CS-UY 4563: Machine Learning

Final Project Written Report

Heart Attack Prediction

April 29, 2024

Professor Linda N. Sellie

Thu Vu, Ariel Wang

Introduction

This project focuses on the development of predictive models to identify the risk factors associated with heart attacks using both supervised and unsupervised machine learning techniques. For the supervised analysis, we implemented models including Logistic Regression, Support Vector Machines (SVM), and Neural Networks. These models were trained to predict heart attack occurrences based on a set of labeled data points. In the realm of unsupervised learning, we employed the k-means clustering algorithm to uncover inherent groupings and patterns within the data that could signify hidden risk factors. Additionally, we incorporated various feature transformation and regularization methods to enhance the performance and generalization ability of our models. The specifics of the feature transformations and regularization strategies employed will be discussed further in subsequent sections of this report.

Data Preparation

Given the nature of the collected dataset, no encoding techniques were required. Both MinMax and StandardScaler were experimented and we chose to use StandardScaler as it returns a significantly higher accuracy. The data was divided into a training set, a validation set, and a testing set, depending on which model is being trained. Specifically, for logistic regression, we divided our dataset into 80% training set and 20% test set. For SVM, the percentages of data allocated to training, validation, and testing sets are 86.49%, 6.51%, and 7%. And for Neural Network, the splits are 42%, 18%, and 40%.

1. Logistic Regression

Given that this is a binary classification problem, Logistic Regression emerges as a well-suited approach. We developed code from scratch as well as utilized the sklearn library for our implementations. Initially, we processed the data without any feature transformation or regularization, testing a range of learning rates: [1e-7, 1e-6, 1e-5, 0.0001, 0.001, 0.01, 0.1, 1, 5] and number of iterations: [10, 30, 100, 300, 1000, 3000, 10000, 30000, 100000]. The rate of 0.00001 and 100000 iterations proved most effective, achieving the highest test accuracy of 88.525%. Figure 1 below shows accuracy for each learning rate and number of iterations.

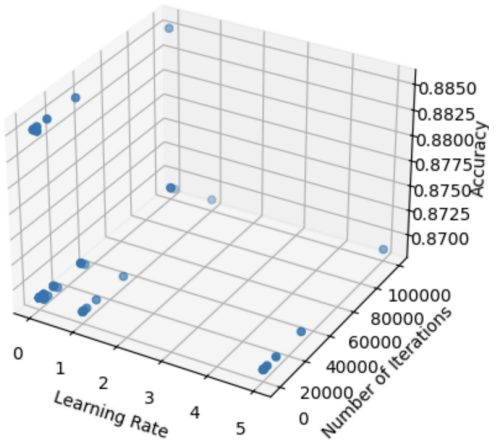

Figure 1. Accuracy, Learning Rate, and Number of Iterations for Logistic Regression

Subsequently, we incorporated both L1 and L2 regularization using the sklearn linear_model library. Results show that both the Lasso (L1) and Ridge (L2) models yielded the same optimal λ value of 0.35938, and identical accuracy scores of 0.852459 and 0.863636 for the validation and training datasets, respectively. This suggests that regularization does not significantly affect accuracy compared to the unregularized model.

The confusion matrices below also show that there is no significant influence of regularization on the model.

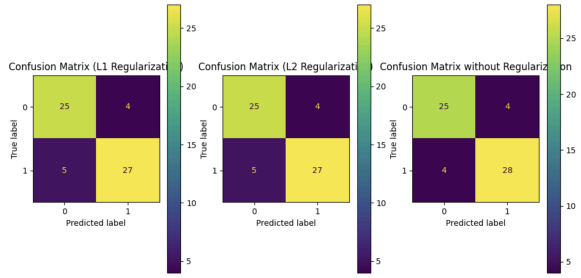

Figure 2. Confusion matrices of L1, L2 and No Regularization on Logistc Regression

Lastlu, polynomial feature transformations with degrees 2, 3, and 4 were applied. The third-degree transformation showed the best performance, getting highest accuracy, precision, and F1 score. The highest recall was observed in the model without any transformation. The comparative results are detailed in Table 1.

