Project Report for Machine Learning Course

Mariia Vidmuk, Daria Yakovchuk, Aleksandra Novikova

***Abstract*—This paper describes the methodology for solving the Machine Learning problem: to identify the Higgs boson. We have gone through all stages of creating a model: exploratory data analysis, pre-processing, feature processing, model optimization, and model evaluation.**

I. Introduction

The data collected by the Atlas experiment at CERN were used by physicists to improve the analysis of the elusive particle known as the Higgs boson. The objective of the analysis is to identify a region of the feature space that has a significant amount of events that are not related to the known background processes. The goal of this research is to improve the procedure that produces the selection region.

II. Model and Methods

A. Data Exploration

Our dataset consists of 250 000 data points and 30 features. The labels can take the values -1(b) and 1(s). Consequently, this is a binary classification problem, so we focused on the Logistic Regression model and KNN additionally implemented for comparison.

All variables are floating points except for ”PRI jet num”, which is an integer with a value range from 0 to 3 (and can be converted to categorical).

In this dataset we have the value of meaningless entries of some variables that are -999, which is outside the normal range of all variables. Exploring this case, we determined columns that contain vast amounts of missing values (in some columns missing values are half of the data) and will tackle the issue in the data processing part.

Likewise, we discovered that our data is unbalanced.

Label	Count
1	85667
-1	164333

B. Data Processing

1) *Imputation of missing values*. The best we could do is to replace missing values (-999) with the median(robust statistic) of the column. That procedure prevents skewness in the prediction of the model.
   First of all, the number of missing values of some columns is equal, so we can simply create new features with 0 or 1 whether there is -999, that will correspond to these columns.
2) *Log transformation*. This approach is used to convert the skewed distribution that we have in our data to a less-skewed distribution. In our case we take absolute value of x, so we can consider all features. Thus, we added the transformed features to the dataset.

$$x = sign(x) * log(1 + |x|) \tag{1}$$

3) *One-hot encoding.*. We used it to transform data from a categorical representation to a numeric representation format for the ”PRI-jet-num”. The dataset will include new variables(subsets where ”PRI-jet-num”==0, ”PRI-jet-num”==1, ”PRI-jet-num”==2, ”PRI-jet-num”==3) that take on values 0 and 1 to define the original categorical values.
   Also, the label takes 0 and 1 values for convenience.
4) *Feature Expansion and Cleaning*. To improve the model performance, we used the new feature engineering approach that is likely to expose the crucial relationships between input and target variables. In order to do feature expansion, we operated by raising input variables to power and concatenated them together.
5) *Standardization*. It is crucial to standardize the data and apply it to both training and testing datasets so that both will be transformed in the same way.
6) *Fitting with unbalanced data*. Our training model spent most of its time on negative examples and did not learn enough from positive ones. Which eventually led to a low F1 score. There are two effective ways that use to handle imbalanced data:
   - Downsample the majority class.
   - Upsample the minority class.

   We think the best decision is to downsample the majority class because it identifies the minority class better than oversampling and has a lower False Negatives Rate.
   However, we won’t use it later on in prediction, because we got pretty low accuracy after applying it to the validation set. It might work on the data with a higher amount of data.

C. Methods

1) *KNN*
   This algorithm as well as all additional functions for it and submission generation are implemented in a separate notebook *knn_algorithms* for convenience. This notebook generates a submission with accuracy 0.808 on the test data.
   We implemented a K-nearest neighbors algorithm for classification. To avoid the curse of dimensionality, the number of features was not increased for the knn method (only standardization and missing values(-999) replacement with the median was used as preprocessing).

Knn function counts predictions for all values of nearest neighbors from 1 to k at once.
Since the function requires counting of pairwise distances, we can't pass the entire train and test to it at once, since it won't fit into memory. Therefore, the test is divided into batches (for example with 512 elements).
Using cross validation, we find the best value of k. By plotting the accuracy for all k from 1 to 1000 it is easy to see the point of global maximum (Fig. 1). The maximum value of accuracy is reached at cross validation at k = 29 with accuracy 0.80592.
For this value (k = 29) classes are further predicted already for the real test data.

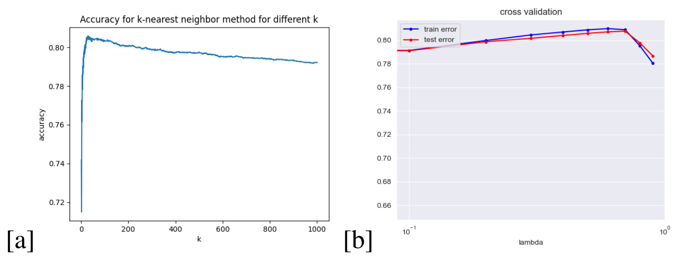

Fig. 1. Accuracy plot for Knn [a] and Logistic Regression [b] methods on cross validation depending on the values of k and lambda respectively

2) *Logistic Regression.*
It is transparent that the most appropriate model to use is Logistic Regression which satisfies our classification task. And in practice, it works faster than KNN. We attempted both Logistic Regression with and without regularisation. In most cases, our model didn't overfit, as shown below. So eventually, we have chosen the Logistic Regression without regularization as our main model to work on.
Since we needed to specify hyperparameters, which is the step size(γ) for gradient descent and degree of polynomial feature expansion, we used Grid Search to find optimal values in 2-fold Cross-Validation to get the best accuracy score. (best gamma $\approx$ 0.6 for 500 iterations, best degree = 2) But ultimately, we came up with an idea to reduce the step size for each 10 000th iteration(out of 40 000), so we used the step size of whatever we prefer with degree 2 to get to the optimal solution eventually.
The train and test accuracy results are presented in the following table for Logistic Regression and KNN.

On the validation set:

Model	Train Accuracy	Test Accuracy
KNN	0.80592	0.808
Logistic Regression	0.816035	0.8135

In the platform:

Model	Accuracy	F1 Score
KNN	0.808	0.711
Logistic Regression	0.812	0.715

The numbers above were produced by splitting the training dataset into two parts (one for training and another for validation) and applying data preprocessing. Another table will show the efficiency of our models submitted to the online platform https://www.aicrowd.com/ to validate the prediction of the test dataset.
Observing the results, we can conclude that Logistic Regression performs better and faster in comparison to KNN.

D. *Possible methods to improve our accuracy:*

1) As our dataset is unbalanced and in this problem, it is crucial to classify "Higgs boson", Random Forest Classifier could possibly be a better solution with its penalization of misclassifying the minority class.
2) It would be interesting to try to feed to one model just "raw" features and to another one - variables computed by physicists. Eventually, the final model will summarize the predictions of the two previous ones.
3) Since we consider only 20 possible gammas and 3 possible degrees, it is not enough for getting truly optimal hyperparameters. A possible solution for it would be the Optuna optimization framework that in fact optimizes hyperparameters.

III. Summary

This project shows explicitly why preprocessing is so important. We went through the main strategies to improve our model and got the result of percentage of accuracy 81 with Logistic regression. But we still believe that there is a lot to discover in this problem, especially using more advanced techniques.

References

[1] Feature Transformation
https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-and-scaling/
[2] Feature Engineering
https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-feat
[3] One-Hot-Encoding
https://vitalflux.com/one-hot-encoding-concepts-python-code-examples/