Haensel AMS
Advanced Mathematical Solutions

Recruitment Challenge - Data Science traineer

Contact: Daniel Carlos dos Santos Machado
Hansaring 25
Mnster, NRW 48155
Phone: (+49) 17 656 912 333
Email: dalabarda@gmail.com
Date: April 4, 2017

1. Initial data analysis.

All steps performed to analyse and work with the data is within the **recruitmentSolution.ipynb** Jupiter notebook file. My first action was looking for an NaN or invalid cell which in further manipulation could hinder the data processing. It turned out that there was no change in the row, containing the same initial 66137 ***Out[6]*** and ***Out[7]***.

the method df.isnull().values.any() were applied and the result was False. which is great. Any row was removed, there was no need to Impute any values and we could directly move to the next step.

There were around 10 columns with unique values. their indexes are 59, 179, 268, 269, 270, 271, 272, 273, 274, 275 and 276. I considered removing them once they would help the classification. However, as there are some Neural Networks that evaluate the interaction between columns. I thought there is no harm in still letting them in.

I've printed a Helper function that displays correlation matrix for each pair of columns in the data-frame by color. Red is most correlated, Blue least. Usually high correlation between columns hinder the the result once different columns might be from similar variables. The observation is that there are only 15 correlations between dimensions that are above 50% of the whole amount. And Only two variables presented significantly high correlation. If I were dealing with an small amount of dimensions. I would consider removing one column. However, in these case, I decided to keep it as it is.

The most notorious characteristic of this data-set is the Bad distribution of values/classes. Over 70% correspond to class C whereas class A Only represents a small amount of data(1,3%). Thus, a standard learning technique might not work very well. In this particular case, we need to use a special advanced technique. Around 10,0% of the data is classified as B. in these case, falls into the same problem as class A which has not many samples The Class C represents the vast majority of the cases covering around 70,9% of the total amount of classes.

2. Fiting some ML model(s).

Linear SVM, Random Forest & Logistic Regression.

First, I tried To use Linear-SVM to classify the data-set. Initially, I expected that the algorithm would at least classify values C and not C. It turned out to classify the majority of rows as B. I thought that I should change the default parameters. But as I don't have much experience, I decided to try another algorithm.

Trying with Random forest, It turned out to find right the opposite. The accuracy resulted trained data is 0,9786 and the Accuracy for the test data is 0.6934, which I believe it is satisfactory for this case. We can enhance it but I will leave it as it is and go to the other phase.

Performing the Logistic Regression algorithm resulted in an interesting insight. The default parameter classify all samples as value C. As c represent the vast majority of the data Class, it turn out to be a good result covering more than 70% of the total universe . When I started to modify the parameters to fit into a multivariate classification, the overall accuracy was hindered and decreasing around 1%.

Like modifying the parameter, The Cross Validation of logistic regression didn't show more accuracy although it tried to guess more values.

3. Results & Next Steps

An alternative to better classify this dataset could be by performing at first, a decision tree approach in which it would classify everything as C and Non-C. Then, in a second level, We could select the sabe amount of data from the remaining categories and execute the Random Forest Algorithm.

2.2 Neural Network

Probably a linear solution like Backpropagation may not be the best strategy once these vast amount of dimensions might have relationship with one-another or much more dimensions. Given the present data-set, the Gradient descent algorithm becomes computationally expensive procedure. A better alternative is using the Stochastic Gradient Descent could be a good solution.

# # #