Project 2 Report

CS 135 Intro to Machine Learning

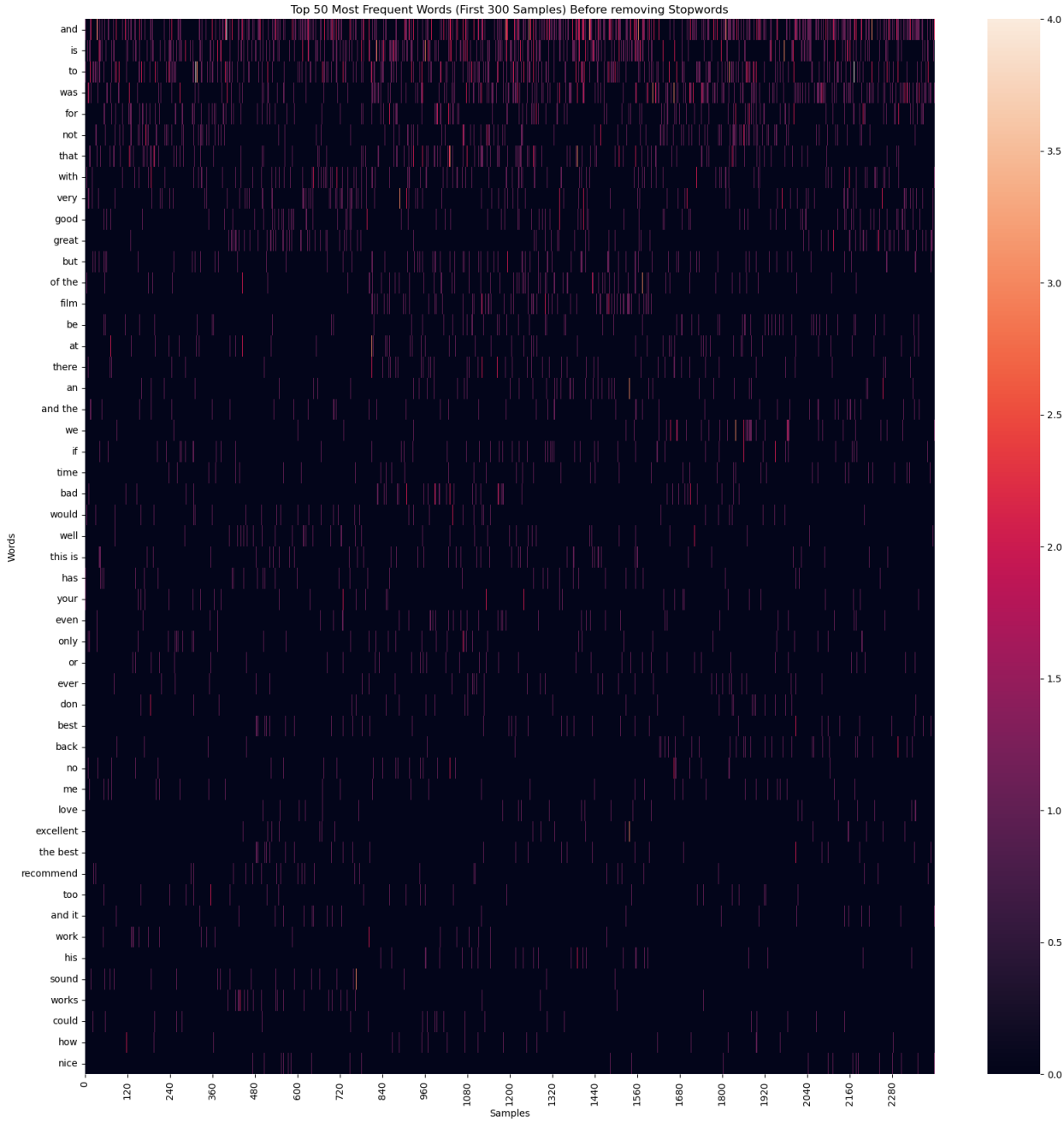
Student ID: 1409557

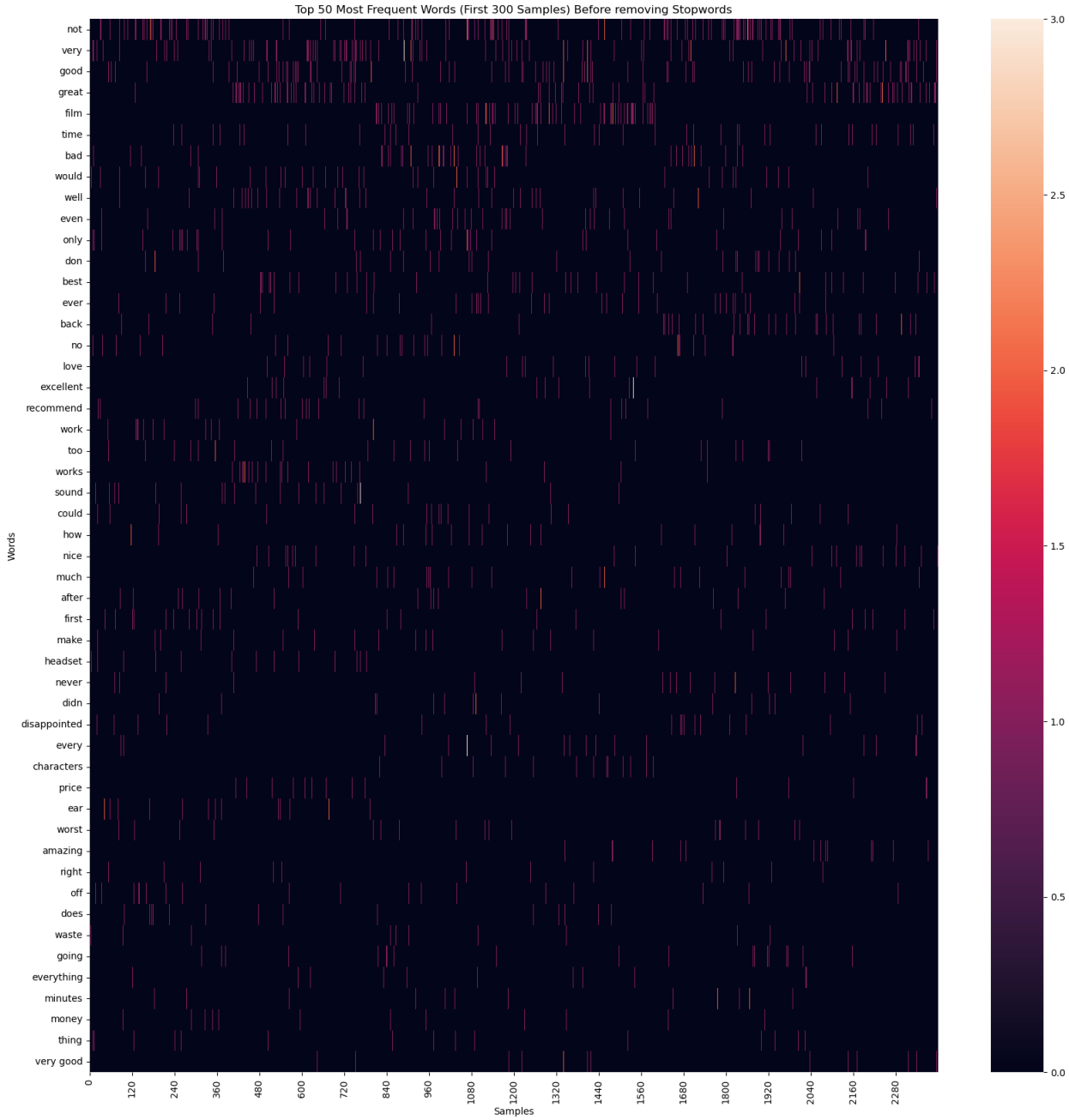
Emmet Hayes

In this project, I am building models for binary classification that generate accurate sentiment labels for single-sentence reviews collected from websites like imdb, amazon, and yelp. The input data consists of two columns: the source website of the review, and the review text itself.
A positive review is given a 1, and a negative review a 0. Before I explore the different models, I will first explain the process of feature selection that I chose to follow for this project.

