

EXPERIMENT 3

DECISION TREE AND RANDOM FOREST ALGORITHM

NAME : SREENIDHI GANACHARI

REGISTRATION NUMBER : 19BCE7230

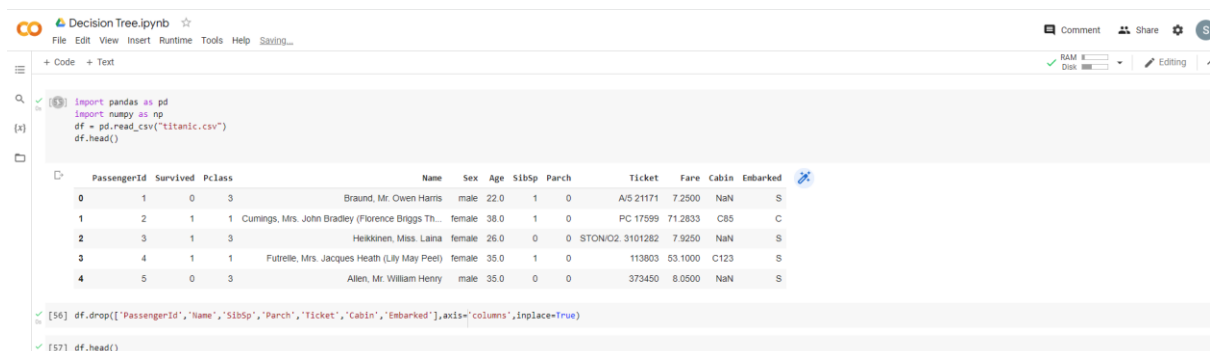
LAB SLOT : L-23+24

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', 'Embarked'], 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 –



The screenshot displays a Jupyter Notebook interface with the following content:

- Code Cell:** Contains the Python code for loading the Titanic dataset, preprocessing it, and training a decision tree model.
- Output:** Shows the first five rows of the dataset as a table.
- Execution Log:** Displays the execution of the code cells, including the output of the decision tree model.

	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	Cummings, 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

Decision Tree.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM 8 Disk 100

Editing

```
[57] Survived Pclass Sex Age Fare
0 0 3 male 22.0 7.2500
1 1 1 female 38.0 71.2833
2 1 3 female 26.0 7.9250
3 1 1 female 35.0 53.1000
4 0 3 male 35.0 8.0500

[58] inputs = df.drop('Survived',axis='columns')
target = df.Survived

[59] inputs.Sex = inputs.Sex.map({'male': 1, 'female': 2})

[60] inputs.Age[:10]

0 22.0
1 38.0
2 26.0
3 35.0
4 35.0
5 NaN
6 54.0
7 2.0
8 27.0
9 14.0
Name: Age, dtype: float64
```

Decision Tree.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM 8 Disk 100

Editing

```
[69] inputs.Age = inputs.Age.fillna(inputs.Age.mean())
inputs.head()

Pclass Sex Age Fare
0 3 1 22.0 7.2500
1 1 2 38.0 71.2833
2 3 2 26.0 7.9250
3 1 2 35.0 53.1000
4 3 1 35.0 8.0500

[62] from sklearn.model_selection import train_test_split

[63] X_train, X_test, y_train, y_test = train_test_split(inputs,target,test_size=0.3)

[64] len(X_train)

712

[65] len(X_test)

179

[66] from sklearn import tree
model = tree.DecisionTreeClassifier()

[67] model.fit(X_train,y_train)
DecisionTreeClassifier()

model.score(X_test,y_test)

0.7486033519553073
```

```

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))
print("Report for 50 trees: \n", classification_report(y_test, predictions2))
print("Report for 100 trees: \n", classification_report(y_test, predictions3))
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))

```

OUTPUT –

Random Forest.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

Editing

ts

[80]

```
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

+ Code + Text

RAM Disk

Editing

ts

[80]

```
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

+ Code + Text

RAM Disk

Editing

ts

[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 0.97 0.97 75
macro avg 0.97 0.97 0.97 75
weighted avg 0.98 0.97 0.97 75
```

