Machine Learning: Assigment 1 Report

Mário Gomes, 54896, Analysis and Engineering of Big Data
Miguel Calado, 54491, Analysis and Engineering of Big Data

***Abstract*— This document is the first assignment report for the Machine learning course. The project aims to compare three machine learning classification models (Naïve Bayes Classifier, Logistic Regression and K-Nearest Neighbors) using the banknote authentication dataset, obtained from the UCI Machine Learning Repository, to effectively predict if a given note is genuine or forged based on the data extracted from the notes image. After preprocessing the data and finding the best parameter for each classifier, the McNemar's test revealed that the Naïve Bayes model was significantly different from the other models.**

I. Introduction

Identification of fraudulent banknotes is a challenging task often resulting in ambiguous labels. Even professional observers sometimes disagree on the label given an image of a banknote. It is a difficult problem that pushes the limits of the visual abilities for both humans and computers.

The UCI Machine Learning Repository dataset provides 4 features (Wavelet Variance, Wavelet Skewness, Wavelet Curtosis and Image Entropy) and the true label, that can be preprocessed and used in a machine learning classifier, with the proposed of predicting a forged banknote. The project objectives are to use preprocessing tools to this dataset and preform K-Fold Cross-Validation in order to obtain the best values of the hyperparameter associated with a given model, then training the model and testing it in order to get a sense of its predictive power and some other associated metric measures. In the end, McNemar's statistical test is used to check significantly differences between models.

II. Cross-Validation and parameter tunning

K-Fold Cross-Validation is associated with finding the parameters that will, most likely, minimize the true error of the model. Given a train set, the algorithm partitioned it into k equal length folds, from which k-1 will be used for training the model and the last fold will be used as a validation fold, this is, for having a sense of the error committed by the model trained. This is an iterative process in which each one of k folds is used for validating and training the model which leads to preventing overfitting and to get a sense of the predictive power of the model based on that data. After the iterative process has terminated, the minimum of the validation error across the k folds will identify the optimal parameter, since each iteration is associated with one and only one parameter. It is also important to mention that given that the validation error was used to pick the best parameter, with which the model will be trained, it is no longer an unbiased estimate of the true error committed.

After preprocessing the data (scaling it between -1 and 1) and splitting it into the train set and test set, which will later be used to get the true error of the model, the classifiers and parameters that were chosen to be cross validated are as shown below:

- Logistic Regression: Regularization Parameter (C);
- KNN (K-Nearest Neighbors: Number neighbors (K);
- Naive Bayes : Band width of the Kernel Density.

The parameter C was tested in the range of 2^N , where N is in between 1 and 20. To find the best C, a Logistic Regression model was trained so that the train error and validation error could be evaluated. This is done in order to check for overfitting. The lowest validation error corresponds to the optimal C. The plot of the errors against the logarithm of the C value is shown bellow.

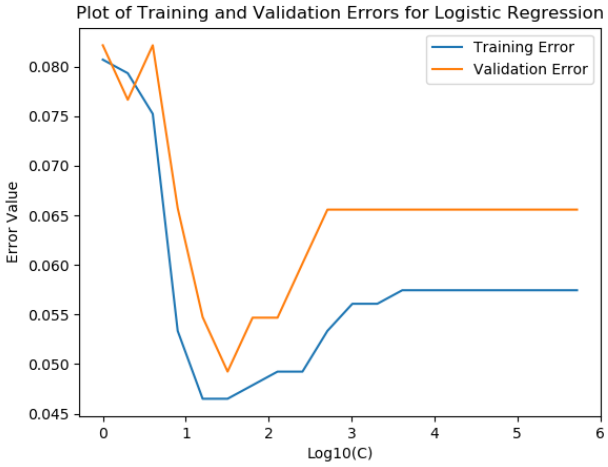

Fig. 1. Training and Validation errors against the Log of C

The parameter K was tested in the range of 1 to 39. The same process was applied, where a KNN model was trained and the train and validation error were plotted.

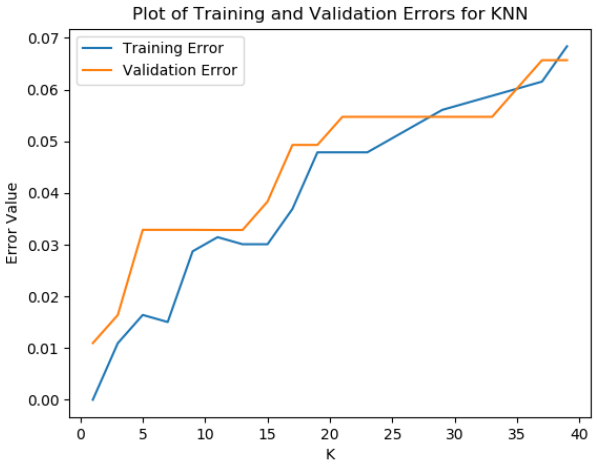

Fig. 2. Training and Validation errors against the K

The best band width was tested from 0.01 to 1 with a step of 0.024 and the same procedure was applied, like previously explained. The graph is as it follows.

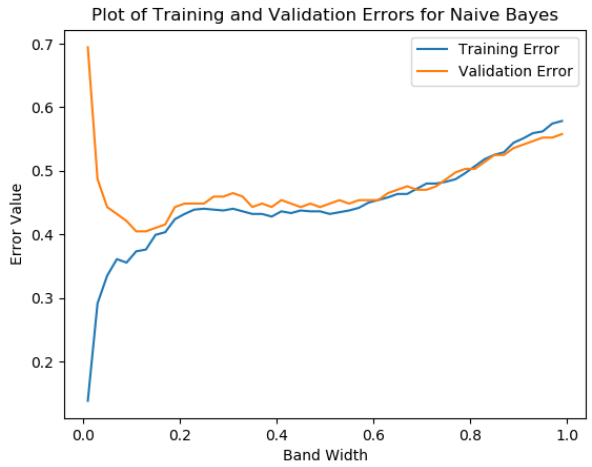

Fig. 3. Training and Validation errors against the Band Width

It is also to note that the validation and train errors in the models were done by subtracting 1 by the mean accuracy in each fold.

III. Logistic Regression and C Regularization

The Logistic Regression model is one of the most common and well-known classification models of supervised learning, in part this has to do with its easy probabilistic understanding and simple idea. The concept behind it is to obtain an optimal decision boundary hyper-plane that best separates the two classes and then considering its equation as part of the sigmoid function, so that the classifier can approximate the probability of a given vector belonging to a class C_1. As any other machine learning model, logistic regression can overfit and in order to prevent that, the classifier starts modeling the noise inside the training data set and fails to generalize well to unseen data. It is often used L2 or L1 regularization. Both regularization methods differ in the way they penalize the model, as it is shown below:

$$\text{L2 regularization method} : \lambda \sum_{j=1}^{n} \beta_j^2$$

$$\text{L1 regularization method} : \lambda \sum_{j=1}^{n} \mid \beta_j \mid \tag{1}$$

λ refers to a positive constant that defines the magnitude of the regularization, higher values of this hyperparameter will lead to an increasing penalization on the model and consequently a more smoother classifier function. In the logistic regression classifier, λ equals $1/C$, so in this sense, higher values of C determine a smaller value of λ and concomitantly a less smoother classifier, on the other hand, little values of C determine high values of λ and a more smoother function.

IV. K Nearest-Neighbours and K Regularization

The KNN classification model is a lazy learning algorithm well known due to its simple idea and high need of computational power. Given the k nearest points of another one, the model will predict the target of this one to be the most frequent target among those k nearest points. Since it is a lazy learning algorithm it does not learn a discriminative function that maps from the data to the target, but it ”memorizes” the training set. The parameter to be optimized on this model it is the number of nearest points k to consider in order to make a prediction. Choice of k is very critical A small value of k means that noise will have a higher influence on the result. A large value make it computationally expensive and sort of defeats the basic philosophy behind KNN (that points that are near might have similar densities or classes).

V. Naïve Bayes and Bandwidth Regularization

The Naïve Bayes classifier predicts the targets of a given data set based on the joint probability between each feature of the data set and each possible target value and their respected frequency on that data set, under the assumption that each feature is conditionally independent from the others.

In order to estimate the joint probability distribution of each feature given each target value it was used a Gaussian kernel density estimator, a non-parametric model that takes a bandwidth value argument. The bandwidth will control the smoothness of the given kernel density function, since it defines the length of the range of values in which each gaussian distribution will be built, in the end, the kernel density function will be sum of all this gaussian distributions

previously defined. If the band width value is very high, the kernel will "traverse all points", which will result in overfitting. The oposite will happen when the value is very low, underfitting. This can be seen in the Figure [3], where very high values result in bigger validation and training errors.

Finally, the classifier will predict the target of a given sample to be the one that maximizes the sum of the joint log-probability for all features plus its prior probability. The corresponding equation in shown below:

$$C^{\text{Naïve Bayes}} = \underset{k \in \{0,1,\ldots,K\}}{\operatorname{argmax}} \ln p(C_k) + \sum_{j=1}^{N} \ln p(x_j | C_k)$$

To implement this classifier a class was created, *NaiveBayesClassifier.py*, with the intent of imitate a machine learning classifier, like Logistic Regression. The class is initialized with the following attributes: a bandwidth value (by default is one), a list that will hold the prior probabilities of the targets in the data set and a dictionary whose keys are pairs of integer numbers, the first one is with respect to the position of the feature and the second with respect to the target category, the values of the dictionary are kernel density estimators that attempt to estimate the probability density function of the feature's values that are associated with the given target category. In this situation since threre's four features and two target categories, the dictionary will have eight elements.

The fit method of the class will fill the list with the two proper prior probabilities and fit the proper data to the kernel density estimator associated with each one of the eight keys in the dictionary.

The predict_probs method returns an array with the number of lines equal to the number of samples in the data set passed as argument and two columns, each entry corresponds to the probability of the sample belonging to the class zero (first column) and class one (second column).

The predict method is the one that predicts the target of each sample to be the one that has higher probability, returning the result as an array with one column and number of lines equal to the number of samples of the data set passed as an argument.

The score method returns the accuracy score of the predicted values, that is, the fraction of correct classifications.

VI. McNemar Statistical Test And Model Comparisons

The McNemar's test is used to evaluate significant differences between different models. With a 95% confidence interval, if the calculated value between models is greater than 3.84, the null hypothesis, H_0, is rejected (the two classifiers perform identically). Below there's a table with the effectuated test across all models:

Models 1 vs Models 2	McNemar's Test	Model 1 Error	Model 2 Error
LR vs KNN	2.25	4	0
LR vs Naive Bayes	12.89	4	24
Naive Bayes vs KNN	22.04	24	0

TABLE I
McNemar's test results

By inspection of the table [I], Logistic Regression and KNN don't have a significant difference, although, KNN is less prompt to misclassifications, having zero while Logistic Regression has four.

The biggest difference among all models is the Naïve Bayes, having significant differences and more misclassifications then the other two models.

VII. Results and Discussion

The results of the cross validation are as follows: C=32, K=1, Bandwidth=0.109. It's also to note that these values might change if the random seed is modified.

To have a better perspective of the overall performance of the models, a table containing different metrics was created. The outcome is below:

Model	ROC AUC	Accuracy	Precision	Recall	F1-score
LR	0.992	0.991	0.981	1.0	0.99
KNN	1.0	1.0	1.0	1.0	1.0
Naive Bayes Classifier	0.946	0.948	0.954	0.926	0.940

TABLE II
Different metrics across all Models

In summary, the worst performing model is Naïve Bayes. This might be due to multiple facts that are present in the dataset. The first one is a bit of class imbalance in the target. This will affect the calculation of the prior probability, that might affect the overall classification.

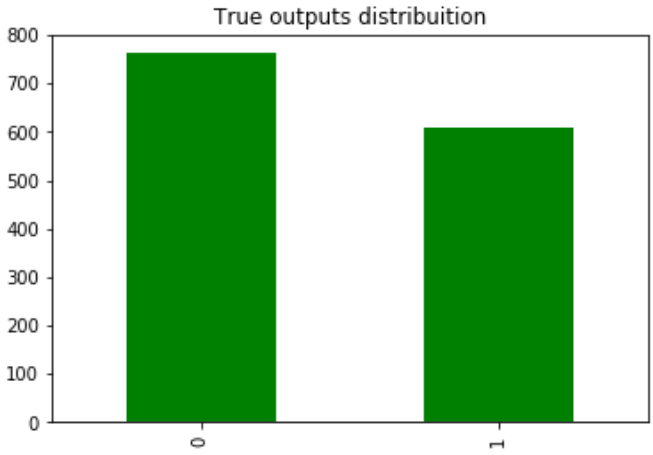

Fig. 4. Distribuition of the target

The second one is related to one of the premises of Bayes Theory, that where the features must be independent to each other. A way that could check for independence is by plotting a correlation matrix. In this case it was used a package from python, *sns*, that graphs a heatmap containing the correlation between the data.

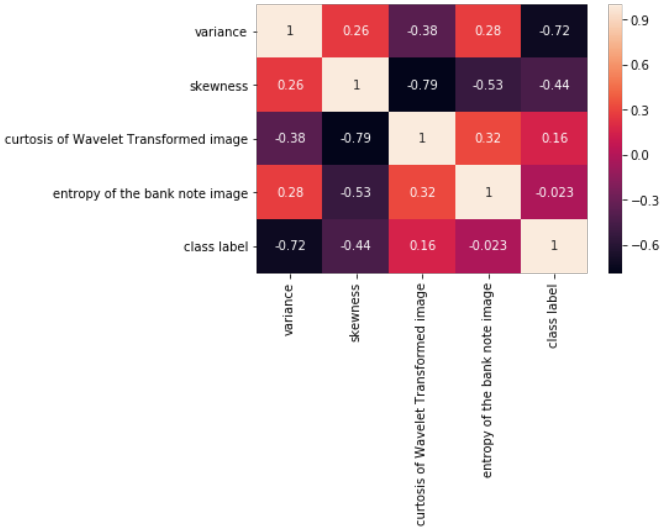

Fig. 5. Heatmap of the correlation matrix

It was also plotted a pair plot to support the Figure [5]:

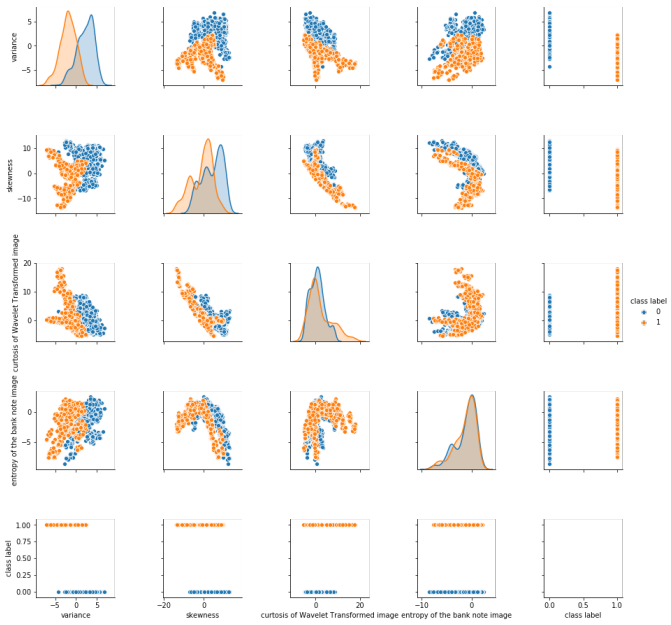

Fig. 6. Pair plot of the entire dataset

By inspection of the two figures [5] and [6] it‘s possible to assume that there‘s two features that are dependent from each other. A test made was removing one of them but unfortunately this produced worst results since the features contain relevant information to the classification. This is doesn‘t prove that the features are independent and that the small class imbalance is affecting the prediction, but the two points made above are still of significance and should be considered.

Although KNN shows the best results, this classifier depends on the given dataset. By observation of the pair plot, Figure [6] there are no significant outliers which decently help the KNN‘s classification. Logistic Regression doesn‘t differ from the previous talked model, according to the McNemar‘s test, and by doing regularization, the chance of performing overfit is less, so this model would be more appropriated to production.