Experiment No 9

Title: To implement CART

Lab Outcome: LO4: To demonstrate learning with classification.

Theory:

Nowadays, Decision Tree algorithm is known by its modern name CART which stands for Classification and Regression Trees. Classification and Regression Trees or CART is a term introduced by Leo Breiman to refer to Decision Tree algorithms that can be used for classification and regression modelling problems.

The CART algorithm provides a foundation for other important algorithms like bagged decision trees, random forest and boosted decision trees. In this kernel, I will solve a classification problem. So, I will refer the algorithm also as Decision Tree Classification problem.

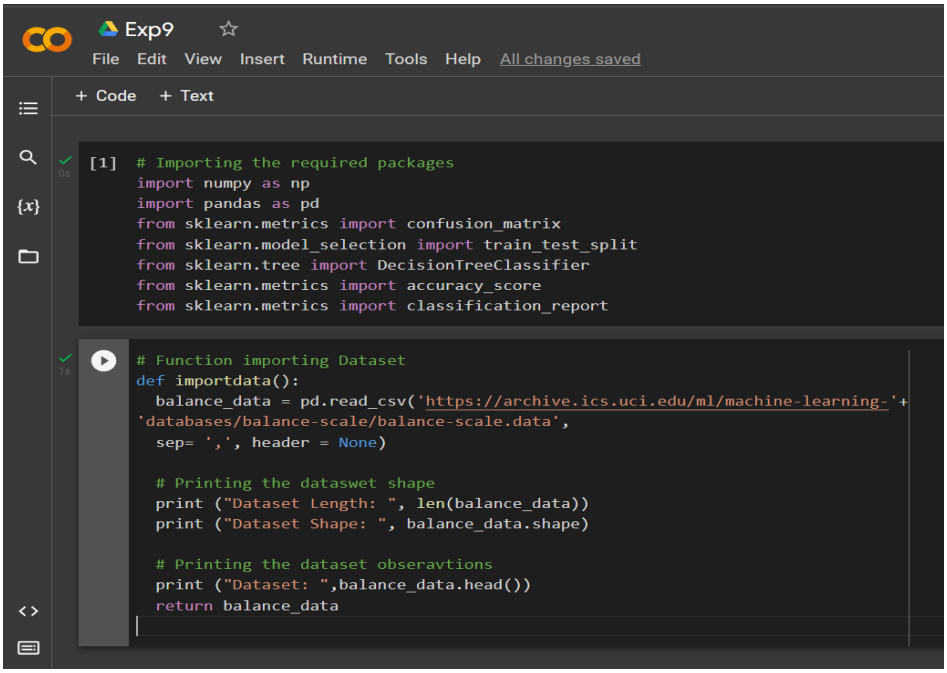
Program Code:

