Problem Statement

Linear Regression

Import Libraries

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

In [4]:
```
a=pd.read_csv("horse1.csv")
a
```

Out[4]:

	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

To display top 10 rows

In [5]:
```
c=a.head(15)
c
```

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	..
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	28.03.2018	Happy Valley	8	1800	Gress	1310000	9	M F Poon	53	Sverige	..
11	11.04.2018	Happy Valley	6	1650	Gress	1310000	11	W M Lai	55	Sverige	..
12	25.04.2018	Happy Valley	3	2200	Gress	1310000	2	W M Lai	54	Sverige	..
13	09.05.2018	Happy Valley	7	1650	Gress	1310000	3	W M Lai	54	Sverige	..
14	22.09.2018	Sha Tin	4	1600	Gress	920000	11	C Y Ho	57	Sverige	..

15 rows × 21 columns

To find Missing values

In [6]:
```
c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
```

```
 0   Dato               15 non-null     object
 1   Track              15 non-null     object
 2   Race Number        15 non-null     int64
 3   Distance           15 non-null     int64
 4   Surface            15 non-null     object
 5   Prize money        15 non-null     int64
 6   Starting position  15 non-null     int64
 7   Jockey             15 non-null     object
 8   Jockey weight      15 non-null     int64
 9   Country            15 non-null     object
 10  Horse age          15 non-null     int64
 11  TrainerName        15 non-null     object
 12  Race time          15 non-null     object
 13  Path               15 non-null     int64
 14  Final place        15 non-null     int64
 15  FGrating           15 non-null     int64
 16  Odds               15 non-null     object
 17  RaceType           15 non-null     object
 18  HorseId            15 non-null     int64
 19  JockeyId           15 non-null     int64
 20  TrainerID          15 non-null     int64
dtypes: int64(12), object(9)
memory usage: 2.6+ KB
```

To display summary of statistics

In [7]:
```
a.describe()
```

Out[7]:

	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	27008.0(
mean	5.268624	1401.666173	1.479445e+06	6.741447	55.867373	5.246408	1.67
std	2.780088	276.065045	2.162109e+06	3.691071	2.737006	1.519880	1.63
min	1.000000	1000.000000	6.600000e+05	1.000000	47.000000	2.000000	0.0(
25%	3.000000	1200.000000	9.200000e+05	4.000000	54.000000	4.000000	0.0(
50%	5.000000	1400.000000	9.670000e+05	7.000000	56.000000	5.000000	1.0(
75%	8.000000	1650.000000	1.450000e+06	10.000000	58.000000	6.000000	3.0(
max	11.000000	2400.000000	2.800000e+07	14.000000	63.000000	12.000000	11.0(

To display column heading

In [8]:
```
a.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')
```

Pairplot

In [9]:
```
s=a.dropna(axis=1)
s
```

Out[9]:

