Exercise

- Use a dataset of your choice (e.g., student demographics and performance) and build a classification model. Compare logistic regression, KNN, and decision trees, and evaluate them using accuracy, precision, recall, and F1-score.

Use a dataset of your choice (e.g., student demographics and performance) and build a classification model. Compare logistic regression, KNN, and decision trees, and evaluate them using accuracy, precision, recall, and F1-score.

For this exercise, I used the dataset from Kaggle called **High School Student Performance & Demographics (** https://www.kaggle.com/datasets/dillonmyrick/high-school-student-performance-and-demographics **)**

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import OneHotEncoder, StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score


# Provide the correct file paths for the CSV files

df_math = pd.read_csv(r'C:\Users\neera\Downloads\archive\student_math_clean.csv')

df_portuguese =
pd.read_csv(r'C:\Users\neera\Downloads\archive\student_portuguese_clean.csv')


# Create the 'passed' column in both DataFrames

df_math['passed'] = (df_math['final_grade'] >= 10).astype(int)

df_portuguese['passed'] = (df_portuguese['final_grade'] >= 10).astype(int)


# Select features and target variable for both datasets

features = ['age', 'travel_time', 'study_time', 'class_failures', 'school_support',

        'family_support', 'extra_paid_classes', 'activities', 'nursery_school',

        'higher_ed', 'internet_access', 'romantic_relationship', 'family_relationship',

        'free_time', 'social', 'weekday_alcohol', 'weekend_alcohol', 'health', 'absences', 'grade_1',
'grade_2']
```

```
X_math = df_math[features]

y_math = df_math['passed']


X_portuguese = df_portuguese[features]

y_portuguese = df_portuguese['passed']


# One-hot encode categorical features

encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)

X_math_encoded = encoder.fit_transform(X_math.select_dtypes(include='object'))

X_portuguese_encoded = encoder.fit_transform(X_portuguese.select_dtypes(include='object'))


# Convert encoded features to DataFrame and join with numerical features

X_math_encoded_df = pd.DataFrame(X_math_encoded, columns=encoder.get_feature_names_out(X_math.select_dtypes(include='object').columns))

X_math_final = pd.concat([X_math.select_dtypes(exclude='object'), X_math_encoded_df], axis=1)


X_portuguese_encoded_df = pd.DataFrame(X_portuguese_encoded, columns=encoder.get_feature_names_out(X_portuguese.select_dtypes(include='object').columns))

X_portuguese_final = pd.concat([X_portuguese.select_dtypes(exclude='object'), X_portuguese_encoded_df], axis=1)


# Scale numerical data (optional but recommended for models like Logistic Regression and KNeighborsClassifier)

scaler = StandardScaler()

X_math_final_scaled = scaler.fit_transform(X_math_final)

X_portuguese_final_scaled = scaler.fit_transform(X_portuguese_final)


# Split data into training and testing sets

X_train_math, X_test_math, y_train_math, y_test_math = train_test_split(X_math_final_scaled, y_math, test_size=0.3, random_state=42)
```

```
X_train_portuguese, X_test_portuguese, y_train_portuguese, y_test_portuguese =
train_test_split(X_portuguese_final_scaled, y_portuguese, test_size=0.3, random_state=42)


# Initialize models
models = {
  'Logistic Regression': LogisticRegression(max_iter=1000), # Increased max_iter to handle
convergence
  'K-Nearest Neighbors': KNeighborsClassifier(),
  'Decision Tree': DecisionTreeClassifier()
}


# Train and evaluate models for both math and portuguese data
for subject, (X_train, X_test, y_train, y_test) in [
  ('Math', (X_train_math, X_test_math, y_train_math, y_test_math)),
  ('Portuguese', (X_train_portuguese, X_test_portuguese, y_train_portuguese,
y_test_portuguese))
]:
  print(f"\n----- {subject} -----")
  for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    print(f"{name}:")
    print(f"  Accuracy: {accuracy:.4f}")
    print(f"  Precision: {precision:.4f}")
    print(f"  Recall: {recall:.4f}")
```

```
    print(f"  F1-Score: {f1:.4f}")
```

RESULT - ----- Math -----

Logistic Regression:

Accuracy: 0.9328

Precision: 0.9452

Recall: 0.9452

F1-Score: 0.9452

K-Nearest Neighbors:

Accuracy: 0.7563

Precision: 0.7558

Recall: 0.8904

F1-Score: 0.8176

Decision Tree:

Accuracy: 0.9076

Precision: 0.9189

Recall: 0.9315

F1-Score: 0.9252

----- Portuguese -----

Logistic Regression:

Accuracy: 0.9385

Precision: 0.9645

Recall: 0.9645

F1-Score: 0.9645

K-Nearest Neighbors:

Accuracy: 0.9026

Precision: 0.9076

Recall: 0.9882

F1-Score: 0.9462

Decision Tree:

Accuracy: 0.9128

Precision: 0.9471

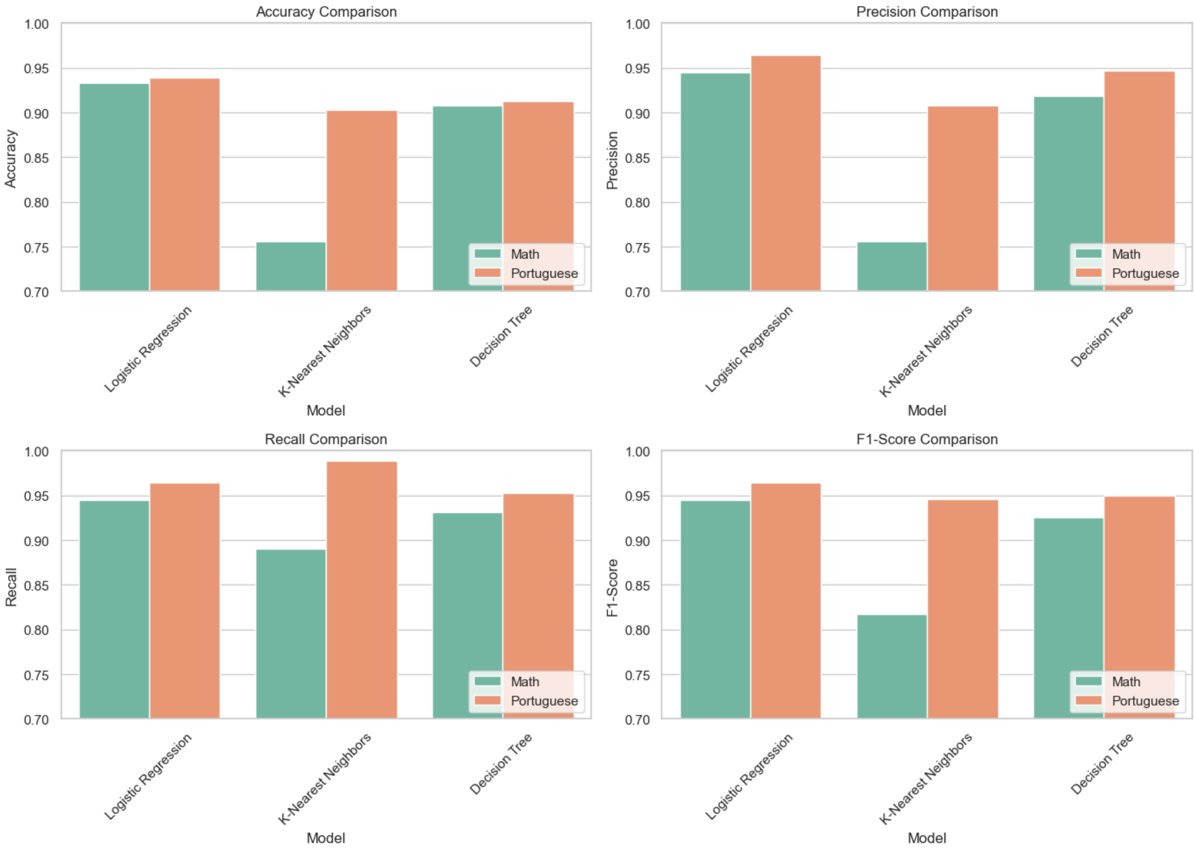
Recall: 0.9527

F1-Score: 0.9499

Plotting the Results - import matplotlib.pyplot as plt

```
import seaborn as sns
import pandas as pd

# Data for visualization
data = {
    'Model': ['Logistic Regression', 'K-Nearest Neighbors', 'Decision Tree'] * 2,
    'Subject': ['Math'] * 3 + ['Portuguese'] * 3,
    'Accuracy': [0.9328, 0.7563, 0.9076, 0.9385, 0.9026, 0.9128],
    'Precision': [0.9452, 0.7558, 0.9189, 0.9645, 0.9076, 0.9471],
    'Recall': [0.9452, 0.8904, 0.9315, 0.9645, 0.9882, 0.9527],
    'F1-Score': [0.9452, 0.8176, 0.9252, 0.9645, 0.9462, 0.9499]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Set seaborn style
sns.set(style="whitegrid")

