

# DATA COLLECTION

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

```
In [3]: # To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\15_Horse.csv")
sd
```

Out[3]:

	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	Austr
27004	21.06.2020	Sha Tin	2	1200	Gress	967000	7	K C Leung	57	Austr
27005	21.06.2020	Sha Tin	4	1200	Gress	967000	6	Blake Shinn	57	Austr
27006	21.06.2020	Sha Tin	5	1200	Gress	967000	14	Joao Moreira	57	N Zeale
27007	21.06.2020	Sha Tin	11	1200	Gress	1450000	7	C Schofield	55	N Zeale

27008 rows × 21 columns



```
In [4]: # to display top 10 rows
sd.head(10)
```

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
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 [5]: `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  FGating                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 [6]: `# to display summary of statistics`  
`sd.describe()`

Out[6]:

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

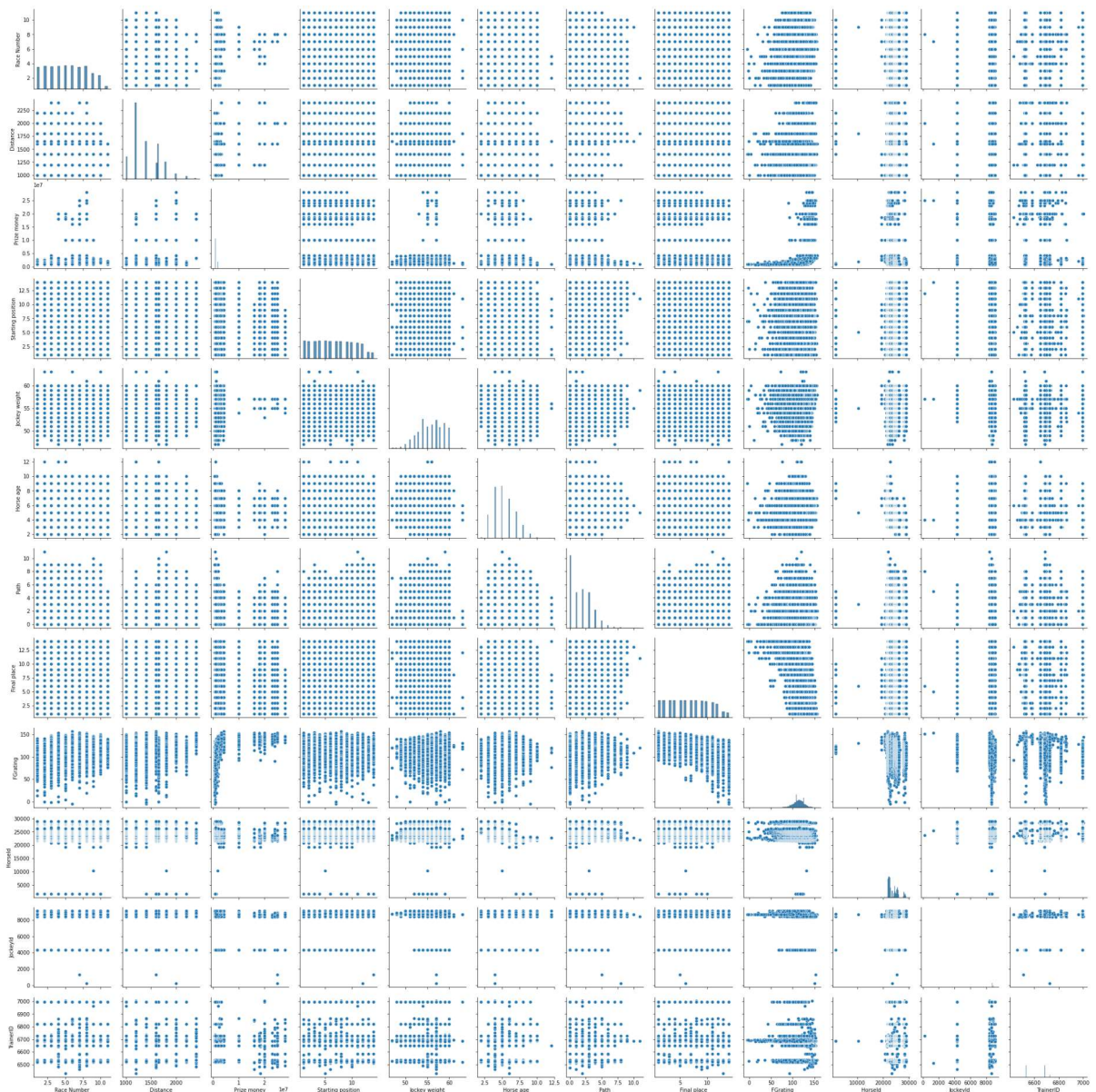
```
In [7]: #to display colums heading
sd.columns
```

```
Out[7]: 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 visualization