	Without Transformation	2	3	
Precision	0.878788	0.9	0.933333	0.86666
Recall	0.90625	0.84375	0.875	0.8125
F1	0.892308	0.870968	0.903226	0.83871
Accuracy	0.885246	0.868852	0.901639	0.83606

Table 1. Performance Metrics of Polynomial Feature Transformations on Logistic Regression

2. Support Vector Machine (SVM)

2. 1. Pegasos Algorithm:

Support Vector Machine (SVM) is another effective tool for binary classification. We explored both the Pegasos Algorithm and three kernel functions: Linear, Radial Basis Function (RBF), and Polynomial. Using the Pegasos algorithm, we achieved a validation accuracy of 70.0%, with default lambda value of 0.005 and 50 iterations. After applying hyperparameter tuning, [0.001, 0.005, 0.01, 0.05, 0.1] for lambda and [10, 20, 30, 40, 50] for number of iterations, the optimal parameters were found to be a lambda of 0.001 and 10 iterations, yielding the best accuracy of 80%. Table 2 shows accuracy for each lambda value and number of iterations.

2. 2. Kernel Functions

For kernel functions, we experimented with linear, polynomial, and RBF kernels. Because training our model using SVC is quite computationally expensive, we first used a subset size of 50 to compare the performance of all three kernels before optimization. The results show that linear kernel yields the highest accuracy at 85%, while 3-degree polynomial and RBF kernels both have an accuracy of 75%.

Next, using GridSearchCV to optimize three different kernels, we varied the regularization parameter C = [0.1, 1, 10], gamma = [0.1, 1, 10] for RBF, and degree = [2, 3, 4] for polynomial kernel. The most effective parameters were a C of 10, a polynomial degree of 2, and a gamma of 0.1 for the RBF kernel, with the linear kernel outperforming the others. The mean accuracy values of all three kernel functions using 5-fold cross-validation are recorded in Table 3. The mean accuracy for linear kernel is the highest and the most consistent, fluctuating very slightly around 86.6%. The mean accuracy for both RBF and polynomial kernels both vary significantly from 55% to 78%, indicating that their performance is dependent on the choice of hyperparameters, such as degree of polynomial or gamma value for RBF. The linear kernel, with C set to 10, delivered a training accuracy of 86.6% and a validation accuracy of 75%.

Because our dataset is high-dimensional, it is difficult to fully visualize every single feature on a graph, we chose two features at a time to graph on a 2D plot. Figure 3 illustrates the decision

boundary of the linear SVM, showcasing the two features Resting Blood Pressure and Age.

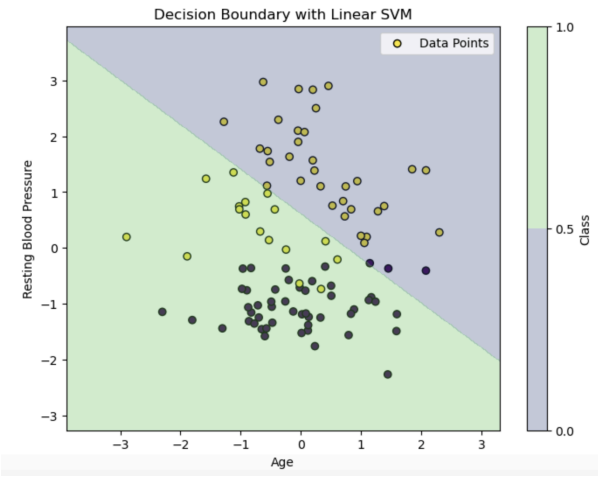

Figure 3. Decision Boundary with Linear SVM for Resting Blood Pressure vs. Age

3. Neural Network

We also incorporated a Neural Network into our analysis, due to the complex nature of our medical dataset, neural networks can be expected to capture complex relationships and non-linear patterns in the data.

For activation functions, we tried Sigmoid, tanh, and ReLU within our model. The model was optimized using Stochastic Gradient Descent. With the default sigmoid activation function, one 8-neuron hidden layer, and learning rate of 0.01, we achieved a training accuracy of 89.6% and precision, recall, F1 of 90.5%, 89.1%, and 89.8%, respectively.

Next, we tested our model on the validation set, where both the accuracy and other performance metrics were lower than that of the training set. The accuracy on the validation set was 78.1818%. L1 and L2 regressions were performed and both showed significant improvement in accuracy on validation set. Without parameter tuning and using default lambda of and learning rate of 0.01, L1 regularization has an accuracy of 87.3% and L2 reached a 90.1% accuracy. After iterating through lambda values ranging from 1E-5 to 1.0, L1 performed the best with lambda of 0.0001, yielding a 98.2% accuracy. L2 with the optimal lambda value of 0.0001 reached a 100% accuracy. When tested on unseen test dataset, L1 and L2 with their optimal parameters have accuracy value of 85.25% and 86.07%, respectively.

