RandomForest

July 9, 2025

Random Forest Classifier on Bike Dataset

[1]:
```
# Import necessary libraries
import pandas as pd
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, mean_squared_error,␣
 ↪classification_report
```

[2]:
```
# Load your dataset

df = pd.read_csv(r"C:\Users\risha\Desktop\Cleaned_Data_Used_Bike.csv")  #␣
 ↪Replace with file name/path
```

[3]:
```
# View the dataset is it load Correctly

df
```

[3]:

	bike_name	price	city	kms_driven	owner	age	power	brand
0	1	1	1	1	1	1	1	1
1	2	2	2	2	1	2	2	2
2	3	3	2	3	1	3	3	3
3	4	4	3	4	1	2	4	1
4	5	5	3	5	1	1	5	4
…	…	…	…	…	…	…	…	…
7319	471	43	2	3720	1	3	5	6
7320	27	1	3	523	1	10	13	7
7321	242	473	185	3721	1	2	25	15
7322	115	744	8	1666	1	2	26	7
7323	90	5	8	3722	1	6	13	7

[7324 rows x 8 columns]

[4]:
```
# Explore the dataset

print(df.head())  # View first few rows
print("_" * 153)
print(df.info())  # Get info about columns and data types
```

```
print("_" * 153)
print(df.describe())  # Statistical summary
```

```
   bike_name  price  city  kms_driven  owner  age  power  brand
0          1      1     1           1      1    1      1      1
1          2      2     2           2      1    2      2      2
2          3      3     2           3      1    3      3      3
3          4      4     3           4      1    2      4      1
4          5      5     3           5      1    1      5      4
```

```
-------------------------------------------------------------------------------
-------------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7324 entries, 0 to 7323
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   bike_name   7324 non-null   int64
 1   price       7324 non-null   int64
 2   city        7324 non-null   int64
 3   kms_driven  7324 non-null   int64
 4   owner       7324 non-null   int64
 5   age         7324 non-null   int64
 6   power       7324 non-null   int64
 7   brand       7324 non-null   int64
dtypes: int64(8)
memory usage: 457.9 KB
None
```

```
-------------------------------------------------------------------------------
-------------------------------------------------------------------------
         bike_name        price         city   kms_driven        owner  \
count  7324.000000  7324.000000  7324.000000  7324.000000  7324.000000
mean    108.185418   215.413708    38.499454  1064.275532     1.107318
std      97.578059   273.960942    75.332862  1154.178851     0.356299
min       1.000000     1.000000     1.000000     1.000000     1.000000
25%      34.000000    36.000000     3.000000   101.750000     1.000000
50%      77.000000   112.000000     8.000000   498.000000     1.000000
75%     167.000000   255.000000    31.000000  1954.250000     1.000000
max     471.000000  1232.000000   443.000000  3722.000000     4.000000

               age        power        brand
count  7324.000000  7324.000000  7324.000000
mean      5.796013     8.304615     5.538094
std       4.148665     6.680548     2.955179
min       1.000000     1.000000     1.000000
25%       2.000000     5.000000     3.000000
50%       5.000000     7.000000     6.000000
75%       7.000000    10.000000     7.000000
max      35.000000    53.000000    23.000000
```

[29]:
```
x = df.drop(['price'],axis=1) # Independent Variable (Input)
y = df['price'] # Dependent Variable (Output)
```

[27]:
```
x_train,x_test,y_train,y_test =␣
↪train_test_split(x,y,random_state=42,test_size=0.2)
```

[23]:
```
rfc = RandomForestClassifier(
    # Optional
    # n_estimators=50,
    # max_depth=10,
    # n_jobs=1,
    # random_state=42
)

rfc.fit(x_train, y_train)
```

[23]:
```
RandomForestClassifier()
```

[8]:
```
print("This Is The Train Score: ",rfc.score(x_train,y_train))
print("This Is The Test Score: ",rfc.score(x_test,y_test))
```

```
This Is The Train Score:  0.9953917050691244
This Is The Test Score:  0.05187713310580205
```

[9]:
```
print(x_train)
```

```
      bike_name  city  kms_driven  owner  age  power  brand
4506        421     4        2238      1   16      1      1
2758        133    11        1328      1    4      8      2
640         147   103         341      1    4      1      6
5824         74     3        2964      1    5      5      4
6703        336    19         186      1    6      8      2
…           …    …           …     …    …     …      …
5191         13    36         600      1   10     10      7
5226        133     8        1843      1    6      8      2
5390        176     6        2724      1    6      5      7
860          63     4         438      1    3      5      4
7270         70     2         186      1   11      9      5

[5859 rows x 7 columns]
```

[10]:
```
print(y_train)
```

```
4506     118
2758     102
640      126
5824      28
6703     115
        …
```

```
5191    1007
5226     328
5390     243
860       28
7270       5
Name: price, Length: 5859, dtype: int64
```

[11]:
```
print(x_test)
```

```
      bike_name  city  kms_driven  owner  age  power  brand
3795         34     4         210      1    4      2      2
1562         21     6         740      1    6      8      2
4480         48    40          24      1    4     13      7
3718        171     2          67      1   13      5      4
5887         82     6        2995      1   10      7      6
…           …    …          …     …    …     …      …
4417         87     3        2197      1    6     25     15
527         203     2         298      1    5      9     14
3309        131     2        1638      1    2      5      8
5077         75     2        2545      1    5      8      2
565          42     2         313      1    4      6      1

[1465 rows x 7 columns]
```

[12]:
```
print(y_test)
```

```
3795    840
1562    528
4480     84
3718     57
5887     22
       …
4417    265
527     102
3309    684
5077    328
565      99
Name: price, Length: 1465, dtype: int64
```

[17]:
```
pred = rfc.predict(x_test)
```

[14]:
```
pred
```

[14]:
```
array([ 65, 575,  39, …, 278,  12,   1], dtype=int64)
```

[15]:
```
data = {"prediction":pred,"actual":y_test}
pd.DataFrame(data)
```

[15]:
```
      prediction  actual
3795          65     840
1562         575     528
4480          39      84
3718          43      57
5887         117      22
…            …      …
4417        1177     265
527          530     102
3309         278     684
5077          12     328
565            1      99

[1465 rows x 2 columns]
```

Random Forest Regressor on Bike Dataset

[22]:
```
rfr = RandomForestRegressor()
rfr.fit(x_train, y_train)
```

[22]:
```
RandomForestRegressor()
```

[24]:
```
print("This Is The Train Score: ",rfr.score(x_train,y_train))
print("This Is The Test Score: ",rfr.score(x_test,y_test))
```

```
This Is The Train Score:  0.8512725776719138
This Is The Test Score:  -0.05492554766296709
```

[25]:
```
pred = rfr.predict(x_test)
```

[19]:
```
pred
```

[19]:
```
array([ 65, 575,  39, …, 278,  12,   1], dtype=int64)
```

[18]:
```
data = {"prediction":pred,"actual":y_test}
pd.DataFrame(data)
```

[18]:
```
      prediction  actual
3795          65     840
1562         575     528
4480          39      84
3718          43      57
5887         117      22
…            …      …
4417        1177     265
527          530     102
3309         278     684
5077          12     328
565            1      99
```

```
[1465 rows x 2 columns]
```

[ ]: