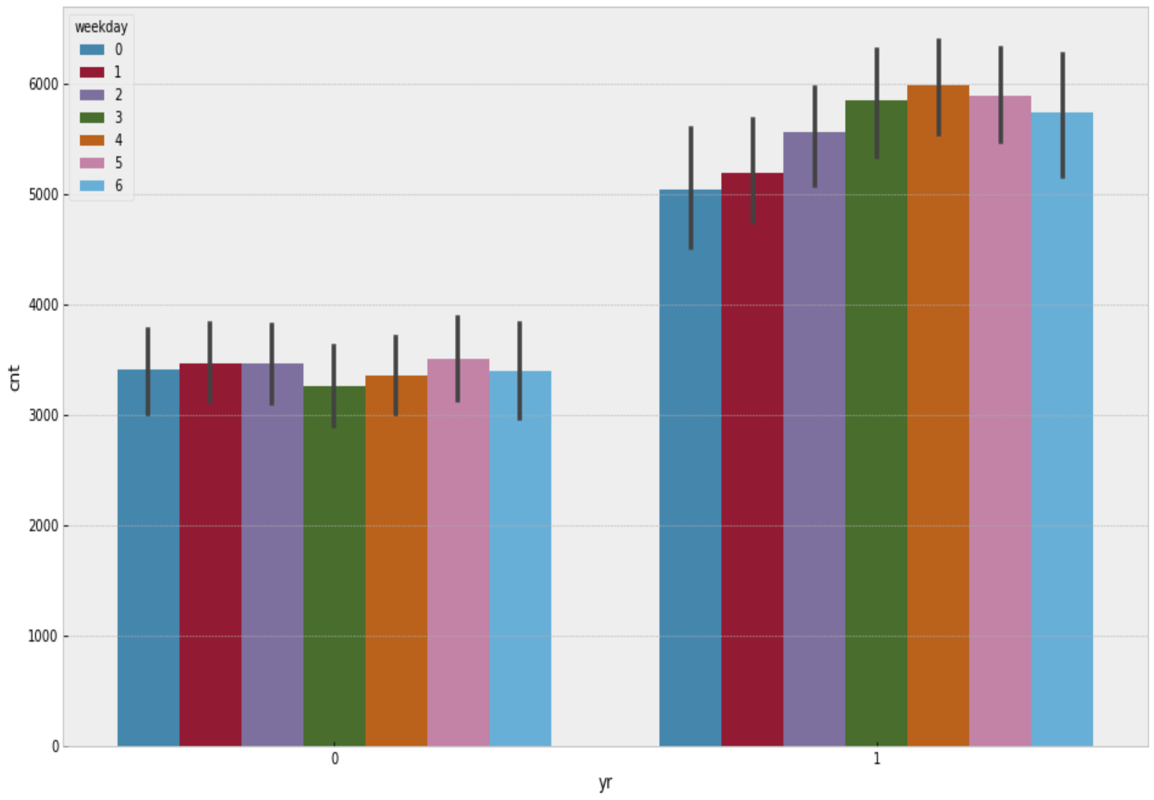
2. Monthly holiday vs total count



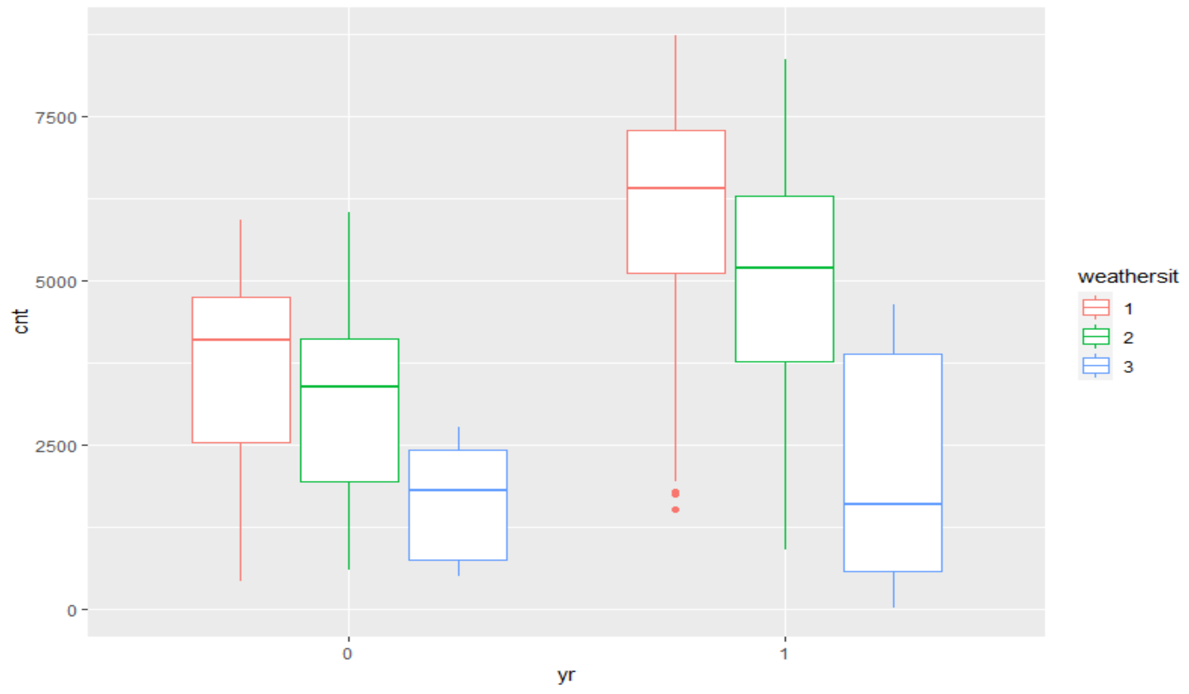
3. Monthly workingday vs total count



4. Weekday vs cnt

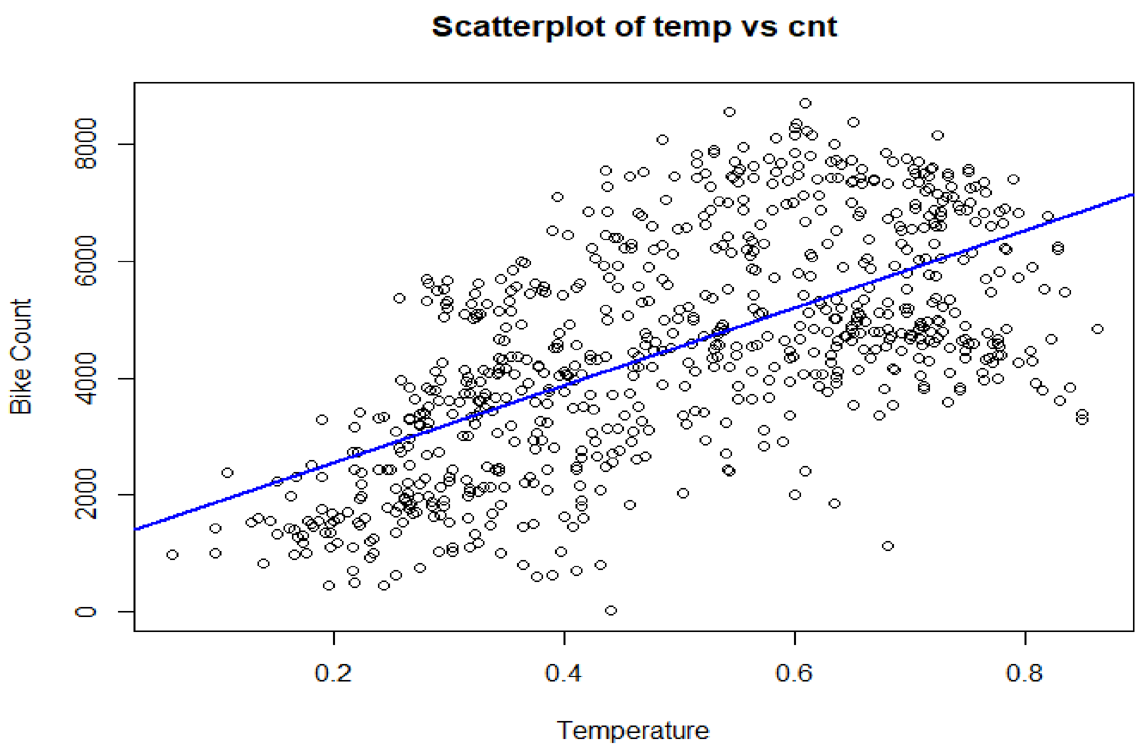


5. Weather vs total count



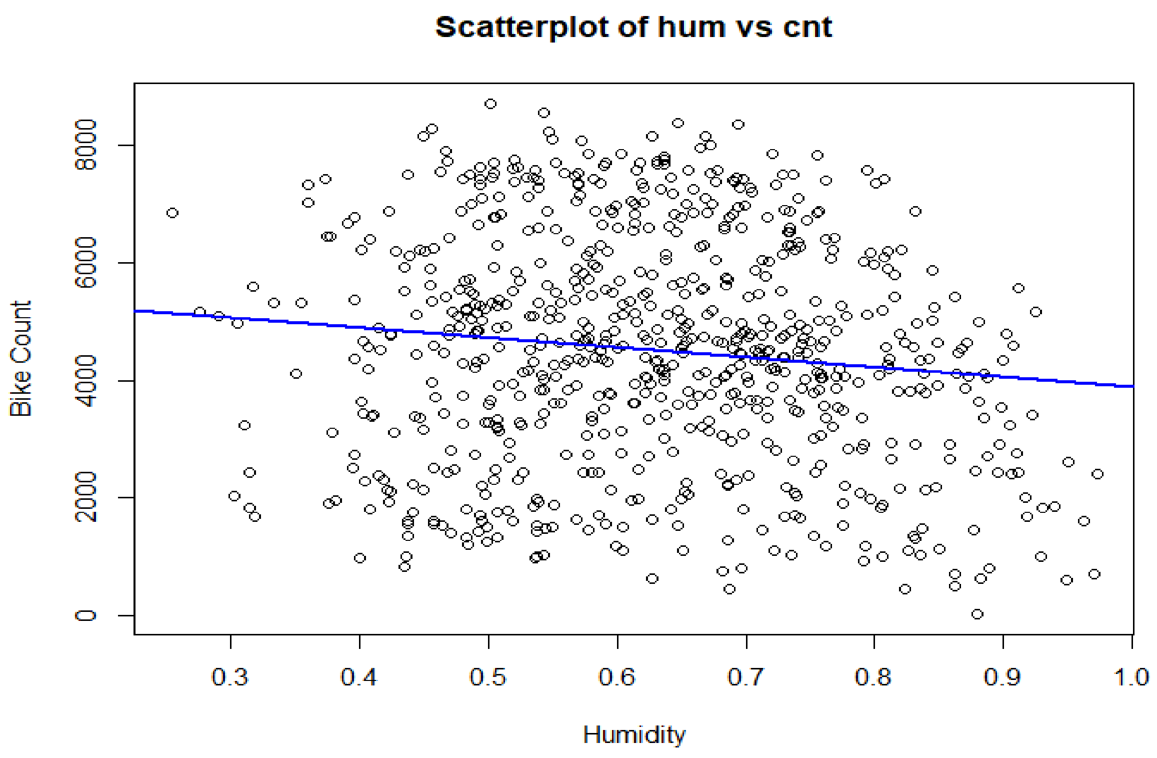
