Human Activity Recognition Using Smartphones

Name:	**ARIJIT SAMAL**
Roll No.:	19051
University Name:	IISER Bhopal
Stream:	EECS
Problem Release date:	JANUARY 13, 2022
Date of Submission:	APRIL 24, 2022

1 INTRODUCTION

1.1 OBJECTIVE

Our main objective in this project is to predict human activities from the 6 given human activities. Hence, it is a multiclass classification problem. As the dataset is rather extensive we will use feature selection to find salient features of the data. Finally we will use various classification models on the training dataset and run the best framework on the test data to output the activity labels onto a text file.

1.2 ABOUT THE DATASET

There are 563 columns and 8239 rows in the training dataset. The training data class labels have two columns that indicate the data index and activity labels for each data point. We have 562 numerical features and a categorical feature called subject that represents the person who provided the reading. Finally, there are six classes, each with three static activities (standing, laying, and sitting) and three dynamic activities (walking, walking upstairs, and walking downstairs).

1.3 UNDERSTANDING THE DATASET

This dataset is gathered from 30 people (referred to as Subjects) performing various activities. The data is recorded using a smartphone's sensors (accelerometer and gyroscope). We have time series data for triaxial linear acceleration (tAcc-XYZ) from accelerometer and triaxial angular velocity (tGyro-XYZ) from Gyroscope with several variations, prefix t in the attributes denotes time whereas suffix XYZ represents 3-axial signals in X , Y, and Z directions. The dataset also contains estimated values of mean, std, max etc. derived from time series data.

2 METHODS

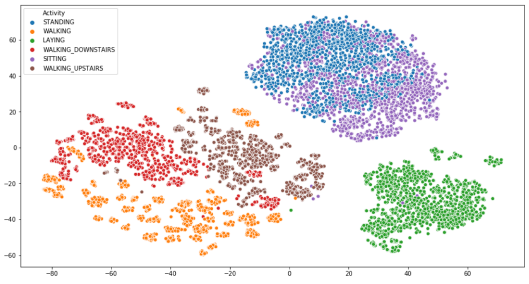
Here is the link of the GitHub repository-
HUMAN ACTIVITY RECOGNITION USING SMARTPHONES
3 CSV files are provided namely train_data, train_labels, and test_data. On the basis of the index (referred to as Unnamed: 0 in both the datasets), we will first generate a merged dataframe including train_data and train_labels which will aid in the process of EDA, data preprocessing and cleaning.

2.1 DATA CLEANING AND PREPROCESSING

First of all we check for outliers in the dataframe and found out that all the features have values lying between -1 to 1 and hence suggesting presence of no outliers in the data.Then, we start with checking for imbalanced data using the value counts function and countplots and we observe that the data is nearly well balanced and we no need to process it. As the next step we check for any null and duplicate values in the data and found that there were no null or duplicate values in the whole datset. We proceed further changing the column names to readable format. Finally, after data analysis we label encode the class labels of the data so that they can be fitted in the classification models.

2.2 EXPLORATORY DATA ANALYSIS (EDA)

We perform EDA on the merged dataset to get insights of the data and its features. We start by plotting the scatter plots and density plots for various features and we observe that some of them were not able to distinguish between classes like tBodyAccJerkmeanX but some of the features were able to distinguish between various classes like tBodyAccJerkentropyX and tBodyAccMagmean which were able to distinguish between static and dynamic activities; angleXgravityMean which was able to distinguish between laying and all other classes. We also use box plots to distiguish various classes and got the bounds of various classes for a particuar feature. Finally, we used the tSNE plot shown below



to map the 563 dimension feature vector to a 2D space and we observed that sitting and standing classes were the hardest to classify whereas the laying class was the easiest to distinguish out of all the classes.

2.3 FEATURE SELECTION

We first encode the activity labels of the data using the label encoder. the encoded labels are as follows 2- STANDING, 3- WALKING, 0- LAYING, 4- WALKING_DOWNSTAIRS, 1- SITTING, 5-WALKING_UPSTAIRS. After encoding we form the X_train dataframe by dropping the subject, index and Activity columns from the merged dataframe obtained earlier, now there are 561 features in X_train. Similarly, we obtain y_train by extracting the target variable Activity column for the merged dataframe. We also obtain X_test by dropping the subject and index columns. We develop feature selection frameworks to run the models after feature selection. We apply hyperparameter tuning on the whole training data without feature selection with a small set of parameters to compare the model before and feature selection. Finally, we fit the classification models namely KNN, Logistic regression, decision trees, random forest, linear and kernel SVM and obtain the results on the whole dataset without feature selection.Then, we proceed to feature selection and We mainly use 4 types of feature selection techniques. These include FILTER METHODS that are based on statistical techniques like ANOVA (analysis of variance) or F-test which is a parametric statistical hypothesis test for determining whether the means from two or more samples of data (often three or more) come from the same distribution or not. We use it here as we have numerical inputs and categorical target variable as we have a multiclass classification problem, Mutual information gain which is similar to the information gain concept of decision trees. Mutual Information measures the entropy drops under the condition of the target value. The MI score will fall in the range from 0 to 1. The higher value, the closer connection between a feature and the target; Extra tree classifier which is an ensemble algorithm that seeds multiple tree models constructed randomly from the training dataset and sorts out the features that have been most voted for.The most important features are found using the feature importance attribute of the classifier.Finally, the whole dataset is fitted into it and it picks out the best features reducing the overall feature space; PCA for dimensionality reduction to 20 principal componenets. We compare the raw model with the models fitted after feature selection and report the findings in a CSV file which can be found in the github link above. This file contains performance measures of different tuned models before and after feature selection. This gives us insights into which feature selection technique we should use for our dataset and by observing we found that extra tree classifier with feature importance provided the best results and we use this to reduce the number of features.

2.4 BUILDING CLASSIFICATION MODELS

First we start by creating a classification framework to run all the classification models whose pseudo code is given below-

```
def classification_framework(model, Xtrain, ytrain, Xtest, class_labels):
    #pass in desired model in model
    #pass reduced X_train, X_test from feature selection in Xtrain, Xtest
    #pass y_train in ytrain
    #pass Activity labels in class_labels

    create a dictionary called results to store results
    assign hyperparameters on basis of models passed
    run GridSearchCV on the model
    fit the model
    predict ytrain labels and store it in results
    predict ytest labels and store it in results
    print and store accuracy in results
    print and store confusion matrix in results
    print and store classification report in results
    print and store Average cross validated score in results
    print and store accuracy in results
    print and store best estimator of grid search in results
    print and store best parameters of grid search in results
    print and store no. of cross validation sets used in results
    print and store average cross validated scores in results
    store the passed model in results

    return results
```

A slightly tweaked version of this code is used as the feature selection framework, it does not print all the parameters of grid search and confusion matrix as we are only trying to select the best feature selection technique there. We also design a function to plot a confusion matrix which gives us insight into the confusion in classifying data into different classes. Before tuning we take ranges of values for a particular hyperparameter and plot them to find the optimal values of the hyperparameter which would be helpful in choosing them.Then we use the models stated above and tune their hyperparameters to find the best set of parameters for the model along with cross validation to avoid overfitting of data. We start with KNN passing in value of n_neighbors (value of k), leaf size and distance metrics as the hyperparameters. Then we use logistic regression in which we use c (Regularization parameter whose strength of the regularization is inversely proportional to c, it is strictly positive) and penalty (either l1 or l2 norm) as a hyperparameter. For decision tree we take max_depth as the hyperparameter which ddenotes the maximum depth upto which the tree is grown.Then we use random forest classifier and pass in n_estimators (number of trees in the forest), max_depth (maximum depth of the tree) and criterion (function to measure the quality of a split. It can take either gini or entropy denoting gini impurity and information gain as the splitting criteria respectively) as the hyperparameters. In linear SVM we use tol (tolerance or stopping criteria), c (regularization parameter as in logistic regression) and penalty (either l1 or l2 norm) as the hyperparameters. Finally, we tune the kernel SVM by using kernel (specifies the kernel type to be used in algorithms), c value and gamma value (which represent the kernel coefficients) as the hyperparameters. We set n_jobs=-1 for all the models so that it can use all the cores in the CPU to make the computations fast. we obtain the best estimator, parameters and scores of all the models and output them to a CSV that can be found in the GitHub link mentioned in the above section.To evaluate model performance comprehensively, we should examine both precision and recall and F1 score serves as a metric that considers both of them. We use the macro-averaged F1 score which is computed by taking the arithmetic mean of all the per-class F1 scores. This method treats all classes equally regardless of their support values. Finally, after obtaining the macro averaged F1 scores of all the models we choose the model with the highest score and predict the class labels of test data i.e. X_test using that model and output it to a text file.

3 EXPERIMMENTAL ANALYSIS

3.1 FOR FEATURE SELECTION

All the findings were reported in an excel file named feature selection results which can be found at the github link mentioned above.

1. For anova feature space is reduced to- 20 features, average F1 score over all models-0.908623362
2. For mutual information gain feature space is reduced to- 47 features, average F1 score over all models-0.922837725
3. For extra tree classifier feature space is reduced to- 97 features, average F1 score over all models-0.975185234
4. For PCA feature space is reduced to- 20 features, average F1 score over all models-0.938299478

Therefore, from the above data we chose extra tree classifier for feature selection and reduced the feature space from 561 to 97 features.
Note that the number of features selected can vary for each run.

3.2 FOR CLASSIFICATION MODELS

we report all the findings onto the excel file named hyperparameter tuning resuts which can be found at the github link mentioned above. Afte reducing the feature space we ran all the models again with more hyperparameters to find the best set of parameters for a model and with cross validation to avoid overfitting of data. For KNN the F1 score is- 0.992645144, best parameters are- 'leaf_size': 34, 'metric': 'manhattan', 'n_neighbors': 3,average cross validated score- 0.977667193. For logistic regression the F1 score is- 0.979563821, best parameters are- 'C': 1000, 'penalty': 'l2',average cross validated score-0.971477121. For decision trees the F1 score is- 0.973081254, best parameters are- 'max_depth': 9,average cross validated score is-0.933122952. For random forest the F1 score is- 0.999774168, best parameters are- 'criterion': 'entropy', 'max_depth': 13, 'n_estimators': 190,average cross validated score is-0.976089331. For linear SVM the F1 score is- 0.97495597, best parameters are-'C': 8 ,average cross validated score is-0.969413764. For kernel SVM the F1 score is- 0.977445714, best parameters are-'C': 16, 'kernel': 'rbf',average cross validated score is-0.970506129.

4 DISCUSSIONS

Our main objective in this project was to predict human activities from the 6 given human activities and we were successfully able to achieve it using the random forest classifier. Merits of the project include successful classification of standing and sitting classes, which was hard for almost all classifiers to separate; we found a suitable feature selection technique which not only reduces the number of features by more than half but also maintains the accuracy of each model which is further tuned to increase the accuracy of models. Future scopes in this project include building stacking classifier by combining already tuned classifiers with comparatively low F1 score and see if it can improve the performance of these classifiers. This is based on actual readings from a smartphone sensors data and therefore a real world project, it can be used for health related applications. This can be extended by integrating this model within a health app.