

Iris Classifier Project.

The Iris Classifier project is a classic example of a supervised machine learning problem that involves predicting the species of an iris flower based on its petal length, petal width, sepal length, and sepal width. The iris dataset contains 150 samples, each with four input features and one target variable.

The goal of this project is to create a machine learning model that can accurately predict the species of an iris flower given its measurements. The model is trained on a subset of the dataset, and its performance is evaluated on a held-out test set.

Once the model is trained, it can be used to predict the species of an iris flower for new data. This prediction can be useful for botanists, gardeners, and plant enthusiasts who need to identify the species of an iris flower based on its measurements.

In conclusion we can say that the Iris Classifier project is a valuable tool for predicting the species of an iris flower based on its measurements. It demonstrates how machine learning can be used to make accurate predictions based on input features, which can be useful in a wide range of applications in fields such as biology and agriculture.

- A model which of Iris classifier project by Get the data from local system not from web

```
import pandas as pd

from sklearn.model_selection import train_test_split

iris_data = pd.read_csv(r' C:\Users\ADMIN\Downloads\iris.csv')

print(iris_data)

X,y =iris_data(return_X_y=True)

Print(X)

Print(y[128])

Print(type(X))

Print(len(y))

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

```

Print(len(y_train))

Print(len(y_test))

from sklearn.tree import DecisionTreeClassifier

d = DecisionTreeClassifier()

d.fit(X_train, y_train)

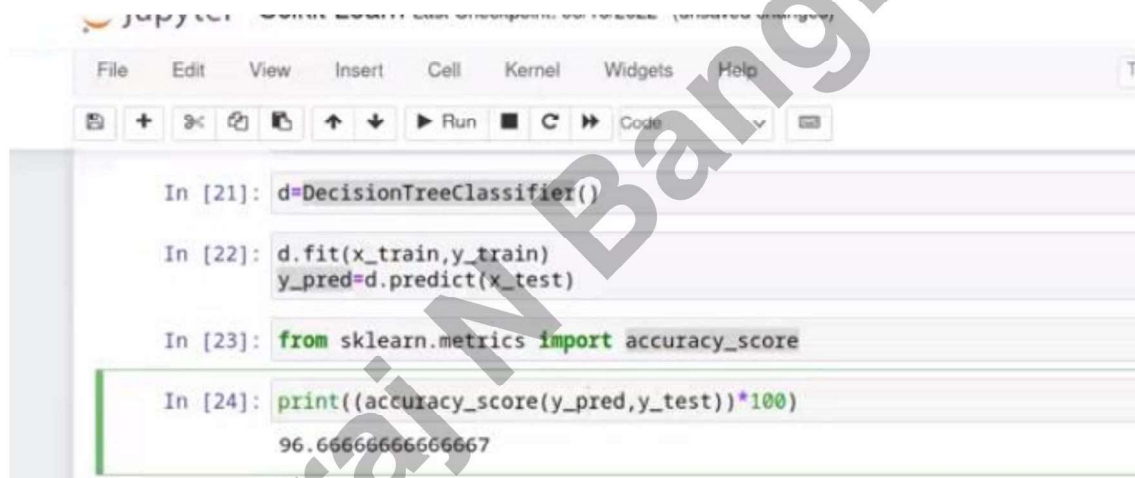
y_pred = d.predict(X_test)

from sklearn.metrics import accuracy_score

print ((accuracy_score(y_test, y_pred))*100)

```

OUTPUT:



```

In [21]: d=DecisionTreeClassifier()

In [22]: d.fit(x_train,y_train)
          y_pred=d.predict(x_test)

In [23]: from sklearn.metrics import accuracy_score

In [24]: print((accuracy_score(y_pred,y_test))*100)
          96.66666666666667

```

Accuracy of **96.667%** is obtained by using above **Decision Tree Classifier** Algorithm

- Trying to evaluate the performance of the model by changing **various parameters.**

```

from sklearn.tree import DecisionTreeClassifier

d = DecisionTreeClassifier(criterion="entropy",max_depth=3)

d.fit(X_train, y_train)

```

```

y_pred = d.predict(X_test)

from sklearn.metrics import accuracy_score

accuracy = ((accuracy_score(y_test, y_pred))*100)

print('Accuracy:', accuracy)

```

OUTPUT:

Accuracy of **97.334%** is obtained by using above **Decision Tree Classifier** Algorithm

- Using **Random forest classifier** algorithms and evaluate the performance of each algorithms in this dataset.

```

from sklearn.ensemble import DecisionTreeClassifier

clf = RandomForestClassifier(n_estimators=100)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

from sklearn import metrics

print("Accuracy:",metrics.accuracy_score(y_test,y_pred))

```

OUTPUT:

```

: #Import Random Forest Model
  from sklearn.ensemble import RandomForestClassifier

: #Create a Gaussian Classifier
  clf=RandomForestClassifier(n_estimators=100)

: #Train the model using the training sets y_pred=clf.predict(X_test)
  clf.fit(X_train,y_train)

: y_pred=clf.predict(X_test)

:

:

:

: #Import scikit-learn metrics module for accuracy calculation
  from sklearn import metrics
  # Model Accuracy, how often is the classifier correct?
  print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

: Accuracy: 0.9777777777777777

```

Accuracy of **97.777%** is obtained by using above **Random Forest Classifier** Algorithm