CS-UY 4563: Machine Learning

Final Project Write Up

Spaceship Titanic Classification

11:00 am Section

Apr 26, 2023

Professor Sellie

Kevin Chen, Nick Ferrara

Introduction:

This project uses data from a Kaggle competition that is called Spaceship Titanic which focuses on the use of machine learning models in order to predict whether or not a passenger was transported to an alternate dimension during the *Spaceship Titanic's* collision with a spacetime anomaly. The dataset given by Kaggle was partially used, with rows that contained missing values dropped from our training and test datasets.

Logistic regression, SVM, and neural networks were used to classify the passengers as transported or not transported to an alternate dimension. All three models were trained multiple times with different regularization techniques.

Preprocessing:

The dataset was split into a training and test set and we ended up with around 5300 training examples and 1300 test examples. Additionally, a cursory glance through the dataset showed that there were some missing values in the dataset, so the rows with missing values were dropped in order to get better performance in model training.

A correlation matrix was used to determine if any of the features had a significant effect on classification, but no specific feature seemed to have a large correlation with the final classification.

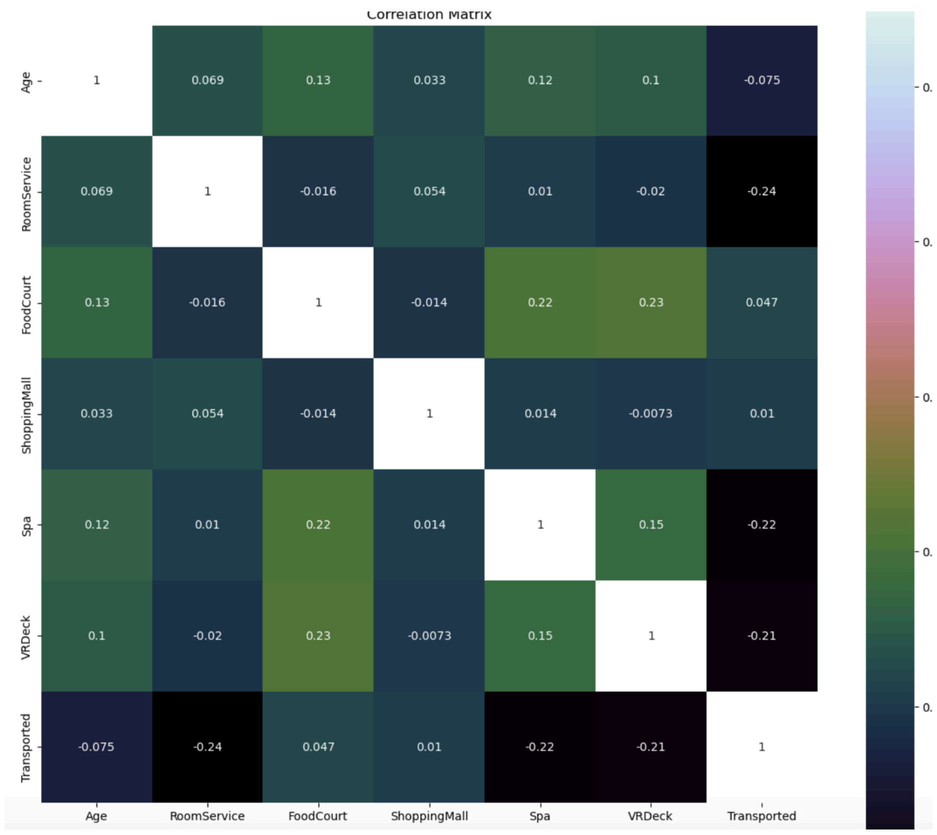


Figure 1: Correlation Matrix of Dataset

There were many columns of data in the dataset that consisted of text data such as HomePlanet, Destination and Cabin. To preprocess the data into numerical features, one-hot-encoding was used to transform these features into numerical data and then the align method was used in order to give the training and validation datasets the same number of columns.

Before we ran all of our models, we did some EDA on the data to get a better sense of the data that we were working with and noticed some interesting features about the dataset. First, the dataset had many passengers from Earth, but not nearly as many passengers from other planets.

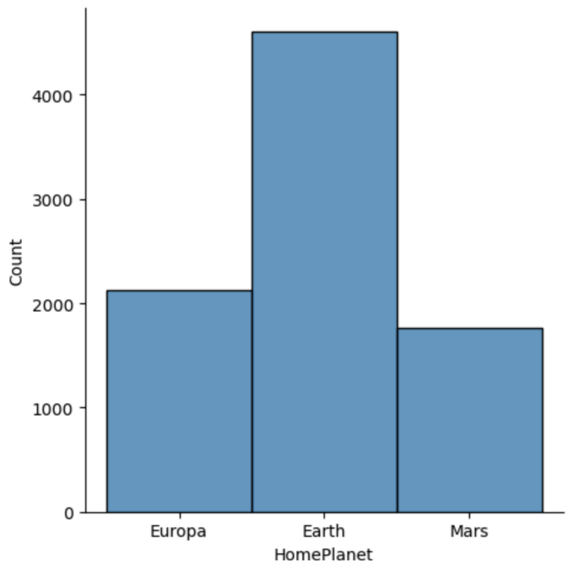


Figure 2: HomePlanet Distribution

Additionally, the age distribution of the passengers seemed to be roughly normally distributed with some outliers among the young passengers which could indicate the presence of many families onboard traveling with young children.

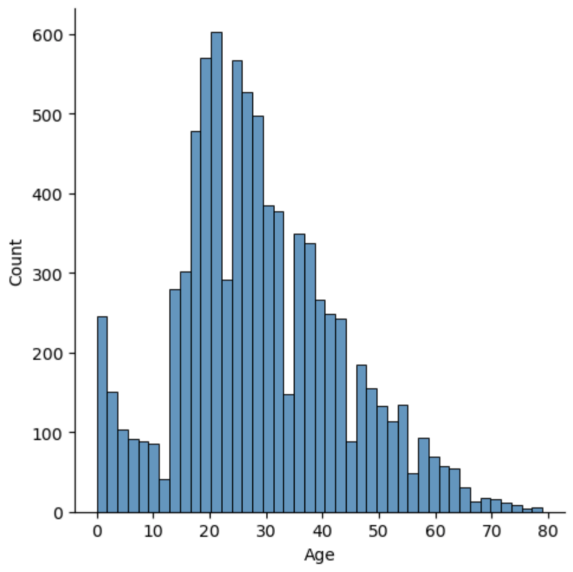


Figure 3: Age Distribution

Finally, we looked at the distribution of transported passengers and we found that there was a pretty even split between passengers that were transported and ones that were not.

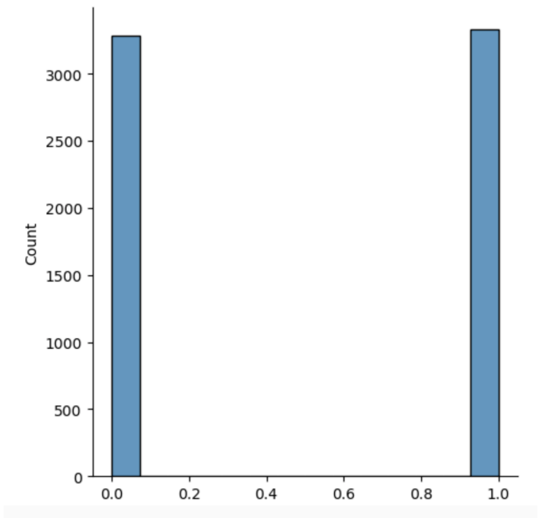


Figure 4: Classification Distribution

Logistic Regression:

The first model that we decided to train was the logistic regression model and we used the sklearn implementation of logistic regression.We started out with no regularization and achieved a training accuracy of 79.30% and a validation accuracy of 75.94%. Then, we trained the logistic regression with no regularization, L1, and L2 regularizations, with hyperparameters of C = [0.01, 0.1, 1, 10, 100, and 1000] for three cases: dataset with no features transformed, dataset with the square of every feature, and a dataset with the square root of every feature.

Logistic Regression Results:

C Values-> Test Cases ↓	0.01	0.1	1	10	100	1000
X: train	0.793	0.793	0.793	0.793	0.793	0.793
X:	0.759	0.759	0.759	0.759	0.759	0.759

validation						
X: L1 train	0.793	0.795	0.795	0.806	0.805	0.805
X: L1 validation	0.782	0.766	0.763	0.770	0.752	0.741
X: L2 train	0.800	0.791	0.793	0.793	0.793	0.793
X: L2 validation	0.782	0.759	0.762	0.760	0.760	0.760
X^2: train	0.759	0.759	0.759	0.759	0.759	0.759
X^2: validation	0.755	0.755	0.755	0.755	0.755	0.755
X^2: L1 train	0.776	0.774	0.776	0.790	0.790	0.790
X^2: L1 validation	0.750	0.741	0.744	0.740	0.731	0.725
X^2: L2 train	0.759	0.759	0.759	0.759	0.759	0.759
X^2: L2 validation	0.755	0.755	0.755	0.755	0.755	0.755
√X: train	0.800	0.800	0.800	0.800	0.800	0.800
√X: validation	0.784	0.784	0.784	0.784	0.784	0.784
√X: L1 train	0.790	0.801	0.803	0.811	0.811	0.812
√X: L1 validation	0.778	0.780	0.783	0.782	0.751	0.735
√X: L2 train	0.797	0.801	0.801	0.801	0.801	0.800
√X: L2 validation	0.787	0.785	0.781	0.782	0.782	0.780

The results of training the models are also visually shown below.

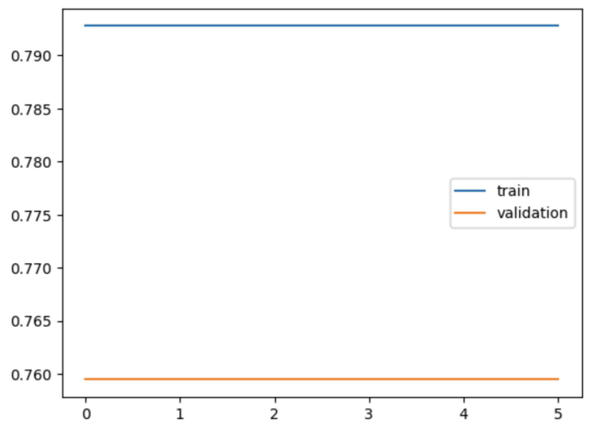


Figure 5: Regression of X

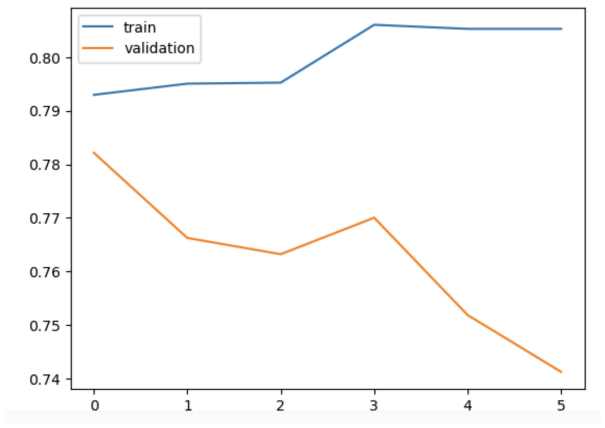


Figure 6: L1 Regression of X

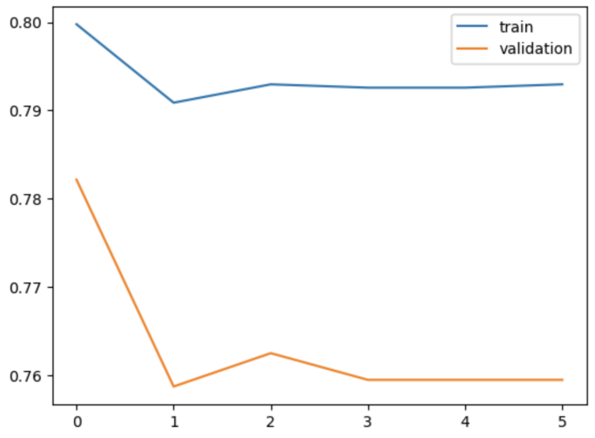


Figure 7: L2 Regression of X

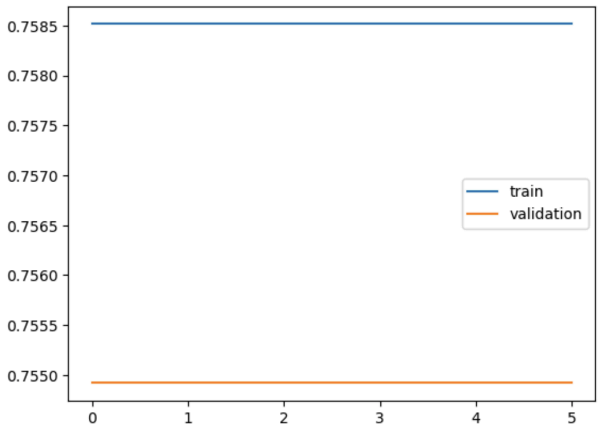


Figure 8: Regression of X Squared

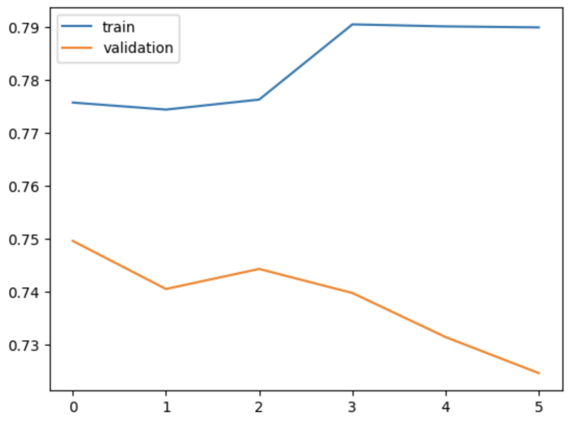


Figure 9: L1 Regression of X Squared

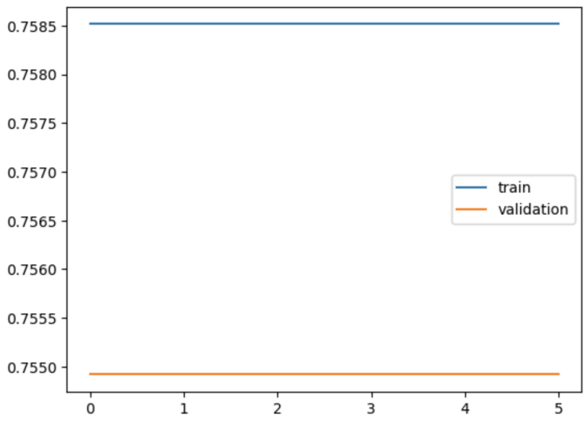


Figure 10: L2 Regression of X Squared

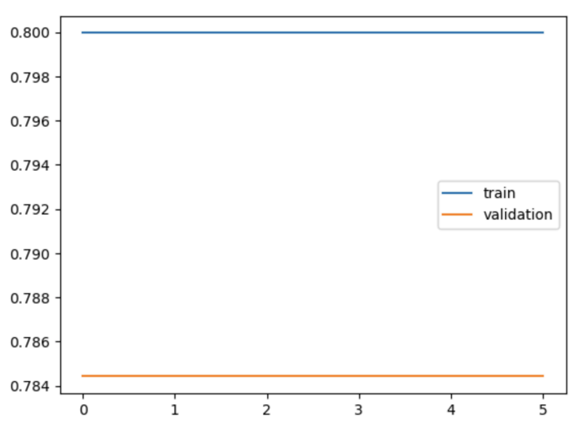


Figure 11: Regression of √X

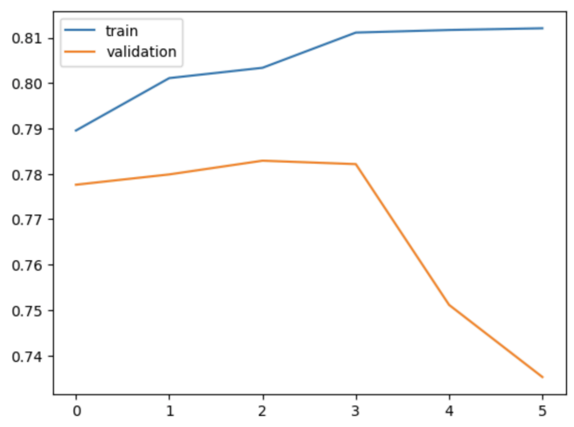


Figure 12: L1 Regression of √X

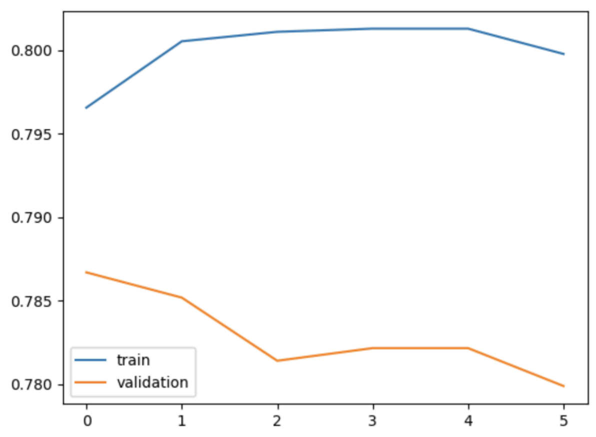


Figure 13: L2 Regression of √X

The model that had the highest test accuracy of 0.787 was the square root feature transformed data with L2 validation and a c value of 0.01. The precision of this model was 0.757 and the recall of this model was 0.848. We retrained on this model and created a confusion matrix.

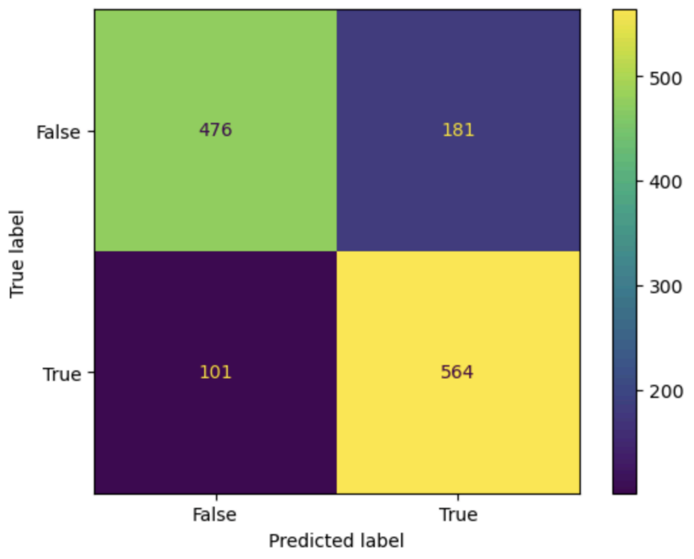


Figure 14: Confusion Matrix for √X with L2 Regularization and c = 0.01

The model showed slight bias towards true, or transported predictions, with more true predictions than false predictions. As we saw earlier, the distribution of transported and not-transported passengers was pretty much evenly split, so this suggests bias in the model towards transported predictions.

We additionally looked at the top ten highest weighted features in the best performing model and saw that the features that mattered the most seemed to be HomePlanet as well as whether the passenger was in CryoSleep. However, no one feature had overwhelming importance compared to the other features, which is what we speculated based on what we saw in the correlation matrix.

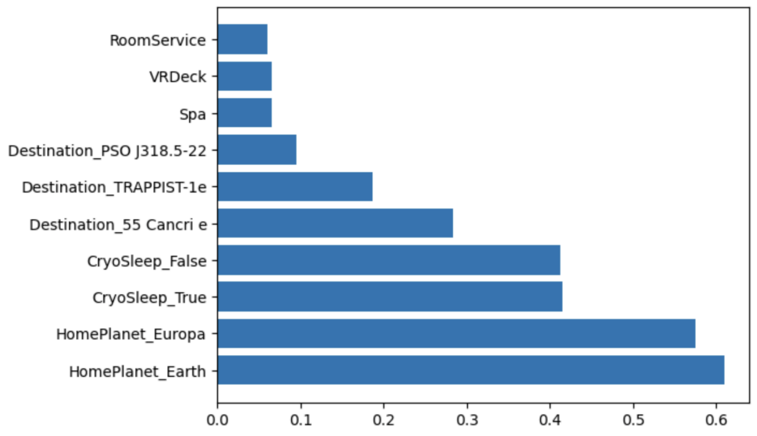


Figure 15: Feature Importance for Logistic Regression

SVM:

The next model that we used was Support Vector Machines. We used sklearn's implementation of SVM, more specifically SVC. The training and validation datasets used to train the model were the original datasets and the square root transformed dataset, scaled using the sklearn standard scaler. The original unscaled datasets were initially used for training these models, however, training using the original datasets was too time consuming and would not have been feasible so a scaled dataset was used instead to make the training more efficient. We speculate that different scales on different features would cause some features to take longer to converge than others. Three kernel functions were also tested: radial-basis function, linear, and polynomial of degree 3. All of these feature transformations were trained with a regularization parameter of L2 squared with c values of [0.01, 0.1, 1, 10, 100, and 1000].

SVM Results:

C Values-> Kernels	0.01	0.1	1	10	100	1000

Types ↓						
X: linear train	0.794	0.802	0.805	0.805	0.805	0.805
X: linear validation	0.784	0.764	0.772	0.774	0.773	0.776
X: poly train	0.544	0.548	0.556	0.567	0.751	0.790
X: poly validation	0.507	0.508	0.507	0.529	0.700	0.723
X: rbf train	0.578	0.726	0.779	0.802	0.816	0.818
X: rbf validation	0.560	0.671	0.696	0.720	0.722	0.722
√X: linear train	0.811	0.812	0.812	0.812	0.811	0.812
√X: linear validation	0.800	0.797	0.796	0.796	0.796	0.796
√X: poly train	0.544	0.549	0.556	0.564	0.772	0.810
√X: poly validation	0.506	0.507	0.506	0.520	0.722	0.736
√X: rbf train	0.701	0.728	0.808	0.816	0.823	0.828
√X: rbf validation	0.655	0.673	0.721	0.736	0.732	0.738

The results of training the models are also visually shown below.

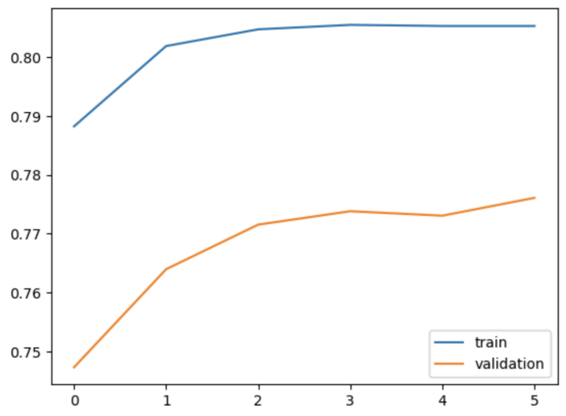


Figure 16: Linear Kernel for Scaled X

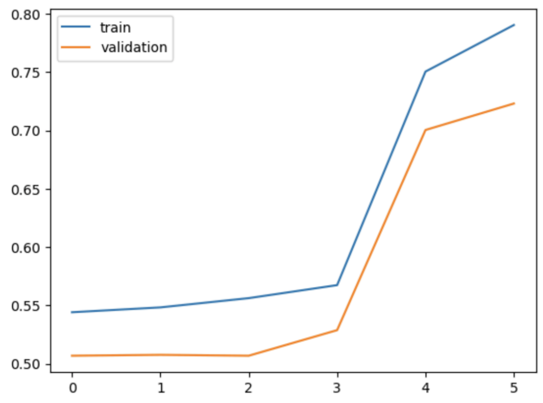


Figure 17: Poly Kernel for Scaled X

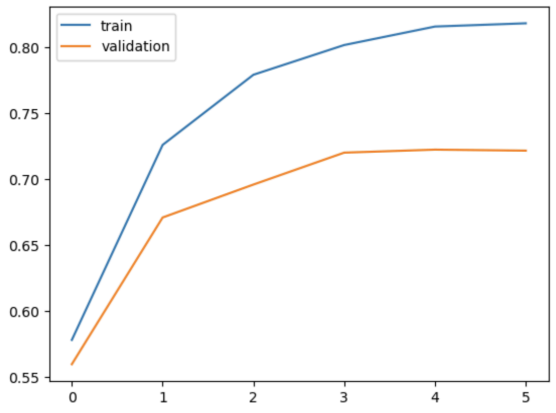


Figure 18: RBF Kernel for Scaled X

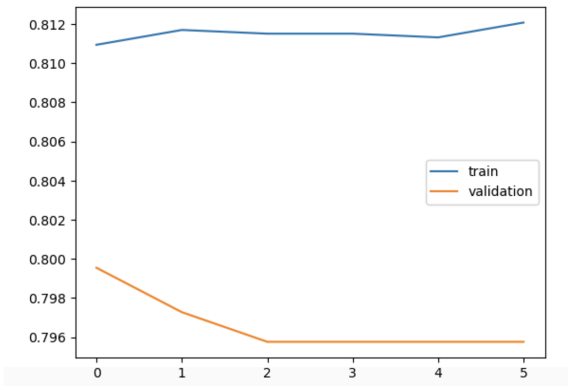


Figure 19: Linear Kernel for √X

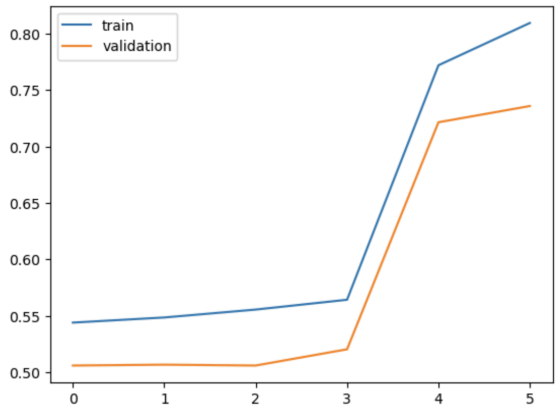


Figure 20: Poly Kernel for √X

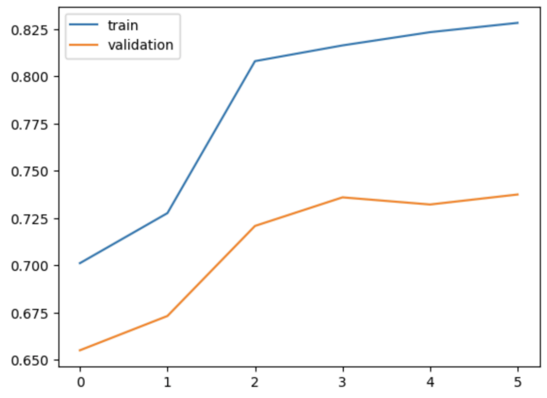


Figure 21: RBF Kernel for √X

The model that had the highest test accuracy, 0.800, was the square root feature transformed data with L2 squared validation, linear kernel and a c value of 0.01. The precision of this model is 0.779 and the recall of this model is 0.839. We retrained on this model and created a confusion matrix.

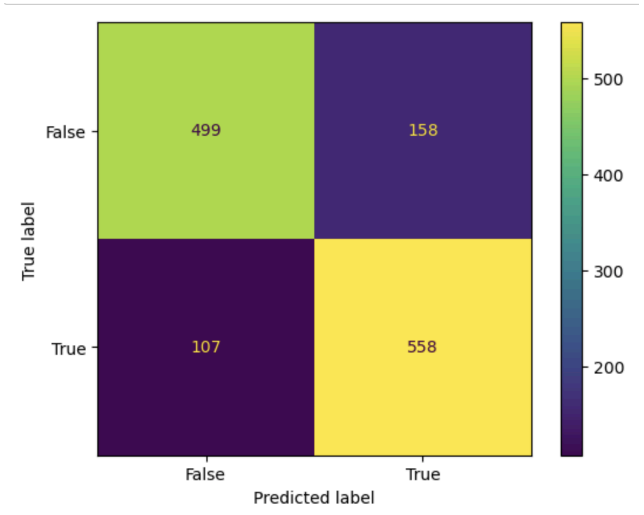


Figure 22: Confusion Matrix for √X with Linear Kernel and c = 0.01

Similar to the logistic regression model, this model showed slight bias towards true, or transported predictions, with more true predictions than false predictions. As we saw earlier, the distribution of transported and not-transported passengers was pretty much evenly split, so this suggests bias in the model towards transported predictions.

We additionally looked at the top ten highest weighted features in the best performing model and saw that the features that mattered the most seemed to be the amount spent on the VRDeck, Spa, RoomService and FoodCourt. This was different from what we saw in the logistic regression model, however, no one feature seemed to have significant weight over the other features, similar to the logistic regression model.

