LDA and MLP Classifiers for Predicting Customer Airline Satisfaction

David Antaki, Maulik Patel
Professor Mark Zolotas
EECE5644 Final Project
6/28/22
Github Repository; Airline Satisfaction Dataset

1 Introduction

For this project, the question that was sought to be answered was "what factors lead to customer satisfaction for an airline?" and can a customer's satisfaction be predicted. To accomplish this, a dataset containing an airline satisfaction survey with the 25 features listed below was used.

Gender: Gender of the passengers (Female, Male)

Customer Type: The customer type (Loyal customer, disloyal customer)

Age: The actual age of the passengers

Type of Travel: Purpose of the flight of the passengers (Personal Travel, Business Travel)

Class: Travel class in the plane of the passengers (Business, Eco, Eco Plus)

Flight distance: The flight distance of this journey

Inflight wifi service: Satisfaction level of the inflight wifi service (0:Not Applicable;1-5)

Departure/Arrival time convenient: Satisfaction level of Departure/Arrival time convenient

Ease of Online booking: Satisfaction level of online booking

Gate location: Satisfaction level of Gate location

Food and drink: Satisfaction level of Food and drink

Online boarding: Satisfaction level of online boarding

Seat comfort: Satisfaction level of Seat comfort

Inflight entertainment: Satisfaction level of inflight entertainment

On-board service: Satisfaction level of On-board service

Leg room service: Satisfaction level of Leg room service

Baggage handling: Satisfaction level of baggage handling

Check-in service: Satisfaction level of Check-in service

Inflight service: Satisfaction level of inflight service

Cleanliness: Satisfaction level of Cleanliness

Departure Delay in Minutes: Minutes delayed when departure

Arrival Delay in Minutes: Minutes delayed when Arrival

Satisfaction: Airline satisfaction level(Satisfaction, neutral or dissatisfaction)

Figure 1 – List of all the features in the dataset

The training dataset included 103,904 samples, while the testing dataset was about 25% of that at 25,976 samples. According to the dataset and the question attempting to be answered, the problem is a supervised binary classification problem as there are 2 outcomes: a customer is either "satisfied" or they are "neutral or dissatisfied". The initial plan was

to train an ML model to predict whether a passenger would be satisfied based on the features in the dataset. Prior to beginning work, there was a chance that prediction might not be possible. An MLP classifier was developed along with an LDA classifier to determine if runtime could be improved for a negligible difference in accuracy.

2 Data Visualization

To better understand the dataset that was being worked on, data science metrics and plots were made to better visualize the data and confirm if the use of "Permutation Importance" and correlation heat map made sense. The figure below shows the distribution of samples labeled "neutral or dissatisfied" vs. "satisfied". As evident, around 56.7% is "neutral or dissatisfied", and 43.3% is "satisfied".

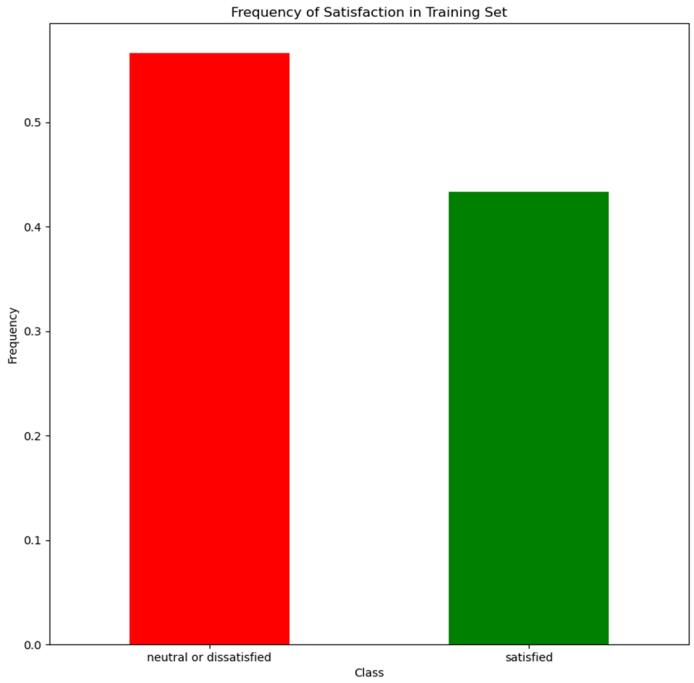

Figure 2-Frequency of Satisfaction in Training Set

It should be noted that the label of "satisfied" is equivalent to 1, and the label of "neutral or dissatisfied" is labeled as 0. From the plot below it can be seen that both genders are equivalently similar in the distribution of satisfaction, so the "gender" feature may not prove to be a useful feature in classification.

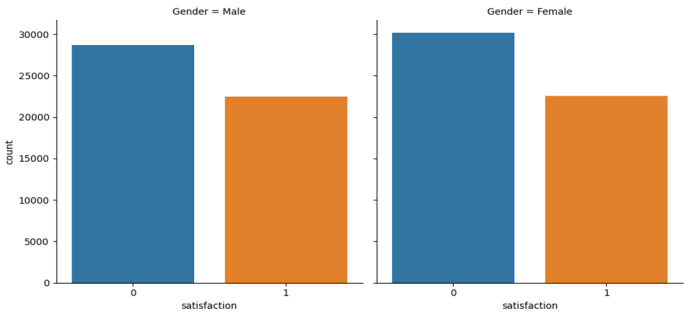

Figure 3 – Classification splits based on gender.

As for customer type, and specifically "loyal customers", an equal number of loyal customers are satisfied, and neutral or dissatisfied.

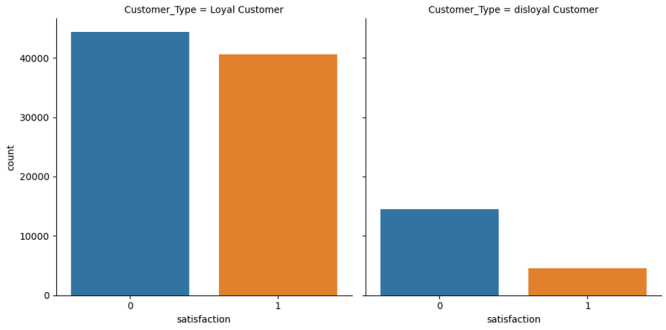

Figure 4 – Classification splits based on customer type.

