Decision Tree Classification

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
import matplotlib.pyplot as plt
```

```
# Step 1: Creating the Dataset
data = {
    "Height": [170, 170, 185, 155, 175],
    "Weight": [75, 90, 85, 50, 78],
    "Gender": ["Female", "Male", "Male", "Female", "Female"]
}

df = pd.DataFrame(data)
```

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

# Step 2: Encoding the Target Variable (Gender)
# df["Gender"] = df["Gender"].map({"Male": 1, "Female": 0})
df['Gender']=label_encoder.fit_transform(df['Gender'])

# Step 3: Splitting Features (X) and Target Variable (y)
X = df[["Height", "Weight"]]
y = df["Gender"]
X.columns
```

```
Index(['Height', 'Weight'], dtype='object')
```

```
y
```

```
0    0
1    1
2    1
3    0
4    0
Name: Gender, dtype: int32
```

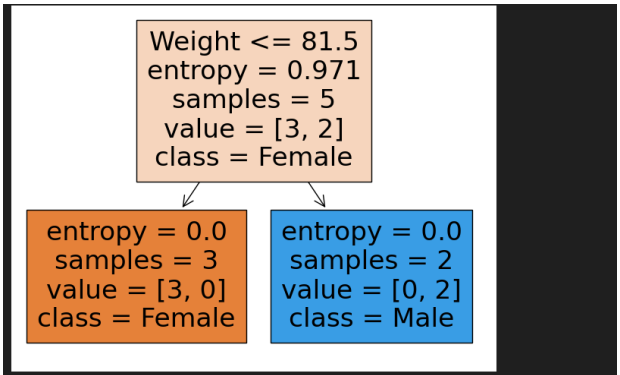
```
# Step 4: Training the Decision Tree Classifier
model = DecisionTreeClassifier(criterion="entropy", max_depth=3, random_state=42)
model.fit(X, y)

# Step 5: Printing Decision Rules
tree_rules = export_text(model, feature_names=X.columns.to_list())
print("\nDecision Tree Rules:\n")
print(tree_rules)

# Step 6: Visualizing the Decision Tree
plt.figure(figsize=(8, 6))
plot_tree(model, feature_names=X.columns.tolist(), class_names=["Female", "Male"], filled=True)
plt.show()
```

```
Decision Tree Rules:

|--- Weight <= 81.50
|   |--- class: 0
|--- Weight >  81.50
|   |--- class: 1
```



Height	Weight	Gender
170	75	Female
170	90	Male
185	85	Male
155	50	Female
175	78	Female

Weight	
50	50+75/2 =62.5
75	75+78/2=76.5
78	78+85/2=81.5
85	85+90/2=87.5
90	

```
df = pd.read_csv("salaries.csv")
df
```

	company	job	degree	salary_more_then_100k
0	google	sales executive	bachelors	0
1	google	sales executive	masters	0
2	google	business manager	bachelors	1
3	google	business manager	masters	1
4	google	computer programmer	bachelors	0
5	google	computer programmer	masters	1
6	abc pharma	sales executive	masters	0
7	abc pharma	computer programmer	bachelors	0
8	abc pharma	business manager	bachelors	0
9	abc pharma	business manager	masters	1
10	facebook	sales executive	bachelors	1
11	facebook	sales executive	masters	1
12	facebook	business manager	bachelors	1
13	facebook	business manager	masters	1
14	facebook	computer programmer	bachelors	1
15	facebook	computer programmer	masters	1

```
inputs = df.drop('salary_more_then_100k',axis='columns')
```

```
target = df['salary_more_then_100k']
```

```
from sklearn.preprocessing import LabelEncoder
le_company = LabelEncoder()
le_job = LabelEncoder()
le_degree = LabelEncoder()
```

```
inputs['company_n'] = le_company.fit_transform(inputs['company'])
inputs['job_n'] = le_job.fit_transform(inputs['job'])
inputs['degree_n'] = le_degree.fit_transform(inputs['degree'])
```

```
inputs
```

	company	job	degree	company_n	job_n	degree_n
0	google	sales executive	bachelors	2	2	0
1	google	sales executive	masters	2	2	1
2	google	business manager	bachelors	2	0	0
3	google	business manager	masters	2	0	1
4	google	computer programmer	bachelors	2	1	0
5	google	computer programmer	masters	2	1	1
6	abc pharma	sales executive	masters	0	2	1
7	abc pharma	computer programmer	bachelors	0	1	0
8	abc pharma	business manager	bachelors	0	0	0
9	abc pharma	business manager	masters	0	0	1
10	facebook	sales executive	bachelors	1	2	0
11	facebook	sales executive	masters	1	2	1
12	facebook	business manager	bachelors	1	0	0
13	facebook	business manager	masters	1	0	1
14	facebook	computer programmer	bachelors	1	1	0
15	facebook	computer programmer	masters	1	1	1

```
inputs_n = inputs.drop(['company','job','degree'],axis='columns')
```

```
X = inputs_n
```

```
Y = target
```

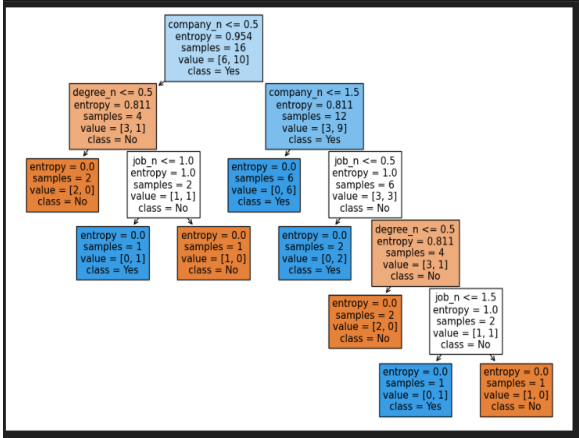
```
#Split data into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.1)
```

```
from sklearn import tree
model = tree.DecisionTreeClassifier(criterion="entropy")
```

```
model.fit(inputs_n, target)
```

```
DecisionTreeClassifier(criterion='entropy')
```

```
# Plot Decision Tree
plt.figure(figsize=(12, 8))
plot_tree(model, feature_names=X.columns.tolist(), class_names=['No', 'Yes'], filled=True)
plt.show()
```




```
y_pred = model.predict(X_test)
```

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

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
cr = classification_report(y_test, y_pred)
print("\nClassification Report:",)
print (cr)
acc = accuracy_score(y_test,y_pred)
print("\nAccuracy:",acc)
```

```
Confusion Matrix:
[[2]]

Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2


Accuracy: 1.0
```

Is salary of Google, Computer Engineer, Bachelors degree > 100 k ?

```
model.predict([[2,1,0]])
```

```
C:\Users\ELWIN G\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\L
  warnings.warn(

array([0], dtype=int64)
```

Is salary of Google, Computer Engineer, Masters degree > 100 k ?

```
model.predict([[2,1,1]])
```

```
C:\Users\ELWIN G\AppData\Local\Packages\PythonSoftwareFoundation.Pyth
  warnings.warn(

array([1], dtype=int64)
```

Generate classification report, confusion matrix and accuracy on the test set (you may consider 30% of the data for testing)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score

# Load dataset
df = pd.read_csv("titanic.csv")[['Pclass', 'Sex', 'Age', 'Fare', 'Survived']]

# Handle missing values in 'Age' (replace with median)
df['Age'] = SimpleImputer(strategy='median').fit_transform(df[['Age']])

# Encode categorical variable 'Sex'
df['Sex'] = LabelEncoder().fit_transform(df['Sex'])

# Split dataset
X = df[['Pclass', 'Sex', 'Age', 'Fare']]
y = df['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train Decision Tree model (small and clear)
model = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
model.fit(X_train, y_train)

# Predictions and accuracy
y_pred = model.predict(X_test)
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Plot simplified decision tree
plt.figure(figsize=(10, 6))
plot_tree(model, feature_names=X.columns, class_names=['Not Survived', 'Survived'], filled=True)
plt.show()
```

```
plt.show()
```

```
Accuracy Score: 0.8059701492537313
```