Classification project work report

December 12, 2020

1 Introduction

The goal of the project is to build a classifier that predicts the class (event type) of the NPF data's observations. This is achieved by first analyzing the provided data. After that, by selecting an appropriate machine learning model and possibly a dimensionality reduction algorithm. Training the model with the provided labeled data, and then testing it by predicting the also given, unlabeled, NPF data.

The main part of the goal is to create a well-working binary classifier. However, good scoring in multi-class classification is a secondary goal. In our project, we held these two flags equally high during feature selection and while choosing the ML model. For this reason, we ended up with one particular classifier for both cases out of 11 different ones.

In this report, we will first go through the process of completing the project work. So first, we will discuss how we analyzed the NPF data. After that, we will explain what machine learning pipeline we ended up choosing and how the selection process proceeded. We will also talk about the optimization of hyperparameters of our model and the dimensionality reduction technique. Lastly, we will have a look at the results of our classifier and discuss the project as a whole.

2 Data analysis

In this chapter, we will explain what steps we took at the begging of the project to analyze and understand the data at hand. Originally we only knew that we were dealing with both labeled and unlabelled NPF data. From which latter one, we were supposed to predict using the first one as training data for supervised machine learning.

The first thing we did after loading the data was, of course, to have a simple plain look at what we are dealing with. Because of this, we then knew that we had data that had columns for means and standard deviations of multiple variables. The labeled data also contains the class (event type) and date of the observation, unlike unlabelled data, which doesn't contain either of them because they are pruned away. We did notice that the mean and std columns of each variable clearly have different distributions regarding the values they contain.

After that, we proceeded by visualizing the data using different plots. One of the more insightful ones was a scatter plot representing the correlation between variables. We executed this for most of the data set and found out that means and stds of variables that are similar to each other have a high correlation between each other. However, this correlation doesn't expand between stds and means nor between completely different variables. So basically, the data set contains groups of columns that highly correlate with each other.

However, to be sure about the correlation between columns in the data set, we computed the correlation matrix of the data sets. Results did support the observation done using plots.

3 Selecting ML model and dimensionality reduction technique

Analyzing the data didn't give out much regarding how it correlates with the target of the classification (event type). For that reason, we took the more computationally harder route to achieve the best results in classifying the NPF data. Because we had to go through many different classifiers, we also decided to combine them all with a bunch of different dimensionality reduction techniques.

For evaluating the classifiers and their performance with different dimensionality reduction techniques, we introduced two parameters. First is the accuracy score of the model using cross-validation, and second is how well they perform in both multi-class and binary classification tasks. To compare the usefulness of dimensionality reduction techniques, we performed cross-validation for all of the models with full data set too. However, the parameters of the dimensionality reduction techniques were optimized before the actual model selection began. However, we will talk about that in the next chapter of the report.

We computed the accuracy score of cross-validation as a mean of n independent repetitions of k-fold cross-validation using labeled data. Because of the limitations of computers, for model selection, we used five and ten for n and k, respectively.

The results of plenty of computing were quite surprising. This is because the top 3 classifiers ended up being Logistic Regression, Gaussian Process classifier, and Multi-layer Perceptron classifier wIth a variety of dimensionality reduction techniques. From them, the most consistently performing model was the Gaussian process classifier using the original data, so without any dimensionality reduction technique, and using multitude of dimensionality reduction techniques. Dimensionality reduction techniques, however, didn't have much beneficial effect on performance. THus, we decided to use Gaussian Process classifier without fature selection.

Another notation that we did during the experimentation was that while standard deviation columns of the data alone are not viable predictors, they still do give some additional information. We were able to get marginally better results across the plain of classifier-technique pairs when using the entire data set instead of using only mean columns. However, there were occasional cases where the performance was inverse.

Even though Gaussian Process classifier wasn't the most accurate one in neither binary nor multi-class classification tasks. In the binary case, the most well working model was a combination of Logistic Regression and K highest scores feature selection. Where we used mutual information estimate for a discrete variables as the scoring method. Whereas in the multi-class case, the most accurate model-technique combination was a pair of Logistic Regression and extra-trees classifier. From which, the extra-trees classifier was used to select useful features for actual classification.

In the end, because of all the evidence, we decided to choose the Gaussian process classifier as our ML model for NPF data. We did, however, run one more cross-validation just for the selected model but with n=10. The result of this experiment did support our initial results. The accuracy of the Gaussian process classifier in the binary classification task was 89.9% with a standard deviation of 0.043, and in the multi-class classification task, it was 67.5% with a standard deviation of 0.061.

4 Optimization of hyperparameters

Before we began searching for a proper machine learning model for the NPF data by iterating through a multitude of them in a combination of different dimensionality reduction techniques, and of course without them too as a baseliner of the model accuracy. We did optimize the parameters of the techniques by comparing their results on different parameters in both binary and multi-class classification tasks. The evaluation was conducted by performing the earlier mentioned cross-validation process using the logistic regression model as the classifier for the techniques.

After we had found seemingly optimal parameters for each dimensionality reduction technique in both cases, we did one more check-up iteration for them with higher n to make sure results didn't change. Because the last round of computation only strengthened our view on the parameters optimality, we decided to choose them for the actual model search.

Besides optimizing the hyperparameters of the dimensionality reduction techniques, we also proceeded to tune the hyperparameters of the top two models of our experiments. We performed an extensive grid search for both Logistic Regression and Gaussian Process classifier on a multitude of parameters with a variety of possible values. Hyperparameter tuning regarding Gaussian Process classifier ended up selecting parameter values that were already initially chosen when doing experiments, meaning that the model accuracy performance didn't benefit from optimization. The same thing goes for Logistic Regression, at least when using an extra-trees classifier as a selective model. The hyperparameter tuning is generally so expensive in computational power that we didn't have enough time to conduct optimization for multitude of machine learning pipelines.

5 Results

Gaussian Process classifier's performance in cross-validation was decently high. During experiments, it was able to achieve cross-validated binary accuracy of about 90% and multi-class accuracy of almost 70%. However, because real-life situations (reality) is not as controlled as any training environment could ever be, it is only natural that the test score will be lower than the cross-validation score.

One of the points of the project was to predict unlabelled test data. For that, there was a challenge on which 43 projects (groups of 1-3) participated. In the challenge, there were 4 categories of which were: binary accuracy, multi-class accuracy, perplexity, and accuracy of binary accuracy estimate.

Our Gaussian Process classifier performed well as it got second place in binary accuracy and first place in multi-class accuracy. On the other hand, the perplexity score of the model was not good at all, which actually was partly because of the mistake that happened during the generation of prediction. Lastly, the estimate of binary accuracy went also quite well, considering that we got fifth place in that category. This is most likely because our estimate base on the cross-validation scores gotten in binary accuracy. But like mentioned above, the binary accuracy score gotten by cross-validation is the absolute upper limit for the real binary accuracy. Thus, we reduced the number by 0.01-0.02 to make it a more realistic value, which in fact was quite an accurate reduction. The accuracy of our accuracy estimate was 0.0038 as the true accuracy of our model in the binary case was 87.9%. The multi-class accuracy instead was 69.4%, meaning that we were actually quite close in that one too.

The perplexity score of our model, while the mistake was still in place, was 3.52. But after we realized the mistake and fixed our code, we actually got a perplexity score of 1.63. The mistake

was that we accidentally output probabilities that the prediction is not part of the class. However, it was a simple fix, as all we had to do was to inverse all the probabilities that were part of our prediction.

6 Conclusion

Overall, the project has definitely gone well. We manage to identify multiple classifiers with a variety of dimensionality reduction techniques that can explain the NPF data fairly well. This is the case in both binary classifications as well as in multi-class classification. However, one machine learning pipeline or, more specifically, one model did perform most consistently. This model is a Gaussian Process classifier.

Gaussian Process classifier performed in repeated cross-validation and challenge really well, considering that its accuracy scores had at most a reduction of a mere one present between them. This means that the Gaussian Process classifier can definitely explain NPF data with significant confidence, which is a good aspect of a machine learning model.

All in all, the project did teach us a lot about machine learning and, more precisely, about the classification of new previously unknown data. Future work in this topic that could be conducted is to perform more hyperparameter optimizing with both dimensionality reduction techniques as well as with the classifiers. Before the actual machine learning pipeline, classifier and dimensionality reduction technique, selection, because this would most likely lead to much better results in binary accuracy, multi-class accuracy as well as in perplexity. This would lead to a model that could theoretically predict the NPF data with accuracy and confidence of 99%.

7 Appendix

7.1 Link to the code

https://github.com/Jhoneagle/IML-term-project