1. Introduction

In the final project, we used a dataset from kaggle that contained the healthcare data that can be used to predict stroke. The dataset contains 10 features, and indicates whether each patient has had a stroke. The original features are: gender, patient id, age, hypertension, heart disease history, marital status, work type, residence type, average glucose level, body mass index (BMI), and smoking status.

This a binary classification problem, with the goal of predicting whether a patient will have a stroke based on health care data. In this project, we experiment with three different classification models: logistic, support vector machines (SVM), and neural networks.

2. Data Preparation

Cleaning the Dataset

The raw dataset contained outliers, incomplete features with null entries, and features that do contribute to the prediction of health status.. To clean up the dataset the outliers and irrelevant features were dropped, and the null values were filled in arbitrarily., The 'id' feature was removed from the dataset as it does not contribute to the actual medical condition of the patients. There were three categories for gender: '*male', 'female',* and '*other'*. As there was only one patient documented as '*other*', that sample was dropped from the dataset as an outlier. The 'bmi' feature was not recorded for all patients and contained null entries.There were 201 null entries in total, which is nearly four percent of the entire dataset. We took a trivial approach to resolve this problem: for each combination of categorical features, we calculated the mean BMI for such combinations and set the samples with null BMI to the mean of all samples with matching combinations of categorical features.

Categorical Encoding

Many features within the data set were categorical and the actual data was linguistic. Applying categorical encoding can help reduce size of the dataset and make the data more accessible for training. The bi-categorical features such as gender, ever married, heart disease, etc, were label encoded into 0 and 1's. However, there are also features that consist of more than two categories. For instance, smoking status was recorded as either: smokes, never smoked, formerly smoked, or unknown. These features were encoded through one-hot encoding, which increases the number of features and typically performs better than label encoding.

Data Scaling

The data was scaled using the StandardScaler() package from Pandas, which zero centered the data and scaled data points by calculating the z-score of that point, transforming the data in each feature to normally distributed data. Normalization of data can help reduce bias within the data. All of the experiments in this rest of the paper will use this normalized version of the input data as the default input.

Data Splitting

The data was split into a training set, a validation set, and a testing set. The training set was 60% of the original set, while the validation set and test set were both 20% of the original set.

3. Models

Logistic Regression

The first model used for stroke prediction with this data set is logistic regression. The outline for acquiring the best hyperparameters was as follows:

1. Create polynomial transforms of the feature matrix of different degrees and experiment for the best feature transformation. Apply no regularization.
2. Using the best feature transformation from the previous step, experiment with L1 regularization, L2 regularization and compare the results with no regularization. The optimal regularization strength hyperparameter for L1 and L2 regularization shall be found with K-fold cross validation.

Polynomial Feature Transformations

In the first experiment, multiple new feature matrices were generated with Sklearn's PolynomialFeatures tool. It offered a parameter named degree that allowed for specification of the degree of the polynomial features. In this experiment, the new feature matrices all included an x_0 term of 1 and bias was included in the logistic regression model. The LogisticRegressionCV model from Sklearn was used because it had K-fold cross validation built in and offered a parameter named cv that allowed for direct specification of the K value. For this first experiment, the K value was fixed to 5, though it did not matter because no regularization was applied. Moreover, this model did not offer the choice of no regularization, so a C value (the inverse of the regularization strength hyperparameter) of 100000 was used with its L2 regularization option to approximate the effect of no regularization.

In the end, only polynomial transformations of degrees 1 (no transformation) and 2 were tested because as the degree increased, the number of features grew huge and the training time was too long. The original feature matrix had 18 features, and the 2nd degree polynomial transformation resulted in 171 features, while the 3rd degree polynomial transformation had 1140 features. Starting from the 3rd degree polynomial transformation, the model became infeasible to train with a laptop's hardware. The results are shown in table 1 below.

Dataset	**Polynomial Transformation Degree**	
	1 (no transformation)	***2***
Train	0.95106	0.953997
Validation	0.962818	0.954012

Table 1: Polynomial transformation results

Since the validation accuracy was higher with no polynomial feature transformation, it was concluded that no polynomial feature transformation for this dataset and model is better.

Regularization with K-fold validation

In this part of the experiment, no polynomial transformation was applied to the input data. K for K-fold cross validation varied from 2 to 10. For each K value, 10 C values from the range of 0.0001 to 10000 were tested on every fold, and the best validation scores were recorded for each K value. The results are shown in tables 2 and 3

L1 Regularization		
K	**Best C**	**Score (Accuracy)**
2	0.0001	0.95106
3	0.0001	0.95106
4	0.0001	0.95106
5	0.35938	0.95139
6	0.35938	0.95139
7	2.78256	0.95139
8	2.78256	0.95139
9	2.78256	0.95139
10	0.0001	0.95106

Table 2: Logistic L1 Regularization Results

L2 Regularization		
K	**Best C**	**Score (Accuracy)**
2	0.0001	0.95106
3	0.0001	0.95106
4	0.0001	0.95106
5	0.35938	0.95139
6	0.35938	0.95139
7	0.35938	0.95139
8	0.35938	0.95139
9	2.78256	0.95139
10	0.0001	0.95106

Table 3: Logistic L2 Regularization results

The best score with L1 regularization applied to the model was 0.95139, and the best score with L2 regularization applied was 0.95139 as well.

Conclusion and Performance on Test Set

As indicated by the results of the validation scores above, the best logistic regression model is the model with no polynomial feature transformation and no regularization. A possible reason for why polynomial feature transformation did not increase the validation accuracy could be that the original feature matrix was already very linearly separable, and applying a polynomial feature transformation made it less separable. A reason for why regularization did not help with the validation accuracy could be that the model did not overfit to the data set, and regularization made the model less flexible, thus leading to a lower validation score.

The accuracy of this model on the test set is 0.941292.

Support Vector Machine

Support vector machines offer a plethora of options and freedom in terms of model type and hyperparameter settings. Using the Scikit-Learn python library, this investigation deployed the linear kernel using both L1 and L2 regularization techniques, and the polynomial kernel with various degrees. The best hyperplanes found in each of the respective models and configurations yielded the test accuracy. We suspect the convergence is a result of the optimized implementation of the Scikit-Learn library package. This motivated an experimentation with k-fold cross validation, which produced models that performed better on the test set.

Linear Kernel & Polynomial Kernels

Three different configurations of the linear kernel SVM were tested: linear kernel without regularization, linear kernel with L1 regularization, and linear kernel with L2 regularization. The results are shown in table 4 below. The three configurations produced identical results in terms of accuracy on both the training and test set. As discussed earlier, this is most likely due to the fact that the Scikit-Learn linear kernel utilizes many optimization techniques, and all three configurations found the best hyperplane for the given dataset.

Dataset	**Configuration**		
	No Regularization	***L1 Norm***	***L2 Norm***
Train	0.955257	0.955257	0.955257
Test	0.941944	0.941944	0.941944

Table 4: linear kernel results

SVM with polynomial kernel was deployed on the same test and training set, from 1-degree to 9-degree. The results are shown in table 5 and figure 1 below. The polynomial kernel performs the best when the degree is 1 and 2, which yields the same training and test accuracy as the linear models.. As the degree increases above 2, the accuracy on the training and test sets begin to diverge, with the training set accuracy increasing and the test set accuracy decreasing. This is within expectation, as the risk of overfitting usually emerges as the polynomial degree rises. The high degree models have higher variance as they become more specific and sensitive to the training data. One would expect low degree models to have high bias, but their good performance on this dataset suggests that there is a strong correlation between the features and prediction target. Also, the normalization step in data preprocessing contributed to low bias in our models.

Degree	Training	Test
1	0.955257	0.941944
2	0.955257	0.941944
3	0.957215	0.939987
4	0.962248	0.938030
5	0.964765	0.936073
6	0.967841	0.934768
7	0.969519	0.930855
8	0.971197	0.930202
9	0.971197	0.927593

Table 5: Polynomial Kernel Accuracy

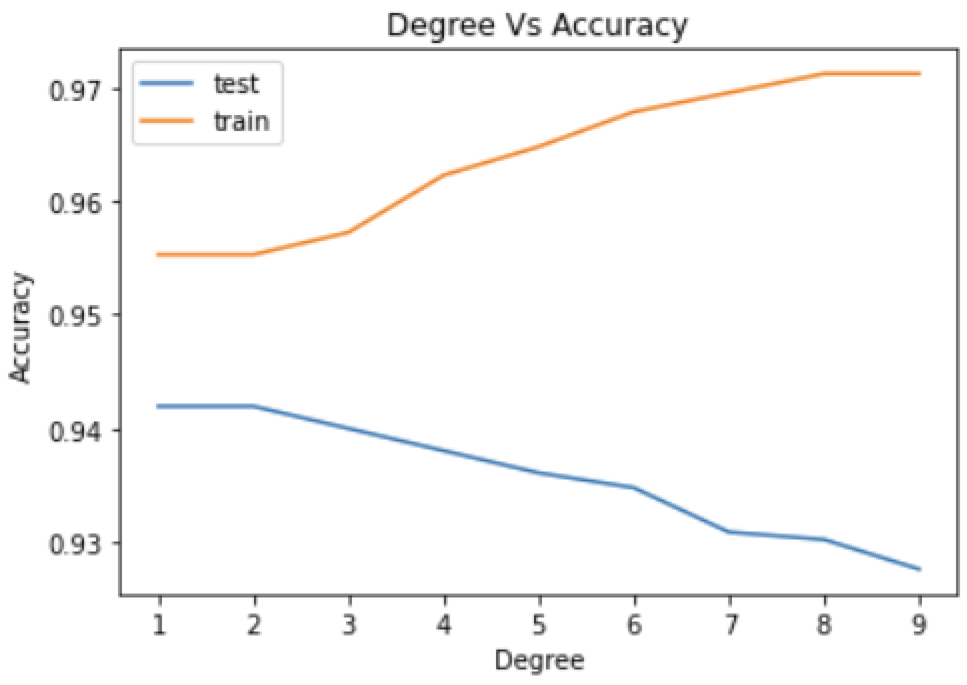
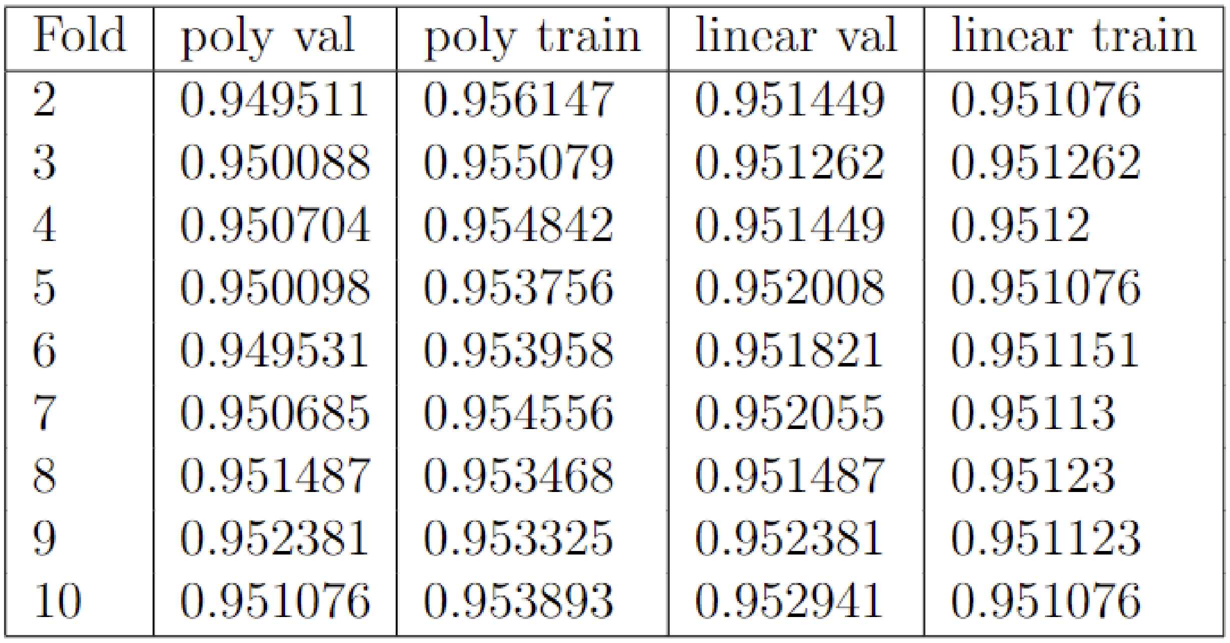


Figure 1: Polynomial Degree Vs Accuracy

K-Fold Cross Validation

The results from the linear kernel and polynomial kernel were not satisfactory as they seem to hit a ceiling in accuracy. While the convergence of the results is partly due to the optimized training of the Scikit-Learn library, it is also plausible to attempt obtaining better partitions of the training and test sets to improve the models. To this reason, a K-fold cross validation was implemented with both linear and polynomial kernels. In the set up to the k-fold implementation, the default Scikit-Learn default settings were used for both linear and polynomial kernels. The library optimizes the result by choosing the optimal hyperparameter setting for regularization, and using the polynomial degree best balances the tradeoff between bias and variance. Cross validation of two to ten folds were trained. Data shuffle and random state were introduced so the data split at each fold would remove any potential dependencies within the ordering of the data. The results are displayed in table 6 and figure 2.