Despite the yearly increase in the count, the trend is the same here too. Clear weather has more rentals and harsh weather has fewer rentals.

6. Temperature vs total count



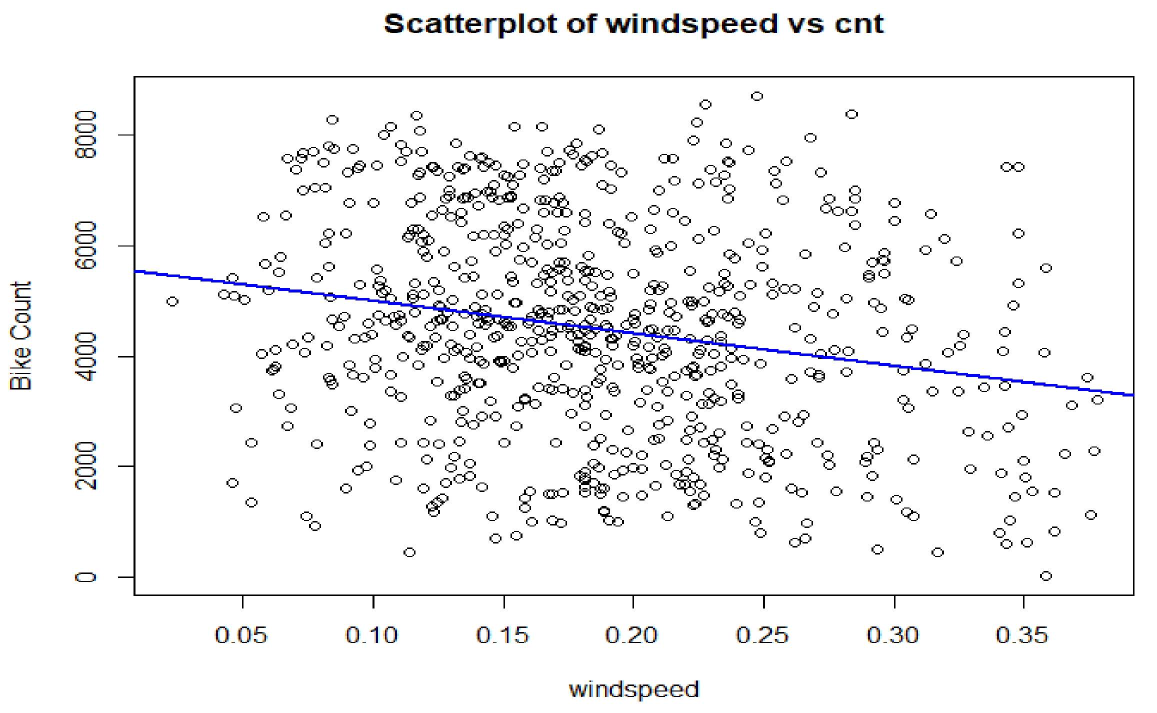
A linear relationship between temperature and count can be seen.

