STUDENT DROPOUT PREDICTION

BY GEORGE SYLVA

OUTLINE

- Introduction
- Methodology
- Results/fiindings
- conclusion

INTRODUCTION

- **PROBLEM STATEMENT**: UNIVERSITIES AND EDUCATIONAL INSTITUTIONS FACE CHALLENGES IN RETAINING STUDENTS, WITH DROPOUT RATES AFFECTING BOTH STUDENT SUCCESS AND INSTITUTIONAL REPUTATION. IDENTIFYING STUDENTS AT RISK OF DROPPING OUT CAN HELP IN TAKING TIMELY INTERVENTIONS AND IMPROVING RETENTION.
- **OBJECTIVE**: THE GOAL OF THIS PROJECT IS TO DEVELOP A MACHINE LEARNING MODEL THAT CAN ACCURATELY PREDICT STUDENT DROPOUT BASED ON ACADEMIC PERFORMANCE, DEMOGRAPHIC FACTORS, AND OTHER KEY VARIABLES. THIS WILL ENABLE EDUCATORS AND ADMINISTRATORS TO FOCUS ON STUDENTS WHO ARE MORE LIKELY TO DROP OUT AND PROVIDE THEM WITH NECESSARY SUPPORT.

- I analyzed student data, including both categorical and numerical features, such as "Course," "Mother's Occupation," "Previous Qualification (Grade)," "Admission Grade," "Curricular Units Enrolled" etc
- The initial dataset has 4424 rows and 37 columns
- A Random Forest model was then trained and evaluated using selected features of this data to predict the likelihood of a student dropping out.

METHODOLOGY
DATA PREPROCESSING

- The features which included numerical and catgorical features had to be ran through a pipeline for preprocessing. The target was seperated during train-test-split and label encoded.
- The numerical features were scaled using Standard_scaler (-1 to 1)
- The categorical features all seem not to need any transoformal since all categorical features appear to be encoded altight.
- Knn imputer was used to handle any missing numerical value, of which there was none but this will come in handy when pipeline Is used in model building and deployed. Missing values will not necessarily affect model because knn is handling it.

FEATURE SELECTION

PROBLEM: NOT ALL FEATURES CONTRIBUTE EQUALLY TO PREDICTING THE TARGET VARIABLE (DROPOUT/GRADUATE/ENROLLED).

SOLUTION:

- WE APPLIED **RECURSIVE FEATURE ELIMINATION (RFE)** USING THE RANDOM FOREST MODEL TO SELECT THE MOST IMPORTANT FEATURES.
- RFE ITERATIVELY REMOVES THE LEAST IMPORTANT FEATURES TO IDENTIFY THE SUBSET THAT MAXIMIZES MODEL PERFORMANCE.
- I USED RFE OF VARIOUS NUMBER OF KEY FEATURES AND WHEN FEATURE WAS 15 MODEL PERFORMANCE WAS CLOSE ENOUGH TO ITS PERFORMANCE WHEN WE USED ALL 36 TRAINING FEATURES.

THE PASSTHROUGH METHOD WAS USED FOR THE CATEGORICAL COLUMNS SINCE WE WILL NOT BE TRANSFORMING THOSE

Data Preprocessing, Hyperparameter Tuning and Model Evaluation

[11]:
```
# Create a Column Transformer for Preprocessing
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline(steps=[
            ('imputer', KNNImputer(n_neighbors=5)),
            ('scaler', StandardScaler())
        ]), numerical_cols),
        ('cat', 'passthrough', categorical_cols)
    ]
)
```

[12]:
```
# Define Hyperparameter Search Spaces
rf_param_dist = {
    'model__n_estimators': [50, 100, 150], #[100, 200, 300]
    'model__max_depth': [None, 5, 10, 15],
    'model__min_samples_split': [4, 10, 20], #[2, 5, 10]
    'model__min_samples_leaf': [2, 4, 6], #[1, 2, 3]
    'model__bootstrap': [True, False]
}

xgb_param_dist = {
    'model__n_estimators': [100, 200, 300],
    'model__max_depth': [1, 2, 3],
    'model__learning_rate': [0.02, 0.2, 0.4],
    'model__subsample': [0.7, 0.8, 0.9],
    'model__colsample_bytree': [0.7, 0.8, 1.0]
}
```