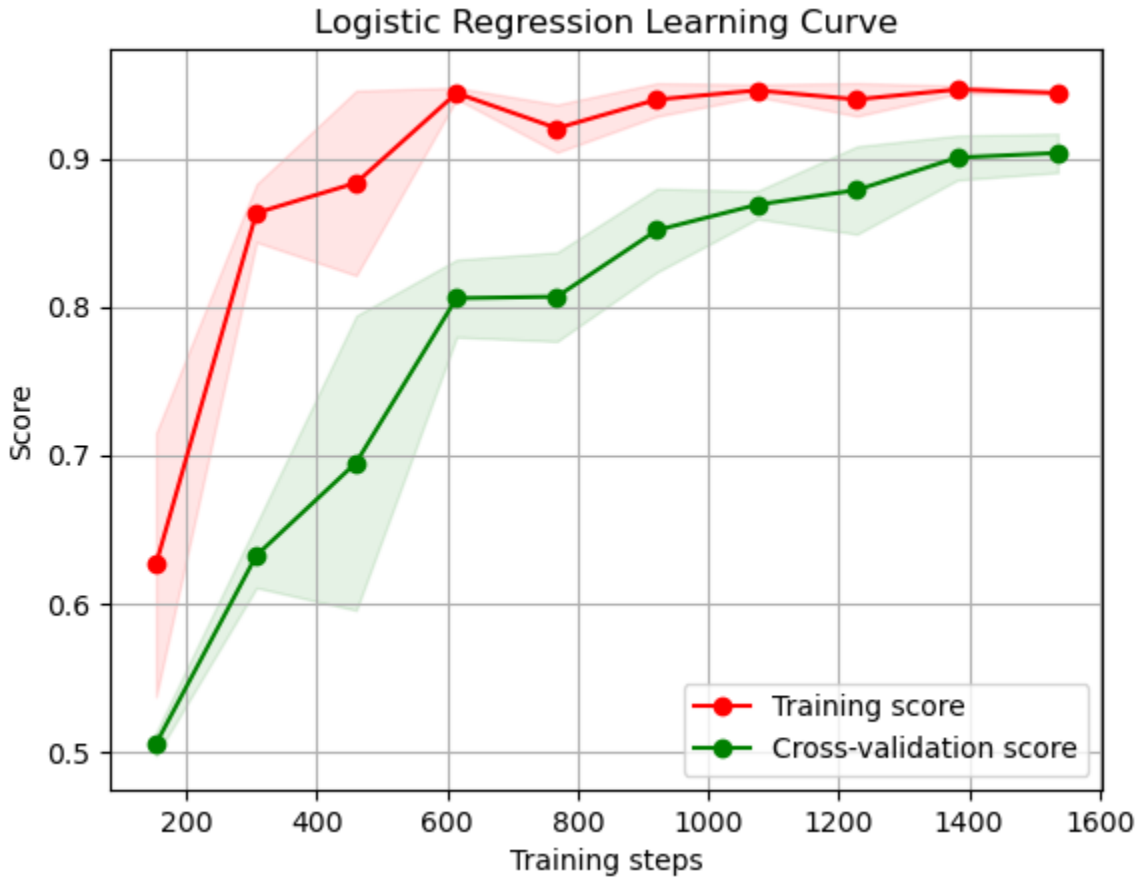
Part 1. Before anything else, I used some regular expressions to do some preliminary preprocessing to the text. I stripped all punctuation, all non-alphabetic and non white-space characters, and set all alphabetic characters to lower-case in the reviews. I did this because I only wanted the model to care about the actual words in the reviews. I wanted to make sure the model would only consider the lower-case form of the word, since it is very unlikely to make a difference if a word is capitalized or not, and would just add more complexity to the model if we didn't make this decision.

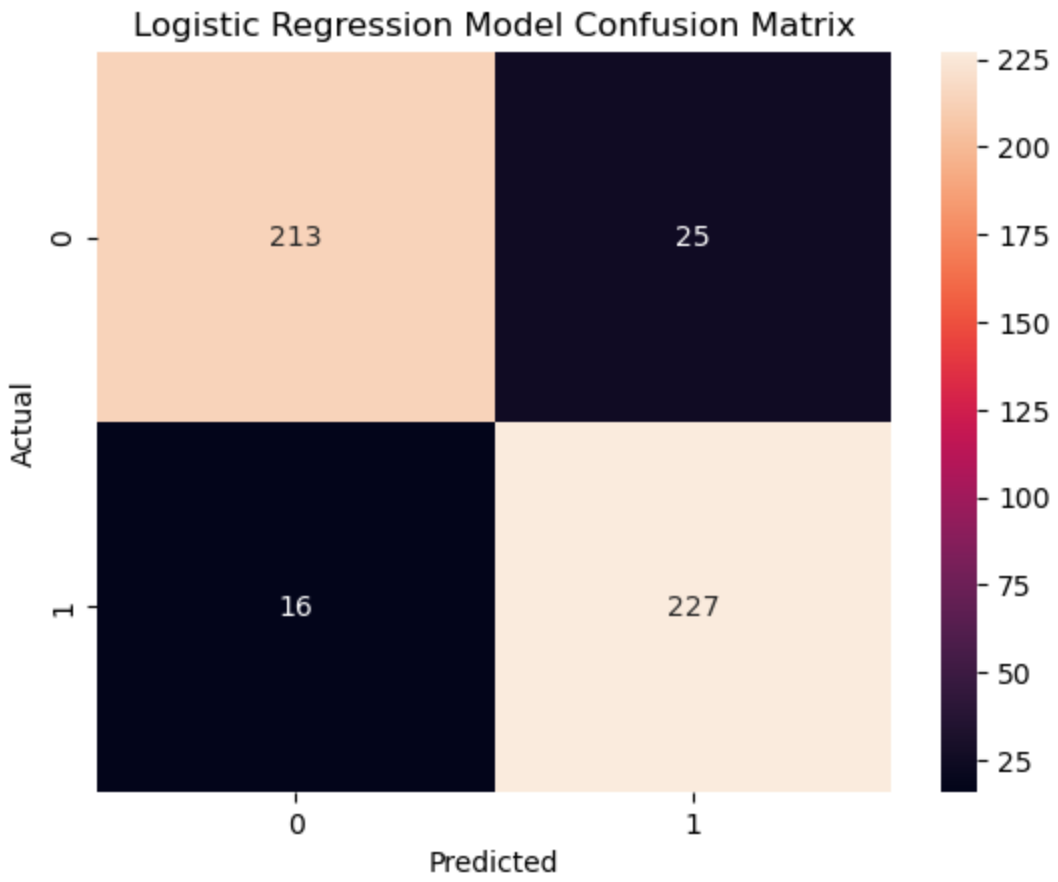
The next step I took after using the regular expressions to strip unnecessary characters, I called the CountVectorizer to vectorize the words in the reviews. After generating a visualization of the occurrences of each word, The visualization helped me pick better stop-words, as the most common words in English like 'the' and 'and' etc. were clearly adding some non-sentimental noise to the data by noticing these words., and I have included visualizations before and after adding the stop-words array to the count vectorizer for this phenomena. Both vectorizer classes I used during experimentation accept stop_words as a parameter, so I came up with a list of stop words that slightly improved the performance of all three of the models. They also both accept n_grams, which I decided to include word-pairs so that reviews that say 'not good' or 'very good' will treat the whole pair as a word, which I believe is more meaningful to the model than when those words are separate. Then, I added a chi-squared test to the pipeline for feature selection, which actually improved the model even more than adding the stop words list! The last two steps I took to preprocess the data is that I scaled it using a standard scaler, and then I split the training data into 80% training and a 20% training evaluation set. During my experiments, I found that this pipeline significantly improved model accuracy across all three models.


