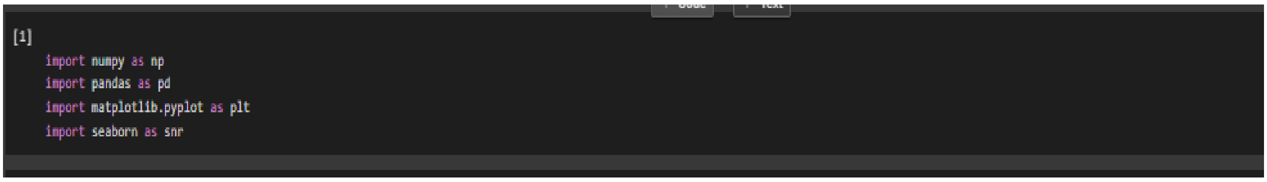
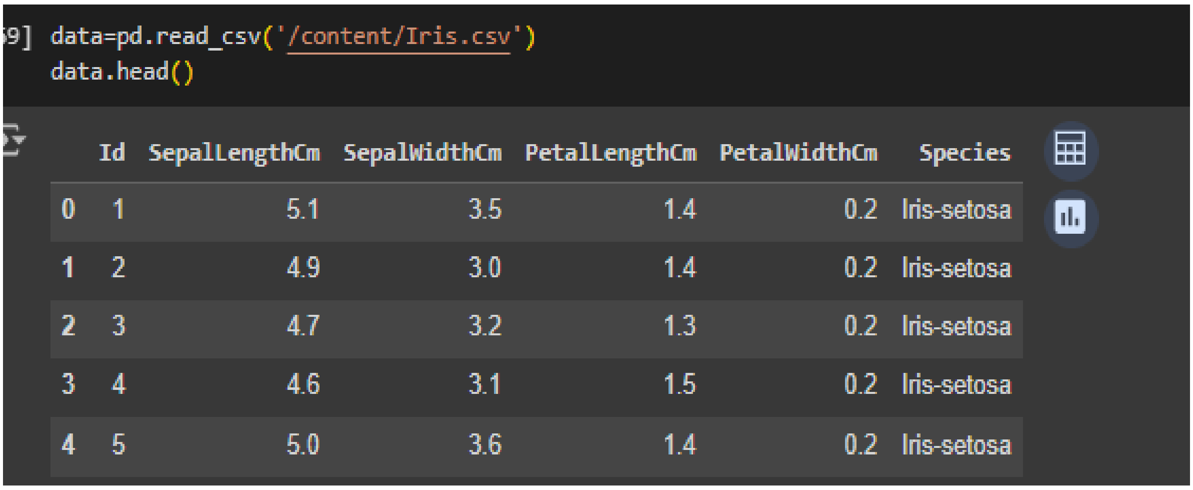
Project on Iris Flower Classification Prediction using Machine Learning

- **Aim:-**To create a Data science Project, Iris flower has three species; setosa, versicolor, and virginica, which differs according to their measurements. Now assume that you have the measurements of the iris flowers according to their species, and here task is to train a machine learning model that can learn from the measurements of the iris species and classify them.
- Steps to be taken in the project is sub-divided into the following sections. These are:
    - Importing the libraries such as 'numpy', 'pandas','sklearn. model' etc.
    - Loading Dataset as a CSV file for training & testing the models.
    - Splitting the data set into independent & dependent sets.
    - Checking if still any null values or any other data types other than float and integers are present into the dataset or not.
    - Importing the train_test_split model from sklearn.model for splitting data into train & test sets.
    - Applying the different kinds of ML Algorithms .which gives Best accuracy of model.
    - Also checking with new data set for predicting the values.
- Steps of creating ML model:-
- Importing numpy as np & pandas as pd for loading and reading the data-set & using matplotlib.pyplot and Seaborn for visualization of data.



```
[1]
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as snr
```

