In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df = pd.read_csv("Horse.csv")
# .dropna(axis="columns")
df
```

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]:

```
df.head()
```

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 rows × 21 columns

Data cleaning and pre processing

In [4]:

```
df.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]:
```
df.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	

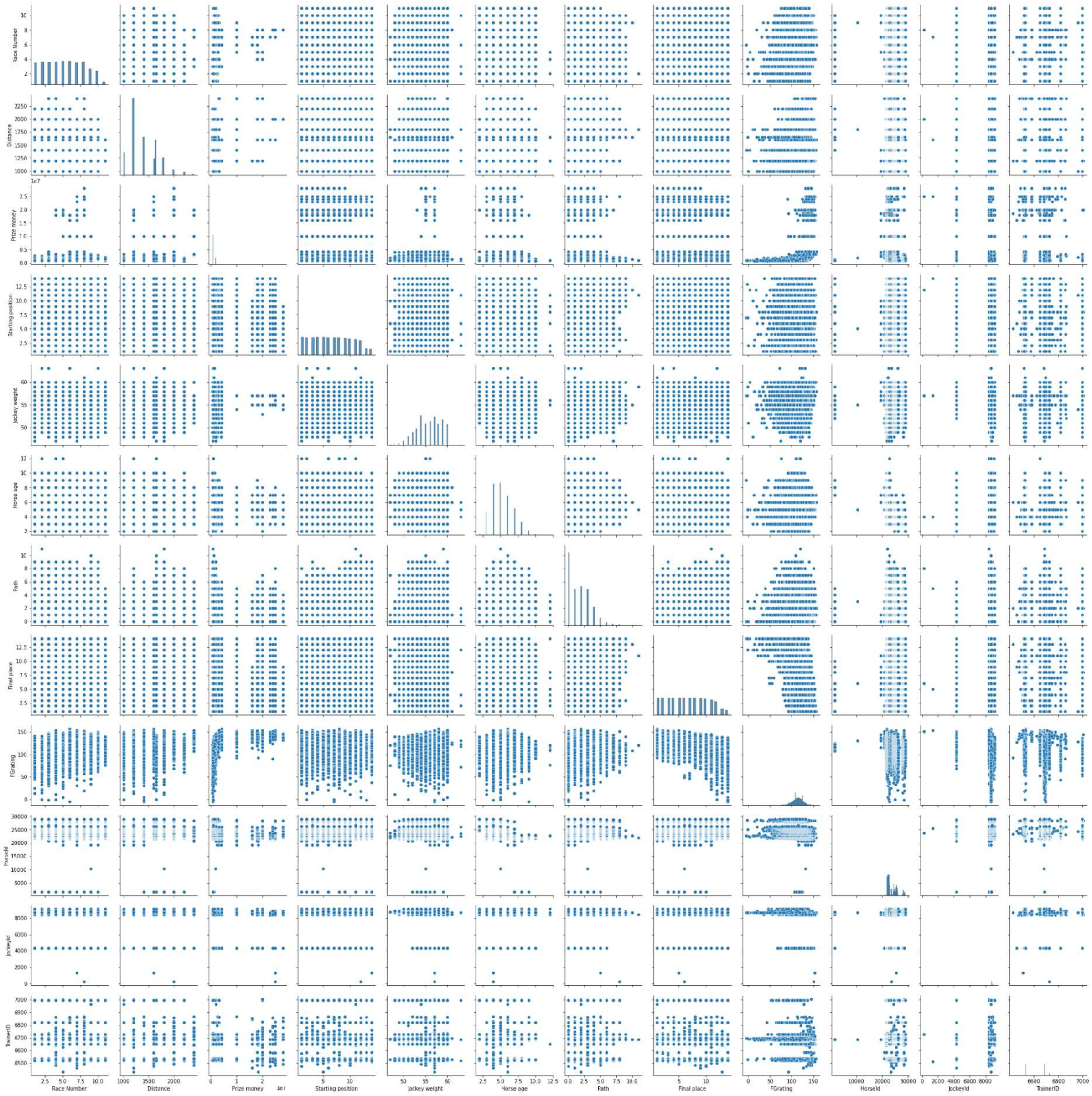
In [6]:
```
df.columns
```

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(df)
```

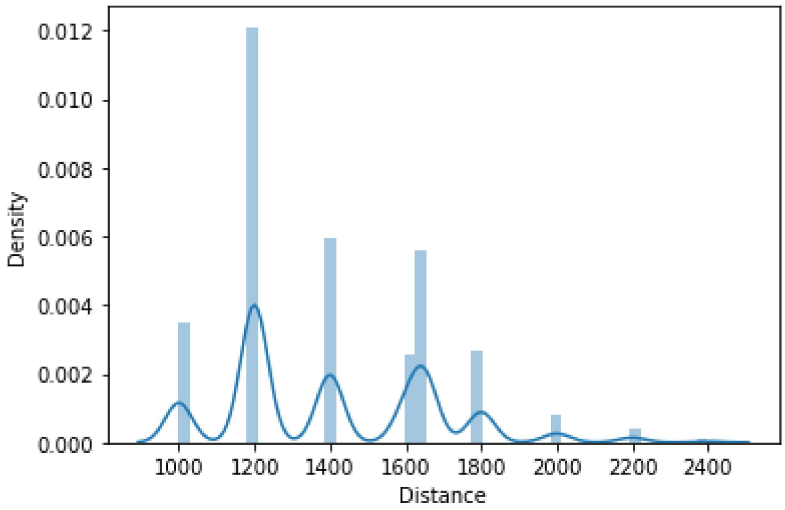
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x16381cb8b50>
```



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

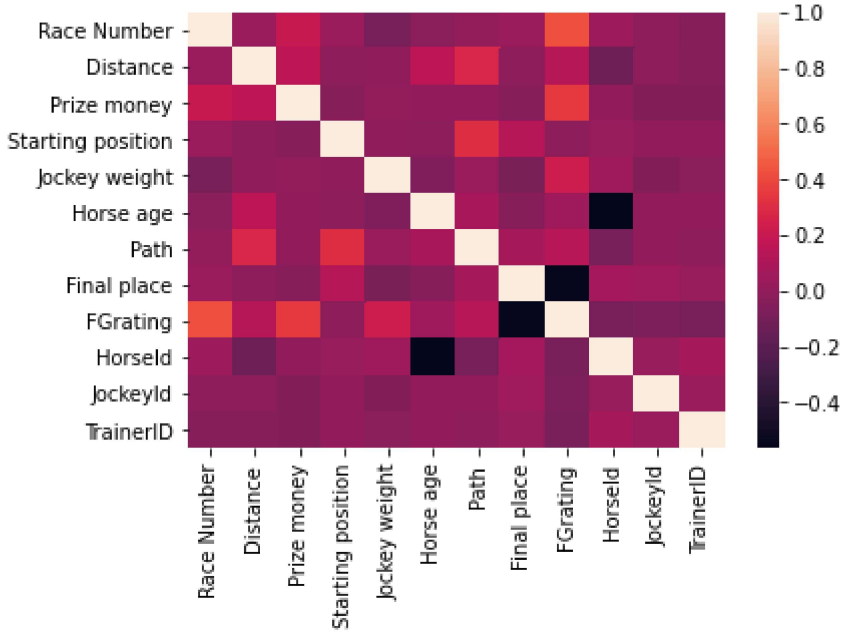
```
Out[8]: <AxesSubplot:xlabel='Distance', ylabel='Density'>
```



```
In [9]: df1 = df[['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize money',
               'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse age',
               'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odds',
               'RaceType', 'HorseId', 'JockeyId', 'TrainerID']]
```

```
In [10]: sns.heatmap(df1.corr())
```

```
Out[10]: <AxesSubplot:>
```



In [11]:
```
x = df1[['Race Number','Distance','Prize money','Starting position','Jockey wei
y = df1[ 'FGrating']
```

split the data into training and test data

In [12]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

In [13]:
```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

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

In [14]:
```
lr.intercept_
```

Out[14]:
```
101.38500963011558
```

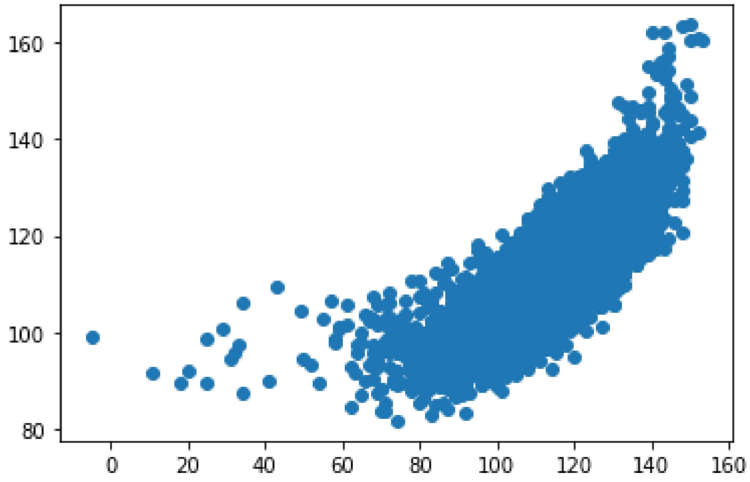
In [15]:
```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
Race Number	1.997801
Distance	0.000797
Prize money	0.000001
Starting position	0.014436
Jockey weight	1.048928
Horse age	0.134051
Path	1.247377
Final place	-1.985124
HorseId	-0.000288
JockeyId	-0.000060
TrainerID	-0.006380

```
In [16]: prediction = lr.predict(x_test)
         plt.scatter(y_test, prediction)

Out[16]: <matplotlib.collections.PathCollection at 0x16391246eb0>
```



```
In [17]: lr.score(x_test,y_test)

Out[17]: 0.6438692690608674
```

ACURACY

```
In [18]: from sklearn.linear_model import Ridge,Lasso
```

```
In [19]: rr=Ridge(alpha=10)
         rr.fit(x_train,y_train)
         rr.score(x_test,y_test)
         rr.score(x_train,y_train)

Out[19]: 0.6411267182253932
```

```
In [20]: rr.score(x_test,y_test)

Out[20]: 0.6438691554560037
```

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

Out[21]: Lasso(alpha=10)
```

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

Out[22]: 0.44887089282073434
```

In [23]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

In [24]:
```
print(en.coef_)
```

```
[ 1.79018006e+00  1.52806735e-03  1.47647609e-06  2.41437525e-02
  9.16744500e-01  0.00000000e+00  8.50859987e-01 -1.86454844e+00
 -3.53067249e-04 -1.00839978e-04 -6.48166348e-03]
```

In [25]:
```
print(en.intercept_)
```

```
111.8594183134557
```

In [26]:
```
print(en.predict(x_test))
```

```
[129.42015878 115.09866688 112.13041113 ... 100.25105845  99.70673682
 124.60852832]
```

In [27]:
```
print(en.score(x_test,y_test))
```

```
0.6380379502227375
```

In [28]:
```
# Evaluation Metrics
from sklearn import metrics
```

In [29]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 5.5874718729549135
```

In [30]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 63.93305087761121
```

In [31]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
Root Mean Squared Error: 7.995814584994528
```