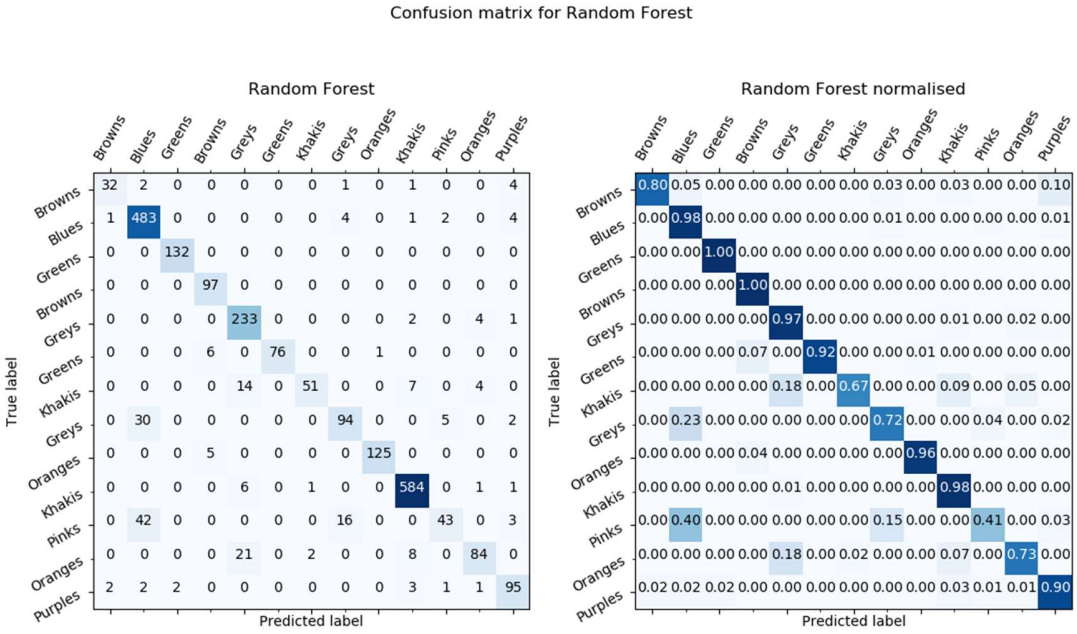
Introduction to Machine Learning HW5

Grigorii Melnikov 961152121

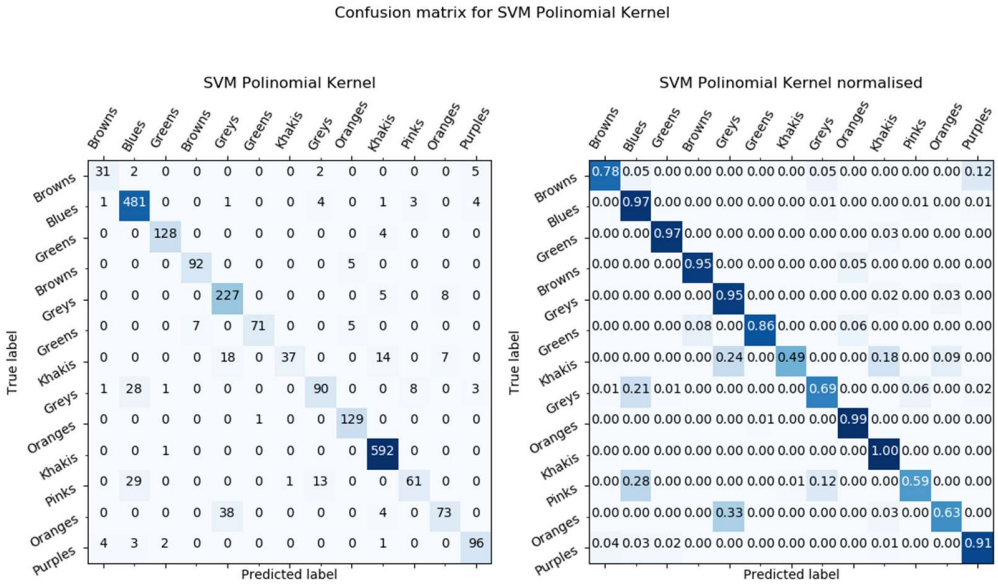
David Koplevatskii 961152089

Model Selection:

During previous assignments we came up with a several models with fair performance:

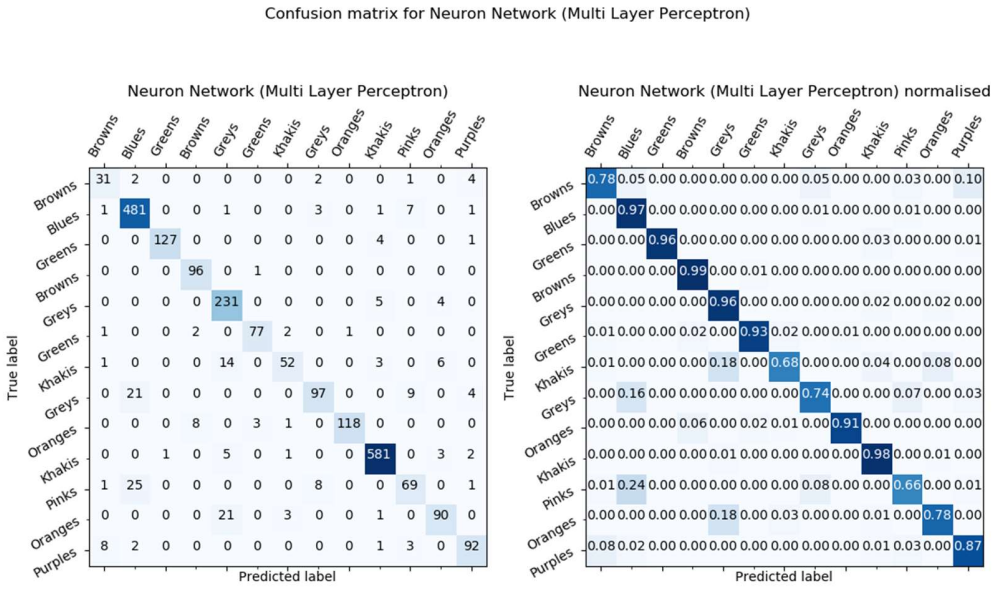


Random Forest accuracy score: 90.90520922288642 %
Random Forest f1 score: 86.99308389033517%



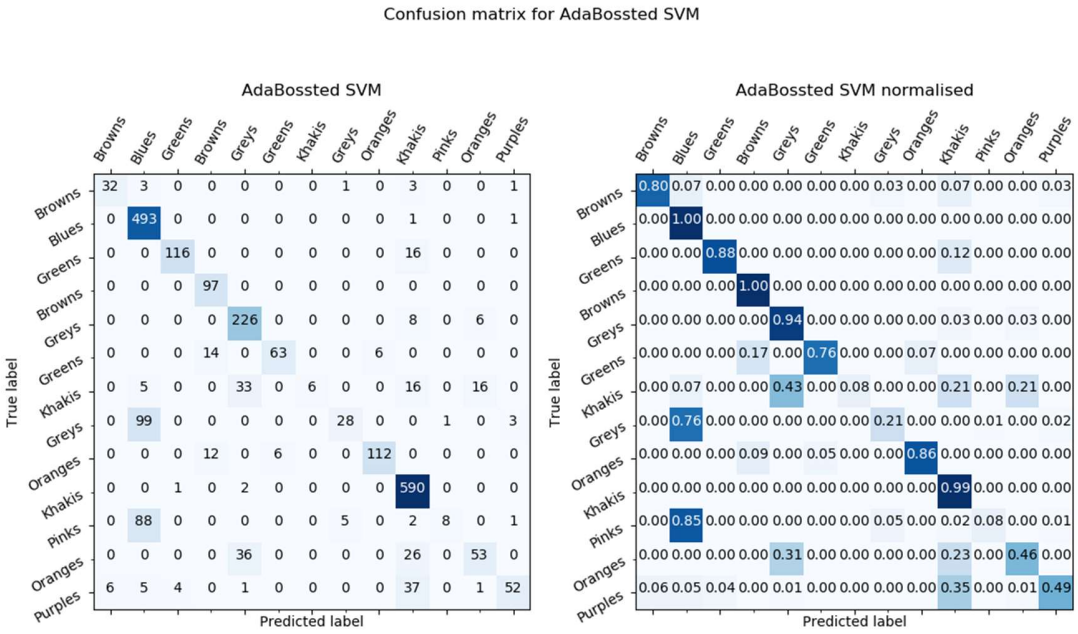
SVM Polinomial Kernel accuracy: 90.00853970964987%

SVM Polinomial Kernel f1 score: 85.12384326200587%



Neuron Network (Multi Layer Perceptron) accuracy: 91.46029035012809%

Neuron Network (Multi Layer Perceptron) f1 score: 87.39921795432171%



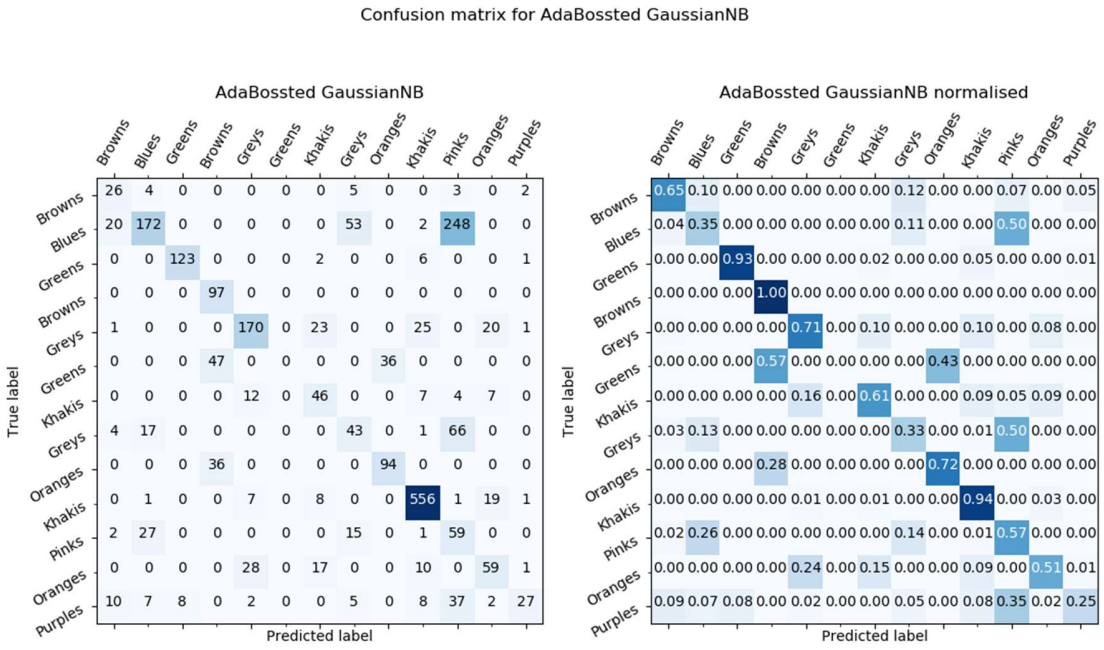
AdaBossted SVM accuracy: 0.8010247651579846

AdaBossted SVM f1 score: 0.67318280258458

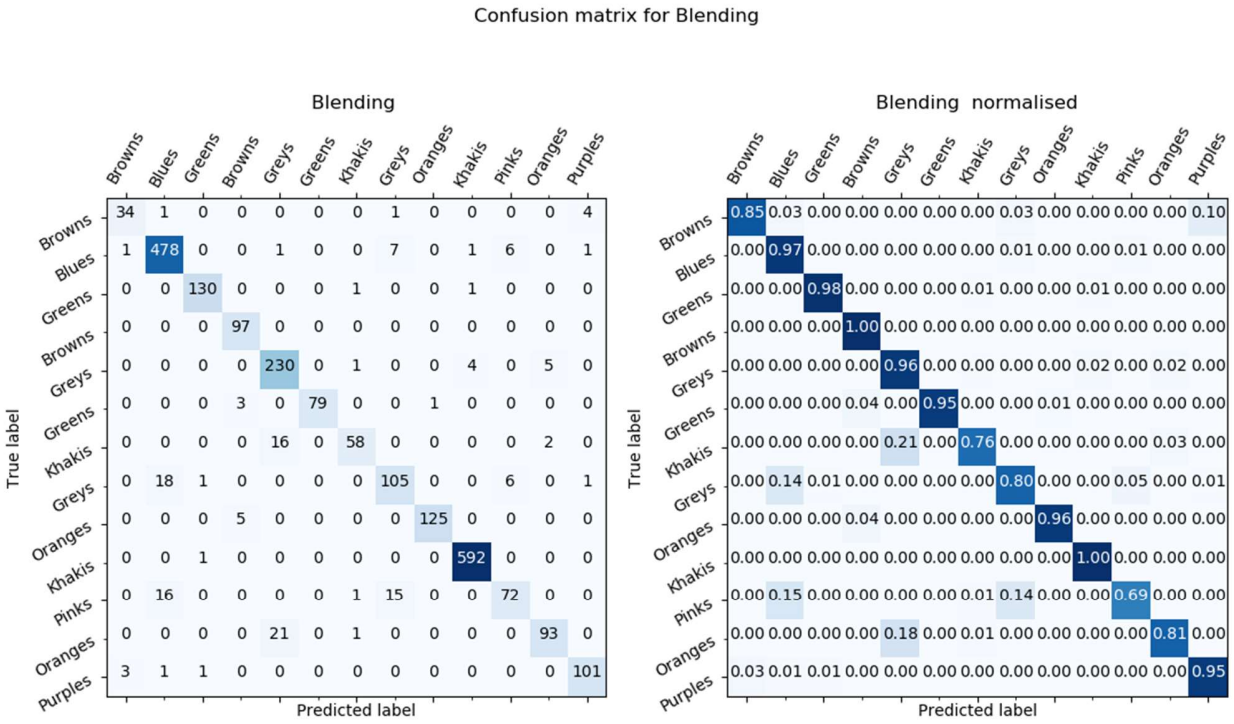
We set as a goal to achieve better f1 score, which is well suitable because some classes in training might be imbalanced in comparison with data set given to prediction. We could observe that Random Forest gets more precise prediction rate for parties like ‘Greys’ and ‘Khakis’, while SVM with polynomial kernel (and polynomial boundary consequently) has higher separation score accordingly to davies_bouldin_score and Neural Network does great job overall.

We decided to train ensemble model, which is expected to combine best aspects of already trained models.

We started with AdaBoost classifier based on a simple model with fair accuracy: simple Random Forest. Due to challenging search of suitable hyperparameters, extreme long computational time and bad overall performance, it was decided to try base estimator of different type (this time generative). We set base estimator to be GaussianNB, but again failed to obtain fair performance, so it was decided to give up with AdaBoost.



We would try combining approach, so models' predictions stored in a new dataframe and then final estimator trained over the new dataframe. Final estimator likely to be linear SVM due to low number of features and ability to improve class separation. We call it stacking or blending:



Stacking method accuracy: 94.5269410267289%

Stacking method f1 score: 92.77154702417879%

This model shows better prediction ability as we it could be seen from correlation matrix and also has better performance accordingly to f1 score. A several techniques was applied to improve predictions rate for classes ‘Pinks’ and ‘Khakis’, but none of them succeed.

```
Party with most probable majority of votes:
Turquoises : 607, 25.753075943996606%
Amount of votes per party:
Turquoises : 607, 25.753075943996606%
Browns : 516, 21.89223589308443%
Khakis : 238, 10.097581671616462%
Reds : 134, 5.685193042002545%
Greens : 132, 5.60033941450997%
Purples : 127, 5.388205345778532%
Whites : 118, 5.006364022061943%
Yellows : 106, 4.497242257106492%
Greys : 98, 4.15782774713619%
Violets : 98, 4.15782774713619%
Oranges : 79, 3.3517182859567245%
Pinks : 69, 2.927450148493848%
Blues : 35, 1.4849384811200679%
```

Coalition.

In this assignment the party distribution may change, although voting characteristic remains similar with train data, thus the coalition may be very different from the one obtained in previous assignment.

We highly appreciated how homogeneous the partition of GMM model is, so we decided to move with clustering approach. As previously, we want clusters to be relatively big to not overfit and let to clusters to describe the similarity between parties' voters based on voter characteristics.

After clusterification is done, we iterated over clusters sorted by variance in ascending order. For each cluster, party is added to coalition if that has more than 10% of its voters in the cluster. Parties in cluster determined accordingly to prediction of stacking model described from above. When coalition reaches more than *51%* search terminates.