Name: Sandeep Vishwakarma
FEYNN LABS
Online Vehicle Booking Market

- **Please find the descriptions of the columns present in the dataset as below.**

1. **datetime - hourly date + timestamp**
2. **season - spring, summer, autumn, winter**
3. **holiday - whether the day is considered a holiday**
4. **workingday - whether the day is neither a weekend nor holiday**
5. **weather - Clear , Cloudy, Light Rain, Heavy**
6. **temp - temperature in Celsius**
7. **atemp - "feels like" temperature in Celsius**
8. **humidity - relative humidity**
9. **windspeed - wind speed**
10. **Total_booking - number of total booking**

In [3]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import math
%matplotlib inline


from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from xgboost import XGBRFRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

In [4]:

```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\gf63\anaconda3\lib\site-packages (1.7.
3)
Requirement already satisfied: numpy in c:\users\gf63\anaconda3\lib\site-packages (from x
gboost) (1.20.3)
Requirement already satisfied: scipy in c:\users\gf63\anaconda3\lib\site-packages (from x
gboost) (1.7.1)
```

# DATA COLLECTION AND PREPERATION

In [5]:

```
df_train= pd.read_csv('train.csv')
```

In [6]:

```
col=['Total_booking']
df_train_label= pd.read_csv('train_label.csv',header=None,names=col)
```

In [7]:

```
#appending the train_label dataset to train.csv as 'Total_booking' column.
train=df_train.join(df_train_label)
```

In [8]:

```
train
```

Out[8]:

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | Total_booking |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 5/02/2012 19:00 | Summer | 0 | 1 | Clear + Few clouds | 22.14 | 25.760 | 77 | 16.9979 | 504 |
| **1** | 9/05/2012 4:00 | Fall | 0 | 1 | Clear + Few clouds | 28.70 | 33.335 | 79 | 19.0012 | 5 |
| **2** | 1/13/2011 9:00 | Spring | 0 | 1 | Clear + Few clouds | 5.74 | 6.060 | 50 | 22.0028 | 139 |
| **3** | 11/18/2011 16:00 | Winter | 0 | 1 | Clear + Few clouds | 13.94 | 16.665 | 29 | 8.9981 | 209 |
| **4** | 9/13/2011 13:00 | Fall | 0 | 1 | Clear + Few clouds | 30.34 | 33.335 | 51 | 19.0012 | 184 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **8703** | 1/16/2012 6:00 | Spring | 1 | 0 | Clear + Few clouds | 4.10 | 6.820 | 54 | 6.0032 | 13 |
| **8704** | 11/10/2011 1:00 | Winter | 0 | 1 | Mist + Cloudy | 16.40 | 20.455 | 87 | 0.0000 | 11 |
| **8705** | 4/12/2011 3:00 | Summer | 0 | 1 | Mist + Cloudy | 23.78 | 27.275 | 56 | 8.9981 | 1 |
| **8706** | 11/07/2012 1:00 | Winter | 0 | 1 | Mist + Cloudy | 11.48 | 13.635 | 61 | 16.9979 | 92 |
| **8707** | 1/10/2011 10:00 | Spring | 0 | 1 | Mist + Cloudy | 5.74 | 6.060 | 50 | 19.9995 | 31 |

8708 rows × 10 columns

In [9]:

```
df_test= pd.read_csv('test-project.csv')
```

In [10]:

```
col=['Total_booking']
df_test_label= pd.read_csv('test_label.csv',header=None,names=col)
```

In [11]:

```
#appending the train_label dataset to train.csv as 'Total_booking' column.
test=df_test.join(df_test_label)
```

In [12]:

```
test
```

Out[12]:

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | Total_booking |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 5/10/2012 11:00 | Summer | 0 | 1 | Clear + Few clouds | 21.32 | 25.000 | 48 | 35.0008 | 256 |
| **1** | 6/09/2012 7:00 | Summer | 0 | 0 | Clear + Few clouds | 23.78 | 27.275 | 64 | 7.0015 | 87 |
| **2** | 3/06/2011 20:00 | Spring | 0 | 0 | Light Snow, Light Rain | 11.48 | 12.120 | 100 | 27.9993 | 11 |
| **3** | 10/13/2011 11:00 | Winter | 0 | 1 | Mist + Cloudy | 25.42 | 28.790 | 83 | 0.0000 | 84 |
| **4** | 6/02/2012 12:00 | Summer | 0 | 0 | Clear + Few clouds | 25.42 | 31.060 | 43 | 23.9994 | 668 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2173** | 3/08/2012 3:00 | Spring | 0 | 1 | Clear + Few clouds | 18.86 | 22.725 | 63 | 26.0027 | 3 |

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | Total_booking |
|---|---|---|---|---|---|---|---|---|---|---|
| **2174** | 1/12/2012 12:00 | Spring | 0 | 1 | Mist + Cloudy | 13.94 | 17.425 | 81 | 7.0015 | 144 |
| **2175** | 3/07/2012 22:00 | Spring | 0 | 1 | Clear + Few clouds | 18.86 | 22.725 | 59 | 19.9995 | 159 |
| **2176** | 5/12/2011 5:00 | Summer | 0 | 1 | Clear + Few clouds | 17.22 | 21.210 | 94 | 8.9981 | 29 |
| **2177** | 7/18/2012 16:00 | Fall | 0 | 1 | Clear + Few clouds | 30.34 | 34.850 | 66 | 16.9979 | 224 |

2178 rows × 10 columns

In [13]:

```
#feature engineering
from datetime import datetime
import calendar
```

