OLA BIKE RIDE REQUEST DEMAND FORCAST PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the degreeof

BACHELOR OF TECHNOLOGY(INT-254)

IN

COMPUTER SCIENCE AND ENGINEERING

Under the Guidance of:

Dr. Pawan Kumar

(Professor)

Submitted By

Hrishikesh Bharti (12014036)

Kanishk Rao (12015255)

CANDIDATE’S DECLARATION AND CERTIFICATE

We hereby certify that the work, which is being presented in this report entitled, **OLA BIKE RIDE REQUEST DEMAND FORCAST PROJECT REPORT**, in partial fulfilment of the requirements for the degree of **Bachelor of Technology, Course Code INT-254** submitted in the **Computer Science and Engineering , Lovely Professional University, Punjab**; by **MD Hrishikesh Bharti (12014036), Kanishk Rao (12015255)** is the authentic recordof our own work carried out under the supervision of **Dr. Pawan Kumar Mall, Professor, Computer Science and Engineering, Lovely Professional University, Punjab.**

We further declare that the matter embodied in this report has not been submitted by us forthe award of any other degree.

Candidate(s) Signature

This is to certify that the above statement made by the candidate is correct to the best of my knowledgeand belief.

Signature of Supervisor**Dr.**

Pawan Kumar Date:

ACKNOWLEDGMENT

It is our pleasure to acknowledge the contributions of all who have helped us and supported us during this Project report.

First, we thank God for helping us in one way or another and providing strength and enduranceto us. We wish to express my sincere gratitude and indebtedness to our supervisor Mr. Pawan Kumar Mall, Computer Science and Engineering, Lovely Professional University, Punjab; for his intuitive and meticulous guidance and perpetual inspiration in completion of this report. In spite of his busy schedule, he rendered help whenever needed, giving useful suggestions and holding informal discussions. His invaluable guidance and support throughout this work cannot be written down in fewwords. We also thank him for providing facilities for my work in the department.

We are also humbly obliged by the support of our group members and friends for their love and caring attitude. The sentimental support they rendered to us is invaluable and everlasting. They have helped us through thick and thin and enabled us to complete the work with joy and vigour. Wethank the group members for entrusting in each other and following directions, without them this report would never have been possible.

We are also thankful to our parents, elders and all family members for their blessing, motivation and inspiration throughout our work and bearing with us even during stress and bad temper. They have always provided us with high moral support and contributed in all possible waysin completion of this report.

TABLE OF CONTENTS

INTRODUCTION

Machine learning is almost everywhere nowadays. It's become more and more necessary day by day. From the recommendation of what to buy to recognizing a person, robotics, everywhere is machine learning. So in this project, we'll create a machine learning model Ola bike ride request demand forecast .

From telling rickshaw-wala where to go, to tell him where to come we have grown up. Yes, we are talking about online cab and bike facility providers like OLA and Uber. If you had used this app some times then you must have paid some day less and someday more for the same journey. But have you ever thought what is the reason behind it? It is because of the high demand at some hours. this is not the only factor but this is one of them.

In this project, we will try to predict ride-request for a particular hour using machine learning.

What is machine learning?

Machine learning is about learning to predict something or extracting knowledge from data. ML is a part of artificial intelligence. ML algorithms build a model based on sample data or known as training data and based upon the training data the algorithm can predict something on new data.

Machine learning is about learning to predict something or extracting knowledge fromdata. ML is a part of artificial intelligence. ML algorithms build a model based on sample data or known as training data and based upon the training data the algorithm can predict something on new data.
Categories of Machine Learning :

Supervised machine learning: Supervised machine learning are types of machine learning that are trained on well-labeled training data. Labeled data means the trainingdata is already tagged with the correct output.

Unsupervised machine learning: Unlike supervised learning, unsupervised learningdoesn't have any tagged data. It learned patterns from untagged data. Basically, it creates a group of objects based on the input data/features.

Semi-supervised machine learning:

Semi-supervised learning falls between supervised and unsupervised learning. It has asmall amount of tagged data and a large amount of untagged data.

OBJECTIVE

---

1. To develop an algorithm to predict ride-request demand.
2. Apply different algorithms to develop an ideal model.
3. we will try to predict ride-request for a particular hour using machine learning. One can refer to the below explanation for the column names in the dataset and their values as well.

Dataset used for deriving the modules: -

```
instant,datetime,season,year,month,holidays,weekday,workingday,weather,temp,atemp,humidity,windspeed,casual,registered,count
1,01-01-2011,1,0,1,0,6,0,2,0.344167,0.363625,0.805833,0.160446,331,654,985
2,02-01-2011,1,0,1,0,0,0,2,0.363478,0.353739,0.696087,0.248539,131,670,801
3,03-01-2011,1,0,1,0,1,1,1,0.196364,0.189405,0.437273,0.248309,120,1229,1349
4,04-01-2011,1,0,1,0,2,1,1,0.2,0.212122,0.590435,0.160296,108,1454,1562
5,05-01-2011,1,0,1,0,3,1,1,0.226957,0.22927,0.436957,0.1869,82,1518,1600
6,06-01-2011,1,0,1,0,4,1,1,0.204348,0.233209,0.518261,0.0895652,88,1518,1606
7,07-01-2011,1,0,1,0,5,1,2,0.196522,0.208839,0.498696,0.168726,148,1362,1510
8,08-01-2011,1,0,1,0,6,0,2,0.165,0.162254,0.535833,0.266804,68,891,959
9,09-01-2011,1,0,1,0,0,0,1,0.138333,0.116175,0.434167,0.36195,54,768,822
10,10-01-2011,1,0,1,0,1,1,1,0.150833,0.150888,0.482917,0.223267,41,1280,1321
11,11-01-2011,1,0,1,0,2,1,2,0.169091,0.191464,0.686364,0.122132,43,1220,1263
12,12-01-2011,1,0,1,0,3,1,1,0.172727,0.160473,0.599545,0.304627,25,1137,1162
13,13-01-2011,1,0,1,0,4,1,1,0.165,0.150883,0.470417,0.301,38,1368,1406
14,14-01-2011,1,0,1,0,5,1,1,0.16087,0.188413,0.537826,0.126548,54,1367,1421
15,15-01-2011,1,0,1,0,6,0,2,0.233333,0.248112,0.49875,0.157963,222,1026,1248
16,16-01-2011,1,0,1,0,0,0,1,0.231667,0.234217,0.48375,0.188433,251,953,1204
17,17-01-2011,1,0,1,1,1,0,2,0.175833,0.176771,0.5375,0.194017,117,883,1000
18,18-01-2011,1,0,1,0,2,1,2,0.216667,0.232333,0.861667,0.146775,9,674,683
19,19-01-2011,1,0,1,0,3,1,2,0.292174,0.298422,0.741739,0.208317,78,1572,1650
20,20-01-2011,1,0,1,0,4,1,2,0.261667,0.25505,0.538333,0.195904,83,1844,1927
21,21-01-2011,1,0,1,0,5,1,1,0.1775,0.157833,0.457083,0.353242,75,1468,1543
22,22-01-2011,1,0,1,0,6,0,1,0.0591304,0.0790696,0.4,0.17197,93,888,981
23,23-01-2011,1,0,1,0,0,0,1,0.0965217,0.0988391,0.436522,0.2466,150,836,986
24,24-01-2011,1,0,1,0,1,1,1,0.0973913,0.11793,0.491739,0.15833,86,1330,1416
25,25-01-2011,1,0,1,0,2,1,2,0.223478,0.234526,0.616957,0.129796,186,1799,1985
26,26-01-2011,1,0,1,0,3,1,3,0.2175,0.2036,0.8625,0.29385,34,472,506
27,27-01-2011,1,0,1,0,4,1,1,0.195,0.2197,0.6875,0.113837,15,416,431
```

Importing the Modules: -

Python libraries make it easy for us to handle the data and perform typical and complex tasks with a single line of code.

Pandas – This library helps to load the data frame in a 2D array format and has multiple functions to perform analysis tasks in one go.

Numpy – Numpy arrays are very fast and can perform large computations in a very short time.
Matplotlib/Seaborn – This library is used to draw visualizations.

