TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Evgeny Chubarov 120951

LANGUAGE FAMILIARITY MODELING ON THE BASIS OF FACIAL MOTIONS

Bachelor's thesis

Supervisor: Sven Nõmm
PhD

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Evgeny Chubarov 120951

KEELTEOSKUSE MODELLEERIMINE NÄO LIIKUMISTE PÕHJAL

Bakalaureusetöö

Juhendaja: Sven Nõmm
PhD

Tallinn 2018

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Evgeny Chubarov

19.05.2018

Abstract

The purpose of this work is to explore whether it is possible to determine if individual is speaking their native language or not on the basis of facial motions.

The result consists of two parts. The first one is a desktop application written on C# to record face motions using Microsoft Kinect sensor and convert saved coordinates to features. The second part is series of Python scripts to display, validate and process collected data, using statistics and different machine learning techniques.

During the data collection phase, 30 people were recorded speaking different languages, however it was not possible to use all the records for models building. While testing hypothesis the actual goal was changed to distinguish between Russian as native and English as foreign, using only half of collected data.

Multiple different classifiers were build on selected features, as a result – the best model is showing accuracy of 75%, which is good enough with such amount of training data.

This thesis is written in English and is 37 pages long, including 6 chapters, 23 figures and 2 tables.

Annotatsioon

Käesoleva töö eesmärk on uurida, kas on võimalik tuvastada kas inimene räägib emakeeles või mitte, kasutades tema näo liikumisi.

Tulemusena on esitatud kaks osa. Esimene on Windows OS rakendus, kirjutatud keeles C#, et registreerida ja salvestada näo liikumisi Microsoft Kinect sensori abil, ja konverteerida neid koordinaate tunnusjoonideks. Teine osa on mitu skripte Python keeles, et kuvada, valideerida ja töödelda andmeid, kasutades mitmesugusi statistika ja masinõpe tehnikaid.

Andmete kogumise ajal olid filmitud ja salvestatud sensori abil 30 inimeste näo liikumised, igaüks rääkis kaks või rohkem keelt umbes 30 sekundit, aga kasutada kõik kogutud andmeid ei ole võimalik. Hüpoteesi testimise ajal oli määratud, et kõikide inimeste segatud andmed on väga sarnased ja ei saa olla eraldatud, vähemalt valitud tunnusjoonide abil. Mudelite eesmärk oli natuke muutunud et eristada vene keelt emakeelena ja inglise keelt võõrkeelena, kasutades ainult poole andmeid.

Töö tulemusena on mitu mudelid, millest parem näitab 75% täpsust. See on pigem hea näit, arvestades, kui vähe andmeid oli kasutatud mudelite koolitamas.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 37 leheküljel, 6 peatükki, 23 joonist, 2 tabelit.

List of abbreviations and terms

Feature	Individual measurable characteristic of observed samples
Model	Algorithm trained on a sample data with developed heuristics during training and weights
API	Application programming interface
KNN	K-nearest neighbors, classification algotithm
SVM	Support vector machine, classification algorithm
IDE	Integrated developnebt environment
CSV	Comma-separated values, file format
OS	Operating system
WPF	Windows presentation foundation
.NET	Software framework by Microsoft, mainly for Microsoft Windows
MVP	Model-View-Presenter, software architectural pattern

Table of Contents

List of Figures

List of Tables

1 Introduction

Face recognition is quite new and challenging field, not so many studies are present regarding facial movements analysis and comparison in different conditions. Machine learning techniques are very good for such data, because of its quantity and structuredness. Sensor used in this work gives a stream of frames containing 1347 face points, about 15-30 frames per second. Analysis of such data is a good chance to discover more about face movement patterns.

1.1 Problem Statement and Objectives

Given two sets of data – people speaking their native language and same people speaking foreign language, the aim is to determine whether it is possible to distinguish between them using machine learning classification methods.

The work consists of three phases:

1) Collecting and converting data.
2) Finding if sets are separable using data analysis.
3) Building models if it makes sense, based on second phase results.

The aim is to build a successful model that is able to distinguish level of language familiarity of individual.

1.2 Limitations

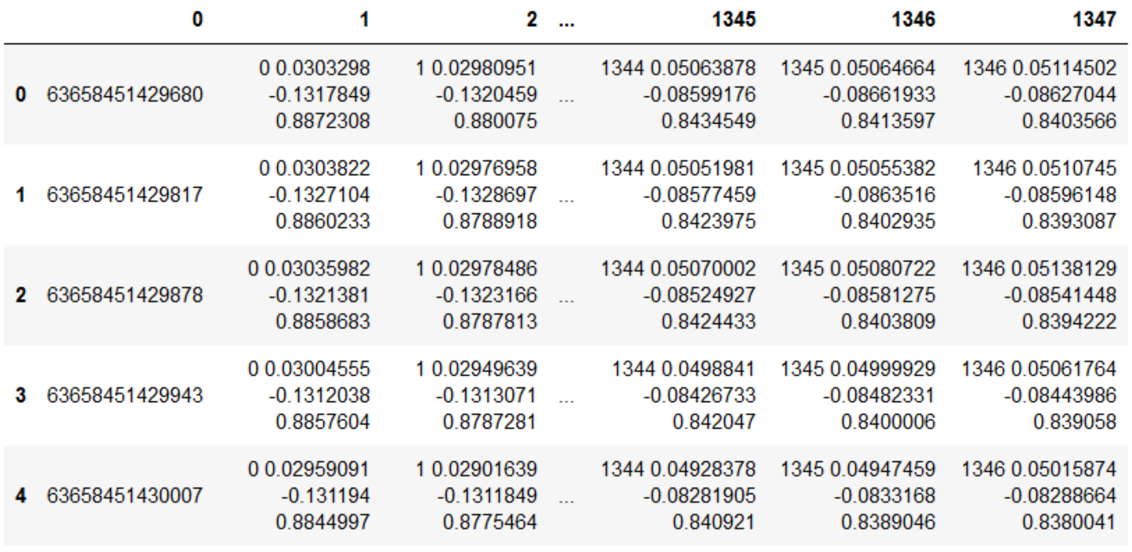
In total were recorded 30 people, each one speaking two or more languages. During analysis it was decided to reduce the working data set to people who speak Russian as native language and English as foreign language. That is only 14 people, which gives 28 entries in total.

Due to not so high number of samples used in this work, built models are not very accurate, however the resulting solution can be easily used to collect any number of training samples and to build new models, improving or modifying used in this work methods if necessary.

2 Theoretical Background

2.1 Feature engineering

Building features from raw data is a very big part of intelligent system creation. Even with new smart methods like deep learning, you have to filter and transform data to correspond specific problem to get better performance. [2] Domain knowledge is what helps to transform raw data to meaningful features, increasing predictive power of resulting machine learning algorithms. [3]



	0	1	2	...	1345	1346	1347
0	63658451429680	0 0.0303298 -0.1317849 0.8872308	1 0.02980951 -0.1320459 0.880075	...	1344 0.05063878 -0.08599176 0.8434549	1345 0.05064664 -0.08661933 0.8413597	1346 0.05114502 -0.08627044 0.8403566
1	63658451429817	0 0.0303822 -0.1327104 0.8860233	1 0.02976958 -0.1328697 0.8788918	...	1344 0.05051981 -0.08577459 0.8423975	1345 0.05055382 -0.0863516 0.8402935	1346 0.0510745 -0.08596148 0.8393087
2	63658451429878	0 0.03035982 -0.1321381 0.8858683	1 0.02978486 -0.1323166 0.8787813	...	1344 0.05070002 -0.08524927 0.8424433	1345 0.05080722 -0.08581275 0.8403809	1346 0.05138129 -0.08541448 0.8394222
3	63658451429943	0 0.03004555 -0.1312038 0.8857604	1 0.02949639 -0.1313071 0.8787281	...	1344 0.0498841 -0.08426733 0.842047	1345 0.04999929 -0.08482331 0.8400006	1346 0.05061764 -0.08443986 0.839058
4	63658451430007	0 0.02959091 -0.131194 0.8844997	1 0.02901639 -0.1311849 0.8775464	...	1344 0.04928378 -0.08281905 0.840921	1345 0.04947459 -0.0833168 0.8389046	1346 0.05015874 -0.08288664 0.8380041

Figure 1. Raw data example

Figure 1 shows an example of raw data, consisting of a timestamp and multiple point coordinates. Such data cannot be used in algorithms and should be transformed to meaningful values. This is not always the case, so sometimes numeric values could be used as is, usually if they represent values or counts. [2] Such features are known as raw features, and the ones that are extracted from other attributes are called derived features. [2]

Each sample often has lots of different features. Such combination is mostly shown as a matrix, where each column represents feature values and rows – individual observations (Table 1). In case of supervised learning the last column usually represents a class of a sample.

Table 1. Feature examples

	321	835	label
0	0.282302	0.244955	foreign
1	0.947206	0.852025	foreign
2	0.145307	0.142516	foreign
3	0.155932	0.151248	foreign
4	0.281287	0.234780	foreign

2.2 Data Analysis

2.2.1 Hypothesis testing

After features are created, but before training any models, we should statistically approve that two sets of data are different and separable, otherwise it makes no sense to build any classificator. This is where we need hypothesis testing. Here is used Student t test or just t-test, which allows to compare the means of two sets.

First the null hypothesis is stated, saying that sets are the same and their means are not different, the purpose of test is to reject it. Two different formulas are used to calculate t-value, depending on sets relativity: Formula 1 is used for not relative data, and Formula 2 for related, or paired data (for example same measurements before and after some action). [8]

$$t=\frac{m_A-m_B}{\sqrt{\frac{S^2}{n_A}+\frac{S^2}{n_B}}} \tag{1}$$

$$t=\frac{m}{s/\sqrt{n}} \tag{2}$$

T-value shows the size of the difference between two given sets in units of standard error. The greater is t-value, the greater are chances that sets are significantly different – and against null hypothesis. [9]

Based on t-value we can calculate p-value, which is a probability of getting the observed or more extreme value if null hypothesis is true. That means the higher is absolute t-value, the lower is probability of null hypothesis being true.

The critical value to reject null hypothesis and approve alternative hypothesis is known as alpha-level, and usually is set to 0.05. The difference between sets is statistically significant if $P < 0.05$.

2.2.2 Feature selection

When features are extracted from raw data, next important step is feature selection. That means we should choose features that are most relevant for the current problem. [3] The reason to use limited number of features and not all of them are:

a) Training time significantly increases with number of features.

b) High number of features increases risk of overfitting. [4]

The most widely used method for selecting best features is Fisher score. Its main idea is to find subset of features, such that in the resulting space, distances between clusters are the largest, while distances between points inside same cluster are the smallest. That shows Formula 3, where k is number of clusters, m – mean value, s – standard deviation, and p – ratio of samples with current label to the total number of samples. [6]

$$F=\frac{\sum_{j=1}^{k} p_j (m-m_j)^2}{\sum_{j=1}^{k} p_j (s_j)^2} \tag{3}$$

2.3 Machine Learning

Machine learning is a technique of building an algorithm not by explicitly programming it, but by giving it enough data to find patterns on it's own in order to achieve result. Those algorithms can be used for different purposes, such as objects detection on images/video, weather prediction or medical diagnosis. [10]

Technique used in this work is called supervised classification, meaning all data was separated and labeled before being used in model training. System is given both input and output variables, and searches for correlation between them to be able to classify unknown data in future. As it has certain set of labels, it would only be capable of assigning those known classes. Here are listed algorithms used in current work.

2.3.1 Decision tree

Decision tree is an algorithm which builds tree in order to predict labels. The model can be easily read by human and in some way mimics human approach to make a decision. While growing, it recursively estimates how much more organized becomes independent variable if we split it according to the dependent variable's value, and does those splits until decision can be made. [10]

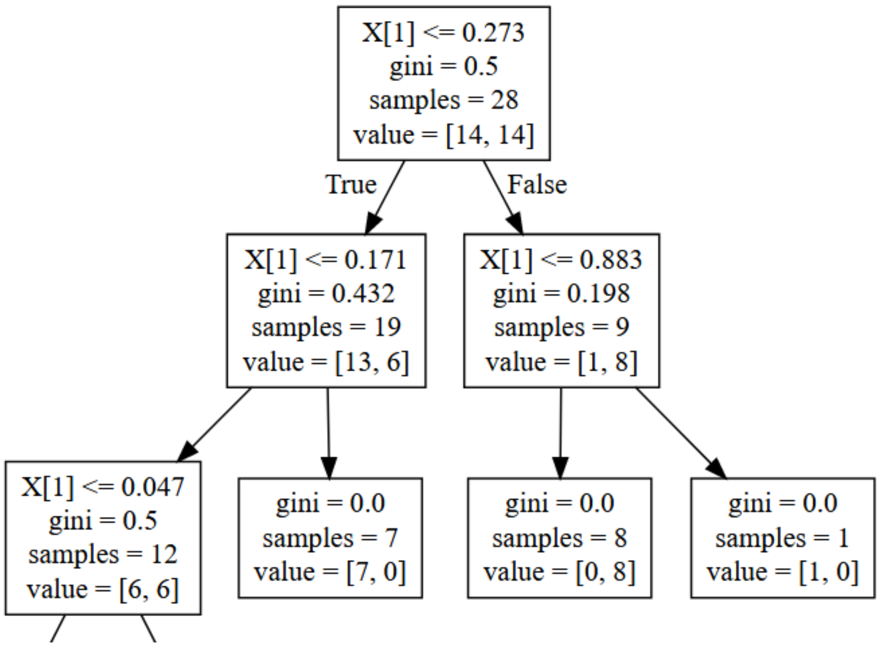


Figure 2. Part of generated decision tree

2.3.2 Random forest

Random forest is a set of individual decision trees, trained by slightly different samples. To predict the value each tree is doing it separately, and then the answer of majority is given as a prediction. [11]

2.3.3 K-means

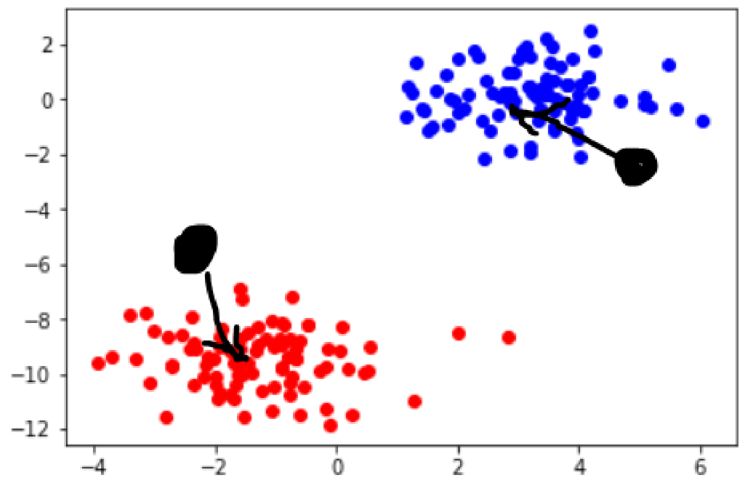


Figure 3. Example of k-means clustering

K-means is not classification, but clustering algorithm. However it can be used to see if data can be separated for given number of classes. The process starts with placing K centroids randomly or by some initializing strategy, and every point gets assigned to one of them, the nearest. Each step centroids is are moved to the average location of points in this cluster and points get reassigned. Process ends when no change is seen in clusters. [12]

2.3.4 KNN (k-Nearest Neighbors)

K-Nearest Neighbors is an algorithm to classify data, according to values of N nearest data points as shown on Figure 7. Such model doesn't need to be trained before using, but has to keep all the data in memory in order to classify new point. [13]

2.3.5 SVM (Support Vector Machine)

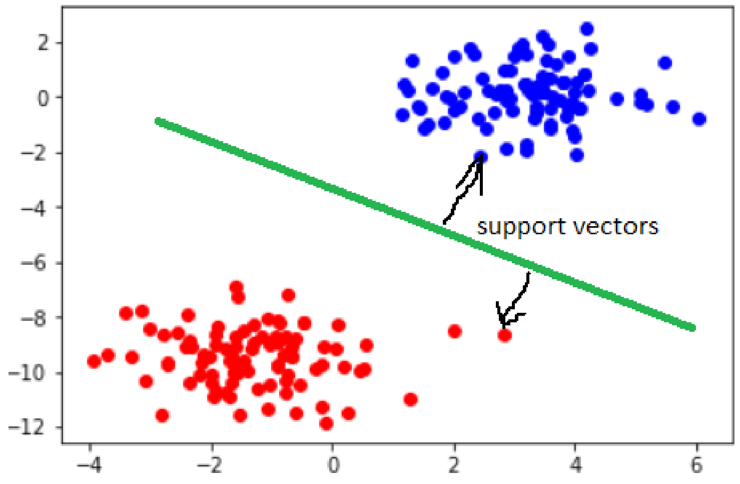


Figure 4. SVM example

Support vector machine is an algorithm that tries to find a hyperplane that best divides a dataset into two classes. Support vectors in this case are the points nearest to the hyperplane, based on which the distance of cluster to the hyperplane is measured. In case of two classes the hyperplane is just a line, three classes – plane etc. If data cannot be clearly separated by a hyperplane, moving to higher dimension is needed. [14]

2.3.6 Boosting

Boosting is an algorithm which takes weak classifying algorithm, and improves its accuracy. The base algorithm is applied iteratively, each time defining new weak prediction rule and in the end of process combines them into a one strong rule. [15] It means that algorithm is learning on own mistakes and next predictor should be more accurate than previous, considering misclassified samples.

3 Tools

3.1 Hardware

The device which was chosen to collect face movements data is Microsoft Kinect sensor. It provides rich API for .NET framework which allows to create various applications. The sensor is capable of streaming 15-30 frames per second, containing data about recognized individual body and face. [16]

For this work was registered and stored all 1347 face points provided by API, recorded while people were speaking in front of sensor.

3.2 Programming Languages

The choice of programming languages is explained by current work field and used device. Because Microsoft Kinect sensor was used to record data, C# was chosen to build the desktop application for collecting and converting data as the most rich and powerful .NET programming language. Visual Studio 2017 was used as an IDE.

For the modeling part was used Python of version 3.6.5, because of its brevity, great choice and quality of machine learning libraries, and lots of existing examples of applying them in practice.

3.3 Libraries and other

Kinect for Windows SDK 2.0 in order to use sensor on computer. [25]

All data used in work is stored in CSV (comma-separated values) files. This format gives maximum of flexibility, because it’s human-readable, is widely used and supported by many tools, and easy to write or access manually or by self-written code.

3.3.1 C#

Besides standard libraries, there was used Microsoft.Kinect reference, which provides API for accessing and managing sensor. [17] Application was built using WPF (Windows Presentation Foundation) graphical subsystem. [18]

3.3.2 Python

Reading and writing of datasets is done using pandas [19] library. For implementation of machine learning algorithms and model training was used scikit-learn [20] , for statistic scipy [27] . Plotting is done using matplotlib [21] and mlxtend [23] . For numeric calculations numpy library is used. [22]

As IDE was chosen Jupyter notebook [24] , because it allows to easily and interactively execute portions of code, display formatted output and draw plots just in browser. The python kernel is running in background, executing given code and holding contexts. Such notebooks could be supplemented by notes and headings, sent to any other machine and executed exactly the same way, that is why this approach is good for educational purposes.

4 Methods

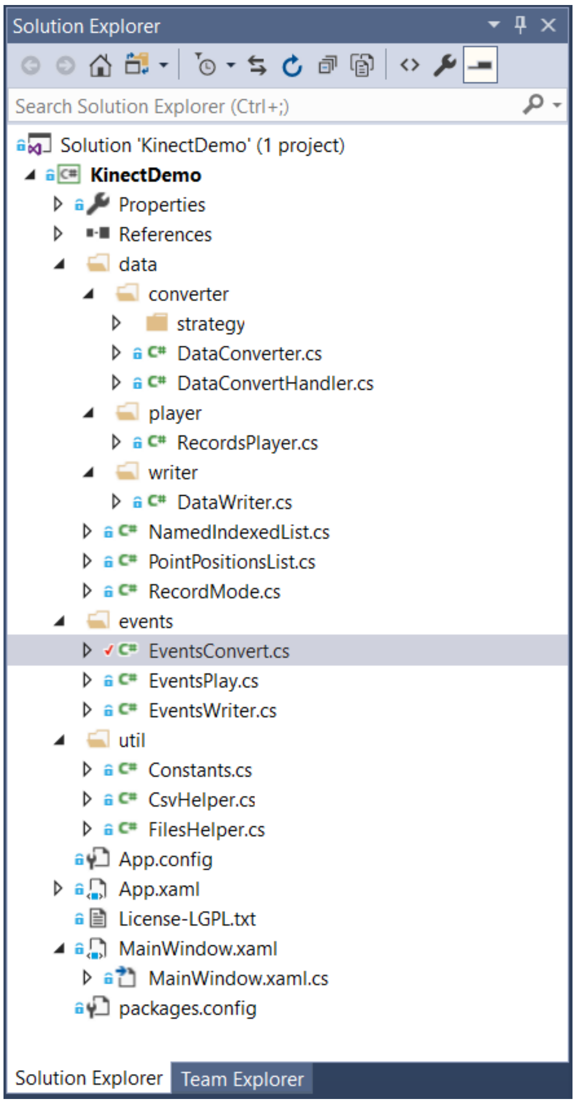
4.1 C#

In order to collect data was build the desktop application that uses Kinect sensor to obtain face points and save them to file.

The application is written on C# with WPF as a view, and has an architecture close to MVP (Model-View-Presenter). Three components are located on separate tabs and serve different purposes.

The view is passive and only fires events on elements. Events* classes serve as presenter layer, receiving events from view, and answering to model about view state or altering it.

Util classes contain static methods to execute common tasks.



```
Solution 'KinectDemo' (1 project)
  KinectDemo
    Properties
    References
    data
      converter
        strategy
        DataConverter.cs
        DataConvertHandler.cs
      player
        RecordsPlayer.cs
      writer
        DataWriter.cs
      NamedIndexedList.cs
      PointPositionsList.cs
      RecordMode.cs
    events
      EventsConvert.cs
      EventsPlay.cs
      EventsWriter.cs
    util
      Constants.cs
      CsvHelper.cs
      FilesHelper.cs
    App.config
    App.xaml
    License-LGPL.txt
    MainWindow.xaml
      MainWindow.xaml.cs
    packages.config
```

Figure 5. C# project structure

4.1.1 Collecting Data

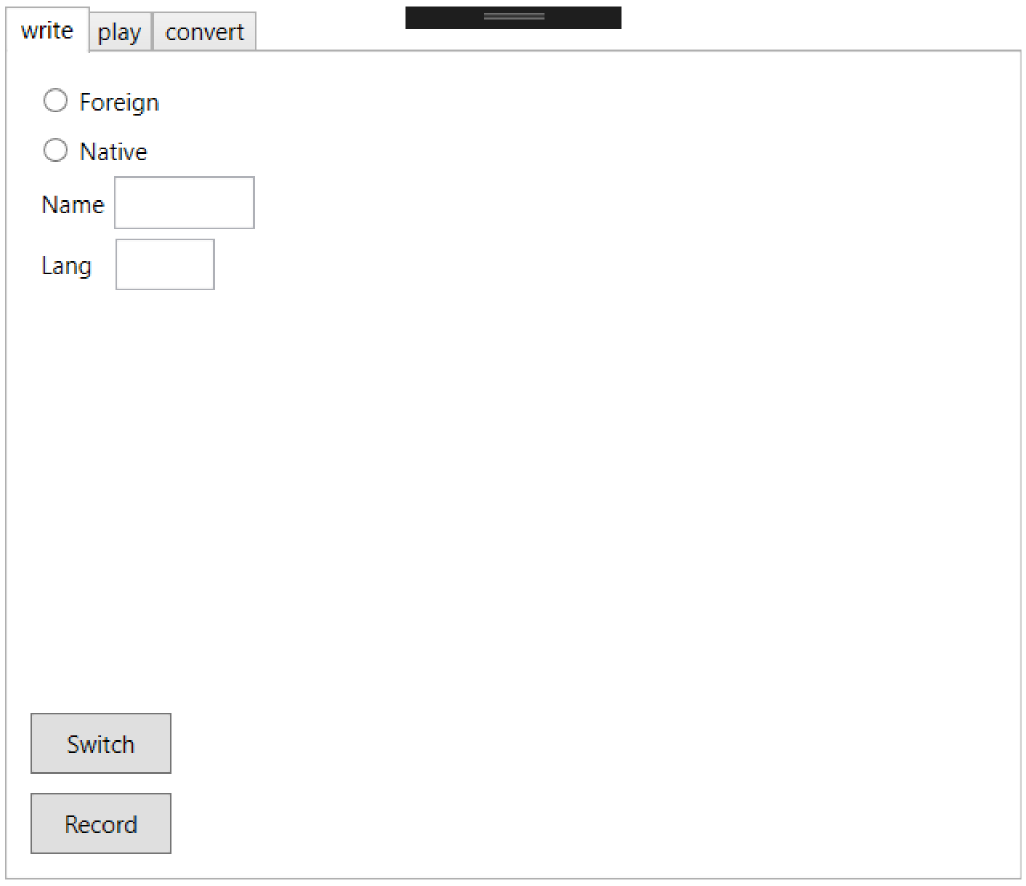


Figure 6. Data collecting interface

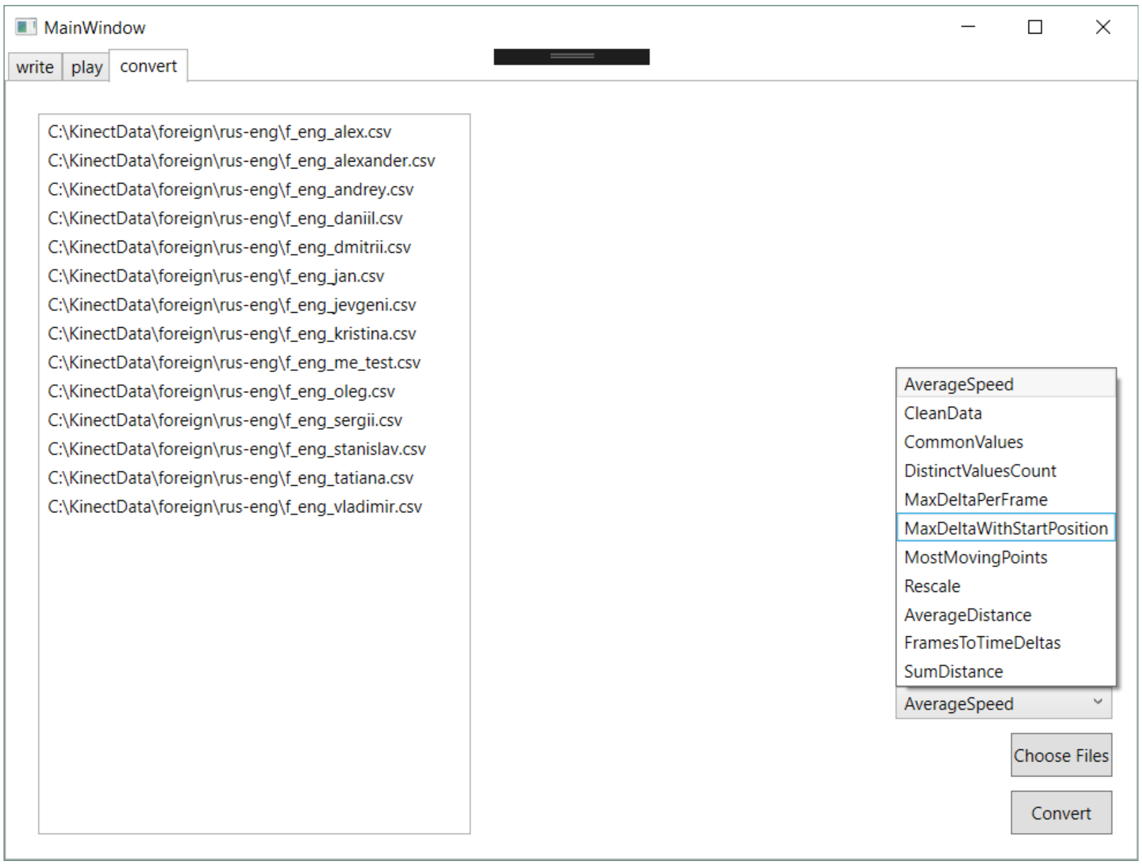
Data collection was done on first tab of application, here are located two switches – for Kinect itself and to start/stop recording. Fields are required to compose file name with metadata: individuals name and recording language.

When sensor is turned on, detected face points are shown on the screen on top of a depth-space video stream. It is not as resource demanding as a color video and can be shown at faster rate. All the shapes of objects and people are well distinguishable. [26]

Csv of such raw data takes a lot of disk space – about 1 MB for 1 second of motions. Saving of it can last more than recording itself, but it happens in parallel so we do not need to wait to start recording next sample. It gives good opportunity to record individual samples fast one after aother. The interface has an indicator showing if record is happening and how many frames are waiting in a queue.

As a result – a file is created with name indicating its language, familiarity level and individuals name. Each row of such CSV file is timestamp of frame plus all points positions at that moment.

4.1.2 Converting Data



```
C:\KinectData\foreign\rus-eng\f_eng_alex.csv
C:\KinectData\foreign\rus-eng\f_eng_alexander.csv
C:\KinectData\foreign\rus-eng\f_eng_andrey.csv
C:\KinectData\foreign\rus-eng\f_eng_daniil.csv
C:\KinectData\foreign\rus-eng\f_eng_dmitrii.csv
C:\KinectData\foreign\rus-eng\f_eng_jan.csv
C:\KinectData\foreign\rus-eng\f_eng_jevgeni.csv
C:\KinectData\foreign\rus-eng\f_eng_kristina.csv
C:\KinectData\foreign\rus-eng\f_eng_me_test.csv
C:\KinectData\foreign\rus-eng\f_eng_oleg.csv
C:\KinectData\foreign\rus-eng\f_eng_sergii.csv
C:\KinectData\foreign\rus-eng\f_eng_stanislav.csv
C:\KinectData\foreign\rus-eng\f_eng_tatiana.csv
C:\KinectData\foreign\rus-eng\f_eng_vladimir.csv
```

Figure 7. Data converting interface

Special converter was created to turn raw data into features. Files related to the same group should be selected to be converted using needed strategy. Converting is generalized so that each file is transformed to one result column. Converter handles process and saves the result, while classes implementing strategy do the main job and define meta parameters such as importance of indexing and result file name. Figure 8

Classes implementing abstract strategy are found in a namespace automatically and loaded into combobox so new ways to convert are added just by creation of new implementation. Convertion at some point may be even more generalized, for example multiple features are working with distance differences between point sequential positions, so this is extracted to `UsingPointsDeltasConvertStrategy.cs`.

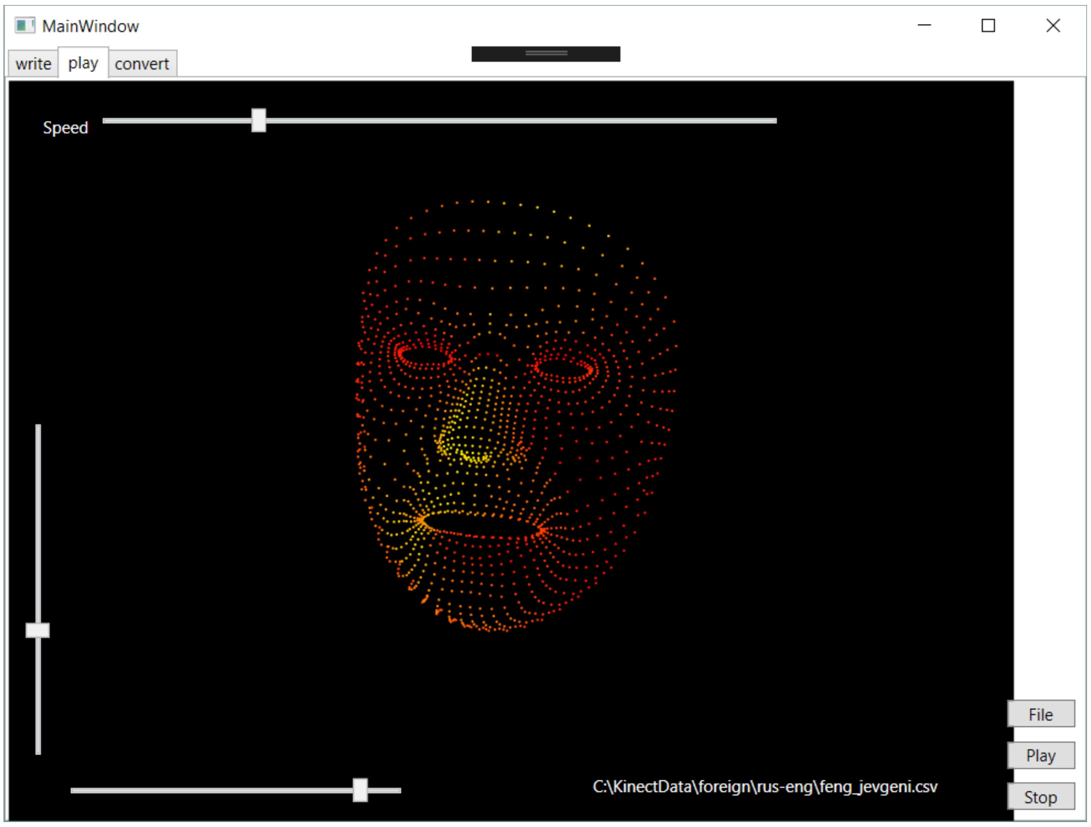
```
public void Convert(string parameters) {
        if (FilesToConvert == null || FilesToConvert.Length == 0)
                throw new Exception("Please choose files");

        result = ConvertStrategy.ConsumeFiles(parameters,FilesToConvert);
        string resultDir = Constants.DIR_BASE_OUTPUT +
                                   Constants.DIR_CONVERTED;
        string resultFilePath = string.Format("{0}{1}.csv", resultDir,
                FilesHelper.GetIncreasedVersionOfFile(resultDir,
                        string.Format("{0}_{1}",
                        ConvertStrategy.ResultFileName, ConvertId)));

        FilesHelper.WriteLogLine(resultDir + Constants.LOG_FILE_NAME,
                ConvertStrategy.GetLogSummary(result));
        CsvHelper.WriteCsv(resultFilePath, result,
                ConvertStrategy.DoIndexesMatter());
}
```

Figure 8. DataConverter main function

4.1.3 Records playing



C:\KinectData\foreign\rus-eng\feng_jevgeni.csv

Figure 9. Recorded motions playing

The last tab is meant to play recorded face motions. Any recorded file could be chosen or drag'n'dropped into the panel to be reproduced. It is meant for the raw records and as not all the people were sitting still and on the same distance from sensor, the position of record can be adjusted by sliders. Also speed can be changed.