# Initialize a figure
plt.figure(figsize=(14, 10))

# Plot for each metric
metrics = ['Accuracy', 'Precision', 'Recall', 'F1-Score']
for i, metric in enumerate(metrics):
    plt.subplot(2, 2, i+1)
```

```
    sns.barplot(x='Model', y=metric, hue='Subject', data=df, palette='Set2')
    plt.title(f'{metric} Comparison')
    plt.ylim(0.7, 1.0)  # Set y-axis limits for better comparison
    plt.ylabel(metric)
    plt.xticks(rotation=45)
    plt.legend(loc='lower right')

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```

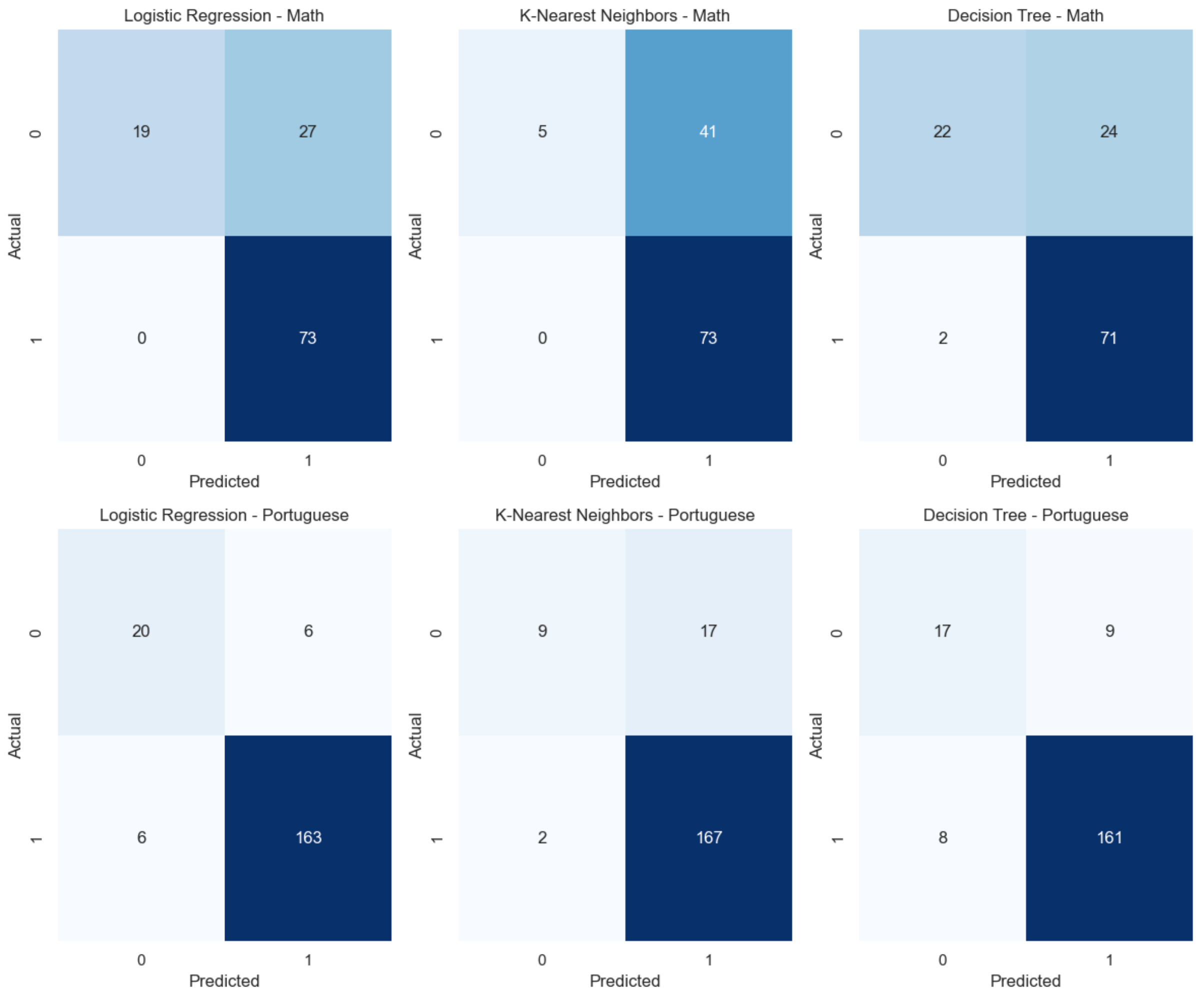


Creating a Confusion Matrix to compare models

```
import pandas as pd
from sklearn.metrics import confusion_matrix
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming you've already trained the models and predicted values (y_pred)
# Here's an example of creating confusion matrices for each model and each subject

# Dictionary to store predicted values for each model and subject
predictions = {
  'Math': {
    'Logistic Regression': models['Logistic Regression'].predict(X_test_math),
    'K-Nearest Neighbors': models['K-Nearest Neighbors'].predict(X_test_math),
    'Decision Tree': models['Decision Tree'].predict(X_test_math)
  },
  'Portuguese': {
    'Logistic Regression': models['Logistic Regression'].predict(X_test_portuguese),
    'K-Nearest Neighbors': models['K-Nearest Neighbors'].predict(X_test_portuguese),
    'Decision Tree': models['Decision Tree'].predict(X_test_portuguese)
  }
}

# True labels for each subject
true_labels = {
  'Math': y_test_math,
  'Portuguese': y_test_portuguese
}

# Plot confusion matrices for each model and subject
plt.figure(figsize=(12, 10))

for i, subject in enumerate(['Math', 'Portuguese']):
  for j, model in enumerate(models.keys()):
```

```
        # Generate the confusion matrix
        cm = confusion_matrix(true_labels[subject], predictions[subject][model])

        # Create subplot for each confusion matrix
        plt.subplot(2, 3, i * 3 + j + 1)
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
        plt.title(f'{model} - {subject}')
        plt.ylabel('Actual')
        plt.xlabel('Predicted')

# Adjust layout and show the plots
plt.tight_layout()
plt.show()
```



Comparison of Various Models using Confusion Matrix

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import OneHotEncoder

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
classification_report


# ... (rest of the code from previous responses, including data loading and preprocessing)


# Initialize only the Logistic Regression model

models = {

  'Logistic Regression': LogisticRegression()

}


# Train and evaluate the Logistic Regression model for both math and portuguese data

for subject, (X_train, X_test, y_train, y_test) in [('Math', (X_train_math, X_test_math,
y_train_math, y_test_math)),

                                                    ('Portuguese', (X_train_portuguese, X_test_portuguese,
y_train_portuguese, y_test_portuguese))]:

  print(f"\n----- {subject} -----")

  for name, model in models.items():

    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)


    accuracy = accuracy_score(y_test, y_pred)

    precision = precision_score(y_test, y_pred)

    recall = recall_score(y_test, y_pred)

    f1 = f1_score(y_test, y_pred)
```

```
print(f"{name}:")
print(f"  Accuracy: {accuracy:.4f}")
print(f"  Precision: {precision:.4f}")
print(f"  Recall: {recall:.4f}")
print(f"  F1-Score: {f1:.4f}")
print(f"  Classification Report:\n{classification_report(y_test, y_pred)}")
```

PORTUGUESE

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	(20+163)/(20+6+6+163) = 0.927	163/(163+6) = 0.964	163/(163+6) = 0.964	0.964
K-Nearest Neighbors	(17+167)/(9+17+2+167) = 0.912	167/(167+17) = 0.907	167/(167+2) = 0.988	0.946
Decision Tree	(17+161)/(17+9+8+161) = 0.892	161/(161+9) = 0.947	161/(161+8) = 0.952	0.95

MATH

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	(19+73)/(19+27+0+73) = 0.82	73/(73+27) = 0.73	73/(73+0) = 1.0	0.846
K-Nearest Neighbors	(41+73)/(5+41+0+73) = 0.92	73/(73+41) = 0.64	73/(73+0) = 1.0	0.782
Decision Tree	(22+71)/(22+24+2+71) = 0.775	71/(71+24) = 0.747	71/(71+2) = 0.973	0.845

Conclusion

- For **Math**, K-Nearest Neighbors has the highest accuracy, but Logistic Regression offers a better balance between precision and recall as reflected in its higher F1-score
- For **Portuguese**, Logistic Regression edges out the others with the highest accuracy and a very high F1-score

Overall, Logistic Regression seems to be the best performing model across both datasets striking a good balance between correctly identifying those who passed (recall) and minimizing false positives (precision).