EXPERIMENT 3

DECISION TREE AND RANDOM FOREST ALGORITHM

NAME : SREENIDHI GANACHARI

REGISTRATION NUMBER : 19BCE7230

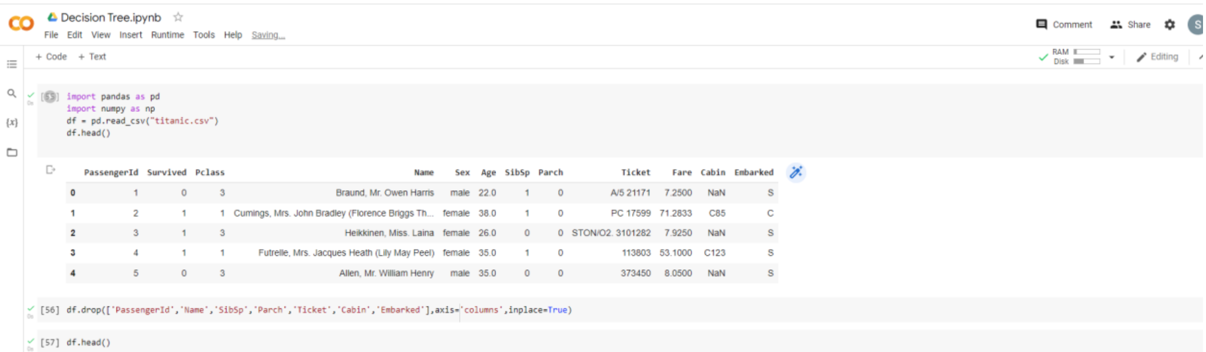
LAB SLOT : L-23+24

DECISION TREE –

CODE-

```
import pandas as pd
import numpy as np
df = pd.read_csv("titanic.csv")
df.head()
df.drop(['PassengerId','Name','SibSp','Parch','Ticket','Cabin','Embarke
d'],axis='columns',inplace=True)
inputs = df.drop('Survived',axis='columns')
target = df.Survived
inputs.Sex = inputs.Sex.map({'male': 1, 'female': 2})
inputs.Age[:10]
inputs.Age = inputs.Age.fillna(inputs.Age.mean())
inputs.head()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs,target,test_
size=0.2)
len(X_train)
len(X_test)
from sklearn import tree
model = tree.DecisionTreeClassifier()
model.fit(X_train,y_train)
model.score(X_test,y_test)
```