Fold	poly val	poly train	linear val	linear train
2	0.949511	0.956147	0.951449	0.951076
3	0.950088	0.955079	0.951262	0.951262
4	0.950704	0.954842	0.951449	0.9512
5	0.950098	0.953756	0.952008	0.951076
6	0.949531	0.953958	0.951821	0.951151
7	0.950685	0.954556	0.952055	0.95113
8	0.951487	0.953468	0.951487	0.95123
9	0.952381	0.953325	0.952381	0.951123
10	0.951076	0.953893	0.952941	0.951076

Table 6: K-Fold Accuracies

The linear kernel's training accuracy remained consistent throughout different folds. The linear kernel's validation accuracy had an increasing trend as folds increased. On the other hand, the polynomial kernel's training accuracy was decreasing as folds increased. This is within expectation. The training set size increases as the number of folds increases, so when the training set is small when folds are small, the polynomial kernel could overfit and have high variance as it is sensitive to small changes in the data. As the training size increases, the variance will decrease and the model becomes less sensitive to changes and has improved generalization error. This is why an increasing trend in test accuracy for the polynomial kernel was observed as folds increased.

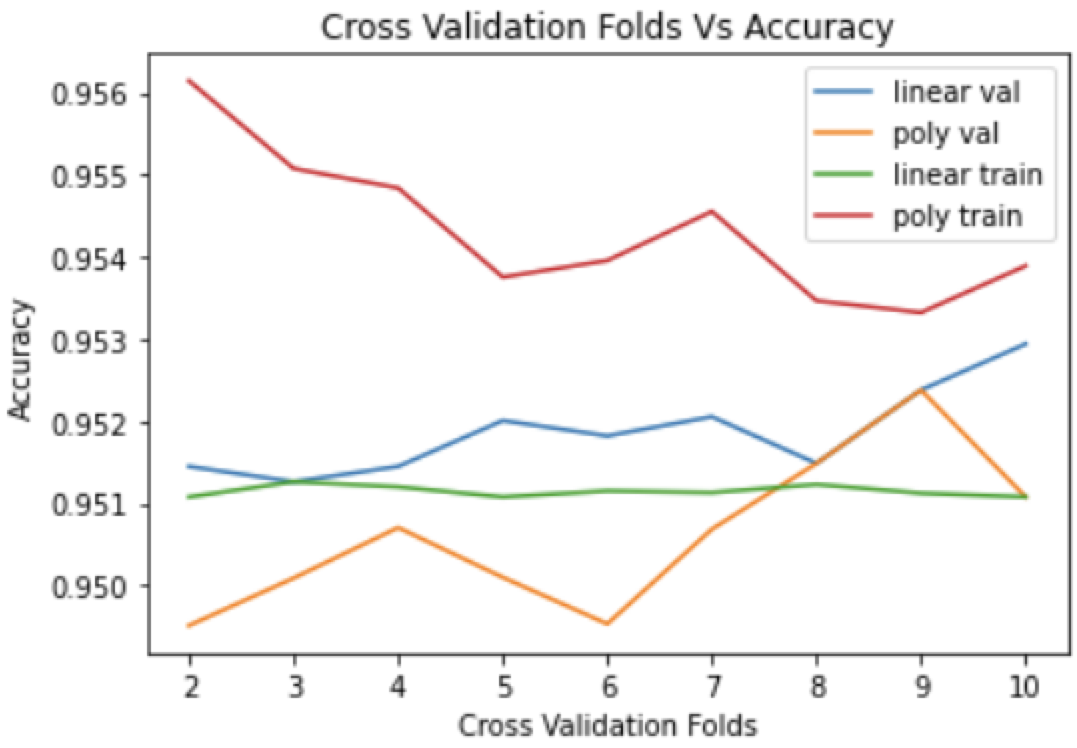


Figure 2: Cross Validation Folds Vs Accuracy

Conclusion

The linear kernel SVM's performance did not change with the introduction of L1 and L2 penalties. Its best test accuracy also converged with that of the polynomial kernel at degree 1 and 2. The fact that low degree polynomial kernel transformation has produced the best performing model suggests that the data is linearly separable.