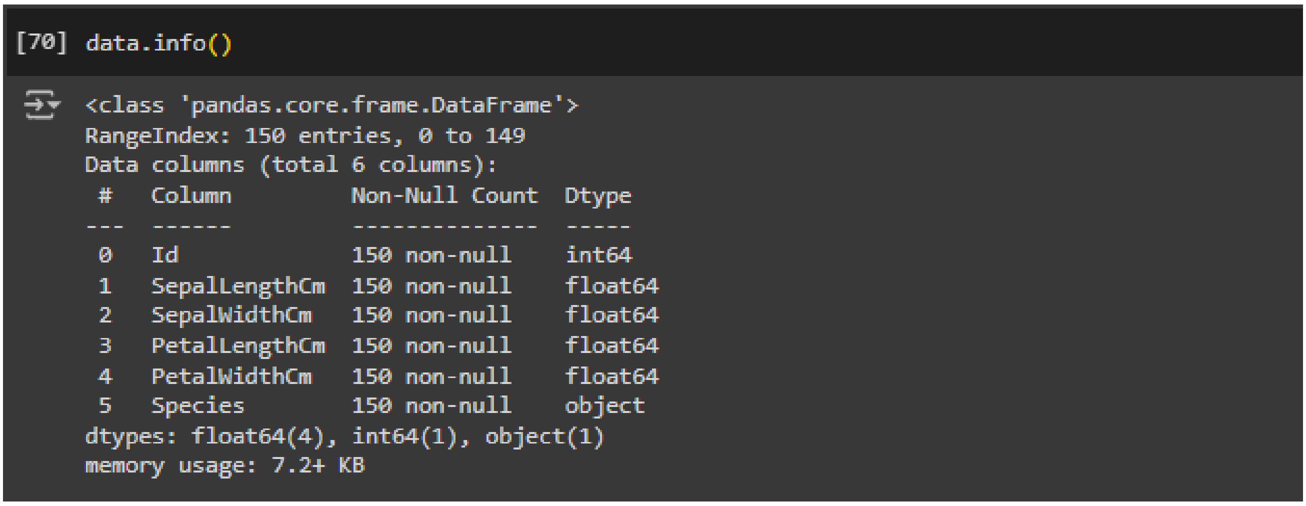
- Loading the csv-dataset in the variable name 'data' Then viewing the data with data.head()



```
9] data=pd.read_csv('/content/Iris.csv')
   data.head()
```

	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

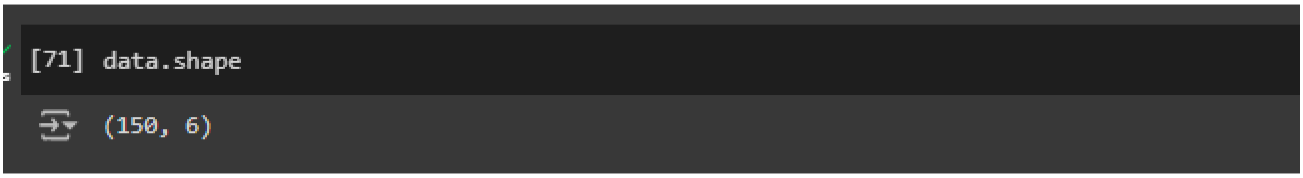
- ❖ Checking the data such as number of columns, rows and type of data(float,integer) with help of data.info()



```
[70] 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
```

We observe that the above data have integer, object and float.

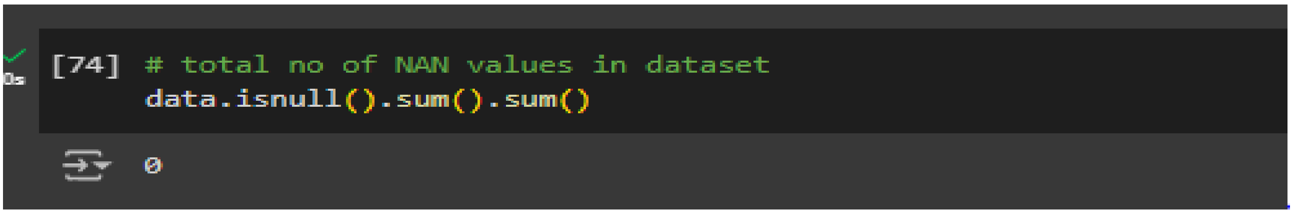


```
[71] data.shape

(150, 6)
```

Train data have 150 Rows and 6 columns

- ❖ Now checking data have Nan value or not.

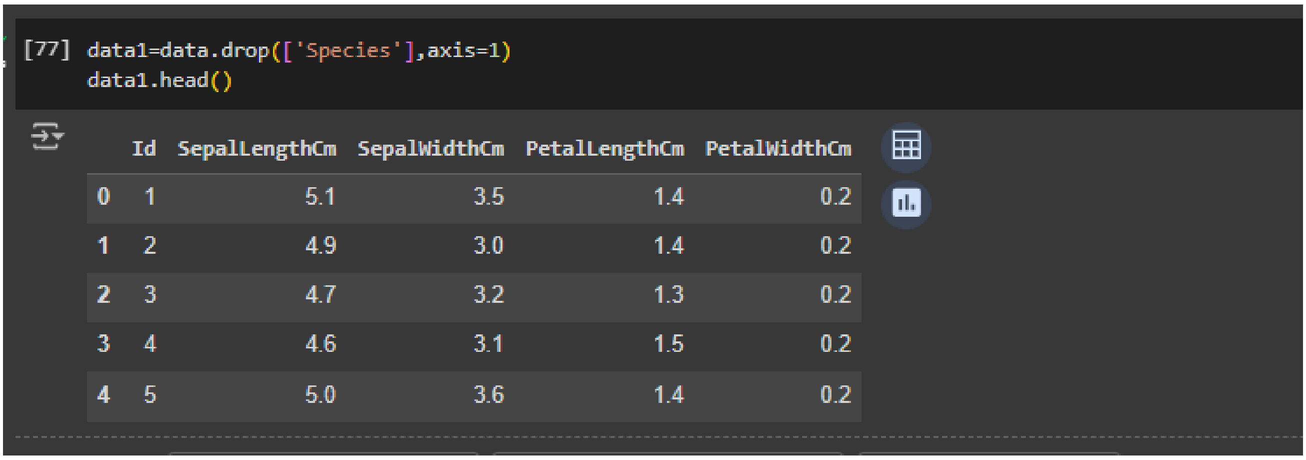


```
[74] # total no of NAN values in dataset
     data.isnull().sum().sum()

0
```

We observe that the above data have not Nan value.