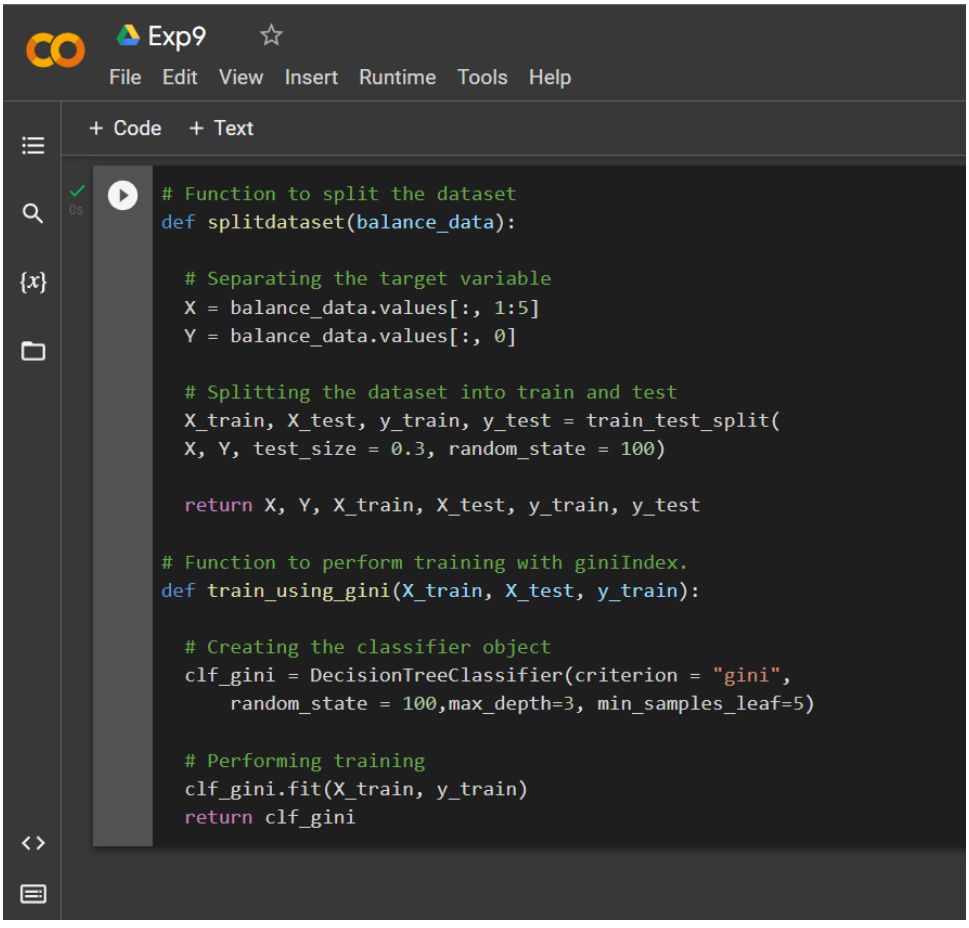
```
# Run this program on your local python
# interpreter, provided you have installed
# the required libraries.

# Importing the required packages
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

# Function importing Dataset
def importdata():
	balance_data = pd.read_csv(
'https://archive.ics.uci.edu/ml/machine-learning-'+
'databases/balance-scale/balance-scale.data',
	sep= ',', header = None)

	# Printing the dataswet shape
	print ("Dataset Length: ", len(balance_data))
	print ("Dataset Shape: ", balance_data.shape)

	# Printing the dataset obseravtions
	print ("Dataset: ",balance_data.head())
	return balance_data

# Function to split the dataset
def splitdataset(balance_data):

	# Separating the target variable
	X = balance_data.values[:, 1:5]
	Y = balance_data.values[:, 0]

	# Splitting the dataset into train and test
	X_train, X_test, y_train, y_test = train_test_split(
	X, Y, test_size = 0.3, random_state = 100)

	return X, Y, X_train, X_test, y_train, y_test
```

```
# Function to perform training with giniIndex.
def train_using_gini(X_train, X_test, y_train):

	# Creating the classifier object
	clf_gini = DecisionTreeClassifier(criterion = "gini",
			random_state = 100,max_depth=3, min_samples_leaf=5)

	# Performing training
	clf_gini.fit(X_train, y_train)
	return clf_gini

# Function to perform training with entropy.
def tarin_using_entropy(X_train, X_test, y_train):

	# Decision tree with entropy
	clf_entropy = DecisionTreeClassifier(
			criterion = "entropy", random_state = 100,
			max_depth = 3, min_samples_leaf = 5)

	# Performing training
	clf_entropy.fit(X_train, y_train)
	return clf_entropy

# Function to make predictions
def prediction(X_test, clf_object):

	# Predicton on test with giniIndex
	y_pred = clf_object.predict(X_test)
	print("Predicted values:")
	print(y_pred)
	return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):

	print("Confusion Matrix: ",
		confusion_matrix(y_test, y_pred))

	print ("Accuracy : ",
	accuracy_score(y_test,y_pred)*100)

	print("Report : ",
	classification_report(y_test, y_pred))

# Driver code
def main():

	# Building Phase
	data = importdata()
	X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
	clf_gini = train_using_gini(X_train, X_test, y_train)
	clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

	# Operational Phase
	print("Results Using Gini Index:")

	# Prediction using gini
	y_pred_gini = prediction(X_test, clf_gini)
	cal_accuracy(y_test, y_pred_gini)
```

```
print("Results Using Entropy:")
# Prediction using entropy
y_pred_entropy = prediction(X_test, clf_entropy)
cal_accuracy(y_test, y_pred_entropy)
# Calling main function
if __name__=="__main__":
main()
```

Implementation:





Exp9

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
# Function to perform training with entropy.
def tarin_using_entropy(X_train, X_test, y_train):

    # Decision tree with entropy
    clf_entropy = DecisionTreeClassifier(
            criterion = "entropy", random_state = 100,
            max_depth = 3, min_samples_leaf = 5)

