

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

```
In [2]: df=pd.read_csv("15_Horse Racing Results - 15_Horse Racing Results.csv")
df
```

Out[2]:

	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	...
...	...	...	...	...	...	...	...	...	...	...	...
27003	14.06.2020	Sha Tin	11	1200	Gress	1450000	6	A Hamelin	59	Australia	...
27004	21.06.2020	Sha Tin	2	1200	Gress	967000	7	K C Leung	57	Australia	...
27005	21.06.2020	Sha Tin	4	1200	Gress	967000	6	Blake Shinn	57	Australia	...
27006	21.06.2020	Sha Tin	5	1200	Gress	967000	14	Joao Moreira	57	New Zealand	...
27007	21.06.2020	Sha Tin	11	1200	Gress	1450000	7	C Schofield	55	New Zealand	...

27008 rows × 21 columns



```
In [3]: df.head()
```

Out[3]:

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	Country	...	Train
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	...	

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	Country	...	Train
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 DATA PREPROCESSING

```
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()
```

	Race Number	Distance	Prize money	Starting position	Jockey weight	Horse age	Pat
count	27008.000000	27008.000000	2.700800e+04	27008.000000	27008.000000	27008.000000	27008.00000
mean	5.268624	1401.666173	1.479445e+06	6.741447	55.867373	5.246408	1.67802
std	2.291281	369.107107	1.479445e+06	3.691071	2.737006	1.519880	1.63178

	Race Number	Distance	Prize money	Starting position	Jockey weight	Horse age	Pat
min	1.000000	1000.000000	6.600000e+05	1.000000	47.000000	2.000000	0.00000
25%	3.000000	1200.000000	9.200000e+05	4.000000	54.000000	4.000000	0.00000
50%	5.000000	1400.000000	9.670000e+05	7.000000	56.000000	5.000000	1.00000
75%	8.000000	1650.000000	1.450000e+06	10.000000	58.000000	6.000000	3.00000
max	11.000000	2400.000000	2.800000e+07	14.000000	63.000000	12.000000	11.00000

In [6]:

```
df.columns
```

Out[6]:

```
Index(['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'],  
      dtype='object')
```

In [7]:

```
df1=df.dropna(axis=1)  
df1
```

Out[7]:

	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	...
...	...	...	...	...	...	...	...	...	...	...	...
27003	14.06.2020	Sha Tin	11	1200	Gress	1450000	6	A Hamelin	59	Australia	...
27004	21.06.2020	Sha Tin	2	1200	Gress	967000	7	K C Leung	57	Australia	...
27005	21.06.2020	Sha Tin	4	1200	Gress	967000	6	Blake Shinn	57	Australia	...
27006	21.06.2020	Sha Tin	5	1200	Gress	967000	14	Joao Moreira	57	New Zealand	...
27007	21.06.2020	Sha Tin	11	1200	Gress	1450000	7	C Schofield	55	New Zealand	...

```
In [8]: df1.columns
```

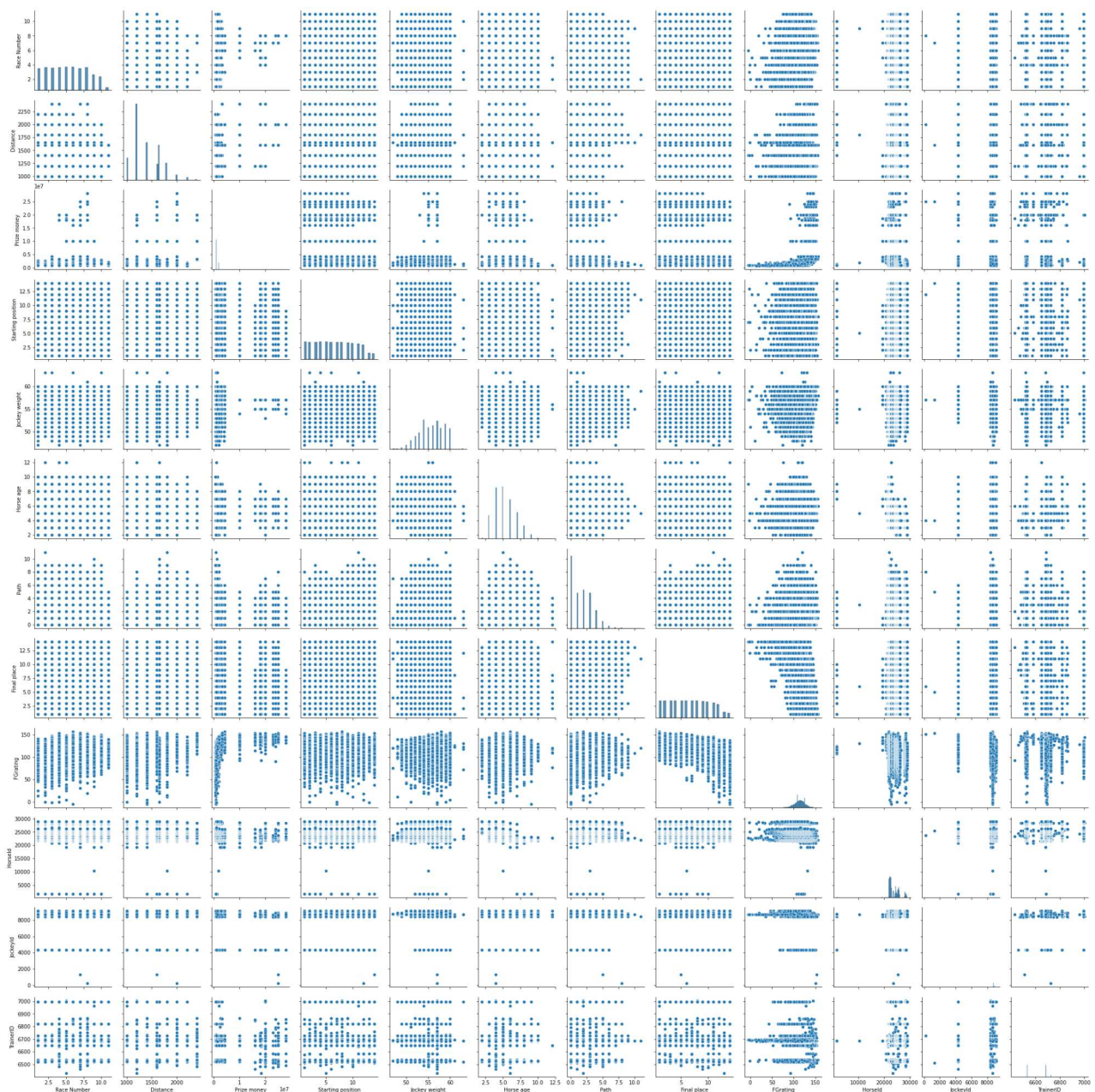
```
Out[8]: Index(['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize money',  
              'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse age',  
              'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odds',  
              'RaceType', 'HorseId', 'JockeyId', 'TrainerID'],  
              dtype='object')
```

```
In [9]: df1=df1[['Race Number', 'Distance', 'Prize money',  
                'Starting position', 'Jockey weight', 'Horse age',  
                'Path', 'Final place', 'FGrating', 'HorseId', 'JockeyId', 'TrainerID']]
```

## EDA AND VISUALIZATION

```
In [10]: sns.pairplot(df1)
```

```
Out[10]: <seaborn.axisgrid.PairGrid at 0x17142578ca0>
```

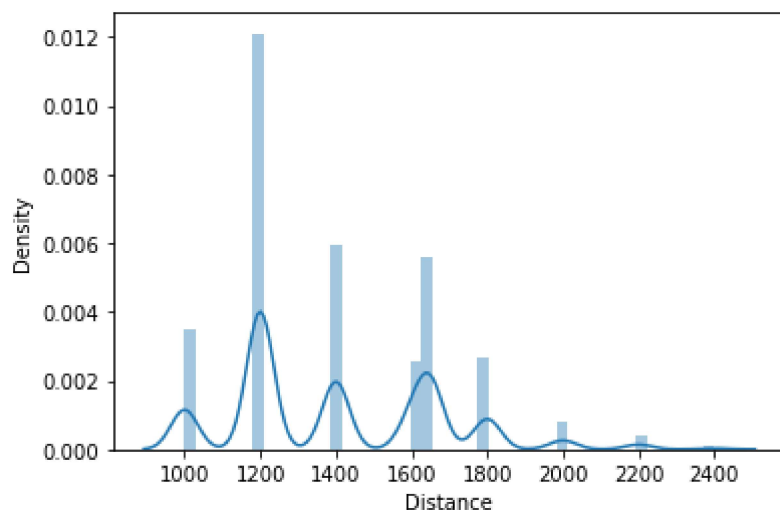


```
In [11]: sns.distplot(df1['Distance'])
```

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

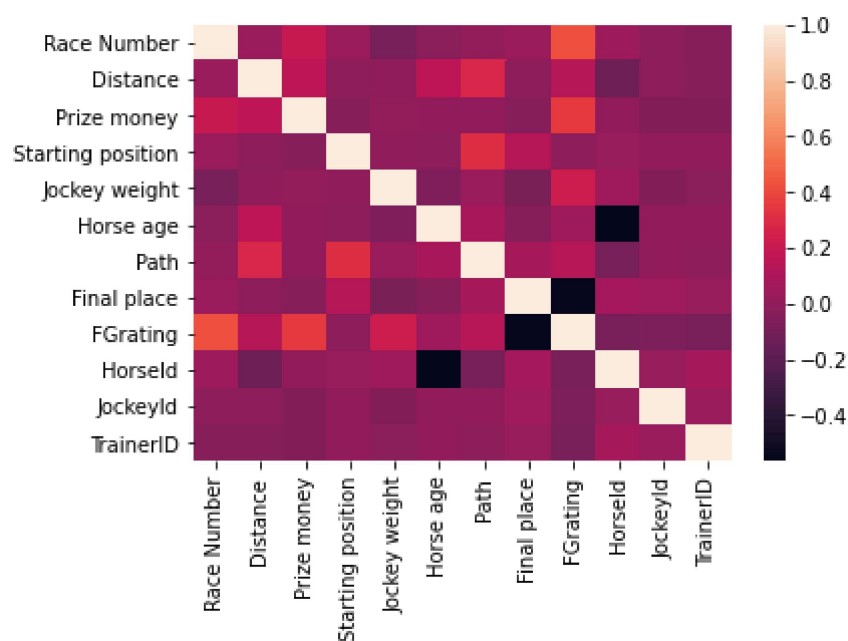
```
Out[11]: <AxesSubplot:xlabel='Distance', ylabel='Density'>
```





```
In [12]: sns.heatmap(df1.corr())
```

```
Out[12]: <AxesSubplot:>
```



## TO TRAIN THE MODEL AND MODEL BULDING

```
In [13]: x=df[['Race Number', 'Prize money',
               'Starting position', 'Jockey weight', 'Horse age',
               'Path', 'Final place', 'FG rating', 'HorseId', 'JockeyId', 'TrainerID']]
          y=df[ 'Distance']
```

```
In [14]: 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 [15]: from sklearn.linear_model import LinearRegression
          model=LinearRegression()
```

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

Out[15]: LinearRegression()

```
In [16]: lr.intercept_
```

Out[16]: 1872.5808312729644

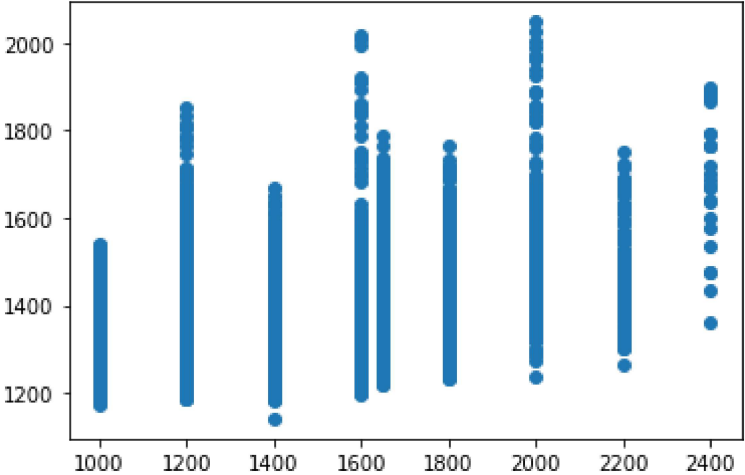
```
In [17]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
Race Number	-0.549491
Prize money	0.000019
Starting position	-7.030367
Jockey weight	-0.928717
Horse age	20.060081
Path	50.639045
Final place	0.285753
FGrating	0.669342
Horseld	-0.004034
Jockeyld	-0.004921
TrainerID	-0.078981

```
In [18]: prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]: <matplotlib.collections.PathCollection at 0x171598365e0>



```
In [19]: lr.score(x_test,y_test)
```

```
Out[19]: 0.13585363619012092
```

```
In [20]: lr.score(x_train,y_train)
```

```
Out[20]: 0.1363572686472284
```

```
In [21]: from sklearn.linear_model import Ridge,Lasso
```

```
In [22]: rr=Ridge(alpha=10)  
rr.fit(x_train,y_train)
```

```
Out[22]: Ridge(alpha=10)
```

```
In [23]: rr.score(x_test,y_test)
```

```
Out[23]: 0.1358550986396566
```

```
In [24]: rr.score(x_train,y_train)
```

```
Out[24]: 0.13635726343978705
```

```
In [25]: la=Lasso(alpha=10)  
la.fit(x_train,y_train)
```

```
Out[25]: Lasso(alpha=10)
```

```
In [26]: la.score(x_test,y_test)
```

```
Out[26]: 0.134258446471762
```

```
In [27]: la.score(x_train,y_train)
```

```
Out[27]: 0.1346925244786028
```