Human Activity Recognition Using Smartphones: Feature Engineering Techniques and Stacking/Ensemble Models for Accurate Classification

Name:	**Priyanshu Kumar Rai**
Registration No./Roll No.:	20215
Institute/University Name:	IISER Bhopal
Program/Stream:	Electrical Engineering and Computer Science
Problem Release date:	January 12, 2023
Date of Submission:	April 16, 2023

Introduction

The project aims to develop a model that accurately recognizes human activity based on data collected from smartphones. The dataset contains 562 different features and 8239 data points. The test date contains 2049 points for which the class labels are to be predicted. In this project, we will be using different feature engineering techniques and classification models to accurately classify human activity into one of the six classes. The percentage of frequency of the different classes in the training dataset is shown in figure 1. We have explored two different feature engineering techiniques namely- **Variance Thresholding** and **Maximum Relevance - Minimum Redundancy** (mRMR) [1] [2]. The project explored the use of sklearn ML models like decison tree, random forest, multilayer perceptron, KNN, SVM and stacking/ensemble classifier [3].

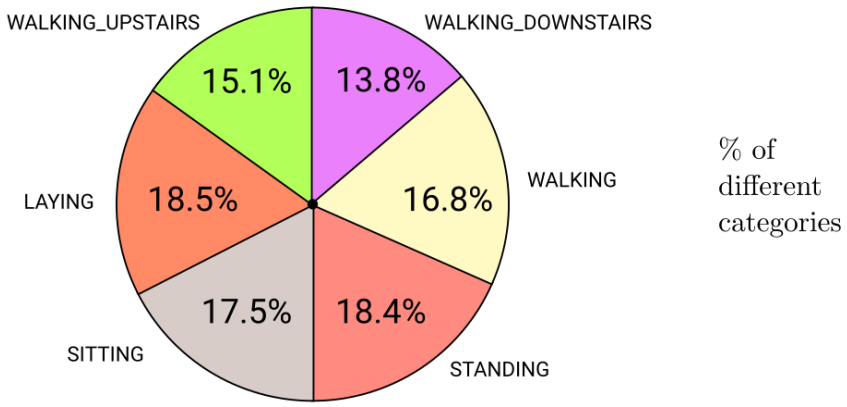


Figure 1: Pie chart for showing the percentage of different activities in the training data.

Feature Engineering Techniques Explored

Low-variance thresholding will eliminate features below a particular variance. To determine the best parameters for the two techniques, we recorded and plotted the macro-averaged F1-scores of the classifiers as mentioned versus variance threshold for the former as shown in figure 2 and macro-averaged F1-score against best K features obtained from the mRMR algorithm as shown in figure 3 .This simulation was performed on untuned classifiers. The classifiers were later tuned after the feature engineering techniques were implemented.

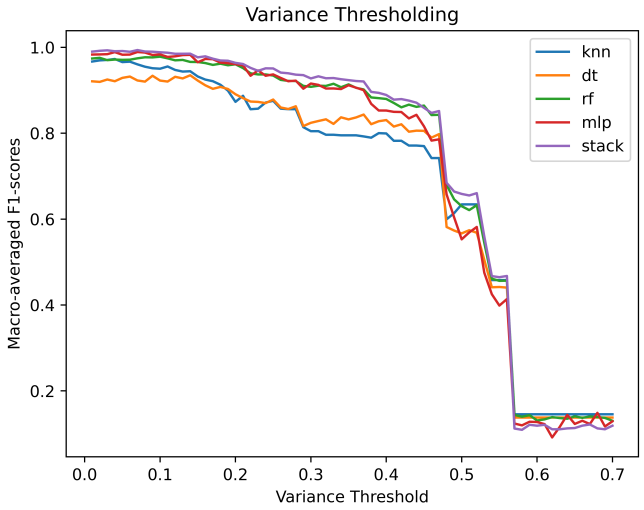


Figure 2: Plot for F1 score vs variance threshold

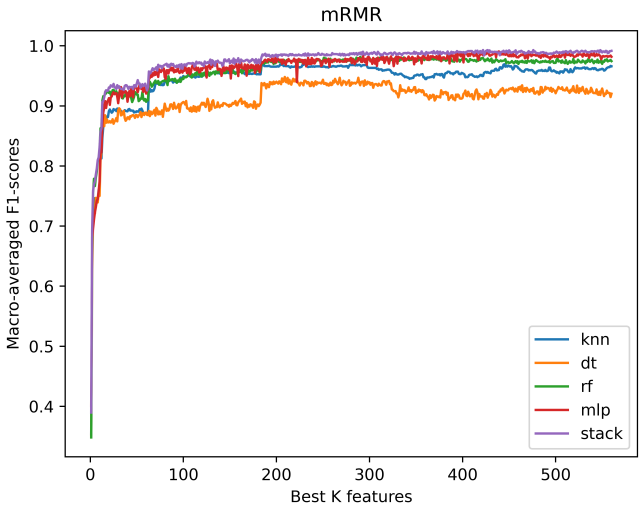


Figure 3: Plot for F1 score vs the best K features for mRMR

From the recorded data we study that variance thresholding gave the best F1-score of at variance **0.07** for the stacking classifier while mRMR gave 0.99292 for **422** as the best K features. We performed this simulation on untuned classifier models. Therefore the proposed framework will use variance threshold of 0.07 and best K features as 422 for mRMR technique.

Methods

We will use two frameworks with the feature engineering techniques as mentioned above. Variance Thresholding removes all features whose variance does not meet some threshold. We set 0.07 as the variance threshold, and the number of features we obtained is 280. The pseudocode for this technique is mentioned below:

variance_thresholder(variance): `// This function performs variance thresholding on the data and remove features having variance less than 0.07.`

```
vt = VarianceThreshold(threshold=variance) // VarianceThreshold object
vt.fit(X) // Obtain variance values for all the feature columns
Obtain a vector of the features selected through variance threshold and store
Take a subset of the dataset using the above selected features
print(f'variance_thresholder() returned X with shape {X.shape}')
```

The mRMR is a feature selection method that eliminates features by evaluating the mutual information to get the value of relevance and redundancy. We choose 422 best features for this technique. The pseudocode for this technique is mentioned below:

mrmr(best_n): `// best n features are passed as a parameter`

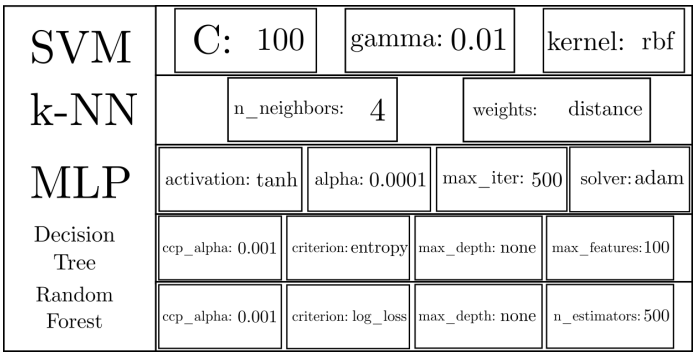
```
selected_features = mrmr_classif(X, y, best_n) // Selected features are obtained
X = X[X.columns.intersection(selected_features)] // Intersection of the feature
    matrix is taken with the selected features to get the reduced dataset.
```

The stacking/ensemble classfier would be used to predict the class labels of the test data. The stack would include k-NN, SVM, Random Forest, Decision Tree, MLP and Logistic Regression as the final estimator. We carry out hyperparameter tuning for the classifiers before stacking to get the most optimized parameters for the individual classifiers using GridSearchCV using macro-averaged F1-score as the metric. Stacked generalization consists in stacking the output of individual estimator and use a classifier to compute the final prediction.

```
estimator_list = [('mlp', mlp), ('svm_rbf', svm_rbf), ('rf', rf), ('knn', knn)]
stack_model = StackingClassifier(estimators=estimator_list, final_estimator=
    LogisticRegression(solver='liblinear'))
```

We split the training data into training and validation set using 8:2 as the split ratio in a stratified fashion and train the model using the best features obtained from variance thresholding and mRMR

using the best parameters we got from the hyperparameter tuning. We will use the feature engineering technique for which we get the best F1-score for predicting the class labels of the test data. The detailed overview of the method, inferences can be viewed on my GITHUB here.
The tuned parameters using GridSearchCV for the classifiers are listed below.



Experimental Analysis

Macro-averaged precision, recall, F-measure and Matthew's Correlation Coefficient (MCC) was used to determine the performance of the different classifiers using the feature engineering techniques to evaluate the best performing framework to predict the class labels. The results of the classifiers by using variance thresholding, mRMR feature engineering techniques on the validation dataset are given in table 1 and table 2 respectively. The results of the classifiers without using any feature engineering technique is given in table 3.

Table 1: Performance Of Different Classifiers Using Variance Thresholding

Classifier	Precision	Recall	F-measure	MCC-Score
Decision Tree	0.93964	0.93955	0.93951	0.92998
Random Forest	0.97845	0.97853	0.97844	0.977448
k-NN	0.96599	0.96582	0.96586	0.95623
SVM	0.98933	0.98923	0.98927	0.98614
MLP	0.99025	0.99001	0.99006	0.98763
Stack	0.99430	0.99423	0.99425	0.99271

Table 2: Performance Of Different Classifiers Using mRMR

Classifier	Precision	Recall	F-measure	MCC-Score
Decision Tree	0.92751	0.92771	0.92758	0.91610
Random Forest	0.97677	0.97705	0.97688	0.97302
k-NN	0.96684	0.96557	0.96593	0.95706
SVM	0.98461	0.98449	0.98452	0.98031
MLP	0.98963	0.98972	0.98965	0.98688
Stack	0.99374	0.99365	0.99369	0.99198

Table 3: Performance Of Different Classifiers without using Feature Engineering

Classifier	Precision	Recall	F-measure	MCC-Score
Decision Tree	0.92425	0.92409	0.92410	0.91100
Random Forest	0.97379	0.97398	0.97385	0.96937
k-NN	0.96956	0.96899	0.96916	0.96064
SVM	0.99321	0.99304	0.99312	0.99124
MLP	0.98894	0.98899	0.98896	0.98614
Stack	0.99489	0.99475	0.99482	0.99343

The F1 score of the framework which used variance thresholding and the one which did not use any feature engineering technique are nearly same but the time taken by the framework using variance thresholding was less than the one that did not use any feature engineering technique. The framework which used mRMR took more time than the one without feature engineering. This is because the mRMR technique used some time to select the best 422 features accurately. As anticipated the stack-/ensemble performed the best among all the classifiers which are used for predicting the class labels of the test data. Therefore, from the experimental results we can conclude that the variance threshold feature engineering proved to be a better method for this problem than the mRMR in terms of both the F1 score and time taken for running compared to the model without using any feature engineering techniques. The simulations were conducted in AMD Ryzen 5 3500U and Intel Core i5-1135G7 processors.

The framework with variance thresholding as the feature engineering technique and the stacking/ensemble classifier for predicting the class labels is an efficient algorithm for this problem because, firstly, the feature technique eliminates the features below certain variance which does not allow the model to overfit on the training data and secondly the stacking/ensemble classifier is a way to improve model predictions by combining the outputs of multiple models and running them through another machine learning model called a meta-learner which significantly improves the F1 score and accuracy of the model thereby helping to correctly classify the test data into the six classes.

Discussions and Analysis

The work uses the method of stacking classifier which helps us to achieve a better-predicting model for this problem which is the merit of this work and the significant finding. This work does not explore the other used feature engineering techniques in machine learning which can prove to be more better and robust for solving the human activity recognition problem. This problem can also be approached using deep learning models such as the convolutional neural network (CNN) which may prove to be better-performing models for their deep network architecture. This work can be used to solve many similar human activity recognition problems such as posture recognition in humans and gait recognition which may be helpful in predicting Parkinson's disease at an early stage in human beings.

Individual Contribution

The simulation for mRMR feature engineering technique and implementing Random Forest, Decision Tree and SVM and performance evaluation was done by me. The design of the ensemble/stacking classifier model was mutually discussed and carried out.

References

[1] A. Doewes, S. E. Swasono, and B. Harjito, "Feature selection on human activity recog nition dataset using minimum redundancy maximum relevance," in *2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*. IEEE, jun 2017.

[2] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, aug 2005.

[3] D. H. Wolpert, "Stacked generalization," vol. 5, no. 2, pp. 241–259.