Top 50 Most Frequent Words (First 300 Samples) Before removing Stopwords
Words
and
is
to
was
for
not
that
with
very
good
great
but
of the
film
be
at
there
an
and the
we
if
time
bad
would
well
this is
has
your
even
only
or
ever
don
best
back
no
me
love
excellent
the best
recommend
too
and it
work
his
sound
works
could
how
nice
0
120
240
360
480
600
720
840
960
1080
1200
1320
1440
1560
1680
1800
1920
2040
2160
2280
Samples
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0


Top 50 Most Frequent Words (First 300 Samples) Before removing Stopwords
Words
not
very
good
great
film
time
bad
would
well
even
only
don
best
ever
back
no
love
excellent
recommend
work
too
works
sound
could
how
nice
much
after
first
make
headset
never
didn
disappointed
every
characters
price
ear
worst
amazing
right
off
does
waste
going
everything
minutes
money
thing
very good
0
120
240
360
480
600
720
840
960
1080
1200
1320
1440
1560
1680
1800
1920
2040
2160
2280
Samples
3.0
2.5
2.0
1.5
1.0
0.5
0.0

Part 2. The logistic regression model that I generated for the feature-data was relatively simple to experiment with. I used a grid-search on the C parameter to find the best value for training, which ended up being 0.0001. As I was generating plots off the learning curve of the model, I found that high values of C would cause the model to hit a high training accuracy very quickly, but would struggle to improve its cross-validation score. I chose to explore different powers of ten for C in a grid search during my experiments because of the logarithmic scale of the parameter, as well as for simplicity's sake. We can see from the plot below that the training and cross-validation scores have similar slopes, which is good because this suggests the model is not overfitting, since it is improving on both the training and evaluation set at a similar rate. Another hyperparameter that I played around with was the solver. I ended up using 'lbfgs', because it produced the model with the highest cross-validation accuracy, and it also trained faster than 'liblinear'. My explorations of models using the liblinear solver yielded slightly worse results, and I believe this is because L-BFGS can make use of curvature information, leading it to converge more quickly, and in this case, more accurately. Uncertainty is being represented in the plot below in the lightly-shaded red and green regions along the learning curve lines.



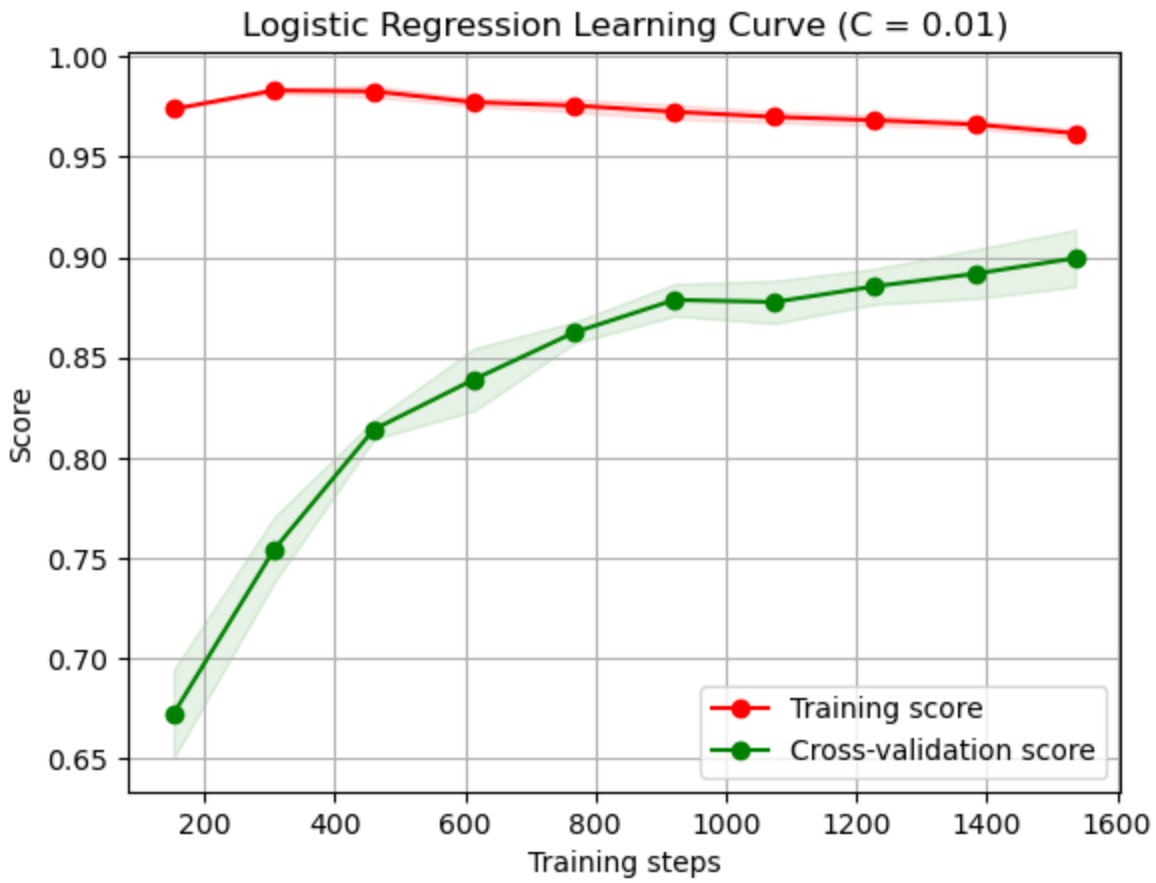


Accuracy: 0.9147609147609148
Classification report:

	precision	recall	f1-score	support
0	0.93	0.89	0.91	238
1	0.90	0.93	0.92	243
accuracy			0.91	481
macro avg	0.92	0.91	0.91	481
weighted avg	0.92	0.91	0.91	481

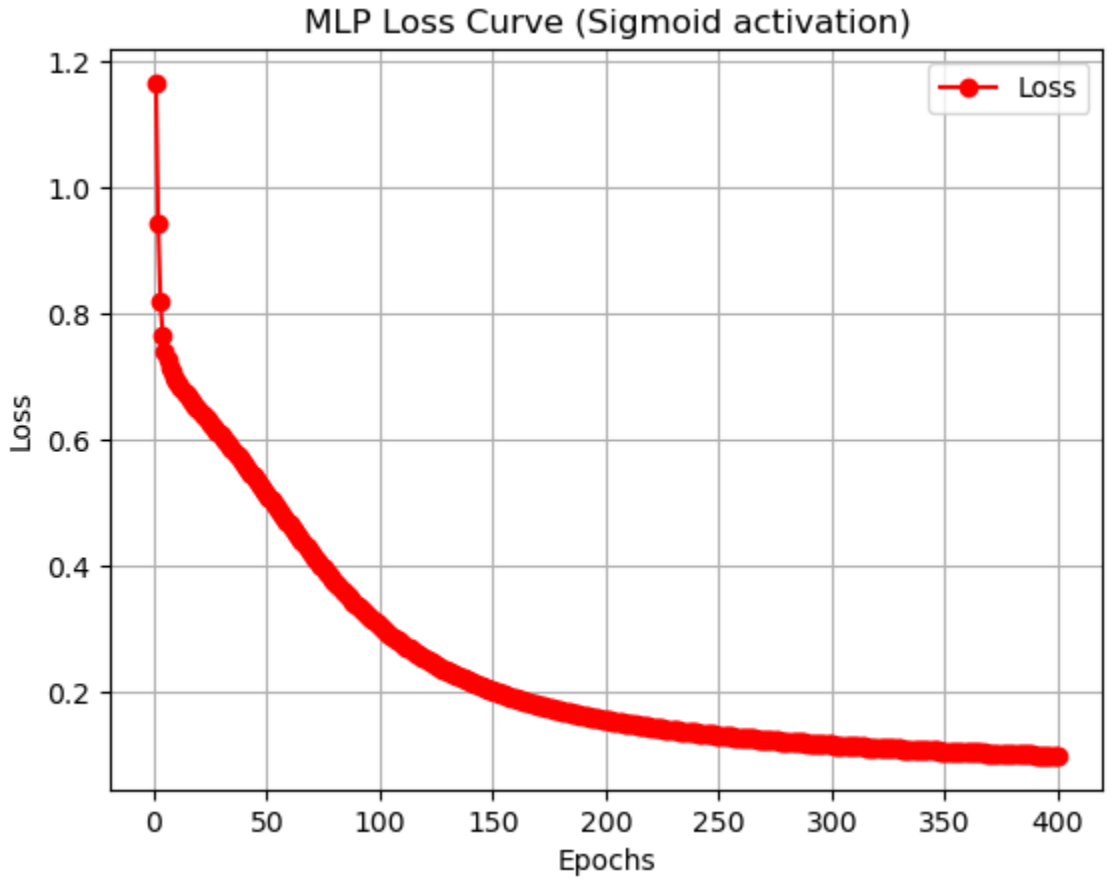
5-Fold Cross-Validation:
- Average accuracy: 0.8005
- Standard deviation: 0.0131

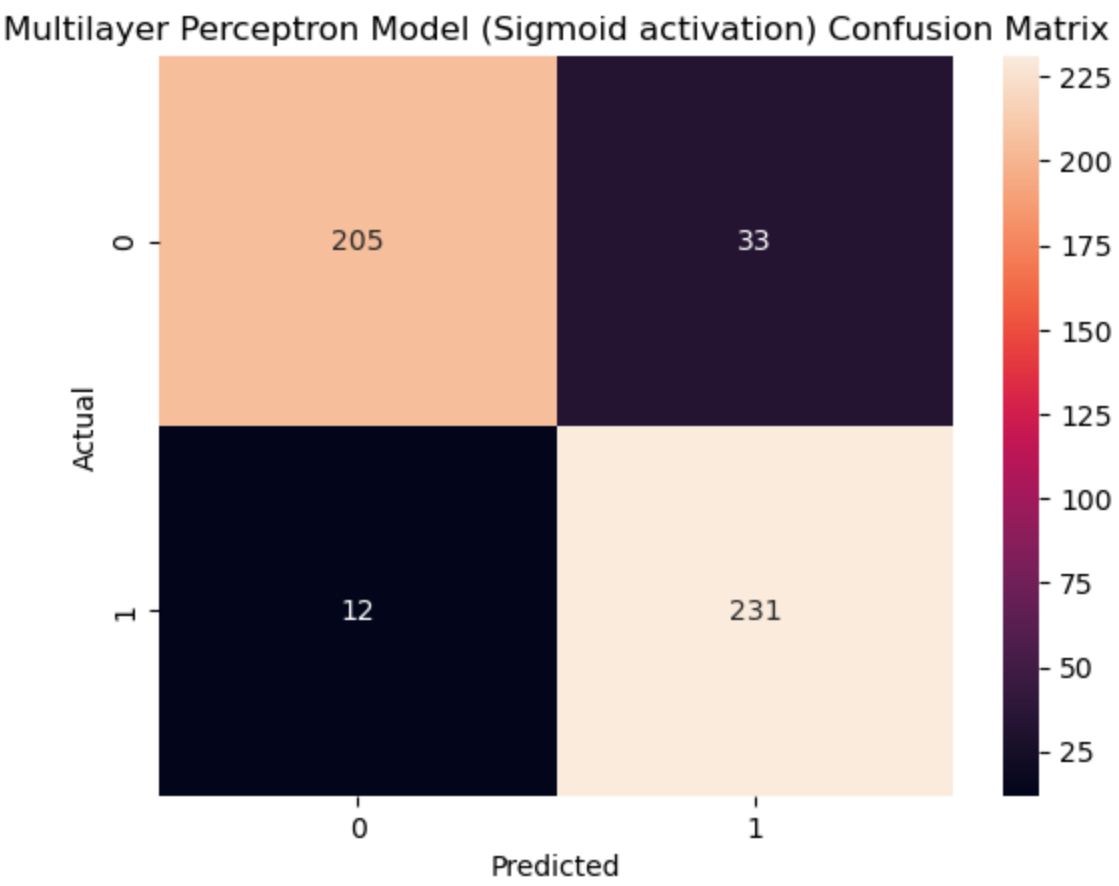
The hyperparameter settings I chose (solver='lbfgs', C=0.0001, penalty='l2', max_iter=100) produced the best model I found during experimentation. This model tends to be slightly better at correctly predicting positive reviews than negative reviews. I found that larger values of C produced models that were overfitting on the training data. Here is one more additional plot on the next page, showing this phenomena:



I included this plot above to show the effects of overfitting by picking a bad C value. We can see that the model is already scoring very highly on the training set at the start, suggesting that this model will have trouble generalizing, and will have to try and improve over training steps.

Part 3. The Neural Network model that I generated for the feature-data was generally harder to find good parameters for than the Logistic Regression model. The size of the hidden layer in the network was probably the hardest parameter to tune in this whole project. I found that a single hidden layer sufficed to produce a model that scored over 90% accuracy during cross-validation. This may be because the problem of sentiment analysis may not need to have multiple hidden layers to learn more complex and abstract features from the data. I also experimented with the solver and activation of the model, and found stochastic gradient descent combined with logistic activation and a single hidden layer (of size 88) produced the most accurate model, at least according to the accuracy and cross-validation scores I was receiving.