7. Humidity vs total count



As the humidity increases, there is a small decrease in the no. of rides. Most rides are in the mid-range.

8. Windspeed vs total count



Windspeed also hurts the count. More rides are in the mid-range.

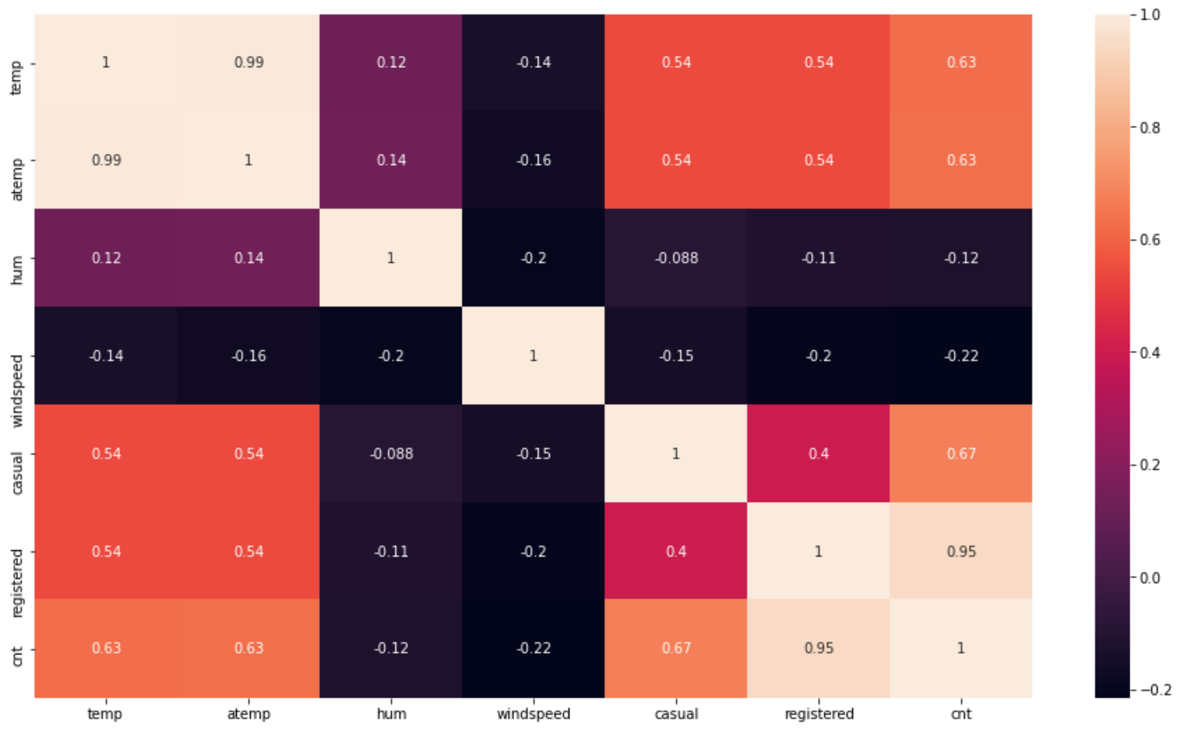
Feature Engineering

Feature engineering is the process of using domain knowledge to extract features from raw data. These features can be used to improve the performance of machine learning algorithms. Feature engineering can be considered as applied machine learning itself.

Feature Selection:

Index and dteday are not useful for the prediction of the bike count as the former is the serial number of the data and the latter is the date in ascending order and also we already have the useful data from the date like the year, month, etc.

Now, let's look into the correlation matrix:



As expected, the temp and atemp are highly correlated. Also, the registered rides and total rides are highly correlated. This indicates that most of the rides are registered rather than casual. Let's concentrate on cnt and drop casual and registered as cnt is their sum and our target variable.

Hence, we will be dropping the following variables:

- instant
- dteday
- atemp
- casual
- registered

Therefore, the head of the data (df in R and python) will look like:

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	hum	windspeed	cnt
0	1	0	1	0	6	0	2	0.344167	0.805833	0.160446	985
1	1	0	1	0	0	0	2	0.363478	0.696087	0.248539	801
2	1	0	1	0	1	1	1	0.196364	0.437273	0.248309	1349
3	1	0	1	0	2	1	1	0.200000	0.590435	0.160296	1562
4	1	0	1	0	3	1	1	0.226957	0.436957	0.186900	1600

Feature Scaling

Machine learning algorithms like linear regression, logistic regression, neural network, etc. that use gradient descent as an optimization technique require data to be scaled.

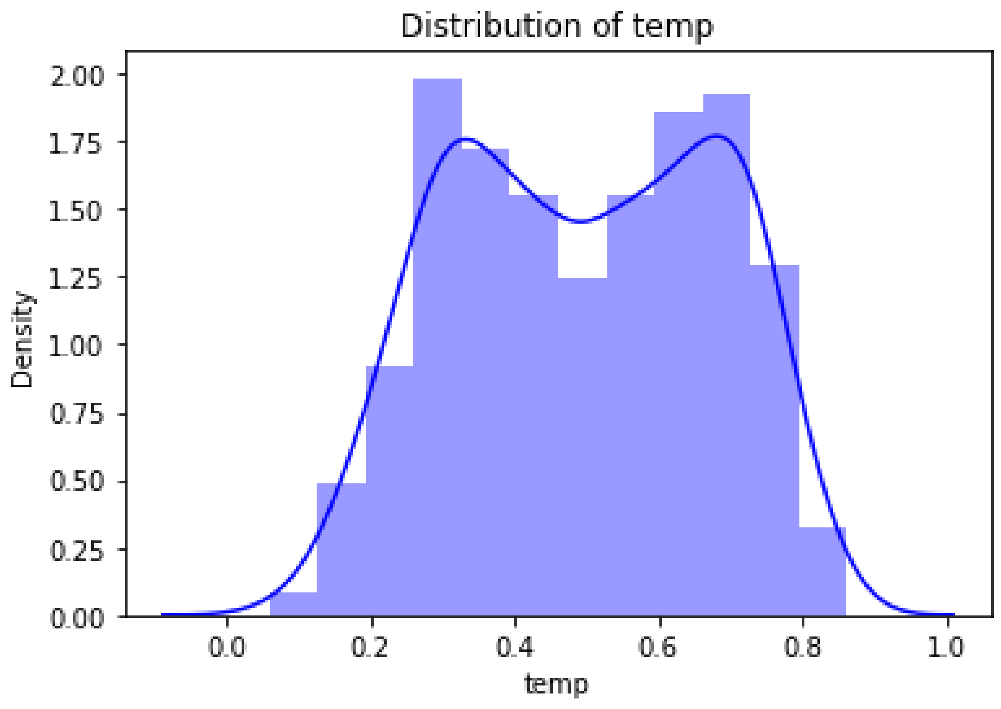
Distance algorithms like KNN, K-means, and SVM are most affected by the range of features. This is because behind the scenes they are using distances between data points to determine their similarity.

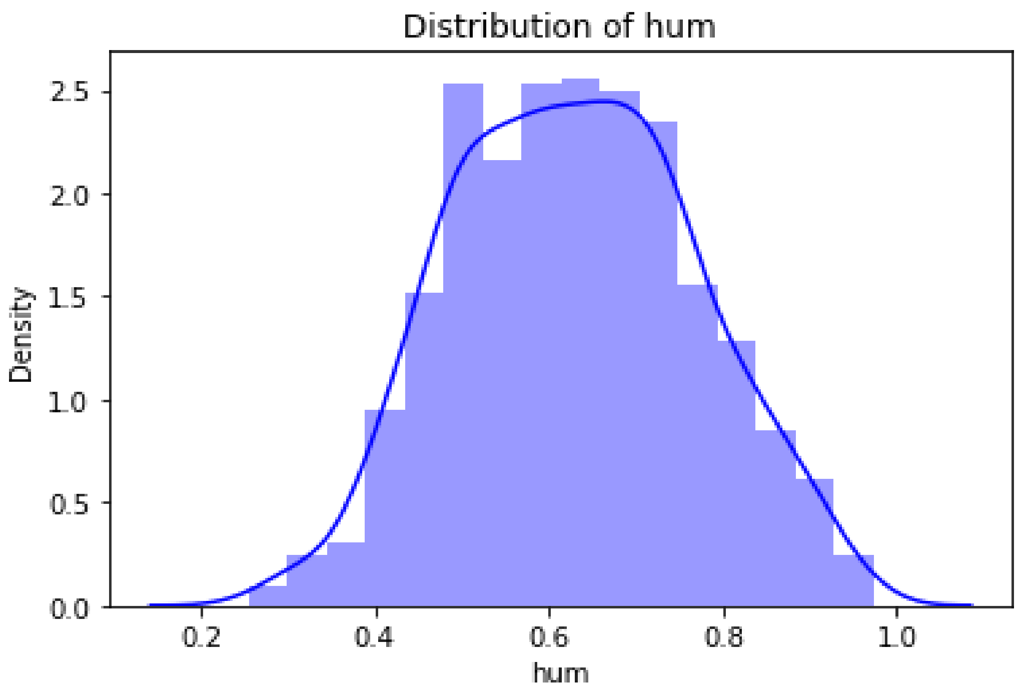
Tree-based algorithms, on the other hand, are fairly insensitive to the scale of the features.

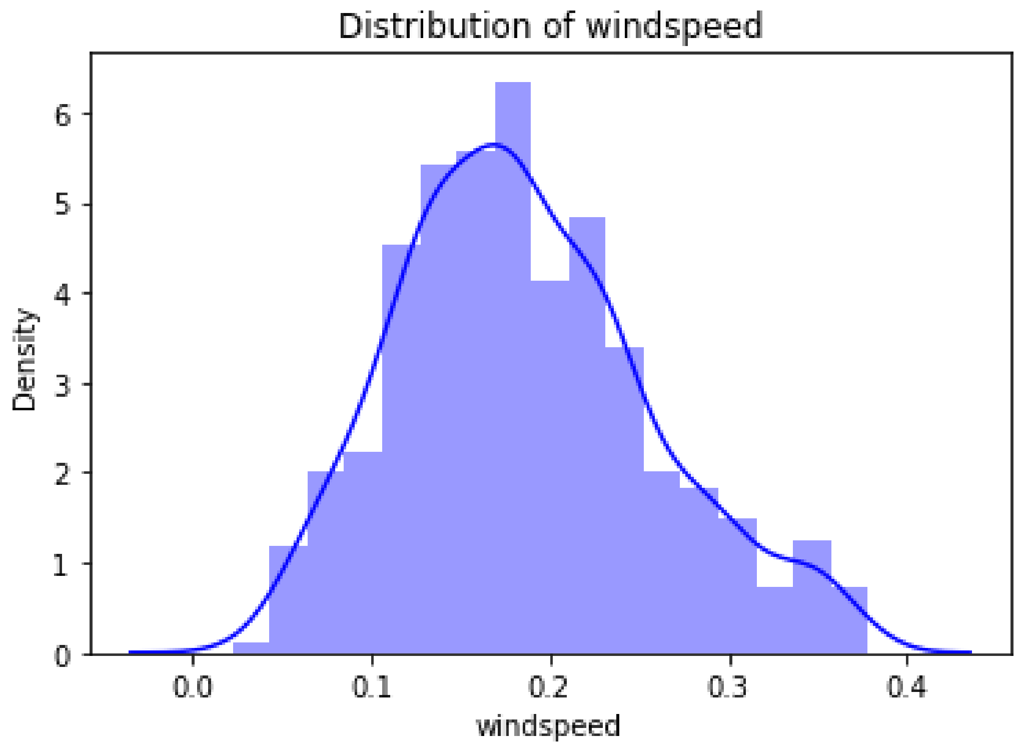
We have a regression problem and we will start from linear regression to test the most suitable model. Therefore, we must avoid skewness in our data.

We have to check the skewness of temp, hum and windspeed as they are continuous and the rest of the variables can be treated as categorical variables.

Even though the data is described as normalized, let’s confirm by plotting the distribution:







Now, the continuous variables appear to be normally distributed so we don't need to use feature scaling techniques like normalization and standardization for further scaling.

We can now go forward with modelling.

Model Building

As per industry standards, there are four categories of models that are derived by classifying the problem statement and goal of the project. These categories are:

o Forecasting

o Classification

o Optimization

o Unsupervised Learning

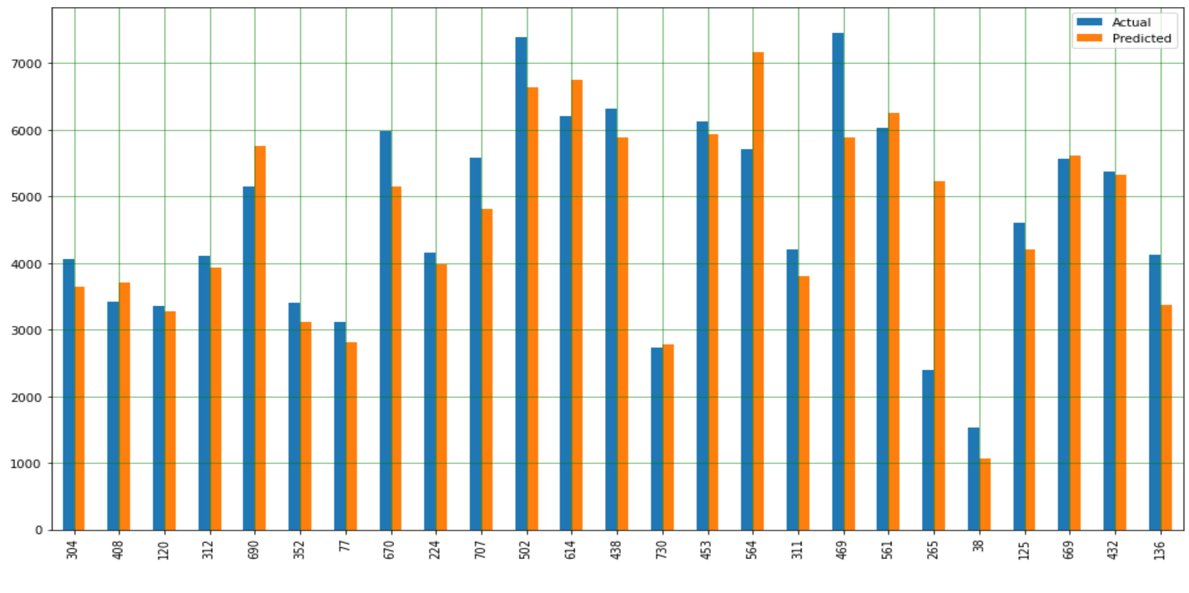
The process of selecting a precise model depends on our goal and the problem statement. In this project, the problem statement is to predict the bike rental count on daily basis, considering the environmental and seasonal settings. Thus, the problem statement is identified as a regression problem and falls under the category of forecasting, where we have to forecast numeric data or continuous variable for the target.

Before running any model, we will split our data into two parts which are train and test data. Here in our case, we have taken 75% of the data as our train data.

Linear Regression

The first approach we tried was Multiple Linear Regression, Multiple linear regression is the most common form of linear regression analysis. Multiple regression is an extension of simple linear regression. It is used as predictive analysis when we want to predict the value of a variable based on the value of two or more other variables. The variable we want to predict is called the dependent variable (or sometimes, the outcome, target or criterion variable.

In python, the predicted vs actual count looks as shown below:



Performance of the model in python:

```
Root Mean Squared Error: 771.8467914588674
Rsquared = 0.8429676265964835
Accuracy =  85.37 %
```

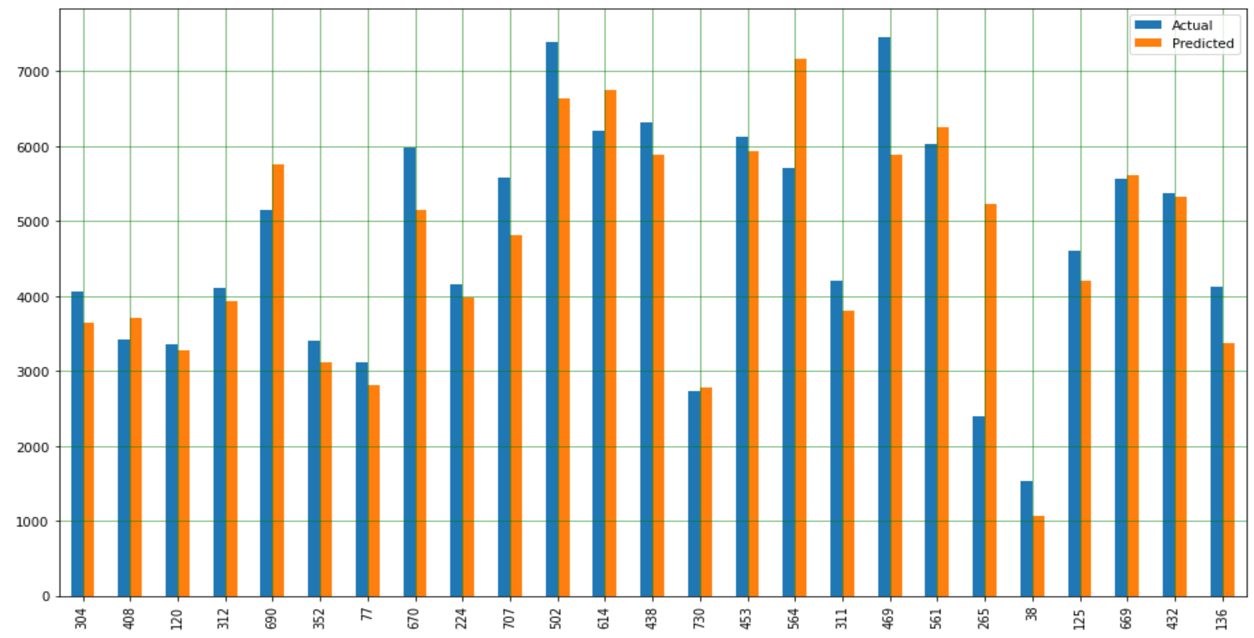
Performance of the model in R:

```
Root Mean Squared Error: 776.0476273
Rsquared = 0.8505296
Accuracy =  84.8 %
```

Decision Tree

The second approach tried was of the decision tree, Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree corresponds to the best predictor called the root node. Decision trees can handle both categorical and numerical data.

In python, the predicted vs actual count looks as shown below:



Performance of the model in python:

```
Root Mean Squared Error: 896.7417950528098
Rsquared = 0.7992748785386647
Accuracy =  82.71 %
```

Performance of the model in R:

```
Root Mean Squared Error: 887.7473305
Rsquared = 0.8061029
Accuracy =  81.4%
```

