Capstone Project José Mauricio Mani

Machine Learning Engineering Nanodegree

July 11, 2020

Ecce Homo: An Application for Automatic Machine Learning Workflows.

Definition

Project Overview

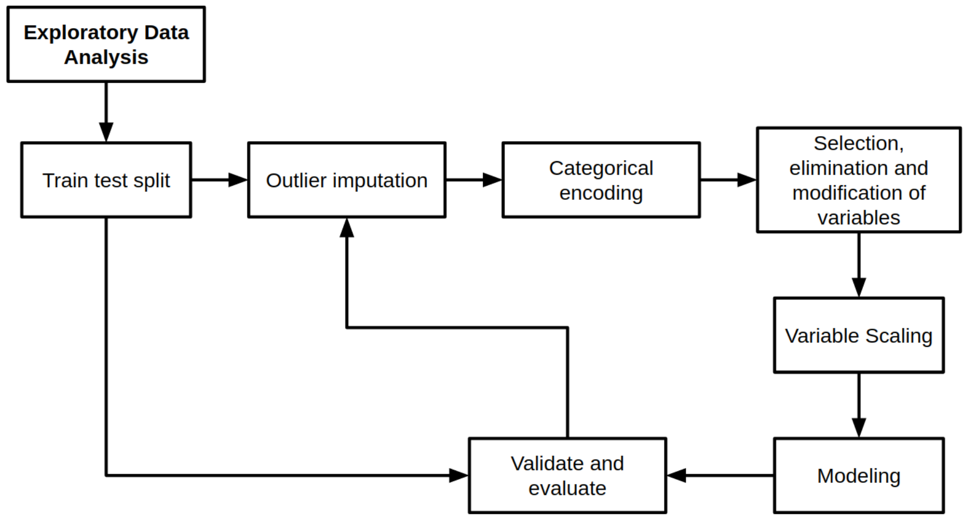
As we have seen, the machine learning workflow is clearly defined. Therefore, we can automate and perform easy implementation tasks. It is not true, we can accurately automate all the tasks, without more advanced techniques, like reinforcement learning. However, in every task we can think of smart techniques to overcome these problems. In this first version I propose automated tasks for solving binary classification tasks, primarily aided by Bayesian optimization and smart imputation methods, plus quick exploratory data analysis basics to focus in more smart deep dives, data acquisition and feature engineering.

Problem Statement

This application expects to help in three ways.

- A benchmark for human beings to achieve better outcomes than an automatic workflow (Ecce homo).
- Speed up the process of creating data products, to focus on other issues, like getting better data, analysis and explorations, but also things like deployment of the products; assuring end to end implementations.
- As stated by Dan Lubanga throughout the course, making easier to create machine learning products would attract more people to the field, which means more teamwork and feedback from different backgrounds.

The first bullet aims to motivate enthusiasts and professionals to achieve better metrics in their model, having competition, and especially non-human competition. The second bullet, and the one I have found the most annoying for some peers in the industry, is the time they spent doing the same tasks, again and again. And finally, the third one, I expect this product help more people develop easier machine learning products, hence being captivated and ready to go deeper on their AI learning.



Inputs and benchmark

Test de application on Kaggle binary classification tasks. On the famous titanic dataset (https://www.kaggle.com/c/titanic) we will show the importance of feature engineering and transformations together with deep dive on EDA, and on churn prediction of customers (https://www.kaggle.com/c/ic20182) and fraud detection challenge (https://www.kaggle.com/c/competetion1/overview) we will use ecce homo as a simplifier of benchmark models; were we will just superficially explore the data (for further research) and perform train test split to compare ROC area under the curve to Kaggle leaderboard. We aim to fastly perform a well segmented test for model benchmarking with ecce homo automatic machine learning workflow.

As stated on the inputs section, the benchmark models, would be the Kaggle competitor's score. I do not look forward best metric on test, but a fast implementation of machine learning with simple lines of code. Not having to repeat myself every time I do start a project and have control of the process. E.g. I do want to handle missing variables, but that is it.

Analysis

Ecce Homo performs automatic Exploratory Data Analysis tasks such as:

- Dataset description, number of rows and columns, name of the features.
- Summary statistics, mean, median, first and third quartile, minimum and maximum values, as well as count.
- Group by, aggregation on categorical variables, for fast decisions and understanding of data classes.
- Boxplot plot categorical variables to have a deep understanding of the distribution of numerical data with information on the target class as well.
- Histograms plot the distribution of continuous data to understand distribution and compare distributions between the classes of target variables.
- Give brief understanding of empty values on each variable.
- Correlations print a correlation matrix with all numerical variables plus a heatmap to visually find highly correlated features.
- Show pairwise scatter plots on n random features (to easily visualize all features on high dimensional spaces). To look for patterns in data.
- Bar plots on categorical data to understand how classes are distributed among.

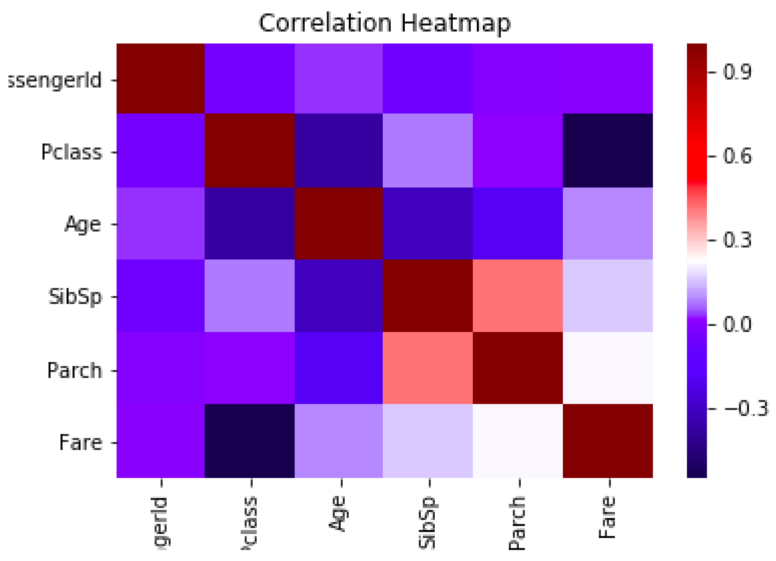
These are just simple tasks to help users find problems faster, bringing them more time to tackle down those issues with smartest solutions. The above results are written to an html file were user can visualize results and share with collaborators. Images will also be saved and simply added to markdown notation (Also helps not going through many cells). Throughout this writing we will be using titanic dataset as example of results. The former image shows the automatic information provided and correlation heatmap, that will provide insights on how to manage family on board variables such as siblings (SibSp) and kids/parents (Parch). The html file includes images and more tables that will provide information for decisions on where to allocate resources (time), for more information please visit summary_titanic_eda.html.

Exploratory Data Analysis

Summary Statistics

The notebook has 891 observations and 12 columns. Having the following names: PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked.

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200



Algorithms and techniques.

Ecce homo performs data imputation with a quite simple dictionary style of the form *{“variable”:”method”}* having multiple methods such as zero imputing, mean, median, max, min (sometimes data is missing and through max or min imputing you can create a simple more robust missing indicator variable), you can also pass a missing indicator value to fill null values and a K-nearest neighbor imputer. The latter will impute data based on the *k* nearest values, using euclidean distance. The main difference between a less robust approach like the mean or median is that we are giving an arbitrary value for all the data, with this method we want to personalize the imputer to the observations that are closer, that might have similar values. When the data is being trained, dictionary with method and imputation value will be stored to be used for test data.

SMOTE[1] and SMOTEENN[2] for data balancing. Due to the nature of our problems, performing data balance is crucial. We can use class weight hyperparameters to penalize certain class, but there are more advanced techniques to over and under sampling. SMOTE stands for Synthetic Minority Oversampling technique. It is all about creating new observations through nearest neighbors, you find the *n* nearest neighbors to a minority class observation after that the nearest neighbor observation is randomly chosen and the difference between the latter and the original observation is multiplied by a random number and add that to the original observation, you perform that for a sample of the minority class observation. SMOTEENN is the combination of three techniques: SMOTE, edited nearest neighbors[3] and Tomek links[4]. Edited nearest neighbors and Tomek links are under sampling techniques, the former removes observations that the class do not match with the rest, this for majority class, the later eliminates observations on the decision boundry.

For outlier's manipulation Ecce Homo can perform isolation forest[5] and DBSCAN[6]. Isolation forest is a more robust approximation than decision trees for outlier detection, based on splits, those observations that deserve a unique split might be consider outliers. DBSCAN stands for Density Based Spatial Clustering of Applications with Noise and can return observations that does not belong to density areas.

For hyperparameter tuning and modeling Ecce Homo performs Bayesian optimization[7] on Gradient Boosting Tree, Random Forest and Support Vector Machine on poly kernel and grid search for logistic regression with L2 regularization. Bayesian optimization is built under the idea of approximating the function to optimize, were it grows we should look forward to points closer, so, more observations might mean knowing were local minima might be found. The idea is

[1] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953
[2] https://imbalanced-learn.readthedocs.io/en/stable/index.html
[3] D. Wilson, Asymptotic" Properties of Nearest Neighbor Rules Using Edited Data," In IEEE Transactions on Systems, Man, and Cybernetics, vol. 2 (3), pp. 408-421, 1972.
[4] Tomek, "Two modifications of CNN," In Systems, Man, and Cybernetics, IEEE Transactions on, vol. 6, pp 769-772, 2010.
[5] Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. "Isolation forest." Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on.
[6] Ester, M., H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, pp. 226-231. 1996
[7] https://github.com/fmfn/BayesianOptimization

reducing the number of fittings we must perform in order to find optimizing points, at least reducing compared with grid search and the idea of testing all possible values for each parameter, that is very expensive. For three based methods, hyperparameters that have been optimized are:

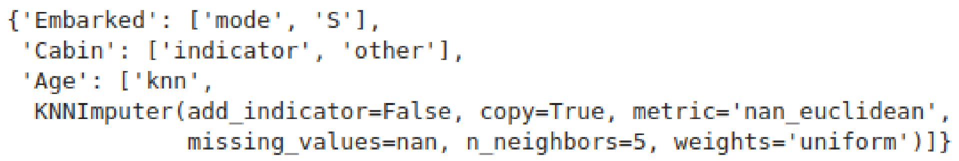
- Number of estimators (number of simple trees).
- Minimum number of samples to perform split.
- Max depth of particular decision tree.
- Minimum number of sample in a leaf.

For logistic regression only regularization parameter is being optimized and for support vector machine degree hyperparameter.

Methodology

We want the audience to be clear of the importance of feature engineering, Ecce Homo does not aim to perform it. For titanic dataset we implement simple variable transformations such as extracting title of passengers, creating a less than 16 years old variable (we have found the tend to survive) and combine family variables as explained above.

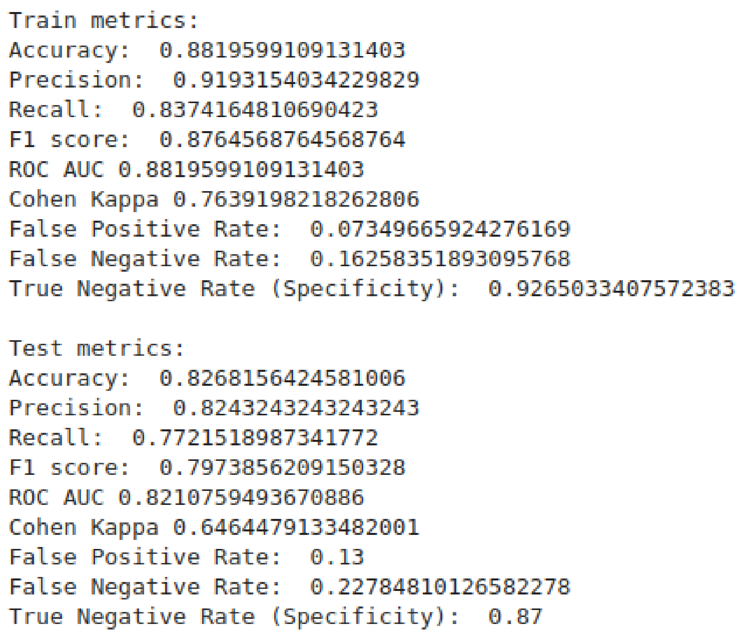
We split the data with 80-20 strategy before implementing any data imputation, if doing after, we can bias our train with test data. We impute categorical variable embarked (even though we did not use it) with the mode, we use a missing indicator for null values on cabin feature and impute age with 5 nearest neighbor approach and we also have a dictionary with the information of our imputation methods that can be used either to impute with same method or with same value from train.



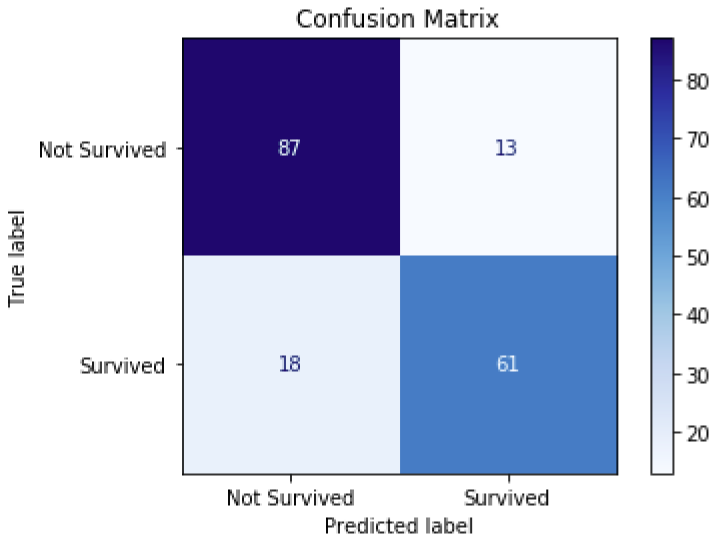
```
{'Embarked': ['mode', 'S'],
 'Cabin': ['indicator', 'other'],
 'Age': ['knn',
  KNNImputer(add_indicator=False, copy=True, metric='nan_euclidean',
             missing_values=nan, n_neighbors=5, weights='uniform')]}
```

We perform on hot variable encoding, that is creating a dummy variable for each categorical feature in “Pclass”, “Cabin”, “Sex” and “Title”. Even though “Pclass” already have numerical output, the variable is categorial and might be interpreted as unit change mean on class change with might not be necessary true.

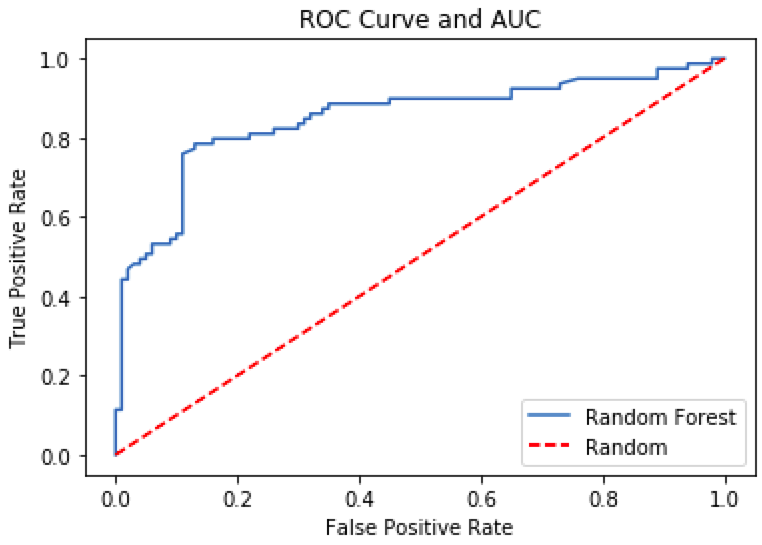
We then balance the dataset using SMOTE, obtaining around 500 observations for each class. We perform minmax scaling, since our distributions are not normal and do not have plenty outliers. Then we fit a random forest with 50 iterations using Bayesian optimization to look forward improve f1 score. We are using F1 score since we want to know wo survived and who did not and there is no clear which class is more important. We found a lower recall, compared to precision, this means we are pretty sure of those observations marked as survived really survived, but not very good at bringing a lot of possible survives. In other words, we are highly penalizing predicting survived when not really survived. Through the metric on accuracy in both train and test we can be confidently that our model is not highly overfitting the data. However, our values on the Cohen Kappa metrics that measure how well our model is compared to random decisions, in train is quite high but in test is not that high. 13% of the time we will false alarm regarding you survived when you did not. And we have an area under the ROC curve of 0.82. As we can observe in the ROC curve our random forest line lies above the random line and closer to the top left bottom. Below this histogram of the titanic leaderboard, whereby automatic imputation and hyperparameter tuning we found ourselves almost 5% above the median.



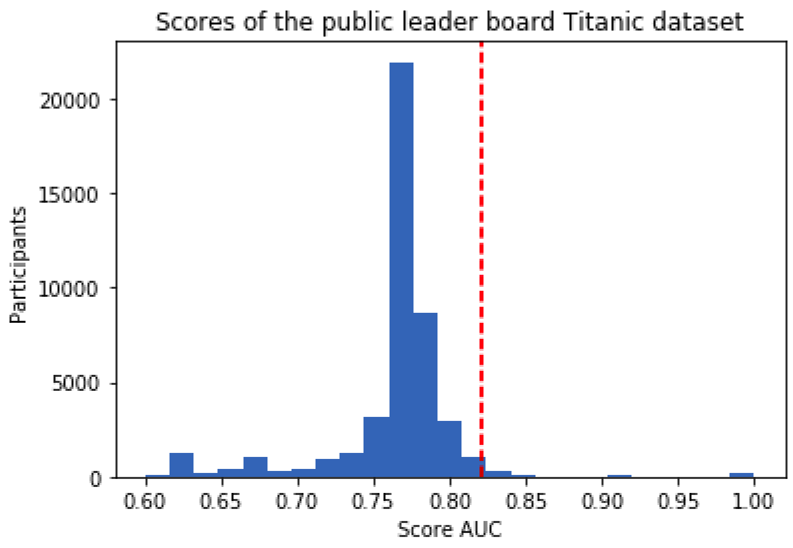
```
Train metrics:
Accuracy:  0.8819599109131403
Precision:  0.9193154034229829
Recall:  0.8374164810690423
F1 score:  0.8764568764568764
ROC AUC 0.8819599109131403
Cohen Kappa 0.7639198218262806
False Positive Rate:  0.07349665924276169
False Negative Rate:  0.16258351893095768
True Negative Rate (Specificity):  0.9265033407572383

Test metrics:
Accuracy:  0.8268156424581006
Precision:  0.8243243243243243
Recall:  0.7721518987341772
F1 score:  0.7973856209150328
ROC AUC 0.8210759493670886
Cohen Kappa 0.6464479133482001
False Positive Rate:  0.13
False Negative Rate:  0.22784810126582278
True Negative Rate (Specificity):  0.87
```



Confusion Matrix
True label
Predicted label
Not Survived
Survived
Not Survived
Survived
87
13
18
61
80
70
60
50
40
30
20



ROC Curve and AUC
True Positive Rate
False Positive Rate
Random Forest
Random
0.0
0.2
0.4
0.6
0.8
1.0



```
Median of scores:  0.7703300000000001
```

For churn and fraud datasets we use recall and precision respectively to optimize our algorithms obtaining less area under the curve ROC metric than the median leaderboard. However, feature engineering was not performed and just work as benchmarks. For churn we obtain 0.74 ROC AUC on test and the median score is 0.81. For fraud dataset the minimum value is 0.69 and we obtain 0.79. As conclusion, we can say than Ecce Homo can work to speed up machine learning engineer workflows, however you still must perform feature engineering to achieve good metrics.

Future improvements

To improve Ecce Homo machine learning Workflows, I propose:

- Prettify html files from EDA results, plus make them more resilient.
- One hot encoding approach to perform on new given category or categories not found in test set.
- More control on the modeling class or test automatically all algorithms and return the one that works the best.
- Add feature selection module.