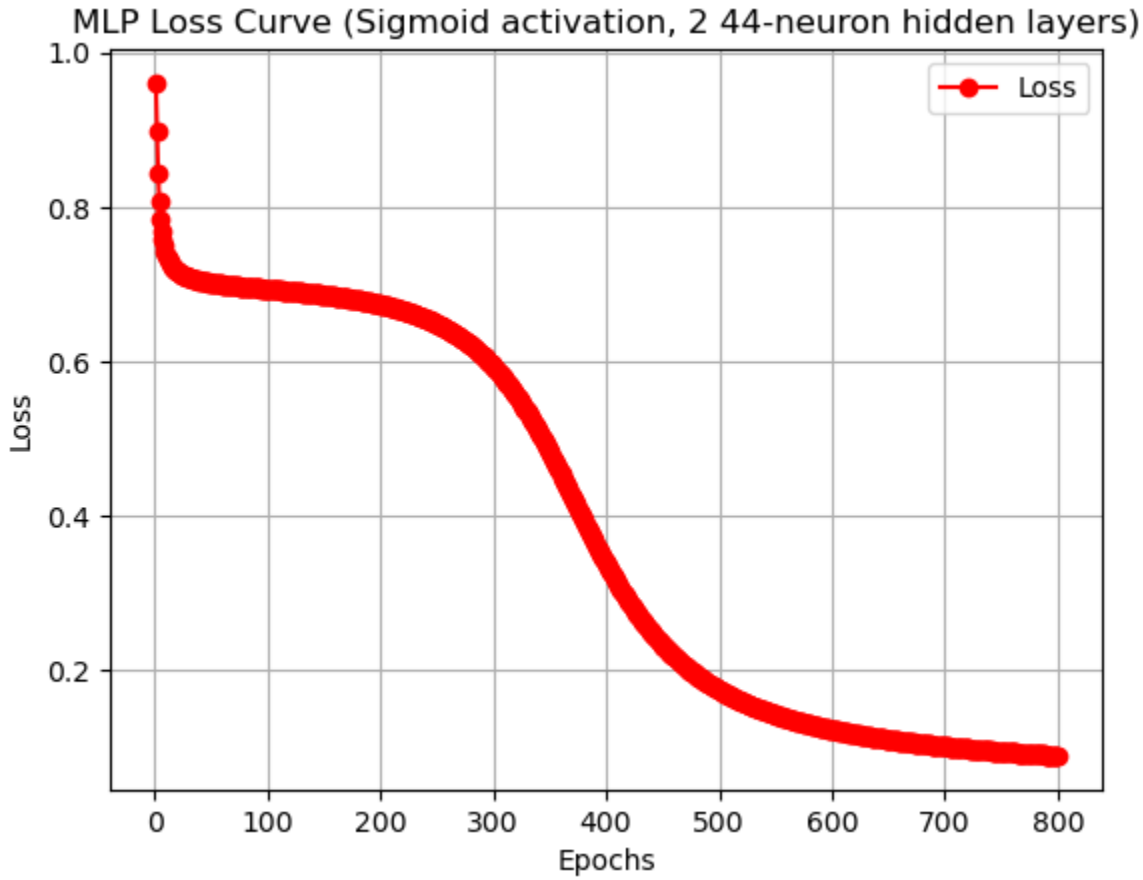
Accuracy: 0.9064449064449065
Classification report:

	precision	recall	f1-score	support
0	0.94	0.86	0.90	238
1	0.88	0.95	0.91	243
accuracy			0.91	481
macro avg	0.91	0.91	0.91	481
weighted avg	0.91	0.91	0.91	481

5-Fold Cross-Validation:
- Average accuracy: 0.8026
- Standard deviation: 0.0132

As we can see from the classification report and confusion matrix above, the model has an improved cross-validation score than the linear regression model, despite achieving lower accuracy. Using sigmoid activation with an SGD model and a single hidden layer proved to yield the most accurate

final model. There is some evidence of over-fitting with models with multiple hidden layers, as shown in the plot below.

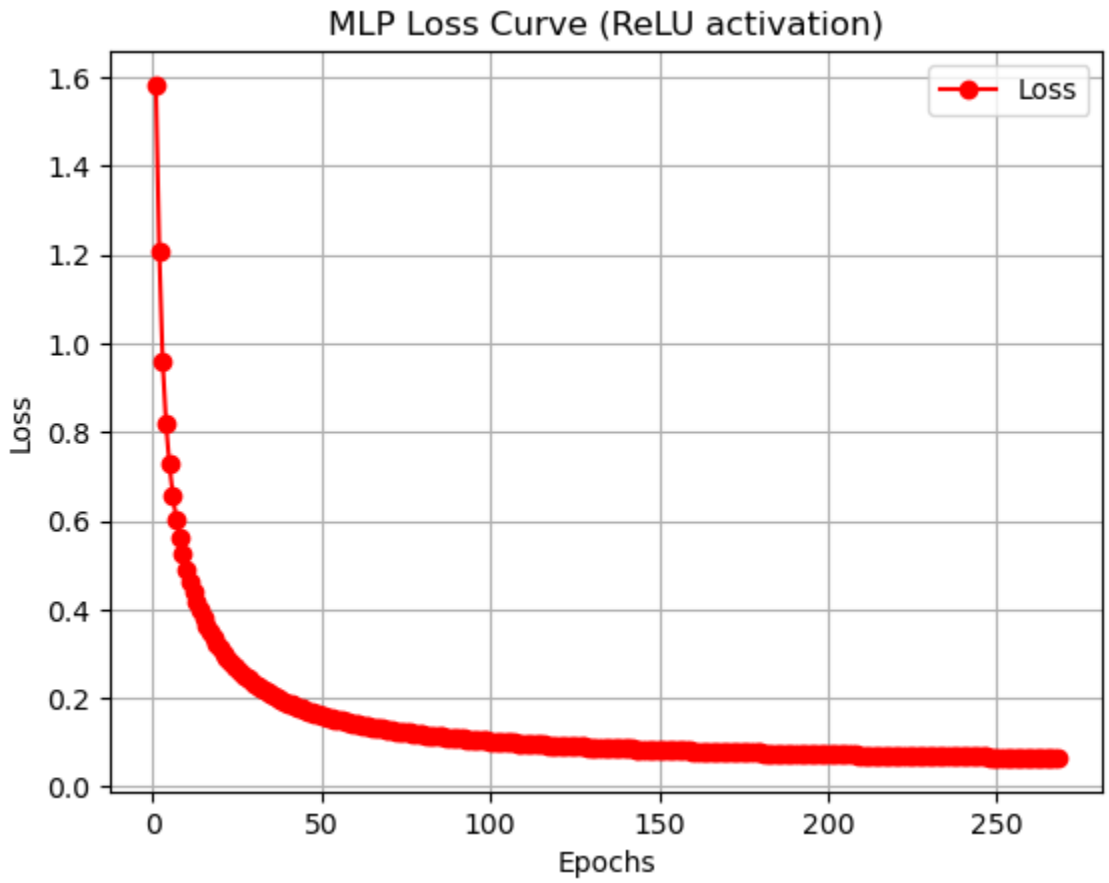


Accuracy: 0.8981288981288982
Classification report:

	precision	recall	f1-score	support
0	0.93	0.86	0.89	238
1	0.87	0.94	0.90	243
accuracy			0.90	481
macro avg	0.90	0.90	0.90	481
weighted avg	0.90	0.90	0.90	481

5-Fold Cross-Validation:
- Average accuracy: 0.7327
- Standard deviation: 0.1295

When experimenting with the activation functions, I found that ReLU converged quickly, but yielded a model that was not as capable of accurate predictions for our use case. Perhaps this is due to the fact that sigmoid activation has a clear probabilistic interpretation for binary classification problems, which is advantageous for positive/negative sentiment analysis, while ReLU does not have this property.

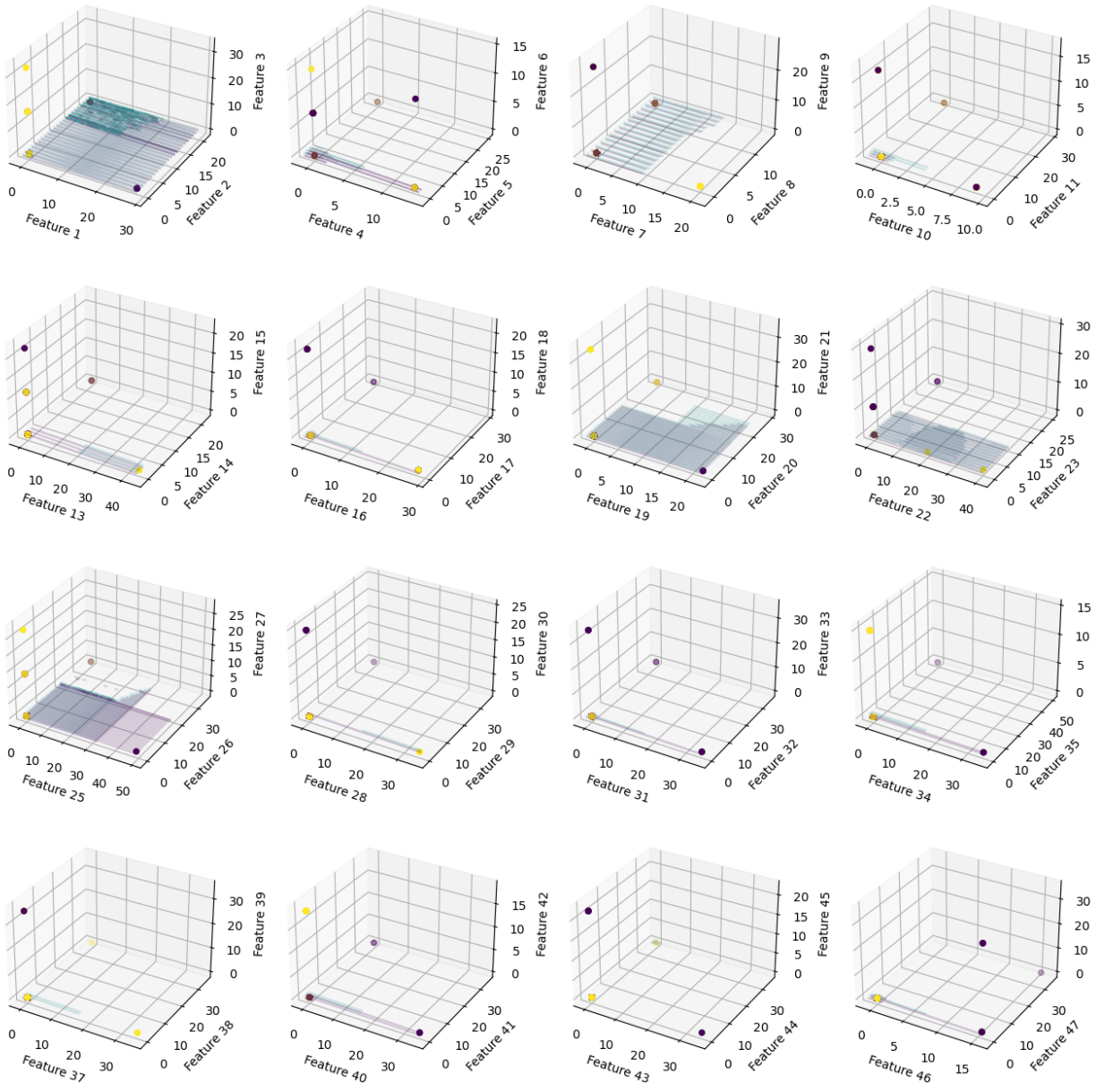


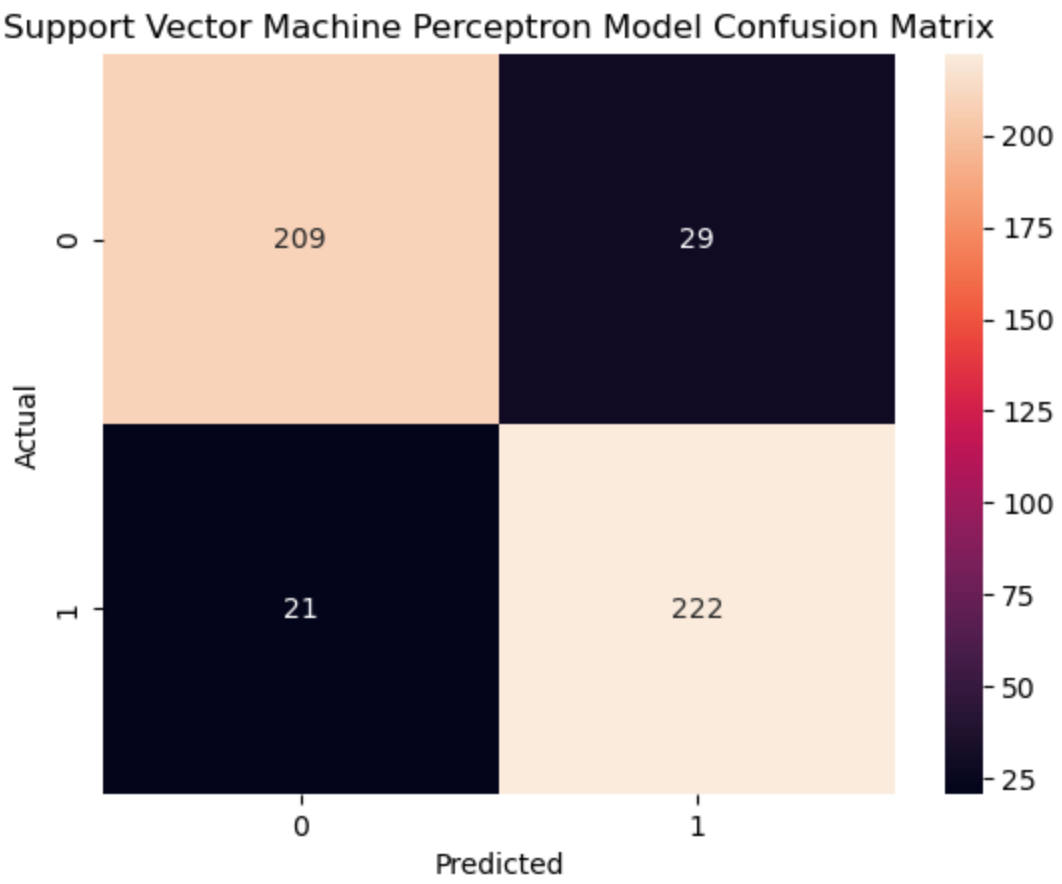
Accuracy: 0.8565488565488566
Classification report:

	precision	recall	f1-score	support
0	0.89	0.81	0.85	238
1	0.83	0.90	0.86	243
accuracy			0.86	481
macro avg	0.86	0.86	0.86	481
weighted avg	0.86	0.86	0.86	481

5-Fold Cross-Validation:
- Average accuracy: 0.7922
- Standard deviation: 0.0091

Part 4. At first, I chose to generate an SVM classifier as my third model, but my final and most accurate model that was submitted to the leaderboard was actually an ensemble model that used all three strategies, and I am happy to say that it actually turned out to be the most accurate model that I was able to produce for completing the given sentiment analysis task. Focusing on the SVM, the first hyperparameter that I played with for the SVM was the kernel. When comparing the performance using the Radial Basis Function to the Linear kernel, there was a clear performance improvement in the model that used RBF. This is likely because the sentiment analysis problem we are working on in this project has lots of non-linear relationships in the data. A linear kernel SVM is only able to find linear decision boundaries, which is probably not sufficient to capture the true complexity of the relationships in the data. The best model used C=1 and kernel=rbf.





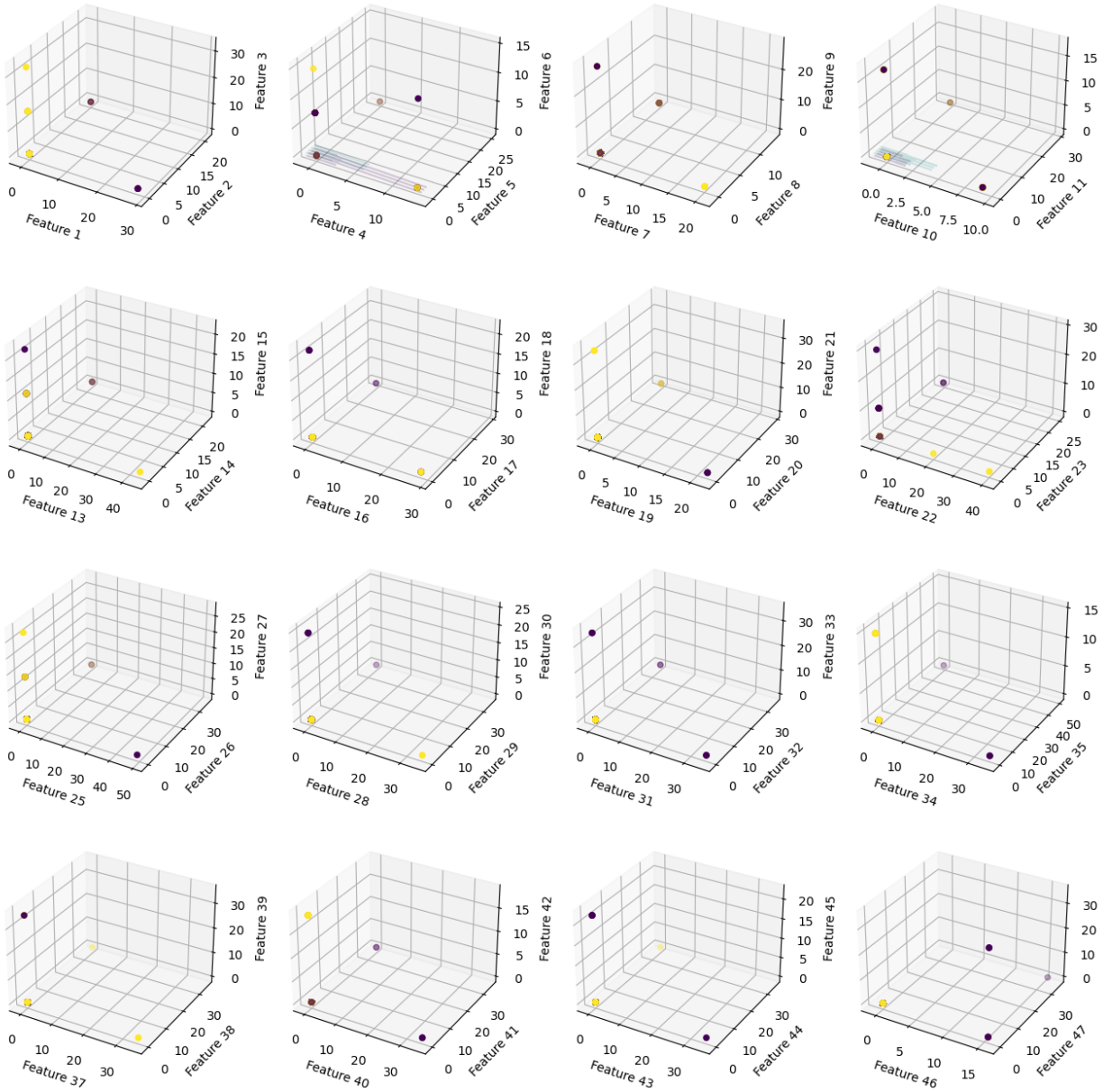
Accuracy: 0.896049896049896
Classification report:

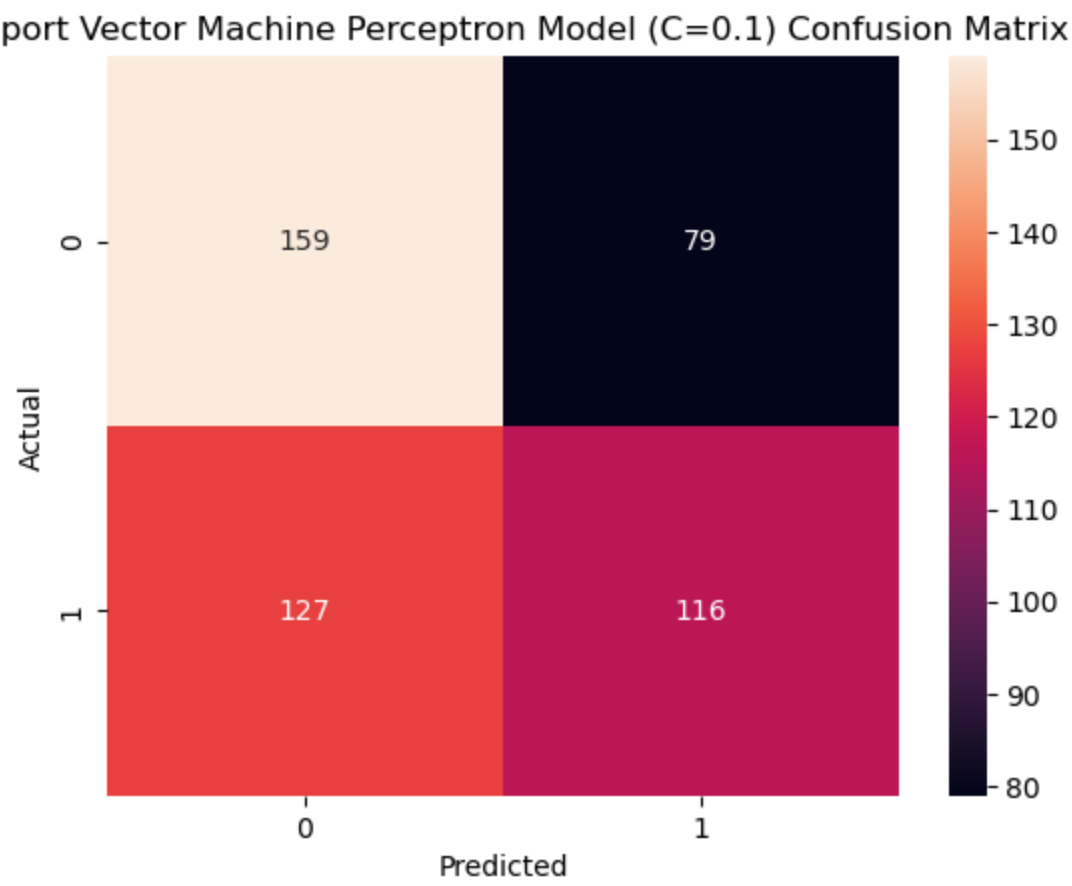
	precision	recall	f1-score	support
0	0.91	0.88	0.89	238
1	0.88	0.91	0.90	243
accuracy			0.90	481
macro avg	0.90	0.90	0.90	481
weighted avg	0.90	0.90	0.90	481

5-Fold Cross-Validation:
- Average accuracy: 0.8038
- Standard deviation: 0.0092

Drawing the decision boundary in a meaningful way proved to be difficult, since there were way too many features in the model to generate a single meaningful plot, so I instead provided some subplots showing some of the decision boundaries placed on points based on 3 features. Perhaps much more useful in this case are the confusion matrix, the classification report and the final cross-validation score, which we can see is the highest accuracy so far, with low deviation.

The most sensitive hyperparameter in this entire project that I encountered was the C parameter for the SVM model. I have included some output from a model that used C=0.1, and it is clear that this model is just completely missing the proper decision boundaries needed to produce an accurate model, as is clearly demonstrated by the following plots, reports and figures (using kernel=rbf and C=0.1). The accuracy of the C=0.1 model is completely tanking at 57%, down from 89.6%, which suggests that the C=0.1 model is underfitting the data. This makes sense, because C is a parameter that manages the trade-off between minimizing classification errors and regularization. If C is too small, the model will fail to capture the complexity of the underlying relationships of the data and produce an overly-simplistic decision boundary, resulting in poor generalization performance on both the training and test datasets. Below, I have included plots that show the result of such an underfit model. Notice the more-distinct lack of decision boundaries in the 3d subplots, and the horrible performance seen in the confusion matrix.





Accuracy: 0.5717255717255717
Classification report:

	precision	recall	f1-score	support
0	0.56	0.67	0.61	238
1	0.59	0.48	0.53	243
accuracy			0.57	481
macro avg	0.58	0.57	0.57	481
weighted avg	0.58	0.57	0.57	481

5-Fold Cross-Validation:
- Average accuracy: 0.6064
- Standard deviation: 0.0261

Part 5. Although all three of the kinds of ML models I experimented with on the sentiment analysis task were able to achieve comparable results (after tuning the hyperparameters), the SVM model did just slightly better than the MLP and Logistic Regression models did on the testing data. The slight discrepancies in the performance of the three models can be attributed to the inner mechanisms of how each model handles the learning process, as well as how they each handle regularization. During the experimentation phase of the project, MLPs model hyperparameters proved the hardest to tune, since the optimum number of hidden layers (and count of neurons in each layer) to include in a neural network can be very difficult to find and interpret. This is because SVMs and Logistic Regression are both convex optimization problems, whereas an MLP model is a non-convex optimization problem, meaning that an MLP model has to deal with local minima.

SVM's, especially ones that use a non-linear kernel like RBF, are highly capable of capturing complicated relationships in the data by implicitly mapping the input features to a higher-dimensional space, which allows the model to find support vectors to provide better decision boundaries, even in cases where the classes are not linearly separable in the original feature space. In general, Logistic Regression models may not perform as well as SVMs on problems with lots of non-linearity in the data, because Logistic Regression can only learn linear decision boundaries. In contrast, SVMs can learn more flexible non-linear decision boundaries that can better separate the classes, leading to performance.

There could be several reasons why the MLP is not performing quite as well as the SVM on the sentiment analysis task. It is certainly possible that the MLP model is overly complex, and is slightly overfitting the training data, which is leading to slightly worse generalization to the test dataset. I tried as best as I could to be thorough when I experimented with different hyperparameters, but I consistently found the final parameters I landed on to produce the best model, though it is probably impossible to say for sure. In contrast, the SVM model appears to achieve a better balance between model complexity and generalization than the MLP. That said, it is very interesting to observe that the MLP confusion matrix shows a much lower tendency to predict a false negative than the SVM model, suggesting that if we were to use ensemble techniques like bagging or boosting to combine the strengths of the MLP and SVM models, we could potentially achieve better performance on the sentiment analysis task.

Now that the best classifier on the testing set has been identified as the SVM model, I will do some analysis on some of the mistakes that the classifier makes. I have included a print out of 50 reviews, along with their true and predicted sentiments on the following page. From analyzing the outputs of the predictions on the reviews, it appears that there are reasonable scenarios that cause misclassifications to happen. The third review in the printout is a false negative predicted by the SVM, that has generally negative-sounding phrases in it like 'loneliness' and 'forces you' that may be to blame for the model's false prediction. The last review in the printout is a false positive that contains the word 'highly', which I highly suspect to have thrown off the model about the sentiment of this review. Lastly, we can see a review from amazon in the middle of the printout that reads 'the nano stated it my son was disapointed'. I believe due to the fact that the word 'disappointed' was misspelled in the review, the model most likely treated it as a completely different word, since I did not introduce any pre-processing that could handle typos like these. That could be a potential way to further improve the performance of the model.

source	review	truth	prediction
imdb	but i thought his acting was skilled	1	1
amazon	so i basically threw my money out the window for nothing	0	0
imdb	definitely worth seeing… it s the sort of thought provoking film that forces you to question your own threshold of loneliness	1	0
yelp	your servers suck wait correction our server heimer sucked	0	0
yelp	the meat was pretty dry i had the sliced brisket and pulled pork	0	0
yelp	dessert panna cotta was amazing	1	1
yelp	i didn t know pulled pork could be soooo delicious	1	1
yelp	at least min passed in between us ordering and the food arriving and it wasn t that busy	0	0
yelp	terrible management	0	0
yelp	my ribeye steak was cooked perfectly and had great mesquite flavor	1	1
amazon	not a good item it worked for a while then started having problems in my auto reverse tape player	0	0
amazon	best headset ever	1	1
yelp	favorite place in town for shawarrrrrrma	1	1
yelp	went in for happy hour great list of wines	1	1
yelp	i ll take my business dinner dollars elsewhere	0	0
yelp	food is way overpriced and portions are fucking small	0	0
yelp	i want to first say our server was great and we had perfect service	1	1

amazon	no ear loop needed it s tiny and the sound is great	1	1
imdb	i thoroughly enjoyed it when christopher eccleston took control of the tardis and the continuation of the series	1	1
imdb	john wayne did an incredible job for being so young in the movie industry	1	1
imdb	as they say in canada this is the fun game aye	1	1
imdb	the writer gorman bechard undoubtedly did his homework because all references are industry and character age appropriate	1	1
imdb	when achille and philippa beautifully sing a duet from don giovanni that perfectly describes their situation in the movie you appreciate the subtle layers of this excellent film	1	1
imdb	non linear narration thus many flashbacks and every part are articulated quite well	1	1
amazon	not worth it	0	0
yelp	the burger i got the gold standard a burger and was kind of disappointed	0	0
amazon	i connected my wife s bluetooth motorola hs to my phone and it worked like a charm whether the phone was in my pocket or the case	1	1
imdb	rating out of	0	0
amazon	it clicks into place in a way that makes you wonder how long that mechanism would last	0	0
amazon	how can that be the audio quality is poor	0	0
yelp	the cashier had no care what so ever on what i had to say it still ended up being wayyy overpriced	0	0
yelp	first time there and might just be the last	0	0
yelp	i was disgusted because i was pretty sure that was human hair	0	0

yelp	i can t tell you how disappointed i was	0	0
imdb	this one just fails to create any real suspense	0	0
imdb	overall i rate this movie a out of a scale	1	1
yelp	never been to hard rock casino before will never ever step forward in it again	0	0
yelp	the food is good	1	1
yelp	the steak and the shrimp are in my opinion the best entrees at gc	1	1
amazon	the nano stated it my son was dissapointed	0	1
yelp	i came back today since they relocated and still not impressed	0	0
yelp	the owners are super friendly and the staff is courteous	1	1
amazon	performed awful muffled tinny incoming sound and severe echo for those on the other end of the call	0	0
imdb	everything from acting to cinematography was solid	1	1
amazon	worst ever	0	0
yelp	the portion was huge	1	1
imdb	it is a true classic	1	1
yelp	the ripped banana was not only ripped but petrified and tasteless	0	0
imdb	all in all a great disappointment	0	0
amazon	the battery life is highly unacceptable	0	1

Part 6. After applying the best classifier from the previous steps to the text data in x_test.csv, I found that the performance shown on the test data set aligned with the performance that was observed during the cross-validation step of each model. The SVM did better than the other two models, but only slightly, which really speaks to the importance of doing the proper data pre-processing and hyperparameter tuning of the models when applying Machine Learning to a problem like sentiment analysis. This is a good thing, since seeing not much difference between performance in training/cross-validation and final testing suggests that the final model is not under or over fit, and that it is generalizing well to unseen data. If I were to see amazing performance during training and cross-validation, but horrible performance during testing, that would have suggested that the model has over fit to the training data and does not generalize well. My final testing scores with the SVM model were as follows:

Error Rate: 0.20167 AUROC: 0.79833

However, once I tried implementing an ensemble model that used Logistic Regression to pick the best predictions of the three inner models, I found that the boosting strategy provided a decent improvement to my final results. I also experimented with a fourth model (a random forest) to see if that would help, and found the improvements to be minor, but enough to be significant:

Error Rate: 0.195 AUROC: 0.805