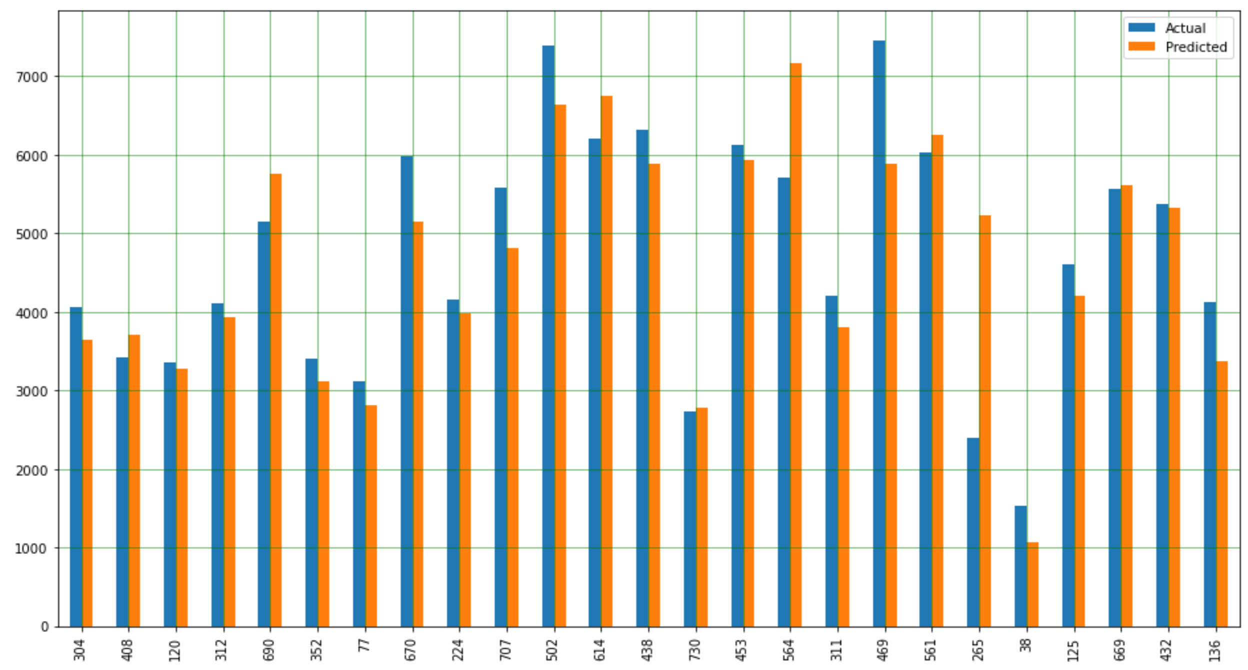
Random Forest

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as **bagging**. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.
Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

We need to approach the Random Forest regression technique like any other machine learning technique

- Design a specific question or data and get the source to determine the required data.
- Make sure the data is in an accessible format else convert it to the required format.
- Specify all noticeable anomalies and missing data points that may be required to achieve the required data.
- Create a machine learning model
- Set the baseline model that you want to achieve
- Train the data machine learning model.
- Provide an insight into the model with test data
- Now compare the performance metrics of both the test data and the predicted data from the model.
- If it doesn't satisfy your expectations, you can try improving your model accordingly or dating your data or use another data modelling technique.
- At this stage, you interpret the data you have gained and report accordingly.

In python, the predicted vs actual count looks as shown below:



Performance of the model in python:

```
Root Mean Squared Error: 639.9340959309582
Rsquared = 0.8920472951571087
Accuracy =  87.79 %
```

Performance of the model in R:

```
Root Mean Squared Error: 688.042105
Rsquared = 0.884561
Accuracy =  87.0 %
```

Model Evaluation

The key metrics used to compare and evaluate the model are as follows,

1. RMSE (Root Mean Square Error): is a frequently used measure of the difference between values predicted by a model and the values actually observed from the environment that is being modelled.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{mo\ del,i})^2}{n}}$$

2. R Squared(R^2): is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. In other words, we can say it explains how much of the variance of the target variable is explained.

In python the error metrics are:

SL	Model Type	RMSE	RSquared	Accuracy
1.	Linear Regression	771.85	0.84	85.37 %
2.	Decision Tree	896.74	0.80	82.71 %
3.	Random Forest	639.93	0.89	87.79 %

In R the error metrics are:

SL	Model Type	RMSE	RSquared	Accuracy
1.	Linear Regression	776.05	0.85	84.8 %
2.	Decision Tree	887.75	0.81	81.4 %
3.	Random Forest	688.04	0.88	87.0 %

By analysing the results obtained from both the approaches the best fit for the current problem statement is Random forest Regression as the metrics are better by the Random Forest model.

Now, let us tune the parameters of the Random Forest model to check if we can optimize the prediction further accurately

Model Optimization:

In Python, I tried to optimize the parameters by GridSearchCV and in R, I optimized by Random Search.

GridSearchCV in python:

In this grid search, I will try different combinations of RF hyperparameters.

Most important hyperparameters of Random Forest:

n_estimators = n of trees; max_features = max number of features considered for splitting a node; max_depth = max number of levels in each decision tree; min_samples_split = min number of data points placed in a node before the node is split; min_samples_leaf = min number of data points allowed in a leaf node; bootstrap = method for sampling data points (with or without replacement)

As for how I decided the numbers to try I simply followed the advice of Aurelion Geron (2017): 'When you have no idea what value a hyperparameter should have, a simple approach is to try out consecutive powers of 10 (or a smaller number if you want a more fine-grained search)'.

The code:

```
In [65]: from sklearn.model_selection import GridSearchCV

         param_grid = [
         {'n_estimators': [10, 50], 'max_features': [10, 30],
          'max_depth': [10, 100, None], 'bootstrap': [True,False]}
         ]

         grid_search_forest = GridSearchCV(RF_regressor, param_grid, cv=5, scoring='neg_mean_squared_error')
         grid_search_forest.fit(X_train, y_train)
```

```
Out[65]: GridSearchCV(cv=5, estimator=RandomForestRegressor(),
                      param_grid=[{'bootstrap': [True, False],
                                   'max_depth': [10, 100, None],
                                   'max_features': [10, 30], 'n_estimators': [10, 50]}],
                      scoring='neg_mean_squared_error')
```

```
In [66]: #now let's see how the RMSE changes for each parameter configuration
         cvres = grid_search_forest.cv_results_
         for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
             print(np.sqrt(-mean_score), params)
```

```
745.4903623642763 {'bootstrap': True, 'max_depth': 10, 'max_features': 10, 'n_estimators': 10}
717.3140221784334 {'bootstrap': True, 'max_depth': 10, 'max_features': 10, 'n_estimators': 50}
761.5800308367545 {'bootstrap': True, 'max_depth': 10, 'max_features': 30, 'n_estimators': 10}
729.4863061927525 {'bootstrap': True, 'max_depth': 10, 'max_features': 30, 'n_estimators': 50}
732.510296584402 {'bootstrap': True, 'max_depth': 100, 'max_features': 10, 'n_estimators': 10}
704.6070698946811 {'bootstrap': True, 'max_depth': 100, 'max_features': 10, 'n_estimators': 50}
771.3399720895575 {'bootstrap': True, 'max_depth': 100, 'max_features': 30, 'n_estimators': 10}
730.7473562433923 {'bootstrap': True, 'max_depth': 100, 'max_features': 30, 'n_estimators': 50}
730.5305789159521 {'bootstrap': True, 'max_depth': None, 'max_features': 10, 'n_estimators': 10}
704.6778520029889 {'bootstrap': True, 'max_depth': None, 'max_features': 10, 'n_estimators': 50}
763.7781592101239 {'bootstrap': True, 'max_depth': None, 'max_features': 30, 'n_estimators': 10}
729.5616553342094 {'bootstrap': True, 'max_depth': None, 'max_features': 30, 'n_estimators': 50}
760.9730653119469 {'bootstrap': False, 'max_depth': 10, 'max_features': 10, 'n_estimators': 10}
710.3747287174308 {'bootstrap': False, 'max_depth': 10, 'max_features': 10, 'n_estimators': 50}
855.1121849691941 {'bootstrap': False, 'max_depth': 10, 'max_features': 30, 'n_estimators': 10}
865.5349010408268 {'bootstrap': False, 'max_depth': 10, 'max_features': 30, 'n_estimators': 50}
754.863082881866 {'bootstrap': False, 'max_depth': 100, 'max_features': 10, 'n_estimators': 10}
698.8086173848193 {'bootstrap': False, 'max_depth': 100, 'max_features': 10, 'n_estimators': 50}
885.7009711157389 {'bootstrap': False, 'max_depth': 100, 'max_features': 30, 'n_estimators': 10}
876.3038464903207 {'bootstrap': False, 'max_depth': 100, 'max_features': 30, 'n_estimators': 50}
746.8047521726968 {'bootstrap': False, 'max_depth': None, 'max_features': 10, 'n_estimators': 10}
713.5814972394035 {'bootstrap': False, 'max_depth': None, 'max_features': 10, 'n_estimators': 50}
879.622853731457 {'bootstrap': False, 'max_depth': None, 'max_features': 30, 'n_estimators': 10}
876.4615198434738 {'bootstrap': False, 'max_depth': None, 'max_features': 30, 'n_estimators': 50}
```

```
In [67]: #find the best model of grid search
         grid_search_forest.best_estimator_
```

```
Out[67]: RandomForestRegressor(bootstrap=False, max_depth=100, max_features=10,
                               n_estimators=50)
```

