Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

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("Horse Racing Results")
```

To display top 10 rows

In [3]:
```
df.head(10)
```

Out[3]:

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	Country	...	TrainerName	Race time	Path	Final place	F
0	03.09.2017	Sha Tin	10	1400	Gress	1310000	6	K C Leung	52	Sverige	...	CH Yip	83,38	2	9	
1	16.09.2017	Sha Tin	10	1400	Gress	1310000	14	C Y Ho	52	Sverige	...	CH Yip	81,56	3	4	
2	14.10.2017	Sha Tin	10	1400	Gress	1310000	8	C Y Ho	52	Sverige	...	CH Yip	82,36	1	6	
3	11.11.2017	Sha Tin	9	1600	Gress	1310000	13	Brett Prebble	54	Sverige	...	CH Yip	96,53	0	8	
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	Sverige	...	CH Yip	94,17	0	3	
5	10.12.2017	Sha Tin	1	1800	Gress	1310000	4	C Y Ho	52	Sverige	...	CH Yip	107,92	2	3	
6	01.01.2018	Sha Tin	9	1800	Gress	1310000	9	C Schofield	54	Sverige	...	CH Yip	108,63	5	1	
7	04.02.2018	Sha Tin	5	1800	Gress	1310000	6	Joao Moreira	57	Sverige	...	CH Yip	108,19	0	4	
8	03.03.2018	Sha Tin	8	1800	Gress	1310000	3	C Y Ho	56	Sverige	...	CH Yip	107,65	1	3	
9	11.03.2018	Sha Tin	10	1600	Gress	1310000	8	C Y Ho	57	Sverige	...	CH Yip	94,46	1	6	

10 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]:
```python
# Display the statistical summary
df.describe()
```

Out[5]:

	Race Number	Distance	Prize money	Starting position	Jockey weight	Horse age	Path	Final place	FGrating
count	27008.000000	27008.000000	2.700800e+04	27008.000000	27008.000000	27008.000000	27008.000000	27008.000000	27008.000000
mean	5.268624	1401.666173	1.479445e+06	6.741447	55.867373	5.246408	1.678021	6.685834	113.428318
std	2.780088	276.065045	2.162109e+06	3.691071	2.737006	1.519880	1.631784	3.664551	13.314949
min	1.000000	1000.000000	6.600000e+05	1.000000	47.000000	2.000000	0.000000	1.000000	-5.000000
25%	3.000000	1200.000000	9.200000e+05	4.000000	54.000000	4.000000	0.000000	4.000000	106.000000
50%	5.000000	1400.000000	9.670000e+05	7.000000	56.000000	5.000000	1.000000	7.000000	114.000000
75%	8.000000	1650.000000	1.450000e+06	10.000000	58.000000	6.000000	3.000000	10.000000	122.000000
max	11.000000	2400.000000	2.800000e+07	14.000000	63.000000	12.000000	11.000000	14.000000	157.000000

In [6]:
```python
# To display the col headings
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]:
```python
cols=df.dropna(axis=1)
```

In [8]:
```python
cols.columns
```

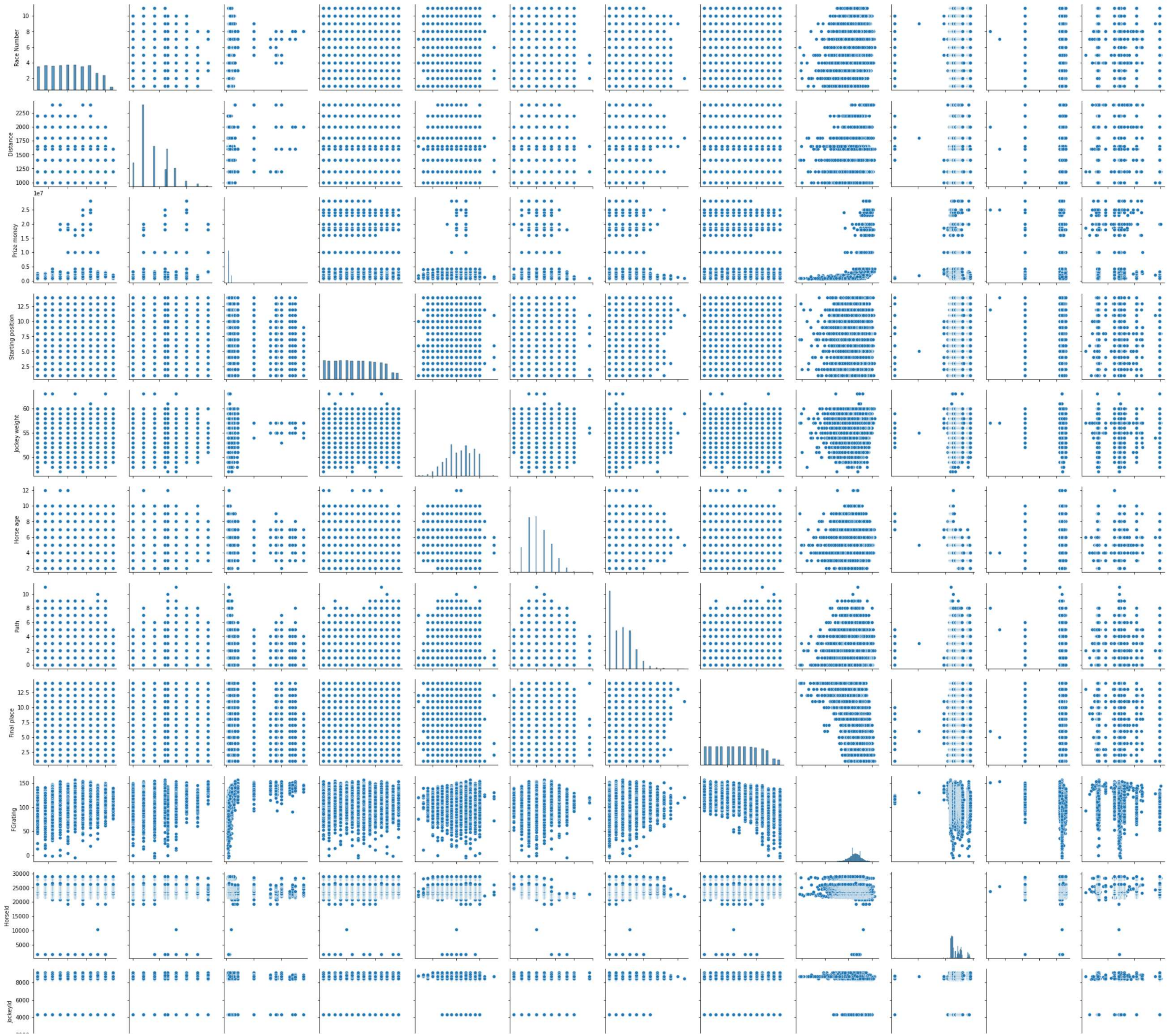
Out[8]:
```
Index(['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize money',
       'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse age',
       'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odds',
       'RaceType', 'HorseId', 'JockeyId', 'TrainerID'],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

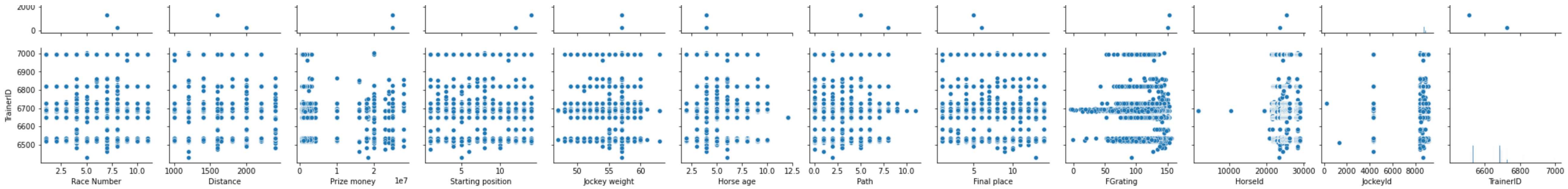
EDA and Visualization

In [9]:

```
sns.pairplot(cols)
```

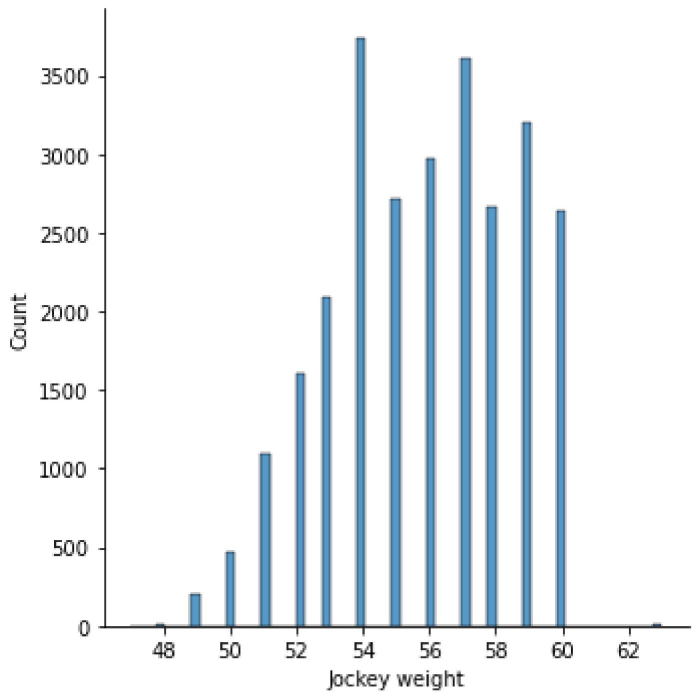
Out[9]: <seaborn.axisgrid.PairGrid at 0x1b6182f23d0>


Race Number
Distance
Prize money
1e7
Starting position
Jockey weight
Horse age
Path
Final place
FGrating
HorseId
JockeyId



In [10]:
```
1 sns.displot(df['Jockey weight'])
```

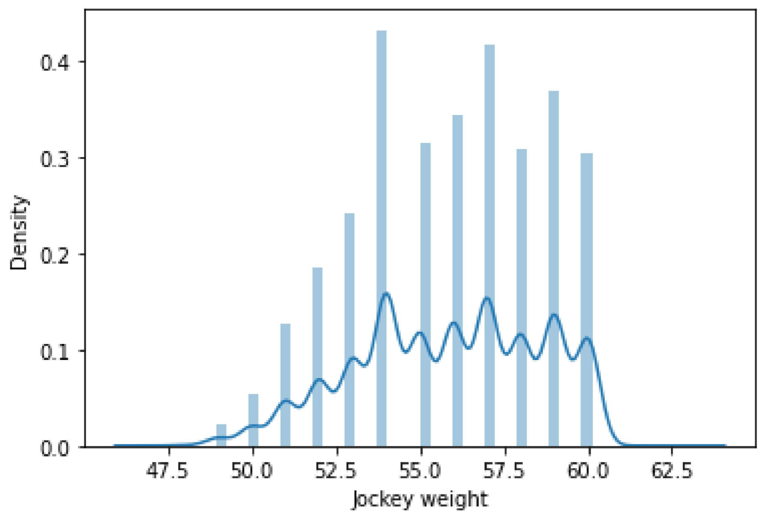
Out[10]: <seaborn.axisgrid.FacetGrid at 0x1b62e14eee0>



In [11]:
```
# We use displot in older version we get distplot use displot
sns.distplot(df['Jockey weight'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated 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='Jockey weight', ylabel='Density'>



In [12]:

```
df1=cols[['Race Number', 'Distance',
       'Jockey weight', 'Horse age']]
df1
```

Out[12]:

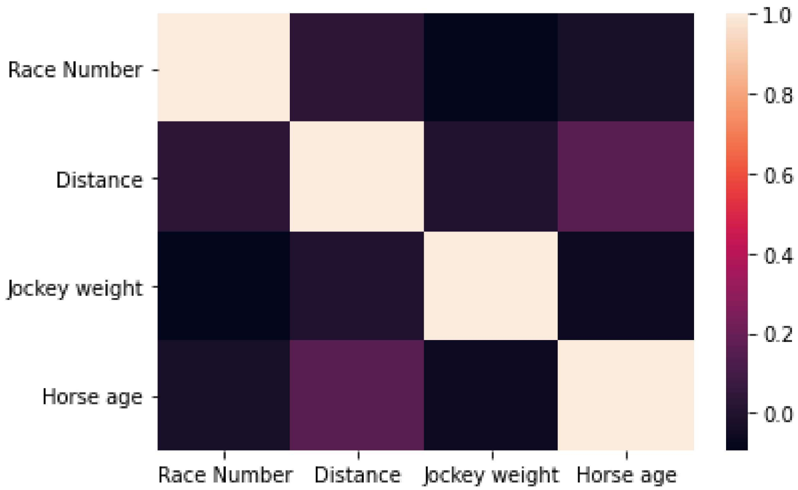
	Race Number	Distance	Jockey weight	Horse age
0	10	1400	52	7
1	10	1400	52	7
2	10	1400	52	7
3	9	1600	54	7
4	9	1600	52	7
...	...	...	...	...
27003	11	1200	59	3
27004	2	1200	57	3
27005	4	1200	57	3
27006	5	1200	57	3
27007	11	1200	55	4

27008 rows × 4 columns

In [13]:

```python
sns.heatmap(df1.corr())
```

Out[13]: <AxesSubplot:>



To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [14]:

```python
x=df1[['Race Number', 'Distance',
       'Jockey weight', 'Horse age']]
y=df1[['Distance']]
```

To split the dataset into test data

In [15]:

```python
# importing lib for splitting test data
from sklearn.model_selection import train_test_split
```

In [16]:
```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [17]:
```python
from sklearn.linear_model import LinearRegression

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

Out[17]:
```
LinearRegression()
```

In [18]:
```python
print(lr.intercept_)
```

```
[-6.82121026e-13]
```

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

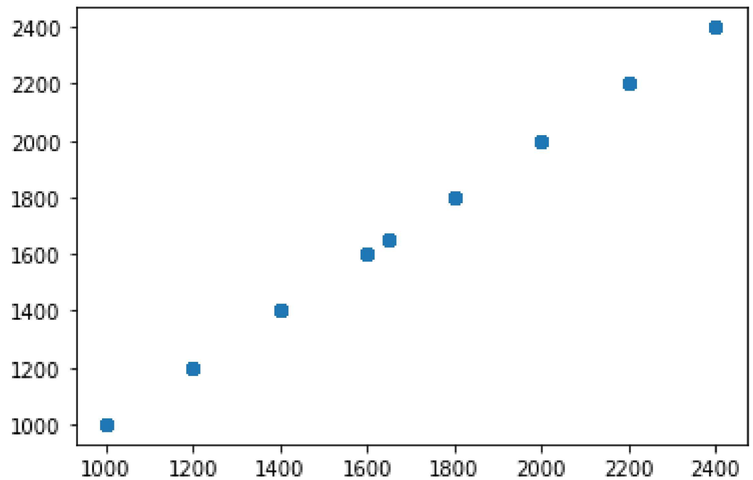
```
1.0
```

In [20]:
```python
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[20]:

	0	1	2	3
0	-1.293701e-15	1.0	8.004529e-17	7.520177e-16

In [21]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```
Out[21]: <matplotlib.collections.PathCollection at 0x1b62ffecd90>



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

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

In [24]:
```
rr.score(x_test,y_test)
```
Out[24]: 1.0

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]: 0.9999999831678777

Elastic net

In [27]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:
```
ElasticNet()
```

In [28]:
```
print(en.coef_)
```

```
[0.         0.99998703 0.         0.        ]
```

In [29]:
```
print(en.intercept_)
```

```
[0.01818225]
```

In [30]:
```
prediction=en.predict(x_test)
prediction
```

Out[30]:
```
array([1200.00261376, 1649.99677557, 1200.00261376, ..., 1649.99677557,
       1400.00001901, 1200.00261376])
```

In [31]:
```
print(en.score(x_test,y_test))
```

```
0.999999999831681
```

EVALUATION METRICS

In [32]:
```
from sklearn import metrics
```

In [33]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.002922189737464215
```

In [34]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 1.2486062600118047e-05
```

In [35]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 0.0035335623102073703
```