1. INTRODUCTION

1.1 About the Project

Bike sharing system is the new generation of traditional bike rentals where the whole process from membership, rental and return back has become automatic. Through these systems, the user is able to get a bike for rent a bike from a particular position and return back at another position. Currently, there are about over 500 bike-sharing programs around the world which are composed of over 500 thousands bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues.

Apart from interesting real-world applications of bike sharing systems, the characteristics of data being generated by these systems make them attractive for the research. Opposed to other transport services such as bus or subway, the duration of travel, departure, and arrival position is explicitly recorded in these systems. This feature turns the bike sharing system into a virtual sensor network that can be used for sensing mobility in the city. Hence, it is expected that most of the important events in the city could be detected via monitoring these data.

The objective is to predict the number of bikes went out on rent provided the data which includes various features.Various python packages like numpy, pandas, seaborn and standardScaler were used to explore and do feature engineering on data. RandomForestRegression and DecisionTreeRegression model built from scratch and for each of the model various metrics such as mean_squared_error and r2_score were analyzed to arrive at the conclusion.

1.2 Company Profile

Company Name: **ROOT - IT Learning Centre**

Address : No:86, Madurai Road, II Floor-BKG Building,
Tiruchirappalli 620008,

Phone: 9790636324

Email: root.anand@gmail.com

External Guide Name: Dr.P.Anand Kumar, MCA., Ph.D.,

2. PROBLEM STATEMENT

2.1 Business Problem

Generally, in bicycle sharing system, it is very important that the administrators should know how many cycles will be needed in each bicycle station, knowing this count enables them to arrange a proper number of cycles at the stations and decide whether a particular station needs to have an extra number of bicycle stands. So in this Project work, there is various prediction algorithms i.e. random forests, decision trees, gradient boosting machines. This Project work focuses on which algorithm can work better for the real world problem of bicycle sharing demand prediction.

2.2 Machine Learning

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

Machine learning algorithms are used in a wide variety of applications, such as email filtering, detection of network intruders, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers.

The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

2.3 Regression

Regression is the machine learning models which predicts the numerical data values. Regression analysis is a form of predictive modeling technique which investigates the relationship between a dependent (target) and independent variables**.**

This technique is used for forecasting, time series modeling and finding the causal effect relationship between the variables. Regression analysis is an important tool for modeling and analyzing data. It fit a curve/line to the data points, in such a manner that the differences between the distances of data points from the curve or line are minimized.

1. It indicates the significant relationships between the dependent variable and independent variable.
2. It indicates the strength of the impact of multiple independent variables on a dependent variable.

Regression analysis also allows us to compare the effects of variables measured on different scales, such as the effect of price changes and the number of promotional activities. These benefits help market researchers/data analysts/data scientists to eliminate and evaluate the best set of variables to be used for building predictive models.

3. DATASET DESCRIPTION

3.1 Getting The Data

Getting data is an important factor in building a predictive model, in most of the real-time situations we cannot have the luxury of having fully structured data every time. In this research work, a public dataset provided by Capital Bike Share on the UCI Repository is used for model construction. The data has the following attributes and are explained in the below table:

Features/Labels	Values
DateTime	hourly date + timestamp
Season	whether the day is considered a holiday
Working Day	whether the day is neither a weekend nor holiday
Weather	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
Temperature	temperature in Celsius
ATemp	"feels like" temperature in Celsius
Humidity	relative humidity
Windspeed	wind speed
Casual	number of non-registered user rentals initiated
Registered	number of registered user rentals initiated
Count	number of total rentals

3.2 History

The bike-sharing rental process is highly correlated to environmental and seasonal settings. For instance, weather conditions,precipitation, the day of the week, season, an hour of the day, etc. can affect the rental behaviors. The core data set is related to the two-year historical log corresponding to years 2011 and 2012 from the Capital Bikeshare system, Washington D.C., the USA which is publicly available in http://capitalbikeshare.com/system-data. We aggregated the data on two hourly and daily basis and then extracted and added the corresponding weather and seasonal information. Weather information is extracted from http://www.freemeteo.com.

Associated tasks:

Regression:

Predication of bike rental count hourly or daily based on the environmental and seasonal settings.

Event and Anomaly Detection:

Count of rented bikes is also correlated to some events in the town which easily are traceable via search engines.

For instance, a query like "2012-10-30 Washington d.c." in Google returns related results to Hurricane Sandy. Some of the important events are

Identified in Therefore the data can be used for validation of anomaly or event detection algorithms as well.

4. PROJECT DEFINITION AND DESCRIPTION

The Bike Share System is encouraging, most of the companies are having their own analytics divisions. Before starting any new Rental Bike Station the company will analyze how much the station will be useful and will it generate appropriate revenue also whether setting up the station is feasible or not. With enhancements in analytic techniques in the current era, with some simple surveys, Bike Share System can forecast the use of the system and then plan accordingly.

4.1 MODULES

- Data Pre-Processing
- Train the model
- Evaluate the model
- Use the Model

Data Pre-Processing

Data pre-processing 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 behaviors or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. There is a need for data pre-processing because the data may be incomplete or inconsistent or noisy. There are many ways to deal with un-processed data viz

i)**Data Cleaning**:

By this term we mean to fill the missing values in data, identifying and removing outliers in data, smoothing data.

ii)**Data Transformation**

In this stage operations like normalization and aggregation are performed.

iii)**Data Reduction**

In this stage, the data set is modified such that the results produced by the model are almost the same but unnecessary values in the dataset are removed.

iv)**Data Integration**

In this stage data is merged from different sources if needed, again redundancies are removed too.

TRAIN THE MODEL

In any Machine Learning model is that we're going to split data-set into two separate sets

- **Training Set**
- **Test Set**

Need for splitting

Well, here it's an algorithm model that is going to learn from your data to make predictions. Generally, we split the data-set into 70:30 ratio or 80:20 what does it mean, 70 percent data take in train and 30 percent data take in the test. However, this Splitting can be varied according to the data-set shape and size.

EVALUATE MODEL

Fraudulent transaction detector (positive class is "fraud"):

- Optimize for **sensitivity**
- Because false positives (normal transactions that are flagged as possible fraud) are more acceptable than false negatives (fraudulent transactions that are not detected)

USE THE MODEL

Model is well-trained after splitting and evaluation of the model under supervised. The machine will understand the new data input and estimate the needs. The trained machine can predict the value with estimate accuracy.

5. EXPLORATORY DATA ANALYSIS

Exploratory data analysis is a statistical way of understanding the data which is usually done in a visual way. The graphs plotted in exploratory data analysis are for better understanding of data by the analyst. For the current data set exploratory data analysis is done as follows:

Since, the system has to predict the number of bikes that will be rented, the best way to begin is with the variable to predict, "count". it can stratify the "count" distribution as boxplots for the categorical variables, and draw the"count" and numeric variables in another plot.

5.1 CORRELATION

#Correlation For Season

```
season        1.000000
holiday       0.029368
workingday   -0.008126
weather       0.008879
temp          0.258689
atemp         0.264744
humidity      0.190610
windspeed    -0.147121
casual        0.096758
registered    0.164011
count         0.163439
Name: season, dtype: float64
```

The Attribute season has Significant,Postive Correlation with temp and Atemp.

0.2-Significant

0.5-Highly Significant

0.00—No Correlation

#Correlation For Holiday

```
season        0.029368
holiday       1.000000
workingday   -0.250491
weather      -0.007074
temp          0.000295
atemp        -0.005215
humidity      0.001929
windspeed     0.008409
casual        0.043799
registered   -0.020956
count        -0.005393
Name: holiday, dtype: float64
```

The Attribute Holiday has Significant,Postive Correlation with Workingday

0.2-Significant

0.5-Highly Significant

0.00—No Correlation

#Correlation For Workingday

```
season       -0.008126
holiday      -0.250491
workingday    1.000000
weather       0.033772
temp          0.029966
atemp         0.024660
humidity     -0.010880
windspeed     0.013373
casual       -0.319111
registered    0.119460
count         0.011594
Name: workingday, dtype: float64
```

The Attribute Workinday has Significant,Postive Correlation with Holiday,casual

0.2-Significant

0.5-Highly Significant

0.00—No Correlation

#Correlation For Weather

```
season        0.008879
holiday      -0.007074
workingday    0.033772
weather       1.000000
temp         -0.055035
atemp        -0.055376
humidity      0.406244
windspeed     0.007261
casual       -0.135918
registered   -0.109340
count        -0.128655
Name: weather, dtype: float64
```

The Attribute Weather has Significant,Postive Correlation with Hunidity

0.2-Significant

0.5-Highly Significant

0.00—No Correlation

#Correlation For Temp

```
season        0.258689
holiday       0.000295
workingday    0.029966
weather      -0.055035
temp          1.000000
atemp         0.984948
humidity     -0.064949
windspeed    -0.017852
casual        0.467097
registered    0.318571
count         0.394454
Name: temp, dtype: float64
```

The Attribute Temp has Significant,Postive Correlation with Season,Count,Registered

0.2-Significant

0.5-Highly Significant

0.00—No Correlation

#Correlation For ATemp

```
season        0.264744
holiday      -0.005215
workingday    0.024660
weather      -0.055376
temp          0.984948
atemp         1.000000
humidity     -0.043536
windspeed    -0.057473
casual        0.462067
registered    0.314635
count         0.389784
Name: atemp, dtype: float64
```

The Attribute ATemp has Significant,Postive Correlation with Season,Count,Registered

0.2-Significant

0.5-Highly Significant

0.00—No Correlation

#Correlation For Humidity

```
season        0.190610
holiday       0.001929
workingday   -0.010880
weather       0.406244
temp         -0.064949
atemp        -0.043536
humidity      1.000000
windspeed    -0.318607
casual       -0.348187
registered   -0.265458
count        -0.317371
Name: humidity, dtype: float64
```

The Attribute Humidity has Significant,Postive Correlation with Season,Registered

0.2-Significant

0.5-Highly Significant

0.00—No Correlation

#Correlation For Casual

```
season        0.096758
holiday       0.043799
workingday   -0.319111
weather      -0.135918
temp          0.467097
atemp         0.462067
humidity     -0.348187
windspeed     0.092276
casual        1.000000
registered    0.497250
count         0.690414
Name: casual, dtype: float64
```

The Attribute Casual has Significant,Postive Correlation with Workingday

0.2-Significant

0.5-Highly Significant

0.00—No Correlation

#Correlation For Registered

```
season        0.164011
holiday      -0.020956
workingday    0.119460
weather      -0.109340
temp          0.318571
atemp         0.314635
humidity     -0.265458
windspeed     0.091052
casual        0.497250
registered    1.000000
count         0.970948
Name: registered, dtype: float64
```