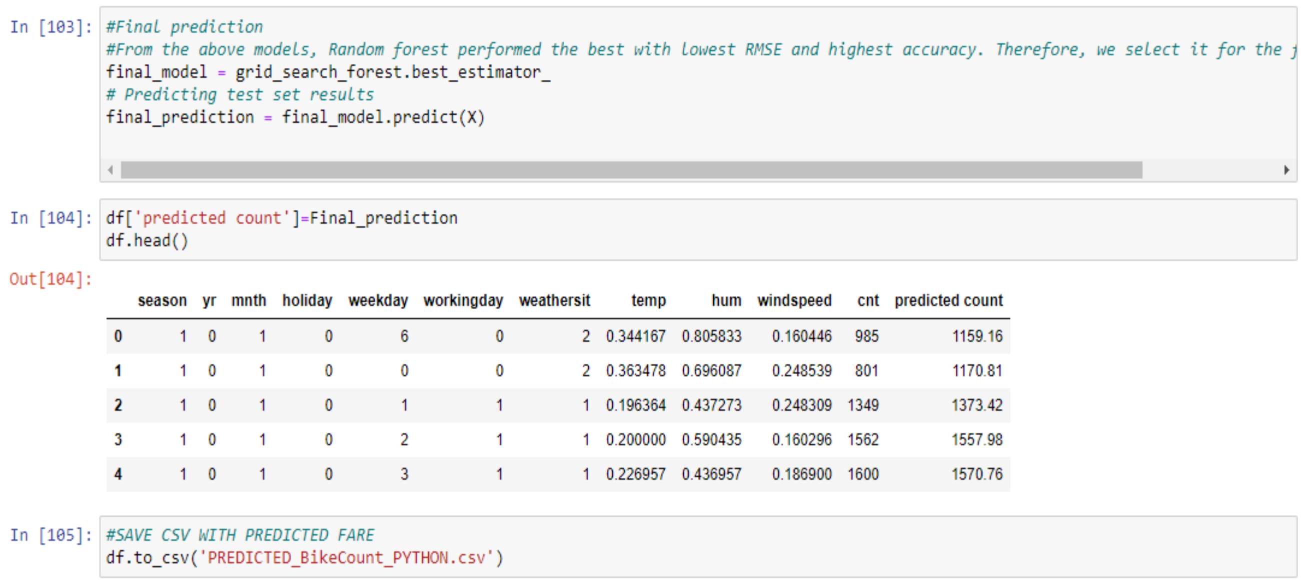
The best model of Grid Search is:

```
RandomForestRegressor(bootstrap=False, max_depth=100, max_features=10,
                      n_estimators=50)
```

The best model from grid-search has an accuracy of 100.0 %

The best model from the grid search has an RMSE of 0.25

Final prediction code:



```
In [103]: #Final prediction
          #From the above models, Random forest performed the best with lowest RMSE and highest accuracy. Therefore, we select it for the
          final_model = grid_search_forest.best_estimator_
          # Predicting test set results
          final_prediction = final_model.predict(X)
```

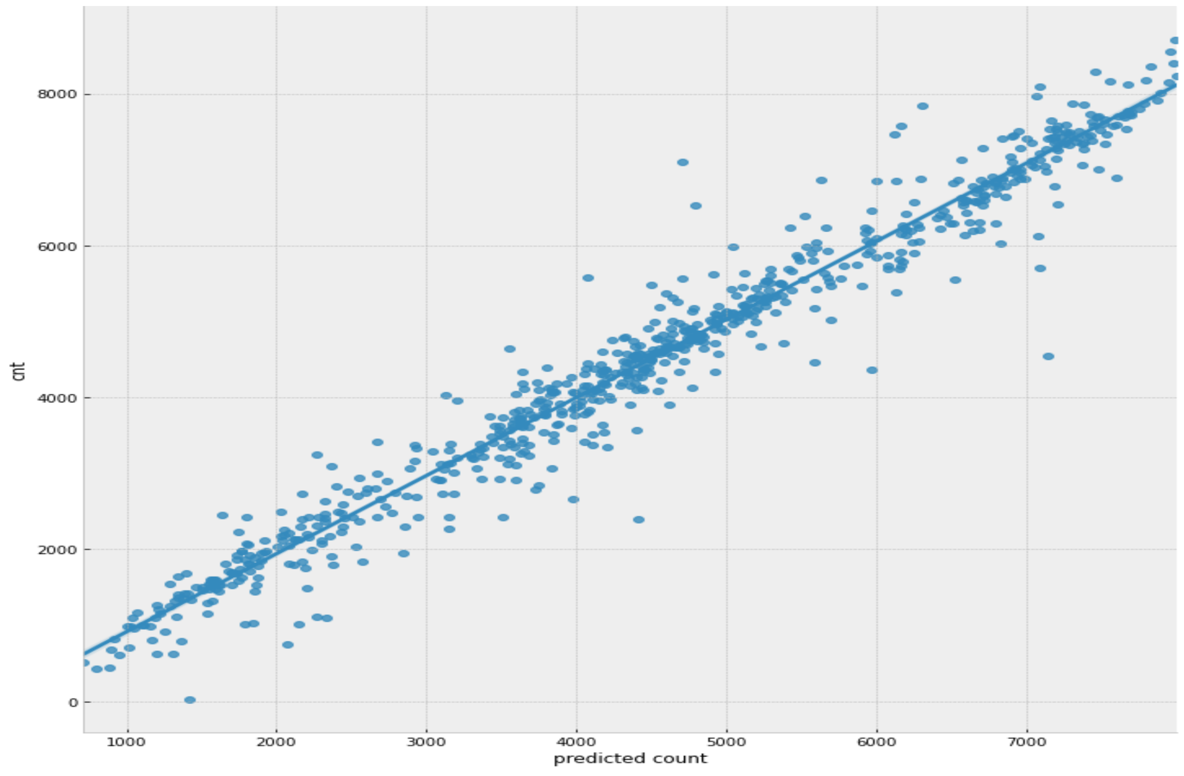
```
In [104]: df['predicted count']=Final_prediction
          df.head()
```

