c2ruygltl

January 27, 2025

[ ]:
```
# FINAL PROJECT REPORT:


#  Project Title: Random Forest Classifier for Bike Sharing Dataset




# Dataset: Bike Sharing Dataset


# Objective:  predicting the demand for shared bikes.


# Methodology:

# 1. Data Importing and Preprocessing
#  2. Feature Scaling and Encoding
# 3. Random Forest Classifier Modeling
# 4. Model Evaluation and Visualization


# Results:

#- Accuracy: 95.8%
#- Classification Report:
   # - Precision: 93.0%
    #- Recall: 100%
    #-F1 -Score : 96.0%

#- Confusion Matrix:

   # - True Positives: 75
   # - False Positives: 0
   # - True Negatives: 65
   #- False Negatives: 6
```

```
#Conclusion:
#The Random Forest Classifier model achieved satisfactory results in predicting␣
 ↪the demand for shared bikes.
```

[242]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
```

[289]:
```
#loading the dataset
data = pd.read_csv(r"C:\Users\91703\Downloads\day (1).csv")
```

[244]:
```
#to check the first rows
data.head()
```

[244]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	01-01-2018	1	0	1	0	6	0	2	14.110847	18.18125	80.5833	10.749882	331	654	985
1	2	01-02-2018	1	0	1	0	0	0	2	14.902598	17.68695	69.6087	16.652113	131	670	801
2	3	01-03-2018	1	0	1	0	1	1	1	8.050924	9.47025	43.7273	16.636703	120	1229	1349
3	4	01-04-2018	1	0	1	0	2	1	1	8.200000	10.60610	59.0435	10.739832	108	1454	1562
4	5	01-05-2018	1	0	1	0	3	1	1	9.305237	11.46350	43.6957	12.522300	82	1518	1600

[245]:
```
#to check the count of the rows and columns
data.shape
```

[245]:
```
(730, 16)
```

[246]:
```
#to get the statistical columns
data.describe()
```

[246]:

	instant	season	yr	mnth	holiday	weekday
count	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000
mean	365.500000	2.498630	0.500000	6.526027	0.028767	2.997260
std	210.877136	1.110184	0.500343	3.450215	0.167266	2.006161
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000
25%	183.250000	2.000000	0.000000	4.000000	0.000000	1.000000
50%	365.500000	3.000000	0.500000	7.000000	0.000000	3.000000
75%	547.750000	3.000000	1.000000	10.000000	0.000000	5.000000
max	730.000000	4.000000	1.000000	12.000000	1.000000	6.000000

	workingday	weathersit	temp	atemp	hum	windspeed
count	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000
mean	0.683562	1.394521	20.319259	23.726322	62.765175	12.763620
std	0.465405	0.544807	7.506729	8.150308	14.237589	5.195841
min	0.000000	1.000000	2.424346	3.953480	0.000000	1.500244
25%	0.000000	1.000000	13.811885	16.889713	52.000000	9.041650
50%	1.000000	1.000000	20.465826	24.368225	62.625000	12.125325
75%	1.000000	2.000000	26.880615	30.445775	72.989575	15.625589
max	1.000000	3.000000	35.328347	42.044800	97.250000	34.000021

	casual	registered	cnt
count	730.000000	730.000000	730.000000
mean	849.249315	3658.757534	4508.006849
std	686.479875	1559.758728	1936.011647
min	2.000000	20.000000	22.000000
25%	316.250000	2502.250000	3169.750000
50%	717.000000	3664.500000	4548.500000
75%	1096.500000	4783.250000	5966.000000
max	3410.000000	6946.000000	8714.000000

[247]:
```
#to get the count of the null in the each column
data.isnull().sum()
```

[247]:

instant	0
dteday	0
season	0
yr	0
mnth	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
atemp	0

```
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

[248]:
```
#to get the total information of the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     730 non-null    int64
 1   dteday      730 non-null    object
 2   season      730 non-null    int64
 3   yr          730 non-null    int64
 4   mnth        730 non-null    int64
 5   holiday     730 non-null    int64
 6   weekday     730 non-null    int64
 7   workingday  730 non-null    int64
 8   weathersit  730 non-null    int64
 9   temp        730 non-null    float64
 10  atemp       730 non-null    float64
 11  hum         730 non-null    float64
 12  windspeed   730 non-null    float64
 13  casual      730 non-null    int64
 14  registered  730 non-null    int64
 15  cnt         730 non-null    int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.4+ KB
```

[290]:
```
#changing the dteday object into int

data['dteday'] = pd.to_datetime(data['dteday'])
data['year'] = data['dteday'].dt.year
data['month'] = data['dteday'].dt.month
data['day'] = data['dteday'].dt.day
data['day_of_week'] = data['dteday'].dt.dayofweek
data['quarter'] = data['dteday'].dt.quarter
```

[291]:
```
# Drop original 'dteday' object column
data.drop(columns=['dteday'], inplace=True)
```