- ❖ Now,Main focus convert the categorical data into Numerical data with help of one hot encoding method.

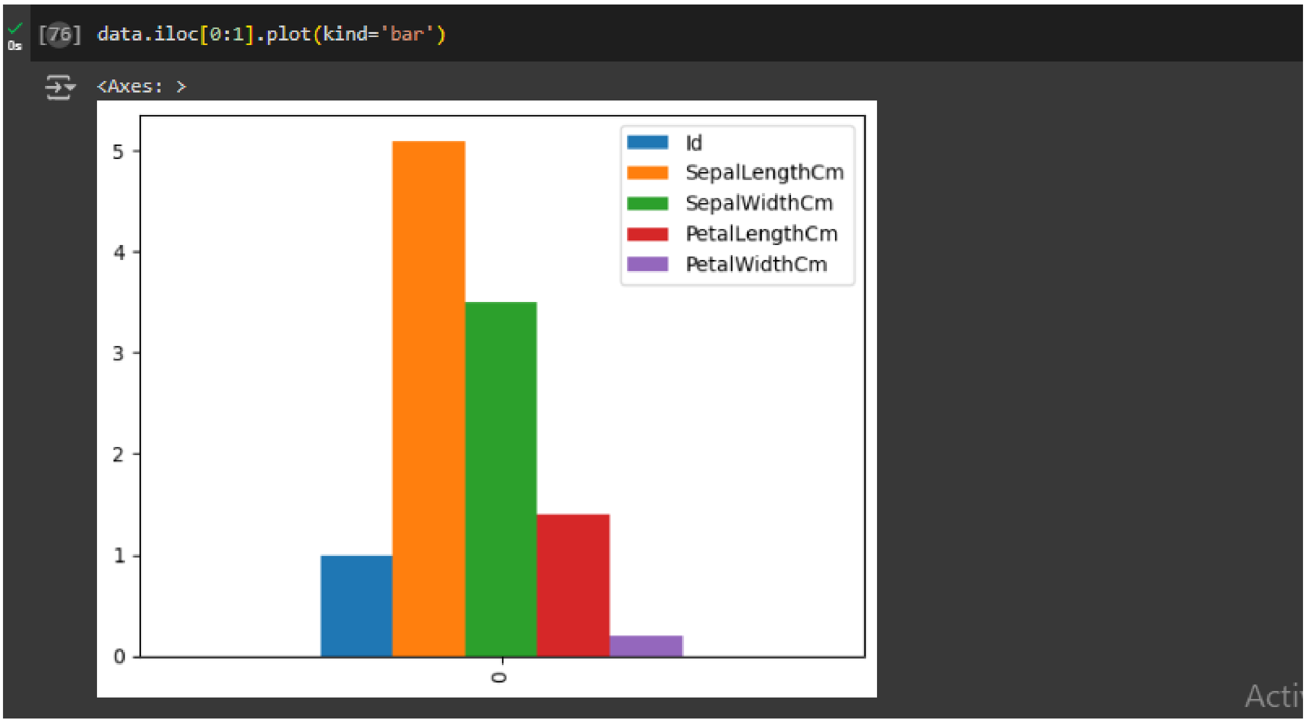


```
[77] data1=data.drop(['Species'],axis=1)
     data1.head()
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0   1            5.1           3.5            1.4           0.2
1   2            4.9           3.0            1.4           0.2
2   3            4.7           3.2            1.3           0.2
3   4            4.6           3.1            1.5           0.2
4   5            5.0           3.6            1.4           0.2
```

