How to Train, Tune, and Evaluate Decision Tree and KNN Models for Student Dropout Prediction

1. Introduction

This guide walks you through the process of training, tuning, and evaluating two classification algorithms — Decision Tree and K-Nearest Neighbors (KNN) — to predict whether a student is at risk of dropping out.

Our dataset comes from the UCI Machine Learning Repository – Predict Students Dropout and Academic Success, containing 4,424 student records and 36 features across demographics, academics, and socio-economic indicators.

The workflow is tailored for CMPT 310 students who want to understand:

- How to prepare a dataset for model training
- How to use Stratified K-Fold Cross-Validation with GridSearchCV for hyperparameter tuning
- How to interpret classification reports and confusion matrices

2. Environment Setup

1. Set up your Environment:

```
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

2. Navigate to the src Folder & Data Cleanining:

```
cd src
python data_clean.py
```

3. Train Baseline Models

```
python dt_train_model.py
python knn_train_model.py
```

4. Tune Models

```
python tune_models.py
```

5. Evaluatie Baseline Models

```
python evaluate_baseline_model.py
```

6. Evaluate Tuned Models

```
python evaluate_tune_model.py
```

3. Dataset Preprocessing (Brief Overview)

1. We convert the dataset from 3 classes to binary classification:

```
data['Target'] = data['Target'].map({0: 0, 1: 1, 2: 1})
```

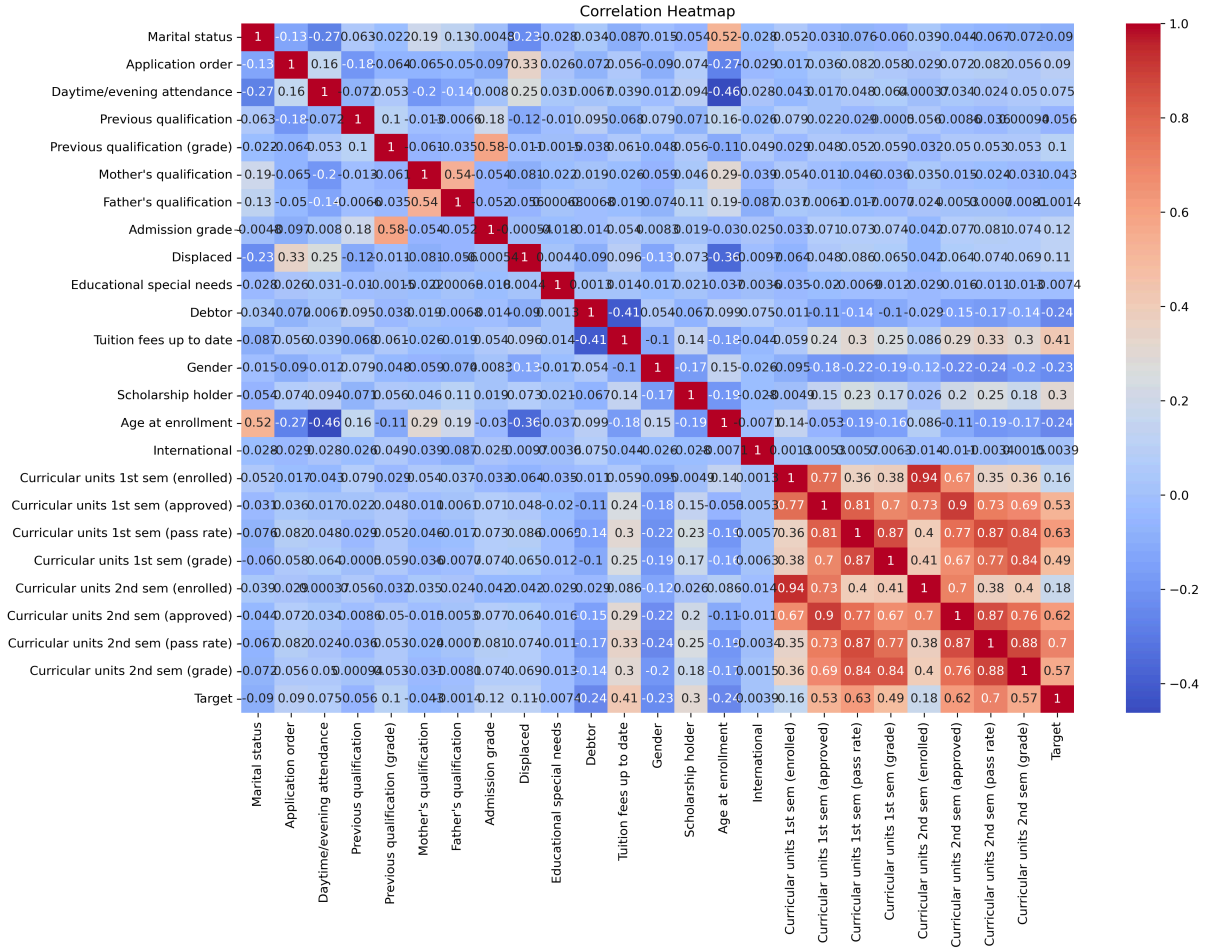
- 0 -> Dropout
- 1 -> Enrolled -> No Risk
- 2 -> Graduate -> No Risk

2. Engineered Features:

```
data['overall_pass_rate'] = (
    data['Curricular units 1st sem (approved)'] +
    data['Curricular units 2nd sem (approved)']
) / (
    data['Curricular units 1st sem (enrolled)'] +
    data['Curricular units 2nd sem (enrolled)']
)

data['grade_diff'] = data['Curricular units 2nd sem (grade)'] - data['Curricular units 1s
data['financial_risk'] = data['Debtor'] - data['Scholarship holder']
data['grade_x_passrate'] = data['Curricular units 1st sem (grade)'] * data['Curricular un
```

3. Correlation Heatmap

A correlation heatmap helped us spot redundant features to simplify KNN's distance calculation



4. Model Training

We trained 2 models:

1. Decision Tree with GridSearchCV:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV, StratifiedKFold

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

dt_params = {
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

dt = DecisionTreeClassifier(random_state=42)
dt_grid = GridSearchCV(dt, dt_params, cv=cv, scoring='accuracy', n_jobs=-1)
dt_grid.fit(X_train, y_train)
print(dt_grid.best_params_)
```

2. KNN with GridSearchCV:

```
from sklearn.neighbors import KNeighborsClassifier

knn_params = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'p': [1, 2]
}

knn = KNeighborsClassifier()
knn_grid = GridSearchCV(knn, knn_params, cv=cv, scoring='accuracy', n_jobs=-1)
knn_grid.fit(X_train, y_train)
print(knn_grid.best_params_)
```

5. Evaluation & Interpretation

We evaluated on a **hold-out test set** of 885 students.

1. Classification Report:

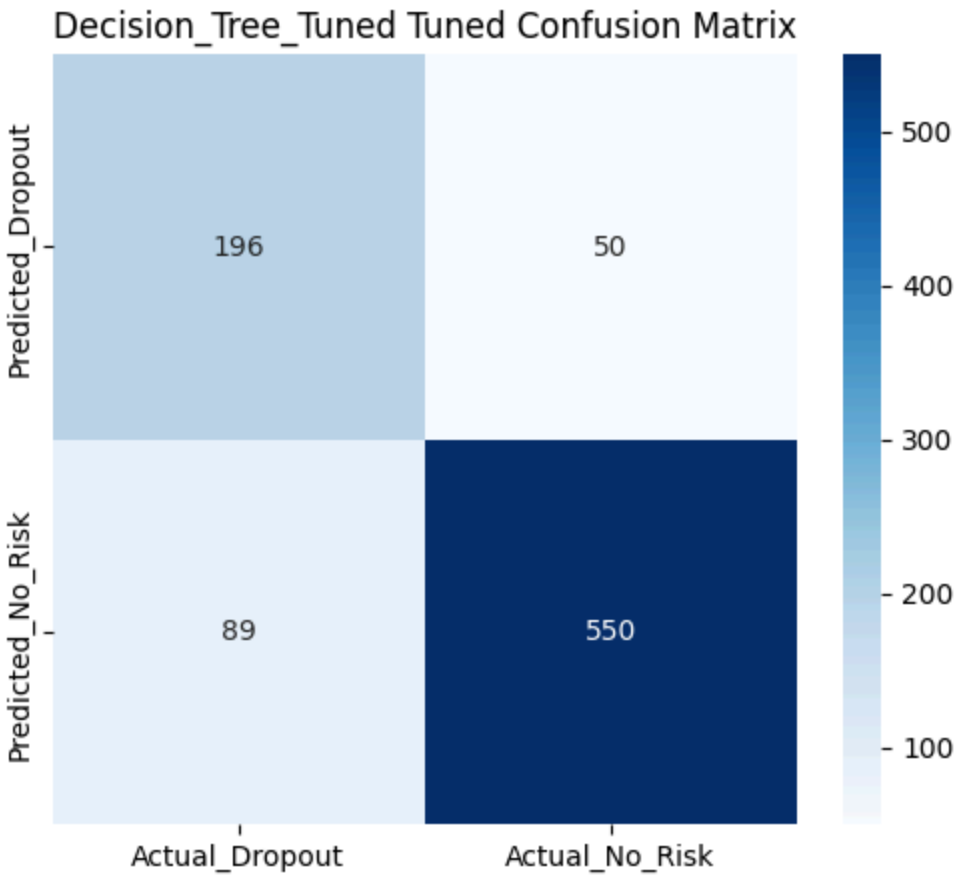
```
from sklearn.metrics import classification_report

y_pred_dt = dt_grid.predict(X_test)
print(classification_report(y_test, y_pred_dt, target_names=["Dropout", "No Risk"]))
```

2. Tuned Model Results:

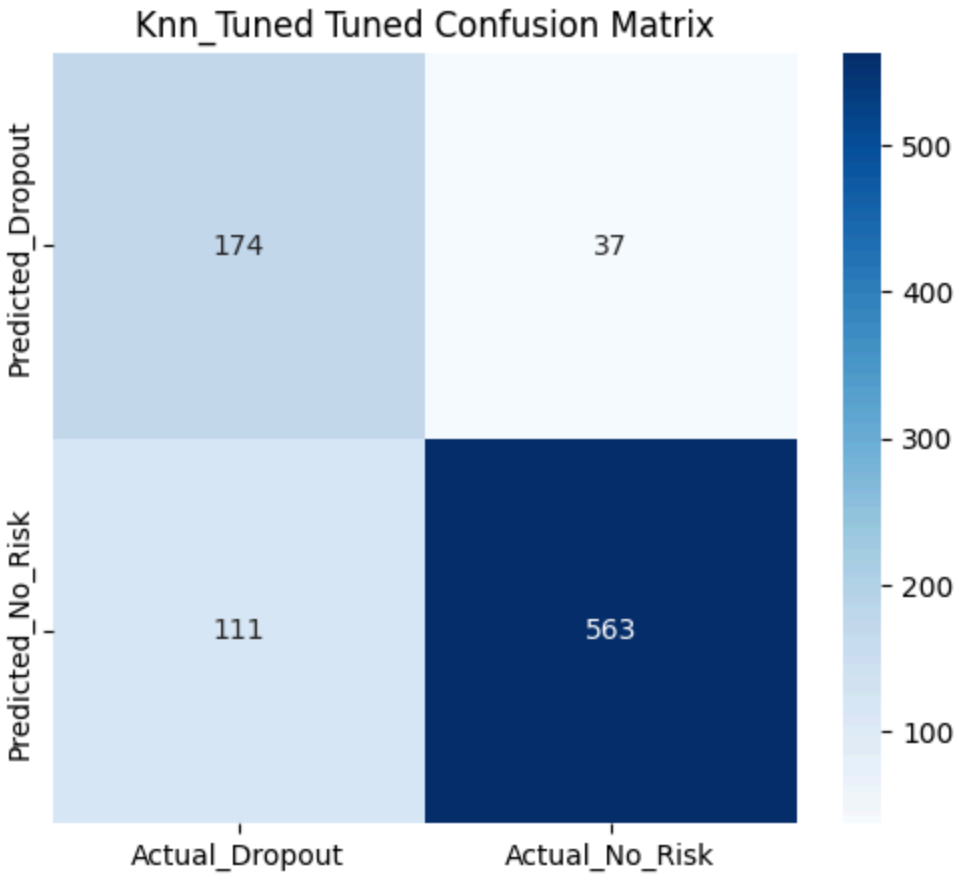
```
=== Decision_Tree_Tuned ===
              precision    recall  f1-score   support

     Dropout       0.80      0.69      0.74       285
     No Risk       0.86      0.92      0.89       600

    accuracy                           0.84       885
   macro avg       0.83      0.80      0.81       885
weighted avg       0.84      0.84      0.84       885


=== KNN_Tuned ===
              precision    recall  f1-score   support

     Dropout       0.82      0.61      0.70       285
     No Risk       0.84      0.94      0.88       600

    accuracy                           0.83       885
   macro avg       0.83      0.77      0.79       885
weighted avg       0.83      0.83      0.83       885
```

3. Confusion Matrix - Decision Tree

- TP: 196, FP: 50, FN: 89, TN: 550



4. Confusion Matrix – KNN



5. Trade-off:

- Decision Tree → Higher recall for Dropouts (better for early intervention) but more false positives.
- KNN → Fewer false positives but misses more dropouts.

6. Troubleshooting:

1. **Broken Virtual Environment After Python Update, fix:**

```
rm -rf .venv
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

2. Scripts Fail to Find the Data File

These scripts use **relative paths** such as:

```
pd.read_csv("../data/raw/data.csv")
```

If you run them from the project root folder, you will get a `FileNotFoundError`.

Fix: Always run the scripts **from inside the `src` directory**:

```
cd src
python dt_train_model.py
```

7. Verify Results

Check:

- Reports in reports/
- Saved models in modes/
- Confusion matrices to compare performance