Predicting Credit Card Debt Defaulting

Tyler Song, Vaikunthan Mathiyalakan

CS-UY 4563: Intro to Machine Learning

Professor Linda Sellie

April 29, 2025

Introduction

The goal of this project is to train a machine learning model to predict whether or not an individual will default on their credit card debt next month based on various factors. The Kaggle dataset (found here) studies the demographics and payment behaviors among Taiwanese credit cardholders from April 2005 through September 2005. It combines demographic information (age, gender, education, marital status), credit account information, and detailed repayment histories, all culminating in a binary indicator of whether a client defaulted on their payment obligations the following month. In total there are thirty-thousand data points with 25 features each.

Three models were trained on this dataset: a logistic regression model, support vector machine model, and a neural network model. Each model was trained using various hyperparameters using the scikit-learn python machine learning library. Throughout the model analysis, various feature transformations like polynomial transformation, principal component analysis (PCA), and radial basis functions (RBF) were implemented to test performance. Additionally, various hyperparameters like the value of the regularization constant, number of principal components, network layer amount and the size of each network, were used to add additional analysis to the model. The goal was to find the best prediction model to determine the likelihood that a given individual would default on their next month's credit payment.

Data Preprocessing & Unsupervised Analysis

The dataset had a few key issues for the purposes of the project. Notably, categorical features like marital status and education level initially caused the model to treat their integer labels as ordinal or continuous—imposing false distances and relationships that bias and degrade performance. Initially the education variable could have taken one of six different values and the marriage variable could have taken one of three different values. One-hot encoding was used to split the education, marriage, and sex variables into six, three, and two different binary indicators, respectively. Four of these new categorical features, EDUCATION_4, EDUCATION_5, and EDUCATION_6, and MARRIAGE_3, were massively imbalanced with less than 1% of the total data being positive for each of those categories. To avoid any overfitting, the column MARRIAGE_3 (representing 'other') was removed. Columns EDUCATION_4-6 were combined to form one feature. In total there were 28 features to predict whether an individual would default on their payment next month

Additional general preprocessing included dropping the index column and using sklearn's StandardScalar to scale the data. StandardScalar standardizes features by using the variable's mean and standard deviation in the z-score equation:

$$z = \frac{x-\mu}{\sigma}$$

Equation 1: StandardScalar Equation

After one lengthy model training, the amount of data being trained on was lowered from thirty-thousand to one-thousand to match hardware and time constraints. 250 points were delegated as test data, while the other 750 were taken as training data.

Various unsupervised analysis methods were used to better understand the data. No test data was observed. To start, 3 points were removed due to incomplete data. In the 997 remaining data points, 761 individuals (majority class) did not default on their debt while 236 did (minority class). Within the training set, 573 individuals did not default on their debt while 174 did. There was not any kind of reweighting or resampling of features to better model the data because the difference in size between the majority and minority class was not perceived to be substantial enough to warrant such action. After preprocessing the data, individual feature distributions were plotted. For discrete categorical variables, the average value in the default vs. non-default groups were displayed for each possible value (i.e. SEX_1, SEX_2, etc).

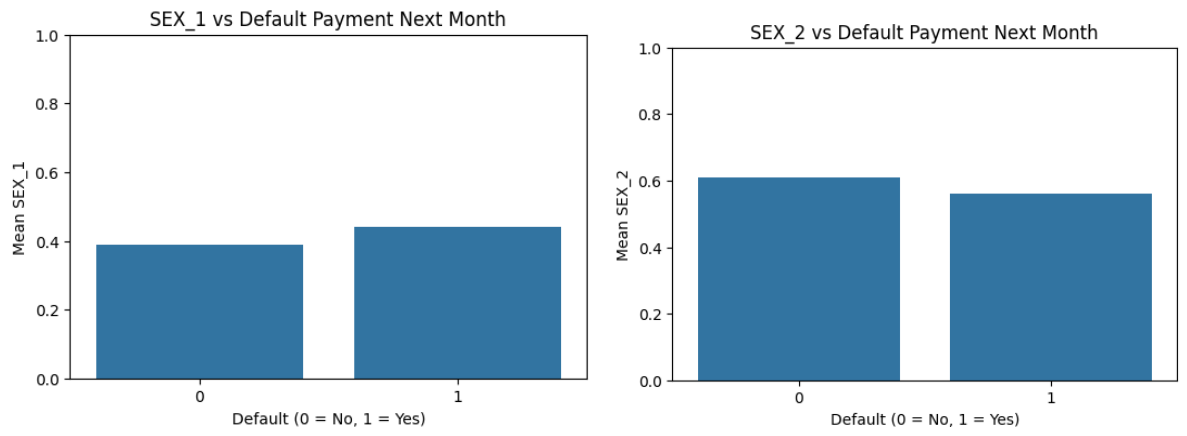

Figure 1: Mean Sex vs Default Class (Sex_1 male, Sex_2 female)

For continuous variables, the individual feature distributions (ex: Figure 2) and a boxplot graph of their relationship with the next-month default status target variable (ex: Figure 3).

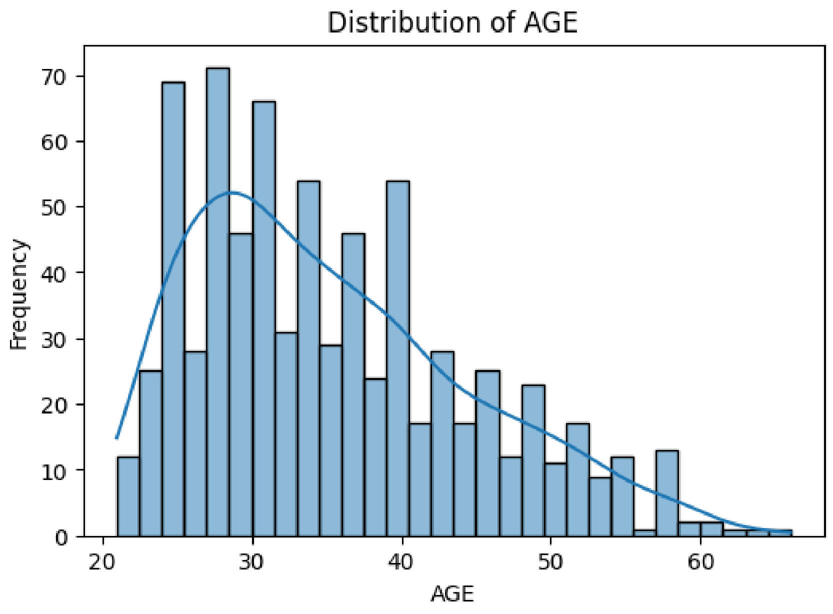

Figure 2: Distribution of Age of Credit Card Users

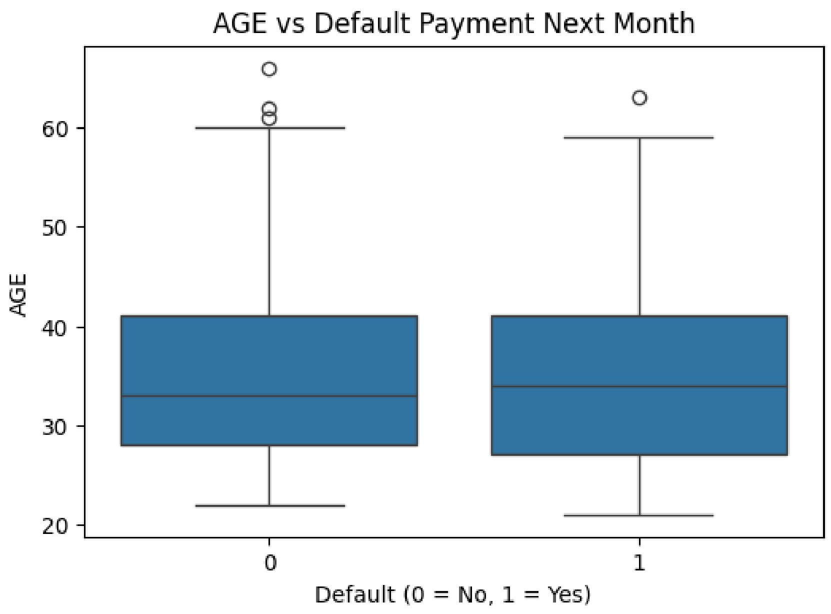

Figure 3: Distribution of Age of Credit Card Users

After analyzing the individual distributions of the features, the correlation between all the variables were measured (Figure 4).

