Capstone Project

Sebastian Jimenez

Machine Learning Engineer Nanodegree

May 29, 2021

OctopusML: A Machine Learning Workflow

I. Definition

Project Overview

One task of a Machine Learning Engineer is to design and build applications that automate the execution of predictive models. The goal of this project is to implement a machine learning workflow to increase the efficiency in supervised learning specifically in **classification problems**.

The datasets used to test the workflow will be the Titanic[1] and the Diabetes[2] Datasets. The first one is useful to test the workflow in a no-complex problem and the second one adds more complexity. The Titanic dataset has *891* records and *10* features. Its class is relatively balanced with 38% / 62%. For the other side, the Diabetes dataset has 101766 records and 50 features. Its class is imbalanced 9% / 91%.

Problem Statement

When Data Scientists and Data Analysts are tackling a Machine Learning problem, there are some steps that they usually must do. Steps like: exploration data analysis, missing values analysis, outlier detection, correlation analysis, data transformation, feature engineering, choose the best metric for the problem, feature selection, hyperparameter tuning, compare models and finally put the best model in production. These can take a lot of time and it can reduce the time placed in to find insights. Some of these steps can be automated to help them to do their work easier and faster. That's why this project looks to build a workflow for ML projects specifically in **classification problems**. I hope that it brings a good baseline and encourages them to obtain better metrics.

I didn't choose to work directly with a use case, because I think that this Nanodegree is a good opportunity to improve my programming skills instead of working on a Data Science project itself. I want to finish by mentioning the following quote "a range of

[1] https://www.kaggle.com/c/titanic/data

[2] https://www.kaggle.com/c/diabetes-hospital-readmission/data

Auto-ML tools will need to be developed to support varying user goals such as simplicity, reproducibility, and reliability" (Xin, et al. 2021).

But at the end what does OctopusML return?

It returns a dictionary with 5 components:

1. train: tuple with the train set, i.e. (X_train, y_train).
2. test: tuple with the test set, i.e. (X_test, y_test).
3. models_trained: a list of tuples with the pipeline trained, i.e. [('LR', pipeline), ('LRR', pipeline), ('RF', pipeline), ('XGB', pipeline)].
4. best_model: a tuple with the best model based on the metric given ('name', pipeline).
5. metrics: a pandas.DataFrame with the following metrics: accuracy, auc roc, recall, precision and f1-score. These ones are calculated for all models in models_trained.

In addition a report.html file is generated with information about the Octopus Process.

Metrics

The idea is that the Data analyst or Data scientist can analyze several metrics around a problem that they are working on. Therefore, the workflow should return different metrics. But, in this particular case, the metric for Titanic dataset is the **Accuracy** and for Diabetes readmission dataset the **AUC ROC** will be used, with the goal to compare with the kaggle's competition.

II. Analysis

Like I said before, two datasets were chosen to test the workflow. I'm going to show the results of just one of them, in this case the diabetes dataset.

Data Exploration

This dataset has 101766 rows and 50 variables. Then, the exploration reported in the report.html file is:

First, it removes categorical variables with more than 10 categories.

Processing Report

Features' Consistency:

Qualitative features removed:

encounter_id removed because has more than 10 categories
Top 5 categories

encounter_id	relative_frecuency
173015040	0.000013
15840348	0.000026
146727732	0.000039
292167198	0.000052
2356308	0.000066

patient_nbr removed because has more than 10 categories
Top 5 categories

patient_nbr	relative_frecuency
88785891	0.000406
37096866	0.000642
88227540	0.000878
23643405	0.001114
23398488	0.001336

Also, it removes variables with a high proportion of samples (> 99%) in one category.

nateglinide removed because has a high proportion in one category

nateglinide	relative_frecuency
No	0.993017
Steady	0.999686
Up	0.999895
Down	1.000000

chlorpropamide removed because has a high proportion in one category

chlorpropamide	relative_frecuency
No	0.999135
Steady	0.999921
Up	0.999987
Down	1.000000

Also, it removes variables with just one category.

examide removed because has 1 category

examide	relative_frecuency
No	1.0

citoglipton removed because has 1 category

citoglipton	relative_frecuency
No	1.0

In this case, none of the numerical features were removed.

Quantitative features removed:

None quantitative feature was removed

Then it checks the missing values, and in this case *race* and *gender* were imputed with other. Then the outlier method LOF was used, and the amount of outliers found were 437.

Check missing values:

Feature race was imputer with "other"
Feature gender was imputer with "other"

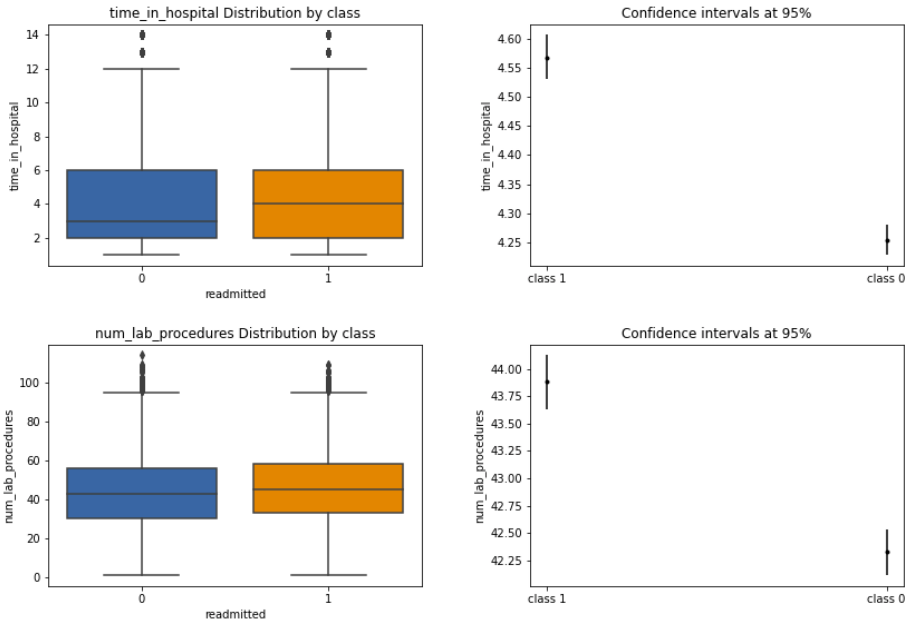
Outlier detection:

Local Outlier Factor (LOF) method used
Total outliers found: 437

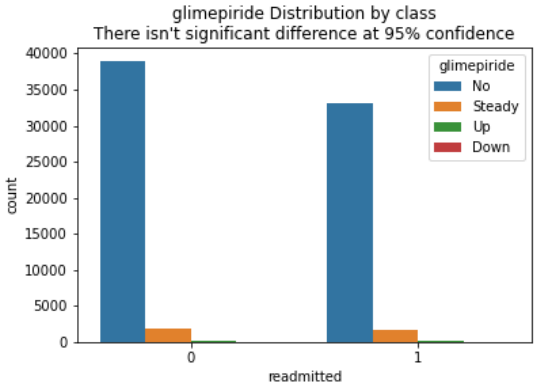
Exploratory Visualization

Then a statistical analysis is done. It's like a combination between descriptive and inferential statistics, for quantitative features, the first one is represented through boxplots and the second one using Bootstrap Confidence Intervals.

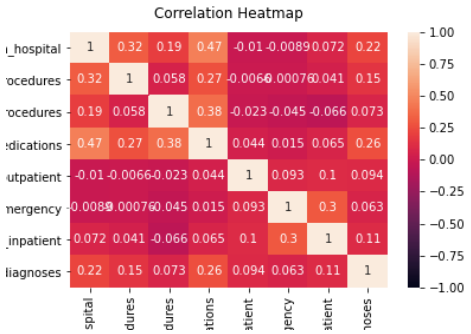
Statistical Analysis:



For qualitative features, the first one is represented through barplots and the second one using the Chi-square Test for independence.



Also, the correlation plot is shown.



Algorithms and Techniques

Regarding the outlier detection, 3 techniques are used: The first one is the **adjusted boxplot** (Hubert, et al. 2008), It's an univariate method used for skewed distributions. I used this technique because most of the numerical features are skewed and this one works fine with symmetric features as well due that is based on medcouple measure. The second one is the **Local Outlier Factor (LOF)** (Breuning, et al. 2000), It's an unsupervised and multivariate method based on distances, which computes the local density deviation between each point and its neighbors. The last one is the **Isolation Forest** (Liu, et al. 2008), It's an unsupervised method based on decision trees and works on the principle of isolating anomalies.

Then, the statistical analysis is implemented with the goal to explore the classes. The **Bootstrap** technique is used because it's a non-parametric method to compute the confidence intervals for the mean; It isn't going to depend on any theoretical distribution in this case. In categorical features, the **Chi-Square Test for independence** was implemented.

The models that OctopusML has implemented are: **Logistic Regression** without and with regularization, **Random Forest** and **XGBoost**. The first one is a straightforward method that we should try (if we can do) in any classification problem, sometimes simpler methods are better. For the second one, a **grid search cross validation** was implemented to tune the regularization hyperparameter. For Random Forest and XGBoost models, a **bayesian optimization** was implemented, because in this case we can have a huge set of possible hyperparameters to test, and this one is a smart way to try to find a local maximum for the metric given, because it doesn't need to do many iterations, due that the next iteration take into account the results obtained in the before iteration to move forward.

Benchmark

In this case the benchmark is going to be the kaggle's competition, therefore, the accuracy benchmark will be 86%, and the AUC ROC will be 69%.

III. Methodology

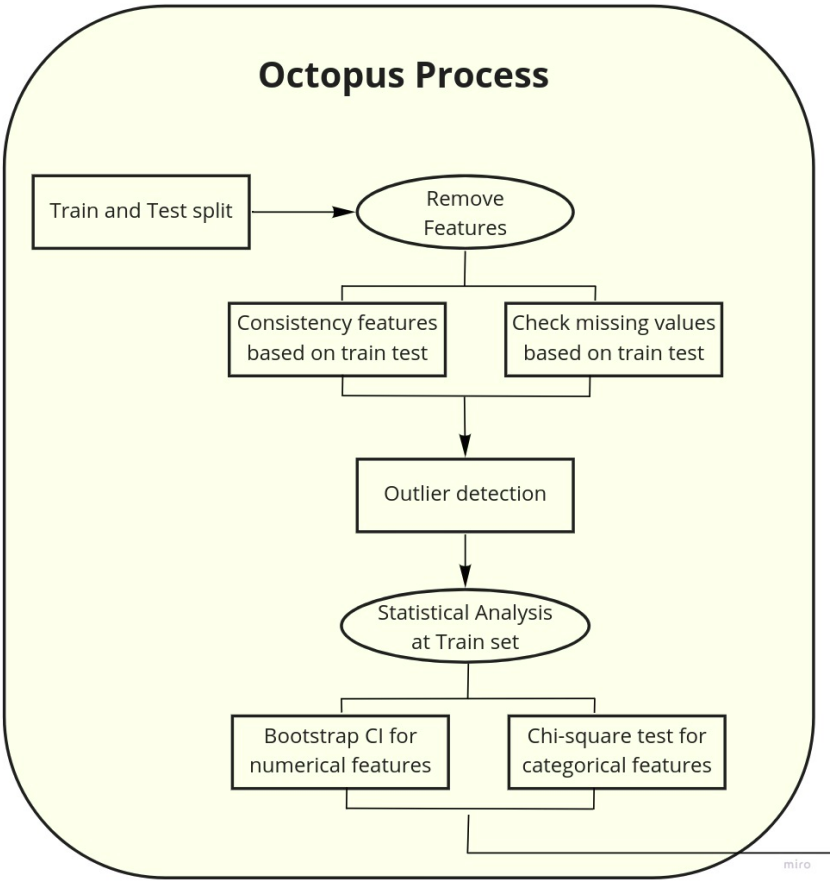
Octopus Process is the component that does all preprocessing tasks and the statistical analysis task, like you can see in the figure.

Preprocessing task

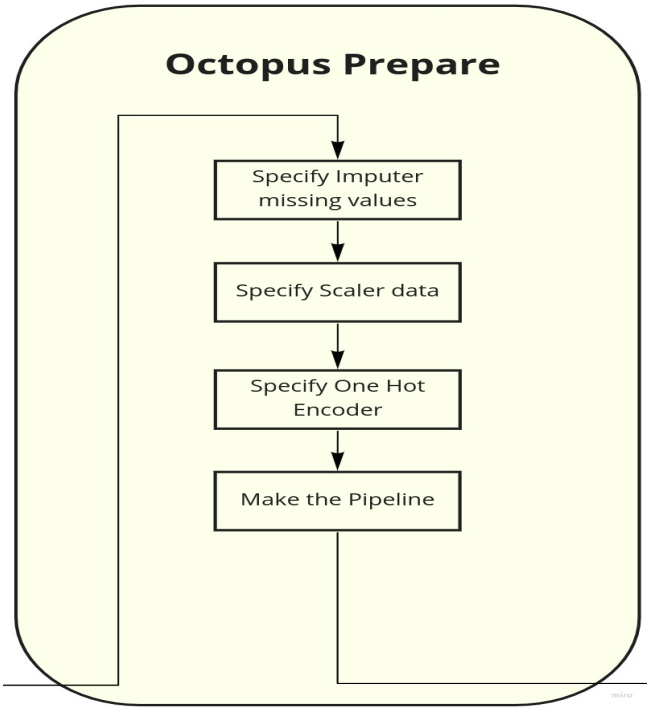
- Split data in train and test sets
- Remove categorical features:

    - It's going to allow a maximum of **10** categories in a feature. If one feature has more than 10 and it has **9** categories with more than **75%** of data, the remaining categories are going to be categorized like *'others'.*
    - If its first 9 categories don't collect at least of the 75% data, the feature will be removed.
    - Also, if the feature has just one category, it will be removed.
    - If just one category collects more than 99% of records, the feature will be removed.
- Remove numerical features:
    - If just one value collects more than 99% of records, the feature will be removed.
- Check the missing values:
    - If one feature has more than a x% of missing values, it's going to be removed. The x% value can be given by the analyst.
- Outlier detection:
    - Three methods are available:
        - Adjusted boxplot: If one sample has at least one feature like outlier, that one will be removed.
        - Local Outlier Factor (LOF).
        - Isolation Forest.
- Statistical analysis:
    - Bootstrap Confidence Intervals: Given a significance level, the app computes the confidence intervals for the mean with the goal to compare the two classes that we are trying to predict. It allows us to identify the features that can explain our target.
    - Chi-square Test: For the categorical features the Chi-square test for independence is used to identify those that aren't independent regarding to the target.

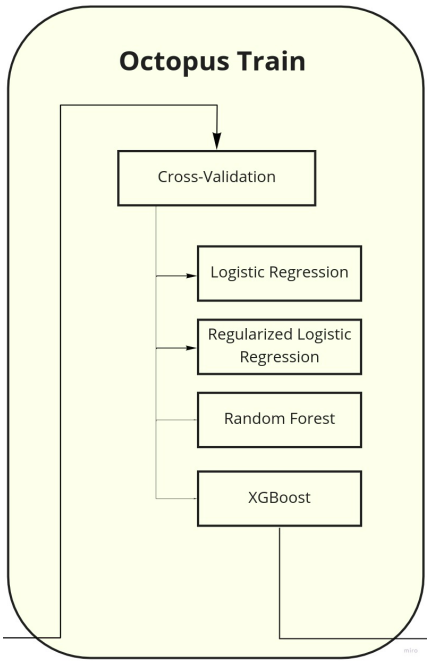
      These results are just informative, but in the future these ones can be added in a functionality to select features for instance.



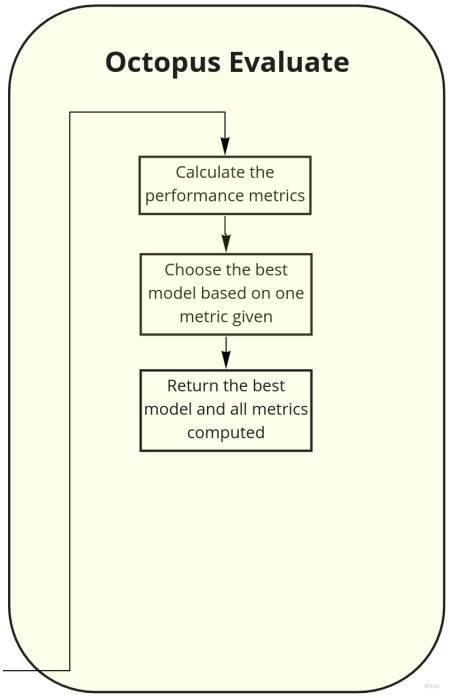
Octopus Prepare is the component that basically builds the pipeline with all transformations that are going to be used in the train process, like you can see in the figure.



Octopus Train is the component that trains 4 kinds of models proposed, like you can see in the figure. The hyperparameter tuning for Regularized Logistic Regression, the Grid Search Cross Validation is used. For Random Forest and XGBoost models, the Bayesian Optimization with Cross Validation is used.



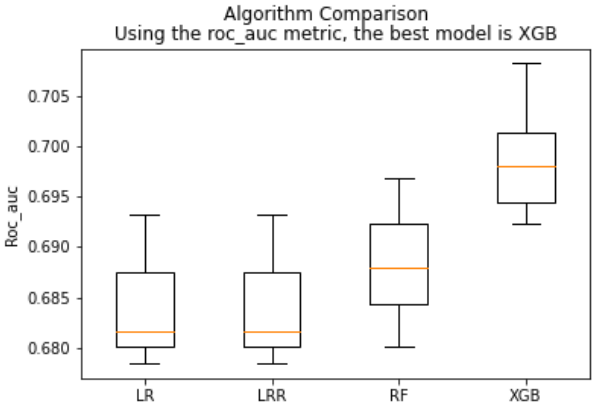
Octopus Evaluate is the component that evaluates the tuned models in Octopus Train. For a metric given, this component calculates the cross-validation performance and chooses the best model. Then, all metrics are computed with the test set.



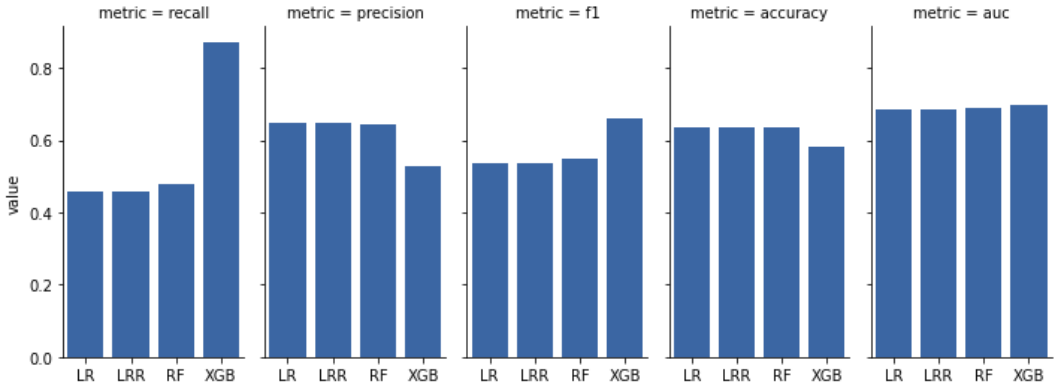
Finally, we can use each Octopus step separately, or there is a module that encapsulates all these steps, and generates useful information about the things done in Octopus Process so far.

IV. Results

OctopusML calculates the metric given for all models using cross validation, for this case, AUC ROC was computed in the 4 models and compared them. The best model, using the average auc (69.8%), was the XGBoost. This one will be returned for OctopusML.



Finally, the model is evaluated in the test set, all metrics were calculated. In addition the metrics for the rest of models are added.



In this case the metric is a little bit higher than the benchmark.

Improvements

I think that some ideas to improve OctopusML are:

- Improve function's documentation.
- More options to handle missing values (imputer).

- To add different possibilities to recognize missing values (nan, none, ?, anything specified by the user).
- Methods to deal with imbalanced datasets.
- To add at HTML the metrics results in evaluation.
- Add feature importances.
- Feature selection; some ideas:
    - Based on statistical analysis.
    - Based on feature importances.
- Recommend if the model is overfitted or not.
- Interpretable Machine Learning.
- Unit test to functions.
- To add more logs to better follow the code.

References

XIN, Doris, et al. Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows. arXiv preprint arXiv:2101.04834, 2021.

BREUNIG, Markus M., et al. LOF: identifying density-based local outliers. En Proceedings of the 2000 ACM SIGMOD international conference on Management of data. 2000. p. 93-104.

HUBERT, Mia; VANDERVIEREN, Ellen. An adjusted boxplot for skewed distributions. Computational statistics & data analysis, 2008, vol. 52, no 12, p. 5186-5201.

LIU, Fei Tony; TING, Kai Ming; ZHOU, Zhi-Hua. Isolation forest. En 2008 eighth ieee international conference on data mining. IEEE, 2008. p. 413-422.