```
In [8]: sns.pairplot(sd)
```

```
Out[8]: <seaborn.axisgrid.PairGrid at 0x1a1c049e4f0>
```

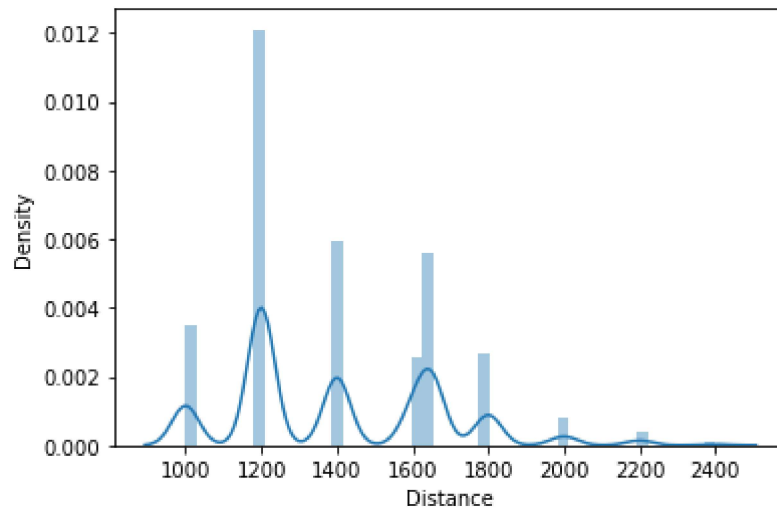


```
In [10]: sns.distplot(sd['Distance'])
```

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).

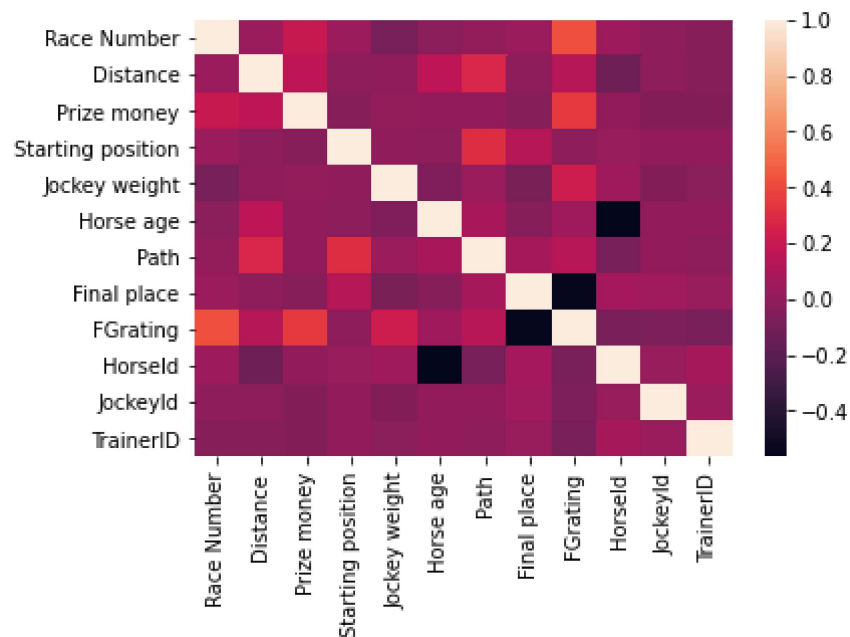
```
warnings.warn(msg, FutureWarning)
```