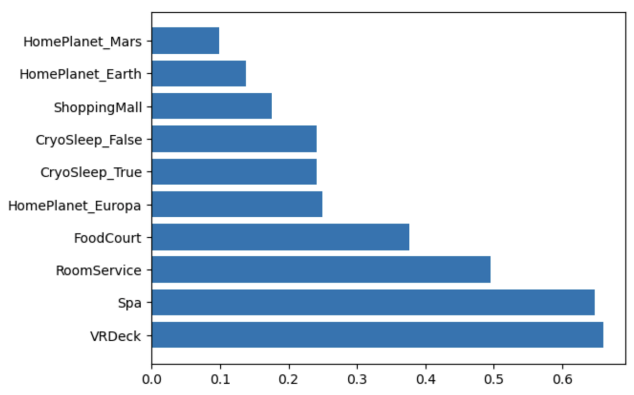


Figure 23: Feature Importance in SVM

Neural Network:

The next model that we used was a supervised neural network. Because neural network models are very versatile, there are many ways we can construct one. In order to find a model that best models whether a passenger has been transported to another dimension, we systematically adjusted the number of layers in our model, applied different regularization parameters, and used differently scaled data. We used tensorflow's keras library to build, train, and test our machine learning models. The first model consists of 349 input neurons (number of features in our data), 64 neurons in the hidden layer, and 1 output neuron with the sigmoid activation function. The second model consists of 349 input neurons, 64 neurons in the first hidden layer with the sigmoid activation function, 32 neurons in the second hidden layer with the ReLU (Rectified Linear Units) activation function, and 1 output neuron with the sigmoid activation function. Finally, the second model consists of 349 input neurons, 64 neurons in the first hidden layer with the sigmoid activation function, 32 neurons in the second hidden layer with the ReLU activation function, 16 neurons in the third hidden layer with the ReLU activation function, and 1 output neuron with the sigmoid activation function. As mentioned before, the training and validation datasets used to train the model were the unscaled, squared, and square-root datasets. The original unscaled datasets were initially used for training these models, however, as stated previously, training using the original datasets was time consuming so a scaled dataset was used instead to make the training more efficient. We speculate that different scales on different features would cause some features to take longer to converge than others. All of these models with the feature transformations were trained with and without regularization parameters of L1 and L2 squared with c parameter values of [0.01, 0.1, 1, 10, 100, and 1000]. Note that the loss function used for this type of binary classification is Mean Squared Error. Below are the best accuracy found while training each model under the conditions of 1. The Model Type, 2. Regularization Parameter, and 3. The dataset.

First Model Results: [349, 64, 1]

C Values-> Test Cases ↓	0.01	0.1	1	10	100	1000
X: No Reg. train	0.808	0.808	0.808	0.808	0.808	0.808
X: No Reg. validation	0.794	0.794	0.794	0.794	0.794	0.794
X: L1 train	0.798	0.788	0.763	0.700	0.572	0.661
X: L1 validation	0.795	0.790	0.781	0.776	0.715	0.675
X: L2 train	0.800	0.799	0.797	0.792	0.786	0.771
X: L2 validation	0.796	0.797	0.794	0.792	0.787	0.778
X^2: No Reg. train	0.799	0.799	0.799	0.799	0.799	0.799
X^2: No Reg. validation	0.787	0.787	0.787	0.787	0.787	0.787
X^2: L1 train	0.791	0.779	0.775	0.764	0.737	0.732
X^2: L1 validation	0.790	0.780	0.784	0.781	0.751	0.739

X^2: L2 train	0.789	0.791	0.792	0.778	0.770	0.761
X^2: L2 validation	0.785	0.786	0.784	0.778	0.756	0.770
√X: No Reg. train	0.812	0.812	0.812	0.812	0.812	0.812
√X: No Reg. validation	0.795	0.795	0.795	0.795	0.795	0.795
√X: L1 train	0.791	0.784	0.638	0.639	0.681	0.558
√X: L1 validation	0.790	0.782	0.639	0.700	0.691	0.533
√X: L2 train	0.801	0.793	0.785	0.773	0.631	0.740
√X: L2 validation	0.793	0.788	0.785	0.780	0.757	0.769

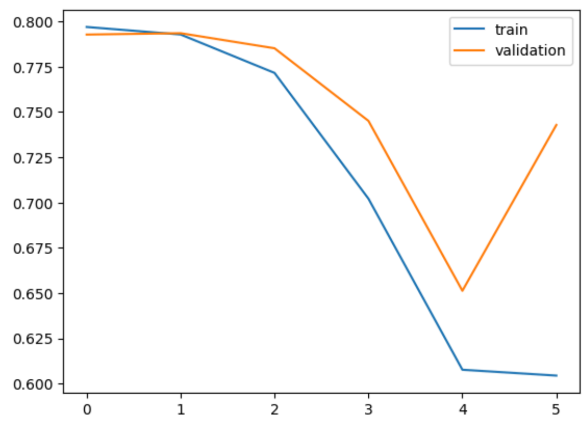


Figure 24: L1 Regularization for Scaled X

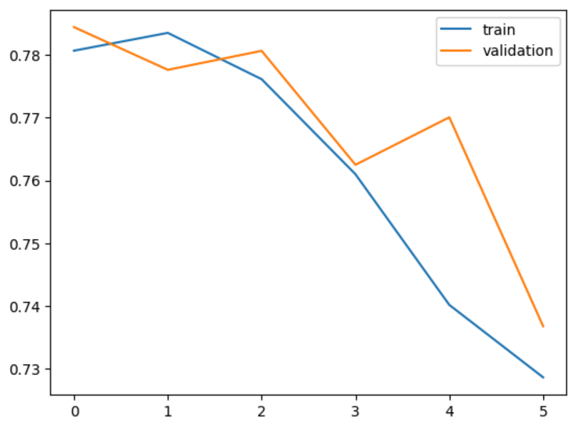
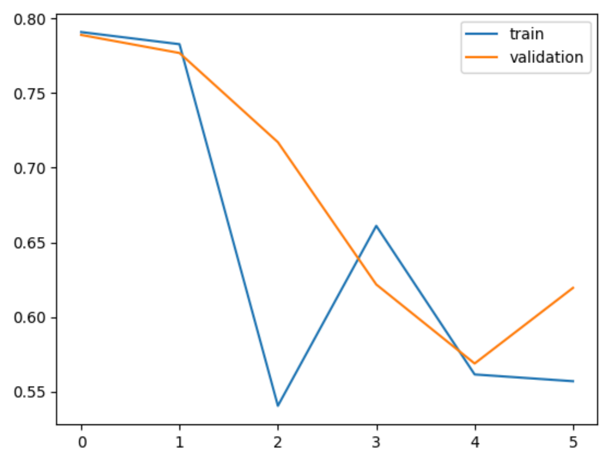


Figure 25: L1 Regularization for Scaled X^2



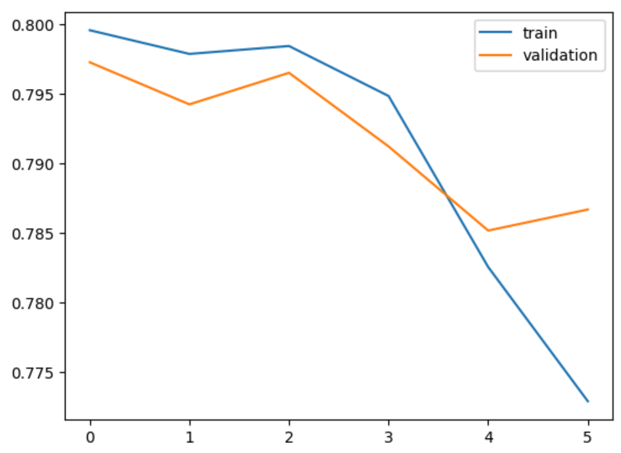
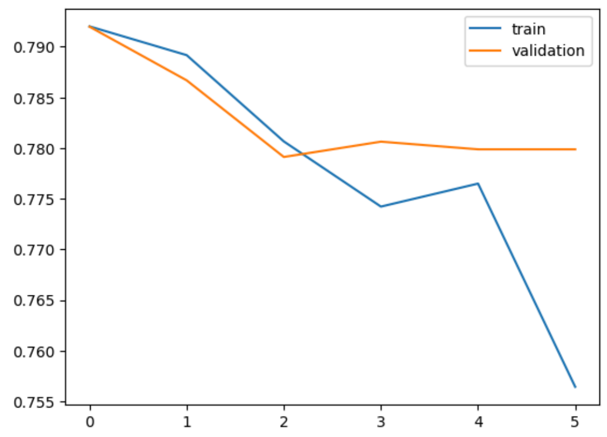


Figure 26: L1 Regularization for Scaled SqrtX **Figure 27: L2 Regularization for Scaled X**



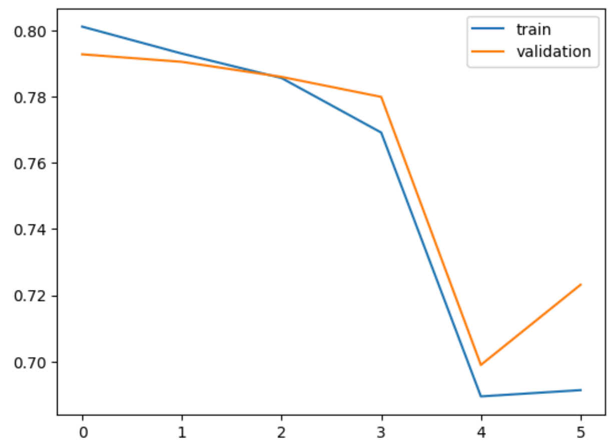
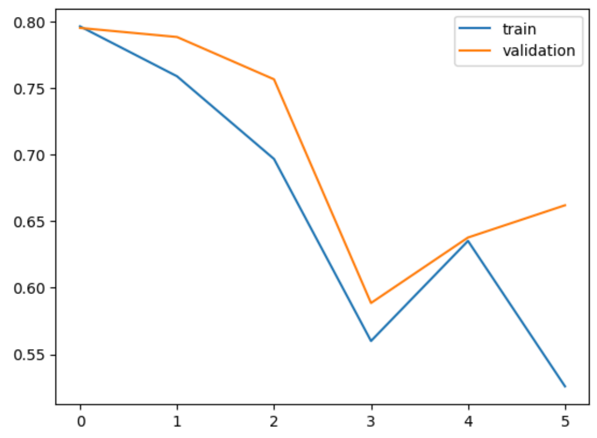


Figure 28: L2 Regularization for Scaled X^2 **Figure 29: L2 Regularization for Scaled sqrtX**

Second Model Results: [349, 64, 32, 1]

C Values-> Test Cases ↓	0.01	0.1	1	10	100	1000
X: No Reg. train	0.812	0.812	0.812	0.812	0.812	0.812
X: No Reg. validation	0.797	0.797	0.797	0.797	0.797	0.797
X: L1 train	0.796	0.781	0.680	0.710	0.555	0.601
X: L1 validation	0.795	0.792	0.679	0.640	0.682	0.558

X: L2 train	0.801	0.798	0.794	0.788	0.788	0.704
X: L2 validation	0.794	0.796	0.792	0.787	0.776	0.685
X^2: No Reg. train	0.805	0.805	0.805	0.805	0.805	0.805
X^2: No Reg. validation	0.793	0.793	0.793	0.793	0.793	0.793
X^2: L1 train	0.782	0.750	0.714	0.683	0.664	0.530
X^2: L1 validation	0.787	0.787	0.706	0.596	0.696	0.580
X^2: L2 train	0.791	0.783	0.774	0.759	0.762	0.759
X^2: L2 validation	0.791	0.784	0.785	0.770	0.750	0.772
√X: No Reg. train	0.816	0.816	0.816	0.816	0.816	0.816
√X: No Reg. validation	0.796	0.796	0.796	0.796	0.796	0.796
√X: L1 train	0.791	0.521	0.639	0.584	0.524	0.538
√X: L1 validation	0.788	0.534	0.660	0.539	0.646	0.537
√X: L2 train	0.799	0.788	0.770	0.645	0.597	0.608
√X: L2 validation	0.793	0.788	0.777	0.744	0.584	0.628



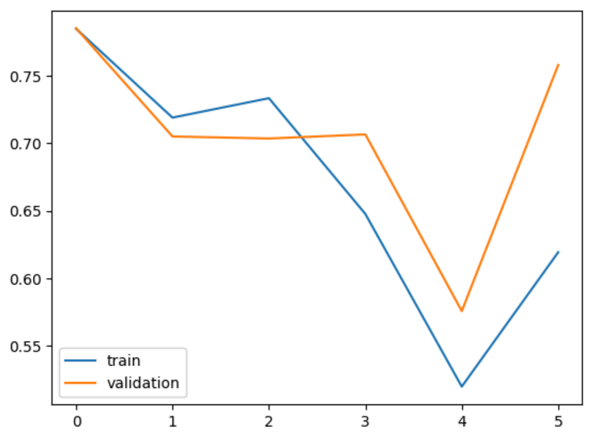
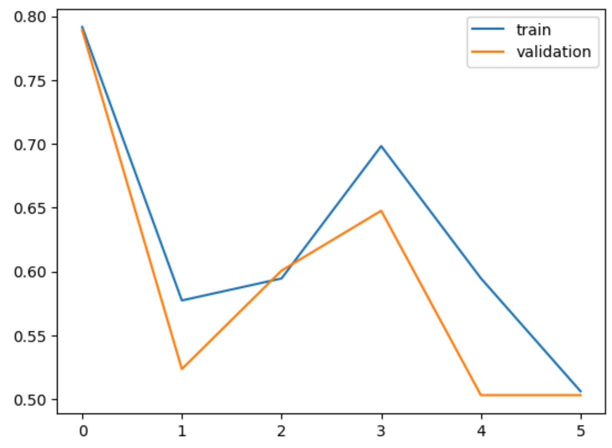


Figure 30: L1 Regularization for Scaled X **Figure 31: L1 Regularization for Scaled X^2**



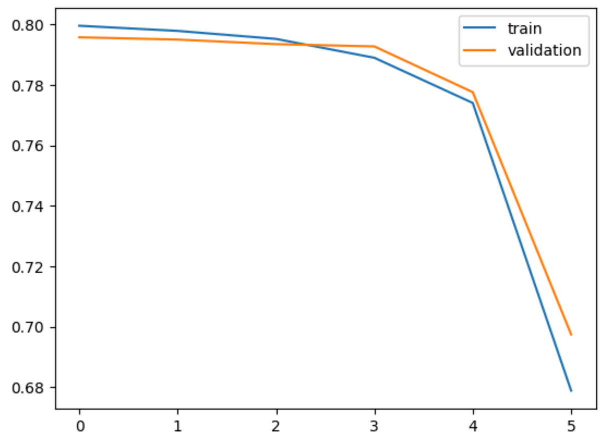
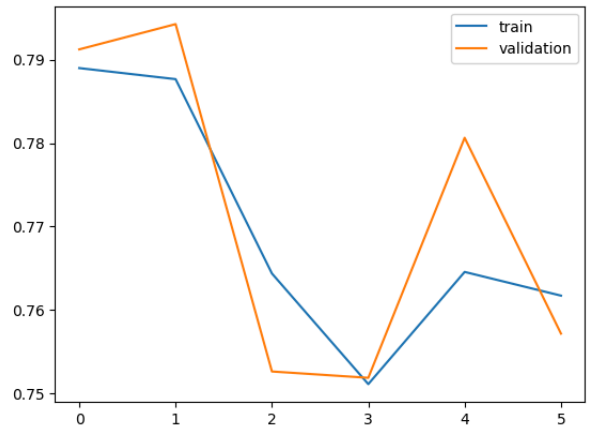


Figure 32: L1 Regularization for Scaled SqrtX **Figure 33: L2 Regularization for Scaled X**



