DAY 10:

Horse Dataset

In [1]:

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

In [2]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\15_horse.csv")[0:500]
df
```

Out[2]:

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	Co
0	03.09.2017	Sha Tin	10	1400	Gress	1310000	6	K C Leung	52	S
1	16.09.2017	Sha Tin	10	1400	Gress	1310000	14	C Y Ho	52	S
2	14.10.2017	Sha Tin	10	1400	Gress	1310000	8	C Y Ho	52	S
3	11.11.2017	Sha Tin	9	1600	Gress	1310000	13	Brett Prebble	54	S
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	S
...	...	...	...	...	...	...	...	...	...	
495	19.11.2017	Sha Tin	3	1400	Gress	880000	3	W M Lai	53	Ze
496	01.01.2018	Sha Tin	2	1600	Gress	660000	1	K Teetan	60	Ze
497	17.01.2018	Happy Valley	2	1650	Gress	660000	9	K Teetan	60	Ze
498	16.09.2017	Sha Tin	9	1000	Gress	1310000	6	M L Yeung	53	Ze
499	01.10.2017	Sha Tin	6	1200	Gress	1310000	10	M Chadwick	53	Ze

500 rows × 21 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Dato               500 non-null    object
 1   Track              500 non-null    object
 2   Race Number        500 non-null    int64 
 3   Distance           500 non-null    int64 
 4   Surface            500 non-null    object
 5   Prize money        500 non-null    int64 
 6   Starting position  500 non-null    int64 
 7   Jockey             500 non-null    object
 8   Jockey weight      500 non-null    int64 
 9   Country            500 non-null    object
 10  Horse age          500 non-null    int64 
 11  TrainerName        500 non-null    object
 12  Race time          500 non-null    object
 13  Path               500 non-null    int64 
 14  Final place        500 non-null    int64 
 15  FGrating           500 non-null    int64 
 16  Odds               500 non-null    object
 17  RaceType           500 non-null    object
 18  HorseId            500 non-null    int64 
 19  JockeyId           500 non-null    int64 
 20  TrainerID          500 non-null    int64 
dtypes: int64(12), object(9)
memory usage: 82.2+ KB
```

In [4]:

```
df.columns
```

Out[4]:

```
Index(['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize mone
y',
       'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse a
ge',
       'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odd
s',
       'RaceType', 'HorseId', 'JockeyId', 'TrainerID'],
      dtype='object')
```

Linear Regression

In [6]:

```
x=df[['Race Number', 'Distance','Prize money',
       'Starting position']]
y=df[ 'TrainerID']
```

In [7]:

```
# to split my dataset into test and train data
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.score(x_test,y_test))
```

```
-0.002183230659517532
```

In [10]:

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

Out[10]:

```
0.04077484063374881
```

Ridge Regression

In [11]:

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

In [12]:

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

Out[12]:

```
-0.0021622517972550437
```

Lasso Regression

In [13]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[13]:

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

In [14]:

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

Out[14]:

```
0.002487067263932996
```

Elastic regression

In [15]:

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

Out[15]:

```
ElasticNet()
```

In [16]:

```
print(en.intercept_)
```

```
6648.405791290872
```

In [17]:

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

In [18]:

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

```
-0.0014372940221814012
```

Evalution matrics

In [19]:

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

```
Mean Absolute Error: 43.598893750357234
```

In [20]:

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

```
Mean Square Error: 8188.4721727437545
```

In [21]:

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

```
Root Mean Square Error: 90.49017721688777
```

In [ ]:

```

```

In [ ]:

```

```