The Attribute Registered has Significant,Postive Correlation with temp,Atemp

0.2-Significant

0.5-Highly Significant

0.00—No Correlation

#Correlation For Count

```
season        0.163439
holiday      -0.005393
workingday    0.011594
weather      -0.128655
temp          0.394454
atemp         0.389784
humidity     -0.317371
windspeed     0.101369
casual        0.690414
registered    0.970948
count         1.000000
Name: count, dtype: float64
```

The Attribute Count has Significant,Postive Correlation with Weather,temp

0.2-Significant

0.5-Highly Significant

0.00—No Correlation

5.2 STATISTICAL SUMMARY

1. temp : Normalized temperature in Celsius. The values are divided to 41 (max)

Mean	20.23086
Standard Error	0.074678
Median	20.5
Mode	14.76
Standard Deviation	7.79159
Sample Variance	60.70887
Kurtosis	-0.91453
Skewness	0.003691
Range	40.18
Minimum	0.82
Maximum	41
Sum	220233.1
Count	10886

2. atemp: Normalized feeling temperature in Celsius. The values are divided to 50 (max)

Mean	23.65508
Standard Error	0.081224
Median	24.24
Mode	31.06
Standard Deviation	8.474601
Sample Variance	71.81886
Kurtosis	-0.85008
Skewness	-0.10256
Range	44.695
Minimum	0.76
Maximum	45.455
Sum	257509.2
Count	10886

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

Mean	61.88646
Standard Error	0.184452
Median	62
Mode	88
Standard Deviation	19.24503
Sample Variance	370.3713
Kurtosis	-0.75982
Skewness	-0.08634
Range	100
Minimum	0
Maximum	100
Sum	673696
Count	10886

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

Mean	12.7994
Standard Error	0.078252
Median	12.998
Mode	0
Standard Deviation	8.164537
Sample Variance	66.65967
Kurtosis	0.630133
Skewness	0.588767
Range	56.9969
Minimum	0
Maximum	56.9969
Sum	139334.2
Count	10886

5. casual: count of casual users

Mean	36.02195
Standard Error	0.478842
Median	17
Mode	0
Standard Deviation	49.96048
Sample Variance	2496.049
Kurtosis	7.551629
Skewness	2.495748
Range	367
Minimum	0
Maximum	367
Sum	392135
Count	10886

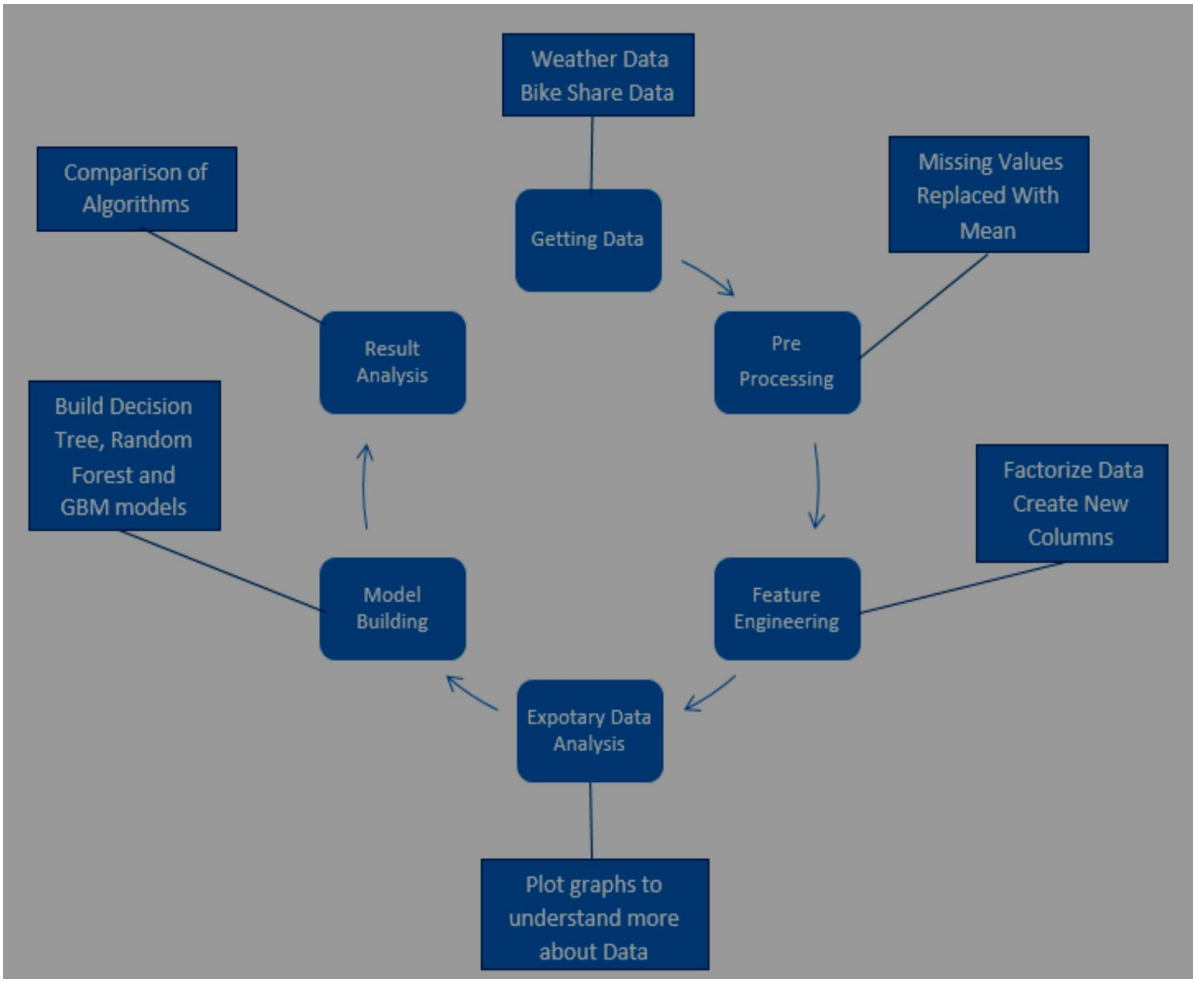
6. registered: count of registered users