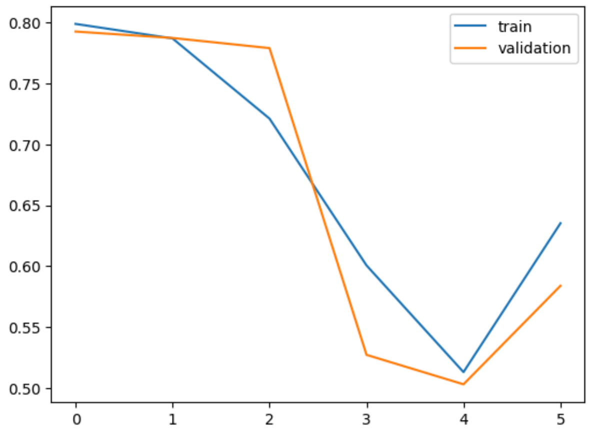
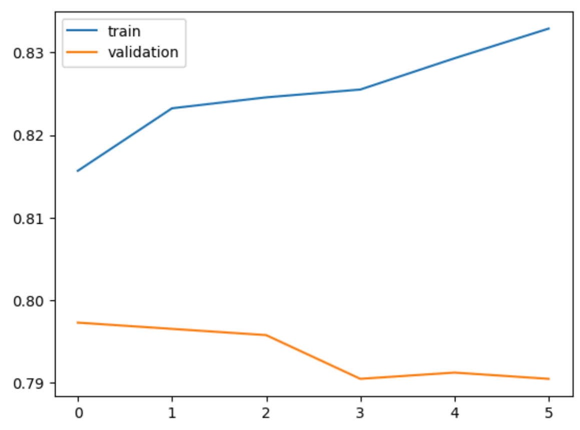


Figure 34: L2 Regularization for Scaled X^2 **Figure 35: L2 Regularization for Scaled sqrtX**

Third Model Results: [349, 64, 32, 16, 1]

C Values-> Test Cases ↓	0.01	0.1	1	10	100	1000
X: No Reg. train	0.815	0.815	0.815	0.815	0.815	0.815
X: No Reg. validation	0.797	0.797	0.797	0.797	0.797	0.797
X: L1 train	0.818	0.820	0.823	0.830	0.833	0.835

X: L1 validation	0.794	0.796	0.796	0.793	0.790	0.786
X: L2 train	0.833	0.838	0.843	0.845	0.850	0.851
X: L2 validation	0.790	0.787	0.781	0.778	0.778	0.775
X^2: No Reg. train	0.801	0.801	0.801	0.801	0.801	0.801
X^2: No Reg. validation	0.791	0.791	0.791	0.791	0.791	0.791
X^2: L1 train	0.814	0.820	0.822	0.825	0.826	0.827
X^2: L1 validation	0.790	0.796	0.790	0.787	0.787	0.784
X^2: L2 train	0.820	0.824	0.825	0.828	0.829	0.831
X^2: L2 validation	0.792	0.791	0.790	0.789	0.789	0.790
√X: No Reg. train	0.818	0.818	0.818	0.818	0.818	0.818
√X: No Reg. validation	0.793	0.793	0.793	0.793	0.793	0.793
√X: L1 train	0.837	0.851	0.863	0.874	0.879	0.887
√X: L1 validation	0.788	0.776	0.766	0.760	0.759	0.759
√X: L2 train	0.856	0.869	0.877	0.885	0.894	0.899
√X: L2 validation	0.775	0.769	0.778	0.775	0.770	0.770



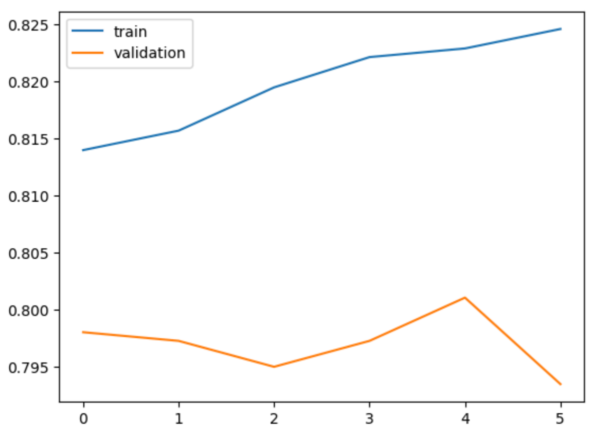
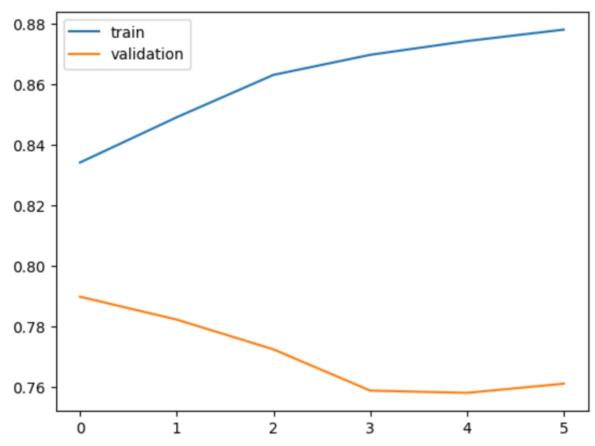


Figure 36: L1 Regularization for Scaled X   Figure 37: L1 Regularization for Scaled X^2



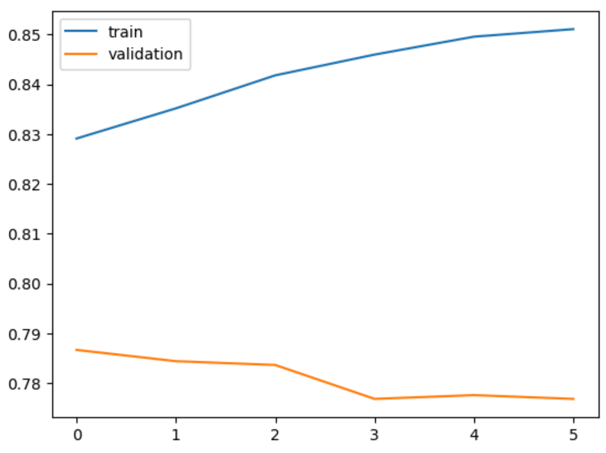
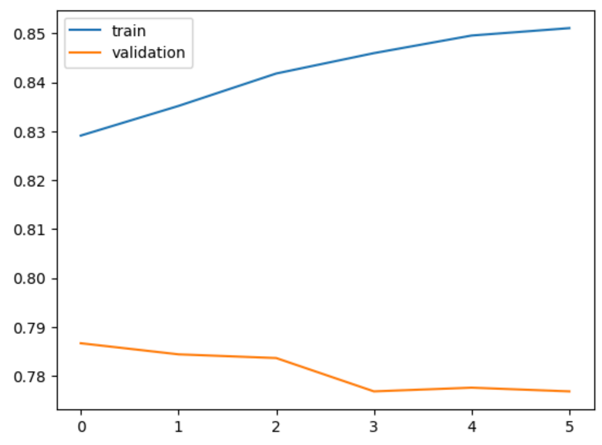


Figure 38: L1 Regularization for Scaled SqrtX   Figure 39: L2 Regularization for Scaled X



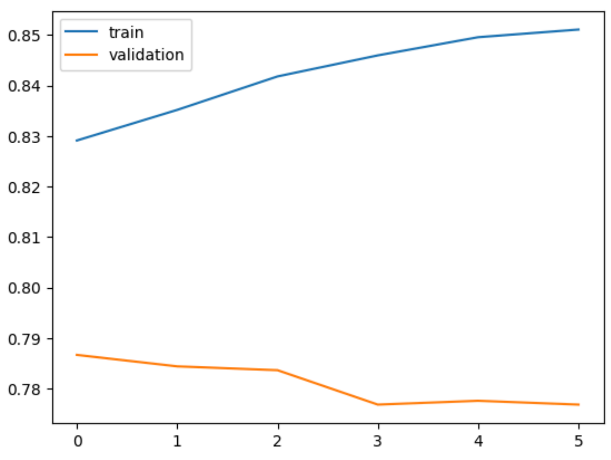


Figure 40: L2 Regularization for Scaled X^2 Figure 41: L2 Regularization for Scaled sqrtX

Although the validation accuracy seems to be similar across the 3 models and transformed data and thus may need more tests to confirm which one is better, for the sake of analysis we will choose the model that yielded the highest validation accuracy and most efficient to train: Model 3 of X with no regularization. We compute the confusion matrix for this model to understand its performance.

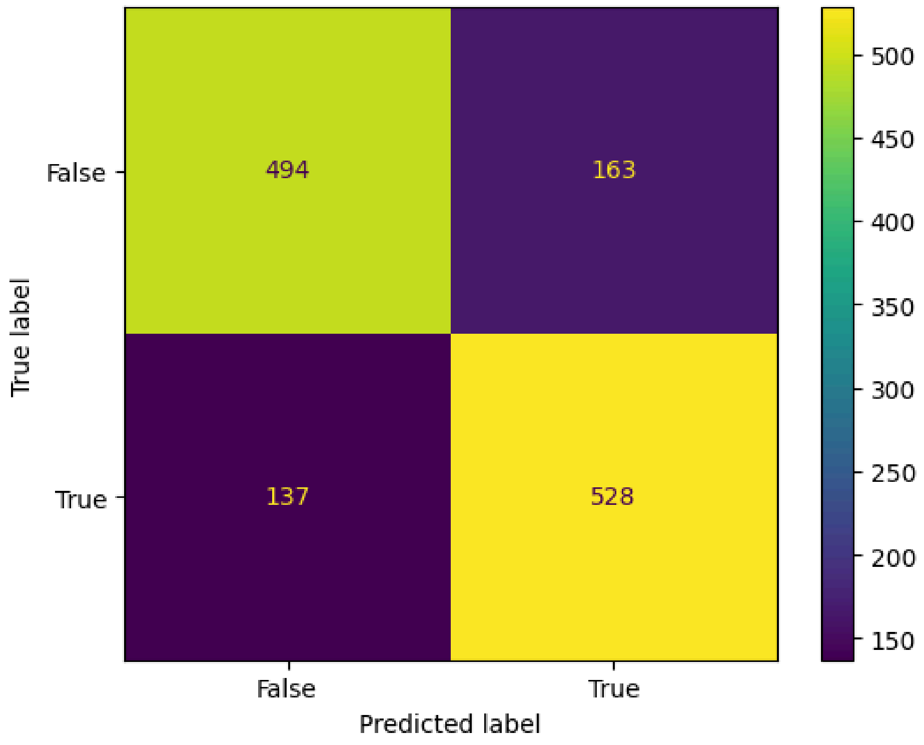


Figure 42: Confusion Matrix for model 3 on X without regularization

Just like we saw for the other two models, the model is slightly biased towards true, but not as much, indicating the neural network model may capture the even distribution of transported and non-transported passengers. Note that the accuracy=0.754, precision=0.764, and recall=0.794, which makes sense as the larger recall ratio than precision indicates the model is slightly more biased towards positive prediction.

Conclusion:

The model that performed the best for logistic regression was the model trained on the square root feature transformed data with L2 regularization and a hyperparameter (c) value of 0.01. In hindsight, training models with no L1 or L2 regularization with differing c values was a futile effort as there is no regularization term for a c value to effect. A trend that we noticed was that the square root feature transformed data generally performed better than the original dataset as well as the squared dataset. This means that the data is likely slightly left skewed, with the square root transformation pushing the dataset closer to a normal distribution. For all the logistic regression models, as the c values increased, so did training accuracy, however, validation accuracy decreased following this same trend. This is evidenced by many of the visualizations shown in the write-up. We speculate that this is likely due to a larger c value putting more weight on the regularization term in the cost function, which in turn would increase overfitting on the noise in the training data. This makes sense considering that training accuracy increased and validation decreased with hyperparameter strength.

Similarly, the best performing SVM model was trained using square root transformed data with a c value of 0.01 and L2 squared regularization. Many of the linear SVM models, similar to the logistic regression models, had a very similar accuracy ranging from around 70 - 80% accuracy on validation data. However, the RBF and polynomial kernel models had very low accuracy of around 50-60% on small c values with the polynomial kernel not showing improvements until a c value of 100. Once again, the best performing model used the square root transformed data and a low c-value which further supports our theory that the data is slightly left-skewed and that high c-values cause the model to overfit on the noise in the training data. For the other kernels, the validation accuracy seemed to increase as the hyperparameter increased which could mean that the regularization parameter needs to be stronger in order to model the patterns in the data.

Furthermore, when analyzing the neural network models, we can see that there is a general trend of less accuracy with an increase in the regularization parameter (c). Although for some models like the third model with the 1000 value regularization parameter and sqrt data we see really high accuracy on the training set, we can see this doesn’t necessarily translate to a better model as the validation accuracy seemed to worsen. This phenomenon is likely overfitting as we see with more complex models like model 3 that there is a trend of increased training accuracy with a decrease in validation accuracy. The more complex model can overfit the data with a higher regularization parameter since its inverse (1/c) is the coefficient of the cost of the regularization, making it less significant, allowing for the model to fit to noise more and thus overfitting.

As aforementioned, the original dataset contained many text features such as HomePlanet, Destination and VIP status. In order to process this data, one-hot encoding was used to turn this data into numerical data and then align was used in order to match up the columns between validation and training data so that the model fitted on the training data could predict using validation data. However, during this process, many of the data columns were dropped as there were many features in the training data that were not present in the validation data. We speculate that many of these data columns that were dropped were the PassengerID and Cabin, however, we were unable to formulate a solution to this problem, so we went ahead and trained on the data that we processed and were still able to produce an accuracy of 80%, so likely these data columns did not have a very significant weight/significance in the overall prediction.

The prediction accuracies were fairly high for all the models with around 80% validation accuracy. There are many possible improvements to our project that could yield better results. One of these improvements is to experiment with lower hyperparameter values for the SVM linear model as well as the logistic regression model as we saw our best performance on the lowest hyperparameter values. Moreover, for the SVM polynomial and RBF kernel models, training on larger hyperparameter values could lead to better results as based on the graphs that we have for these kernel functions, they have not reached an asymptote yet in validation accuracy, so we may get better results training on these values. Another possible improvement is to utilize a better preprocessing method to convert our textual data features to numeric features such as the Bag of Words or the Word2Vec algorithms. Amongst all the models, it seemed that the SVM linear kernel model provided the best results, however, all three of the best models of each type had a very similar validation accuracy of around 80 percent which suggests that all three types of models work fairly well with this dataset.

Works Cited:

Spaceship Titanic Dataset:

https://www.kaggle.com/competitions/spaceship-titanic/data

Sklearn Logistic Regression Documentation:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Sklearn SVC Documentation:

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Feature Transformation Techniques:

https://www.geeksforgeeks.org/feature-transformation-techniques-in-machine-learning/

Feature Importance Function:

https://stackoverflow.com/questions/41592661/determining-the-most-contributing-features-for-svm-classifier-in-sklearn

Keras Documentation (Neural Network):

https://keras.io/api/models/sequential/

Tensorflow Documentation (Neural Network):

https://www.tensorflow.org/guide/keras/functional