```
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    0.99    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    0.99    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]]

```

```

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

```

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, accuracy_score
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))

```

OUTPUT –

```

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, accuracy_score
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test, y_pred))

```

```

[[['Id', 'SepallengthCm', 'SepalwidthCm', 'PetallengthCm', 'PetalwidthCm', 'Species']
 [1.00e+00 5.10e+00 3.50e+00 1.40e+00 2.00e-01]
 [2.00e+00 4.90e+00 3.00e+00 1.40e+00 2.00e-01]
 [3.00e+00 4.70e+00 3.20e+00 1.30e+00 2.00e-01]
 [4.00e+00 4.60e+00 3.10e+00 1.50e+00 2.00e-01]
 [5.00e+00 5.00e+00 3.60e+00 1.40e+00 2.00e-01]
 [6.00e+00 5.40e+00 3.90e+00 1.70e+00 4.00e-01]
 [7.00e+00 4.60e+00 3.40e+00 1.40e+00 3.00e-01]
 [8.00e+00 5.00e+00 3.40e+00 1.50e+00 2.00e-01]
 [9.00e+00 4.40e+00 2.90e+00 1.40e+00 2.00e-01]
 [1.00e+01 4.90e+00 3.10e+00 1.50e+00 1.00e-01]
 [1.10e+01 5.40e+00 3.70e+00 1.50e+00 2.00e-01]
 [1.20e+01 4.80e+00 3.40e+00 1.60e+00 2.00e-01]
 [1.30e+01 4.80e+00 3.00e+00 1.40e+00 1.00e-01]
 [1.40e+01 4.30e+00 3.00e+00 1.10e+00 1.00e-01]
 [1.50e+01 5.80e+00 4.00e+00 1.20e+00 2.00e-01]
 [1.60e+01 5.70e+00 4.40e+00 1.50e+00 4.00e-01]
 [1.70e+01 5.40e+00 3.90e+00 1.30e+00 4.00e-01]
 [1.80e+01 5.10e+00 3.50e+00 1.40e+00 3.00e-01]
 [1.90e+01 5.70e+00 3.80e+00 1.70e+00 3.00e-01]
 [2.00e+01 5.10e+00 3.80e+00 1.50e+00 3.00e-01]
 [2.10e+01 5.40e+00 3.40e+00 1.70e+00 2.00e-01]
 [2.20e+01 5.10e+00 3.70e+00 1.50e+00 4.00e-01]
 [2.30e+01 4.60e+00 3.60e+00 1.00e+00 2.00e-01]]

```

1s completed at 10:24 PM

```

[[1.38e+02 6.40e+00 3.10e+00 5.50e+00 1.80e+00]
 [1.39e+02 6.00e+00 3.00e+00 4.80e+00 1.80e+00]
 [1.40e+02 6.90e+00 3.10e+00 5.40e+00 2.10e+00]
 [1.41e+02 6.70e+00 3.10e+00 5.60e+00 2.40e+00]
 [1.42e+02 6.90e+00 3.10e+00 5.10e+00 2.30e+00]
 [1.43e+02 5.80e+00 2.70e+00 5.10e+00 1.90e+00]
 [1.44e+02 6.80e+00 3.20e+00 5.90e+00 2.30e+00]
 [1.45e+02 6.70e+00 3.30e+00 5.70e+00 2.50e+00]
 [1.46e+02 6.70e+00 3.00e+00 5.20e+00 2.30e+00]
 [1.47e+02 6.30e+00 2.50e+00 5.00e+00 1.90e+00]
 [1.48e+02 6.50e+00 3.00e+00 5.20e+00 2.00e+00]
 [1.49e+02 6.20e+00 3.40e+00 5.40e+00 2.30e+00]
 [1.50e+02 5.90e+00 3.00e+00 5.10e+00 1.80e+00]]

```

```

[[11 0 0]
 [0 13 0]
 [0 0 6]]

```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	6
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

1.0