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 [39]:
```
#import the dataset
data=pd.read_csv(r"C:\Users\user\Desktop\Vicky\14_Iris.csv")[0:500]
```

In [40]:
```
#to display top 10 rows
data.head()
```

Out[40]:

	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 [41]:
```
#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 [42]:
```
data.shape
```

Out[42]:
```
(150, 6)
```

In [43]:
```
#to display summary of statistics
data.describe()
```

Out[43]:

	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 [44]:
```
#to display columns name
data.columns
```

Out[44]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthC
m',
       'Species'],
      dtype='object')
```

In [45]:
```
data.fillna(value=5)
```

Out[45]:

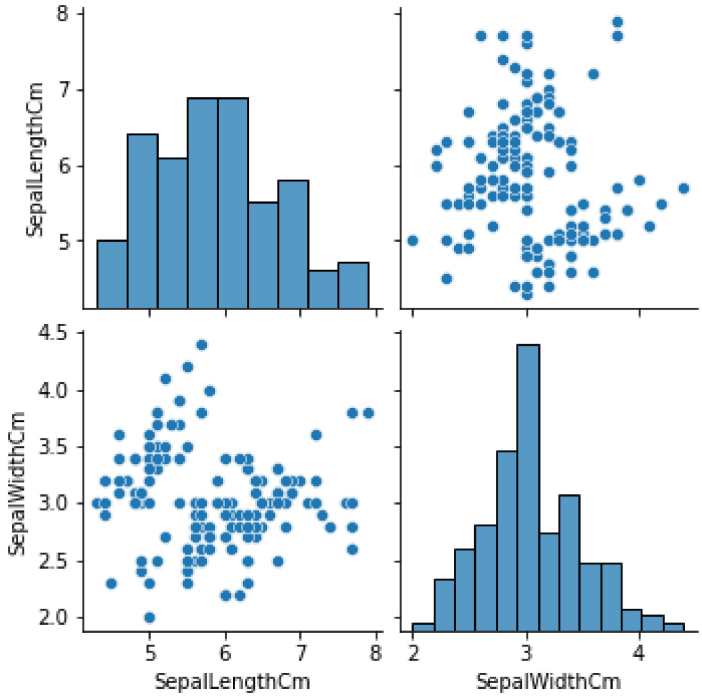
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	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 [51]:
```
data1=data[['SepalLengthCm', 'SepalWidthCm']]
```

In [52]:
```
sns.pairplot(data1)
```

Out[52]:
```
<seaborn.axisgrid.PairGrid at 0x12561175ee0>
```

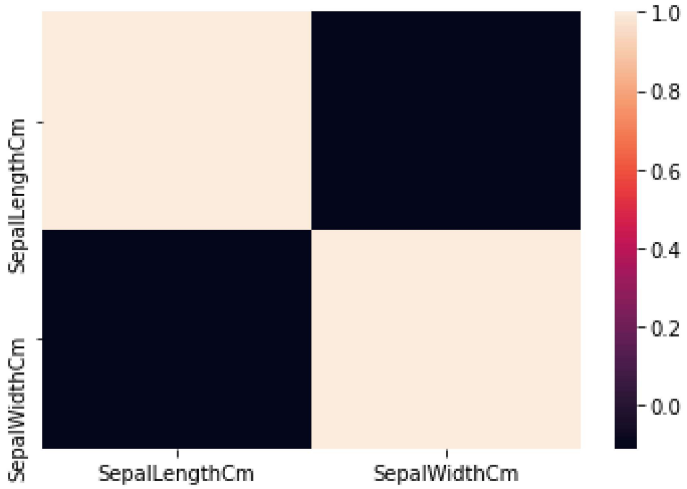


EDA and Visualization

In [55]:
```
#sns.distplot(data['Co2-Emissions'])
```

In [56]:
```
sns.heatmap(data1.corr())
```

Out[56]:
```
<AxesSubplot:>
```



To train the model

we are going to train the linear regression model ;We need to split the two variable x and y where x in independent variable (input) and y is dependent of x(output) so we could ignore address columns as it is not requires for our model

In [58]:
```
x=data[['PetalLengthCm', 'PetalWidthCm']]
y=data1['SepalLengthCm']
```

In [59]:
```
#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=0.1)
```

In [60]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[60]:
```
LinearRegression()
```

