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

Data Collection

In [1]:

```
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [50]:

```
#import the dataset
data=pd.read_csv(r"C:\Users\user\Desktop\Vicky\14_Iris.csv")[0:500]
```

In [51]:

```
#to display top 10 rows
data.head()
```

Out[51]:

	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

In [52]:

```
#to display null values
data.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 [53]:
```
data.shape
```

Out[53]:
```
(150, 6)
```

In [54]:
```
#to display summary of statistics
data.describe()
```

Out[54]:

	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

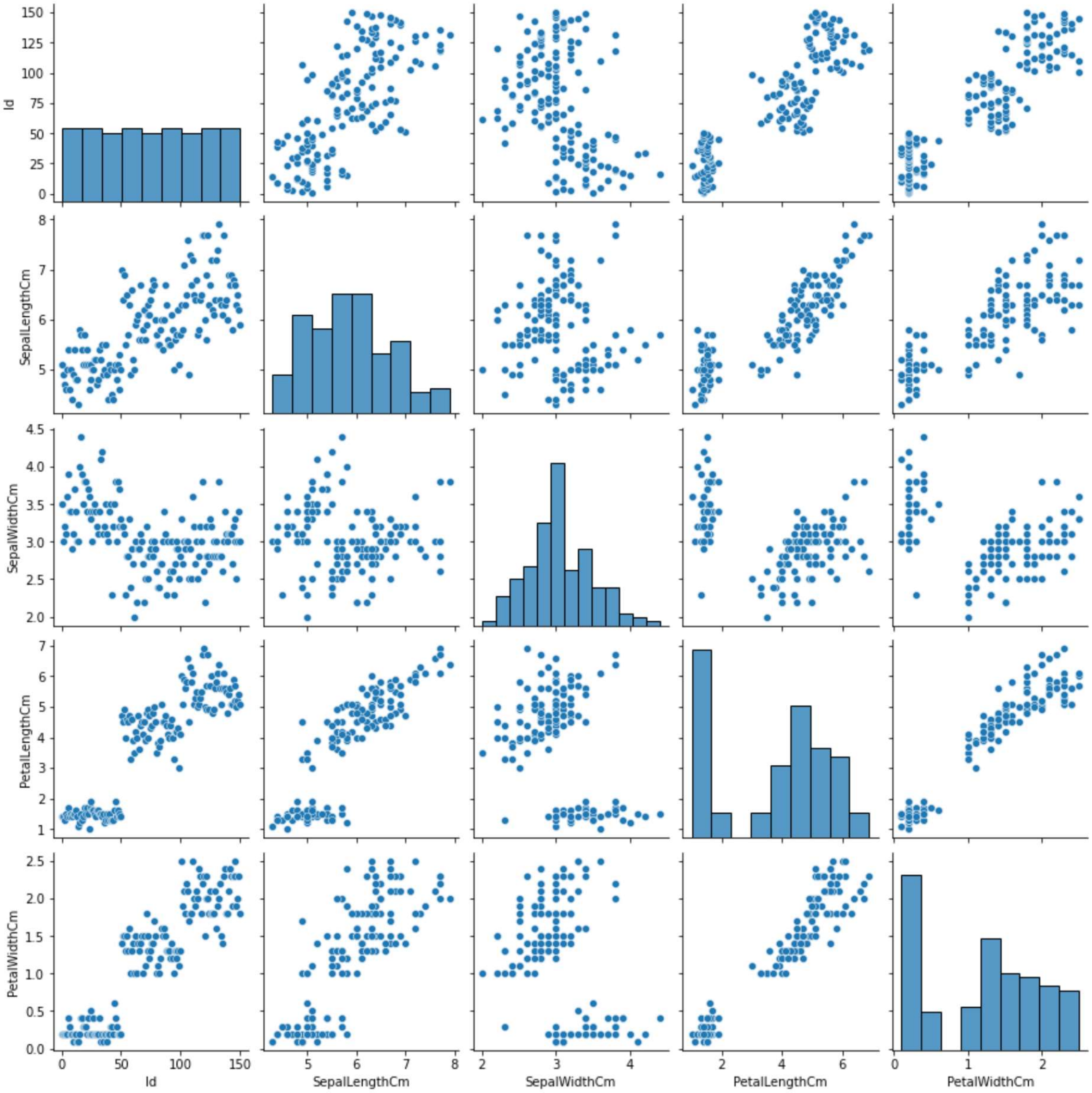
In [55]:
```
#to display columns name
data.columns
```

Out[55]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthC
m',
       'Species'],
      dtype='object')
```

In [56]:
```
data1=data[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidt
       'Species']]
```