PIPELINES FOR RF MODEL AND XGBOOST AND HYPERPARAMETER TUNING USING PIPELINES

```
[13]: # Create Pipelines
rf_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', RandomForestClassifier(random_state=42))
])

xgb_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', XGBClassifier(eval_metric='mlogloss', random_state=42))
])
```

```
[14]: # Perform hyperparameter tuning with RandomizedSearchCV to the RandomForest pipeline
rf_random_search = RandomizedSearchCV(
    estimator=rf_pipeline,
    param_distributions=rf_param_dist,
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
    scoring='accuracy',
    n_iter=10,
    n_jobs=-1,
    verbose=1,
    random_state=42
)

# Perform hyperparameter tuning with RandomizedSearchCV to the XGBoost pipeline
xgb_random_search = RandomizedSearchCV(
    estimator=xgb_pipeline,
    param_distributions=xgb_param_dist,
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),
    scoring='accuracy',
    n_iter=10,
    n_jobs=-1,
    verbose=1,
    random_state=42
)
```

MODELS PERFORMANCES AFTER USING HYPERPARAMETR TUNING

```
rf_y_pred = rf_random_search.best_estimator_.predict(X_test)
xgb_y_pred = xgb_random_search.best_estimator_.predict(X_test)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[16]: # Evaluation on Test Set
print("Random Forest Test Accuracy:", accuracy_score(y_test, rf_y_pred))
print(classification_report(y_test, rf_y_pred))

print("XGBoost Test Accuracy:", accuracy_score(y_test, xgb_y_pred))
print(classification_report(y_test, xgb_y_pred))
```

```
Random Forest Test Accuracy: 0.7796610169491526
              precision    recall  f1-score   support

           0       0.81      0.75      0.78       284
           1       0.62      0.43      0.51       159
           2       0.80      0.93      0.86       442

    accuracy                           0.78       885
   macro avg       0.74      0.70      0.71       885
weighted avg       0.77      0.78      0.77       885

XGBoost Test Accuracy: 0.7728813559322034
              precision    recall  f1-score   support

           0       0.80      0.74      0.77       284
           1       0.55      0.48      0.52       159
           2       0.82      0.90      0.86       442

    accuracy                           0.77       885
   macro avg       0.72      0.71      0.71       885
weighted avg       0.77      0.77      0.77       885
```

DURING FEATURE SELECTION IT WAS DISCOVERED THAT THE BEST PERFORMANCE WAS WHEN NUMBER OF FEATURE WAS 15. THESE WERE THEN USED TO TRAIN OUR MODEL

```
print("Selected Features:", selected_features)
```

