Double-click (or enter) to edit

Import Libraries

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import precision_recall_curve, roc_curve, auc
```

Data Loading and Preprocessing

```
# Load the dataset
data = pd.read_csv('water_potability.csv')  # Update path if needed

# Handle missing values by replacing them with the median
data = data.fillna(data.median())

# Define features and target variable
X = data.drop(columns='Potability')  # Features
y = data['Potability']  # Target (0 = Not Drinkable, 1 = Drinkable)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Feature Scaling

```
# Standardize features (important for KNN and SVM)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Model Initialization, Training, and Prediction

```
# Initialize and fit the KNN model
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
y_pred_prob_knn = knn.predict_proba(X_test)[:, 1]

# Initialize and fit the SVM model
svm = SVC(kernel='rbf', C=1, gamma='scale', probability=True)
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)
y_pred_prob_svm = svm.predict_proba(X_test)[:, 1]

# Initialize and fit the Decision Tree model
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
y_pred_prob_dt = dt.predict_proba(X_test)[:, 1]
```

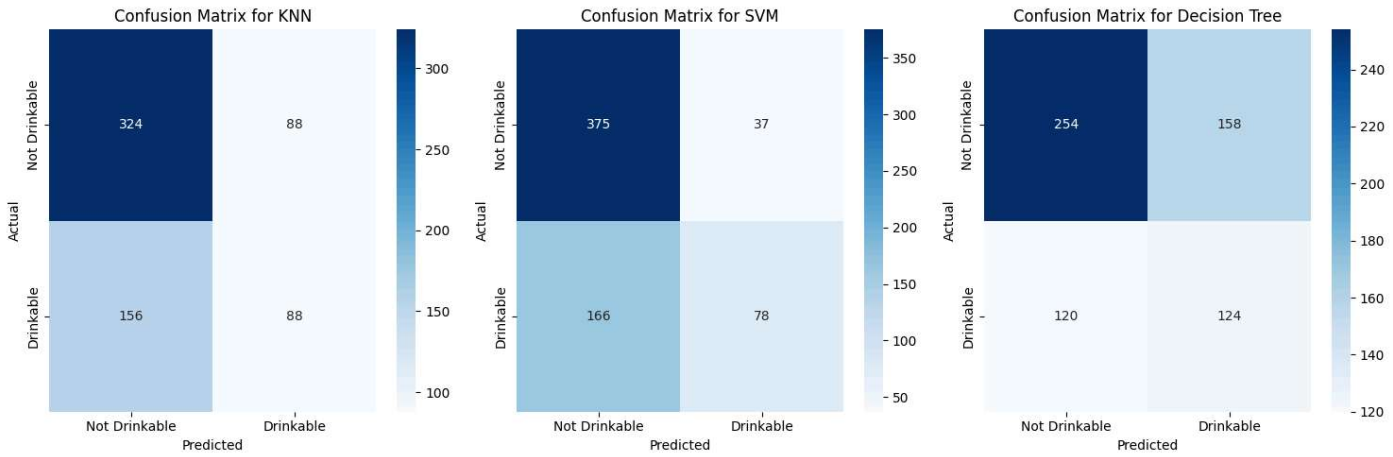
Model Evaluation

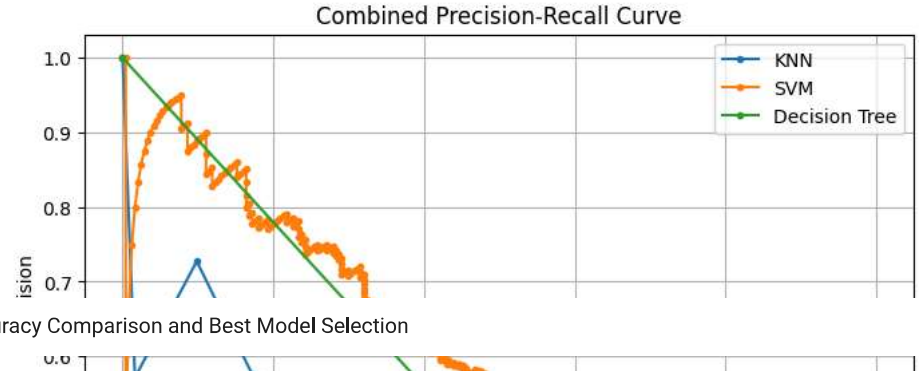
```
# Accuracy Scores
accuracy_knn = accuracy_score(y_test, y_pred_knn)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
accuracy_dt = accuracy_score(y_test, y_pred_dt)

print("KNN Accuracy:", accuracy_knn)
print("SVM Accuracy:", accuracy_svm)
print("Decision Tree Accuracy:", accuracy_dt)

