Capstone Project - Machine Learning Engineer Nanodegree

SUBJECT IDENTIFICATION BY EYES MOVEMENT: A MACHINE LEARNING APPROACH

Guilherme Lima
August 16th, 2019

Definition

Project Overview

Human identification has always been a very important problem in our world. Not only for knowing if a person is who he says he is (authentication problem), but also for identifying who that person is (recognition problem). This can be seen in several areas as financial services, security & safety, health care, telecommunication, etc. when referred to giving someone access to resources, personalized services or individual care. [1][2] Once biometrics identification apply technology to accurately identify a person by his intrinsic physical or behavioral characteristics [3], it has been increasingly used for human identification.

In this project I used machine learning techniques to develop an identification method based on eyes movement data, as proposed by Kasprowski and Komogortsev in the First Eye Movements' Verification and Identification Competition in the IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS 2012) [5]. All data provided for the competition was collected by Dr. Paweł Kasprowski at Silesian University of Technology, Poland [6] and consists as samples from 37 subjects, each sample consisting in 2048 measures, obtained through the following process [5]:

> *Step stimulus was presented in a form of a jumping dot interpolated on the 3x3 grid. The stimulus consisted of eleven dot position changes giving twelve consecutive dot positions. Subjects were given instructions to follow the dot. First dot appeared in the middle of the screen. After 1600 ms the dot in the middle disappeared and for 20 ms a screen was blank. Subsequently, a dot appeared in the upper right corner. The sequence continued until all locations of the 3x3 grid were visited. Dot movements on the grid were interspersed with dots presented at the central screen location.*

Problem Statement

The competition proposed in BTAS 2012 employed four datasets: A, B, C and D. Dataset A was used for the main competition (http://www.kasprowski.pl/emvic2012) and also for an additional competition on Kaggle web service (http://www.kaggle.com/c/emvic) [5].

This project is focused on solving the problem employing dataset A: determining how people may be identified based on their eyes' movement characteristics using a classification model, i.e., to predict the subject (class). Development went through the following workflow:

- Data exploration in order to get a better understanding of data provided before using it.
- Data preprocessing for feature extraction and dimensionality reduction.
- Implementation of the machine learning model with basic setup.
- Refinement in order to refine model and improve its performance.

Metrics

Once the dataset provided is imbalanced, i.e., classes are not represented equally, the traditional accuracy metric is not the best choice as performance metric as it can be misleading due to accuracy paradox [8].

In this project model performance was evaluated using F1-Score. It can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal [9]. Formula:

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

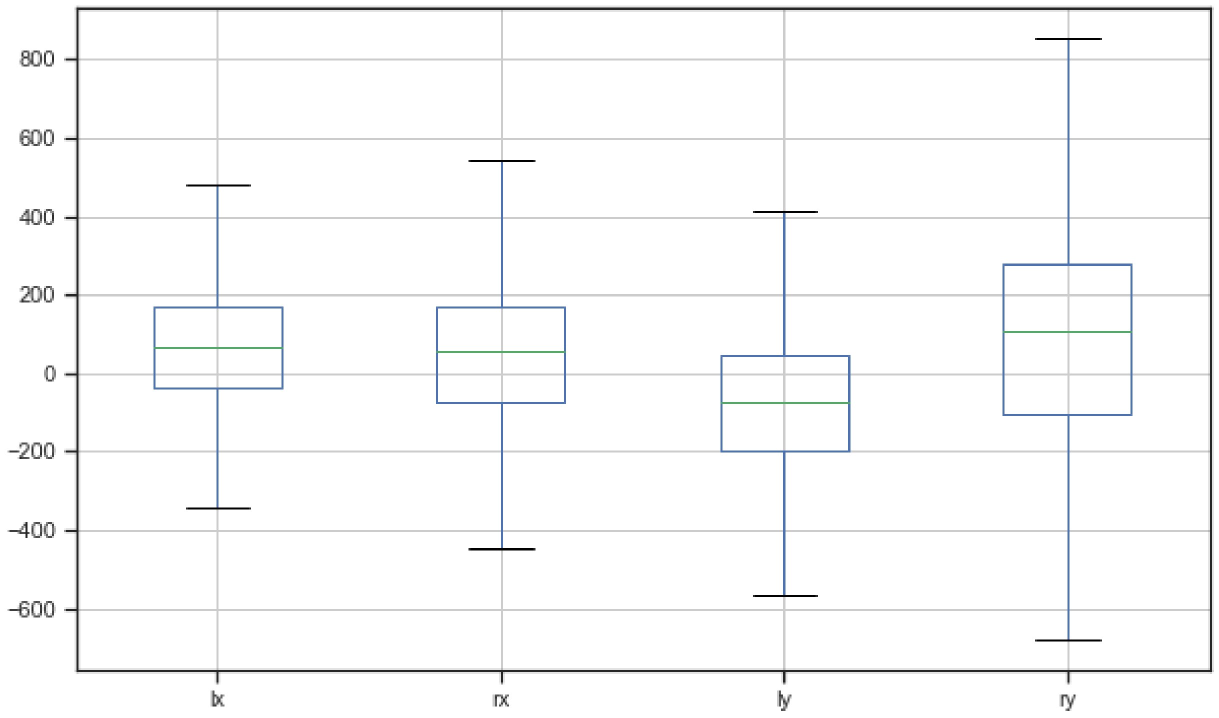
Accuracy metric was used only as a support metric, for it was the official metric employed by BTAS 2012 competition [5].

