Predicting Students' Mental Health using Collected Features from Smartphones and Methods Comparison

Group Name: Black Sheep

Group Members:

Xiaohan Zhu z5187021
Wentao Zou z5229938
Yaosong Huang z5222792
Junyu Ren z5195715

Introduction

In this project, we want to find out if it is possible to predict a person's mental status from their daily behaviors using machine learning methods. Moreover, we wish to compare the performance of different machine algorithms in solving this problem.

To accomplish this project, three different machine learning methods should be implemented to predict three different mental measures, Flourishing scores and PANAS scores including positive and negative aspects.

These two kinds of scores are student's self-evaluation in several mental aspects. Assuming it may be hard to predict the specific score, we separate scores into to two groups of "High" and "Low" based on the median number. Moreover, because the features are collected in a period for 10 weeks, we assume that these features are more likely to indicate the students' mental health change. So, we also calculate the difference between pre and post scores and want to predict the trend of psychological status change during the term, whether the student's mental status improved or not after a ten-week study.

Thus, to solve these classification problems, our group selects three different machine learning methods: 1) SVM, 2) Random Forest, 3) XGBoost. In the third part we will discuss them in detail.

Dataset

Input Data

The raw data is collected from students' smartphones for 10 weeks, which contain 10 different sensor data: physical activity, audio inferences, conversation inferences, Bluetooth scan, light sensor, GPS, phone charge, phone lock, Wi-Fi and Wi-Fi location .Each sensing feature record each student's behavior separately during the term.

1) Activity feature: Recording each student's activity type like stationary and moving every two or three seconds.
2) Audio feature: Monitoring students' microphone and making audio inference each timestamp.
3) Conversation feature: Recording each students' every conversation's start and end time.
4) Bluetooth feature: Recording connected Bluetooth's MAC address, class ID and signal frequency every ten minutes

5) Light feature: Recoding start and end time when the phone is in dark environment.
6) GPS feature: Recoding phones' geographical location (latitude and altitude) and the source of GPS coordinates.
7) Phone charge feature: The start time and end time of each phone charge.
8) Phone lock feature: Recording start and end time of a charge duration if it is more than one hour.
9) Wi-Fi feature: Including BBSID, wi-fi signal frequency and strength.
10) Wi-Fi location feature: Inside or near the building.

Output Data

The output data consists of Flourishing score and PANAS score:

The Flourishing scale consists of 8 self-evaluated statement scaled 1 to 7. A higher mark represents more positive emotion a student has.

PANAS scale has two components: **Positive** affect scores and **Negative** affect scores, all marked from 1 to 5. The sum of each affect indicates a person's level of positive or negative emotion. For each scale, every student attempted twice, one is before the term another is after the term. We assume that the sensing features may have different relevance to pre and post score. Thus, we treat pre and post scores in a same scale as different set. For classification, our group divides each score set into two classes: 'High' and 'Low' based on each set's median score. This could make each class in equal size.

Moreover, we assume that the sensing data may reveal more information about the difference between pre and post scores. Thus, for each mental measure, we calculate the difference between pre and post scores and same as above, binarizing them into two classes based on the median difference.

Output Datasets					
Flourishing	Pre-Score	PANAS Positive	Pre-Score	PANAS Negative	Pre-Score
	Post-Score		Post-Score		Post-Score
	Difference		Difference		Difference

Table1. Output datasets and size

Methods

Methods introduction

For solving classification problem, our group choose 1) Support Vector Machine, 2) Random Forest and 3) XGBoost. These three machine learning methods are common supervised learning methods. For svm, it can be applied to find a hyperplane by using kernel trick. RF and XGBoost both are ensemble method, but former is based on bagging and the latter is based on boosting. The difference in ensemble may cause different performance on these classifications.

SVM Description

SVM in skleaen has 4 kind of kernel models. Based on the characteristics of our dataset, "rbf" is the most suitable model because firstly, our classification problem cannot be classified by linear model. Moreover, in this project, the sample size is relatively small which has only 49 students. It is not too small or not too big.

Random Forest Description

For our dataset, the train data is small, and the features are complexity, so it is beneficial to use random forest to avoid overfitting. And random forest algorithm is a strong algorithm which based on many decision trees can provide a high accuracy and have more significant generalization performance than other model.

XGBoost Description

1) XGBoost is an algorithm based on GradientBoosting algorithm with advanced regularization to prevent overfitting.

2) XGBoost enumerates several candidates that may become the dividing points according to the percentile method, and then find the best dividing point from the candidate according to the above formula for calculating the dividing point.

3) Xgboost considers the case where the training data is sparse and can specify the default direction of the branch for the missing value or the specified value, which can greatly improve the efficiency of the algorithm.

Pre-processing

Firstly, we generate all features' total time according to the timestamps. However, the features were not collected every day, so the average data may imply more information about a student's mental states. In addition to the simply processing of the raw data, we want to gain more

information from the combination of raw data. For example, a student's sleep duration may be a very important to indicate a person's mental health. Thus, generating new features from original data is the next step. Here are some new features generated from raw data, for example:

1) Sleep duration: The sleep duration of person is the linear combination of 5 different raw features: Stationary feature, Light feature, Phone Usage features including phone lock and phone charging features, and Silence feature.
2) Daily Travel Distance: This feature indicates a person's daily mobility. According to altitude and latitude information from the GPS feature, the specific daily moving distance could be generated.

Feature extraction and Feature dimensional reduction

Overall, 50 different features are generated from the original sensing data. However, compared to the limited size of input, the number of features is overwhelming. This may make the instance space too spares and increase the training complexity. Hence, 1) we selected some of the important features that has more diversity and more relationship with students' mental status according to the StudentsLife report which means we drop out some redundant features. Then, the new feature size decreases to 15. 2) Feature Dimensionality-reduction: Sklearn-PCA was implemented to reduce the features' dimension. We are surprised to find that the dimension space of the features could be decreased to 3, and more importantly the variance ratio of the new is over 0.99 which means it can contain almost all information in the raw features:

variance_ratio of new Features after PCA		
New Feature1	New Feature2	New Feature3
0.79320551	0.14214903	0.06286136
Total: 0.9982		

Now, we have three different feature sets: 1) Raw feature sets including 50 generated features. 2) Simpler features after feature selection. 3) Dimension decreased features implemented PCA.

Feature sets	Size
Raw Features	50
Simpler Features	15
PCA Features	3

Thus, we wish to find the best feature set for each method.

Normalization

Before we start developing each model, we need the scale each feature in the feature set into the same scale. So, we use min-max scalar to pre-process each feature:$Xnew = \frac{X - \min(x)}{\max(x) - \min(x)}$

Method development

After feature processing, each mental measure has 3 different feature sets. We need to find the best feature set for each measure. Considering the number of instances in each set is limited, we decide to use K-fold cross validation in training our data, which could offset the inefficient data

K-fold cross validation

In this project, we do not set test set because the number of instances in a dataset is just about 40 (excluding output has missing value). Setting the test set will make the train set even smaller. Then, 5–fold cross validation is used in this project. This is also because of the limited set size, if we select small number such as 3 will make the train set too small, which will cause underfitting. The main metric to judge the model's performance is 'roc_auc'.

Steps of developing method

For each dataset, we wish to generate a best model for each method. For a given dataset, we need to compare models generated by each of the three features (Raw data, extracted data and PCA data). Then pick the best model as the method's model in this dataset.

1) Pick a feature set to be trained
2) Use 5-fold cross validation to tune hyperparameters to find the best model which has highest average AUC score.
3) Select rest of feature sets and train again.
4) Compare models generated by each feature and choose the best one.

The detailed tuning steps is discussed in the next section.

Evaluation metrics

After generating a best model for a given dataset, the best model should do cross-validation again to get metrics such as accuracy, recall and precision for the comparison of different methods. We not only focus on the model's overall accuracy but also want to compare their ability to correctly classify true positive instances and the percentage true positive instances can be classified.

Results

SVM

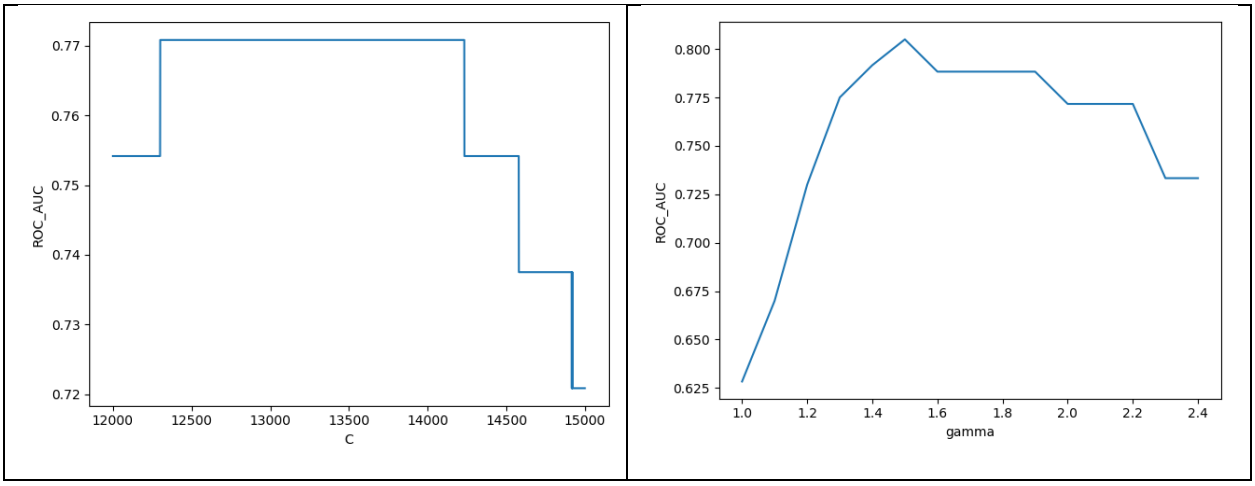
Hyper-Parameters explanation

SVM model have a collection of hyper-parameters. For solving this problem, we decided to mainly focus on 2 parameters: C and gamma. C means penalty coefficient, if C is bigger, the model's ability to tolerate misclassification is weaker. Gamma is a parameter that goes with kernel 'rbf'. It determines the distribution of data when they are projected to new feature spaces. When gamma is bigger, supporting vectors are less and vice versa.

Hyper-Parameter adjustment strategy

1.Initialize C range list: c_range_list = [0.00001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]
Initialize gamma range list: gamma_range_list = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]

2. Finding the best c, gamma in these lists using cross validation, and set cv = 5.
3. Construct a range (i.e. best value-50, best value+50) based on best values in step 2.
4. Observe the result of best parameter pairs (usually more than 1), if the AUC score strictly raises in current range, range boundary should be increased to see what happen. And if the AUC score is strictly going down in current range, range boundary should be decrease. If the current range contains the best parameter value, this value is fixed.
5. When one of C or gamma is fixed, the same as step4, dynamically adjust the range to retrieve the best value of another parameter.



Best Parameters & metrics for 9 datasets

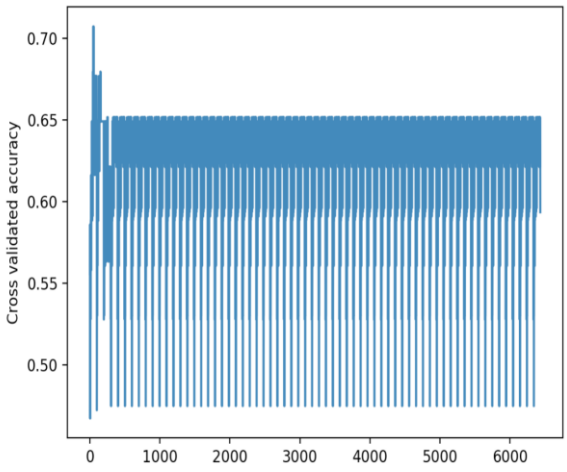
Model SVM kernel='rbf'						
Dataset	C	gamma	precision	recall	accuracy	auc
Flourishingpost	8500	0.07	0.62	0.68	0.66	0.76
Flourishingpre	10500	0.35	0.64	0.67	0.67	0.76
Flourishingdiffer	8300	0.015	0.67	0.63	0.74	0.78
PanasPositivepost	0.05	0.78	0.67	0.62	0.65	0.76
PanasPositivepre	1.0	6166	0.63	0.62	0.64	0.67
PanasPositivediffer	22000	76	0.58	0.57	0.63	0.68
PanasNegativepost	40000	22	0.62	0.58	0.60	0.71
PanasNegativepre	0.0005	2116	0.55	0.56	0.52	0.59
PanasNegativediffer	36955	0.11	0.56	0.73	0.58	0.69

Random Forest

Hyper-parameter adjustment selection

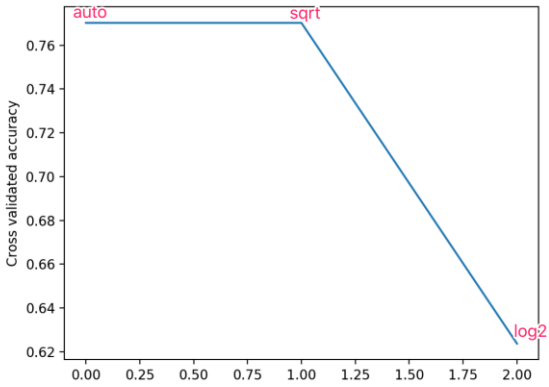
N_estimators: This is number of trees in forest. The result of a tree and a hundred trees must be very different, so it a very important parameter for our model. Our team choose the number of n_estimators from 1 to 100, some we set it from 1 to 200.

max_depth: This is also a very important parameter. The graph on the right side illustrates the combinations of different max depth and different number of estimators can provide a high roc_auc score, but there are many repetitions. It is shows that we can get a not bad score when the depth is shallow. So, our group choose the max_depth from 1 to10, for some test we choose it from 1 to 5.

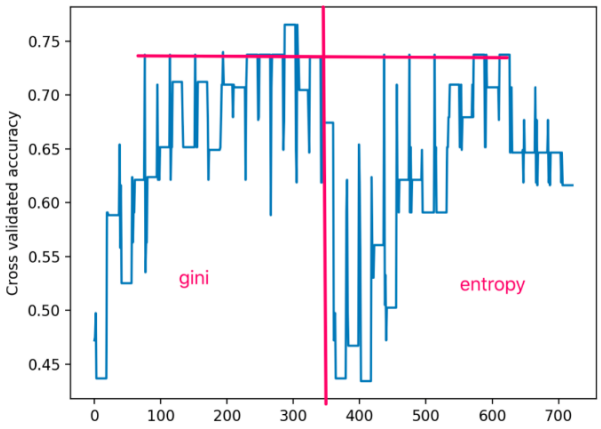


min_samples_split: This is the minimum number of samples required to split an internal node. However, our sample data is not big, and this parameter is very important for big sample data do not have much impact on our sample data, so our team just set default for this parameter.

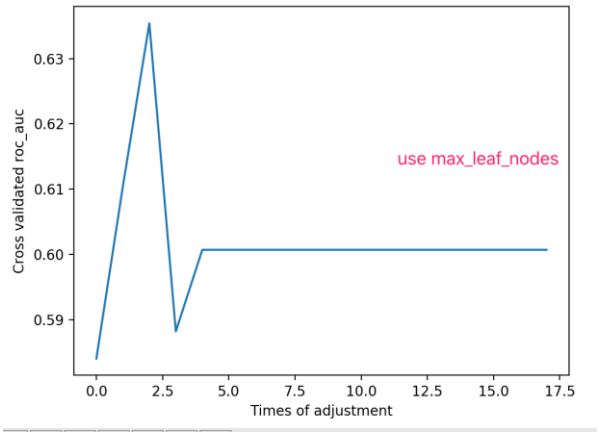
max_feature: This parameter set the number of features to be considered. when looking for the best split, there are three choices for this parameter: auto, sqrt and log2. From the figure, the performance of ‘log2’ is the worst, and ‘auto’ and ‘sqrt’ have the same level of performance

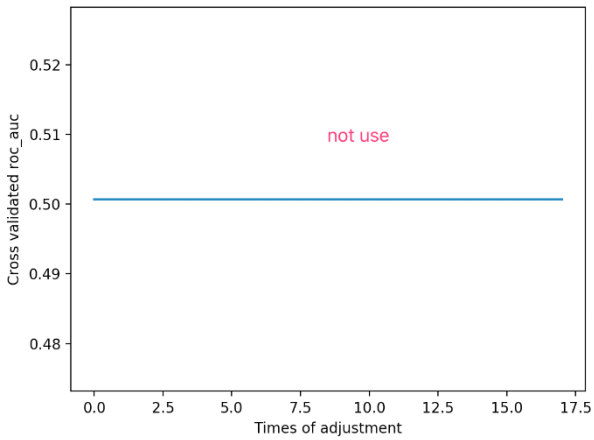


criterion: This hyper-parameter has two choices: “gini” and “entropy”, which determine the tree’s split. Our team test it on different situations and get result as below. It is obvious that the average accuracy in ‘gini’ is higher than ‘entropy’. Thus, ‘gini’ should be chosen.



max_leaf_nodes: Grow trees with max_leaf_nodes in best-first fashion. For most our sample data if we use it, we can have a good performance, as below showing,





We can see that if we use max_leaf_nodes for some points we can get a better performance than not use. So, from so many tests, we choose the max_leaf_nodes from 2 to 6.

min_weight_fraction_leaf: This is the minimum weighted fraction of the sum of weights (of all the input samples) required to be at a leaf node. For many tests, our team find 0.1 is the best figure for this parameter.

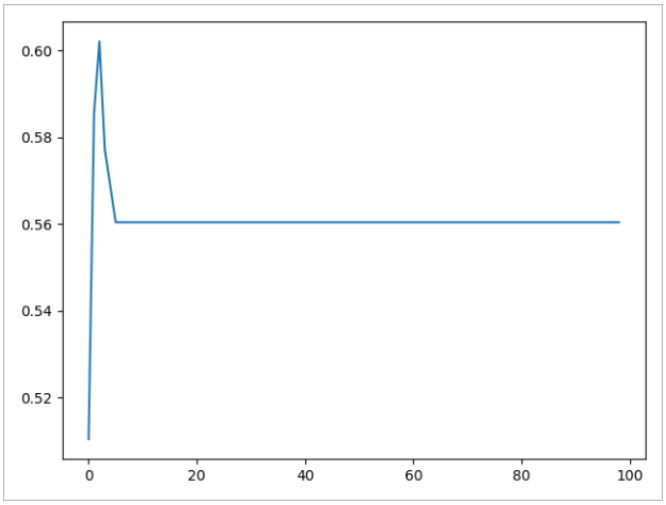
Best Parameters & metrics for 9 datasets

Random Forest							
Dataset	max_depth	N_estimators	Max_lesf_nodes	precision	recall	accuracy	Roc_auc
Flourishingpost	1	145	2	0.66	0.63	0.614	0.63
Flourishingpre	2	16	3	0.58	0.61	0.615	0.622
Flourishingdiffer	1	19	2	0.64	0.58	0.67	0.69
PanasPositivepost	1	72	2	0.68	0.57	0.58	0.80
PanasPositivepre	2	136	3	0.695	0.65	0.57	0.668
PanasPositive differ	3	1	4	0.48	0.63	0.50	0.39
PanasNegativepost	2	4	4	0.42	0.85	0.51	0.53
PanasNegativepre	1	7	2	0.45	0.35	0.56	0.5
PanasNegativediffer	1	139	2	0.64	0.54	0.67	0.61

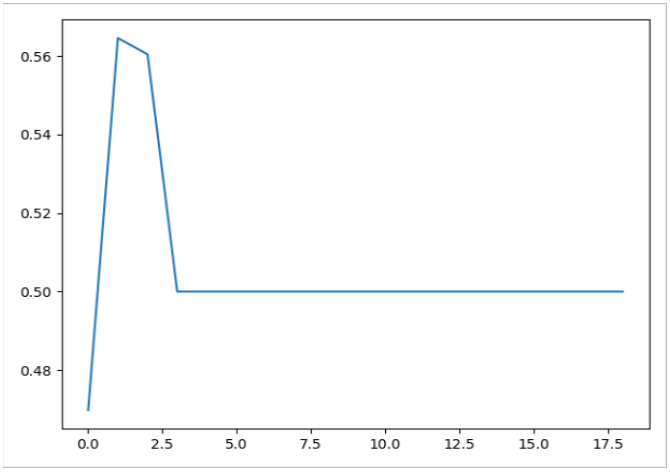
XGBoost

Hyper-parameter adjustment selection

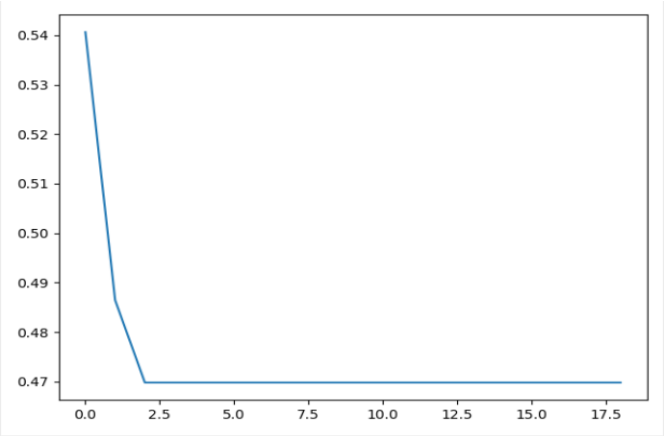
n_estimators: The graph on the right side shows the cross validated accuracy decreased with the value of K for n_estimators increasing. Thus, choosing a better performed range (0, 25).



min_child_weight: This parameter significantly affects the result, which means the smaller the value of the parameter, the easier it is to overfitting. The larger the value, the more conservative. Thus, range (0, 7) could be a suitable range.



Max_depth: This parameter indicates the maximum depth of the tree. The greater the depth of the tree, the higher the fit to the data (the higher the degree of overfitting). Thus, we get a suitable range (0, 2.5)



Best Parameters & metrics for 9 datasets

XGBoost							
	max_depth	min_child_weight	n_estimators	precision	recal	accuray	roc_auc
FlourishingPost	2	0	10	0.8	0.666	0.7143	0.6431
FlourishingPre	1	1	11	0.7	0.666	0.595	0.6229
FlourishingDiffer	3	0	10	0.67	0.533	0.6655	0.6433
PanasNegativePost	1	0	3	0.75	0.4	0.675	0.6313
PanasNegativePre	2	0	7	0.444	0.47	0.52	0.3235
PanasNegativeDiffer	3	1	4	0.5333	0.45	0.475	0.5271
PanasPositivePost	1	1	17	0.4876	0.6	0.5333	0.6382
PanasPositivePre	3	1	11	0.5383	0.61	0.57	0.628
PanasPositiveDiffer	2	2	1	0.95	0.583	0.7857	0.7146

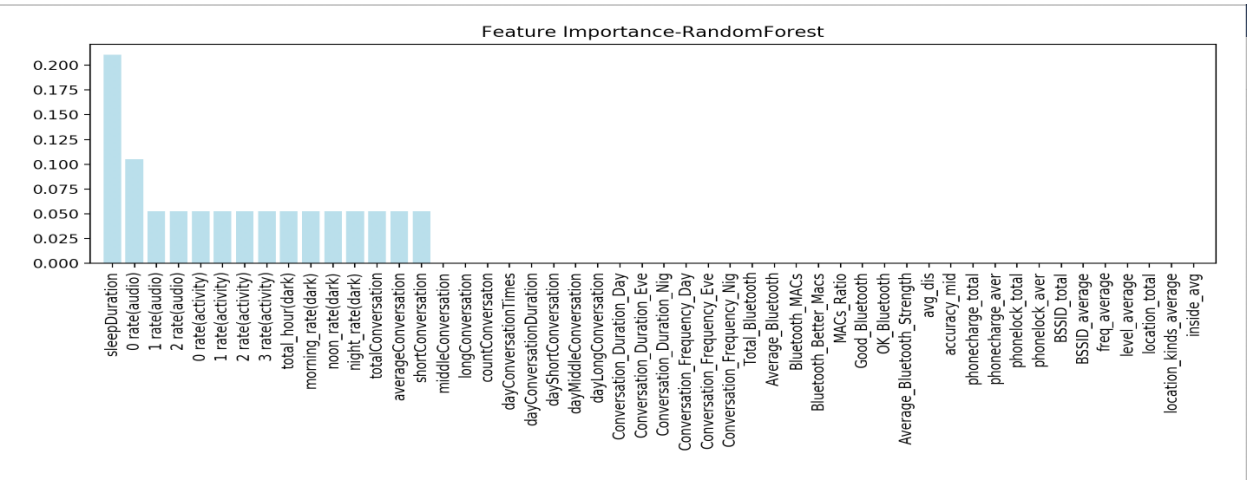
Feature sets comparison

When a given model was trained on a mental dataset, we tried three different features set as former section mentioned. Obviously, we should compare the three different models generated by the different feature sets and select one which has the best classification ability. Here we take the Flourising Difference as the example to explain the performances using different features.

	Raw Feature	**Extracted Feature**	**PCA Feature**
SVM	0.78	0.687	0.77
Random Forest	0.69	0.67	0.55
XGBoost	0.6808	0.7233	0.6692

From the table, SVM's classification performance remain in a relatively high level in both raw and PCA dataset compared to the extracted feature. However, the situation is opposite if Random Forest is used. The dimensional decreased feature just has 0.55 AUC score. This may be because the feature dimension may influence the Random Forest's performance. XGBoost has the highest metric when it is implemented in the extracted feature.

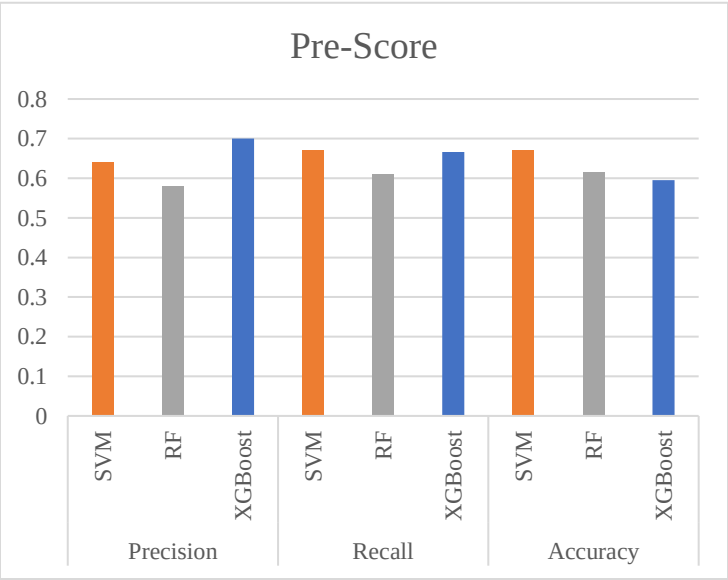
Feature Importance



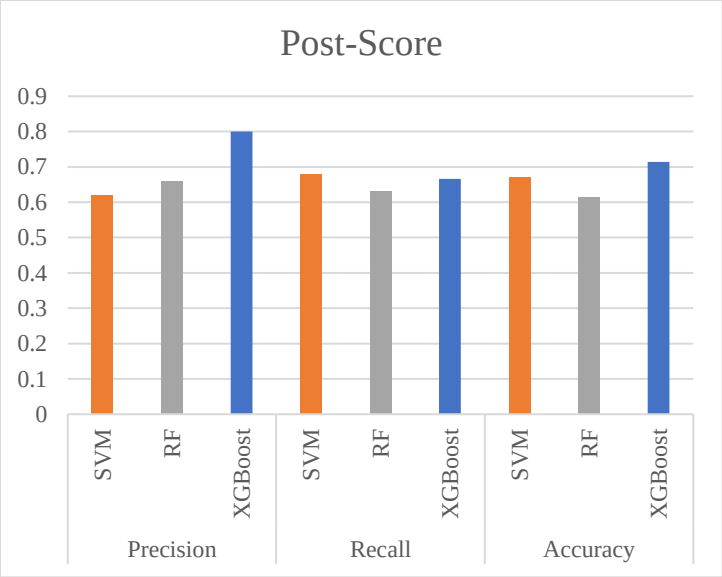
We use Random Forest to evaluate each feature's importance. The figure above indicates that most of features in the raw feature set are not helpful to the model development. And the top 15 important features are extracted to generated as the new feature set which could not affect the model performance and decrease the training time.

Discussion

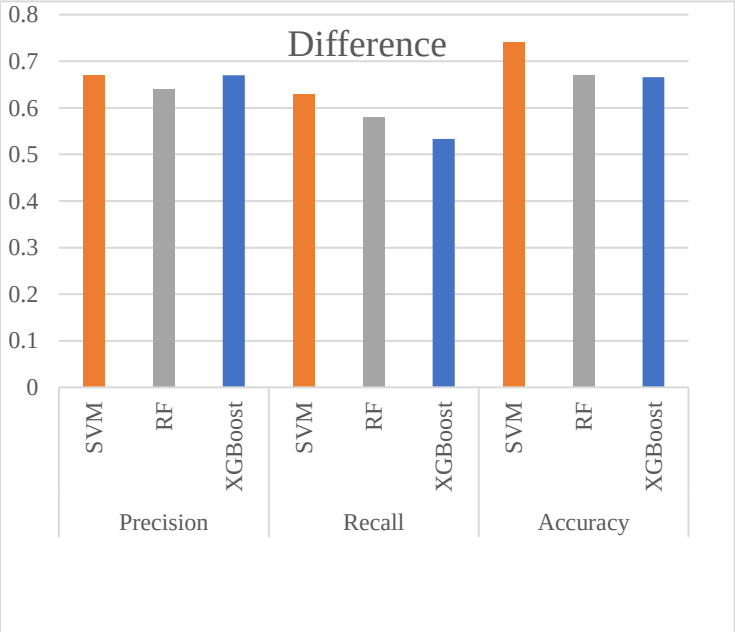
Flourishing



For predicting Flourishing pre-score, SVM methods has the highest accuracy (0.67) while XGBoost comes last. However, the XGBoost model has the highest precision score whereas Random Forest cannot correctly classify 'High' score students compared to the other two method

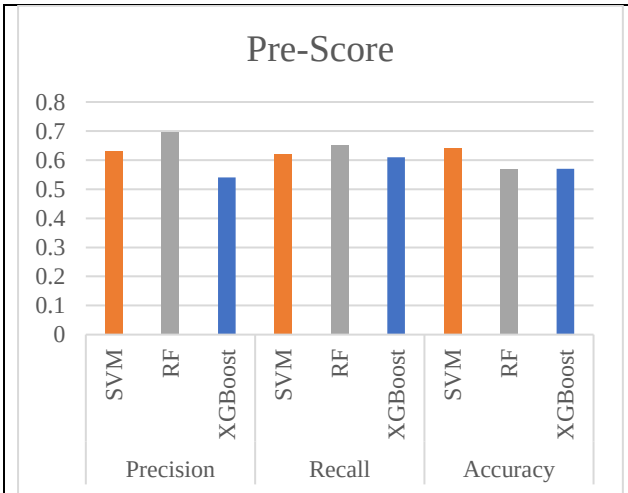
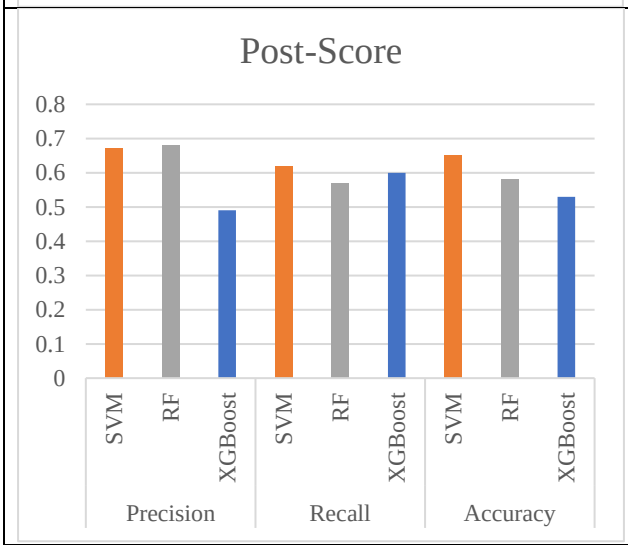
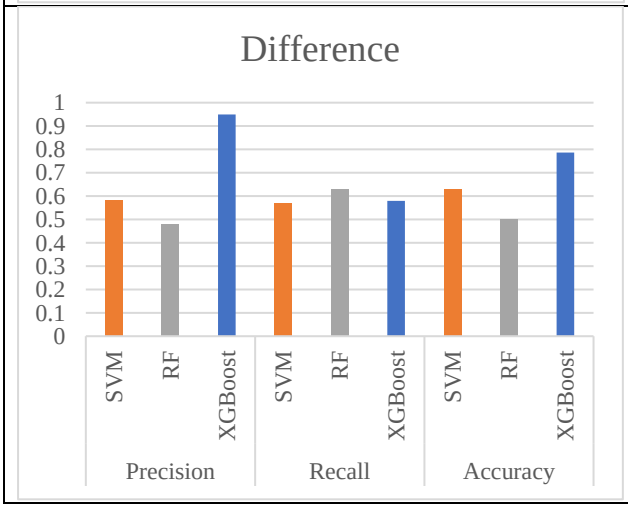


The model trained by XGBoost has the best performance in predicting post-score in Flourishing dataset. Meanwhile, this model also has the highest precision score which is about 0.8. Random Forest has least accuracy but is still over 0.6.

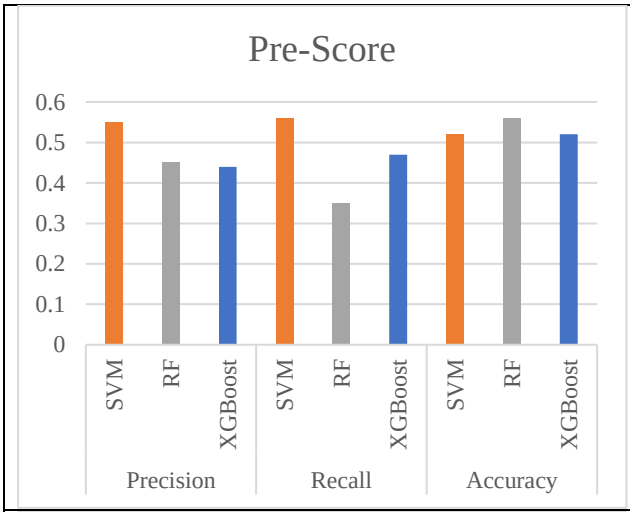
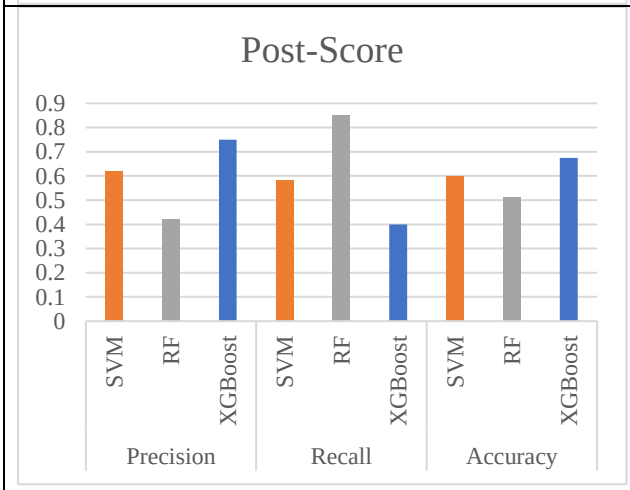
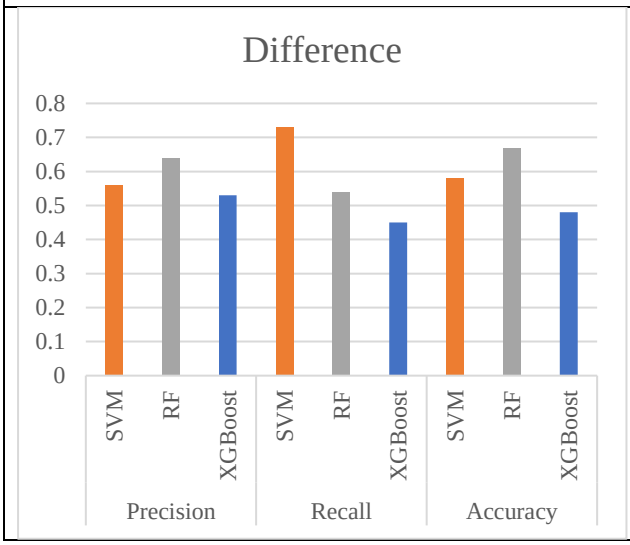


The SVM method is the best method to predict the difference between the pre score and post score in Flourishing dataset since it has the highest scores in all three metrics. The other two method's performance is almost same. Random Forest is better at correctly identifying positive class instances while the precision rate of RF is the least.

PANAS_Positive

	The SVM models is the best choice to classify the pre-score in Positive data set. Compared with the other two methods which both have 0.56 accuracy. The Random Forest has the highest recall and precision rate in this classification.
	Same as the pre-score, model generated by SVM performs best while Random Forest model still remain around 0.56 accuracy and XBGBoost performs slightly better than the random classification. Moreover, XGBoost's precision rate could not reach 0.5 which means the 'High' class predicted by the model is almost misclassification.
	However, in score difference classification, the XGBoost's accuracy score is almost 0.8 which is significantly higher than the other two models. Besides that, its' precision rate is 0.95 which indicates that the XGBoost is the best method to classify 'High' class in score difference among the three methods.

PANAS_Negative

	For classifying pre-score in PANAS_Negative dataset, all these three models' performance are not very ideal. RF model has the highest accuracy (0.56) while the other two both have just 0.52. However, the recall rate of the RF is just 0.35 which indicates that its ability to correctly classify 'High' class is not very ideal.
	The accuracy of model generated by Random Forest is only 0.51, however, it can correctly identify most of the positive class instances (0.85 recall rate). This means that the Random Forest model tends to classify an instance as a 'High' score class. The SVM model performs best in this classification, which has the highest accuracy and the highest precision rate.
	In this dataset, the Random Forest has highest accuracy (0.68) whereas the XGBoost's performance is even worse than the random classification. According to the recall rate, the SVM model could correctly identify 72% positive class instances. Thus, in classifying score difference in Negative dataset, the XGBoost is worst one.

Comparison and Findings

Overall, the SVM method has the highest average accuracy which is about 0.63. SVM method comes next with 0.61 accuracy. The Random Forest has the least average accuracy (0.58). To be more specific, SVM is good at classifying pre and score difference in Flourishing scale, pre and post score in PANAS_Positive scale. XGBoost is best method in score difference classification in both Flourishing and PANANS_Positive set. Random Forest only excels in classifying score difference in Negative scale.

Flourishing			PANAS_Positive			PANAS_Negative		
Pre	Post	Difference	Pre	Post	Difference	Pre	Post	Difference
0.626	0.662	0.69	0.593	0.587	0.64	0.53	0.595	0.575

The figure above illustrates each dataset's average accuracy. It is obvious that the Flourishing measure's accuracy is slightly higher than the other two sets. This means that the sensing data is more relevant to students' Flourishing measures. In addition, the score difference has the highest score among all score scale. Thus, the sensing feature could indicate more information of students' mental change during the 10 weeks.

Conclusion

In this project, we generate and extracted features from the raw sensing data to predict 49 students in 3 different mental aspects: Flourishing, PANAS_Positive and PANAS_Negative. To solve the problem, we implement 3 different machine learning methods: SVM, Random Forest and XGBoost to generate best models for each dataset.

1) Feature: There are 15 features that significantly influence the model performance. So we extract the top 15 importance features into a new feature set for training.
2) Methods: The XGBoost has the highest average accuracy followed by SVM. The random forest is not very ideal in this project, most models' accuracy is below 60. The XGBoost has best performance in classifying score difference, especially in PANAS_Postive dataset where the model's accuracy is around 0.8.
3) Mental measures: Flourishing measure has the highest average accuracy which means that the sensing data is more relevant to students' Flourishing measures. Moverover the score difference has the highest score among all score scale. Thus, the sensing feature could indicate more information of students' mental change during the 10 weeks.

In conclusion, the sensing data could predict students' mental status to some extends.

Reference

1. Z. Chen, M. Lin, F. Chen, N. D. Lane, G. Cardone, R. Wang,T. Li, Y. Chen, T. Choudhury, and A. T. Campbell. Unobtrusivesleep monitoring using smartphones. In *Proc. of PervasiveHealth*, 2013.
2. Wang, Rui & Chen, Fanglin & Chen, Zhenyu & Li, Tianxing & Harari, Gabriella & Tignor, Stefanie & Zhou, Xia & Ben-Zeev, Dror & Campbell, Andrew. (2014). StudentLife: Assessing mental health, academic performance and behavioral trends of college students using smartphones. UbiComp 2014 - Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing. 10.1145/2632048.2632054.
3. Zeng, & Tan, & Matsunaga, Tsuneo & Shirai,. (2019). Generalization of Parameter Selection of SVM and LS-SVM for Regression. Machine Learning and Knowledge Extraction. 1. 745-755. 10.3390/make1020043.
4. Liu, C. & Chamberlain, Benjamin & Little, Duncan & Cardoso, Ângelo. (2017). Generalising Random Forest Parameter Optimisation to Include Stability and Cost. 10.1007/978-3-319-71273-4_9.
5. Bentéjac, Candice & Csörgő, Anna & Martínez-Muñoz, Gonzalo. (2019). A Comparative Analysis of XGBoost.
6. Forsyth, David. (2019). SVMs and Random Forests. 10.1007/978-3-030-18114-7_2.