Documentation

Step 1: Importing Libraries and Setting Up Environment
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, ConfusionMatrixDisplay

# Suppress warnings for cleaner output
import warnings
warnings.filterwarnings("ignore")
```

Step 2: Loading the Data
Explanation
This step loads training and testing data from files. Adjust file paths as needed.
```python
# Load data
try:
    X_train = pd.read_csv('/content/X_train.txt', sep="\s+", header=None)
    Y_train = pd.read_csv('/content/y_train.txt', sep="\s+", header=None)
    X_test = pd.read_csv('/content/X_test.txt', sep="\s+", header=None)
    Y_test = pd.read_csv('/content/y_test.txt', sep="\s+", header=None)
except Exception as e:
    print(f"Error loading files: {e}")
    raise

# Preview data
print("X_train sample:")
print(X_train.head())
print("Y_train sample:")
print(Y_train.head())
```

Step 3: Preprocessing the Data

Explanation
This step:
1. Assigns column names to the data.
2. Handles missing values by imputing the mean.
3. Ensures the data is numeric for ML models.
```python
# Assign column names
num_features = X_train.shape[1]
X_train.columns = [f'feature_{i}' for i in range(num_features)]
X_test.columns = [f'feature_{i}' for i in range(num_features)]
Y_train.columns = ['target']
Y_test.columns = ['target']

# Handle missing values
imputer = SimpleImputer(strategy='mean')
X_train = pd.DataFrame(imputer.fit_transform(X_train), columns=X_train.columns)
X_test = pd.DataFrame(imputer.transform(X_test), columns=X_test.columns)
```

Step 4: Exploratory Data Analysis (EDA)
Explanation
Visualize the data to understand distributions and potential issues.
```python
# Visualize target distribution
sns.countplot(x='target', data=Y_train)
plt.title("Target Distribution in Training Data")
plt.show()

# Correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(X_train.corr(), cmap="coolwarm", annot=False)
plt.title("Feature Correlation Heatmap")
plt.show()
```

Step 5: Training and Evaluating Models
Explanation
Train multiple models and evaluate their performance on test data.
```python
# Initialize models
models = {
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'Random Forest': RandomForestClassifier(random_state=42),
    'Logistic Regression': LogisticRegression(max_iter=1000, random_state=42),
```

```python
    'AdaBoost': AdaBoostClassifier(random_state=42)
}

# Train and evaluate models
results = []
for name, model in models.items():
    # Train
    model.fit(X_train, Y_train['target'])

    # Predict
    predictions = model.predict(X_test)

    # Evaluate
    acc = accuracy_score(Y_test['target'], predictions)
    prec = precision_score(Y_test['target'], predictions, average='weighted')
    rec = recall_score(Y_test['target'], predictions, average='weighted')
    f1 = f1_score(Y_test['target'], predictions, average='weighted')

    results.append({
        'Model': name,
        'Accuracy': acc,
        'Precision': prec,
        'Recall': rec,
        'F1 Score': f1
    })

    # Confusion Matrix
    cm = confusion_matrix(Y_test['target'], predictions)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot()
    plt.title(f"Confusion Matrix - {name}")
    plt.show()

# Results DataFrame
results_df = pd.DataFrame(results)
print(results_df)
```

Step 6: Visualizing Model Performance
Explanation
Compare model performance using a bar chart.

```python
# Plot performance
```

```
results_df.set_index('Model')[['Accuracy', 'Precision', 'Recall', 'F1 Score']].plot(kind='bar', figsize=(10, 6))
plt.title("Model Performance Comparison")
plt.ylabel("Score")
plt.ylim(0, 1)
plt.xticks(rotation=45)
plt.legend(loc='lower right')
plt.show()
```

Step 7: Conclusion
Explanation
- Discuss which model performed best and why.
- Suggest next steps for improving performance, such as hyperparameter tuning or additional feature engineering.
```markdown
### Observations
1. Logistic Regression achieved the highest accuracy and F1-score, suggesting it handles this dataset well.
2. AdaBoost underperformed, indicating it might not be well-suited for this data.
3. Feature scaling and hyperparameter tuning could further improve performance.
```