Mean	155.5522
Standard Error	1.447622
Median	118
Mode	3
Standard Deviation	151.039
Sample Variance	22812.79
Kurtosis	2.626081
Skewness	1.524805
Range	886
Minimum	0
Maximum	886
Sum	1693341
Count	10886

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

Mean	191.5741
Standard Error	1.736165
Median	145
Mode	5
Standard Deviation	181.1445
Sample Variance	32813.31
Kurtosis	1.300093
Skewness	1.242066
Range	976
Minimum	1
Maximum	977
Sum	2085476
Count	10886

6.3 SYSTEM ARCHITECTURE FOR PREDICTION MODELS



6. DATA PROCESSING

6.1 ENCODING

ONE-HOT ENCODING

The Standard Approach for Categorical Data. One hot encoding is the most widespread approach, and it works very well unless your categorical variable takes on a large number of values (i.e. you generally won't it for variables taking more than 15 different values. It'd be a poor choice in some cases with fewer values, though that varies.)

One hot encoding creates new (binary) columns, indicating the presence of each possible value from the original data. Let's work through an example.

In this case, a one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

In the “color” variable example, there are 3 categories and therefore 3 binary variables are needed. A “1” value is placed in the binary variable for the color and “0” values for the other colors.

6.2 COLUMNS DROP BASED ON EXPLORATORY DATA ANALYSIS

Remove the fields from a dataset which doesn’t have a proper meaning like categorical columns. If it is not removed, on training dataset the machine will get confuses with these data.

- “temp” and “humidity” features have a positive and negative correlation with count respectively. Although the correlations between them are not very prominent, still the count variable has got a little dependency on “temp” and “humidity”.
- “windspeed” is not going to be a really useful numerical feature and that is visible from the correlation value with “count”.

- “atemp” variable is not taken into account since “atemp” and “temp” has a strong correlation with each other. During model building, any one of the variables has to be dropped since they will exhibit multicollinearity in the data.
- “casual” and “registered” attributes are also not taken into account since they are *leakage variables* in nature and need to be dropped during model building.

6.3 STANDARDIZATION

Standardization of datasets is a common requirement for many machine learning estimators implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance.

In practice, we often ignore the shape of the distribution and just transform the data to center it by removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation.

For instance, many elements used in the objective function of a learning algorithm assume that all features are centered around zero and have variance in the same order. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

7. FEATURE ENGINEERING

Feature engineering is the process of using **Domain Knowledge** of the data to create features that make machine learning algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process. Feature engineering is the most important art in machine learning which creates a huge difference between a good model and a bad model.

Typically, feature engineering is a drawn-out manual process, relying on domain knowledge, intuition, and data manipulation. This process can be extremely tedious and the final features will be limited both by human subjectivity and time. Automated feature engineering aims to help the data scientist by automatically creating many candidates features out of a dataset from which the best can be selected and used for training.

Feature engineering means building additional features out of existing data which is often spread across multiple related tables. Feature engineering requires extracting the relevant information from the data and getting it into a single table which can then be used to train a machine learning model. The process of constructing features is very time-consuming because each new feature usually requires several steps to build, especially when using information from more than one table.

As we see from the above results, the columns season, holiday, workingday and weather should be of categorical data type. But the current data type is int for those columns. Let us transform the dataset in the following ways so that we can get started with our EDA.

- Create new columns “date, ”hour”, ”weekday”, ”month” from “datetime” column.
- Coerce the datatype of “season”,”holiday”,”workingday” and weather to categorical datatype.
- Drop the datetime column as we already extracted useful features from it.

Lets start with a very simple visualization of variables data type count.

7.1 Split the model

Module introduced dividing your data set into two subsets:

- **Training set**—a subset to train a model.
- **Test set**—a subset to test the trained model.

Need of splitting

Well, here it's your algorithm model that is going to learn from your data to make predictions. Generally, we split the data-set into 70:30 ratio or 80:20 what does it mean, 70 percent data take in train and 30 percent data take in a test. However, this Splitting can be varied according to the data-set shape and size.

Make sure that your test set meets the following two conditions:

- ✓ Is large enough to yield statistically meaningful results?
- ✓ Is representative of the data set as a whole. In other words, don't pick a test set with different characteristics than the training set?

Assuming that your test set meets the preceding two conditions, your goal is to create a model that generalizes well to new data. Our test set serves as a proxy for new data. Notice that the model learned for the training data is very simple. This model doesn't do a perfect job—a few predictions are wrong. However, this model does about as well on the test data as it does on the training data. In other words, this simple model does not overfit the training data.

Never train on test data. If you are seeing surprisingly good results on your evaluation metrics, it might be a sign that you are accidentally training on the test set. For example, high accuracy might indicate that test data has leaked into the training set.

7.2 Missing Value Analysis

Once we get the hang of the data and attributes, the next step we generally do is to find out whether we have any missing values in our data. Luckily we do not have any missing value in the data. One way which I generally prefer to visualize missing value in the data is through the missing library in python.

It is quite a handy library to quickly visualize missing values in attributes. As I mentioned earlier we got lucky this time as there were no missing values in the data. But we have a lot of 0's in "windspeed" column which we will deal later when we build machine learning models.

8. MODEL SELECTION

The data set was modeled as a multi-class classification problem (i.e., The outcome variable “Number of rides” has multiple class labels for values.) in the beginning, and different multi-class classification algorithms were used up front in the analysis. Because the accuracy for those models was small, granularity was reduced by decreasing the number of class labels to two. We asserted that no learning algorithm can uniformly outperform other algorithms for all data sets.

We use regression analysis to predict bike trips during morning rush hours for the station. We use log-log regression models and test the model in different scenarios based on weather and temporal characteristics. Hence as the data is understood properly, random forest, linear regression predictive model can be built for this data to predict the count variable

.

9.1 ENSEMBLE TECHNIQUE

Ensemble learning is a machine learning paradigm that uses multiple learning models to perform predictions. It is based on a perspective that using multiple models and combining them to obtain the final prediction can produce better results compared to using a single model. Some representative ensembles include boosting, bagging, stacking and a bucket of models. It could be seen from the literature that bagging and boosting are the most extensively used ensembles.

State-of-threat techniques include Random Forest (Bagging based ensemble), Gradient Boosted Trees (GBT), Ada-Boost and XG Boost (Boosting based ensembles). Bagging is a type of ensemble modeling that aims to create multiple training models and passes them with several overlapped subsets of data for training. Each base learner in a bagged ensemble is an independent fully trained model. The data to be predicted is then passed to each of these learners and multiple predictions are obtained. The combiner aggregates these results and provides a single prediction result.

9. IMPLEMENTATION

Source code for machine learning model

#neccessary packages

```
import pandas as pd
import numpy as np
```

#import necessary file

```
bike=pd.read_csv("bikeSharing.csv")
bike.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

```
bike.info()
```

#Finding similar item

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
datetime      10886 non-null object
season        10886 non-null int64
holiday       10886 non-null int64
workingday    10886 non-null int64
weather       10886 non-null int64
temp          10886 non-null float64
atemp         10886 non-null float64
humidity      10886 non-null int64
windspeed     10886 non-null float64
casual        10886 non-null int64
registered    10886 non-null int64
count         10886 non-null int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.6+ KB
```

```
bike.describe()
```

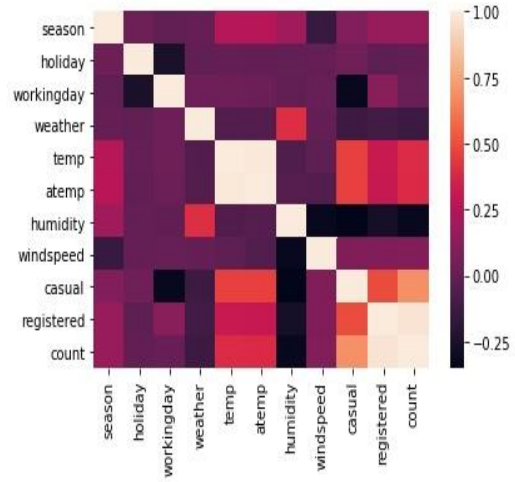
	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	cc
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000

bike.corr()

