Miner user: lalvarez

Rank: 34

F1 Score: 0.80

Introduction

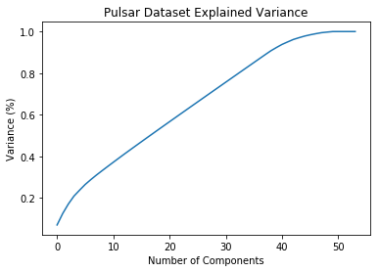
In this project I have analyzed the performance of the following classifiers: Neural Networks, Decision Trees, Naïve Bayes, SVC and K-NN in the given dataset. I decided to apply SVC and K-NN in order to get the best predictions for an imbalanced dataset.

Model evaluation procedure

To analyze the performance of my classifiers, I split my training data in two sets in order to train and test my model on different data. Due to I have more instances from one class than from the others, I used confusion matrices as a metric to compare my classifiers. With this metric, I can estimate how well a classifier will predict the class that are less frequent in my training data, in other words, how well my model will generalize the problem.

Preprocessing

Dimensionality reduction



This problem has a high number of variables so a dimensionality reduction technique could be applied to improve performance and execution time. The graphic on the right shows how PCA will perform on this data. From this results we can infer that we will need at least 45 components to maintain an acceptable level of variance. After this analysis, I decided to not apply PCA, due to the high level of variables needed. My project focuses on the problem of dealing with imbalanced data so I wanted to keep as much information as possible.

Imbalance dataset

Some methods that I will cover in this report needs to manage the weight that should be given to each class in order to deal with the imbalanced nature of the data and have a good performance predicting the test labels. To manage this problem I realized a deep analysis on my training data:

CLASS	1	2	3	4	5	6	7
INSTANCES	5296	7083	894	69	238	435	513

This table shows the number of times a class appears on my training data, I used this information to decide the weights that each class should have in order to deal with unbalance.

Classifiers

Neural Networks

Neural Networks provide a decent performance to this problem, regardless is not the best classifier for this dataset, a perceptron can deal with the imbalanced data and provide an accuracy of 64%.

To analyze the performance of this classifier I built different networks with different parameters.

Eventually, I came with the conclusion than the activation function that performs better when we have at least one hidden layer is the logistic function otherwise I used the default function relu.

Activation function	Number of hidden layers	Max. iterations	Score	Time
logistic	(15,15,15,15)	3500	0.75	2''
relu	(100)	2000	0.79	1'30''
relu	(100)	500	0.78	40''

The conclusion is that we get good results using a perceptron, we do not get better results if we increase the number of hidden layers. If we increase the number of neurons and the number of iterations, we overfit the model and the F1 score will be smaller. Also the time increases considerably. The best results are given when using a simple perceptron with 500 iterations.

Decision Trees & SVD

To implement my decision tree I used the scikit-learn library, this allows me to choose the parameters of the tree I want to build. Due to the imbalanced nature of my data, I built my tree with a balanced mode. The tree was balanced with the following weights:

class_weight={1:1.7, 2:1, 3:4.5, 4:7, 5:6, 6:5, 7:5.5}

This values are based on the distribution of the classes in my training data discussed before.

	F1 SCORE	Time
Balanced Mode	0.74	4 s
Default Mode	0.44	4 s

An interesting finding is that the results are clearly better when we increment the probability of the less frequent class to be chosen.

To complete my research using classifiers where I can explicitly change the weight of the classes I used a SVD classifier to see if I can get better results. I applied the same weights that I used in my decision tree and I found that this approach provided a performance as good as the decision tree with a 0.72 F1 score. The running time was also

Naïve Bayes

This algorithm is also well known for multi class prediction feature. Our problem has seven possible classes so out first intuition is that Bayes will provide a good classification of our test data. This algorithm has been applied using Complement, Gaussian and a Multinominal functions. MultinominalNB required to transform the first feature of my training data before because it contained negative values and this algorithm doesn't allow negative values.

From the results obtained, the best approach is to choose a GuassianNB, this type of Bayes classifier assume a Gaussian distribution of our data.

Naïve Bayes Model	F1 SCORE	Time
MULTINOMIAL	0.27	3 s
COMPLEMENT	0.37	2 s
GAUSSIAN	0.41	2 s

This classifier provides the worst results compared with the other classifiers.

KNN

My project uses a K-NN classifier to predict the forest type.

Neighbors	3	5	15
F1 Score	0.8	0.78	0.77
Time	3 s	5 s	6 s

The KNN classifier gave me the better results, I used the Euclidean distance and 3 neighbors. The score I get using this classifier is 0.80.

This algorithm does a great performance because it doesn't care about the number of instances of one class in the training data. The frequency of one class is not going to change the performance of the classifier.

I decided to use this approach because it gives the better results with an insignificant execution time compared with other classifiers such as neural networks that also gives good results but have higher computational time.