Analysis

Data Exploration

Dataset was provided in two parts, already cleansed and without any missing values: one to be used as the training set, containing 66,7% of data (652 observations) and the other as the test set containing 33,3% of data (326 observations). Distribution of values for each eye for X and Y axis shows something interesting (Figure 1). As someone would expect, both left and right eyes have similar distributions for axis X. But for axis Y, distributions differ: it's like the right eye movement in Y axis has a wider range.

Figure 1 – Eyes position values



Stimulus position data are not subject related, since they only describe the experiment and these columns were discarded in this implementation. A total of 8192 data columns remained, so all these values must be converted into a set of features, giving us information about the subject.

Both datasets have the following structure:

Table 1- Datasets structure

Column	**Description**
sid	subject identifier (in textual format: aXX)
sx	list of 2048 tab separated values of stimulus point placements on X axis
sy	list of 2048 tab separated values of stimulus point placements on Y axis
lx	list of 2048 tab separated values of left eye gaze points on X axis

Column	Description
ly	list of 2048 tab separated values of left eye gaze points on Y axis
rx	list of 2048 tab separated values of right eye gaze points on X axis
ry	list of 2048 tab separated values of right eye gaze points on Y axis

Exploratory Visualization

Visualization of observations for different subjects (figures 1 and 2) shows us how each individual respond to the same stimulus. Our model must capture these differences.

Figure 2 - Eye coordinates subject 17

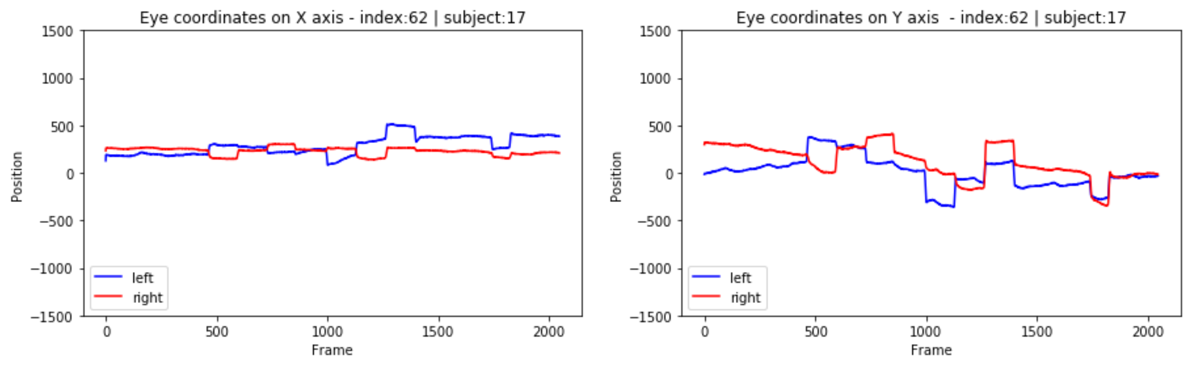
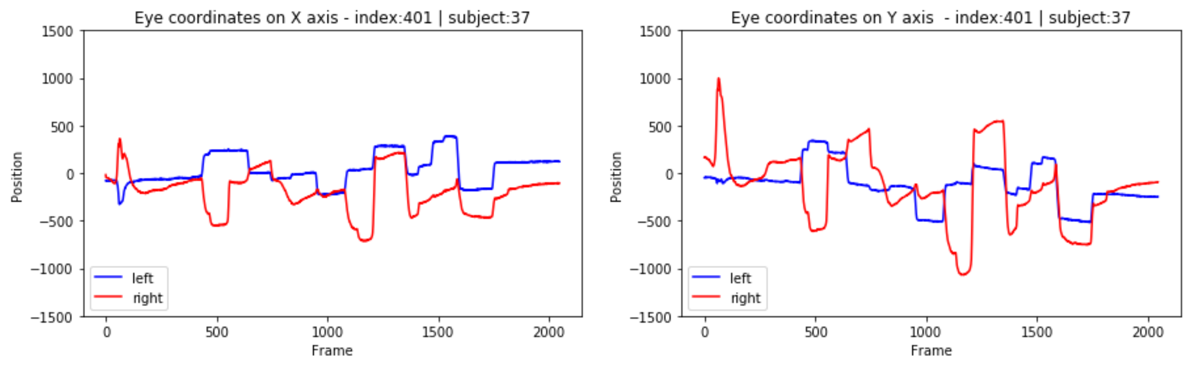
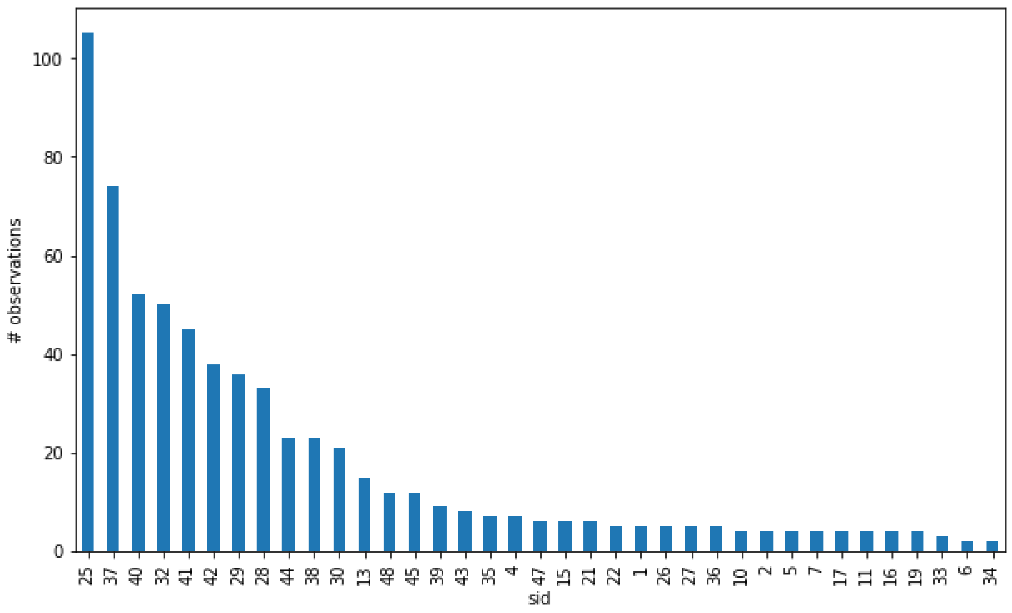


