CS-UY 4563: Machine Learning:

Final Project Written Report

Video Game Success Prediction

11:00 AM & 2:00 PM Section

April 26, 2023

Professor Linda N. Sellie

Andy Quintuna, Yujun Lu

Introduction

This final project focuses on using machine learning models to predict the rating of video games and classify if they are successful or not. The dataset was taken from kaggle and contained 10 features: indexing, game title, maximum score, game leaderboard ratio, amount of gamers, percentage of gamers who 100% completed the game, the rating, url to the games page, minimum completion time, and maximum completion time.

The project starts off as a regression problem but becomes more of a classification one, with the goal to predict what rating a game may receive as an alternative or verification to game journalists or reviewers, completely free of bias. We experimented with three different classification models: Linear Regression, Support Vector Machine (SVM), and Neural Networks.

Data Preparation

Expanding the Dataset

The data set did not contain any null values but two of the features, indexing and url, wouldn't provide the models any useful information so we decided to ignore them and to add in our own data. The four new additions being: the price of the game, if the game was classified as a AAA, the file size of the base foregoing dlc, and platforms it was available on. These were collected from the url website the dataset provided where the other features were recorded, with the platforms they were available on was taken by searching how to purchase the games from their official stores, websites, or other records online. Bringing the number of features to 11 with rating becoming the target value. These are traits we believed would have an effect on the popularity of a game and therefore its rating. Although the purpose for the project is rate games in general, this particular dataset focused on PS4 games so our project is just slightly skewed towards this category of games.

One hot encoding

Originally we wanted the production value of each game as we believed there to be a correlation to how successful a game can be to the amount of money put into it, but this information wasn't readily available for every game on the data set so instead we opted to include what companies created and published the games. However, with the vast amount of different publishers this would have resulted in a large-scale categorical encoding issue, so instead we chose to incorporate a one hot encoding where 0 meant a game wasn't made or distributed by a major publisher, and 1 meaning it was.

Numeric Transformation

The title of the game needed to be transformed from a string into an integer and we decided to represent it by its length to show if longer or shorter titles contributed to a game's rating. The line of thinking for this decision was factoring in that popular games tend to have a shorter, catcher, or memorable name.

Data Scaling

The data was then scaled using StandardScaler() imported from Pandas. All the models utilized standardization with some testing of normalization to observe differences, using MinMaxScaler() from Pandas.

Data Splitting

The dataset included 1581 games but because we needed to add additional values we decided to pick the first 125 games to append with the new data. The training set was made up of 100 games of 80% and the validation/test set were 25 games or 20%.

Models

Linear regression

The first model for the prediction of game ratings was linear regression. Methods to find the optimal version of the model involved using standard linear regression, polynomial transformation of the features with no regularization, and then with ridge regularization.

