LAB # 05

SUPERVISED LEARNING (DECISION TREE)

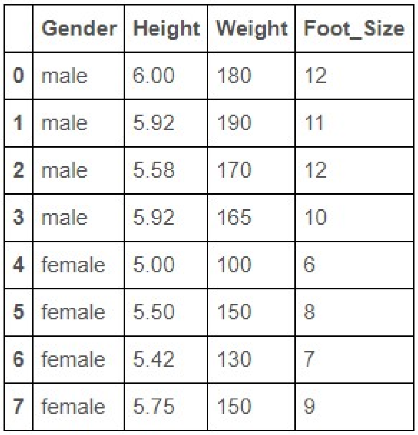
OBJECTIVE

Implementing supervised learning, DTS algorithm for training, testing and classification.

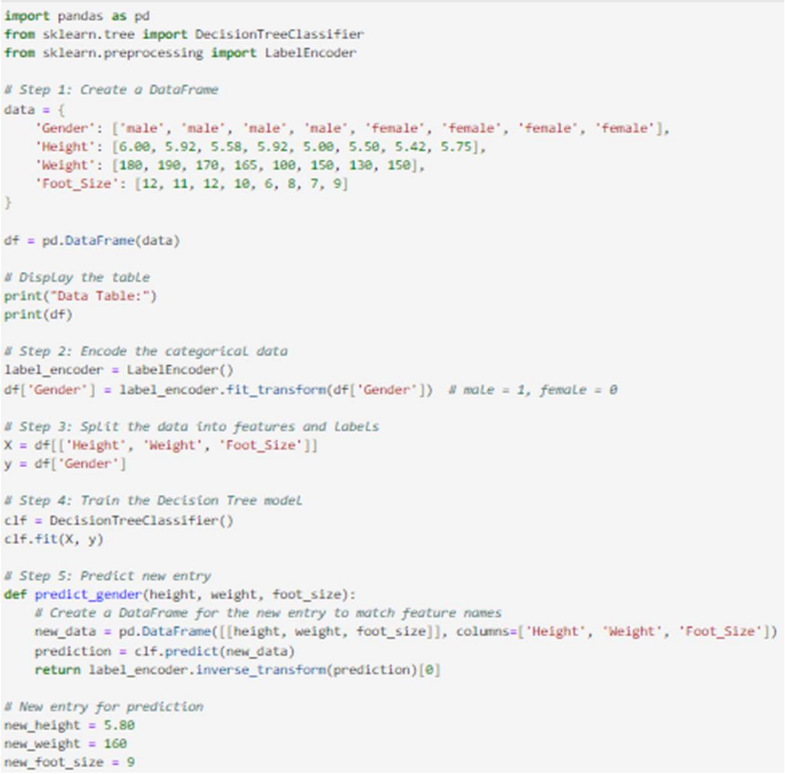
Lab Tasks

1. Implement the Decision tree algorithm on the data given in the table. 1 and predict the new entry entered by the user.

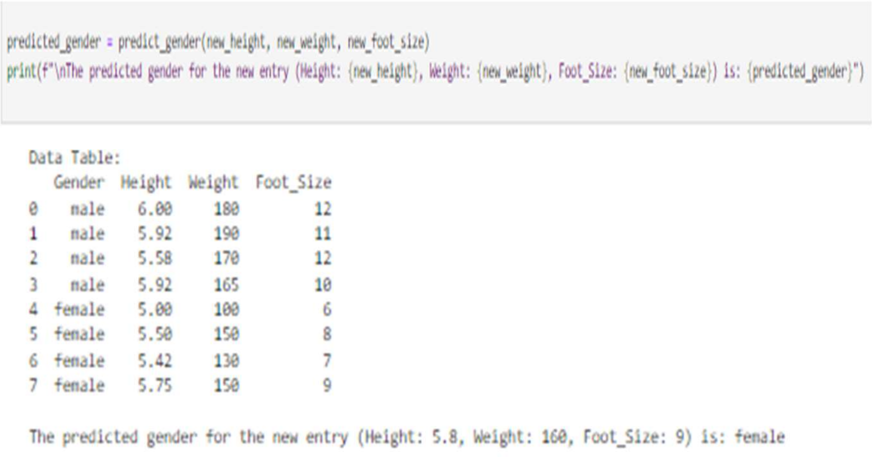
Table. 1



	Gender	Height	Weight	Foot_Size
0	male	6.00	180	12
1	male	5.92	190	11
2	male	5.58	170	12
3	male	5.92	165	10
4	female	5.00	100	6
5	female	5.50	150	8
6	female	5.42	130	7
7	female	5.75	150	9



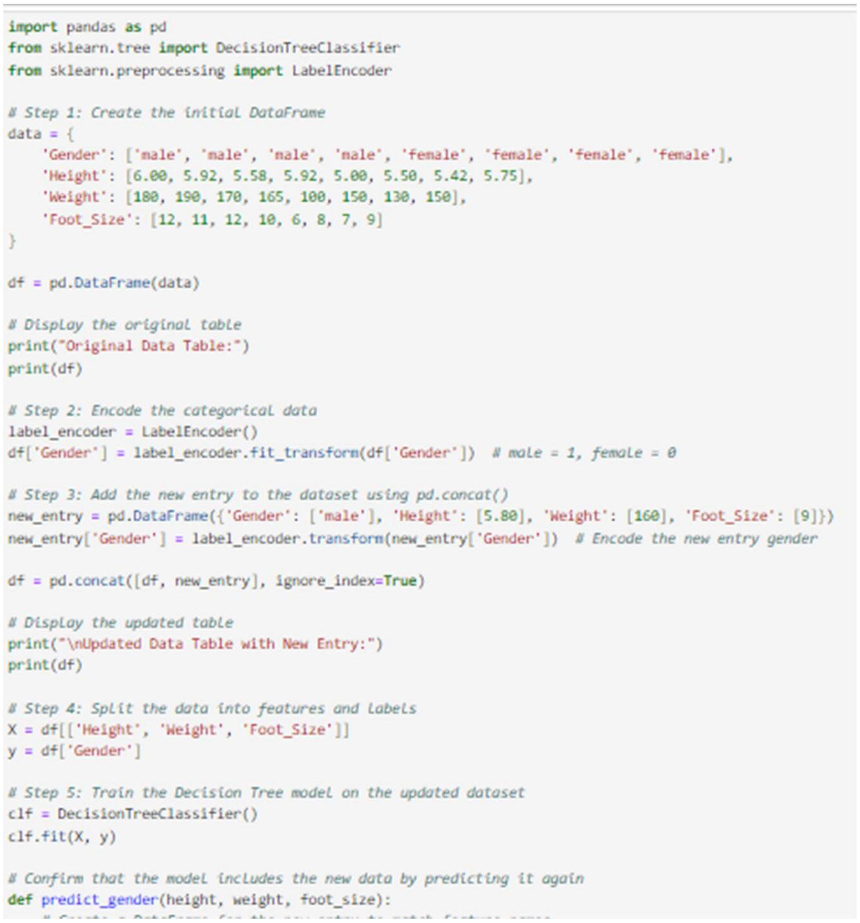
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder

# Step 1: Create a DataFrame
data = {
    'Gender': ['male', 'male', 'male', 'male', 'female', 'female', 'female', 'female'],
    'Height': [6.00, 5.92, 5.58, 5.92, 5.00, 5.50, 5.42, 5.75],
    'Weight': [180, 190, 170, 165, 100, 150, 130, 150],
    'Foot_Size': [12, 11, 12, 10, 6, 8, 7, 9]
}

df = pd.DataFrame(data)

# Display the table
print("Data Table:")
print(df)

# Step 2: Encode the categorical data
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])  # male = 1, female = 0

# Step 3: Split the data into features and labels
X = df[['Height', 'Weight', 'Foot_Size']]
y = df['Gender']

# Step 4: Train the Decision Tree model
clf = DecisionTreeClassifier()
clf.fit(X, y)

# Step 5: Predict new entry
def predict_gender(height, weight, foot_size):
    # Create a DataFrame for the new entry to match feature names
    new_data = pd.DataFrame([[height, weight, foot_size]], columns=['Height', 'Weight', 'Foot_Size'])
    prediction = clf.predict(new_data)
    return label_encoder.inverse_transform(prediction)[0]

# New entry for prediction
new_height = 5.80
new_weight = 160
new_foot_size = 9
```



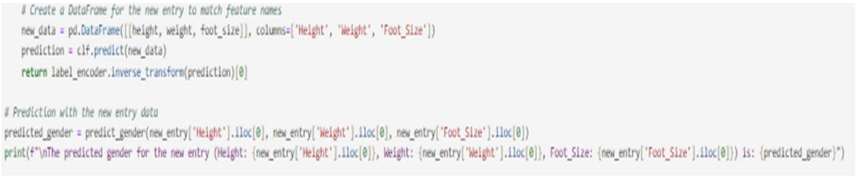
```
predicted_gender = predict_gender(new_height, new_weight, new_foot_size)
print(f"\nThe predicted gender for the new entry (Height: {new_height}, Weight: {new_weight}, Foot_Size: {new_foot_size}) is: {predicted_gender}")
```

