Decision Tree Classification

```
import io
import pandas as pd
import numpy as nm
import matplotlib.pyplot as mtp
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving suv_data.csv to suv_data.csv
```

```
df2 = pd.read_csv(io.BytesIO(uploaded['suv_data.csv']))
```

```
#Extracting Independent and dependent Variable
x= df2.iloc[:, [2,3]].values
y= df2.iloc[:, 4].values
df2.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)

#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

```
#Fitting Decision Tree classifier to the training set


from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(x_train, y_train)
```

```
▾                DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

```
import pandas as pd
from tabulate import tabulate

# Assuming y_pred and y_test are numpy arrays or lists
data = {'y_pred': y_pred, 'y_test': y_test}
df = pd.DataFrame(data)

# Display the DataFrame with bold headings, margins, and outline
table = tabulate(df, headers='keys', tablefmt='fancy_grid', showindex=False)
```

```
print(table)
```

```
╒══════════╤══════════╕
│   y_pred │   y_test │
╞══════════╪══════════╡
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        1 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        0 │
├──────────┼──────────┤
│        1 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        1 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        1 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        1 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        1 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
```

```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
```

```
# Convert the confusion matrix to a DataFrame for better display
confusion_matrix_df = pd.DataFrame(cm, columns=['Predicted 0', 'Predicted 1'], index=['Actual 0', 'Actual 1'])

# Display the confusion matrix with bold headings, margins, and outline
confusion_matrix_table = tabulate(confusion_matrix_df, headers='keys', tablefmt='fancy_grid', showindex=True)

print("Confusion Matrix:")
print(confusion_matrix_table)
```

```
Confusion Matrix:
╒══════════╤═══════════════╤═══════════════╕
│          │   Predicted 0 │   Predicted 1 │
╞══════════╪═══════════════╪═══════════════╡
```

```
│ Actual 0 │          62 │           6 │
├──────────┼─────────────┼─────────────┤
│ Actual 1 │           3 │          29 │
╘══════════╧═════════════╧═════════════╛
```

POST-LAB Task

1. Download the Titanic dataset. Perform all the preprocessing required. Predict the survival for the test set using the Decision Tree Classification and compare the accuracy with the Naive Bayes Algorithm

Importing Data

```
dataset = pd.read_csv('https://github.com/umairbinmansoor/Datasets/raw/main/titanic_train.csv')
#X = dataset.iloc[:,:4].values
#y = dataset['species'].values
dataset.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

```
x = dataset.iloc[:, [2, 3]].values  # Assuming columns at index 2 and 3 are independent variables
y = dataset.iloc[:, 4].values      # Assuming column at index 4 is the dependent variable
dataset.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

Splitting dataset into training and testing

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
```

```
numerical_features = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']  # Replace with actual numerical column names if d
x = dataset[numerical_features].values
y = dataset['Survived'].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
st_x = StandardScaler()
x_train = st_x.fit_transform(x_train)
x_test = st_x.transform(x_test)
```

Fitting Decision Tree classifier to the training set

```
from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(x_train, y_train)
```

```
▾              DecisionTreeClassifier          i ?
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
y_pred= classifier.predict(x_test)
```

```
import pandas as pd
from tabulate import tabulate
data = {'y_pred': y_pred, 'y_test': y_test}
df = pd.DataFrame(data)
table = tabulate(df, headers='keys', tablefmt='fancy_grid', showindex=False)
print(table)
```

```
╒══════════╤══════════╕
│   y_pred │   y_test │
╞══════════╪══════════╡
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        1 │
├──────────┼──────────┤
│        1 │        1 │
├──────────┼──────────┤
│        0 │        1 │
├──────────┼──────────┤
│        1 │        1 │
├──────────┼──────────┤
│        1 │        1 │
├──────────┼──────────┤
│        0 │        1 │
├──────────┼──────────┤
│        0 │        1 │
├──────────┼──────────┤
│        1 │        0 │
├──────────┼──────────┤
│        0 │        1 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        1 │
├──────────┼──────────┤
│        0 │        1 │
├──────────┼──────────┤
│        1 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        1 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        1 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        1 │
├──────────┼──────────┤
│        0 │        0 │
├──────────┼──────────┤
│        1 │        1 │
```

Creating Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
```

```
confusion_matrix_df = pd.DataFrame(cm, columns=['Predicted 0', 'Predicted 1'], index=['Actual 0', 'Actual 1'])
confusion_matrix_table = tabulate(confusion_matrix_df, headers='keys', tablefmt='fancy_grid', showindex=True)
print("Confusion Matrix:")
print(confusion_matrix_table)
```

```
Confusion Matrix:
╒══════════╤═══════════════╤═══════════════╕
│          │   Predicted 0 │   Predicted 1 │
╞══════════╪═══════════════╪═══════════════╡
│ Actual 0 │           108 │            31 │
├──────────┼───────────────┼───────────────┤
│ Actual 1 │            42 │            42 │
╘══════════╧═══════════════╧═══════════════╛
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the Decision Tree model: {accuracy}")
```

```
Accuracy of the Decision Tree model: 0.672645739910314
```

Importing Libraries

```
from sklearn.naive_bayes import GaussianNB
from tabulate import tabulate
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Finding Accuracy Via Naive Bayes Model

```
# Naive Bayes Classifier
nb_classifier = GaussianNB()
nb_classifier.fit(x_train, y_train)
nb_y_pred = nb_classifier.predict(x_test)
nb_accuracy = accuracy_score(y_test, nb_y_pred)
print(f"Accuracy of the Naive Bayes model: {nb_accuracy}")
```

```
Accuracy of the Naive Bayes model: 0.7219730941704036
```

Comparing Results

```
print("\nComparison:")
if accuracy > nb_accuracy:
    print("Decision Tree model has higher accuracy.")
elif nb_accuracy > accuracy:
    print("Naive Bayes model has higher accuracy.")
else:
    print("Both models have the same accuracy.")
```

```

Comparison:
Naive Bayes model has higher accuracy.
```

Upon comparing the Decison Tree Model provides less accuracy when being compared to Naive bayes Model.