Daniel Choy
Spring 2023
CSE 144

Assignment 1 Logistic Regression & Forward NN (Report)

Validation Accuracy Results:

Logistic Regression Model Validation Accuracy:

```
[81] ###### Your codes start here.######
     # Generating predictions on the validation dataset
     Y_val_pred = LR_model.predict(X_val)

     # Assessing the model's predictive performance
     correct_predictions = (Y_val_pred == Y_val).sum()
     total_samples = len(Y_val_pred)
     accuracy = correct_predictions / total_samples
     print("Logistic Regression Model Validation Accuracy: ", accuracy)
     ###### Your codes end here.######

     Logistic Regression Model Validation Accuracy:  0.8682352941176471
```

Neural Network Model Validation Accuracy when H=6:

```
[ ] with torch.no_grad():
      y_val_pred = model(X_val)
      _, predicted_val = torch.max(y_val_pred, 1)
      val_acc = (Y_val == predicted_val).sum().float() / len(Y_val)

      # Print validation accuracy
      print(f"Neural network model validation accuracy: {val_acc}")

    Neural network model validation accuracy: 0.8717647194862366
```

Neural Network Model Validation Accuracy when H=40:

```
[88] with torch.no_grad():
       y_val_pred = model(X_val)
       _, predicted_val = torch.max(y_val_pred, 1)
       val_acc = (Y_val == predicted_val).sum().float() / len(Y_val)

       # Print validation accuracy
       print(f"Neural network model validation accuracy: {val_acc}")

     Neural network model validation accuracy: 0.8623529672622681
```

What I learned from this Assignment 1?

Preprocessing the Data is Important

Before using the data to train a machine learning model, we have to preprocess it in order for the model to be able to understand it and use it. In this excercise the preprocessing of the data involved shuffling the training data, selecting the features for training and testing, checking if their are missing data in the datasets, replacing the churn labels of either no or yes to 0 or 1, splitting X and Y into training set and validation set, and converting the categorical features in the dataframe to one-hot encoding.

We shuffled the data in order to ensure that the data samples are given to the model in a random order during training which helps the model to learn a more robust representation of the data and generalize better to new/unseen data. The reason we check if their are missing data in the training and test dataset is because we want to handle the situation by filling up the value so that the model doesn't get confused when it sees missing data. In our case, we didn't have any missing data. The reason we replaces the churn labels of either no or yes to 0 or 1 is because it makes it easier for the machine learning algorithm to work with these numerical labels mathematically. The reason we split X and Y into training set and validation set is because we need the two sets for different reasons. The training set is used to train the model's parameters in order to learn the relationships between the input data and output labels. The validation set is used to evaluate the performance of the model and adjust the parameters accordingly in order to prevent overfitting. The reason we converted the categorical features in the dataframe to one-hot encoding is because it allows the machine learning algorithm to properly handle categorical data and prevent it from assigning any numerical importance or order to the categories. One-hot encoding instead represents each category as a separate binary feature with no inherent numerical meaning.

Evaluating the Accuracy of the Model is Important

Evaluating the accuracy of the model on validation set is important because it ensures that it's working as expected. If the accuracy is low, there are multiple possible reasons for this. One being that the model has underfitted which is a result of having a training set to small or not representative of the true distribution of the data. Another reason is that the model has overfitted which is a result of the model capturing the unnecessary noise of the training set or training for too many epochs.

Hyperparameter Tuning is Important

Hyperparamter tuning is important in machine learning as they govern how a machine algorithm works and how well they do. Some of the hyperparanters used in this assignment for our neural network are the learning rate, the number of hidden units, and the number of epochs. The learning rate is important to adjust because a too small of a learning rate can lead to gradient descent to be very slow and a too large learning rate can cause gradient descent to overshoot the minimum or even fail to converge (might even diverge). The number of hidden units is important to adjust because this can help with the performance of the model. If the number of hidden units is too high as we saw in our neural network when we changed H from 6 to 40, then

the neural network can become too complex and overfit the training data. The number of epochs is important to adjust as well as too many epochs can also lead to overfitting.

Choosing the Right Algorithm Matters

Choosing the right algorithm matters because it determines how accurate your model can be. The model you choose is determined by many factors including size of the dataset and the complexity of the task you are trying to accomplish. In our case we used a logistic regression model and a neural network. Both models came relatively close in accuracy when compared to each other.