CS-UY 4563: Machine Learning: Final Project Written Report

IMDB Score Classification of Movies and TV Shows

Section A

April 26, 2023

Professor Linda N. Sellie

Ariel Hsieh, Kevin Ng

Table of Contents

Introduction

The purpose of this project was to utilize various machine learning models to ultimately predict whether or not a Netflix movie/tv show was good based upon different features. To categorize whether a movie/tv show was good, the IMDB score system was used, and a score of 8 or higher meant that the movie/tv show was good, while a score lower than 8 would mean that the movie/tv show was not good. A Kaggle dataset updated in July 2022 of more than 5000 data points was used for this project.

The machine learning models that were used in this project were: logistic regression, support vector machine, and neural network. The models were instructed to classify movies into two classes: IMDB score >= 8 and IMDB score < 8. All the models were run multiple times using different parameters, regularizations, and normalizations.

Preprocessing

Preprocessing is important because it improves performance and handles dirty data. Our Netflix dataset preprocessing goes though the following steps: data cleanup, encoding, balancing, splitting, and finally normalization. First, the data was cleaned up by removing any rows with NaN values. The genres and production_country columns of the dataset are then one-hot-encoded by creating new data frames with binary values for each genre and country respectively. These dataframes are merged into the dataset and the original columns genres and production_country are dropped. Ordinal values for age certification and media type are encoded using the OrdinalEncoder() function. Binary values for imdb score based on whether it's greater than 8 or not are also encoded (1 for greater than 8, 0 for less than 8) so that we can perform binary classification across all 3 models. Then, the dataset was balanced by dropping rows of data until the number of rows with imdb_score = 0 matched imdb_score = 1. This ensures that our model is not biased to classify one class over the other. The dataset is then split into training, validation, and testing sets using train_test_split() function from scikit-learn library. The training set is 72% of the original dataset, validation set is 18%, and testing set is 10% by convention. The function returns four arrays: X_train, X_test, y_train, y_test. The first two arrays contain the features for training and testing sets respectively, while the last two arrays contain the target variable for training and testing sets respectively. Finally, X_train, X_test were normalized using StandardScaler() from scikit-learn library.

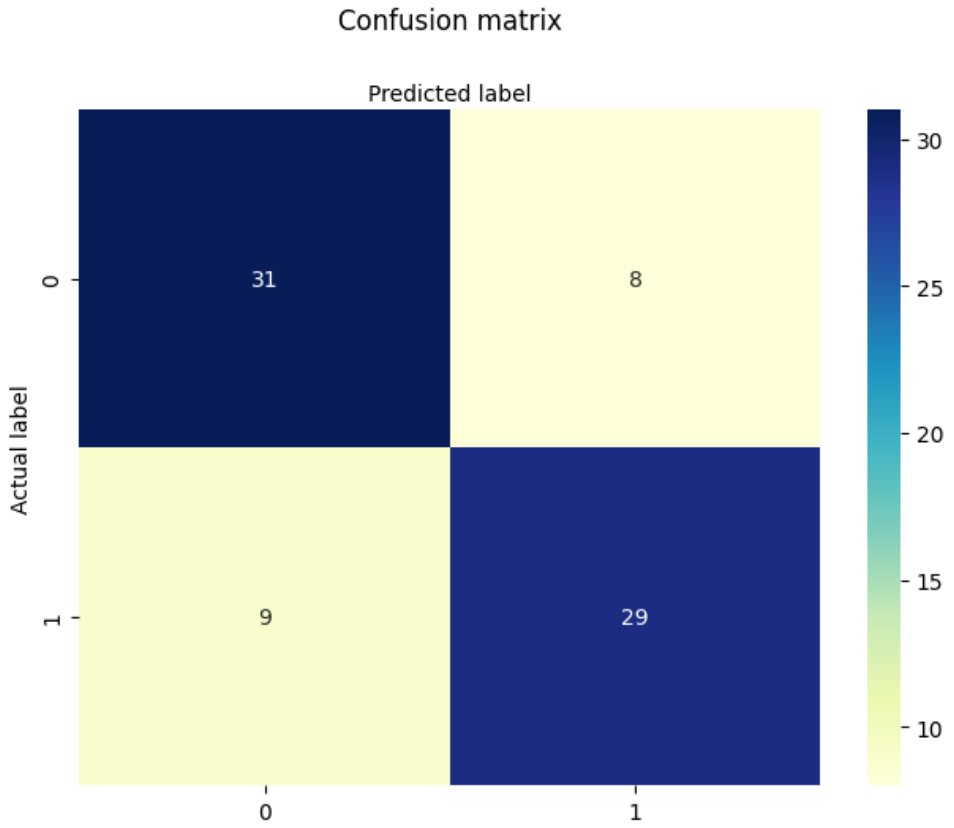
Supervised Analysis Results

Logistic Regression

We used the logistic regression model from the sklearn model library. The first trial was done with no feature transformations and no regularization penalty, which yielded a validation accuracy of 0.72 and a test accuracy of 0.78. The Classification report and confusion matrix are shown below:

	precision	recall	f1-score	support
imdb score < 8	0.78	0.79	0.78	39
imdb score >= 8	0.78	0.76	0.77	38
accuracy			0.78	77
macro avg	0.78	0.78	0.78	77
weighted avg	0.78	0.78	0.78	77

Logistic Regression: Classification Metric Table (No Transformations, No Regularization)

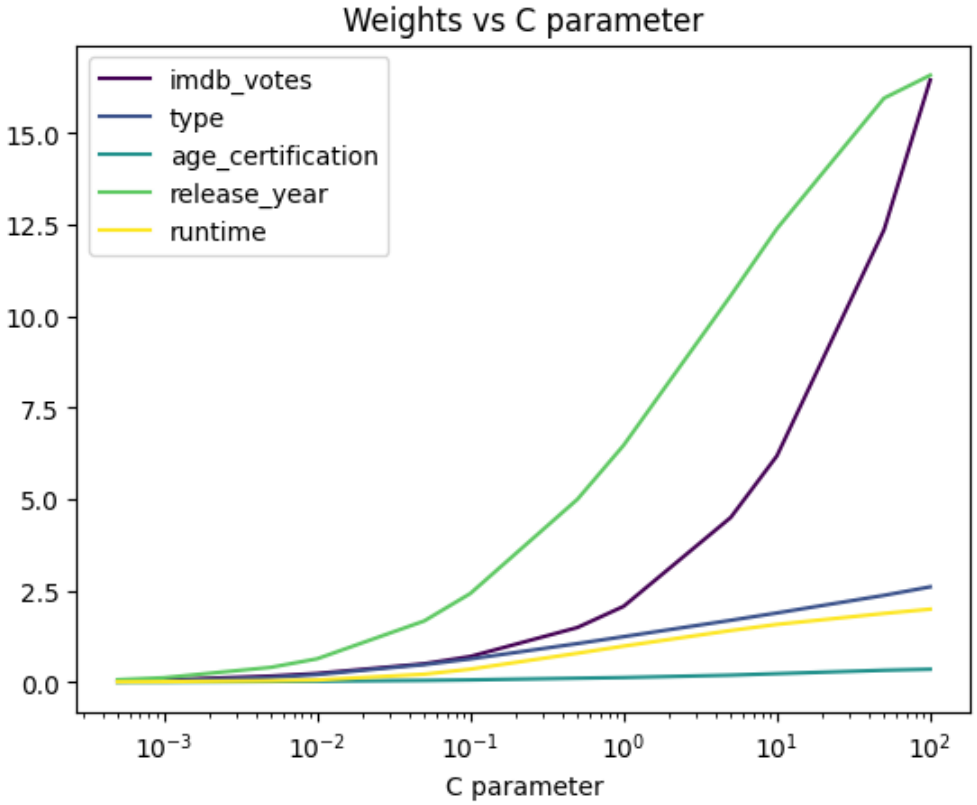


Logistic Regression: Confusion Matrix (No Transformation, No Regularization)

