Student Performance Prediction

Dataset: Student performance (in Portuguese schools)

Amulya Kallakuri (kallakur@usc.edu)
Nithyashree Manohar (nithyash@usc.edu)

05/02/2022

1. Abstract

In this project, we explored the student performance in two Portuguese schools. The dataset contains several attributes which include student grades, demographic, social and school related features. We used several machine learning models to predict the score of a student to intervene and help at-risk students do better. We predicted the first-period academic performance without any prior academic performance and predicted the final-period academic performance with and without prior academic performance. For the multiclass classification problem, the trivial system randomly outputs class labels with probability based on class priors. The baseline system used was the nearest means classifier. We also used K Nearest Neighbors, Support Vector Machine Classifier and an Artificial Neural Network for all the missions. It was found that the Support Vector Classifier was the best performing model for mission one with a test accuracy of 40.49%, Artificial Neural Network was the best performing model for mission two with a test accuracy of 42.33%. Support Vector Classifier was the best performing model for mission three with a test accuracy of 76.07%. For the regression problem, the trivial system outputs the mean of the training output data for all points. The baseline system used was the 1 nearest neighbor regressor and the linear regressor. Additionally, we used K Nearest Neighbors, Support Vector Regressor, Artificial Neural Networks, Decision Tree Regressor, Random Forest Regressor and XG Boost Regressor for all the missions. The ANN and the XGBoost are shown to be the best performing models with r2_score 0.91 and 0.929, followed by the Random Forest and the Decision Tree.

2. Introduction

2.1. Problem Statement and Goals

The student performance dataset consists of student data from two schools in Portuguese. The goal for both problems is to predict the first-period academic performance without prior academic performance and final-period academic performance with and without prior academic performance. The number of input attributes per datapoint is 30 excluding the first-period, second-period and final-period academic score. For the classification problem, the output target score is converted into a 5-class categorical value based on certain rules. However, if the first-period and second-period scores are used as features, then they are left in numeric form ranging from 0 to 20, which we used for regression. 13 of the attributes are integer valued, 13 are binary-valued categorical, and 4 are multi-valued categorical.

3. Approach and Implementation

3.1. Dataset Usage

For Classification:

For the trivial system, the training dataset was used to calculate the prior probabilities of each class and this probability was used to output a random class for the data points in the test dataset. The baseline system used is a Nearest Means Classifier, in this system the training dataset is used to find the centroids of each class. Once the centroids are determined, we use the test dataset to predict the class the datapoint belongs to. For the baseline and the trivial system, we do not use the 4 categorical on-binary features. For the classification models, K Nearest Neighbors and Support Vector Machine Classifier, we use the train dataset to perform a grid search to find the best parameters. We then use the test dataset to make predictions using the best parameters. For the Artificial Neural Network, the target variable is one hot encoded and we pass the train dataset through two hidden layers and one output layer. The test dataset is passed to the trained model to make predictions. The dimensions of the train and test datasets of each mission are as follows

Mission 1	Trivial	Baseline	KNN	SVM	ANN
X_train	486 × 26	486 × 26	486 × 39	486 × 39	486 × 39
y_train	486 × 1	486 × 1	486 × 1	486 × 1	486 × 1
X_test	163 × 26	163 × 26	163 x 39	163 x 39	163 x 39
y_test	163 × 1	163 × 1	163 x 1	163 x 1	163 x 1

Mission 2	Trivial	Baseline	KNN	SVM	ANN
X_train	486 × 26	486 × 26	486 × 39	486 × 39	486 × 39
y_train	486 × 1	486 × 1	486 × 1	486 × 1	486 × 1
X_test	163 × 26	163 × 26	163 x 39	163 x 39	163 x 39
y_test	163 × 1	163 × 1	163 x 1	163 x 1	163 x 1

Mission 3	Trivial	Baseline	KNN	SVM	ANN
X_train	486 × 28	486 × 28	486 × 44	486 × 44	486 × 44
y_train	486 × 1	486 × 1	486 × 1	486 × 1	486 × 1
X_test	163 × 28	163 × 28	163 x 44	163 x 44	163 x 44
y_test	163 × 1	163 × 1	163 x 1	163 x 1	163 x 1

For Regression:

For the trivial system, we calculated the mean of the training output data and chose that as our test output, essentially meaning a null model since there is no learning happening with our train data. For the first baseline system, we used the 1NN model which chooses the nearest neighbor with the value of k as 1 in the kNN model. For the second baseline system, we used Linear Regression as our model. For the baseline and the trivial systems, we do not use the 4 categorical non-binary features. For the regression models kNN, Support Vector Regression, ANN, Decision Tree

Regression, Random Forest Regression and XG Boost Regression, we use the train dataset to perform a grid search to find the best parameters. We then use the test dataset to make predictions using the best parameters.

Mission 1	**Trivial**	**Baseline**	**All other models (Before feature selection)**	**All other models (After feature selection)**
X_train	486 × 26	486 × 26	486 × 27	486 x 14
y_train	486 × 1	486 × 1	486 × 1	486 x 1
X_test	163 × 26	163 × 26	163 x 27	163 x 14
y_test	163 × 1	163 × 1	163 x 1	163 x 1

Mission 2	**Trivial**	**Baseline**	**All other models (Before feature selection)**	**All other models (After feature selection)**
X_train	486 × 26	486 × 26	486 × 27	486 x 14
y_train	486 × 1	486 × 1	486 × 1	486 x 1
X_test	163 × 26	163 × 26	163 x 27	163 x 14
y_test	163 × 1	163 × 1	163 x 1	163 x 1

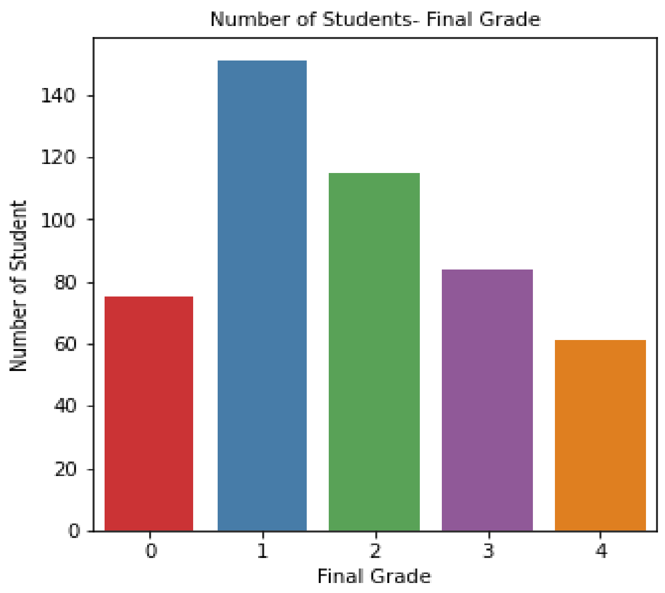
Mission 3	**Trivial**	**Baseline**	**All other models (Before feature selection)**	**All other models (After feature selection)**
X_train	486 × 26	486 × 26	486 × 27	486 x 14

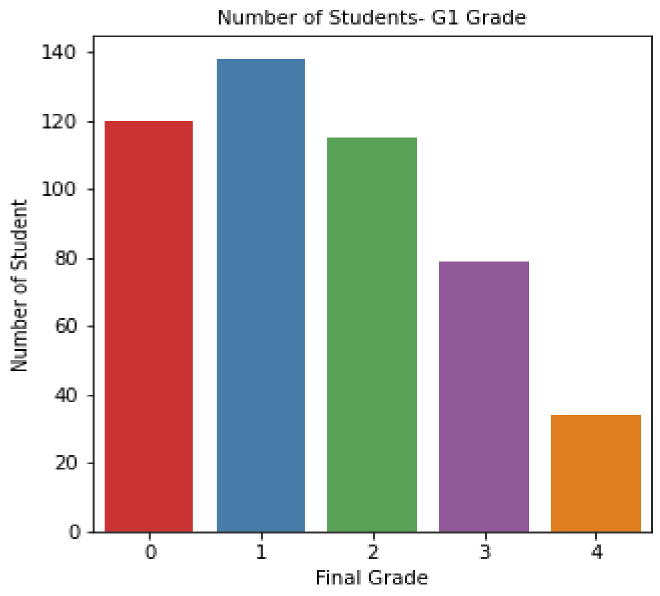
y_train	486 × 1	486 × 1	486 × 1	486 x 1
X_test	163 × 26	163 × 26	163 x 27	163 x 14
y_test	163 × 1	163 × 1	163 x 1	163 x 1

3.2. Preprocessing

In terms of preprocessing, for both classification and regression, the 13 categorical attributes were converted to numeric values using a label encoder from the sklearn preprocessing package. The 4 multivalued categorical attributes were mapped to numeric values and dummies were generated.

For classification in all the three missions, there was a class imbalance. We used the SMOTE oversampling technique from the scikit-learn module to balance the classes. However, this method did not produce good results. Hence, we proceeded without oversampling. We performed min max scaling and standard scaling using the scikit-learn preprocessing package and arrived at the conclusion that this technique did not produce desired results. The class imbalance can be seen from the images below. The number of students who scored an excellent score are much lower than the number of students who scored a sufficient score.





For regression in all three missions, we z-scaled the training and testing input data to see if that made a difference in the accuracy since the numbers varied by a lot (for instance, in the case of age, absences etc.)

3.3. Feature engineering

Classification:

It was found that the mother's education was an important feature in predicting academic performance. A new column consisting binary values where 1 denotes that the student's mother has completed secondary or higher education and 0 otherwise was added. The same technique was used for father's education. Using these two new columns, another column with binary values where 1 denotes that both the parents have completed secondary or higher education and 0 otherwise was added. For the third mission, a new column, same_grade, which has binary values was added. In this column, 1 denotes that the student scored more than 15 in period one and period two, 0 otherwise. Another column, average, which is the mean of the first-period score and the second-period score was added. A new column, failure was added which has binary values. 1 denotes 3 or more failures and 0 denotes no failures. It was found that the absences attribute ranges from 0 to 32, this was mapped to categorical numeric values. We used the Chi2 and Anova test to find the importance of each feature. The chi2 test determines if the feature is correlated to the target variable or not using the p value. Generally, the threshold for the p value is 0.05. However, for our problem we dropped six of the least correlated attributes. The attributes dropped for each mission are shown below.

Mission one

Attribute	**p value**
goout	8.435803e-01
famsize	8.879097e-01
guardian	9.237833e-01
romantic	8.766715e-01
Pstatus	9.951690e-01
age	9.991033e-01

Mission two

Attribute	**p value**
famsup	8.057053e-01
nursery	8.422840e-01
famrel	8.530165e-01
guardian	9.466250e-01
age_	9.899682e-01
Pstatus	9.979086e-01

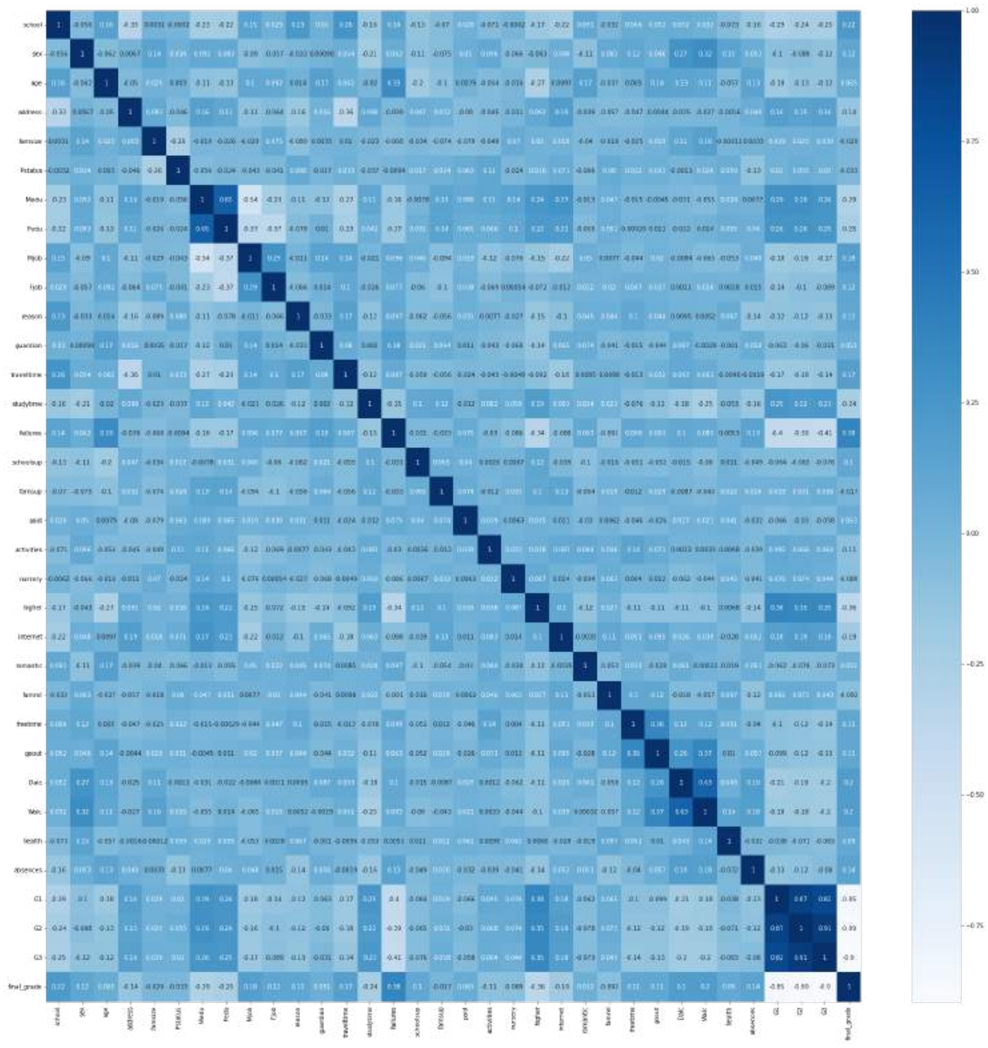
Mission three

Attribute	**p value**
famsup	8.057053e-01
nursery	8.422840e-01
famrel	8.530165e-01
guardian	9.466250e-01
age_	9.899682e-01
Pstatus	9.979086e-01

Regression:

For the regression models, the features were selected based on the following methods:

1. Analysis of the correlation matrix between all the features versus the grade to be predicted.
2. SelectKBest features
3. Random Forest model feature importances attribute



Correlation matrix of the training data
(Image is not legible in this document since we had to follow format according to protocol. For clarity, the reader may zoom in the image or increase size)

The final features were selected based on which features were determined to be most important given at least two out of three methods and those features were selected to run our models. Using domain knowledge, additional features such as health were included and famrel was excluded.

In the following table, the cells highlighted green depict features that are included in our model for training and testing.

Correlation Matrix	SelectKBest	Random Forest Feature Importances
Higher	Higher	Higher
Medu	Medu	Medu
Fedu	Fedu	Fedu
Studytime	Studytime	Studytime
Absences	Absences	Absences
Failures	Failures	Failures
Walc	Walc	Walc
Dalc	Dalc	Dalc
Traveltime	Traveltime	Traveltime
Address	Address	-
-	School	School
-	Internet	Internet
-	Freetime	Freetime
-	Goout	Goout
Famrel	-	Famrel
Reason	-	-
-	Sex	-
-	-	Health

3.4. Feature dimensionality adjustment

Principal component analysis (PCA) is a technique for reducing the dimensionality of datasets, increasing interpretability but at the same time minimizing information loss [1]. We used Principal Component Analysis for feature dimensionality adjustment. However, this is not improving the accuracy of the

models.
For regression, we tried selecting the features using the feature importance attribute of the random forest regressor with gini importance greater than 0.2 and ran the models using just those features. We also followed the correlation and the SelectKBest features in order to remove the features that all three methods found were important and ran the model on just these features.

3.5. Training, classification, regression, and model selection

- **For the classification problem**, the trivial system randomly outputs class labels based on the class priors. The class prior was calculated by dividing the number of occurrences of each class in the train dataset by the length of the train dataset. The probabilities are passed as weights to a random class generator. We ran this system 10 times and took the average of the results to be the output class.
- The baseline system used is the nearest means. This was implemented using the Nearest Centroids module from scikit-learn library. This system calculates the centroid for each target class in the training phase. In the testing phase, the system calculates the distance between the centroids and the new point. The minimum distance is picked and this point is assigned to the class with the least distance to the centroid.
- The first classification algorithm used is the K Nearest Neighbors. The algorithm was implemented using the K Neighbors Classifier from the scikit-learn library. This system calculates the k nearest neighbors for a given point using the given distance metric. The number of occurrences of each class is calculated and the given point is assigned to the class with the maximum occurrence. The parameters passed to the model are number of neighbors, leaf size, p value, algorithm, weights and metric. The best parameters were chosen using a grid GridSearchCV which is a library function that is a member of sklearn's model_selection package. We used this

function to loop through predefined hyperparameters and fit the model on the training set. Using this we found the best parameters to pass to the model. The different values used for each parameter can be seen in the table below

Neighbors	1 - 10
Leaf size	1, 2, 4, 8, 16, 32, 64, 128, 256, 512
P value	1, 2
Algorithm	auto, kd_tree, brute, ball_tree
Weights	uniform, distance
Metric	minkowski, euclidean, manhattan, chebyshev

The following are the best parameters produced by grid search for different missions
Mission one - {'algorithm': 'kd_tree', 'leaf_size': 128, 'metric': 'minkowski', 'n_neighbors': 7, 'p': 2, 'weights': 'uniform'}
Mission two - {'algorithm': 'ball_tree', 'leaf_size': 1, 'metric': 'minkowski', 'n_neighbors': 7, 'p': 1, 'weights': 'uniform'}
Mission three - {'algorithm': 'ball_tree', 'leaf_size': 32, 'metric': 'minkowski', 'n_neighbors': 4, 'p': 1, 'weights': 'uniform'}

- The second classification algorithm is the Support Vector Machine for Classification. SVM works by finding a line that maximizes the separation between a two-class data set of 2-dimensional space points [2]. In a multiclass problem, the SVM works by finding a hyperplane that maximizes the separation of the data points to their potential classes in an n-dimensional space [2]. For the multiclass classification, there are two approaches - One vs One and One vs Rest. In the one vs one approach, the system breaks down the multiclass problem into multiple binary classification problems. A binary classifier for each pair of classes. In the one vs rest approach, the system separates each class from the rest of the classes using a hyperplane.

C	0.1, 1, 10, 100, 1000
Gamma	1, 0.1, 0.01, 0.001, 0.0001
Kernel	rbf, poly

The following are the best parameters produced by grid search for different missions
Mission one - {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
Mission two - {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
Mission three - {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}

- The final classification algorithm is the Artificial Neural Network. ANN consists of input, hidden and output layers. This system we implemented comprises two hidden layers and one output layer. The two hidden layers have ReLu activation function and the output layer uses the sigmoid activation function. The input dimension is 39 for mission one, mission two and 44 for mission three. Dummies were created for the target variable in the classification problem. The first hidden layer has 20 neurons and the second hidden layer has 10 neurons. The optimizer we used is adam, loss function is binary_crossentropy, batch size is 10 and number of epochs is 200. We also use a dropout layer to reduce overfitting.

- **For the regression problem**, the trivial system takes the mean of the training outputs and uses this for all values of the testing output, essentially imitating a null model since there is no learning happening based on training data. Inevitably, this system has a low R2 score for all three missions.

- The first baseline system uses 1-nearest neighbor for regression and this also has a low R2 score for all three missions since there is only one neighbor that is used for the regression.
- The second baseline system uses Linear Regression which by far gives the best results since it is a proper linear model that trains, learns and then predicts the outputs based on all the data that we have. The R2 score for Linear Regression is the highest amongst the ones calculated so far.
- In all three systems, the third mission achieves the highest R2 score since it makes use of prior grades data to calculate the final grade.
- We implemented six different models, viz., kNN, SV Regressor, ANN, Decision Tree Regressor, Random Forest Regressor and XGBoost Regressor and the results are as follows. For all the models and all the missions, we ran a grid search to find the best parameters and then ran the final model with the best parameters for all three missions.
- kNN: kNN Regression uses the k nearest neighbors in order to predict the target value of the test data. This regressor takes in the algorithm we want to specify, the metric for calculating the distance between the point and the nearest neighbors, the number of neighbors we want to select, the power for the minkowski metric and the weights.

Parameters provided for Grid Search **Algorithm: KD Tree, Brute, Ball Tree, Auto** **Metric: Minkowski, Euclidean, Manhattan, Chebyshev** **N_Neighbors: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10** **P: 1, 2, 3** **Weights: Uniform, Distance**			
Mission	**Best Parameters**	**Best Grid Score**	**R2 Score**
Goal 1	{'algorithm': 'ball_tree', 'metric': 'minkowski',	16%	0.05

	'n_neighbors': 10, 'p': 1, 'weights': 'uniform'}		
Goal 2	{'algorithm': 'brute', 'metric': 'minkowski', 'n_neighbors': 10, 'p': 1, 'weights': 'distance'}	14%	0.2
Goal 3	{'algorithm': 'kd_tree', 'metric': 'minkowski', 'n_neighbors': 8, 'p': 2, 'weights': 'distance'}	78%	0.86

- SVR: Support Vector Regression is a supervised learning algorithm used to predict discrete values. It uses the same principle as Support Vector Classification in that it finds the best fit line or the hyperplane with maximum number of points for prediction.[3] In our case, we tune the regressor only for the regularization parameter C, and the kernel to be used in the algorithm.

Parameters provided for Grid Search **C: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15** **Kernel: Linear, RBF**			
Mission	**Best Parameters**	**Best Grid Score**	**R2 Score**
Goal 1	{'C': 3, 'kernel': 'linear'}	25.1%	0.26

Goal 2	{'C': 1, 'kernel': 'linear'}	26.1%	0.24
Goal 3	{'C': 15, 'kernel': 'rbf'}	82.8%	0.91

- MLP Regressor (ANN): The MLP Regressor works on optimizing the squared error using any one of the Limited-memory BFGS optimizer, Adam optimizer or the Stochastic Gradient Descent. We ran a grid search in order to find out which parameters work best given the values as described below.

Parameters provided for Grid Search **Activation: 100, 150, 200** **Hidden_Layer_Sizes: Relu, Tanh, Logistic** **Max_Iter: 100, 150, 200, 250, 300, 350, 400** **Momentum: 0.7, 0.8, 0.9** **N_Iter_No_Change: 10, 20**			
Mission	**Best Parameters**	**Best Grid Score**	**R2 Score**
Goal 1	{'activation': 'tanh', 'hidden_layer_sizes': 100, 'max_iter': 400, 'momentum': 0.9, 'n_iter_no_change': 20}	28.4%	0.27
Goal 2	{'activation': 'tanh', 'hidden_layer_sizes': 100, 'max_iter': 350, 'momentum':	29%	0.27

	0.7, 'n_iter_no_change': 10}		
Goal 3	{'activation': 'relu', 'hidden_layer_sizes': 200, 'max_iter': 150, 'momentum': 0.7, 'n_iter_no_change': 10}	83%	0.91

- Decision Tree Regressor: Decision Trees are predictive models that use a set of binary rules to calculate a target value. Each individual tree is a model that contains branches, nodes and leaves. A decision tree arrives at an estimate by asking a series of questions to the data, each question narrowing possible values until the model gets confident enough to make a single prediction. [4] In our model, we run a grid search specifying criterion as the tuning parameter and determining which parameter works best for which goal.

Parameters provided for Grid Search Criterion: Squared Error, Friedman MSE, Absolute Error, Poisson			
Mission	**Best Parameters**	**Best Grid Score**	**R2 Score**
Goal 1	{'criterion': 'friedman_mse'}	24%	0.26
Goal 2	{'criterion': 'friedman_mse'}	26%	0.28
Goal 3	{'criterion': 'poisson'}	72%	0.75

- Random Forest Regressor: Random Forest Regressors use the ensemble learning method in order to predict the target value. Ensemble learning is a manner of learning in which the model combines predictions from a number of machine learning algorithms in order to make a more accurate prediction compared to just using a single model. [5] In our case, we provide the criterion and the number of estimators as the parameters.

Parameters provided for Grid Search **Criterion: Squared Error, Friedman MSE, Absolute Error, Poisson** **N_Estimators: 100, 150, 200, 250, 300, 350, 400, 450**			
Mission	**Best Parameters**	**Best Grid Score**	**R2 Score**
Goal 1	{'criterion': 'absolute_error', 'n_estimators': 100}	23%	0.26
Goal 2	{'criterion': 'squared_error', 'n_estimators': 100}	24%	0.25
Goal 3	{'criterion': 'squared_error', 'n_estimators': 70}	81.6%	0.91

- XGBoost Regressor: XGBoost is similar to Random Forests in that it makes use of multiple trees to arrive at the target value. The main difference is that the XG Boost algorithm adds one tree at a time in order to correct the prediction errors made by prior models. This is where the boost part of it comes in. This method of using period knowledge and carrying the model forward is what is known as boosting. For our model, we provided the step size shrinkage and the maximum depth of the trees.

Parameters provided for Grid Search **Eta: 0.01, 0.1, 0.2, 0.3, 0.4, 0.5** **Max_Depth: 1, 2, 3, 4, 5, 6, 7**			
Mission	**Best Parameters**	**Best Grid Score**	**R2 Score**
Goal 1	{'eta': 0.01, 'max_depth': 1}	26.7%	0.27
Goal 2	{'eta': 0.01, 'max_depth': 1}	28.3%	0.28
Goal 3	{'eta': 0.01, 'max_depth': 2}	83%	0.929

4. Results and Analysis: Comparison and Interpretation

Classification:

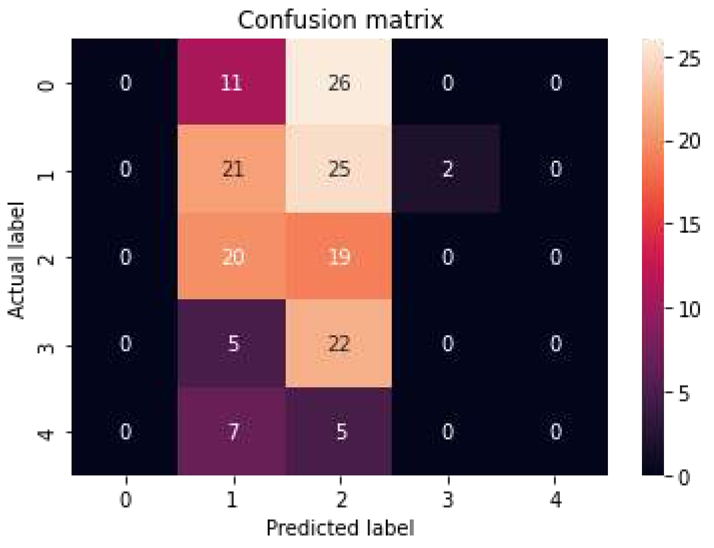
Mission one: Predicting the first-period academic performance without any prior academic performance. For mission one, the three classification models performed better than the trivial and the baseline model. For this mission, the baseline system which is the nearest centroids gave us an accuracy of 26.38% on the test dataset. Out of the three classification models we implemented, the support vector classifier gave us the best results. It has an accuracy of 37.44 % on the train dataset and 40.49 % on the test dataset. The support vector classifier was closely followed by the k nearest neighbors classifier which produced an accuracy of 34.57% and 35.58% on the test dataset respectively. The train and test accuracies, test f1 score and the confusion matrix are provided below. It can be seen that for the trivial system the predictions are centered around class 1 and class 2. This is because the weights of class 1 and class 2 and high.

Algorithm	**Train Accuracy**	**Test Accuracy**	**Test f1 score**
Trivial	-	24.53 %	0.13
Baseline	24.07 %	26.38 %	0.25
KNN	34.57 %	35.58 %	0.30

SVC	37.44 %	40.49 %	0.28
ANN	53.50 %	34.97 %	0.28

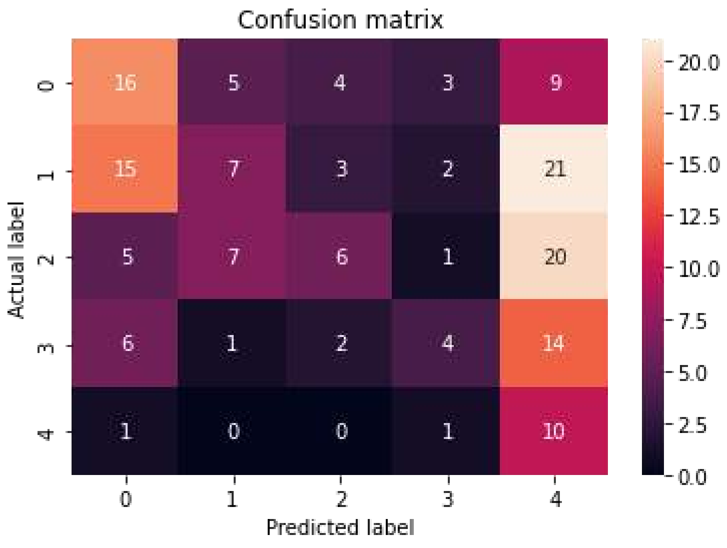
Confusion matrix

Trivial system



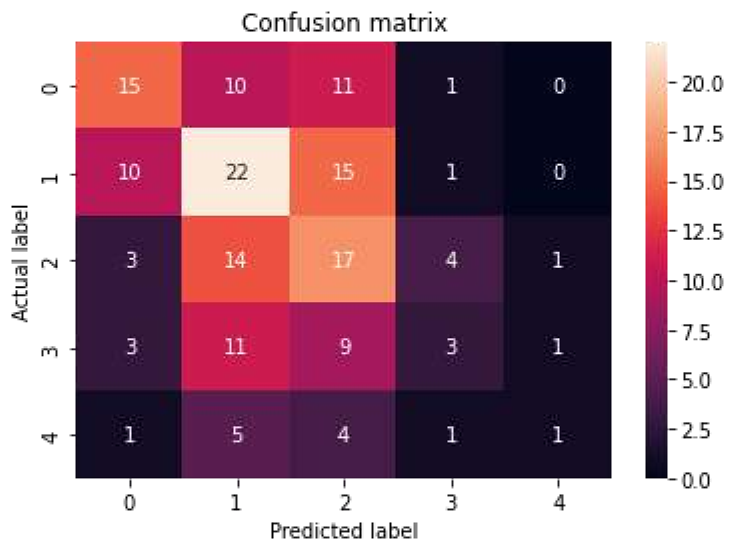
	0	1	2	3	4
0	0	11	26	0	0
1	0	21	25	2	0
2	0	20	19	0	0
3	0	5	22	0	0
4	0	7	5	0	0

Baseline system

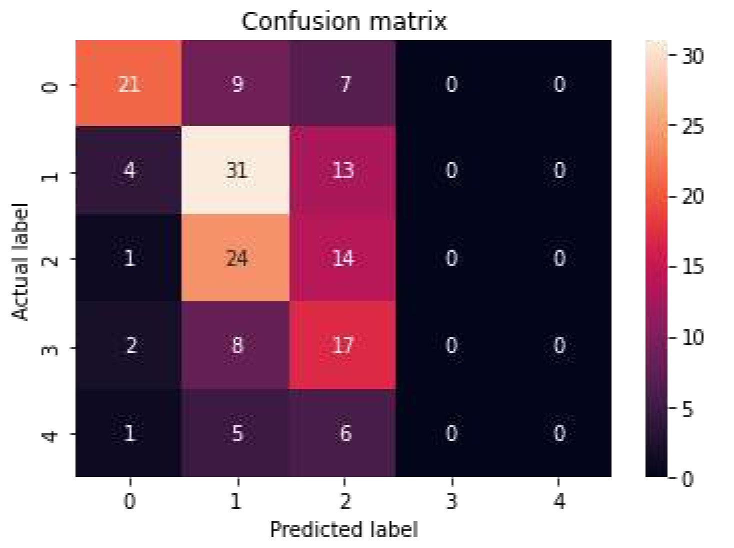


	0	1	2	3	4
0	16	5	4	3	9
1	15	7	3	2	21
2	5	7	6	1	20
3	6	1	2	4	14
4	1	0	0	1	10

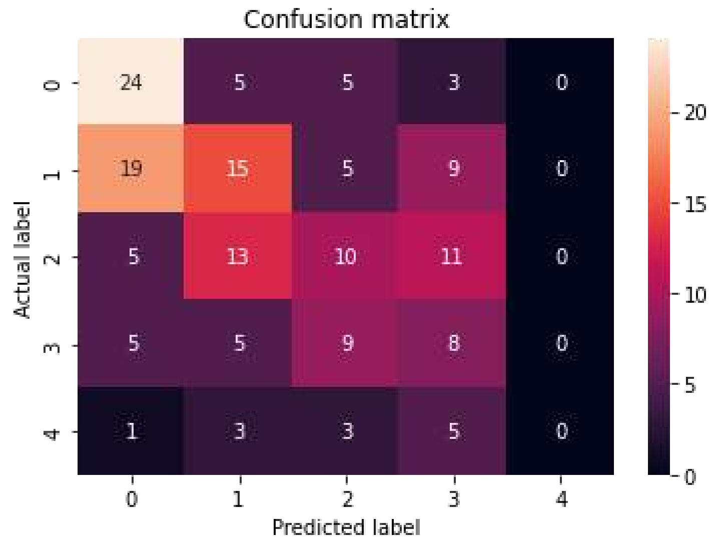
K Nearest Neighbor



Support Vector Classifier



Artificial Neural Network

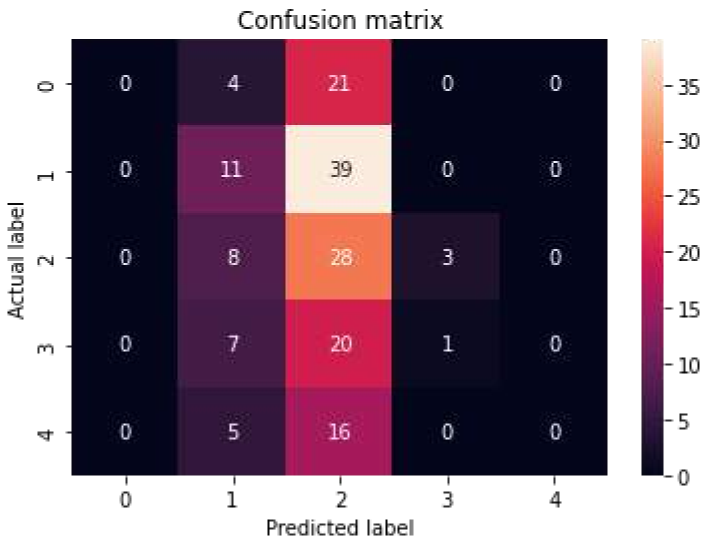


Mission two: Predicting the final-period academic performance without any prior academic performance. For this mission, the baseline system which is the nearest centroids gave us an accuracy of 26.38 % on the test dataset. All the three classification models implemented performed better than the baseline and the trivial models. Out of the three classification models we implemented, the artificial neural network gave us the best results. It has an accuracy of 47.94 % on the train dataset and 42.33 % on the test dataset. The artificial neural network was closely followed by the support vector classifier and the k nearest neighbors classifier which produced an accuracy of 36.20 % and 32.52 % on the test dataset respectively. The train and test accuracies, test f1 score and the confusion matrix are listed below. The scores for the second mission and first mission are almost similar. The models are not performing so well without previous academic performance.

Algorithm	**Train Accuracy**	**Test Accuracy**	**Test f1 score**
Trivial	-	24.85 %	0.13
Baseline	29.62 %	26.38 %	0.25
KNN	36.01 %	32.52 %	0.30
SVC	36.83 %	36.20 %	0.33
ANN	47.94 %	42.33 %	0.39

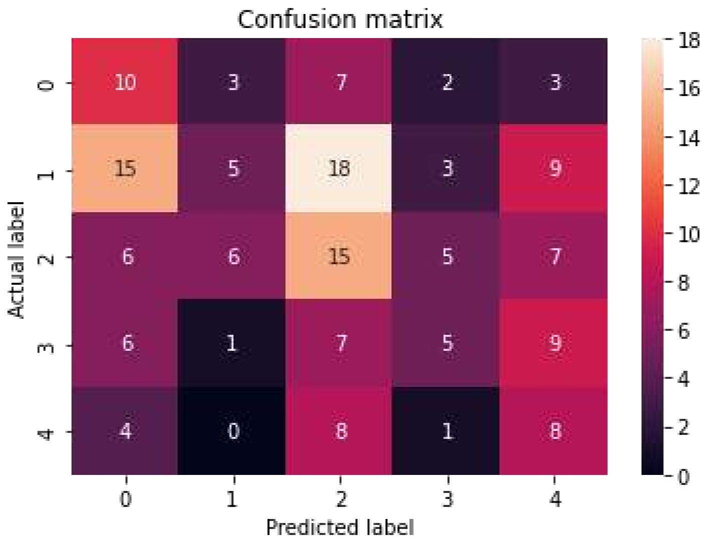
Confusion matrix

Trivial system



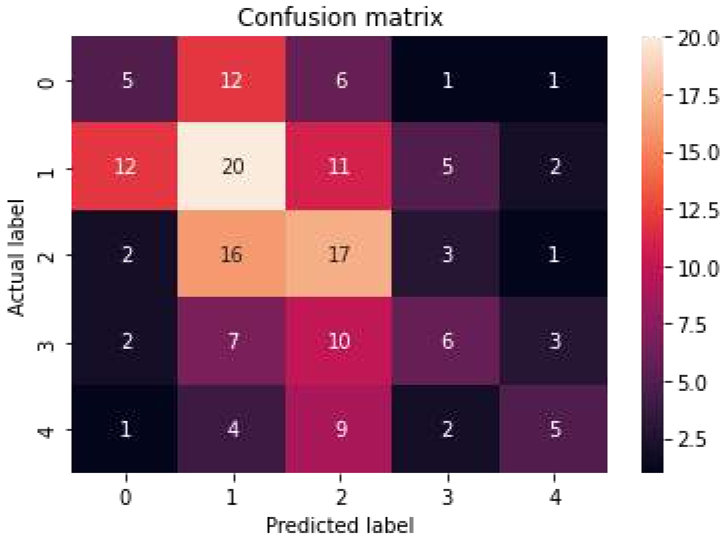
Actual \ Predicted	0	1	2	3	4
0	0	4	21	0	0
1	0	11	39	0	0
2	0	8	28	3	0
3	0	7	20	1	0
4	0	5	16	0	0

Baseline

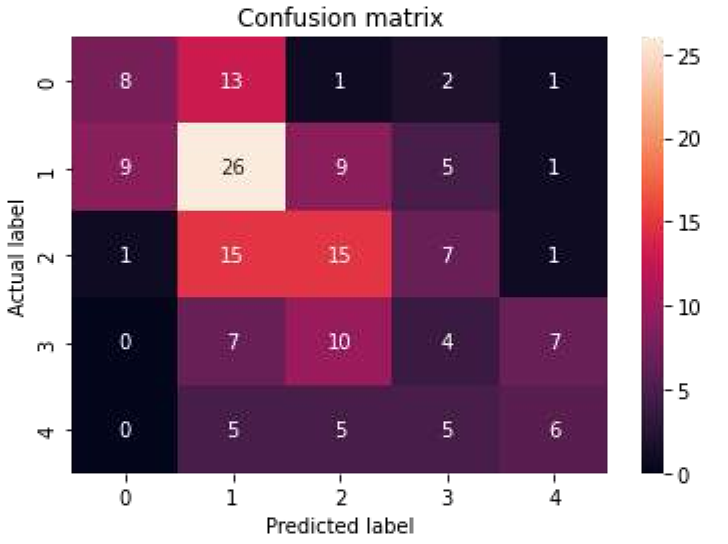


Actual \ Predicted	0	1	2	3	4
0	10	3	7	2	3
1	15	5	18	3	9
2	6	6	15	5	7
3	6	1	7	5	9
4	4	0	8	1	8

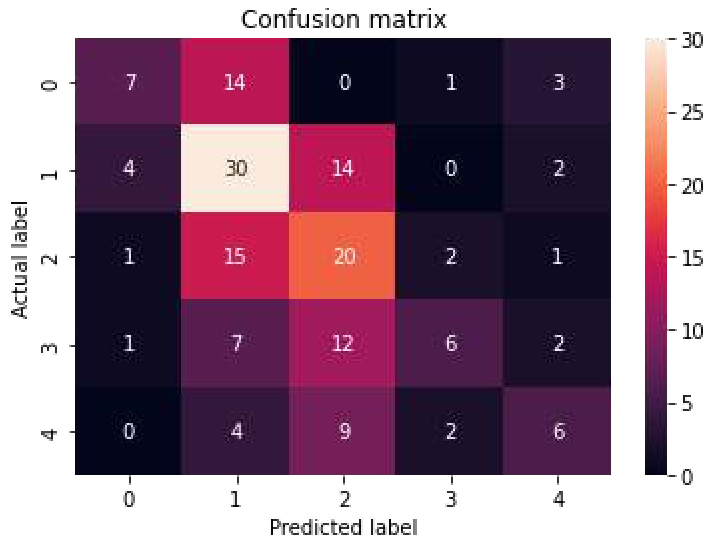
K Nearest Neighbor



Support Vector Classifier



Artificial Neural Network

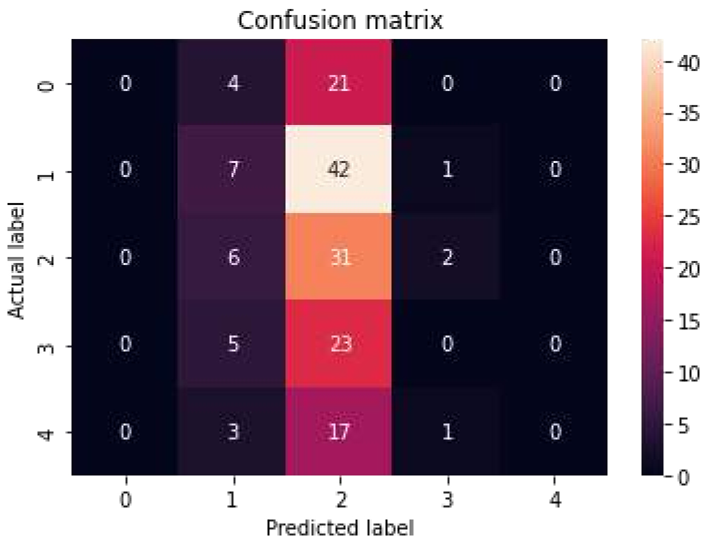


Mission three: Predicting the final-period academic performance with prior academic performance. For this mission, the baseline system which is the nearest centroids gave us an accuracy of 60.12 % on the test dataset. Out of the three classification models we implemented, the support vector classifier gave us the best results. It has an accuracy of 72.01 % on the train dataset and 76.07 % on the test dataset. The support vector classifier was closely followed by the artificial neural network and the k nearest neighbors classifier which produced an accuracy of 74.23 % and 74.84 % respectively. The train and test accuracies, test f1 score and the confusion matrix are listed below. All the three models perform better than the baseline and the trivial system.

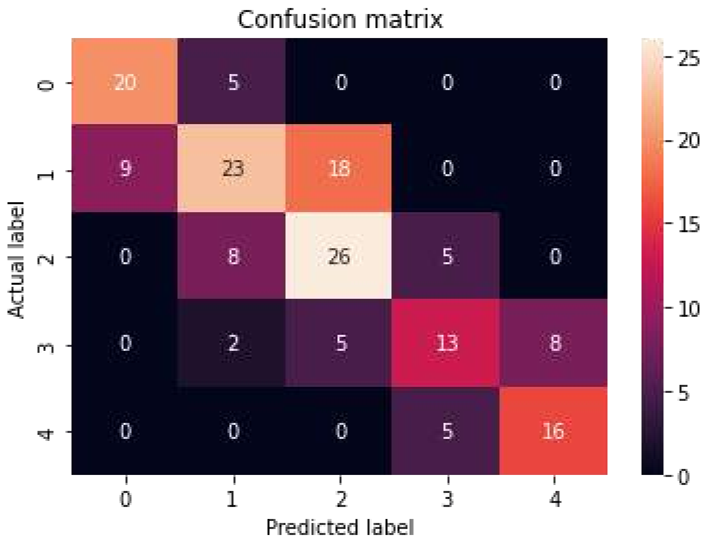
Algorithm	**Train Accuracy**	**Test Accuracy**	**Test f1 score**
Trivial	-	23.31 %	0.10
Baseline	67.69 %	60.12 %	0.61
KNN	70.77 %	74.84 %	0.75
SVC	72.01 %	76.07 %	0.75
ANN	73.86 %	74.23 %	0.71

Confusion matrix

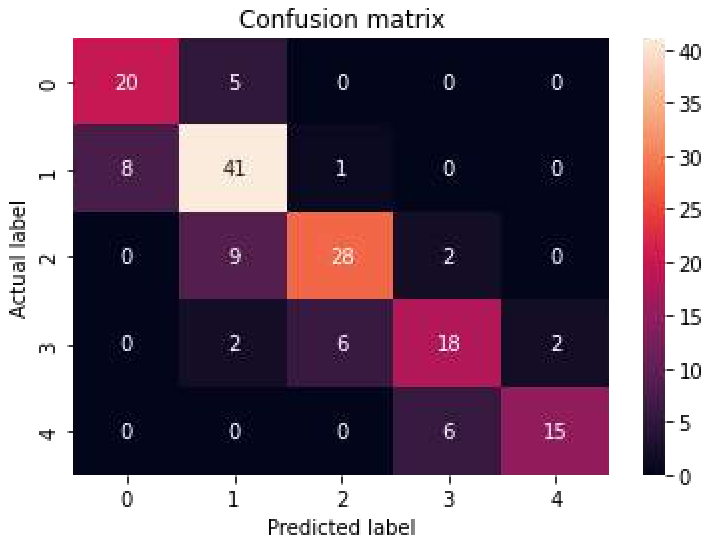
Trivial system



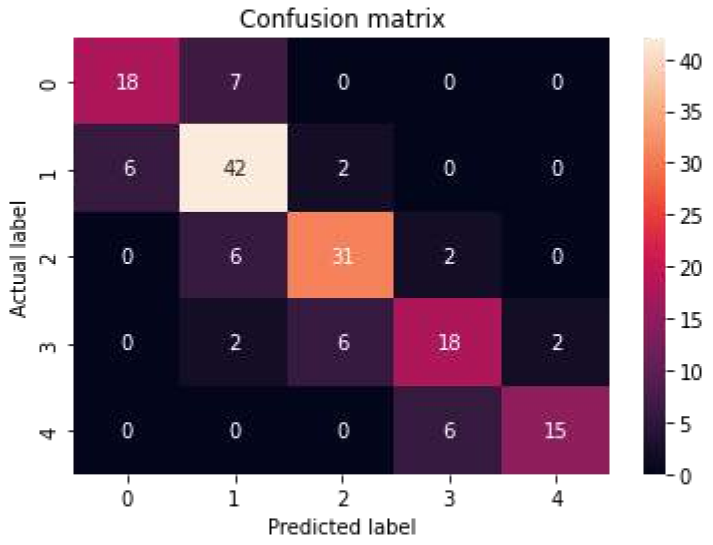
Baseline



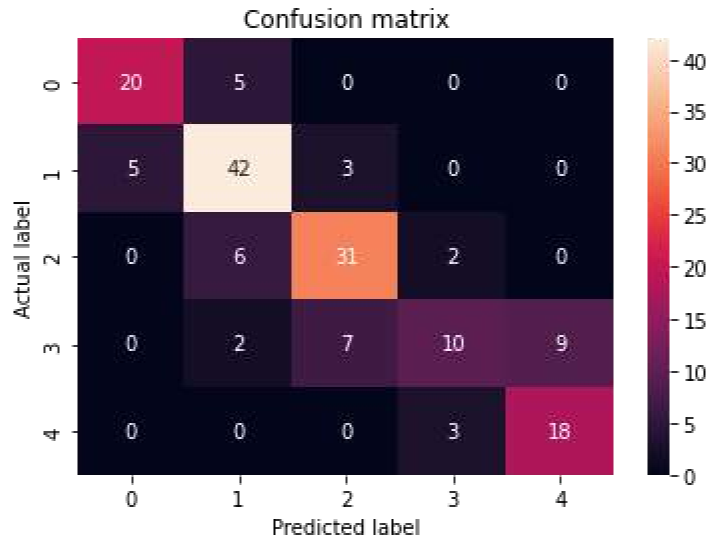
K Nearest Neighbor



Support Vector Classifier



Artificial Neural Network



From the results for all three missions, it can be seen that the classification models perform better when the prior academic performance is available.

Regression:

Mission 1:

Model	Best Grid Score %	R2 Score	Rank by R2 Score
KNN	16	0.05	3
SVR	25.1	0.26	2
ANN	28.4	0.27	1
Decision Tree	24	0.26	2
Random Forest	23	0.26	2
XG Boost	26.7	0.27	1

Mission 2:

Model	**Best Grid Score %**	**R2 Score**	**Rank by R2 Score**
KNN	14	0.2	5
SVR	26.1	0.24	4
ANN	29	0.27	2
Decision Tree	26	0.28	1
Random Forest	24	0.25	3
XG Boost	28.3	0.28	1

Mission 3:

Model	**Best Grid Score %**	**R2 Score**	**Rank by R2 Score**
KNN	78	0.86	3
SVR	82.8	0.91	2
ANN	83	0.91	2
Decision Tree	72	0.75	4
Random Forest	81.6	0.91	2
XG Boost	83	0.929	1

As we can see from the tables above, ANN and XG Boost are the best models for our dataset, followed by Random Forest and Decision Trees.

5. Libraries used and what you coded yourself

- Preprocessing

  For preprocessing, we mapped the 4 multivalued categorical attributes to numeric values without the use of libraries. The libraries used for different functions are listed below

Library	**Module**	**Function**
sklearn	preprocessing	LabelEncoding
sklearn	preprocessing	StandardScaler
sklearn	preprocessing	MinMaxScaler
imblearn	over sampling	SMOTE
pandas	-	get_dummies

- Feature engineering

  For feature engineering, we added new features without the use of libraries. We used libraries to perform feature selection. The libraries used are listed below

Library	**Module**	**Function**
sklearn	feature selection	chi2
sklearn	feature selection	f_classif
sklearn	feature selection	SelectKBest
sklearn	feature selection	RFE, RFECV

- Feature dimensionality reduction

  For dimensionality reduction, we used principal component analysis from the sklearn library

Library	Module	Function
sklearn	decomposition	PCA

- Trivial model

  The trivial model was coded from scratch and no libraries were used.

- Baseline model

  To implement the baseline model, we used the NearestCentroid, Linear Regressor and kNeighborsRegressor function from the sklearn library

Library	Module	Function
sklearn	neighbors	NearestCentroid
sklearn	linear_model	LinearRegressor
sklearn	linear_model	kNeighborsRegressor

- K Nearest Neighbors

  The k nearest neighbors classifier was implemented using functions from sklearn as stated below

Library	Module	Function
sklearn	neighbors	KNeighborsClassifier
sklearn	model_selection	GridSearchCV
sklearn	neighbors	kNeighborsRegressor

- Support Vector

  The support vector was implemented using functions from sklearn as stated below

Library	**Module**	**Function**
sklearn	svm	SVC
sklearn	svm	SVR
sklearn	model_selection	GridSearchCV

- Artificial Neural Network

  The artificial neural network was implemented using different functions as stated below

Library	**Module**	**Function**
keras	models	sequential
keras	layers	dense dropout

- All other models

Library	**Module**	**Function**
sklearn	tree	DecisionTreeRegressor
sklearn	ensemble	RandomForestRegressor
xgboost	-	XGBRegressor
sklearn	model selection	GridSearchCV KFold cross_val_score

- Metrics

Library	**Module**	**Function**

sklearn	metrics	accuracy_score r2_score confusion_matrix mean_squared_error f1_score

6. Contributions of each team member

Amulya Kallakuri - Regression problem

Nithyashree Manohar - Classification Problem

7. Summary and conclusions

To summarize, all the classification and regression models perform better than the baseline and trivial models. The models perform better when prior academic performance is available. Overall, Support Vector Classifier and Artificial Neural Network perform well for classification. Artificial Neural Network and XGBoost perform well for regression. We intend on working further on this project by implementing a couple of more models.

References

[1] Jolliffe Ian T. and Cadima Jorge, 2016. Principal component analysis: a review and recent developments *Phil. Trans. R. Soc. A.*37420150202220150202

[2] Understanding Support Vector Machine algorithm from examples [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code

[3] Unlocking the True Power of Support Vector Regression [Online]. Available: https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0

[4] Decision Tree Regressor Explained In Depth [Online]. Available: https://gdcoder.com/decision-tree-regressor-explained-in-depth/

[5] Random Forest Regression [Online]. Available: https://levelup.gitconnected.com/random-forest-regression-209c0f354c84