Lastly, it is observed that between ages 7 and 40 and between ages 62 to 85, customers are generally neutral/dissatisfied. The remaining are generally satisfied as shown below.

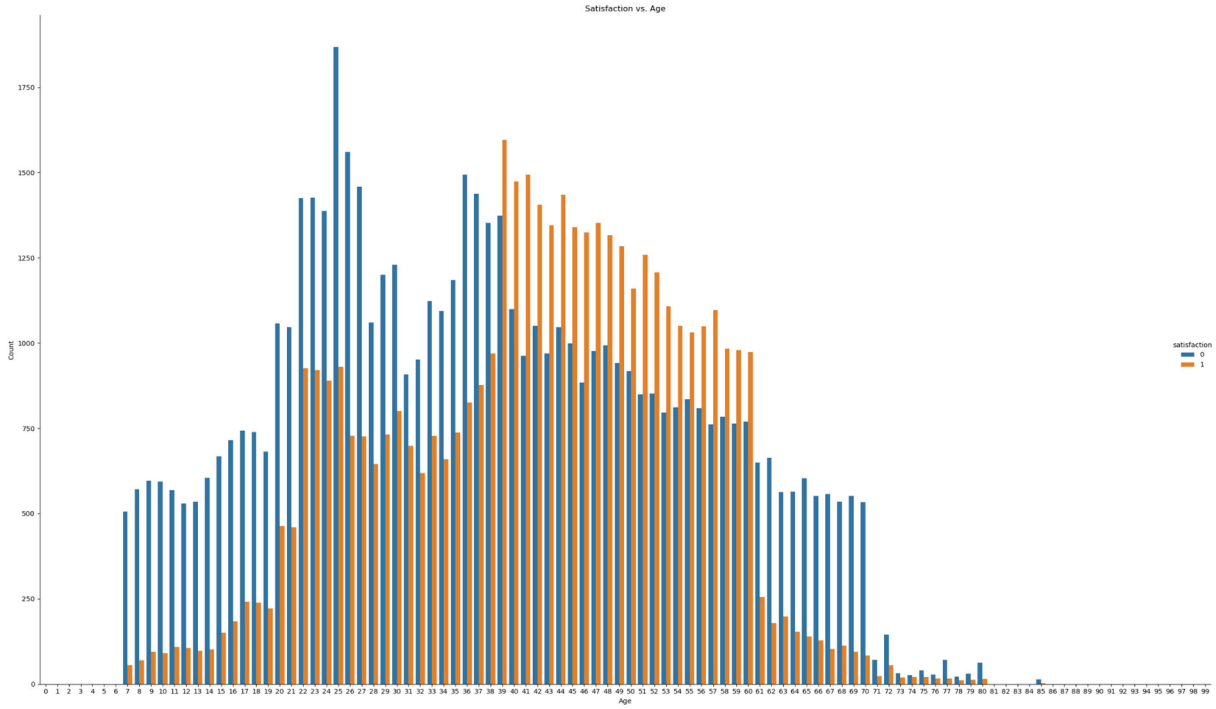

Figure 5 - Classification splits based on age.

The correlation heatmap for the features was plotted and can be seen in Figure 6. Strong feature correlations can be noticed between “Ease of Online Booking” & “Inflight Wifi Service”, “Arrival Delay” & “Departure Delay”, and “Inflight Entertainment” & “Cleanliness” which were 0.72, 0.82, and 0.71 respectively. The largest correlation between some feature and satisfaction was online boarding (whether a customer could board online). “Type of travel” and “seat class” were highly correlated with satisfaction with 0.45 correlation coefficients (they are negative because of the categorical to numeric labeling conversion). “Flight distance”, “seat comfort”, “inflight entertainment”, “onboard service”, “leg room”, and “cleanliness” were also somewhat correlated to satisfaction all above 0.3 correlation coefficients.

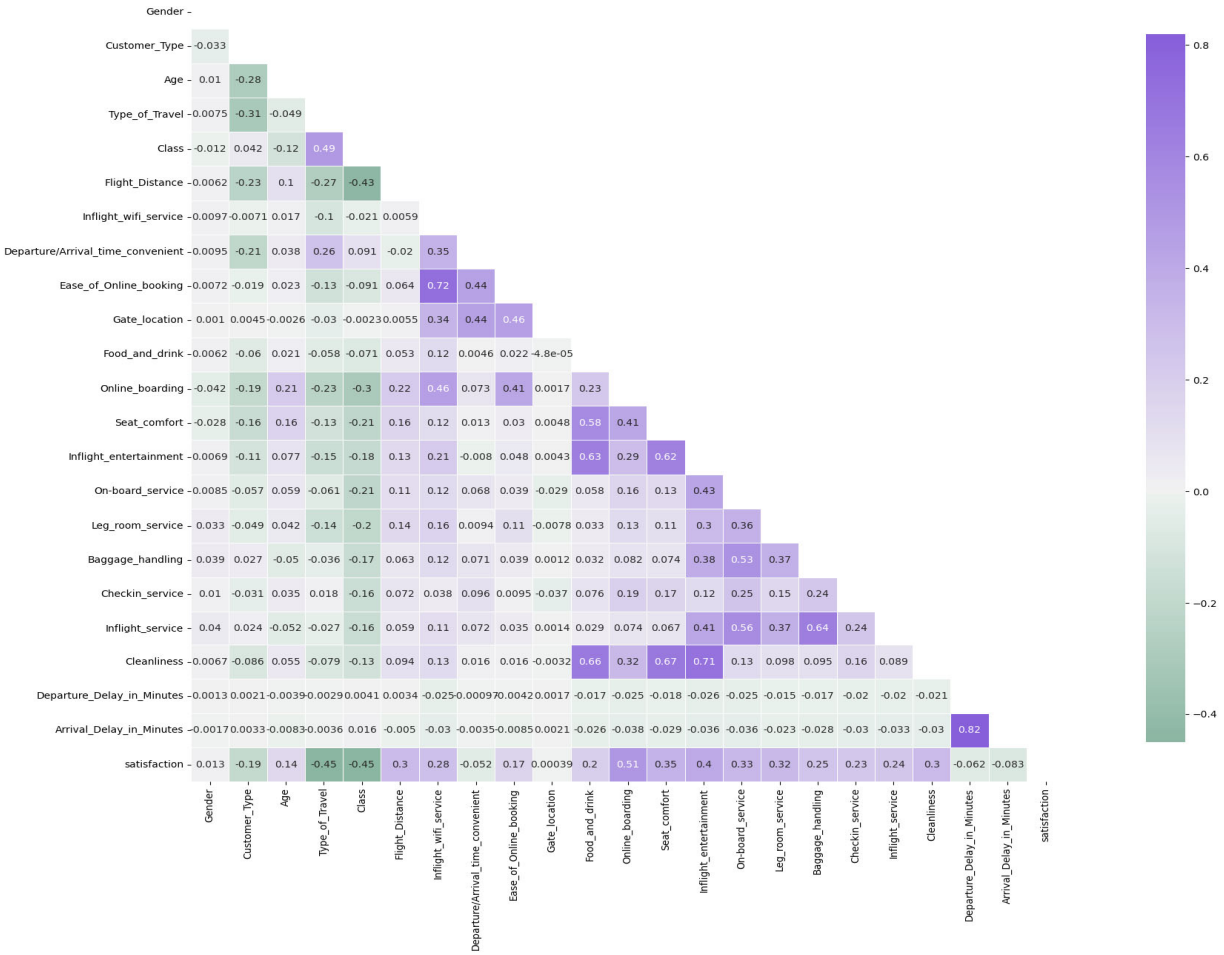

Figure 6 – Correlation heatmap between features.

3 Linear Discriminant Analysis (LDA)

LDA GitHub Code

Next, the function, *PermutationImportance*, was used to determine the importance/weight of the features in the dataset. This function calculates the feature importance of nonlinear or opaque estimators for a given dataset. This technique benefits from the model being agnostic and can be calculated many times with different permutations of the feature. The weights of the features using this function can be seen below.

Weight	Feature
0.2723 ± 0.0039	Type_of_Travel
0.1278 ± 0.0026	Inflight_wifi_service
0.0435 ± 0.0011	Online_boarding
0.0424 ± 0.0013	Seat_comfort
0.0355 ± 0.0009	Checkin_service
0.0294 ± 0.0014	Inflight_service
0.0289 ± 0.0008	Baggage_handling
0.0246 ± 0.0006	Cleanliness
0.0177 ± 0.0007	On-board_service
0.0172 ± 0.0007	Class

Figure 7 – Permutation important of features.

The top 10 features from this technique were used to plot the ROC curve using linear discriminant analysis with the training data set as shown below. Linear discriminant analysis is a classifier with a linear decision boundary, which is

generated by fitting class conditional densities to the data and using Bayes' rule. This model fits a gaussian density to each class, assuming that they all share the same covariance matrix. The fitted model was also used to reduce the dimensionality of the input by projecting it to the most discriminative direction. The LDA classifier was fit on the training dataset and tested on the test dataset. This LDA classifier was used as a baseline for the MLP classifier discussed later.

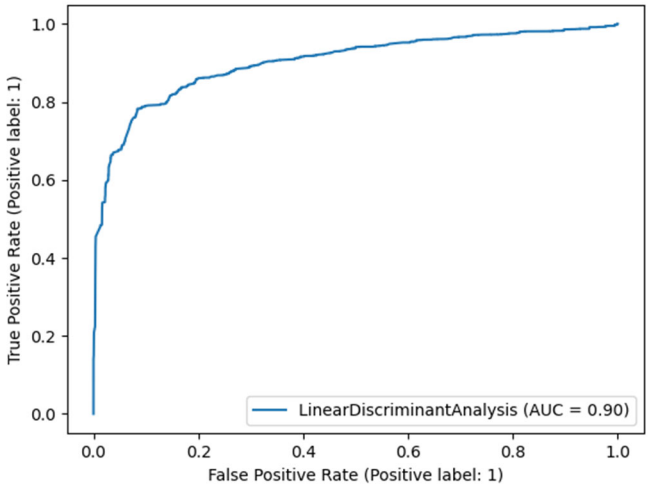

Figure 8 – ROC curve for LDA classifier.

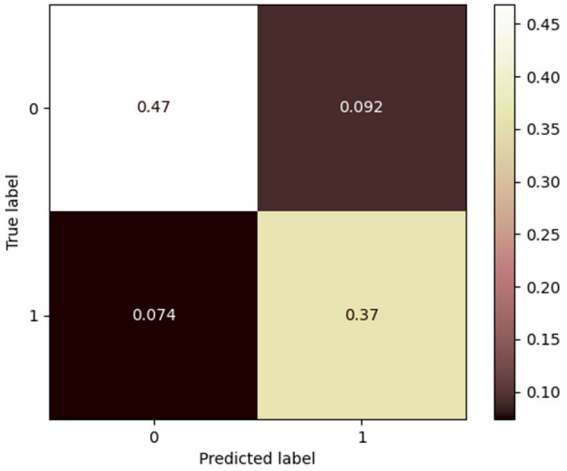

Figure 9 – Confusion matrix for the LDA classifier.

The accuracy from LDA classifier was determined to be 83.4% with the ROC Area under the curve to be 0.8337.

4 MLP Classifier

MLP Classifier GitHub Code

4.1 Neural Network Structure

A neural network was also used to classify the airline satisfaction data. The number of hidden layers and the number of neurons in the hidden layers were determined with grid search cross validation (along with additional hyperparameters) as discussed later. Fully connected linear hidden layers were used with ReLu activation functions. PyTorch's Linear layer and ReLu implementations were used. Stochastic gradient descent was used as the optimization function with CrossEntropyLoss as the loss function. PyTorch's implementation was used for both functions. SoftMax activation was

not used at the output layer of the network as CrossEntropyLoss expects raw prediction values and performs a SoftMax. Early stopping and dropout were implemented and are discussed later.

4.2 Feature Scaling

At first, no feature scaling was performed on the data. Figure 10 shows how training the network works (somewhat) at first since the loss function decreases slightly, but then stops decreasing. The probability of error for that model was 43.3% (i.e. all samples were classified as 1 class and none of the other classes, see Figure 11).

Feature scaling was then added to the training and test sets by scaling the data to 0-mean unit variance. SkLearn's StandardScalar() was used. Figure 11 shows the learning curve with feature scaling: a significant improvement with P(error)=8.16%. Momentum=0.9, learning rate=0.1, 4 layers, and 64 hidden neurons were used as hyperparameters. This model is still not great as none of the other hyperparameters were tuned yet, but the significance of feature scaling was clearly observed.

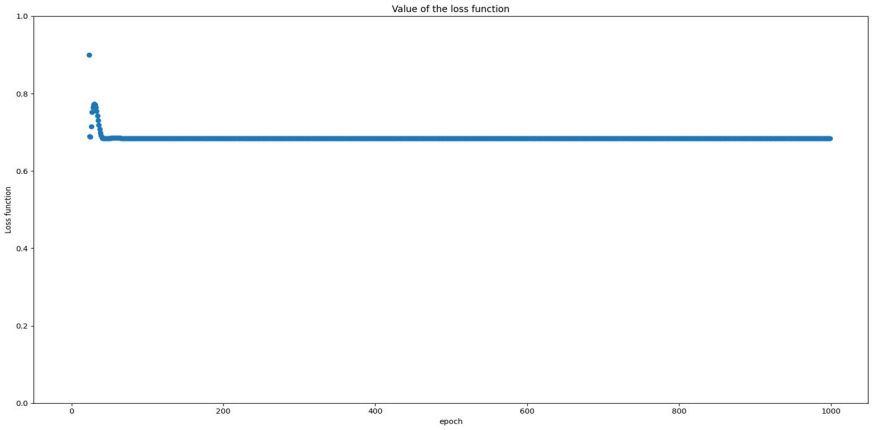

Figure 10 – Learning curve with no feature scaling.

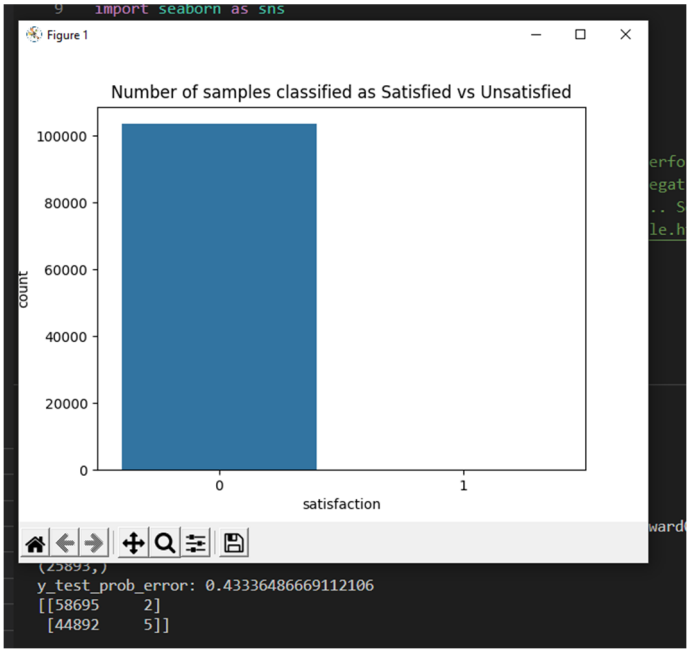

Figure 11 - Classifications with no feature scaling.

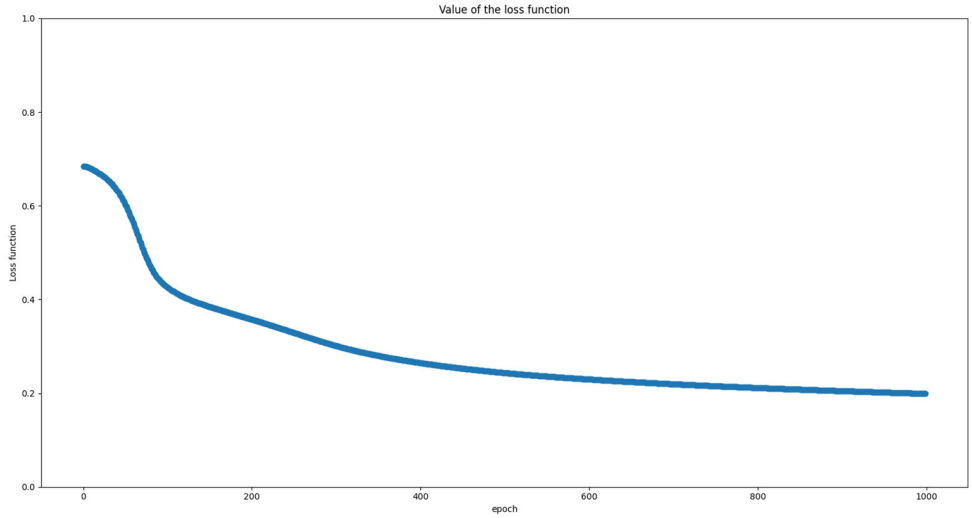

Figure 12 - Loss function with feature scaling.

Figure 13 - Classifications with feature scaling.

4.3 Optimizing Hyperparameters

To find the optimal hyperparameters for the network, implemented grid search cross validation was implemented. A grid of possible hyperparameter combinations was generated by implementing the function generate_hyper_params_grid(). For each combination of hyperparameters from this grid, 10-fold cross validation was performed with maximizing accuracy as the objective function.

Grid search was run multiple times to obtain the optimal hyperparameters. The following sets of parameters were used in the first run. The network was then retrained on the entire training set and tested on the test set. The network produced an accuracy of 93%. Figure 14 shows the learning curve for the parameters determines in this run.

Parameters Used in Grid Search CV:

n_hidden_neurons = [2, 4, 8, 16, 32, 64, 128, 256]
learning rate = [.1, .01, 10e-3, 10e-4, 10e-5]
momentum = [0.1, 0.5, 0.9, 0.99]
num_epochs = [200]

Determined Optimal Parameters:

n_hidden_neurons = 256
learning rate = 0.1
momentum = 0.99
num_epochs = 200

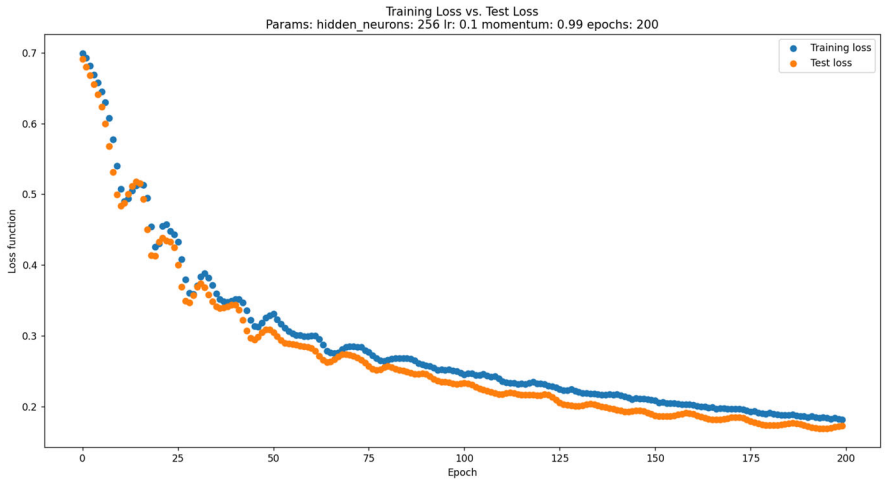

Figure 14 – Learning curve from first run of grid search CV.

The second run of the grid search CV was done with the following parameters. The network produced an accuracy of 94.4% and its learning curve can be seen in Figure 15.

Parameters Used in Grid Search CV:

n_hidden_neurons = [256, 512, 1028]
learning rate = [.1, .2, .3]
momentum = [0.99]
num_epochs = [200]

Determined Optimal Paramters:

n_hidden_neurons = 1028
learning rate = 0.3
momentum = 0.99
num_epochs = 200

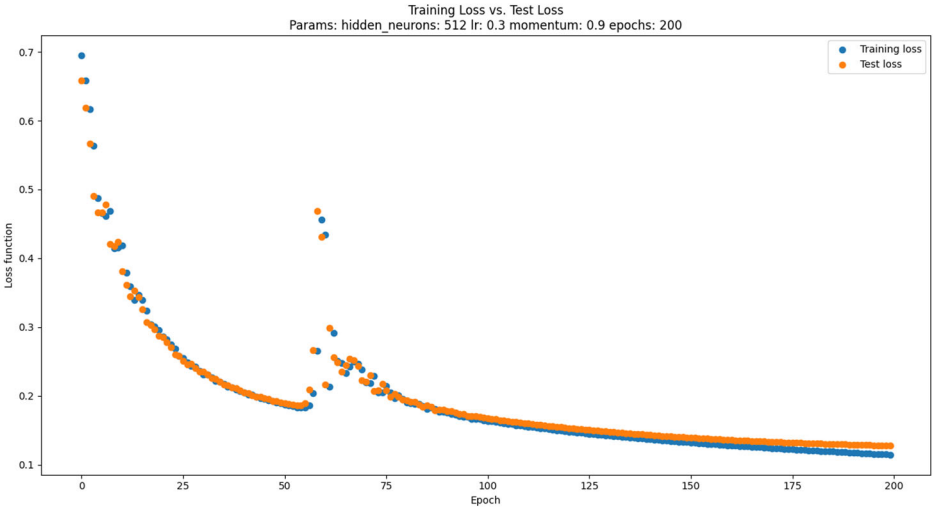

Figure 15 – Learning curve from the second run of grid search CV.

Due to computing constraints, the learning rate was optimized further on its own with 5-fold CV and the following parameters. The results of the cross validation to select learning rate can be seen in Figure 16**.** As seen in the figure, a learning rate of 0.8 was found to be optimal.

n_hidden_neurons_list = [128]
lr_list = [.1, .2, .3, .4, .5, .6, .7, .8, .9 , .99]
momentum_list = [0.99]
num_epochs_list = [200]

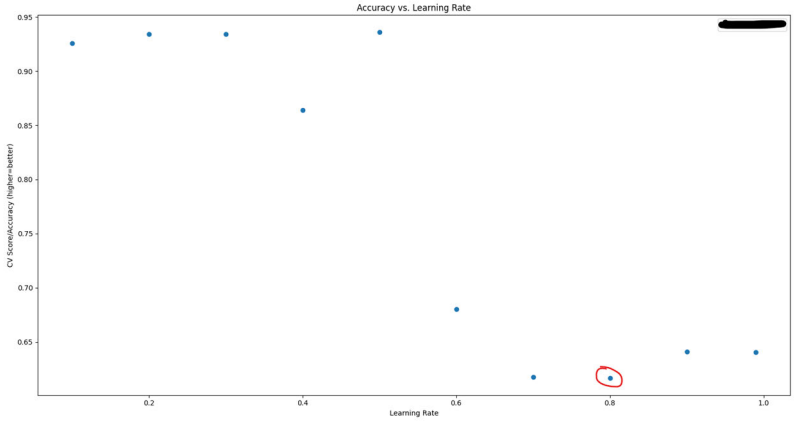

Figure 16 – CV score (accuracy) vs learning rate.

Although 1028 neurons were the optimal number of neurons from the grid search CV (and most likely more neurons would have yielded better performance), due to computing constraints, the number of hidden neurons was set to 512. It should also be noted that in the final training, the model was found to be unstable, so the learning rate was reduced to 0.3. The parameters used for the final model are found below in Figure 17.

# Hidden Neurons	512
# Hidden Layers	4
Learning rate	0.3
Momentum	0.99
# Epochs	200

Figure 17 - Final hyperparameters used.

The parameters above produced a model with 94.7% accuracy. A more detailed classification report (using SkLearn's classification_report() implementation) can be found in Figure 18**.** The learning curve for these parameters can be seen in Figure 19. As can be seen in the learning curve, the model generalizes well, but begins to diverge slightly. This will be addressed in the regularization section below, although it was later determined that the model was not significantly overfit.

	precision	recall	f1-score	support
neutral or dissatisfied	0.94	0.97	0.95	14528
satisfied	0.96	0.91	0.94	11365
accuracy			0.95	25893
macro avg	0.95	0.94	0.95	25893
weighted avg	0.95	0.95	0.95	25893

Figure 18 – Classification report of final trained model. Note: Dropout and early stopping were not implemented here, but these regularization techniques did not have a significant impact on the model. This is discussed later.

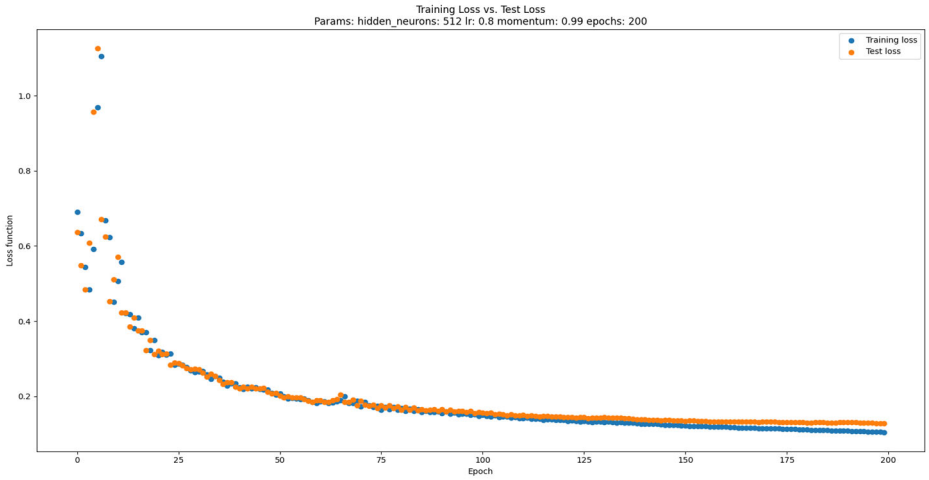

Figure 19 – Learning curve of final trained model.

4.4 Regularization (Underfitting and Overfitting)

Before addressing the slight overfitting seen in the model and learning curve above, we further explored underfitting and overfitting of the model. The following hyperparameters were used for such experiments:

Learning rate = 0.8
momentum = 0.99
Epochs = 500

4.4.1 Underfitting

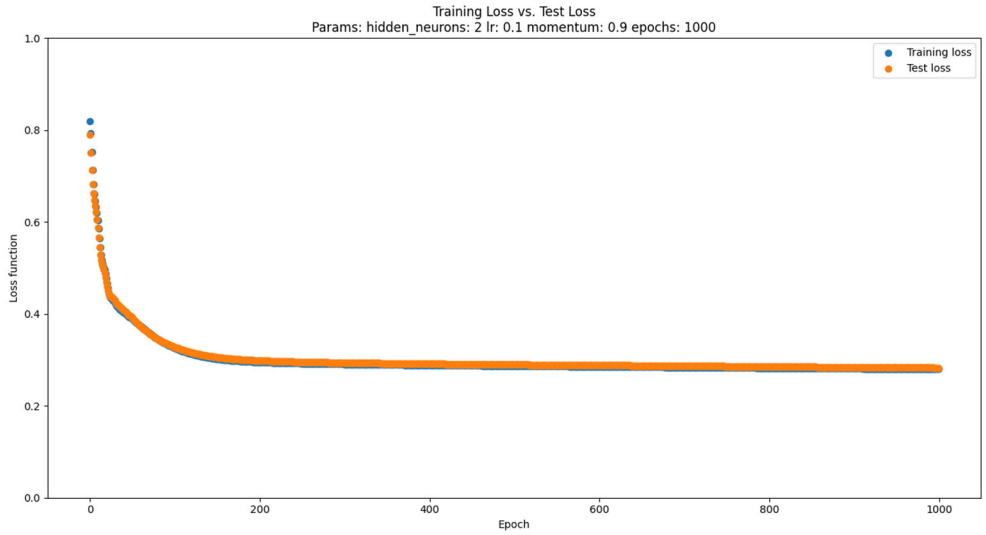

Figure 20 – Learning curve when attempting to underfit data.

We attempted to intentionally underfit the model by using a 1-layer, 2-neuron network, however, the network still achieved an accuracy of 84% classification. Figure 20 shows the learning curve when underfitting was attempted.

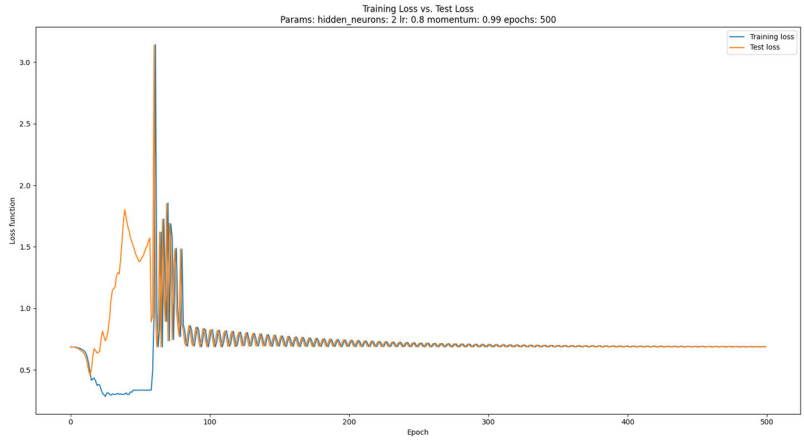

Figure 21 – Learning curve of model when less training samples were used to visualize underfitting of the model.

We reduced the training set to only 100 samples while keeping the same ~26,000 test set samples. A 4-layer, 2 neuron network was used this time. Underfitting was observed as seen in Figure 21. As seen in the figure, the loss from the test set initially goes down, but then increases while the training loss continually goes down (when looking at an epoch range of 0-50). The accuracy of the model on the test set was 56%. Underfitting is addressed by increasing the capacity of the network (i.e. increasing the number of layers and the number of neurons per layer.

4.4.2 Overfitting

Like underfitting, we reduced the training set to 100 samples and used a 4-layer, 512-neuron network. As can be seen in Figure 22, the testing loss and training loss are near perfectly fitting the data, but then at epoch 15 thy spike up indicating an overfit model that does not generalize well to unseen data. The accuracy of this model on the test set was 56%.

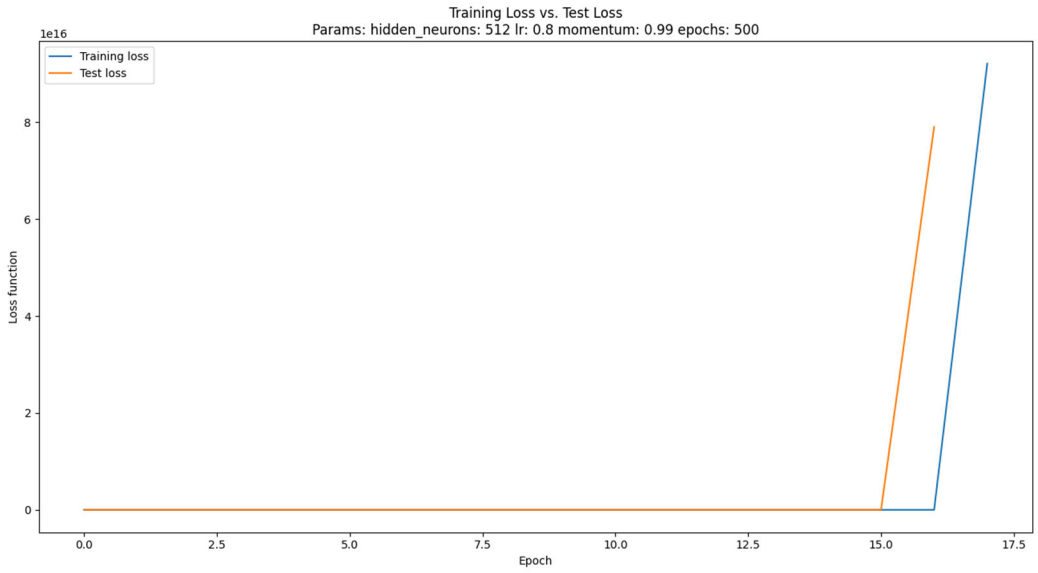

Figure 22 – Learning curve of intentionally overfit model.

A potentially more likely overfitting scenario is with a 5,000 sample training set and with the parameters listed in Figure 23 below. As can be seen in the figure, the training loss and test/validation loss match each other well at first, but then diverge, indicating overfitting. The model still produced an accuracy of 90%

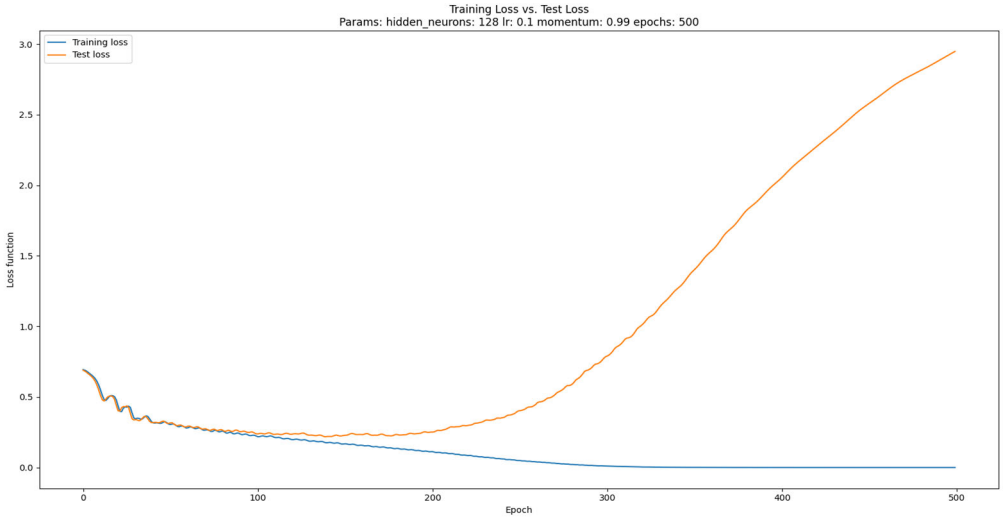

Figure 23 - Learning curve without dropout.

To prevent overfitting, dropout was implemented using PyTorch's Dropout() implementation. Dropout was added after each hidden layer with a probability of 25%. The effect of using dropout can be seen in Figure 24 below: the validation loss does not run away nearly as much as in the model without dropout. The accuracy of the model with dropout was 93%, a 3% increase.

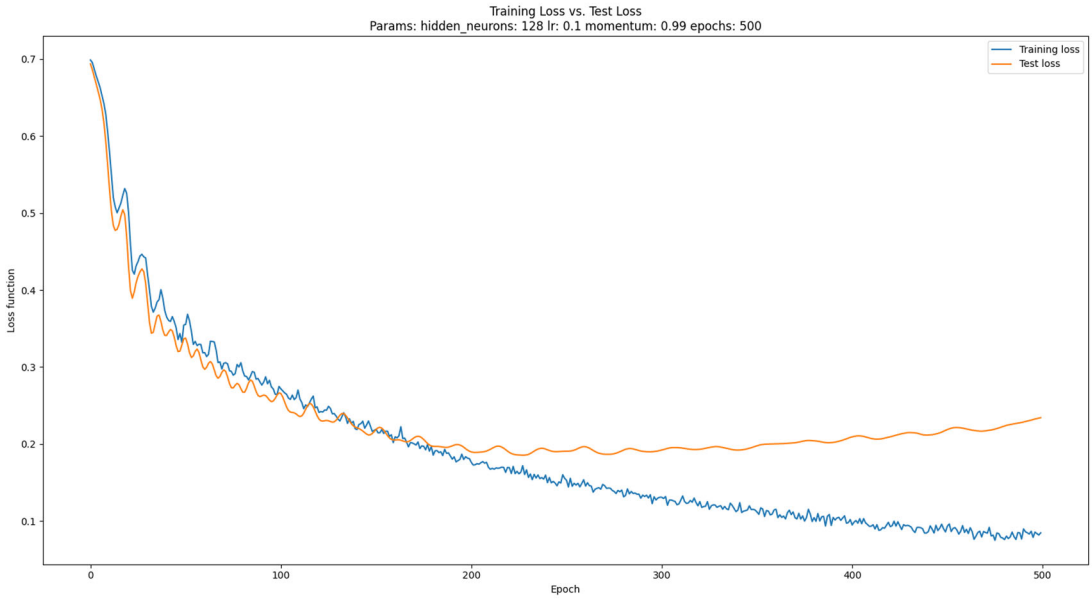

Figure 24 – Learning curve with dropout.

Early stopping was implemented in the train_model() function. Early stopping was implemented as follows: training stops if a certain number of early stopping events are triggered consecutively. This "certain number of events" is the "patience". An early stopping event is triggered if the validation loss increases from the previous epoch or if the training loss and validation loss are separated by a given threshold. Figure 25 shows the learning curve of the model with dropout and early stopping implemented. The accuracy of the new model was unchanged at 93%.

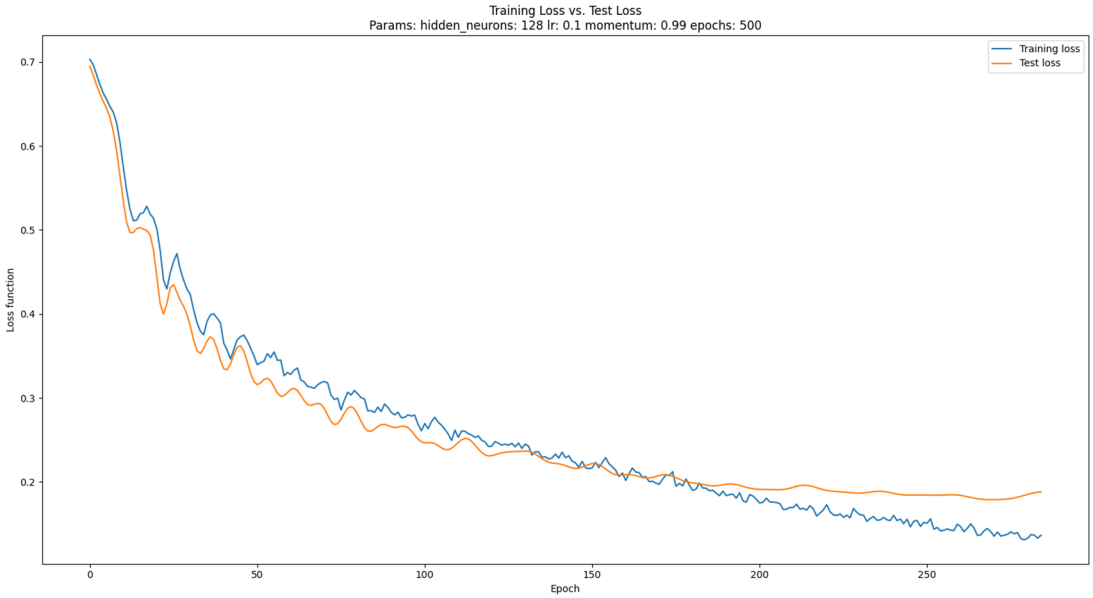

Figure 25 – Learning curve with dropout and early stopping.

The final model was trained with the parameters from Figure 17 and with dropout and early stopping. It produced an accuracy of 93.5%, indicating that dropout and early stopping did not help the final model, but these regularization methods also did not make the model worse. This indicates that the model was not overfitting the data to begin with and is most likely due to the large dataset used. Nonetheless, early stopping did improve runtime as the training stopped before the max number of epochs. The learning curve can be seen in Figure 26 below with the classification report in Figure 27.

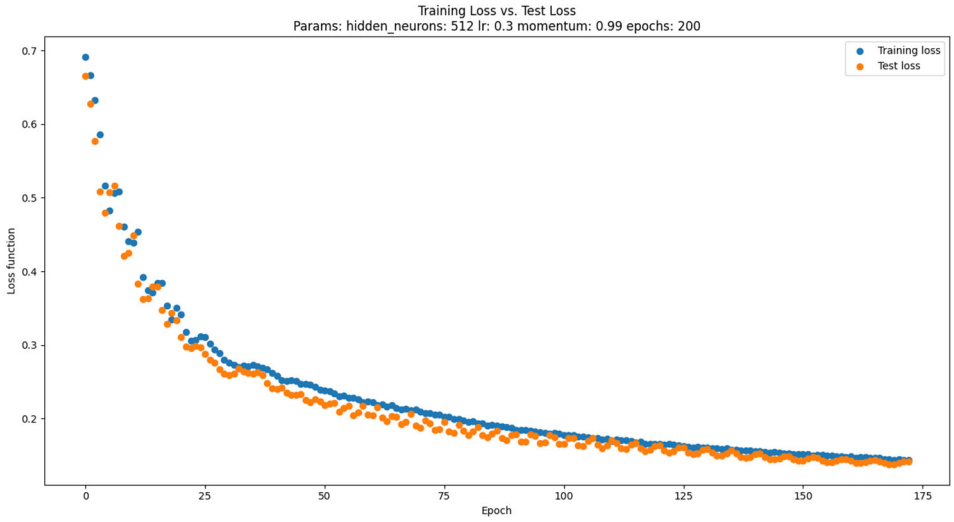

Figure 26 - Final trained model with early stopping, dropout, and the hyperparameters determined from grid search cross validation.

	precision	recall	f1-score	support
neutral or dissatisfied	0.92	0.98	0.95	14528
satisfied	0.98	0.88	0.93	11365
accuracy			0.95	25893
macro avg	0.95	0.93	0.94	25893
weighted avg	0.94	0.94	0.94	25893

Figure 27 – Final classification report for the final model.

4.5 Conclusion

An LDA classifier was trained on the dataset and produced an accuracy of 83.4%. A 4 layer MLP was trained that produced a final accuracy of 93.5%, nearly 10% better than the LDA classifier. To train the MLP model, feature scaling was first performed on the data which had a significant impact (i.e. a neural network must have data scaled). Hyperparameters were then tuned using grid search 10-fold cross validation, and regularization was introduced to the training process via early stopping and dropout though the model was not significantly overfit to begin with.