	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

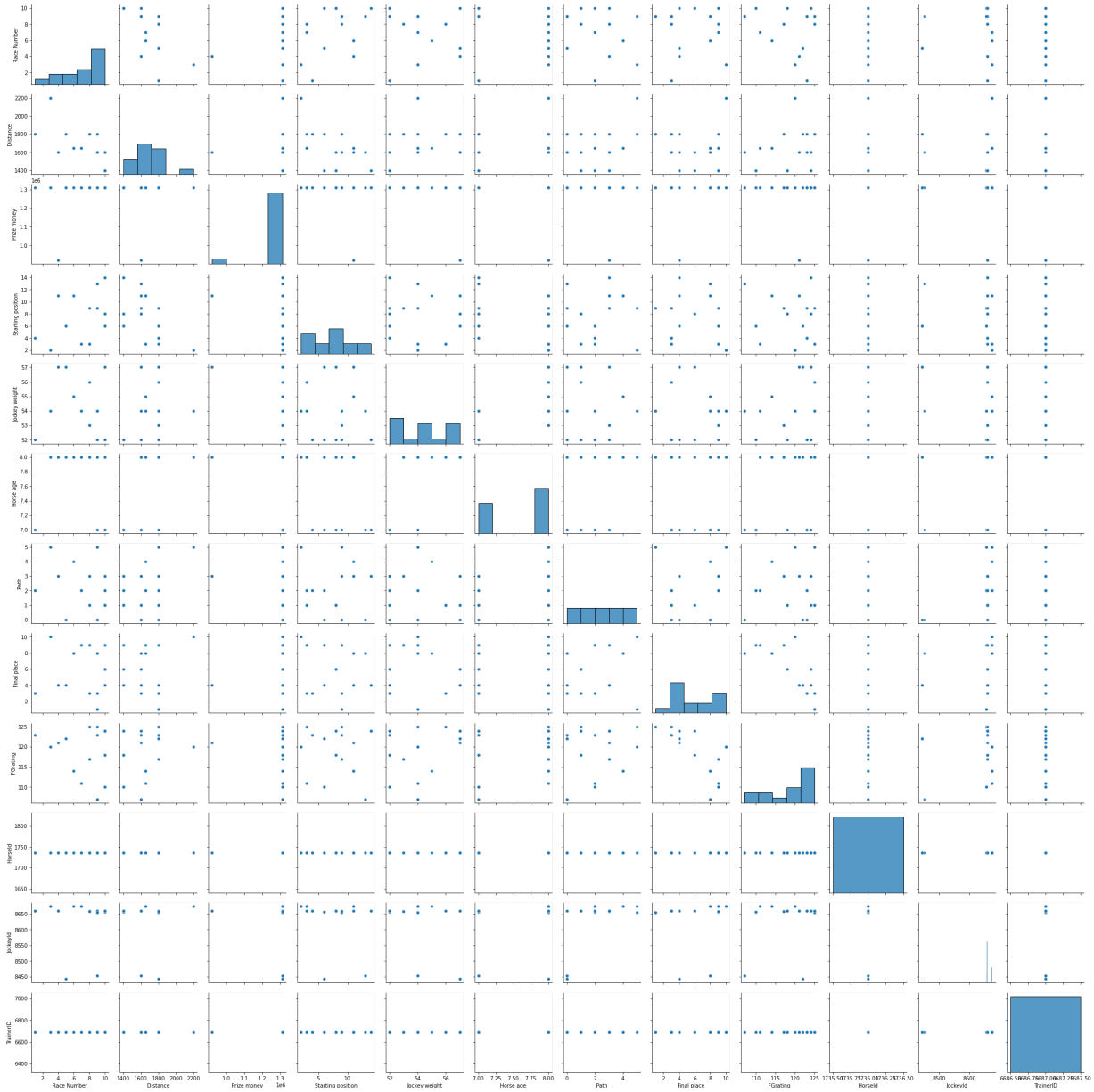
27008 rows × 21 columns

In [10]:
```
s.columns
```

Out[10]:
```
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 [11]:
```
sns.pairplot(c)
```

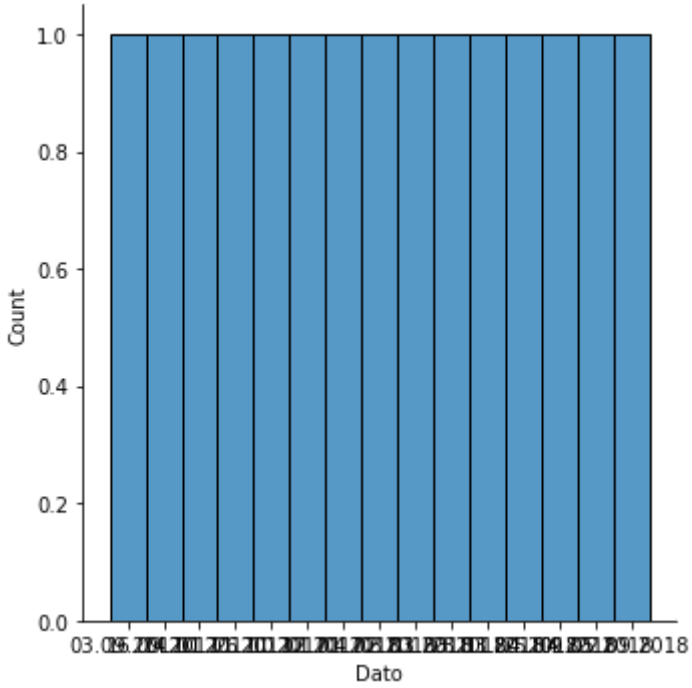
Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x24f8ce17b50>
```



