import pandas as pd   
import numpy as np   
import matplotlib.pyplot as plt   
import seaborn as sns

df = pd.read\_csv('Used\_Bikes.csv')  
df.head()

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha

# target variable = Y dependent variable  
# features x variables , independent var.

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 32648 entries, 0 to 32647  
Data columns (total 8 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 bike\_name 32648 non-null object   
 1 price 32648 non-null float64  
 2 city 32648 non-null object   
 3 kms\_driven 32648 non-null float64  
 4 owner 32648 non-null object   
 5 age 32648 non-null float64  
 6 power 32648 non-null float64  
 7 brand 32648 non-null object   
dtypes: float64(4), object(4)  
memory usage: 2.0+ MB

df.isnull().sum()

bike\_name 0  
price 0  
city 0  
kms\_driven 0  
owner 0  
age 0  
power 0  
brand 0  
dtype: int64

df.duplicated().sum()

25324

df.shape[0] - df.duplicated().sum()

7324

df.drop\_duplicates(inplace=True)

df.shape

(7324, 8)

cat\_col = df.select\_dtypes(include='O')  
cat\_col.head()

bike\_name city owner brand  
0 TVS Star City Plus Dual Tone 110cc Ahmedabad First Owner TVS  
1 Royal Enfield Classic 350cc Delhi First Owner Royal Enfield  
2 Triumph Daytona 675R Delhi First Owner Triumph  
3 TVS Apache RTR 180cc Bangalore First Owner TVS  
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition Bangalore First Owner Yamaha

num\_col = df.select\_dtypes(exclude='O')  
num\_col.head()

price kms\_driven age power  
0 35000.0 17654.0 3.0 110.0  
1 119900.0 11000.0 4.0 350.0  
2 600000.0 110.0 8.0 675.0  
3 65000.0 16329.0 4.0 180.0  
4 80000.0 10000.0 3.0 150.0

#feature selection   
df['bike\_name'].nunique()

471

cat\_col = cat\_col.drop(['bike\_name','city'],axis='columns')  
cat\_col.head()

owner brand  
0 First Owner TVS  
1 First Owner Royal Enfield  
2 First Owner Triumph  
3 First Owner TVS  
4 First Owner Yamaha

# label encoding , one hote encoding  
df['owner'].value\_counts()

owner  
First Owner 6642  
Second Owner 588  
Third Owner 84  
Fourth Owner Or More 10  
Name: count, dtype: int64

#one hot encoding  
pd.get\_dummies(cat\_col).astype('int')

owner\_First Owner owner\_Fourth Owner Or More owner\_Second Owner \  
0 1 0 0   
1 1 0 0   
2 1 0 0   
3 1 0 0   
4 1 0 0   
... ... ... ...   
9362 1 0 0   
9369 1 0 0   
9370 1 0 0   
9371 1 0 0   
9372 1 0 0   
  
 owner\_Third Owner brand\_BMW brand\_Bajaj brand\_Benelli brand\_Ducati \  
0 0 0 0 0 0   
1 0 0 0 0 0   
2 0 0 0 0 0   
3 0 0 0 0 0   
4 0 0 0 0 0   
... ... ... ... ... ...   
9362 0 0 0 0 0   
9369 0 0 1 0 0   
9370 0 0 0 0 0   
9371 0 0 1 0 0   
9372 0 0 1 0 0   
  
 brand\_Harley-Davidson brand\_Hero ... brand\_LML brand\_MV \  
0 0 0 ... 0 0   
1 0 0 ... 0 0   
2 0 0 ... 0 0   
3 0 0 ... 0 0   
4 0 0 ... 0 0   
... ... ... ... ... ...   
9362 0 1 ... 0 0   
9369 0 0 ... 0 0   
9370 1 0 ... 0 0   
9371 0 0 ... 0 0   
9372 0 0 ... 0 0   
  
 brand\_Mahindra brand\_Rajdoot brand\_Royal Enfield brand\_Suzuki \  
0 0 0 0 0   
1 0 0 1 0   
2 0 0 0 0   
3 0 0 0 0   
4 0 0 0 0   
... ... ... ... ...   
9362 0 0 0 0   
9369 0 0 0 0   
9370 0 0 0 0   
9371 0 0 0 0   
9372 0 0 0 0   
  
 brand\_TVS brand\_Triumph brand\_Yamaha brand\_Yezdi   
0 1 0 0 0   
1 0 0 0 0   
2 0 1 0 0   
3 1 0 0 0   
4 0 0 1 0   
... ... ... ... ...   
9362 0 0 0 0   
9369 0 0 0 0   
9370 0 0 0 0   
9371 0 0 0 0   
9372 0 0 0 0   
  
[7324 rows x 27 columns]

cat\_col['owner'].value\_counts()

owner  
First Owner 6642  
Second Owner 588  
Third Owner 84  
Fourth Owner Or More 10  
Name: count, dtype: int64

dt = {"First Owner":1,"Second Owner":2,"Third Owner":3,"Fourth Owner Or More":4}  
dt

{'First Owner': 1,  
 'Second Owner': 2,  
 'Third Owner': 3,  
 'Fourth Owner Or More': 4}

cat\_col['owner'] = cat\_col['owner'].map(dt)

cat\_col['owner'].value\_counts()

owner  
1 6642  
2 588  
3 84  
4 10  
Name: count, dtype: int64

{i:key for key , i in enumerate(list(cat\_col['brand'].unique()))}

{'TVS': 0,  
 'Royal Enfield': 1,  
 'Triumph': 2,  
 'Yamaha': 3,  
 'Honda': 4,  
 'Hero': 5,  
 'Bajaj': 6,  
 'Suzuki': 7,  
 'Benelli': 8,  
 'KTM': 9,  
 'Mahindra': 10,  
 'Kawasaki': 11,  
 'Ducati': 12,  
 'Hyosung': 13,  
 'Harley-Davidson': 14,  
 'Jawa': 15,  
 'BMW': 16,  
 'Indian': 17,  
 'Rajdoot': 18,  
 'LML': 19,  
 'Yezdi': 20,  
 'MV': 21,  
 'Ideal': 22}

cat\_col['brand'].nunique()

23

dt2 = {'TVS': 0,  
 'Royal Enfield': 1,  
 'Triumph': 2,  
 'Yamaha': 3,  
 'Honda': 4,  
 'Hero': 5,  
 'Bajaj': 6,  
 'Suzuki': 7,  
 'Benelli': 8,  
 'KTM': 9,  
 'Mahindra': 10,  
 'Kawasaki': 11,  
 'Ducati': 12,  
 'Hyosung': 13,  
 'Harley-Davidson': 14,  
 'Jawa': 15,  
 'BMW': 16,  
 'Indian': 17,  
 'Rajdoot': 18,  
 'LML': 19,  
 'Yezdi': 20,  
 'MV': 21,  
 'Ideal': 22}  
dt2

{'TVS': 0,  
 'Royal Enfield': 1,  
 'Triumph': 2,  
 'Yamaha': 3,  
 'Honda': 4,  
 'Hero': 5,  
 'Bajaj': 6,  
 'Suzuki': 7,  
 'Benelli': 8,  
 'KTM': 9,  
 'Mahindra': 10,  
 'Kawasaki': 11,  
 'Ducati': 12,  
 'Hyosung': 13,  
 'Harley-Davidson': 14,  
 'Jawa': 15,  
 'BMW': 16,  
 'Indian': 17,  
 'Rajdoot': 18,  
 'LML': 19,  
 'Yezdi': 20,  
 'MV': 21,  
 'Ideal': 22}

cat\_col['brand'] = cat\_col['brand'].map(dt2)

cat\_col

owner brand  
0 1 0  
1 1 1  
2 1 2  
3 1 0  
4 1 3  
... ... ...  
9362 1 5  
9369 1 6  
9370 1 14  
9371 1 6  
9372 1 6  
  
[7324 rows x 2 columns]

complete\_df = pd.concat([cat\_col,num\_col],axis=1)  
complete\_df.head()

owner brand price kms\_driven age power  
0 1 0 35000.0 17654.0 3.0 110.0  
1 1 1 119900.0 11000.0 4.0 350.0  
2 1 2 600000.0 110.0 8.0 675.0  
3 1 0 65000.0 16329.0 4.0 180.0  
4 1 3 80000.0 10000.0 3.0 150.0

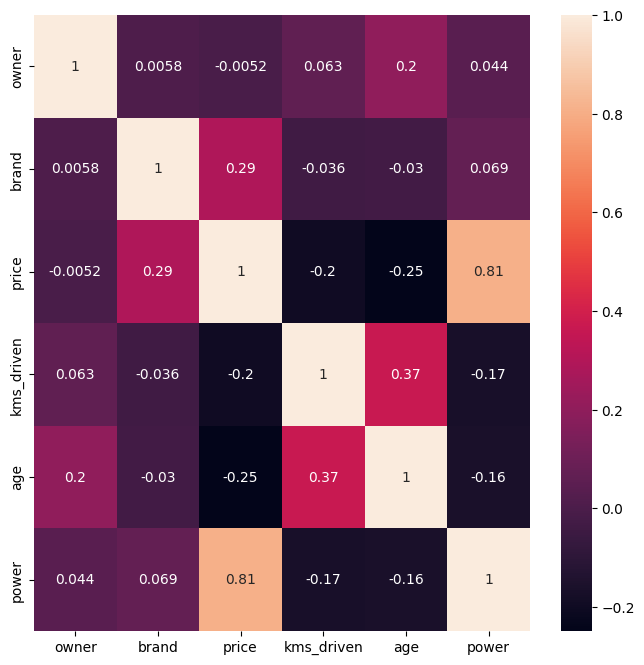
complete\_df.dtypes

owner int64  
brand int64  
price float64  
kms\_driven float64  
age float64  
power float64  
dtype: object

metrix = complete\_df.corr()  
metrix

owner brand price kms\_driven age power  
owner 1.000000 0.005844 -0.005188 0.062612 0.203151 0.043746  
brand 0.005844 1.000000 0.288409 -0.036161 -0.030263 0.068647  
price -0.005188 0.288409 1.000000 -0.199951 -0.248752 0.807641  
kms\_driven 0.062612 -0.036161 -0.199951 1.000000 0.367518 -0.168295  
age 0.203151 -0.030263 -0.248752 0.367518 1.000000 -0.164211  
power 0.043746 0.068647 0.807641 -0.168295 -0.164211 1.000000

plt.figure(figsize=(8,8))  
sns.heatmap(metrix,annot=True)  
plt.show()



price = metrix['price']

list(price[price<0.6].keys())

['owner', 'brand', 'kms\_driven', 'age']

x = complete\_df.drop('price',axis=1)  
y = complete\_df[['price']]

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=42)

x\_train

owner brand kms\_driven age power  
5789 1 0 68857.0 15.0 110.0  
3451 1 1 5740.0 6.0 500.0  
735 1 5 28329.0 6.0 110.0  
7533 1 3 42966.0 7.0 150.0  
8461 1 1 8000.0 5.0 500.0  
... ... ... ... ... ...  
6522 1 6 56000.0 9.0 200.0  
6583 1 1 22493.0 5.0 500.0  
6856 1 6 17477.0 5.0 150.0  
1028 1 3 14836.0 8.0 150.0  
9250 1 4 8000.0 10.0 250.0  
  
[5859 rows x 5 columns]

from sklearn.linear\_model import LinearRegression

lr = LinearRegression()

lr.fit(x\_train,y\_train)

LinearRegression()

lr.score(x\_train,y\_train)

0.7053826605671762

lr.score(x\_test,y\_test)

0.7586900869386048

from sklearn.ensemble import RandomForestRegressor  
rd = RandomForestRegressor()  
rd.fit(x\_train,y\_train)  
print(rd.score(x\_train,y\_train))  
print(rd.score(x\_test,y\_test))

c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:1474: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel().  
 return fit\_method(estimator, \*args, \*\*kwargs)

0.9848207452127526  
0.900527533182769

from sklearn.metrics import mean\_squared\_error,mean\_absolute\_error

pred = lr.predict(x\_test)

y\_test['prediction'] = pred

y\_test

price prediction  
4909 88400.0 125382.545235  
1942 102850.0 216422.203336  
5763 67000.0 96966.485952  
4800 30000.0 7571.713344  
7614 20000.0 8532.577046  
... ... ...  
5653 395000.0 476134.279312  
609 140000.0 175120.608085  
4211 54500.0 72233.836151  
6379 114000.0 208856.676207  
647 36000.0 9305.600876  
  
[1465 rows x 2 columns]

mean\_squared\_error(y\_test['price'],y\_test['prediction'])

4697994405.450356

# score   
# adjusted square  
# correlation  
# mean squared

import joblib, pickle

joblib.dump(lr,'linear-model.lb')

['linear-model.lb']

model = joblib.load('linear-model.lb')

from sklearn.tree import DecisionTreeClassifier  
from sklearn.ensemble import RandomForestRegressor

dt = DecisionTreeClassifier(criterion="entropy",splitter='random')

joblib.dump(rd,'RD\_model.lb')

['RD\_model.lb']

x\_test

owner brand kms\_driven age power  
4909 1 1 22500.0 6.0 350.0  
1942 1 1 3198.0 5.0 500.0  
5763 1 6 15000.0 6.0 220.0  
4800 1 3 27000.0 12.0 150.0  
7614 1 5 16764.0 9.0 100.0  
... ... ... ... ... ...  
5653 1 14 16523.0 5.0 750.0  
609 1 13 2881.0 7.0 250.0  
4211 1 7 23833.0 4.0 150.0  
6379 1 1 9282.0 7.0 500.0  
647 1 0 2980.0 6.0 160.0  
  
[1465 rows x 5 columns]

rd.predict([[1, 1, 22500.0, 6.0, 350.0]]) #for single data point,single row we can do this process

c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names  
 warnings.warn(

array([109110.])

model = joblib.load('RD\_model.lb')

bike\_numbers= {'TVS': 0,  
 'Royal Enfield': 1,  
 'Triumph': 2,  
 'Yamaha': 3,  
 'Honda': 4,  
 'Hero': 5,  
 'Bajaj': 6,  
 'Suzuki': 7,  
 'Benelli': 8,  
 'KTM': 9,  
 'Mahindra': 10,  
 'Kawasaki': 11,  
 'Ducati': 12,  
 'Hyosung': 13,  
 'Harley-Davidson': 14,  
 'Jawa': 15,  
 'BMW': 16,  
 'Indian': 17,  
 'Rajdoot': 18,  
 'LML': 19,  
 'Yezdi': 20,  
 'MV': 21,  
 'Ideal': 22}