MACHINE LEARNING PROJECT

PROBLEM STATEMENT:-

The given dataset contains students' data, and we need to predict whether a given student has dropped out, graduated, or is still enrolled in the course.

DATASET DESCRIPTION AND PREPROCESSING:-

The given dataset has the following columns:-
['Marital_Status_Code', 'Application_Method', 'Application_Sequence', 'Attendance_Type', 'Prior_Qualification_Code', 'Prior_Qualification_Score', 'Nationality_Code', "Mother's_Education_Level", "Father's_Education_Level", "Mother's_Job_Category", "Father's_Job_Category", 'Admission_Score', 'Student_Displacement_Flag', 'Special_Educational_Needs', 'Outstanding_Debts_Flag', 'Tuition_Fees_UpToDate_Flag', 'Gender_Code', 'Scholarship_Recipient_Flag', 'Enrollment_Age', 'International_Status', 'Credits_1st_Semester', 'Enrolled_1st_Semester', 'Evaluations_1st_Semester', 'Passed_1st_Semester', 'Grade_1st_Semester', 'No_Evaluations_1st_Semester', 'Credits_2nd_Semester', 'Enrolled_2nd_Semester', 'Evaluations_2nd_Semester', 'Passed_2nd_Semester', 'Grade_2nd_Semester', 'No_Evaluations_2nd_Semester', 'Local_Unemployment_Rate', 'Inflation_Rate', 'Regional_GDP', 'Outcome']

In the given columns, ‘Outcome’ is our target column, and the rest are our features.

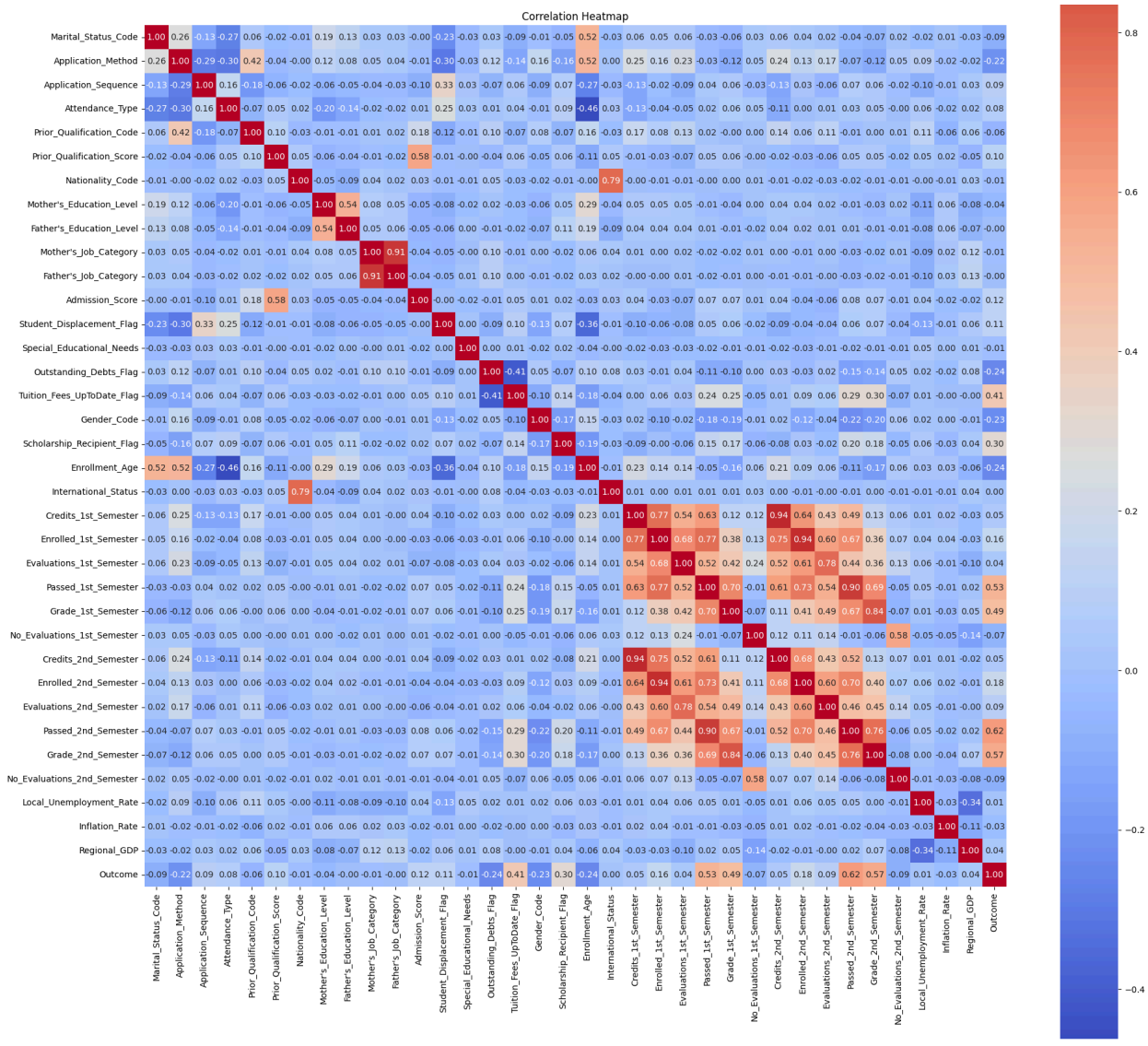
The given dataset has no null values for any of the features, but there is a slight imbalance for our target:-

