# 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("Iris")
```

# To display top 10 rows

In [3]:
```
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]:
```
1 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]:
```
1 # Display the statistical summary
2 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]:
```
# To display the col headings
df.columns
```

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

In [7]:
```
cols=df.dropna(axis=1)
```

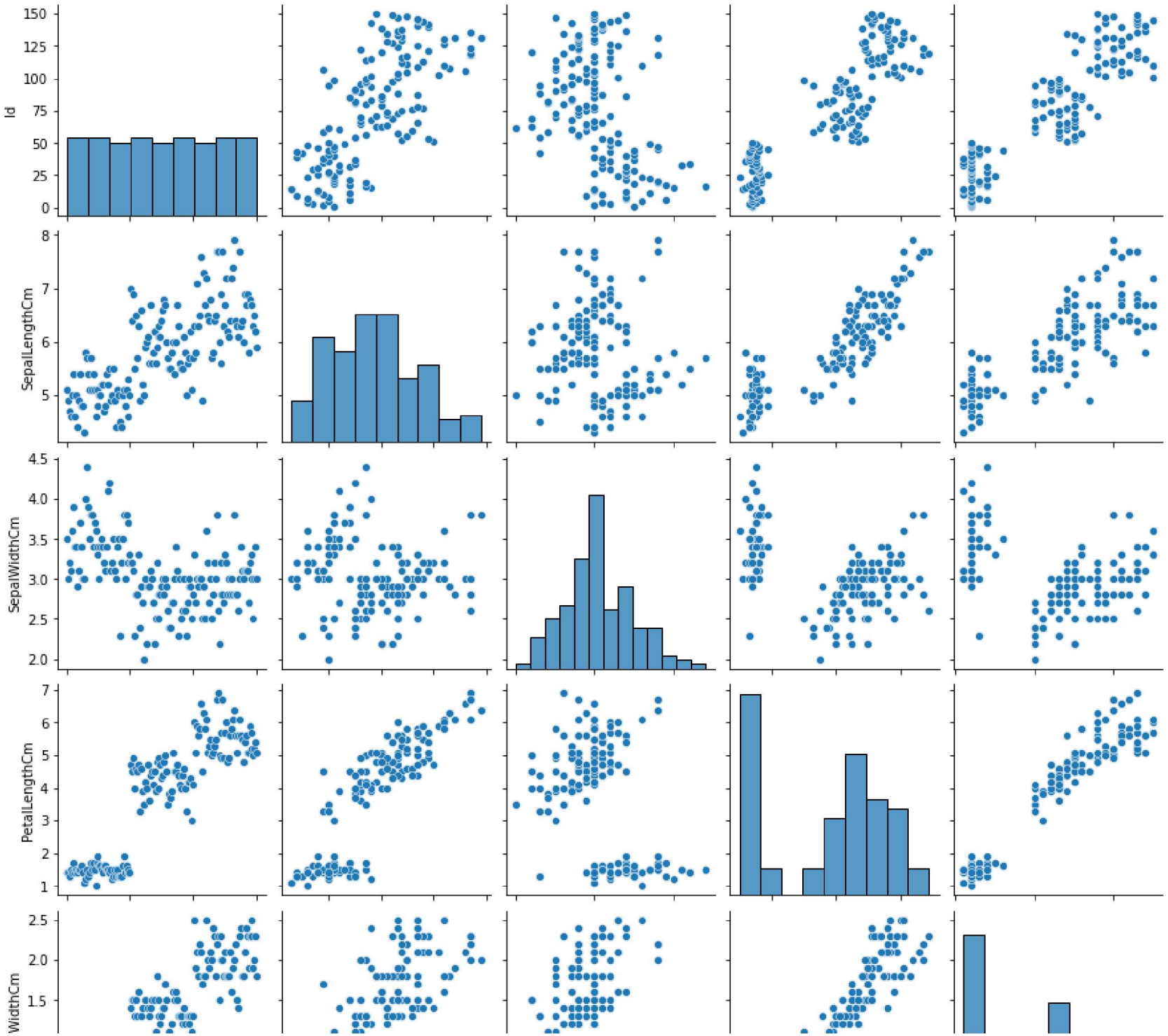
In [8]:
```
cols.columns
```

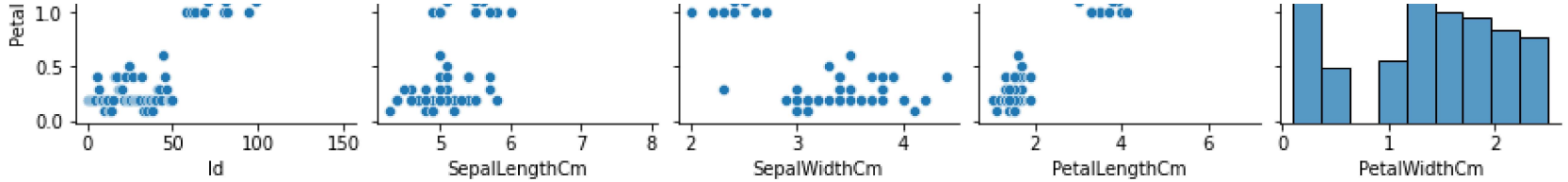