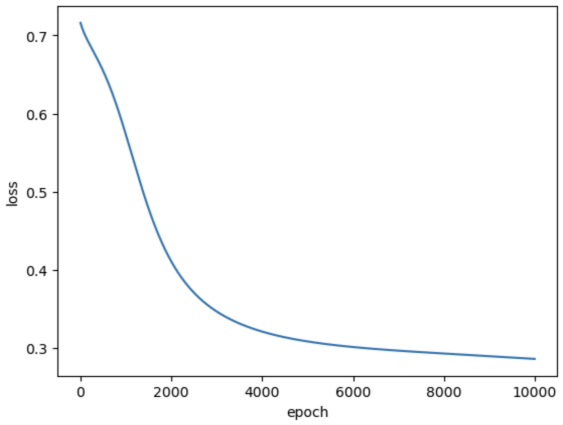

Figure 4. Loss Curve of Neural Network Model with Sigmoid Activation Function

Lastly, we used GridSearchCV to tune hyperparameters including number of hidden layers (from 2 to 4), learning rate from 1E-5 to 0.1, and activation functions including ReLU, Sigmoid, and tanh. The best hyperparameters are one hidden layers, learning rate of 0.01, and ReLU activation function. This setting yields an accuracy of 87.3%, without regularization.

Result

1. Logistic Regression

On training data, our logistic regression model yields an accuracy of 88.53% without regularization, and at optimal learning rate and number of iterations of 0.00001 and 100000, respectively. With regularization, the optimal lambda value, training and validation accuracy for k-fold cross validation are recorded in Table 2.

K-Fold Cross Validation with L1			
k	**Optimal lambda**	**training accuracy**	**validation accuracy**
2	0.35938	0.86364	0.85246
3	2.78256	0.84711	0.86885
4	0.35938	0.86364	0.85246
5	0.35938	0.86364	0.85246
6	0.35938	0.86364	0.85246
7	0.04642	0.86364	0.85246
8	0.35938	0.86364	0.85246
9	2.78256	0.84711	0.86885
10	0.35938	0.86364	0.85246

K-Fold Cross Validation with L2			
k	**Optimal lambda**	**training accuracy**	**validation accuracy**
2	21.54435	0.85124	0.88525
3	2.78256	0.86364	0.86885
4	2.78256	0.86364	0.86885
5	0.35938	0.86364	0.85246
6	2.78256	0.86364	0.86885
7	2.78256	0.86364	0.86885
8	2.78256	0.86364	0.86885
9	21.54435	0.85124	0.88525
10	0.35938	0.86364	0.85246

Table 2. Optimal lambda and Accuracy for L1 and L2 Regularization with K-Fold Cross Validation on Logistic Regressions

With 2-, 3-, and 4-degree feature transformation, the training versus validation accuracy values can be seen in Table 3 below. Figures 5-7 demonstrate the accuracy over number of iterations.

Polynomial Feature Transformation		
Degree	**Validation accuracy**	**training accuracy**
2	0.85246	0.97521
3	0.90164	0.93388
4	0.86885	0.96694

Table 3. Validation and Training Accuracy for Polynomial Feature Transformation

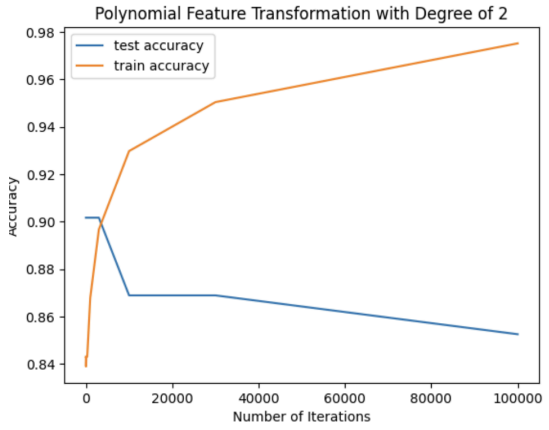

Figure 5: Accuracy versus Number of Iterations for 2-Degree Feature Transformation

