Group 25 2025-11-21

IT3212 Assignment 3: Basic modelling

Table of Contents

1. Develop a problem statement (real world and machine learning)

a. This is one of the most important skills that a Machine Learning Engineer/Scientist should have. Select a dataset and frame a machine learning problem and then connect this machine learning problem to the real world scenario.

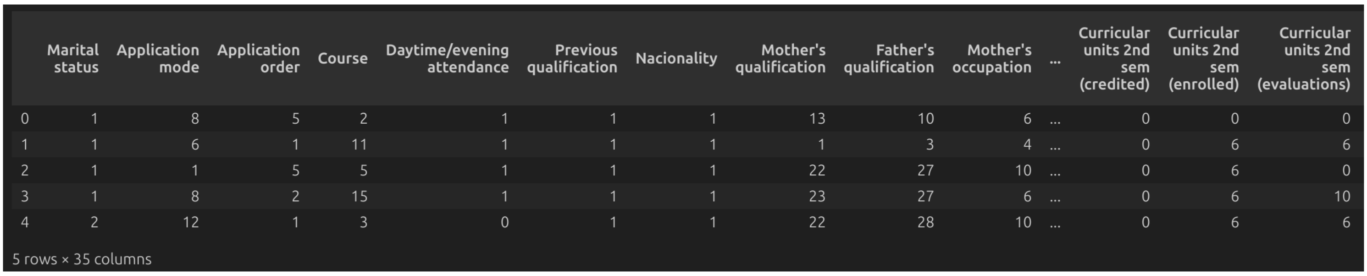
Real World Problem
As the education sector becomes more data-driven, collected data can unlock substantial value. Universities want to reduce course dropout and capture students who are likely to still be enrolled beyond the normal time to graduate, so institutions can allocate extra resources proactively and help students get back on track. This improves student success and workforce readiness, strengthens institutional outcomes, and generates insights useful for policymakers.

Machine Learning Problem
With this in mind, we selected the Student Graduation dataset, which records students across multiple undergraduate programs and includes socio-economic factors, prior academic background, and performance at the end of the first and second semesters. Our goal is to train machine learning models that predict three outcomes: dropout, extended enrollment beyond the normal time, or successful completion of the course. These predictions directly support the real-world problem by enabling early, targeted interventions for students at risk of dropping out or in need of assistance.

2. Implement the preprocessing and justify the preprocessing steps

We first looked at the data to try to find out what preprocessing steps were necessary. Since we had the source of the data, we had a description available for every column.

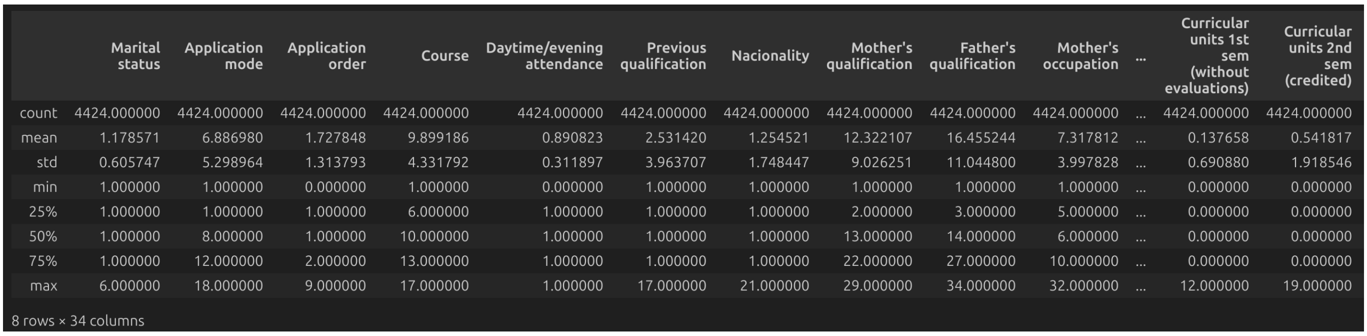
All categorical columns are label encoded, and has a mapping from number to category in the data description, but we found the numbers didn't match what was described in the data source. Assuming we were supposed to use the provided dataset over data directly from the source, we couldn't know what each class in the column meant.



	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Mother's qualification	Father's qualification	Mother's occupation	...	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)	Curricular units 2nd sem (evaluations)
0	1	8	5	2	1	1	1	13	10	6	...	0	0	0
1	1	6	1	11	1	1	1	1	3	4	...	0	6	6
2	1	1	5	5	1	1	1	22	27	10	...	0	6	0
3	1	8	2	15	1	1	1	23	27	6	...	0	6	10
4	2	12	1	3	0	1	1	22	28	10	...	0	6	6

5 rows × 35 columns

Figure 1: First 5 rows of our data

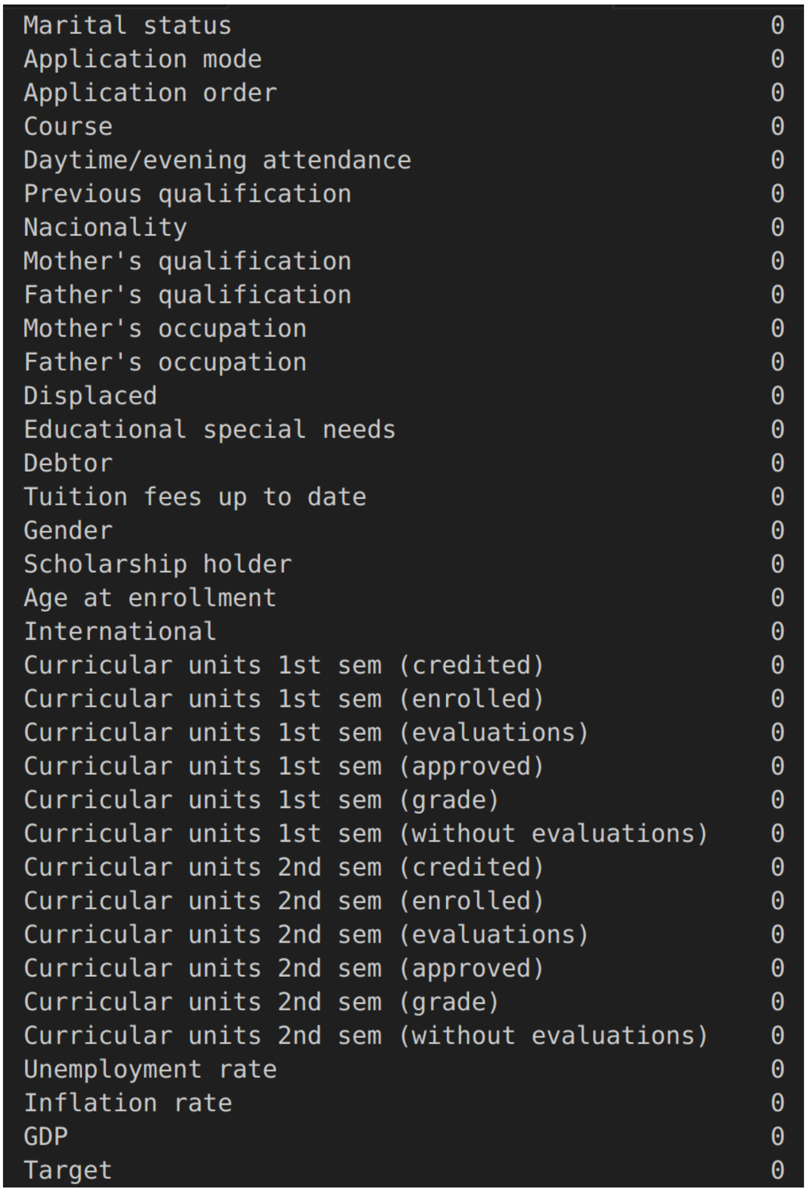


	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Mother's qualification	Father's qualification	Mother's occupation	...	Curricular units 1st sem (without evaluations)	Curricular units 2nd sem (credited)
count	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	...	4424.000000	4424.000000
mean	1.178571	6.886980	1.727848	9.899186	0.890823	2.531420	1.254521	12.322107	16.455244	7.317812	...	0.137658	0.541817
std	0.605747	5.298964	1.313793	4.331792	0.311897	3.963707	1.748447	9.026251	11.044800	3.997828	...	0.690880	1.918546
min	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	0.000000	0.000000
25%	1.000000	1.000000	1.000000	6.000000	1.000000	1.000000	1.000000	2.000000	3.000000	5.000000	...	0.000000	0.000000
50%	1.000000	8.000000	1.000000	10.000000	1.000000	1.000000	1.000000	13.000000	14.000000	6.000000	...	0.000000	0.000000
75%	1.000000	12.000000	2.000000	13.000000	1.000000	1.000000	1.000000	22.000000	27.000000	10.000000	...	0.000000	0.000000
max	6.000000	18.000000	9.000000	17.000000	1.000000	17.000000	21.000000	29.000000	34.000000	32.000000	...	12.000000	19.000000

8 rows × 34 columns

Figure 2: Description of data columns

From looking at the data source, we knew the dataset had been undergone rigorous data preprocessing to handle data from anomalies, unexplainable outliers, and missing values. Still, we decided to see ourselves if there were anomalies or outliers.



```
Marital status                                    0
Application mode                                  0
Application order                                 0
Course                                            0
Daytime/evening attendance                        0
Previous qualification                            0
Nacionality                                       0
Mother's qualification                            0
Father's qualification                            0
Mother's occupation                               0
Father's occupation                               0
Displaced                                         0
Educational special needs                         0
Debtor                                            0
Tuition fees up to date                           0
Gender                                            0
Scholarship holder                                0
Age at enrollment                                 0
International                                     0
Curricular units 1st sem (credited)               0
Curricular units 1st sem (enrolled)               0
Curricular units 1st sem (evaluations)            0
Curricular units 1st sem (approved)               0
Curricular units 1st sem (grade)                  0
Curricular units 1st sem (without evaluations)    0
Curricular units 2nd sem (credited)               0
Curricular units 2nd sem (enrolled)               0
Curricular units 2nd sem (evaluations)            0
Curricular units 2nd sem (approved)               0
Curricular units 2nd sem (grade)                  0
Curricular units 2nd sem (without evaluations)    0
Unemployment rate                                 0
Inflation rate                                    0
GDP                                               0
Target                                            0
```

Figure 3: Null values in dataset

There were no null values in the dataset, as seen in figure 3. We also found that all categorical columns had as many values as expected, and all value columns, like age and curricular units, had expected values.

The first step in our preprocessing is one hot encoding on categorical columns. We use one hot encoding over label encoding for this dataset, because the categorical columns don't have any real order, meaning a higher or lower value when label encoded wouldn't mean anything, only the exact numbers. To prevent creating an order where there is none, we use one hot encoding.

One problem with using one hot encoding on our dataset is that our categorical columns have many categories, turning the dataset from having 35 to having 246 columns. This makes our dataset more sparse, making each column contain less information, which can be harmful to some models. We prioritized not creating a non-existent order, and will instead remove excess features later.

We then split the dataset into a train and test set. This was done with a 75-25 split. Splitting the dataset as such enables the model to learn patterns from most of the data while keeping a suitable portion for unbiased evaluation.

Then we min-max scaled the dataset, ensuring no feature is weighted too highly based on having higher values than the others.

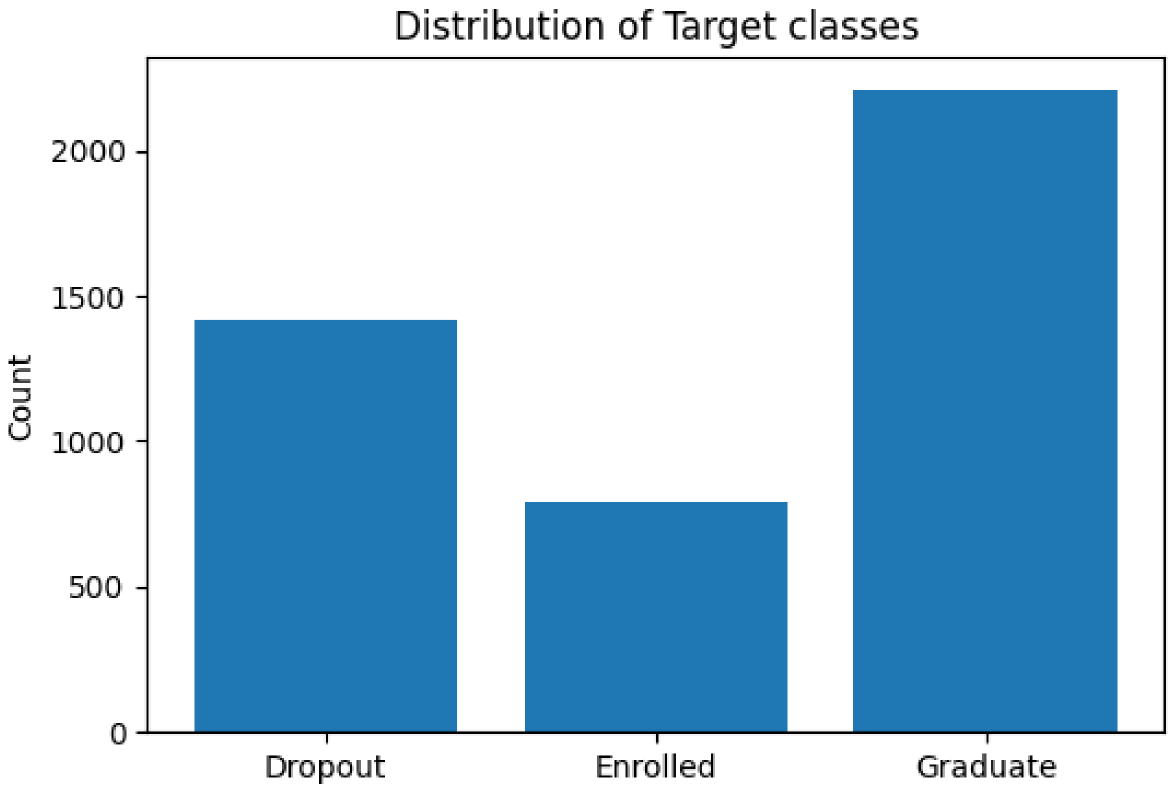


Figure 4: Distribution of target classes

As seen in figure 4, our dataset had very imbalanced target classes. With one class making up half the dataset, we had to modify it so the classes would be weighted fairly. To do this, we chose oversampling. We randomly selected rows in the training set of the underpopulated classes, and duplicated them until each class had the same number of rows.

Oversampling will improve the performance of the models, especially for predicting students with `Enrolled` in the target column. This column made up less than a fifth of the dataset, making it vulnerable to being mostly ignored by models seeking accuracy by prioritizing the more populated `Graduate` class, since they also have very similar data distributions, as seen in figure 3 and #.

We chose oversampling instead undersampling because we thought our dataset wasn't large enough to justify removing almost half of it to balance the classes. We also thought it was more important to correctly predict graduate and dropout over enrolled, as our main goal with the model is to figure out which students are at risk of dropping out.

3. Extract features and justify the methods used

For feature extraction, we used PCA. PCA creates principal components that are linearly independent, meaning a lot of the variance in the dataset can be explained using much fewer components. The components are also sorted by which explain most of the variance in the dataset. This can be seen in figure 5, where the total explained variance increases quickly with few components while the explained variance per principal component quickly approaches 0.

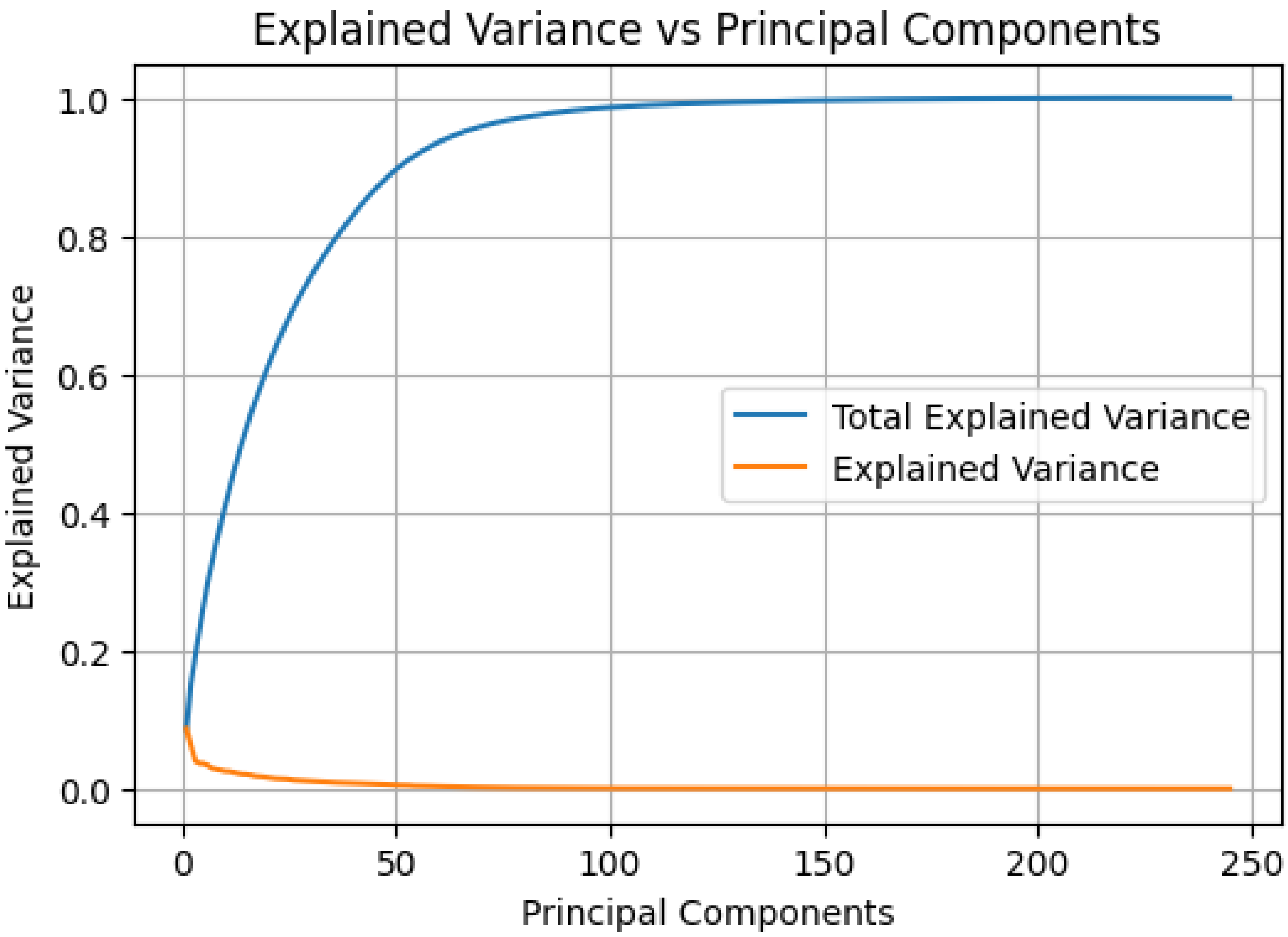


Figure 5: Graph of explained variance for PCA

We can get a lot of the information from this dataset using much less than all our components by selecting all components up until they explain 95% of the total variance in the dataset. Selecting components up to 95% explained variance will use 66 of our components. This threshold and our chosen principal components can be seen in figure 6. Removing most of our components while still keeping 95% variance should make our models have similar performance while cutting down training time.

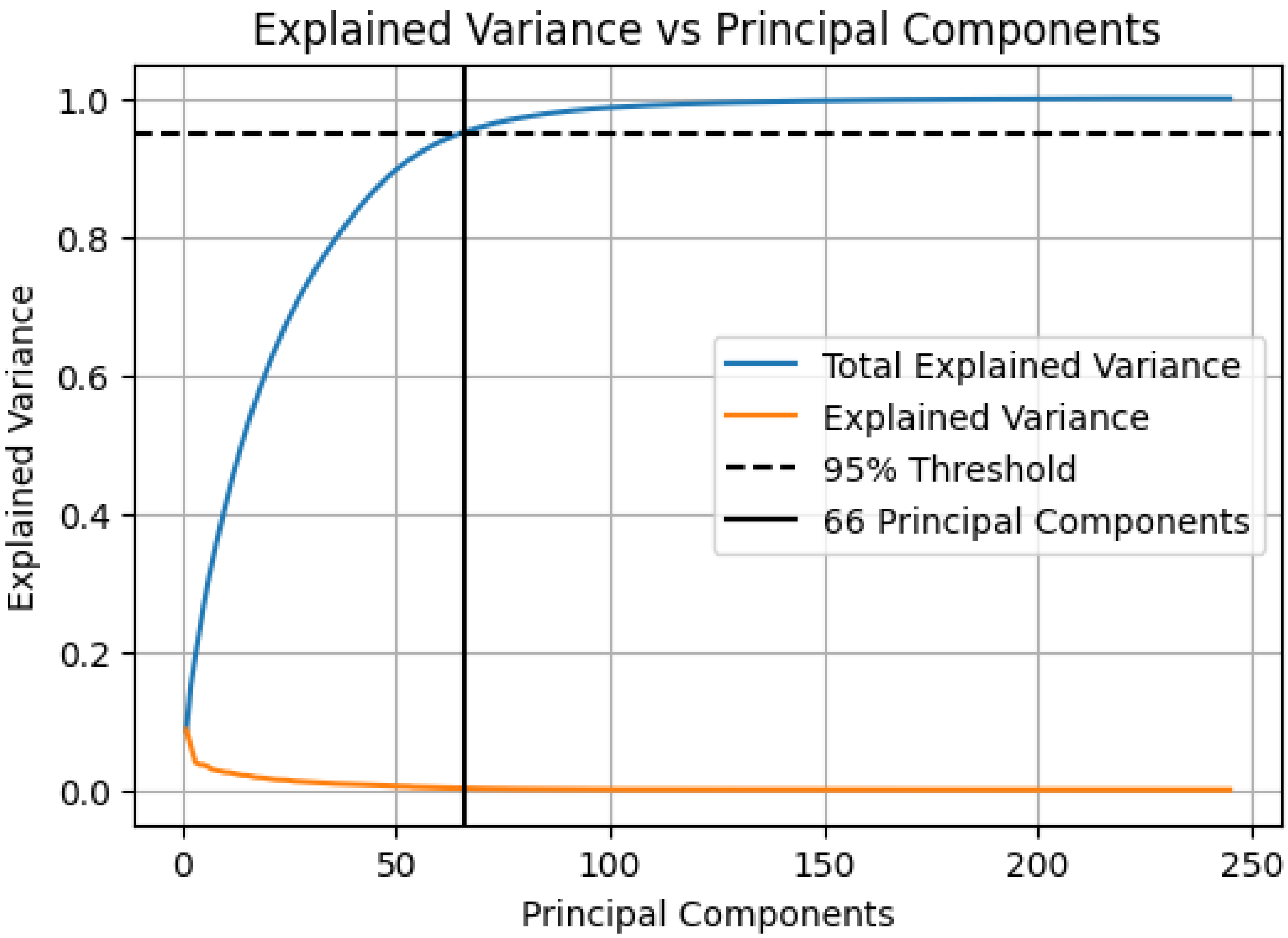


Figure 6: Graph of chosen principal components at 95% variance threshold

PCA is also very useful for visualizing data, as it can show a lot of variance in the first few components, with the drawback of it being hard to understand what the visualization is supposed to represent in the actual dataset. A visualization of our data using PCA can be seen in figure 7.

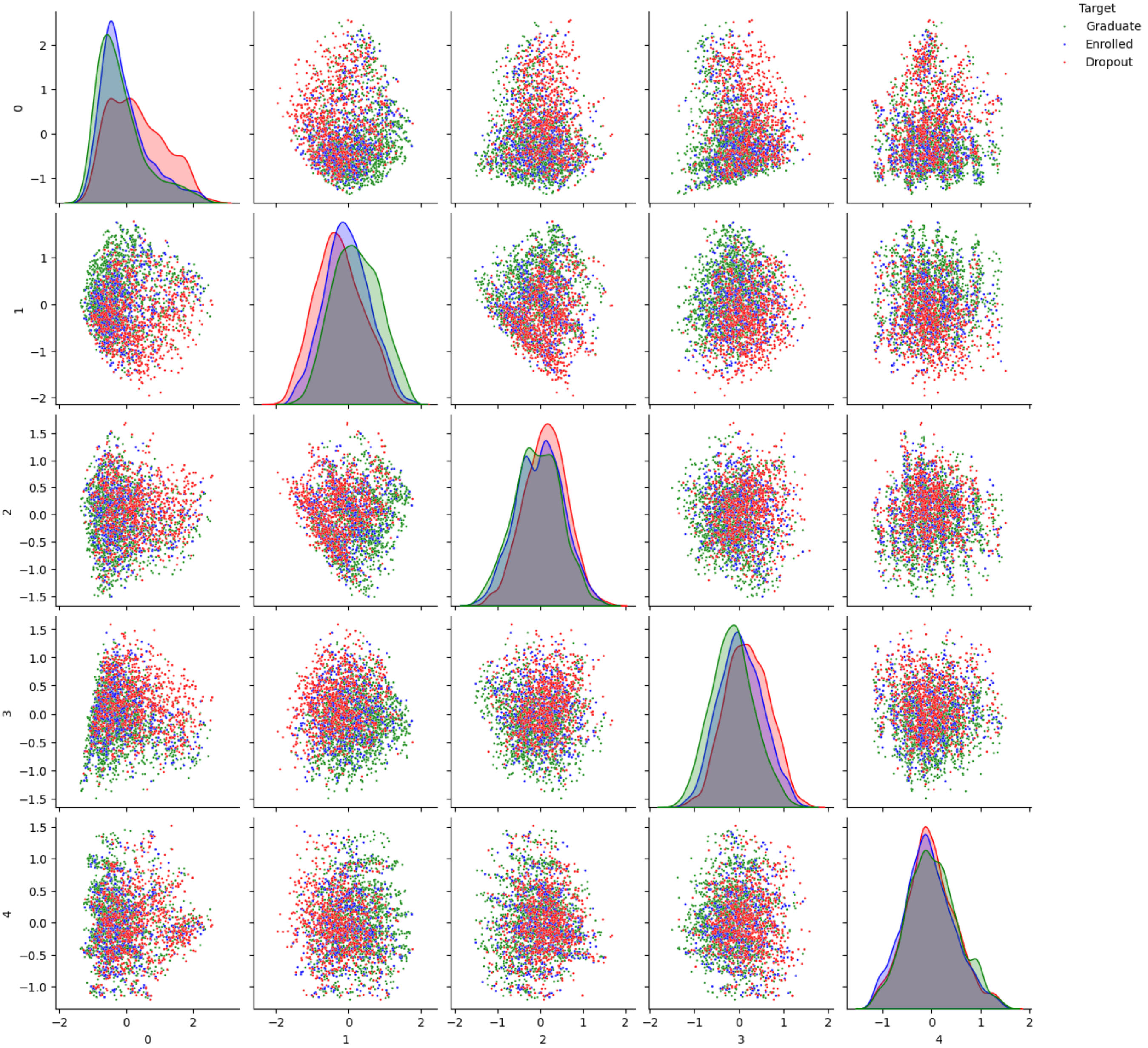


Figure 7: Distribution of data using PCA

4. Select features and justify the methods used

For feature selection, we decided to use our preprocessed dataset (before PCA) and remove one hot encoded columns with a low frequency of rows containing `true`. There are 2 reasons for this. The first is that it should prevent overfitting, as one hot encoded columns with very few rows containing `true` could all be of one target category in the training data, leading to the feature being incorrectly correlated with a certain category. Doing this should improve model performance as they won't be overfitting on the one hot encoded columns.

Another reason to remove columns with a low frequency of `true` is that it will make the dataset less sparse by getting rid of the columns containing the least information. It will reduce the amount of columns in our dataset to something similar to our 95% threshold of explained variance in PCA. This will reduce training by removing a lot of our sparse columns.

We decided to remove columns with a frequency of `true` less than 3%, as this requires ~100 rows containing it. This will definitely prevent overfitting on one hot encoded columns, while also removing a large chunk of our columns without much data.

Removing columns with a threshold of 3% removed 167 columns from our dataset. This left us with 78 features, which is more than the dataset before one-hot-encoding, but has made the categories easier to understand for our models.

We can also look at the distribution of our target categories using the columns chosen from our feature selection, even though the variance of the data can't be visualized as clearly as with PCA in figure 3. This visualization is shown in figure 8.

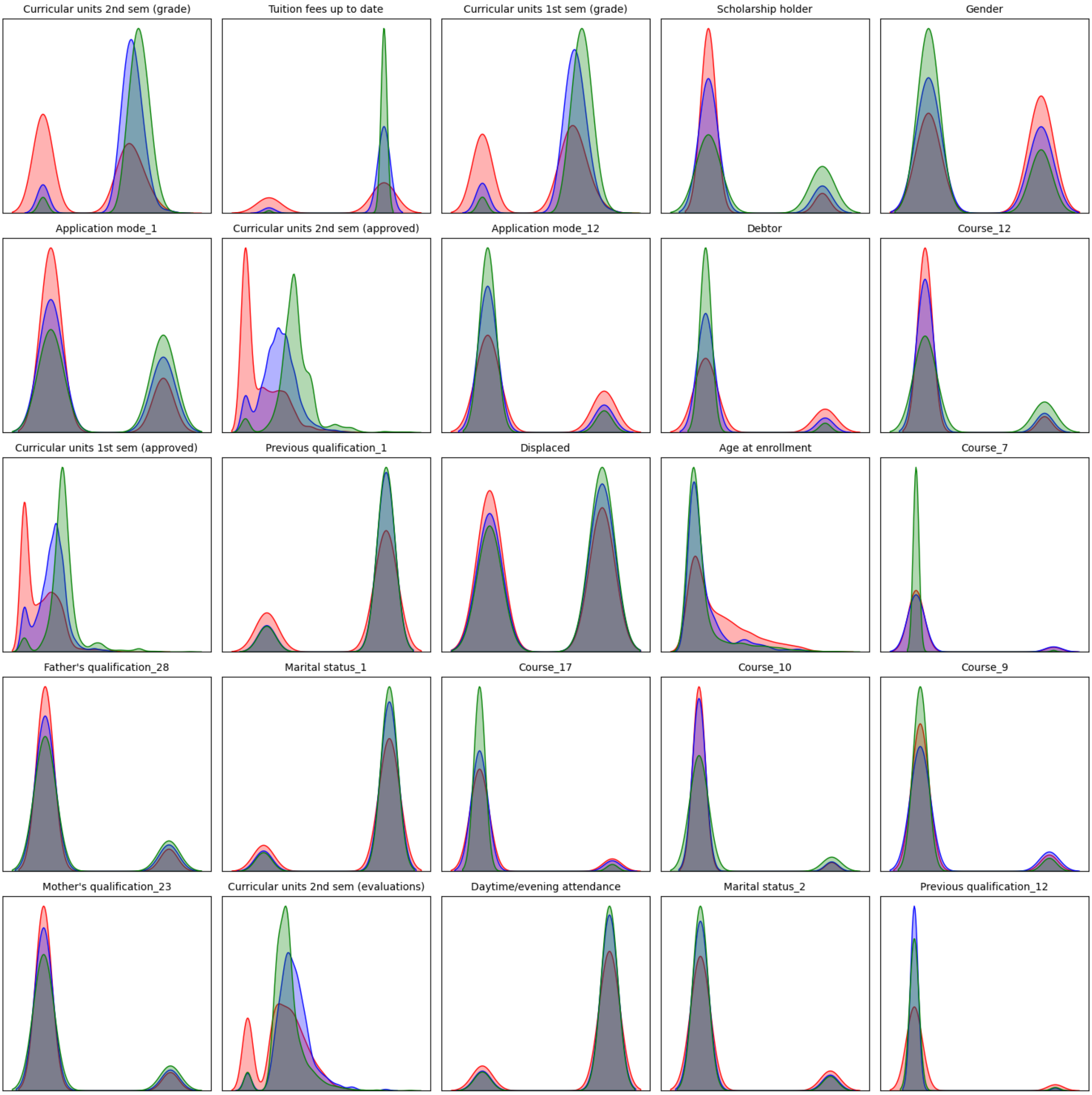


Figure 8: Distribution of data using features sorted by highest variance

To decide whether to use our dataset after feature extraction using PCA or after feature selection, we trained one of the models that don't take that long to train, SVM, with all the variations of our dataset.

From this point on in this document, the following label numbers, \$0\$, \$1\$, \$2\$, displayed inside of figures represent `Dropout`, `Enrolled` and `Graduate` respectively.

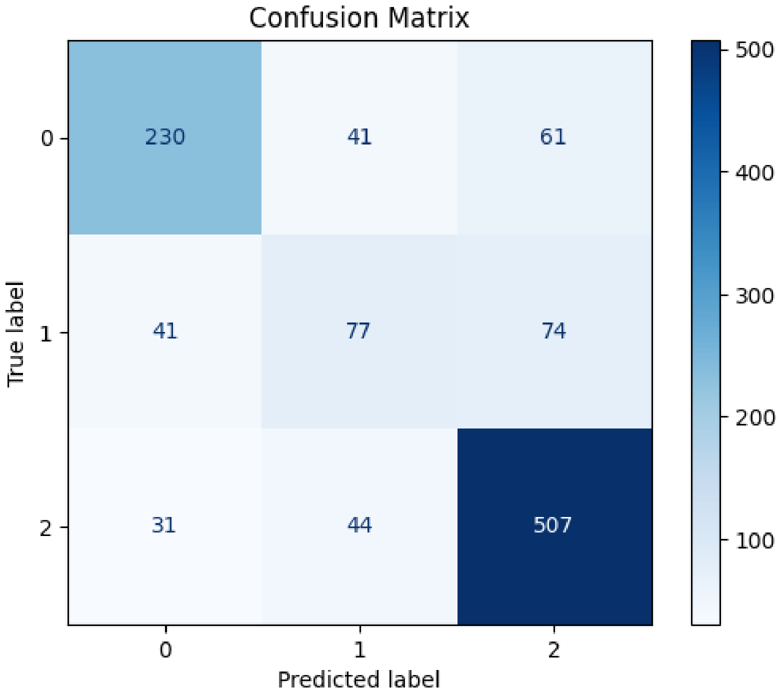


Figure 9: Confusion matrix for SVM (PCA, all principal components)

Figure 9 shows the confusion matrix for SVM using all principal components from our PCA. It has an accuracy of 73.6%, taking 1 minute and 57 seconds to train.

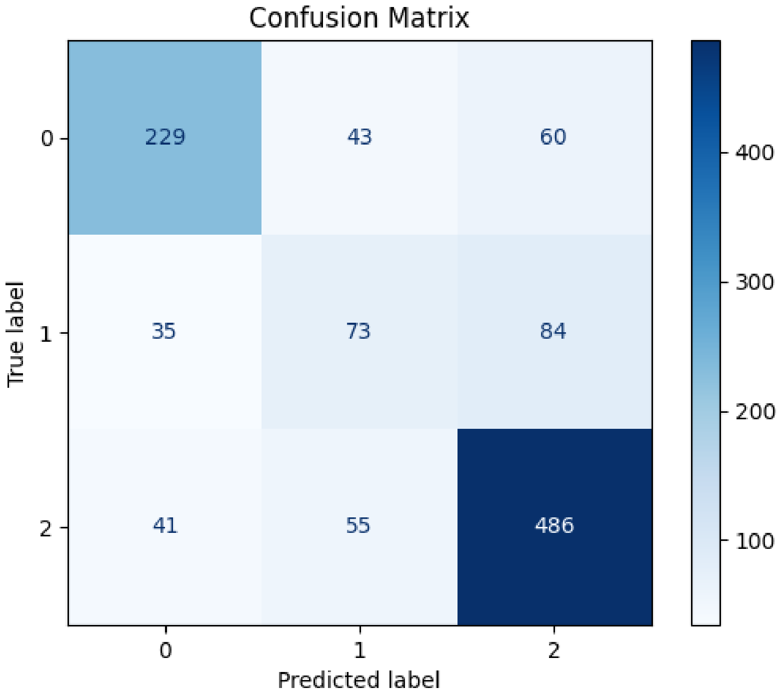


Figure 10: Confusion matrix for SVM (PCA, 95% explained variance threshold)

Figure 10 shows the confusion matrix for SVM using the principal components from the 95% explained variance threshold after PCA. It has an accuracy of 71.2%, taking 32 seconds to train.

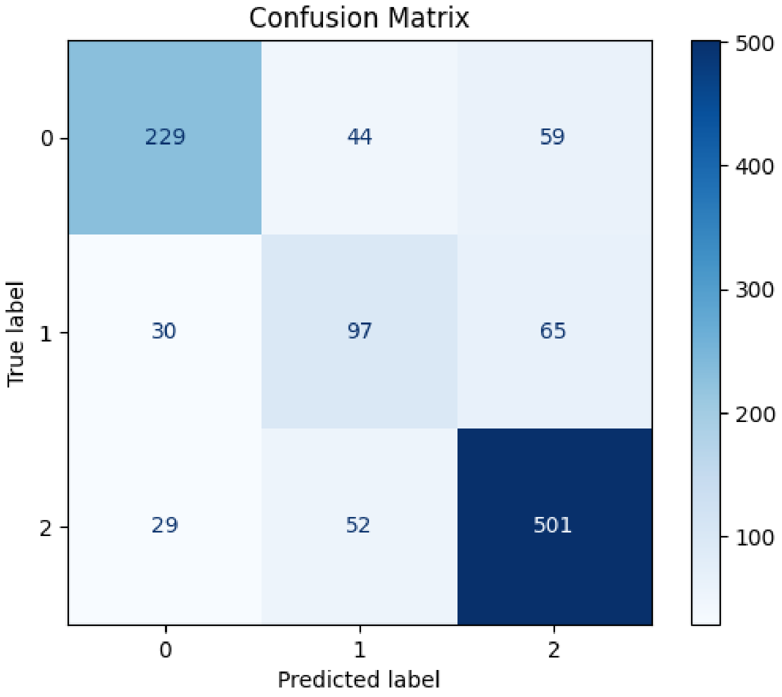


Figure 11: Confusion matrix for SVM (all features)

Figure 11 shows the confusion matrix for SVM using all preprocessed features (one hot encoded and scaled, not PCA). It has an accuracy of 74.8%, taking 1 minute and 40 seconds to train.

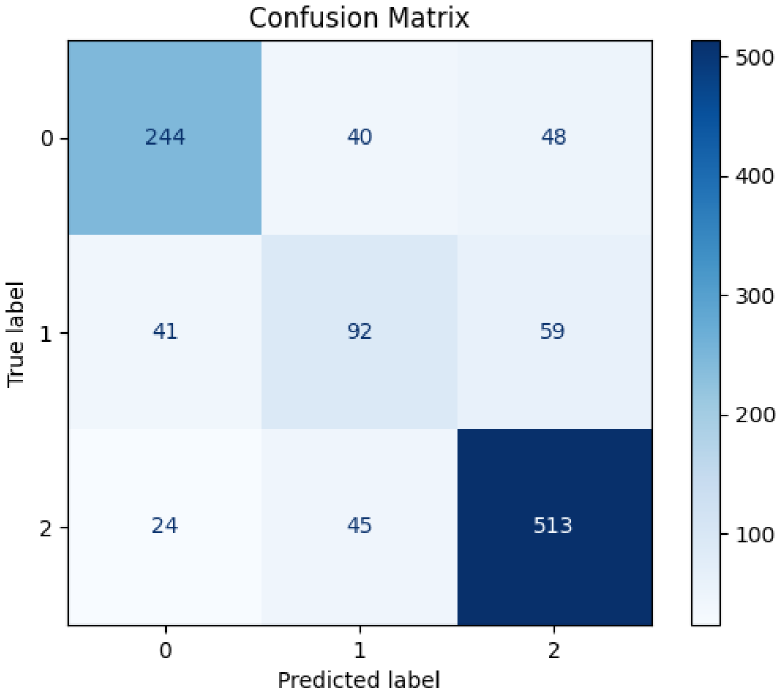


Figure 12: Confusion matrix for SVM (feature selection)

Figure 12 shows the confusion matrix for SVM using features kept after feature selection. It has an accuracy of 76.8%, taking 41 seconds to train.

As expected, feature selection has both better performance and takes less time to train than with all features, but we found it surprising how much better it performed than using PCA. This might be a result of features with little variance being more important for the classification of our target class than the features with more variance. If this is the case, it would make sense that removing low-variance features from our PCA would decrease performance. It would also explain why the preprocessed dataset without PCA and the dataset with the all principal components had much more similar performance.

After looking at the results, we decided to use feature selection over PCA, as it has much better performance and the features are more understandable.

5. Implement five out of the following algorithms and justify the choice

a. Logistic regression

How it works

Multinomial logistic regression models the log odds of each class as a linear function of the inputs and uses a softmax function to output class probabilities.

Why we chose it

Multinomial logistic regression is a strong baseline for multiclass classification, it works well with our one-hot encoded categorical features. It fits well for target classification when the target classes are unordered, as they are in our dataset (The target can be viewed as having a intuitive order, but in our view it's not a scale of bad-good in the same way that low-medium-high could be viewed). A known limitation is the linearity assumption, which can miss non-linear patterns.

b. Decision trees

How it works

A decision tree can be seen as a tree of different choices, with each leaf node in the tree corresponding to a target class. Each node/choice in the tree will split the data that reaches it between the later nodes, which will then further separate the data until it reaches a target class.

Why we chose it

Decision trees are a good fit for classification tasks because they can capture both linear and non-linear relationships and splits the data in an intuitive way. They are also easy to understand the inner workings of, as they just go through the tree doing if/then checks. One problem with decision trees is that they can easily be overfitted if not given correct hyperparameters.

c. Random forest

How it works

A random forest builds many decision trees on bootstrap samples (samples of randomly selected data that include removing and duplicating data points) while randomly selecting subsets of features at each split. The final prediction is the majority vote across trees.

Why we chose it

It usually delivers higher accuracy than a single decision tree and handles many attributes well, including our one-hot encoded features and mixed numeric inputs. Although ensembles can be computationally heavier, our dataset is small enough that training won't take too long. The ensemble voting and bootstrap samples make the majority vote of all the slightly overfitted decision trees less prone to overfitting and often more accurate.

d. SVM with kernels

How it works

A support vector machine (SVM) tries to draw a line/boundary that best separates classes by keeping the distance to the closest points as large as possible. With kernels like the radial basis function, it implicitly maps data to a higher dimensional space to separate complex patterns. Only the support vectors, or the points closest to the boundary, determine its position.

Why we chose it

Support vector machines performs well in the high-dimensional feature spaces created by one-hot encoding and often achieves strong accuracy when appropriately regularized. At prediction time it's efficient and reasonably memory-friendly, since the decision function only depends on a subset of the training points (the support vectors). Training can be slow on very large datasets, but for our dataset size this isn't a problem.

e. Neural network - MLP

How it works

MLP is a feedforward neural network that stacks linear layers, each node being a linear transformation of all nodes in the previous layer, with nonlinear activations that allow the model to recognize non-linear relationships. The model tweaks parameters by backpropagation. For multi class outputs it ends with a softmax layer that produces probabilities of each class.

Why we chose it

Our problem is a multi class classification problem with three target categories to classify. As a result we chose to implement our neural network with a MLP classifier as they are designed for multi class problems. Furthermore, neural nets can also learn complex interactions among our features that simpler linear models might miss.

6. Compare the performance of the five algorithms with respect to your problem, explain the results

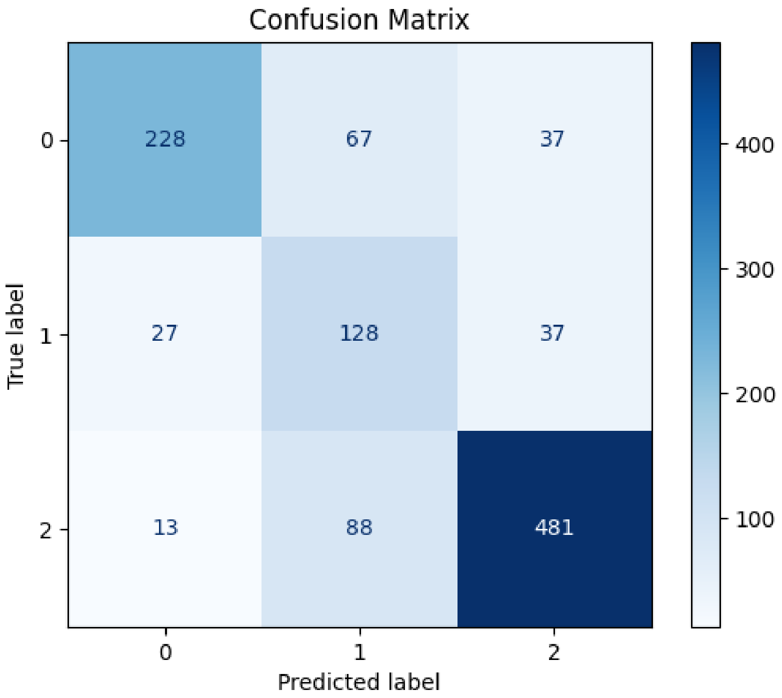


Figure 13: Confusion matrix for logistic regression

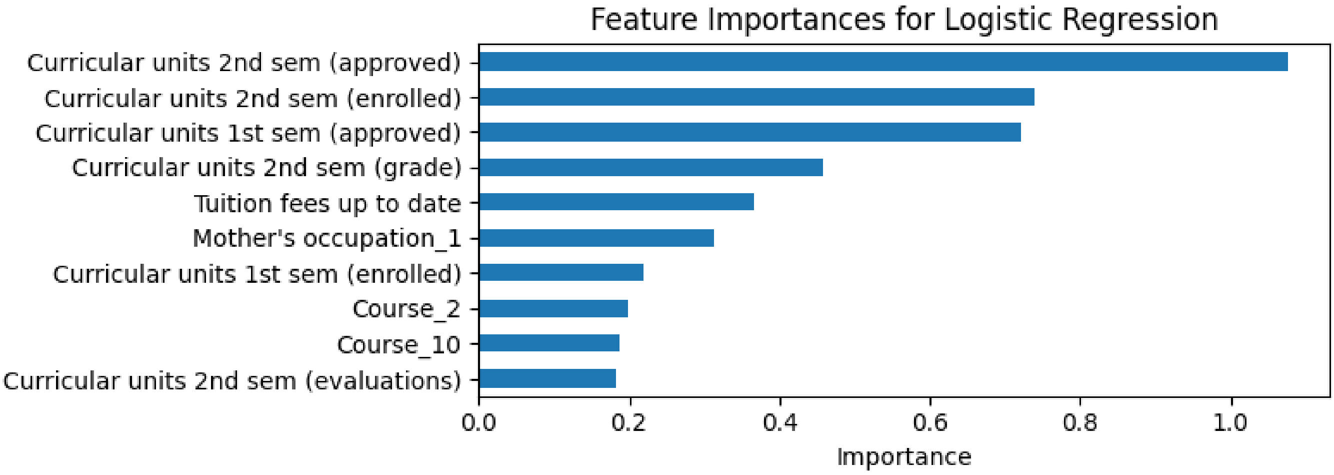


Figure 14: Feature importances for logistic regression

In figure 13, we can see the confusion matrix of our logistic regression model. It achieved an accuracy of 75.7% and it correctly predicted a majority of the students that have graduated. However, it performed worse in regards to predicting dropout, but was still accurate in most cases. When predicting students that would still be enrolled beyond normal completion time it misses 50% of the time. One important note for this model is that it is the most accurate of the basic models when predicting the enrolled category.

As seen in figure 14 we can see that the most important feature for logistic regression in determining classifications were the different curricular features. How many curricular units a student has gotten approved, enrolled in and what grade was achieved, were more important for making predictions.

Two course classes are also in the top ten feature importance, which is expected because different courses will have different dropout/enrolled/graduate ratios.

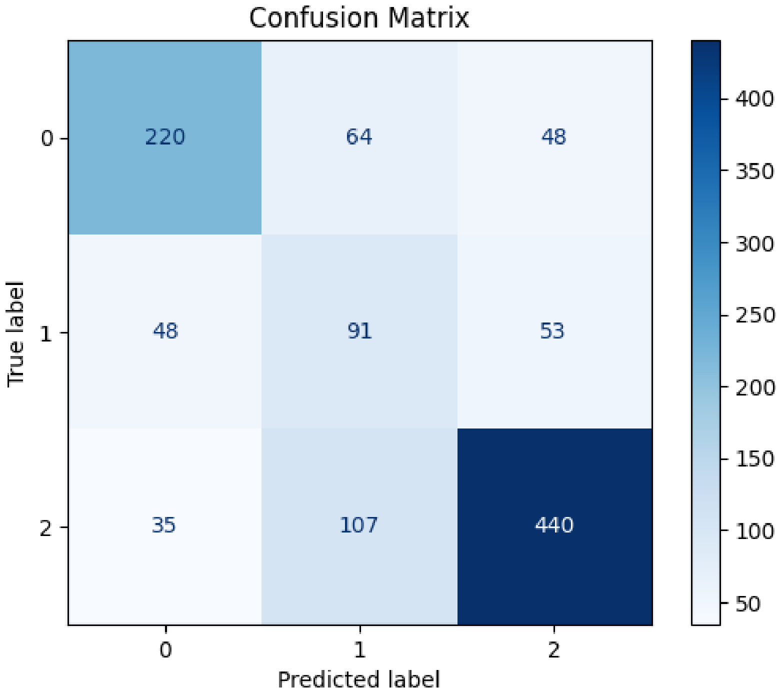


Figure 15: Confusion matrix for decision tree

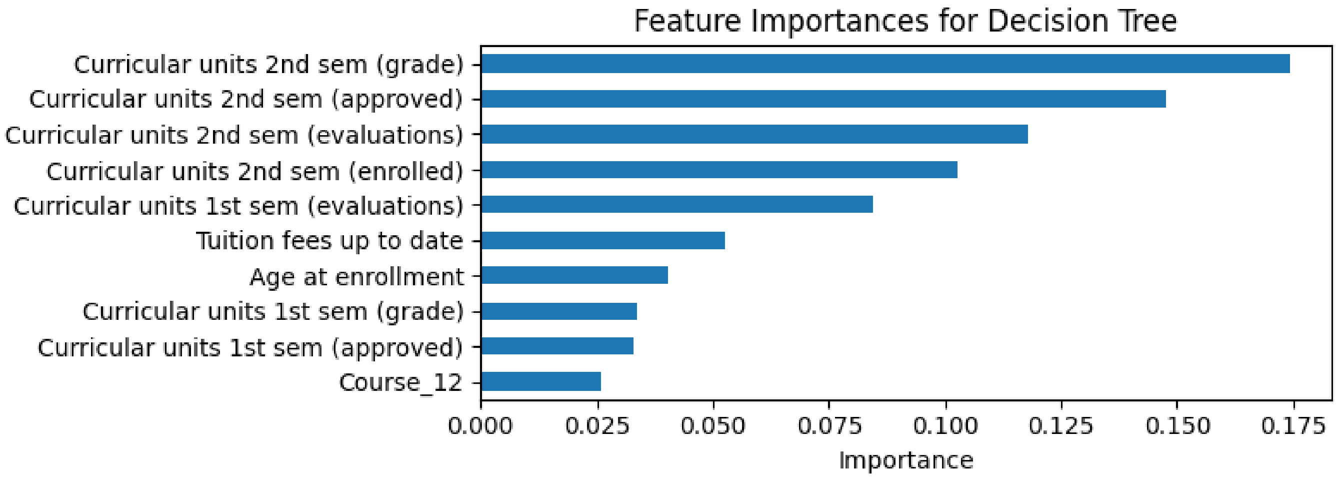


Figure 16: Feature importance for decision tree

Our decision tree model achieved a 67.9% accuracy and figure 15 displays the confusion matrix for the model. When predicting dropout it performed about the same as the logistic regression model, but performed slightly worse when predicting graduates and enrolled students.

In figure 16 we can see that like logistic regression, the decision tree model relied mostly on the different curricular features when making its predictions. It had both tuition fees up to date and age at enrollment in its top ten features, showing that factors other than academic performance matter to the model even though this may not be inherently obvious.

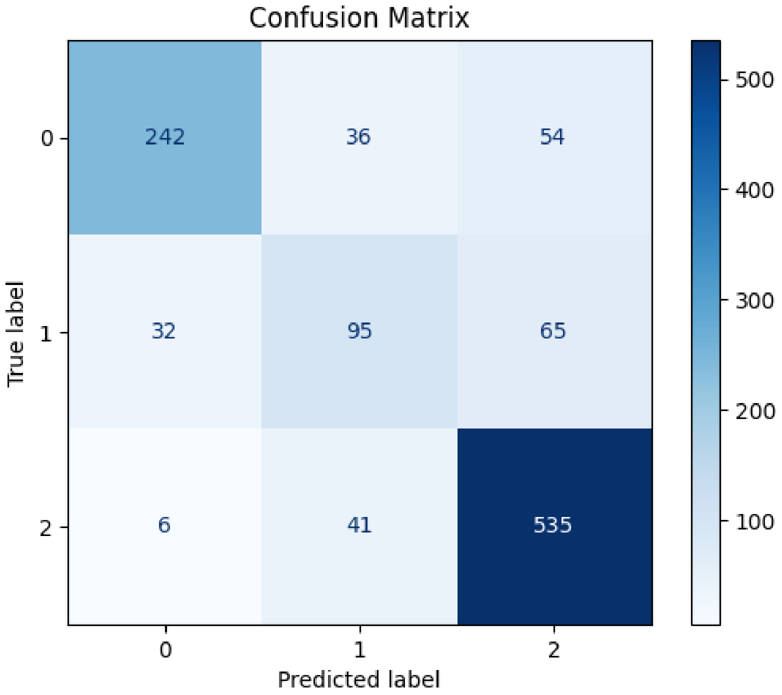


Figure 17: Confusion matrix for random forest

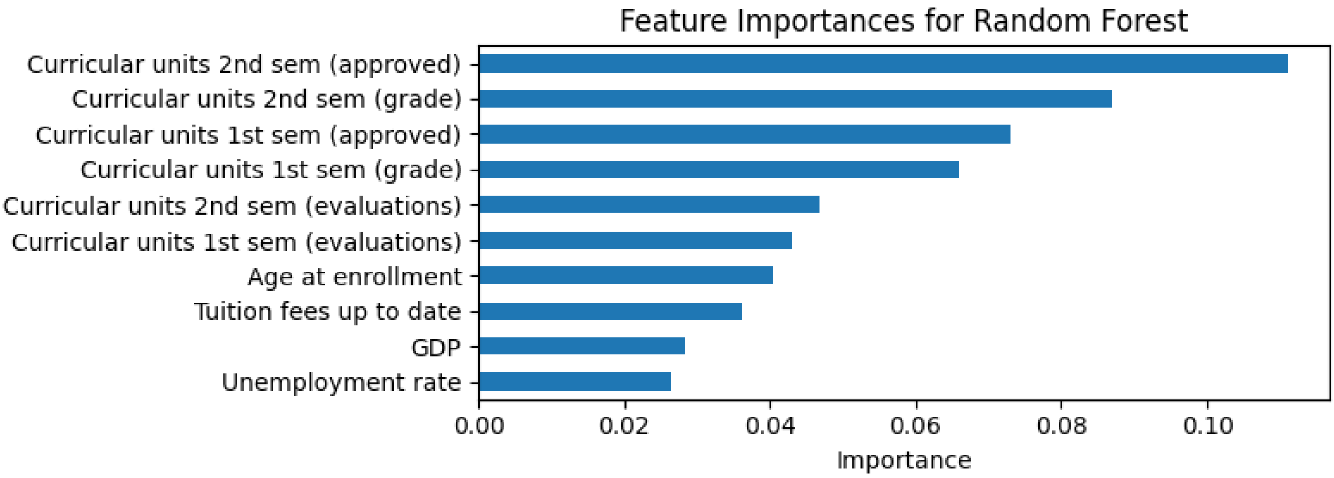


Figure 18: Feature importance for random forest

Our random forest model implementation achieved an accuracy of 78.8%, our best performing model. As seen in figure 17 it is very accurate in predicting students that graduate. Random forest has a better overall accuracy than all other models although it performs worse at predicting enrolled compared to logistic regression.

Additionally, when we analyze the top then feature importance for our random forest model we see similarities to the previous models. Interestingly, this marks the first instance of a model prioritizing GDP and unemployment rate, encoding for information outside of the student.

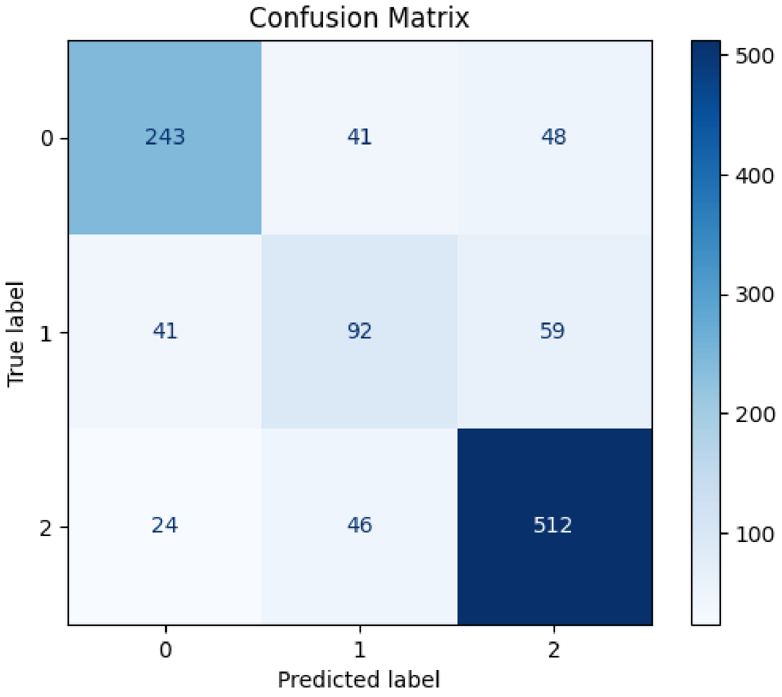


Figure 19: Confusion matrix for SVM with RBF kernel

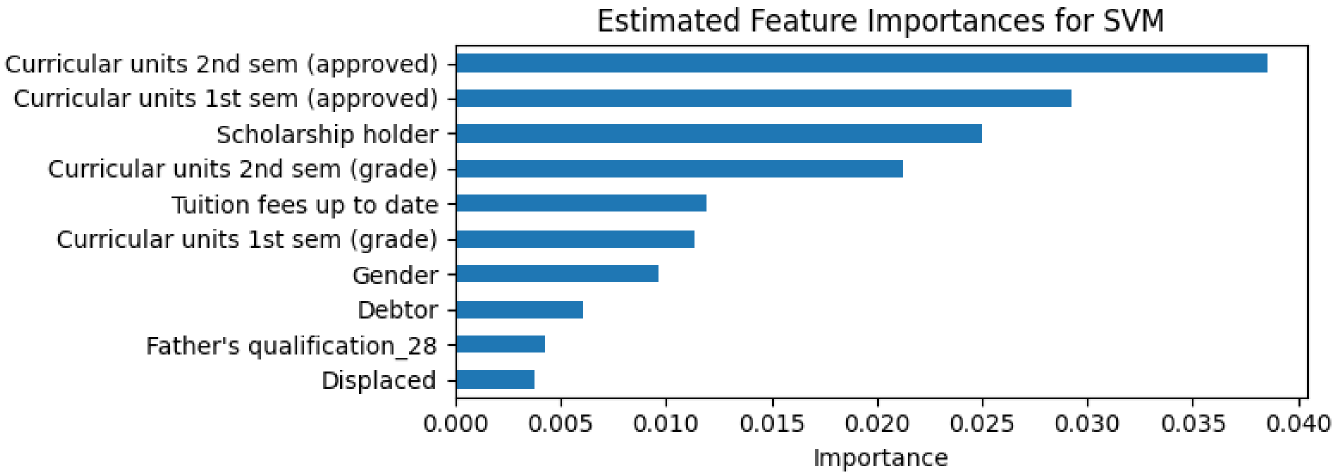


Figure 20: Feature importance for SVM with RBF kernel

In figure 19 we can see that the SVM model performs almost as well as random forest. It achieved an accuracy of 76.6% and predicts graduate and dropout correctly a majority of the time, but struggles to predict enrolled like most of the previous models.

When we estimated feature importance it revealed that also SVM relies on mostly the same features to make its predictions. The most important being the curricular features. In figure 20 we also see, with comparison to the previous models, the addition of debtor, gender, displaced, scholarship holder, and a father's qualification feature as a model's top feature importance. Here we notice that features not related to academic performances, albeit not as important, do contribute to model's predictions.

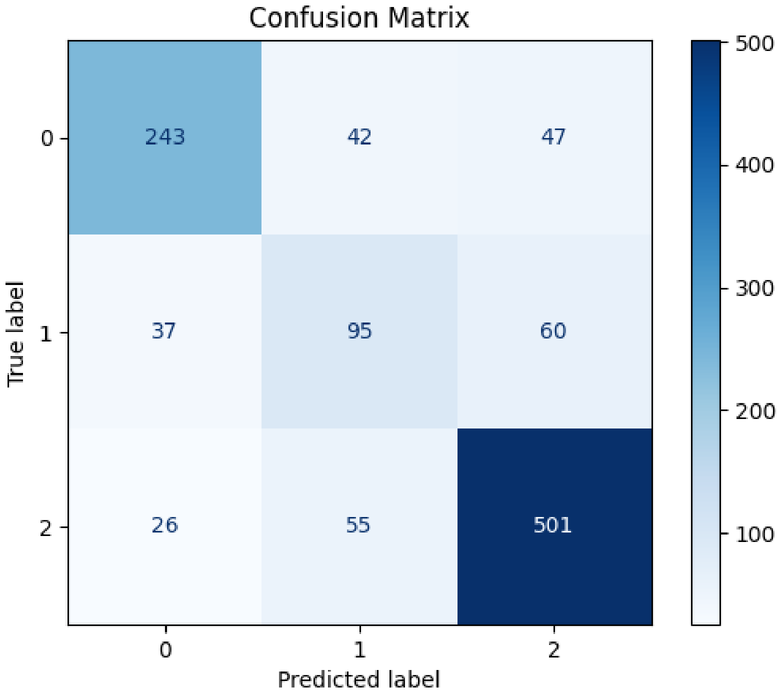


Figure 21: Confusion matrix for MLP

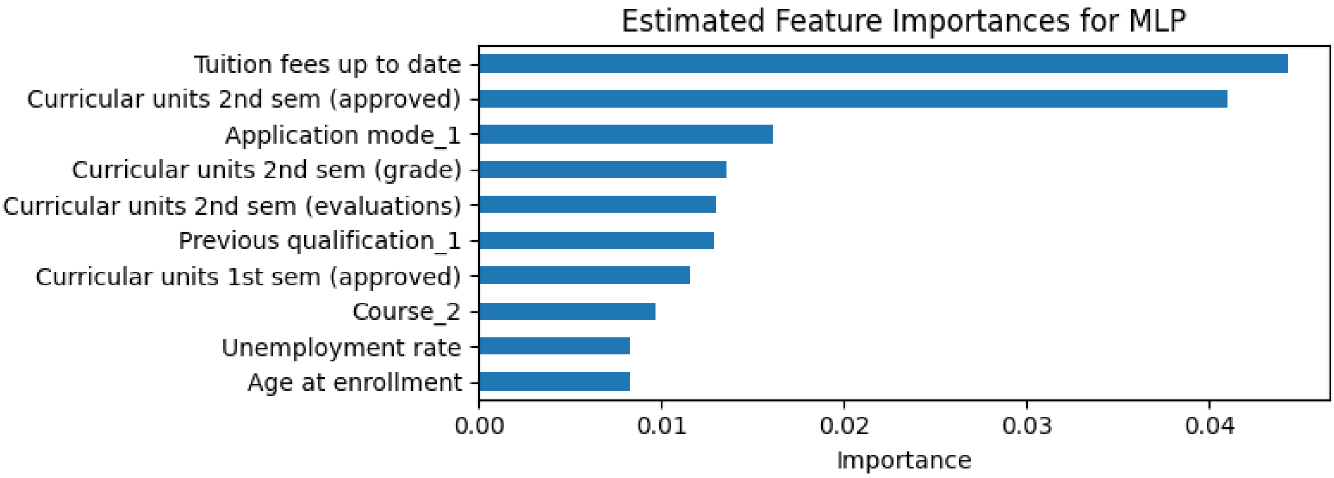


Figure 22: Feature importance for MLP

Lastly, we have our MLP neural network model. It achieved a 75.9% accuracy and performed well when predicting graduate and dropout. Similarly to previous models it also struggled with predicting enrolled, as it missed about 50% of the time on this classification.

In our estimated feature importance for our MLP model, we see in figure 22, that tuition fees up to date is the most important feature. Like the other models it also utilized the curricular features to make predictions. Here we also noticed that it had two previously unseen features in its top ten feature importance, these were a previous qualifications feature and an application mode feature. It is interesting to note that factors other than academic performance not only contribute more than in previous models, but are actually more important. Note that these are estimated feature importance based on permutation importance, and might not match the actual importance.

Overall comparison

Because we see features like approved, enrolled, and graded curricular units repeating in each model's feature importance, we can confidently say they have a high correlation with the target in our dataset compared to the other features. We thought of this as an intuitive correlation, since we thought getting many curricular units approved and achieving good grades would have a high correlation to finishing your course within its normal span.

We also noticed data from the second semester is more relevant than data from the first semester, which is also intuitively understood, as more up-to-date information would have a higher correlation with the results later in time.

Socio-economic factors like tuition fees, unemployment rate, debtor, and GDP also repeat in the feature importance of the models. This category of features seems to be the next most important factor after school performance for predicting our target. Even though we can't see how the target categories are influenced by these features in the feature importance graphs above, we think economically disadvantaged students are probably shifted more towards `Dropout` and `Enrolled` than the average. This leads us to think that having an overview of the students' socio-economic status would be important for figuring out which students likely need more help to get back on track in their academic trajectory.

For figuring out which students likely need more academic help or other interventions, the most important factor is of course their academic performance, but as seen from the feature importance of the models above, other factors are also important to get the most accurate predictions. Consequently, this could lead to a more appropriate use of the institution's resources when helping students.

7. Implement boosting and bagging with your choice of base models and explain all the steps

We implemented several ensemble learning methods: **Bagging with Logistic Regression, MLPs, SVMs, and Decision Trees**, as well as **AdaBoost with Logistic Regression and Decision Trees**.

Bagging models train multiple independent base learners on bootstrap samples, meaning each use random subsets of observations and features. The ensemble prediction is generated through majority voting, reducing variance and improving stability. Logistic Regression, MLPs, SVMs, and Decision Trees serve as base learners.

AdaBoost models build ensembles sequentially, reweighting misclassified samples so that later learners focus on harder cases. Using Logistic Regression and Decision Trees as base learners, AdaBoost reduces bias by combining their weighted predictions.

Note that **Random Forest** already uses bagging, so bagging it again is redundant.

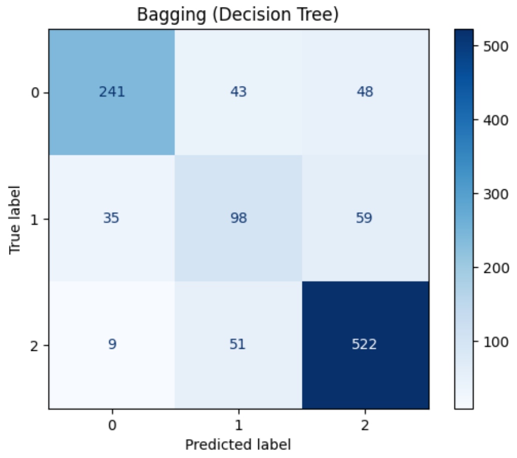


Figure 23.a: Confusion matrix (Bagging with Decision Trees)

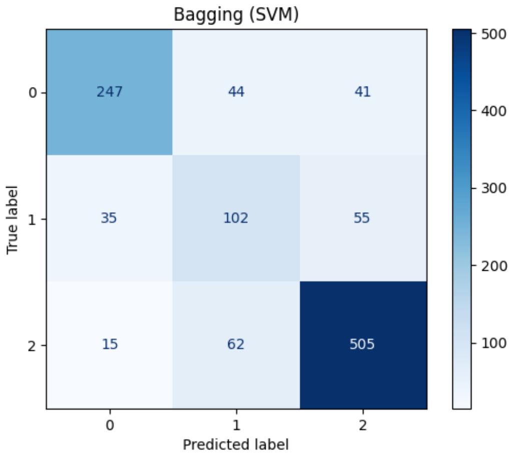


Figure 23.b: Confusion matrix (Bagging with Support Vector Machines)

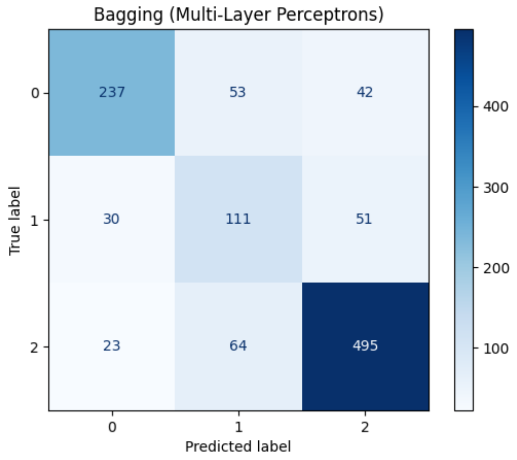


Figure 23.d: Confusion matrix (Bagging with Multi-Layer Perceptrons)

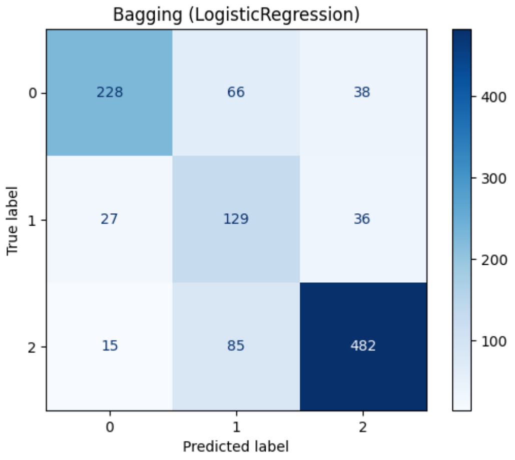


Figure 23.c: Confusion matrix (Bagging with Logistic Regression)

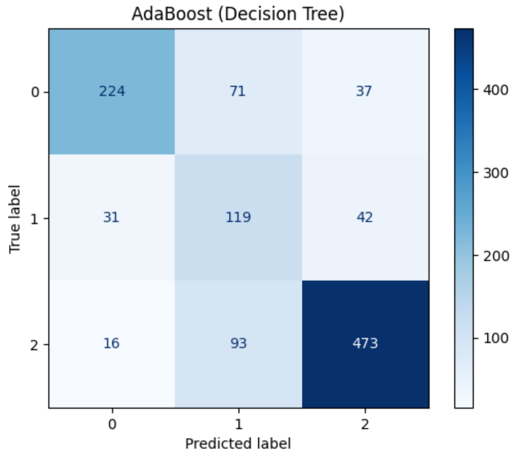


Figure 23.e: Confusion matrix (AdaBoost with Decision Trees)

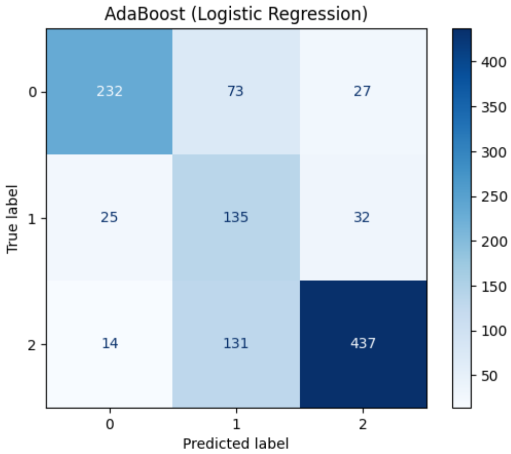


Figure 23.f: Confusion matrix (AdaBoost with Logistic Regression)

As seen in the confusion matrices above, the number of correct predictions each model makes varies with bagging and boosting.

When it comes to bagging, using it on `Decision Trees` seems to work best. The other models also perform relatively well.

Boosting, on the other hand, performs overall worse than bagging. `Decision Trees` once again performed best. `Boosting with Logistic Regression` performed the worst.

Bagging tends to outperform boosting on our dataset because these it contains several less relevant features, causing boosting to overfit misclassified or ambiguous cases. Bagging instead reduces variance by averaging multiple independent models, making it more robust and better suited to the structure and quality of our features.

Despite the usage of boosting or bagging, the best performing models are still biased towards graduates.

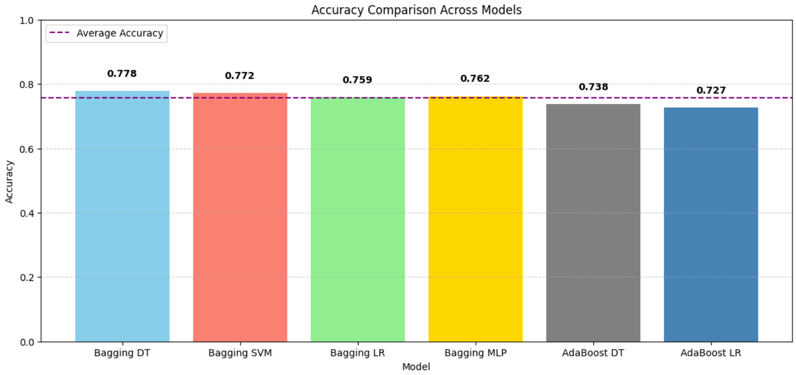


Figure 24a: Bagging vs Boosting accuracy

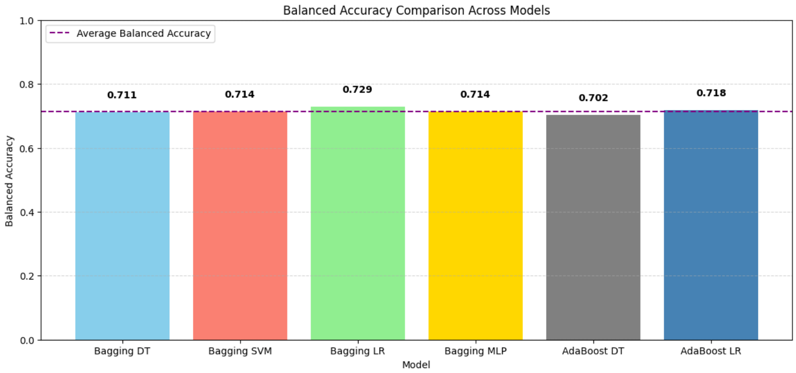


Figure 24b: Bagging vs Boosting balanced accuracy

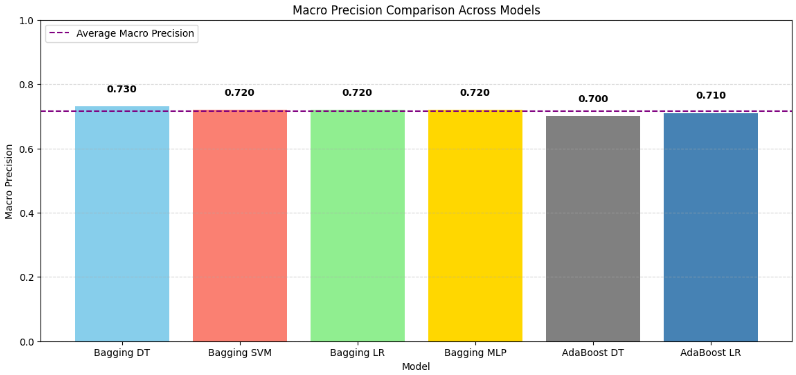


Figure 24c: Bagging vs Boosting macro precision accuracy

From ***Figure 24a***, we notice that **accuracy** ranges from 0.727 to 0.778 across all models. The best model, `Bagging DT`, correctly predicts about 78% of students, which is just as good as `Random Forest`, which is expected since the structure of `Bagging DT` is very similar to `Random Forest`. The worst model is `AdaBoost LR` with 0.727 accuracy.

However, accuracy can be misleading because the classes are unbalanced since many students are graduates. To resolve this, we have also used **Balanced Accuracy**.
Balanced accuracy goes from 0.702 to 0.729 which is lower than the raw accuracy (see ***Figure 24b***).
This indicates that the bagged and boosted models predict graduates and dropouts well, but struggle to predict enrolled students. Interestingly, `Bagging LR` performed the best here. This is explained from the fact base Logistic Regression outperformed all other base models when predicting enrolled students.

Finally, **Macro precision** goes from 0.7 up to 0.73 (see ***Figure 24c***). Once again, `Bagging DT` performed the best. This means that when the model predicts a class, it is correct roughly 73% of the time.
`AdaBoost DT` performed the worst, because its base learners cannot model the complexity of our imbalanced dataset, causing more misclassifications and therefore more false positives in multiple classes.

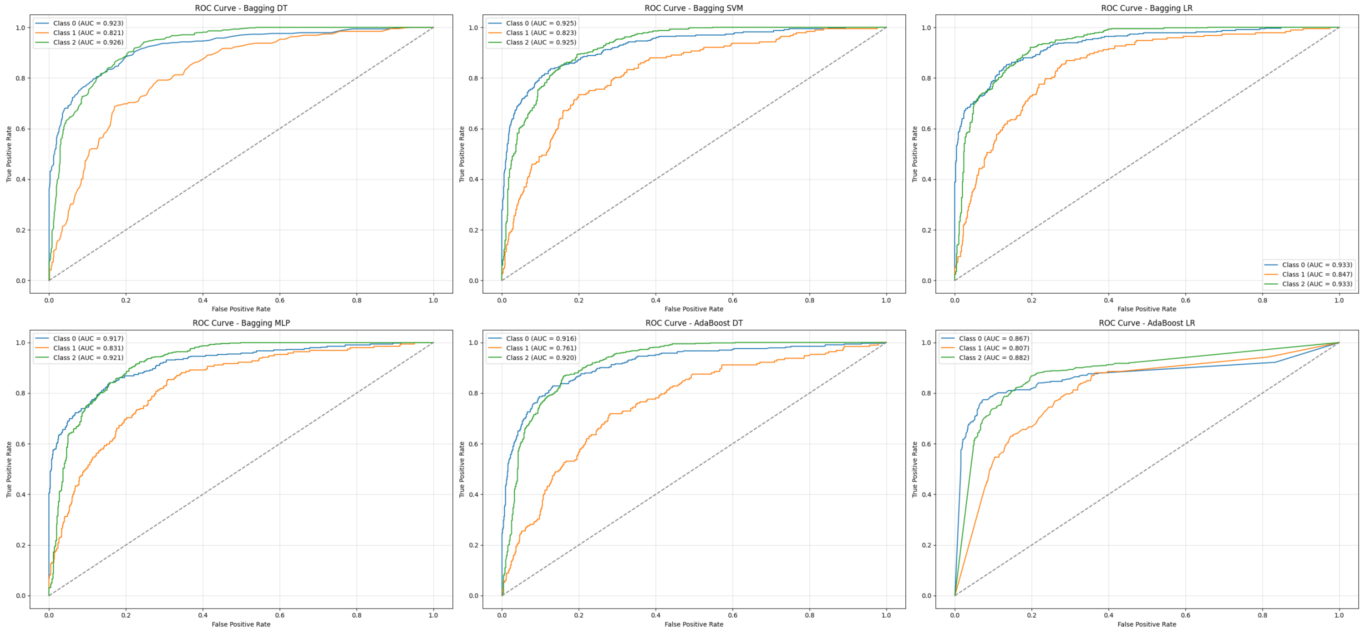


Figure 25: ROC graphs for all models

ROC-AUC measures how well each model separates classes. Our models range from 0.867 to 0.933 when predicting dropout and graduates which is a very good result. When predicting enrolled students, the models range from 0.761 to 0.847, which is significantly worse, explained by the fact that enrolled students have overlapping features with the other classes. This overlap is observed in ***figure 7***, which uses PCA to visualize features which is still useful in this case even though we don't use the PCA dataset.

Bagging Decision Trees performs best overall, while Bagging MLPs and SVMs do well, and Bagging Logistic Regression excels in ROC-AUC, due to its more equally distributed classification curves.

8. Implement one instance of transfer learning (find a related bigger dataset online) and explain all the steps

First we looked online to find a dataset related to our student-graduation dataset. We were only able to find a single larger dataset related to student graduation/performance. The dataset we found was Student Performance & Behavior Dataset found on Kaggle.

Unfortunately this dataset shares very few features with our dataset, but as previously stated it was the only somewhat related and larger dataset we could access. This meant we had to map seemingly correlated features between the dataset based on our own intuition on what makes sense. The features where we could not make any sensible mappings had to be dropped so only the set of overlapping features between the dataset were used for this transfer learning.

Since we had to drop the non overlapping features from the new dataset, this meant the pretrained model was trained on a small fraction of the original data. As a result, not even this pretrained model reached a desirable accuracy. When transferring this pretrained model over to our original dataset (with only overlapping features) the accuracy got even worse. This was probably because what we considered to be the most logical mappings between the dataset features were not very accurate.

Due to these limitations we consider transfer learning to be an unsuitable training approach for a model predicting our dataset (given the other dataset we were able to find).

a. Explain the bigger dataset with visualization and summary statistics.

Age and gender were features in both datasets so they were kept as they were.

Our main dataset had two features named "Curricular units 1st sem (grade)" & "Curricular units 2nd sem (grade)". We chose to use the "Quizzes_Avg" feature in our found dataset to map to these features based on the intuition that higher average score on quizzes would correspond to higher grades in the student's semesters. We transformed the value range of this feature from 0-100 to the range 0-20.

We also chose to map the feature "Family_Income_Level" in our found dataset over to the "Debtor" feature in our main dataset. Choosing families with low income as debtors.

Furthermore, we also mapped the feature "Family_Income_Level" == 'High' in the found dataset over to "Tuition fees up to date" in the original based on the assumption that if a student comes from a high earning family they would have paid all tuition fees on time.

Finally, the "Grade" feature in the found dataset was chosen to become the "Target" column where A & B grades where mapped to graduate, C to enrolled if the students stress level was above 5 and graduate if it was below, D & F to dropout. This was based on the assumption that the poorer(D & F) grades would be an accurate mapping of dropout students, and the best grades(A & B) would be an accurate mapping to graduate. We also made an assumption that a C grade would be graduate if they were adequately calm as measured by the self reported lower stress levels. Consequently, we also thought they would be enrolled if they had higher stress levels indicating they where struggling to maintain a C grade. We do concede that these assumptions are flawed, but we thought this to be the most sensible way split the data to maintain a reasonable distribution of the target classes.

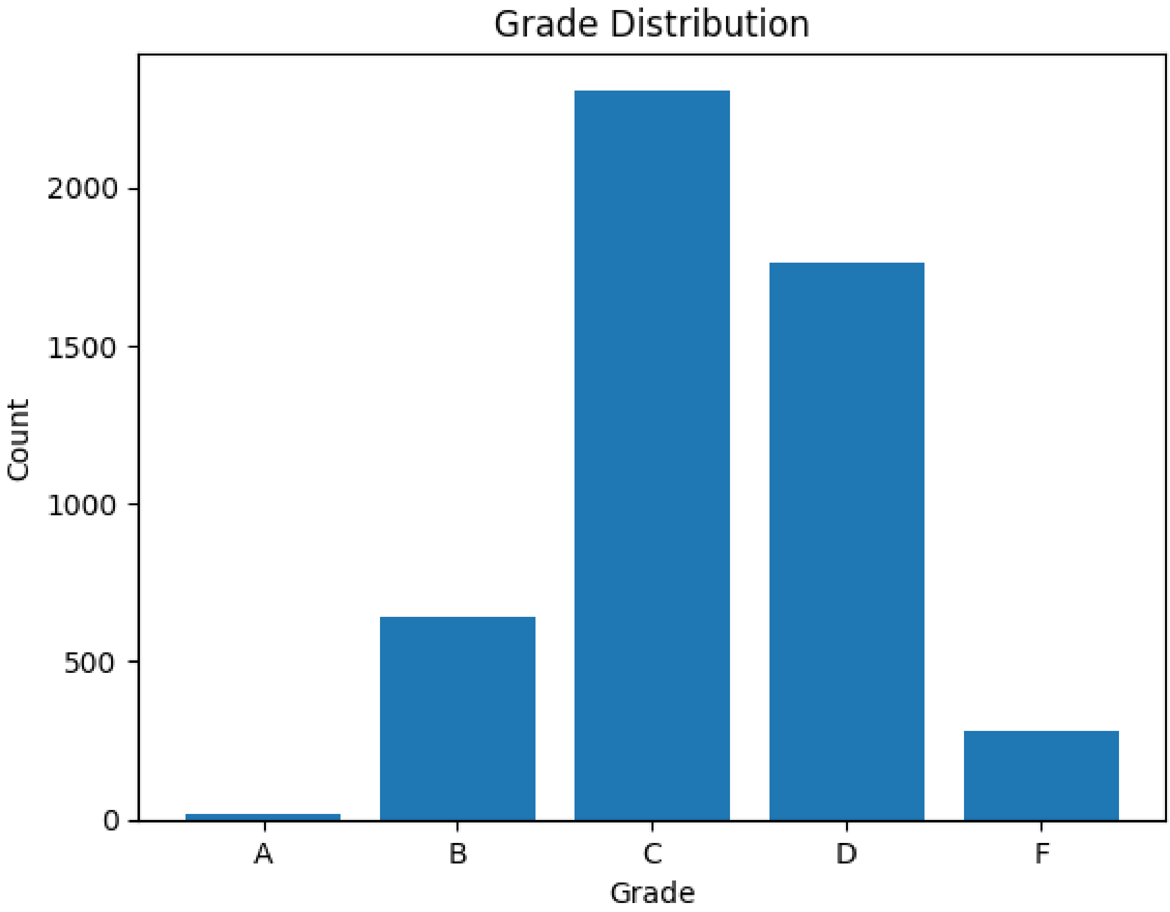


Figure 26: Distribution of grades

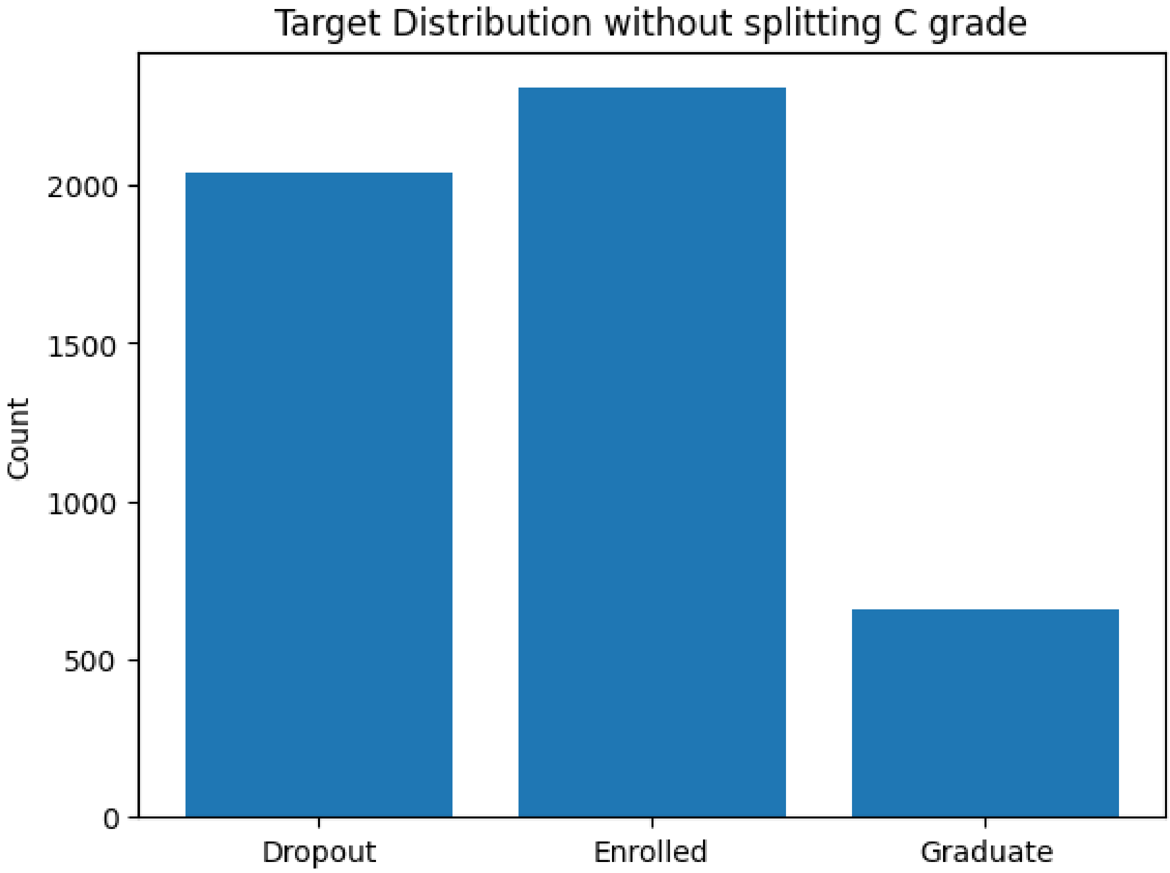


Figure 27: Distribution of target without splitting C grade

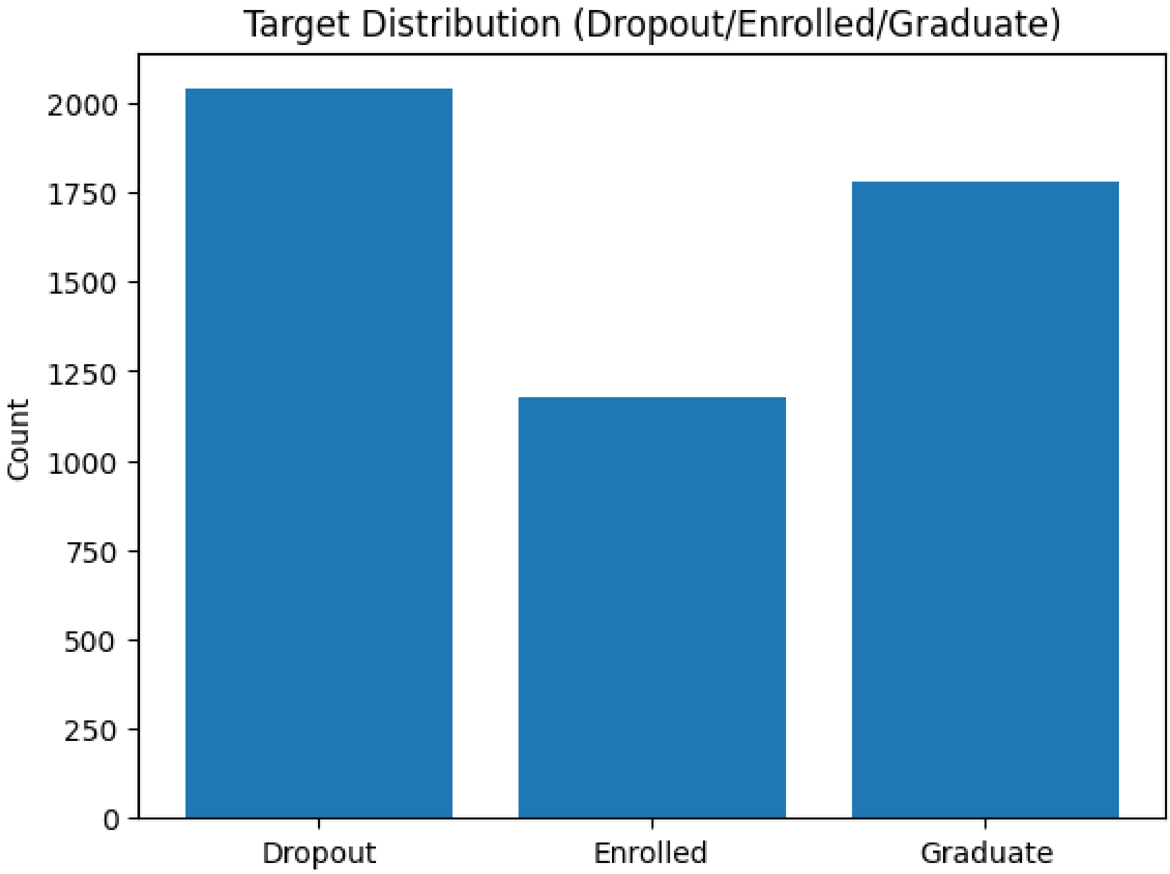


Figure 28: Distribution of target with split of C grade

To further even out the target distributions we oversampled the graduate and enrolled classes to make them level in comparison to dropout. Without the splitting of the C grade into enrolled and graduate, the classes were so uneven that oversampling would definitely cause overfitting.

With the features of the found dataset now mapped to the original dataset we pretrained a MLP neural network model on the new dataset.

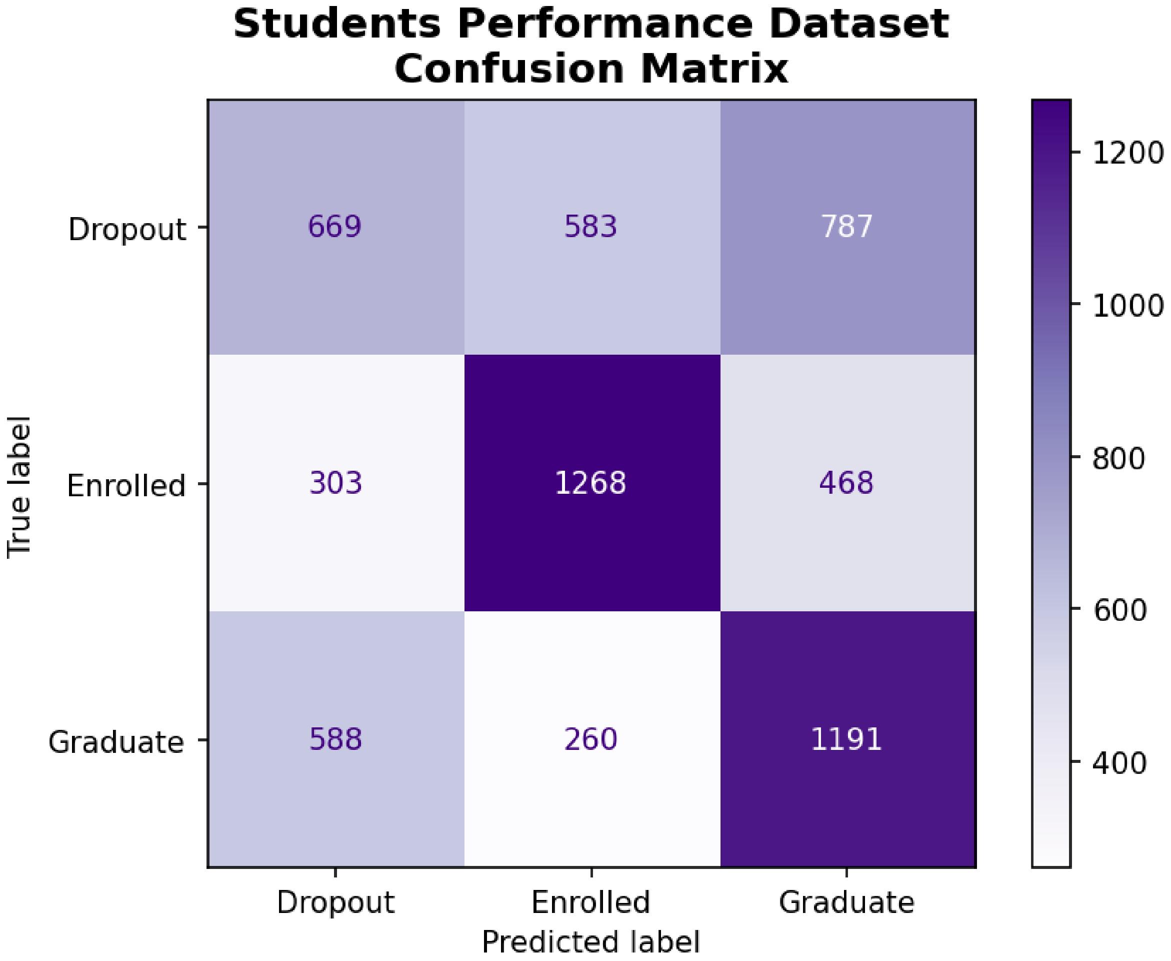


Figure 29: Confusion matrix for pre-trained model

The pre-trained model resulted in the confusion matrix seen in ***Figure 29***. This model had an accuracy of 51.1%. This is likely because we had to drop a lot of features to make the two datasets compatible.

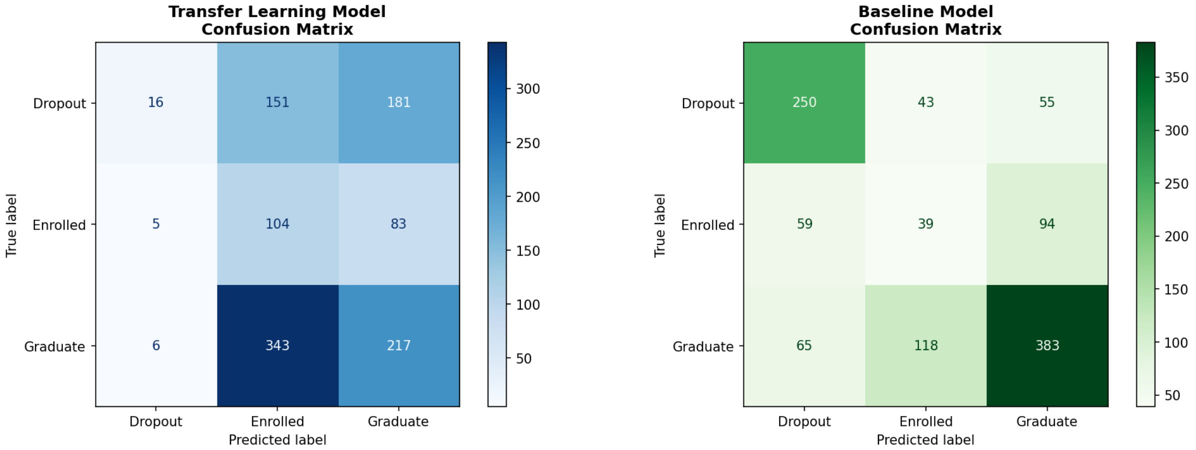


Figure 30: Confusion matrix for post trained model and base MLP model

The post-trained model achieved a low accuracy of 30.5% and was a clear step down in quality from the baseline model which had an accuracy of 60.7% when trained on the same subset of features.

9. Compare the performance of the algorithms (basic VS boosting VS bagging VS transfer) with respect to your machine learning problem and explain the results

Basic models like `SVM` and `MLP` performed well, but they are biased towards the majority **Graduate** class despite oversampling.
The best overall model is `Random Forest`, with an accuracy of 78.8%. It is robust to less relevant features which turns out to be especially important for our dataset since several features turned out to be less relevant than expected for this model. Furthermore, it also captures nonlinear relationships and interactions between factors that influence student persistence.
`Radial Basis Function Support Vector Machine` came in second with 76.6% accuracy. `RBF SVM` does not scale as well as `Random Forest` with the number of samples and features, so it does not fully explore complex interactions as efficiently as `Random Forest`.
`Multilayer perceptrons` come after with an accuracy of 75.9%. Neural networks need a lot of data to learn complex patterns, with only 4,425 rows, `MLP` cannot generalize as well as the better performing models.
The `Logistic regression` models comes in fourth place with an accuracy 75.7%. We believe that dropout risk depends not only on linear interactions, meaning linear models like `Logistic regression` cannot capture them as well. Finally, `Decision Trees` come in last place with an accuracy of 67.9%. A single `Decision Tree` performs the worst most likely because it is a high-variance model, so even small changes in the training data can lead to very different trees which makes its predictions unstable and prone to overfitting.

Bagging emerged as the strongest overall approach because the student-performance dataset contains overlapping class boundaries (Graduate and Enrolled overlap in terms of features) and features that the models consider less relevant. All of this benefits from the variance reduction that comes from using bagging. Despite this, bagging did not completely resolve the bias most models had towards graduates.
Once again `Random Forest` performed the best, because it already implements bagging in addition to bootstrap sampling by design.
`Bagging with Decision Trees` comes after, performing much better than before with an accuracy of 77.8%. It performs much better because it combines many trees, reducing overfitting and variance compared to a single tree.
`Bagging with Support Vector Machines` performed slightly better than before with 77.2% accuracy. Combining multiple `SVM` models on different bootstrap samples reduces variance and smooths out individual model errors.
`Bagging with Multilayer perceptrons` performed about the same, with an accuracy increase of about 0.03%. We get almost no improvement because neural networks like `MLP` are already high-variance and flexible models. `Bagging with MLPs` is not considered a suitable method for improving the performance of the aforementioned method. Finally, `Bagging with Logistic regression` performed the same, with an accuracy increase of only 0.02%. `Logistic Regression` is a low-variance and linear model, so averaging multiple models doesn't do much. This model performed the best when it comes to balanced accuracy, as it is better at predicting enrolled than the other models.

Boosting performed worse compared to bagging because its reweighting strategy forces models to focus on misclassified data which leads to overfitting.
`AdaBoost with Decision Trees` performed better than `AdaBoost with Logistic Regression`, with accuracies of 73.8% and 72.7% respectively.

`AdaBoost DT` outperforms `AdaBoost LR` because decision trees are flexible, high-variance base learners, while logistic regression is a low-variance and linear base model.

As mentioned earlier, transfer learning performed poorly because the new dataset was incompatible and required heavy feature dropping. We tried our best a relevant dataset and apply transfer learning on it. Despite this, we didn't find a good dataset, and the model only got an accuracy of 30.5%.

With regards to our problem to our real world problem, we believe that using the `Random Forest` model on this dataset to make predictions for student interventions, such as preventing dropout or prolonging enrollment.
However, should a university institution rather prioritize checking on students that are at risk of not graduating on time (enrolled), the `Logistic Regression with Bagging` model would be better at accurately predicting those students.