Out[8]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

# EDA and Visualization

In [9]:
```
1 sns.pairplot(cols)
```

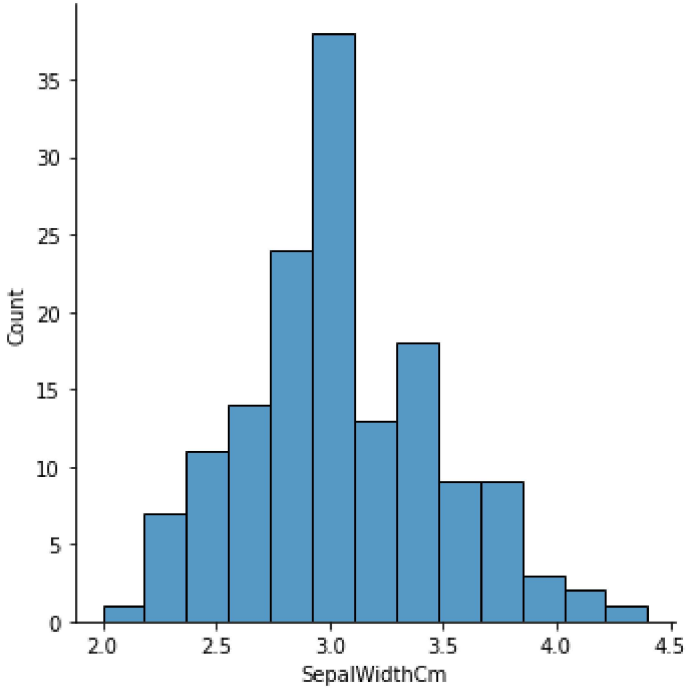
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x12b9e8f1760>
```


Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
WidthCm




In [11]:
```
1 sns.displot(df['SepalWidthCm'])
```

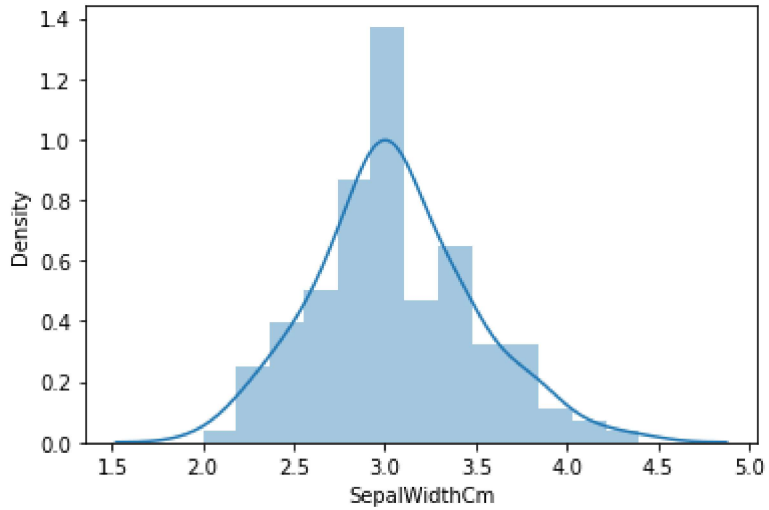
Out[11]: <seaborn.axisgrid.FacetGrid at 0x12b9e894f70>



In [12]:
```
# We use displot in older version we get distplot use displot
sns.distplot(df['SepalWidthCm'])
```

```
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[12]:
```
<AxesSubplot:xlabel='SepalWidthCm', ylabel='Density'>
```



In [13]:
```
df1=cols[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
df1
```

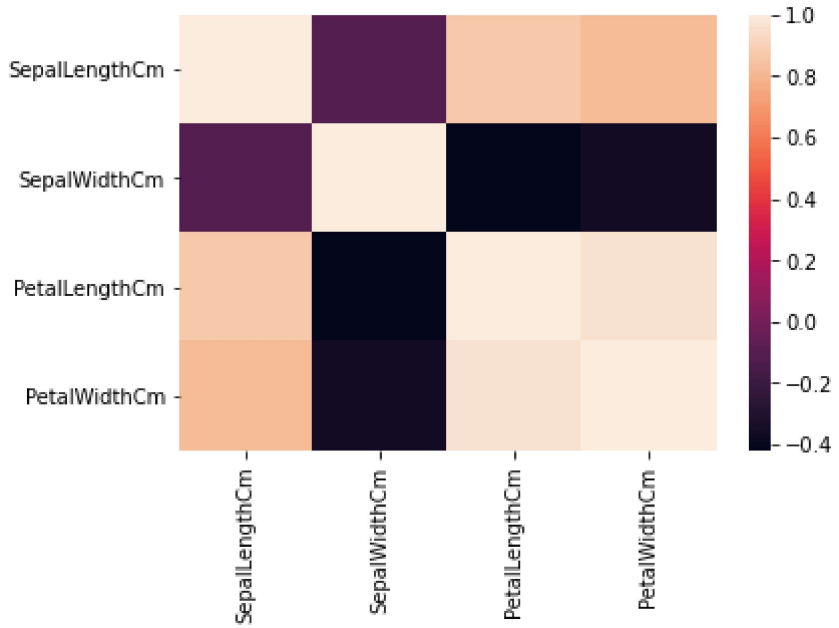
Out[13]:

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

150 rows × 4 columns

In [14]:
```
1 sns.heatmap(df1.corr())
```

Out[14]: <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 [15]:
```
1 x=df1[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
2 y=df1[['SepalWidthCm']]
```

# To split the dataset into test data

In [16]:
```
1 # importing lib for splitting test data
2 from sklearn.model_selection import train_test_split
```

In [17]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [18]:
```
from sklearn.linear_model import LinearRegression

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

Out[18]:
```
LinearRegression()
```

In [19]:
```
print(lr.intercept_)
```

```
[8.8817842e-16]
```

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

```
1.0
```

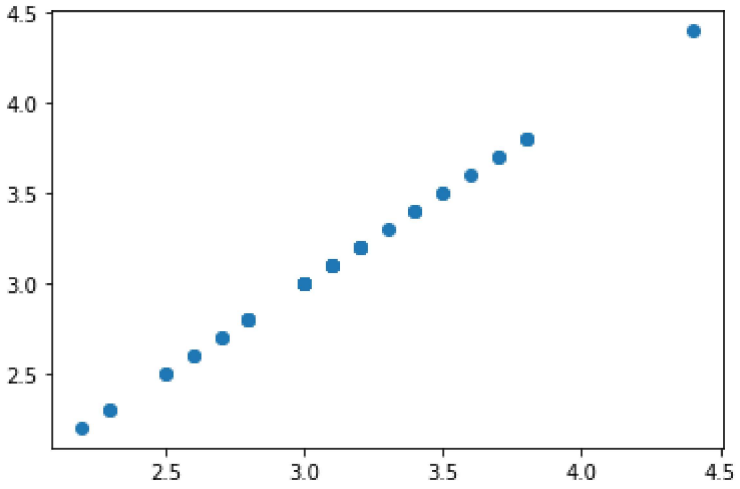
In [21]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[21]:

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **0** | -1.681678e-16 | 1.0 | -7.374996e-17 | -4.346048e-16 |

In [22]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[22]: <matplotlib.collections.PathCollection at 0x12ba1c03880>




In [23]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [24]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[24]: Ridge(alpha=10)

In [25]:
```
rr.score(x_test,y_test)
```

Out[25]: 0.8765619212163441

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]: -0.021883618312189768

In [ ]:
```
1
```