Machine Learning – Exercise 3

Lavi.Lazarovitz – 065957383

Aharon Sharim – 052328523

# General notes

Part2.py is the data cleansing module

Part2\_model is the model selection and prediction process. The execution halts when the graphs are shown. To complete the execution, please close the graphs after you review them.

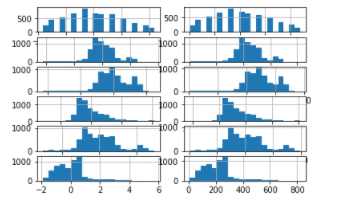
To evaluate the model I used the njobs=-1, to utilize all available processors. This might overload a machine with low resources.

# Data Cleansing

The data cleansing part is implemented in the part2.py module and in the included notebook. The module process the electionsdata.csv file according to the following steps:

* Loading data
* Splitting and writing to disk the raw version of the training, validation and tests data sets - line 17
* Dividing the different features into class dependent / independent and numeric / non numeric. This categorization is used later to decide how outliers are handled, how gaps are filled and how to scale the data - line 36
* Handling outliers – line 78
* Filling gaps – line 102
* Converting non numeric data to numeric – line 160
* Scaling the data – line 185
* Changing Vote column to label – line 230
* Filtering the features to the selected features as specified in the exercise notes – line 227
* Saving the processed data set to disk – line 243

The data cleansing process is the same we used in the previous exercise which proved to be accurate and does not remove or change the information within the data. To make sure the data still reflects the same information after the cleansing process, we compared distribution of the processed and raw data:



We took the different numeric features and plotted the histograms just as a sanity check. It looks pretty much the same.

# Choosing a model

We divided this part to three main stages:

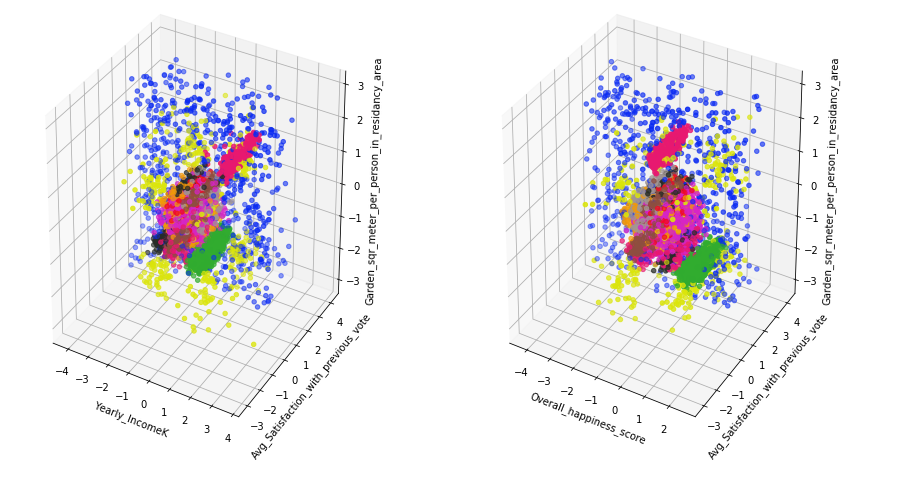
1. Trials – just trying out different models with different models and different parameters to see and feel how each model performs.
2. Examination – after the trials we zoomed in on two models that has shown similar results with the intention to choose the best for the data we have
3. Executing – Training and predicting based on the chosen model.

The process of choosing the model and using it, can be found in the part2\_model.py file and in the included notebook.

## Trials

First, as suggested in the exercise, we loaded the processed data. We loaded both the train and validation sets and combine those to one bigger set which we will later use to run stratified-k-fold training which utilizes as much information as possible to train the model (line 42).

Next, in order to see and feel the data, we plotted the numeric fields ( Number of valued Kneset members, yearly income, overall happiness, average satisfaction with previous vote, garden square meter per person and weighted education rank) with each other in a 3D graphs – overall we had 6 choose 3 options, as shown below:



* The rest of the plots can be found in the attached notebook (line 7)
* The different colors are the different parties (the color match)

Just by looking at the data we tried to think what we could learn from the data and this is what we came up with:

* The blue party seems to be all over the place and we might face high false positive rate trying to predict dots associated with the blue party or “close by” parties
* We noticed that there are set of parties that are overlap quite a lot. We feared that this fact might produce over fitting and might rule out decision trees.
* It is also obvious that the different features do seem to separate each party in pretty good…

We noted the above information and continued to evaluate the following classifiers:

* SVC – we tried it both with linear and non-linear kernels and OneVS rest option
* OneVsOne classifier – We tried this classifier as the data suggested that certain features can separate well different types of classes
* Decision trees
* KNN
* Naïve Bayes

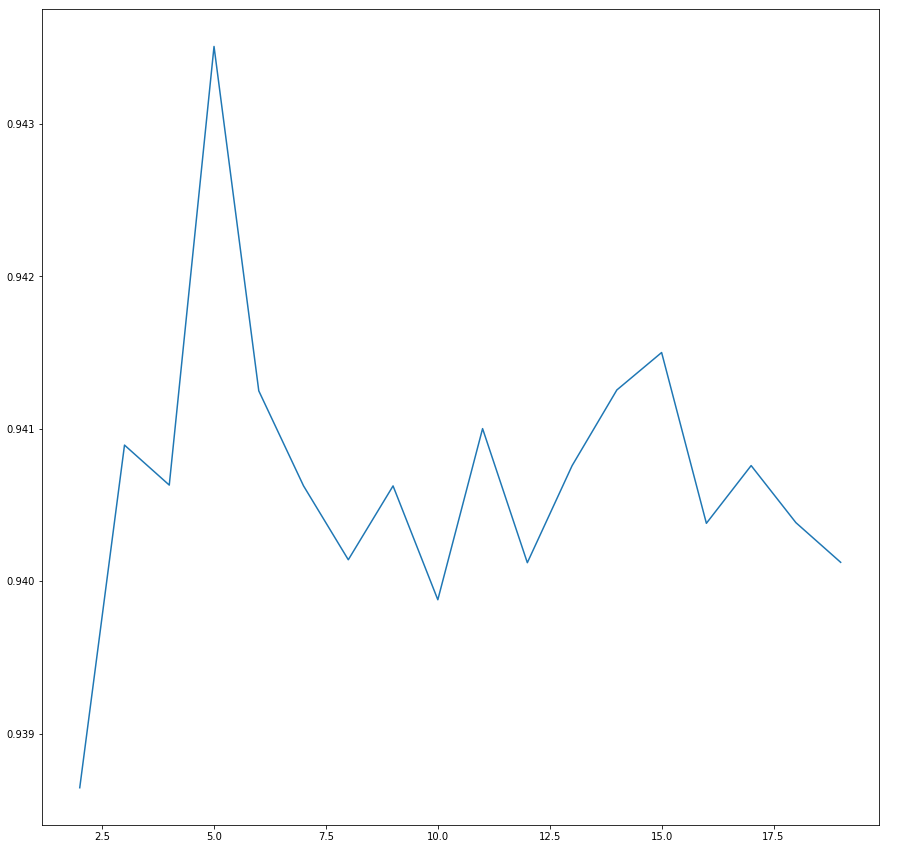
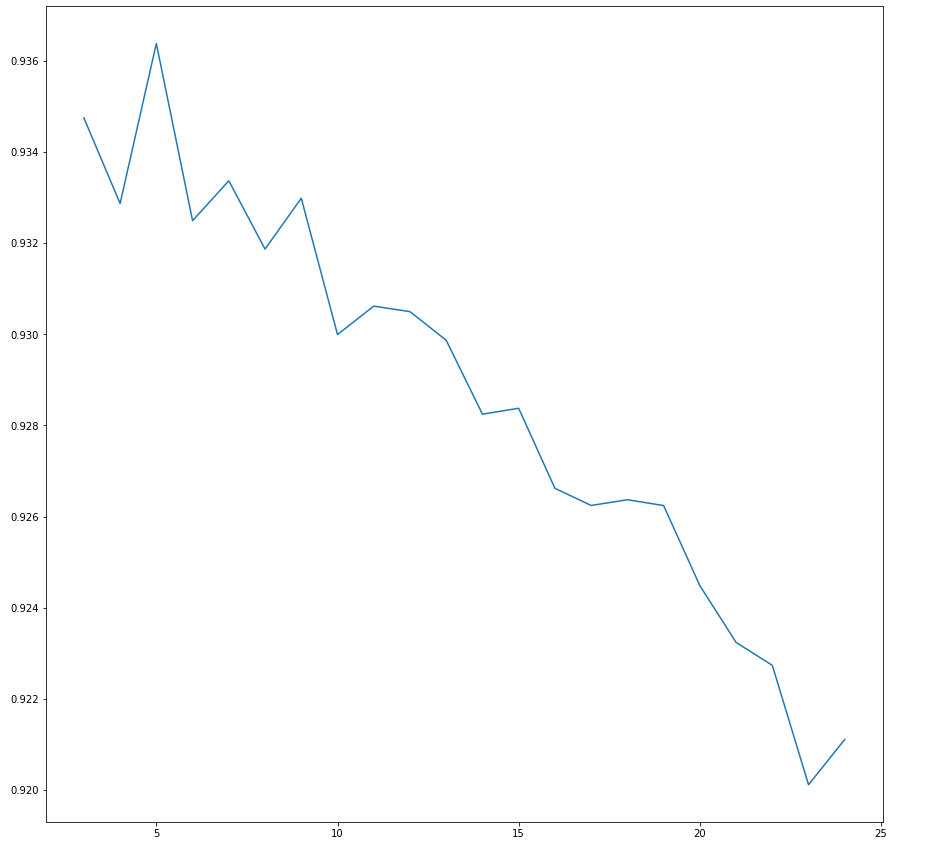
To evaluate each classifier we used cross validation scoring which uses stratified k-fold splitting, 5 times for each classifier. The cross validation allowed us to utilize larger set of data for the training while the stratified k-fold (which is default for cross validation for numeric multiclass data) took care of the bias due to class sizes. The whole process can be found in line 54 in the python code and in line 44 in the notebook. Those are the average score we got for the above classifiers:

* SVC - linear: 0.93; non-linear: 0.89; oVr: 0.91
* OvO: 0.94
* Decision tree: 0.94
* KNN: 0.93
* Naïve Bayes: 0.89

For both the decision tree and KNN we ran several executions with different parameters:

* For decision tree we have ran a range of “minimum items per split” to limit the number of splits and eventually avoid overfitting.
* For KNN we ran a range of k-neighbors to examine the optimal average score.

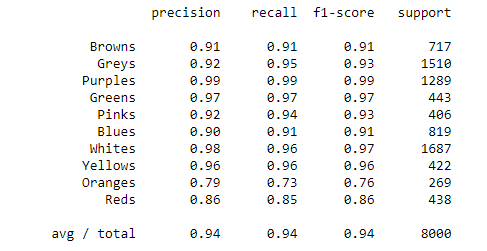
In both cases we found that about 5 samples limit for decision treens and less than 5 neighbors for KNN optimized the results. The analysis can be seen in the graph we plotted here and in the notebook – in 43 and 17. (On the left: average score per samples limit; On the right average score per number of neighbors)

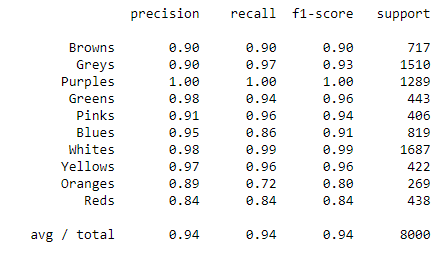
## Examination

As shown we found that OvO and decision trees (against our initial thought) had the best results so we continued to examine both. To do so, we created a classification report for each classifier with different scorings and based on the different classes. This is what we found (in python script line 121, in notebook n 74):

* For decision tree:



And for OvO:



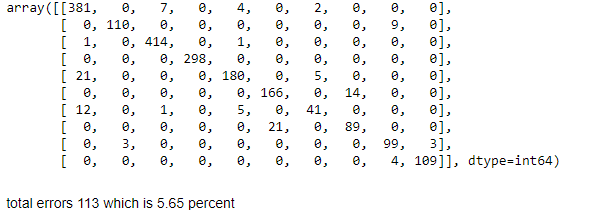
Although both seemed to be very close in almost every parameter, we noticed that OvO is better in evaluating the larger groups (Purples, Greys, and Whites). This insight might allow us to be just a bit more accurate as the larger groups compose more than 50% of the samples in the training and validation sets.

To answer the question in the exercise, we actually looked at precision, recall and f1-score to evaluate our models. Our decision to proceed with the OvO was based on the fact that it seems to predict better the larger groups which resembled more than 50% of the data.

## Execution

At last, we ran repeated stratified k-fold training on the train and validation data sets and predicted the full validation set with the expected 94.5% accuracy. The execution takes place on line 142 in the python script and in 85 in the notebook.

According to the prediction which is included in the validation-prediction.csv file there are about 5.5 percent error or 110 errors (according to our recent run using the included python module), as presented by the confusion matrix:



According to the prediction, the **Purple party** gets highest number of votes and the vote distribution is as follows:

Reds: 5.8%

Greens: 16.1%

Whites: 2.75%

Yellows: 4.8%

Greys: 5.35%

Oranges: 6.0%

Browns: 20.0%

Pinks: 8.45%

Blues: 9.5%

Purples: 21.25%