Sklearn – This module contains multiple libraries are having pre-implemented functions to perform tasks from data preprocessing to model development and evaluation.

XGBoost – This contains the eXtreme Gradient Boosting machine learning algorithm which is one of the algorithms which helps us to achieve high accuracy on predictions.

In [107]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.ensemble import RandomForestRegressor
import matplotlib
import warnings
warnings.filterwarnings('ignore')
from datetime import datetime
import calendar
from datetime import date
import holidays
```

In [108]:
```
df = pd.read_csv('day.csv')
df.head()
```

Out[108]:

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

Determining the shape and size of dataset: -

Let’s check which column of the dataset contains which type of data

```
In [109]: shape=df.shape
          df.info()
          df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   instant     731 non-null    int64  
 1   datetime    731 non-null    object 
 2   season      731 non-null    int64  
 3   year        731 non-null    int64  
 4   month       731 non-null    int64  
 5   holidays    731 non-null    int64  
 6   weekday     731 non-null    int64  
 7   workingday  731 non-null    int64  
 8   weather     731 non-null    int64  
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  humidity    731 non-null    float64
 12  windspeed   731 non-null    float64
 13  casual      731 non-null    int64  
 14  registered  731 non-null    int64  
 15  count       731 non-null    int64  
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

As per the above information regarding the data in each column we can observe that there are no null values.

Out[110]:

	instant	season	year	month	holidays	weekday	workingday	weather	temp	atemp	humidity	windspeed	
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.
mean	366.000000	2.496580	0.500684	6.519836	0.028728	2.997264	0.683995	1.395349	0.495385	0.474354	0.627894	0.190486	848.
std	211.165812	1.110807	0.500342	3.451913	0.167155	2.004787	0.465233	0.544894	0.183051	0.162961	0.142429	0.077498	686.
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.059130	0.079070	0.000000	0.022392	2.
25%	183.500000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000	1.000000	0.337083	0.337842	0.520000	0.134950	315.
50%	366.000000	3.000000	1.000000	7.000000	0.000000	3.000000	1.000000	1.000000	0.498333	0.486733	0.626667	0.180975	713.
75%	548.500000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000	2.000000	0.655417	0.608602	0.730209	0.233214	1096.
max	731.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000	3.000000	0.861667	0.840896	0.972500	0.507463	3410.

Feature Engineering

There are times when multiple features are provided in the same feature, or we have to derive some features from the existing ones. We will also try to include some extra features in our dataset so, that we can derive some interesting insights from the data we have. Also if the features derived are meaningful then they become a deciding factor in increasing the model’s accuracy significantly.

In [111]:
```python
parts = df["datetime"].str.split(" ", n=2, expand=True)
df["datetime"] = parts[0]
df.head()
```

Out[111]:

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

In the above step, we have separated the date and time. Now let’s extract the day, month, and year from the date column

In [112]:
```python
parts = df["datetime"].str.split("-", n=3, expand=True)
df["day"] = parts[0].astype('int')
df["month"] = parts[1].astype('int')
df["year"] = parts[2].astype('int')
df.head()
```

Out[112]:

	instant	datetime	season	year	month	holidays	weekday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	day
0	1	01-01-2011	1	2011	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985	1
1	2	02-01-2011	1	2011	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801	2
2	3	03-01-2011	1	2011	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349	3
3	4	04-01-2011	1	2011	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562	4
4	5	05-01-2011	1	2011	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600	5

Whether it is a weekend, or a weekday must have some effect on the ride request count.

```
In [113]: def weekend_or_weekday(year, month, day):
              d = datetime(year, month, day)
              if d.weekday() > 4:
                  return 0
              else:
                  return 1
```

```
In [114]: df['weekday'] = df.apply(lambda x:weekend_or_weekday(x['year'],x['month'],x['day']),axis=1)
          df.head()
```

Out[114]:

	instant	datetime	season	year	month	holidays	weekday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	day
0	1	01-01-2011	1	2011	1	0	0	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985	1
1	2	02-01-2011	1	2011	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801	2
2	3	03-01-2011	1	2011	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349	3
3	4	04-01-2011	1	2011	1	0	1	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562	4
4	5	05-01-2011	1	2011	1	0	1	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600	5

It would be nice to have a column which can indicate whether there was any holiday on a particular day or not.

```
In [115]: def is_holiday(x):

              india_holidays = holidays.country_holidays('IN')

              if india_holidays.get(x):
                  return 1
              else:
                  return 0
```

```
In [116]: df['holidays'] = df['datetime'].apply(is_holiday)
          df.head()
```

Out[116]:

	instant	datetime	season	year	month	holidays	weekday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	day
0	1	01-01-2011	1	2011	1	0	0	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985	1
1	2	02-01-2011	1	2011	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801	2
2	3	03-01-2011	1	2011	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349	3
3	4	04-01-2011	1	2011	1	0	1	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562	4
4	5	05-01-2011	1	2011	1	1	1	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600	5

Now let’s remove the columns which are not useful for us

```
In [117]: df.drop(['datetime',],axis=1,inplace=True)
          df.head()
```

Out[117]:

	instant	season	year	month	holidays	weekday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	day
0	1	1	2011	1	0	0	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985	1
1	2	1	2011	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801	2
2	3	1	2011	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349	3
3	4	1	2011	1	0	1	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562	4
4	5	1	2011	1	1	1	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600	5

There may be some other relevant features as well which can be added to this dataset but let's try to build a build with these ones and try to extract some insights as well.

Exploratory Data Analysis

EDA is an approach to analyzing the data using visual techniques. It is used to discover trends, and patterns, or to check assumptions with the help of statistical summaries and graphical representations.

We have added some features to our dataset using some assumptions. Now let's check what are the relations between different features with the target feature.

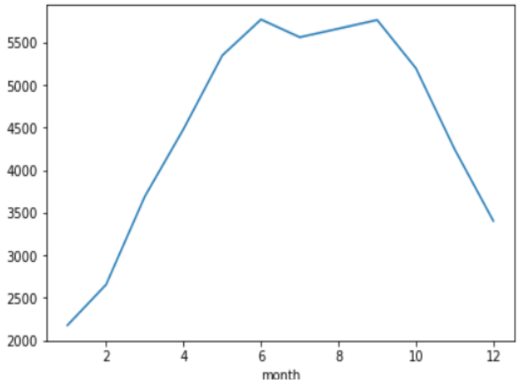
```
In [118]: df.isnull().sum()

Out[118]: instant       0
          season        0
          year          0
          month         0
          holidays      0
          weekday       0
          workingday    0
          weather       0
          temp          0
          atemp         0
          humidity      0
          windspeed     0
          casual        0
          registered    0
          count         0
          day           0
          dtype: int64
```

Now, we will check for any relation between the ride request count with respect to the month.

```
In [21]: features = ['month']
```

```
In [22]: plt.subplots(figsize=(15, 10))
for i, col in enumerate(features):
    plt.subplot(2, 2, i + 1)
    df.groupby(col).mean()['count'].plot()
plt.show()
```

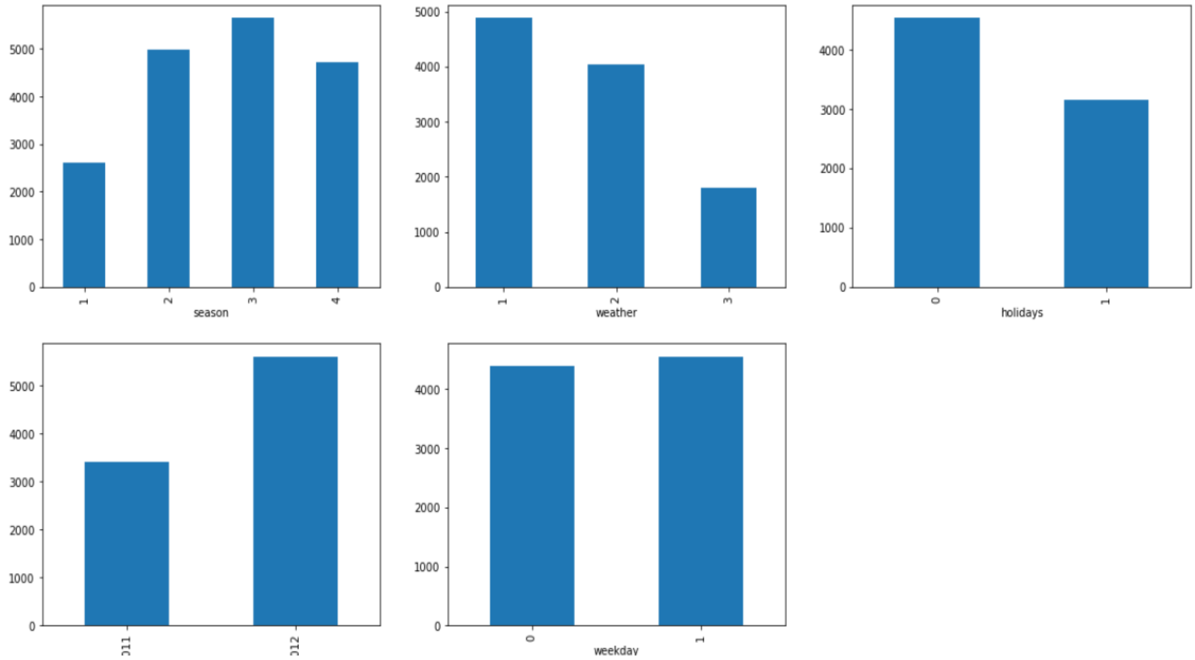


From the above line plots we can confirm some real-life observations:

- The average ride request count has dropped in the month of festivals that is after the 7th month that is July that is due to more holidays in these months.

```
In [23]: features = ['season', 'weather', 'holidays','year', 'weekday']

plt.subplots(figsize=(20, 10))
for i, col in enumerate(features):
    plt.subplot(2, 3, i + 1)
    df.groupby(col).mean()['count'].plot.bar()
plt.show()
```



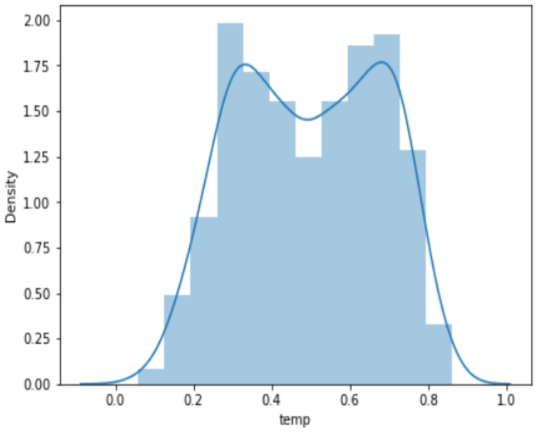
From the above bar plots, we can confirm some real-life observations:

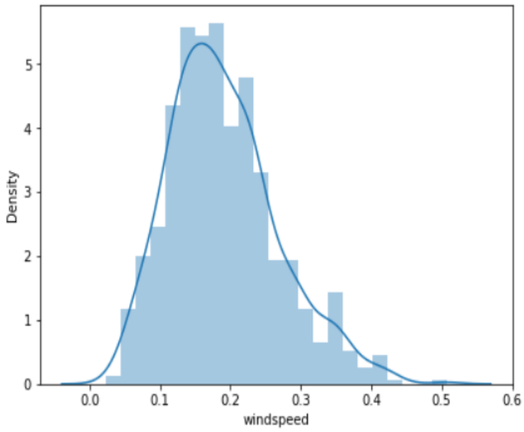
- Ride request demand is high in the summer as well as season.
- The third category was extreme weather conditions due to this people avoid taking bike rides and like to stay safe at home.
- On holidays no college or offices are open due to this ride request demand is low.

- More ride requests during working hours as compared to non-working hours.
- Bike ride requests have increased significantly from the year 2011 to the year 2012.

In [24]:
```
features = ['temp', 'windspeed']

plt.subplots(figsize=(15, 5))
for i, col in enumerate(features):
    plt.subplot(1, 2, i + 1)
    sb.distplot(df[col])
plt.show()
```

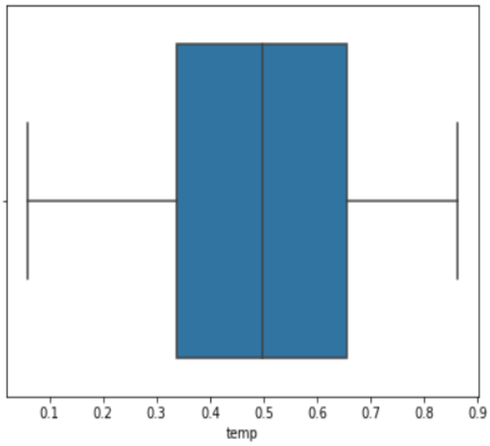


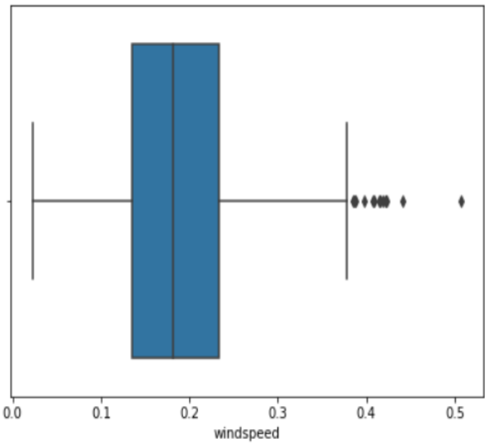


Temperature values are normally distributed but due to the high number of 0 entries in the windspeed column, the data distribution shows some irregularities.

In [25]:
```
features = ['temp', 'windspeed']

plt.subplots(figsize=(15, 5))
for i, col in enumerate(features):
    plt.subplot(1, 2, i + 1)
    sb.boxplot(df[col])
plt.show()
```





Ah! outliers let’s check how much data we will lose if we remove outliers.

temp windspeed

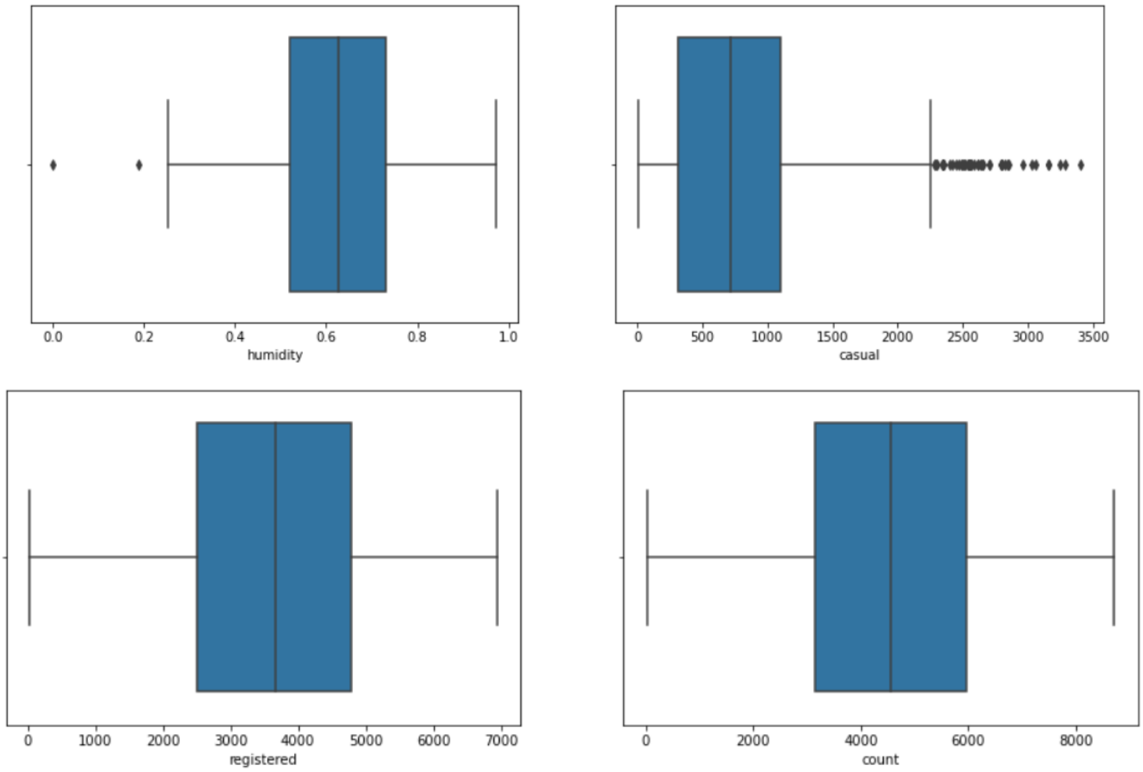
```
In [26]: num_rows = df.shape[0] - df[df['windspeed']<32].shape[0]
         print(f'Number of rows that will be lost if we remove outliers is equal to {num_rows}.')
```

```
Number of rows that will be lost if we remove outliers is equal to 0.
```

