DATA COLLECTION

In [1]:
```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\15_Horse.csv")
sd
```

Out[2]:

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	Coun
0	03.09.2017	Sha Tin	10	1400	Gress	1310000	6	K C Leung	52	Sver
1	16.09.2017	Sha Tin	10	1400	Gress	1310000	14	C Y Ho	52	Sver
2	14.10.2017	Sha Tin	10	1400	Gress	1310000	8	C Y Ho	52	Sver
3	11.11.2017	Sha Tin	9	1600	Gress	1310000	13	Brett Prebble	54	Sver
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	Sver
...	...	...	...	...	...	...	...	...	...	
27003	14.06.2020	Sha Tin	11	1200	Gress	1450000	6	A Hamelin	59	Austra
27004	21.06.2020	Sha Tin	2	1200	Gress	967000	7	K C Leung	57	Austra
27005	21.06.2020	Sha Tin	4	1200	Gress	967000	6	Blake Shinn	57	Austra
27006	21.06.2020	Sha Tin	5	1200	Gress	967000	14	Joao Moreira	57	N Zeala
27007	21.06.2020	Sha Tin	11	1200	Gress	1450000	7	C Schofield	55	N Zeala

27008 rows × 21 columns

In [3]:
```
# to display top 10 rows
sd.head(10)
```

Out[3]:

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	Country
0	03.09.2017	Sha Tin	10	1400	Gress	1310000	6	K C Leung	52	Sverige
1	16.09.2017	Sha Tin	10	1400	Gress	1310000	14	C Y Ho	52	Sverige
2	14.10.2017	Sha Tin	10	1400	Gress	1310000	8	C Y Ho	52	Sverige
3	11.11.2017	Sha Tin	9	1600	Gress	1310000	13	Brett Prebble	54	Sverige
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	Sverige
5	10.12.2017	Sha Tin	1	1800	Gress	1310000	4	C Y Ho	52	Sverige
6	01.01.2018	Sha Tin	9	1800	Gress	1310000	9	C Schofield	54	Sverige
7	04.02.2018	Sha Tin	5	1800	Gress	1310000	6	Joao Moreira	57	Sverige
8	03.03.2018	Sha Tin	8	1800	Gress	1310000	3	C Y Ho	56	Sverige
9	11.03.2018	Sha Tin	10	1600	Gress	1310000	8	C Y Ho	57	Sverige

10 rows × 21 columns

DATA CLEANING AND PRE_PROCESSING

In [4]:
```
sd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27008 entries, 0 to 27007
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Dato               27008 non-null  object
 1   Track              27008 non-null  object
 2   Race Number        27008 non-null  int64 
 3   Distance           27008 non-null  int64 
 4   Surface            27008 non-null  object
 5   Prize money        27008 non-null  int64 
 6   Starting position  27008 non-null  int64 
 7   Jockey             27008 non-null  object
 8   Jockey weight      27008 non-null  int64 
 9   Country            27008 non-null  object
 10  Horse age          27008 non-null  int64 
 11  TrainerName        27008 non-null  object
 12  Race time          27008 non-null  object
 13  Path               27008 non-null  int64 
 14  Final place        27008 non-null  int64 
 15  FGrating           27008 non-null  int64 
 16  Odds               27008 non-null  object
 17  RaceType           27008 non-null  object
 18  HorseId            27008 non-null  int64 
 19  JockeyId           27008 non-null  int64 
 20  TrainerID          27008 non-null  int64 
dtypes: int64(12), object(9)
memory usage: 4.3+ MB
```

In [5]:
```
# to display summary of statistics
sd.describe()
```

Out[5]:

	Race Number	Distance	Prize money	Starting position	Jockey weight	Horse age	
count	27008.000000	27008.000000	2.700800e+04	27008.000000	27008.000000	27008.000000	270
mean	5.268624	1401.666173	1.479445e+06	6.741447	55.867373	5.246408	
std	2.780088	276.065045	2.162109e+06	3.691071	2.737006	1.519880	
min	1.000000	1000.000000	6.600000e+05	1.000000	47.000000	2.000000	
25%	3.000000	1200.000000	9.200000e+05	4.000000	54.000000	4.000000	
50%	5.000000	1400.000000	9.670000e+05	7.000000	56.000000	5.000000	
75%	8.000000	1650.000000	1.450000e+06	10.000000	58.000000	6.000000	
max	11.000000	2400.000000	2.800000e+07	14.000000	63.000000	12.000000	

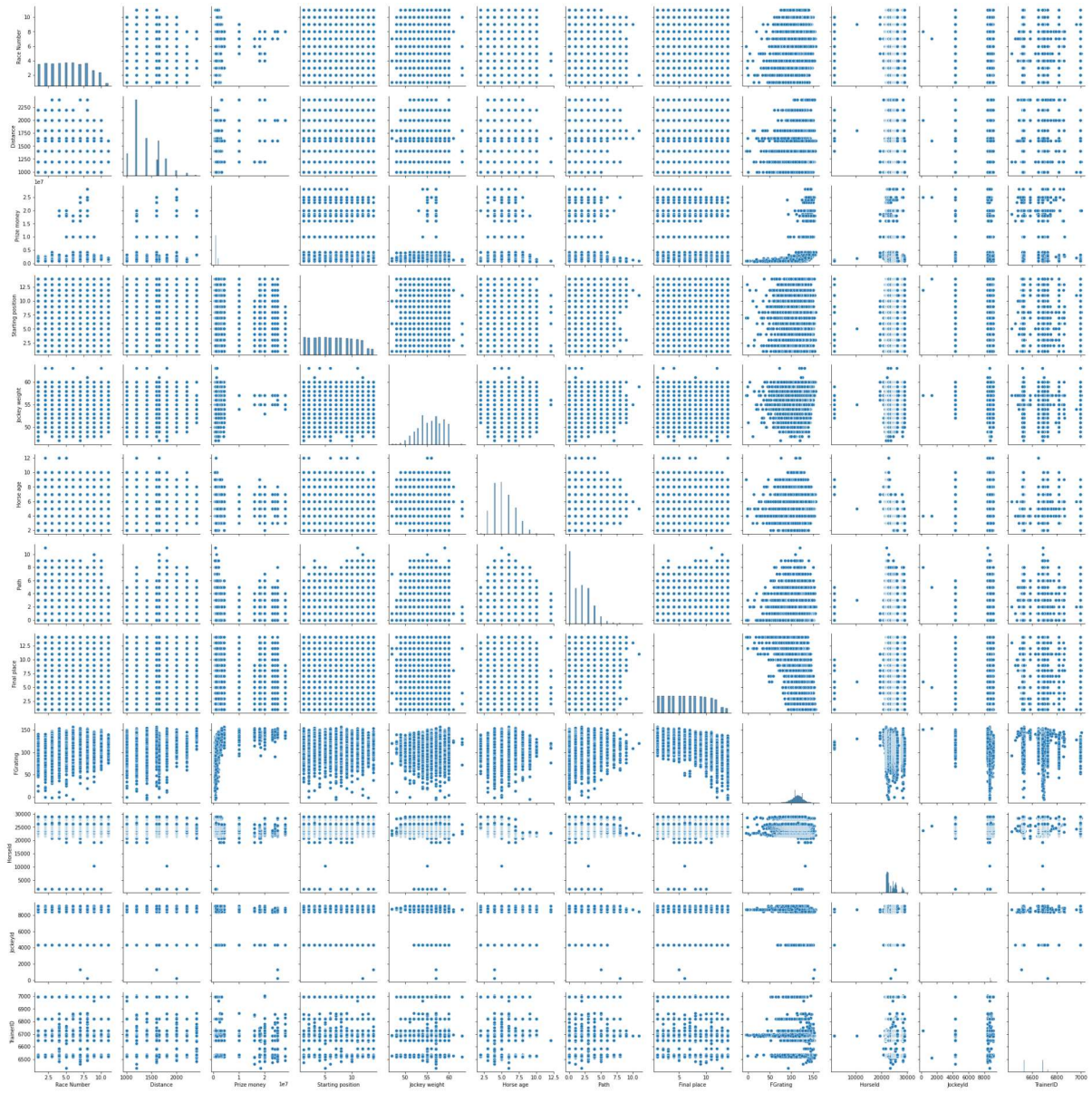
In [6]:
```
#to display colums heading
sd.columns
```

Out[6]:
```
Index(['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize money',
       'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse ag
e',
       'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odds',
       'RaceType', 'HorseId', 'JockeyId', 'TrainerID'],
      dtype='object')
```

EDA and visualization

In [7]:
```
sns.pairplot(sd)
```

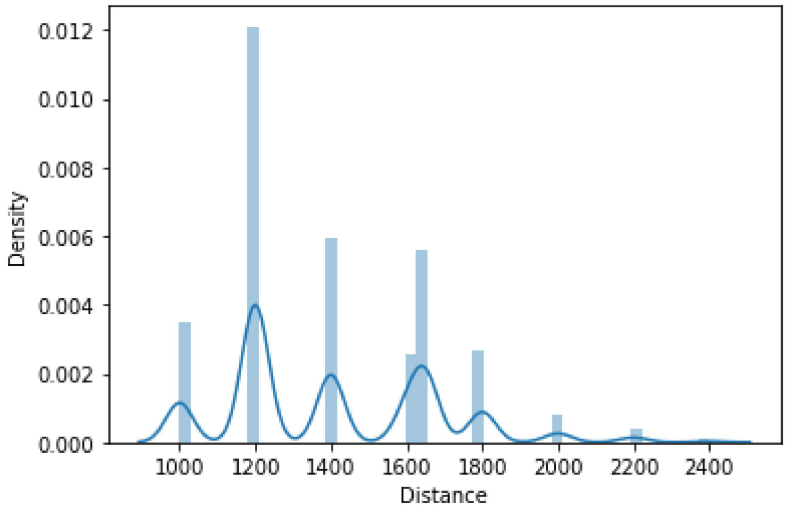
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x13da303ca00>
```



In [8]:
```
sns.distplot(sd['Distance'])
```

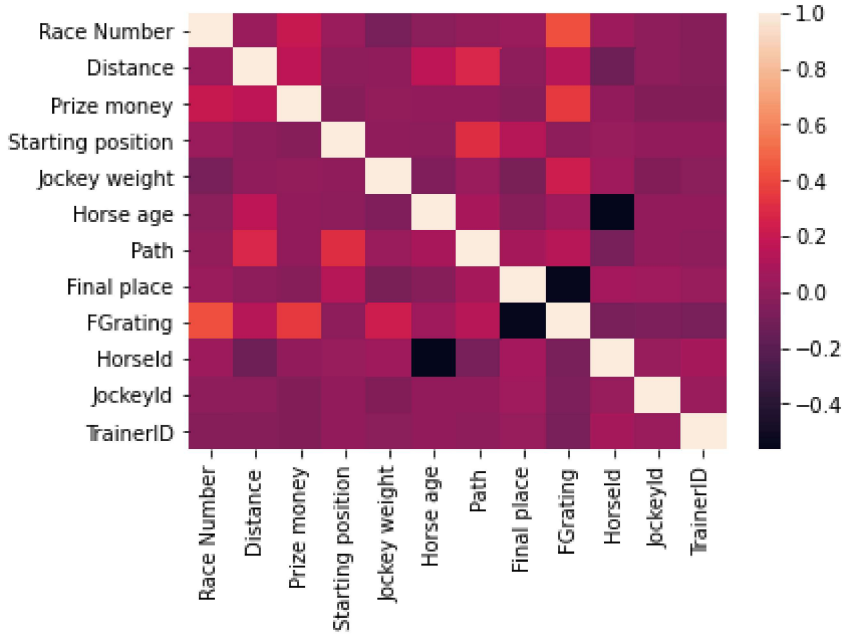
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:
```
<AxesSubplot:xlabel='Distance', ylabel='Density'>
```



In [9]:
```
sns.heatmap(sd.corr())
```

Out[9]:
```
<AxesSubplot:>
```



In [10]:
```
sd1=sd[['Race Number', 'Distance', 'Prize money',
       'Starting position', 'Jockey weight']]
```

TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

In [11]:
```
x= sd1[['Race Number', 'Distance', 'Prize money',
       'Starting position']]
y=sd1[ 'Jockey weight']
```

In [12]:
```
# To split my dataset  into training data and test 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 [13]:
```
from sklearn.linear_model import LinearRegression

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

Out[13]:
```
LinearRegression()
```

In [14]:
```
from sklearn.linear_model import LinearRegression

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

Out[14]:
```
LinearRegression()
```

In [15]:
```
print(lr.intercept_)
```

```
56.29139413877686
```

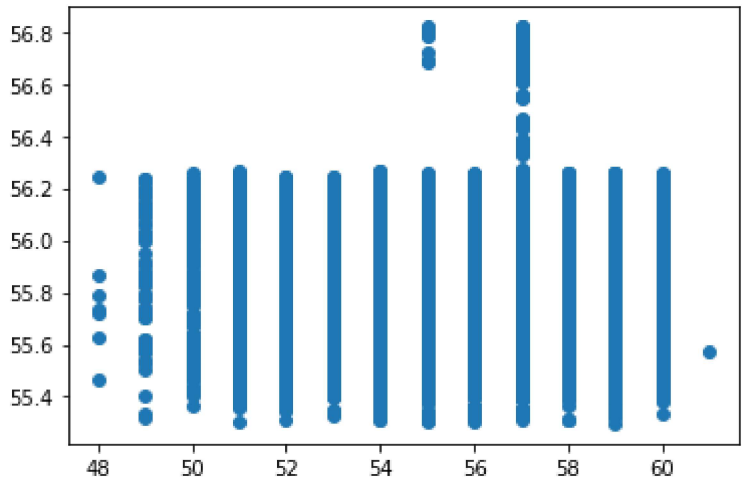
In [16]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Race Number	-9.066383e-02
Distance	-3.913607e-05
Prize money	4.643470e-08
Starting position	4.300373e-03

In [17]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]: <matplotlib.collections.PathCollection at 0x13dbb83b880>



In [18]:
```
print(lr.score(x_test,y_test))
```

```
0.012470199596210652
```

In [19]:
```
lr.score(x_train,y_train)
```

Out[19]: 0.008440924886162104

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[21]: Ridge(alpha=10)

In [22]:
```
dr.score(x_test,y_test)
```

Out[22]: 0.012469878005481494

In [23]:
```
dr.score(x_train,y_train)
```

Out[23]: 0.0084409248447328

In [24]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]: Lasso(alpha=10)

In [25]:
```
la.score(x_test,y_test)
```

Out[25]: -0.00017477995385428713

In [26]:
```
la.score(x_train,y_train)
```

Out[26]:
```
0.00030885069270014665
```

ElasticNet

In [27]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:
```
ElasticNet()
```

In [28]:
```
print(en.coef_)
```

```
[-2.16840242e-02 -3.31644059e-05  2.85658774e-08  0.00000000e+00]
```

In [29]:
```
print(en.intercept_)
```

```
55.97421586564249
```

In [30]:
```
prediction=en.predict(x_test)
```

In [31]:
```
print(en.score(x_test,y_test))
```

```
0.004306640009266727
```

Evaluation metric

In [32]:
```
from sklearn import metrics
```

In [33]:
```
print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
mean Absolute Error: 2.2856105542816803
```

In [34]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean squared Error: 7.399911888621687
```

In [35]:
```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root mean Absolytre Error: 2.7202779065054523
```

In [ ]: