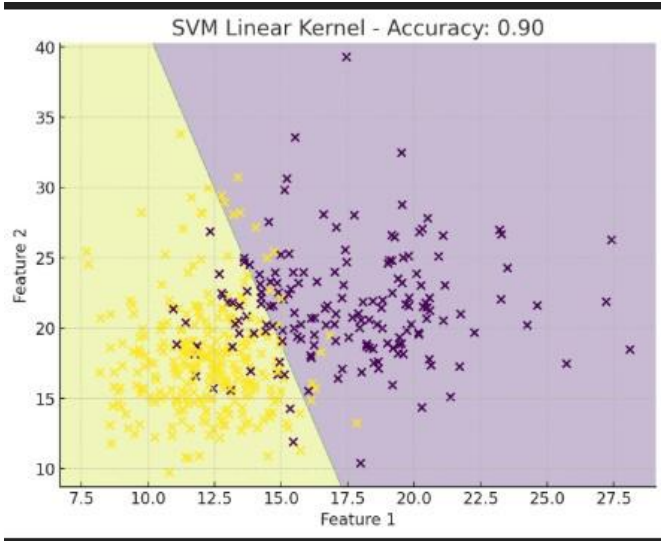
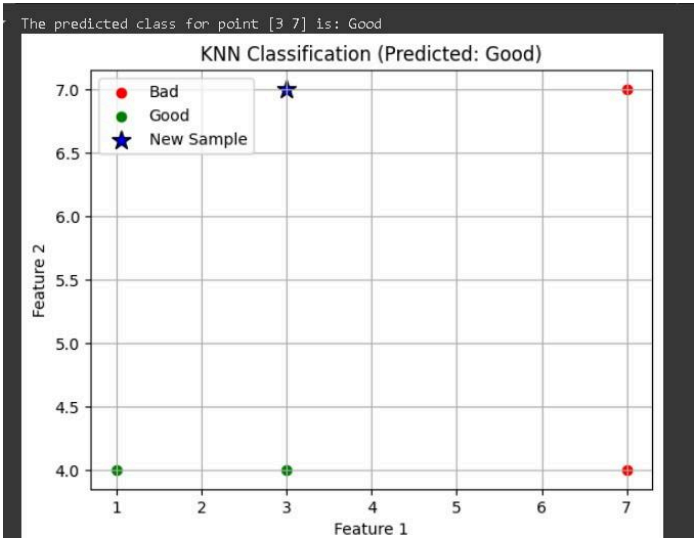
```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
data = datasets.load_breast_cancer()
X, y = data.data, data.target
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = SVC(kernel='linear')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

Output :



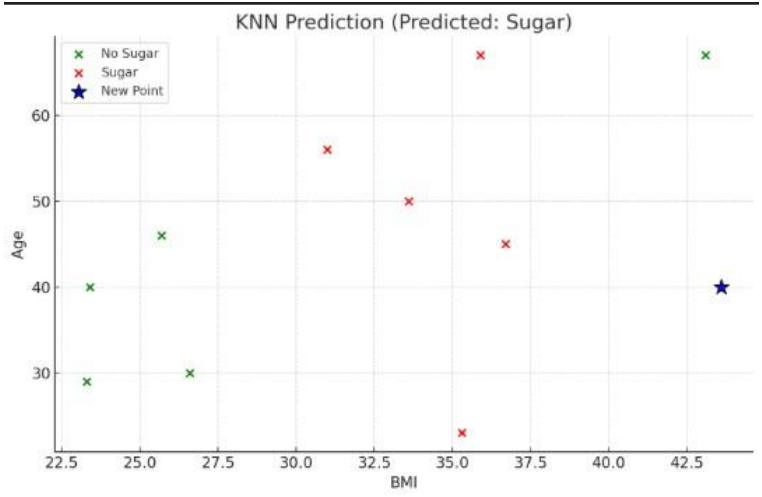
```
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
X = np.array([
    [7, 7],
    [7, 4],
    [3, 4],
    [1, 4]
])

y = np.array(['Bad', 'Bad', 'Good', 'Good'])
new_sample = np.array([[3, 7]])
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X, y)
predicted_class = knn.predict(new_sample)
print(f"The predicted class for point {new_sample[0]} is:
{predicted_class[0]}")
```

Output :



```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
bmi = np.array([33.6, 26.6, 23.4, 43.1, 35.3, 35.9, 36.7, 25.7, 23.3,
31])
age = np.array([50, 30, 40, 67, 23, 67, 45, 46, 29, 56])
sugar = np.array([1, 0, 0, 0, 1, 1, 1, 0, 0, 1])
X = np.column_stack((bmi, age))
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X, sugar)
new_point = np.array([[43.6, 40]])
prediction = knn.predict(new_point)
print(f"The predicted sugar level for BMI=43.6 and Age=40 is:
{prediction[0]}")
```

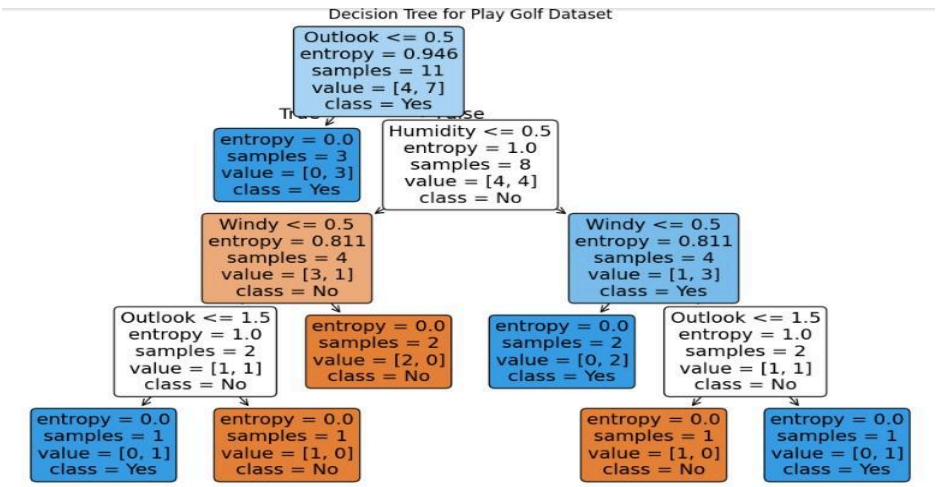
Output :



```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
# Create the Play Golf dataset
data = {
    "Outlook": ["Sunny", "Sunny", "Overcast", "Rain", "Rain", "Rain",
"Overcast",
                "Sunny", "Sunny", "Rain", "Sunny", "Overcast", "Overcast",
"Rain"],
    "Temp": ["Hot", "Hot", "Hot", "Mild", "Cool", "Cool", "Cool",
             "Mild", "Cool", "Mild", "Mild", "Mild", "Hot", "Mild"],
```

```
    "Humidity": ["High", "High", "High", "High", "Normal", "Normal",
"Normal",
                 "High", "Normal", "Normal", "Normal", "High", "Normal",
"High"],
    "Windy": [False, True, False, False, False, True, True,
              False, False, False, True, True, False, True],
    "Play": ["No", "No", "Yes", "Yes", "Yes", "No", "Yes",
             "No", "Yes", "Yes", "Yes", "Yes", "Yes", "No"]
}
df = pd.DataFrame(data)
df
encoder = LabelEncoder()
df["Outlook"] = encoder.fit_transform(df["Outlook"])
df["Temp"] = encoder.fit_transform(df["Temp"])
df["Humidity"] = encoder.fit_transform(df["Humidity"])
df["Windy"] = df["Windy"].astype(int)
df["Play"] = encoder.fit_transform(df["Play"])
X = df[["Outlook", "Temp", "Humidity", "Windy"]]
y = df["Play"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
clf = DecisionTreeClassifier(criterion="entropy", random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on Test Data: {accuracy:.2f}")
tree_rules = export_text(clf, feature_names=list(X.columns))
print("\nDecision Tree Rules:")
print(tree_rules)
plt.figure(figsize=(12, 8))
plot_tree(clf, feature_names=list(X.columns),
class_names=encoder.classes_, filled=True, rounded=True)
plt.title("Decision Tree for Play Golf Dataset")
plt.show()
```

Output:



```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from collections import Counter
import matplotlib.pyplot as plt
data = {
    "Outlook": ["Sunny", "Sunny", "Overcast", "Rain", "Rain", "Rain",
"Overcast",
                "Sunny", "Sunny", "Rain", "Sunny", "Overcast", "Overcast",
"Rain"],
    "Temp": ["Hot", "Hot", "Hot", "Mild", "Cool", "Cool", "Cool",
             "Mild", "Cool", "Mild", "Mild", "Mild", "Hot", "Mild"],
    "Humidity": ["High", "High", "High", "High", "Normal", "Normal",
"Normal",
                 "High", "Normal", "Normal", "Normal", "High", "Normal",
"High"],
    "Windy": [False, True, False, False, False, True, True,
```

```
                False, False, False, True, True, False, True],
    "Play": ["No", "No", "Yes", "Yes", "Yes", "No", "Yes",
             "No", "Yes", "Yes", "Yes", "Yes", "Yes", "No"]
}
df = pd.DataFrame(data)
outlook_encoder = LabelEncoder()
temp_encoder = LabelEncoder()
humidity_encoder = LabelEncoder()
play_encoder = LabelEncoder()
df["Outlook"] = outlook_encoder.fit_transform(df["Outlook"])
df["Temp"] = temp_encoder.fit_transform(df["Temp"])
df["Humidity"] = humidity_encoder.fit_transform(df["Humidity"])
df["Windy"] = df["Windy"].astype(int)
df["Play"] = play_encoder.fit_transform(df["Play"])
X = df[["Outlook", "Temp", "Humidity", "Windy"]]
y = df["Play"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
model = RandomForestClassifier(n_estimators=5, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on Test Data: {accuracy:.2f}")
new_data = pd.DataFrame({
    "Outlook": [outlook_encoder.transform(["Sunny"])[0]],
    "Temp": [temp_encoder.transform(["Cool"])[0]],
    "Humidity": [humidity_encoder.transform(["High"])[0]],
    "Windy": [0]
})
print("\nIndividual tree predictions for new data:")
votes = []
label_map = {i: label for i, label in enumerate(play_encoder.classes_)}
for i, tree in enumerate(model.estimators_):
    pred = tree.predict(new_data)[0]
    votes.append(pred)
    print(f"Tree {i+1}: {label_map[pred]}")
majority_vote = Counter(votes).most_common(1)[0][0]
print(f"\nMajority Voting Result for new data:
{label_map[majority_vote]}")
```

Output :

```
Accuracy on Test Data: 0.67

Individual tree predictions for new data:
Tree 1: Yes
Tree 2: No
Tree 3: No
Tree 4: Yes
Tree 5: Yes
```