In [14]:

```
# Creating new columns from date time column
train['date']= train.datetime.apply(lambda x : x.split()[0])
train['hour']= train.datetime.apply(lambda x : x.split()[1].split(":")[0])
train["weekday"]= train.date.apply(lambda dateString : calendar.day_name[datetime.strptim
e(dateString,"%m/%d/%Y").weekday()])
train["month"]= train.date.apply(lambda dateString : calendar.month_name[datetime.strpti
me(dateString,"%m/%d/%Y").month])

test['date']= test.datetime.apply(lambda x : x.split()[0])
test['hour']= test.datetime.apply(lambda x : x.split()[1].split(":")[0])
test["weekday"]= test.date.apply(lambda dateString : calendar.day_name[datetime.strptime(
dateString,"%m/%d/%Y").weekday()])
test["month"]= test.date.apply(lambda dateString : calendar.month_name[datetime.strptime(
dateString,"%m/%d/%Y").month])
```

In [15]:

```
train
```

Out[15]:

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | Total_booking | date | hou |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 5/02/2012 19:00 | Summer | 0 | 1 | Clear + Few clouds | 22.14 | 25.760 | 77 | 16.9979 | 504 | 5/02/2012 | 1 |
| **1** | 9/05/2012 4:00 | Fall | 0 | 1 | Clear + Few clouds | 28.70 | 33.335 | 79 | 19.0012 | 5 | 9/05/2012 | |
| **2** | 1/13/2011 9:00 | Spring | 0 | 1 | Clear + Few clouds | 5.74 | 6.060 | 50 | 22.0028 | 139 | 1/13/2011 | |
| **3** | 11/18/2011 16:00 | Winter | 0 | 1 | Clear + Few clouds | 13.94 | 16.665 | 29 | 8.9981 | 209 | 11/18/2011 | 1 |
| **4** | 9/13/2011 13:00 | Fall | 0 | 1 | Clear + Few clouds | 30.34 | 33.335 | 51 | 19.0012 | 184 | 9/13/2011 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **8703** | 1/16/2012 6:00 | Spring | 1 | 0 | Clear + Few clouds | 4.10 | 6.820 | 54 | 6.0032 | 13 | 1/16/2012 | |
| **8704** | 11/10/2011 1:00 | Winter | 0 | 1 | Mist + Cloudy | 16.40 | 20.455 | 87 | 0.0000 | 11 | 11/10/2011 | |
| **8705** | 4/12/2011 3:00 | Summer | 0 | 1 | Mist + Cloudy | 23.78 | 27.275 | 56 | 8.9981 | 1 | 4/12/2011 | |
| **8706** | 11/07/2012 | Winter | 0 | 1 | Mist + | 11.48 | 13.635 | 61 | 16.9979 | 92 | 11/07/2012 | |

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | Total_booking | date | hou |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8707 | 1/10/2011 10:00 | Spring | 0 | 1 | Mist + Cloudy | 5.74 | 6.060 | 50 | 19.9995 | 31 | 1/10/2011 | 1 |

8708 rows × 14 columns

# Task 1:

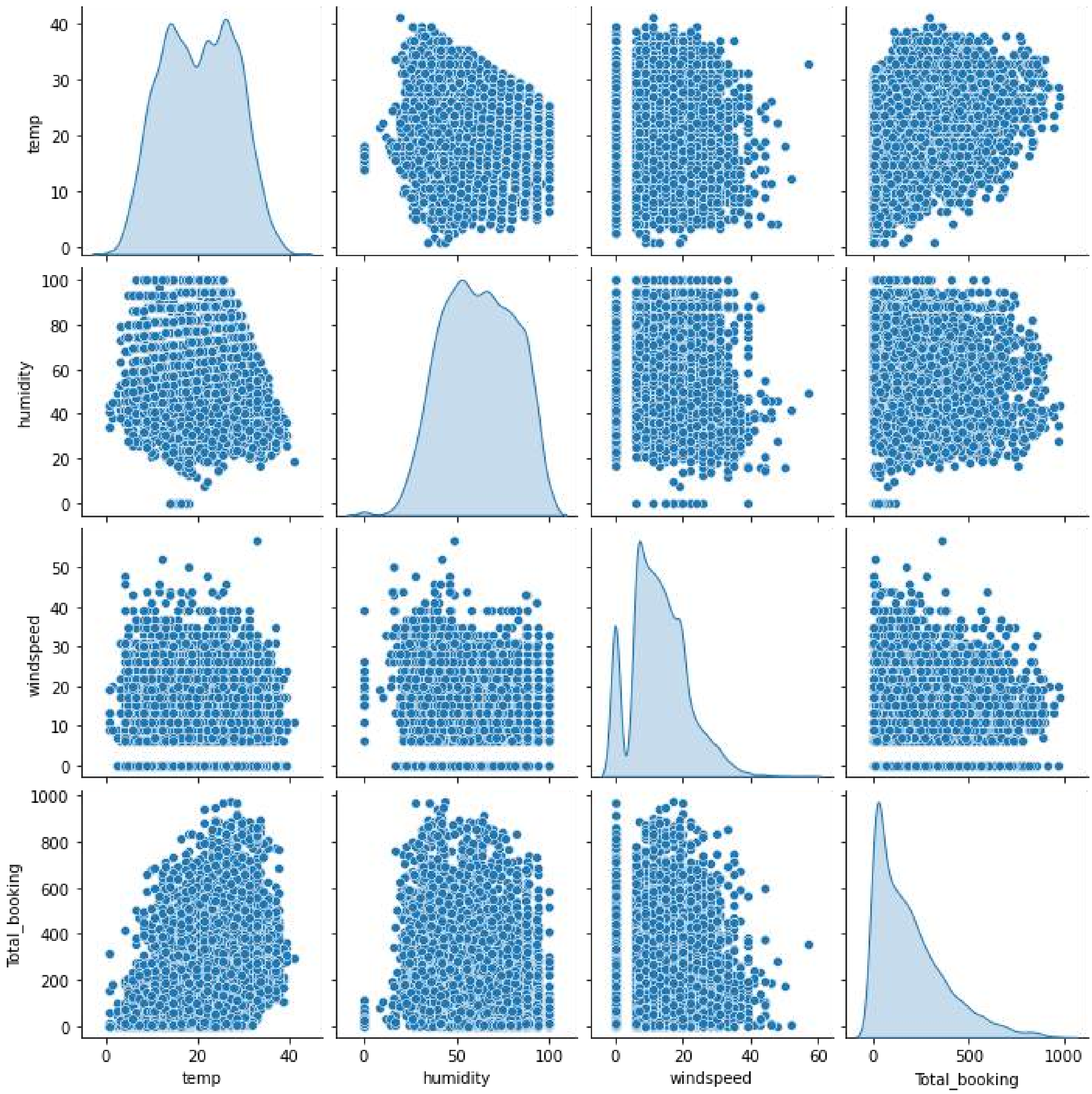
# 1. Visualize data using different visualizations to generate interesting insights

In [16]:

```
#Explore data
sns.pairplot(train[["temp","humidity","windspeed","Total_booking"]], diag_kind ='kde')
```

Out[16]:

<seaborn.axisgrid.PairGrid at 0x1f1e7ee7670>




# 2. Outlier Analysis

In [17]:

```
train.describe()
```

Out[17]:

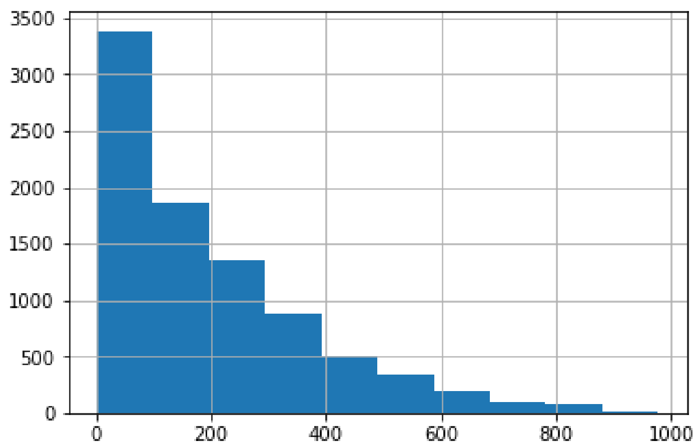
| | holiday | workingday | temp | atemp | humidity | windspeed | Total_booking |
|---|---|---|---|---|---|---|---|
| **count** | 8708.000000 | 8708.000000 | 8708.000000 | 8708.000000 | 8708.000000 | 8708.000000 | 8708.000000 |
| **mean** | 0.028480 | 0.680294 | 20.243020 | 23.655552 | 61.649173 | 12.843891 | 193.007005 |
| **std** | 0.166348 | 0.466389 | 7.813494 | 8.501107 | 19.199578 | 8.190314 | 181.552211 |
| **min** | 0.000000 | 0.000000 | 0.820000 | 0.760000 | 0.000000 | 0.000000 | 1.000000 |
| **25%** | 0.000000 | 0.000000 | 13.940000 | 16.665000 | 46.000000 | 7.001500 | 43.000000 |
| **50%** | 0.000000 | 1.000000 | 20.500000 | 24.240000 | 61.000000 | 12.998000 | 148.000000 |
| **75%** | 0.000000 | 1.000000 | 26.240000 | 31.060000 | 77.000000 | 16.997900 | 286.000000 |
| **max** | 1.000000 | 1.000000 | 41.000000 | 45.455000 | 100.000000 | 56.996900 | 977.000000 |

In [18]:

```
train.Total_booking.hist()
```

Out[18]:

```
<AxesSubplot:>
```




In [19]:

```
rev_stat=train.Total_booking.describe()
print(rev_stat)
# calculating interquartile range
iqr=rev_stat['75%']-rev_stat['25%']
upper=rev_stat['75%']+1.5*iqr
lower=rev_stat['25%']-1.5*iqr
print('iqr=',iqr)
print('The upper and lower bounds for suspected outliers are {} and {}'.format(lower,uppe
r))
```

```
count     8708.000000
mean       193.007005
std        181.552211
min          1.000000
25%         43.000000
50%        148.000000
75%        286.000000
max        977.000000
Name: Total_booking, dtype: float64
iqr= 243.0
The upper and lower bounds for suspected outliers are -321.5 and 650.5
```

In [20]:

```
outliers=train[train.Total_booking>upper].index.tolist()
```

```
train[train.Total_booking>upper]
```

Out[20]:

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | Total_booking | date | hou |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 8/17/2012 17:00 | Fall | 0 | 1 | Clear + Few clouds | 33.62 | 12.120 | 36 | 22.0028 | 791 | 8/17/2012 | 1 |
| 27 | 5/02/2012 18:00 | Summer | 0 | 1 | Clear + Few clouds | 22.96 | 26.515 | 73 | 15.0013 | 813 | 5/02/2012 | 1 |
| 30 | 7/10/2012 17:00 | Fall | 0 | 1 | Clear + Few clouds | 31.98 | 35.605 | 49 | 12.9980 | 872 | 7/10/2012 | 1 |
| 52 | 6/05/2012 18:00 | Summer | 0 | 1 | Clear + Few clouds | 24.60 | 31.060 | 43 | 12.9980 | 790 | 6/05/2012 | 1 |
| 70 | 5/18/2012 18:00 | Summer | 0 | 1 | Clear + Few clouds | 26.24 | 31.060 | 38 | 15.0013 | 669 | 5/18/2012 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | . |
| 8478 | 3/14/2012 8:00 | Spring | 0 | 1 | Clear + Few clouds | 18.04 | 21.970 | 82 | 0.0000 | 662 | 3/14/2012 | |
| 8582 | 10/15/2012 8:00 | Winter | 0 | 1 | Mist + Cloudy | 24.60 | 30.305 | 64 | 26.0027 | 737 | 10/15/2012 | |
| 8676 | 5/12/2012 13:00 | Summer | 0 | 0 | Clear + Few clouds | 26.24 | 31.060 | 36 | 12.9980 | 659 | 5/12/2012 | 1 |
| 8681 | 8/08/2012 17:00 | Fall | 0 | 1 | Mist + Cloudy | 32.80 | 37.880 | 55 | 19.0012 | 858 | 8/08/2012 | 1 |
| 8695 | 10/11/2012 17:00 | Winter | 0 | 1 | Clear + Few clouds | 20.50 | 24.240 | 39 | 19.0012 | 827 | 10/11/2012 | 1 |