Graduated - 49%
Dropped out - 32%
Enrolled - 18%

We use an Ordinal Encoder to encode the Outcome

Now with the target encoded, we plot the correlation matrix. The higher the correlation between a feature and the outcome, the better that feature contributes to the predictive power of our Machine Learning Model. The colour of a cell across a row and column depicts the value.

Correlation matrix Plot:-



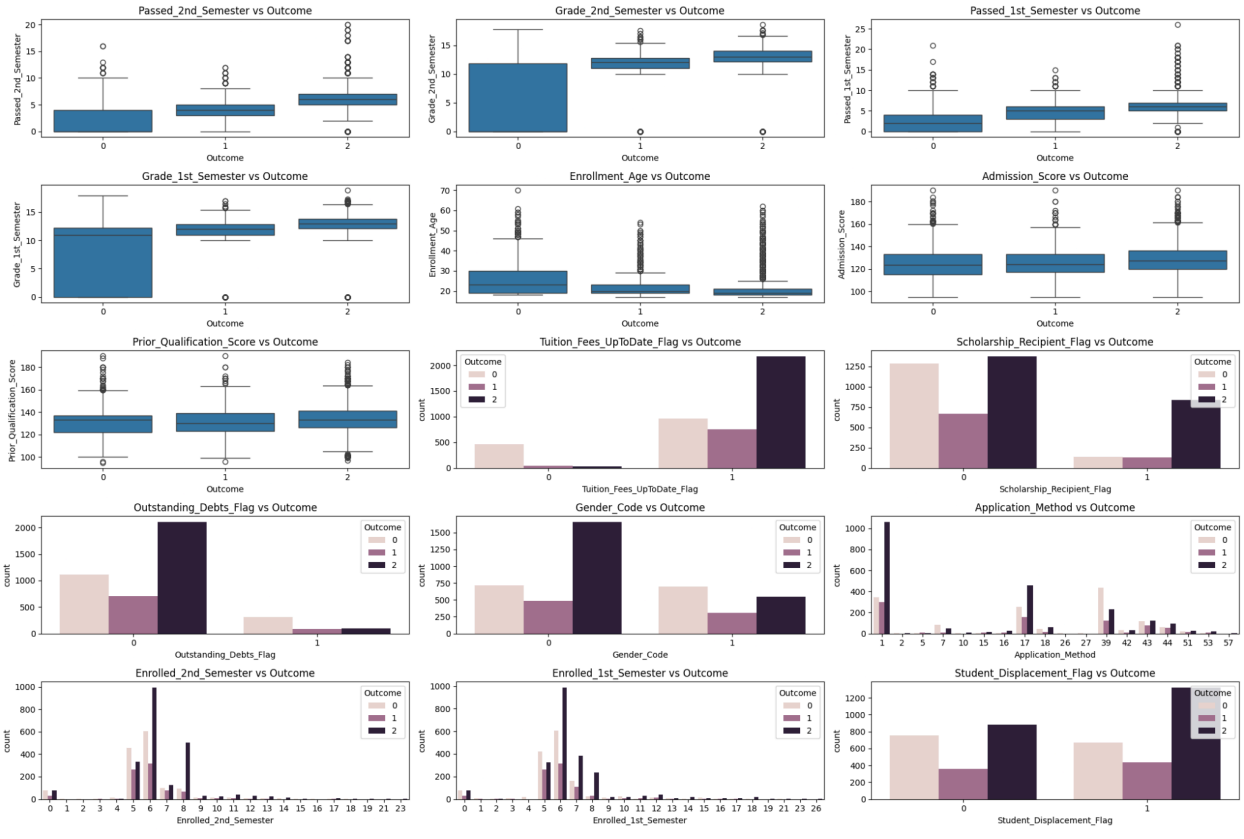
We extract the best features with a correlation value > 0.1 with the Outcome.

We are left with the following features:-

['Outcome', 'Passed_2nd_Semester', 'Grade_2nd_Semester',
'Passed_1st_Semester', 'Grade_1st_Semester',
'Tuition_Fees_UpToDate_Flag', 'Scholarship_Recipient_Flag',
'Enrollment_Age', 'Outstanding_Debts_Flag', 'Gender_Code',
'Application_Method', 'Enrolled_2nd_Semester', 'Enrolled_1st_Semester',
'Admission_Score', 'Student_Displacement_Flag',
'Prior_Qualification_Score']

We Visualise Numerical Features with Boxplots and Categorical Features with Countplots

Visualisation for Best Features:-



FEATURE ENGINEERING:-

We add the following features from the best shortlisted features

- Overall_Grade = Average Grade Across both the Semesters
- Grade_Improvement = the trajectory of grades, whether upwards or downwards
- Sems_Passed = How many semesters passed successfully
- Total_Credits = Total credits enrolled in
- Age_Group = Age Grouped into categories
- Tuition_Fees_Flag = Financial Indicator

OUTLIER CLIPPING: We Clip Outliers with quantile

SCALING: With the Standard Scaler, We scale the data

ML MODEL TRAINING AND EVALUATION:-

First, we fit the data to a general Function for Model Training. To do this, we split the data with train_test_split.
Random_state is set to 18
We train the following ML models in this function:-

LogisticRegression, RandomForestClassifier, GradientBoostingClassifier, SVC, KNeighborsClassifier, XGBoostClassifier

We evaluate against Accuracy, Precision, Recall, F1-Score:-

	Model	Accuracy	Precision:	Recall	F1
1	RandomForest	0.800000	0.761671	0.705706	0.724078
2	GradientBoosting	0.790960	0.746389	0.693735	0.710891
5	XGBoost	0.770621	0.710625	0.674548	0.686714
0	LogisticRegression	0.769492	0.702616	0.663481	0.674100
3	SVC	0.697175	0.609768	0.542252	0.521951
4	KNN	0.655367	0.582655	0.559806	0.566952

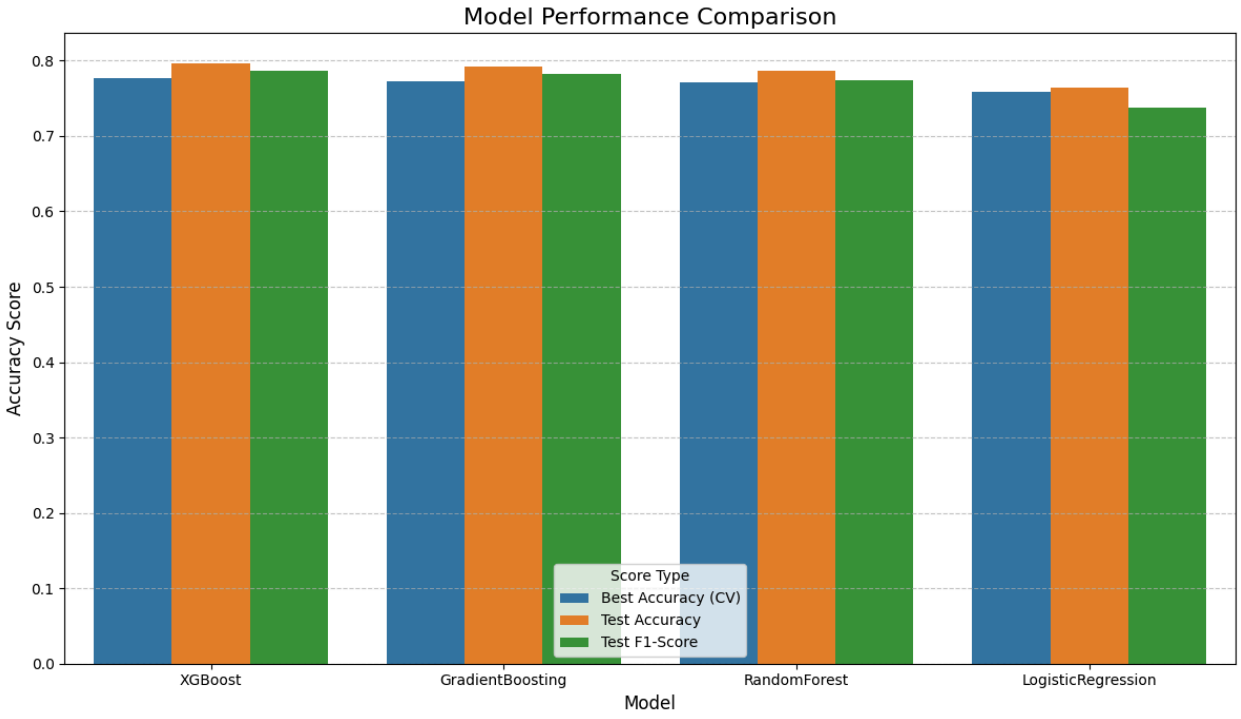
Next, we create a new Function where we fine-tune the models using GridSearchCV to get the best models, where we limit the function to the following models because of time and resource constraints: Logistic Regressor, Random Forest, Gradient Boost and XGBoost

From the new function, we get the following results:-

	Model	Best Accuracy (CV)	Test Accuracy	Test Precision	Test Recall	Test F1-Score	Best Parameters
3	XGBoost	0.776207	0.796610	0.789688	0.796610	0.785986	{'learning_rate': 0.1, 'max_depth': 3, 'n_esti...
2	GradientBoosting	0.773100	0.792090	0.784038	0.792090	0.782493	{'learning_rate': 0.1, 'max_depth': 3, 'n_esti...
1	RandomForest	0.771122	0.786441	0.776671	0.786441	0.774001	{'max_depth': 20, 'min_samples_leaf': 1, 'n_es...
0	LogisticRegression	0.758692	0.763842	0.737957	0.763842	0.738251	{'C': 0.1, 'solver': 'liblinear'}

RESULT VISUALISATION:-

We visualise our Best results with a Bar Chart Plot and conclude with an approximate **80 percent Accuracy**



DL MODEL TRAINING AND EVALUATION

First we perform the same steps as we did for the ML Model for processing the data then :-

TENSORIZING: We convert our data:-

Pandas DataFrame -> Tensors

Tensors -> Tensor Dataset

Tensor Dataset -> DataLoader (batchsize = 64)

MODEL ARCHITECTURE:-

```python
# Outccome Model Architecture
class OutcomeModel(nn.Module):
  def __init__(self, input_dim, hidden_dim1, hidden_dim2, output_dim):
    super(OutcomeModel, self).__init__()
    self.fc1 = nn.Linear(input_dim, hidden_dim1)
    self.batchnorm1 = nn.BatchNorm1d(hidden_dim1)

    self.fc2 = nn.Linear(hidden_dim1, hidden_dim2)
    self.batchnorm2 = nn.BatchNorm1d(hidden_dim2)

    self.out = nn.Linear(hidden_dim2, output_dim)
    self.relu = nn.ReLU()
    self.dropout = nn.Dropout(p=0.3)

  def forward(self, x):
    x = self.relu(self.batchnorm1(self.fc1(x)))
    x = self.dropout(x)
    x = self.relu(self.batchnorm2(self.fc2(x)))
    x = self.dropout(x)
    return self.out(x)
```

MODEL PARAMETERS:-

- Adam Optimizer
- CrossEntropyLoss
- 0.001 Learning Rate with a Scheduler to reduce Learning Rate with drop in progress
- Early Stopper with Patience 7
- 50 Epochs

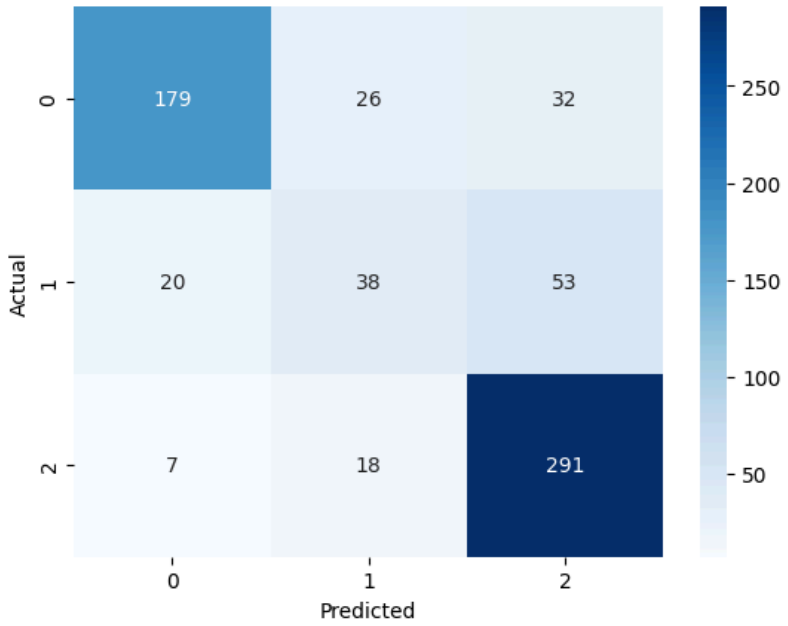
MODEL TRAINING:-

```
Epoch 1/50, Training Loss: 0.8037, Validation Loss: 0.6717, Validation Accuracy: 0.7380
Epoch 2/50, Training Loss: 0.6640, Validation Loss: 0.6207, Validation Accuracy: 0.7590
Epoch 3/50, Training Loss: 0.6361, Validation Loss: 0.6131, Validation Accuracy: 0.7605
Epoch 4/50, Training Loss: 0.6106, Validation Loss: 0.5959, Validation Accuracy: 0.7575
Epoch 5/50, Training Loss: 0.5943, Validation Loss: 0.5901, Validation Accuracy: 0.7636
Epoch 6/50, Training Loss: 0.5875, Validation Loss: 0.5957, Validation Accuracy: 0.7545
Epoch 7/50, Training Loss: 0.5819, Validation Loss: 0.5915, Validation Accuracy: 0.7530
Epoch 8/50, Training Loss: 0.5755, Validation Loss: 0.5826, Validation Accuracy: 0.7696
Epoch 9/50, Training Loss: 0.5695, Validation Loss: 0.5870, Validation Accuracy: 0.7575
Epoch 10/50, Training Loss: 0.5707, Validation Loss: 0.5859, Validation Accuracy: 0.7590
Epoch 11/50, Training Loss: 0.5615, Validation Loss: 0.5804, Validation Accuracy: 0.7681
Epoch 12/50, Training Loss: 0.5707, Validation Loss: 0.5765, Validation Accuracy: 0.7666
Epoch 13/50, Training Loss: 0.5586, Validation Loss: 0.5893, Validation Accuracy: 0.7605
Epoch 14/50, Training Loss: 0.5573, Validation Loss: 0.5871, Validation Accuracy: 0.7666
Epoch 15/50, Training Loss: 0.5482, Validation Loss: 0.5888, Validation Accuracy: 0.7726
Epoch 16/50, Training Loss: 0.5487, Validation Loss: 0.5806, Validation Accuracy: 0.7666
Epoch 17/50, Training Loss: 0.5442, Validation Loss: 0.5789, Validation Accuracy: 0.7666
Epoch 18/50, Training Loss: 0.5322, Validation Loss: 0.5787, Validation Accuracy: 0.7651
Epoch 19/50, Training Loss: 0.5337, Validation Loss: 0.5803, Validation Accuracy: 0.7651
Early stopping at epoch 19
Training Complete. Loading Best Model Weights
```

RESULT

We conclude the Result from the Neural Network with 76.5 percent accuracy

CONFUSION MATRIX:-



CLASSIFICATION REPORT:-

	precision	recall	f1-score	support
0	0.87	0.76	0.81	237
1	0.46	0.34	0.39	111
2	0.77	0.92	0.84	316
accuracy			0.77	664
macro avg	0.70	0.67	0.68	664
weighted avg	0.76	0.77	0.75	664