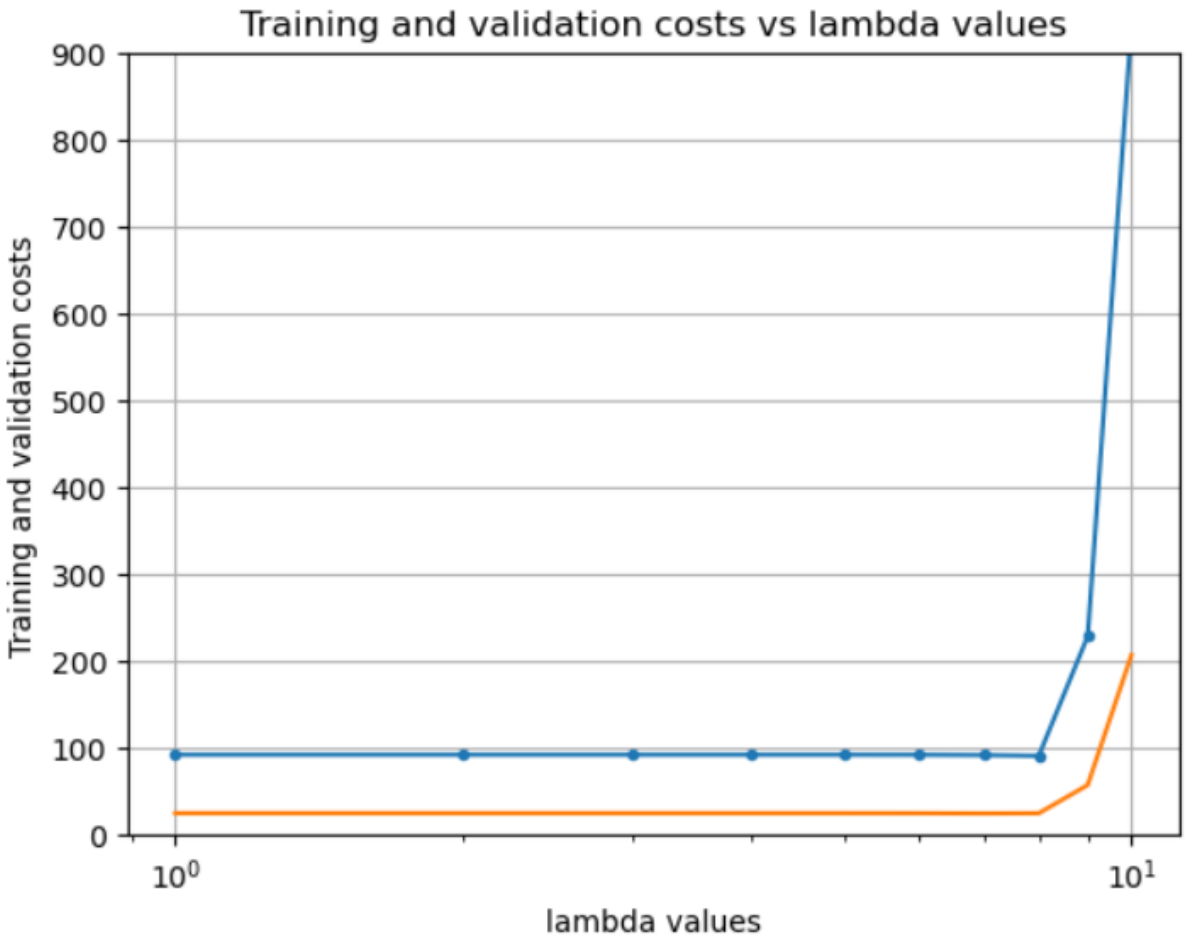
Polynomial Feature Transformations

For the first experiment, polynomial transformation was performed on the features utilizing PolynomialFeatures() from Sklearn. Taking in a parameter that represented the degree of a polynomial we were able to transform our features to up to a degree of 3, meaning being able to cube our feature vectors. Any higher degree resulted in the number of features growing too large to be of practical use. The first experiment used no regularization and the valdition and training costs were recorded to determine the best model.

Dataset	Degree of 1	Degree of 2	Degree of 3
Training	92.91391	128.17886	21955102.54549
Validation	24.91391	371937.28213	4.22232e+31

The costs continued to grow exponentially as the degree of the transformation increased so a degree of 1 or no transformation at all was selected as the best for the first experiment.

For the second experiment of the first model, ridge regression was implemented and the results were recorded based on the lambda values used. Similar results were obtained when performing feature transformation as before so the degree of transformation remained 1 for this experiment.



This time around with a value of λ = 0.0359381366380464 we were able to obtain a lower training and validation cost of 90.94369 and 24.85188 respectively. This was only a very slight improvement however.

λ->	1e-10	1.66e-09	1.78e-08	4.64e-07	7.74e-06
Training Set	92.12508	92.12508	92.12506	92.12483	92.12094
Validation Set	24.91391	24.91391	24.91390	24.91369	24.91031

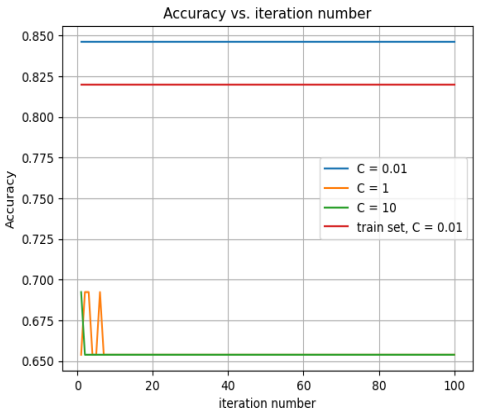
λ->	1.29e-4	2.15e-3	3.59e-2	.599	10
Training Set	92.06472	91.73155	90.94369	228.33767	931.21718
Validation Set	24.86099	24.62125	24.85188	57.32438	206.75667

With this information we decided the best linear regression model would be with lambda value 3.59e-2 and a feature transformation of degree 1 or no transformation. Resulting in the values using the training set:
RSS = 32.282295
TSS = 61.717900
R^2 = 0.476938

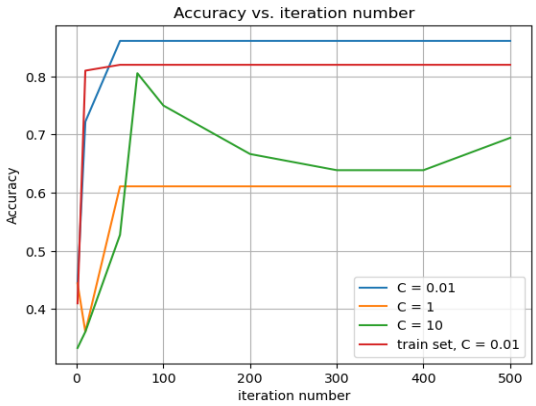
And using validation set:
RSS = 9.072885
TSS = 14.498000
R^2 = 0.374197

SVM

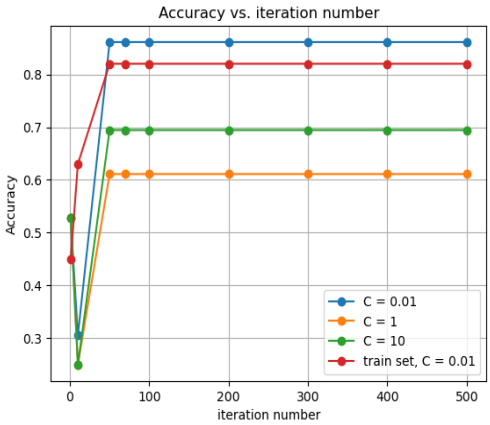
The second model we tested with was SVMs. This time we slightly shifted the objective of the project to not only calculate the rating of the games but to then categorize them as successful or not if the rating was over a 2.5 or not. Using the Scikit-Learn Python Library we were able to test using a linear kernel function with L1 and L2 regularization, a polynomial kernel with degrees up to 2, and radial-basis kernel functions.
The charts below show the level of accuracy of the model with the validation and train set in relation to the number of iterations. Generally the accuracy slightly shifts upward and as the iterations increase eventually flatline and this trend is shared with all of the SVM models.
SVM using L1, Linear Kernel SVM using L2, Linear Kernel

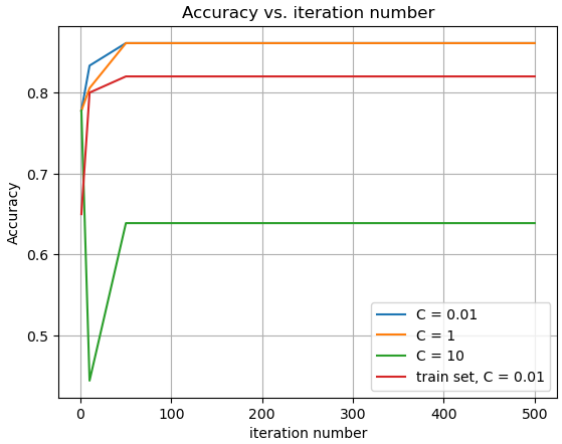


SVM using L2, RBF kernel



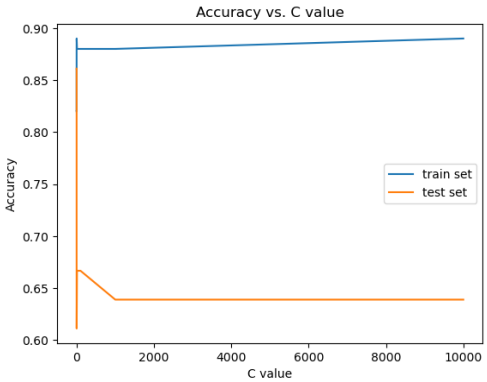
SVM using L2, polynomial kernel of degree 2



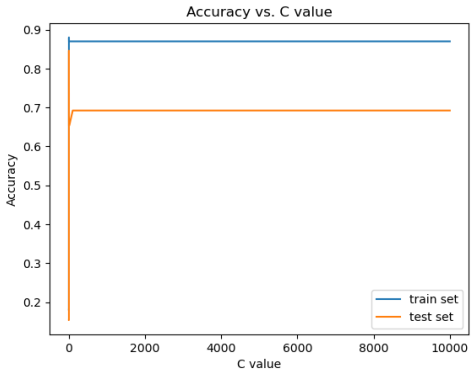


For the change of C value:

SVM using L2, Linear kernel

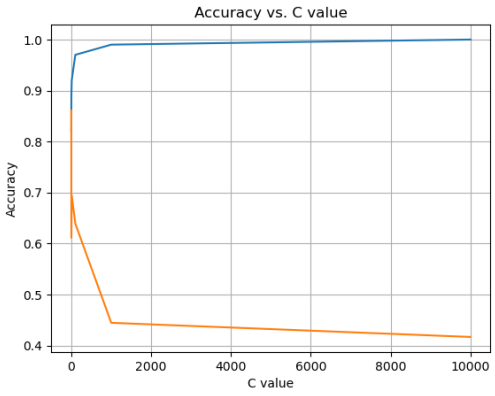


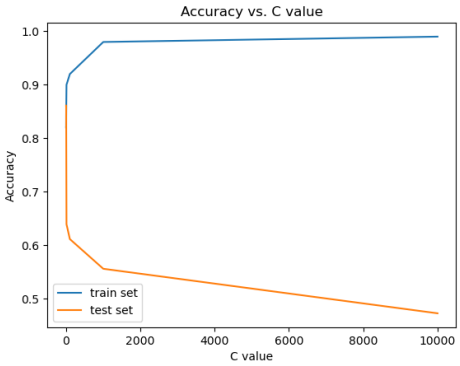
SVM using L1, Linear kernel



SVM using L2, RSF kernel:

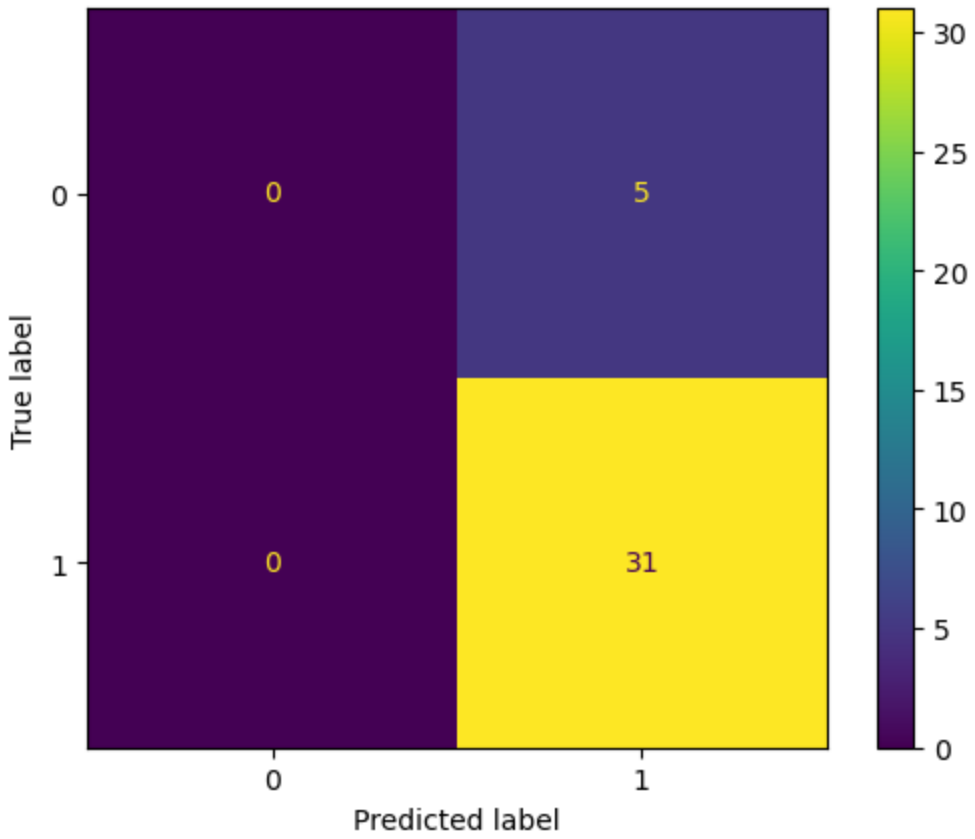
SVM using L2, polynomial kernel with degree 2:





C kernels	0.0000001	0.000001	0.001	0.01	0.1	1	10	100	1000
X : rbf with L2 train	0.82	0.82	0.82	0.82	0.82	0.89	0.92	0.97	0.99
X : rbf with L2 test	0.86	0.86	0.86	0.86	0.86	0.61	0.69	0.63	0.44
X : linear train	0.82	0.82	0.82	0.82	0.88	0.89	0.88	0.88	0.88
X : linear test	0.86	0.86	0.86	0.86	0.63	0.61	0.67	0.67	0.67
X : poly train	0.82	0.82	0.82	0.82	0.82	0.84	0.9	0.92	0.98
X : poly test	0.86	0.86	0.86	0.86	0.86	0.86	0.63	0.61	0.55
X : rbf with L1 train	0.18	0.18	0.18	0.82	0.88	0.87	0.87	0.87	0.87
X : rbf with L1 test	0.15	0.15	0.15	0.84	0.76	0.65	0.65	0.69	0.69