# Classification Reports
print("\nKNN Classification Report:\n", classification_report(y_test, y_pred_knn))
print("\nSVM Classification Report:\n", classification_report(y_test, y_pred_svm))
print("\nDecision Tree Classification Report:\n", classification_report(y_test, y_pred_dt))
```

```
KNN Accuracy: 0.6280487804878049
SVM Accuracy: 0.6905487804878049
Decision Tree Accuracy: 0.5762195121951219

KNN Classification Report:
               precision    recall  f1-score   support

           0       0.68      0.79      0.73       412
           1       0.50      0.36      0.42       244

    accuracy                           0.63       656
   macro avg       0.59      0.57      0.57       656
weighted avg       0.61      0.63      0.61       656


SVM Classification Report:
               precision    recall  f1-score   support

           0       0.69      0.91      0.79       412
           1       0.68      0.32      0.43       244

    accuracy                           0.69       656
   macro avg       0.69      0.61      0.61       656
weighted avg       0.69      0.69      0.66       656


Decision Tree Classification Report:
               precision    recall  f1-score   support

           0       0.68      0.62      0.65       412
           1       0.44      0.51      0.47       244

    accuracy                           0.58       656
   macro avg       0.56      0.56      0.56       656
weighted avg       0.59      0.58      0.58       656
```

Visualization Functions

```
# Function to plot combined confusion matrices
def plot_combined_confusion_matrix(y_test, y_preds, model_names):
    cm_list = [confusion_matrix(y_test, y_pred) for y_pred in y_preds]
    fig, axes = plt.subplots(1, len(model_names), figsize=(15, 5))

    for ax, cm, model_name in zip(axes, cm_list, model_names):
        sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", ax=ax,
                    xticklabels=['Not Drinkable', 'Drinkable'],
                    yticklabels=['Not Drinkable', 'Drinkable'])
        ax.set_title(f'Confusion Matrix for {model_name}')
        ax.set_xlabel('Predicted')
        ax.set_ylabel('Actual')

    plt.tight_layout()
    plt.show()

# Plot combined confusion matrices
plot_combined_confusion_matrix(y_test, [y_pred_knn, y_pred_svm, y_pred_dt], ["KNN", "SVM", "Decision Tree"])

# Function to plot combined precision-recall curves
def plot_combined_precision_recall(y_test, y_pred_probs, model_names):
    plt.figure(figsize=(8, 5))

    for y_pred_prob, model_name in zip(y_pred_probs, model_names):
        precision, recall, _ = precision_recall_curve(y_test, y_pred_prob)
        plt.plot(recall, precision, marker='.', label=model_name)

    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.title('Combined Precision-Recall Curve')
    plt.legend()
    plt.grid()
    plt.show()

# Plot combined precision-recall curves
plot_combined_precision_recall(y_test, [y_pred_prob_knn, y_pred_prob_svm, y_pred_prob_dt], ["KNN", "SVM", "Decision Tree"])

# Function to plot combined ROC curves
def plot_combined_roc_curve(y_test, y_pred_probs, model_names):
    plt.figure(figsize=(8, 5))

    for y_pred_prob, model_name in zip(y_pred_probs, model_names):
        fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, marker='.', label=f'{model_name} (AUC = {roc_auc:.2f})')

    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Combined ROC Curve')
    plt.legend()
    plt.grid()
    plt.show()

# Plot combined ROC curves
plot_combined_roc_curve(y_test, [y_pred_prob_knn, y_pred_prob_svm, y_pred_prob_dt], ["KNN", "SVM", "Decision Tree"])
```





Accuracy Comparison and Best Model Selection

```
# Final Accuracy Comparison
print("\nComparison of Model Accuracies:")
print("KNN Accuracy:", accuracy_knn)
print("SVM Accuracy:", accuracy_svm)
print("Decision Tree Accuracy:", accuracy_dt)

# Determine the best-performing model
accuracies = {"KNN": accuracy_knn, "SVM": accuracy_svm, "Decision Tree": accuracy_dt}
best_model = max(accuracies, key=accuracies.get)
print(f"The best-performing model is {best_model} with an accuracy of {accuracies[best_model]:.2f}.")
```

```
Comparison of Model Accuracies:
KNN Accuracy: 0.6280487804878049
SVM Accuracy: 0.6905487804878049
Decision Tree Accuracy: 0.5762195121951219
The best-performing model is SVM with an accuracy of 0.69.
```