Joppe Vos
Haensel AMS
22 Februari 2019

Internship Recruitment-challenge

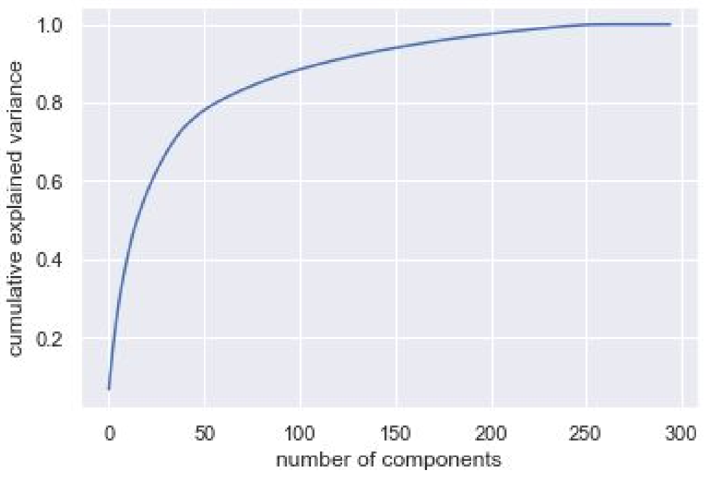
This write-up is the give some insight in the thinking process and approaches I took. I cut straight into the task and keep the code paste to a limit. For any clarification of the code, please open the attached .py files.

The given data is a set of 66k columns, all numeric except for the target. The goal is to predict the classification. There are no missing values.

```
RangeIndex: 66136 entries, 0 to 66135
Columns: 296 entries, 0 to B
dtypes: float64(4), int64(291), object(1)
```

Interesting to note is that there are no equal number of observations in each class. Of the 5 target classes, class-3 named 'C' is dominant above all. The set is imbalanced I will keep this in mind further on.

Dimension reduction: The number of dimension is high and there is a change there is some overlap and correlation in the columns. High dimensional data sets are often at risk of being sparse. Making the model prone towards overfitting and less reliable than in lower dimensions. I rescale the data beforehand such that they have the properties of a normal distribution.

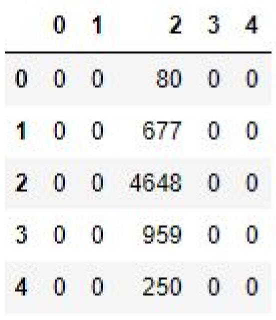


After applying PCA and plotting the component against the variance I found there are dimensions i'm able to reduce without losing much variance in the dataframe. I will wait with dropping and first to a the complete model rundown without removing. I'm interested in the difference it will make and the problems it can create.

Spot-checking: I change the target from categorical to numerical and split the data frame into a test and training set. Now I start spot-checking. The test set I will use to check after I have cross validated on the training set. I choose for a variety of algorithms to get a good starting point. A mix of linear, non-linear and a neural network. This allows me to see different models in a very short time. It gives an clear overview where to put my focus.

Below are the top 5 performing algorithms of a batch of 10. The results are cross-validating 5 folds on a random sample of 5000. Performance is taken by F1-Micro. It shows an overall good performance in predicting, but micro can hide some insights about the specific class performance.

```
SVM test: 0.7033
XGBoost test: 0.7013
RandomForestClassifier test: 0.6973
Keras NN test: 0.6968
GradientBoost test: 0.6839
```

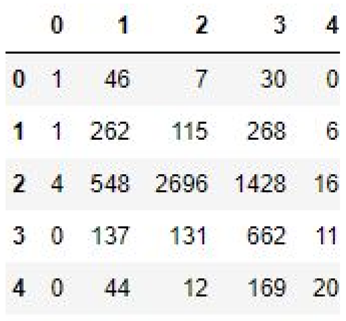


	0	1	2	3	4
0	0	0	80	0	0
1	0	0	677	0	0
2	0	0	4648	0	0
3	0	0	959	0	0
4	0	0	250	0	0

Confusion matrix: I calculated the predictions in a confusion matrix on the right. There I can clearly see that SVM is just predicting “Class-3” all the time regardless of the data asked to predict. To get the highest score.

Decision trees based algorithms perform well on imbalanced datasets. To give more value to for splits to the underrepresented classes I will make use of weights.

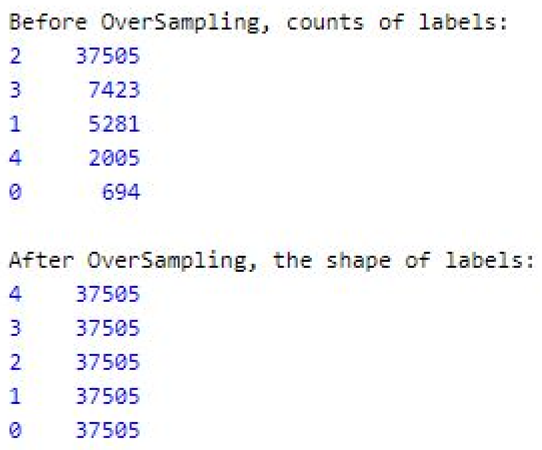
XGBoost: I will take a look at XGBoost with experimenting on using different weights on the classes. I also take the the F1-Macro now since it takes the average for each class. I do not know the goal of the prediction, but probably the prediction for each class is just as, if not more important! The baseline of XGBoost f1-macro is **0.19** and overall micro is **0.70** on the test set. By running the model multiple times with different weights assigned I try to find a balance between the two. ‘Class-0’ appears only 80 times in the dataset, but I it could be quite important. I assign the most weight to the least common. After experimenting with the weights and finding a good number I start a small grid-search on the values of interest, cross validate and predict on the test set. The overall prediction has dropped from **0.70** to **0.55.**
F1-Macro returns **0.32** crossval and **0.35** on the test set. An increase of **85**% on the individual class predictions. The confusion matrix shows the increase of class predictions compared to the first matrix from SVM, but a decrease in the overall prediction.



	0	1	2	3	4
0	1	46	7	30	0
1	1	262	115	268	6
2	4	548	2696	1428	16
3	0	137	131	662	11
4	0	44	12	169	20

To get a different view on the data I will take a different approach. I drop a lot of columns with the PCA I took in the beginning. I reduce the set to 60 features to keep 80% of the variance. I also start by using stratified splits, to make sure the distribution is around the same of the prediction classes. After done that I take a look at feature selection and see if there are columns I could drop by using RFE. Recursive feature elimination will drop each iteration a given number of features with the lowest importance. After letting that run for around 15 minutes stratified cross val it returns 48 columns. Running the same XGBoost model as before it returns me a worse score of **0.25**. The columns are important and I continue with the whole set.

Oversampling The next thing I would like to try is 'over-sampling'. If I'm able to create more data points on the less common classes the model would be able to train better on those specific classes. SMOTE is an algorithm that creates minority points between existing points. On the right you could see the number of appearances in each class before and after SMOTE applied. Important to watch out for that only the training set should be taken for oversampling. otherwise there would be a big data leakage since the algorithm will be instances with a lot of correlation to the already existing points.



```
Before OverSampling, counts of labels:
2    37505
3     7423
1     5281
4     2005
0      694

After OverSampling, the shape of labels:
4    37505
3    37505
2    37505
1    37505
0    37505
```

The end results is that XGB was able to predict 'class-0' much better. but worse on all the other classes. It returned an f1-macro of **0.28**

Conclusion My impression after experimenting these days with the dataset in that there is not much predictance in it or I don't have the knowledge yet to get it out of the set. The next step would be to see if there are possibilities to find new and more predictive datastreams to add to the dataset. With more background information available about the type of dataset there also will be the option to do some feature engineering by retrieving new columns from existing ones. I think I coul have tried multiple approaches from reducing to oversampling. I am happy with the overall process at the challenge and learned oversampling with SMOTE.

I am eager to learn more about the internship position and would appreciate the opportunity to speak in an interview about my qualifications.