Day-10

Vanda Barath Statement

In [1]:

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

In [2]:

```
d=pd.read_csv(r"C:\Users\user\Downloads\Bharat.csv")
d
```

Out[2]:

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City
0	1	New Delhi - Varanasi Vande Bharat Express	22435/22436	Delhi	New Delhi	Varanasi
1	2	New Delhi - Shri Mata Vaishno Devi Katra Vande...	22439/22440	Delhi	New Delhi	Katra
2	3	Mumbai Central - Gandhinagar Capital Vande Bha...	20901/20902	Mumbai	Mumbai Central	Gandhinagar
3	4	New Delhi - Amb Andaura Vande Bharat Express	22447/22448	Delhi	New Delhi	Andaura
4	5	MGR Chennai Central - Mysuru Vande Bharat Express	20607/20608	Chennai	Chennai Central	Mysuru
5	6	Bilaspur - Nagpur Vande Bharat Express	20825/20826	Bilaspur, Chhattisgarh	Bilaspur Junction	Nagpur
6	7	Howrah - New Jalpaiguri Vande Bharat Express	22301/22302	Kolkata	Howrah Junction	Siliguri
7	8	Visakhapatnam - Secunderabad Vande Bharat Express	20833/20834	Visakhapatnam	Visakhapatnam Junction	Hyderabad
8	9	Mumbai CSMT - Solapur Vande Bharat Express	22225/22226	Mumbai	Chhatrapati Shivaji Terminus	Solapur
9	10	Mumbai CSMT - Sainagar Shirdi Vande Bharat Exp...	22223/22224	Mumbai	Chhatrapati Shivaji Terminus	Shirdi
10	11	Rani Kamalapati (Habibganj) - Hazrat Nizamuddi...	20171/20172	Bhopal	Habibganj (Rani Kamalapati)	Delhi
11	12	Secunderabad - Tirupati Vande Bharat Express	20701/20702	Hyderabad	Secunderabad Junction	Tirupati
12	13	MGR Chennai Central - Coimbatore Vande Bharat ...	20643/20644	Chennai	Chennai Central	Coimbatore
13	14	Delhi Cantonment - Ajmer Vande Bharat Express	20977/20978	Delhi	Delhi Cantonment	Ajmer
14	15	Kasaragod - Thiruvananthapuram Vande Bharat Ex...	20633/20634	Kasaragod	Kasaragod	Thiruvananthapuram
15	16	Howrah - Puri Vande Bharat Express	22895/22896	Kolkata	Howrah Junction	Puri

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City
16	17	Anand Vihar Terminal - Dehradun Vande Bharat E...	22457/22458	Delhi	Anand Vihar Terminal	Dehradun
17	18	New Jalpaiguri - Guwahati Vande Bharat Express	22227/22228	Siliguri	New Jalpaiguri Junction	Guwahati
18	19	Mumbai CSMT - Madgaon Vande Bharat Express	22229/22230	Mumbai	Chhatrapati Shivaji Terminus	Madgaon
19	19	Mumbai CSMT - Madgaon Vande Bharat Express	22229/22230	Mumbai	Chhatrapati Shivaji Terminus	Madgaon
20	20	Patna - Ranchi Vande Bharat Express	22349/22350	Patna	Patna Junction	Ranchi
21	21	KSR Bengaluru - Dharwad Vande Bharat Express	20661/20662	Bangalore	Bangalore City	Hubbali - Dharwad
22	22	Rani Kamalapati (Habibganj) - Jabalpur Vande B...	20173/20174	Bhopal	Habibganj (Rani Kamalapati)	Jabalpur
23	23	Indore - Bhopal Vande Bharat Express	20911/20912	Indore	Indore Junction	Bhopal
24	24	Jodhpur - Sabarmati (Ahmedabad) Vande Bharat E...	12461/12462	Jodhpur	Jodhpur Junction	Ahmedabad
25	25	Gorakhpur - Lucknow Charbagh Vande Bharat Express	22549/22550	Gorakhpur	Gorakhpur Junction	Charbagh

In [3]:
```
d.columns
```

Out[3]:

```
Index(['Sr. No.', 'Train Name', 'Train Number', 'Originating City',
       'Originating Station', 'Terminal City', 'Terminal Station', 'Operat
or',
       'No. of Cars', 'Frequency', 'Distance', 'Travel Time', 'Speed',
       'Average Speed', 'Inauguration', 'Average occupancy'],
      dtype='object')
```

In [4]:

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Sr. No.              26 non-null     int64 
 1   Train Name           26 non-null     object
 2   Train Number         26 non-null     object
 3   Originating City     26 non-null     object
 4   Originating Station  26 non-null     object
 5   Terminal City        26 non-null     object
 6   Terminal Station     26 non-null     object
 7   Operator             26 non-null     object
 8   No. of Cars          26 non-null     int64 
 9   Frequency            26 non-null     object
 10  Distance             26 non-null     object
 11  Travel Time          26 non-null     object
 12  Speed                26 non-null     object
 13  Average Speed        26 non-null     object
 14  Inauguration         26 non-null     object
 15  Average occupancy    26 non-null     object
dtypes: int64(2), object(14)
memory usage: 3.4+ KB
```

In [6]:

```
x=d[['Sr. No.']]
y=d['No. of Cars']
```

In [7]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [8]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[8]:

```
LinearRegression()
```

In [9]:

```
print(lr.intercept_)
```

```
16.696071342492168
```

In [10]:

```
print(lr.score(x_test,y_test))
```

```
0.5456909482273775
```

In [11]:

```
print(lr.score(x_train,y_train))
```

```
0.2989425162689805
```

Ridge Regression

In [12]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [13]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[13]:

```
0.5420367235948864
```

Lasso Regression

In [14]:

```
la=Lasso(alpha=10)
```

In [15]:

```
la.fit(x_train,y_train)
```

Out[15]:

```
Lasso(alpha=10)
```

In [16]:

```
la.score(x_test,y_test)
```

Out[16]:

```
0.15966243026131
```

Elastic Regreesion

In [17]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[17]:

```
ElasticNet()
```

In [18]:

```
predict=(en.predict(x_test))
print(predict)
```

```
[11.86227473 13.31769901 15.06420814 14.48203843 16.22854756 13.60878386
 10.69793531 10.1157656 ]
```

In [19]:

```
print(en.score(x_test,y_test))
```

```
0.5310323920677906
```

Evaluation Method

In [20]:

```
from sklearn import metrics
```

In [21]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute Error: 2.0884055376536574
```

In [22]:

```
print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root Mean Square Error: 2.372258690612496
```

In [23]:

```
print("Mean Square Error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Square Error: 5.627611295186513
```

In [25]:

```
import pickle
```

In [26]:

```
filename="predict"
```

In [27]:

```
pickle.dump(lr,open(filename,'wb'))
```

In [ ]:

```

```