Machine Learning Challenge Report

1. Problem Overview:

Dataset with high dimensionality has always been a hard challenge to deal with for Data scientist. In our dataset, we have approx 66K records and 295 features/column. The target variable has 5 classes namely A,B,C,D,E. We have to make a Neural Network model to predict the classes.

2. Initial Data Analysis:

Firstly, we generated a correlation matrix from matrix of features to show how columns are connected to each other and to what extent they are connected. Now let's have a look at the distribution of the class labels in our dataset.

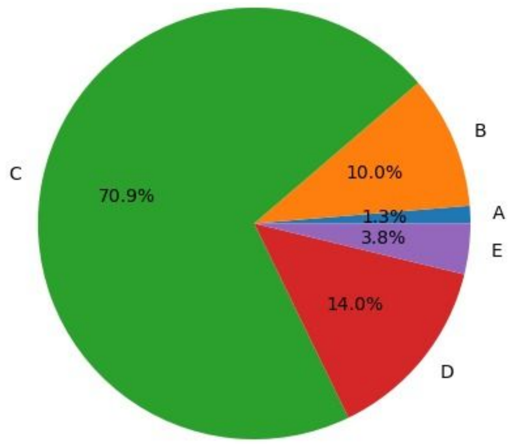


Fig: Class Distribution

Pretty much clear that class C owns the bigger part of the chart. Initially, we applied PCA as a dimension reduction technique. An average of explained variance of all columns were taken as threshold. The constraint was to exclude a particular column if it's explained variance is less than the threshold. 51 columns/features were left after applying PCA.

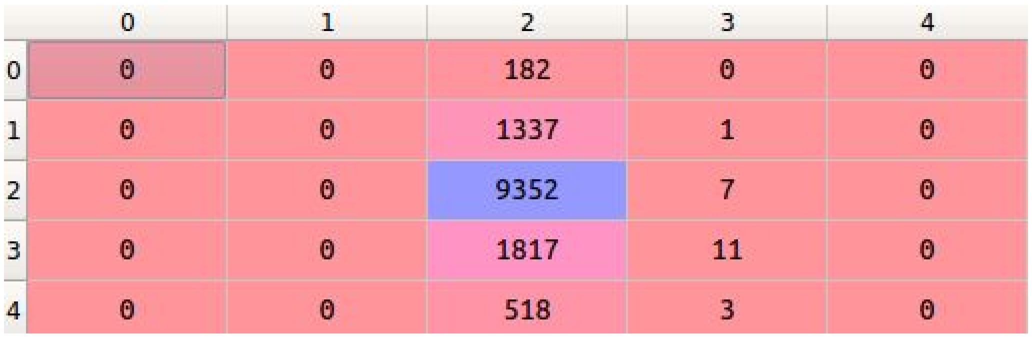
We also performed performed K-Means (K=5) clustering algorithm in this segment. The finding here is, we found all clusters being dominated by class C. This is something unusual and this is happening because the primary problem is that these classes are imbalanced: the other classes are greatly outnumbered by class C. C took 51.23%, 74.87%, 67.76%, 71.05% and 86.63% area in five consecutive clusters. So the fact is classes other than C might get ignored to some extent while applying predictive analysis and a strong strategy should be there to solve the problem.

2. Artificial neural network with k-fold cross validation:

In this section, we build a model to predict the given classes. Artificial neural network was chosen as a model. We have 3 hidden layers and all of them with rectifier activation function. Again, PCA does the job of dimension reduction and according to that we have 51 neurons in the input layer. Number of neurons in hidden layers are set by the following way: (number of neurons in input layer + numbers of neurons in output layer)/2. Its one of the common practices and is able to come up with good results.

Optimizer backpropagate the error through the network and optimizes the weights. This backpropagation can be done after each observation(Reinforcement learning) or after batch of observations (Batch learning). We have picked batch learning for our problem and the batch size is 64. We chose ‘rmsprop’ as our optimizer. When all the observations have gone through this forward propagation and backpropagation process, that completes an epoch and we have done 100 epochs.

0.2% of the split goes to test set and 0.8% goes to the training set. For k-fold cross validation, value of k is 15. Let’s observe the confusion matrix over testset generated by our program.



	0	1	2	3	4
0	0	0	182	0	0
1	0	0	1337	1	0
2	0	0	9352	7	0
3	0	0	1817	11	0
4	0	0	518	3	0

Fig: Confusion matrix over testset

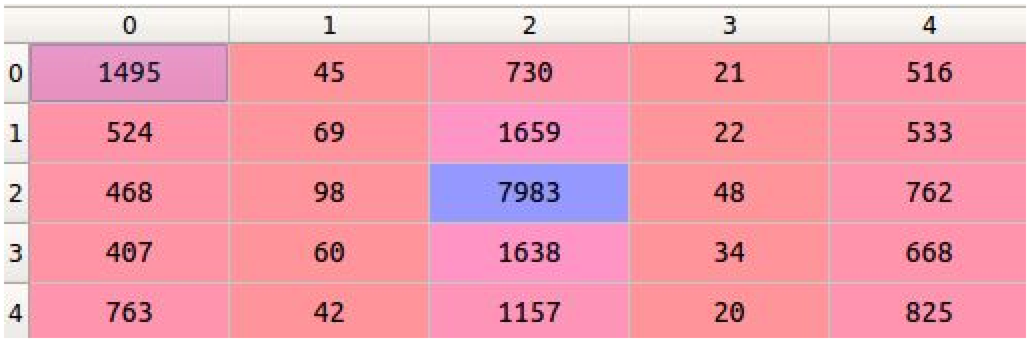
0,1,2,3,4 represents class A,B,C,D and E consecutively.Average precision and recall is 57.01% and 70.78%. Average accuracy achieved while performing k-fold cross validation was 70.92% and according to confusion matrix accuracy of the testset is 70.64%. The only problem we see here in the confusion matrix is except class C all the other classes were underestimated or ignored by the model. This is taking place due to the class imbalance problem as we mentioned earlier.
We have two good approaches to deal with this solution:

1. Undersampling the majority class.
2. Oversampling the minority class.

We will choose to proceed with Oversampling the minority class to solve the imbalanced class problem.

2.1 Oversampling the minority class:

By oversampling only on the training data, none of the information in the test data is being used to create synthetic observations. So these results should be generalizable. We are using SMOTE(Synthetic Minority Oversampling technique) algorithm to oversample data. After generating data we do shuffle out target variable and matrix of features. We perform oversampling on minor class for both Training set and Test set separately.
Let's have a look at the confusion matrix in this case.



	0	1	2	3	4
0	1495	45	730	21	516
1	524	69	1659	22	533
2	468	98	7983	48	762
3	407	60	1638	34	668
4	763	42	1157	20	825

Fig: Confusion matrix over testset

Average precision and recall is 42.73% and 51%. The average accuracy over training set was 52.02% and confusion matrix shows us 50.55% accuracy on test set. The accuracy was decreased than the previous model but we solved the class imbalance problem. Through oversampling the minority class, we have put weight to each of the classes so that our model didn't ignore any of them. The value of precision and recall also decreased as more samples were added to the test data but still all the classes are contributing in precision and recall unlike previous model.

3. Possible steps to improvise the model:

1. Autoencoder can be used as a dimensionality reduction technique
2. Grid Search can be used to tune the parameters of our deep learning model. It takes one or set of parameters and one or several values for each parameters. Then the model is executed for each combination and finally returns the best combination of values of the parameters with which the model performance was best. Though this technique requires a high computational cost.