The Fisher scores saved during feature selection are shown on the record using color of face points. Gradient goes form yellow (high score) to red (low score).

4.2 Python

4.2.1 T-Test

In order to start analysis, all created features should be combined to two sets – people speaking native and foreign language respectively, with samples and features in the same order. Each file from conversion contains samples for one feature for one language (Figure 10).

As half of sample data was initially separated (only Russian-English speaking people), it was decided to execute two t-tests independently and then choose larger or smaller set of data to proceed with modeling, so in process of merging, four different datasets were created – two for foreign and native language of 14 Russian-English speaking individuals, and two sets for all people.

As implementation of t-test was used `scipy.stats.ttest_rel` function, it works with related sets. Our datasets are related to each other, because all the

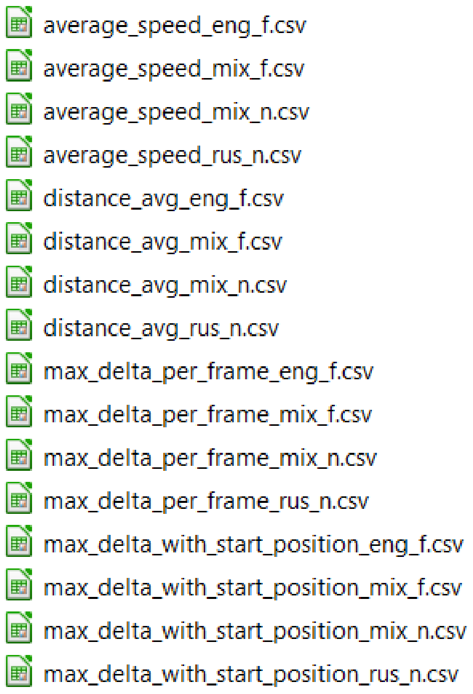

average_speed_eng_f.csv
average_speed_mix_f.csv
average_speed_mix_n.csv
average_speed_rus_n.csv
distance_avg_eng_f.csv
distance_avg_mix_f.csv
distance_avg_mix_n.csv
distance_avg_rus_n.csv
max_delta_per_frame_eng_f.csv
max_delta_per_frame_mix_f.csv
max_delta_per_frame_mix_n.csv
max_delta_per_frame_rus_n.csv
max_delta_with_start_position_eng_f.csv
max_delta_with_start_position_mix_f.csv
max_delta_with_start_position_mix_n.csv
max_delta_with_start_position_rus_n.csv

Figure 10. Convert results

samples of two sets correspond to each other, as it is the same person speaking different languages.

The features used for analysis are: average speed of point (mean of ratios of distance delta to time delta between all sequential frames), average distance traveled, maximum distance traveled per frame, and maximum deviation of point from starting position. Every feature is connected to one face point. They are all combined to one dataset for the ease of processing. On Figure 11, Figure 12 we can see that all the features are clearly distinguishable and have different values. All values were normalized before processing.

First plot (Figure 11) is showing p-values for all people, so we can see that not a single feature here can be used to distinguish two sets, in other words all mixed data is very similar for us to be able to separate two sets.

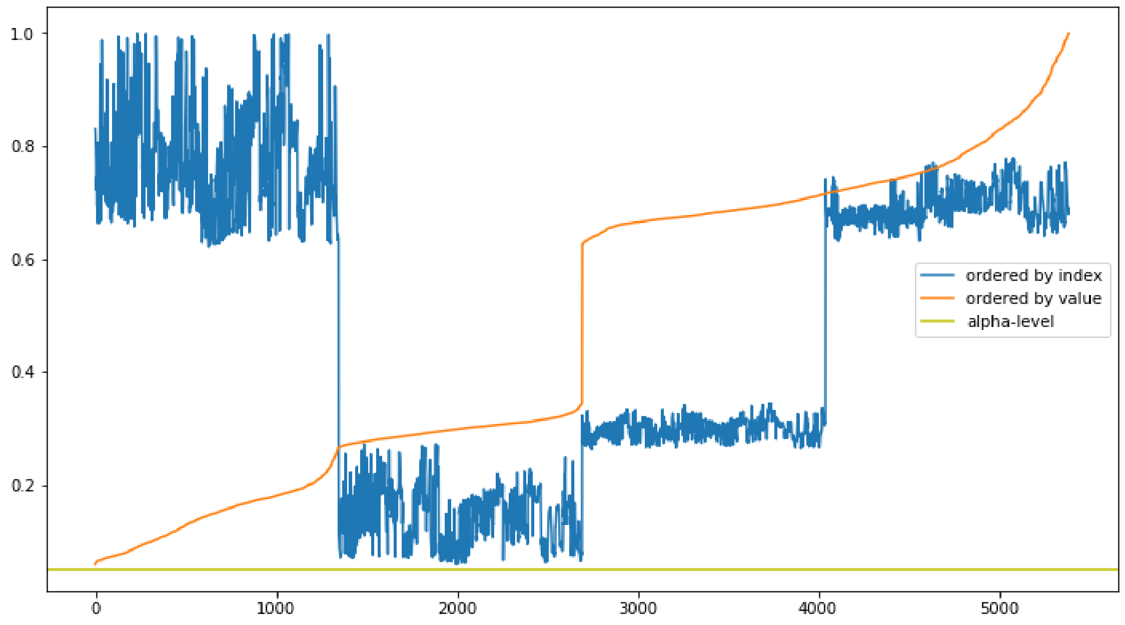


Figure 11. P-values of all people

However, the subset of only Russian-English speaking people show us that there are 166 features with p-value small enough to reject null hypothesis (Figure 12). It means that we could build a classificator good enough for separating those two classes. We can see that all those features belong to the third set – maximum distance delta of point per frame.

Table 2. Partial list of features for foreign language of all people

	laurent	alex	alexander	andrey	...	flavia	egert	urmas	vladimir
0	0.000021	0.000018	0.000030	0.000023	...	0.000016	0.000020	0.000023	0.000018
1	0.000021	0.000018	0.000030	0.000024	...	0.000016	0.000020	0.000024	0.000018
2	0.000022	0.000019	0.000032	0.000026	...	0.000017	0.000022	0.000025	0.000019
3	0.000023	0.000019	0.000033	0.000026	...	0.000018	0.000023	0.000026	0.000020
4	0.000023	0.000019	0.000033	0.000027	...	0.000018	0.000023	0.000027	0.000020

Here we can see a part of one table, containing all features of all people (Table 2). It has size of (5388, 28), the same as corresponding table of native language features. Russian-English speaking subsets have sizes of (5388, 14).

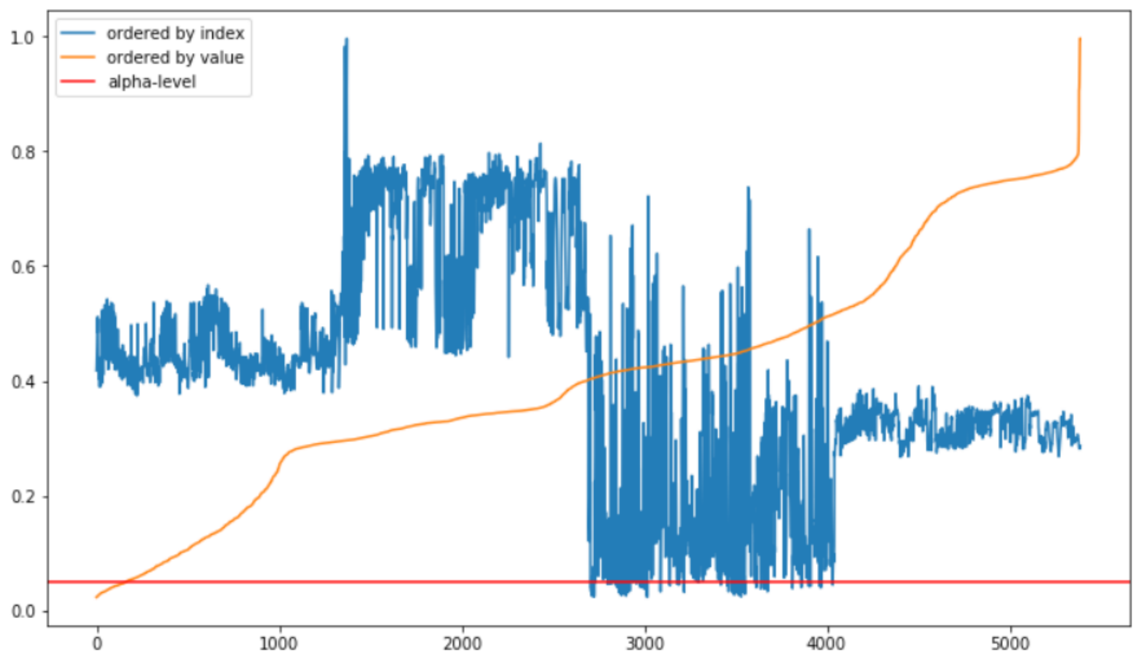


Figure 12. P-values of Russian-English speaking people

4.2.2 Fisher score and correlation

Now we can compare Fisher scores of features to choose the best ones for training models. The formula shown in chapter 2.2.2 was applied to every feature and results could be seen on Figure 13, red points show top 30 selected features. We can see that fisher score and p-value are connected – the lower is p-value, the higher is Fisher score and importance of given feature.

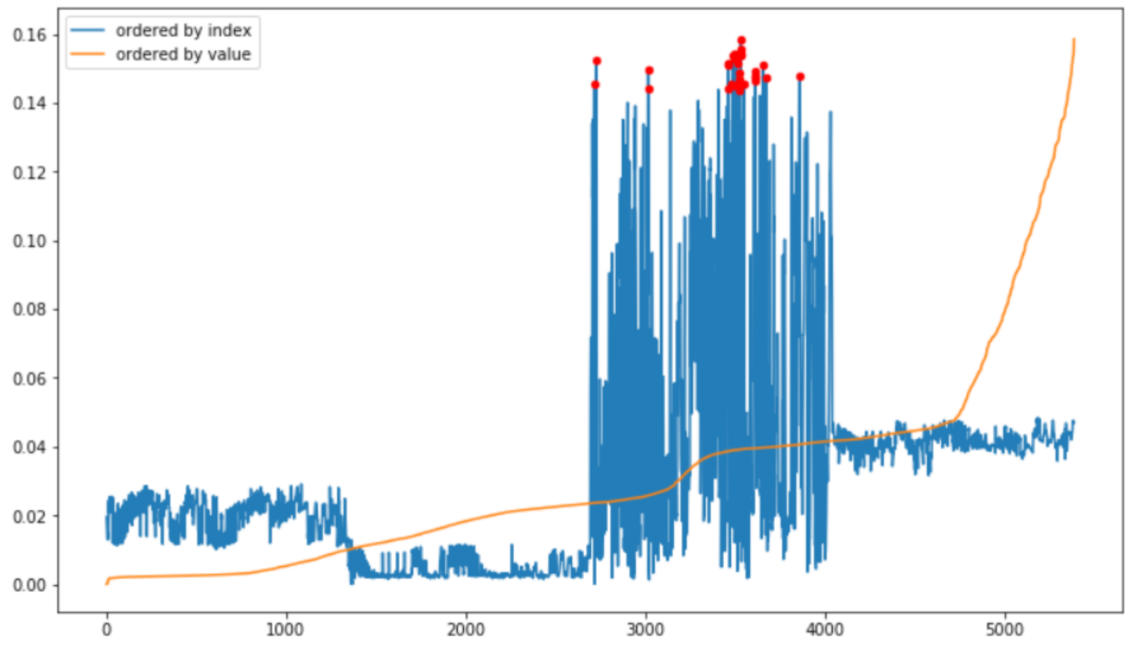


Figure 13. All Fisher scores with top 30 marked

One moment to notice is that features with the highest Fisher score are correlated with each other. On Figure 14 we can see their values through all 28 samples of data, which are (although normalized) are pretty close. It doesn't mean this is bad or will decrease models efficiency, but has to be considered.

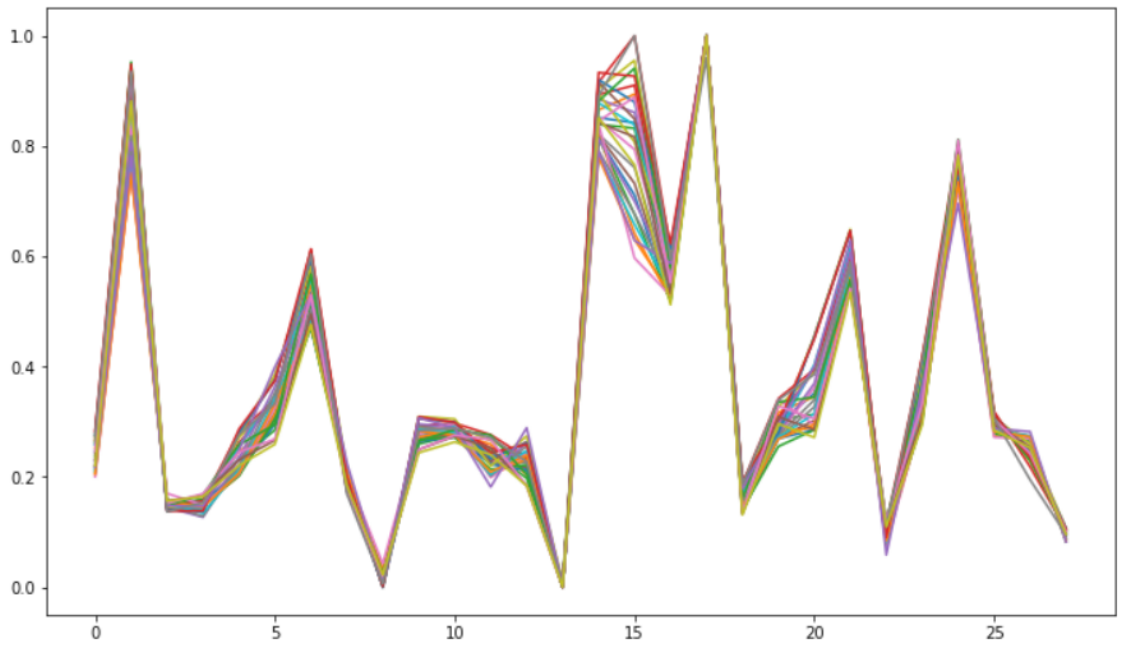


Figure 14. Top 30 features according to Fisher score

Attempt to remove correlated features was made by calculating correlation matrix and removing everything with correlation score to any other feature higher that 0.8. Only three features left after that, their value change could be seen on Figure 15. It can seem as nice features to keep, but only one of them has high enough Fisher score and low p-value, so the use of those features will not help in given datasets separation.

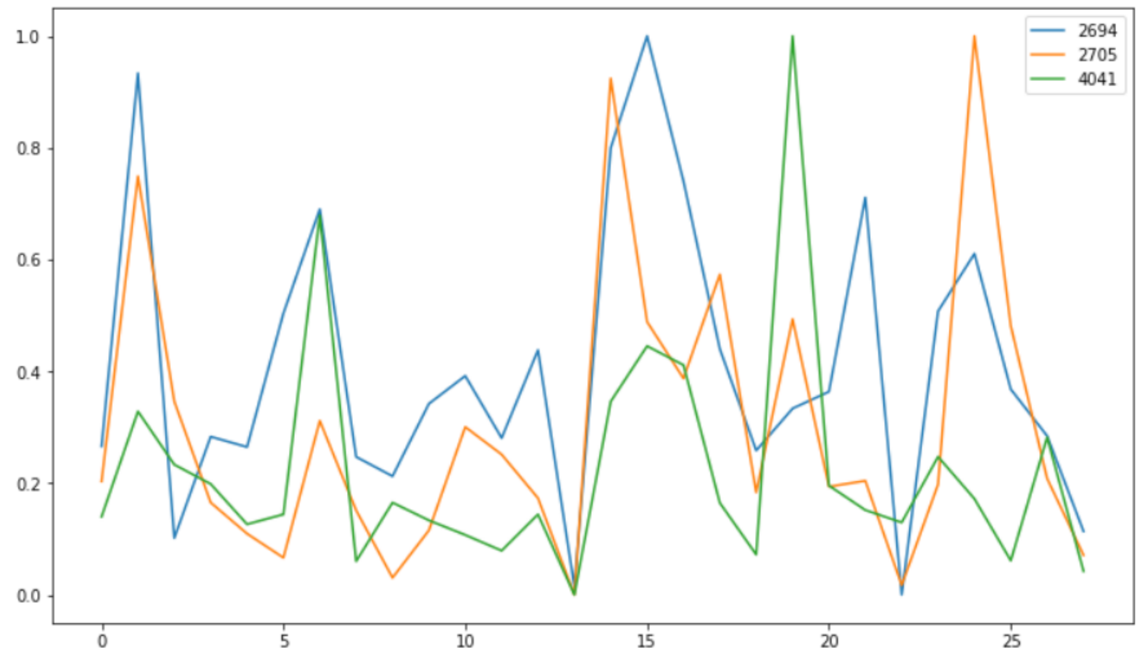


Figure 15. Most not correlated features values

Was decided to use two features to avoid overfitting on such a small number of samples. Although two best features are with numbers 3529 and 3530, they are located very close to each other, so one of them was replaced by feature 3015 with almost same Fisher score. We can get actual numbers of original points by taking a remainder after division by 1347 (total number of points, four features for each), which are 321 and 836. As we can see on Figure 9, they all are located on the bottom part of a nose, as the most yellow ones.