Figure 3 - Eye coordinates subject 37



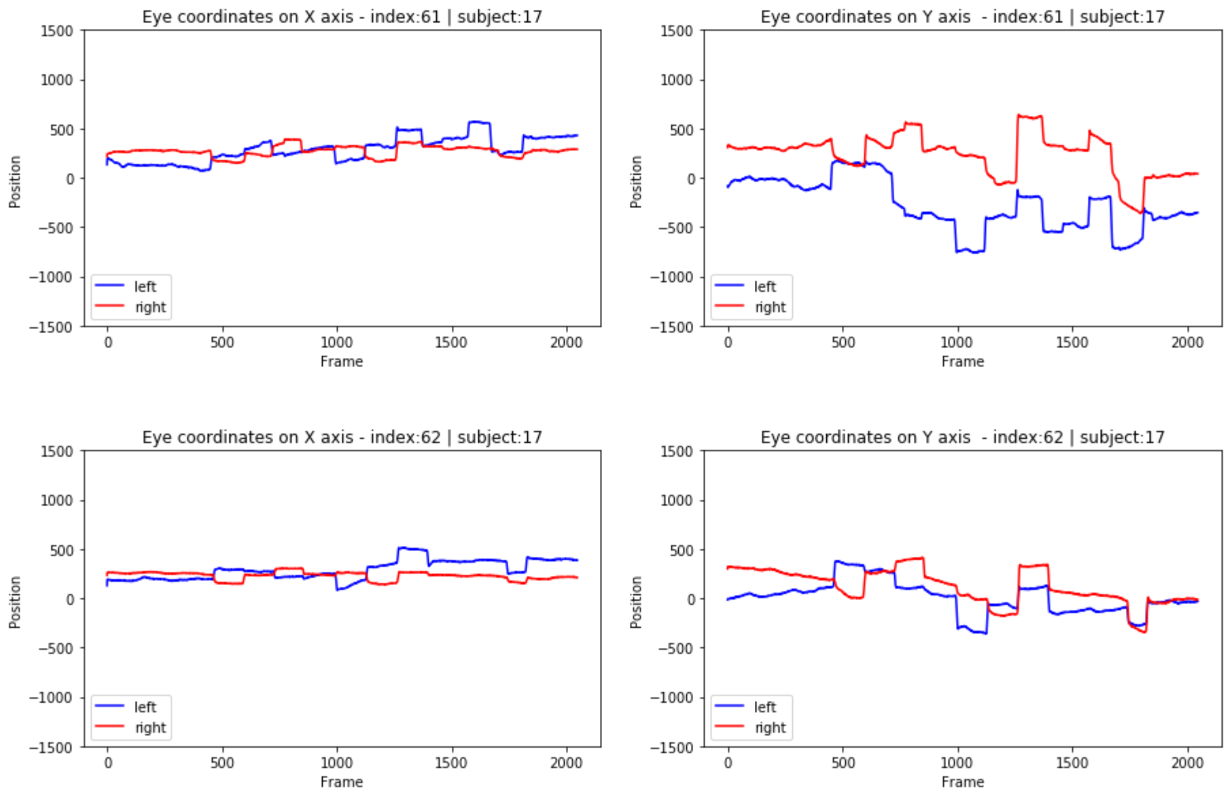
Inspecting frequency for each class in training set (Figure 3) we can see that classes are not represented equally, so our data is unbalanced.

Figure 4- Class frequency



When analyzing different observations for the same subject, we see clearly that even the same subject produces different values for exactly the same experiment (Figure 4). Our model must extract a set of features which values are as similar as possible for different observations of the same class and with values as different as possible for a different class [4].

Figure 5 - Eye coordinates subject 17

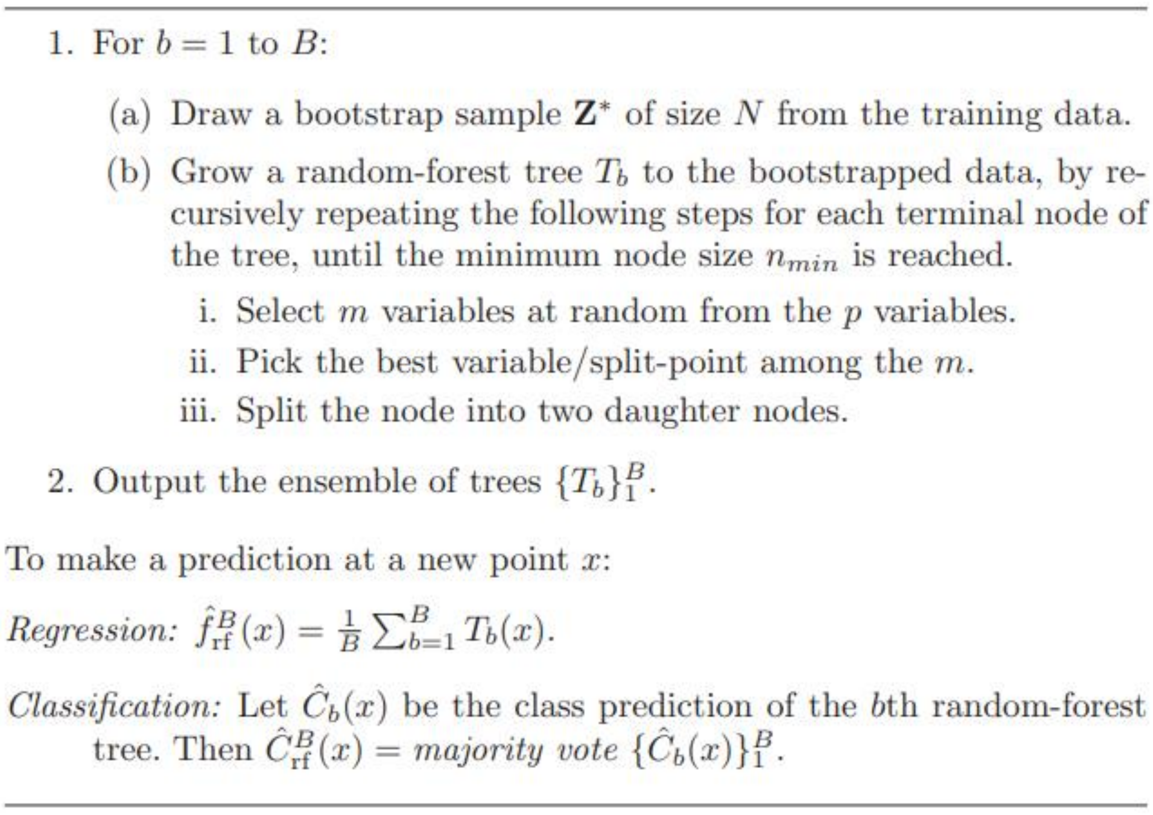


Algorithms and Techniques

There are several machine learning algorithms to use in such a multiclassification problem. My choice was the Random Forest, due to his great accuracy. Random Forest is a modification of bagging algorithm, but simpler to train and tune (the essential idea in bagging is to average many noisy but approximately unbiased models, and hence reduce the variance) [13].

Random Forest builds a large collection of de-correlated trees, and then averages them. It uses a random sample of m predictors as split candidates from all p predictors at each split. Algorithm is shown in Figure 6.