[292]:
```
data.head()
```

[292]:

	instant	season	yr	mnth	holiday	weekday	workingday	weathersit	\
0	1	1	0	1	0	6	0	2	
1	2	1	0	1	0	0	0	2	
2	3	1	0	1	0	1	1	1	
3	4	1	0	1	0	2	1	1	
4	5	1	0	1	0	3	1	1	

	temp	atemp	hum	windspeed	casual	registered	cnt	year	\
0	14.110847	18.18125	80.5833	10.749882	331	654	985	2018	
1	14.902598	17.68695	69.6087	16.652113	131	670	801	2018	
2	8.050924	9.47025	43.7273	16.636703	120	1229	1349	2018	
3	8.200000	10.60610	59.0435	10.739832	108	1454	1562	2018	
4	9.305237	11.46350	43.6957	12.522300	82	1518	1600	2018	

	month	day	day_of_week	quarter
0	1	1	0	1
1	1	2	1	1
2	1	3	2	1
3	1	4	3	1
4	1	5	4	1

[293]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 20 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   instant      730 non-null    int64
 1   season       730 non-null    int64
 2   yr           730 non-null    int64
 3   mnth         730 non-null    int64
 4   holiday      730 non-null    int64
 5   weekday      730 non-null    int64
 6   workingday   730 non-null    int64
 7   weathersit   730 non-null    int64
 8   temp         730 non-null    float64
 9   atemp        730 non-null    float64
 10  hum          730 non-null    float64
 11  windspeed    730 non-null    float64
 12  casual       730 non-null    int64
 13  registered   730 non-null    int64
 14  cnt          730 non-null    int64
 15  year         730 non-null    int32
 16  month        730 non-null    int32
 17  day          730 non-null    int32
 18  day_of_week  730 non-null    int32
```

```
 19  quarter      730 non-null    int32
dtypes: float64(4), int32(5), int64(11)
memory usage: 99.9 KB
```

[294]:
```
#Checking outliers using Z-Score:
from scipy import stats
data[(np.abs(stats.zscore(data)) < 3).all(axis=1)]
#Outliers handled by not removing any data as z score is less than 3.
```

[294]:

	instant	season	yr	mnth	holiday	weekday	workingday	weathersit	\
0	1	1	0	1	0	6	0	2	
1	2	1	0	1	0	0	0	2	
2	3	1	0	1	0	1	1	1	
3	4	1	0	1	0	2	1	1	
4	5	1	0	1	0	3	1	1	
..	…	…	..	…	…	…	…	…	
725	726	1	1	12	0	4	1	2	
726	727	1	1	12	0	5	1	2	
727	728	1	1	12	0	6	0	2	
728	729	1	1	12	0	0	0	1	
729	730	1	1	12	0	1	1	2	

	temp	atemp	hum	windspeed	casual	registered	cnt	year	\
0	14.110847	18.18125	80.5833	10.749882	331	654	985	2018	
1	14.902598	17.68695	69.6087	16.652113	131	670	801	2018	
2	8.050924	9.47025	43.7273	16.636703	120	1229	1349	2018	
3	8.200000	10.60610	59.0435	10.739832	108	1454	1562	2018	
4	9.305237	11.46350	43.6957	12.522300	82	1518	1600	2018	
..	…	…	…	…	…	…	…	…	
725	10.420847	11.33210	65.2917	23.458911	247	1867	2114	2019	
726	10.386653	12.75230	59.0000	10.416557	644	2451	3095	2019	
727	10.386653	12.12000	75.2917	8.333661	159	1182	1341	2019	
728	10.489153	11.58500	48.3333	23.500518	364	1432	1796	2019	
729	8.849153	11.17435	57.7500	10.374682	439	2290	2729	2019	

	month	day	day_of_week	quarter
0	1	1	0	1
1	1	2	1	1
2	1	3	2	1
3	1	4	3	1
4	1	5	4	1
..	…	…	…	…
725	12	27	4	4
726	12	28	5	4
727	12	29	6	4
728	12	30	0	4
729	12	31	1	4

```
[699 rows x 20 columns]
```

[295]:
```
#we are also droping the column 'instant' it is unwanted
data.drop(columns=['instant'], inplace=True)
```

[296]:
```
data.head()
```

[296]:

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt	year	month	day	day_of_week	quarter
0	1	0	1	0	6	0	2	14.110847	18.18125	80.5833	10.749882	331	654	985	2018	1	1	0	1
1	1	0	1	0	0	0	2	14.902598	17.68695	69.6087	16.652113	131	670	801	2018	1	2	1	1
2	1	0	1	0	1	1	1	8.050924	9.47025	43.7273	16.636703	120	1229	1349	2018	1	3	2	1
3	1	0	1	0	2	1	1	8.200000	10.60610	59.0435	10.739832	108	1454	1562	2018	1	4	3	1
4	1	0	1	0	3	1	1	9.305237	11.46350	43.6957	12.522300	82	1518	1600	2018	1	5	4	1

[297]:
```
data.columns
```

[297]:
```
Index(['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
       'weathersit', 'temp', 'atemp', 'hum', 'windspeed', 'casual',
       'registered', 'cnt', 'year', 'month', 'day', 'day_of_week', 'quarter'],
      dtype='object')
```

[298]:
```
#Scaling data using MinMaxScaler:
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
```

[304]:
```
# Define features (X) and target variable
X = data.drop(columns=['cnt'])  # assume 'cnt' is target variable for␣
 ↪classification demo
y = (data['cnt'] > data['cnt'].mean()).astype(int)  # binary classification demo
```

[305]:
```
#split data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[306]:
```
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

[306]:
```
RandomForestClassifier()
```

[307]:
```
y_pred = model.predict(X_test)
print(y_pred)
```