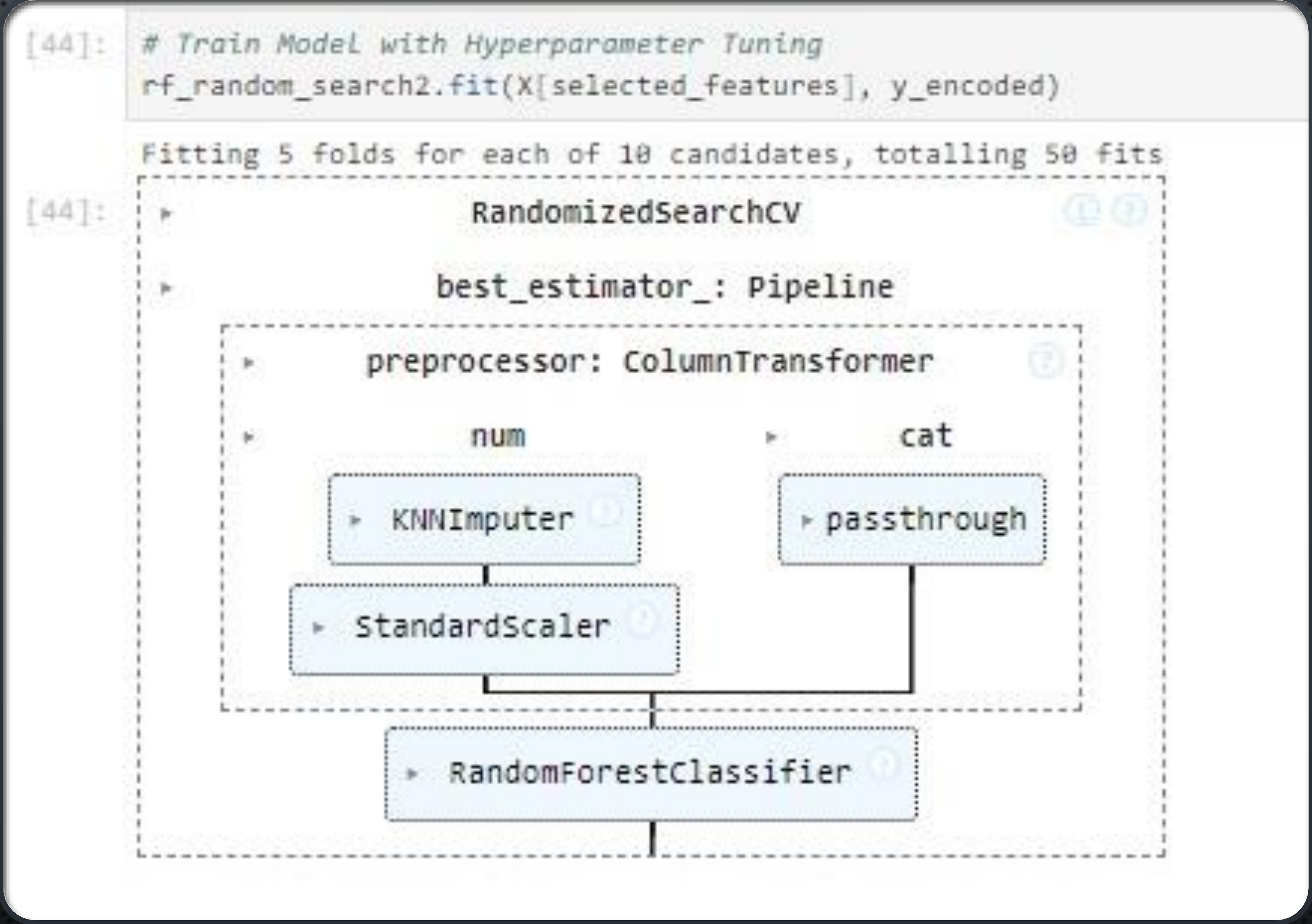
```
Selected Features: Index(['Course', 'Previous qualification (grade)', 'Mother_occupation',
       'Admission grade', 'Tuition fees up to date', 'Age at enrollment',
       'Curricular units 1st sem (enrolled)',
       'Curricular units 1st sem (evaluations)',
       'Curricular units 1st sem (approved)',
       'Curricular units 1st sem (grade)',
       'Curricular units 2nd sem (enrolled)',
       'Curricular units 2nd sem (evaluations)',
       'Curricular units 2nd sem (approved)',
       'Curricular units 2nd sem (grade)', 'GDP'],
      dtype='object')
```

[31]:

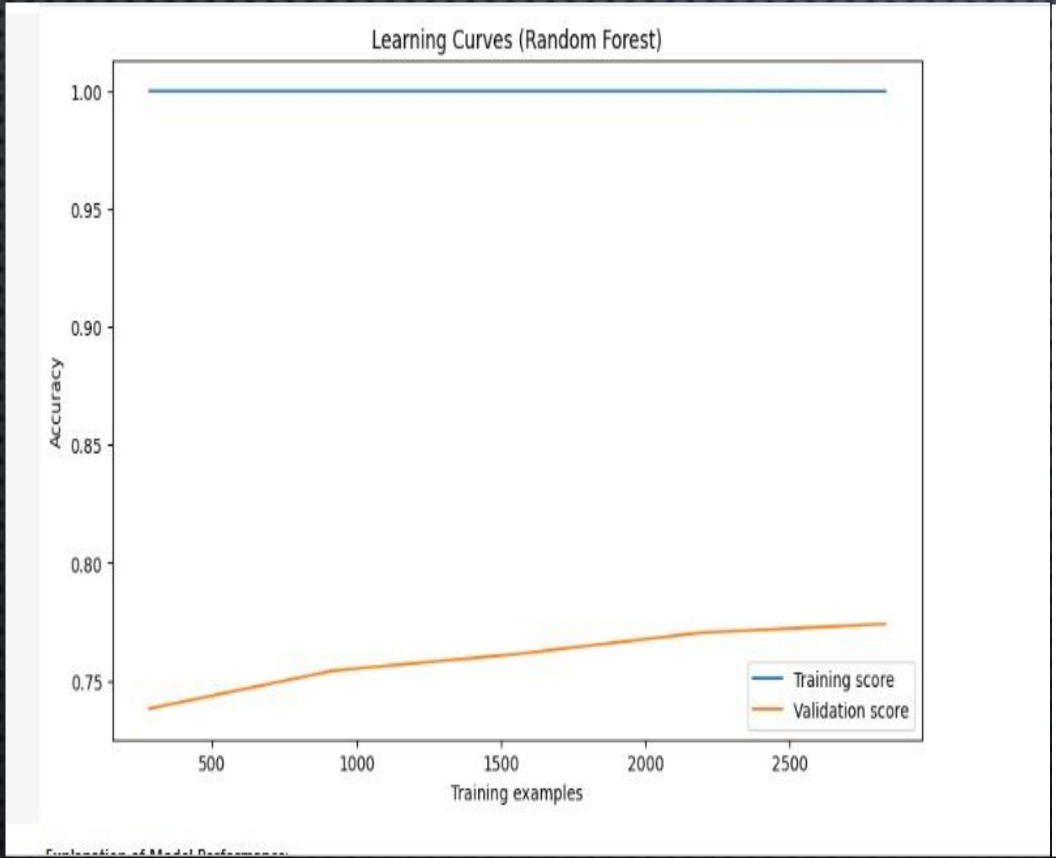
```
# Train model with selected features
rf_model.fit(X_train[selected_features], y_train)

# Predict with test set
y_pred_rfe = rf_model.predict(X_test[selected_features])

# Evaluate the model
print("Random Forest Test Accuracy with RFE:", accuracy_score(y_test, y_pred_rfe))
print(classification_report(y_test, y_pred_rfe))
```

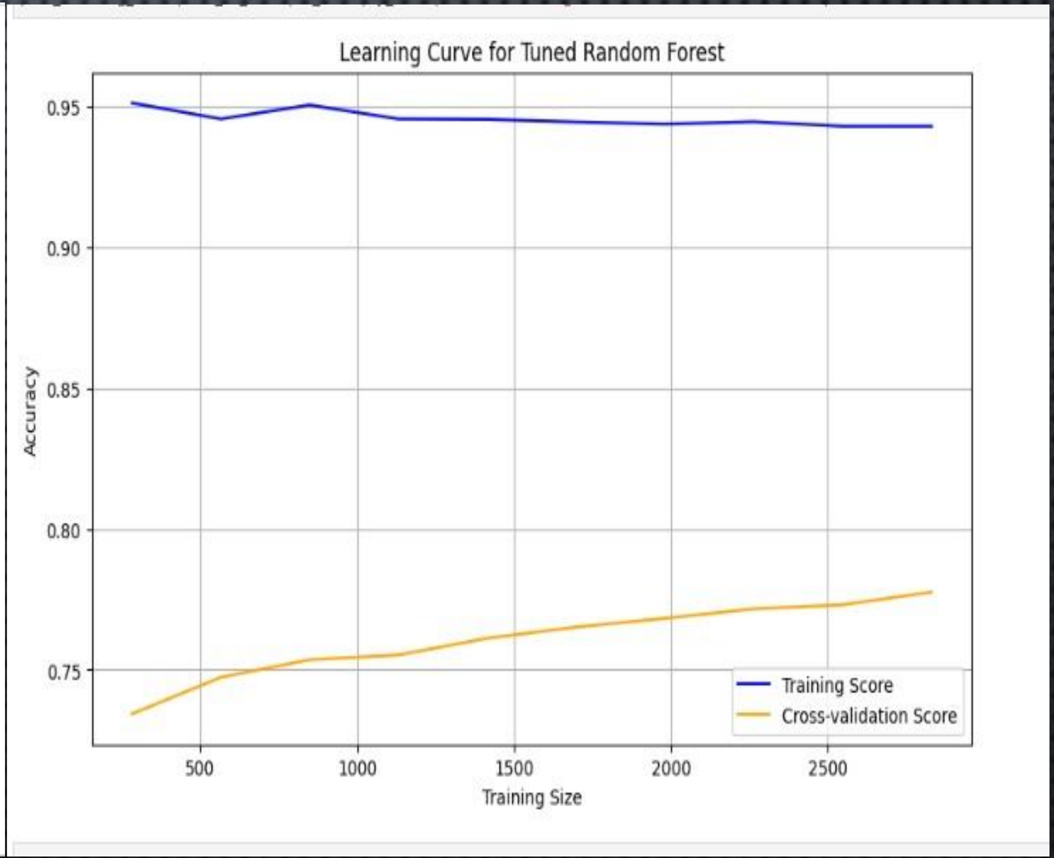
```
Random Forest Test Accuracy with RFE: 0.768361581920904
              precision    recall  f1-score   support

           0       0.80      0.75      0.78       284
           1       0.54      0.36      0.44       159
           2       0.80      0.92      0.86       442

    accuracy                           0.77       885
   macro avg       0.71      0.68      0.69       885
weighted avg       0.75      0.77      0.76       885
```

FLOW CHART OF OUR RF MODEL USING ITS BEST ESTIMATOR. WITHIN THE PIPELINE IS OUR PREPROCESSOR FOR BOTH NUMERICAL AND CATEGORICAL FEATURES.

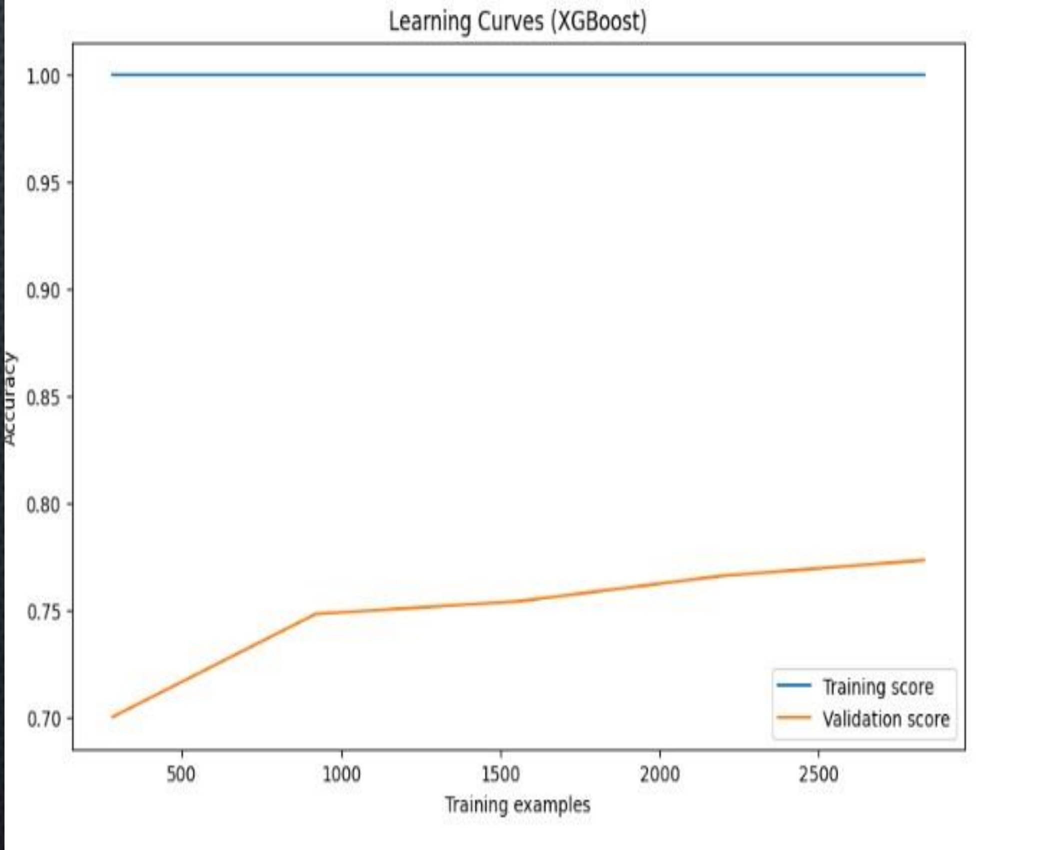


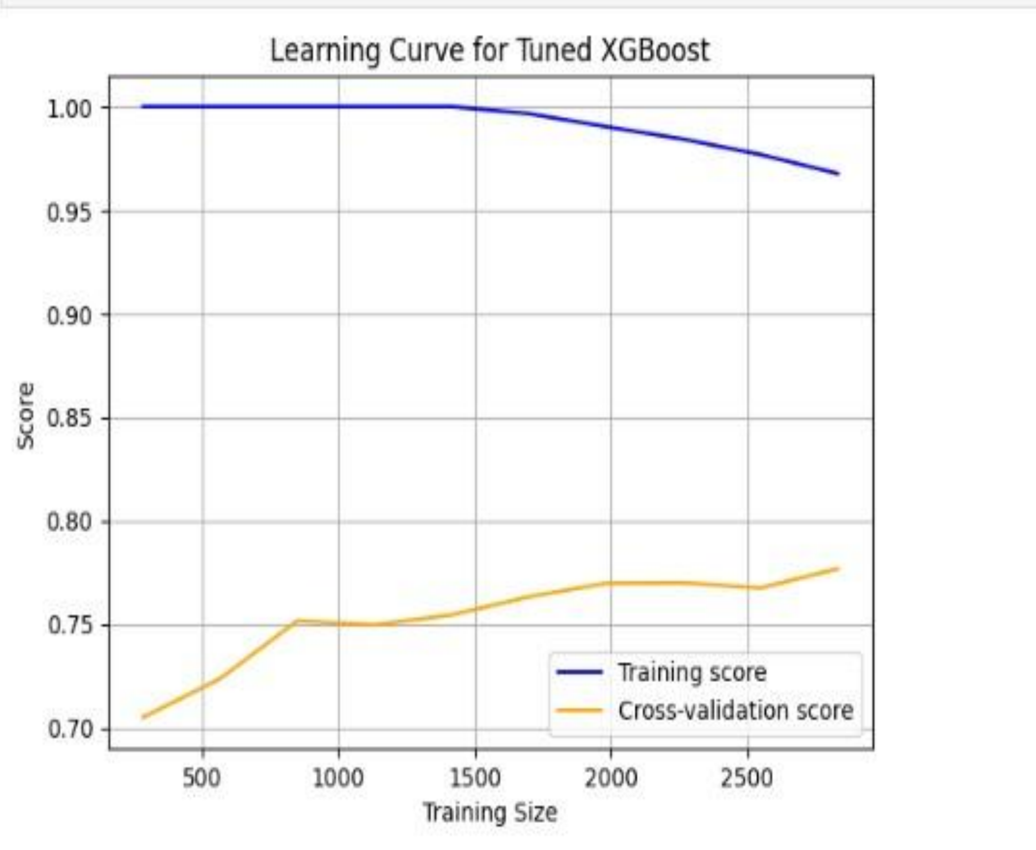
RF MODEL EVALUATION PLOT





XGBOOST MODEL EVALUATION PLOT





MODEL DEPLOYEMNT

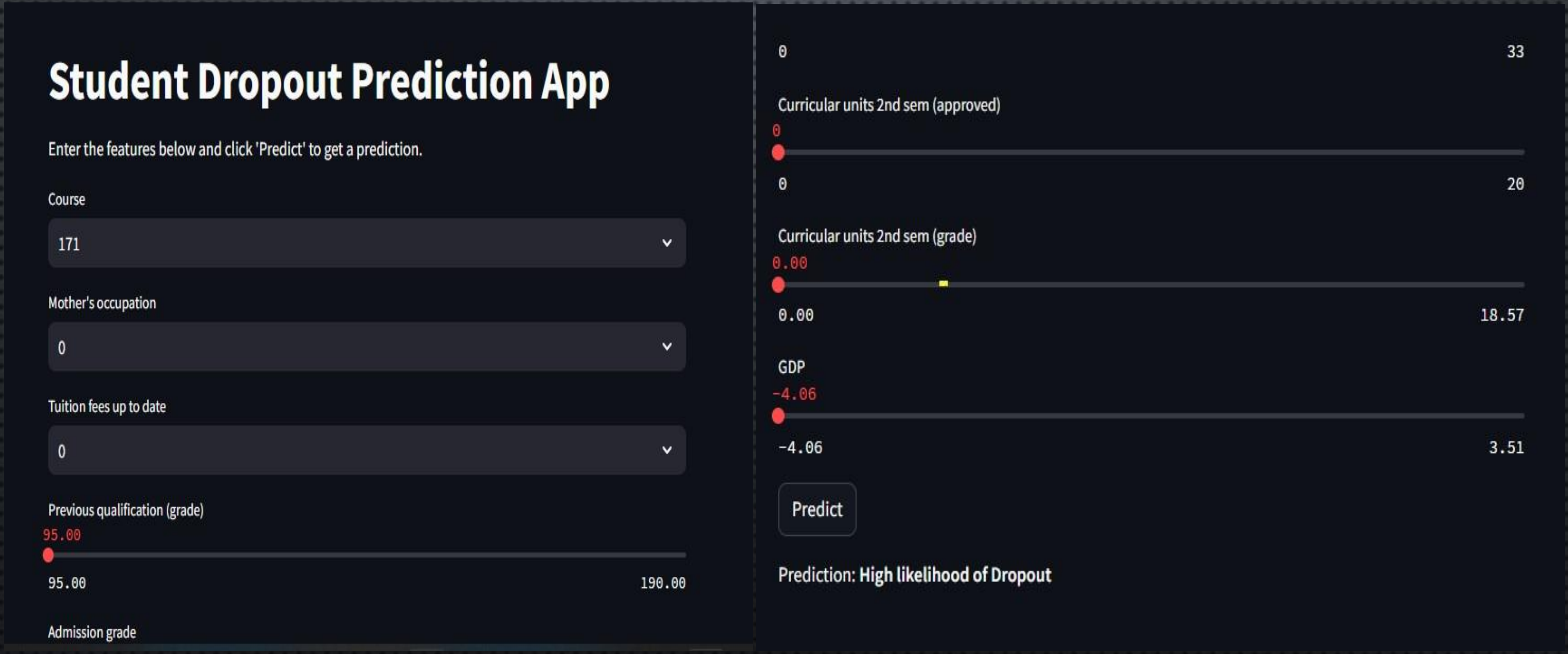
STUDENT DROPOUT PREDICTION APP

After training and fine-tuning the machine learning model, the next step was deploying it for practical use. In this project, we deployed the trained **Random Forest model** in the form of a web application using **Streamlit**. The app allows users to interact with the model and make predictions based on student data, helping to forecast the likelihood of student dropout or graduation.

- After tuning the Random Forest model, it was saved as a serialized object using **Joblib**. Along with the model. the **Label Encoder** used to transform the target variable into numeric labels was also saved.

Reason: Saving the model and encoder ensures that we can easily reload them in the deployment environment for making predictions without retraining.

STREAMLIT APP



Conclusion

- In this project, we successfully built a machine learning pipeline to predict student outcomes, with a focus on identifying students at risk of dropping out. We started with thorough data analysis and preprocessing, followed by training a robust **Random Forest model**, which performed well in predicting the likelihood of dropout, enrollment, or graduation. The model was then deployed in an easy-to-use **Streamlit web application**.

Key Takeaways:

- **Data Insights:** Through data exploration, we identified key factors influencing student performance and dropout risk, such as **previous academic performance, family background** which is reflected in parents occupation and how updated childs tuition fee payments is.
- **Model Performance:** Our **Random Forest model** provided high accuracy in predicting student outcomes, thanks to careful feature selection, hyperparameter tuning, and robust cross-validation techniques.
- **App Deployment:** The interactive Streamlit app makes the model accessible to stakeholders, allowing educators and administrators to input student data and quickly predict their risk of dropout.

REFERENCE

- GitHub Repo: https://github.com/georgesylva1/3signet_task_1
- Streamlit app: https://3signettask1-version3.streamlit.app/