242 rows × 14 columns

In [21]:

```
#Original data with outliers
train.shape
```

Out[21]:

```
(8708, 14)
```

In [22]:

```
for index in outliers:
    train.drop(index,inplace=True)
```

In [23]:

```
# After removing outliers
train.shape
```

Out[23]:

```
(8466, 14)
```

# 3. Missing value analysis

In [24]:

```
#we observed there is no missing values in our dataset
train.isnull().sum()
```

Out[24]:

```
datetime         0
season           0
holiday          0
workingday       0
weather          0
temp             0
atemp            0
humidity         0
windspeed        0
Total_booking    0
date             0
hour             0
weekday          0
month            0
dtype: int64
```

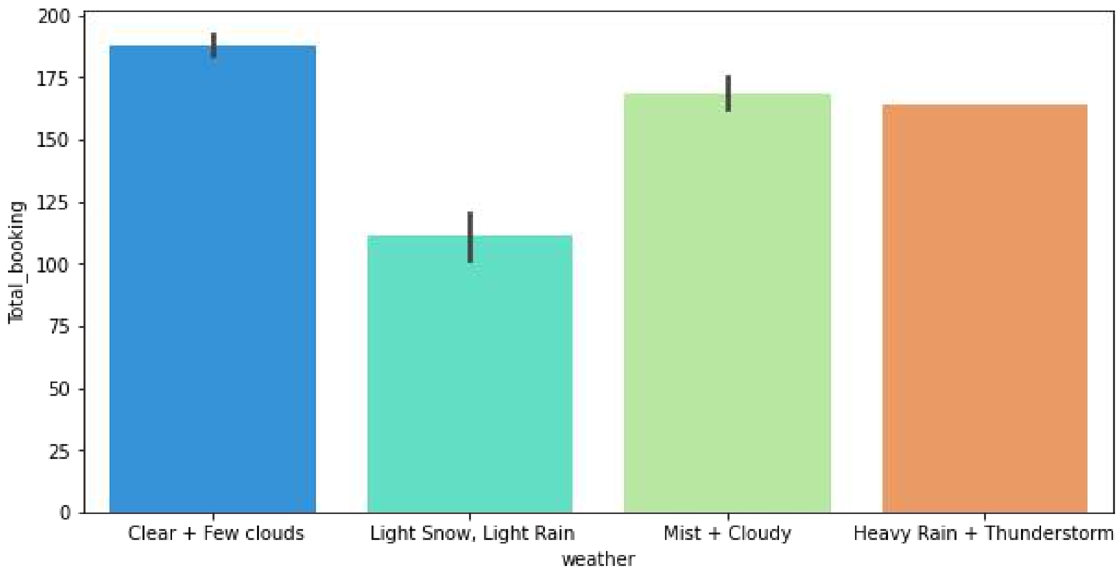
# 4. Visualizing Total_booking Vs other features to generate insights

In [25]:

```
plt.figure(figsize=(10,5))
sns.barplot(x = 'weather', y = 'Total_booking', data = train,palette='rainbow')
```

Out[25]:

```
<AxesSubplot:xlabel='weather', ylabel='Total_booking'>
```

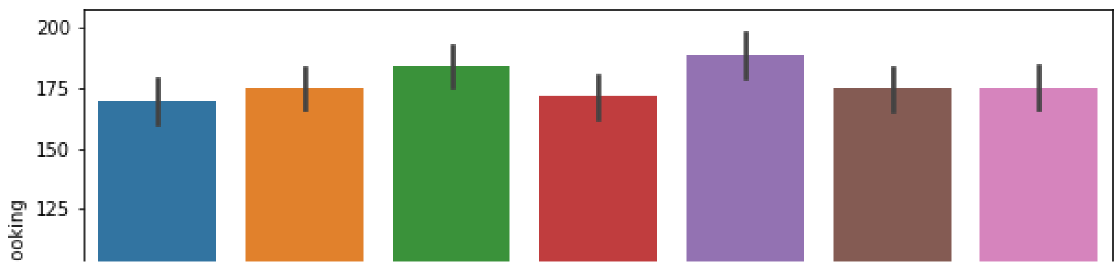


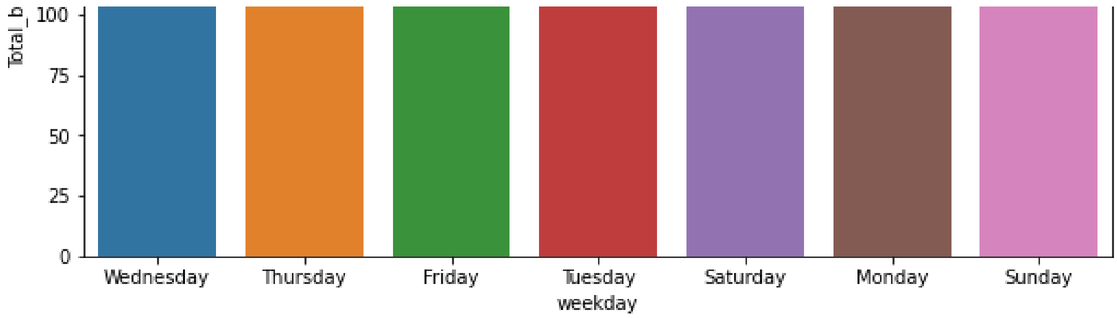

In [26]:

```
plt.figure(figsize=(10,5))
sns.barplot(x = 'weekday', y = 'Total_booking', data = train)
```

Out[26]:

```
<AxesSubplot:xlabel='weekday', ylabel='Total_booking'>
```



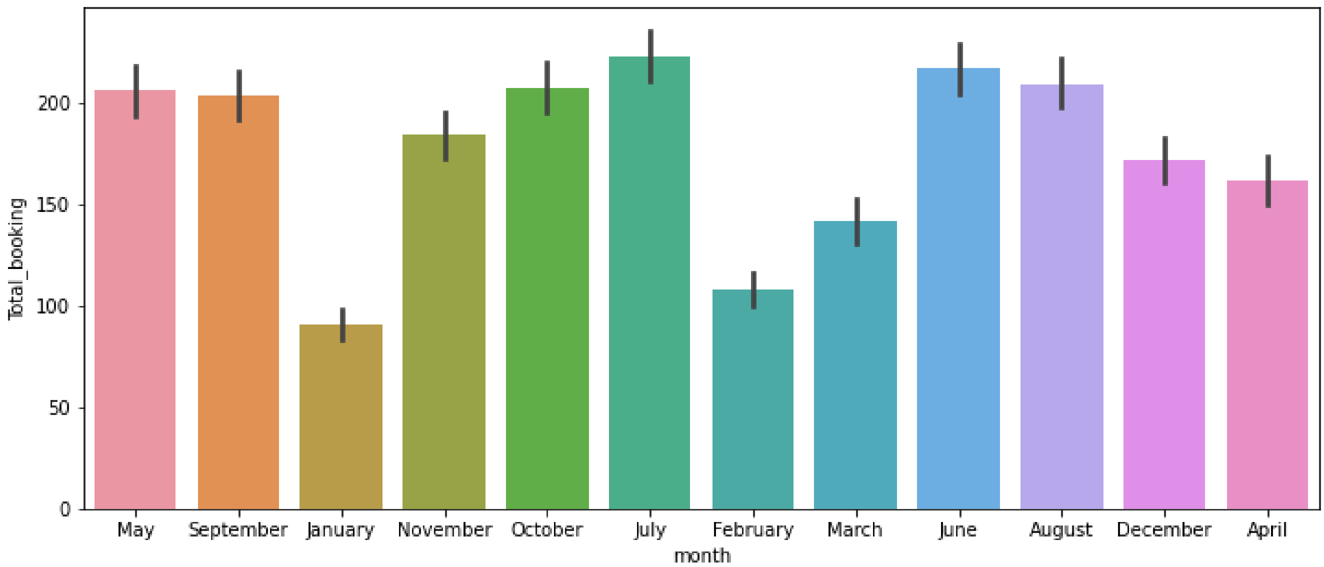



In [27]:

```
plt.figure(figsize=(12,5))
sns.barplot(x = 'month', y = 'Total_booking', data = train)
```

Out[27]:

```
<AxesSubplot:xlabel='month', ylabel='Total_booking'>
```

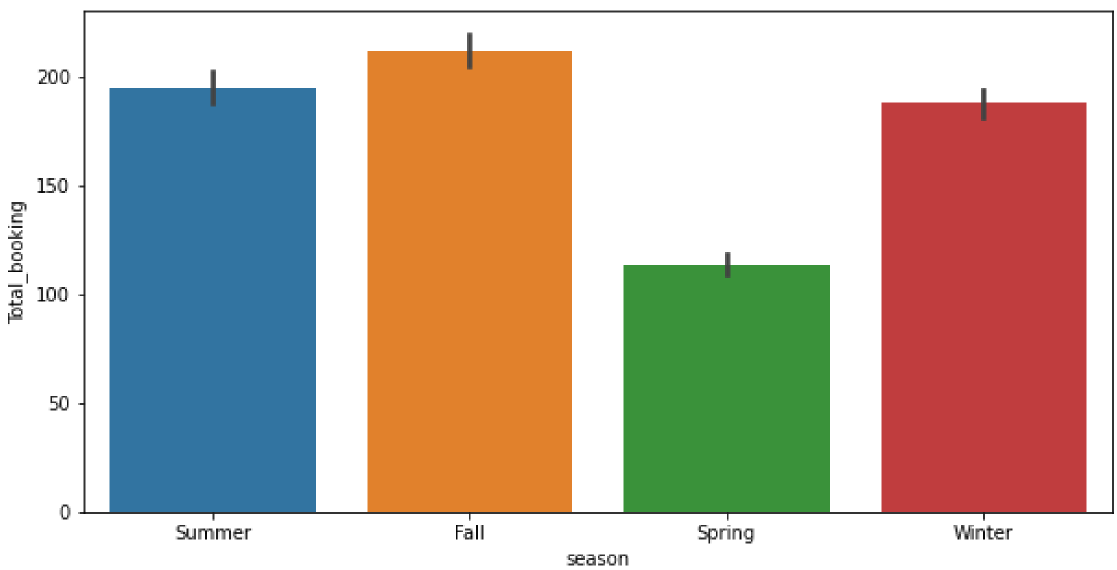



In [28]:

```
plt.figure(figsize=(10,5))
sns.barplot(x = 'season', y = 'Total_booking', data = train)
```

Out[28]:

```
<AxesSubplot:xlabel='season', ylabel='Total_booking'>
```




# 5. Correlation Analysis

In [29]:

```
correlation = train.corr()
fig = plt.figure(figsize = (12,10))
sns.heatmap(correlation, cmap ='viridis', annot = True, vmax = 1, square = True,vmin=-1)
plt.show()
```

```
C:\Users\gf63\AppData\Local\Temp/ipykernel_19204/2368513804.py:1: FutureWarning: The defa
ult value of numeric_only in DataFrame.corr is deprecated. In a future version, it will d
efault to False. Select only valid columns or specify the value of numeric_only to silenc
e this warning.
  correlation = train.corr()
```


| | holiday | workingday | temp | atemp | humidity | windspeed | Total_booking |
|---|---|---|---|---|---|---|---|
| holiday | 1 | -0.25 | 0.0026 | -0.0028 | -0.0061 | 0.0074 | 0.01 |
| workingday | -0.25 | 1 | 0.026 | 0.021 | -0.011 | 0.016 | -0.023 |
| temp | 0.0026 | 0.026 | 1 | 0.99 | -0.052 | -0.031 | 0.39 |
| atemp | -0.0028 | 0.021 | 0.99 | 1 | -0.031 | -0.073 | 0.39 |
| humidity | -0.0061 | -0.011 | -0.052 | -0.031 | 1 | -0.32 | -0.31 |
| windspeed | 0.0074 | 0.016 | -0.031 | -0.073 | -0.32 | 1 | 0.1 |
| Total_booking | 0.01 | -0.023 | 0.39 | 0.39 | -0.31 | 0.1 | 1 |

In [30]:

```
del(train["atemp"]) # remove to avoid multicollinearity- Temp & atemp are highly correlat
ed
```

# Task 2:

# 1. Feature Engineering

In [31]:

```
# Creating new columns from date time column
train['date']= train.datetime.apply(lambda x : x.split()[0])
train['hour']= train.datetime.apply(lambda x : x.split()[1].split(":")[0])
train["weekday"]= train.date.apply(lambda dateString : calendar.day_name[datetime.strptim
e(dateString,"%m/%d/%Y").weekday()])
train["month"]= train.date.apply(lambda dateString : calendar.month_name[datetime.strpti
me(dateString,"%m/%d/%Y").month])

test['date']= test.datetime.apply(lambda x : x.split()[0])
```

```
test['hour']= test.datetime.apply(lambda x : x.split()[1].split(":")[0])
test["weekday"]= test.date.apply(lambda dateString : calendar.day_name[datetime.strptime(
dateString,"%m/%d/%Y").weekday()])
test["month"]= test.date.apply(lambda dateString : calendar.month_name[datetime.strptime(
dateString,"%m/%d/%Y").month])
```

In [32]:

```
del(train["datetime"])
```

In [33]:

```
del(train["date"])
```

In [34]:

```
train.dtypes
```

Out[34]:

```
season            object
holiday            int64
workingday         int64
weather           object
temp             float64
humidity           int64
windspeed        float64
Total_booking      int64
hour              object
weekday           object
month             object
dtype: object
```

In [35]:

```
train.columns.nunique()
```

Out[35]:

```
11
```

In [36]:

```
train = pd.get_dummies(train)
train.head()
```

Out[36]:

| | holiday | workingday | temp | humidity | windspeed | Total_booking | season_Fall | season_Spring | season_Summer | season_Wint |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 22.14 | 77 | 16.9979 | 504 | 0 | 0 | 1 | |
| **1** | 0 | 1 | 28.70 | 79 | 19.0012 | 5 | 1 | 0 | 0 | |
| **2** | 0 | 1 | 5.74 | 50 | 22.0028 | 139 | 0 | 1 | 0 | |
| **3** | 0 | 1 | 13.94 | 29 | 8.9981 | 209 | 0 | 0 | 0 | |
| **4** | 0 | 1 | 30.34 | 51 | 19.0012 | 184 | 1 | 0 | 0 | |

**5 rows × 57 columns**

In [37]:

```
train.shape
```

Out[37]:

```
(8466, 57)
```

# Splitting into X and y

In [38]:

```
X = train.drop('Total_booking', axis =1)
y = train['Total_booking']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state
= 0)
print(X_train.shape)
print(X_test.shape)
```

```
(6772, 56)
(1694, 56)
```

# 2.Regression Analysis

In [39]:

```
# Decision Tree Model
DT_model = DecisionTreeRegressor()
DT_model.fit(X_train,y_train)
y_pred_DT = DT_model.predict(X_test)
y_pred_train = DT_model.predict(X_train)

# RANDOM FOREST
RF_model = RandomForestRegressor( random_state = 0)
RF_model.fit(X_train,y_train)
y_pred_RF = RF_model.predict(X_test)
y_pred_train = RF_model.predict(X_train)


# XG BOOST
XG_model = XGBRFRegressor()
XG_model.fit(X_train, y_train)
y_pred_XG = XG_model.predict(X_test)

# SVM
SVM_model = SVR()
SVM_model.fit(X_train, y_train)
y_pred_SVM = SVM_model.predict(X_test)

# KNN
KNN_model = KNeighborsRegressor()
KNN_model.fit(X_train, y_train)
y_pred_KNN = KNN_model.predict(X_test)

# Gradient Boosting
GB_model = GradientBoostingRegressor()
GB_model.fit(X_train, y_train)
y_pred_GB = GB_model.predict(X_test)
```

# PERFORMANCE METRICS

In [ ]:

```
from sklearn.metrics import r2_score, mean_squared_error
print("R squared value for DT :", r2_score(y_test, y_pred_DT))
print("R squared value for RF:", r2_score(y_test, y_pred_RF))
print("R squared value for XG:", r2_score(y_test, y_pred_XG))
print("R squared value for SVM:", r2_score(y_test, y_pred_SVM))
print("R squared value for KNN:", r2_score(y_test, y_pred_KNN))
print("R squared value for GB:", r2_score(y_test, y_pred_GB))


print("MSE for DT :", mean_squared_error(y_test, y_pred_DT))
print("MSE for RF:", mean_squared_error(y_test, y_pred_RF))
```

```
print("MSE for XG:", mean_squared_error(y_test, y_pred_XG))
print("MSE for SVM:", mean_squared_error(y_test, y_pred_SVM))
print("MSE for KNN:", mean_squared_error(y_test, y_pred_KNN))
print("MSE for GB:", mean_squared_error(y_test, y_pred_GB))
```

```
R squared value for DT : 0.6696310894622542
R squared value for RF: 0.8354197771760357
R squared value for XG: 0.5130508896569834
R squared value for SVM: 0.1679652652268958
R squared value for KNN: 0.23901017222862841
R squared value for GB: 0.7178058017450006
MSE for DT : 8295.258116883117
MSE for RF: 4132.45733999822
MSE for XG: 12226.842270076166
MSE for SVM: 20891.623476073764
MSE for KNN: 19107.751499409682
MSE for GB: 7085.635599917184
```

# 3. Ensemble Model

In [ ]:

```
# RANDOM FOREST-

from sklearn.ensemble import RandomForestRegressor
# 1000 trees, samples creation with replacement(bootsrap = true), n_jobs = -1 full proce
ssing of system

RF_reg = RandomForestRegressor(n_estimators = 1000, n_jobs = -1, random_state=0)

#fit the model
RF_reg = RF_reg.fit(X_train, y_train)

#Predict the model
y_pred_RFR = RF_reg.predict(X_test)

print("R squared:",r2_score(y_test,y_pred_RFR))
```

```
R squared: 0.8382149833105221
```

# 4. Grid search cv

In [ ]:

```
from sklearn.model_selection import GridSearchCV
# Create the parameter grid
param_grid = {
    'bootstrap': [True],
    'max_depth': [20,30,40,],
    'n_estimators': [300, 500, 1000, 1200]
}

# Instantiate the grid search model
grid_search = GridSearchCV(estimator = RF_reg, param_grid = param_grid,
                          cv = 3, n_jobs = -1, verbose = 2)
grid_search.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  36 out of  36 | elapsed:  7.3min finished
```

Out[ ]:

```
GridSearchCV(cv=3, error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                             criterion='mse', max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
```

```
                                                max_samples=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=1000, n_jobs=-1,
                                                oob_score=False, random_state=0,
                                                verbose=0, warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'bootstrap': [True], 'max_depth': [20, 30, 40],
                         'n_estimators': [300, 500, 1000, 1200]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=2)
```

In [ ]:

```
grid_search.best_params_
```

Out[ ]:

```
{'bootstrap': True, 'max_depth': 40, 'n_estimators': 1200}
```

In [ ]:

```
grid_search.best_estimator_
```

Out[ ]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=40, max_features='auto', max_leaf_nodes=None,
                      max_samples=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=1200, n_jobs=-1, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

In [ ]:

```
cvrf_grid = grid_search.best_estimator_
#Predict the model
y_pred_clf = cvrf_grid.predict(X_test)
print("R squared value for GridSearch :", r2_score(y_test, y_pred_clf))
print("MSE for GridSearch :", mean_squared_error(y_test, y_pred_clf))
```

```
R squared value for GridSearch : 0.8382697341494291
MSE for GridSearch : 4060.897553461924
```

In [ ]:

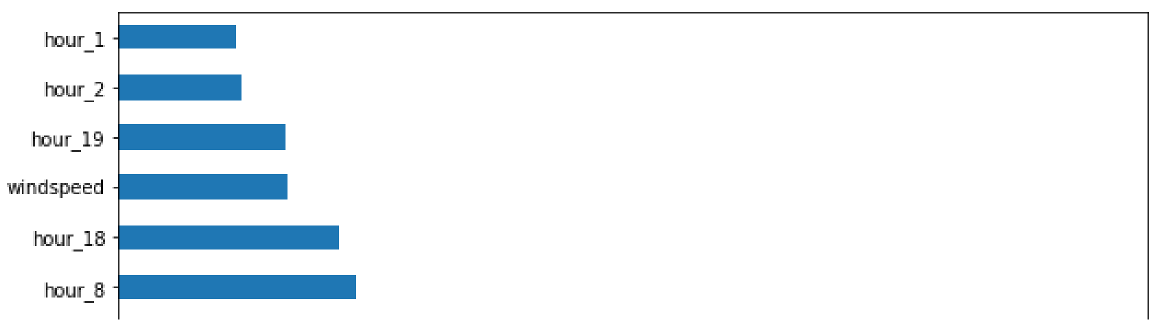
```

```

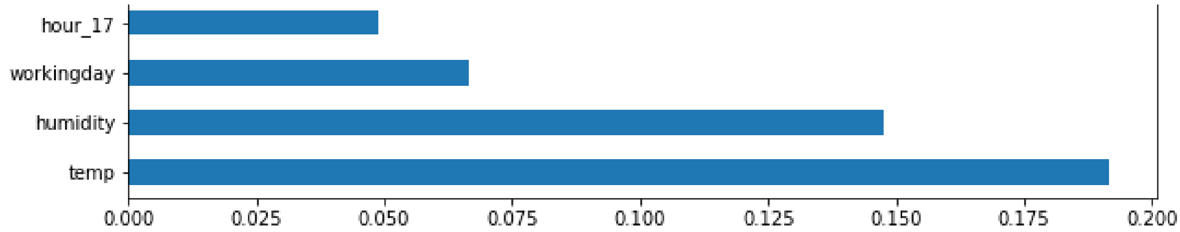
In [ ]:

```
# FEATURE IMPORTANCE
feat_importances = pd.Series(cvrf_grid.feature_importances_, index=X_train.columns)
f, ax = plt.subplots(figsize=(10,5))
feat_importances.nlargest(10).plot(kind='barh')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x476c2e8508>
```






# Comparing with the test-project dataset and predicting the output with grid search cv model

In [ ]:

```
test
```

Out[ ]:

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | Total_booking | date | hou |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 5/10/2012 11:00 | Summer | 0 | 1 | Clear + Few clouds | 21.32 | 25.000 | 48 | 35.0008 | 256 | 5/10/2012 | 1 |
| **1** | 6/09/2012 7:00 | Summer | 0 | 0 | Clear + Few clouds | 23.78 | 27.275 | 64 | 7.0015 | 87 | 6/09/2012 | |
| **2** | 3/06/2011 20:00 | Spring | 0 | 0 | Light Snow, Light Rain | 11.48 | 12.120 | 100 | 27.9993 | 11 | 3/06/2011 | 2 |
| **3** | 10/13/2011 11:00 | Winter | 0 | 1 | Mist + Cloudy | 25.42 | 28.790 | 83 | 0.0000 | 84 | 10/13/2011 | 1 |
| **4** | 6/02/2012 12:00 | Summer | 0 | 0 | Clear + Few clouds | 25.42 | 31.060 | 43 | 23.9994 | 668 | 6/02/2012 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **2173** | 3/08/2012 3:00 | Spring | 0 | 1 | Clear + Few clouds | 18.86 | 22.725 | 63 | 26.0027 | 3 | 3/08/2012 | |
| **2174** | 1/12/2012 12:00 | Spring | 0 | 1 | Mist + Cloudy | 13.94 | 17.425 | 81 | 7.0015 | 144 | 1/12/2012 | 1 |
| **2175** | 3/07/2012 22:00 | Spring | 0 | 1 | Clear + Few clouds | 18.86 | 22.725 | 59 | 19.9995 | 159 | 3/07/2012 | 2 |
| **2176** | 5/12/2011 5:00 | Summer | 0 | 1 | Clear + Few clouds | 17.22 | 21.210 | 94 | 8.9981 | 29 | 5/12/2011 | |
| **2177** | 7/18/2012 16:00 | Fall | 0 | 1 | Clear + Few clouds | 30.34 | 34.850 | 66 | 16.9979 | 224 | 7/18/2012 | 1 |

2178 rows × 14 columns

In [ ]:

```
del(test["datetime"])
```

In [ ]:

```
del(test["atemp"])
```

In [ ]:

```
del(test["date"])
```

In [ ]:

```
test.shape
```

Out[ ]:

```
(2178, 11)
```

In [ ]:

```
test.columns
```

Out[ ]:

```
Index(['season', 'holiday', 'workingday', 'weather', 'temp', 'humidity',
       'windspeed', 'Total_booking', 'hour', 'weekday', 'month'],
      dtype='object')
```

In [ ]:

```
test = pd.get_dummies(test)
test.head()
```

Out[ ]:

| | holiday | workingday | temp | humidity | windspeed | Total_booking | season_Fall | season_Spring | season_Summer | season_Wint |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 21.32 | 48 | 35.0008 | 256 | 0 | 0 | 1 | |
| **1** | 0 | 0 | 23.78 | 64 | 7.0015 | 87 | 0 | 0 | 1 | |
| **2** | 0 | 0 | 11.48 | 100 | 27.9993 | 11 | 0 | 1 | 0 | |
| **3** | 0 | 1 | 25.42 | 83 | 0.0000 | 84 | 0 | 0 | 0 | |
| **4** | 0 | 0 | 25.42 | 43 | 23.9994 | 668 | 0 | 0 | 1 | |

**5 rows × 56 columns**

In [ ]:

```
test.shape
```

Out[ ]:

```
(2178, 56)
```

In [ ]:

```
# Fitting Grid Search Model to test dataset

New_pred = cvrf_grid.predict(test)
print("R squared for prediction v/s test_label:",r2_score(df_test_label, New_pred))
print("MSE for prediction v/s test_label :", mean_squared_error(df_test_label, New_pred))
```

```
R squared for prediction v/s test_label: 0.797753772723576
MSE for prediction v/s test_label : 6508.497062539411
```

# Appending the prediction values column with the test-project dataset

In [ ]:

```
New_pred = pd.DataFrame(New_pred)
New_pred.columns = ['Predictions']
New_pred
```

Out[ ]:

| | Predictions |
|---|---|
| 0 | 214.892812 |
| 1 | 157.762059 |
| 2 | 77.260000 |
| 3 | 164.358417 |
| 4 | 444.000000 |
| ... | ... |
| 2173 | 12.273333 |
| 2174 | 138.090833 |
| 2175 | 135.948958 |
| 2176 | 24.306667 |
| 2177 | 381.943333 |

2178 rows × 1 columns

In [ ]:

```
df_test= pd.read_csv('test-project.csv')
col=['Total_booking']
df_test_label= pd.read_csv('test_label.csv',header=None,names=col)
test=df_test.join(df_test_label)
Dataset_testpred = test.join(New_pred)
Dataset_testpred
```

Out[ ]:

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | Total_booking | Predictions |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5/10/2012 11:00 | Summer | 0 | 1 | Clear + Few clouds | 21.32 | 25.000 | 48 | 35.0008 | 256 | 214.892812 |
| 1 | 6/09/2012 7:00 | Summer | 0 | 0 | Clear + Few clouds | 23.78 | 27.275 | 64 | 7.0015 | 87 | 157.762059 |
| 2 | 3/06/2011 20:00 | Spring | 0 | 0 | Light Snow, Light Rain | 11.48 | 12.120 | 100 | 27.9993 | 11 | 77.260000 |
| 3 | 10/13/2011 11:00 | Winter | 0 | 1 | Mist + Cloudy | 25.42 | 28.790 | 83 | 0.0000 | 84 | 164.358417 |
| 4 | 6/02/2012 12:00 | Summer | 0 | 0 | Clear + Few clouds | 25.42 | 31.060 | 43 | 23.9994 | 668 | 444.000000 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2173 | 3/08/2012 3:00 | Spring | 0 | 1 | Clear + Few clouds | 18.86 | 22.725 | 63 | 26.0027 | 3 | 12.273333 |
| 2174 | 1/12/2012 12:00 | Spring | 0 | 1 | Mist + Cloudy | 13.94 | 17.425 | 81 | 7.0015 | 144 | 138.090833 |
| 2175 | 3/07/2012 22:00 | Spring | 0 | 1 | Clear + Few clouds | 18.86 | 22.725 | 59 | 19.9995 | 159 | 135.948958 |
| 2176 | 5/12/2011 5:00 | Summer | 0 | 1 | Clear + Few clouds | 17.22 | 21.210 | 94 | 8.9981 | 29 | 24.306667 |
| 2177 | 7/18/2012 16:00 | Fall | 0 | 1 | Clear + Few clouds | 30.34 | 34.850 | 66 | 16.9979 | 224 | 381.943333 |

2178 rows × 11 columns

```
In [ ]:
```