    # Performing training
    clf_entropy.fit(X_train, y_train)
    return clf_entropy
```

Exp9

File Edit View Insert Runtime Tools Help

+ Code + Text

```
# Function to make predictions
def prediction(X_test, clf_object):

  # Predicton on test with giniIndex
  y_pred = clf_object.predict(X_test)
  print("Predicted values:")
  print(y_pred)
  return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):

  print("Confusion Matrix: ",
    confusion_matrix(y_test, y_pred))

  print ("Accuracy : ",
  accuracy_score(y_test,y_pred)*100)

  print("Report : ",
  classification_report(y_test, y_pred))
```

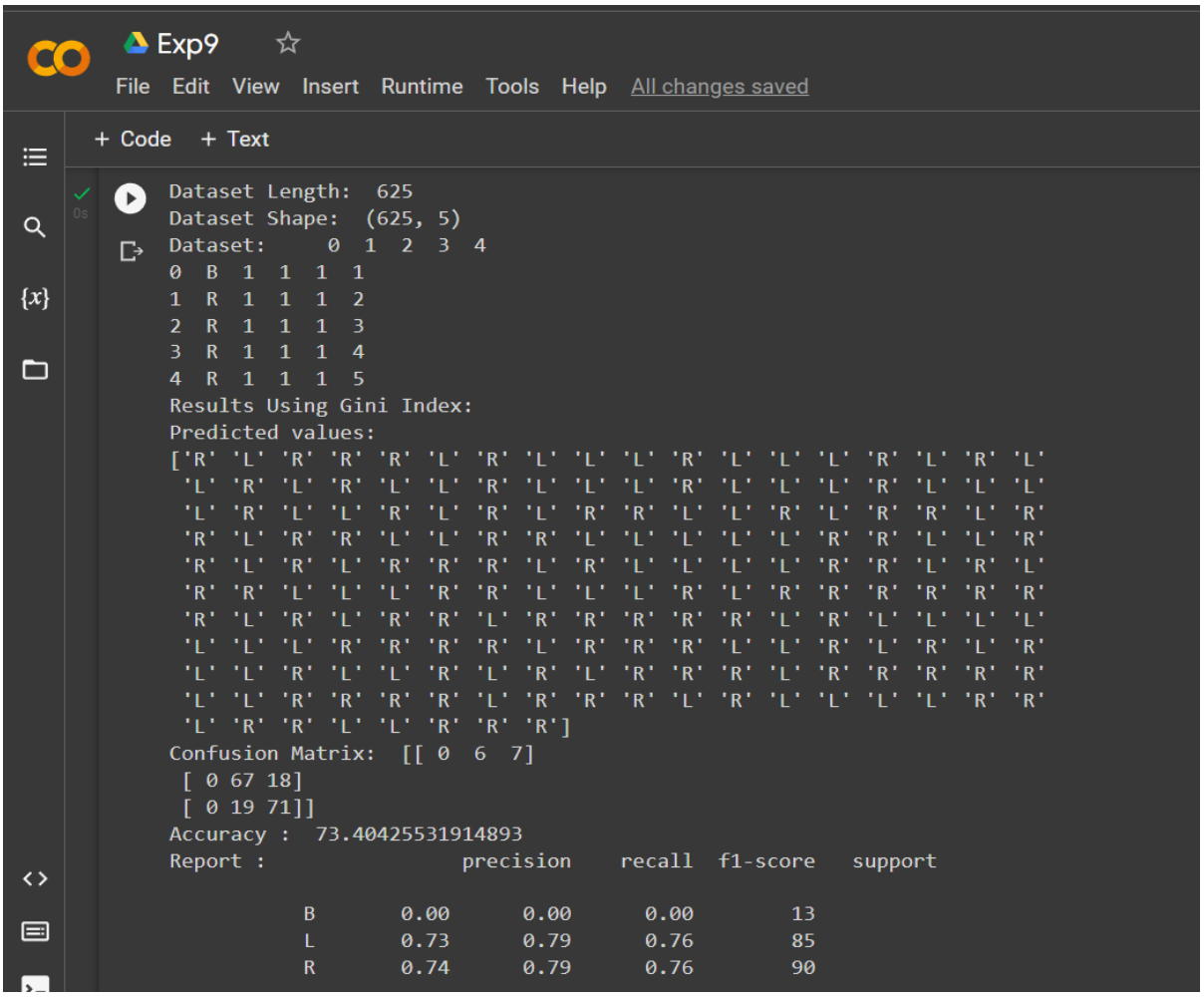
Exp9

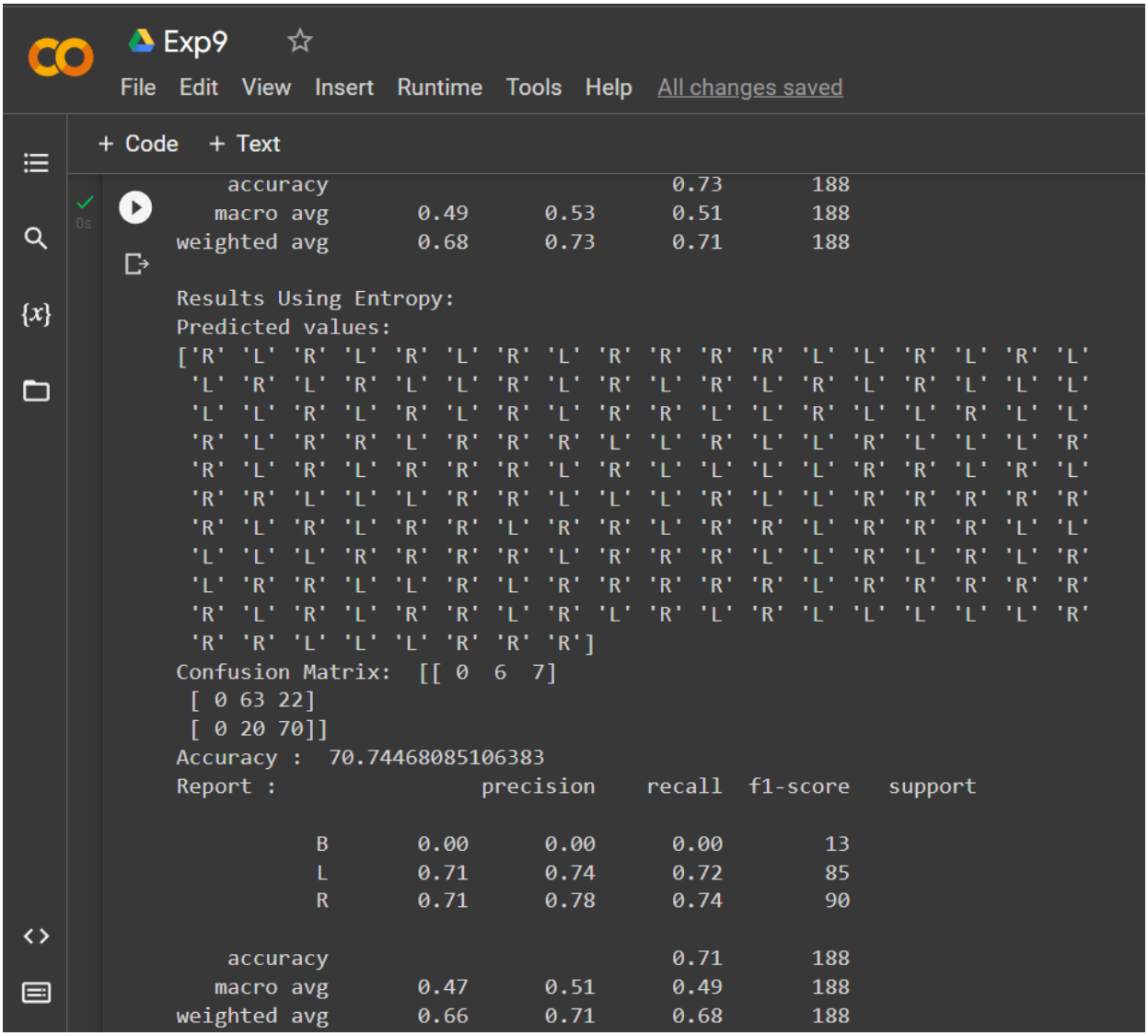
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
# Building Phase
data = importdata()
X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
clf_gini = train_using_gini(X_train, X_test, y_train)
clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

# Operational Phase
print("Results Using Gini Index:")

# Prediction using gini
y_pred_gini = prediction(X_test, clf_gini)
cal_accuracy(y_test, y_pred_gini)

print("Results Using Entropy:")
# Prediction using entropy
y_pred_entropy = prediction(X_test, clf_entropy)
cal_accuracy(y_test, y_pred_entropy)
```

Output:





Conclusion: Thus in this practical we studied performed CART using python programming language.