Hamish Wood – hcw0026@auburn.edu, Nhat Nguyen – nhn0001@auburn.edu,

and Nathan Hunter – nbh0018@auburn.edu

Coordinator: Nathan Hunter

COMP 5600 – Project Final Report

Tuesday, April 30, 2024

Bean Classification Final Report

Problem Description

The problem that our group decided to tackle was to create machine learning models that were able to identify the type of a particular dry bean when given the bean’s shape characteristics. We located a large dataset on Kaggle.com that fit the requirements of our assignments in that it contained enough beans and enough features that described them. We proposed that if we were able to create models that have a high enough accuracy in identifying the beans, the models could potentially be used in the agricultural or shipping industries in applications such as verifying that a shipment of beans is correct or in sorting beans when given an unsorted inventory.

Our requirement as a 5600 group was to have a dataset of audio, visual, textual, or tabular elements that contained more than 5000 training samples, 1000 testing samples, and more than 6 attributes. We fulfilled these requirements as our dry bean dataset contains 13,611 tabular sample entries which we split up into 10889 training samples and 2722 testing samples. Each sample in the dataset has 16 attributes to describe them which are measured up to the 7th decimal place or greater. The models that we were required to implement were two models which use logistic regression and multi-layer perceptron (MLP).

Machine Learning Theory and Implementation

-Logistic Regression

Logistic regression is a supervised learning method of predicting an object’s class based on the object’s features and the model’s logistic regression equation’s coefficients that it established from previous training data. The object’s features are the independent variables which are plugged into the multi-variable regression equation and the predicted class is the output.

To build our logistic regression model, we used the Logistic Regression package from scikit-learn python library and various other packages used to process our dataset. Once we’ve imported our dataset into the program as a standard .csv array using pandas, we first use sklearn dataset splitting function, which will split our primary dataset into randomly selected training and testing datasets. The ratio we chose for this was to have an 80:20 split of entries resulting in a training dataset of 10,889 samples and a testing dataset of 2,722 samples. These datasets are then preprocessed using fit\_transform() and transform() to standardize our features so our model does not become biased towards any feature.

The preprocessed training dataset is next used to train the logistic regression model itself. We use fit() function with our training data to build the model, then we can evaluate the model using our testing data and chosen evaluation metrics and store the results.

-Multi-Layer Perceptron (MLP)

-Multi-Layer Perceptron is a form of feed-forward neural network in which the model passes forward information through a system of perceptron to extract necessary features from the input information. These features can then be used to predict a target class in the final output layer. Generally, there are three types of layers in an MLP mode. The input layer receives raw input data which will often be features of the target classes, then passed on to the hidden layer for processing. The hidden layer applies weights and bias to the input and passes forward to the next layers through an activation function to extract and capture pattern of features from the data. Finally, the output layer represents the target classes and produces the final output.

For data preprocessing, our MLP implementation was also created in the same way our logistic regression model was implemented, with sklearn. We preprocess the data in the same way as our LR model: 80:20 train-test split, data scaling with StandardScaler transform() and fit\_transform(), then feed the preprocessed data into MLPClassifier model provided by sklearn library.

The activation function that we choose for our model is ReLU(x) = max(0, x), a simple activation function that only allows useful information to be retained and propagated toward the next layer. Another activation function that is typically used for multi-class problems are softmax function, which MLPClassifier has already automatically applied to the output layer.

To compile our program into a GUI for a user to interact with, we used tkinter Python GUI library to display a window which has our program’s configuration options. The program uses the file explorer in Windows to allow the user to select the primary dataset that they wish to use for the process. The user is then able to choose which model implementation between Logistic Regression and MLP they want to use when training the model, and the evaluation results from that training are displayed as various metrics which are explained later in the report.

Software Documentation and Instructions for Use

The program was developed using the latest available versions of each library and Python. The Python packages required to run the program include:

* Python 3, the earliest version we tested on was Python 3.8.
* pandas 2.2.2
* scikit-learn 1.4.2

Instructions:

1. Running the code from interface.py will bring up the following GUI window where all options can be configured including importing a dataset and choosing the model implementation to use for training.

A screenshot of a computer program

Description automatically generated

Step 1: The window displayed when first starting the program.

1. The next step is to choose the dataset to use for the model. The dataset must be a .csv format file for the program to import the data properly. The included Dry\_Bean\_Dataset.csv file is the expected file to select for training in this project, but other datasets will work given that they are in the same format. Clicking the “Browse” button will open a Windows file explorer window with which to select the dataset with.

A screenshot of a computer

Description automatically generated

Step 2: Selecting a dataset, navigate to the .csv file and open it using the Windows file explorer window.

1. The file will now display in the main program window and the button to train the model will now be able to be clicked. To choose the model that you want to use for training and testing, use the radio buttons which list the available model implementations. The options for our program are Logistic Regression and Multi-layer Perceptron (MLP).

A screenshot of a computer

Description automatically generated

Step 3: The file imported is now shown in the program window. Selecting the model to train and test is done using the highlighted radio buttons.

1. Once the model implementation has been chosen and dataset imported, click the “Train Model” button to launch the training and testing session. The process will take time to complete depending on how fast the machine it is run on is. Logistic Regression finishes very quickly while MLP will take a few times longer. Once the process has trained and tested the model, the results will be displayed in a separate window listing all of the evaluation metrics and total model accuracy.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Step 4: The end results of training and test displayed in another window. The results of training a Logistic Regression model are shown on the left window, and the results for an MLP model are shown in the right window.

Evaluation Metrics, Results, and Findings

Evaluation Metrics:

As with our progress report, we still grade our models in seven categories. The categories include Accuracy, Precision, Recall, f1-score, Support, Macro Average, and Weighted Average. The definitions of each of these metrics are carried over and are as follows:

* Accuracy – Measures the overall correctness of the model.
* Precision – Measures the ratio of the number of true positive predictions compared to the number of positive predictions made by the model.
* Recall – Measures how many true positives occur out of all the actual positive instances.
* F1-score – The harmonic mean of precision and recall which considers false positives and negatives.
* Support – Number of samples in that class's dataset.
* Macro average – The unweighted mean of evaluation metrics of all classes.
* Weighted average – The weighted mean of evaluation metrics which weighs classes with more support more heavily.

The result window displayed after testing and training the models displays the metrics of Precision, Recall, F1-score, and Support on a per class basis, while Accuracy, Macro Average, and Weighted Average are displayed as an overall model metric including all of the classes in one statistic.

Results:

The results for our evaluation metrics were consistent across multiple runs of each model and are formatted into the tables below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Logistic Regression | | | | | |
| Bean Type | Class Number | Precision | Recall | F1-score | Support |
| Barbunya | 0 | 0.92 | 0.91 | 0.92 | 261 |
| Bombay | 1 | 1.00 | 1.00 | 1.00 | 117 |
| Cali | 2 | 0.95 | 0.94 | 0.94 | 317 |
| Dermason | 3 | 0.92 | 0.90 | 0.91 | 671 |
| Horoz | 4 | 0.97 | 0.96 | 0.97 | 408 |
| Seker | 5 | 0.97 | 0.94 | 0.95 | 413 |
| Sira | 6 | 0.85 | 0.90 | 0.87 | 536 |
| Macro Average | | 0.94 | 0.94 | 0.94 | 2723 |
| Weighted Average | | 0.93 | 0.93 | 0.93 | 2723 |
| Accuracy | | 92.66% |

Table 1: The metrics resulting from the Logistic Regression model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Multi-layer Perceptrons | | | | | |
| Bean Type | Class Number | Precision | Recall | F1-score | Support |
| Barbunya | 0 | 0.93 | 0.92 | 0.93 | 261 |
| Bombay | 1 | 1.00 | 1.00 | 1.00 | 117 |
| Cali | 2 | 0.93 | 0.95 | 0.94 | 317 |
| Dermason | 3 | 0.92 | 0.93 | 0.93 | 671 |
| Horoz | 4 | 0.97 | 0.94 | 0.96 | 408 |
| Seker | 5 | 0.97 | 0.94 | 0.96 | 413 |
| Sira | 6 | 0.88 | 0.89 | 0.89 | 536 |
| Macro Average | | 0.94 | 0.94 | 0.94 | 2723 |
| Weighted Average | | 0.93 | 0.93 | 0.93 | 2723 |
| Accuracy | | 93.21% |

Table 2: The metrics resulting from the MLP model.

Findings:

Our results show that the MLP model is more accurate than the Logistic Regression model by a small margin, with the MLP scoring 93.21% accuracy and the Logistic Regression model scoring 92.66%. The MLP model’s score was higher because even though most of the beans have similar precision, it was able to identify the Sira dry beans at a much higher rate than the regression model. We believe that the regression model had a harder time identifying Sira beans because the shape characteristics of that bean may be very similar to the other types of beans in our dataset.

Other than the Sira bean, the overall results of the two models were very similar, with the models correctly identifying the beans more than 90% of the time. The Bombay bean is an outlier here as the model was always correct in identifying it. This may be because that type of bean has unique shape characteristics and could not be mistaken for another type with sufficient training.

In our project proposal, we said that our goal for the overall accuracy of our models was going to be 95%. Our results are close enough to that initial optimistic goal that we are happy to settle on these implementations of Logistic Regression and Multi-layer Perceptron. The MLP model is closer to our goal, however the time it takes to train the model far exceeds the time of the Logistic Regression model. Both models train and test quickly because of our small dataset, but with a larger dataset that may include millions of unique samples, the MLP model may not scale feasibly. This would cause our preference to shift to the Logistic Regression model as it is only half of a percentage behind the MLP model in accuracy.

Project Group Individual Credits

Nathan Hunter: Implementation of the Logistic Regression model and program user interface.

Nhat Nguyen: Implementation of the Multi-Layer Perceptron model and model’s research.

Hamish Wood: Compilation and writing of project proposal, progress report, and final report.