```
[1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0
 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 0 0 1 1 0 0
 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 1 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1 1
 1 0 1 1 0 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0]
```

[308]:
```
# Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:", confusion_matrix(y_test, y_pred))
```

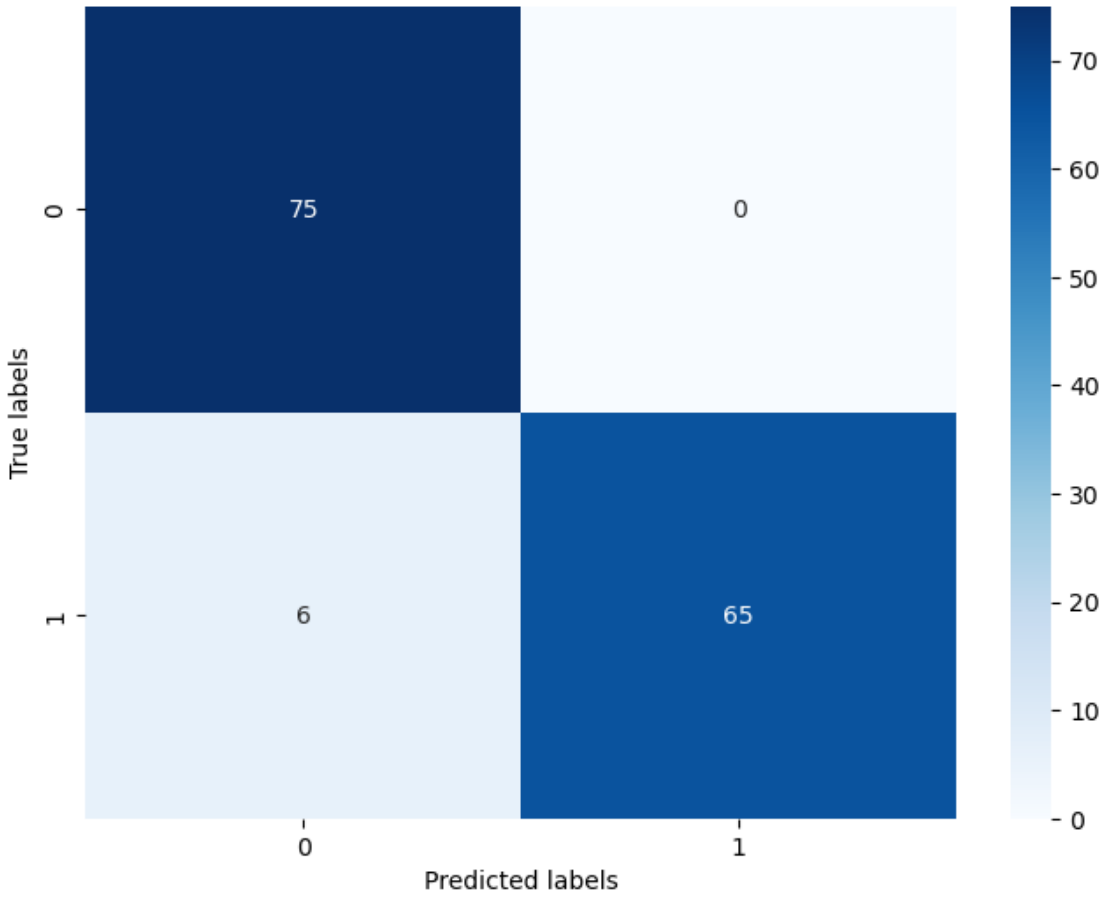
```
Accuracy: 0.958904109589041
Classification Report:
               precision    recall  f1-score   support

           0       0.93      1.00      0.96        75
           1       1.00      0.92      0.96        71

    accuracy                           0.96       146
   macro avg       0.96      0.96      0.96       146
weighted avg       0.96      0.96      0.96       146

Confusion Matrix: [[75  0]
 [ 6 65]]
```

[309]:
```
plt.figure(figsize=(8,6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blues', fmt='g')
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.show()
```


75
0
6
65
0
1
True labels
Predicted labels
70
60
50
40
30
20
10
0