2019103555 PRANAVA RAMAN B M S 29/11/2021 CS6301

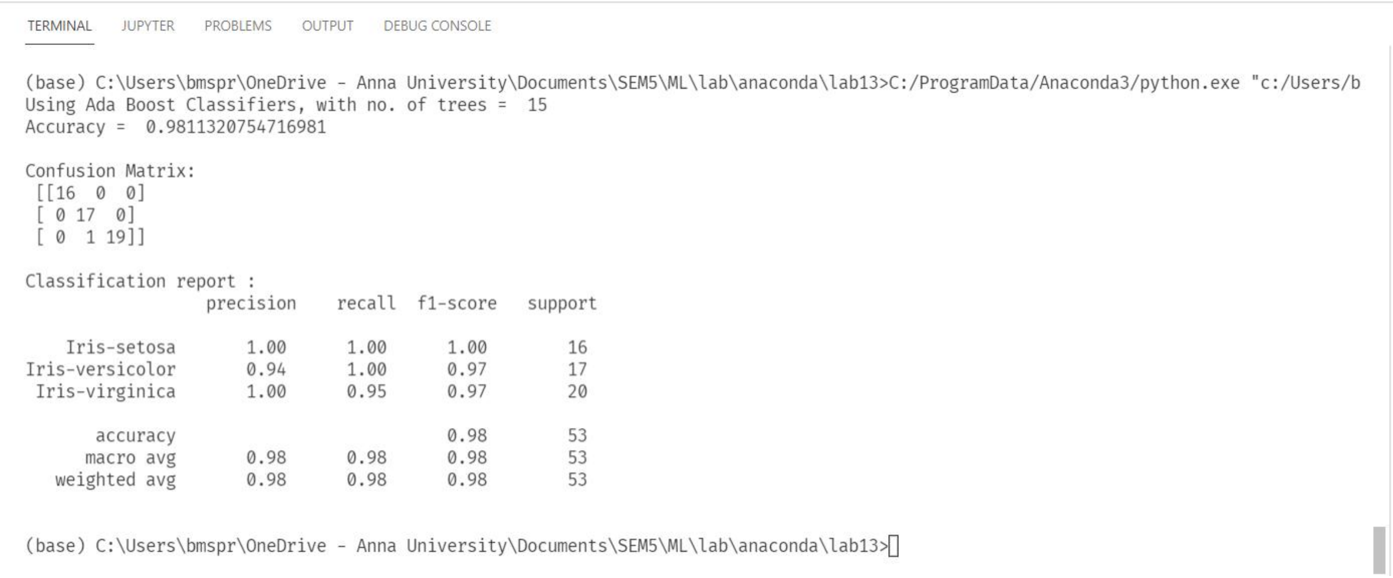
MACHINE LEARNING - LAB - 13 - EXECUTION - ENSEMBLE CLASSIFIERS:-

1. ADABOOST CLASSIFIER

CODE:-

```
import pandas as pd
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split

df = pd.read_csv("Iris.csv")
array = df.values

X = df.iloc[:, :-1]
y = df.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)

seed = 10
num_trees = 15
print("Using Ada Boost Classifiers, with no. of trees = ", num_trees)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

print("Accuracy = ", accuracy_score(y_pred, y_test))

y_true = y_test
print("\nConfusion Matrix: \n", confusion_matrix(y_true, y_pred))

matrix = classification_report(y_true, y_pred)
print("\nClassification report : \n", matrix)
```

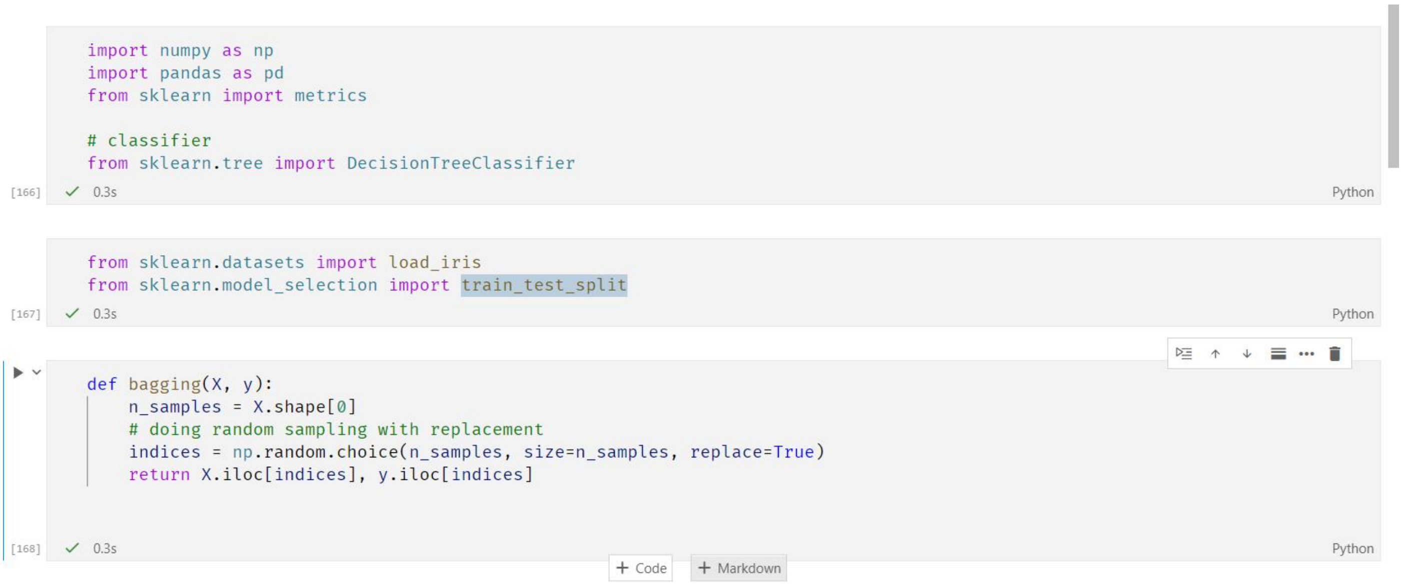
OUTPUT:-



```
(base) C:\Users\bmspr\OneDrive - Anna University\Documents\SEM5\ML\lab\anaconda\lab13>C:/ProgramData/Anaconda3/python.exe "c:/Users/b