5 Models

All images illustrating data separation sectors by models are created using Python library `mlxtend` and its function `plotting.plot_decision_regions`. It is very useful to get impression of model structure, as we use two features and data can be displayed as 2D plot. The initial data consisting of two chosen features looks as shown on Figure 16.

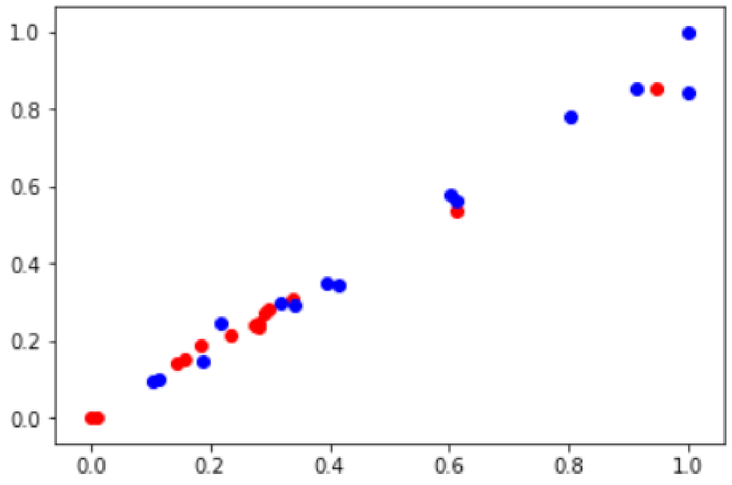


Figure 16. All training data

To choose the best hyperparameters was used function `model_selection.GridSearchCV` from package `sklearn.import`. It iterates over all possible combinations of given parameters and selects the one with the highest score.

Cross validation of model is used to get average accuracy of a predictor. The algorithm separates all the data for *k* folds and sequentially uses every one of them as a test set with all others combined as a training set. As a result we get *k* results and take the mean. Here is used 7 folds, because the number of samples is 28 and it gives us equal folds of 4 entries each. There is not so many samples in total, so this is just fine.

```
def validateClfAndDraw(clf):
    clf.fit(x,y)
    scores = cross_val_score(clf, x, y, cv=7, scoring='accuracy')

    print('k folds accuracies: ',scores)
    print('mean: ',scores.mean())
    if x.shape[1] == 2:
        plot_decision_regions(X=x.values,
                              y=y,
                              clf=clf,
                              legend=2)
```

Figure 17. Classificator validation function

Below are the outputs of function shown on Figure 17 for different classificators, they are basically completed models to use for predicting new samples.

5.1 KNN

```
k folds accuracies:  [0.75 0.25 0.5  0.5  1.   0.75 0.75]
mean:  0.6428571428571429
```

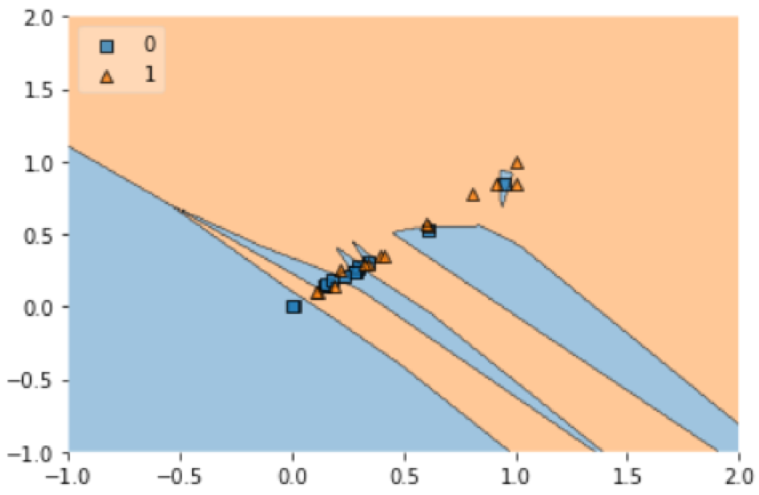


Figure 18. KNN model

Quite good result, basing only on nearest data points.

As we can see, cross validation strategy saves us from cases of dataset being not so well divided to train and test data, by using different combinations of them. Some folds give 100% accuracy and some give only 25%.

5.2 K-means

```
k folds accuracies:  [0.75 0.75 0.   1.   0.5  0.75 0.75]
mean:  0.6428571428571429
```

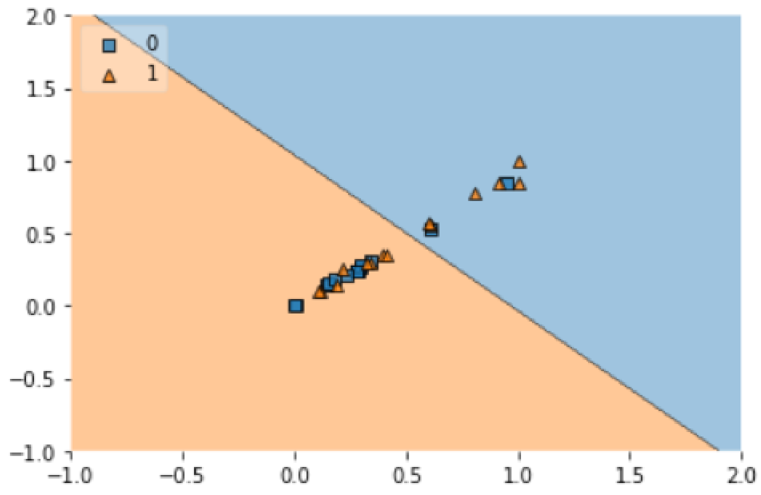


Figure 19. K-means model

Quite straightforward solution by separating dataset in half. Kind of accurate, but as the data is not very well grouped to separate clusters, this model doesn't seem very good.

5.3 SVM

```
k folds accuracies:  [0.75 1.   0.5  0.5  0.5  0.75 0.5 ]
mean:  0.6428571428571429
```

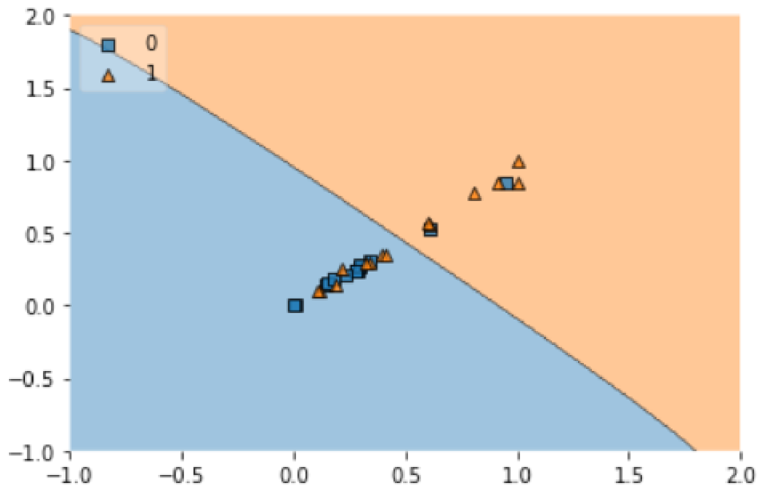


Figure 20. SVM model

Looks similar to k-means model because of the algorithm mechanism.

