Iris Flower Classification with Machine Learning

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

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

First five Values

```
print (iris.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
```

Last Five Values

```
print (iris.tail())
```

```
       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica
```

Statistics

```
print(iris.describe())
```

```
                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
```

```
print("Target Labels", iris["Species"].unique())
```

```
Target Labels ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

Separate features (X) and target variable (y)

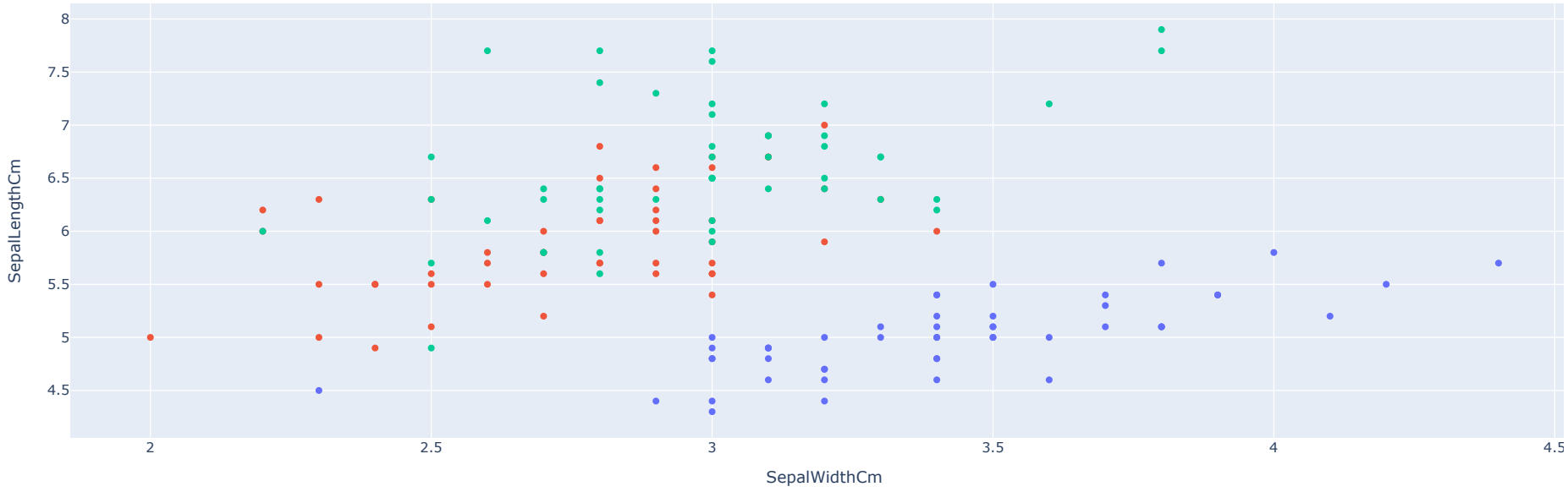
```
X = data[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = data['Species']
```

Split the data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Scatter Plot

```
import plotly.express as px
fig = px.scatter(iris, x="SepalWidthCm", y="SepalLengthCm", color="Species")
fig.show()
```



KNN Clasification Aligorithm

```
x = iris.drop("Species", axis=1)
y = iris["Species"]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.2,
                                                    random_state=0)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(x_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

```
x_new = np.array([[5, 2.9, 1, 0.2,6]])
prediction = knn.predict(x_new)
print("Prediction: {}".format(prediction))
```

```
Prediction: ['Iris-setosa']
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:465: UserWarning:

X does not have valid feature names, but KNeighborsClassifier was fitted with feature names
```

Train the model

```
model.fit(X_train, y_train)
```

```
LogisticRegression()
```

Make predictions on the test set

```
y_pred = model.predict(X_test)
```

Evaluate the model

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.3
```

Decision Tree model

```
from sklearn.tree import DecisionTreeClassifier

dt_model = DecisionTreeClassifier()
```

```
Decision Tree Accuracy: 0.3333333333333333
```

Train the model

```
dt_model.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

Make predictions on the test set

```
dt_y_pred = dt_model.predict(X_test)
```

Evaluate the model

```
dt_accuracy = accuracy_score(y_test, dt_y_pred)
print("Decision Tree Accuracy:", dt_accuracy)
```

```
Decision Tree Accuracy: 0.3333333333333333
```

Double-click (or enter) to edit

Support Vector Machine model

```
from sklearn.svm import SVC

svm_model = SVC()

svm_model.fit(X_train, y_train)

svm_y_pred = svm_model.predict(X_test)

svm_accuracy = accuracy_score(y_test, svm_y_pred)

print("SVM Accuracy:", svm_accuracy)
```

```
SVM Accuracy: 0.3333333333333333
```

Random Forest model

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier()

rf_model.fit(X_train, y_train)

rf_y_pred = rf_model.predict(X_test)

rf_accuracy = accuracy_score(y_test, rf_y_pred)

print("Random Forest Accuracy:", rf_accuracy)
```

```
Random Forest Accuracy: 0.3
```

Naive Bayes model

```
from sklearn.naive_bayes import GaussianNB
```

```
from sklearn.naive_bayes import GaussianNB

nb_model = GaussianNB()

nb_model.fit(X_train, y_train)

nb_y_pred = nb_model.predict(X_test)

nb_accuracy = accuracy_score(y_test, nb_y_pred)

print("Naive Bayes Accuracy:", nb_accuracy)
```