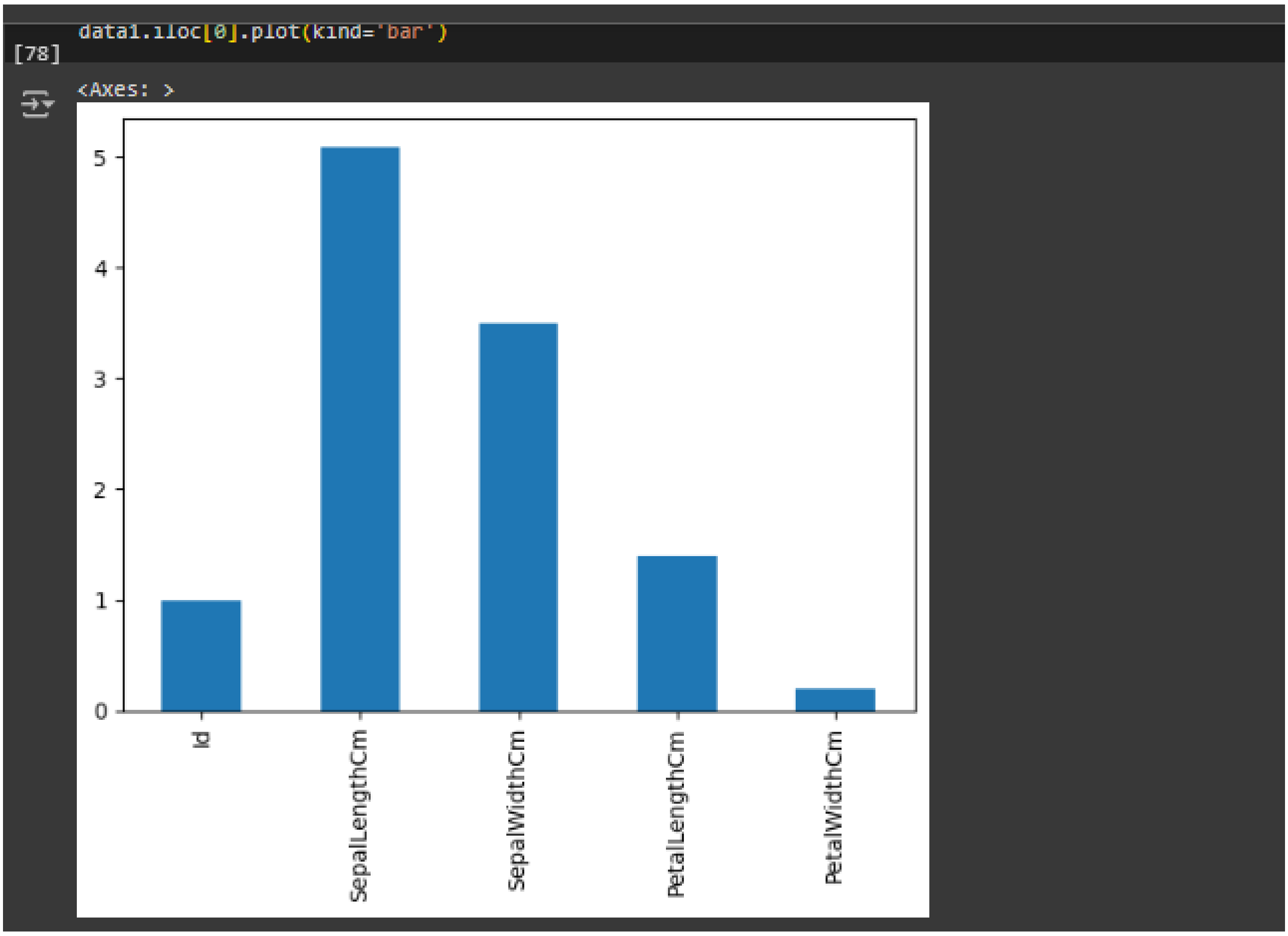
Finally we observe the data are fully cleaned.

❖ Visualizing the iris data set.



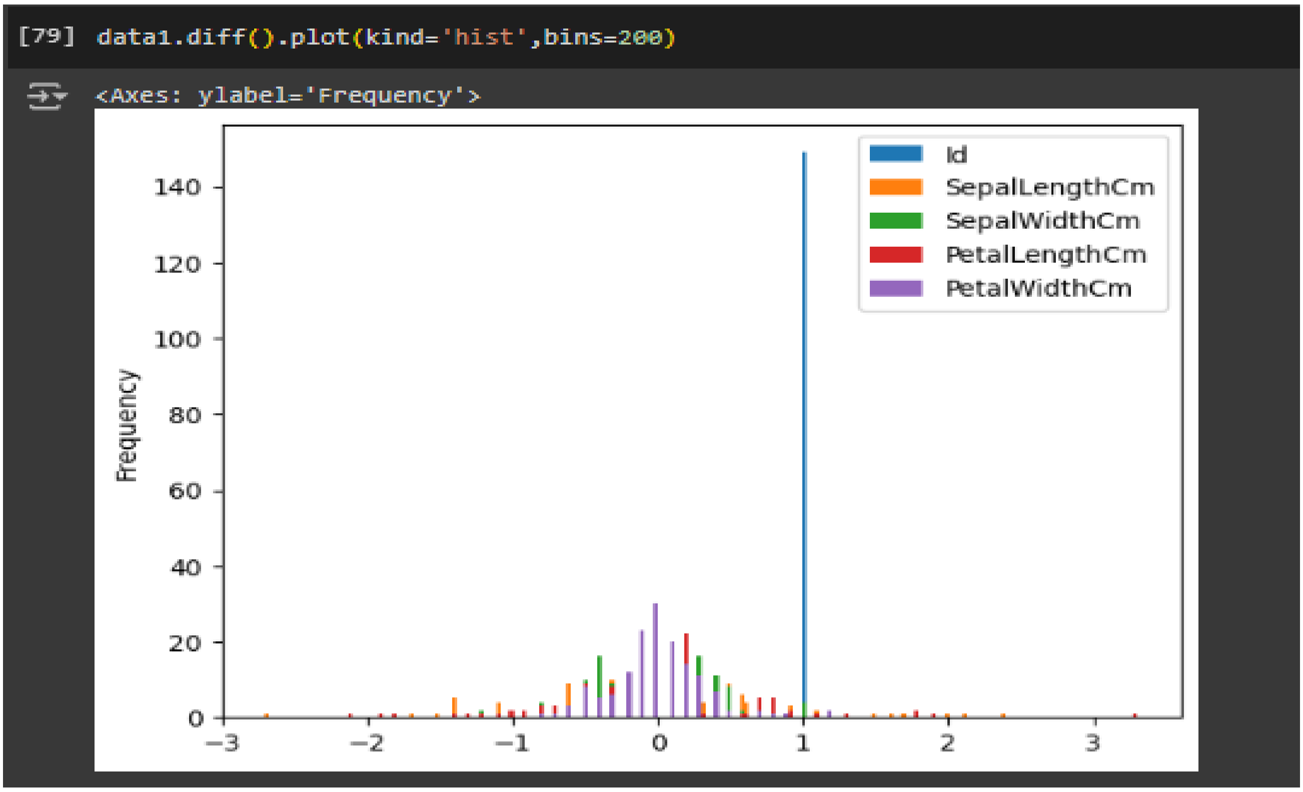
```
[76] data.iloc[0:1].plot(kind='bar')
```

```
<Axes: >
```



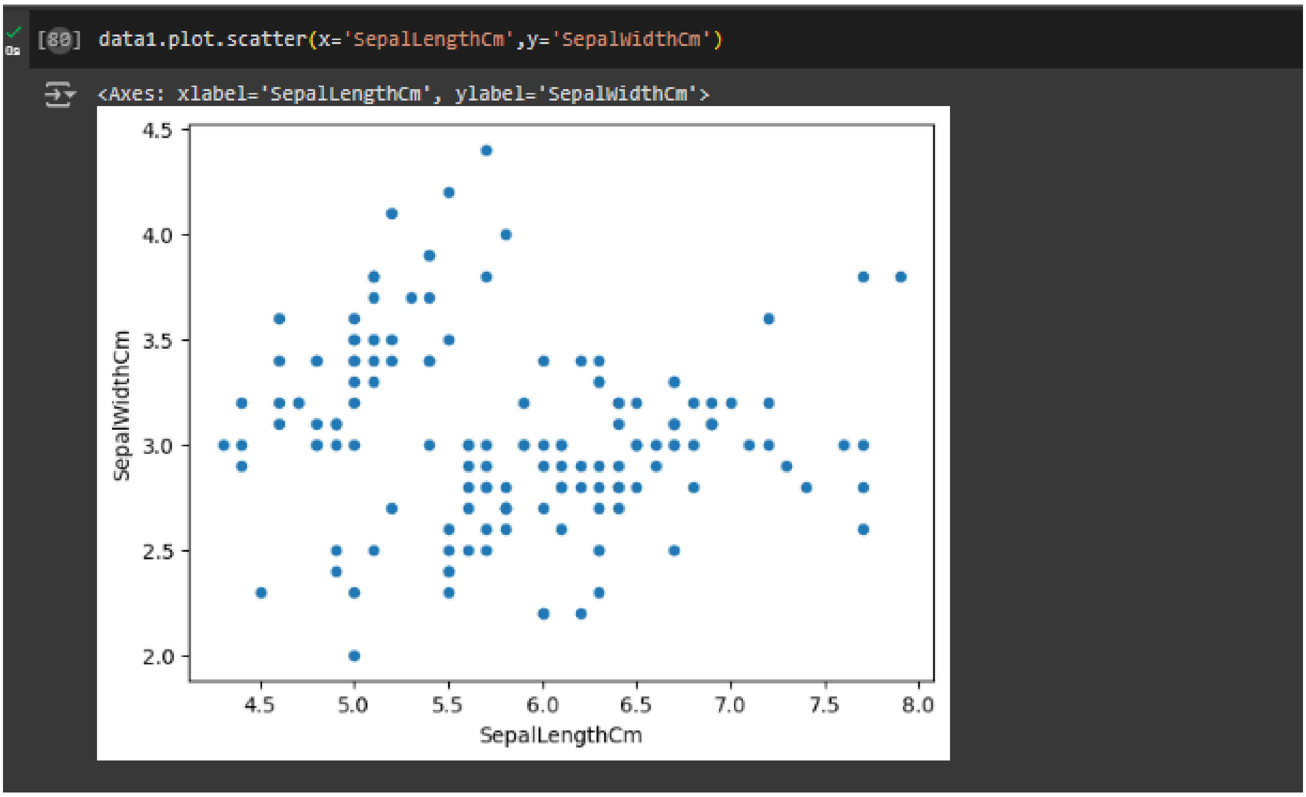
```
[78] data1.iloc[0].plot(kind='bar')
```

```
<Axes: >
```



```
[79] data1.diff().plot(kind='hist',bins=200)
```

```
<Axes: ylabel='Frequency'>
```

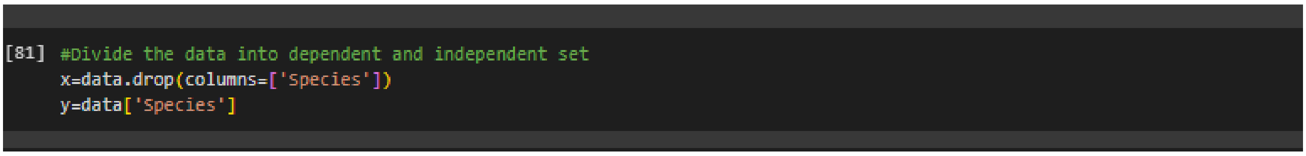


```
[80] data1.plot.scatter(x='SepalLengthCm',y='SepalWidthCm')

<Axes: xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```

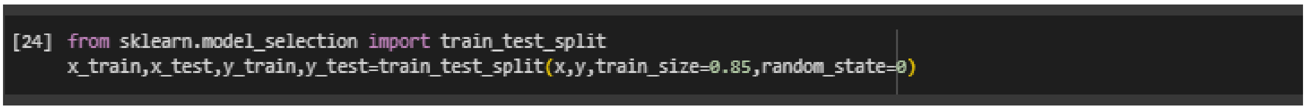
After visualization of data, we predict iris data using Machine Learning .

- Splitting the dataset into dependent(y) & independent(x) sets



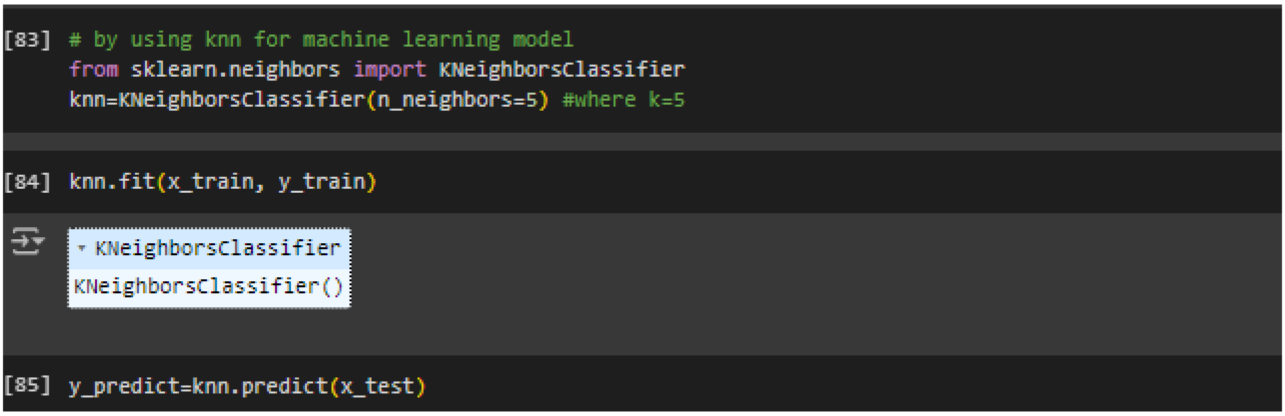
```
[81] #Divide the data into dependent and independent set
     x=data.drop(columns=['Species'])
     y=data['Species']
```

- Importing train_test_split from sklearn.model library for splitting the data into train and test sets.



```
[24] from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.85,random_state=0)
```

- Importing Knn from sklearn Libaray & then activating the Machine learning Model .Then used regression.fit() to training the model by providing train & test sets as x & y. And then predicted the trained model with help of MLM & the checked score as regression.score(x,y)



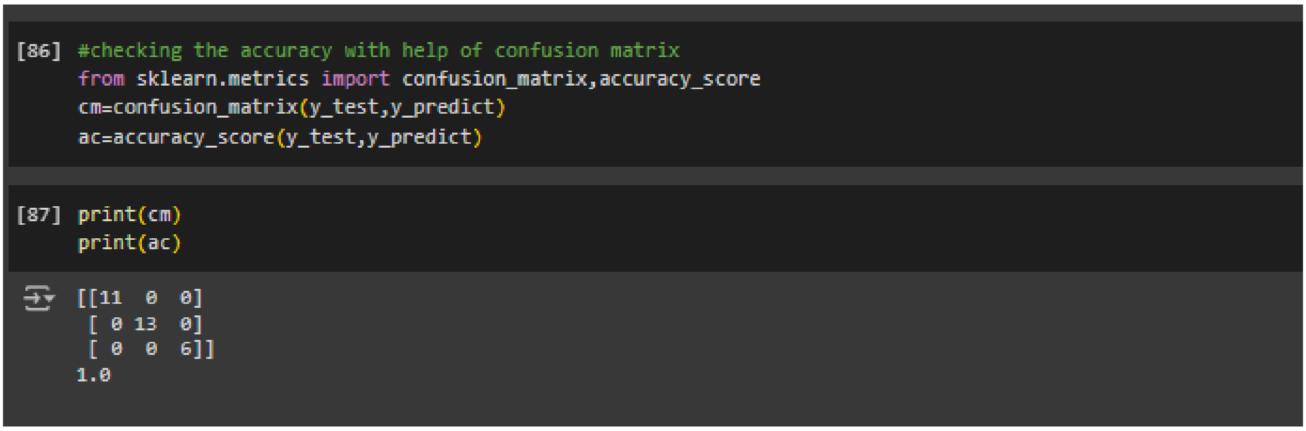
```
[83] # by using knn for machine learning model
     from sklearn.neighbors import KNeighborsClassifier
     knn=KNeighborsClassifier(n_neighbors=5) #where k=5

[84] knn.fit(x_train, y_train)

     ▾ KNeighborsClassifier
     KNeighborsClassifier()

[85] y_predict=knn.predict(x_test)
```

- Checking the accuracy with help of confusion Matrix.



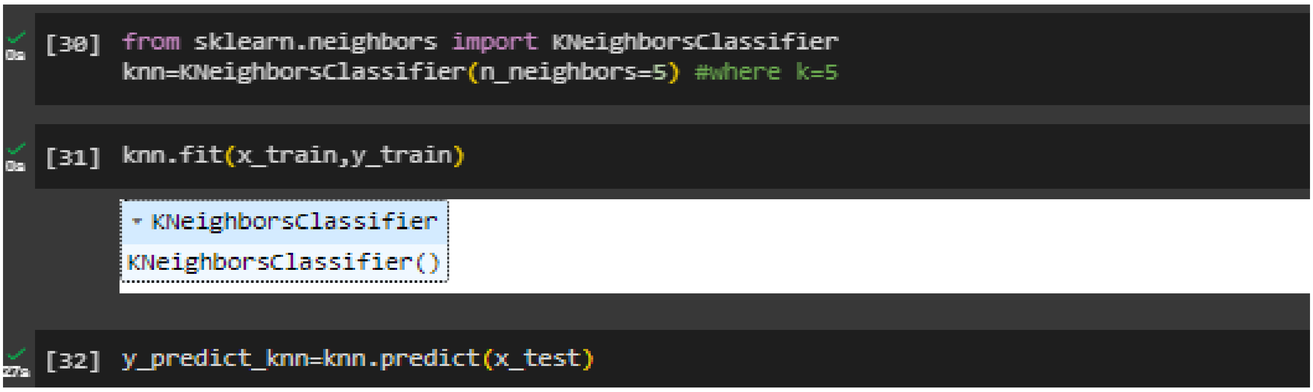
```
[86] #checking the accuracy with help of confusion matrix
     from sklearn.metrics import confusion_matrix,accuracy_score
     cm=confusion_matrix(y_test,y_predict)
     ac=accuracy_score(y_test,y_predict)

[87] print(cm)
     print(ac)

     [[11  0  0]
      [ 0 13  0]
      [ 0  0  6]]
     1.0
```

In the above model we can see that the accuracy obtained is 100%

- Now applying new algorithm Decision tree, then checked score.

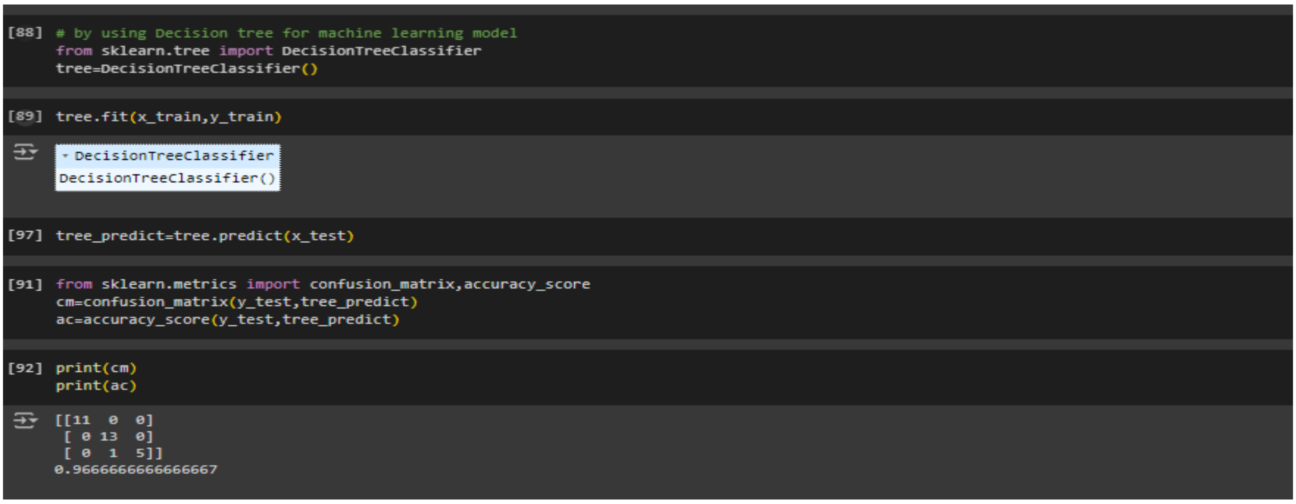


```
[30] from sklearn.neighbors import KNeighborsClassifier
     knn=KNeighborsClassifier(n_neighbors=5) #where k=5

[31] knn.fit(x_train,y_train)

     ▾ KNeighborsClassifier
     KNeighborsClassifier()

[32] y_predict_knn=knn.predict(x_test)
```



```
[88] # by using Decision tree for machine learning model
     from sklearn.tree import DecisionTreeClassifier
     tree=DecisionTreeClassifier()

[89] tree.fit(x_train,y_train)

     ▾ DecisionTreeClassifier
     DecisionTreeClassifier()

[97] tree_predict=tree.predict(x_test)

[91] from sklearn.metrics import confusion_matrix,accuracy_score
     cm=confusion_matrix(y_test,tree_predict)
     ac=accuracy_score(y_test,tree_predict)

[92] print(cm)
     print(ac)

     [[11  0  0]
      [ 0 13  0]
      [ 0  1  5]]
     0.9666666666666667
```

we can see that the accuracy obtained is 96%

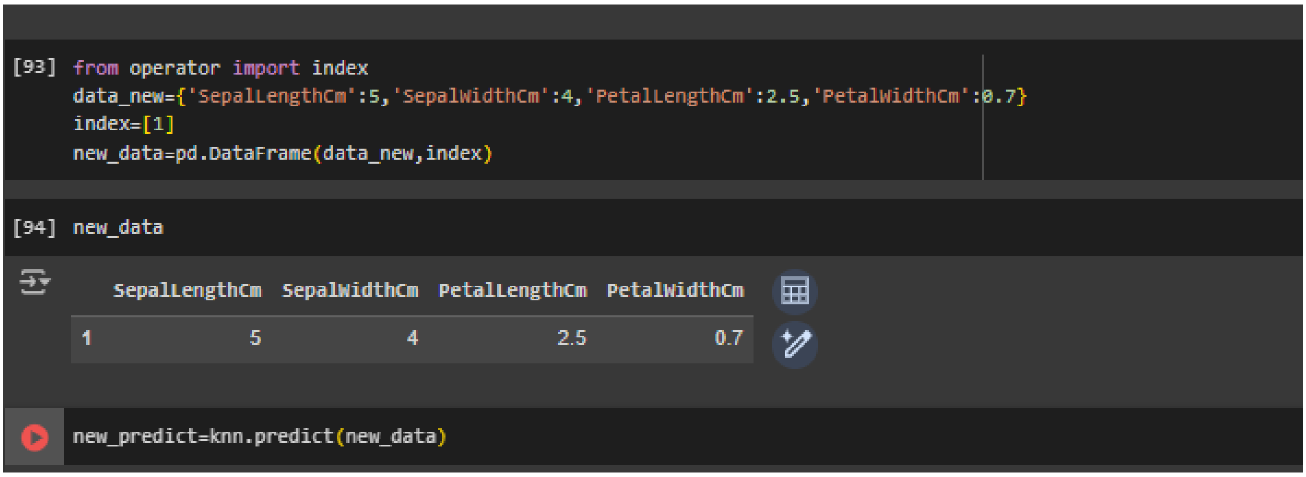
We see the accuracy is good but less than Decision Tree and Random forest algorithms.

- Now we compare all algorithms with accuracy

Algorithms	accuracy
KNN	100%
Decision Tree classifier	96%

KNN machine algorithms is better than Decision Tree Classifier

- Now applying for new data set.



```
[93] from operator import index
     data_new={'SepalLengthCm':5,'SepalWidthCm':4,'PetalLengthCm':2.5,'PetalWidthCm':0.7}
     index=[1]
     new_data=pd.DataFrame(data_new,index)

[94] new_data

        SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
     1              5             4            2.5           0.7

     new_predict=knn.predict(new_data)
```

Conclusion:- In this test data set we analysed the data we found the best result of new iris data.

Thank you