In [61]:
```
lr.intercept_
```

Out[61]:
```
4.1739181148958115
```

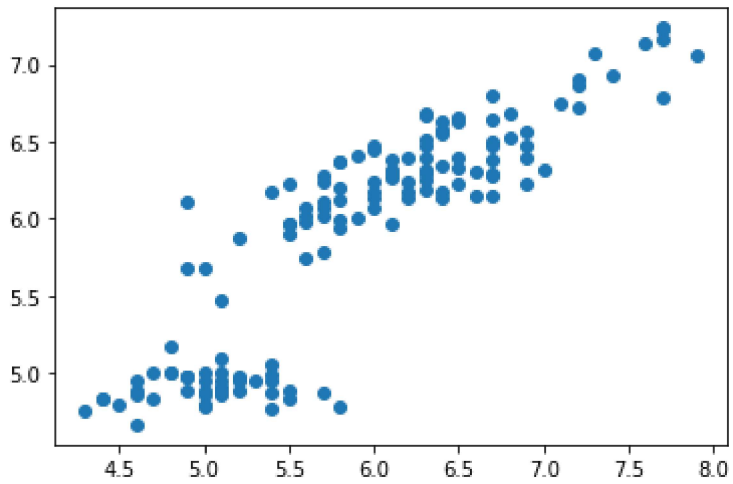
In [62]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[62]:

	Co-efficient
PetalLengthCm	0.559501
PetalWidthCm	-0.346296

In [63]:
```
prediction = lr.predict(x_train)
plt.scatter(y_train,prediction)
```

Out[63]:
```
<matplotlib.collections.PathCollection at 0x1257cc355b0>
```



In [64]:
```
lr.score(x_test,y_test)
```

Out[64]:
```
0.49701843337807705
```

In [65]:
```
lr.score(x_train,y_train)
```

Out[65]:
```
0.7765060366147293
```

In [66]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [67]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[67]:
```
0.5147809853652496
```

In [68]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_test,y_test)
```

Out[68]:
```
-0.4747641768787396
```

In [69]:
```
from sklearn.linear_model import ElasticNet
en= ElasticNet()
en.fit(x_train,y_train)
```

Out[69]:
```
ElasticNet()
```

In [70]:
```
print(en.coef_)
```

```
[0.21790278 0.        ]
```

In [71]:
```
print(en.intercept_)
```

```
5.054534570816629
```

In [72]:
```
prediction = en.predict(x_test)
prediction
```

Out[72]:
```
array([6.1004679 , 5.35959846, 5.35959846, 6.1004679 , 6.20941929,
       5.35959846, 5.88256512, 5.35959846, 6.27479012, 5.40317901,
       5.81719429, 5.42496929, 6.12225818, 6.03509707, 5.42496929])
```

In [73]:
```
print(en.score(x_test,y_train))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-73-48d1f0543252> in <module>
----> 1 print(en.score(x_test,y_train))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in score(self, X,
y, sample_weight)
    552         from .metrics import r2_score
    553         y_pred = self.predict(X)
--> 554         return r2_score(y, y_pred, sample_weight=sample_weight)
    555
    556     def _more_tags(self):

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inn
er_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py in
r2_score(y_true, y_pred, sample_weight, multioutput)
    674     -3.0
    675     """
--> 676     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    677         y_true, y_pred, multioutput)
    678     check_consistent_length(y_true, y_pred, sample_weight)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py in
_check_reg_targets(y_true, y_pred, multioutput, dtype)
     86         the dtype argument passed to check_array.
     87     """
---> 88     check_consistent_length(y_true, y_pred)
     89     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
     90     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in che
ck_consistent_length(*arrays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent num
bers of"
    263                          " samples: %r" % [int(l) for l in lengths])
    264

ValueError: Found input variables with inconsistent numbers of samples: [135,
15]
```

In [74]:
```
from sklearn import metrics
```

In [75]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error: 0.4576038894075581
```

In [76]:
```
print("Mean Absolute Square error:",metrics.mean_squared_error(y_test,predicti
```

```
Mean Absolute Square error: 0.259524809848262
```

In [77]:
```
print("Root mean Square error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
Root mean Square error: 0.5094357759799187
```

In [ ]:
```

```