```
In [57]: sns.pairplot(data1)
```

```
Out[57]: <seaborn.axisgrid.PairGrid at 0x238878021c0>
```



EDA and Visualization

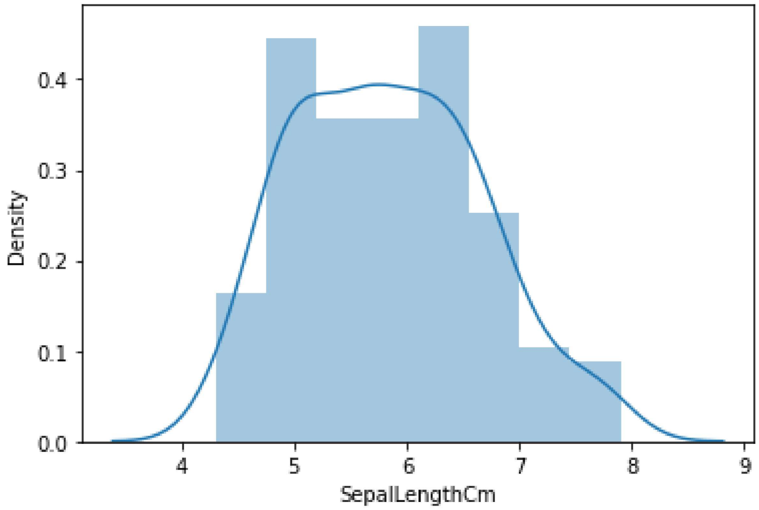
In [66]:

```
sns.distplot(data['SepalLengthCm'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[66]:

```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='Density'>
```

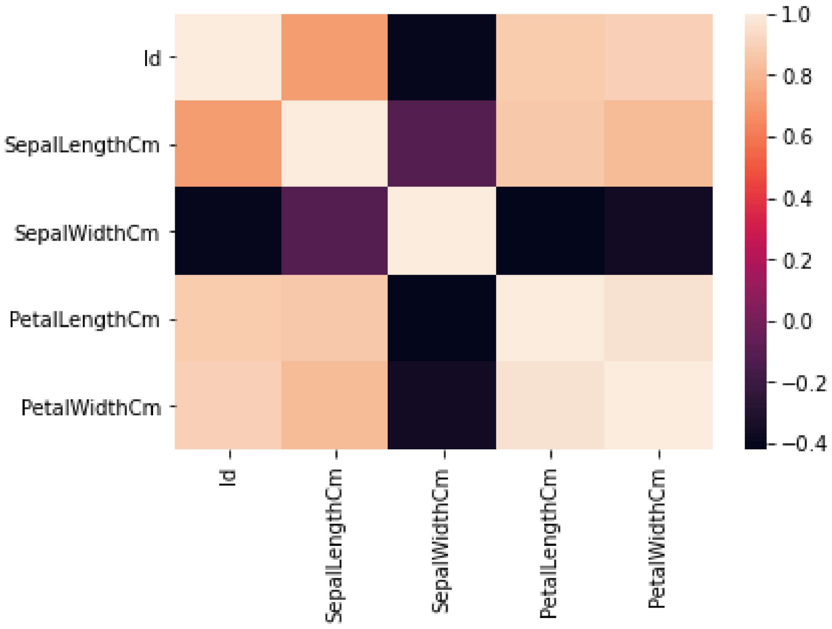


In [67]:

```
sns.heatmap(data1.corr())
```

Out[67]:

```
<AxesSubplot:>
```



To train the model

we are going to train the linear regression model ;We need to split the two variable x and y

In [74]:
```
x=data1[[ "PetalWidthCm","PetalLengthCm"]]
y=data1["SepalLengthCm"]
```

In [75]:
```

#To split test and train data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.6)
```

In [76]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[76]:
```
LinearRegression()
```

In [77]:
```
lr.intercept_
```

Out[77]:
```
3.791797830974231
```

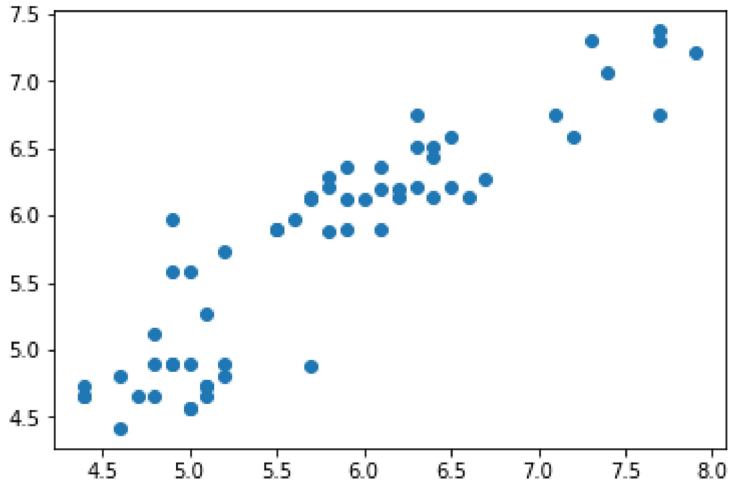
In [78]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[78]:

	Co-efficient
PetalWidthCm	-0.795634
PetalLengthCm	0.784375

In [79]:
```
prediction = lr.predict(x_train)
plt.scatter(y_train,prediction)
```

Out[79]:
```
<matplotlib.collections.PathCollection at 0x2388eb20460>
```



In [80]:
```
lr.score(x_test,y_test)
```

Out[80]:
```
0.6575905172987584
```

```
In [81]: lr.score(x_train,y_train)

Out[81]: 0.818119145498935

In [82]: from sklearn.linear_model import Ridge,Lasso

In [83]: rr=Ridge(alpha=10)
         rr.fit(x_train,y_train)
         rr.score(x_test,y_test)

Out[83]: 0.7157001934591718

In [84]: la=Lasso(alpha=10)
         la.fit(x_train,y_train)
         la.score(x_test,y_test)

Out[84]: -0.017597028057483755

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```