We can remove this many rows because we have around 0 rows of data so, this much data loss won't affect the learning for our model.

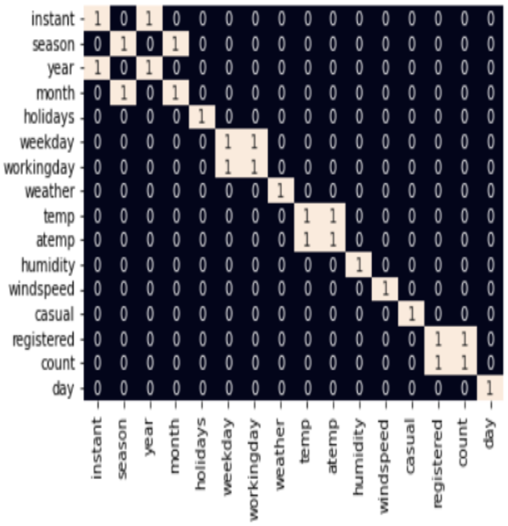
```
In [27]: features = ['humidity', 'casual', 'registered', 'count']

         plt.subplots(figsize=(15, 10))
         for i, col in enumerate(features):
             plt.subplot(2, 2, i + 1)
             sb.boxplot(df[col])
         plt.show()
```



Now let's check whether there are any highly correlated features in our dataset or not.

```
In [28]: sb.heatmap(df.corr() > 0.8,annot=True,cbar=False)
         plt.show()
```



Here the registered feature is highly correlated with our target variable which is count. This will lead to a situation of data leakage if we do not handle this situation. So, let’s remove this ‘registered’ column from our feature set.
Now, we have to remove the outliers we found in the above two observations that are for the humidity and wind speed.

```
In [29]: df.drop(['registered'], axis=1, inplace=True)
         df = df[(df['windspeed'] < 32) & (df['humidity'] > 0)]
         features = df.drop(['count'], axis=1)
         target = df['count'].values
         df.head()
```

Out[29]:

	instant	season	year	month	holidays	weekday	workingday	weather	temp	atemp	humidity	windspeed	casual	count	day
0	1	1	2011	1	0	0	0	2	0.344167	0.363625	0.805833	0.160446	331	985	1
1	2	1	2011	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	801	2
2	3	1	2011	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1349	3
3	4	1	2011	1	0	1	1	1	0.200000	0.212122	0.590435	0.160296	108	1562	4
4	5	1	2011	1	1	1	1	1	0.226957	0.229270	0.436957	0.186900	82	1600	5

Model Training

Now we will separate the features and target variables and split them into training and the testing data by using which we will select the model which is performing best on the validation data.

```
In [30]: X_train, X_val, Y_train, Y_val = train_test_split(features,target,test_size = 0.1,random_state=22)
         print(X_train.shape, X_val.shape)
         scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_val = scaler.transform(X_val)
```

```
(657, 14) (73, 14)
```

Normalizing the data before feeding it into machine learning models helps us to achieve stable and fast training.

