Using Scikit-learn to Predict Bank Telemarketing Outcomes

Tyler Little
University of Maryland,
Balitmore County
Baltimore, Maryland 21227
Email: tyler10@umbc.edu

***Abstract*—This document is a report for my CSMC 471 - Artificial Intelligence final project. Using Scikit-learn, a Python machine learning software library, and the Bank Marketing Data Set provided by the University of California, Irvine, I am tasked with creating a Python script which utilizes different supervised learning algorithms to predict whether a client will subscribe to a term deposit or not, based on a collection of twenty attributes regarding the client. In banking, a term deposit is a monetary investment made to a financial institution that grows over an agreed time span.**

I. The Data Set

The Bank Marketing Data Set is comprised of attributes regarding a Portuguese banking institution's direct marketing (telephone call) campaign. According to the University of California, Irvine, data was collected from May 2008 to November 2010. During this time span around 45,000 samples are taken. There are twenty columns out of twenty-one total in the data set which refer to individual clients being called. Some examples are the age, occupation, and marital status of a client. The twenty-first column in each sample refers to whether they subscribed to a term deposit or not.

II. Preprocessing

First, the data set is imported into the Python script using the pandas module. Before utilizing Scikit-learn's learning algorithms, the data set must be modified accordingly. Each learning algorithm requires a set of samples of length n and a corresponding set of labels. To aggregate the Bank Marketing Data Set, it must be split into to individual sets: one set containing the first twenty columns which describe the client, and another set containing the last column which describes the client's final decision. One final aggregation must be made to the first split set: transforming categorical data columns, such as marital status (single or married) into numerical data. To do this, Scikit-learn has a label encoder available which assigns each unique category with a unique integer value.

III. Processing

At this point, the data set is modified to a format acceptable by Scikit-learn. I can now utilize the module to create various supervised learning models to predict whether a client will subscribe to a term deposit or not, based on a set of attributes regarding the client. The input values and features will be split in half for training and testing each model. The following sections will illustrate analysis of each learning algorithm and the model's results of prediction given the test data.

IV. Gaussian Naive Bayes

Gaussian Naive Bayes is a form of the Naive Bayes classification method. Naive Bayes a probabilistic classifier which assumes that the probability of attributes for a given data set are independent from one another. Gaussian Naive Bayes is a branch of Naive Bayes which assumes the probabilistic distribution of each attribute follows a Gaussian (or normal) distribution. I anticipate there being several attributes in the data set that do not follow this distribution, such as marital status, or the month the client was last contacted. Because these kind of attributes do not necessarily follow a normal distribution, these attributes will hinder the accuracy of the model.

The resulting model using this algorithm was able to accurately predict around seventy-four percent of outcomes given the test input data.

V. Perceptron

Perceptron is another machine learning algorithm which identifies a binary classifier (e.g. subscribing to a term deposit or not) in a vector space using a number of inputs multiplied by their weights. Given the large amount of attributes in the data set, I think the model will have trouble finding an area of complete convergence.

The resulting model using this algorithm was able to accurately predict around eighty-five percent of outcomes given the test input data.

VI. Decision Tree Classifier

The Decision Tree Classifier algorithm in Scikit-learn creates and utilizes a decision tree to predict the outcome of a given input. A decision tree is a tool which assesses the outcome of an attribute's probability in a tree like structure given a set of attributes. Since all outcomes will be assessed,

I have a fair amount of confidence in the performance of this algorithm. However, the documentation of Scikit-learn warns the algorithm tends to overfit the data.

The resulting model using this algorithm was able to collectively predict the outcome of each test sample with one hundred percent accuracy. At first I was hesitant to accept this result, because I initially tested the model using the same training data. However, after splitting the data into training and test sets I am confident the model is not overfitting; it is just really accurate.

VII. Multilayer Perceptron (MLP) Classifier

This algorithm is similar to the Perceptron classifier, except the structure is modified such that the vector inputs and weights follow a sequential Perceptron so to speak; the output given input and weight values of a vector equation become the resulting inputs of another Perceptron layer. This can repeat for any amount of iterations. I believe this method may be worse than the standard Perceptron algorithm, because the MLP specializes in prediction using a non-linear activation function to produce the output, while our data seems to be the opposite.

The resulting model using this algorithm was able to predict seventy-seven to eighty-eight percent of outcomes using the test data, which is either a tad worse or a tad better than the Perceptron learning algorithm.

VIII. Bagging Classifier using K-Nearest Neighbor

K-nearest neighbors is a classification algorithm which groups the closest K resulting attributes and outputs a class the attributes belong to. The Bagging Classifier algorithm is a method provided by Scikit-learn which typically creates base classes for a black-box classification algorithm (i.e. method which assesses outcomes and not the probabilities of outcomes) such as decision trees using random subsets of a specified portion of the original data set. This randomness allows for more variance within decisions made by the algorithm.

The resulting model using this algorithm was able to predict around eighty-seven percent of outcomes using the test data.

IX. Conclusion

This assignment has helped me better understand machine learning using real-world data with a very simple and easy to use software library. In retrospect, there are a number of flaws I would have addressed if given a wider time span. First, regarding utilizing the Gaussian Naive Bayes algorithm, I would have modified the data set to better accommodate the algorithm by removing attributes that did not follow a normal distribution. I believe this modification would ultimately raise the model's accuracy. Secondly, I wished to better explore the Perceptron algorithm by modifying set parameters to find a better area of convergence.

References

[1] [Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014