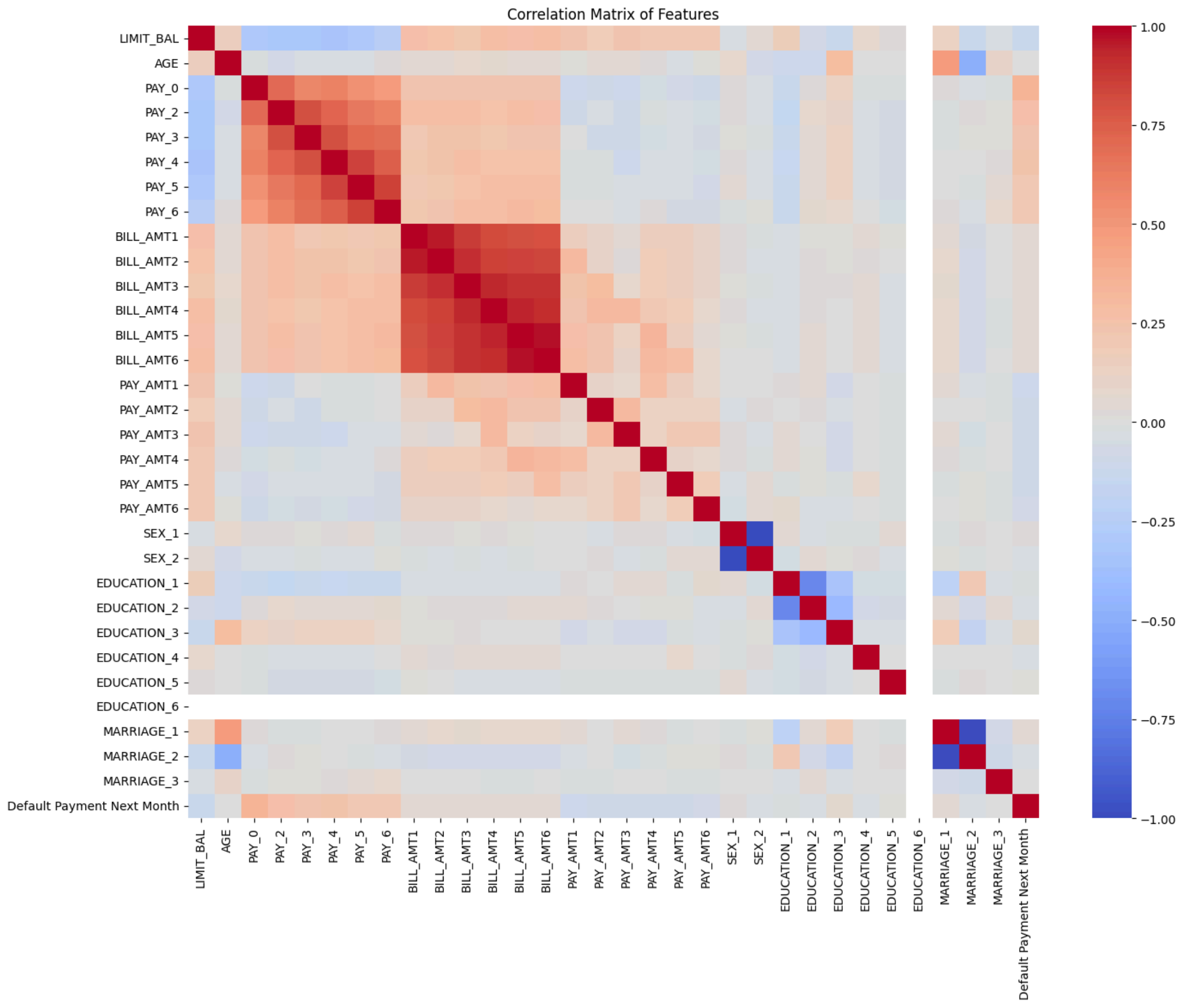

Figure 4: Correlation Map of Dataset

The graph illustrates that all repayment-status variables—from on-time payments to multi-month delays—are strongly intercorrelated, and the billing-amount variables exhibit similarly high mutual correlations. The pay variables in particular have the strongest correlation with the default payment for the next month. Intuitively this is likely because if an individual misses payments one month, they risk accumulating more debt throughout the next month which makes paying their next bill even more difficult.

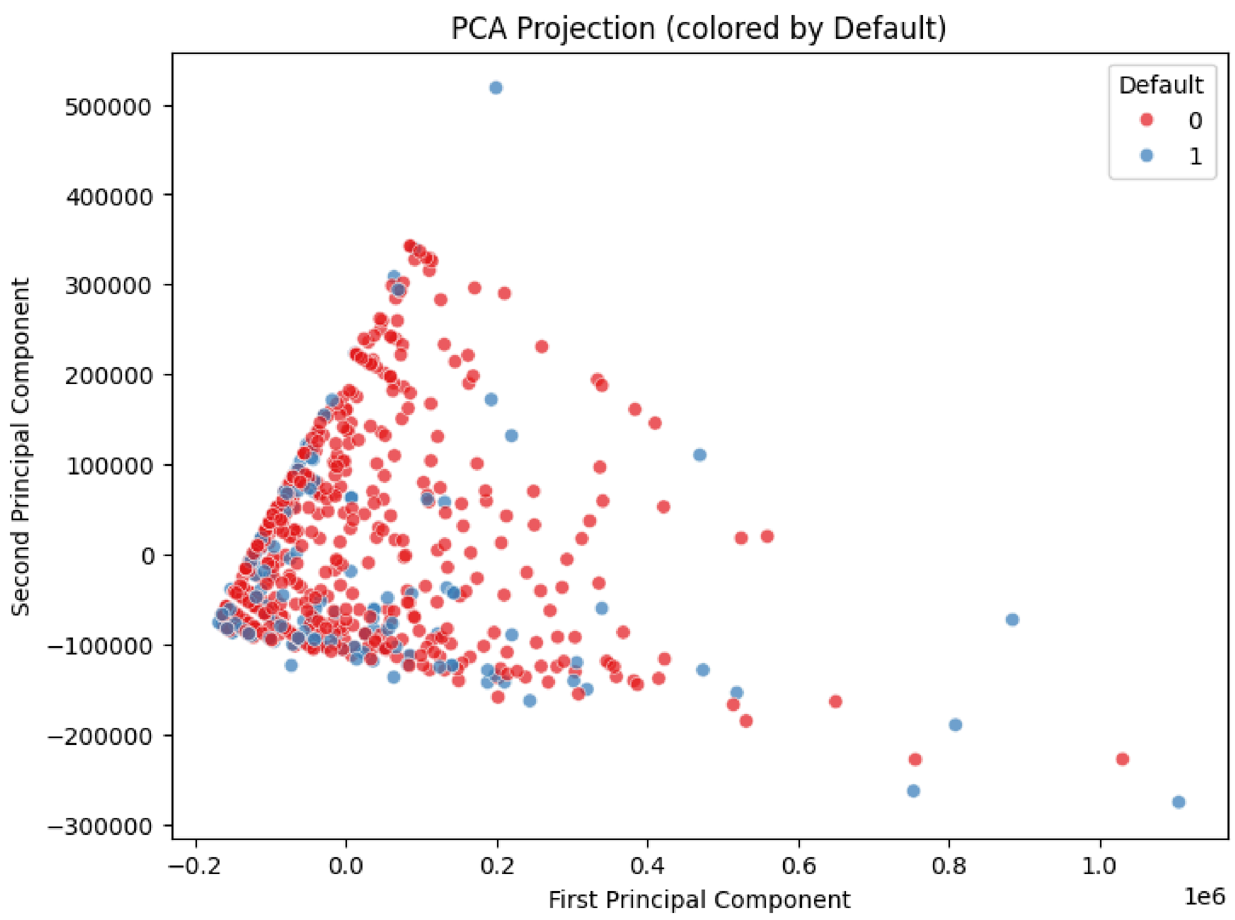

Figure 5: Principal Component Analysis Map with Two Components

The scatterplot shows each client projected onto the first two principal components—linear combinations of the original features that capture the greatest variance—but because the raw variables weren't standardized, the first and second principal components take on very large values dominated by the highest-scale inputs. There is a high degree of mixture between the positive and negative classes which suggests that there is not a clear gap or cluster that can separate the points. This lack of separability prompts the need for feature engineering and a well-trained supervised model to calculate default risk. In simple terms, principal component analysis alone won't clearly distinguish defaulters from non-defaulters, so additional methods were needed to separate the two groups.

Logistic Regression

Logistic regression was used as a baseline model to predict default rates. The *grid_evaluate* function in *utils.py* was used to take in:

1. *estimator* - a particular classifier (logistic regression in this case)
2. *param_grid* - an object with lists of values for each of the feature transformation, degree, number of component, gamma, C, and penalty parameters (some parameters may not always be used)
3. *X_train*
4. *X_test*
5. *Y_train*
6. *Y_test*

Each hyperparameter was tested using a range of values. Feature transformations were either linear (no transformation), polynomial (of degrees 2 and 3), used the radial basis function transformation (RBF), or used the principal component analysis transformation (PCA). The gamma parameter for RBF transformations was taken to be 0.1 or 0.5. The amount of principal components were taken to be 5 or 10, chosen based on unsupervised analysis. C values were chosen at [0.001,0.01, 0.1, 1.0, 10.0, 100.0].

