

# Students' performance Prediction

Data Set: Student performance (in Portuguese schools)

Authors: Jiazhi Dai, Weiqiang Qian

email: [jiazhida@usc.edu](mailto:jiazhida@usc.edu), [qianweiq@usc.edu](mailto:qianweiq@usc.edu)

## 1. Abstract

Our problem is to predict students' academic performance in secondary schools with both classification methods and regression methods. The data is from students in secondary education of two Portuguese schools which have 30 input attributes based on student questionnaires and 3 scores at 3 periods. We have set up a trivial Model and a baseline Model for both the classification system and the regression system to compare the final predictions produced by our own model with those produced by these models. The model we designed included SVM, kNN, decision Tree, Naive Bayes, PCA+Random Forest and rain for predicting regression problems SVR, kNN, Ridge regression and Lasso regression. And we use different indicators to comprehensively measure the performance of the model.

For classification mission 1 & 2, we will choose PCA+Random Forest to be the final classification model and the results (accuracy and macro f1-score) are 0.337423, 0.337494 for mission 1; 0.386503, 0.307730 for mission 2. For classification mission 3, we will choose SVM to be the final classification model and the results (accuracy and macro f1-score) are 0.717791, 0.775741. For regression mission 1 & 2, we will choose linear regression to be the best regression model which is also the baseline model; the results (RMSE, MAE, R-square) are 2.416933, 1.765720, 0.275955 for mission 1 and 2.717781, 1.949769, 0.266989. For regression mission 3, we will choose lasso regression to be the best regression model, the results (RMSE, MAE, R-square) are 0.956400, 0.693627, 0.909226.

We finally found that it was not very accurate to directly predict scores with this data set without considering G1 and G2 scores when only a simple pretreatment was done. After the addition of G1 and G2, the accuracy of the whole data set was significantly improved, and the training data had the best effect after a certain degree of dimension extraction.

**Key words:** SVM, kNN, decision Tree, Naive Bayes, PCA+Random Forest, Ridge regression, Lasso regression

## 2. Introduction

### 2.1. Problem Statement and Goals

We choose only the Portuguese dataset in the students' performance data set which is posted on the UCI website (includes mathematics and Portuguese topics) as the full dataset. The number of input attributes per data point is 30, not counting the grade features (Gx). 13 of the attributes are integer valued categorical, and 4 are multi-valued categorical. There are no missing values. Our goal is trying to find a suitable ML method to predict 3 kinds of students' performance well:

1. Remove G2 and G3 to predict G1.
2. Remove G1 and G2 to predict G3.
3. Keep G1 and G2 as features to predict G3.

For classification problem, the grade level has been divided into 5-classes:

Table 1. Classification rules

I	II	III	IV	V
16-20	14-15	12-13	10-11	0-9
A	B	C	D	F

### 2.2. Examine methods

We will use both classification and regression methods to predict the dataset. Using Accuracy, Macro f1-score and confusion matrix to represent classification performance. Using RMSE, MAE and  $R^2$  to represent regression performance.

Accuracy: as for classification, accuracy is the normal way to represent classification performance. The formula are as follows:

$$Accuracy = \frac{\text{the number of test cases that we predict correctly}}{\text{the total number of test cases}}$$

The higher accuracy is, the better performance will be.

Macro f1-score: it is the mean of f1-score over all classes. f1-score, which consist of recall and precision:[1]

(If we define the target class(e.g. if I want to test the f1-score of class 'A', 'A' is my target class now and the other are the non-target class) as P(positive), non-target class as N(negative), we predict correctly will be marked T and predict wrong will be marked F.)

$$recall = \frac{TP}{TP+FN} = \frac{TP}{P}$$

$$Prediction = \frac{TP}{TP+FP}$$

$$F1 = \frac{2*recall*Precision}{recall+Precision}$$

From the formula we can find that recall represents the ability to recognize target class while precision represents the ability to distinguish non-target class. The higher recall/precision score is, the better the model can recognize target class/distinguish non-target class.

Confusion matrix: it is a visualization matrix to reveal the differences of the predicted classes and actual classes. The ratio of the diagonal is the accurate rate of each predicted class.

RMSE: it is the root of MSE(mean square error). We often use RMSE to represent the degree of deviation from the correct value.

$$RMSE = \sqrt{\frac{1}{n} \sum (\hat{y}(predict) - y(true))^2}$$

MAE: we also use MAE(mean absolute error) to represent the degree of deviation from the correct value. Compared with MSE, MAE lowers the effects of bias to the whole system.

$$MAE = \frac{1}{n} \sum |\hat{y}(predict) - y(true)|$$

R-square: given a series of truth values  $\hat{y}$  and corresponding predictions  $y$  and the mean of true values  $\bar{y}$ . [2]The formula is:

$$R^2 = 1 - \frac{\sum (\hat{y} - y)^2}{\sum (y - \bar{y})^2}$$

- R-square =1: In the best case, all predicted values are equal to true values.
- R-square =0: one possibility is simply to predict that all  $y$  values are equal to the mean of  $y$ , where all  $\hat{y}$  are equal to  $\bar{y}$  (i.e. the average of true  $Y$  values), but there are other possibilities.
- R-square <0: the prediction ability of the model is poor, which is worse than "simply predicting that all  $y$  values are equal to the mean value of  $y$ ". This means that the wrong model may be used, or the assumptions of the model may not make sense.

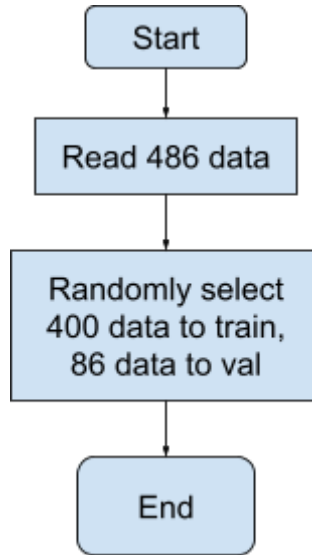
There is no lower bound for the minimum value of R squared, because predictions can be as bad as they want. So the range of R squared is (0,1].

### 3. Approach and Implementation

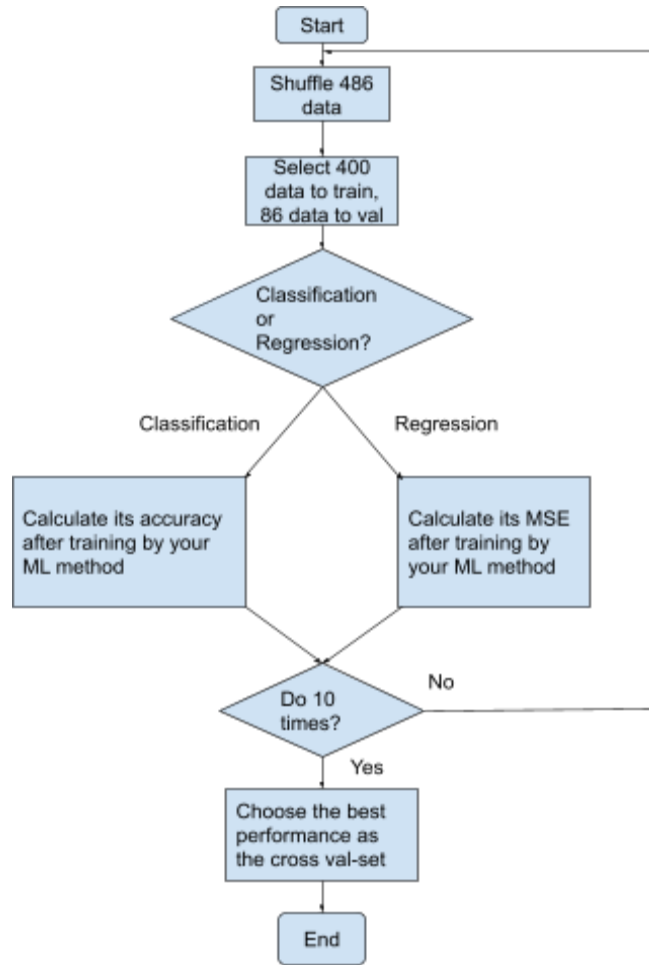
#### 3.1. Dataset Usage

There are 486 data in the training set, 163 data in the test set, we extract 86 data from the train set as our validation set, the rest 400 data as our train set. For cross validation, we randomly choose 400 data as our train set and 86 data as our validation set. Then we train them 10 times.

Here is the usage of validation set and cross validation set:



Flowchart1. Validation set.



Flowchart2. Cross validation set.

We also implement data normalization, one-hot, feature-select to the dataset.

### 3.2. Preprocessing

Both non-binary categorical variable and binary categorical variable are in this dataset, we use one-hot to convert non-binary categorical variables to binary one-hot variables (e.g., values1 or 0).

A one-hot is a group of bits among which the legal combinations of values are only those with a single high (1) bit and all the others low (0). In statistics, dummy variables represent a similar technique for representing categorical data.

For example, feature Mjob, Fjob, reason, guardian are 4 categorical non-binary features, we use one-hot to include them.

Table 2. Example for one-hot.

	feature1	feature2	feature3
sample1	01	1000	100

sample2	10	0100	010
sample3	01	0010	010
sample4	10	0001	001

The value of each feature has different scale, we use feature-wise minmax normalization to convert all the features in the range (0,1), it can prevent the features which include large values from having a huge weight (cause misprediction).

It's obvious that G1, G2, G3 have a strong correlation to each other. We chose not to normalize these three features but the rest of the features.

Equation for feature-wise normalization:

$$v_n[i] = \frac{x_n[i] - \min_n x_n[i]}{\max_n x_n[i] - \min_n x_n[i]}$$

PCA (Principal Component Analysis) is simply taking only the T most important e-directions or principal components. It can reduce the dimensionality for this dataset.

There are 33 features in this dataset, using one-hot will increase the dimension, many features may not have correlation to the academic performance. Here is how PCA works:

Steps involved in the PCA:

1. Standardize the dataset.
2. Calculate the covariance matrix for the features in the dataset.
3. Calculate the eigenvalues and eigenvectors for the covariance matrix.
4. Sort eigenvalues and their corresponding eigenvectors.
5. Pick k eigenvalues and form a matrix of eigenvectors.
6. Transform the original matrix.

### 3.3. Training, classification and/or regression, and model selection

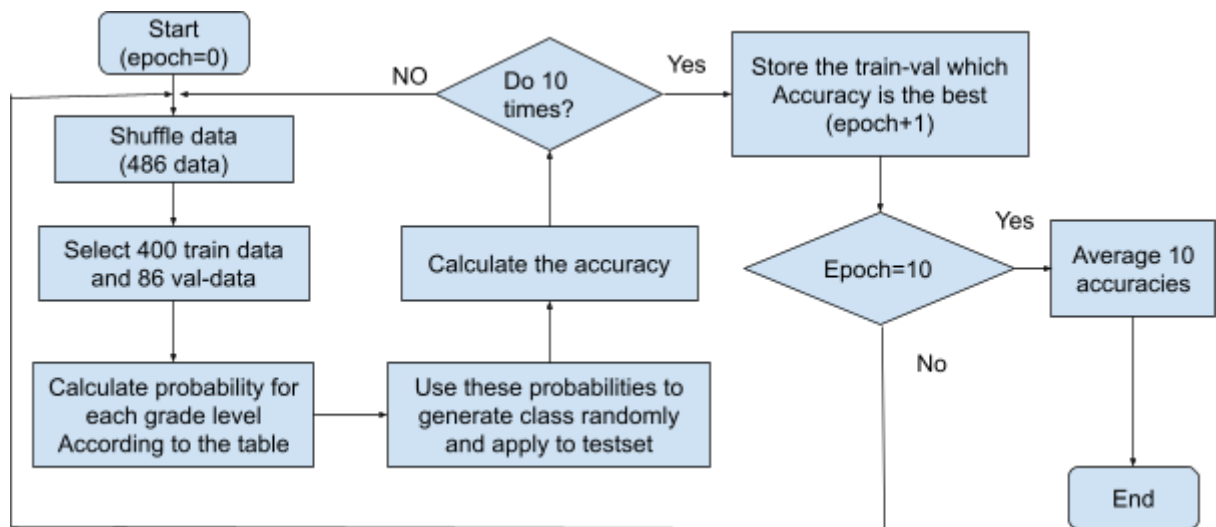
Besides trivial systems and baseline systems, there are other machine learning models used in our project.

For classification: the algorithms we used are Naive Bayes, kNN(k-nearest neighbors), SVM(Support Vector Machine), Decision tree, PCA+Random Forest.

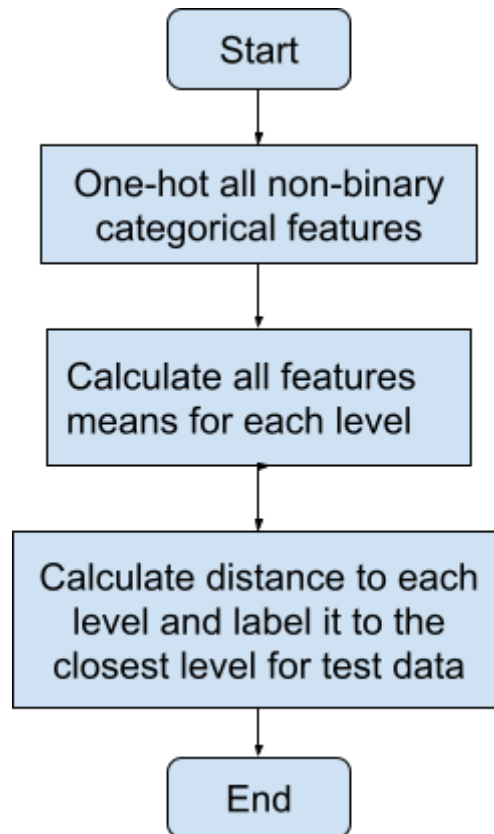
For regression: the algorithms we used are Linear regression, kNN regression, SVR(Support Vector Regression), Ridge regression, Lasso Regression.

where kNN and SVM models are used both for classification and regression.

#### 3.3.1 Trivial and baseline system for Classification

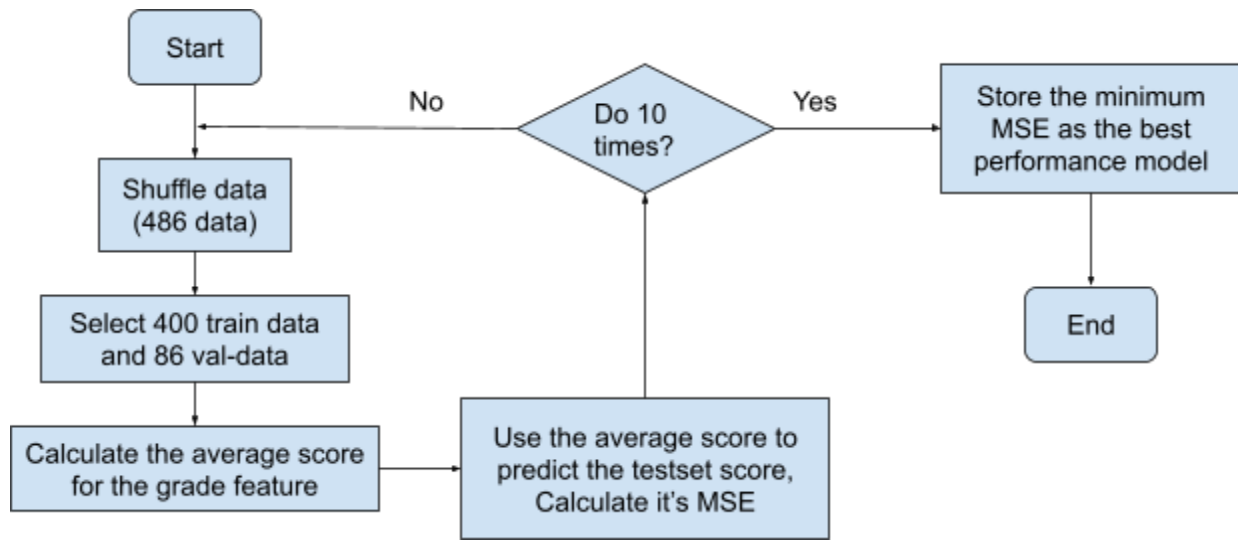


Flowchart 3. Trivial System



Flowchart 4. Baseline System

### 3.3.2 Trivial and baseline system for Regression



Flowchart 5. Trivial System

Baseline is the specific condition for KNN ML algorithm, and the rest flowchart for will be the same except some differences in the training part.

### 3.3.3 SVM classification and regression

The method of Non-linear SVM is to add the eigenvector from low-dimension to high-dimension, so we need kernel function to realize this. There are several commonly used functions like linear, RBF, polynomial and sigmoid kernels. Then we use kernel function and penalty parameter to construct and solve a convex quadratic programming problem, also we need to consider its KKT conditions, then we can get the decision function to classify our test data, In our own svm classifier. We use `sklearn.svm.SVC` to realize classification. We choose the RBF kernel, and use l2 regularization as the penalty, and let the coefficient gamma as  $1/n\_features$ . The other parameters are default. [4]Then for regression, SVR uses slack variable to consider some linear indivisibility condition but linear regression does not, we use `sklearn.svm.SVR`, the kernel is the default rbf, also with l2 regularization, the elipson tube is 0.2.

### 3.5.4 KNN classification and regression

KNN method is easy to understand, Its working mechanism is as follows: given test samples, k training samples closest to them are found based on some distance measurement, and then predictions are made based on the information of these K "neighbors".



There are three main methods of measuring some distance mentioned above: Manhattan distance, Euclidean distance and Minkowski distance. In our project, Euclidean distance is used for distance calculation.

KNN is a kind of lazy learning model, which performs little or incomplete processing on the training data set. Unlike diligent learning such as simple linear regression, KNN does not generate parameters from the model in the training stage. Simply speaking, KNN is a nonparametric model, the number of parameters is not fixed, it may increase with the increase of the number of training columns.

It's similar to regression. New prediction example looks for k-nearest neighbors and then takes the mean value of the target value of the K samples as the predicted value of the new sample.

We tried 10 times for the parameter(which is the number of neighbors) in classification and regression. Just pick it randomly from 3-20 and finally we select 5, 15 respectively for the neighbors of classification and regression.

In our project, we import sklearn KNeighborsClassifier and KNeighborsRegressor to implement the dataset.

#### 3.5.5 Naive Bayesian classification

It's a classifier based on probability. The training process of the Bayesian classifier is parameter estimation. The process of estimating parameters by maximum likelihood method is generally divided into the following four steps:

- Write the likelihood function;
- Take logarithm of likelihood function and arrange;
- Take the derivative, set the partial derivative to 0, and get the likelihood system;
- Solve the likelihood equations and get all the parameters.

In our project, we use sklearn.naive\_bayes.MultinomialNB to train the data. The parameters all use the default setting.

#### 3.5.6 Decision tree classification

Decision tree is a basic classification method. The decision tree model is a tree structure. In the classification problem, it represents the process of classifying instances based on features. It can be regarded as a set of if-then rules or a conditional probability distribution defined in feature space and class space.

Its main advantages are readable models and fast classification speed. During learning, the decision tree model is established according to the principle of minimizing loss function by using training data. In prediction, the new data are classified by decision tree model

Where each non-leaf node represents a test on a feature attribute, each branch represents the output of this feature attribute in a certain range, and each leaf node stores a category.

The process of using decision trees to make decisions is to test the corresponding characteristic attributes in the items to be classified from the root node, and select the output branches according to their values until it reaches the leaf node, and take the categories stored in the leaf node as the decision result.

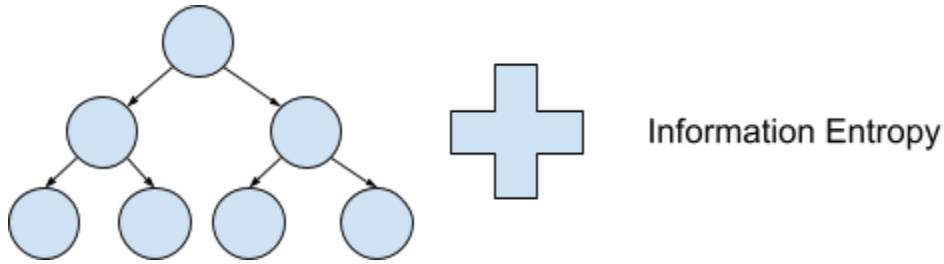


Figure1. Decision tree method

The generation process of a decision tree is mainly divided into the following three parts:

- Feature selection: Feature selection refers to the selection of a feature from the numerous features in the training data as the splitting standard of the current node. There are many different quantitative evaluation standards for how to select features, so as to derive different decision tree algorithms.
- Decision tree generation: recursively generate sub-nodes from top to bottom according to the selected feature evaluation criteria until the data set is not separable and the decision tree stops growing. Recursion is the easiest way to understand a tree structure.
- Pruning: Decision trees are easy to overfit. Generally, pruning is required to reduce the size of the tree structure and alleviate overfitting. There are two kinds of pruning techniques: pre-pruning and post-pruning

Information entropy is uncertainty. When the distribution is uniform, the uncertainty is maximum, and the entropy is maximum. When a feature is selected to classify the data set, the information entropy of the data set after classification is smaller than that before classification, and the difference is represented as information gain. Information gain can measure the influence of a feature on classification results.

The calculate method for information entropy is :

$$Info(D) = - \sum_{i=1}^c p_i \log_2(p_i)$$

In our project, we import `sklearn.tree.DecisionTreeClassifier` to implement a decision tree to train our data.

### 3.5.7 PCA+Random Forest classification

Random forest is an algorithm that integrates multiple trees through the idea of Ensemble Learning.[6] Its basic unit is the decision tree, and its essence belongs to the Ensemble Learning method, a branch of machine Learning. Random forest solves the shortcoming of weak generalization ability of decision trees. There are two key words in the name of random forest, one is "random", the other is "forest".

Random means random selection of samples, random selection of features. That is, each tree selects a fixed number of sample sets from the whole training sample set, and then selects a fixed number of feature sets, so as to construct each decision tree in the random forest.

From an intuitive perspective, each decision tree is a classifier, so for an input sample, N trees will have N classification results. A random forest, on the other hand, aggregates the results of all categories, designating the category with the most votes as the final output, which is the simplest Bagging idea.

Random forest is supposed to perform better than decision trees. Also, we use PCA to select T-th important feature to deduce the feature dimension to make the training faster.

We use `sklearn.ensemble.RandomForestClassifier` to realize it and train our data.

### 3.5.8 Ridge Regression and Lasso Regression

Ridge regression is to add a regularizer term  $\lambda ||\omega||_2^2$  (also called L2 regularizer) to MSE criterion, Lasso Regression is to add a L1 regularizer  $\lambda ||\omega||_1$  to MSE criterion. [3] We use `sklearn.linear_model.Ridge` and `sklearn.linear_model.Lasso` to realize these two regression methods. [5] The  $\lambda$  is set to be default value (which is 1).

## 4. Results and Analysis: Comparison and Interpretation

### 4.1. Result

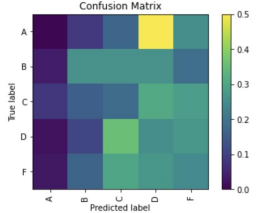
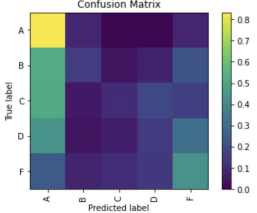
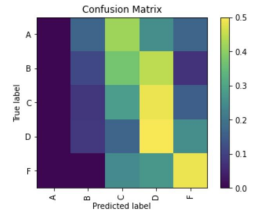
Classification (accuracy and macro f1-score):

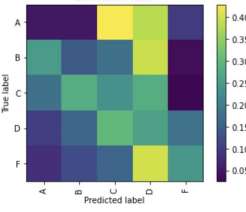
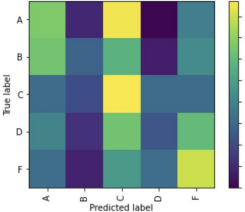
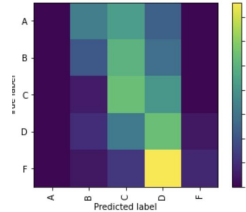
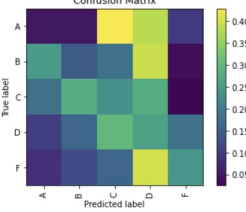
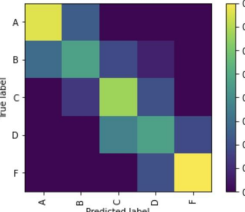
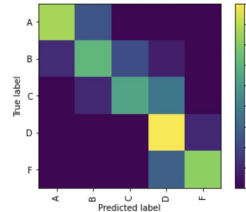
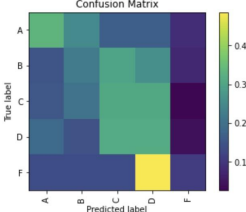
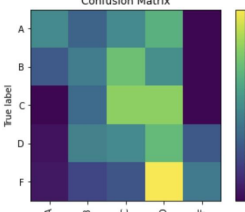
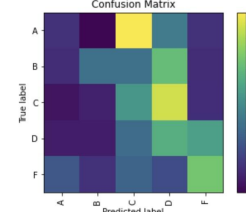
Table 3. Classification models performance(accuracy and macro f1-score)

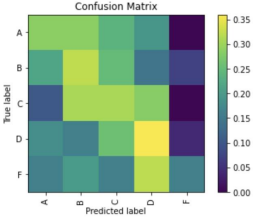
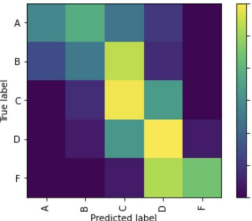
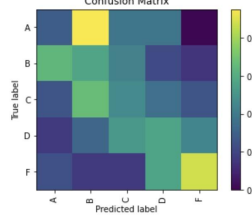
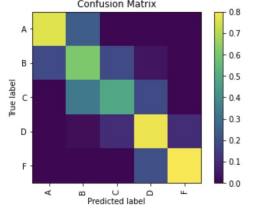
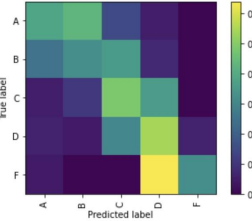
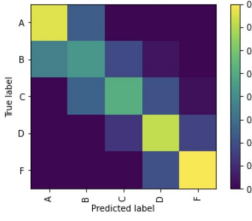
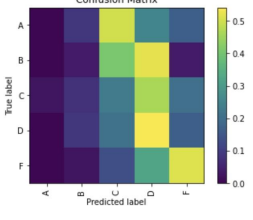
		trivial	baseline	SVM	kNN(5)	Naive Bayes
mission 1	accuracy	0.220245	0.251533	0.343558	0.251533	0.306748
	macro f1	0.184261	0.238501	0.260692	0.234982	0.297341
mission 2	accuracy	0.220245	0.269938	0.343558	0.300613	0.306748
	macro f1	0.187868	0.256153	0.208572	0.285215	0.279254
mission 3	accuracy	0.220245	0.601226	0.717791	0.680981	0.435582
	macro f1	0.187868	0.615058	0.715741	0.683336	0.420864
		trivial	baseline	decision tree	PCA(20)+Random Forest	
mission 1	accuracy	0.220245	0.251533	0.325153	0.337423	
	macro f1	0.184261	0.238501	0.289183	0.243094	
mission 2	accuracy	0.220245	0.269938	0.276073	0.386503	
	macro f1	0.187868	0.256153	0.269722	0.307730	
mission 3	accuracy	0.220245	0.601226	0.638036	0.705521	
	macro f1	0.187868	0.615058	0.558505	0.681784	

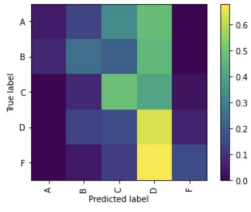
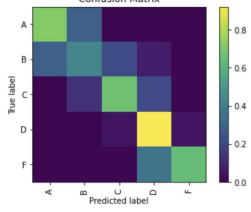
Classification(confusion matrices):

Table 4. Classification models performance(confusion matrices)

	confusion matrix	trivial	baseline	SVM
mission 1	data	[[ 0, 1, 2, 6, 3], [ 1, 7, 7, 7, 5], [ 3, 6, 7, 12, 11], [ 1, 5, 17, 12, 13], [ 1, 6, 11, 10, 9]]	[10 1 0 0 1] [14 4 1 2 6] [20 2 4 7 6] [21 2 3 7 15] [ 9 3 4 5 16]	[ 0 2 5 3 2] [ 0 3 10 12 2] [ 0 3 11 19 6] [ 0 4 8 24 12] [ 0 0 9 10 18]
	map			

mission 2	data	[[ 1, 1, 9, 8, 2], [ 7, 4, 5, 11, 1], [ 7, 11, 9, 11, 1], [ 5, 8, 15, 13, 9], [ 2, 3, 4, 10, 6]]	[ 7 1 9 0 4] [ 9 4 8 1 6] [ 6 4 17 6 6] [10 3 16 6 15] [ 4 1 6 4 10]	[ 0 7 9 5 0] [ 0 6 14 8 0] [ 0 2 21 16 0] [ 0 5 16 27 2] [ 0 1 3 19 2]
	map			
mission 3	data	[[ 1, 1, 9, 8, 2], [ 7, 4, 5, 11, 1], [ 7, 11, 9, 11, 1], [ 5, 8, 15, 13, 9], [ 2, 3, 4, 10, 6]]	[16 5 0 0 0] [ 8 13 5 2 0] [ 0 5 26 8 0] [ 0 0 18 23 9] [ 0 0 0 5 20]	[16, 5, 0, 0, 0], [ 3, 17, 6, 2, 0], [ 0, 4, 21, 14, 0], [ 0, 0, 0, 45, 5], [ 0, 0, 0, 7, 18]
	map			
	confusion matrix	kNN	Naive Bayes	decision tree
mission 1	data	[ 4 3 2 2 1] [ 4 6 8 7 2] [ 6 8 12 12 1] [ 9 7 15 15 2] [ 5 5 5 18 4]	[ 3 2 3 4 0] [ 4 6 10 7 0] [ 0 7 16 16 0] [ 1 11 12 17 7] [ 1 4 5 19 8]	[ 1 0 7 3 1] [ 2 6 6 11 2] [ 1 2 12 21 3] [ 2 2 10 18 16] [ 6 3 7 5 16]
	map			
mission 2	data	[ 6 6 5 4 0] [ 6 9 7 4 2] [ 4 12 12 11 0] [ 9 8 13 18 2]	[ 6 8 5 2 0] [ 4 7 15 2 0] [ 0 3 23 13 0] [ 0 2 16 30 2]	[ 3 10 4 4 0] [ 9 8 6 3 2] [ 5 13 9 7 5] [ 4 8 13 14 11]

		[ 4 5 4 8 4]	[ 0 0 1 13 11]	[ 3 2 2 7 11]
	map	 <p>Confusion Matrix</p> <p>True label \ Predicted label: A, B, C, D, E</p> <p>Color scale: 0.00 to 0.35</p>	 <p>Confusion Matrix</p> <p>True label \ Predicted label: A, B, C, D, E</p> <p>Color scale: 0.0 to 0.6</p>	 <p>Confusion Matrix</p> <p>True label \ Predicted label: A, B, C, D, E</p> <p>Color scale: 0.0 to 0.4</p>
mission 3	data	[16 5 0 0 0] [5 17 5 1 0] [0 13 19 7 0] [0 1 5 39 5] [0 0 0 5 20]	[8 9 3 1 0] [7 9 10 2 0] [2 4 19 14 0] [3 2 15 27 3] [1 0 0 16 8]	[16 5 0 0 0] [10 12 5 1 0] [0 10 20 8 1] [0 0 6 36 8] [0 0 0 5 20]
	map	 <p>Confusion Matrix</p> <p>True label \ Predicted label: A, B, C, D, E</p> <p>Color scale: 0.0 to 0.8</p>	 <p>Confusion Matrix</p> <p>True label \ Predicted label: A, B, C, D, E</p> <p>Color scale: 0.0 to 0.6</p>	 <p>Confusion Matrix</p> <p>True label \ Predicted label: A, B, C, D, E</p> <p>Color scale: 0.0 to 0.8</p>
	confusion matrix	PCA(20)+RF		
mission 1	data	[0 1 6 3 2] [0 1 11 14 1] [1 3 9 18 8] [0 4 10 26 8] [0 1 5 12 19]		
	map	 <p>Confusion Matrix</p> <p>True label \ Predicted label: A, B, C, D, E</p> <p>Color scale: 0.0 to 0.5</p>		
mission 2	data	[1 3 7 10 0] [2 7 6 13 0] [0 3 19 16 1] [0 7 8 32 3] [0 1 3 17 4]		

	map			
mission 3	data	<pre> [15 6 0 0 0] [ 8 12 6 2 0] [ 0 5 26 8 0] [ 0 0 2 46 2] [ 0 0 0 9 16] </pre>		
	map			

Regression:

Table 5. Regression models performance

		trivial	1NN	Linear	kNN(15)	SVR
mission 1	RMSE	<b>2.840431</b>	<b>3.174862</b>	<b>2.416933</b>	2.647549	2.4635431
	MAE	<b>2.243742</b>	<b>2.386503</b>	<b>1.765720</b>	2.048261	1.866445
	R-square	<b>~0</b>	<b>-0.249353</b>	<b>0.275955</b>	0.131192	0.247760
mission 2	RMSE	<b>3.175514</b>	<b>3.270053</b>	<b>2.717781</b>	2.918296	2.781410
	MAE	<b>2.471288</b>	<b>2.496932</b>	<b>1.949769</b>	2.248261	2.063565
	R-square	<b>~0</b>	<b>-0.061185</b>	<b>0.266989</b>	0.154837	0.232264
mission 3	RMSE	<b>3.175514</b>	<b>1.356646</b>	<b>1.082780</b>	1.116182	1.132787
	MAE	<b>2.471288</b>	<b>0.944785</b>	<b>0.797725</b>	0.791820	0.731710
	R-square	<b>~0</b>	<b>0.817351</b>	<b>0.883651</b>	0.876362	0.872656

		trivial	1NN	Linear	ridge	lasso
mission 1	RMSE	2.840431	3.174862	2.416933	2.433797	2.538547
	MAE	2.243742	2.386503	1.765720	1.831852	1.935571
	R-square	~0	-0.249353	0.275955	0.265816	0.201258
mission 2	RMSE	3.175514	3.270053	2.717781	2.771181	2.807155
	MAE	2.471288	2.496932	1.949769	2.037844	2.142118
	R-square	~0	-0.061185	0.266989	0.237900	0.217986
mission 3	RMSE	3.175514	1.356646	1.082780	1.071758	0.956400
	MAE	2.471288	0.944785	0.797725	0.798337	0.693627
	R-square	~0	0.817351	0.883651	0.886007	0.909226

All of the above data are the results of the optimal training set predicted by the test set after 10 cross-validation(We didn't show the result on the cross-validation set because For this data set, its validation set and test set have the same universality. The result of the best validation set is often a little better than that of the test set, because there is less bias in the best validation set, not because of overfitting. If the test data from the validation set were also included in the results, it would make the results very difficult to read and would not prove that the model overfits.). The optimal training set's selection is based on the accuracy/MSE for classification/regression. The parameter of kNN is 5/15 for classification and regression. The parameter of ridge/lasso regression is default value(which is 1). The parameter of PCA is 20(which is the number of features we finally picked).

Based on the table above and compared with trivial and baseline models, we will pick the model with the best performance.

Classification:

For mission 1 & 2, we will choose PCA+Random Forest to be the final classification model.

For mission 3, we will choose SVM to be the final classification model.

The reason why PCA+Random Forest doesn't perform best but we still choose it to predict class in mission 1 is that the accuracy of this model is very close to the best one and it performs greatly better than other classification models in mission 2. Because the G1 and G3 scores are very similar features, we believe PCA+Random Forest is also a good model for mission 1.

Regression:



For mission 1 & 2, we will choose linear regression to be the best regression model.

For mission 3, we will choose lasso regression to be the best regression model.

All of the categorical features are turned into one-hot format. All of the features in this dataset are normalized by min-max normalization (except G1, G2, G3 score for Mission 3 because they are highly related and weigh a lot for the prediction).

## 4.2. Analysis

Classification:

For mission 1&2, we can find that the trivial system, which totally abandons the related feature and only focuses on G1/G3 and outputs based on prior probability, performs very terribly. It means students' scores are not randomly formed by prior probability. While if we take advantage of other features, we actually don't improve many which indicates that these 30 input attributes didn't contribute much to the final prediction. PCA+Random Forest performs better, maybe implies there are some useless or imbalanced features affecting results.

For mission 3, we can find that the accuracy of all models(except trivial) improves a lot. Apparently, G1 and G2 greatly impact the prediction accuracy. All the models except trivial and Bayes have close accuracy. SVM performs best possibly because the sample is small.

Regression:

For mission 1&2, we can find that all the models are beat by linear regression. It seems that G1/G3 has a linear relationship with those features. And even the best model, it doesn't improve a lot more than a trivial system, which further illustrates that the features of those 30 inputs contribute very little to the overall dataset.

For mission 3, we surprisingly find lasso regression performs best. Actually, lasso, ridge regression are some kind of linear regression with regularization. Lasso limits the value range of  $w$  to angular squares, while the Ridge limits the value range of  $w$  to circles. Tangent points of contour lines and square regions are more likely to be on the coordinate axis, whereas tangent points of contour lines and circular regions are less likely to be on the coordinate axis. This is why Lasso (L1 regularization) makes it easier to have partial weights set to 0, making the weights sparse.

Because lasso is easy to set partial weight to 0, feature selection can be used. The feature with a weight of 0 has no contribution to the regression problem, so the feature with a weight of 0 is directly removed, and the output value of the model remains unchanged. All in all, adding G1 and G2 and abandoning irrelevant features can make the model perform better.

## 5. Libraries used and the model what we coded ourselves

Here are the libraries we used:

Numpy, csv, pandas, math, random, matplotlib.pyplot, sklearn.preprocessing,

(Note)These are for model:

```
sklearn.tree,  
sklearn.svm,  
sklearn.neighbors.KNeighborsRegressor,  
sklearn.neighbors.KNeighborsClassifier,  
sklearn.naive_bayes.MultinomialNB, sklearn.decomposition.PCA,  
sklearn.ensemble.RandomForestClassifier,  
sklearn.linear_model.Ridge,  
sklearn.linear_model.Lasso,  
sklearn.linear_model.LinearRegression.
```

(Note)These are for examination:

```
sklearn.metrics.f1.score,  
sklearn.metrics.confusion_matrix,  
sklearn.metrics.mean_absolute_error,  
sklearn.r2_score,  
sklearn.metrics.mean_squared_error,
```

For pandas, except using it to read csv files, we also use `pd.get_dummies` and `pd.DataFrame` to do the one-hot data processing, so that we can consider more features and it may make the prediction more accurately.

Also, we code some function by ourselves:

For running cross-validation, we follow the process in the flowchart above. In the self defined function `shuffle_data`, we just shuffle the data once , choose the first 400 data as the training data and the rest 86 data as the validation data. When using the ML method to process the data, we use a for loop to shuffle the data 10 times and choose the best performance(High accuracy for classification or low MSE in regression) as our cross validation.

We also code the classification trivial system, baseline system and regression trivial system by ourselves.

## 6. Contributions of each team member

Jiazhi Dai's contribution:

- Design all the classification models;
- Mainly responsible for writing code;
- Join, discuss and modify the report.

Weiqliang Qian's contribution:

- Design all the regression models;
- Mainly responsible for writing report;

- Join, discuss and modify the code.

## 7. Summary and conclusions

Classification:

For mission 1 & 2, we will choose PCA+Random Forest to be the final classification model.

For mission 3, we will choose SVM to be the final classification model.

Regression:

For mission 1 & 2, we will choose linear regression to be the best regression model.

For mission 3, we will choose lasso regression to be the best regression model.

All the results showed in the above tables in part4.1.

In general, this project enabled us to learn and practice how to apply the model taught in class to predict multi-dimensional data sets. We finally found that it was not very accurate to directly predict scores with this data set without considering G1 and G2 scores when only a simple pretreatment was done. After the addition of G1 and G2, the accuracy of the whole data set was significantly improved, and the training data had the best effect after a certain degree of dimension extraction. In fact, if you simply want to predict data without considering interpretation, deep learning will further improve the accuracy of prediction. If there is still time, we will consider targeted pretreatment of G1 and G2. It is also expected that the raw data will be more sensitive to the dynamics of some characteristics, such as changes in parents' jobs between G1 and G2, which are also important factors affecting grades. At the same time, we will try to hide some data randomly to make the whole model more accurate to the Missing dataset.

## References

- [1] “Machine Learning F1-Score, Recall, Precision\_Matrix\_11blog\_fi Score.” *Blog.csdn.net*, 31 Dec. 2015, [blog.csdn.net/matrix\\_space/article/details/50384518](http://blog.csdn.net/matrix_space/article/details/50384518). Accessed 30 Apr. 2022.
- [2]chen. “Understanding and usage of R square.” 30 May 2020, [zhuanlan.zhihu.com/p/143132259](http://zhuanlan.zhihu.com/p/143132259). Accessed 30 Apr. 2022.
- [3]“Andrew Wu machine learning assignment-Linear Regression (Python) -CSDN blog.” *Blog.csdn.net*, 30 Nov. 2021, [blog.csdn.net/qq\\_48642405/article/details/121567148](http://blog.csdn.net/qq_48642405/article/details/121567148). Accessed 30 Apr. 2022.
- [4]Wu, Yuxiong. “Python Machine learning support vector machine nonlinear regression SVR model” *Www.zzvips.com*, 22 July 2021, [www.zzvips.com/article/178045.html](http://www.zzvips.com/article/178045.html). Accessed 30 Apr. 2022.
- [5]Adapting. “Ridge regression and Lasso regression.” 16 Aug. 2021, [zhuanlan.zhihu.com/p/400443773](http://zhuanlan.zhihu.com/p/400443773). Accessed 30 Apr. 2022.
- [6]“Machine learning-Random Forest -CSDN blog.” *Blog.csdn.net*, 6 Apr. 2022, [blog.csdn.net/beiye\\_/article/details/123831768](http://blog.csdn.net/beiye_/article/details/123831768).