OUTPUT –

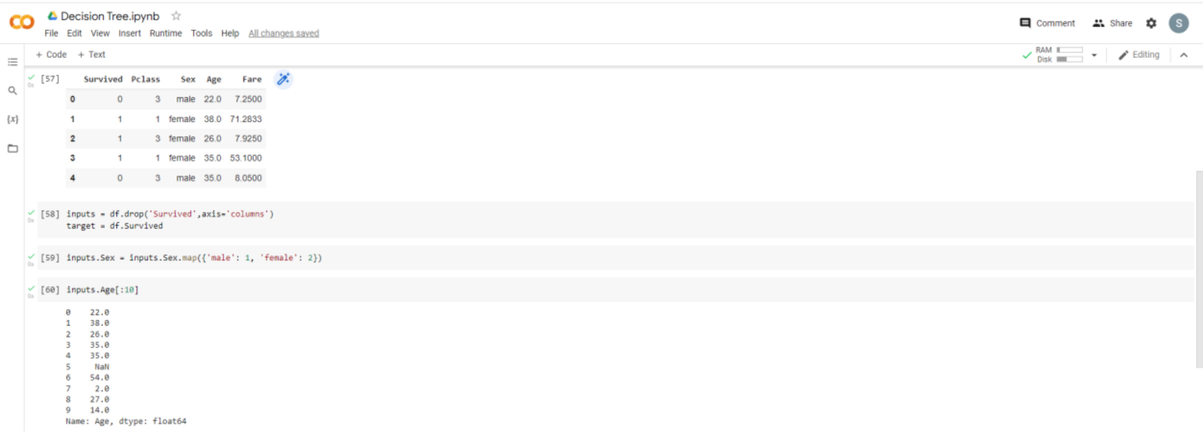


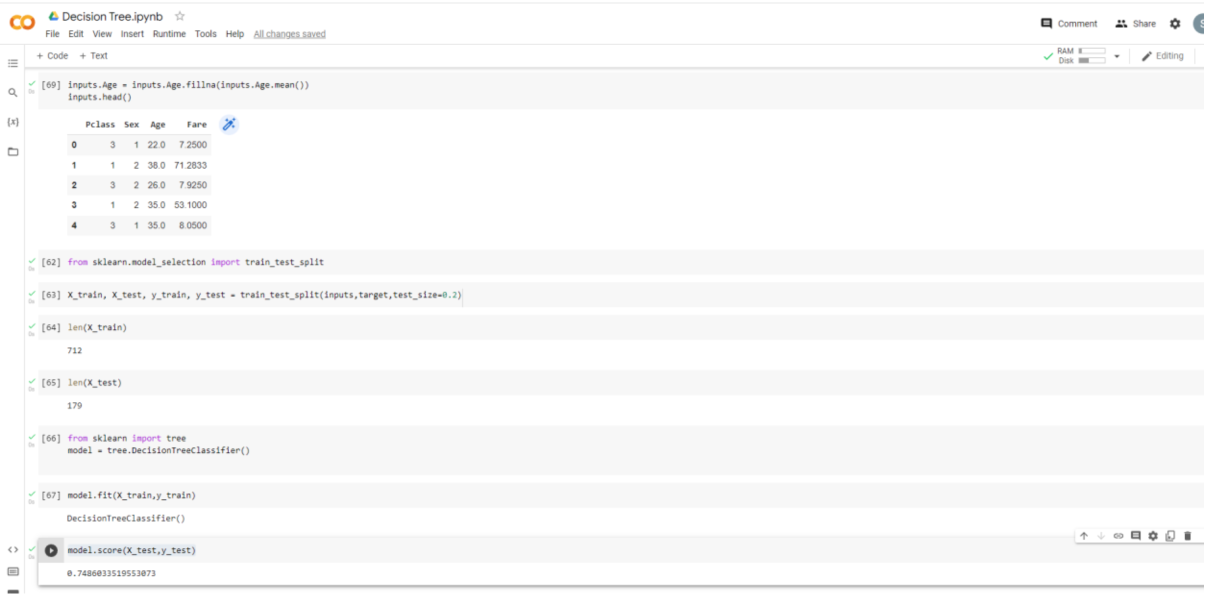
```
[55] import pandas as pd
import numpy as np
df = pd.read_csv("titanic.csv")
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
[56] df.drop(['PassengerId','Name','SibSp','Parch','Ticket','Cabin','Embarked'],axis='columns',inplace=True)
```

```
[57] df.head()
```





RANDOM FOREST ALGORITHM

CODE –

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
df = sns.load_dataset("iris")
df.head()
df.species.unique()
from sklearn.model_selection import train_test_split as tts
X = df.drop('species', axis=1)
y = df['species']
X_train, X_test, y_train, y_test = tts(X, y, test_size=0.5, random_stat
e=42)
from sklearn.ensemble import RandomForestClassifier
rfc1 = RandomForestClassifier(n_estimators=10)
```

```
rfc2 = RandomForestClassifier(n_estimators=50)
rfc3 = RandomForestClassifier(n_estimators=100)
rfc4 = RandomForestClassifier(n_estimators=150)
rfc5 = RandomForestClassifier(n_estimators=200)
rfc1.fit(X_train, y_train)
rfc2.fit(X_train, y_train)
rfc3.fit(X_train, y_train)
rfc4.fit(X_train, y_train)
rfc5.fit(X_train, y_train)
predictions1 = rfc1.predict(X_test)
predictions2 = rfc2.predict(X_test)
predictions3 = rfc3.predict(X_test)
predictions4 = rfc4.predict(X_test)
predictions5 = rfc5.predict(X_test)
from sklearn.metrics import confusion_matrix, classification_report, ac
curacy_score
print("Report for 10 trees: \n", classification_report(y_test, predicti
ons1))
print("Report for 50 trees: \n", classification_report(y_test, predicti
ons2))
print("Report for 100 trees: \n", classification_report(y_test,predicti
ons3))
print("Report for 150 trees: \n", classification_report(y_test,
predictions4))
print("Report for 200 trees: \n", classification_report(y_test,
predictions5))
print("Accuracy for 10 trees: ", accuracy_score(y_test, predictions1))
print("Accuracy for 50 trees: ", accuracy_score(y_test, predictions2))
print("Accuracy for 100 trees: ", accuracy_score(y_test, predictions3))
print("Accuracy for 150 trees: ", accuracy_score(y_test, predictions4))
print("Accuracy for 200 trees: ", accuracy_score(y_test, predictions5))
print()
print("Confusion Matrix for 10 trees:\n", confusion_matrix(y_test, pred
ictions1))
print("Confusion Matrix for 50 trees:\n", confusion_matrix(y_test, pred
ictions2))
print("Confusion Matrix for 100 trees:\n", confusion_matrix(y_test, pre
dictions3))
print("Confusion Matrix for 150 trees:\n", confusion_matrix(y_test, pre
dictions4))
print("Confusion Matrix for 200 trees:\n", confusion_matrix(y_test, pre
dictions5))
print("The Random Forest having highest accuracy is: ", accuracy_score(
y_test, predictions5))
print()
print(confusion_matrix(y_test, predictions5))
```

OUTPUT –

Random Forest.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

\+ Code + Text

[80]

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
df = sns.load_dataset("iris")
df.head()
df.species.unique()
from sklearn.model_selection import train_test_split as tts
X = df.drop('species', axis=1)
y = df['species']
X_train, X_test, y_train, y_test = tts(X, y, test_size=0.5, random_state=42)
from sklearn.ensemble import RandomForestClassifier
rfc1 = RandomForestClassifier(n_estimators=10)
rfc2 = RandomForestClassifier(n_estimators=50)
rfc3 = RandomForestClassifier(n_estimators=100)
rfc4 = RandomForestClassifier(n_estimators=150)
rfc5 = RandomForestClassifier(n_estimators=200)
rfc1.fit(X_train, y_train)
rfc2.fit(X_train, y_train)
rfc3.fit(X_train, y_train)
rfc4.fit(X_train, y_train)
rfc5.fit(X_train, y_train)
predictions1 = rfc1.predict(X_test)
predictions2 = rfc2.predict(X_test)
predictions3 = rfc3.predict(X_test)
predictions4 = rfc4.predict(X_test)
predictions5 = rfc5.predict(X_test)
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print("Report for 10 trees: \n", classification_report(y_test, predictions1))
```

0s completed at 8:28 PM

Random Forest.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

\+ Code + Text

[80]

```python
rfc3.fit(X_train, y_train)
rfc4.fit(X_train, y_train)
rfc5.fit(X_train, y_train)
predictions1 = rfc1.predict(X_test)
predictions2 = rfc2.predict(X_test)
predictions3 = rfc3.predict(X_test)
predictions4 = rfc4.predict(X_test)
predictions5 = rfc5.predict(X_test)
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print("Report for 10 trees: \n", classification_report(y_test, predictions1))
print("Report for 50 trees: \n", classification_report(y_test, predictions2))
print("Report for 100 trees: \n", classification_report(y_test,predictions3))
```

```
Report for 10 trees:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        29
  versicolor       0.88      1.00      0.94        23
   virginica       1.00      0.87      0.93        23

    accuracy                           0.96        75
   macro avg       0.96      0.96      0.96        75
weighted avg       0.96      0.96      0.96        75

Report for 50 trees:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        29
  versicolor       0.96      1.00      0.98        23
   virginica       1.00      0.96      0.98        23

    accuracy                           0.99        75
   macro avg       0.99      0.99      0.99        75
weighted avg       0.99      0.99      0.99        75
```


0s completed at 8:28 PM

Random Forest.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

\+ Code + Text

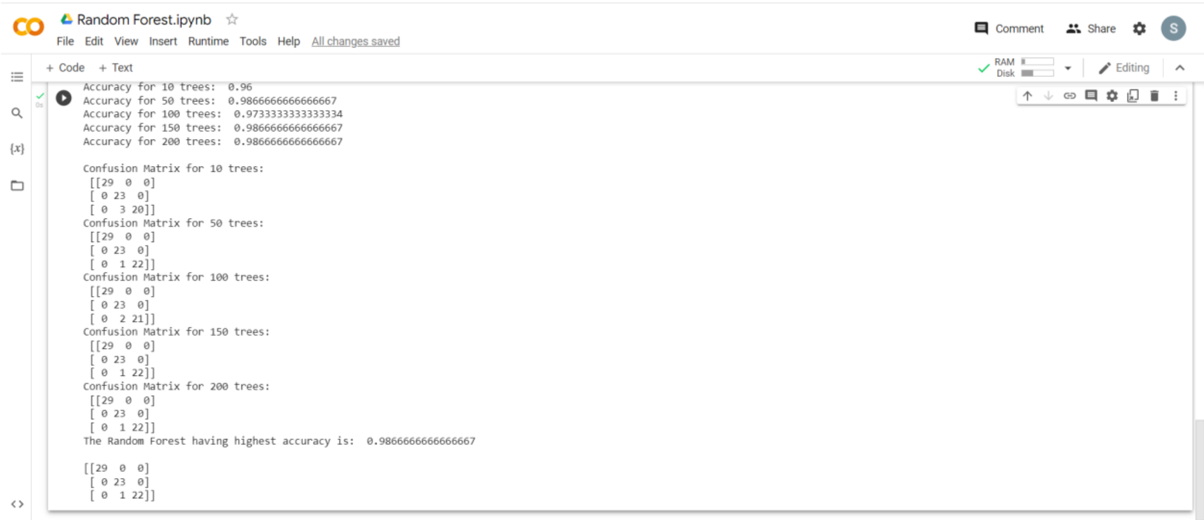
[80]

```
Report for 100 trees:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        29
  versicolor       0.92      1.00      0.96        23
   virginica       1.00      0.91      0.95        23

    accuracy                           0.97        75
   macro avg       0.97      0.97      0.97        75
weighted avg       0.98      0.97      0.97        75
```

```python
print("Report for 150 trees: \n", classification_report(y_test,
predictions4))
print("Report for 200 trees: \n", classification_report(y_test,
predictions5))
print("Accuracy for 10 trees: ", accuracy_score(y_test, predictions1))
print("Accuracy for 50 trees: ", accuracy_score(y_test, predictions2))
print("Accuracy for 100 trees: ", accuracy_score(y_test, predictions3))
print("Accuracy for 150 trees: ", accuracy_score(y_test, predictions4))
print("Accuracy for 200 trees: ", accuracy_score(y_test, predictions5))
print()
print("Confusion Matrix for 10 trees:\n", confusion_matrix(y_test, predictions1))
print("Confusion Matrix for 50 trees:\n", confusion_matrix(y_test, predictions2))
print("Confusion Matrix for 100 trees:\n", confusion_matrix(y_test, predictions3))
print("Confusion Matrix for 150 trees:\n", confusion_matrix(y_test, predictions4))
print("Confusion Matrix for 200 trees:\n", confusion_matrix(y_test, predictions5))
print("The Random Forest having highest accuracy is: ", accuracy_score(y_test, predictions5))
print()
print(confusion_matrix(y_test, predictions5))
```


0s completed at 8:28 PM

```
Report for 150 trees:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        29
  versicolor       0.96      1.00      0.98        23
   virginica       1.00      0.96      0.98        23

    accuracy                           0.99        75
   macro avg       0.99      0.99      0.99        75
weighted avg       0.99      0.99      0.99        75

Report for 200 trees:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        29
  versicolor       0.96      1.00      0.98        23
   virginica       1.00      0.96      0.98        23

    accuracy                           0.99        75
   macro avg       0.99      0.99      0.99        75
weighted avg       0.99      0.99      0.99        75

Accuracy for 10 trees:  0.96
Accuracy for 50 trees:  0.9866666666666667
Accuracy for 100 trees:  0.9733333333333334
Accuracy for 150 trees:  0.9866666666666667
Accuracy for 200 trees:  0.9866666666666667

Confusion Matrix for 10 trees:
 [[29  0  0]
 [ 0 23  0]
 [ 0  3 20]]
Confusion Matrix for 50 trees:
 [[29  0  0]
 [ 0 23  0]
 [ 0  1 22]]
```

```
Accuracy for 10 trees:  0.96
Accuracy for 50 trees:  0.9866666666666667
Accuracy for 100 trees:  0.9733333333333334
Accuracy for 150 trees:  0.9866666666666667
Accuracy for 200 trees:  0.9866666666666667

Confusion Matrix for 10 trees:
 [[29  0  0]
 [ 0 23  0]
 [ 0  3 20]]
Confusion Matrix for 50 trees:
 [[29  0  0]
 [ 0 23  0]
 [ 0  1 22]]
Confusion Matrix for 100 trees:
 [[29  0  0]
 [ 0 23  0]
 [ 0  2 21]]
Confusion Matrix for 150 trees:
 [[29  0  0]
 [ 0 23  0]
 [ 0  1 22]]
Confusion Matrix for 200 trees:
 [[29  0  0]
 [ 0 23  0]
 [ 0  1 22]]
The Random Forest having highest accuracy is:  0.9866666666666667

[[29  0  0]
 [ 0 23  0]
 [ 0  1 22]]
```


RANDOM FOREST ALGORITHM 2

CODE

```python
import pandas as pd
import numpy as np
dataset = pd.read_csv("Iris.csv")
dataset.head()
import numpy as np
feature_list=list(dataset.columns)
print(feature_list)
labels = np.array(dataset['Species'])
dataset= dataset.drop('Species', axis = 1)
dataset = np.array(dataset)
print(dataset)
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, y_train, y_test = train_test_split(dataset, labels,
test_size=0.20, random_state=0)
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=500, random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix,acc
uracy_score
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

OUTPUT –