In [2]:

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
```

In [4]:

```python
# Load the CSV file (since it's in the same folder)
df = pd.read_csv('StudentsPerformance.csv')

# Preview the dataset
df.head()
```

Out[4]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

In [6]:

```python
# Label encode categorical columns
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

In [8]:

```python
# Calculate average score
df['average_score'] = df[['math score', 'reading score', 'writing score']].mean(axi

# Create binary classification target
df['pass'] = (df['average_score'] >= 60).astype(int)

# Drop the average column to avoid data leakage
df.drop(columns=['average_score'], inplace=True)
```

In [10]:

```python
# Separate features and target
X = df.drop('pass', axis=1)
y = df['pass']
```

```
# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [14]:

```
lr_model = LogisticRegression(max_iter=1000, random_state=42)
lr_model.fit(X_train_scaled, y_train)
y_pred_lr = lr_model.predict(X_test_scaled)
```

In [16]:

```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_scaled, y_train)
y_pred_knn = knn_model.predict(X_test_scaled)
```

In [18]:

```
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train_scaled, y_train)
y_pred_dt = dt_model.predict(X_test_scaled)
```

In [20]:

```
print("📊 Logistic Regression:\n", classification_report(y_test, y_pred_lr))
print("📊 k-NN:\n", classification_report(y_test, y_pred_knn))
print("📊 Decision Tree:\n", classification_report(y_test, y_pred_dt))
```

```
📊 Logistic Regression:
               precision    recall  f1-score   support

           0       1.00      0.98      0.99        62
           1       0.99      1.00      1.00       138

    accuracy                           0.99       200
   macro avg       1.00      0.99      0.99       200
weighted avg       1.00      0.99      0.99       200

📊 k-NN:
               precision    recall  f1-score   support

           0       0.95      0.84      0.89        62
           1       0.93      0.98      0.95       138

    accuracy                           0.94       200
   macro avg       0.94      0.91      0.92       200
weighted avg       0.94      0.94      0.93       200

📊 Decision Tree:
               precision    recall  f1-score   support

           0       0.94      0.95      0.94        62
           1       0.98      0.97      0.97       138

    accuracy                           0.96       200
   macro avg       0.96      0.96      0.96       200
weighted avg       0.97      0.96      0.97       200
```

In [ ]: