kaviyadevi 20106064

In [2]:
```
#to import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
#to import dataset
data1=pd.read_csv(r"C:\Users\user\Downloads\15_Horse Racing Results.CSV - 15_Hors
data1
```

Out[4]:

Jockey weight	Country	...	TrainerName	Race time	Path	Final place	FGrating	Odds	RaceType	HorseId	JockeyI
52	Sverige	...	CH Yip	83,38	2	9	110	22	Handicap	1736	865
52	Sverige	...	CH Yip	81,56	3	4	124	48	Handicap	1736	865
52	Sverige	...	CH Yip	82,36	1	6	118	11	Handicap	1736	865
54	Sverige	...	CH Yip	96,53	0	8	107	11	Handicap	1736	845
52	Sverige	...	CH Yip	94,17	0	3	123	40	Handicap	1736	865
...	...	...	...	...	...	...	...	...	...	...	.
59	Australia	...	WY So	70,87	1	9	104	25	Handicap	29038	911
57	Australia	...	KL Man	69,91	2	5	110	124	Handicap	29056	865
57	Australia	...	P O'Sullivan	69,49	0	3	114	88	Handicap	29057	877
57	New Zealand	...	AS Cruz	70,08	2	7	109	22	Handicap	29058	844
55	New Zealand	...	WY So	69,51	2	9	118	55	Handicap	29059	865

In [5]:

```
#to display top 5 rows
data=data1.head(100)
data
```

Out[5]:

	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
...	...	...	...	...	...	...	...	...	...	...
95	10.12.2017	Sha Tin	5	1200	Gress	18500000	13	Francois-Xavier Bertras	57	Great Britain
96	10.12.2017	Sha Tin	7	1600	Gress	23000000	11	Ryan Moore	57	USA
97	01.10.2017	Sha Tin	7	1000	Gress	3000000	10	Brett Prebble	59	New Zealand
98	22.10.2017	Sha Tin	7	1200	Gress	4000000	9	Brett Prebble	59	New Zealand
99	19.11.2017	Sha Tin	7	1200	Gress	4000000	3	Brett Prebble	56	New Zealand

100 rows × 21 columns

DATA CLEANING AND PREPROCESSING

In [6]:
```
#
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Dato               100 non-null    object
 1   Track              100 non-null    object
 2   Race Number        100 non-null    int64 
 3   Distance           100 non-null    int64 
 4   Surface            100 non-null    object
 5   Prize money        100 non-null    int64 
 6   Starting position  100 non-null    int64 
 7   Jockey             100 non-null    object
 8   Jockey weight      100 non-null    int64 
 9   Country            100 non-null    object
 10  Horse age          100 non-null    int64 
 11  TrainerName        100 non-null    object
 12  Race time          100 non-null    object
 13  Path               100 non-null    int64 
 14  Final place        100 non-null    int64 
 15  FGrating           100 non-null    int64 
 16  Odds               100 non-null    object
 17  RaceType           100 non-null    object
 18  HorseId            100 non-null    int64 
 19  JockeyId           100 non-null    int64 
 20  TrainerID          100 non-null    int64 
dtypes: int64(12), object(9)
memory usage: 16.5+ KB
```

In [7]:
```
#to display summary of statistics(here to know min max value)
data.describe()
```

Out[7]:

	Race Number	Distance	Prize money	Starting position	Jockey weight	Horse age	Path	Fina
count	100.000000	100.000000	1.000000e+02	100.000000	100.000000	100.00000	100.000000	100.
mean	6.910000	1446.000000	3.562200e+06	6.170000	55.870000	6.58000	1.510000	6.
std	2.099038	334.820923	4.486259e+06	3.440857	2.942736	1.35721	1.573101	3
min	1.000000	1000.000000	9.200000e+05	1.000000	49.000000	3.00000	0.000000	1.
25%	6.000000	1200.000000	1.380000e+06	3.000000	54.000000	6.00000	0.000000	4.
50%	7.000000	1400.000000	1.950000e+06	6.000000	56.000000	7.00000	1.000000	6.
75%	8.000000	1650.000000	3.000000e+06	9.000000	58.000000	8.00000	3.000000	9.
max	10.000000	2400.000000	2.300000e+07	14.000000	60.000000	9.00000	6.000000	12.

