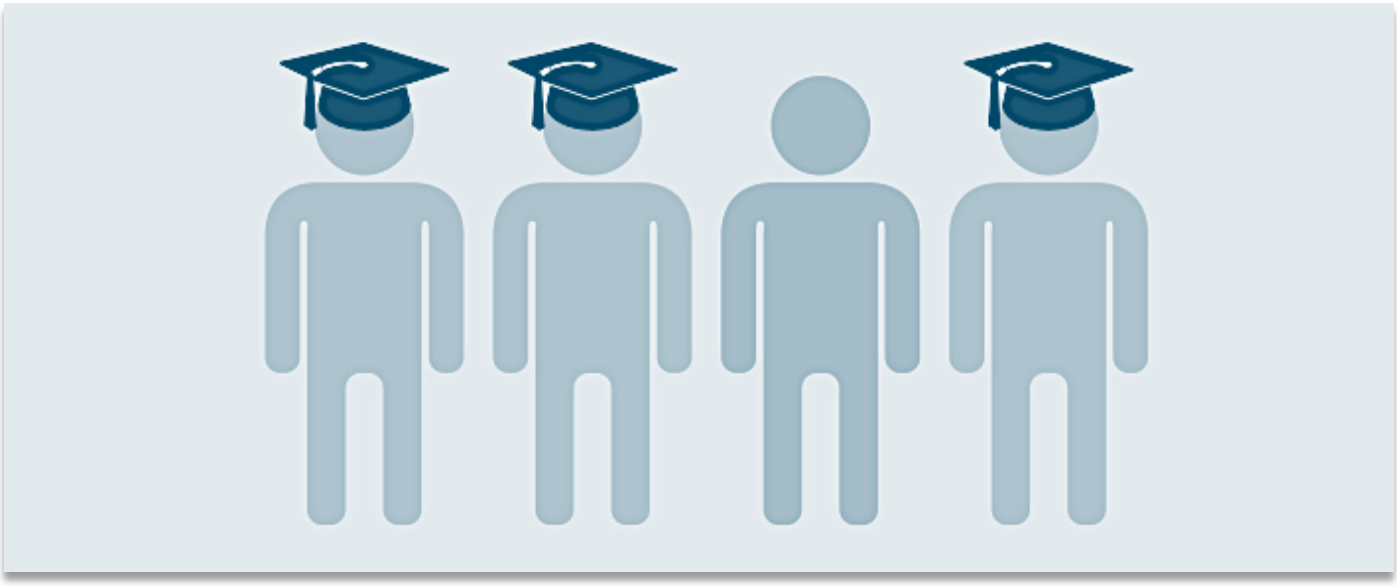
Student Drop Out Prediction

Shashank Baluni

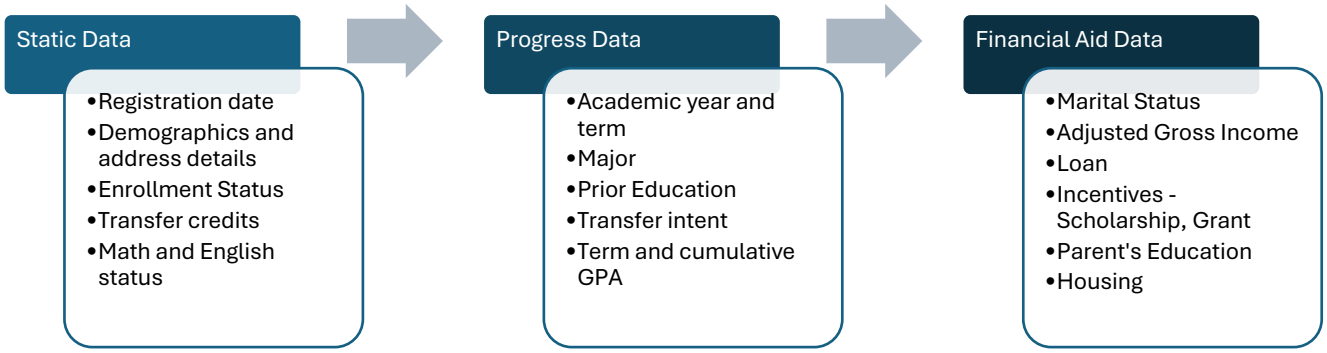


Introduction

About the project

In this project we are required to predict whether an undergraduate student will dropout from the course. We are given student static data, progress data and financial aid data to determine all the key drivers which affect the dropout decision. We will train a machine learning classification-based model on the data given to us and run the model on test data to predict student's probability to dropout from the course.

Datasets



To predict students (undergraduate) will drop out or not, we had been provided with raw data consisting of three types of data, Static Data, Progress Data and Financial Data. Static Data mainly provides Students with their IDs Cohort, Address, Gender Birthdate etc. Progress Data provides Student academic performance TermGPA and Cumulative GPA based up on their academic year. Financial Data provides Student financial loans and scholarships information on a yearly basis. The dataset used for this project has been extracted from a collection of data which had sub- groups in excel and csv formats. Within each subgroup were numerous data we needed to perform the predictions on.

Data provided:

- Student Statistic Data – consist of Fall 2011 to Spring 2016 data. Collected once a year.
- Student Progress Data – consists of Fall 2011 to Summer 2017 data. Collected every term.
- Student Financial Aid Data – consists of student's personal info (IDs, marital status, address etc.). Collected once a year.
- DropoutTrainLabels – consist of StudentID for training data with dropout labels.
- Test data – consist of StudentID for test data in the format to be submitted in Kaggle.

Static Data:

Static data that we have been provided includes student ID that is common with the progress data and with the financial data. Columns included are cohort, cohort term, campus, address, city, state, ZIP, registration, gender, Birth year and Ethnicities of the students etc. Static data include 13,261 observations of 35 variables. There was a total of 11 files of static data. We will concatenate all the files together using python's function to create one master static dataset.

Progress Data:

The second and the most important data that we have been provided to doing the analysis to accurately predict the dropout rate for the undergraduate students is the progress data. In the progress data, we have been provided with 17 different files that include different kind of data of Students. There are certain columns that are common in progress and static data like Student ID and cohort and cohort term. Progress data consists of 57,945 observations of 17 variables. The variables that are included in progress data are student ID, cohort, cohort, term, Academic year, complete, Major1 etc. We combined all the progress data together with Python "Concat" function. Since this data has multiple rows per student, we have used the most recent term information of the student to keep a unique entry for each student. We then combined the static data and progress data together by using the inner join Python merge function. Student ID is common in both data, so we can join the data by the student ID.

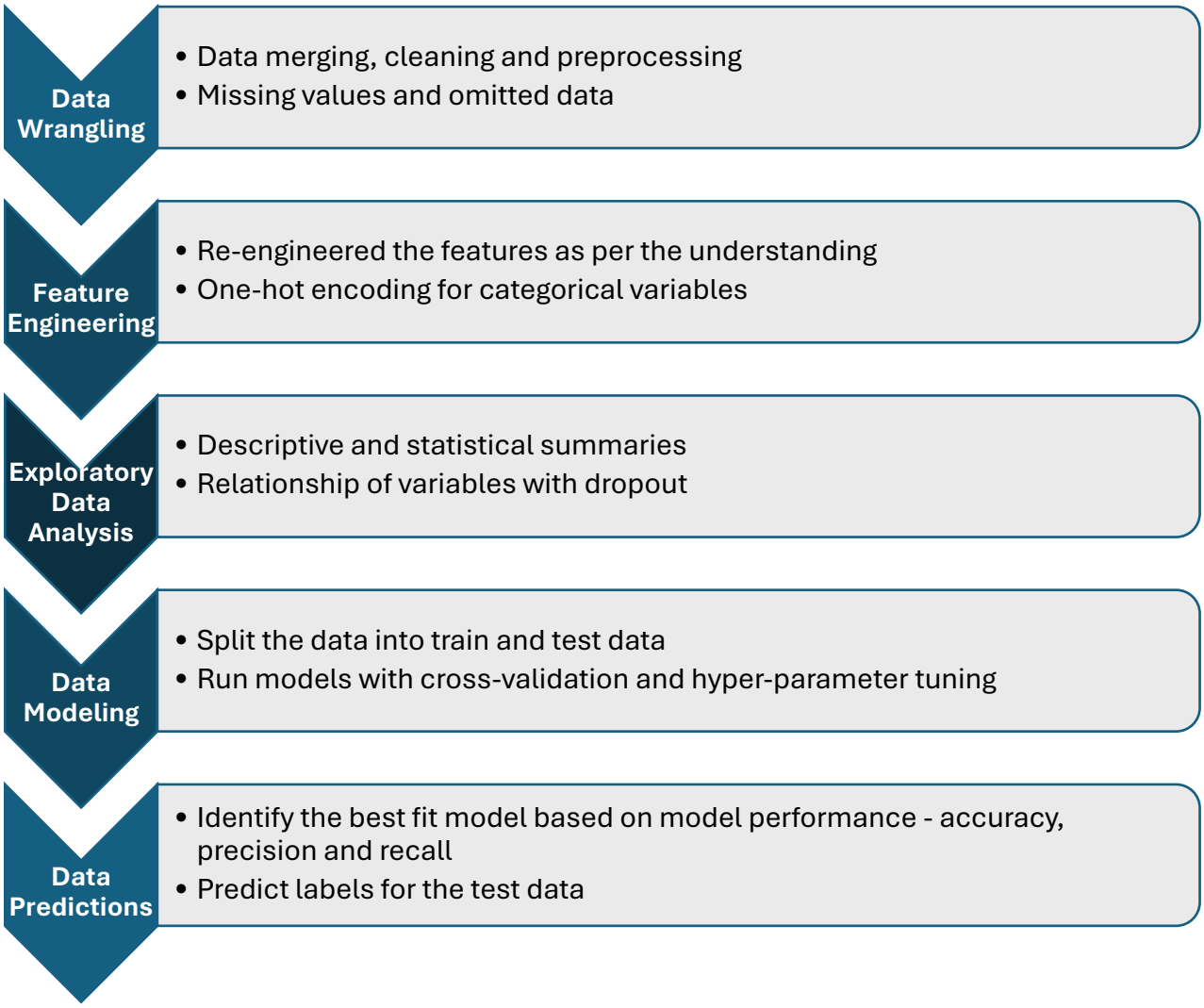
Financial Data:

Financial data has data for student's housing, loan, scholarship, grant, parent's income, and their education level. We have joined all the three datasets together by using Python merge function by taking common variable Student ID, Cohort and Cohort term. This leads us with a final-data dataset consisting of 13,261 observations of 82 variables.

Focus of the research

The focus of this research is to build a classification model on training data by identifying the key features to predict student's dropout rate and then predict on test data whether a student will dropout or not.

Overview of the steps performed



Data Wrangling and Feature Engineering

Data wrangling, also known as data munging or data preprocessing, refers to the process of cleaning, structuring, and transforming raw data into a more suitable format for analysis or modeling. It involves handling missing values, dealing with outliers, transforming variables, and restructuring data to make it more organized and ready for use in statistical analysis, machine learning, or visualization.

Key steps in data wrangling include:

Data Cleaning:

- Handling missing values: Imputing missing data or removing rows/columns with missing values.
- Dealing with duplicates: Identifying and removing duplicate records.
- Correcting errors: Fixing errors or inconsistencies in the data.

Data Transformation:

- Encoding categorical variables: Converting categorical variables into a format suitable for analysis (e.g., one-hot encoding).
- Scaling and normalization: Scaling numerical variables to a standard range or normalizing them.
- Feature engineering: Creating new features or transforming existing ones to better represent patterns in the data.

Data Aggregation:

- Grouping and summarizing: Aggregating data at different levels, computing statistics or creating new variables.

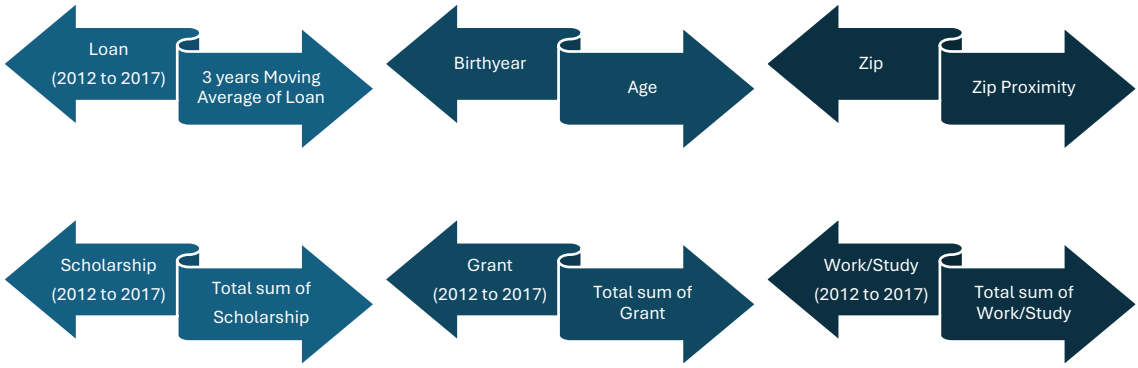
Handling Outliers:

- Identifying and addressing outliers that might affect the analysis.

Data Integration:

- Combining data from multiple sources or datasets.

We have performed these steps in our dataset as well to get the data ready for modeling phase. After merging all three datasets together, we first performed some feature engineering on our variables to transform them into more meaningful features.



Created a moving average of loan data and replaced the missing values with 0. Aggregated scholarship data from 2012 to 2017 into one and did the same for grant and work/study and replaced the missing values with 0. Replaced missing values in birthyear with mode of the birthyear from the respective admission cohort year and instead of using birthyear column directly we then created an age column. To use Zip column, we created a flag of zip proximity by looking at number of students from each zip and those zips with more than 100 students were considered as in proximity to the campus.

Next, we removed all the columns with same values for all the entries. We then checked missing value rate of each column and deleted all the columns with >70% missing rate. For the reaming columns with missing values, replaced object type columns with mode value and numerical columns with mean values and dropped some irrelevant variables like city, state, address, etc. We are then left with a clean data with no missing values; however, we still have categorical variables in our master data which need to be treated before feeding them to machine learning algorithms. To do so we will perform one-hot encoding on the categorical variables. This converted our dataset into a data with 13,261 rows and 67 columns.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyzing and summarizing the main characteristics of a dataset to gain insights and discover patterns. The primary goal of EDA is to understand the underlying structure, relationships, and distributions within the data before formal modeling or hypothesis testing. It involves visualizing and summarizing the main features of the data to uncover patterns, trends, and anomalies.

Key components of exploratory data analysis include:

Descriptive Statistics and Value Counts:

- Computing basic statistics such as mean, median, mode, standard deviation, and percentiles to describe the central tendency and spread of the data.
- Check value counts for categorical variables to understand the data balance.

Data Visualization:

- Creating graphical representations of the data using plots and charts.
- Examples of visualizations include histograms, box plots, scatter plots, bar charts, and heatmaps.

Correlation Analysis:

- Exploring relationships between variables through correlation coefficients or scatter plots.
- Understanding how changes in one variable relate to changes in another.

Outlier Detection:

- Identifying and investigating data points that deviate significantly from the overall pattern.

Exploratory Data Analysis is often the initial step in the data analysis process and is crucial for informing subsequent modeling decisions. It helps data scientists or analysts form hypotheses, validate assumptions, and decide on appropriate modeling techniques.

Summary Statistics

	StudentID	CohortTerm	Term	CompleteDevMath	CompleteDevEnglish	Major1	Major2	Complete1	CompleteCIP1
count	13261.000000	13261.000000	13261.000000	13261.000000	13261.000000	13261.000000	13261.000000	13261.000000	13261.000000
mean	316150.611266	1.391222	2.990046	-1.271548	-1.436543	38.187844	0.041927	2.160244	10.964935
std	45170.648889	0.793371	1.560801	1.001927	0.949234	16.716307	5.853482	3.460927	21.500747
min	20932.000000	1.000000	1.000000	-2.000000	-2.000000	-1.000000	-1.000000	0.000000	-2.000000
25%	305254.000000	1.000000	3.000000	-2.000000	-2.000000	26.000000	-1.000000	0.000000	-2.000000
50%	321478.000000	1.000000	3.000000	-2.000000	-2.000000	43.000000	-1.000000	0.000000	-2.000000
75%	343511.000000	1.000000	3.000000	0.000000	-1.000000	51.000000	-1.000000	7.000000	26.000000
max	359783.000000	3.000000	6.000000	1.000000	1.000000	54.000000	54.000000	8.000000	54.000000

(only for the first few numerical variables)

Value Counts

```
Student count by Marital Status:

Single       11850
Married        992
Divorced       226
Separated      193
```

Data visualization

Created a count plot to show the distribution of students based on their housing situation and cumulative GPA. This shows that students who live on campus have comparatively low cumulative GPA.

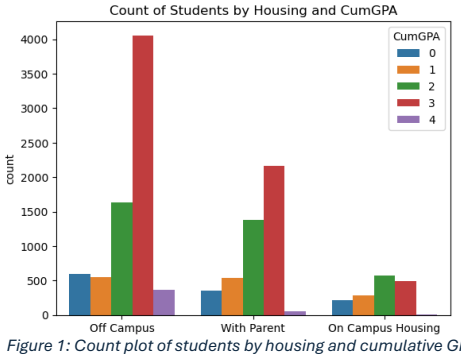


Figure 1: Count plot of students by housing and cumulative GPA

The histogram of student age shows that most of the students are in 20-25 years of age-groups with some outliers from age buckets outside this group.

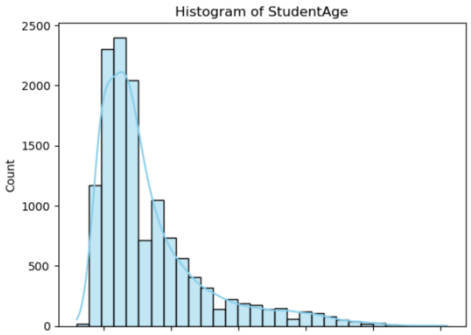
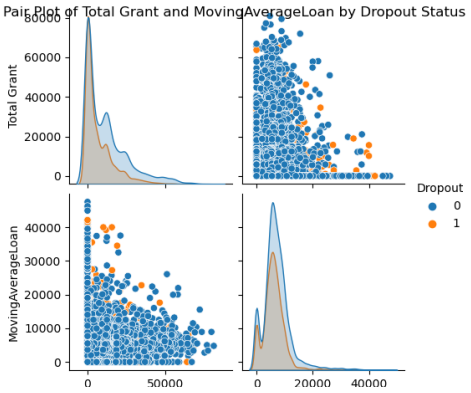


Figure 2: Histogram of student age

We also plotted a scatter plot between parent's adjusted income and loan's moving average, however, there is no correlation between the two variables.

Next, we separated our data into test and training data using the test IDs provided as part of this project. On our training data we then performed some further exploratory analysis by studying the relationships of variables with dropout. We plotted total grant and moving average of loan against dropout status. No big differences but students who didn't dropout had a comparatively high average grant than those who didn't.



Next, we plotted student age and major1 by dropout status. We saw that more students dropped out when in lower age groups and in lower major groups.

We looked at the count plot of students by cumulative GPA and dropout status which helped us understand that less students dropped out who had higher cumulative GPAs.

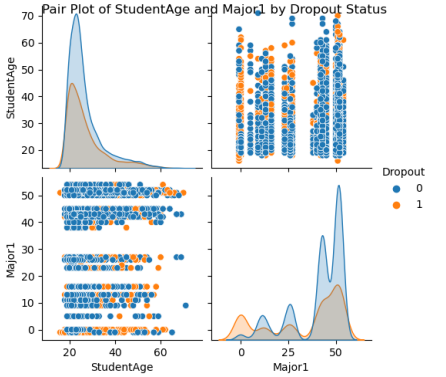


Figure 4: Pair plot of student age and major1 by dropout status

All these exploratory analyses helped us validate some of our initial hypothesis and assumptions about this data and we are now ready to perform classification modeling on our training data.

Data Modeling and Data Predictions

The goal of classification modeling is to learn the underlying patterns and relationship between the input variables and the output classes and use this knowledge to predict the class of new unseen data. The model can be trained using a variety of algorithms. We have looked at Random Forest, Gradient Boosting, Logistic Regression and KNN as part of this analysis.

From the modeling steps, we choose several different models to interpret the data whether an undergraduate student will drop out or not. Several models were fitted and tested in the analysis. Our analysis says that for a problem like this ensemble models performed better than Logistic and KNN.

1. Random Forest
2. Gradient Boosting
3. Logistic Regression
4. K-Nearest Neighbor (KNN)

Model's accuracy, precision, and recall, all of these together decide whether the model is performing well on the data on not. Normally in classification problem, the common metrics for evaluating classification Models include accuracy, precision, recall, F1 score, and receiver operating characteristics curve. The ability to accurately classify new data can provide valuable insights.

Accuracy: (TP + TN) / (TP + TN + FP + FN)

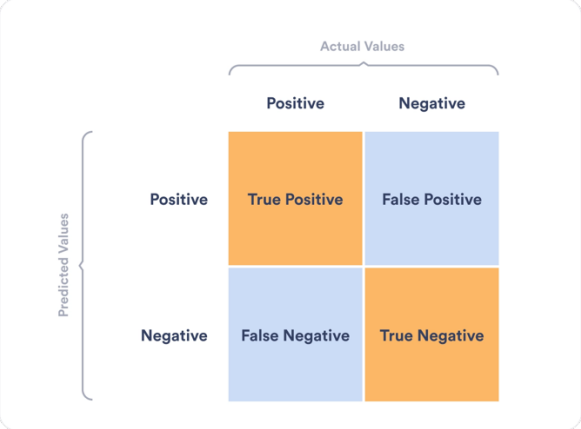
Precision: TP / (TP + FP)

Recall: TP / (TP + FN)

F1-Score: 2 * (Precision * Recall) / (Precision + Recall)

Specificity: TN / (TN + FP)

To improve our model performance, we used cross validation and hyper-parameter tuning using grid-search.



Benefits of cross-validation:

Cross-validation is used to assess how well a predictive model will generalize to an independent dataset. It helps in estimating the performance of a model on an unseen dataset.

- Reduces the risk of overfitting by assessing model performance on different subsets of the data.
- Provides a more reliable estimate of the model's performance compared to a single train-test split.
- Utilizes the entire dataset for training and testing, maximizing data usage.

Benefits of hyper-parameter tuning:

Hyperparameter tuning involves finding the optimal set of hyperparameters for a machine learning model. Hyperparameters are parameters that are not learned during training and must be set before training.

- Improves model performance by selecting hyperparameters that yield the best results.
- Helps in avoiding overfitting or underfitting by finding the right balance in model complexity.
- Enhances the model's ability to generalize to new, unseen data.

After performing optimization techniques, we got the best results with random forest classifier.

Random forest also provided feature importance which helped us in narrow down our selection of best features from 66 to 16 (with more than 1% feature importance). We also looked at correlation matrix of the selected variables to check their dependencies on each other and on the dropout variable. “Complete1” variable was highly correlated with “CompleteCIP1”, so we removed “Complete1” from the final model. Here is a quick model comparison:

Random Forest Classifier with cross-validation and hyper-parameter tuning

Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. We split our training data into train and test sets with 80-20 ratio and then fed them to random forest classifier.

Best parameters from grid search:

```
# best model results based on grid search
best_model = RandomForestClassifier(n_estimators=150,
                                    max_depth=20,
                                    min_samples_split=7,
                                    min_samples_leaf=2,
                                    random_state=42)
```

Model performance:

```
Test accuracy score: 0.9507
              precision    recall  f1-score   support

           0      0.942     0.980     0.961      1519
           1      0.966     0.903     0.933       934

    accuracy                          0.951      2453
   macro avg      0.954     0.941     0.947      2453
weighted avg      0.951     0.951     0.950      2453
```

We then evaluated the performance of this model on our test data and received ~93% score in Kaggle competition. This model performed the best for us.

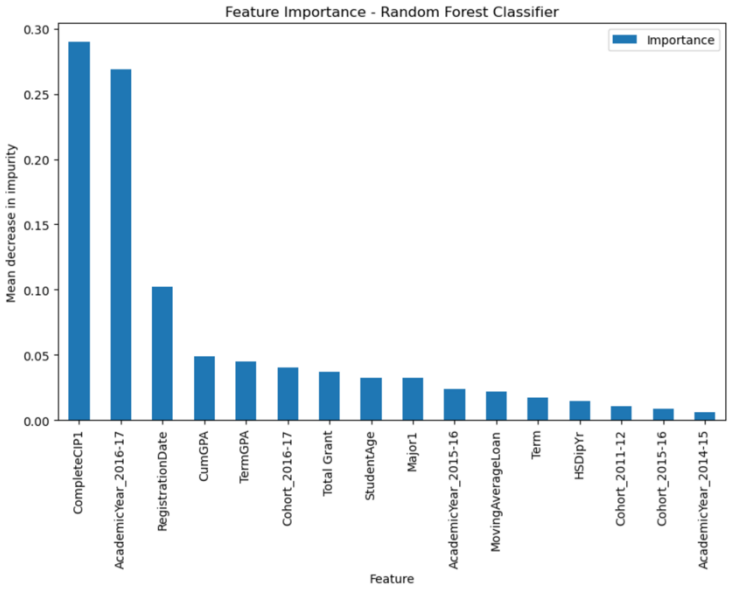


Figure 5: Random forest classifier feature importance (after removing features with importance <=1%)

Here is the correlation matrix of the final variables used in this model:

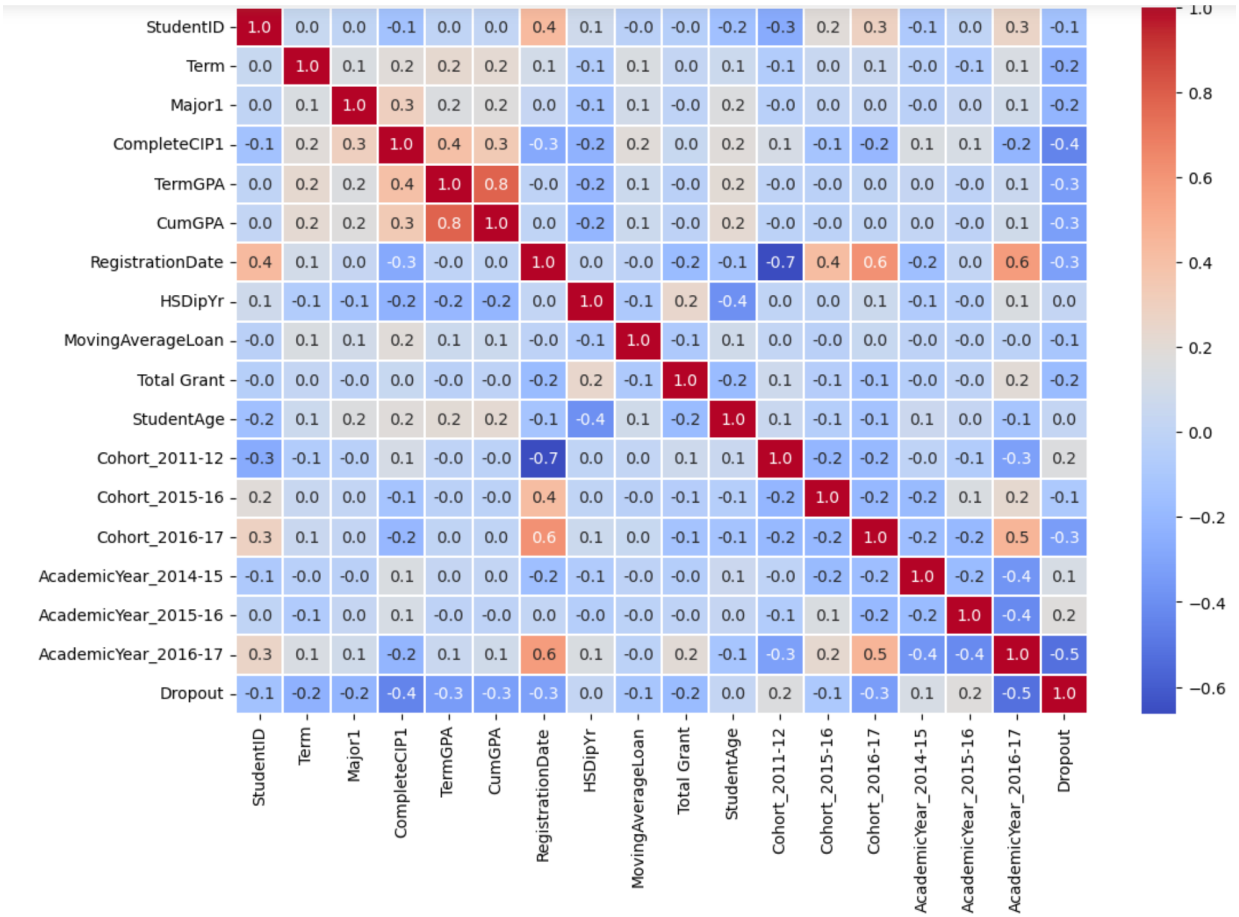


Figure 6: Correlation matrix of variables used in random forest classifier model

Gradient Boosting Classifier with cross-validation and hyper-parameter tuning

Gradient Boosting is an ensemble learning technique that builds a series of weak learners (typically decision trees) sequentially. Each tree corrects the errors of the previous one, and the final model is an aggregation of these weak learners.

Best parameters from grid search:

```
# Train the model with the best parameters
best_gb_model = GradientBoostingClassifier(learning_rate=0.1,
                                           max_depth=5,
                                           n_estimators=200,
                                           random_state=42)
```

Model performance:

```
Test accuracy score: 0.9503
              precision    recall  f1-score   support

           0      0.947     0.974     0.960      1519
           1      0.956     0.911     0.933       934

    accuracy                          0.950      2453
   macro avg      0.952     0.943     0.947      2453
weighted avg      0.950     0.950     0.950      2453
```

We then evaluated the performance of this model on our test data and received ~92% score in Kaggle competition. This model performed slightly less good than random forest classifier.

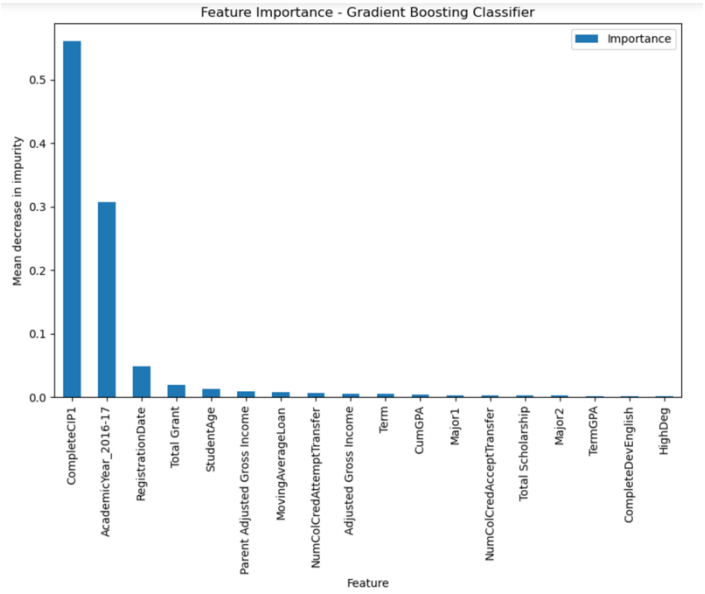


Figure 7: Gradient Boosting classifier feature importance (after removing features with importance <=0.1%)

Here is the correlation matrix of the final variables used in this model:

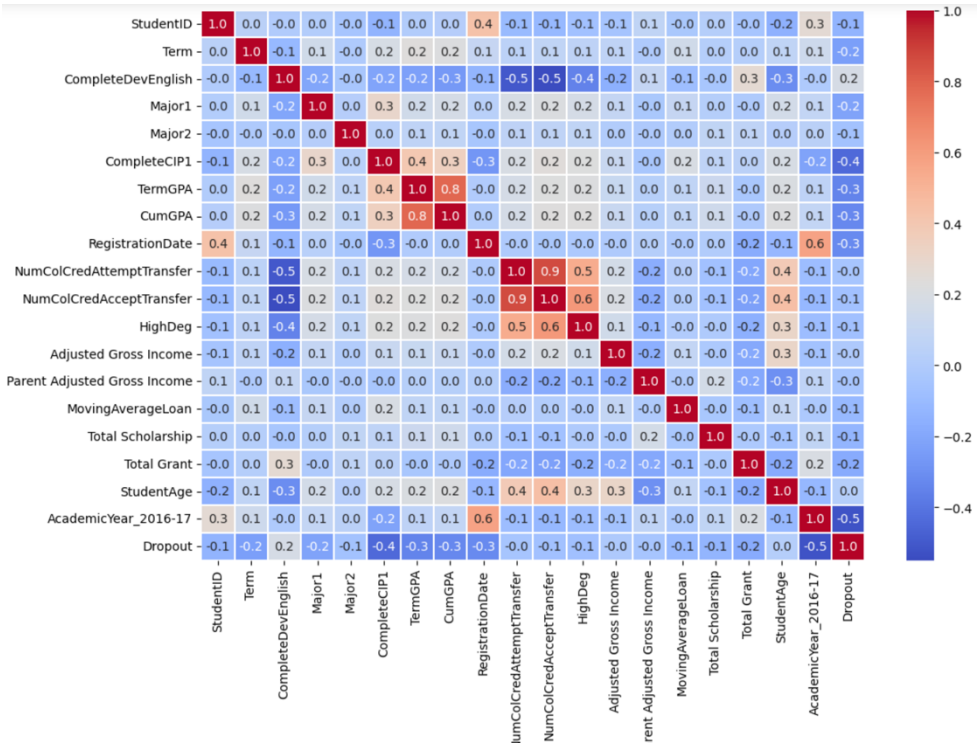


Figure 8: Correlation matrix of variables used in gradient boosting classifier model

Logistic Regression with cross-validation and ROC

Logistic Regression is a statistical model used for binary classification tasks. Despite its name, it is a classification algorithm, not a regression algorithm. It models the probability that an instance belongs to a particular category.

Model performance:

```
Test accuracy score: 0.6592
              precision    recall  f1-score   support

           0      0.664     0.912     0.768      1519
           1      0.634     0.248     0.357       934

    accuracy                          0.659      2453
   macro avg      0.649     0.580     0.563      2453
weighted avg      0.652     0.659     0.612      2453
```

```
Feature Coefficients:
                  Feature   Coefficient
9             StudentAge  1.527314e-04
10        Cohort_2011-12  5.649887e-05
14  AcademicYear_2015-16  5.208309e-05
6                 HSDipYr  4.604563e-05
13  AcademicYear_2014-15  3.466028e-05
5       RegistrationDate  1.847592e-08
11        Cohort_2015-16 -3.224132e-05
8             Total Grant -3.651557e-05
7       MovingAverageLoan -5.629909e-05
12        Cohort_2016-17 -1.036655e-04
15  AcademicYear_2016-17 -1.933536e-04
4                  CumGPA -2.435408e-04
0                    Term -2.569662e-04
3                 TermGPA -3.436920e-04
1                   Major1 -2.483935e-03
2            CompleteCIP1 -6.728765e-03
```

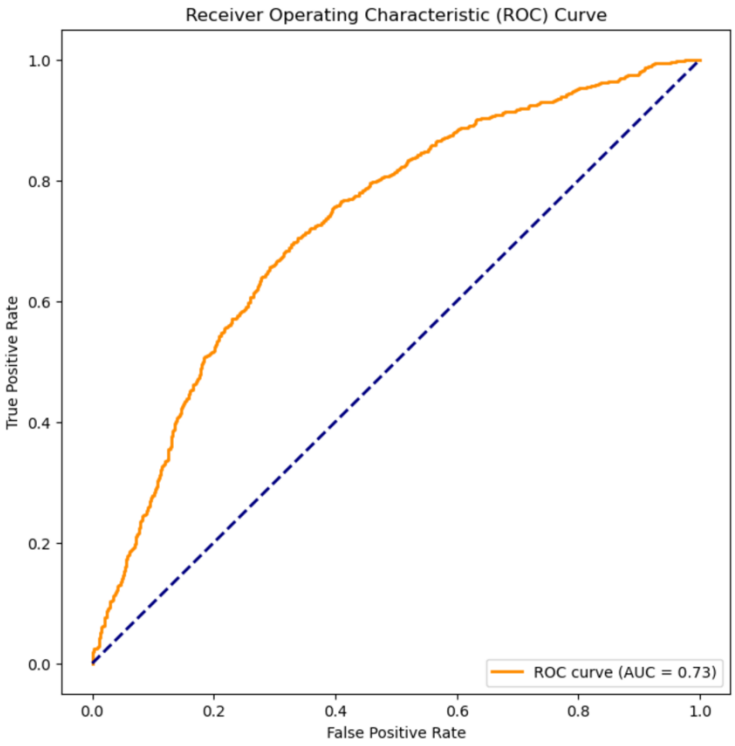


Figure 9: ROC for Logistic regression model

The model performance metrics show that the model didn't perform well. We evaluated the performance of this model on our test data and received low score in Kaggle competition.

KNN with best k-value

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression tasks. It makes predictions based on the majority class of its k nearest neighbors in the feature space.

Best value of k = 9

Model performance:

```
Test accuracy score: 0.7106
              precision    recall  f1-score   support

           0      0.755     0.789     0.771      1519
           1      0.629     0.584     0.606       934

    accuracy                          0.711      2453
   macro avg      0.692     0.686     0.688      2453
weighted avg      0.707     0.711     0.708      2453
```

KNN performed better than Logistic regression but not as good as ensemble models like Random Forest and Gradient boosting.

Hence, the best Kaggle Competition Score was achieved using Random Forest model with grid search for hyperparameter tuning and by only retaining those features with importance >1%.

Conclusion

In conclusion, our team embarked on a predictive analytics project focused on forecasting undergraduate student dropout rates. Leveraging a comprehensive dataset encompassing static information, academic progress, and financial details, we aimed to construct a robust classification model to predict the likelihood of student dropout.

Our journey commenced with meticulous data wrangling and feature engineering. We amalgamated various datasets, addressed missing values, and engineered new features, including the creation of a moving average for loan data, aggregation of financial aid information, and flagging of zip code proximity. Data cleaning and preprocessing were pivotal to preparing a clean and structured dataset for modeling.

Subsequently, we delved into exploratory data analysis (EDA) to gain insights into the relationships between variables. Visualization and statistical analysis helped us understand patterns, trends, and potential drivers of student dropout. EDA provided valuable context for the subsequent modeling phase.

For the modeling stage, we explored multiple classification algorithms, including Random Forest, Gradient Boosting, Logistic Regression, and K-Nearest Neighbors (KNN). Through cross-validation and hyperparameter tuning, we aimed to optimize model performance.

Random Forest emerged as the most effective model, achieving the highest accuracy and predictive power. The ensemble nature of Random Forest, along with its ability to handle non-linear relationships, made it the preferred choice. Feature importance analysis provided further insights into the key drivers of dropout.

Despite the strong performance of Random Forest, we also explored other models, such as Gradient Boosting, Logistic Regression, and KNN, to provide a comprehensive analysis of available options. Each model underwent careful evaluation, considering metrics like accuracy, precision, recall, and the area under the ROC curve.

In conclusion, our predictive modeling efforts culminated in a Random Forest classifier that demonstrated excellent performance, yielding a high Kaggle competition score. This model, trained on a diverse dataset and optimized through rigorous testing, showcases the potential for machine learning to contribute meaningfully to educational analytics. The insights gained from this project can aid institutions in identifying at-risk students early and implementing targeted interventions to support student success.