Lab 6 Solution key

1,2.

```
import numpy as np
import pandas as pd

# Dataset
data = {
    "Outlook": [
        "Sunny",
        "Sunny",
        "Overcast",
        "Rain",
        "Rain",
        "Rain",
        "Overcast",
        "Sunny",
        "Sunny",
        "Rain",
    ],
    "Temperature": [
        "Hot",
        "Hot",
        "Hot",
        "Mild",
        "Cool",
        "Cool",
        "Cool",
        "Mild",
        "Cool",
        "Mild",
    ],
    "Humidity": [
        "High",
        "High",
        "High",
        "High",
        "Normal",
        "Normal",
        "Normal",
        "High",
        "Normal",
        "Normal",
    ],
```

```
    "Wind": [
        "Weak",
        "Strong",
        "Weak",
        "Weak",
        "Weak",
        "Strong",
        "Strong",
        "Weak",
        "Weak",
        "Weak",
    ],
    "PlayTennis": ["No", "No", "Yes", "Yes", "Yes", "No", "Yes", "No", "Yes", "Yes"],
}
df = pd.DataFrame(data)


def entropy(target):
    vals, counts = np.unique(target, return_counts=True)
    probs = counts / len(target)
    return -np.sum(probs * np.log2(probs))


total_entropy = entropy(df["PlayTennis"])
print(f"Total Entropy: {total_entropy:.4f}\n")


def info_gain(df, feature, target_col="PlayTennis"):
    total_ent = entropy(df[target_col])
    vals, counts = np.unique(df[feature], return_counts=True)
    weighted_ent = sum(
        (counts[i] / len(df)) * entropy(df[df[feature] == vals[i]][target_col])
        for i in range(len(vals))
    )
    return total_ent - weighted_ent


for col in ["Outlook", "Temperature", "Humidity", "Wind"]:
    print(f"Information Gain ({col}): {info_gain(df, col):.4f}")

print("\n--- Text Decision Tree ---")
print("Outlook?")
print(" |-- Overcast -> Yes")
print(" |-- Rain -> Check Wind (Weak: Yes, Strong: No)")
```

```
print(" |-- Sunny -> Check Humidity (High: No, Normal: Yes)")
```

3.

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

# Data
data = {
    "Outlook": [
        "Sunny",
        "Sunny",
        "Overcast",
        "Rain",
        "Rain",
        "Overcast",
        "Rain",
        "Sunny",
        "Overcast",
        "Sunny",
    ],
    "Temperature": [
        "Hot",
        "Mild",
        "Cool",
        "Mild",
        "Cool",
        "Mild",
        "Mild",
        "Cool",
        "Hot",
        "Mild",
    ],
    "Humidity": [
        "High",
        "High",
        "Normal",
        "Normal",
        "Normal",
        "High",
        "High",
        "Normal",
        "Normal",
        "Normal",
```

```
    ],
    "Wind": [
        "Weak",
        "Strong",
        "Weak",
        "Weak",
        "Strong",
        "Strong",
        "Weak",
        "Weak",
        "Strong",
        "Weak",
    ],
    "Picnic": ["No", "No", "Yes", "Yes", "No", "Yes", "Yes", "Yes", "Yes", "Yes"],
}
df = pd.DataFrame(data)

# One-hot encoding
X = pd.get_dummies(df.drop("Picnic", axis=1))
y = df["Picnic"]

clf = DecisionTreeClassifier(criterion="entropy")
clf.fit(X, y)

plt.figure(figsize=(10, 6))
plot_tree(
    clf, feature_names=X.columns, class_names=clf.classes_, filled=True
)
plt.show()
```

4.

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {
        "Area": [850, 900, 1000, 1200, 1500, 1700, 2000],
        "Rooms": [2, 3, 3, 4, 4, 5, 5],
        "Price": [100, 120, 150, 200, 250, 275, 310],
    }
)


def calc_split_mse(df, feature, threshold):
```

```
    left = df[df[feature] <= threshold]["Price"]
    right = df[df[feature] > threshold]["Price"]

    if len(left) == 0 or len(right) == 0:
        return float("inf")

    mse_left = np.mean((left - np.mean(left)) ** 2) if len(left) > 0 else 0
    mse_right = np.mean((right - np.mean(right)) ** 2) if len(right) > 0 else 0

    return (len(left) * mse_left + len(right) * mse_right) / len(df)


best_mse = float("inf")
best_split = None

for col in ["Area", "Rooms"]:
    thresholds = (df[col].iloc[:-1].values + df[col].iloc[1:].values) / 2
    for t in np.unique(thresholds):
        mse = calc_split_mse(df, col, t)
        if mse < best_mse:
            best_mse = mse
            best_split = (col, t)

print(
    f"Best Split -> Feature: {best_split[0]}, Threshold: {best_split[1]}, Min MSE: {best_mse:.2f}"
)
```

5.

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.tree import DecisionTreeRegressor, plot_tree

df = pd.DataFrame(
    {
        "Engine Size": [1.2, 1.5, 1.8, 2.0, 2.2, 2.5, 3.0, 3.5, 4.0],
        "Horsepower": [70, 90, 110, 130, 150, 170, 200, 250, 300],
        "Price": [12, 15, 20, 25, 28, 35, 45, 60, 70],
    }
)

X = df[["Engine Size", "Horsepower"]]
y = df["Price"]

model = DecisionTreeRegressor()
```

```
model.fit(X, y)

pred = model.predict([[2.5, 180]])[0]
print(f"Predicted Price for (Engine Size=2.5, HP=180): ${pred:.2f}k")

plt.figure(figsize=(10, 6))
plot_tree(model, feature_names=X.columns, filled=True)
plt.show()
```