Smartphone User Identification

Madalina-Mihaela Badescu

January 5, 2023

1 Introduction

This paper describes in great detail the approach I used in the Practical Machine Learning project for Smartphone User Identification. The main goal of the competition was to develop a classifier that can group accelerometer signal recordings by user.

The dataset contains information from 20 users, each with 450 recordings. The signals are recorded for 1.5 seconds each, in the time in which a user taps the screen of his smartphone. The accelerometer records at 100 Hz and the values are in a 3D space, so they are expressed on x, y, z axes.

For the project, I trained three machine learning models, in order to find the best performer for the classifying of smartphone users through accelerometer recordings.

- The first approach I followed was with a Support Vector Machine (SVM) model. While training this classifier I tried different augmentations of the data, in order to improve the performance and the overall accuracy. The predictions made with the SVM model are used for my first submissions for the Kaggle competition, because they showed a good score.

- My second approach was to train a Linear Discriminant Analysis. I used the same preprocessing techniques as for the SVM model, but the test scores where lower than the first method. Because of this, I didn't submit any predictions made with this model to the competition.

- In search of a better performance, my third and last approach was using a Neural Network (NN) model for the task. With the NN model, I made many trials, using different parameters for the number of layers, epochs, batch sizes etc. I picked the best performers and submitted the predictions to the competition, with better results than the SVM model.

In the following sections I describe my process in detail. First, I approach the subject of the preprocessing of the data [section 2]. Second, I present the three models that I used [section 3, section 4, section 5] by defining them, then describing what attempts I made with them and, lastly, by showing their performance on different types of validation.

2 Preprocessing and augmentation of data

The first step when training a machine learning model is the preprocessing of the data. This step transforms the raw information from the dataset into data that the computer can easily interpret. The preprocessing of data creates a consistent base of information that produces reliable and accurate information.

In my approach for the augmentation of data, the first step I took was making sure that the data has the same number of values for each class. I computed the mean of the number of recordings per user, which was 150 values, and then checked every set of data, in order to bring all the values to the same total. If the number of recordings was greater than 150, then I deleted the last sets of coordinates. If the number of recordings was smaller than 150, I computed the mean for each coordinate from the recordings of that user and added the information as new recordings.

The second step of preprocessing that I followed was to flatten the recordings of each user, so that the training data that I used for the models would be two-dimensional. I transformed the information for each user into a numpy.array() and then applied numpy.ndarray.flatten(). With this approach, the training data was transformed into a list of one dimensional arrays, each representing the recordings for a user.

The last step that I took was feature scaling and the function that I used for this was sklearn.preprocessing.StandardScaler(). This function standardizes features, using the formula $f(x) = \frac{x - mean(x)}{s}$, where x represents the training data and s is the standard deviation of x. The standardization of the dataset is useful for learning algorithms such as the SVM model, because it transforms the data into features that center around 0, changing any information that could be dominant and potentially difficult to interpret.

These were the approaches that I followed for the preprocessing of data that I used in preparation for training my models. The augmentation was first created for the SVM model and, by testing with the other two models, it was proven to be helpful for both the LDA model and the NN model.

3 Support Vector Machines model

3.1 Definition of the model

Support Vector Machines are a set of supervised learning methods used mainly for classification and regression problems. C-Support Vector Classification (SVC) is a class of the SVMs that can perform multi-class classification on a set of data, as it was necessary in the case of smartphone user identification (there where 20 users, i.e. 20 classes).

SVMs assign the training samples to a hyper-plane or a set of hyper-planes, working in a high dimensional space. The classification of the samples is mapped according to the space they fall into. The mathematical formulation for SVC is trying to separate the training vectors into classes, by maximizing the margin between classes and making penalties when a value is wrongly classified.

3.2 Trial and error

For the training of the classifier I used sklearn.svm.SVC(). The implementation of the function has a number of parameters that can be configured, such as the kernel, the gamma parameter, the C parameter etc. The default kernel for SVC is the Radical Basis Function (RBF) kernel SVM. The default value for C is 1 and the default value for gamma is "scale". If the gamma parameter uses "scale" as its value, then it is equal to $\frac{1}{n_{features} \cdot X.var()}$. The C parameters is related to the margin, a bigger C meaning a smaller margin and vice-versa. The gamma parameter tells the influence of a training value and how far it reaches.

Intuitively, my first trial was with the default parameters. In order to test the results, I used the score function and had results above the base line of the competition. Determined to try different approaches, I modified the parameters, with mixed results.

- Changing the kernel from RBF to linear or to poly scores lower than the default.
- Changing the gamma parameter from scale to auto doesn't affect the score (auto has the formula $\frac{1}{n_{features}}$)
- Changing the C parameter to bigger numbers than 1 increases the score (I made training tests with numbers in the interval $[1, 2]$)

After these tests, I concluded that the best approach for the parameters was to change the C parameter to a number close to 2 and leave the others with their default values.

3.3 5-fold cross-validation

The 5-fold cross-validation is a method of validation used to compare different machine learning models and their performance. For the SVM model I computed Table 1 in order to compare the accuracy of the results using different parameters.

Table 1: Table 1

Kernel type	C	Gamma	5-fold cross validation accuracy	Score
RBF	1	scale	0.799	0.895
RBF	1.5	scale	0.815	0.910
RBF	2	scale	0.826	0.918
RBF	1	auto	0.798	0.895
Linear	1	scale	0.72	0.761
Poly	1	scale	0.543	0.761

3.4 Confusion Matrix

A confusion matrix is used to evaluate the accuracy of a classification. It represents a matrix C, in which $C_{i,j}$ computes the number of elements known to be in class i which are predicted to be in class j.

For the SVM model I computed two confusion matrices that reflect the best two algorithms: Figure 1 and Figure 2

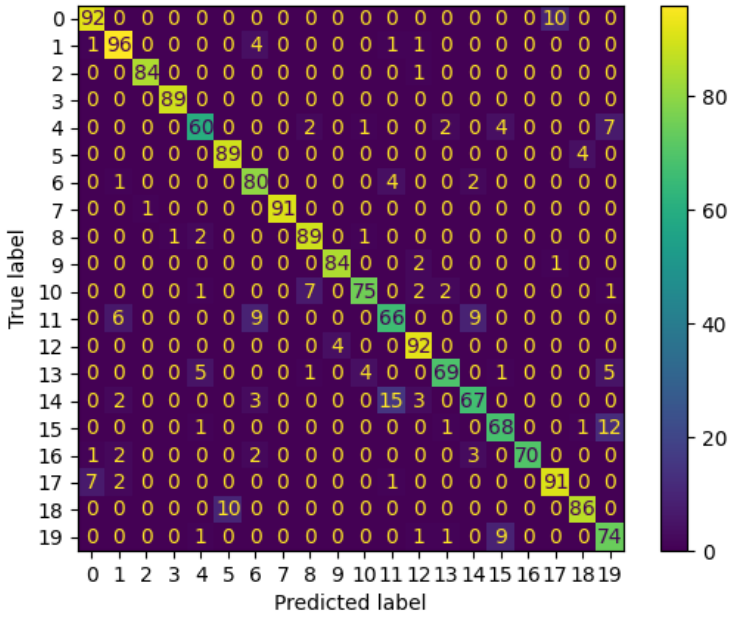

Figure 1: Confusion Matrix using the SVM model with default parameters

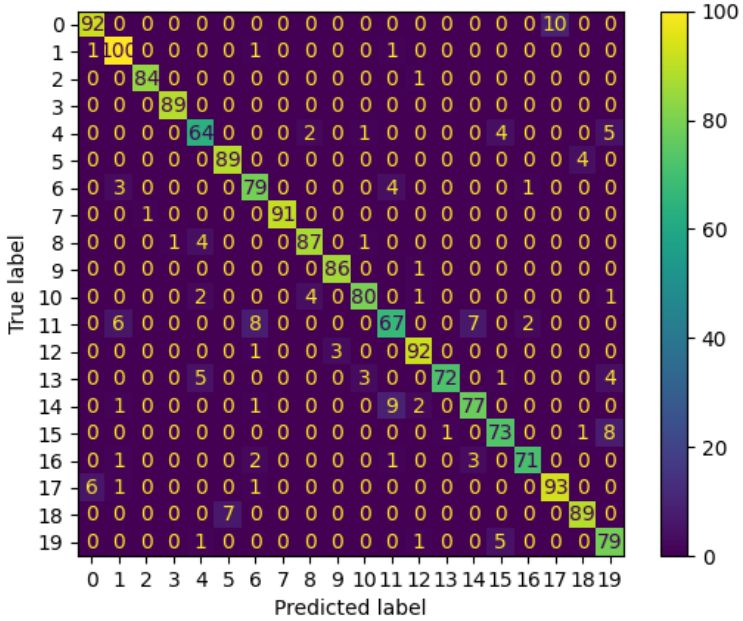

Figure 2: Confusion Matrix using the SVM model with C = 2

4 Linear Discriminant Analysis

4.1 Definition of the model

Linear Discriminant Analysis is supervised classification algorithm that reduces the dimensionality. The main idea is to find a hyperplane on which the data is projected, in a way that the distance between classes is being maximized and the variance between classes is being minimized. The Linear Discriminant Abalysis model gives a Gaussian density to classes.

4.2 Trial and error

After the results given by the SVM model, I used the Linear Discriminant Analysis model, since it tackles multi-class problems directly. The model's default parameters are the Singular Value Decomposition (SVD) solver and no shrinkage.

Since the SVD solver is the recommended option for a large number of features and it doesn't use shrinkage, I trained my model with the default parameters. The results scored lower than the SVM model's results. I tried different parameters, but the results didn't improve considerably.

4.3 5-fold cross-validation

For the SVM model I computed Table 2 in order to compare the accuracy of the results using different parameters with the 5-fold cross-validation.

Table 2: Table 2

Solver type	Shrinkage	5-fold cross validation accuracy	Score
LSQR	Auto	0.684	0.748
LSQR	None	0.549	0.693
SVD	None	0.549	0.693

4.4 Confusion Matrix

For the Linear Discriminant Analysis model I computed a confusion matrix that reflects the results: Figure 3

5 Neural Network

5.1 Definition of the model

A neural network (NN) is a set of algorithms that mimic the way the human brain works, by recognizing relationships between the set of data. Neural networks function with an input layer, then work with multiple hidden layers and produce an output layer.

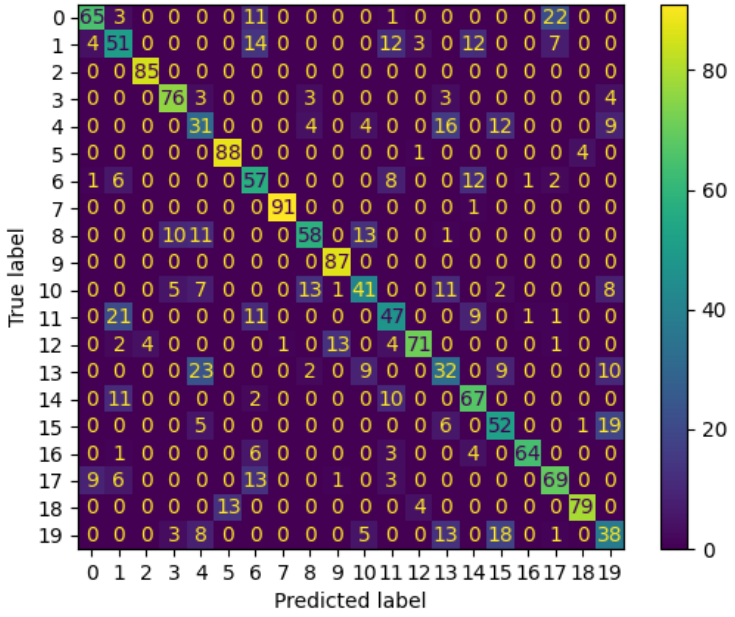

Figure 3: Confusion Matrix using the Linear Discriminant Analysis model

Each artificial neuron of the network has a weight and a threshold. Every neuron is connected to other neurons and is activated if their output meets the threshold. An active node has the function of sending data towards the next layer of the network. A neural network learns in epochs and their accuracy increases over time.

5.2 Trial and error

For the approach of training the NN model I tried many different variations. I made changes to the densities of the layers, to the dropouts, to the optimizer, to the number of epochs, to the batch size and so on. The loss of the model I kept the same, that being sparse categorical crossentropy, because it was the only one which computed good results.

The configurations of layers densities and dropouts that I tested the model for are:

- dense = 512, dense = 512, dense = 1024, dense = 1024, dropout = 0.2. In the Table 3 there is represented each trial, with their specific differences in parameters and results
- dense = 512, dense = 512, dropout = 0.2, dense = 1024, dense = 1024, dropout = 0.2. In the Table 4 there is represented each trial, with their specific differences in parameters and results.
- dense = 256, dense = 256, dense = 512, dense = 512, dropout = 0.2. In the Table 5 there is represented each trial, with their specific differences in parameters and results. The accuracy and val accuracy are registered from the last epoch.

Table 3: Table 3

Optimizer	No. of epochs	Batch size	Accuracy	Val Accuracy
SGD	10	16	0.964	0.897
SGD	15	16	0.979	0.907
SGD	20	16	0.987	0.918
Adam	20	16	0.975	0.905
Adam	25	16	0.977	0.908
SGD	20	32	0.989	0.930

Table 4: Table 4

Optimizer	No. of epochs	Batch size	Accuracy	Val Accuracy
SGD	20	16	0.957	0.897
SGD	20	32	0.976	0.934

Table 5: Table 5

Optimizer	No. of epochs	Batch size	Accuracy	Val Accuracy
SGD	20	16	0.977	0.906
SGD	20	32	0.985	0.885

After analyzing the results from these computation, I concluded that the first set of density for layers and dropout is the best, combined with the SGD optimizer, 20 epochs and a batzh size of 32.

6 Conclusion

In conclusion, there is more than one good approach for handling Smartphone User Identification with machine learning models. My best results were computed using Neural Networks, but the SVM model also represents a good choice.

As future developments for the algorithms, there is a possibility of increase in accuracy if more complex feature processing would be done. Also, more fine-tuning of the parameters and hyper-parameters would probably lead to even better results.