The following dataframe displays the top 5 performing hyperparameters based on F1 validation score[1].

Index	Feature	Degree	PCA Components	Gamma	C	Penalty
1	Polynomial	3	NA	NA	0.01	l2
2	Linear	1	NA	NA	10	l2
3	Linear	1	NA	NA	100	l2
4	Polynomial	2	NA	NA	0.1	l2
5	Linear	1	NA	NA	1	l2

Figure 6.1: Parameters for Best Models by F1 Validation Score for Logistic Regression

Index	Accuracy (val)	Accuracy (train)	Precision (val)	Precision (train)	Recall (val)	Recall (train)	F1 (val)	F1 (train)
1	0.764	0.904953	0.534884	0.963964	0.370968	0.614943	0.438095	0.750877
2	0.792	0.815261	0.678571	0.730769	0.306452	0.327586	0.422222	0.452381
3	0.792	0.815261	0.678571	0.730769	0.306452	0.327586	0.422222	0.452381
4	0.792	0.815261	0.678571	0.730769	0.306452	0.327586	0.422222	0.452381
5	0.764	0.904953	0.534884	0.963964	0.370968	0.614943	0.438095	0.750877

Figure 6.2: Classification Metrics for Best Models by F1 Validation Score for Logistic Regression[2]

[1] Please see the attached logistic_regression_results.csv which contains performance results for all models that were tested

[2] Due to the large table of data, an index column was created to match the entries of figure 6.1 and figure 6.2 to the same data point in the set

Additionally, various graphs were produced to evaluate the performance of the model:

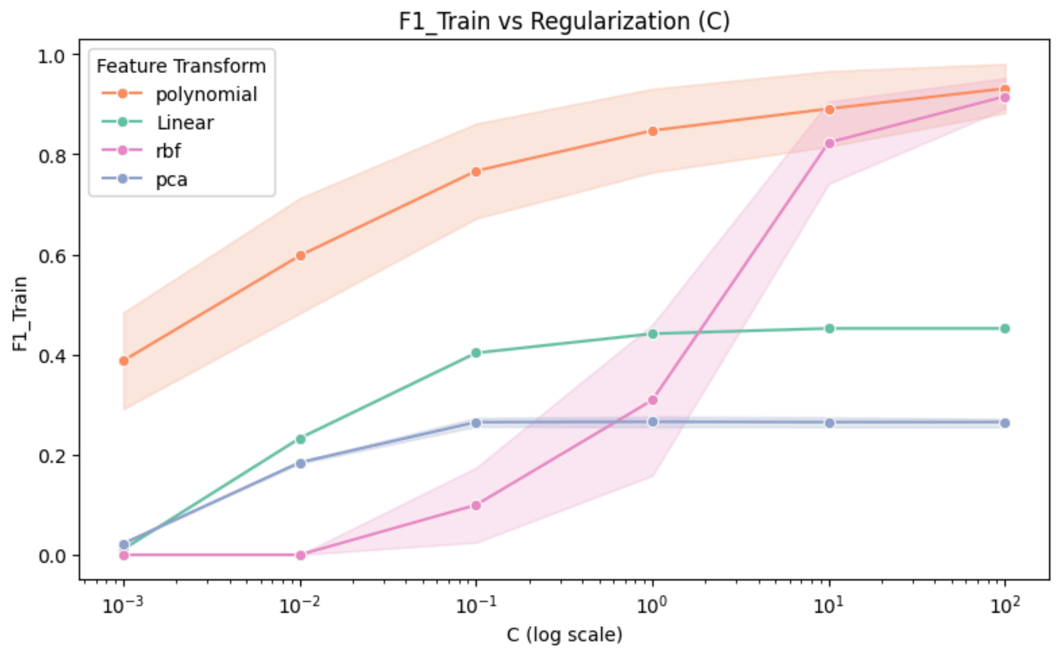

Figure 8: Logistic Regression F1 Train Score vs. C for Different Transformations

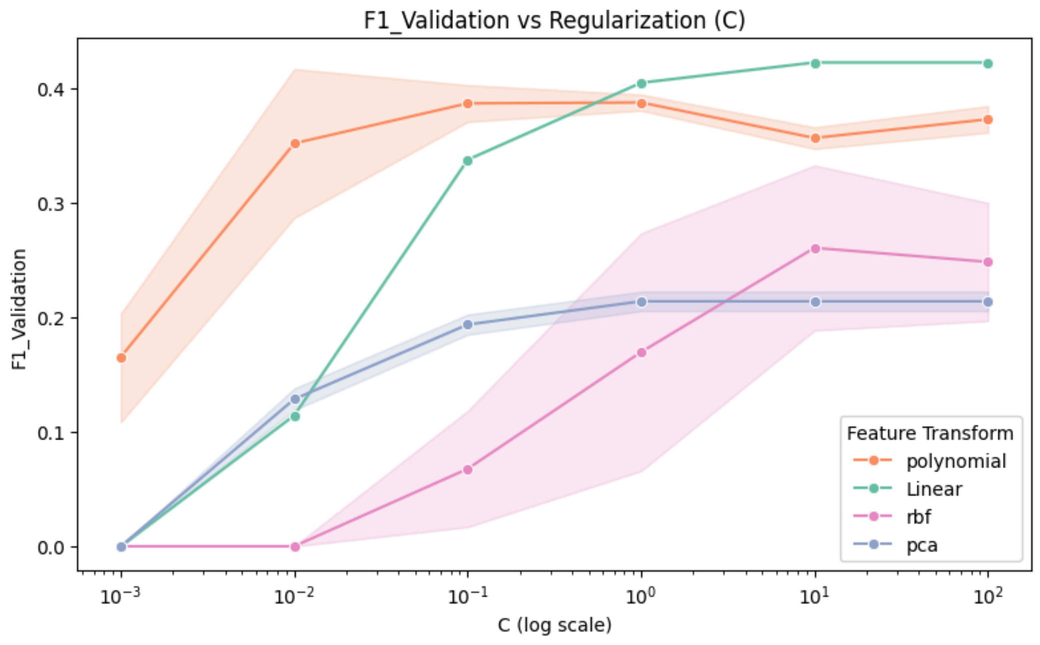

Figure 7: Logistic Regression F1 Validation Score vs. C for Different Transformations[3]

[3] Due to maintaining reasonable space, please visit the code on logistic regression, support vector machines, and neural networks to view all the classification metrics in the training and test set

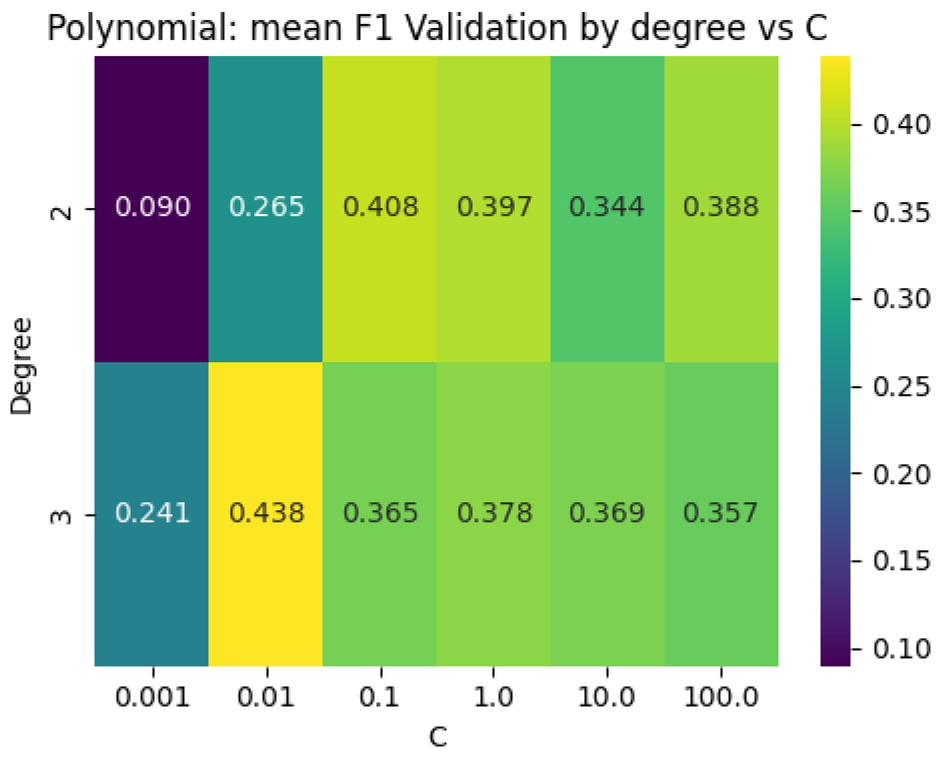

Figure 9: Logistic Regression F1 Validation Score By Degree vs. C for Polynomial Transformation

