FIT5149 S2 2021 Assessment 2 Report

Siyu You

Abstract

This report provides a general evaluation of the performance of three supervised classification models(support vector classifier, gradient boosting, neural networks) on classifying the labels of 19 different material abstracts. The labels are heavily imbalanced as shown in Fig. 1, which could lead to label bias in classification models. I'll be presenting two approaches to handle this problem, 1) undersampling the majority label, 2) pseudo labelling(a variant of semi-supervised learning) the unlabelled data.

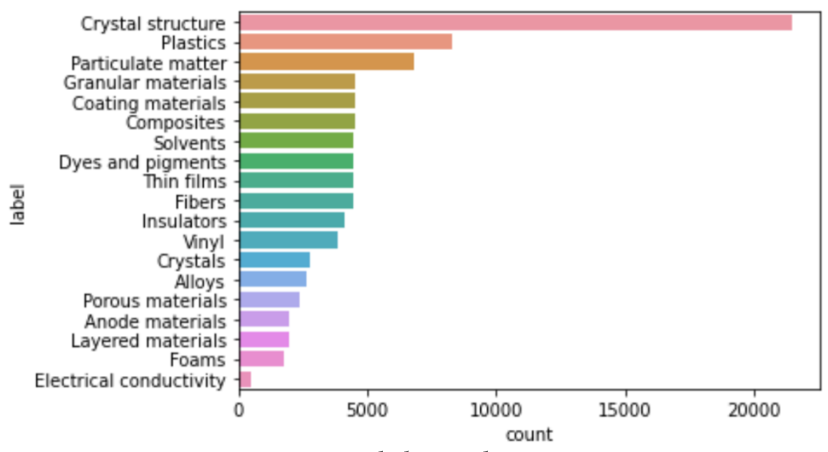

Fig. 1 Label Distribution

Table of Contents

1 Introduction 1

2 Pre-processing and features used........ 1

3 Models 1

3.1 LinearSVC 1

3.2 LightGBM 1

3.3 Neural Networks(GRU) 2

3.4 Discussion of model difference(s) 2

4 Experiment setups 2

5 Experimental results 3

5.1 using original data 3

5.2 undersampled data 4

5.3 pseudo labeling 5

6 Conclusion 5

7 Future works 5

References 6

Appendix 7

1 Introduction

The project takes on a set of scientific documents, more specifically, chemical material abstracts crawled from American Chemical Society, aiming to develop several classification models to make predictions on the abstract labeling. Along the way, some text preprocessing steps such as lower casing, stopword removal were applied, and the performance of the classification models was evaluated.

Since I am the only member in my group, the entire work was done on my own.

2 Pre-processing and features used

I've done the following preprocessing steps on abstracts in both training and testing datasets, and created a 'clean' column in both datasets to store the processed abstracts

1. Lower casing: normalize the abstracts to lower cases to reduce the chance of word duplications
2. Contraction expansion and greek symbol conversion: convert various forms(` ′“) of apostrophy to the uniform(') one, then perform contraction expansion along with greek symbol convertion to English letters. These help reducing any syntactic anomalies
3. HTML tagging and web link removal: remove the web links and HTML tags(<p>) as they would not generate any domain specific information
4. Noise removal: noise here refers to punctuations(except space), stopwords(including words with length equals 1) and numbers, I've transformed them to white spaces. The reason to get rid of these is because they are universal to all abstracts, which are irrelevant to the classification models.
5. Lemmatization: finds the original base form of the word in regards to the context. The reason to choose it over stemming is because stemming might stem off some valuable information

The next step was done solely on the training dataset

6. Remove duplicated rows: there might be duplicated abstracts that belongs to multiple labels, by removing these could make the classification model better generalized to the data

The last step was to fill abstracts with missing values in both training and test dataset with word 'NA' in order for vectorizer to recognize(since the regex pattern I've used for vectorizer is '\w{1, }', which skips missing values)

Label encoding: dependent variable 'label' contains set of strings, for which I've encoded it to numerical values that are suitable for classification models

3 Models

3.1 LinearSVC

The traditional (hard margin)SVM aims to find a optimal hyperplane that maximizes the margin between classes without any misclassifications in between, and the optimal hyperplane is solely dependendent on the support vectors. However, a shortage of SVM is that it cannot be applied to linearly inseparable datasets. That's where (soft margin)SVC being introduced. Similar to SVM, SVC aims to find a(binary classification)/many(multilabel classification) hyperplane(s) that separates the classes. However, in contracts to the hard margin SVM, it allows some misclassifications to be made, thus more flexible in the sense that it mitigates the overfitting issue when applied to linearly inseparable datasets. The optimization problem is defined as $L^* = \min_{w,b} \frac{\|w^2\|}{2} + C\sum_i \epsilon_i$ where ϵ_i denotes loss function. The C acts as a penalization factor, when $C \to \infty$, $\frac{\|w^2\|}{2} \to 0$, in this case SVC has the same goal as the traditional SVM, which is to find a margin with no classification errors(low bias, high variance), however, this violates the primary purpose of SVC. On the other hand, when $C \to 0$, restriction becomes weaker, which allows come misclassifications(high bias, low variance).

3.2 LightGBM

Gradient boosting method ensembles multiple weak learners(decision trees for lightGBM), such that each learner is trained sequentially by fitting on the negative gradients of the loss function conducted via the predictions of the previous learner, which eventually arrives at one final strong learner. By combining the results of multiple decision trees, gradient boosting reduces the variance of using a single tree $\left(\frac{\sigma^2_{single}}{n}\right)$, thereby mitigates overfitting. LightGBM is one of the gradient boosting models that is proven to be the fastest amongst others, it uses Gradient-based One-Side Sampling(GOSS) which amounts to use only the instances with larger gradients, and Exclusive Feature Bundling(EFB) to bundle the mutually exclusive features that achieves the goal of feature reduction.[1]

3.3 Neural Networks(GRU)

In the traditional neural networks, the observations are used as input units, which pass through some hidden layers that capture more and more implicit information of the observations, in the end, they are feeded into an activation function(sigmoid/softmax is used for binary/multi-label classification respectively) that outputs the most probable labels for each of the observations, which then gets evaluated through a loss function and uses algorithms such as back propagation to optimize the loss. This process repeats until the loss converges.
RNN is a variant of traditional neural networks where the previous round's hidden layer results are used as inputs for current round's hidden layers along with the input units, thus it has a sense of 'remembering'. GRU is more robust than RNN in a way that it is not only capable of using previous round's result, but also capable of using results from several rounds earlier.

3.4 Discussion of model difference(s)

LinearSVC:

Advantages: very efficient and less prone to outliers since support vectors are sufficient enough to determine the optimal hyperplane(s), due to this, it can handle very large datasets, like the one in this project; allow some misclassifications to be made thus it is able to handle linearly inseparable datasets
Disadvantages: needs to find a suitable kernel function for the linearly inseparable data which can be time consuming(luckly, text classification is generally proven to be linearly separable[4])

LightGBM:

Advantages: reduces the likelihood of overfitting; effective for handling imbalanced data as in each turn, the weak classifier focuses more on the wrongly classified samples(even if the first tree encounters label bias issue, subsequent trees will try and fix it), thus it is rather unaffected by class imbalance.
Disadvantages: computationally expensive in terms of both training and parameter tuning since it amounts to fit multiple decision trees sequentially; less interpretable of feature interactions, ie. two interactive features could be randomly selected by different learners but not together

Neural Networks:

Advantages: able to capture hidden complex informations within observations through hidden layers, thus it can handle nonlinearity within data
Disadvantages: the loss function is typically non-convex, thus the result is likely to converge to a local optima; may get affected by outliers if the networks is too complex

4 Experiment setups

Validation set setup: train_test_split from sklearn is used to split the data for which 65% of the data are used as training set and the remaining 35% are used as validation set. Also due to label imbalance, stratify is applied on the dependent variable for preserving the original label distribution, thus prevents validation set having all samples with the same label

Cross validation setup: StratifiedKFold from sklearn is used with K setting to 5, this again preserves label distribution

TfidfVectorizer: used for generating document-term matrix for LinearSVC and LightGBM. Key concept here is that if a term appears in only a few documents while much frequent in the current document, this document is likely to be dependent on the term, therefore the word gets assigned a higher score. Likewise, if the term appears in most documents but rather rare in any particular document, it gets assigned a low score. For parameters, ngram_range is set to [1, 2] to extract both uni-gram and bi-gram features, max_df is set to 0.95 and min_df is set to 5 for excluding words that either occured in over 94% of the texts or occured less than 6 times

Tokenizer: text tokenization for the neural networks model which is provided by keras

Evaluation metric: accuracy_score from sklearn which provides computation of the classification model accuracy

Pretrained embedding vector: GloVe from Stanford is used as weight matrix in the embedding layer in the neural networks model, which hoping to be more generalized on the data

RandomizedSearchCV: used for finding best hyperparameters for each models

Undersampling: use NearMiss from imblearn with sampling strategy set to dictionary of label frequencies, and n_neighbours set to 5

Semi-supervised Learning(Pseudo Labelling): inspired by [2], the idea is to first train the data on the original sample using a classifier, make predictions on test data, and concatenate predictions with original data. The concatenated data then gets trained on another classifier that makes final predictions on the test data

Initial parameter setting:

LinearSVC	C: 1; max_iter: 3000
LightGBM	num_leaves: 31; max_depth: -1; objective: multiclassova; metric: softmax; subsample: 1; colsample_bytree: 1
Neural Networks	1 embedding layer: GloVe; 1 BiGRU layer: (128,); 1 Cov1D layer: (128, 3); 2 Pooling layers; 1 Attention layer; 2 Dense layer :(64,) and (32,); activation function: relu; batch_size: 128; epochs: 10; loss: categorical_crossentropy

5 Experimental results

5.1 using original data

model / result	**LinearSVC**	**LightGBM**	**Neural Networks**
Accuracy(%)	cv: 80.6 validation set: 82.6	cv: 80.9 validation set: 82.7	cv: 83.1 validation set: 81.3
Computational Time(minutes)	0.5	10	8(use TPU on colab)
Best hyperparamters	C: 0.2	subsample: 0.2 num_leaves: 31 colsample_bytree: 0.5 max_depth: -1	optimizer: Adam kernal_initializer: he_unifrom
Average validation set accuracy(%)			82.2

Table 1. contains cross validation and validation set accuracies for 3 models; computational cost of models; best hyperparameters found by RandomizedSearchCV; overall average validation set accuracy using original data

From Table 1 we can see that

- LightGBM in general performs the best among the models, with the highest validation set accuracy, thus we can see that, gradient boosting model did handle class imbalance fairly well comparing to other models. However, it also has the highest computation complexities(takes around 10 minutes to train). We can see the optimal subsample is 0.2, which corresponds to 20% of the samples being randomly selected to train each decision tree comparing with the default which is 100%, thus prevents overfitting
- LinearSVC has also made a fairly accurate prediction, which suggests that the features are linearly separable. The small C which gets determined via RandomizedSearchCV indicates that a larger margin is more suitable in this case, thereby allowing some classification errors to be made
- Although neural networks has the highest cv accuracy, it also has the lowest validation set accuracy, which is a result of overfitting. This might suggest that the layers are a bit complex.

- The overall average validation set accuracy achieves 82.2%

In terms of confusion matrices(see appendix.)

- Due to class imbalance, all three models seem to be biased toward 'Crystal structure', which has the highest number of correct predictions, whilst also having large number of false positive predictions(vertical). In particular, 'Crystal', 'Granular material', 'Particulate matter', 'Solvents', 'Thin films' and 'Vinyl' are the ones with false predictions over 100. This issue might be fixed by resampling the data
- Comparing the two machine learning models, LightGBM handles class imbalance better than LinearSVC, which gives lesser weights to the most frequent label, while boosting the correct predictions of other labels.
- For the deep learning model, it captures too much complexities(noises/outliers) of the features, which leads to lesser number of accurate predictions, as well as a lot more false positive predictions than the two machine learning models.

5.2 undersampled data

result \ model	**LinearSVC**	**LightGBM**	**Neural Networks**
Accuracy(%)	cv: 79.9 validation set: 82.8	cv: 81.2 validation set: 82.8	cv: 82.5 validation set: 81.3
Best hyperparamters	C: 0.2	subsample: 0.5 num_leaves: 40 colsample_bytree: 0.5 max_depth: 50	optimizer: Adam kernal_initializer: he_unifrom
Average validation set accuracy(%)			82.3

Table 2. validation set accuracies for 3 models on undersampled data using NearMiss Version 2, specifically, it downsamples 'Crystal structure' to 70%(selected via cross validation) of the original frequency; overall average validation set accuracy using undersampled data

The reasons of choosing undersampling over oversampling are as follows

- The most number of false predictions are associated with the most frequent label
- Oversampling is likely to lead to overfitting due to random noise(repeated samples) being generated
- Undersampling might remove samples(outliers) within 'Crystal structure' that could be considered implicitly correlated with other materials.

There are mainly two types of undersampling, one is random undersampling, which selects subset of samples from the majority label at random, because of the randomness, it can easily discard useful samples, thereby causes information loss. The other type uses k-NN to select the sample subset, which tries to retain as much information as possible. The algorithm I've chosen is of the second type.

In addition, as for preventing the overfitting issue occured in neural networks, I've added 2 dropout layers that act as feature regularizers by randomly setting some proportion of input units to 0. Furthermore, I've used earlystopping as an input for the callback functions which monitors the validation loss and stops training when the loss starts to increase.

From Table 2, the results demonstrate that

- The overall validation set accuracy has increased by 0.1% comparing with the one in original data. Thus we can see that by down sampling the data, label distribution balances out slightly, yielding a classification model with lesser bias towards the most frequent label.
- The deep learning model again overfits the data, however, the issue is less severe than the original one, thus it proves that using the overfitting prevention techniques(dropout & early stopping) is indeed affective

- This time the optimal tree depth for LightGBM has pruned from unconstrained down to 50, which again prevents overfitting, also it increases num_leaves to boost accuracy(as indicated in Parameter Tunning section in LightGBM documentation)

From confusion matrices

- Classifiers greatly penalize the most frequent label, which is indicated by the decrement in number of correctly predictions, along with an increment of correct predictions for other labels
- The number of false positive predictions for the most severe labels that we've previously identified are dropped(except for the neural networks model with the overfitting issue)

5.3 pseudo labeling

secondary / primary	LinearSVC	LightGBM	Neural Networks	Average Accuracy
LinearSVC	82.7	82.7	83.1	82.8
LightGBM	82.5	82.3	82.0	82.3
Neural Networks	82.3	82.1	81.8	82.1
Average validation set accuracy(%)				82.4

Table 3. validation set accuracies of the secondary models, where the pseudo labels are generated via primary models; average validation set accuracy grouped by primary models; overall average validation set accuracy using pseodo labeling

Pseudo labelling differs from undersampling by alleviating label bias with extra data in a semi-supervised manner[2], doing so would minimize the entropy(uncertainty) of unlabelled data, hence gets a better generalization performance[3].

From the results in table 3

- The model pairs with LinearSVC being the primary model generally outperforms other pairs, with average validation set accuracy reaching 82.8%.
- The overall average validation set accuracy also outperforms undersampling by 0.1%
- The highest accuracy model pair corresponds to using LinearSVC as primary model and Neural Networks as the secondary model, which achieves an accuracy over 83%

The confusion matrix in the appendix corresponds to the best model pair. Comparing it to the neural networks confusion matrix with the undersampled data, we can see that the pseudo labelling approach adds more penalization on 'Crystal structure', whilst retains a fairly accurate predictions on other labels.

6 Conclusion

Through training and evaluation, the optimal classifier pair is found to be LinearSVC + Neural Networks with pseudo labelling applied on unlabelled data, which is also the final model chosen for submission. From this project, I've learnt

- Various text preprocessing techniques such as lemmatization and vectorization
- How the change of C parameter in LinearSVC would affect the accuracy, for this project, a small C is more suitable which correpsonds to a larger margin between decision boundaries
- How to reduce overfitting in neural networks with the use of dropout layers and early stopping
- How label imbalance could be an issue to the classification modeling such that it makes the models be biased toward the majority label
- Handlation of label imbalance by either undersampling of the majority label or pseudo labelling of the unlabelled data to achieve a better generalization. Comparing their results, pseudo labelling tend to perform a bit better than undersampling, especially when using LinearSVC as the primary model

7 Future works

- In terms of preprocessing, some spelling corrections could be made
- In terms of imbalance handlation, another approach could be to split the majority labelled sample into multiple parts of smaller sizes, then create dataframes equalling number of the parts where each dataframe contains the combination of one of the parts and rest of the samples. At last, train different models on those dataframes and make predictions, the results are then averaged to arrive at the final prediction.
- Further improvements on the pseudo labelling algorithm, eg. select subset of the unlabeled data predictions, such that the set contains lesser samples with prediction labels equivalent to the majority label, and more samples with prediction labels equivalent to the minority labels, the best subset can be selected through cross validation.
- In terms of deep learning, BERT(Bidirectional Encoder Representations from Transformers) might be used as a replacement of the embedding layer, where it contains a large set of pretrained word embeddings similar to GloVe.

References

[1] Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye and Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." NIPS (2017)

[2] Yuzhe Yang and Zhi Xu. Rethinking the value of labels for improving class-imbalanced learning. arXiv preprint arXiv:2006.07529 (2020)

[3] Lee, Dong-Hyun. Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networkss. ICML Workshop : Challenges in Representation Learning (2013)

[4] Joachims, Thorsten. Text Categorization with Support Vector Machines. Proc.   European Conf. Machine Learning (ECML'98). 10.17877/DE290R-5097 (1988)

Appendix

```
0: Alloys 1: Anode materials 2: Coating materials 3: Composites 4: Crystal structure 5: Crystals 6: Dyes and pigments 7: Elect
rical conductivity 8: Fibers 9: Foams 10: Granular materials 11: Insulators 12: Layered materials 13: Particulate matter 14: P
lastics 15: Porous materials 16: Solvents 17: Thin films 18: Vinyl
```

Fig. 2 Label Mapping

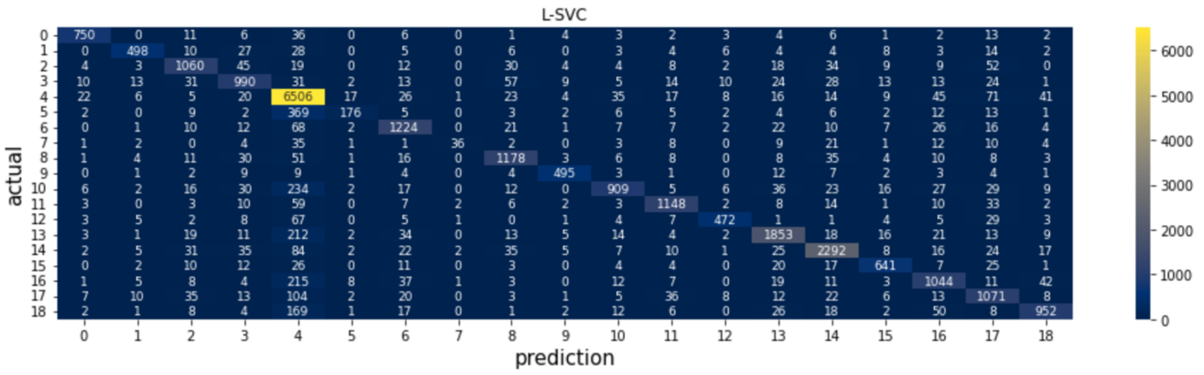

Fig. 3 LinearSVC Confusion Matrix with Original Data

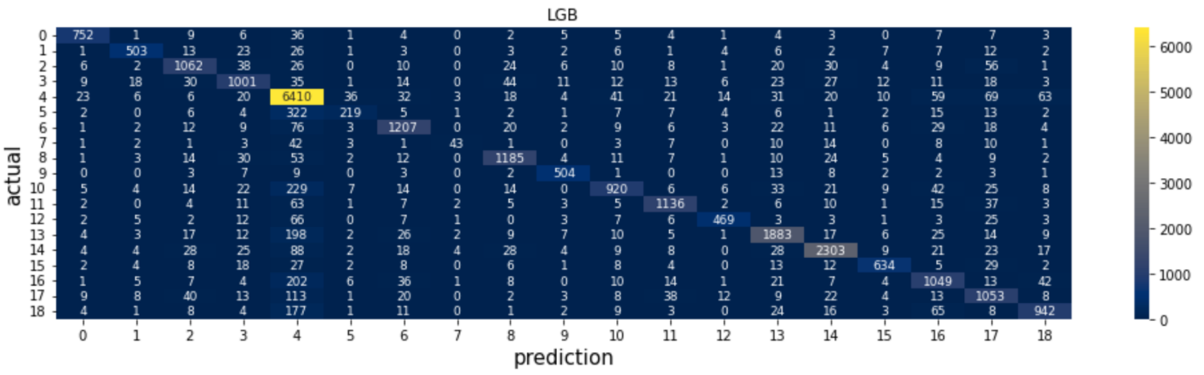

Fig. 4 LightGBM Confusion Matrix with Original Data

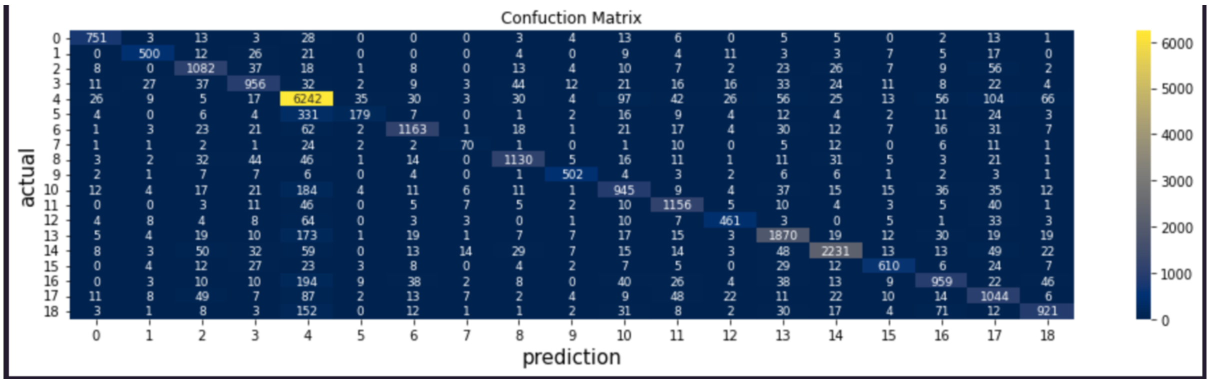

Fig. 5 Neural Networkss Confusion Matrix with Original Data

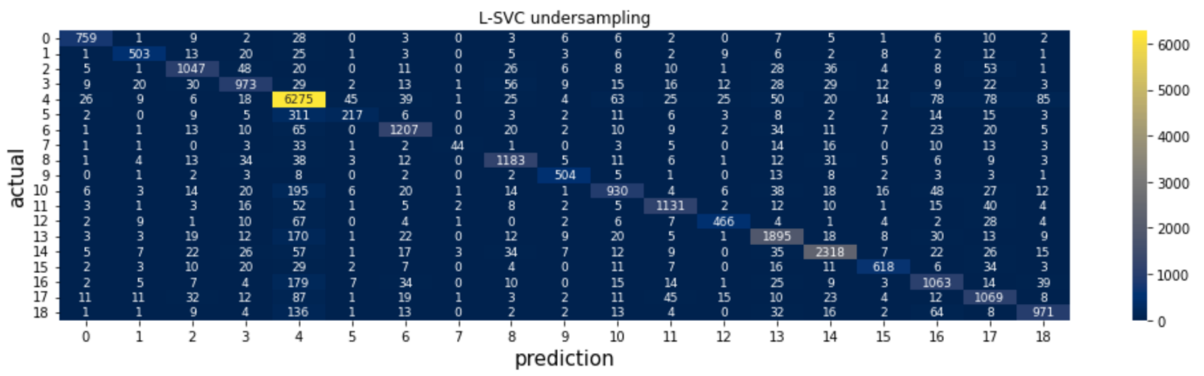

Fig. 6 LinearSVC Confusion Matrix with Undersampled Data

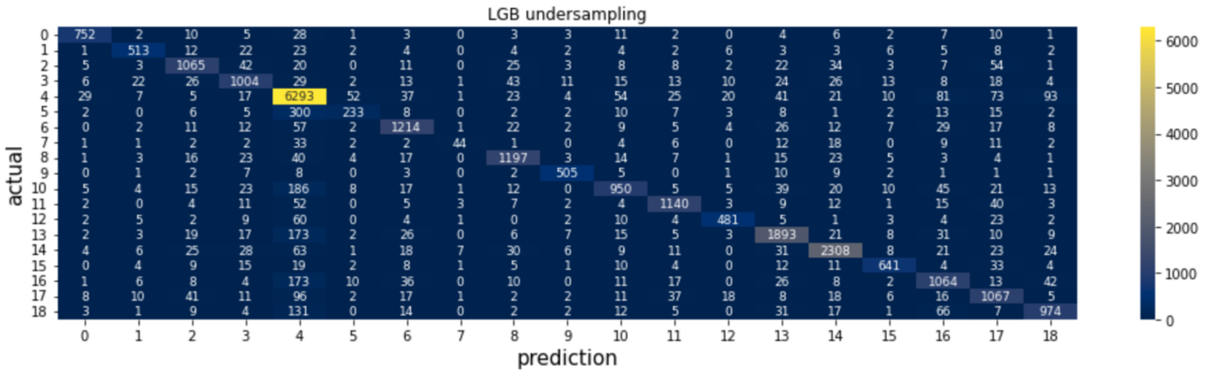

Fig. 7 LightGBM Confusion Matrix with Undersampled Data

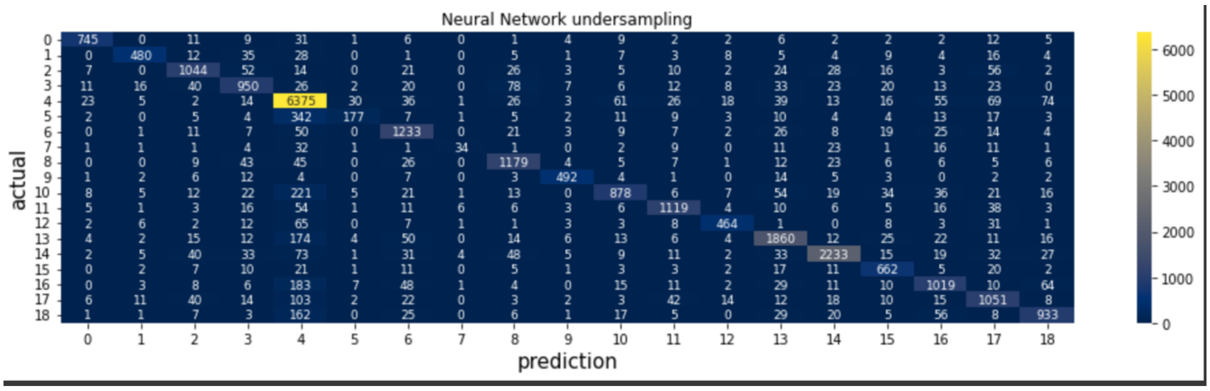

Fig. 8 Neural Networks Confusion Matrix with Undersampled Data

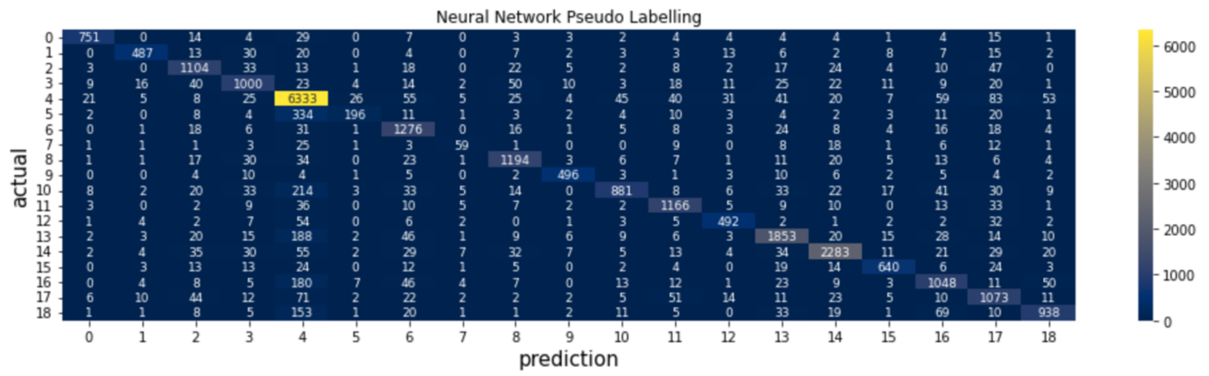

Fig. 9 Confusion Matrix for Best Model Pair using Pseudo labelling