# Higgs Boson ML Project Report

*Gaurav Pasari, Karunya Tota, Oussama Abouzaid*
Department of Data Science, EPF Lausanne, Switzerland

## I. Introduction

THE discovery of the Higgs boson particle was announced in late 2012. However, this discovery embarked a new quest to identify the particle's characteristics and determine if it fits the present model of nature. The ATLAS experiment at CERN has identified a signal of Higgs boson decaying into two tau particles. But this signal is quite small and hidden in background noise. The overarching goal of the original Kaggle Higgs Boson Machine Learning challenge was to *explore the potential of advanced machine learning methods to improve the discovery significance of the experiment.* In other words, the required task was to identify if independent recordings of the particle classified as "tau tau decay of a Higgs boson" or "background noise". Our course's project is based on this challenge. [1]

## II. Preprocessing

Prior to applying any algorithm to generate the weights from our training data, we preprocessed the original data using different methods: (1) partitioning into buckets, (2) cleaning undefined values, (3) normalization, (4) polynomial, (5) one hot encoding, and (6) concatenating an intercept column. Further descriptions for each method are provided here:

**1. Partitioning into Buckets**
We observe that the column `PRI_jet_num` has 4 categorical values - $0, 1, 2$ and 3. For each such value, the `DER_mass_MMC` is either defined or $-999.0$. Hence, we divide the entire dataset into 8 buckets on the basis of 4 `PRI_jet_num` values and if `DER_mass_MMC` is defined or undefined. The idea is to use ridge regression on each of the 8 buckets to generate a set of weights for each corresponding bucket. Finally, when we are given the test data, we partition it into 8 buckets as well. Then, we use the corresponding set of weights from training to predict the y values for the corresponding bucket.

**2. Cleaning Undefined Values**
A quick look at the dataset reveals that there are many undefined values indicated by "-999". In order to prevent these values from interfering with the training of our model, we decided to replace them with the average of all of the other values in the feature column.

**3. Normalization**
We standardized our data by subtracting each feature value by the mean and then dividing by the corresponding standard deviation. In the cases of an undefined value, we only replaced the "-999"s with the mean of the feature column. If the mean was undefined as well, we replaced it with 0.

**4. Polynomial Fit**
In order to enrich our model and increase the representational power, we augmented our model by appending polynomial features to our original training data, up to degree $d$ = 13. This improved our accuracy significantly.

**5. One Hot Encoding**
We also observed that column 22 "PRI_jet_num" contains categorical data identifying the specific jet value ranged from 0 to 3 for that recording. Therefore, we decided to use one hot encoding to transform these variables into a binary representation because it works better for classification and regression algorithms. Hence, this column is removed from the training set and replaced by 4 columns containing 0's and 1's to identify the jet value for that row.

**6. Intercept Fit**
When generating the weights or coefficients for a regression model, it is common to append a column of ones to the training data to correspond with an intercept for the final model. We realized that slightly better results were produced when fitting our model with an intercept.

## III. Methods & Results

We tested with several methods to generate our model: Least Squares, Least Squares with Gradient Descent, and Ridge Regression with $L_2$-Regularization. Before preprocessing the data, we split the training data into two subsets: 80% for a local training set $S_{train}$ and 20% for a local testing set $S_{test}$.

After testing with the Least Squares method *(accuracy = 66.48%)* and the Least Squares using Gradient Descent *(accuracy = 53.08%)*, we decided to use Ridge Regression to generate our model as it gave us a better accuracy. We then ran a grid search to find an optimal $\lambda$ and polynomial degree $d$ to give us a high accuracy after training our model on $S_{train}$ and testing it on $S_{test}$, as shown in the following table:

Here is an overview of the grid search to find the best $\lambda$ :

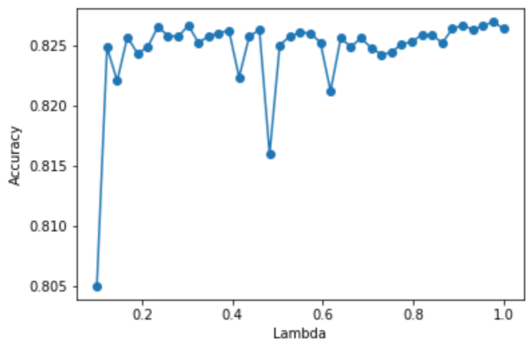

Table 1 contains the optimal parameters and accuracy from the grid search.

| | |
|---|---|
| $\lambda$ | 0.9 |
| degree $d$ | 13 |
| accuracy | 82.676 % |
| Kaggle accuracy | 83.035 % |

TABLE I: Optimal parameters & accuracies

## IV. Conclusion

Our highest accuracy *(83.035%)* on the Kaggle leaderboard is produced by preprocessing our training data using the methods described earlier and generating a model via Ridge Regression with $L_2$-Regularization. Further ways to improve our model could be to discover more efficient methods of preprocessing and using a k-fold cross validation technique to ensure that we are not overfitting the data.

## References

[1] Higgs Boson Machine Learning Challenge. https://www.kaggle.com/c/higgs-boson