Out[104]:

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	hum	windspeed	cnt	predicted count
0	1	0	1	0	6	0	2	0.344167	0.805833	0.160446	985	1159.16
1	1	0	1	0	0	0	2	0.363478	0.696087	0.248539	801	1170.81
2	1	0	1	0	1	1	1	0.196364	0.437273	0.248309	1349	1373.42
3	1	0	1	0	2	1	1	0.200000	0.590435	0.160296	1562	1557.98
4	1	0	1	0	3	1	1	0.226957	0.436957	0.186900	1600	1570.76

```
In [105]: #SAVE CSV WITH PREDICTED FARE
          df.to_csv('PREDICTED_BikeCount_PYTHON.csv')
```

Graphical representation predicted bike count vs actual count in python:



Random Search tuning in R:

```
#RANDOM SEARCH
#library(doParallel)
# cores <- 7
# registerDoParallel(cores = cores)
#mtry: Number of random variables collected at each split. In normal equal square number columns.
mtry <- sqrt(ncol(X_train))
#ntree: Number of trees to grow.
ntree <- 3


control <- trainControl(method='repeatedcv',
                        number=10,
                        repeats=3,
                        search = 'random')

#Random generate 15 mtry values with tuneLength = 15
set.seed(1)
rf_random <- train(cnt ~ .,
                   data = df_dummy,
                   method = 'rf',
                   metric = 'RMSE',
                   tuneLength  = 15,
                   trControl = control)
print(rf_random)

#predict test data by rf_random model
rf_random_pred_test=predict(rf_random,X_test)

print(postResample(pred=rf_random_pred_test,obs = X_test$cnt))
# RMSE        Rsquared        MAE
#289.2255373   0.9807255     212.6599990


#Calculate MAPE

#library(MLmetrics)

rf_random_mape=MAPE(X_test$cnt,rf_random_pred_test)
print(rf_random_mape)
#MAPE=  0.06054491
#Error rate=6%
#Accuracy=94.0%
##WE GOT MAXIMUM ACCURACY AND MIN RMSE FROM THE TUNED MODEL###

###FINAL PREDICTION###
Final_prediction=predict(rf_random,X)
```

Random Search model:
Random Forest

731 samples
35 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 659, 658, 659, 658, 658, 659, ...
Resampling results across tuning parameters:

mtry RMSE   Rsquared   MAE

```
  1   1398.2467  0.8172217  1134.5211
 14    669.1282  0.8825592   466.8824
 18    671.8454  0.8814093   467.4246
 23    677.2957  0.8791448   471.8203
 33    689.0080  0.8746459   478.0482
 34    690.5367  0.8740872   479.6477

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 14.
```

The model accuracy is 94.0%

The model an RMSE of 297.0516922

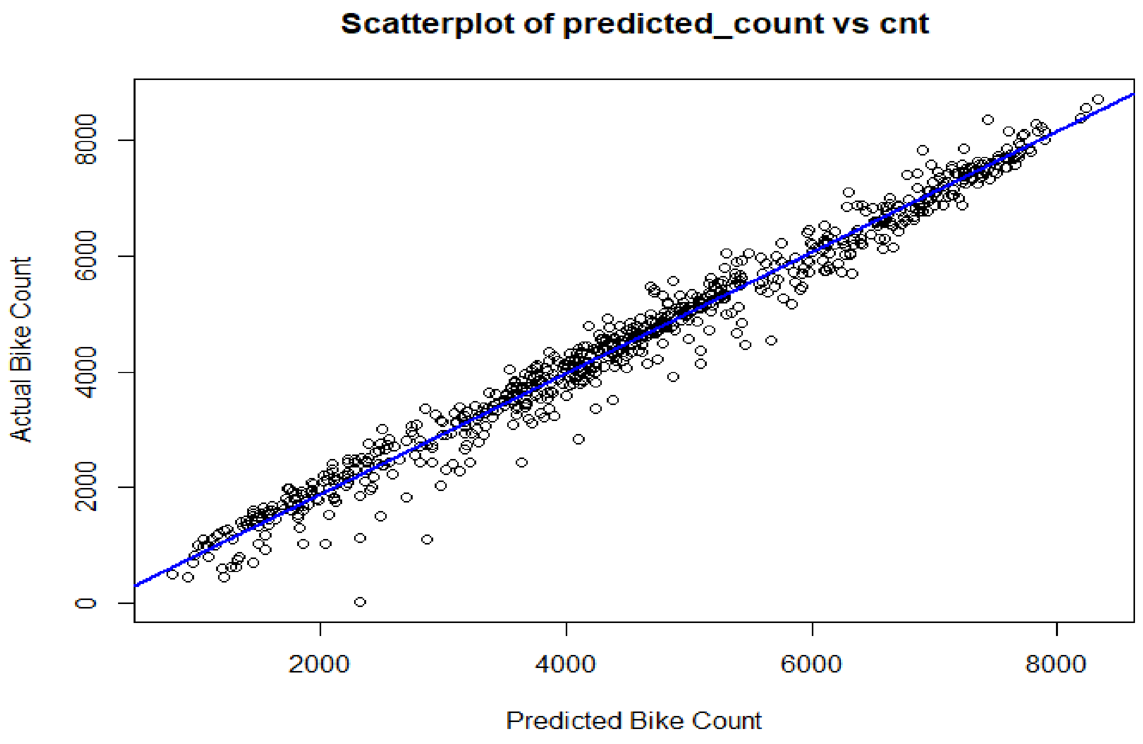
Final Prediction code:

```
###FINAL PREDICTION###
Final_prediction=predict(rf_random,X)

#LETS STORE AS A VARIABLE IN OUR DATASET FOR REFERENCE
df$predicted_count=round(Final_prediction)
head(df)

#SAVE PREDICTION
write.csv(df,file = "Bikecount_predicted_by_R.csv",row.names = F)
```

Graphical representation predicted bike count vs actual count in R:



END OF REPORT

References

https://learning.edwisor.com/

https://www.analyticsvidhya.com/

https://towardsdatascience.com/

https://stackoverflow.com/

https://datatofish.com/

https://en.wikipedia.org/wiki/Wikipedia