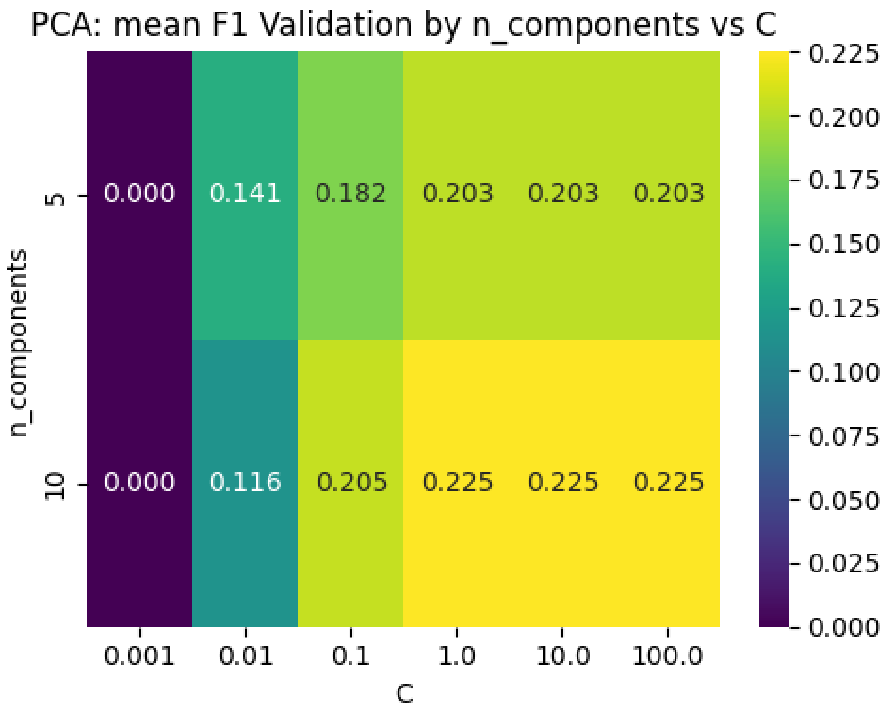

Figure 10: Logistic Regression F1 Validation Score By Component Amount vs. C for PCA

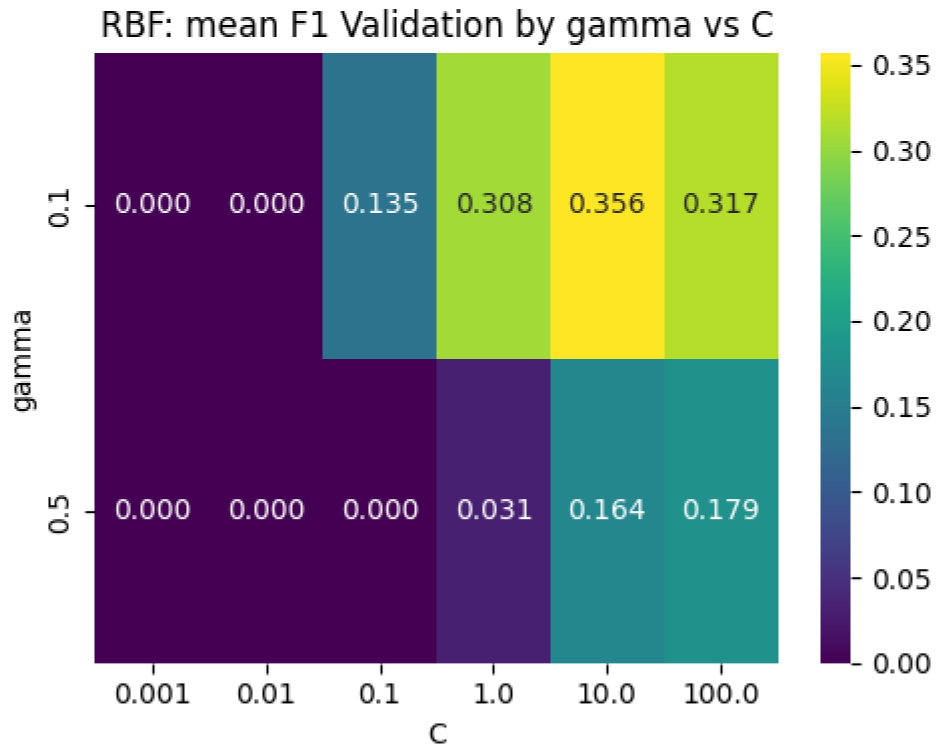

Figure 11: Logistic Regression F1 Test Score By Gamma vs. C for RBF

While accuracy, precision, and recall were all used to understand the model, the F1 score of each model was used to determine performance because it factored in both precision and recall. Accuracy was viewed with a grain of salt due to the moderate imbalance between the number of negative examples (764) and the number of positive examples (236). Hence, when polynomial transformation was applied to logistic regression with a degree 3, the F1 score in the validation set was the highest of all logistic regression models at approximately 0.438.

Support Vector Machine

A support vector machine (SVM) was also used for this binary classification problem because of its ability to deal with non-linear or high-dimensional data due to its use of kernel functions and margin maximization. However, it is important to note that while svm's can fit better to the data, logistic regression is oftentimes faster, more interpretable, and better suited for large datasets. For this model, the same *grid_evaluate* function was used to evaluate the SVM model across a variety of hyperparameters. The hyperparameters used to evaluate the SVM model were the same used to evaluate the logistic regression model.

Index	Feature	Degree	PCA Components	Gamma	C	Penalty
1	Polynomial	2	NA	NA	0.01	l2
2	Polynomial	3	NA	NA	10	l2
3	Polynomial	3	NA	NA	100	l2
4	Polynomial	3	NA	NA	0.1	l2
5	Polynomial	3	NA	NA	1	l2

Figure 12.1:Parameters for Best SVM Models by F1 Test Score

Index	Accuracy (val)	Accuracy (train)	Precision (val)	Precision (train)	Recall (val)	Recall (train)	F1 (val)	F1 (train)
1	0.76	0.864793	0.523810	0.861386	0.354839	0.500000	0.423077	0.632727
2	0.76	0.906292	0.523810	0.964286	0.354839	0.620690	0.423077	0.755245
3	0.668	0.981258	0.356164	0.993827	0.419355	0.925287	0.385185	0.958333
4	0.636	0.967871	0.329412	0.898936	0.451613	0.971264	0.380952	0.933702
5	0.66	0.977242	0.346667	0.943503	0.419355	0.959770	0.379562	0.951567

Figure 12.2: Classification Metrics for Best SVM Models by F1 Validation Score[4]

[4] Similar to Figures 6.1 and 6.2, an index column was created to match parameters to the same data point

Additionally, various graphs were produced to evaluate the performance of the model:

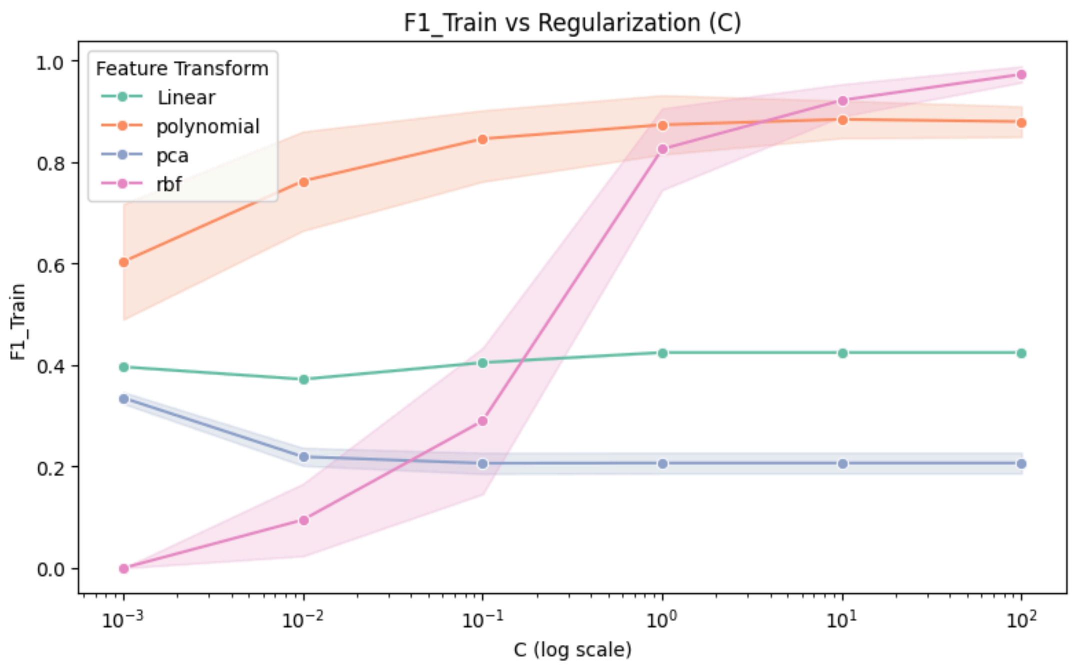

Figure 14: SVM F1 Training Score vs. C for Different Feature Transformations

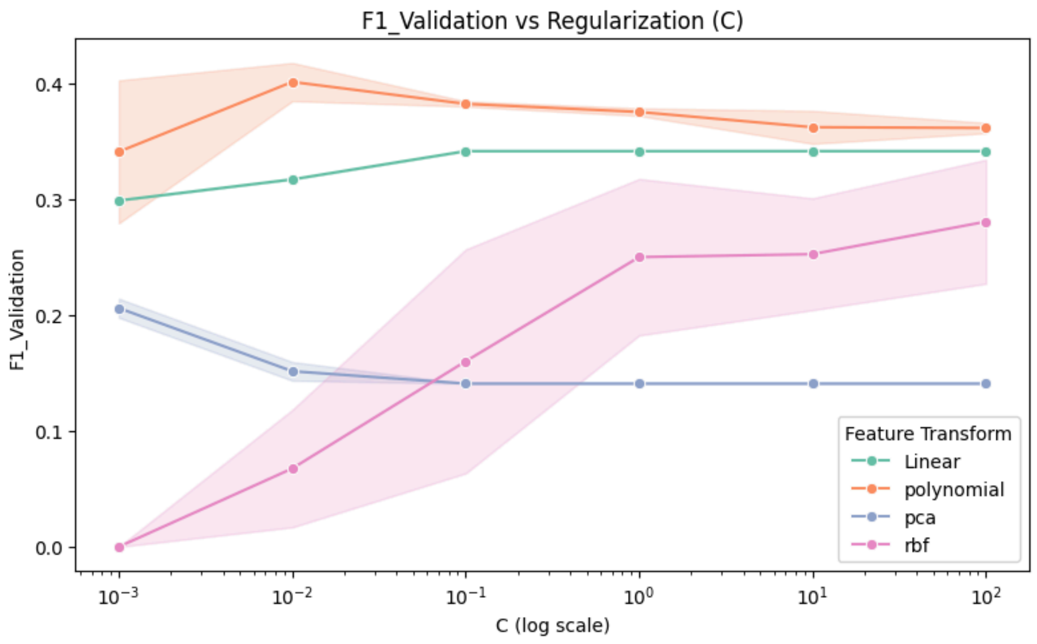

Figure 13: SVM F1 Validation Score vs. C for Different Feature Transformations[5]

[5] Due to maintaining reasonable space, please visit the code on logistic regression to view all the classification metrics in the training and test set

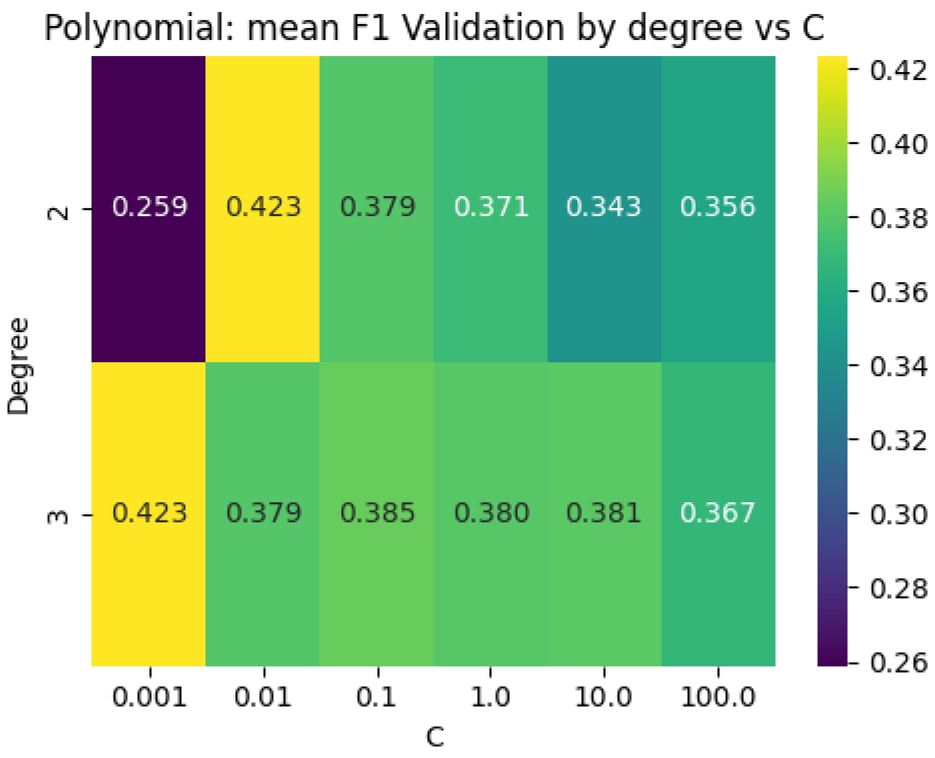

Figure 15: SVM F1 Validation Score By Degree vs. C for Polynomial Transformation

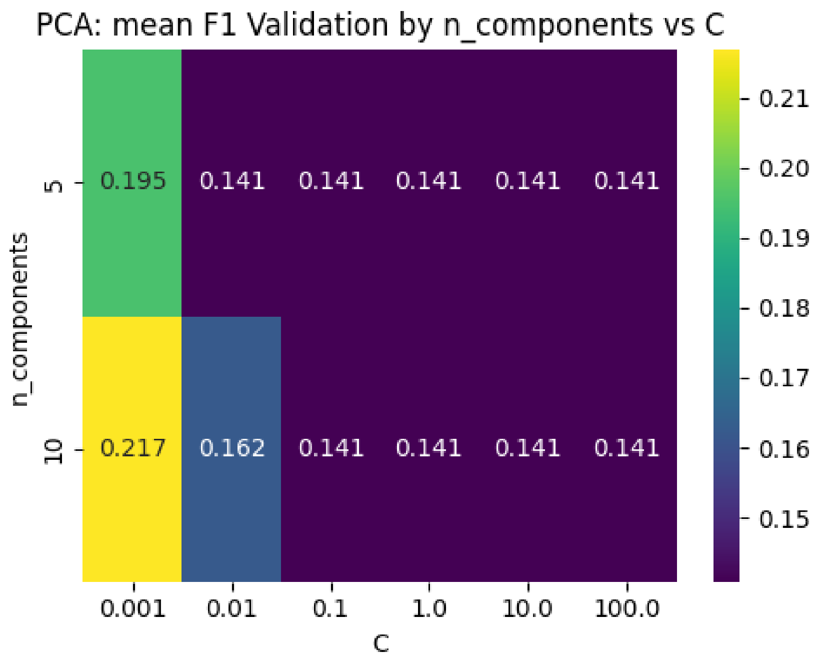

Figure 16: SVM F1 Validation Score By Component Amount vs. C for PCA

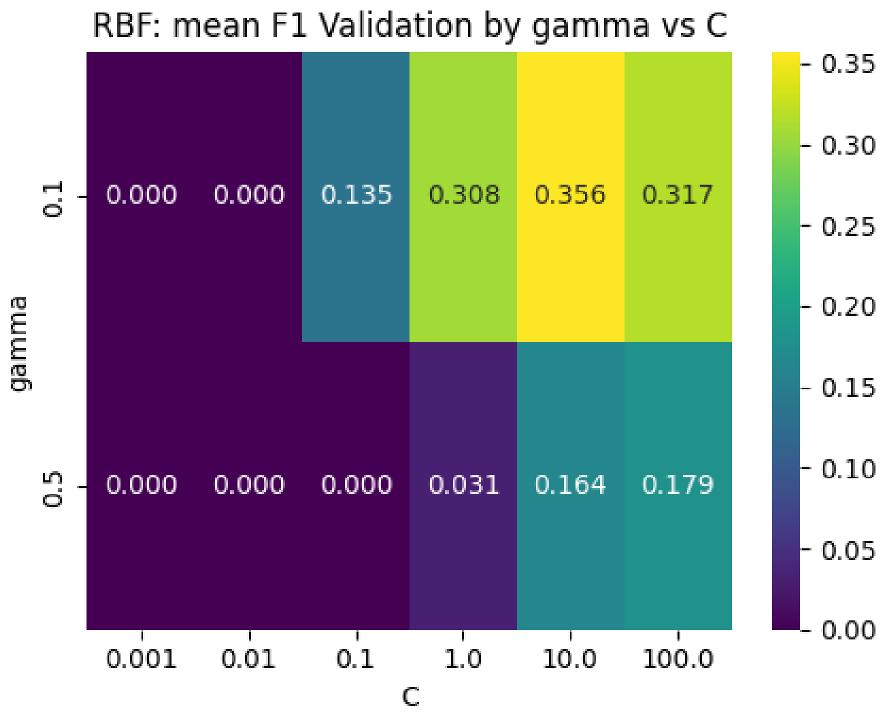

Figure 17: SVM F1 Test Score By Gamma vs. C for RBF

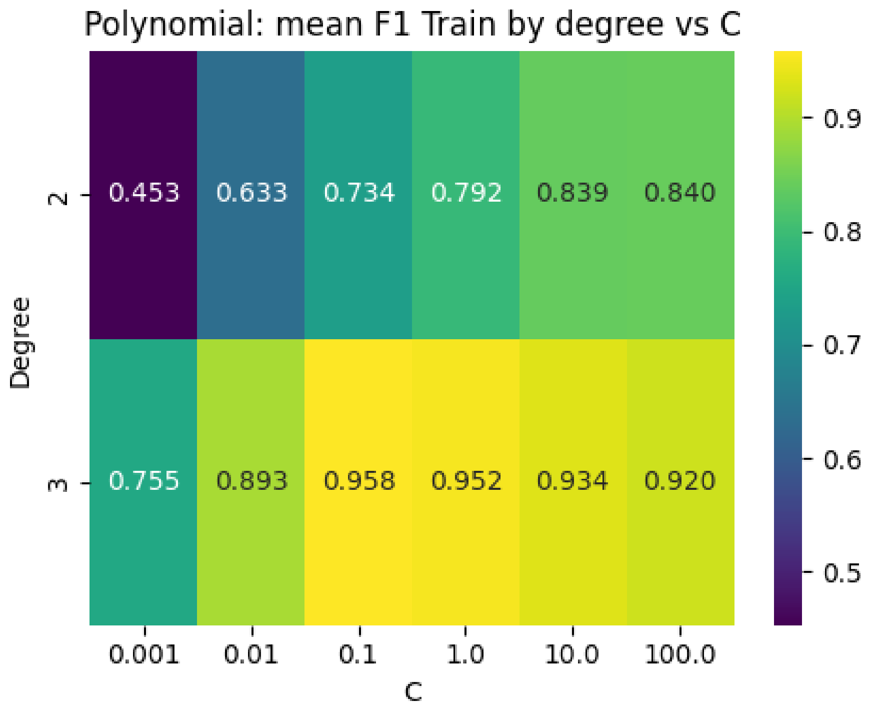

Figure 18: SVM F1 Train Score By Degree vs. C for Polynomial Transformation

Ultimately, the model with degree 2 polynomial transformation and the smallest C value of 0.01 (strong regularization) had the highest F1 validation score, and thus was deemed to be the best model out of the different SVM models tested.

Neural Network

The hyperparameters/transformations experimented with for the neural network model were different from the hyperparameters and transformations experimented with for the SVM and logistic regression models. The same function, *grid_eval*, was used. The parameters changed included: activation function for the hidden layer, alpha value (representing the L2 regularization rate), learning rate initialization value, and hidden layer structure. Two activation functions were experimented with, Rectified Linear Unit (ReLU) and tanh. Alpha values were chosen at [0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1]. Three types of neural network architectures were used: one hidden layer with 15 nodes, one hidden layer with 20 nodes, and two hidden layers with 20 nodes each. The top 5 results ranked by F1 score are shown in Figures 19.1 and 19.2.

Index	Alpha	Hidden Layer Structure	Accuracy (val)	Learning Rate Init.	Activation
1	0.001	(15, 15)	0.728	0.01	ReLU
2	0.01	(15, 15)	0.724	0.001	ReLU
3	0.001	(15, 15)	0.728	0.001	ReLU
4	1.00E-05	(15, 15)	0.728	0.001	ReLU
5	1.00E-06	(15, 15)	0.724	0.001	ReLU

Figure 19.1: Neural Network Models with Highest F1 Score from Parameter Analysis

Index	Accuracy (train)	Precision (val)	Precision (train)	Recall (val)	Recall (train)	F1 (val)	F1 (train)
1	0.973226	0.448276	0.942529	0.419355	0.942529	0.433333	0.942529
2	0.961178	0.438596	0.961783	0.403226	0.867816	0.420168	0.912387
3	0.963855	0.444444	0.974194	0.387097	0.867816	0.413793	0.917933
4	0.963855	0.444444	0.962264	0.387097	0.879310	0.413793	0.918919
5	0.959839	0.436364	0.973684	0.387097	0.850575	0.410256	0.907975

Figure 19.2: Neural Network Models with Highest F1 Score from Parameter Analysis

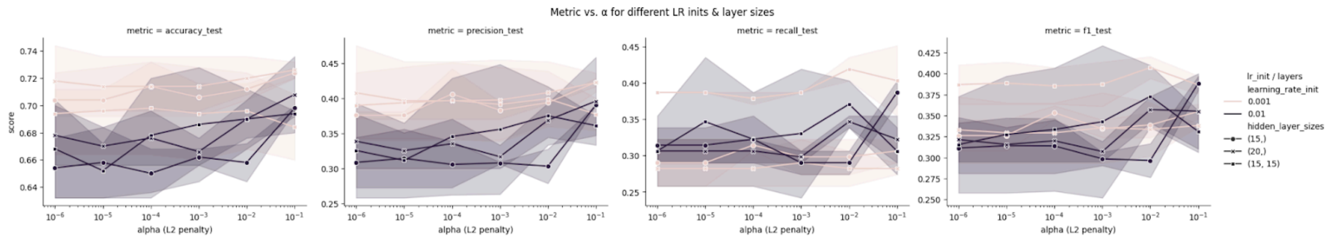

Figure 20: Metrics vs Different Alpha Values for the Validation Set

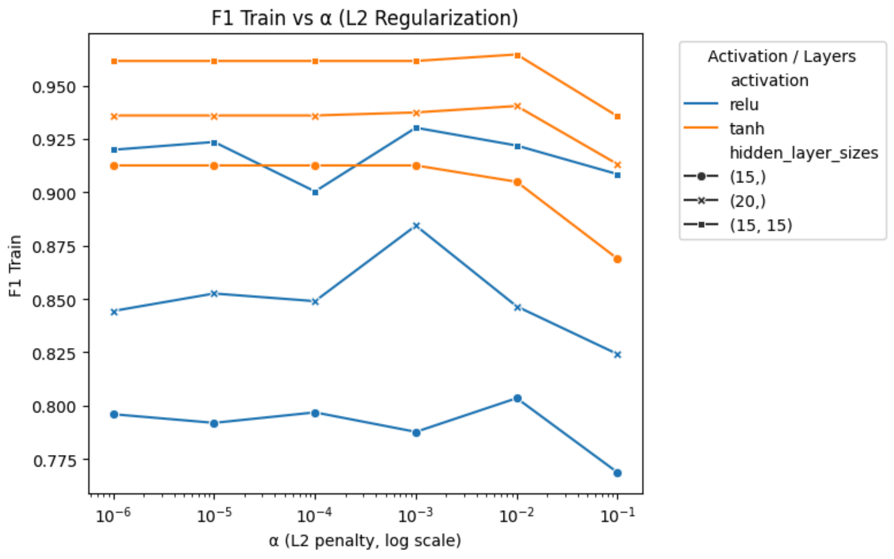

Figure 21: Training F1 Score vs Alpha Values for Validation and Training Set

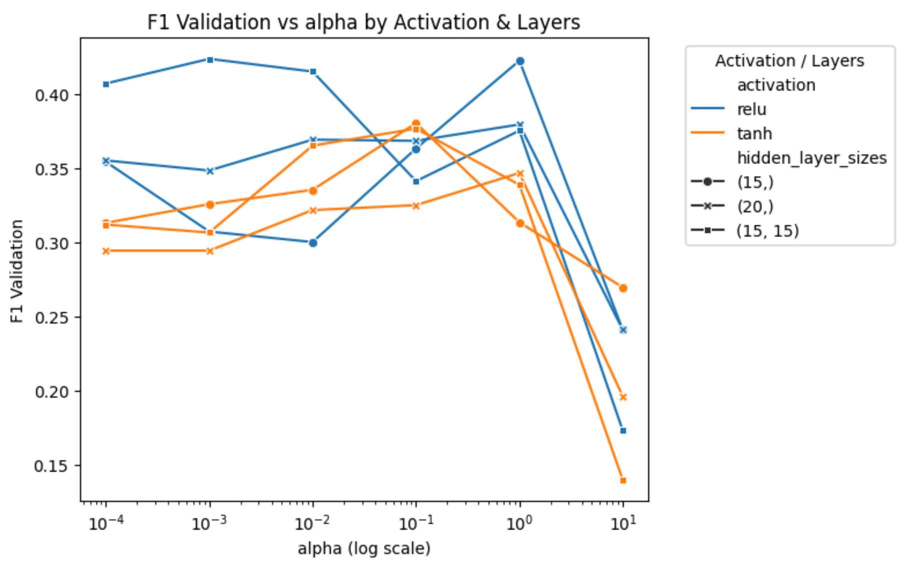

Figure 22: Training F1 Score vs Alpha Values for Validation and Training Set

Looking at the data, the neural network was overfitting strongly to the training set. As a result, the range of alphas was changed to [0.0001, 0.001, 0.01, 0.1, 1, 10] to increase the amount of regularization and improve generalization. The new data is plotted below:

Index	Alpha	Hidden Layer Structure	Learning Rate Init.	Accuracy (val)	Activation
1	0.001	(15, 15)	0.01	0.728	ReLU
2	1	(15,)	0.001	0.78	ReLU
3	0.01	(15, 15)	0.001	0.724	ReLU
4	0.001	(15, 15)	0.001	0.728	ReLU
5	1	(15,)	0.01	0.76	ReLU

Figure 23.1: Neural Network Models with Highest F1 Score from Parameter Analysis after Regularization Increase

Index	Accuracy (train)	Precision (val)	Precision (train)	Recall (val)	Recall (train)	F1 (val)	F1 (val)
1	0.973226	0.448276	0.942529	0.419355	0.942529	0.433333	0.942529
2	0.840696	0.600000	0.816092	0.338710	0.408046	0.432990	0.544061
3	0.961178	0.438596	0.961783	0.403226	0.867816	0.420168	0.912387
4	0.963855	0.444444	0.974194	0.387097	0.867816	0.413793	0.917933
5	0.859438	0.525000	0.828571	0.338710	0.500000	0.411765	0.623656

Figure 23.2: Neural Network Models with Highest F1 Score from Parameter Analysis after Regularization Increase

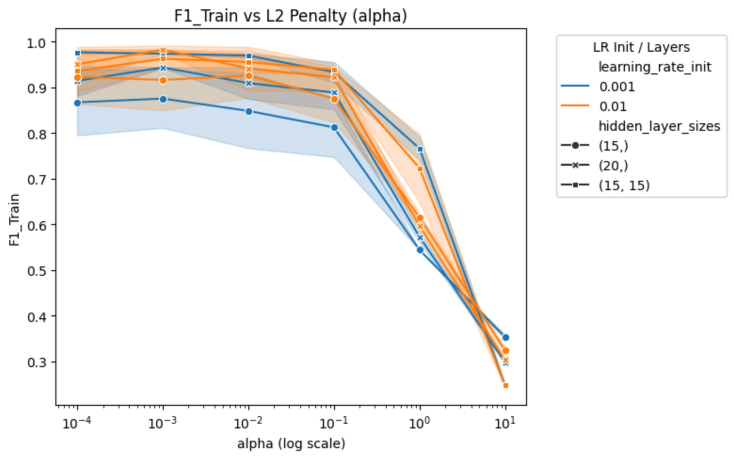

Figure 24: Neural Network F1 Scores vs Alpha Values for Training Set after Regularization Increase

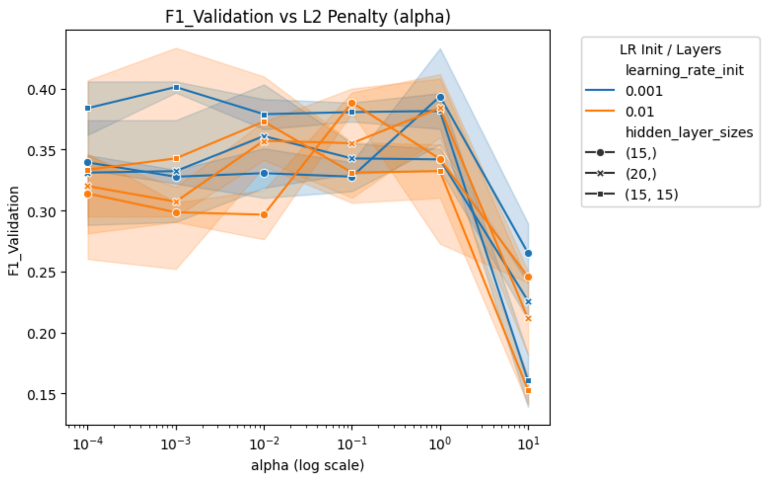

Figure 25: Neural Network F1 Scores vs Alpha Values for Validation Set after Regularization Increase

By increasing the regularization, on average, the F1 test score went up while the F1 train score went down. Interestingly, the top model stayed the same. Additionally, heat maps were graphed comparing alpha vs initial learning rate:

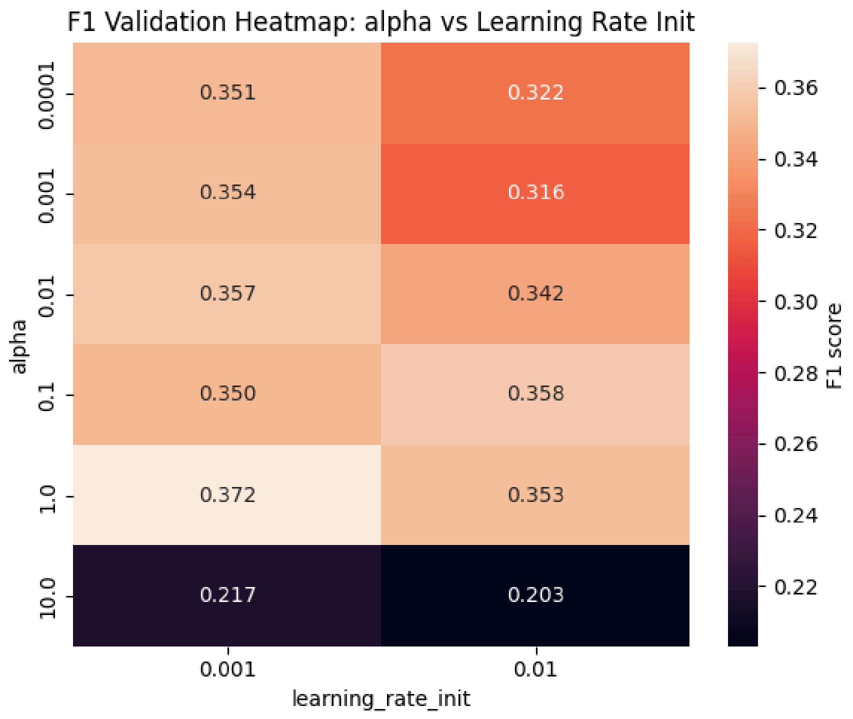

Figure 26: F1 Validation Scores Matrix By Alpha vs Learning Rate Initial Value

Test Set Performance

After observing the performances of logistic regression, SVM, and neural network models on the training and validation sets, the best models (as determined by F1 score on the validation set) were benchmarked on the test set. Below is a summary of performance of each of the models[6]:

[6] For information about the hyperparameters of each of the models, please locate the best performing model by F1 score on the validation set in each of their respective sections.

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.852	0.7647058824	0.4727272727	0.5842696629
SVM	0.836	0.6590909090	0.5272727272	0.5858585858
Neural Network	0.792	0.5263157895	0.5454545455	0.5357142857

Figure 26: Metrics Table for the Best Hyperparameters of Each Model Type

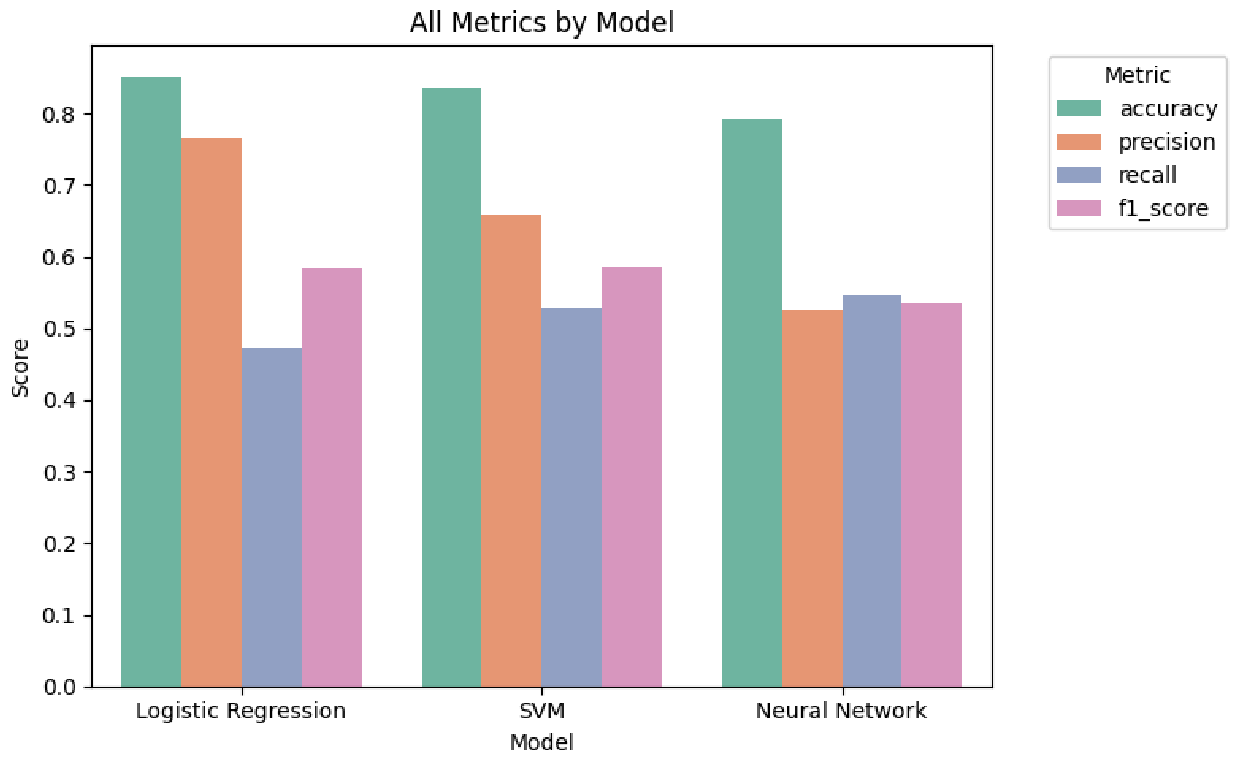

Figure 27: Key Metrics of Each Model on the Test Set

Conclusion & Analysis

The model that best performed on an unseen test set was the SVM model with a 2nd degree polynomial feature transformation with a C of 0.01. The model had an F1 score of about 0.586 and an accuracy of 0.836. This could be either because of noise or intrinsic model structure. It is plausible that the SVM was able to effectively maximize margin, especially given the 2nd degree polynomial feature transformation which allowed the SVM to create flexible nonlinear boundaries. The SVM can strike a balance between complexity and regularization which could be good in this complex dataset that requires fitting to the data but also generalizable performance. Another plausible reason the SVM performed best is due to random chance. The top performing SVM and logistic regression models had extremely similar F1 scores, with the neural network model not far behind. This indicates that the three models were able to model the data at similar levels. If the other data points were sampled, or if an another random seed were used, it is entirely plausible that the SVM and neural network perform drastically differently. The logistic regression model would be most likely to hold its current classification metrics.

It's important to note that, as mentioned earlier, accuracy is not the best predictor of model performance due to the target class imbalance. Also, the noise level of F1 with a smaller positive class is high. Since there are smaller amounts of positives, mispredictions of positive class data points have large effects on the F1 score.

Broadly, across all three models, a few key trends were evident. Overfitting was a consistent problem across all three models. Despite this, it was often that lower to middling regularization (for instance a C of 0.01 in the best performing model) performed the best. This C value indicates that moderate regularization was needed to allow each of the models to capture complex relationships but not shift the weights of the model too high. Analyzing the top performing models across the three different architectures used, polynomial transformation seemed to be most prominent followed by no transformation. Principal component analysis and radial basis functions were particularly ineffective at generating effective models for a variety of reasons. As seen through figure four, there were approximately five clusters that were visibly correlated with each other. These clusters being the various PAY_i features with each other, the

BILL_AMT_i features with each other, the PAY_AMT_i features with each other, the correlation between limit balance and age, and the correlation between BILL_AMT_i and PAY_i. In figure 5, the PCA graph displayed very intermingled data points between the various zero and one values as the two principal components failed to meaningfully separate data points. This lack of separability prompts the need for more features and helps explain why polynomial transformation was successful but PCA and RBF were not. A degree two and three polynomial transformation was introduced in an attempt to capture the non-linear relationships between variables. PCA simplified the model down into clusters which were just shown (in the extreme case of two principal components) to not provide an effective boundary. RBF likely struggled due to potentially too high of a gamma value. The RBF kernel consistently underperformed at $\gamma = 0.1$ and 0.5, likely because those gamma values caused the model to fit noise rather than the true decision boundary. The poor performance of the RBF kernel indicates that γ was potentially too high and lower values should have been tested.

The top performing logistic regression model was a 3rd degree polynomial transformation with a C of 0.01. This is interesting since the other top performing logistic regression models were mostly degree 2 or linear transform models. In general, linear models outperformed the other types of feature transformations, most likely due to better generalization. Although the linear model achieved slightly lower F1 than the degree-3 polynomial, its relatively smaller gap between training and validation sets demonstrated strong robustness under class imbalance. As model complexity increased (e.g., higher polynomial degree), the optimal C decreased, reflecting the need for stronger regularization in the more flexible model. The second best logistic regression model only lagged behind the top model by a slightly smaller F1 Validation score, but had a much smaller gap between F1 training and validation scores. This indicates that this second best model generalizes to the data set better. A possible reason why the degree 3 model outperformed the models with better generalization is that despite the overfit, the degree 3 was still better able to match the complexity of the data.

The best SVM model had an F1 score of about 0.586 on the test set. It had a 2nd degree polynomial feature transformation with a C of 0.01. The best model exhibited slight overfitting, with a gap between the F1 training and validation scores. The polynomial transformation outperformed all other transformation types. Linear transformations had similar but lower performance. RBF and PCA transformations seemed ineffective in improving model accuracy. The C value for the best model was 0.01. Increasing the C value past 0.01 had small effects on the F1 validation score, except for the RBF transformation, whose F1 score steadily increased as C did. Ultimately, there did not seem to be a strong relationship between C value for performance in SVM rather than feature transformation type. This is likely because the number of inputs and relationships the model can investigate was much more important in this scenario than the particular size of the weights.

The best performing neural network model had an F1 score of about 0.536 on the test set. It had an alpha of 0.001, two hidden layers of 15 nodes, and an initial learning rate of 0.01. The two layer structure indicates that the decision surface is complex and benefits from deeper architectures. However, the high difference between the F1 training score and F1 validation score is strongly indicative of overfitting. Despite this, this model performed the best on the validation set. The second best model, with a stronger regularization term of 1 and a simpler structure (one 15 node hidden layer) most likely generalizes better to the data despite having a lower F1 score on the validation set. Initial learning rate seemed to have a small impact on the overall performance of the model. The most interesting trend seen across experimentation with neural networks is the trend towards more complex models. 3 out of the top 5 performing neural networks had two hidden layers. This reinforces the idea that the data set is probably linearly inseparable. The balance between overfitting in an attempt to model the complexity and underfitting for better generalization was volatile.

One strength of the analysis was through the grid search methodology that exhausted every combination of parameters. Rather than tweaking parameters one at a time and siloing the model to find the best possible parameters constrained to the order of parameters that were examined, the *grid_evaluate* function examined all possibilities to explore and provide breadth and rigor to the analysis. There were some limitations of this approach, including high computation time and less precise tweaks of hyperparameters. Many possible improvements could be made to create a more accurate model. The imbalance between the two classes may have had a larger impact on the models than expected. Resampling techniques or class weights can be implemented in future attempts to rectify this issue. Additionally, with more time or stronger hardware, more hyperparameters could have been tested. Hardware and time limitations means that a select few hyperparameters were able to be tested. The grid search function used also left room for optimization. Combinations of transforms and hyperparameters created redundant data (e.g. running a model with different gamma values without using a RBF transformation) that caused unnecessary iterations. Data also had to be cleaned to remove redundant hyperparameter combinations. Lastly, stronger feature engineering may have helped with linear separability. With stronger domain expertise, feature transformations could have been better tailored to the data set.

Works Cited

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

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

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

https://scikit-learn.org/stable/modules/neural_networks_supervised.html

https://cs229.stanford.edu/notes2020fall/notes2020fall/CS229_ML%20advice_presented-slides.pdf

https://matplotlib.org/stable/users/explain/colors/index.html#

https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset/code

Link to GitHub:

https://github.com/tsong28/ML-Final-Project