**MACHINE LEARNING FOUNDATION(INT 247)**

**MACHINE LEARNING MODEL FOR HTRU2 DATASET**



Student Name: Joseph Kobina Ansah Acquah

Roll Number: 36

Email ID: [kobiansah@gmail.com](mailto:kobiansah@gmail.com)

Section: K18XC

Date of Submission: 12/04/2021

Faculty Member: Dr. Sanjay Kumar Singh

**CONTENTS**

1. ABSTRACT
2. DATASET
3. METHODOLOGY
4. EXPERIMENT SETUP
5. RESULT & DISCUSSION
6. CONCLUSION
7. REFERENCES

**ABSTRACT**

Pulsars, a rare type of Neutron star produce which produces radio emission detectable here on Earth, are of high interest to space scientists but the detection often requires a large-scale computation and the participation of astronomical experts. Each pulsar produces a slightly different emission pattern, which varies slightly with each. Thus, a potential signal detection known as a 'candidate', is averaged over many rotations of the pulsar, as determined by the length of an observation. In the absence of additional info, each candidate could potentially describe a real pulsar. However, in practice almost all detections are caused by radio frequency interference (RFI) and noise, making legitimate signals hard to find. The problem here is basically a binary-classification problem. Machine learning systems have been adopted to solving pulsar automatic detection problems. HTRU\_2 is a data set with 17,898 pulsar candidates. In the absence of additional info, each candidate could potentially describe a real pulsar. However, in practice almost all detections are caused by radio frequency interference (RFI) and noise, making legitimate signals hard to find. The constraint of this problem is that it is complicated to derive a physical model for this problem and also interpreting require some expertise in physics and astronomy.

A lot of researchers have applied machine learning techniques to establish a robust identification classifier to detect real pulsars from noise. Various researchers used Artificial Neural Network to process over millions of pulsar candidates obtained by reprocessing data from surveys. They were able to reject 99% of the noise candidates and detect 85% of the pulsar candidates through a blind analysis. Various stream classifiers were studied and they revealed the sensitivity of pulsar data to

For this report, I applied various machine learning algorithms to the dataset, namely logistic regression, decision trees, random forest, and SVM to build a binary classifier that is able to identify the pulsar candidates.

**DATASET**

Pulsar candidates collected during the HTRU survey. Candidates must be classified in to pulsar and non-pulsar classes to aid discovery. As pulsars rotate, their emission beam sweeps across the sky, and when this crosses our line of sight, produces a detectable pattern of broadband radio emission. As pulsars rotate rapidly, this pattern repeats periodically. Thus pulsar search involves looking for periodic radio signals with large radio telescopes.

Each pulsar produces a slightly different emission pattern, which varies slightly with each rotation Thus a potential signal detection known as a 'candidate', is averaged over many rotations of the pulsar, as determined by the length of an observation. The candidate data sets as binary classification problems. Here the legitimate pulsar examples are a minority positive class, and spurious examples the majority negative class. At present multi-class labels are unavailable, given the costs associated with data annotation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Set Characteristics : | Multivariate | Number of Instances: | 17898 | Area | Physical |
| Attribute Characteristics: | Real | Number of Attributes | 9 | Date Donated | 2017-02-14 |
| Associated Tasks: | Classification, Clustering | Missing Values | Null | Number Of Web Hits | 73040 |

Attribute Information:

Each candidate is described by 8 continuous variables, and a single class variable. The first four are simple statistics obtained from the integrated pulse profile (folded profile). This is an array of continuous variables that describe a longitude-resolved version of the signal that has been averaged in both time and frequency. The remaining four variables are similarly obtained from the DM-SNR curve. These are summarised below:

ATTRIBUTES

1. Mean of the integrated profile.

2. Standard deviation of the integrated profile.

3. Excess kurtosis of the integrated profile.

4. Skewness of the integrated profile.

5. Mean of the DM-SNR curve.

6. Standard deviation of the DM-SNR curve.

7. Excess kurtosis of the DM-SNR curve.

8. Skewness of the DM-SNR curve.

9. Class

HTRU 2 Summary

17,898 total examples.

1,639 positive examples.

16,259 negative examples.

**METHODOLOGY**

IMPORT HTRU2 DATASET

EDA & FEATURE IDENTIFICATION

DATASET

FEATURE ENGINEERING

**MODEL SELECTION**

UKNOWN TEST DATA

TRAIN DATA

TEST

TRAIN

DUMMY CLASSIFIER TO SEE WHERE TO EXPECT RESULTS

IMPLIMENT MODELS

IMPLEMENTED WITHOUT SAMPLED DATA

WITH UNDERSAMPLED DATA AND CROSS VALIDATION

WITH OVERSAMPLING DATA AND CROSS VALIDATION

HYPER-PARAMETER TUNED MODEL

**EXPLORATORY DATA ANALYSIS & FEATURE ENGINEERING**

The dataset is clean and does not contain missing data with 8 attributes. The issues that need to be addressed are the over-fitting problem due to imbalance in the dataset that can cause false scores and the other is the large distribution range of the features.

**STANDARDIZATION**

In standardization we take care of the variation in the 8 features. The maximum value can reach 1191 and the minimum 0.2. Standardization is involved to prevent the model from being drastically influenced due to change in values of the features.

**SAMPLING**

Since there are not many candidates found, the positive class volume is small, while the negative class volume has 16259, which is about tens of times the positive candidates. Direct training with the provided data can easily lead to over-fitting of negative samples. Under-sampling and over-sampling, have been used to solve this problem in training the model which can modify the number of positive and negative samples in this dataset.

**Random Under-Sampling**

Random Under-sampling aims to balance class distribution by randomly eliminating majority class examples. This is done until the majority and minority class instances are balanced out. It can help improve run time and storage problems by reducing the number of training data samples when the training data set is huge. It can discard potentially useful information which could be important for building rule classifiers. The sample chosen by random under sampling may be a biased sample. And it will not be an accurate representative of the population. Thereby, resulting in inaccurate results with the actual test data set.

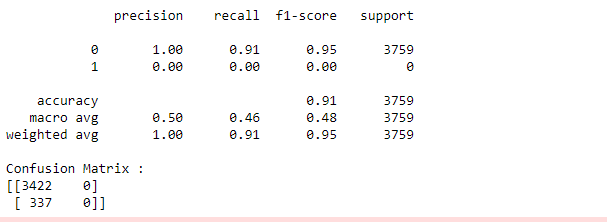
**Random Over-Sampling**

Over-Sampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample. Unlike under sampling this method leads to no information loss. Outperforms under sampling. It increases the likelihood of overfitting since it replicates the minority class events.

**MODEL SELECTION**

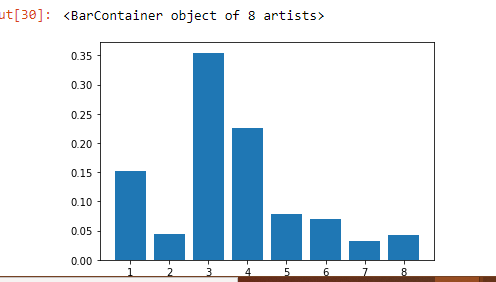
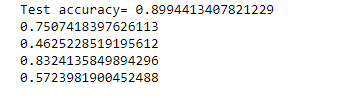
After the dataset is finally ready, it is further split into two datasets, train dataset and unknown test dataset. The unknown test dataset is what would be used to implement the final model.

The train dataset would further be split into two datasets, train datasets and test datasets. The first classifier used is Dummy Classifier Picking an appropriate metric is an essential yet challenging part of any data science project. If we pick a metric not suited for the evaluation of our models, we could be led to choose a wrong model, whose generalization ability is poorer than expected. First I used a dummy classifier that labels all instances as negative (0) to see how common classification metrics should not be used with an imbalanced dataset as they are misleading about the actual performance of the model.

The dummy classifier results are 

These are clearly false assumptions regarding the performance since the scores for positive values have 0.

Next I use random forest classifier on the raw dataset to study the dataset because it handles high-dimensionality very well, extremely versatile, requires very little pre-processing , avoids overfitting and most importantly it helps check the importance of the features.



Logistic regression predicts the probability of an outcome that can only have two values. The prediction is based on the use of one or several predictors (numerical and categorical). Logistic regression is a linear model, which has good anti-noise performance. The number of features in this problem is not large, and thus logistic regression can be applied to this problem.

The model is implemented using sklearn.linear\_model.LogisticRegression The hyper-parameters are penalty (l1 or l2, used to specify the norm used in the penalization) and C (inverse of regularization strength).

Support Vector Machine (SVM) optimizes over support vectors and associated hyperplanes. This method is often called large margin classifier. The model is implemented using sklearn.svm.SVC. The hyper-parameters are *C* (penalty parameter of the error term), *kernel* and *gamma* (kernel coefficient).

After performing the models on the raw data. I resample the data with over- sampling and and under-sampling whiles also cross validating each one.

**HARDWARE/SOFTWARE SETUP**

* ANACONDA ENVIRONMENT/ JUPYTER NOTEBOOK
* WINDOWS 10 OS

**RESULT AND DISCUSSION**

Resampling methods work for tree-based algorithms, namely Decision Tree and Random Forest. It makes sense because Decision Tree is sensitive to class imbalance, and Random Forest is built upon decision trees. Logistic regression model is a probabilistic model which means it does not care much about unbalanced problem. However, it still shows improvement on F1 scores when using resampled data set.

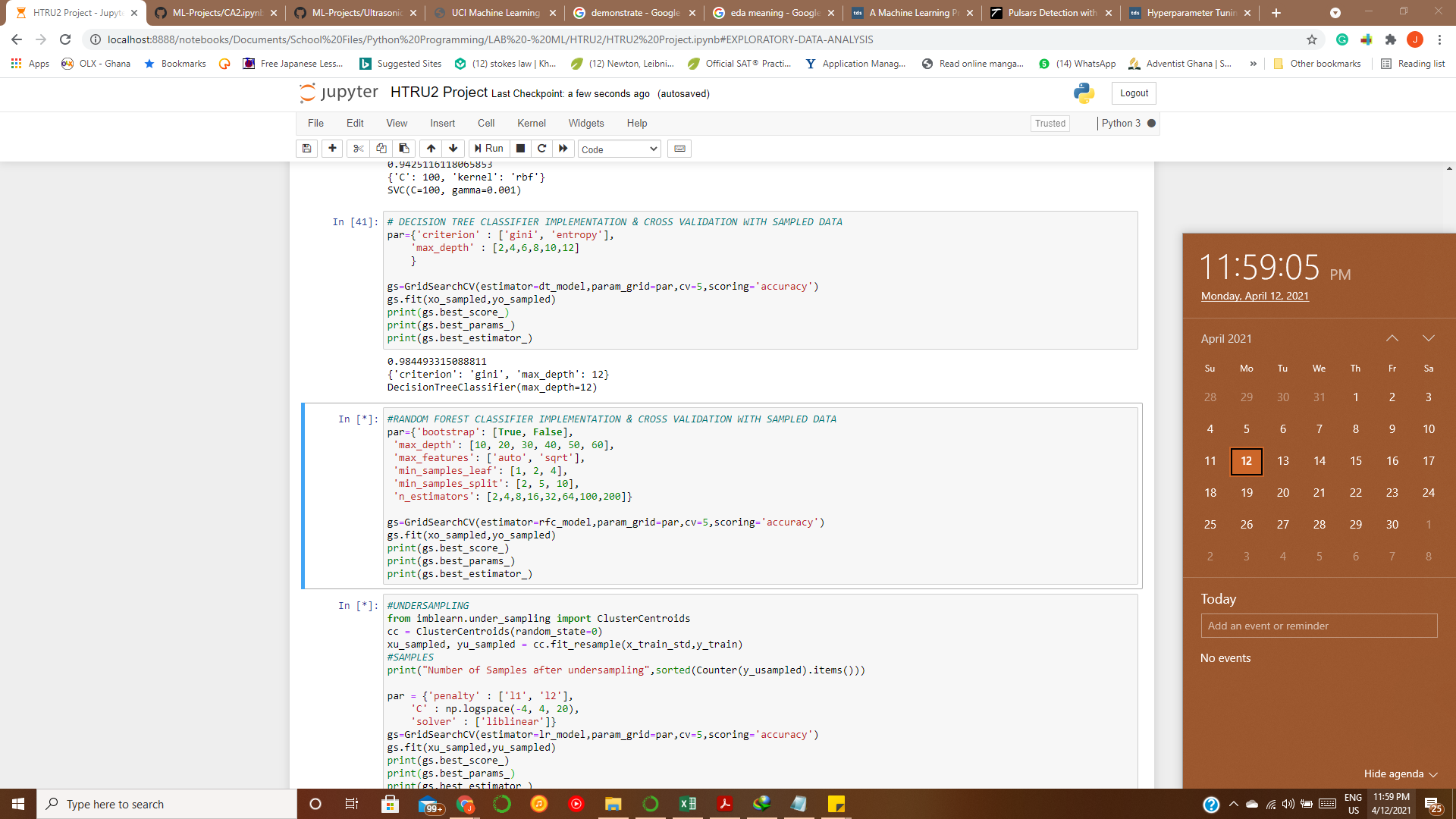
SVMs work fine on sparse and unbalanced data. Class-weighted SVM is designed to deal with unbalanced data by assigning higher misclassification penalties to training instances of the minority class. From Figure 3, it is clear that using random over-sampling is the best resampling choice for this problem.

As mentioned above, I split a test set from the training set to do hyper-parameter tuning. The original test set remains untouched. I use the gridseach to optimize the hyper-parameters, and each time through cross validation to get the model's score. Specifically, I define in advance what the hyper-parameters are, and their scope, and then the method picks the best hyper-parameters one by one to pick the best classifier. The function I used here is sklearn.model\_selection.GridSearchCV.

**CONCLUSION**

In this project, I aimed at building a classifier based on HTRU2 data set. First, I tested several resampling methods (random under-sampling, random over-sampling) to handle with unbalanced data issue. It turned out that random under-sampling works well with this dataset. Then a variety of machine learning methods (logistic regression, decision trees, random forest, SVM) are experimented on this resampled data set. According to the final result, the simple KNN model outperforms all other models. However, due to lack of time from cross validating the various samples and then going back to tuning each one , I was unable to show the highest accuracy with a confusion matrix . This is interesting to do in the future.

CHECKPOINT



LINKS AND REFERENCES

* <https://imbalanced-learn.org/stable/under_sampling.html>
* <https://archive.ics.uci.edu/ml/datasets/HTRU2>
* <https://www.kdnuggets.com/2020/01/5-most-useful-techniques-handle-imbalanced-datasets.html>
* <https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28>
* <https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e>
* <https://towardsdatascience.com/logistic-regression-model-tuning-with-scikit-learn-part-1-425142e01af5>
* <https://towardsdatascience.com/a-machine-learning-project-predicting-used-car-prices-efbc4d2a4998>