```
Data Table:
   Gender  Height  Weight  Foot_Size
0    male    6.00     180         12
1    male    5.92     190         11
2    male    5.58     170         12
3    male    5.92     165         10
4  female    5.00     100          6
5  female    5.50     150          8
6  female    5.42     130          7
7  female    5.75     150          9

The predicted gender for the new entry (Height: 5.8, Weight: 160, Foot_Size: 9) is: female
```

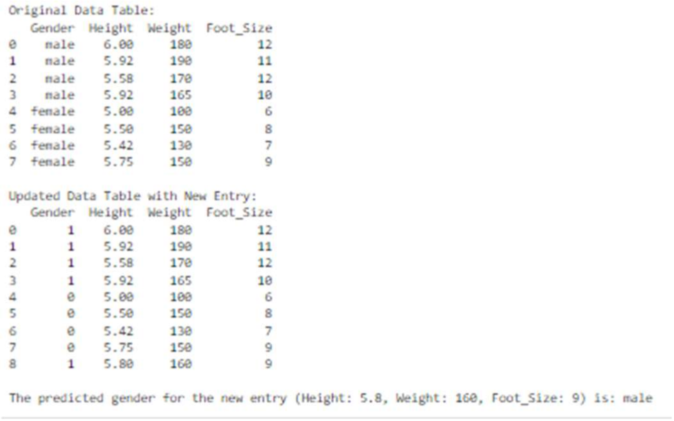
2. Implement Decision Tree using table. 1 in such a way that the new entry becomes the part of the given dataset.



```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder

# Step 1: Create the initial DataFrame
data = {
    'Gender': ['male', 'male', 'male', 'male', 'female', 'female', 'female', 'female'],
    'Height': [6.00, 5.92, 5.58, 5.92, 5.00, 5.50, 5.42, 5.75],
    'Weight': [180, 190, 170, 165, 100, 150, 130, 150],
    'Foot_Size': [12, 11, 12, 10, 6, 8, 7, 9]
}

df = pd.DataFrame(data)

# Display the original table
print("Original Data Table:")
print(df)

# Step 2: Encode the categorical data
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])  # male = 1, female = 0

# Step 3: Add the new entry to the dataset using pd.concat()
new_entry = pd.DataFrame({'Gender': ['male'], 'Height': [5.80], 'Weight': [160], 'Foot_Size': [9]})
new_entry['Gender'] = label_encoder.transform(new_entry['Gender'])  # Encode the new entry gender

df = pd.concat([df, new_entry], ignore_index=True)

# Display the updated table
print("\nUpdated Data Table with New Entry:")
print(df)

# Step 4: Split the data into features and labels
X = df[['Height', 'Weight', 'Foot_Size']]
y = df['Gender']

# Step 5: Train the Decision Tree model on the updated dataset
clf = DecisionTreeClassifier()
clf.fit(X, y)

# Confirm that the model includes the new data by predicting it again
def predict_gender(height, weight, foot_size):
```

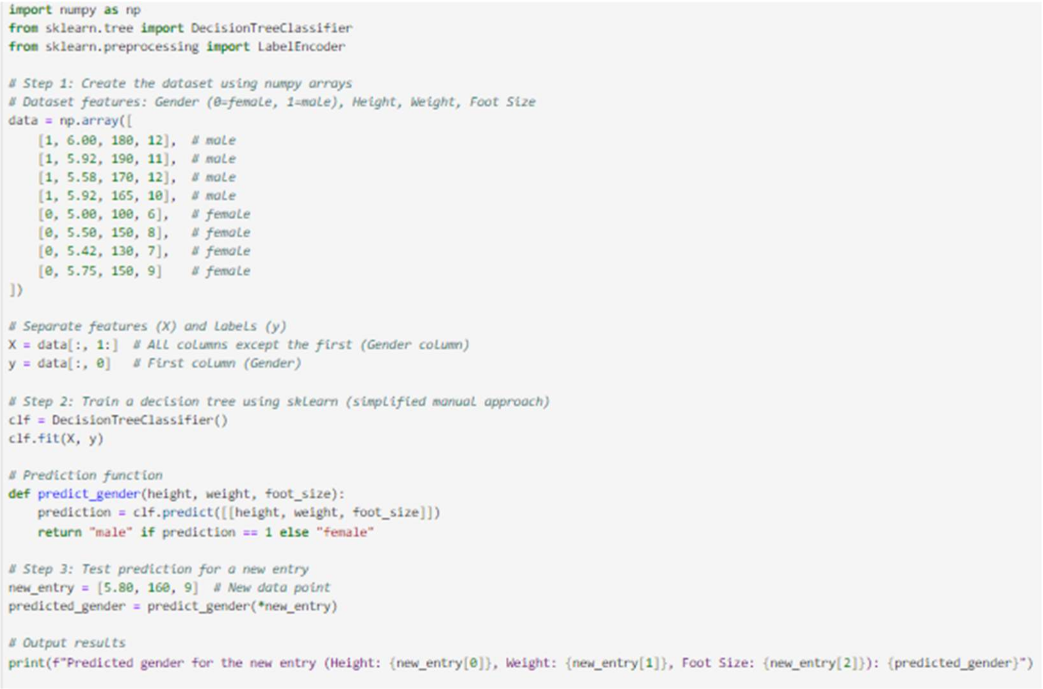


```
    # Create a DataFrame for the new entry to match feature names
    new_data = pd.DataFrame([[height, weight, foot_size]], columns=['Height', 'Weight', 'Foot_Size'])
    prediction = clf.predict(new_data)
    return label_encoder.inverse_transform(prediction)[0]

# Prediction with the new entry data
predicted_gender = predict_gender(new_entry['Height'].iloc[0], new_entry['Weight'].iloc[0], new_entry['Foot_Size'].iloc[0])
print(f"\nThe predicted gender for the new entry (Height: {new_entry['Height'].iloc[0]}, Weight: {new_entry['Weight'].iloc[0]}, Foot_Size: {new_entry['Foot_Size'].iloc[0]}) is: {predicted_gender}")
```

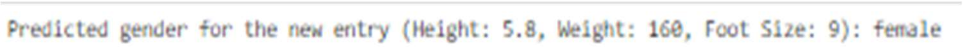


```
Original Data Table:
   Gender  Height  Weight  Foot_Size
0    male    6.00     180         12
1    male    5.92     190         11
2    male    5.58     170         12
3    male    5.92     165         10
4  female    5.00     100          6
5  female    5.50     150          8
6  female    5.42     130          7
7  female    5.75     150          9

Updated Data Table with New Entry:
   Gender  Height  Weight  Foot_Size
0       1    6.00     180         12
1       1    5.92     190         11
2       1    5.58     170         12
3       1    5.92     165         10
4       0    5.00     100          6
5       0    5.50     150          8
6       0    5.42     130          7
7       0    5.75     150          9
8       1    5.80     160          9

The predicted gender for the new entry (Height: 5.8, Weight: 160, Foot_Size: 9) is: male
```

3. Implement Decision Tree using table. 1 without the use of Pandas library. You can use numpy.



```python
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder

# Step 1: Create the dataset using numpy arrays
# Dataset features: Gender (0=female, 1=male), Height, Weight, Foot Size
data = np.array([
    [1, 6.00, 180, 12],  # male
    [1, 5.92, 190, 11],  # male
    [1, 5.58, 170, 12],  # male
    [1, 5.92, 165, 10],  # male
    [0, 5.00, 100, 6],   # female
    [0, 5.50, 150, 8],   # female
    [0, 5.42, 130, 7],   # female
    [0, 5.75, 150, 9]    # female
])

# Separate features (X) and labels (y)
X = data[:, 1:]  # All columns except the first (Gender column)
y = data[:, 0]   # First column (Gender)

# Step 2: Train a decision tree using sklearn (simplified manual approach)
clf = DecisionTreeClassifier()
clf.fit(X, y)

# Prediction function
def predict_gender(height, weight, foot_size):
    prediction = clf.predict([[height, weight, foot_size]])
    return "male" if prediction == 1 else "female"

# Step 3: Test prediction for a new entry
new_entry = [5.80, 160, 9]  # New data point
predicted_gender = predict_gender(*new_entry)

# Output results
print(f"Predicted gender for the new entry (Height: {new_entry[0]}, Weight: {new_entry[1]}, Foot Size: {new_entry[2]}): {predicted_gender}")
```



```
Predicted gender for the new entry (Height: 5.8, Weight: 160, Foot Size: 9): female
```