CS4373 Course Project
Classification Algorithms

TeamNN: Juan Diaz, Hercules Hjalmarsson

04/29/2019

Abstract: We were tasked with analyzing and implementing three different algorithms used in data mining as well as comparing their performance. We choose to research and implement classification algorithms such as Naive Bayes Classifier, Artificial Neural Network and K-Nearest neighbor. We used 10-Fold tests to benchmark the data and compared our algorithms to already available ones through Keras, Tensorflow, and Scikit-Learn.

1. Introduction:

In this project, we analyzed the Pima Indians Diabetes Dataset, Breast Cancer Wisconsin Dataset and the 1984 United States Congressional Voting Records Database Dataset. All of these datasets can be found online. Each dataset contains both features and target data which can be used as training data for our algorithms to measure and compare performance. If an algorithm can be developed to predict outcomes with high probability on unseen data in any of these datasets it will be very valuable for researchers to use in order to predict any outcome with simple inputs. The goal of this project is to compare performance of different algorithms and perhaps even come up with new and improved algorithms. We were able to see many improvements with our neural network implementations on the Pima Indians dataset which performed better than Keras algorithms. We also compared our Naive Bayes and K-Nearest Neighbors implementations with their Scikit Learn equivalents and found that we managed to achieve similar accuracy rates, however performance and time wise Scikit-learn models outperform ours.

2. Description of Data Sets

In this section we will describe the datasets we used for training and testing the algorithms we implemented. We used three different datasets.

1. Pima Indians Diabetes Dataset:

Context: “This dataset is from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.”

Attributes: The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

2. Breast Cancer Wisconsin (Diagnostic) Data Set:

Context: “Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.” The goal is to predict whether it can be classified as malignant or benign.

Attributes:ID number, Diagnosis (M = malignant, B = benign)
3-32) Ten real-valued features are computed for each cell nucleus: radius, texture, perimeter, area, smoothness, compactness, concavity, etc.

3. 1984 United States Congressional Voting Records Database:

Context: “This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).” The goal is to predict whether a congressman is Republican or Democrat based their votes.

Attributes: Attribute Information: Class Name: 2 (democrat, republican), handicapped-infants, water-project-cost-sharing, adoption-of-the-budget-resolution, physician-fee-freeze, el-salvador-aid, religious-groups-in-schools, anti-satellite-test-ban, aid-to-nicaraguan-contras, mx-missile, immigration, synfuels-corporation-cutback, education-spending, superfund-right-to-sue, crime, duty-free-exports, export-administration-act-south-africa.

3. Classification Algorithms

3.1 Naive Bayes Classifier

3.1.1 Implementation:

The following is a brief overview of the steps taken and functions used to implement Naive Bayes.

1. Separate data based on class:

Whenever the fit function is called with X_train and y_train datasets, the first step taken is to create a dictionary of the data wherein the objects are separated based on their outcome class label. The dictionary uses outcome class labels(the class we are trying to predict) as keys and each key maps to all rows which have that class as their label.
Note: See separateBasedOnOutcomeClass() function for code for this.

2. Calculate attribute statistics:

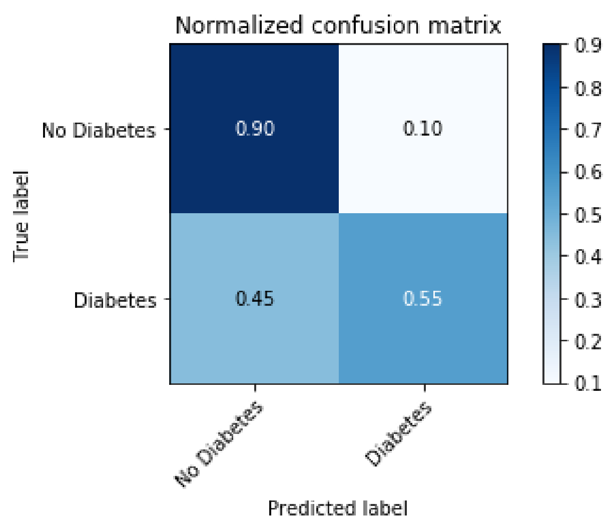
We now take the dictionary of rows separated based on class labels and calculate the mean and standard deviation of all attributes respective to their outcome class and place them into a dictionary of class summaries. So assuming we have to outcome classes (0 and 1) and 1 attribute, we would find the mean and standard deviation of this attribute for rows that map to 0 and also for rows that map to 1.
Note: See summarize() function for code.

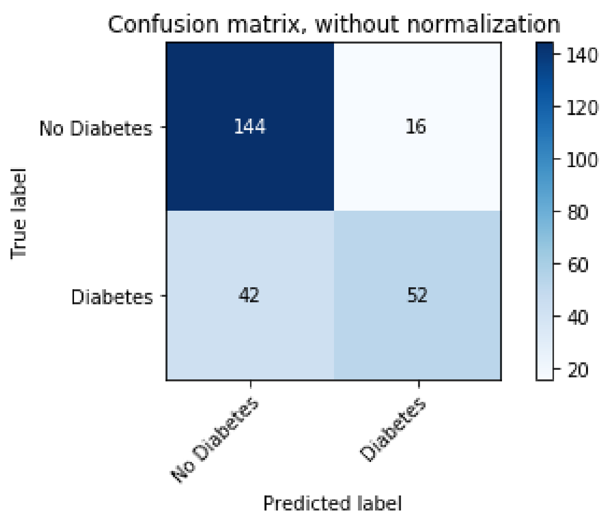
3. Predict label based on Gaussian Probability Distribution:

Given an input row which we are trying to classify, we find the probability that the attributes for this row belong to class 0 and the probability that it belongs to class 1. This is done by calling the scikit norm pdf function given the mean and standard deviation for this particular attribute and outcome class found previously. The algorithm assigns this row the label with the highest probability.

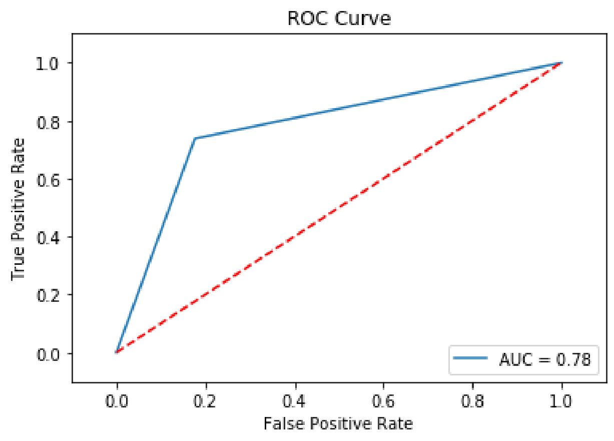
3.1.2 Testing:

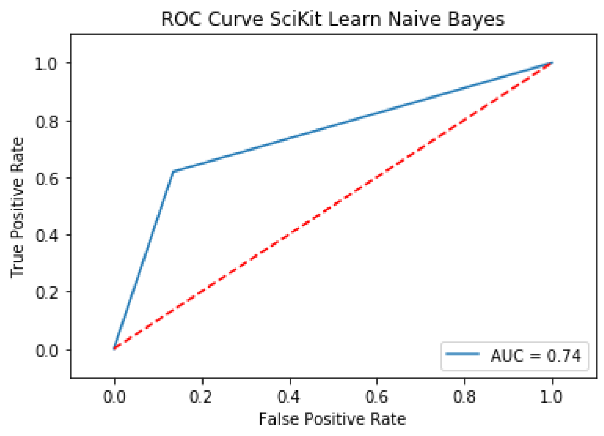
We conducted a 10-Fold test using the Pima Indians database and obtained the following scores:
[0.70, 0.79, 0.71, 0.66, 0.74, 0.75, 0.75, 0.81, 0.75, 0.76]
Mean Score: 76%
Time: 0.93 Seconds
We also computed a confusion matrix in order to illustrate the overall accuracy and sensitivity of our model.





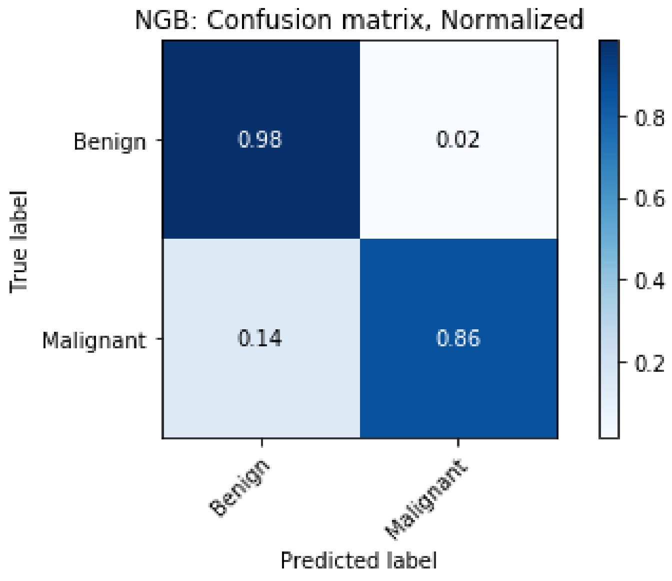
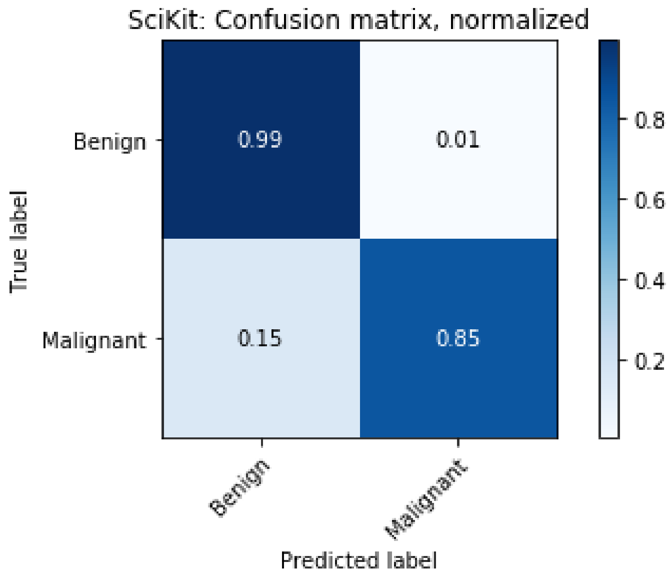
We decided to compare this with Scikit Learn Naive Bayes model in order to gage how accurate our own implementation was.
SciKit Learn 10-fold scores: [0.68, 0.81, 0.75, 0.71, 0.73, 0.77, 0.81 0.82, 0.74, 0.75]
Mean Score: 0.76%
Time: 0.0015 seconds
Since the scores are similar to the scores we got for our implementation we decided to perform a pairwise T-Test between them in order to determine if the two models are statistically similar. The test was inconclusive as the P-value was greater than our 0.05 planned confidence level. Following this we decided to plot the ROC curves for each model (ours vs. SciKit Learn) to see if there would be a difference between the two.





Cancer Diagnosis Dataset

We also ran K-fold tests on the cancer diagnosis data set and compared the two models and got the following results:

Our Implementation	SKLearn Naive Bayes
10-Fold Test Mean Accuracy: 93% Time: 2.53 Seconds 	10-Fold Test Mean Accuracy: 94% Time: 0.002 seconds 

Clearly the two models have similar accuracy rates, however Scikit-Learn outperforms our model by significant amount.

3.2 K-Nearest Neighbors Classifier

For our implementation of the K-Nearest Neighbor classifier we first decided to implement a linear neighbor search and later improved it using SCIPY's KD-Tree as a way to improve our algorithm's performance.

3.2.1 Implementation:

Fitting Data: First, whenever the fit function is called the X and Y values are stored, initially this part required no computation as we did not plan to use a KD-Tree to store the X values. This was improved when we added the KD-Tree implementation as a class object and it is initialized with the X_train values(objects) whenever the fit function is called. The K value is set automatically to the square root of the number of rows in the data set as we found various suggestions to do so in online resources and we also found the highest accuracy levels when K was set accordingly.

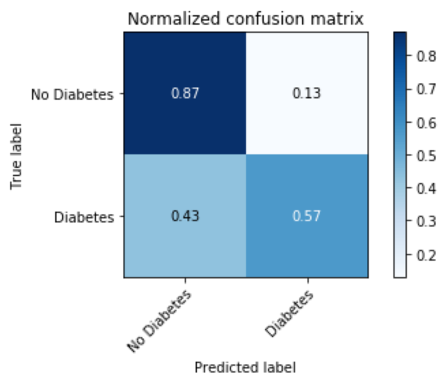
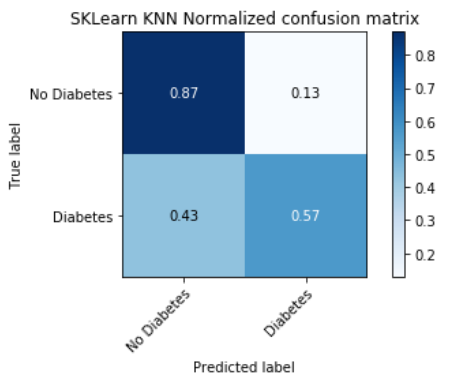
Predicting Labels: When the predict function is called, the algorithm will loop through the X_test vectors and either query the KD-tree or the linear search function we implemented (explained below) to get the nearest K neighbors to the vectors we are trying to predict. Following this the predictLabel function is called and passed an array of the indices of the nearest neighbors. The predictLabel function will now count the occurances of the difference outcome classes (stored in a dict) among the group of neighbors and return the class with highest number of neighbors which have it as their outcome class.

Linear Search: The linear neighbor search function we implemented before using the KD-Tree works as follows: Given a unseen vector, it will loop through the training set and using Euclidean distance, store and return the K neighbors which are closer to the unseen vector we are trying to predict.

We ran 10-Fold to tests to compare the average time elapsed between the KD-Tree query and linear neighbor search and the difference was 0.24 vs. 16.12 seconds respectively and therefore decided to keep the KD-tree as the default implementation method in our model, but left the linear search function as well as an illustration of how it is implemented.

3.2.1 Testing: Comparing our implementation to Scikit Learn's KNN

Firstly we ran 10-Fold tests on both models using the pima indians dataset and got the following results:

	Our KNN model	Scikit Learn KNN
10-fold tests mean accuracy	74.7%	74.4%
10-fold tests mean time	0.24 seconds	0.002 seconds
Confusion Matrix		

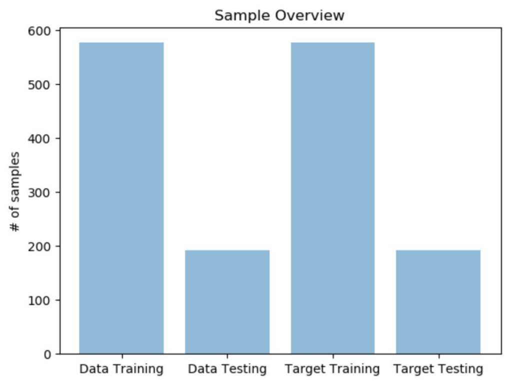
Clearly, here the two models also have similar accuracy rate and again the only difference is in time performance.

3.3 Neural Network

3.3.1 Implementation:

The following is a brief overview of the functions used to implement a Neural Network.

1. Separate and gut-check data



After separating the data in to the following categories the training data was used to train on the target data. Generally this was done through sklearn train_test_split function.

1. Create the model

I decided to implement the model as a class due to it being easier to connect to UI's as well as extending the functionality to for example sklearn baseEstimator class. After setting up the class and the constructor I implemented the fit function along with the training loop and the weights and biases.

The most difficult of this was to make sure the input layer shape of the matrix and output layer shape are correct since there will be hidden layers between the two. Using the Numpy shape functionality of the 2d matrices made this a lot easier.

Once the loop was done I implemented the forward and back propagation functions. These functions are the main tools to train a neural network as one will feed data to the other back and forward. I setup the necessary math functions and separated the training of the derived weights to a separate function.

Once this was done I implemented the predict function which uses the forward propagation function and evaluate function which uses the predict function to output accuracy on both training and testing data.

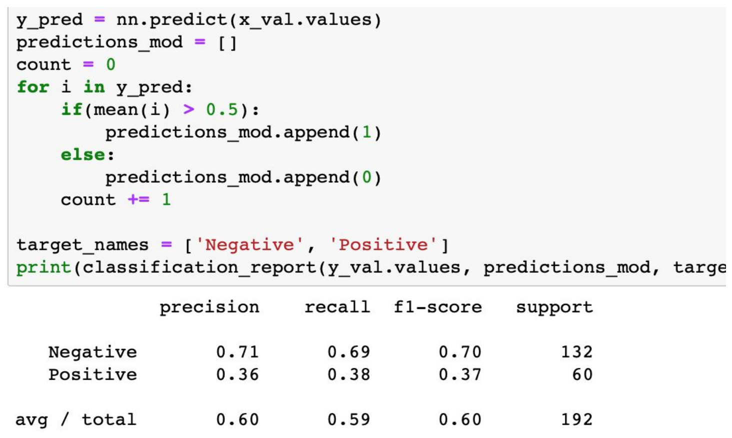
The biggest difficulty with the predict function was that it will only work without altering the input data if the weights and biases can be multiplied and added with both the training and testing data. However if the testing data is larger or smaller than the training data it cannot perform the predictions. My initial solution failed as I was taking the GCD of the testing to accommodate the size issue, this resulted in data corruption. I solves this by either increasing or decreasing the testing data by multiplying it with itself and then subtracting rows depending on the weight size which is equal to the training data size. I then reduce the test size again to its original input size before outputting it to compare with the target test data for accuracy scoring so there is no difference in the number of rows.

1. Evaluation

I created another Keras model similar to my neural network and compared the performance of the two with multiple activation functions.

3.3.2 Testing:
My model was faster when training in comparison to the Keras model on the Pima Indian Dataset, the accuracy however was slightly worse. The overall accuracy of the Neural Networks were worse than the 10-Fold test. I believe implementing more complex Neural Networks or increasing the size of the data input would have made my Neural Network and the Keras Neural Network perform up to par with other algorithms.

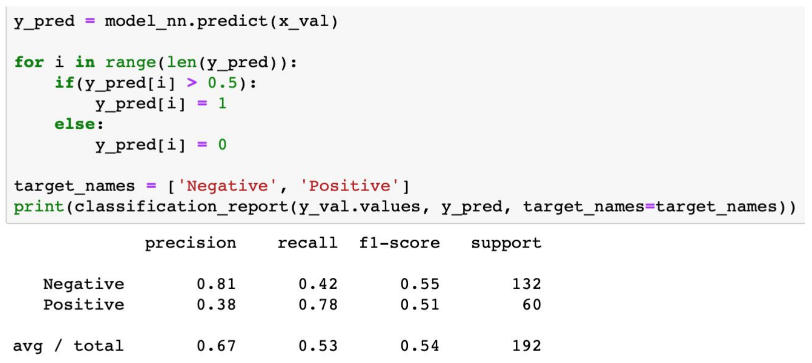
My Neural Network:



```
y_pred = nn.predict(x_val.values)
predictions_mod = []
count = 0
for i in y_pred:
    if(mean(i) > 0.5):
        predictions_mod.append(1)
    else:
        predictions_mod.append(0)
    count += 1

target_names = ['Negative', 'Positive']
print(classification_report(y_val.values, predictions_mod, targe
```

```
             precision    recall  f1-score   support

   Negative       0.71      0.69      0.70       132
   Positive       0.36      0.38      0.37        60

avg / total       0.60      0.59      0.60       192
```

Keras Neural Network:



```
y_pred = model_nn.predict(x_val)

for i in range(len(y_pred)):
    if(y_pred[i] > 0.5):
        y_pred[i] = 1
    else:
        y_pred[i] = 0

target_names = ['Negative', 'Positive']
print(classification_report(y_val.values, y_pred, target_names=target_names))
```

```
             precision    recall  f1-score   support

   Negative       0.81      0.42      0.55       132
   Positive       0.38      0.78      0.51        60

avg / total       0.67      0.53      0.54       192
```

Note: JupyterNotebook “Semester Project” contains all model implementations, data preprocessing, and k-fold tests performed.

<u>Summary:</u>

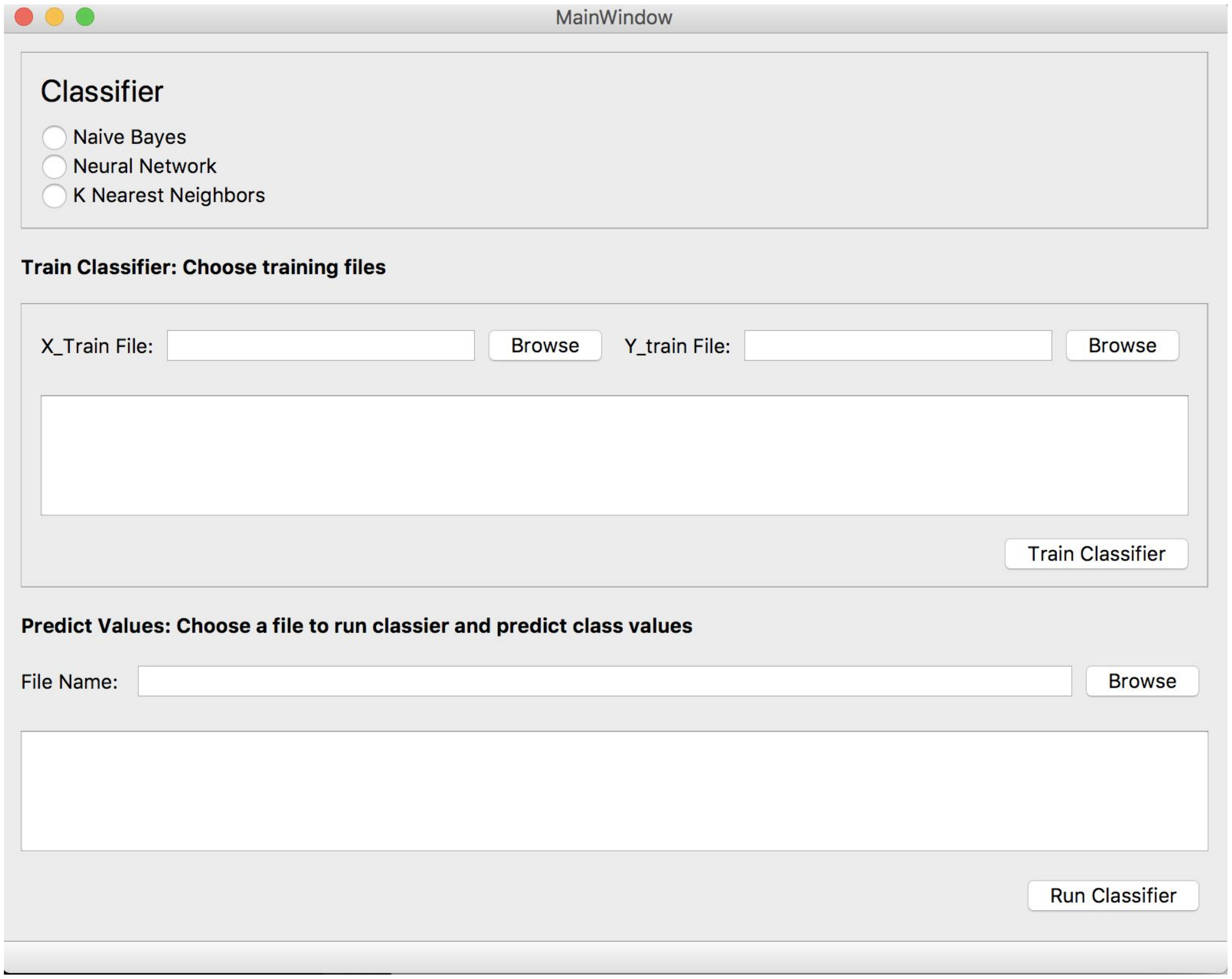
Given the opportunity to choose and work on our own areas of interest has been an invaluable experience. Since we were only two persons doing a project meant for three individuals we had to take on a heavier workload than normally however, our experience in this project include researching and reading about data mining algorithms, pre-processing data and evaluating the data. The most difficult part was to balance research and implementation of the project along with other course work. We did learn a lot by utilizing the knowledge gained throughout the data mining course and our homework to effectively get results. If the course project was given in the beginning of class I think we could’ve most likely achieved most if not all of the extensions to the project.

Possible extensions to this project:

1. Include more test cases and variations of test data
2. Pre-process input data in different ways and measure the impact of it when evaluating the results
3. Expand on the Neural network to include RNN, CNN and other types.
4. Fine-tune current algorithms to improve overall performance

We worked on the project proposal together and created this report as well as the presentation together as a team.

UI User Manual



Train Classifier: Choose training files

X_Train File:

Y_train File:

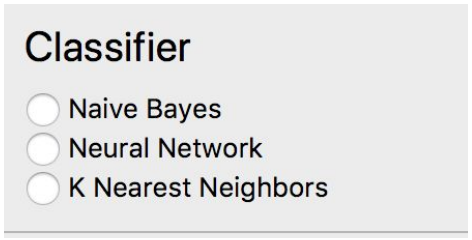
Predict Values: Choose a file to run classier and predict class values

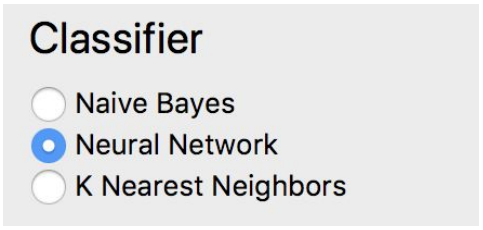
File Name:

Installation: To run UI, run Exec file "UI Classification Algorithms"

Tutorial: Note: Data sets for demo are in folder UIDatasets

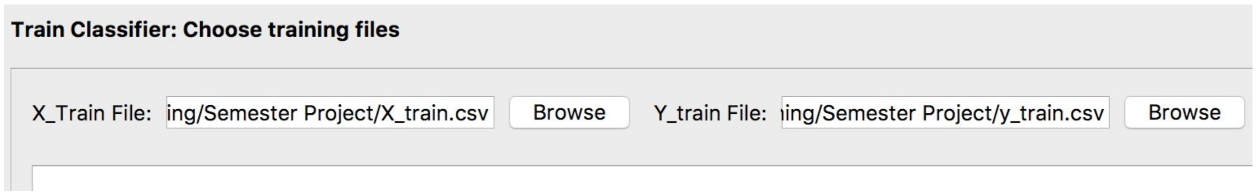
Step 1: Select an Algorithm:





Classifier: Naive Bayes [✓ Neural Network] K Nearest Neighbors

Step 2: Load X_train and Y_train files to fit model: (see folder UI Datasets to find them)

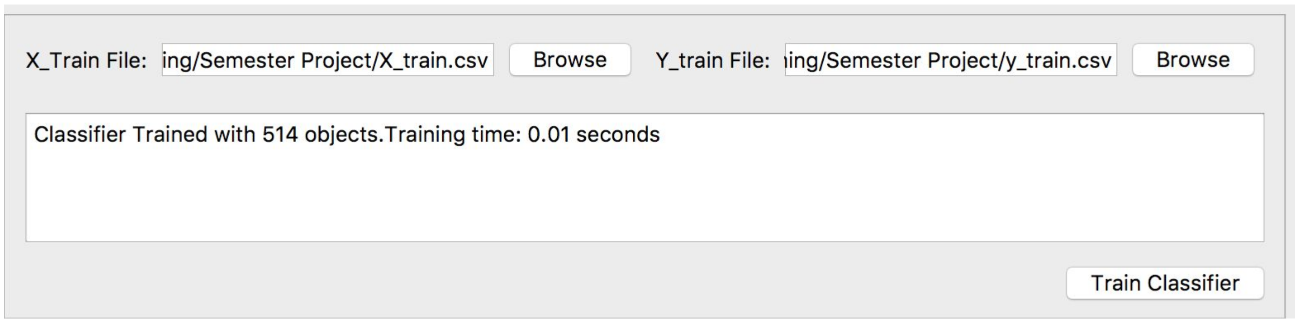


Train Classifier: Choose training files

X_Train File: ing/Semester Project/X_train.csv

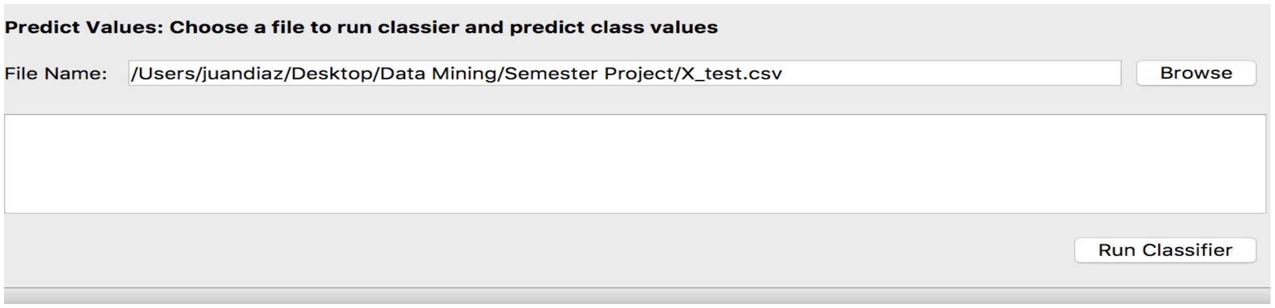
Y_train File: ing/Semester Project/y_train.csv

Step 3: Press "Train Classifier" Button after loading data

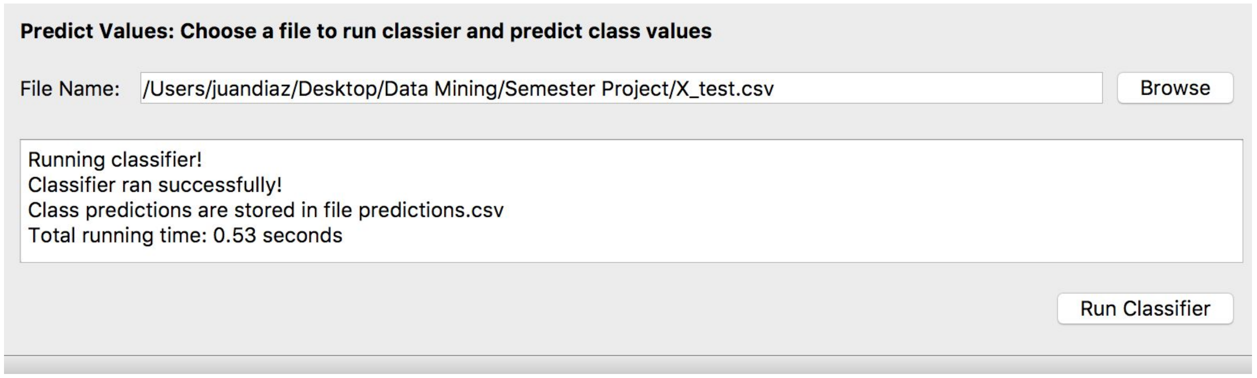

X_Train File:
ing/Semester Project/X_train.csv
Browse
Y_train File:
ıing/Semester Project/y_train.csv
Browse
Train Classifier

```
Classifier Trained with 514 objects.Training time: 0.01 seconds
```

Step 4: Load X_Test Data (**see folder UIDatasets to find them**) **and Press "Run Classifier" Button after doing so**


Predict Values: Choose a file to run classier and predict class values
File Name:
/Users/juandiaz/Desktop/Data Mining/Semester Project/X_test.csv
Browse
Run Classifier

Result: Predictions will be stored in file predictions.csv and displayed on UI


Predict Values: Choose a file to run classier and predict class values
File Name:
/Users/juandiaz/Desktop/Data Mining/Semester Project/X_test.csv
Browse
Run Classifier

```
Running classifier!
Classifier ran successfully!
Class predictions are stored in file predictions.csv
Total running time: 0.53 seconds
```

Note y_test file was also included in UIDatasets to compute accuracy if wished.

Team Work Breakdown:

KNN, Naive Bayes, UI, Presentation, Reports - Juan

Neural Network, Presentation, Reports - Hercules

Dataset Sources:
https://www.kaggle.com/uciml/pima-indians-diabetes-database
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records