COMP9417 Group Project of Assessing Student Life Model (2019 T3)

Group name : Crazy Fire

Group member : Ce Song (z5221101)

Wei Song (z5198433)

Longcheng Zhao (z5227264)

Yinghong Zhong (z5233608)

1 Introduction

In this project, we would focus on a paper, StudentLife[1] which study the mental health, academic performance and behavior via a smart phone app and we would use different machine learning methods to train models to classify the mental health of a student in negative PANAS, positive PANAS and Flourishing score.

Considering the small dataset of our samples and the complexity of our models, we analyzed the details of the paper, input and output dataset and finally extracted 14 features from the 10 raw input datasets to implement our solution.

2 Dataset

2.1 Output Dataset

The object of our project is to predict the flourishing score, negative PANAS and positive PANAS scores.

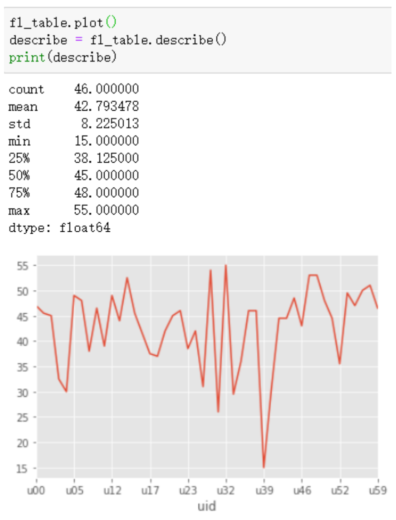
The first task is to pre-process the data and fill out the missing data. After analyzing the raw data, we find that over 80% of students have pre and post scores of the questionnaires which were done at the start and end of term respectively and most of the scores of the same questions in these two questionnaires don’t have significant changes, so we decide to use the average of pre and post scores as the score of each question. After getting the score of each question, we sum them up as the final score of each student.

Due to the small amount of missing data of our output dataset, we have two filling rules for these missing data:

1) if a student had done both pre and post survey but there are some missing data in his dataset, we would fill the missing data with the existing data of that question.
2) if a student just finished one survey and there are some missing data in his dataset, we would fill the missing data with the average value of the whole dataset of this student.

The second task is to set a threshold for each output dataset to divide the scores into two classes of “High” and “Low”. We use the median as the threshold instead of the mean because median is robust against outliers, whereas the mean is sensitive to outliers. What’s more, using median as the threshold can make the data set balanced.

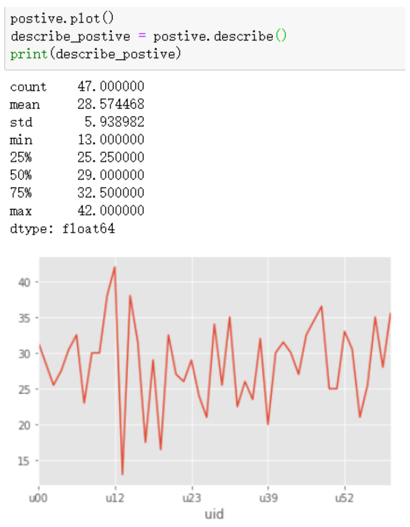
As for flourishing dataset, we only have 46 samples in flourishing.csv as our output dataset. From data description(Figure 1), we can find that the median is 45(50%) and the mean is 43 (37%) which means that over 50% students have a score over 45 ,so we choose median 45 as the threshold, meaning that we would have 2 classes and if the score is larger than 45, it belongs to the “High” class, otherwise it belongs to the “Low” class.



```
fl_table.plot()
describe = fl_table.describe()
print(describe)
```

```
count    46.000000
mean     42.793478
std       8.225013
min      15.000000
25%      38.125000
50%      45.000000
75%      48.000000
max      55.000000
dtype: float64
```

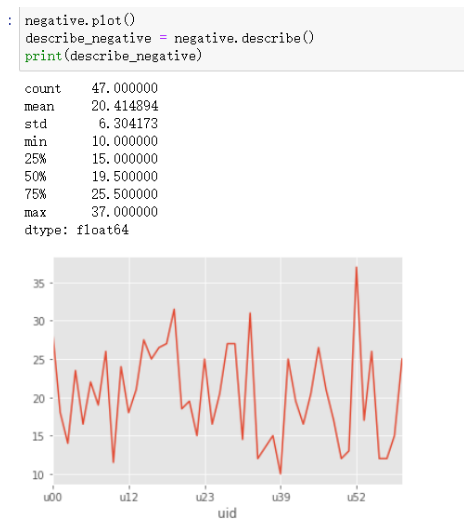
Figure 1. Flouring data description



```
postive.plot()
describe_postive = postive.describe()
print(describe_postive)
```

```
count    47.000000
mean     28.574468
std       5.938982
min      13.000000
25%      25.250000
50%      29.000000
75%      32.500000
max      42.000000
dtype: float64
```

Figure 2. Positive data description



```
negative.plot()
describe_negative = negative.describe()
print(describe_negative)
```

```
count    47.000000
mean     20.414894
std       6.304173
min      10.000000
25%      15.000000
50%      19.500000
75%      25.500000
max      37.000000
dtype: float64
```

Figure 3. Negative data description

As for PANAS, we have 47 samples in panas.csv and then we collect the data following the instructions of PANAS_Scoring.pdf to form 2 sections of data, including positive questions and negative questions. Following same classification method as flourishing data, we choose the median 29 and 21 as the threshold of positive dataset (Figure 2) and negative dataset (Figure 3) respectively. Table 1 is the detail of our output classes.

Table 1. Labels Classification

Label **Class**	**High**	**Low**
Flourishing	≥ 45	< 45
Positive	≥ 29	< 29
Negative	≥ 21	< 21

3 Methods

3.1 Preprocessing input data

There are 10 features provided in this project and these features have influence on four aspects, such as activity, conversation, sleep and location, of a student's life. Figure 4 shows the architecture of the StudentLife app[1]. Hence, we would focus on these four behavioral classifiers to extract and group the feature following their relationship analyzed in the StudentLife study.

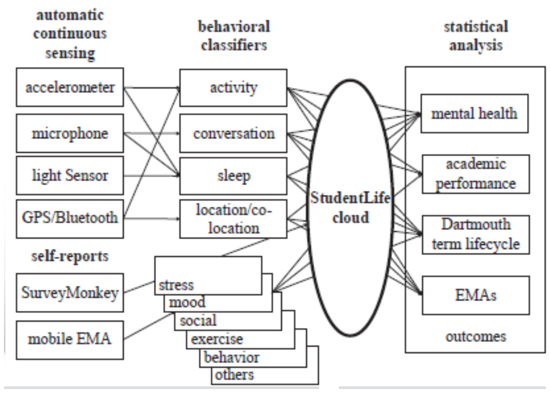


Figure 4. StudentLife app, sensing and analytics system architecture

1) Sleep Features

The StudentLife sleep classifier extracts four types of features: dark features, phone usage features including the phone lock state, stationary from activity features, and sound features from the microphone. According to the paper [1], we learn that any of these features alone is a weak classifier for sleep duration because there are several phone usage patterns having influence on sleeping, so we would combine these features to form a sleeping duration feature.

Firstly, we collect the average dark time, phone lock time, phone charging time and stationary time per day of each student as the raw data. Then we use a linear model in the paper, which is affected by 4 factors: Sleeping Duration = $\sum_{i=1}^{n} \alpha i. \mathrm{Fi}$, where α_i is the weight of the corresponding factor.

We learn the model in Unobtrusive sleep monitoring using smartphones[2] and finally select the coefficients of the features as Table 2.

Feature	Coefficient
Light (F_1)	0.0415
Phone-lock (F_2)	0.0512
Phone-off (F_3)	0.0000
Phone-charging (F_4)	0.0469
Stationary (F_5)	0.5445
Silence (F_6)	0.3484

Table 2. Coefficient of Sleeping Model

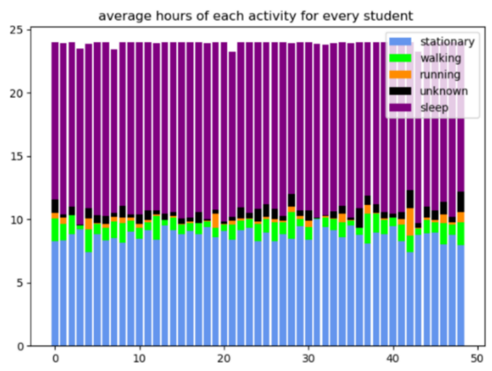


Figure 5. Average time of activities/day of student

2) Activity Features

From StudentLife[1] , we learn that activity classifier extracts features from the preprocessed accelerometer stream and in the activity dataset, we can separate the data into 3 major different activities, including walking which is a measure of indoor mobility, running which represents outdoor mobility and stationary. However, when we talk about average stationary time of a student per day, we find that it includes sleeping time, so we would use pure stationary time as the stationary time which would subtracts the sleeping time from raw stationary time.

In Figure 5, we would find that stationary, running, walking and sleeping make up 24 hours of the day which indicates that all these activities should have some impact on students' life, so we would use the average time of running, walking and pure stationary per day as our features.

3) Conversation Features

In StudentLife, there are two classifiers which are audio and conversation classifier to do the conversation detection. However, from the StudentLife paper[1], we learn that the output of audio classifier is treated as the input of a conversation classifier, so in our model, we would only use the data from conversation dataset.

We consider the conversation frequency and duration of conversations per day as a measure of sociability which would affect the mental health of a student. Figure 6 shows the average time of conversation duration and average times of conversation per day for every student.

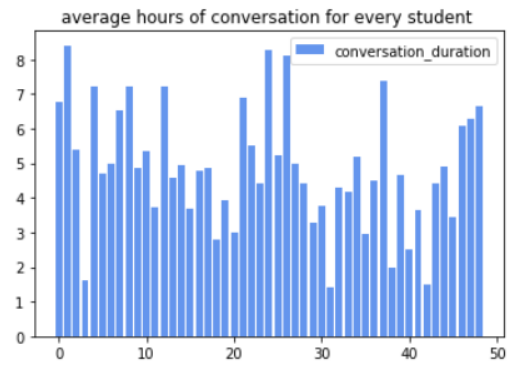


Figure 6(a). Conversation Duration / day

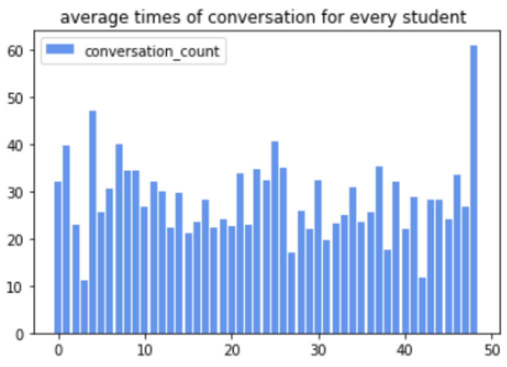


Figure 6(b). Conversation times / day

4) Location/ Co-location Features

In terms of location/co-location, we have four input datasets, including Bluetooth, GPS, Wi-fi and Wi-Fi location.

From Result Section of StudentLife[1], we find that there is a significant correlation between flourishing scale and Bluetooth encounters (number of co-loacations) per day which indicates the sociability of a student. Hence, we calculate the average Bluetooth encounters per day of each student as one of the features.

According to the activity detection of StudentLife, it says that the overall mobility of a student consist of indoor mobility which is computed as the distance a student travels inside the building using WIFI and outdoor mobility which is computed as the distance a student travels around the campus using GPS. We assume that outdoor travel distance indicates that a student would have more social activities which would affect their mental health, so we compute the outdoor mobility (a.k.a. traveled distance) as a feature by using latitude, longitude and altitude in gps.csv. Figure 7 is the average travel distance around the campus for each student.

What's more, considering duration staying in different places like dorm, study places, restaurants, etc., may have relationship with a student's mental health, we calculate the average hours of staying in dorm, study places and other places (restaurants, entertainment) using data in wifi_location.csv. Firstly, we filter the places where students stay in, then check the places in Google Map and give them tags --- dorm, study and other. Finally, we can group the data by different places and find three features about average hours of staying in different places. Figure 8 shows average hours of staying in different places.

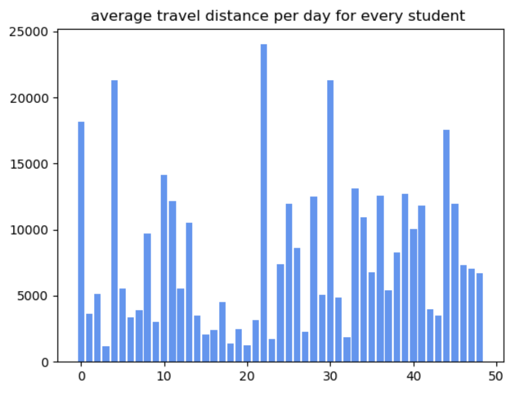


Figure 7

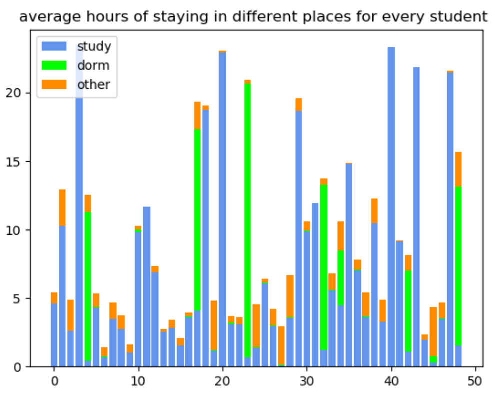


Figure 8

Table 3 shows the initial features we extract from the raw input data.

Table 3 Initial features

NO.	**Feature Class**	**Feature Name**	**Unit**	**Note**
1	Activity	Stationary Time	average hours/day	Extracted from activity dataset and exclude the sleeping time
2		Running Time	average hours/day	Extracted from activity dataset
3		Walking Time	average hours/day	
4	Sleep	Sleeping Time	average hours/day	Calculated with dark, phone lock, phone charge and stationary time
5		Dark Environment Time	average hours/day	Used to calculate sleeping time Extracted from dark dataset
6		Phonecharge Time	average hours/day	Used to calculate sleeping time Extracted from phonecharge dataset
7		Phonelock Time	average hours/day	Used to calculate sleeping time Extracted from phonelock dataset
8	Conversation	Conversation Duration	average hours/day	Extracted from conversation dataset
9		Conversation Frequency	average count/day	

10	Location	Encounter Device Count	average count/day	Extracted from Bluetooth dataset
11		Travel distance	average km/day	Extracted from GPS dataset
12		Dorm Duration	average hours/day	Extracted from WifiLocation dataset
13		Study Duration	average hours/day	
14		Other Duration	average hours/day	

3.2 Evaluation metrics

In our solution, we decide to implement three models including SVM, Random Forest and KNN to do the classification. We would use ROC-AUC score and accuracy score as the metrics to evaluate performance of our models.

Accuracy which shows the percentage of the correct classifications with respect to all samples is a simple metric to evaluate model performance, but it doesn’t show the performance of positive and negative classes. As for ROC-AUC score, it applies to binary classifiers that have some notion of a decision threshold internally and it is helpful to have a good view of the overall performance of the classifier, measuring sensitivity and specificity for our 2 classes, so ROC-AUC score would be a more important metric in our solution.

3.3 Model 1 --- Support Vector Machine (SVM)

Support Vector Classification (SVM) is one popular type of classifiers whose classification boundary can be learned by a support vector machine maximizes the margin.

The reason why we choose SVM is that it is effective in high dimensional spaces and can still have reasonable effect in cases where number of dimensions is greater than the number of samples. In our case, we only have about 50 samples and 14 features, so SVM is suitable for this dataset. Also, it can use different Kernel functions to do prediction, which is helpful for us because we do not know which kind of model (linear or non-linear) is suitable for our dataset.

We use the Support Vector Classifier from scikit-learn library of Python to implement our SVM model and have normalized our data before training the model. Also, GridSearchCV module from scikit-learn library is used to help us to find the best hyper-parameters. For this model, we need to find the best kernel function and penalty parameter. The hyper-parameters setting is as follows:

For predicting flourishing score, kernel is set to 'sigmoid', penalty parameter is set to ‘5’. For predicting PANAS negative score, kernel is set to 'linear', penalty parameter is set to ‘5’. For predicting PANAS positive score, kernel is set to 'rbf', penalty parameter is set to ‘1’.

After finding the best parameters and configuring our model, we do a 5-fold cross validation to get criteria of each fold.

In order to train better models, we then use ‘SelectFromModel’ method from scikit-learn library to select the features for each label. We use L2-Regularization to select features instead of L1-Regularization because L2-SVM is differentiable and imposes a bigger (quadratic vs. linear) loss for points which violate the margin, according to Yichuan Tang, 2013[3].

3.4 Model 2 --- Random Forest (RF)

Random forest is a supervised learning algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forest creates several decision trees on randomly selected data samples and features, gets prediction from each tree and then selects the best solution by means of voting. It also provides a pretty a good indicator of the feature importance.

In our case, we totally have around 50 samples and 14 features, random forest may help us to solve overfitting problem comparing with single decision trees because random samples and features would be selected and it is the initial reason that we choose random forest as one of our models.

In our RF model, we directly use the random forest classifier model from scikit-learn and we need to focus on five parameters including the number of trees, the maximum depth for a tree, minimum samples to split, minimum samples of leaf nodes and maximum features.

We use GridSearch to tune these five parameters step by step. Firstly, via GridSearch we find the optimal number of trees is 60 and then we find that the optimal maximum depth with the optimal number of trees 60 is 3. Next, we find that the optimal minimum samples to split and optimal minimum samples of leaf nodes is 3 and 2 based on the optimal parameters from first two steps. At last, we tune the maximum features and find the optimal value is 8. After tuning hyperparameters, we can build our random forest classifier.

Before we implement our random forest classifier, we divide the 80% of original data set as train data set and left 20% of original data set as test data set. We use training set to train our random forest classifier and test data set to test it.

3.5 Model 3 K- Nearest Neighbors (KNN)

KNN is a lazy learner, which simply stores instances of the training set instead of constructing an internal model like other methods, so it does not need the training stage. As for KNN classifier, the prediction of test set is computed from majority vote of the K nearest training neighbors, which means the test data is assigned the most representatives data class.

In this case, our data set is small as well as balanced with 50% positive label and 50% negative label, so KNN could be a suitable model for out data set and it is easy to implement.

In our KNN model, we use the KNN classifier model from scikit-learn library and we select Euclidean distance which is the most commonly used distance function to calculate the distance when we choose the nearest neighbors. The reason we choose Euclidean distance function is that the addition of new point (e.g. outlier) does not have an influence on the distance between two points and it could transform the object with dimensions into similar scales to input standard data into module[4]. In addition, from previous experience the Euclidean distance function performs reasonably well over the categorical data set. We obtain the best hyper-parameters through the Gridsearch method, for example K, which is the number of nearest neighbors, and then the use these parameters to build our KNN module.

In order to avoid overfitting in our data set, we use 5-fold as cross validation which evenly divides the whole data set into several parts and iteratively test each part while using the rest part as the training set.

4 Results

4.1 SVM model

We do a 5-fold cross validation on our dataset and calculate the average score for AUC-ROC and accuracy for both training set and test set.

Table 4 shows the features we select for SVM and Figure 9 shows the results of our three targets.

Table 4 Feature Selection for SVM

Feature Name	Flourishing score	PANAS negative	PANAS positive
Stationary Time			√
Running Time	√		
Walking Time	√		√
Sleeping Time			
Dark Environment Time		√	√
Phonecharge Time	√		√
Phonelock Time	√	√	
Conversation Duration	√		
Conversation Frequency			
Encounter Device Count	√		√
Travel distance	√	√	√
Dorm Duration	√		√
Study Duration	√	√	
Other Duration	√	√	√

In Figure 9.2, we can see that we achieve about 0.78 on both test set and training set for flourishing score which means that our model for flourishing label is not overfitting. As for PANAS negative score, our model gets about 0.81 and 0.78 on AUC-ROC and accuracy score separately on the training set, and about 0.72 of AUC-ROC and 0.61 of accuracy on the test set. About PANAS positive score, we achieved about 0.96 AUC-ROC and 0.9 accuracy on training set and 0.7 AUC-ROC and 0.68 accuracy on test set.

Comparing all-feature model (Figure 9.1) and some-feature model(Figure 9.2) above, we can observe that overall ROC-AUC and accuracy score of all target labels increase after selecting features. It is also worth noting that before selecting features, for flourishing and PANAS negative score, the metrics of the training set are significantly higher than that of the test set, the reason of this may be the overfitting of our model. Selecting features could alleviate this problem so the model would have a better performance.

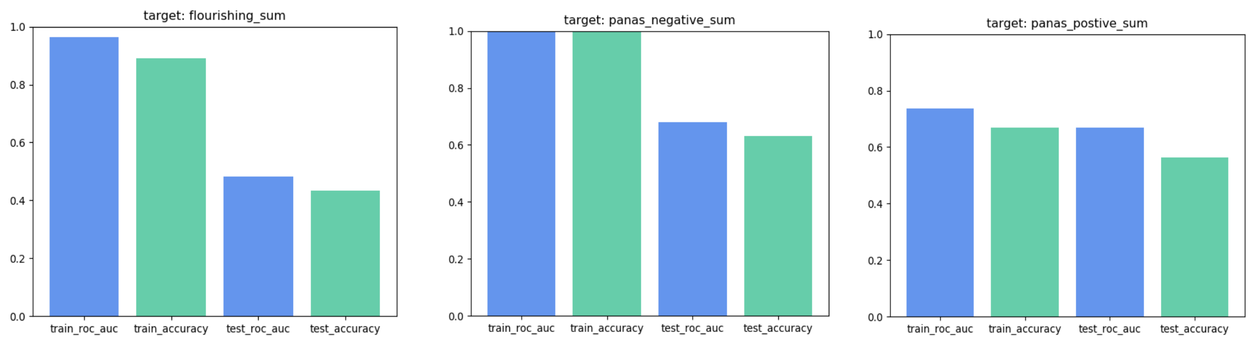


Figure 9.1 Metrics value before selecting features

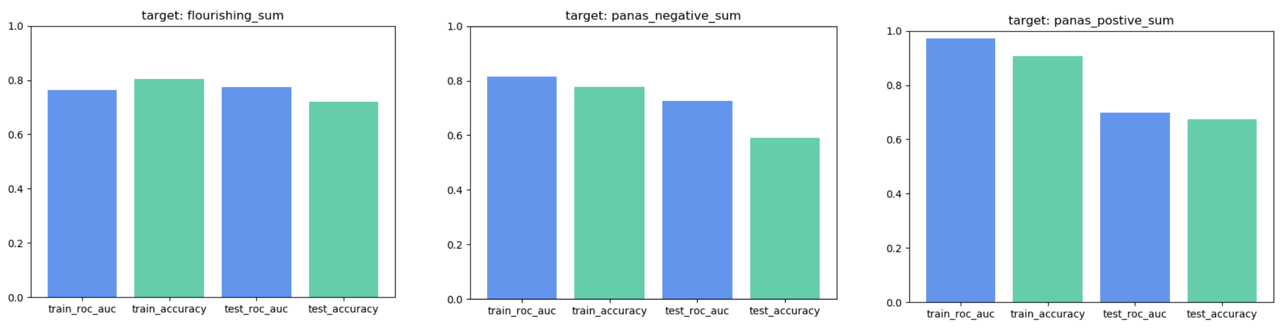


Figure 9.2 Metrics value after selecting feature

To sum up, for flourishing score and PANAS negative score, the SVM model has a similar train and test score and can achieve a significantly better score on the test set. We can infer that this model has a good generalization capacity and can have a decent accuracy on predicting those two labels. However, it performs slightly worse on the PANAS positive label.

4.2 Random Forest model

In random forest classifier, feature importance is based on impurity which is related to Gini impurity/information gain (entropy), so when training a tree, we can compute how much each feature contributes to decreasing the weighted impurity. Figure 10 displays the feature importance of our RF model for each label.

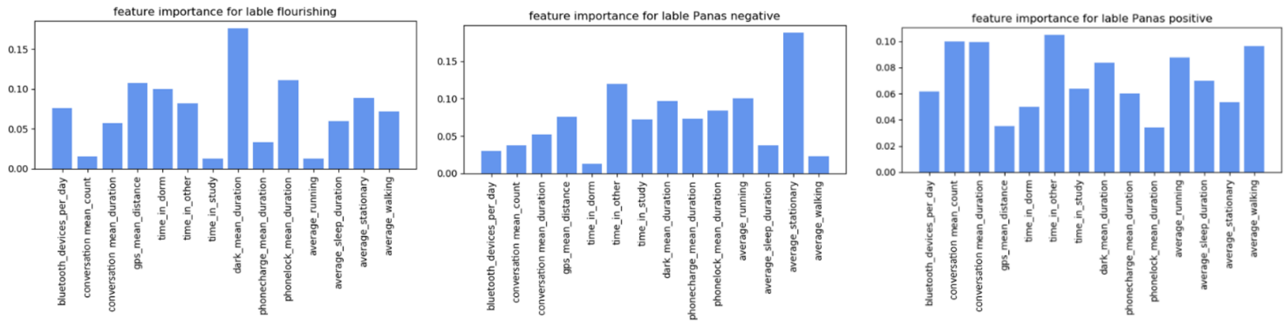


Figure10   Feature importance of each label (RF model)

According to the optimal maximum features which is 8 selected via GridSearch, we know that our RF classifier would choose the top 8 important features at most for each label.

Figure 11 shows the average ROC-AUC score and accuracy for the training set and test set using cross validation with 5-folds.

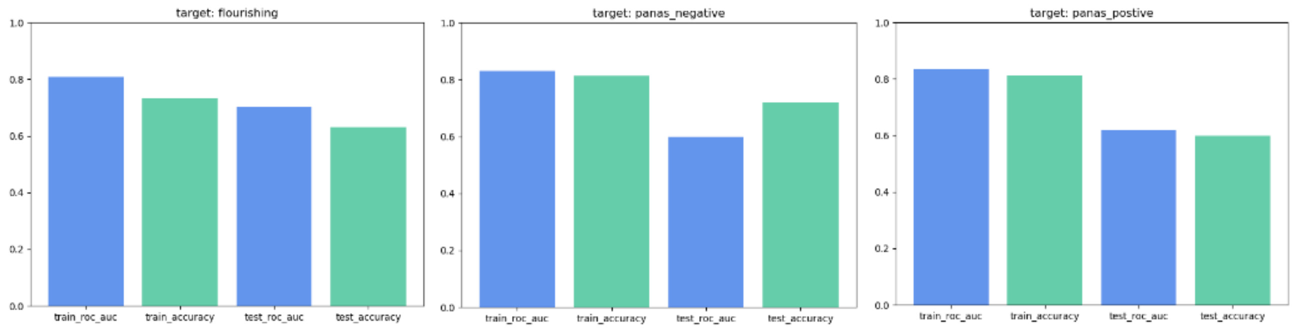


Figure 11   Metrics value for each label (RF model)

From Figure 11, we can observe that AUC-ROC score of training set is over 0.8 while the ROC-AUC score of test set is smaller which is around 0.72. Similarly, the training set accuracy is larger than the test set, which are around 0.72 and 0.68 respectively. Therefore, we can infer that this model is good.

However, this model performs not so good in negative_Panas score --- ROC-AUC score and accuracy of test set are quite different from metric scores in training set, which means that the generalization ability of our RF model is not so good.

When we use this model to fit the label positive_Panas, we can see it performs very well in training set with the 0.82 AUC-ROC score and 0.8 accuracy, but both ROC-AUC score and accuracy of test set are not very high which are only around 0.62. Thus, our random forest classifier performs best at label flourishing.

Overall, random forest performs good with our dataset (14 features and around 50 samples). The accuracy score is general without a high overfitting.

4.3 K Nearest Neighbors model

In this study, we test a range of k values (1 to 7) for choosing the optimal parameter k of the KNN classifier. Figure 12 shows the average test ROC-AUC score with different k values for 3 labels and we find the best k which have highest test ROC-AUC score for flourishing score, positive_Panas and negative_Panas is 1, 4 and 2 respectively.

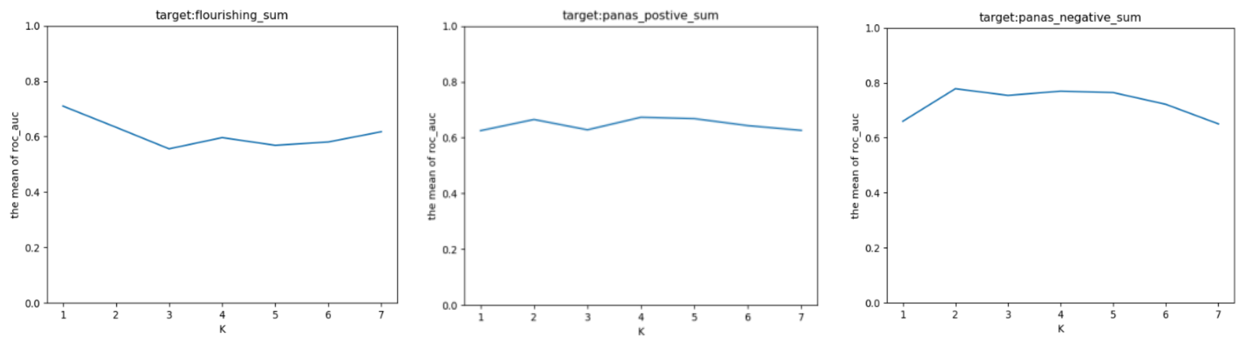


Figure 12 ROC-AUC score with k value parameter (1 to 7) for each label (KNN model)

Figure 13 shows the average ROC-AUC score and accuracy for the training set and test set using cross validation with 5-folds.

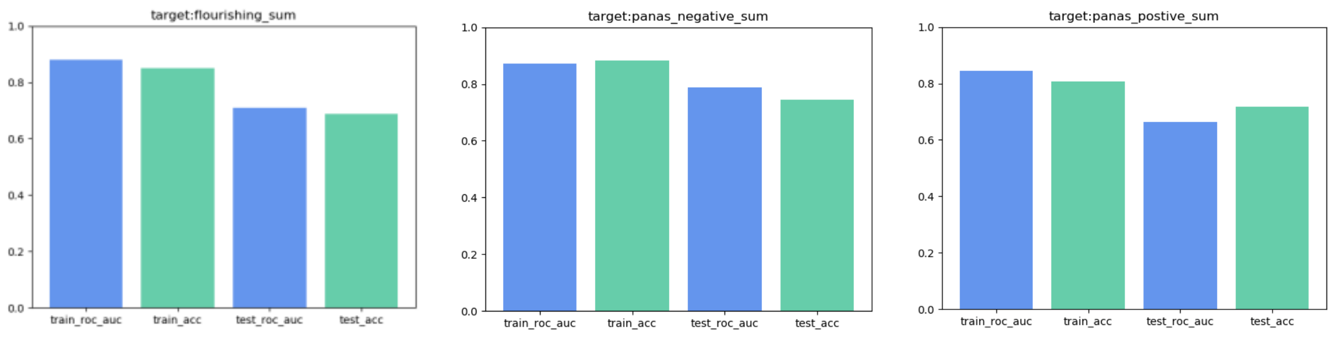


Figure 13 Metrics value for each label (KNN model)

For predicting flourishing class, the ROC-AUC score and accuracy of test set are about 0.7 and approximately 20 percent higher than the baseline.

As for PANAS positive and negative class, both ROC-AUC score and accuracy of test set is a bit lower than the training set with a gap about 0.12.

Generally speaking, our KNN model for all of 3 labels are relatively successful because metrics are generally higher than 0.7. Besides, there is only a small gap of the metrics which is about 0.12 between training set and test set metrics and it indicates that our KNN model handle overfitting problem well.

5 Discussion

5.1 The Performance of Different Classifiers

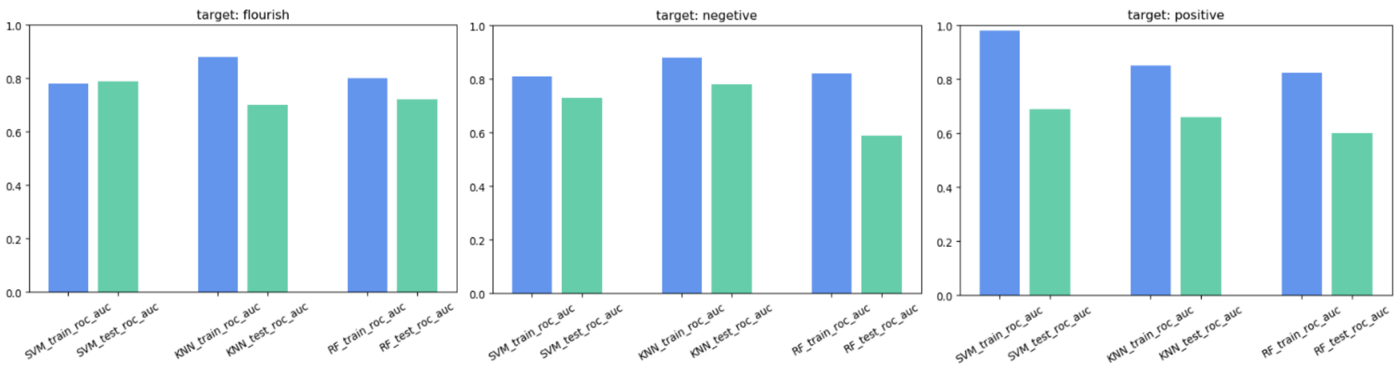


Figure 14 ROC-AUC scores of 3 models for each label

As shown in Figure 14, with average ROC-AUC score of 5-fold cross validation, we find that the best ROC-AUC score of test set for flourishing score and positive_Panas score are about 0.8 and 0.68 in SVM models respectively which are slightly better than RF and KNN. As for negative_Panas score, KNN has the best performance with the highest ROC-AUC score of test set (0.8) and Random Forest has the worst performance with 0.6 as the ROC-AUC score.

In flourishing score and positive_Panas score, SVM has the slightly better performance which is 0.8 and 0.7, than RF and KNN in the test set. What’s more, SVM has a smaller gap of ROC-AUC score between training set and test set in both flourishing score(0.1) and negative_Panas score(0.6), indicating that SVM has better generalization capabilities and more stable than the KNN and Random Forest.

However, in positive_Panas score, we find that all of three models have higher difference which are 0.28, 0.2 and 0.23 respectively between training set and test set and the average

ROC-AUC score is only 0.63 which is not very high. It is probably because our features do not have a strong relationship with the positive_Panas score.

In conclusion, SVM is slightly more stable and accurate than the other two models in our solution according to the result.

5.2 Advantages and disadvantages of various methods

In our project, our target is a binary classifier with a small and balanced dataset which means that 50% of the training set is classified as 1 and another 50% is regarded as 0.

As for SVM classifier, it has the best performance. SVM has good generalization capabilities which prevent it from overfitting and it is stable which means that a small change of data does not have a huge influence on the hyperplane. What's more, since the kernel implicitly contains a non-linear transformation which can handle non-linear data efficiently, we don't need to transform data linearly separable in our model. Actually, selecting an appropriate kernel function is difficult if we have a high-dimensional feature, but because of the small dataset, we can easily find the best kernel for each model.

As for Random Forest, it is considered as a highly accurate and robust method because it is based on bagging algorithm and uses Ensemble Learning technique, creating as many trees on the subsets of data and combines the output of all trees. Since Random Forest would choose features and samples randomly, it is also a good method to handle overfitting. In our RF model, we get the most important features to train our model with the help of RF classifier which selects the most contributing features. However, Random Forest is not the best model for our solution, because RF is a good model to handle a large dataset but not a small dataset.

As for KNN, it is a non-parametric and lazy classification model which doesn't learn anything in training period. And it is easy to implement, because there are only two parameters to tune for the best model, which is value of K and the distance function. However, there are only 49 samples in our project which is a small dataset, so the best K is about 2 or 3 for our models which means that the model would choose 2 or 3 nearest neighbors each time and it is easy to be overfitting.

Conclusion

To sum up, possibly due to the limited quantity of samples, we did not get an exceptional prediction model on this data set. However, we still managed to get about 0.8, 0.78 and 0.68 of maximal ROC-AUC score on flourish, PANAS negative and positive label, using SVM, KNN and SVM model separately. Also, the SVM model is considered as the most suitable model on this data set, which achieves about 0.8, 0.73 and 0.68 of the ROC-AUC score when predicting flourish, negative and positive score separately on the test set. It performs stably and tends to not overfit the data.

References

[1] Rui Wang, Fanglin Chen, Zhenyu Chen. StudentLife: Assessing Mental Health, Academic Performance and Behavioral Trends of College Students using Smartphones, 2014

[2] Z. Chen, M. Lin, F. Chen, N. D. Lane, G. Cardone, R. Wang, T. Li, Y. Chen, T. Choudhury, and A. T. Campbell. Unobtrusive sleep monitoring using smartphones. In Proc. Of PervasiveHealth, 2013.

[3] Yichuan Tang. Deep Learning using Linear Support Vector Machines, eprint arXiv: 1306.0239 (2013)

[4] Dibya Jyoti Bora, Dr. Anil Kumar Gupta.Effect of Different Distance Measures on the Performance of K-Means Algorithm: An Experimental Study in Matlab, arXiv: 1405.7471 (2014)