From the table we can observe that when C = 0.01 we have the highest accuracy for all the SVM types. So we can create the confusion matrix:



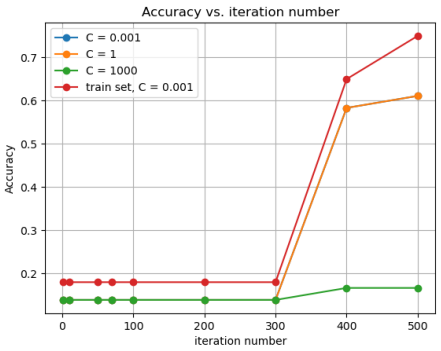
So for this confusion matrix, we have Recall equal to 31/(31+0) = 1,
Precision equal to 31/(31+5) = 0.86.

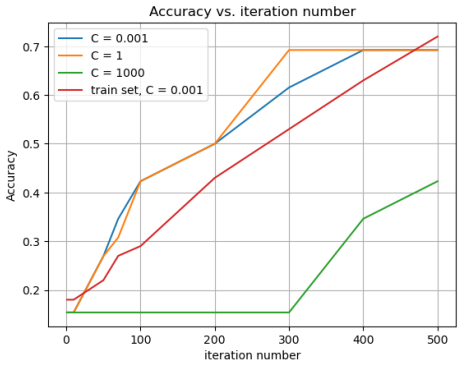
Neural Networking:

We used neural networks as the third model to train our dataset. To select the best training model, we used different activation functions and we modified hyperparameters including the number of layers of neural networks, learning rate, the iteration number and the C value. To implement the neural networks, we selected to use MLPClassifier from Scikit-Learn Library. For activation functions, we selected the tanh, ReLu and the logistic model from the library. In all of the cases we used L2 regularization. We tried different combinations of hyperparameters and the result are the following:

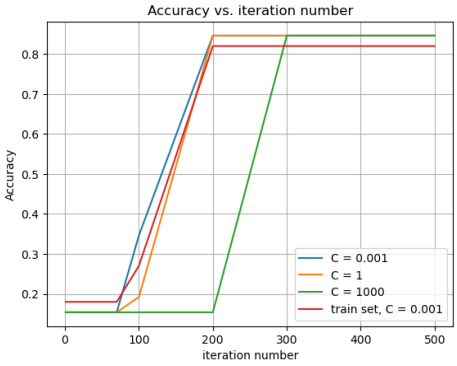
For different iteration number:

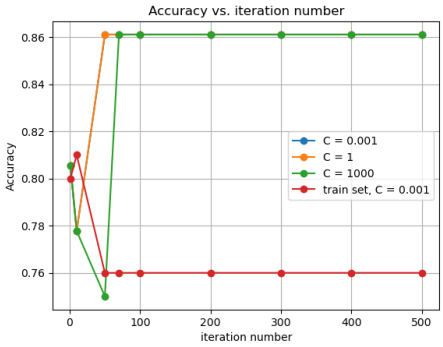
ReLu, hidden_layer_sizes=(5, 2) Tanh, hidden_layer_sizes=(5, 2)





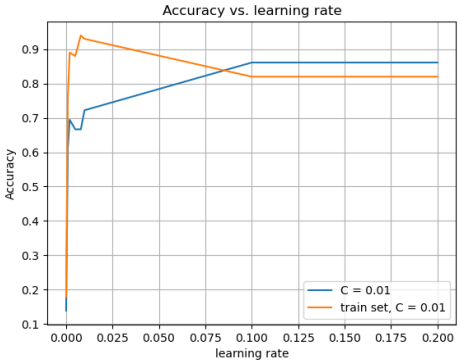
Logistic, hidden_layer_sizes=(5, 2) ReLu, Ploy, hidden_layer_sizes=(5, 2)

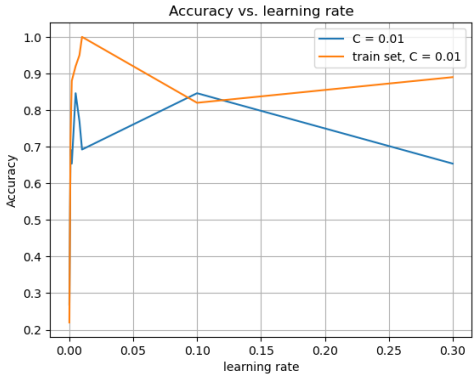




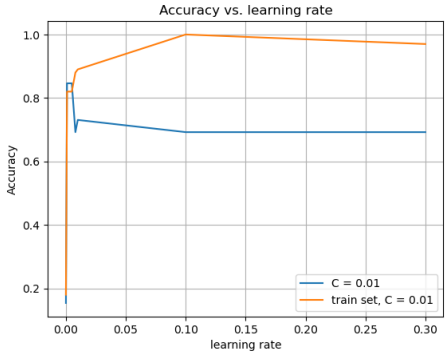
For different learning rate:

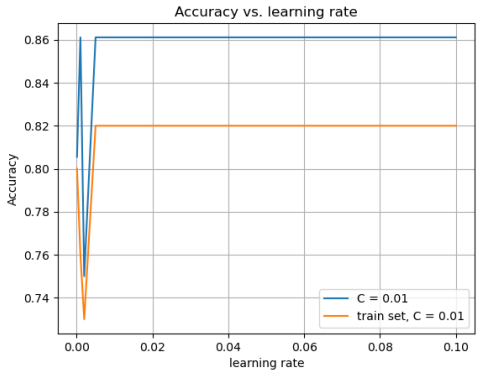
ReLu, hidden_layer_sizes=(5, 2) Tanh, hidden_layer_sizes=(5, 2)





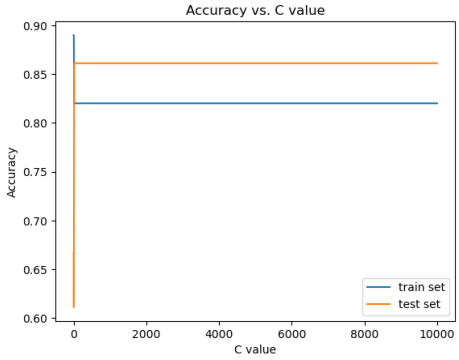
Logistic, hidden_layer_sizes=(5, 2) ReLu, Ploy, hidden_layer_sizes=(5, 2)



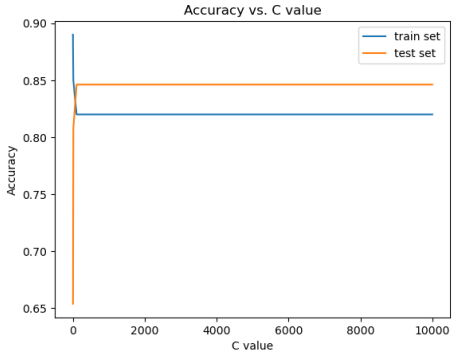


For different value of C in 2 layer Neural Networks:

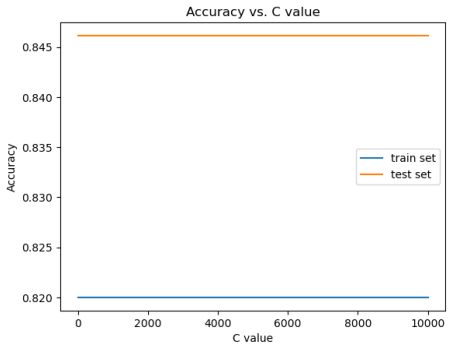
ReLu, hidden_layer_sizes=(5, 2)



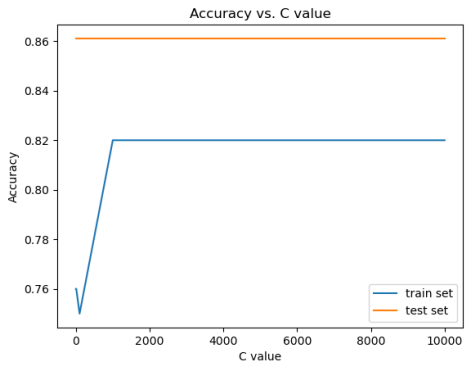
Tanh, hidden_layer_sizes=(5, 2)



Logistic, hidden_layer_sizes=(5, 2)



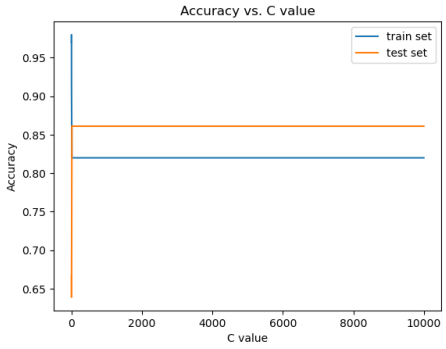
ReLu, Ploy, hidden_layer_sizes=(5, 2)

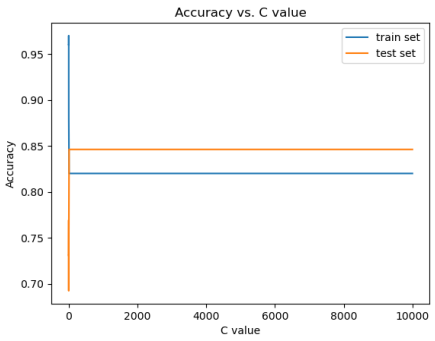


For different value of C in 3 layer Neural Networks:

ReLu, hidden_layer_sizes=(5, 5, 2)

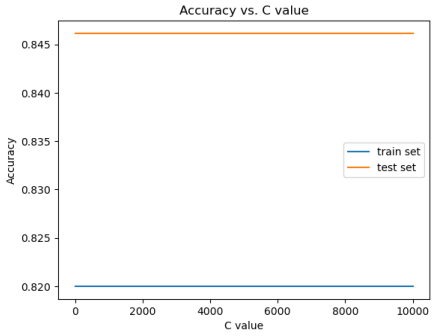
Tanh, hidden_layer_sizes=(5, 5, 2)

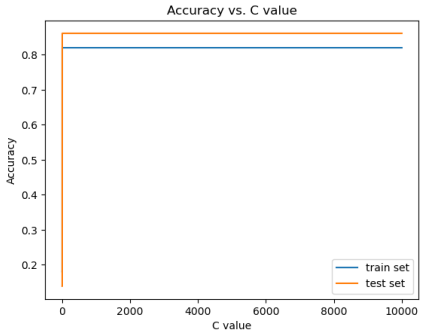




Logistic, hidden_layer_sizes=(5, 5, 2)

ReLu, Ploy, hidden_layer_sizes=(5, 5, 2)

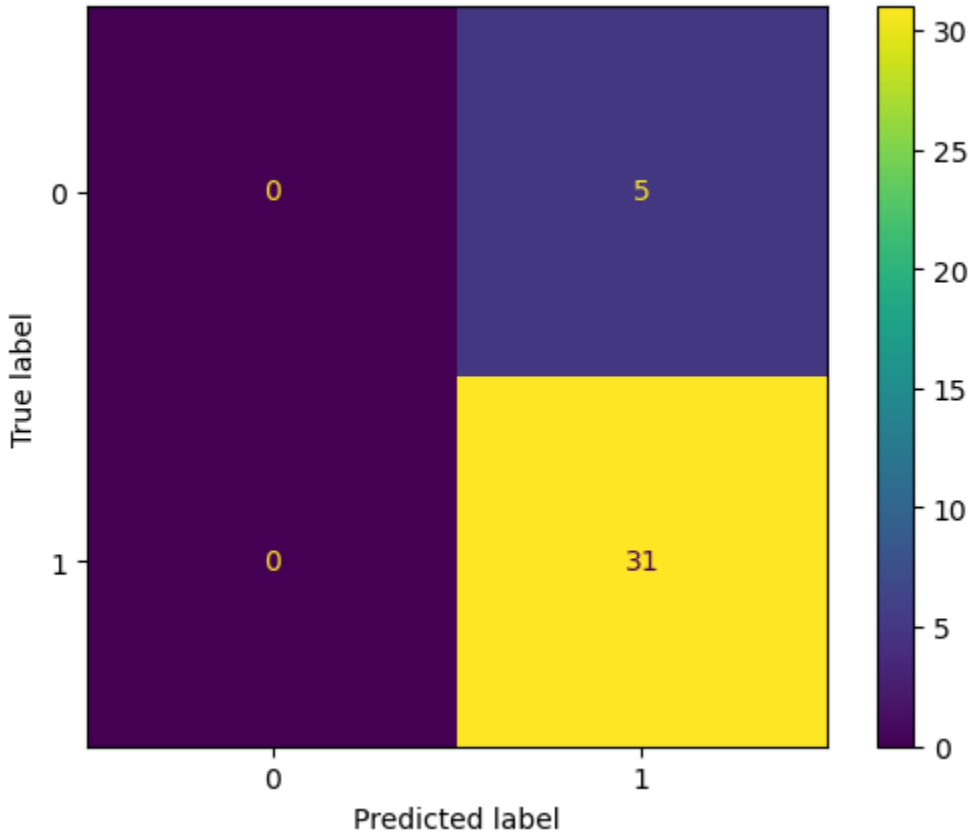




C layers	0.0000001	0.000001	0.001	0.01	0.1	1	10	100	1000
2 layers : ReLu with poly train	0.76	0.76	0.76	0.76	0.76	0.76	0.76	0.75	0.82
2 layers : ReLu with poly test	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
2 layers: Logistic train	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
2 layers: Logistic test	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84
2 layers: tanh train	0.89	0.89	0.89	0.89	0.89	0.88	0.85	0.82	0.82
2 layers: tanh test	0.65	0.65	0.65	0.65	0.65	0.65	0.8	0.84	0.84
2 layers : ReLu train	0.89	0.89	0.89	0.89	0.89	0.89	0.82	0.82	0.82
2 layers : ReLu test	0.66	0.66	0.66	0.66	0.66	0.61	0.86	0.86	0.86
3 layers : ReLu with poly train	0.18	0.18	0.18	0.18	0.18	0.82	0.82	0.82	0.82
3 layers : ReLu	0.14	0.14	0.14	0.14	0.14	0.86	0.86	0.86	0.86

with poly test									
3 layers: Logistic train	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82
3 layers: Logistic test	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84
3 layers: tanh train	0.96	0.96	0.96	0.97	0.92	0.88	0.82	0.82	0.82
3 layers: tanh test	0.73	0.73	0.73	0.76	0.69	0.73	0.84	0.84	0.84
3 layers : ReLu train	0.97	0.97	0.98	0.98	0.94	0.91	0.82	0.82	0.82
3 layers : ReLu test	0.67	0.67	0.67	0.67	0.67	0.63	0.86	0.86	0.86

We can see that the best model we can get is 2 layer structure using ReLu with C value equal to 10. From the learning rate and the iteration graphs we can see that when the learning rate equals 0.125 and iteration equals to 500 the model has the best performance. So we can create the confusion matrix:



So for this confusion matrix, we have Recall equal to 31/(31+0) = 1,
Precision equal to 31/(31+5) = 0.86.

Conclusion:

The performance of the linear regression model did not improve with the transformation of its features. As the degree increased the mean square error of the models would so as well, and when using the validation set outputted an MSE vastly larger than with the training set, indicating high levels of overfitting. Meaning the model had high bias and low variance as it wasn't able to distinguish on any trend when new data was introduced. When tested with ridge regression similar results were shown with only a small improvement when using values of lambda 2.15e-3 and 3.59e-2. Since transformation did not improve the accuracy of the model perhaps an introduction of more base features and examples may better boost its performance for linear regression

For SVM with a recall of 1 and precision 0.86 our model is more likely to predict positive outcome, or games with a high rating. With regularization the accuracy of the model with the training and test set were quick to flatline as c increased indicating the lack of data as it was quick to become very overfit. The test set was always much lower than the training set which may have been because of the size of the test not being big enough to match up or a lack of variety in the dataset. As the recall being 1 meant everything was being classified as positive so sets with more negative ratings wouldn't do as well.

For the Neural Network, the best model we get is using ReLu activation with C value equal to 10 (for L2 regularization), and learning rate shall be equal to 0.125 and iteration number be equal to 500 iterations. The final precision is around 0.86 which shows that our model has a high accuracy. As the C value increases, we can see that the test accuracy of these 8 types of model are higher than their training accuracy, which finally reached higher than 0.85. This result shows that our model is a little bit underfitting. One reason that can cause underfitting is that the number of instances in the training set is too small to train a good neural network model, either with 2 layer structures or 3 layer structures. Also, because the number of instances in the training set is small, the confusion matrix lacks some information. In this case, we don't have FN value, which could be caused by the small size of the validation set.

With the accuracy of the project with SVM and Neural Network are somewhat high with 86 and with linear regression having an RSS of 47% with the training set and 37% validation set, only a 10% decrease in reliability, overall our project can be considered somewhat a success. The ratings it provides may not be exact but it can still

provide a reasonable value as an indication of how good a game may be that others can use to validate their own ratings with.

Overall we believe we could have benefited from having more data for training purposes to be able to utilize different transformations or regularization techniques better as currently they tend to quickly overfit and underperform when outside data is used. As well as the type of data utilized may be more diverse and something we could have looked more into, especially with the types of features included with the dataset may not be as relevant to the rating as others and possibly better features of a game exist outside the dataset that can be added.