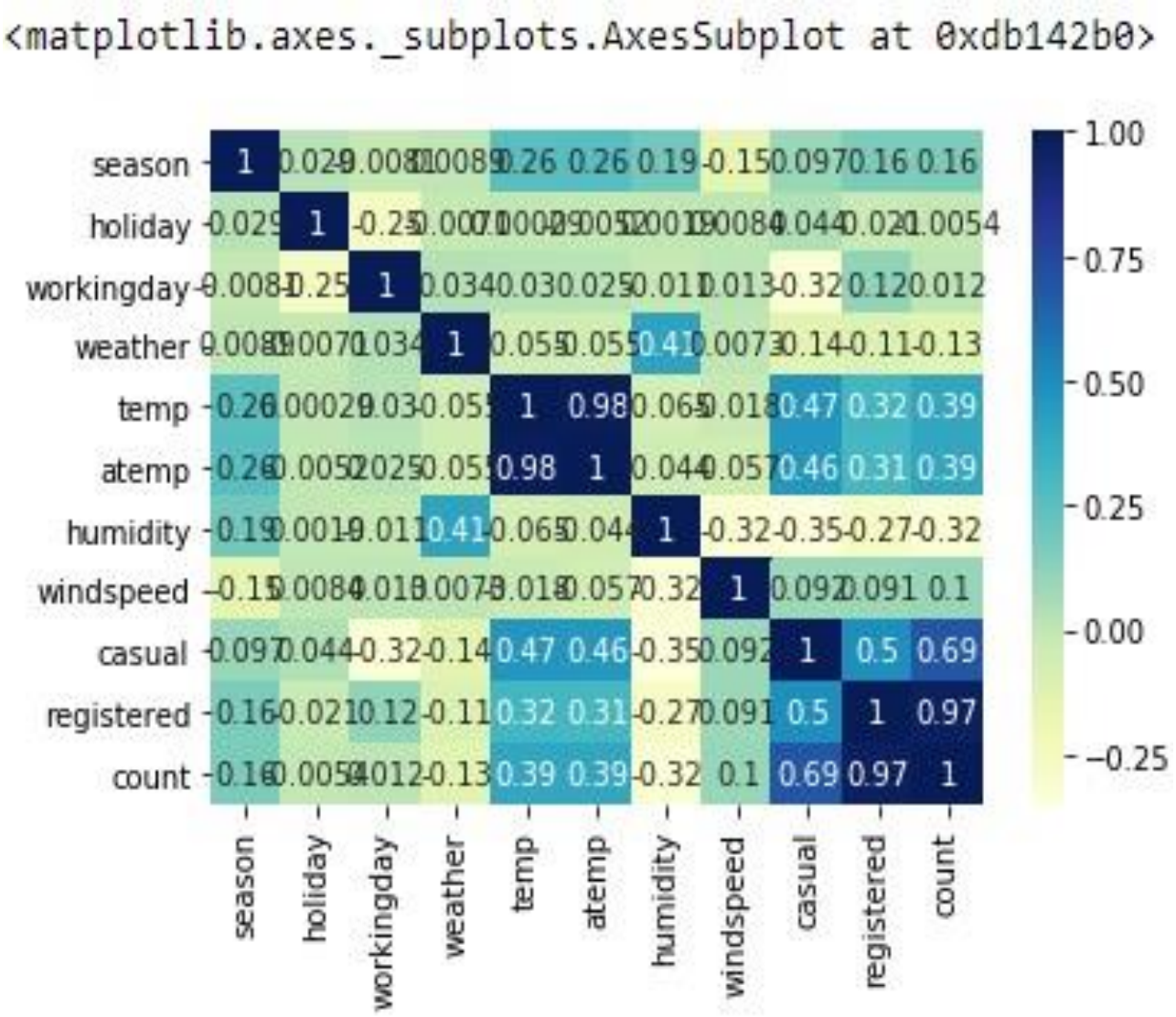
	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
season	1.000000	0.029368	-0.008126	0.008879	0.258689	0.264744	0.190610	-0.147121	0.096758	0.164011	0.163439
holiday	0.029368	1.000000	-0.250491	-0.007074	0.000295	-0.005215	0.001929	0.008409	0.043799	-0.020956	-0.005393
workingday	-0.008126	-0.250491	1.000000	0.033772	0.029966	0.024660	-0.010880	0.013373	-0.319111	0.119460	0.011594
weather	0.008879	-0.007074	0.033772	1.000000	-0.055035	-0.055376	0.406244	0.007261	-0.135918	-0.109340	-0.128655
temp	0.258689	0.000295	0.029966	-0.055035	1.000000	0.984948	-0.064949	-0.017852	0.467097	0.318571	0.394454
atemp	0.264744	-0.005215	0.024660	-0.055376	0.984948	1.000000	-0.043536	-0.057473	0.462067	0.314635	0.389784
humidity	0.190610	0.001929	-0.010880	0.406244	-0.064949	-0.043536	1.000000	-0.318607	-0.348187	-0.265458	-0.317371
windspeed	-0.147121	0.008409	0.013373	0.007261	-0.017852	-0.057473	-0.318607	1.000000	0.092276	0.091052	0.101369
casual	0.096758	0.043799	-0.319111	-0.135918	0.467097	0.462067	-0.348187	0.092276	1.000000	0.497250	0.690414
registered	0.164011	-0.020956	0.119460	-0.109340	0.318571	0.314635	-0.265458	0.091052	0.497250	1.000000	0.970948
count	0.163439	-0.005393	0.011594	-0.128655	0.394454	0.389784	-0.317371	0.101369	0.690414	0.970948	1.000000

sns.heatmap(bike.corr())

```
<matplotlib.axes._subplots.AxesSubplot at 0xbfdb320>
```



sns.heatmap(bike.corr(),cmap='YlGnBu',annot=True)