Using Ada Boost Classifiers, with no. of trees =  15
Accuracy =  0.9811320754716981

Confusion Matrix:
 [[16  0  0]
 [ 0 17  0]
 [ 0  1 19]]

Classification report :
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        16
Iris-versicolor       0.94      1.00      0.97        17
 Iris-virginica       1.00      0.95      0.97        20

       accuracy                           0.98        53
      macro avg       0.98      0.98      0.98        53
   weighted avg       0.98      0.98      0.98        53

(base) C:\Users\bmspr\OneDrive - Anna University\Documents\SEM5\ML\lab\anaconda\lab13>
```

2. BAGGING CLASSIFIER

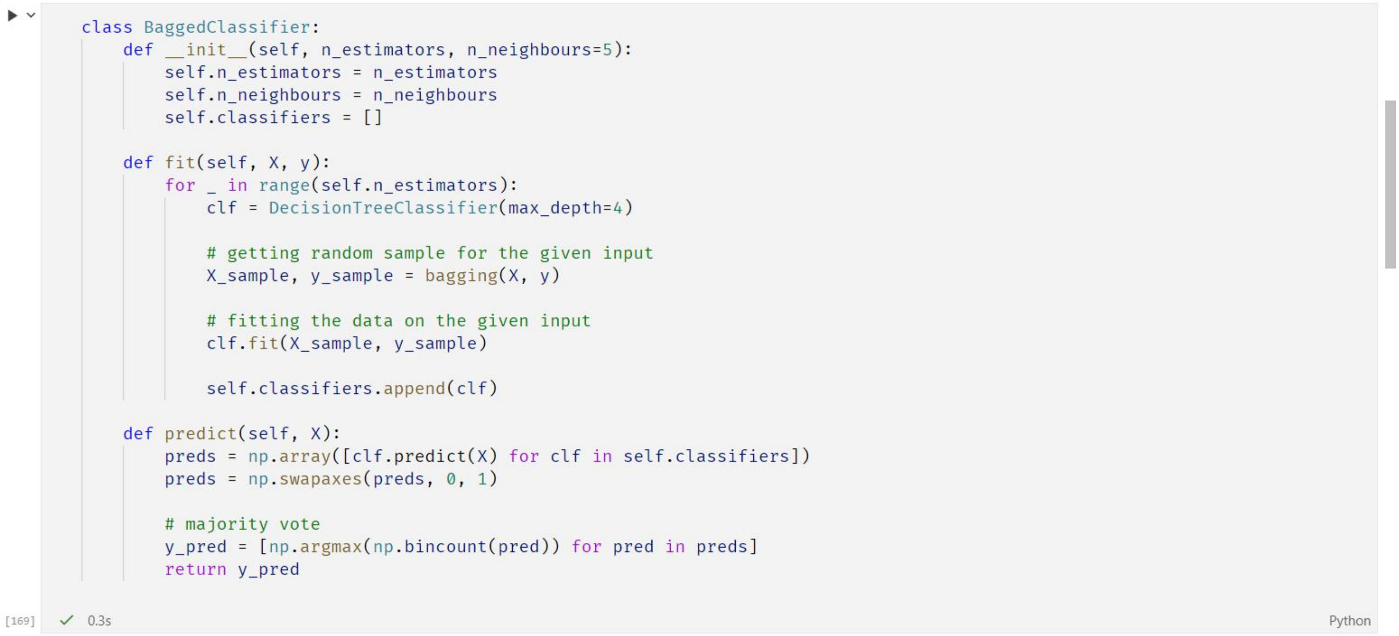
CODE:-



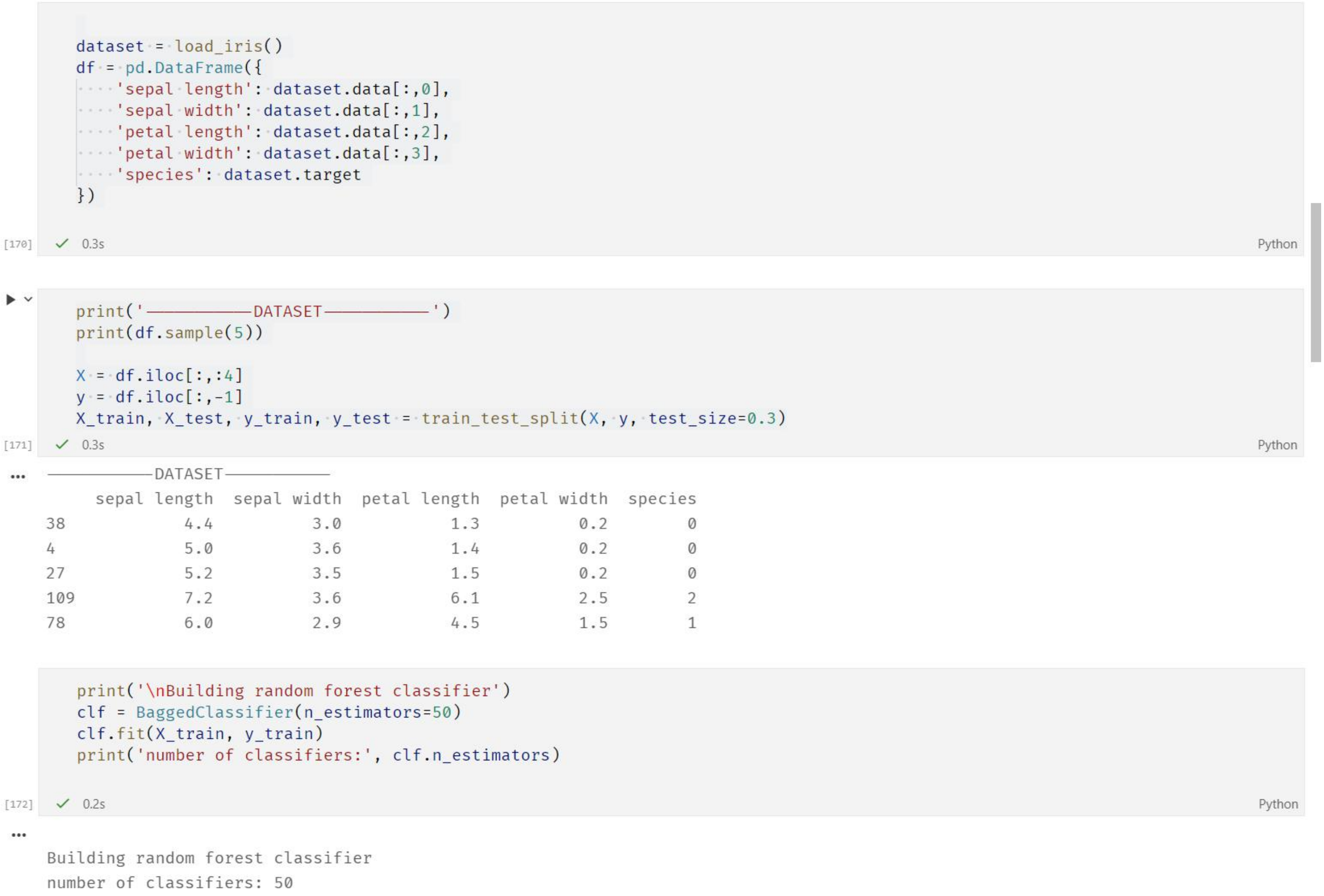
```
import numpy as np
import pandas as pd
from sklearn import metrics

# classifier
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
```

```