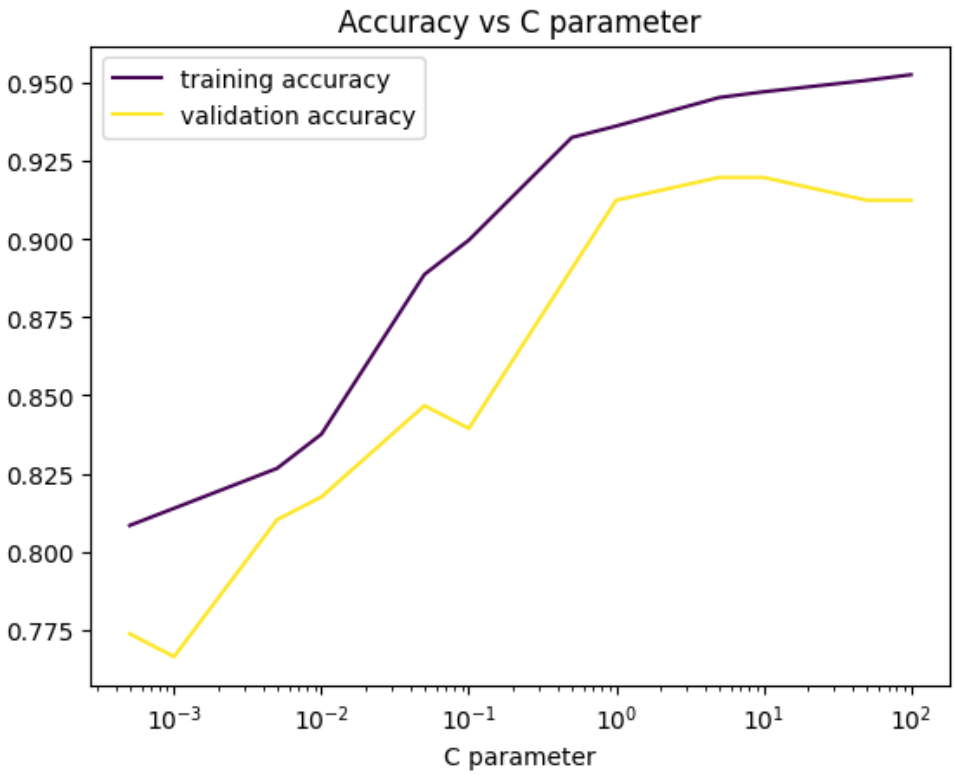
Regularization

The model was then trained with L2 regularization with 2 cases: model with no feature transformations and model with polynomial transformed to the second degree. The regularization constant parameters that were used were C = [0.0005,0.001,0.005,0.01,0.05,0.1,0.5,1,5,10,50,100]. The regularization parameter that yielded the highest training accuracy was then used on the test dataset to observe how the feature weights and accuracy changed with changes in the regularization parameter constant (AKA "C parameter").

With no feature transformation, the regularization parameters were all tested, and the following graphs for feature weights and accuracy were yielded:



Logistic Regression: Feature Weights vs Regularization Parameter (No Transformation)

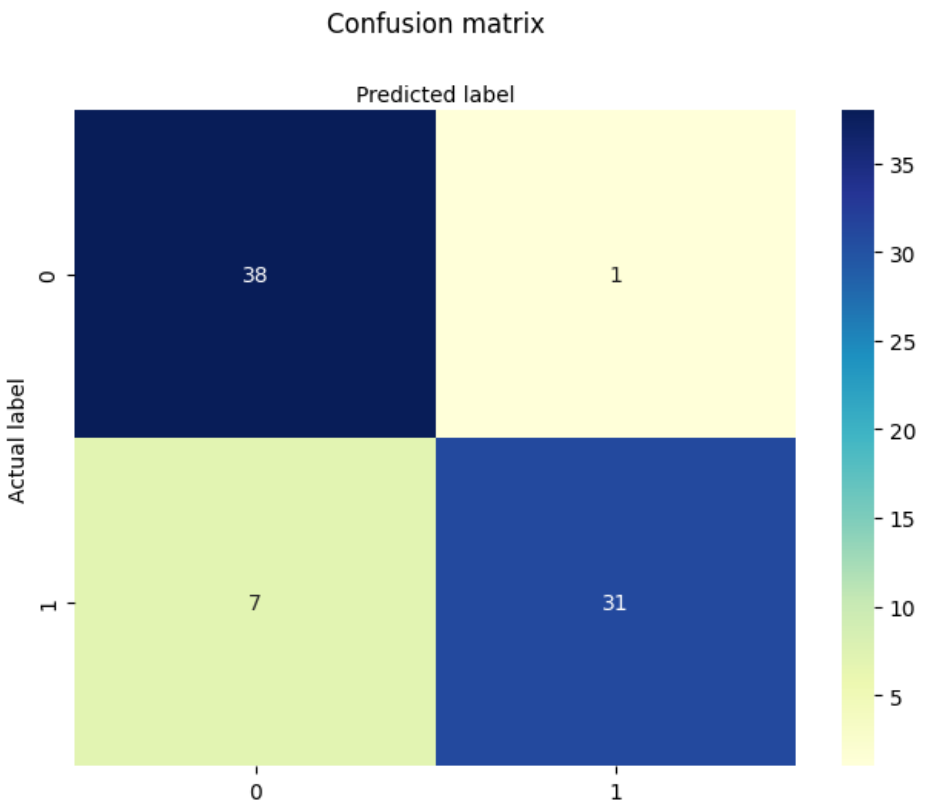


Logistic Regression: Accuracy vs Regularization Parameter (No Transformation)

The best C value was found to be 5, with a validation accuracy of 0.92. Using that model on the test dataset, the following classification metrics and confusion matrix were yielded:

	precision	recall	f1-score	support
imdb score < 8	0.84	0.97	0.90	39
imdb score >= 8	0.97	0.82	0.89	38
accuracy			0.90	77
macro avg	0.91	0.90	0.90	77
weighted avg	0.91	0.90	0.90	77

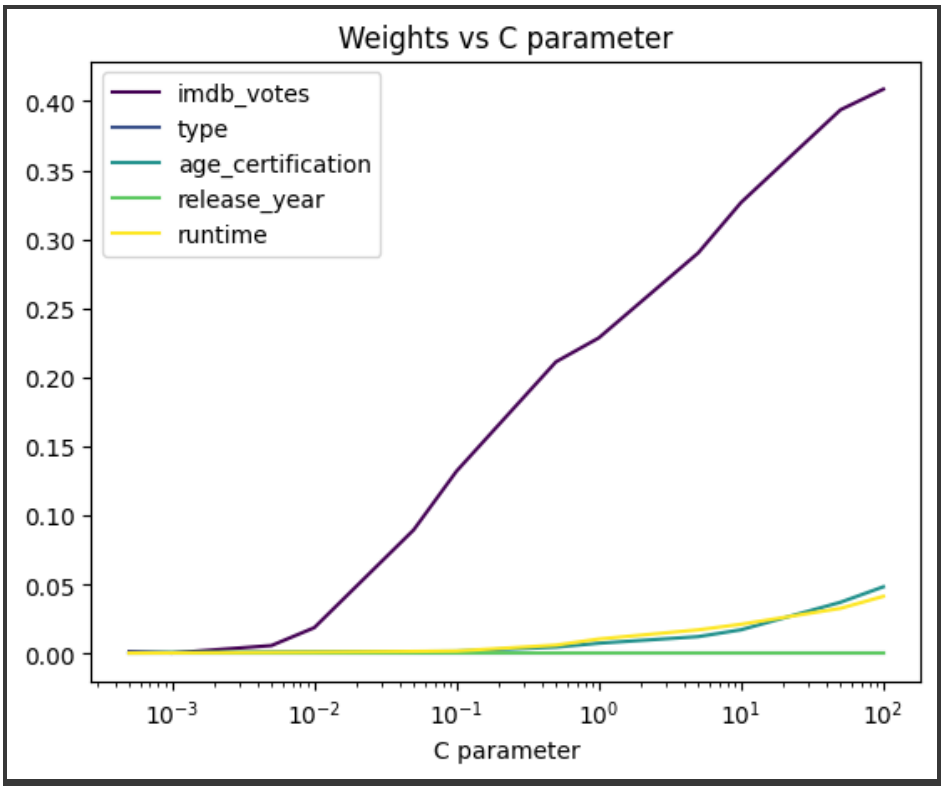
Logistic Regression: Classification Metric Table (No Transformations, C = 5)



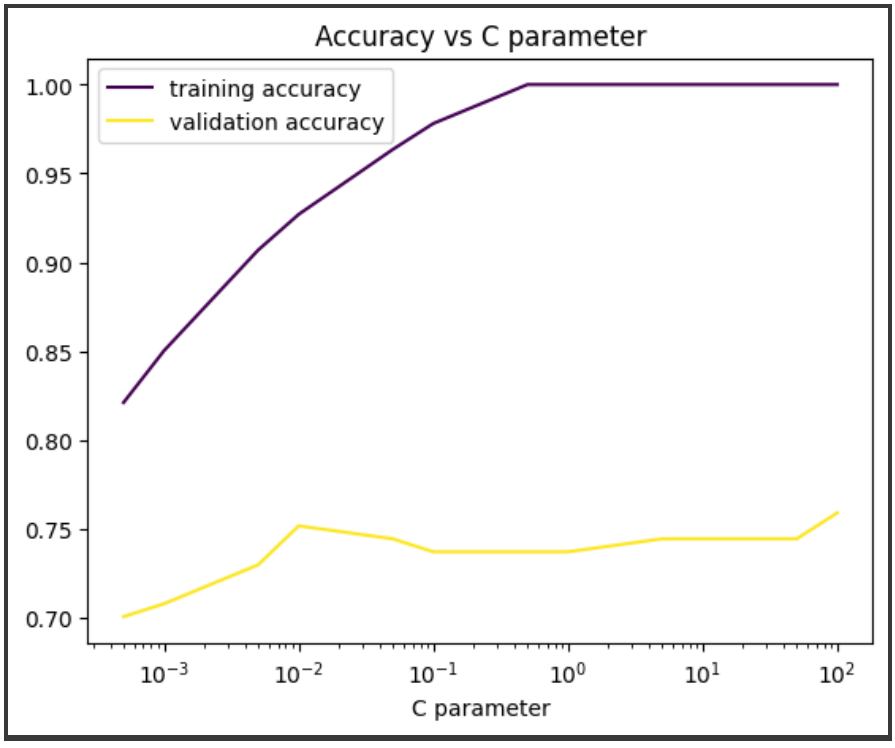
Logistic Regression: Confusion Matrix (No Transformation, C = 5)

Polynomial Transformation

The dataset was then put through a polynomial transformation of degree 2, where the following graphs were yielded:



Logistic Regression: Feature Weights vs Regularization Parameter (X^2 Transformation)

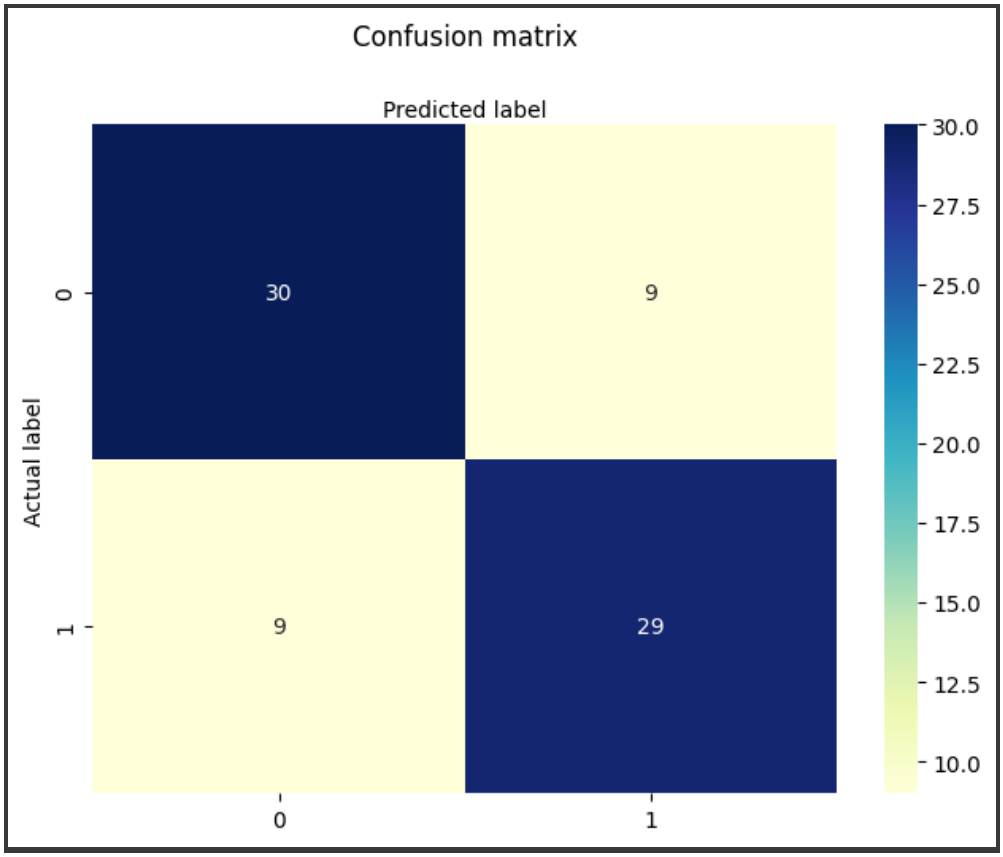


Logistic Regression: Accuracy vs Regularization Parameter (X^2 Transformation)

The best C value was found to be 100, with a validation accuracy of 0.76. Using that model on the test dataset, the following classification metrics and confusion matrix were yielded:

	precision	recall	f1-score	support
imdb score < 8	0.77	0.77	0.77	39
imdb score >= 8	0.76	0.76	0.76	38
accuracy			0.77	77
macro avg	0.77	0.77	0.77	77
weighted avg	0.77	0.77	0.77	77

Logistic Regression: Classification Metric Table (X^2 Transformations, C = 100)



Logistic Regression: Confusion Matrix (X^2 Transformation, C = 100)

Support Vector Machine

We used the SVM model from the sklearn model library. We decided to use three different types of kernels: linear, polynomial, and rbf. With C = 1 regularization applied at first, as that is the default for SVM models, the three kernel functions yielded different accuracies. The validation accuracy with the linear kernel was 0.91, the validation accuracy with the polynomial kernel was 0.66, and the validation accuracy with the RBF kernel was 0.79. Those models were then applied to the test dataset, and the test dataset classification metrics and confusion matrices are shown below:

	precision	recall	f1-score	support
imdb score < 8	0.81	1.00	0.90	39
imdb score >= 8	1.00	0.76	0.87	38
accuracy			0.88	77
macro avg	0.91	0.88	0.88	77
weighted avg	0.91	0.88	0.88	77

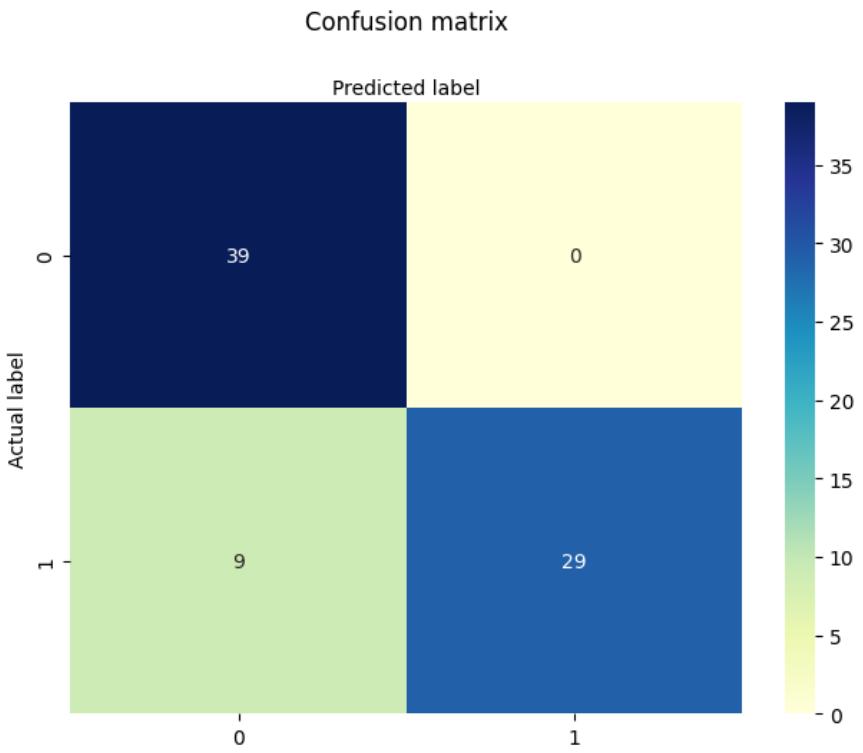
SVM: Classification Metric Table (Linear Kernel, No Regularization)

	precision	recall	f1-score	support
imdb score < 8	0.72	0.46	0.56	39
imdb score >= 8	0.60	0.82	0.69	38
accuracy			0.64	77
macro avg	0.66	0.64	0.63	77
weighted avg	0.66	0.64	0.62	77

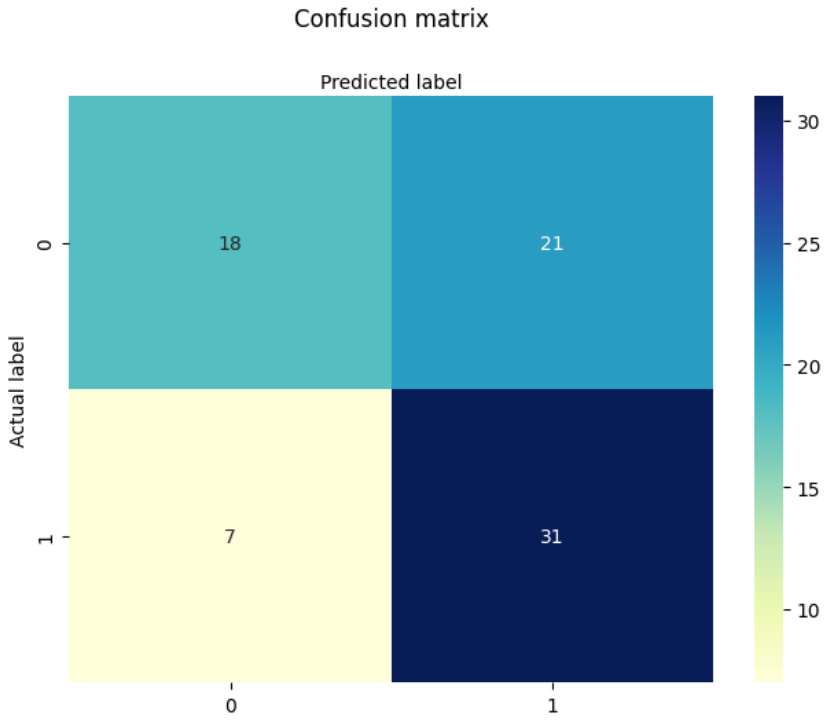
SVM: Classification Metric Table (Polynomial Degree 3 Kernel, No Regularization)

	precision	recall	f1-score	support
imdb score < 8	0.73	0.92	0.82	39
imdb score >= 8	0.89	0.66	0.76	38
accuracy			0.79	77
macro avg	0.81	0.79	0.79	77
weighted avg	0.81	0.79	0.79	77

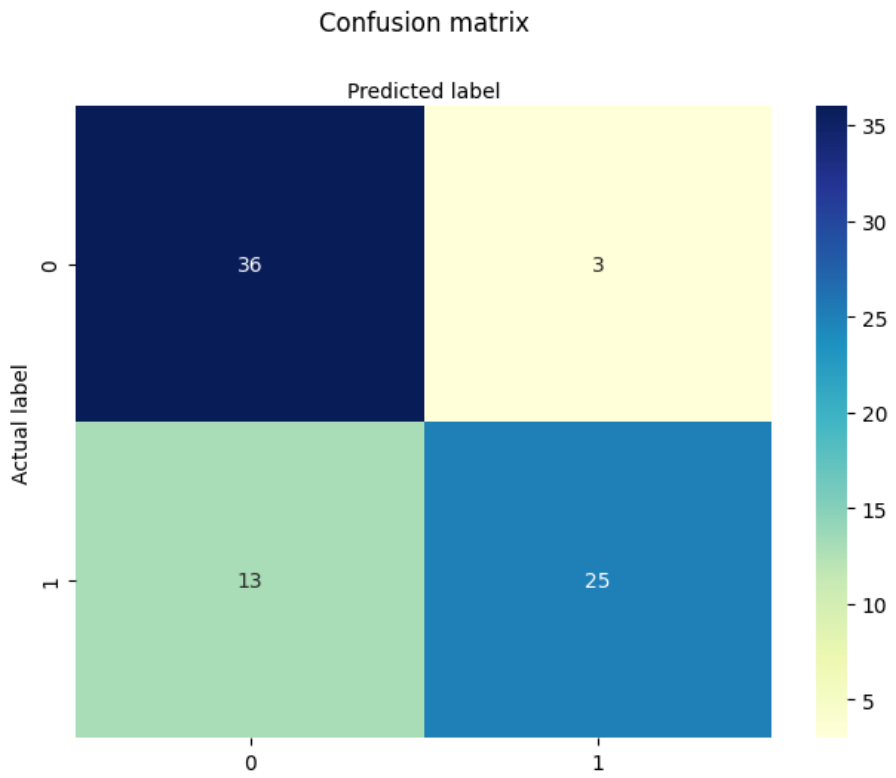
SVM: Classification Metric Table (RBF Kernel, No Regularization)