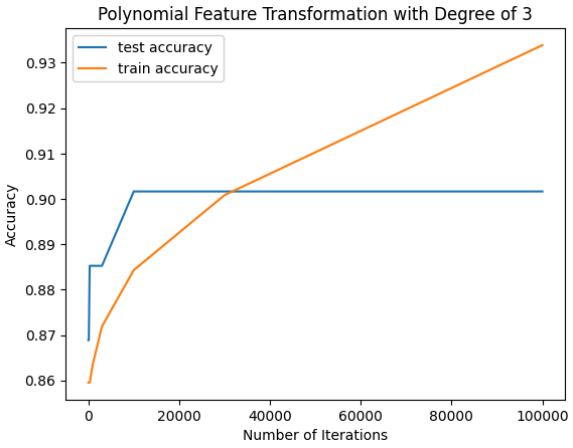

Figure 6: Accuracy versus Number of Iterations for 3-Degree Feature Transformation

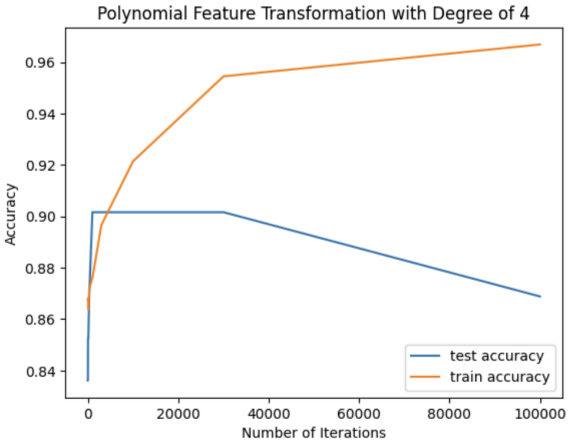

Figure 7: Accuracy versus Number of Iterations for 4-Degree Feature Transformation

2. Support Vector Machine

Using Pegasos algorithm, hyperparamter tuning results including lambda value, number of iterations, and validation accuracy are recorded in Table 4. Training and validation results of our best-performing kernel model with optimal parameters are shown in Table 5.

After hyperparameter tuning, the most optimal kernel is the linear kernel with regularization parameter C of 10. This model yields the best accuracy score of 86.589%.

Pegasos Tuning Hyperparameters		
lambda	**number of iterations**	**validation accuracy**
0.001	10	80.00000
0.001	20	70.00000
0.001	30	60.00000
0.001	40	65.00000
0.001	50	75.00000
0.005	10	75.00000
0.005	20	70.00000
0.005	30	75.00000
0.005	40	75.00000
0.005	50	70.00000
0.01	10	75.00000
0.01	20	75.00000
0.01	30	75.00000
0.01	40	75.00000
0.01	50	75.00000
0.05	10	75.00000
0.05	20	75.00000
0.05	30	75.00000
0.05	40	75.00000
0.05	50	75.00000
0.1	10	75.00000
0.1	20	75.00000
0.1	30	75.00000
0.1	40	75.00000
0.1	50	75.00000

Table 4. Pegasos Algorithm Hyperparameters Tuning Results for SVM

Regularization parameter	Training accuracy	Validation accuracy
10	86.589%	75.00%

Table 5. Training and Validation Accuracy for Optimal Linear Kernel

3. Neural Network:

Without regularization and before hyperparameter tuning, the training and validation accuracy of our neural network model are shown in Table 6.

Without Regularization	Training accuracy	Validation accuracy
	89.683%	78.182%

Table 6. Training and Validation Accuracy for Neural Network without Regularization

After adding regularization and tuning the lambda value, the accuracy on validation versus unseen test data of L1 and L2 regularization are recorded in Table 7.

Regularization	Validation accuracy	Test accuracy
L1	98.182%	85.246%
L2	98.182%	86.066%

Table 7: Validation and Test Accuracy for L1 and L2 Regularization on Neural Network

The loss curves for L1 and L2 are shown in Figures 8 and 9.

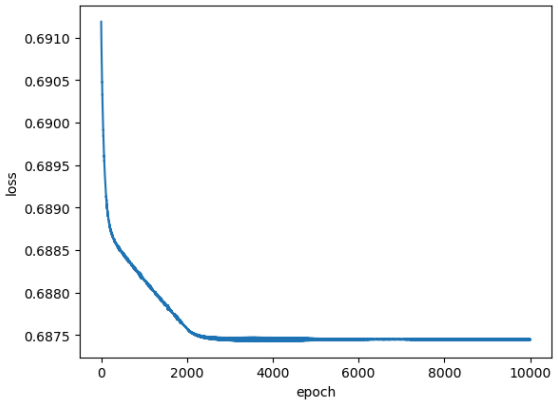

Figure 8. Loss Curve for L1 Regression (default lambda and learning rate)

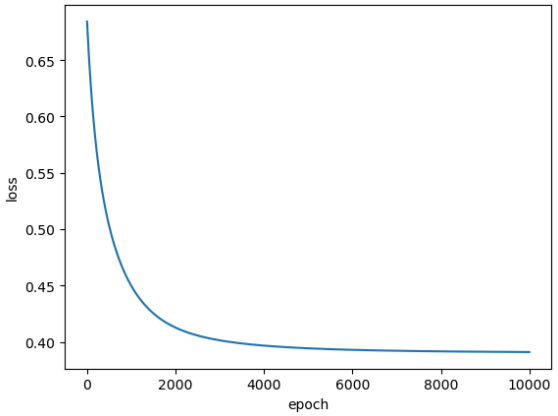

Figure 9. Loss Curve for L2 Regression (default lambda and learning rate)

After tuning all parameters, the most optimal neural network structure has the following settings: one hidden layer, learning rate of 0.01, and ReLU as the activation function. This neural network yields a mean accuracy of 83.6364%.

Conclusion

1. Logistic Regression

Polynomial feature transformation with degree of 2 gives the highest training accuracy of 97.521%. However, the validation accuracy is relatively low, which is 85.246%, showing quite a gap to the very high training accuracy. This implies that the model with 2nd degree polynomial transformation is most likely overfitting to the training dataset, and thus fails to generalize well to unseen validation data.

A polynomial degree of 3 resulted in the highest validation accuracy, indicating that this level of complexity captures the underlying patterns more effectively. However, the higher training accuracy with a degree of 2 or 4, especially the latter, suggests overfitting, as the model performs exceptionally well on the training data but not as well on the validation data. This is characteristic of a high-variance scenario, where the model is sensitive to the noise in the training set.

K-Fold Cross Validation with L1 regularization revealed that while the training accuracy remained relatively stable, the validation accuracy peaked at k=3 and k=9, suggesting an optimal balance at these points. However, the fluctuating optimal lambda values indicate a sensitivity to the model's complexity, hinting at potential overfitting when lambda is minimized. In contrast, K-Fold Cross Validation with L2 regularization showed a less dramatic variance in lambda values but peaked in validation accuracy at k=2 and k=9, indicating a better generalization when the model complexity was appropriately constrained. In summary, while the logistic regression model demonstrated a reasonable degree of predictive power, careful attention to the balance between bias and variance is crucial. Selecting the appropriate degree for polynomial feature transformation and the regularization method and its corresponding lambda is a delicate optimization task that is central to achieving a model that generalizes well to unseen data.

2. Support Vector Machine

The Pegasos SVM model tuning suggests an optimal lambda value lies between 0.005 and 0.01, providing consistent validation accuracy and indicating a balance between bias and variance. Lower lambda values, especially 0.001, may lead to overfitting, as seen by the initial high accuracy that drops with more iterations, likely due to the model capturing noise in the training data. Higher lambdas seem to combat overfitting effectively, as indicated by stable validation accuracies, but setting lambda too high could risk underfitting, where the model is too constrained to capture the underlying data structure. Therefore, the chosen lambda must be carefully tuned to minimize both bias and variance, avoiding overfitting and underfitting, to maintain model generalization.

Contrast to our initial prediction of a non-linear relationship between the input features and the target variable due to the likely complex nature of the dataset, our SVM model proved most effective using linear kernel, significantly more than polynomial and RBF. This suggests that the relationship is more linear in nature, and that a more simple model like the linear kernel which are less prone to overfitting will perform better. Another reason might be that the input features are independent of each other, thus displaying a nearly linear relationship. This is in agreement with our unsupervised analysis at the beginning that showed no clear relationship between the features through K-means clustering.

3. Neural Network

Comparison accuracy of neural network and other two models shows that neural network does not perform well. It might be due to a variety of factors. Neural networks are highly flexible models that can capture complex relationships in the data, but this flexibility comes with the requirement for enough data and careful tuning. Due to the limited size of our dataset, the neural network may not have enough information to effectively learn the underlying patterns, while logistic regression, being a simpler model, can generalize better with fewer data points. Comparing the performance of ReLU and sigmoid activation functions, it appears that ReLU consistently outperforms sigmoid or matches its performance at higher learning rates, which might be attributed to the avoidance of the vanishing gradient problem that can afflict sigmoid functions in deeper networks.

Work Cited

Heart Attack Analysis & Prediction Dataset
https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset