Using Neural Networks to target Customers for BRCC

Group 2: Shirly Wang(sw983), Yiran Wang(yw2236), Rosy Zhang(yz2548), Emily Yang(ty366)

NBA 4920 Final Project

December 8th. 2019

Introduction

The main goal of this project is to target customers who will make the purchase for Big Red Clothing Company (BRCC) in the future. Based on the related topics covered in class and the experiments we conducted, we finally decided to use an ensemble method with neural networks to predict the targeted customers. The data was provided by BRCC.

We first manipulated the training data by setting the response and independent variables. Then we split the data to 80% training data and 20% validation data followed by normalizing the data. After that, we are ready to train the model. We built multilayer perceptron neural network models and experimented them with different activation functions, different layers and the voting method, in order to achieve better performance and higher accuracy. And the main metric we chose to evaluate the performance of the model is Mean F1-score. The Mean F1-score is calculated by the statistics precision p and recall r as below:

$$F1 = 2\frac{p \cdot r}{p + r} \text{ where } p = \frac{tp}{tp + fp}, \quad r = \frac{tp}{tp + fn}$$

In the Bonus questions, we further compared the neural network model performance with the logistic regression and classification trees models using the validation data.

Method used in the submission: (6 points)

I. What activation functions you used, how it affected your Kaggle score, and why do you think a specific activation function worked?

In the project, we tried multilayer perceptron neural network models with all 4 available activation functions in the sklearn package. The activation function decides if given node should be "activated" or not based on the weighted sum. We first used ReLu, logistic, identity, and Tanh activation methods separately to train the model. In order to keep other variables constant to see how different activation functions affect the result, we chose 3 layers with 500 neurons in each layer for all 4 models below and used the validation data to evaluate the model. For the ReLu activation method, the Mean F1-score of this model is 0.617.

```
print(classification_report(y_test,pred1))
```

	precision	recall	f1-score	support
0	0.69	0.73	0.71	1269
1	0.47	0.42	0.44	731
accuracy			0.62	2000
macro avg	0.58	0.58	0.58	2000
weighted avg	0.61	0.62	0.61	2000

```
print(accuracy_score(y_test,pred1))
```

```
0.617
```

For logistic activation method, we obtained an accuracy score of 0.687. Please see the following matrix for more specific details.

```
print(classification_report(y_test,pred2))
```

	precision	recall	f1-score	support
0	0.71	0.88	0.78	1301
1	0.59	0.33	0.43	699
accuracy			0.69	2000
macro avg	0.65	0.61	0.61	2000
weighted avg	0.67	0.69	0.66	2000

```
print(accuracy_score(y_test,pred2))
```

```
0.687
```

Following the same setting with identity activation function, the Mean F1-score for the model is 0.689, which is very similar to the logistic activation method.

```
print(classification_report(y_test,pred3))
```

	precision	recall	f1-score	support
0	0.72	0.86	0.78	1301
1	0.59	0.37	0.46	699
accuracy			0.69	2000
macro avg	0.65	0.62	0.62	2000
weighted avg	0.67	0.69	0.67	2000

```
print(accuracy_score(y_test,pred3))
```

```
0.689
```

We used the Tanh activation method for the last multilayer perceptron model. Using 3 layers with 500 neurons in each layer, the accuracy score is 0.698.

```
print(classification_report(y_test,pred4))
```

	precision	recall	f1-score	support
0	0.73	0.86	0.79	1301
1	0.60	0.40	0.48	699
accuracy			0.70	2000
macro avg	0.66	0.63	0.63	2000
weighted avg	0.68	0.70	0.68	2000

```
print(accuracy_score(y_test,pred4))
```

```
0.698
```

Obviously, the multilayer perceptron with ReLu activation function has the lowest Mean F1-score compared to the other three activation functions. Although models with ReLu activation methods may be computationally less expensive and be used as default method in many settings, it can only be used within hidden layers of the model and the gradients might go toward zero during training. In other words, ReLu could result in dead neurons in some situations. In contrast, the logistics activation function can be a good fit for classification problem as it has a very steep curve

in Y values in the range between -2 to 2 in X values. Thus, any changes in X in that region will result in significant changes in Y. In other words, logistic function tends to bring Y values to the end of the curve, which, as a result, is good for classification problems. Similarly, Tanh activation function shares the same advantages as logistic function since it's actually a simple mathematical transformation of logistic functions. The identity activation function is a linear function in essence. Therefore, we choose the three activation functions models with higher accuracies—logistic, Tanh, and identity, as sub-models to be trained in our voting classifier. In our experiment, adding sub-model with ReLu activation function did decrease the model accuracy by around 4%.

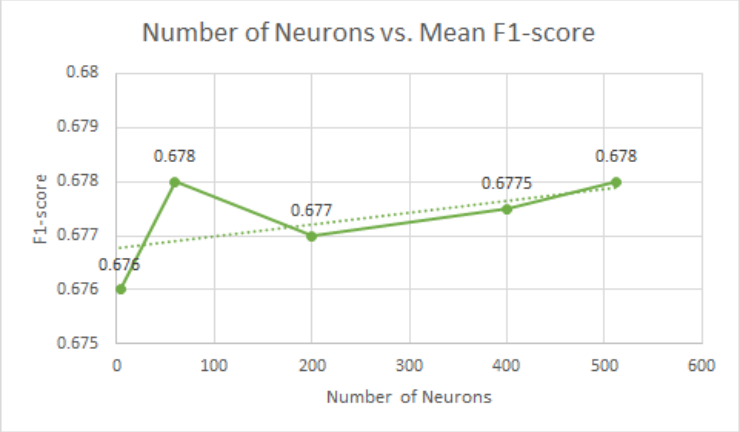
II. How many layers in the neural network you used, how it affected your Kaggle score, and why do you think a specific number of layers worked?

In the project, we used 3 layers in the neural network with 500 neurons. To determine the best number of layers for the final model, we tried 4 layers with 2 hidden layers and 5 layers with 3 hidden layers. We kept the number of epochs at 50, which is the max number allowed for this project, and kept the max iteration at 200. When testing the different Mean F1-scores obtained using different layers, we set each layer with 500 neurons. We used validation data to test the model as well.

By applying different layers to the multilayer perceptron model with ReLu activation method, we found that the Mean F1-score did not change much among 3, 4, or 5 layers. Accordingly, our Kaggle score did not improve with an increasing number of layers. Thus, we used 3 layers, the minimum number of layers for neural network models for simplicity in all of our multilayer perceptron sub-models in our final voting classifier. A neural networks have at least one input layer, one hidden layer, and one output layer. Please see the table below for our Mean F1-scores with different layers of multilayer perceptron models.

Column1	Column2	Column3	Column4
# of Layers	3	4	5
F1-score	0.626	0.633	0.62

We also did experiments to test the best number of neurons in each layer. In multilayer perceptron model with Tanh activation method, we tried 60, 100, 200, 400, and 512 numbers of neurons in 3 layers respectively. There is only around 0.01 difference in the Mean F1-score with validation data within 60, 100, 200, and 400 neurons. The model with 512 neurons in each layer slightly increased Mean F1-score in validation data. Theoretically, the Mean F1-score should increase as the number of neurons increases until it reaches a maximum and becomes stable. In our experiment, we did not see an obvious trend of increase in Mean F1-score as we increased the number of neurons. We suspected that the optimal number of neurons has not been reached yet. Therefore, we chose 512 neurons in our model for each layer.



III. Did you use ensemble methods, how it affected your Kaggle score, and why do you think ensemble worked?

We used voting method to combine three MLP models and a random forest model when training the model. Random forest uses bagging as the ensemble method and decision tree as the

individual model. We used gini-index as our purity measurement, and the minimum leaf in each tree and minimum split are both set to 13. By applying our training data, we obtained 0.6785 Mean F1-score for random forest model in the validation data. We also tried AdaBoost, which is an ensemble method using boosting. It works well especially with decision trees, and can learn from previous mistakes. We experimented to add adaboost on our random forest model. Unexpectedly, both the validation Mean F1-score and the final Kaggle score decreased from the voting model combined with AdaBoosted random forest model.

In our complete model, we used a voting method including 3 multilayer perceptron models with different activation functions and a random forest classification model as sub-models. It increased the accuracy rate with our validation data, so it produced the same increase as using the test data. The Kaggle score was increased using the ensemble voting method compared to a single neural network method.

In general, Ensemble learning makes predictions based on several sub-models. Ensemble methods are able to combine multiple machine learning techniques into one model. It thus decreases variance, bias, or improves predictions. Also, based on our data, the voting method did produce higher accuracy rate than single neural network model.

Bonus Question: (2 points)

I. Neural Network vs. Logistic Regression

After we performed Neural Network algorithms with different activation functions, we also applied Logistic Regression model to the validation data set. In fact, the accuracy doesn't have a huge difference compared with each other. However, the difference between the accuracy of Neural Network with logistic as activation function and the single logistic regression is worth noticing. Here's a straightforward comparison table about the models' accuracy:

NN: ReLu	NN: Logistic	NN: tanh	Logistic Regression
0.673	**0.669**	0.675	**0.673**

On one hand, the classic application of logistic regression model is binary classification, which is commonly used in real life. One of the great properties of logistic regression is that the logistic cost function is convex, which guarantees us to find the global cost minimum and brings a relatively nice accuracy result in this case study. On the other hand, neural networks are somewhat related to logistic regression if we think of logistic regression as a one layer neural network. The logistic sigmoid function is often used as activation function in the hidden layer of a neural network. However, once we stack logistic activation functions in a multi-layer neural network, we lose the convexity advantage of logistic regression because the cost function in a multi-layer perceptron results in multiple local minima and damages the optimization algorithm. That's the reason why Logistic Regression by itself performs slightly better than Neural Network with logistic as activation function. Obviously, Logistic Regression is not the best model in this case study. The Neural Network using hyperbolic tangent as activation function performs better,

and it indeed works better in practice since it's not limited to only positive outputs in the hidden layers.

II. Neural Network vs. Classification Trees

Neural networks are often compared to classification trees because both methods can model data that has non-linear relationships between variables and both of the models can handle interactions between variables. Neural networks of course are powerful models, especially from the perspective of image processing and character recognition, but they usually do not outperform classification trees model because they have a number of drawbacks compared to classification trees. First, it can't handle categorical variables with multiple classes such as living states. Classifying a result into multiple categories can be completed by setting arbitrary value thresholds for discriminating one feature from another. However, classification trees can easily solve this problem. Second, neural network algorithms have a lot of hyper-parameters (number of layers, neurons per layer, activation functions, optimizers, regularizers, etc.) to be tuned in order to figure out the best configuration. What's more, neural networks are prone to overfitting. On the other hand, decision trees model is very intuitive, easy to interpret and the number of hyper-parameters to be tuned is almost null so it requires less effort in data pre-processing.

However, the accuracy of the Classification Tree in this case is 0.666, which slightly underperforms the Neural network models. We think there are two possible reasons. First, the data set is not large enough for us to fully train the models and separate their differences. Second, there's a popular theorem in the machine learning world: no free lunch theorem, so there are no absolutes. Based on the results of this case study, we prefer neural networks to classification trees.