Distribution Plot

In [12]:
```
sns.displot(c['Dato'])
```

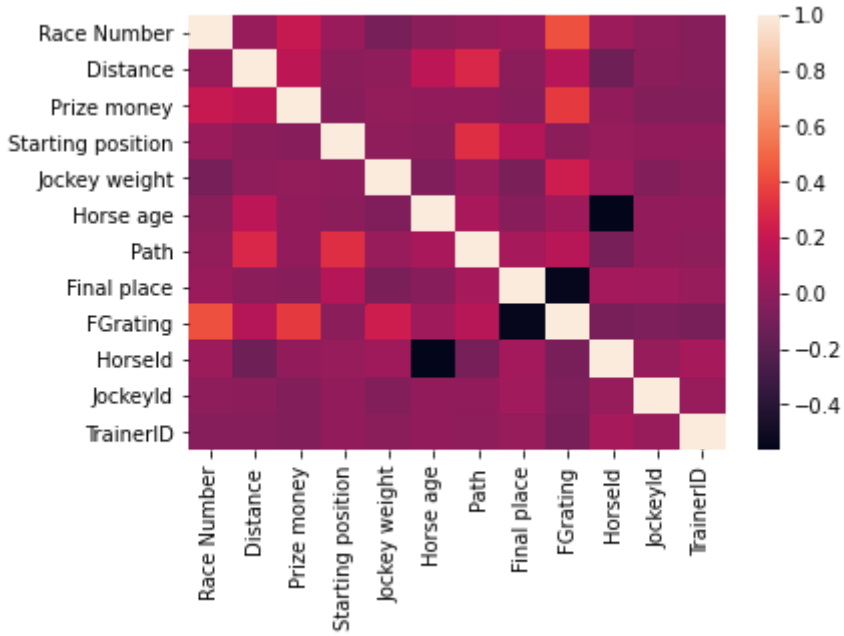
Out[12]:
```
<seaborn.axisgrid.FacetGrid at 0x24f90eb0370>
```



Correlation

In [24]:
```
b=a[['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']]
sns.heatmap(b.corr())
```

Out[24]: <AxesSubplot:>



Train the model - Model Building

In [47]:
```
g=c[['Distance','Prize money',
       'Starting position','HorseId','JockeyId']]
h=c['TrainerID']
```

To split dataset into training end test

In [48]:
```
from sklearn.model_selection import train_test_split
g_train,g_test,h_train,h_test=train_test_split(g,h,test_size=0.6)
```

To run the model

In [49]:
```
from sklearn.linear_model import LinearRegression
```

In [50]:
```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

Out[50]:
```
LinearRegression()
```

In [51]:
```
print(lr.intercept_)
```

```
6687.0
```

Coeffecient

In [52]:
```
coeff=pd.DataFrame(lr.coef_,g.columns,columns=['Co-effecient'])
coeff
```

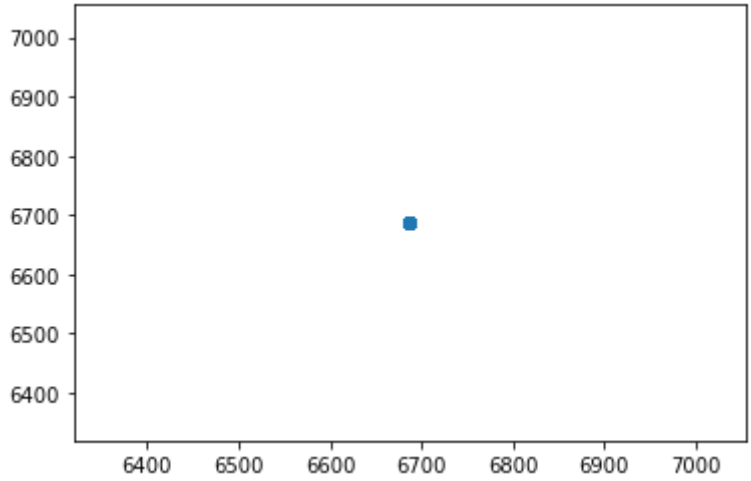
Out[52]:

	Co-effecient
Distance	0.0
Prize money	0.0
Starting position	0.0
HorseId	0.0
JockeyId	0.0

Best Fit line

In [53]:
```
prediction=lr.predict(g_test)
plt.scatter(h_test,prediction)
```

Out[53]:
```
<matplotlib.collections.PathCollection at 0x24f9417c9d0>
```



To find score

In [54]:
```
print(lr.score(g_test,h_test))
```

```
1.0
```

Import Lasso and ridge

In [55]:
```
from sklearn.linear_model import Ridge,Lasso
```

Ridge

In [56]:
```
ri=Ridge(alpha=5)
ri.fit(g_train,h_train)
```

Out[56]:
```
Ridge(alpha=5)
```

In [57]:
```
ri.score(g_test,h_test)
```

Out[57]:
```
1.0
```

In [58]:
```
ri.score(g_train,h_train)
```

Out[58]:
```
1.0
```

Lasso

In [59]:
```
l=Lasso(alpha=6)
l.fit(g_train,h_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.
py:530: ConvergenceWarning: Objective did not converge. You might want to increase t
he number of iterations. Duality gap: 0.0, tolerance: 0.0
  model = cd_fast.enet_coordinate_descent(
```

Out[59]:
```
Lasso(alpha=6)
```

In [60]:
```
l.score(g_test,h_test)
```

Out[60]:
```
1.0
```

In [61]:
```
ri.score(g_train,h_train)
```

Out[61]:
```
1.0
```