Figure 6 - Random Forest Algorithm [13]



```
1. For $b = 1$ to $B$:
    (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$ from the training data.
    (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by re-
        cursively repeating the following steps for each terminal node of
        the tree, until the minimum node size $n_{min}$ is reached.
          i. Select $m$ variables at random from the $p$ variables.
         ii. Pick the best variable/split-point among the $m$.
        iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point $x$:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the $b$th random-forest
    tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.
```

Also, it runs efficiently on large data bases and generates an internal unbiased estimate of the generalization error as the forest building progresses. It also decorrelates the trees, i.e., brings more reliable results, it's relatively robust to outliers and noise hard to overfit [10][11]. Also, decision trees tend to perform well on imbalanced datasets, since the splitting rules can force minority classes to be addressed.

Benchmark

Kaggle project administrators made available code for benchmarks for the competition (https://github.com/benhamner/emvic): one for random forest implementation, one for support vector machine and one uniform. I used as benchmark random forest and support vector machine implementations provided.

For both, the implementation provided runs a basic model, without any data preprocessing, with only the following parameters:

- Random Forest:

```
RandomForestClassifier(n_estimators=100, min_samples_split=2)
```

- Support Vector Machine:

```
svm.SVC(probability=True)
```

These codes were reproduced and evaluated with our metrics defined, yielding the results in test set as shown in Table 2.

Table 2 - Benchmark score (Test set)

Model	**F1 Score**	**Accuracy**
Random Forest – Kaggle benchmark	0.617	0.675
Support Vector Machine – Kaggle benchmark	0.067	0.174

It is important to notice that benchmarks performance may be affected by the absence of data normalization, especially support vector machine.

Methodology

Data Preprocessing

Since we have x and y positions for left eye, as also for right eye, in each experiment are collected 2048 x 4 = 8192 integers. This brings two main considerations to keep in mind: feature extraction and dimensionality reduction. With that in mind, our data (both training and test sets) was preprocessed in the following steps (for each observation):

- Grouping values for each eye signal (left eye X, left eye Y, right eye X, right eye Y).
- Data segmentation into equally sized subsets

- Creating features summarized for each segment
- Data normalization using a scaler (details in Implementation section)

Implementation

A utility file was implemented to help with preprocessing routines. The implementation went into these steps:

Preprocessing:

- Left eye X coordinates where grouped as an array *lx* of shape (*num observations*, 2048)
- Left eye Y coordinates where grouped as an array *ly* of shape (*num observations*, 2048)
- Right eye X coordinates where grouped as an array *rx* of shape (*num observations*, 2048)
- Right eye Y coordinates where grouped as an array *ry* of shape (*num observations*, 2048)
- Each array was divided into multiple sub-arrays. I used 5-fold cross-validation and a classifier with default parameters to identify a reasonable number of segments in 2 steps: one for a wider range, the second one a zoom to refine the number of splits (Figure 5 and Figure 6). Final number of splits: 14.

Figure 7 - Segmentation search – Step 1

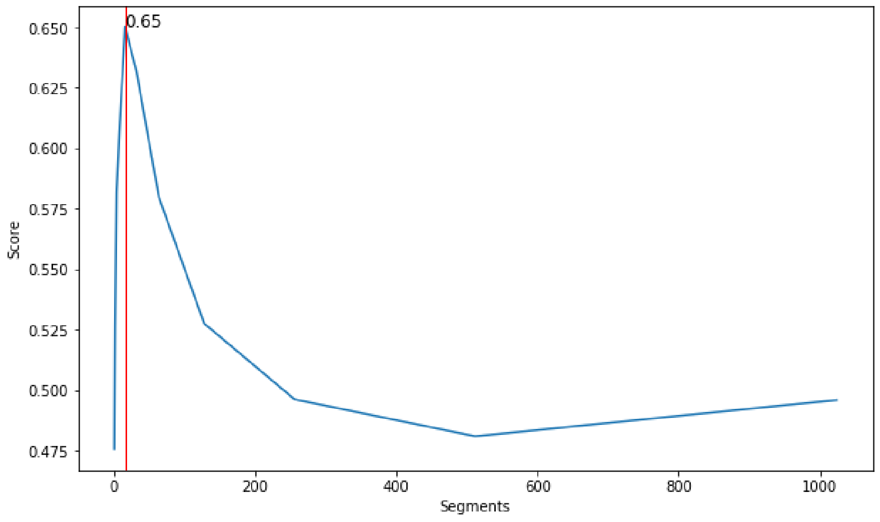
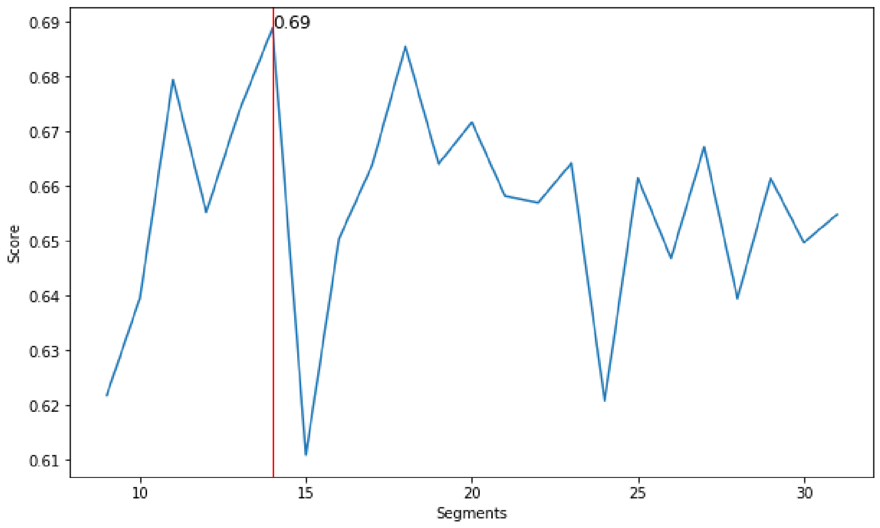


Figure 8 - Segmentation search – Step 2



- Each segment was then summarized into features, averaging values of position, velocity (derivative of first order from position) and acceleration (derivative of second order from position).
- Data normalization: features were standardized by removing the mean and scaling to unit variance using StandardScaler from scikit-learn. The scaler was fitted on training data and used to scale both training and test data.

Modeling:

- Model was trained on preprocessed data using a Random Forest classifier with 100 estimators and minimum number of samples required to split an internal node equal to 2.
- With predicted classes for the test set then scores were computed. Results are shown in Table 3.

Table 3 – Initial solution results (Test set)

Model	F1 Score	Accuracy
Random Forest (version 1)	0.791	0.825

During the coding process I've encountered no major complications.

Refinement

A refinement process was implemented in two steps to find a combination of parameters that performs better than scikit-learn defaults.

Steps:

1. Random Search (using RandomSearchCV from scikit-learn) over the following parameters:
    a. n_estimators: [10, 100, 1000, 2000, 10000]
    b. min_samples_split: [2,3,4]
2. Based on step 1 results, min_samples_split was set equal to 2 and max_depth equal to None. Also, Gini measure of impurity tends to isolate most frequent classes, while Entropy tends to produce slightly more balanced trees. So, I decided to include criterion parameter in this step. Then a grid search was conducted over the following parameters:
    a. n_estimators: [1940, 1941, 1942, …, 2058, 2059, 2060]
    b. criterion: ['gini', 'entropy']

Results

Model Evaluation and Validation

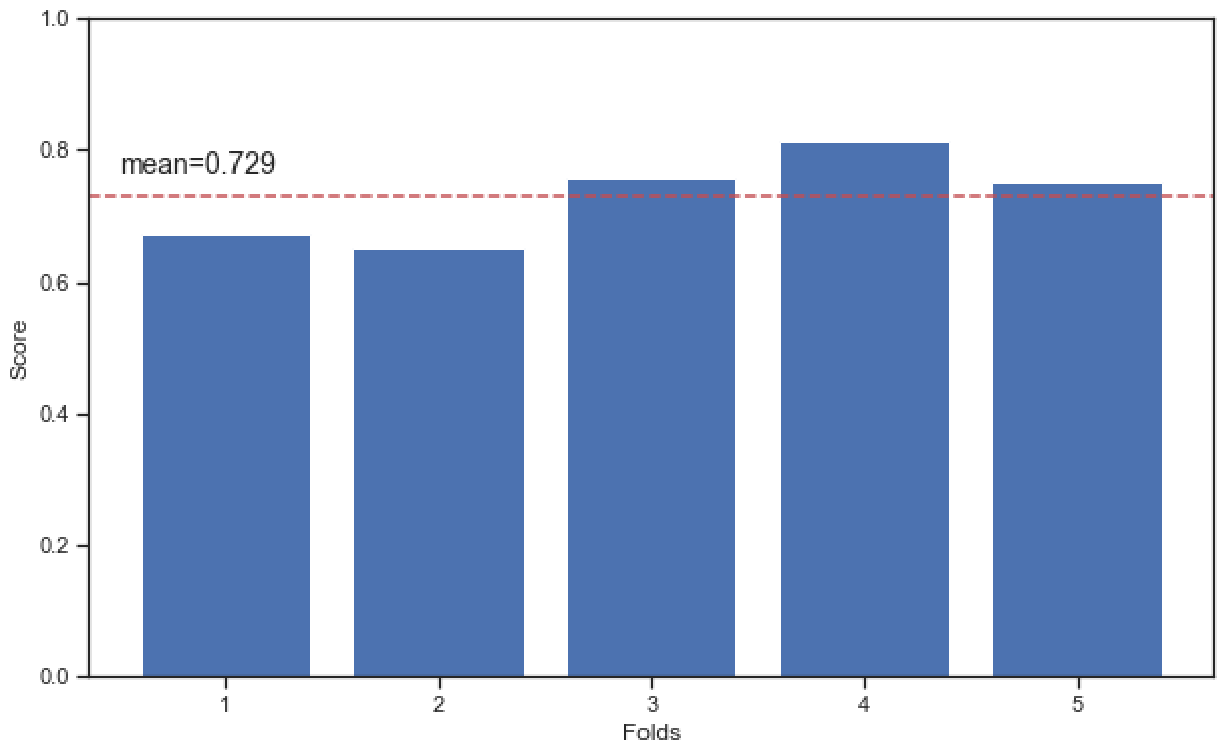
The final model after refinement process has the following parameters:

> *'bootstrap': True, 'class_weight': None, 'criterion': 'entropy', 'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 2059, 'n_jobs': None, 'oob_score': False, 'random_state': 7, 'verbose': 0, 'warm_start': False*

There is no extra data provided to test the model with new observations. However, results are still solid when tested over a random sample of 75% of the test set (results in Table 4).

Additionally, analyzing the cross-validation data for the best model it performed equally well across each individual validation fold, as shown in Figure 9.

Figure 9 - Model performance across each individual validation fold



Justification

The initial model already performed better than benchmarks provided by Kaggle in respect to all metrics. With refinement, the final model achieved yet better results as shown in Table 4. As already mentioned, results in a random sample of the test set also outperforms benchmarks.

Table 4 - Final results (Test set)

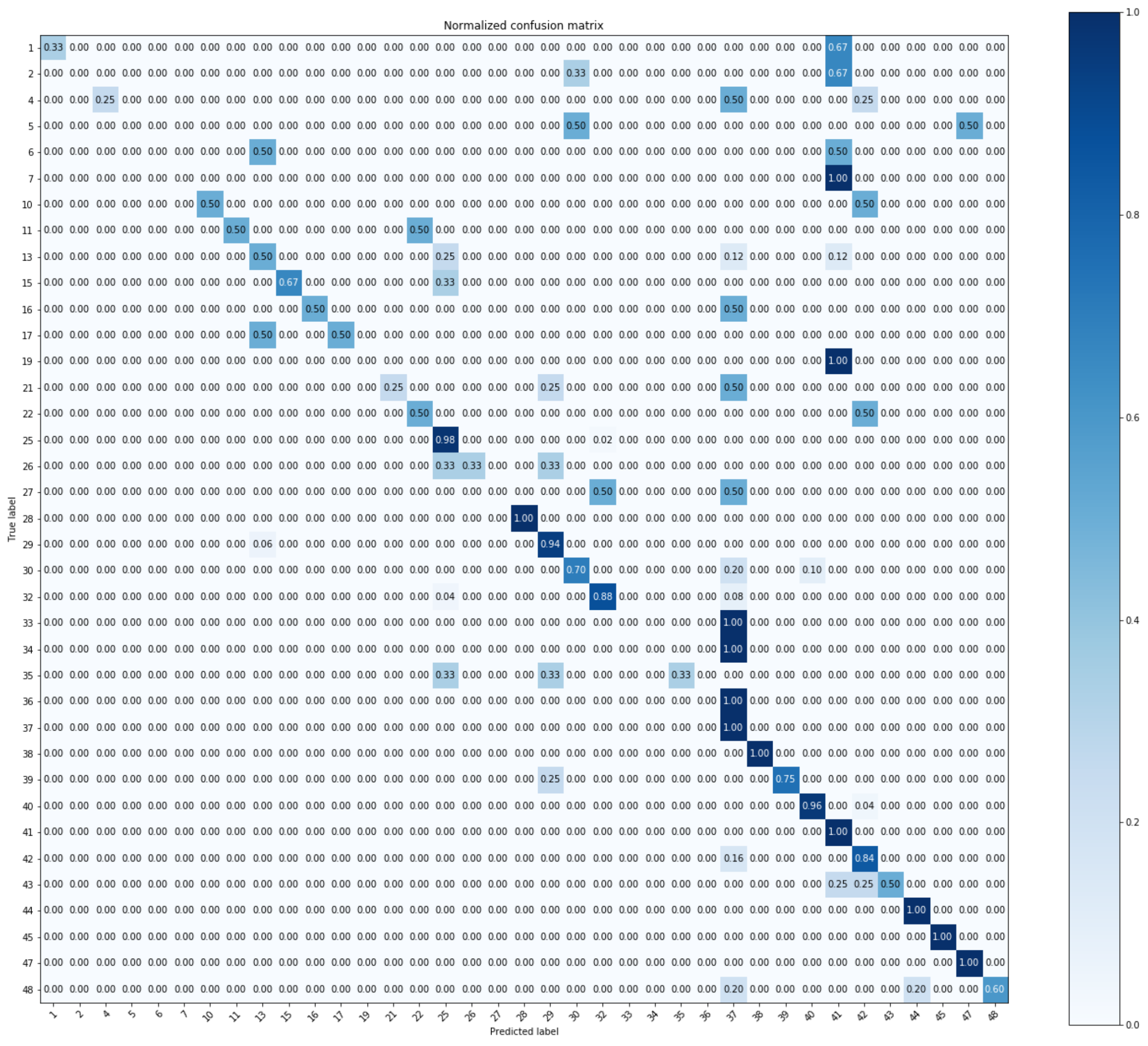
Model	**F1 Score**	**Accuracy**
Random Forest – Kaggle benchmark	0.617	0.675
Support Vector Machine – Kaggle benchmark	0.067	0.174
Random Forest (initial)	0.791	0.825
Random Forest (Final)	0.794	0.825
Random Forest (Final – test sample 75%)	0.759	0.799

Conclusion

Free-Form Visualization

The confusion matrix (Figure 7) for the final model shows us that classes with low number of observations (e.g. 16, 17, 11 and 15) have good scores. This is interesting, thus the model is performing well even for some imbalanced classes.

Figure 10- Confusion Matrix Final Model



Reflection

The process used in this project can be summarized as:

- Choosing an interesting and relevant problem at Kaggle.
- Obtaining data provided.
- Exploring data.
- Preprocess data to better deal with large number of features and feature extraction.
- Training the model.
- Tunning the model.

Without a doubt the most difficult part was the preprocessing of data. Choosing a good technique to transform data in less, but relevant features, was challenging, specially because it was my first problem with large datasets.

The final solution fits my expectations, once it proved to be robust and also obtained a good score, specially when compared with Kaggle leaderboard, placing around top 15.

Improvement

There are further improvements that could be made. One is to process data using general signal processing techniques and identify features specific for the nature of the eye movement signal: average reaction time, average stabilization time and saccade velocity. This would generalize more the solution. Another improvement would be deal better with the imbalanced using a penalized model or generating synthetic samples using an algorithm like SMOTE (Synthetic Minority Over-sampling).

If I knew how to implement, I would try a combination of Multivariate Wald-Wolfowitz test with kNN, described by the winner of BTAS 2012 [12]. The technique described there and results are very interesting and would be a huge learning.

References

[1]Kasprowski, P. (2012), "EMVIC - Eye Movements' Verification and Identification Competition", http://www.kasprowski.pl/emvic2012/biometrics.php. [Accessed: 10-May-2019].

[2]Anil Jain, Lin Hong, and Sharath Pankanti. 2000. Biometric identification. Commun. ACM 43, 2 (February 2000), 90-98. DOI: https://doi.org/10.1145/328236.328110

[3]Rouse, M. (2019). What is biometrics? Available at: https://searchsecurity.techtarget.com/definition/biometrics [Accessed May, 10th 2019].

[4]Kasprowski, Pawel & Ober, Józef. (2004). Eye Movements in Biometrics. 248-258. 10.1007/978-3-540-25976-3_23.

[5]Kasprowski, Pawel & Komogortsev, Oleg & Karpov, Alex. (2012). First Eye Movement Verification and Identification Competition at BTAS 2012. 2012 IEEE 5th International Conference on Biometrics: Theory, Applications and Systems, BTAS 2012. 10.1109/BTAS.2012.6374577.

[6]Kasprowski, P., Ober, J. 2004. Eye Movement in Biometrics, In Proceedings of Biometric Authentication Workshop, European Conference on Computer Vision in Prague 2004, LNCS 3087, Springer-Verlag.the IEEE/IARP International Conference on Biometrics (ICB), pp. 1-8.

[7]Rabiner, L. R., Schafer, R. W.: Digital Processing of Speech Signals, Prentice Hall, Englewood Cliffs, NJ (1978)

[8]https://en.wikipedia.org/wiki/Accuracy_paradox [Accessed August, 16th 2019].

[9]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html [Accessed August, 16th 2019].

[10] Breiman, L., Cutler, A., https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm [Accessed August, 16th 2019]

[11]Breiman, L. Machine Learning (2001) 45: 5. https://doi.org/10.1023/A:1010933404324

[12] I. Rigas, G. Economou and S. Fotopoulos, "Human eye movements as a trait for biometrical identification," 2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS), Arlington, VA, 2012, pp. 217-222. doi: 10.1109/BTAS.2012.6374580

[13]T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning. Springer, 2008.