Now we have to , We have split our data into training and validation data also the normalization of the data has been done. Now let’s train some state-of-the-art machine learning models and select the best out of them using the validation dataset.

```
In [33]: for i in range(5):
             models[i].fit(X_train, Y_train)

             print(f'{models[i]} : ')

             train_preds = models[i].predict(X_train)
             print('Training Error : ', mae(Y_train, train_preds))

             val_preds = models[i].predict(X_val)
             print('Validation Error : ', mae(Y_val, val_preds))
             print()
```

```
LinearRegression() :
Training Error :  471.59432421294855
Validation Error :  507.1428704051535

XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
             grow_policy='depthwise', importance_type=None,
             interaction_constraints='', learning_rate=0.300000012, max_bin=256,
             max_cat_threshold=64, max_cat_to_onehot=4, max_delta_step=0,
             max_depth=6, max_leaves=0, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, ...) :
Training Error :  7.528764806018754
Validation Error :  346.4679222629495

Lasso() :
Training Error :  472.9170003439712
Validation Error :  497.5230563613014

RandomForestRegressor() :
Training Error :  144.1314459665145
Validation Error :  367.09890410958906

Ridge() :
Training Error :  472.20463457605354
Validation Error :  501.28463356945826
```

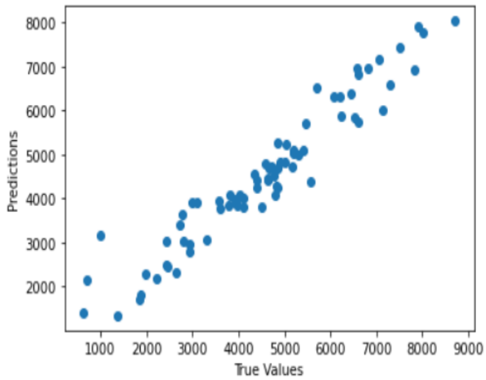
Now we predict the values of test set.

In [34]:
```
val_preds = models[3].predict(X_val)
val_preds
```

Out[34]:
```
array([2780.89, 7436.31, 2457.53, 5705.59, 4566.38, 3804.46, 1338.41,
       1823.82, 2486.85, 4256.54, 4829.18, 3041.64, 3177.89, 6529.22,
       3774.38, 4396.09, 1722.29, 3831.15, 3898.11, 6321.25, 1414.68,
       3946.77, 6914.59, 3649.46, 7765.34, 5235.85, 3826.16, 5106.57,
       4087.09, 4010.14, 5834.12, 4070.4 , 3991.27, 3038.4 , 4425.37,
       2947.54, 7929.95, 4677.9 , 4410.29, 3841.73, 2135.11, 4731.6 ,
       5098.59, 4466.79, 4298.76, 6396.34, 6024.84, 6834.91, 2286.7 ,
       5040.27, 6585.42, 5754.12, 2190.49, 7175.59, 5868.94, 3908.34,
       2331.4 , 8040.53, 6974.26, 4821.82, 5005.99, 4241.4 , 6955.94,
       4079.87, 4717.95, 4797.14, 3403.47, 4519.44, 3063.64, 3898.52,
       6313.32, 4699.3 , 5281.98])
```

In [36]:
```
plt.scatter(Y_val, val_preds)
plt.xlabel("True Values")
plt.ylabel("Predictions")
```

Out[36]: Text(0, 0.5, 'Predictions')



Here we check that how accurate our data is:

In [39]:
```
for i in range(5):
    models[i].fit(X_train, Y_train)
    print(f'{models[i]} : ')
    print('score:', models[i].score(X_val, Y_val))
```

```
LinearRegression() :
score: 0.8588748964578496
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
             grow_policy='depthwise', importance_type=None,
             interaction_constraints='', learning_rate=0.300000012, max_bin=256,
             max_cat_threshold=64, max_cat_to_onehot=4, max_delta_step=0,
             max_depth=6, max_leaves=0, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, ...) :
score: 0.9342839933689902
Lasso() :
score: 0.8645907035078849
RandomForestRegressor() :
score: 0.9176294678656121
Ridge() :
score: 0.8629785006457908
```

The predictions made by the XGBRegressor are really amazing compared to the other model. In the case of XGBRegressor, there is a little bit of overfitting but we can manage it by hyperparameter tuning.

GITHUB LINK: -

https://github.com/Rockingrishu/ML-Project.git

REFERENCES

- GOOGLE.COM
- TUTORIALSPOINT.COM
- GEEKFORGEEKS.COM
- WIKIPEDIA
- YOUTUBE