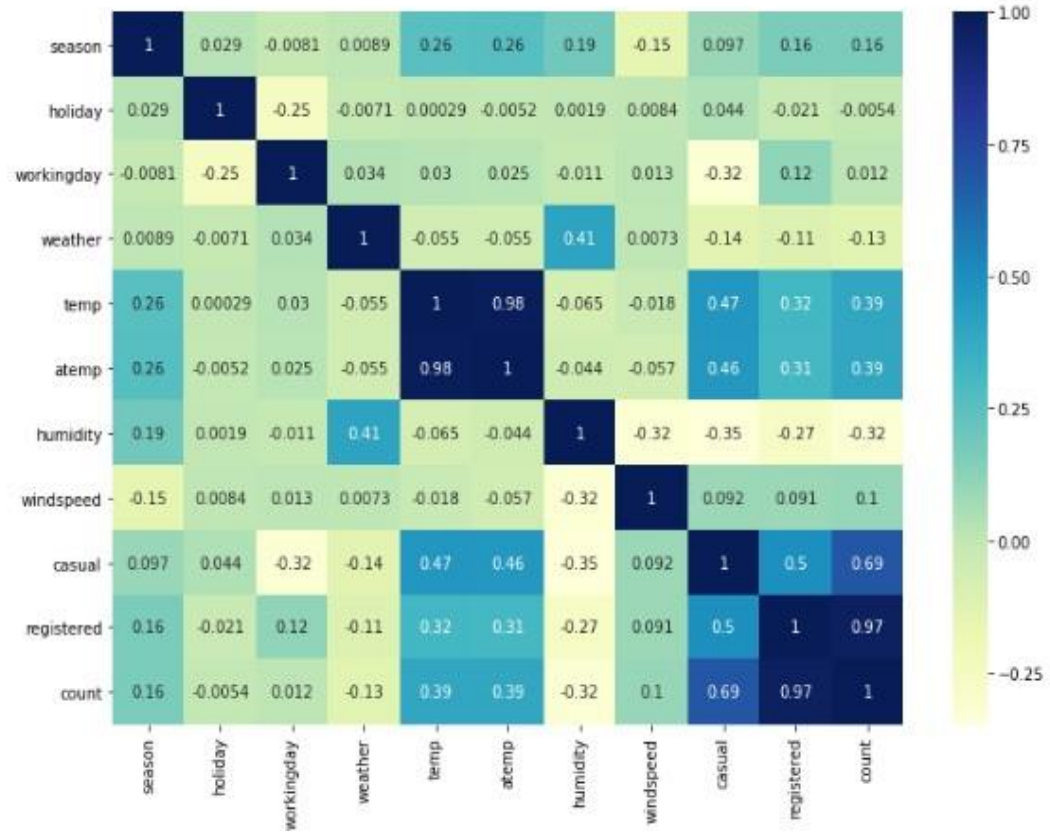
from matplotlib import pyplot

a4_dims = (11.7, 8.27)

fig, ax = pyplot.subplots(figsize=a4_dims)

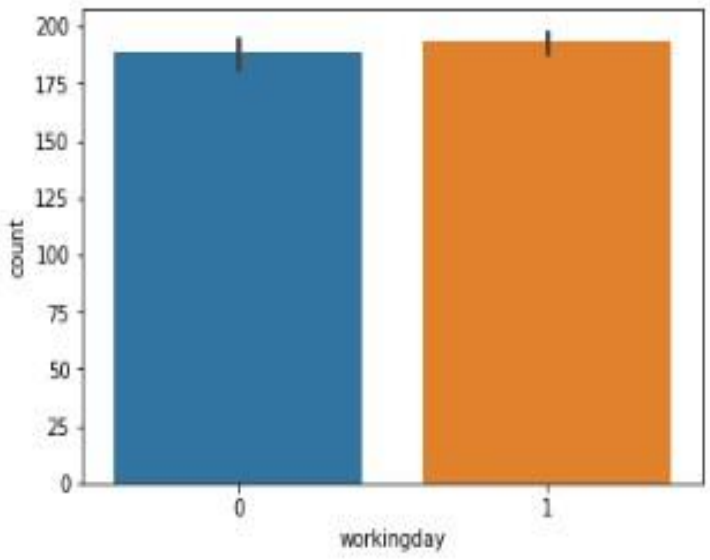
sns.heatmap(ax=ax,data=bike.corr(),cmap='YlGnBu',annot=True)

```
<matplotlib.axes._subplots.AxesSubplot at 0xe1ac240>
```




```
sns.barplot(data=bike,x='workingday',y='count')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x129b2438>
```



#Univariate analysis of all variables

#Categorical data--> Season, Holiday, WorkingDay, Weather,Temp,Atemp,Count

correlation=bike.corr()

correlation['count']

```
season        0.163439
holiday      -0.005393
workingday    0.011594
weather      -0.128655
temp          0.394454
atemp         0.389784
humidity     -0.317371
windspeed     0.101369
casual        0.690414
registered    0.970948
count         1.000000
Name: count, dtype: float64
```

```
correlation=np.abs(correlation['count'])
correlation
```

```
season        0.163439
holiday       0.005393
workingday    0.011594
weather       0.128655
temp          0.394454
atemp         0.389784
humidity      0.317371
```

```
windspeed     0.101369
casual        0.690414
registered    0.970948
count         1.000000
Name: count, dtype: float64

correlation.sort_values(inplace=True)
correlation

holiday       0.005393
workingday    0.011594
windspeed     0.101369
weather       0.128655
season        0.163439
humidity      0.317371
atemp         0.389784
temp          0.394454
casual        0.690414
registered    0.970948
count         1.000000
Name: count, dtype: float6
tobeRemoved=correlation[correlation<0.3]
tobeRemoved

holiday       0.005393
workingday    0.011594
windspeed     0.101369
weather       0.128655
season        0.163439
Name: count, dtype: float64
```

#model creation

```
bikeFS=bike.drop(tobeRemoved.index,axis=1)
bikeFS.head()
```

	datetime	temp	atemp	humidity	casual	registered	count
0	2011-01-01 00:00:00	9.84	14.395	81	3	13	16
1	2011-01-01 01:00:00	9.02	13.635	80	8	32	40
2	2011-01-01 02:00:00	9.02	13.635	80	5	27	32
3	2011-01-01 03:00:00	9.84	14.395	75	3	10	13
4	2011-01-01 04:00:00	9.84	14.395	75	0	1	1

from sklearn.preprocessing import OneHotEncoder

import sklearn

#One Hot Encoding

enc = OneHotEncoder(handle_unknown='ignore')

enc.fit(churnNN[tobeEnc])

```
OneHotEncoder(categorical_features=None, categories=None,
       dtype=<type 'numpy.float64'>, handle_unknown='ignore',
       n_values=None, sparse=True)
```

enc.categories_

```
[array(['00001', '00002', '00003', '00004', '00005', '00006', '00007',
        '00008', '00009', '0000A', '0000B', '0000C', '0000D', '0000E',
        '0000F', '00010', '00011', '00012', '00013', '00014', '00015',
        '00016', '00017', '00018', '00019', '0001A', '0001B', '0001C',
        '0001D', '0001E', '0001F', '00020', '00021', '00022', '00023',
        '00024', '00025', '00026', '00027', '00028', '00029', '0002A',
        '0002B', '0002C', '0002D', '0002E', '0002F', '00030', '00031',
        '00032', '00033', '00034', '00035', '00036', '00037', '00038',
        '00039', '0003A', '0003B', '0003C', '0003D', '0003E', '0003F',
        '00040', '00041', '00042', '00043', '00044', '00045', '00046',
        '00047', '00048', '00049', '0004A', '0004B', '0004C', '0004D',
        '0004E', '0004F', '00050', '00051', '00052', '00053', '00054',
        '00055', '00056', '00057', '00058', '00059', '0005A', '0005B',
        '0005C', '0005D', '0005E', '0005F', '00060', '00061', '00062',
        '00063', '00064', '00065', '00066', '00067', '00068', '00069',
        '0006A', '0006B', '0006C', '0006D', '0006E', '0006F', '00070',
        '00071', '00072', '00073', '00074', '00075', '00076', '00077',
        '00078', '00079', '0007A', '0007B', '0007C', '0007D', '0007E',
        '0007F', '00080', '00081', '00082', '00083', '00084', '00085',
```

model evaluation

bike.drop('datetime',axis=1,inplace=True)

bike.head()

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	1	0	0	1	9.84	14.395	75	0.0	0	1	1

from sklearn.model_selection import train_test_split

X=bike.drop('count',axis=1).values

y=bike['count']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

from sklearn.ensemble import RandomForestRegressor

rf=RandomForestRegressor()

rf.fit(X_train,y_train)

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False
)
```

rf.feature_importances_

```
array([2.67142929e-05, 2.64846970e-06, 1.55510302e-04, 2.81303598e-05,
       9.89316759e-05, 5.91701907e-05, 9.84868125e-05, 7.75545874e-05,
       5.09013535e-02, 9.48551500e-01])
```

rf.predict(X_test)

```
array([128.4,  13. , 162. , ..., 294.6, 464.1, 385.4])
```

predictions=rf.predict(X_test)

predComp=pd.DataFrame({'Predictions':predictions,'Actual':y_test})

predComp.head()

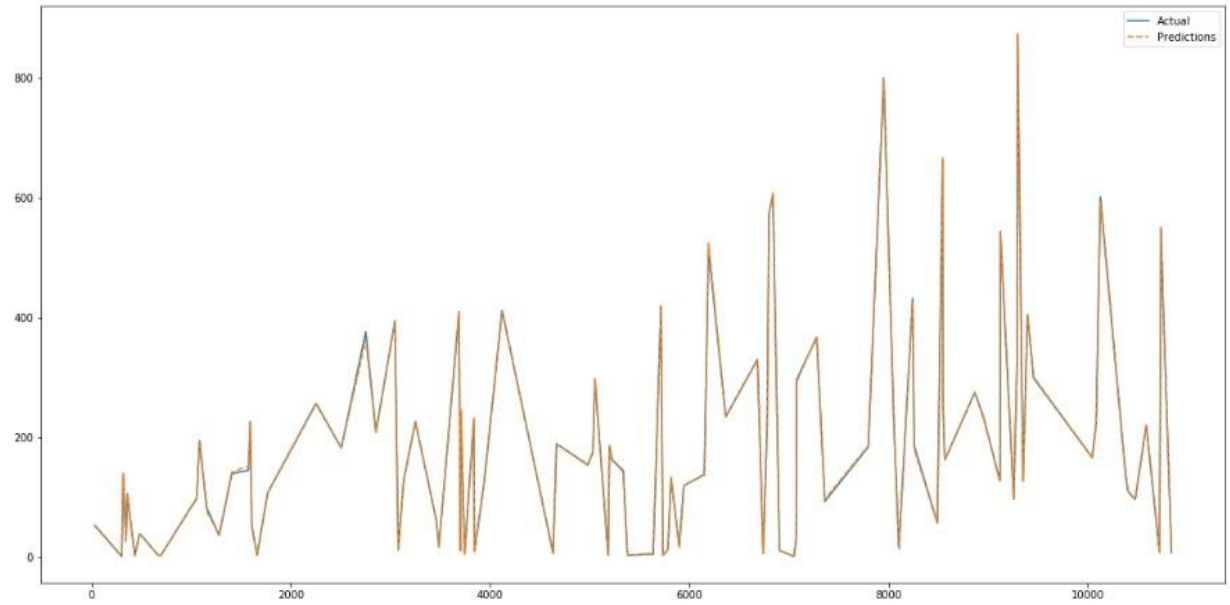
	Actual	Predictions
3133	127	128.4
5786	13	13.0
5224	163	162.0
8953	233	234.9
8054	222	223.1

a4_dims = (20, 10)

fig, ax = pyplot.subplots(figsize=a4_dims)

sns.lineplot(ax=ax,data=predComp[:100],palette='tab10')

```
<matplotlib.axes._subplots.AxesSubplot at 0x1209b550>
```



predComp.to_csv("finalPred.csv")

prediction

10. CONCLUSION

Bike sharing system can be the new boom in India, with the use of various prediction models the ease of operations will be increased. The four algorithms are applied to the bike-share dataset for predicting the count of bicycles that will be rented per hour. Results shows that accuracy with random forest and using TuneRF function with the original random forest algorithm. The accuracy and performance have been compared between the models using Root Mean Squared Logarithmic Error (RMSLE). If these systems include the use of analytics the probability of building a successful system will increase.

REFERENCES

- Data set information: https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset
- Original Source: http://capitalbikeshare.com/system-data
- Weather Information: http://www.freemeteo.com
- Holiday Schedule: http://dchr.dc.gov/page/holiday-schedule