Problem Statement

A real estate agent want help to predict the house price for regions in USA.He gave us the dataset to work on to use linear regression model.Create a model that helps him to estimate of what the house would sell for

Import libraries

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

In [2]:
```
# To import dataset
df=pd.read_csv('14 Iris csv')
df
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [3]:
```
# To display top 10 rows
df.head(10)
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

Data Cleaning and Pre-Processing

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

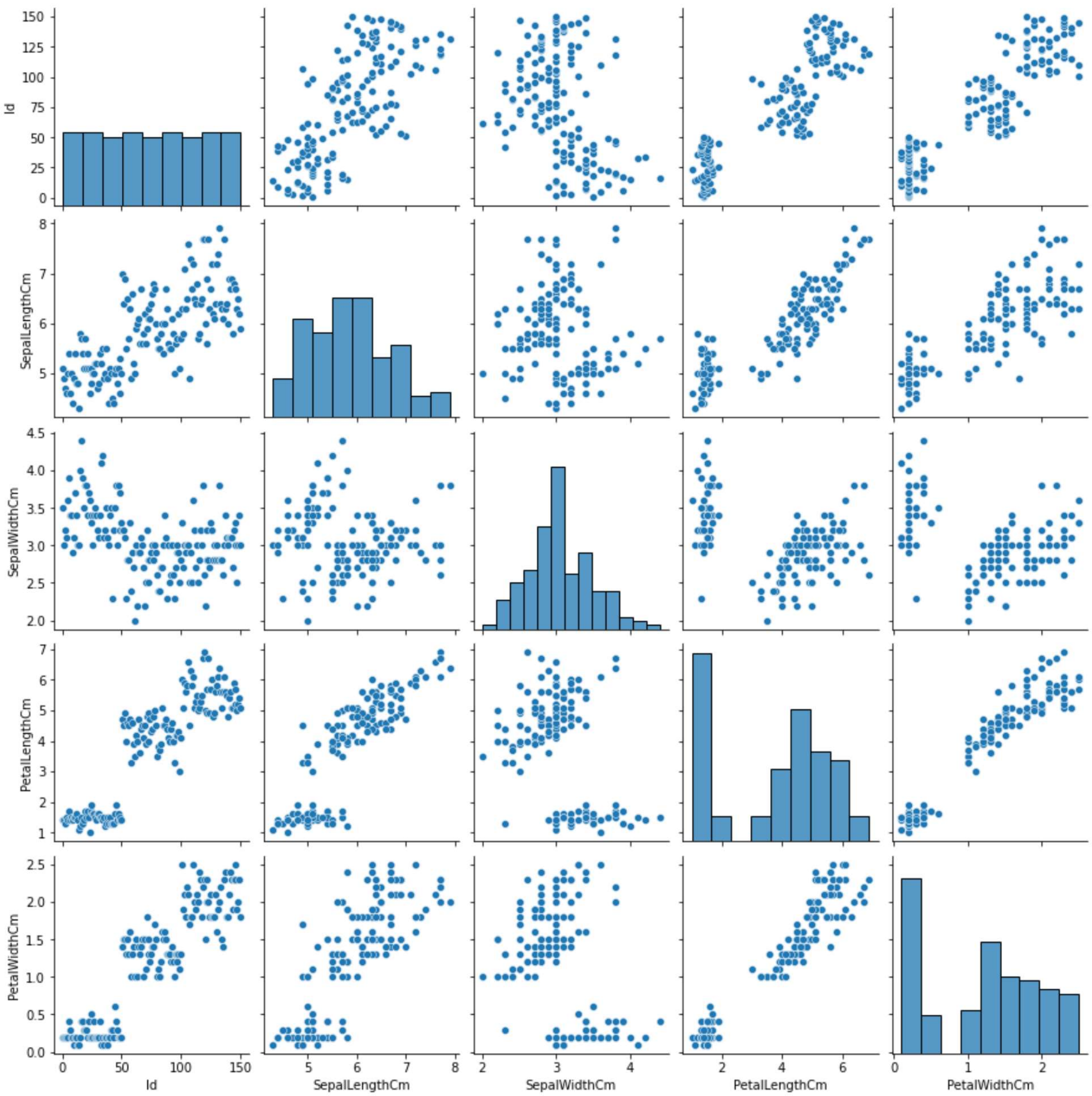
In [7]:
```
a = df.dropna(axis='columns')
a.columns
```

Out[7]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

EDA and Visualization

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

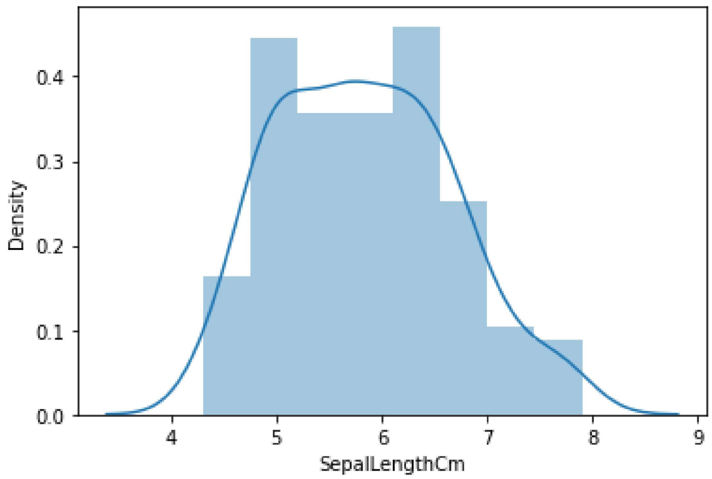
Out[8]: <seaborn.axisgrid.PairGrid at 0x15a5ec71910>



In [9]:
```
sns.distplot(a['SepalLengthCm'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `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 flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

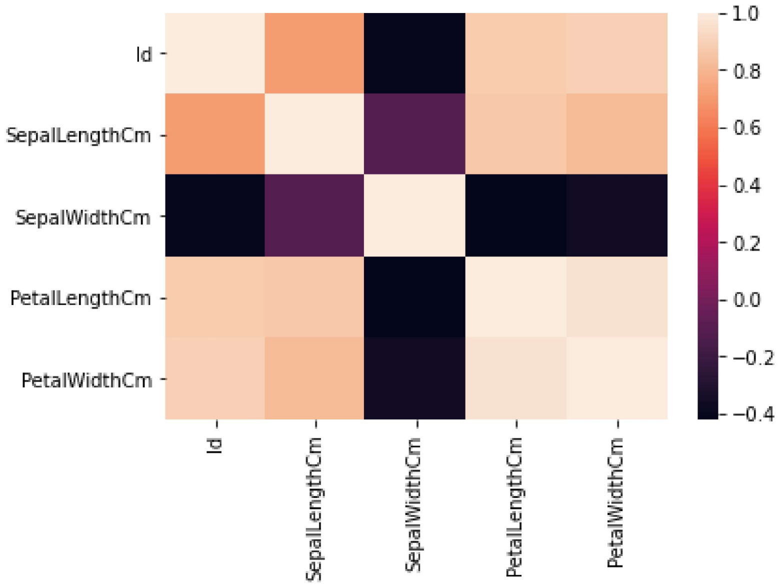
Out[9]: <AxesSubplot:xlabel='SepalLengthCm', ylabel='Density'>



In [10]:
```
a1=a[['Id','SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',]]
```

In [11]:
```
sns.heatmap(a1.corr())
```

Out[11]: <AxesSubplot:>



To Train the Model - Model Building

We are going to train Linear Regression model;We need to split out data into two variables x and y where x is independent variable (input) and y is dependent on x(output). We could ignore address column as it is not required for our model.

In [12]:
```
x=a1[['Id', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',]]
y=a1['SepalLengthCm']
```

To split my dataset into training and test data

In [13]:
```
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 [14]:
```
from sklearn.linear_model import LinearRegression

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

Out[14]:
```
LinearRegression()
```

In [15]:
```
print(lr.intercept_)
```

```
1.7026234125487694
```

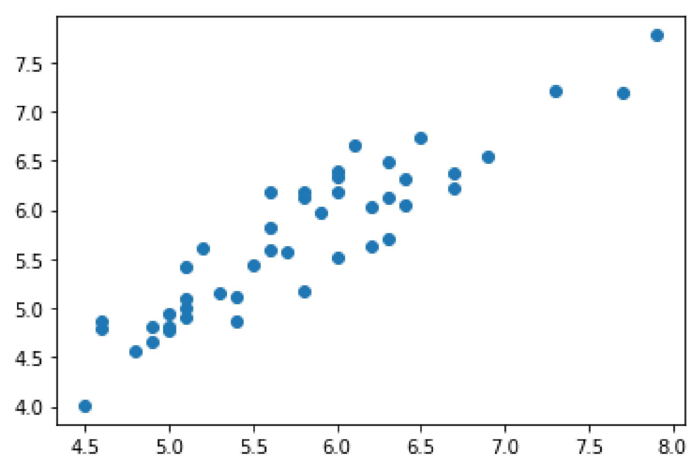
In [16]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Id	-0.001125
SepalWidthCm	0.682829
PetalLengthCm	0.714102
PetalWidthCm	-0.471901

In [17]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x15a61cc7460>
```



In [18]:
```
print(lr.score(x_test,y_test))
```

```
0.8254751586975164
```

ACCURACY

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[20]: 0.8411430653884009

In [21]:
```
rr.score(x_test,y_test)
```

Out[21]: 0.8005662509460001

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

Out[22]: Lasso(alpha=10)

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

Out[23]: 0.4095506812994495

In [24]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[24]: ElasticNet()

In [25]:
```

print(en.coef_)
```

```
[0.01366795 0.         0.         0.        ]
```

In [26]:
```
print(en.intercept_)
```

```
4.765804415504331
```

In [27]:
```

print(en.predict(x_test))
```

```
[6.20093902 5.58588133 5.24418262 6.56997363 5.70889287 5.39453005
 6.24194287 5.96858389 6.10526338 5.33985826 6.15993518 5.61321723
 5.12117108 4.86148006 5.68155697 5.05283134 6.61097748 5.54487749
 5.23051467 5.9139121  4.79314031 6.06425953 4.98449159 6.72032107
 5.76356466 5.09383518 4.77947236 5.3808621  5.66788902 5.20317877
 5.95491595 5.4355339  5.13483903 5.79090056 5.941248   6.29661466
 5.72256082 6.3922903  5.44920185 5.49020569 6.66564927 5.62688518
 4.97082364 5.06649929 4.82047621]
```

In [28]:
```
print(en.score(x_test,y_test))
```

```
0.4481170731441402
```

Evaluation Metrics

In [29]:
```
from sklearn import metrics
```

In [30]:
```

print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 0.2801697240151496
```

In [31]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.10799510994690874
```

In [32]:
```
print("Root Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 0.10799510994690874
```

In [33]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[33]:
```
ElasticNet()
```

In [34]:
```

print(en.coef_)
```

```
[0.01366795 0.         0.         0.        ]
```

In [35]:
```
print(en.intercept_)
```

```
4.765804415504331
```

In [36]:
```

print(en.predict(x_test))
```

```
[6.20093902 5.58588133 5.24418262 6.56997363 5.70889287 5.39453005
 6.24194287 5.96858389 6.10526338 5.33985826 6.15993518 5.61321723
 5.12117108 4.86148006 5.68155697 5.05283134 6.61097748 5.54487749
 5.23051467 5.9139121  4.79314031 6.06425953 4.98449159 6.72032107
 5.76356466 5.09383518 4.77947236 5.3808621  5.66788902 5.20317877
 5.95491595 5.4355339  5.13483903 5.79090056 5.941248   6.29661466
 5.72256082 6.3922903  5.44920185 5.49020569 6.66564927 5.62688518
 4.97082364 5.06649929 4.82047621]
```

In [37]:
```
print(en.score(x_test,y_test))
```

```
0.4481170731441402
```

In [38]:
```
from sklearn import metrics
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Mean Absolytre Error: 0.2801697240151496
Mean Squared Error: 0.10799510994690874
Root Mean Squared Error: 0.3286260944400319
```