5.4 Decision tree

```
k folds accuracies:  [0.75 0.5  0.5  0.75 1.   0.75 0.75]
mean:  0.7142857142857143
```

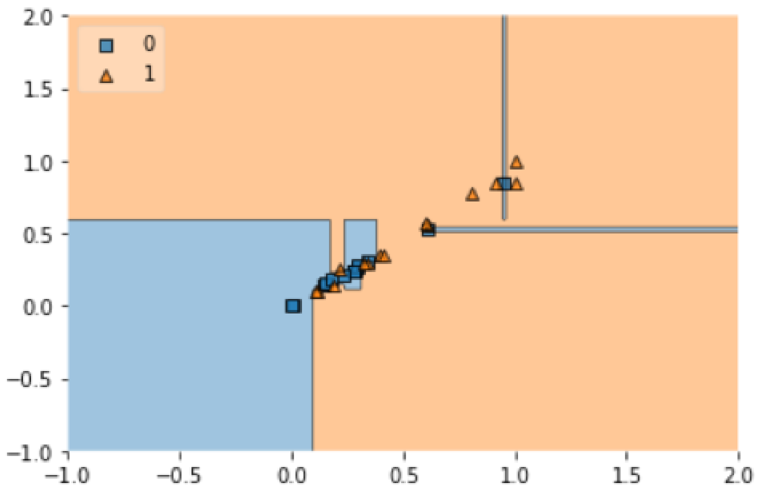


Figure 21. Decision tree model

Good result because of good and universal heuristic designed in decision tree algorithm.

5.5 Random forest

```
k folds accuracies:  [0.75 0.5  0.5  0.75 1.   0.75 0.75]
mean:  0.7142857142857143
```

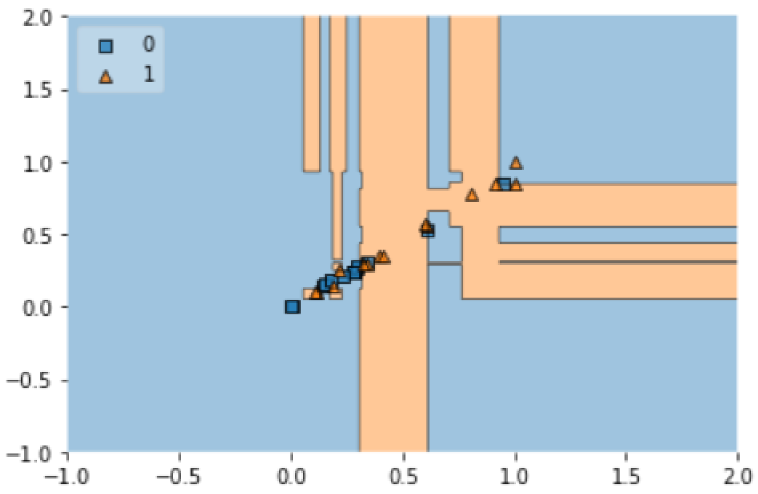


Figure 22. Random forest model

Multiple decision trees show the same result as a single one, no improvements here. Probably because the most successful ones look very similar to the one made separately.

5.6 Boosting

```
k folds accuracies:  [0.75 0.75 0.5  0.75 1.   0.75 0.75]
mean:  0.75
```

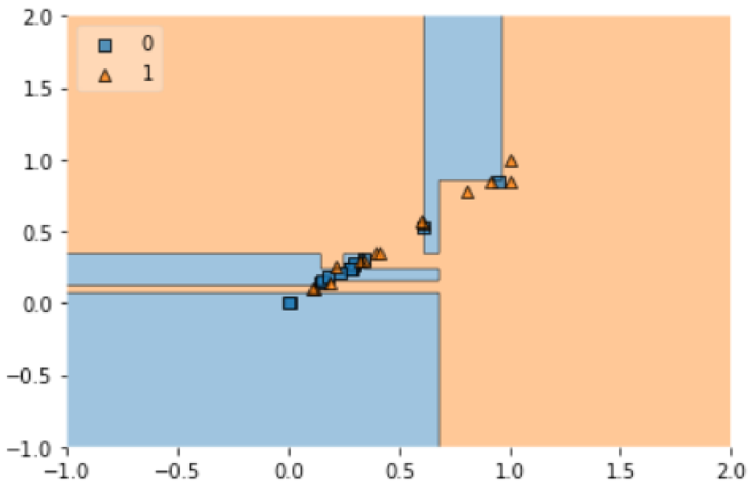


Figure 23. Adaptive boosting model

For the boosting model, previously made decision tree and random forest were taken as a base classificator, and as the best result, the decision tree was boosted to get accuracy of 75%.

6 Results

As a result we got good decision tree and random forest models with accuracy of 71.4%, and with adaptive boosting improved one for even better 75%. This is quite good achievement, if we consider small number of training samples, only 14 pairs. Also, technologies used are not state-of-the-art, because currently already exist very complex multilayered neural networks, capable of finding deep correlations in data.

Nevertheless, the goal was to explore the possibility of detecting if individual is speaking their native language or not, based only on facial movements data, and the result is positive. With built models, in 3 cases of 4 we can detect if the language being spoken is native or not.

References

[1] Towards Data Science. Introduction to Machine Learning [Online] https://towardsdatascience.com/introduction-to-machine-learning-db7c668822c4 (Accessed 21.05.2018)

[2] Towards Data Science. Understanding Feature Engineering [Online] https://towardsdatascience.com/understanding-feature-engineering-part-1-continuous-numeric-data-da4e47099a7b (Accessed 21.05.2018)

[3] Medium. What is Feature Engineering [Online] https://medium.com/mindorks/what-is-feature-engineering-for-machine-learning-d8ba3158d97a (Accessed 21.05.2018)

[4] Towards Data Science. Why How and When to Apply Feature Selection [Online] https://towardsdatascience.com/why-how-and-when-to-apply-feature-selection-e9c69adfabf2 (Accessed 21.05.2018)

[5] Machine Learning Mastery. An Introduction to Feature Selection [Online]

https://machinelearningmastery.com/an-introduction-to-feature-selection/ (Accessed 21.05.2018)

[6] Generalized Fisher Score for Feature Selection [Online] https://arxiv.org/ftp/arxiv/papers/1202/1202.3725.pdf (Accessed 21.05.2018)

[7] Matatat. Hypothesis testing [Online] http://matatat.org/p-values-statistical-testing.html (Accessed 21.05.2018)

[8] STHDA. T-test [Online] http://www.sthda.com/english/wiki/t-test-formula (Accessed 21.05.2018)

[9] The Minitab Blog. What Are T Values and P Values in Statistics [Online]

http://blog.minitab.com/blog/statistics-and-quality-data-analysis/what-are-t-values-and-p-values-in-statistics (Accessed 21.05.2018)

[10] Towards Data Science. Introduction to Machine Learnign [Online] https://towardsdatascience.com/decision-trees-understanding-explainable-ai-620fc37e598d (Accessed 21.05.2018)

[11] Medium. Random Forest Simple Explanation [Online]

https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d (Accessed 21.05.2018)

[12] Bigdata Made Simple. Possibly the simplest way to explain K-Means algorithm [Online]

http://bigdata-madesimple.com/possibly-the-simplest-way-to-explain-k-means-algorithm/ (Accessed 21.05.2018)

[13] Medium. A Quick Introduction to K-Nearest Neighbors Algorithm [Online]

https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7 (Accessed 21.05.2018)

[14] KD nuggets. Support Vector Machines [Online]

https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html (Accessed 21.05.2018)

[15] Abaytics Vidhya. Quick Introduction to Boosting Algorithms in Machine Learning [Online] https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/ (Accessed 21.05.2018)

[16] Microsoft. Kinect for Windows [Online] https://developer.microsoft.com/en-us/windows/kinect (Accessed 21.05.2018)

[17] Microsoft. Microsoft.Kinect Namespace [Online] https://msdn.microsoft.com/en-us/library/microsoft.kinect.aspx (Accessed 21.05.2018)

[18] Microsoft. Introduction to WPF [Online] https://docs.microsoft.com/en-us/visualstudio/designers/introduction-to-wpf (Accessed 21.05.2018)

[19] Pandas [Online] https://pandas.pydata.org/ (Accessed 21.05.2018)

[20] Scikit-learn. Machine Learning in Python [Online] http://scikit-learn.org/stable/ (Accessed 21.05.2018)

[21] Matplotlib [Online] https://matplotlib.org/ (Accessed 21.05.2018)

[22] Numpy [Online] http://www.numpy.org/ (Accessed 21.05.2018)

[23] Github. rasbt/**mlxtend** repository [Online] https://github.com/rasbt/mlxtend (Accessed 21.05.2018)

[24] Jupiter [Online] http://jupyter.org/ (Accessed 21.05.2018)

[25] Microsoft. Kinect for Windows SDK 2.0 [Online] https://www.microsoft.com/en-us/download/details.aspx?id=44561 (Accessed 21.05.2018)

[26] Vangos Pterneas. How to use Kinect HD Face [Online] https://pterneas.com/2015/06/06/kinect-hd-face/ (Accessed 21.05.2018)

[27] SciPy.org [Online] https://www.scipy.org/ (Accessed 21.05.2018)