def bagging(X, y):
    n_samples = X.shape[0]
    # doing random sampling with replacement
    indices = np.random.choice(n_samples, size=n_samples, replace=True)
    return X.iloc[indices], y.iloc[indices]
```



```
class BaggedClassifier:
    def __init__(self, n_estimators, n_neighbours=5):
        self.n_estimators = n_estimators
        self.n_neighbours = n_neighbours
        self.classifiers = []

    def fit(self, X, y):
        for _ in range(self.n_estimators):
            clf = DecisionTreeClassifier(max_depth=4)

            # getting random sample for the given input
            X_sample, y_sample = bagging(X, y)

            # fitting the data on the given input
            clf.fit(X_sample, y_sample)

            self.classifiers.append(clf)

    def predict(self, X):
        preds = np.array([clf.predict(X) for clf in self.classifiers])
        preds = np.swapaxes(preds, 0, 1)

        # majority vote
        y_pred = [np.argmax(np.bincount(pred)) for pred in preds]
        return y_pred
```



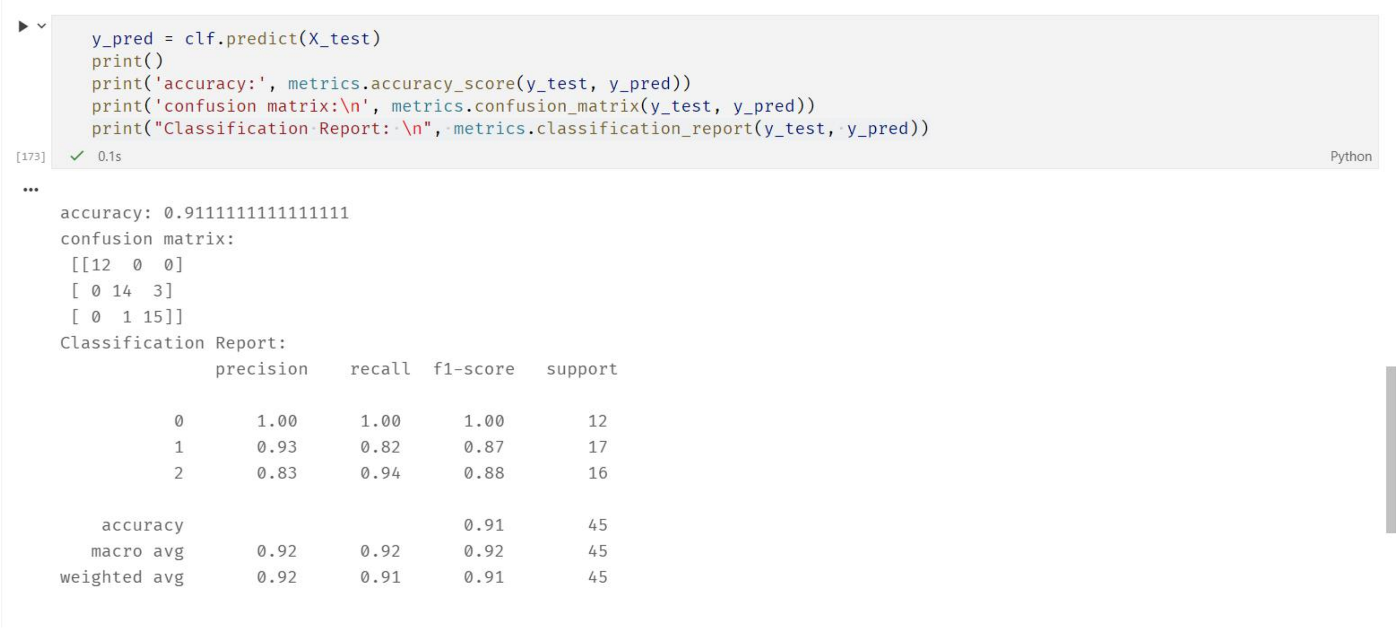
```
dataset = load_iris()
df = pd.DataFrame({
    'sepal length': dataset.data[:,0],
    'sepal width': dataset.data[:,1],
    'petal length': dataset.data[:,2],
    'petal width': dataset.data[:,3],
    'species': dataset.target
})
```

```
print('——————————DATASET——————————')
print(df.sample(5))

X = df.iloc[:,:4]
y = df.iloc[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
——————————DATASET——————————
     sepal length  sepal width  petal length  petal width  species
38            4.4          3.0           1.3          0.2        0
4             5.0          3.6           1.4          0.2        0
27            5.2          3.5           1.5          0.2        0
109           7.2          3.6           6.1          2.5        2
78            6.0          2.9           4.5          1.5        1
```

```
print('\nBuilding random forest classifier')
clf = BaggedClassifier(n_estimators=50)
clf.fit(X_train, y_train)
print('number of classifiers:', clf.n_estimators)
```

```
Building random forest classifier
number of classifiers: 50
```



```
y_pred = clf.predict(X_test)
print()
print('accuracy:', metrics.accuracy_score(y_test, y_pred))
print('confusion matrix:\n', metrics.confusion_matrix(y_test, y_pred))
print("Classification Report: \n", metrics.classification_report(y_test, y_pred))
```

```
accuracy: 0.9111111111111111
confusion matrix:
 [[12  0  0]
 [ 0 14  3]
 [ 0  1 15]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.93      0.82      0.87        17
           2       0.83      0.94      0.88        16

    accuracy                           0.91        45
   macro avg       0.92      0.92      0.92        45
weighted avg       0.92      0.91      0.91        45
```