In [8]:

```
#to display the column heading
data.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')
```

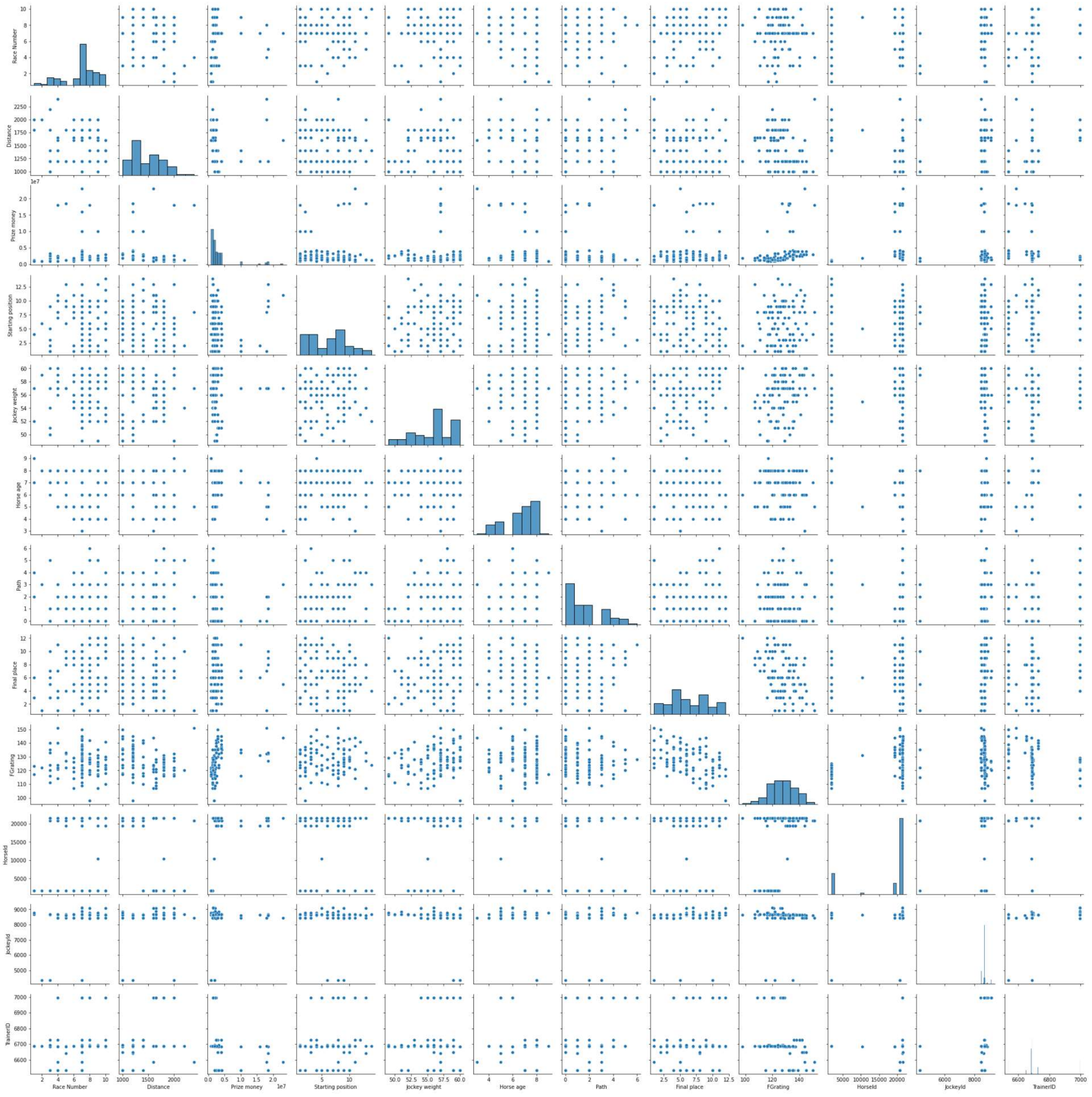
EDA and DATA VISUALIZATION

In [10]:

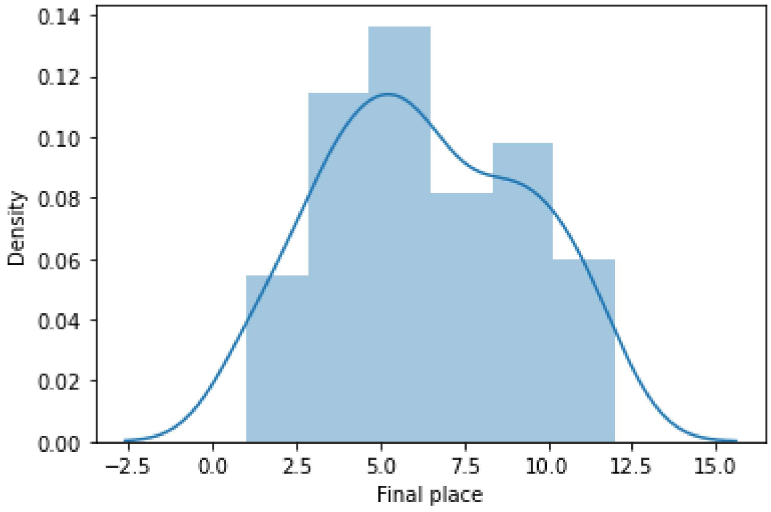
```
sns.pairplot(data)
```

Out[10]:

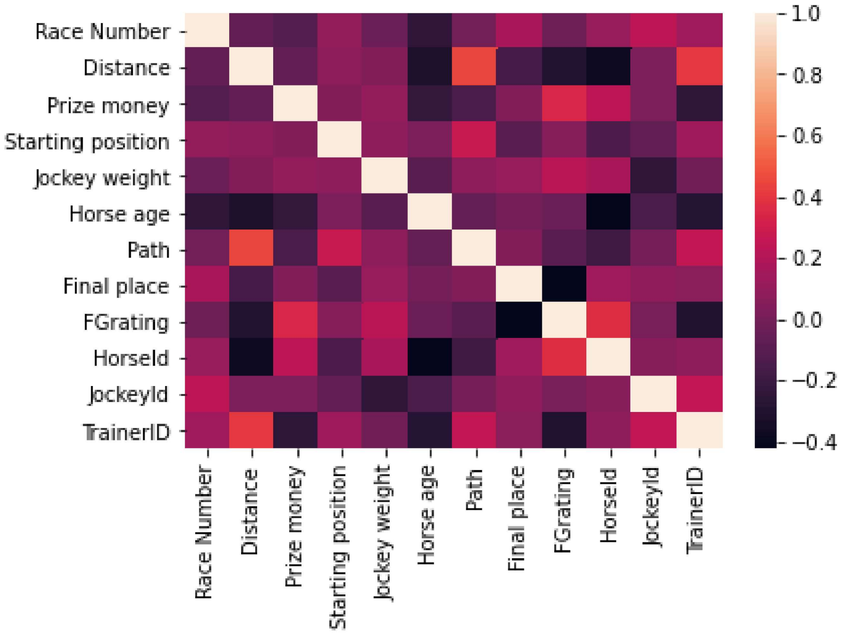
```
<seaborn.axisgrid.PairGrid at 0x1c8671ec190>
```



```
In [12]: sns.distplot(data['Final place'])

Out[12]: <AxesSubplot:xlabel='Final place', ylabel='Density'>
```



```
In [20]: f=data[['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize money',
                'Starting position', 'Jockey', 'Jockey weight', 'Horse age', 'Path', 'Final

In [21]: sns.heatmap(df.corr())

Out[21]: <AxesSubplot:>
```



TRAINING MODEL

In [37]:
```
x=df[['Prize money', 'Jockey weight', 'Horse age',  'Path', 'HorseId', 'JockeyId'
y=df['Final place']
```

In [38]:
```
#to split my dataset into trainning and test

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 [39]:
```
from sklearn.linear_model import LinearRegression

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

Out[39]:
```
LinearRegression()
```

In [40]:
```
#to find intercept
print(lr.intercept_)
```

```
-35.3547214124721
```

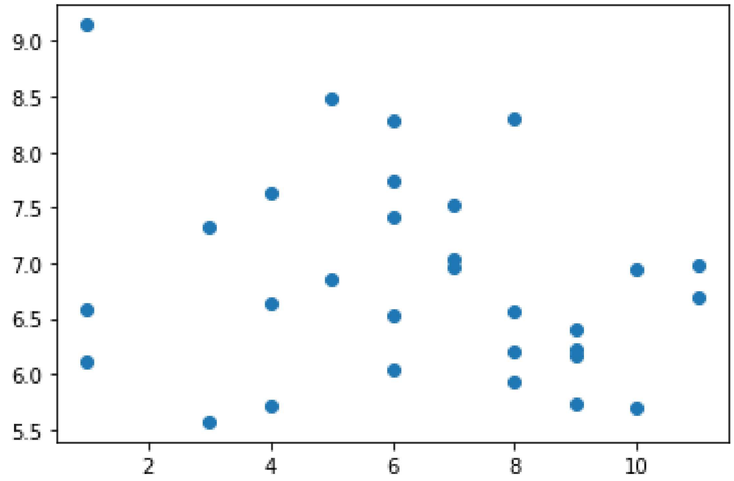
In [41]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[41]:

	Co-efficient
Prize money	8.037970e-08
Jockey weight	1.179449e-01
Horse age	3.642823e-01
Path	1.377315e-01
HorseId	2.003394e-05
JockeyId	-5.285258e-04
TrainerID	5.443788e-03

In [42]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[42]:
```
<matplotlib.collections.PathCollection at 0x1c8712ffe80>
```



In [43]:
```
print(lr.score(x_test,y_test))
```

```
-0.270819847095747
```

RIDGE AND LASSO REGRESSION

In [44]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [45]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[45]:
```
Ridge(alpha=10)
```

In [46]:
```
rr.score(x_test,y_test)
```

Out[46]:
```
-0.26805738520286426
```

In [47]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[47]:
```
Lasso(alpha=10)
```

In [48]:
```
la.score(x_test,y_test)
```

Out[48]:
```
-0.230323676449393
```