SVM: Confusion Matrix (Linear Kernel, No Regularization)



SVM: Confusion Matrix (Polynomial Degree 3 Kernel, No Regularization)

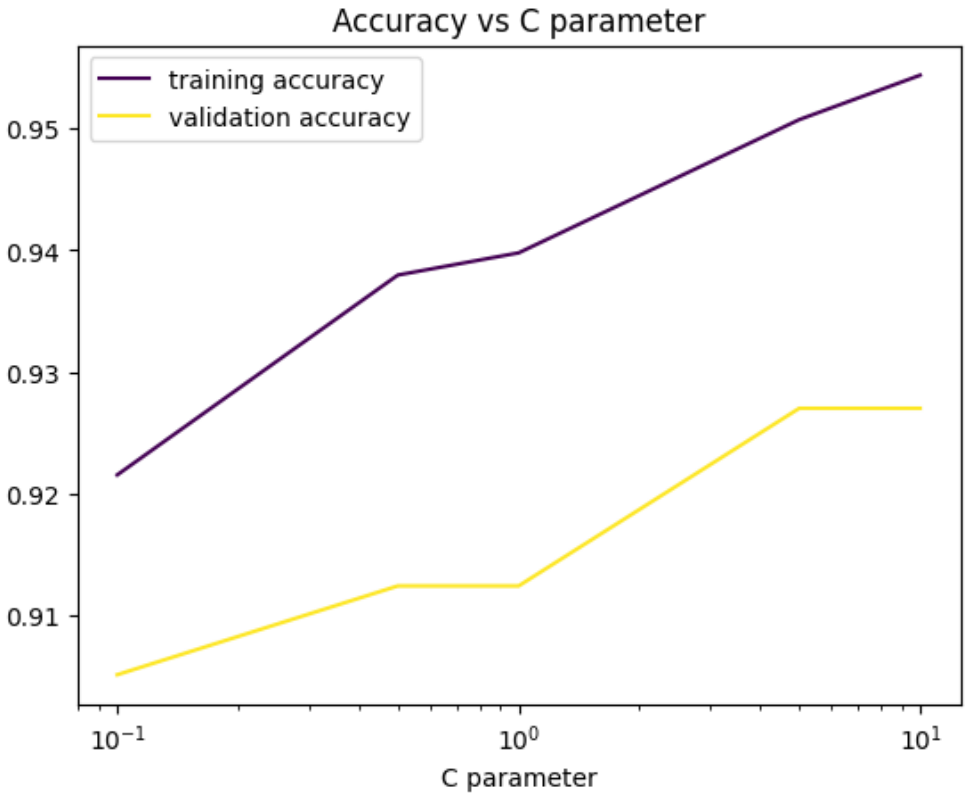


SVM: Confusion Matrix (RBF Kernel, No Regularization)

Regularization

Linear Kernel

Regularization constant parameters were then applied to the model with the three different kernel functions. First, the regularization parameters, C = [0.1,0.5,1.0,5.,10.0], were used with the linear kernel function of the model. The accuracy vs regularization parameter graph is shown below:

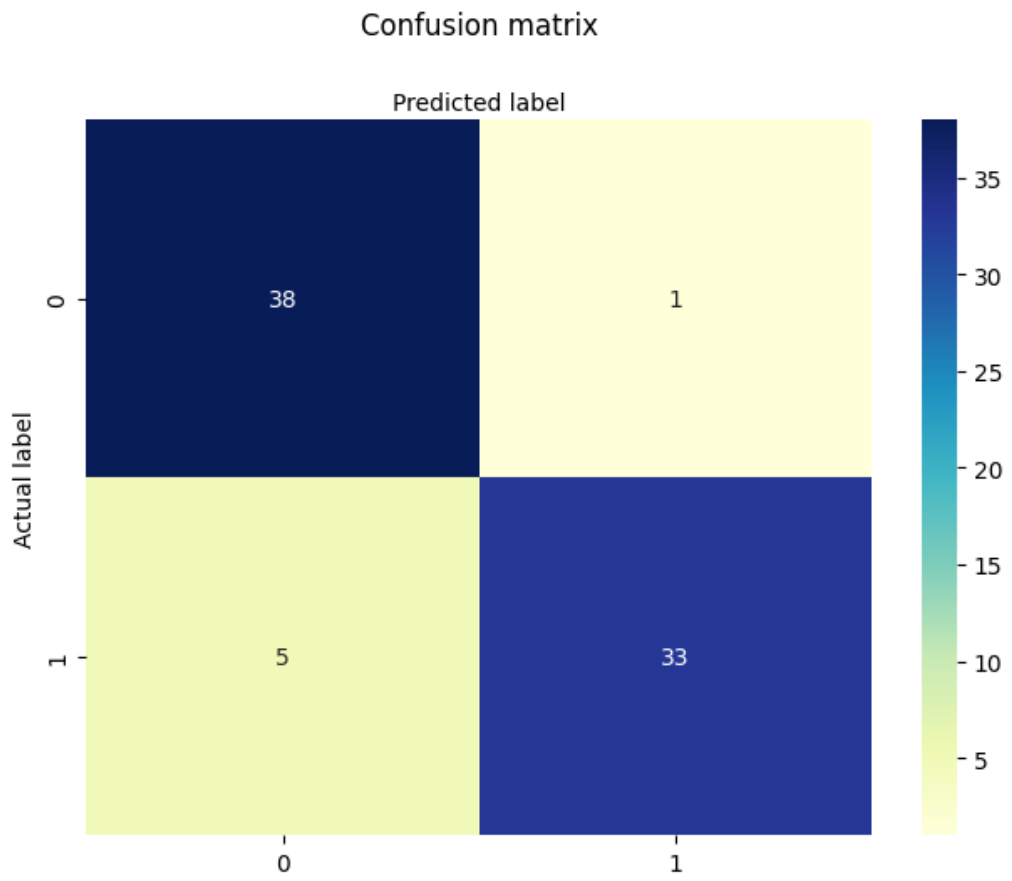


SVM: Accuracy vs Regularization Parameter (Linear Kernel)

The best regularization parameter, which was a value of C = 5.0, yielded a validation accuracy of 0.93, which was higher than the model with no regularization. This model was then used on the test dataset to obtain the following classification metrics and confusion matrix:

	precision	recall	f1-score	support
imdb score < 8	0.88	0.97	0.93	39
imdb score >= 8	0.97	0.87	0.92	38
accuracy			0.92	77
macro avg	0.93	0.92	0.92	77
weighted avg	0.93	0.92	0.92	77

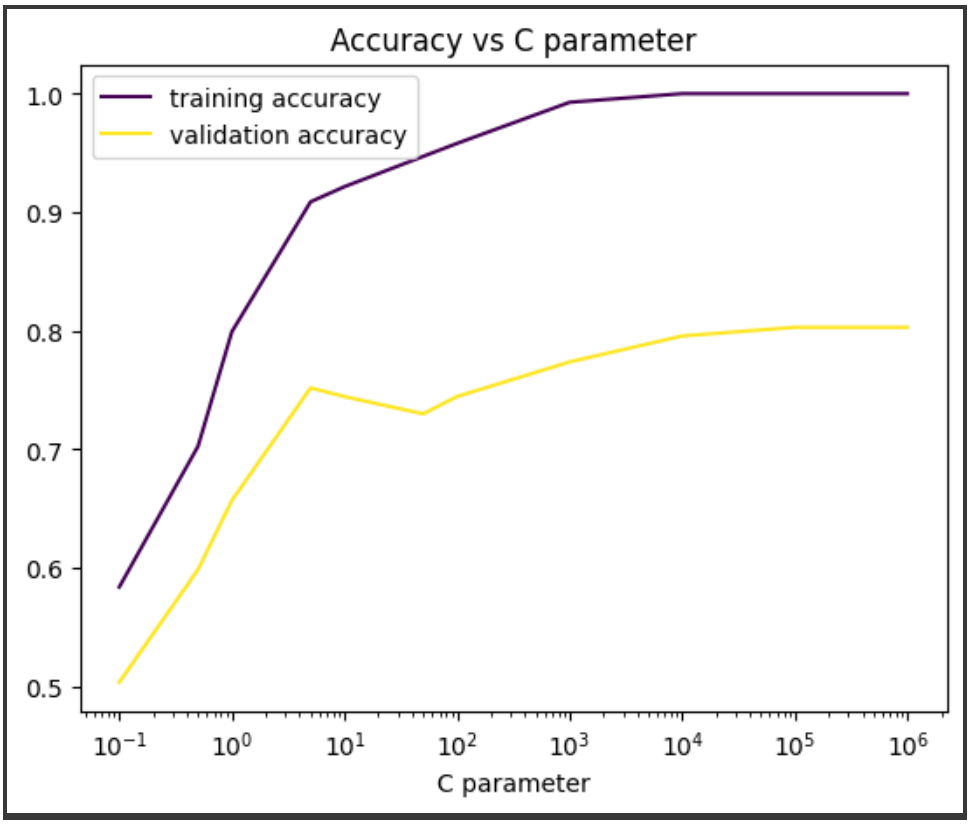
SVM: Classification Metric Table (Linear Kernel, C = 5)



SVM: Confusion Matrix (Linear Kernel, C = 5)

Polynomial Kernel

The regularization parameters were then used with the RBF kernel function. The regularization parameters used for this kernel function model was: C = [0.1,0.5,1.0,5.,10.0,50.,100.,1000.,10000.,100000.,1000000.]. The accuracy vs regularization parameter graph is shown below:

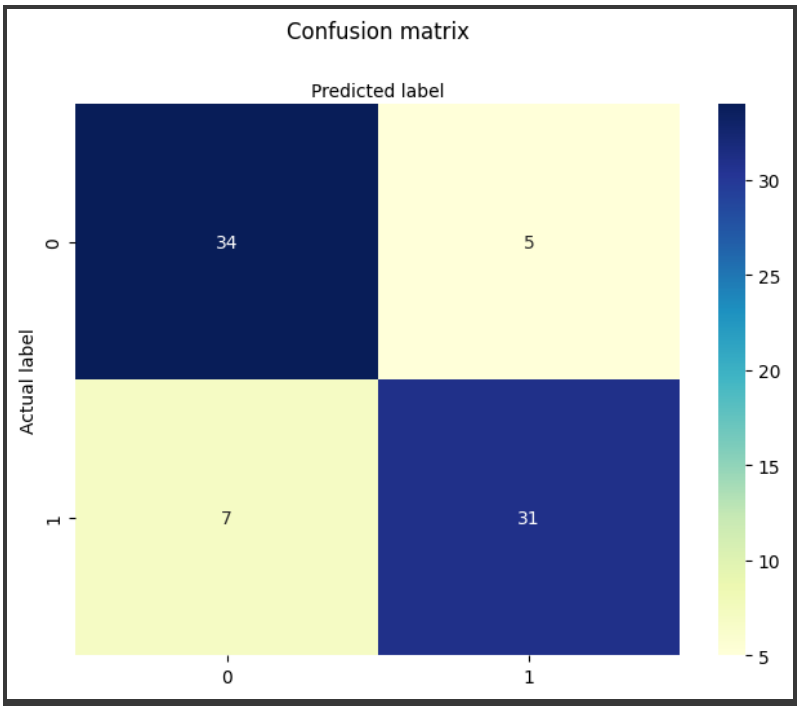


SVM: Accuracy vs Regularization Parameter (Polynomial Kernel)

The best regularization parameter, which was a value of C = 100000, yielded a validation accuracy of 0.80, which was higher than the model with no regularization. This model was then used on the test dataset to obtain the following classification metrics and confusion matrix:

	precision	recall	f1-score	support
imdb score < 8	0.83	0.87	0.85	39
imdb score >= 8	0.86	0.82	0.84	38
accuracy			0.84	77
macro avg	0.85	0.84	0.84	77
weighted avg	0.84	0.84	0.84	77

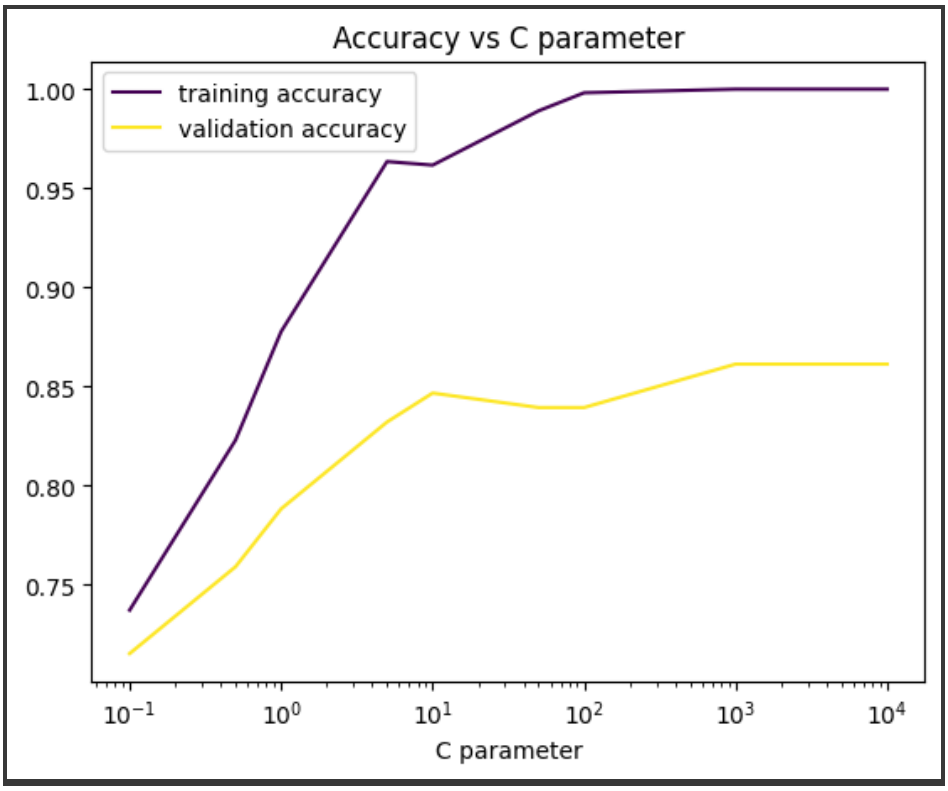
SVM: Classification Metric Table (Polynomial Kernel, C = 100000)



SVM: Confusion Matrix (Polynomial Kernel, C = 100000)

RBF Kernel

The regularization parameters were then used with the RBF kernel function. The regularization parameters used for this kernel function model was: C = [0.1,0.5,1.0,5.,10.0,50.,100.,500.,1000.,10000.] The accuracy vs regularization parameter graph is shown below:

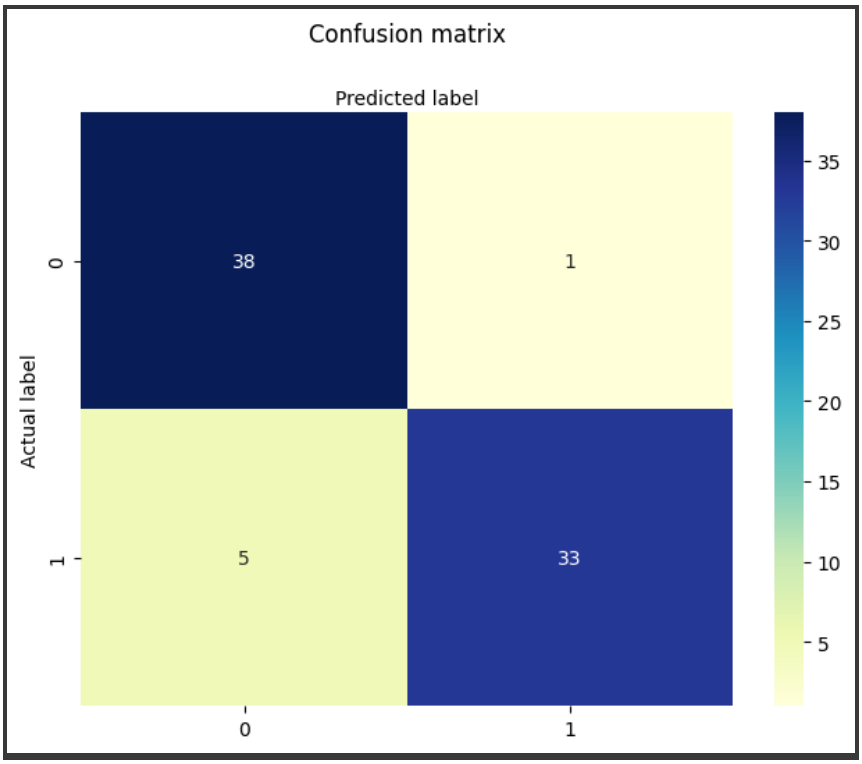


SVM: Accuracy vs Regularization Parameter (RBF Kernel)

The accuracy of the model seemed to increase as the regularization parameter was increased. The best regularization parameter, which was a value of C = 1000, yielded a validation accuracy of 0.86, which was higher than the model with no regularization. This model was then used on the test dataset to obtain the following classification metrics and confusion matrix:

	precision	recall	f1-score	support
imdb score < 8	0.88	0.97	0.93	39
imdb score >= 8	0.97	0.87	0.92	38
accuracy			0.92	77
macro avg	0.93	0.92	0.92	77
weighted avg	0.93	0.92	0.92	77

SVM: Classification Metric Table (RBF Kernel, C = 1000)

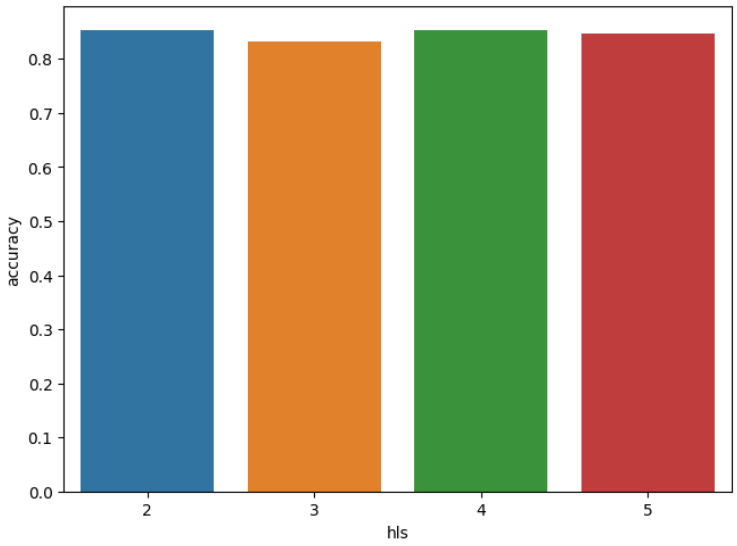


SVM: Confusion Matrix (RBF Kernel, C = 1000)

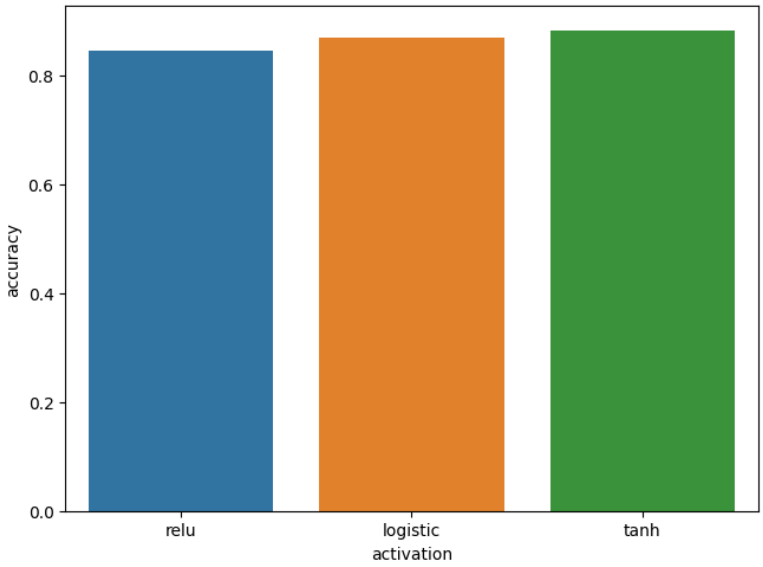
Neural Network

For the neural network (NN) binary classification problem, the MLPClassifier function from the sklearn.neural_network module was used. The hyperparameters used for this project are number of hidden layers: [2,3,4,5] and type of activation function: [‘relu’, ‘sigmoid’, ‘tanh’]. A custom NN function generates, trains, and evaluates the twelve different combinations of NN models. In the function, each model is trained on the X_train data. The models are then used to predict the target variable by calling the .predict() method on the training, validation, and testing datasets.

Finally, the function generates a heat map of confusion matrix and classification report for each model using the heat_map() and classification_report() function. To analyze the relationship between the hyperparamters: # of hidden layers, type of activation function versus the testing accuracy, the following plots were generated using the seaborn library and accuracy_score() function from the sklearn.metrics library. The accuracy score is generated using the predictions made by the predict() method and the actual targets from the test and training data set.



Neural Networks: Accuracy vs # of hidden layers
(2: (100,), 3: (100, 75,), 4: (100, 75, 50,), 5: (100, 75, 50, 25))

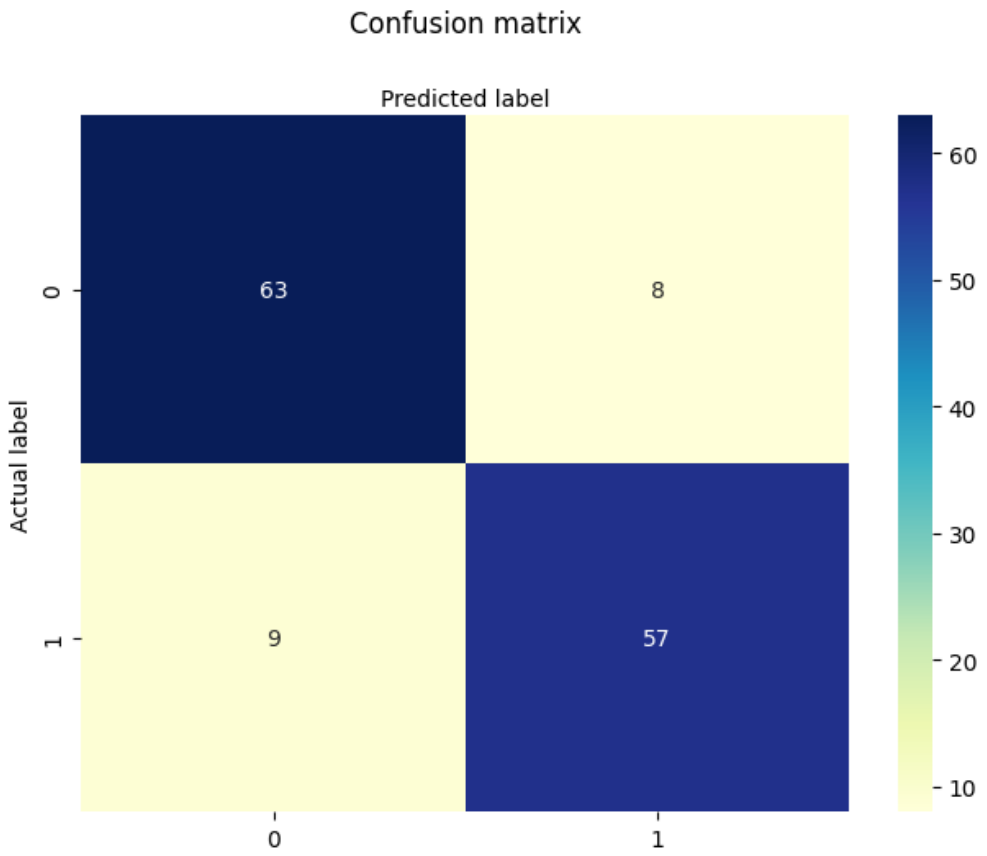


Neural Networks: Accuracy vs # of activation function

Therefore, the classification report and heat map displayed shows the model with the highest testing accuracy (due to the small size of the dataset as result of balancing, the training accuracy is 100% across all model configurations).

	precision	recall	f1-score	support
imdb score < 8	0.88	0.90	0.89	71
imdb score >= 8	0.89	0.86	0.88	66
accuracy			0.88	137
macro avg	0.88	0.88	0.88	137
weighted avg	0.88	0.88	0.88	137

Neural Networks: Classification Report (Tanh Activation Function, 2 hidden layers (100,))



Neural Networks: Heat Map (Tanh Activation Function, 2 hidden layers (100,))

Conclusion

Our objective for this project is to perform and analyze three models: logistic regression, SVMs, and neural networks on a Netflix IMDB score dataset. We chose the best configuration for each model based on the validation accuracy and compared the accuracies across each model.

It seemed that for all logistic regression models, as regularization strength decreased (hyperparameter 1/C), the training accuracy of the model increased, which shows that with less regularization, the model is able to fit the data a bit more closely. When there was no feature transformation, the validation accuracy increased as regularization strength decreased, but plateaued once the C parameter hit around 0.5 and decreased after 100, which shows that the model fit the validation dataset well. When there was a polynomial feature transformation on the training dataset, the validation accuracy did not converge with the training accuracy when the regularization changed, which suggests that when the features were polynomially transformed, the model was overfitting the training dataset. Some of the feature weights of the model increased when the regularization strength decreased, some more drastically than others, which made sense because stronger regularization is supposed to prevent weights from being weighed too heavily to prevent overfitting, so when the regularization strength decreased, the feature weights increased. Without regularization or feature transformation, the validation accuracy was 0.72 and the test accuracy was 0.78. Overall, the logistic regression model that had the highest validation accuracy of 0.92 was the one with no feature transformations and a regularization C-value of 5, which then yielded a test accuracy of 0.90, which suggests that the regularization applied to the model helped improve the accuracy of the model and decreased variance slightly.

With all the SVM models, as regularization strength decreased (hyperparameter 1/C), the training accuracy of the model increased, which shows that with less regularization, the model is able to fit the data a bit more closely. With the linear kernel function, as the regularization strength decreased, the validation accuracy increased around the same magnitude as the training accuracy, so there may have been some bias with the linear kernel function model. This bias may also be exhibited in the test dataset, so the test accuracy still remained relatively high. With the polynomial kernel function, as the regularization strength decreased, the validation accuracy difference from the training accuracy increased, which shows that a higher regularization strength was keeping the model from overfitting to the training dataset, even though there is still bias. The test accuracy of this model was higher than the validation accuracy, but that may be

due to variance, because we were using an overfitting model. The RBF kernel function model exhibited similar patterns to the polynomial kernel function in that the model started overfitting as the regularization strength decreased, as seen in how the training accuracy and validation accuracy diverged. This also led to this model having high variance, and explains why the validation accuracy and test accuracy had different values. With default regularization of C = 1.0, the validation accuracy with the linear kernel was 0.91, the validation accuracy with the polynomial kernel was 0.66, and the validation accuracy with the RBF kernel was 0.79. The test accuracy with the linear kernel was 0.88, the test accuracy with the polynomial kernel was 0.64, and the test accuracy of the RBF kernel was 0.79. When regularization was applied, it increased all the accuracies of the models. However, there was higher variance because the regularization strength was decreased. Overall, the most validation-accurate model was the SVM linear kernel function model, with C = 5.0 and a validation accuracy of 0.93, which yielded a test accuracy of 0.92. This was likely due to how the model was able to fit slightly better to the training dataset than the default C = 1, but not overfit by too much as the variance was low.

Finally, for the neural networks (NN) model, we observed that the highest validation accuracy occurred for the following neural network configuration: 2 hidden layers, tanh activation function, with a validation accuracy of 89%. It is also observed that the training accuracy for each configuration is 100%. This is most likely due to overfitting as the NN model overfits the training data resulting in a high training accuracy but lower validation and test accuracy. However, the model has performed adequately well when trying to predict on the test set with a test accuracy of 88%. Thus, it can be reasonably concluded that the tanh activation function and 2 hidden layers is the best neural network configuration for the objective.

In conclusion, the SVM model with a linear kernel function and a C parameter of 5 performed the best amongst the three models with the highest test accuracy of 92%. To improve model performance, we could increase our dataset, since our dataset was scaled down significantly in an effort to balance the dataset. We would want to look for data that falls within the imdb >= 8 score so that the minority class could be increased. We could also potentially increase our dataset beyond Netflix content. We could also try other types of regularization to see if the validation accuracy could get better and in turn the test accuracy as well. To make more predictions on this dataset, we could also do multiclass classification on the dataset to make more complex models and yield more predictions on data.

Works Cited

Dataset:

https://www.kaggle.com/datasets/victorsoeiro/netflix-tv-shows-and-movies

Referenced Logistic Regression Resource:
https://www.datacamp.com/tutorial/understanding-logistic-regression-python

Referenced SVM Resource:
https://github.com/christianversloot/machine-learning-articles/blob/main/creating-a-simple-binary-svm-classifier-with-python-and-scikit-learn.md

Referenced Neural Network Resource:
https://www.pluralsight.com/guides/machine-learning-neural-networks-scikit-learn

Sklearn Logistic Regression Documentation:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Sklearn SVM Documentation:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

Sklearn Neural Network Documentation:
https://scikit-learn.org/stable/modules/neural_networks_supervised.html