```
Out[10]: <AxesSubplot:xlabel='Distance', ylabel='Density'>
```



```
In [11]: sns.heatmap(sd.corr())
```

```
Out[11]: <AxesSubplot:>
```



```
In [12]: sd1=sd[['Race Number', 'Distance', 'Prize money',  
                'Starting position', 'Jockey weight']]
```

# TO TRAIN THE MODEL \_MODEL BUILDING

we are going to train Linear Regression model; we need to split out the data into two variables x and y where x is independent on x (output) and y is dependent on x(output) address column as it is not required our model

```
In [13]: x= sd1[['Race Number', 'Distance', 'Prize money',  
              'Starting position']]  
y=sd1['Jockey weight']
```

```
In [14]: # 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 [15]: from sklearn.linear_model import LinearRegression  
  
lr=LinearRegression()  
lr.fit(x_train,y_train)
```

```
Out[15]: LinearRegression()
```

```
In [16]: from sklearn.linear_model import LinearRegression  
  
lr=LinearRegression()  
lr.fit(x_train,y_train)
```

```
Out[16]: LinearRegression()
```

```
In [17]: print(lr.intercept_)
```

```
56.40025543461258
```

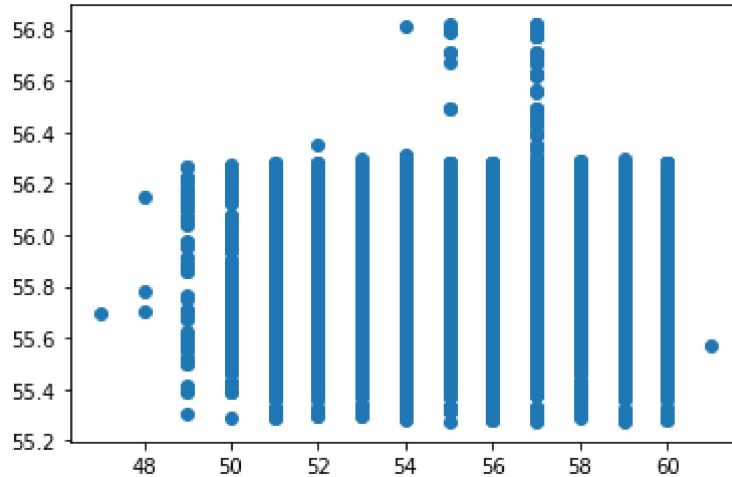
```
In [18]: coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[18]:
```

	Co-efficient
Race Number	-9.810761e-02
Distance	-5.999573e-05
Prize money	4.729432e-08
Starting position	-9.058640e-04

```
In [19]: prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[19]: <matplotlib.collections.PathCollection at 0x1a1d6f4e160>



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

0.009287953407006166

```
In [21]: lr.score(x_train,y_train)
```

Out[21]: 0.009824264440380648

```
In [22]: from sklearn.linear_model import Ridge,Lasso
```

```
In [23]: dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[23]: Ridge(alpha=10)

```
In [24]: dr.score(x_test,y_test)
```

Out[24]: 0.00928795787671921

```
In [25]: dr.score(x_train,y_train)
```

Out[25]: 0.00982426439192996

```
In [26]: la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[26]: Lasso(alpha=10)

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

Out[27]: 0.00013780058443946253

In [28]: `la.score(x_train,y_train)`

Out[28]: 0.00025664757221044443

In [ ]:

In [ ]: