June 27, 2019

Maksim Kumundzheiv

Company: Haensel AMS
Position: MACHINE LEARNING ENGINEER / DATA SCIENTIST
This solution was submitted and prepared by Kumundzhiev Maksim for the Data Challenge. I declare that this solution is my own work. I have not copied or used third party solutions. I have not passed my solution to another candidates.

Data description
(CSV file, sample.zip - 40 MB unzipped, 2 MB zipped) containing 66k records/rows and 295 features/columns. The target variable is the last column (the 296th) with values/classes A,B,C,D,E.

The task

1. Make an initial data analysis. Visualize the main characteristics of the datset and try to highlight potential helpful structures in the data.
2. Fit some ML model(s). Train and evaluate different models and please explain briefly your choices for the models and their pros cons.
3. Show with some X-validation the power of your model and comment the results. We are of course interested in the overall performance, but much more in the performance per class and especially in the under represented ones.
4. Present us the results and the steps you have taken. If possible add also some critical thinking and next possible steps. But mainly explain why your results are good and what insights we can obtain from it.

Solution

Defining the problem
Among all Machine Learning problems we can highlight the most basic:

- Classification of an instance to one of the categories based on its features;
- Regression - prediction of a numerical target feature based on other features of an instance;
- Clustering - identifying partitions of instances based on the features of these instances so that the members within the groups are more similar to each other than those in the other groups;
- Anomaly detection - search for instances that are "greatly dissimilar" to the rest of the sample or to some group of instances;
- In our case it was the multiclass classification problem

Part 1. Untuned solution:

- Check the target - the number of classes - understand which type of task we have
  if numeric - okay;
  categorical - one hot encoding or dummy function;
- The target - check balanced / imbalanced situation + visualisation
- Check the types of features
  if numeric - okay;
  if categorical - one hot encoding or dummy function
- Check the distribution of data
  if distribution is NOT high(all the random values is near with mean) - okay
  if distribution is high - normalize / standardize
- Split data (lots of ways(Validation, CrossValidation))
- Straight solution with few models, define benchmark

Part 2. Tuned solution:

- Implement Feature Selection and define features with the most Information Gain (reduce the number of features)
- Split data (lots of ways(Validation, CrossValidation))
- Straight solution with few models, define benchmark for modified data
- Compare results

"Defining" the evaluating metrics

- Accuracy - intuitive, obvious metric - the proportion of correct answers of the algorithm.

  $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

EDA - Exploration Data Analysis

- Exploring the data, we could assume lots of 0 values - that looked like sparse matrix. In order to fix we are demanded to make more attentive analysis or to reduce the number of futures. (PCA, SVD, RFE, Borura etc)

- Also, it was possible to notice that the values in the data were very scattered. In order to fix we need to use normalization or standardization(In my case I used normalization).
- As well, the target value was imbalanced. In this case there were few ways: to use oversampling/undersampling techniques, to combine smaller classes with each other(combine smaller 4 classes out of 5) and get the Binary Classification task or try to go straight and make downscale benchmark.
- Likewise, after creating a correlation matrix(which is one of the most simple and effective statistical filters) it is very clearly visible that lots of variables are strongly correlated. To fix this, you can remove one value in a bunch of correlated objects or to use Feature Engineering approaches.

Machine Learning approaches

Due to the fact that we have the trivial problem of Multi Classification, I decided to try the standard algorithms of machine learning, such as:

- - Logistic Regression ($sklearn.linear_model.LogisticRegression$);
- - SGD Classifier ($sklearn.linear_model.SGDClassifier$);
- - KNeighbors Classifier ($sklearn.neighbors.KNeighborsClassifier$);
- - RandomForest Classifier ($sklearn.ensemble.RandomForestClassifier$);
- - GradientBoostingClassifier($sklearn.ensemble.GradientBoostingClassifier$);
- - Feature Engineering (Selection) - Boruta ($fromborutaimportBorutaPy$)

Due to the condition that the test data was not given, it was necessary to validate on the train data. Here I used KFold cross validation approach. Also, in some algorithms I used a loop in order to find the best parameters of the model. Regarding the hyper parameters, I used well-known values, confirmed by prooved mathematics theorems and experience.

Feature Engineering (Selecting)
I decided to use the Boruta
Boruta is an all relevant feature selection method, while most other are minimal optimal; this means it tries to find all features carrying information usable for prediction, rather than finding a possibly compact subset of features on which some classifier has a minimal error.
As a result I got 14 features with the highest Information Gain.

Comparison of evaluation
Algorithm's Evaluating

	Result
$KNeighborsClassifier$	0.6298
$RandomForestClassifier$	0.7088
$Logisticregression$	0.7085
$SGDClassifier$	0.7090
GBM	0.9258
$KNeighborsClassifier+Boruta$	0.6253
$RandomForestClassifier+Boruta$	0.7088
$Logisticregression+Boruta$	0.7085
$SGDClassifier+Boruta$	0.7090
$GBM+Boruta$	0.9973

The ways of improving the r insuresults
it is correct;

- - First of all I would like to test on new test data(develop synthetic data) and actually check on overfitting;
- - Also, try to decrease the time complexity for GBM or implement LightGBM etc;
- - As well, check another Feature Selector in order to insure the selected features are correct;
- - Re-sampling Dataset (SMOTE) etc;