Linear	Polynomial degree 1	Linear CV	Polynomial CV
0.941292	0.941292	0.941973	0.943901

Table 7: Final Best Results

The typical expectation is for bias to be high when the number of folds is low, but bias does not seem to be significant throughout. The normalization done in preprocessing helped reduce bias, but many of the features are also categorical data that have few types. The best results for each kernel were obtained at 10 folds for the linear kernel and 9 folds for the polynomial kernel, with respective validation accuracies of 0.952941 and 0.952381. On the test set, the best linear kernel model obtained an accuracy of 0.941973 and the best polynomial kernel model obtained an accuracy of 0.943901. Slight improvement in performance is observed in both models after introducing cross validation.

Neural Network

For the neural network architecture, the plan to obtain a well-performing model was as follows:

1. Use no regularization and test three neural networks with different numbers of layers and neurons in the hidden layers.
2. For each of the three neural networks, apply L1 and L2 regularization and observe whether their performance improves.

Three-Layer Neural Network

First, a three layer neural network was constructed. The input layer had 17 neurons, corresponding to the 17 prediction features, the hidden layer had 10 neurons, and the output layer was just one neuron, indicating the likelihood of a stroke. The activation functions of the first and second layer were ReLU; the activation was sigmoid for the output layer for a likelihood value between 0 and 1. The structure of the model is shown in the figure below:

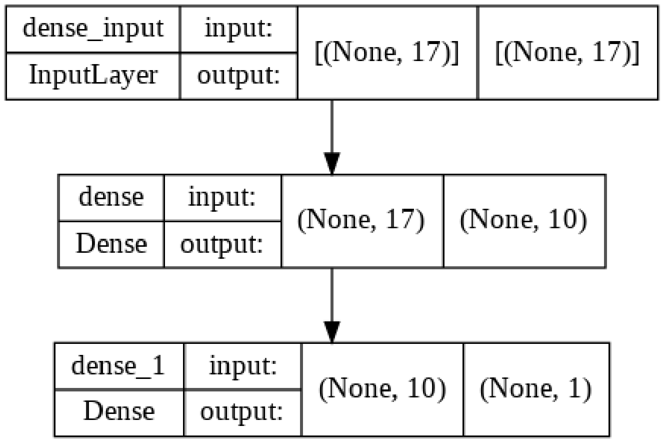


Figure 3: Three Layer Structure

Using the Keras library from Tensorflow, this model was trained with a batch size of 10 for 70 epochs. No regularization was applied. Keras evaluated the cross entropy loss and accuracy of the model for both the training set and the validation set after each epoch. The lowest validation loss of 0.1719 occurred at epoch 24, with a validation accuracy of 0.9499.

Next, with L1 regularization applied and a regularization strength of 0.01, the model's validation loss converged at 0.192 towards the end of the 70 epochs, and the validation accuracy was 0.9499 as well.

When L2 regularization was applied to this model, the validation converged at around 0.1877 towards the end of the 70 epochs, and the validation accuracy was 0.9499.

The results for this neural network are summarized in the table below.

3-layer Neural Network			
Metric	**Regularization**		
	No Regularization	***L1 Norm***	***L2 Norm***
Best Validation Loss	0.1719	0.1920	0.1877
Best Validation Accuracy	0.9499	0.9499	0.9499
Best Validation Epoch	24	69	68

Table 7: Three Layer Results

Four-Layer Neural Network

In the second experiment, the neural network had two hidden layers; the first hidden layer had 10 neurons while the second hidden layer had 5 neurons. All activation functions were ReLU except for the output layer's neuron, which was still sigmoid. The structure of the model is shown in the figure below:

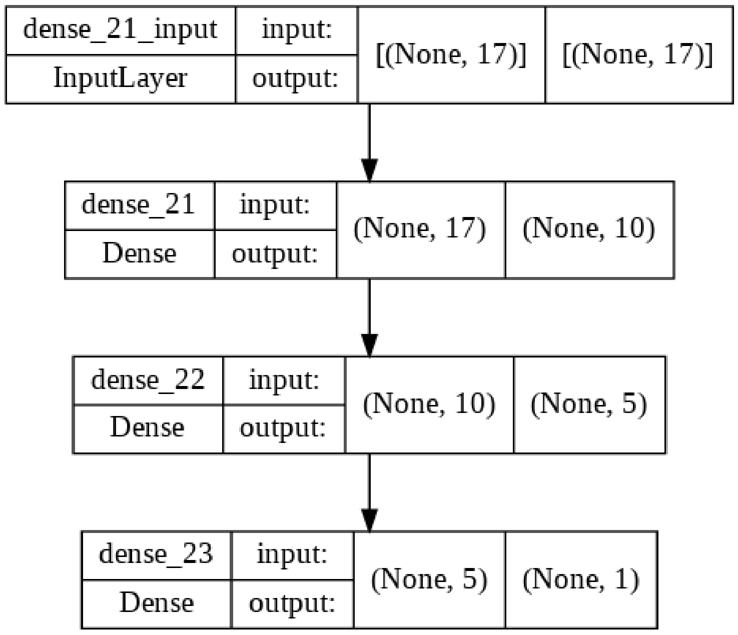


Figure 4: Four Layer Structure

The experiment was carried out in the same fashion as for the previous neural network. The results are summarized in the table below.

4-layer Neural Network			
Metric	**Regularization**		
	No Regularization	***L1 Norm***	***L2 Norm***
Best Validation Loss	0.1687	0.2020	0.1967
Best Validation Accuracy	0.9499	0.9499	0.9499
Best Validation Epoch	9	69	51

Table 8: Four Layer Results

Five-Layer Neural Network

In the last experiment, the neural network had three hidden layers; the first hidden layer had 12 neurons, the second hidden layer had 7 neurons, and the third hidden layer had 4 neurons. All activation functions were ReLU except for the output layer's neuron, which was still sigmoid. The structure of the model is shown in the figure below:

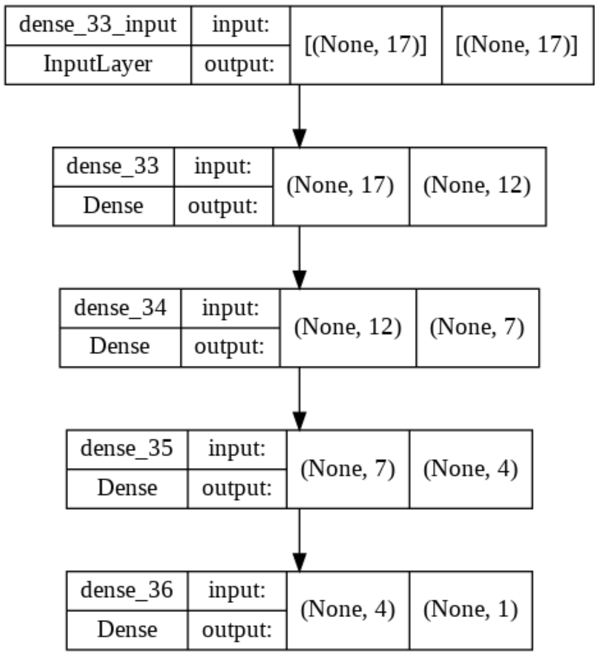


Figure 5: Five Layer Structure

The experiment was carried out in the same fashion as for the previous neural networks. The results are summarized in the table below.

5-layer Neural Network			
Metric	**Regularization**		
	No Regularization	***L1 Norm***	***L2 Norm***
Best Validation Loss	0.1964	0.1964	0.1965
Best Validation Accuracy	0.9499	0.9499	0.9499
Best Validation Epoch	17	52	40

Table 9: Five Layer Results

Conclusion

In conclusion, the best neural network model for this data set is a 4-layer neural network with a 10-neuron hidden layer then a 5-neuron hidden layer and no regularization. Running this model on the test set, it achieved a cross entropy loss of 0.1808 and an accuracy of 0.9413.

Moreover, interesting phenomena observed from these experiments include:

1. All the neural networks with no regularization at all outperformed the ones that did utilize L1 or L2 regularization. However, they easily start overfitting to the training data and thus it is crucial to have a validation set to evaluate its generalization performance for every epoch. In this experiment, early stopping worked very well for these no-regularization neural networks.
2. When regularization was applied, no sign of overfitting was observed at all; that is, for all of the neural networks with regularization, the validation loss kept decreasing until convergence. This indicates that regularization leads to significant underfitting of the model for this data set.

Final Conclusion

The best model obtained each of the logistic, support vector machine, and neural network models are displayed in table 10. The best performing model was the polynomial kernel SVM model that was trained using 9 fold cross validation, with a test accuracy of 0.943901.

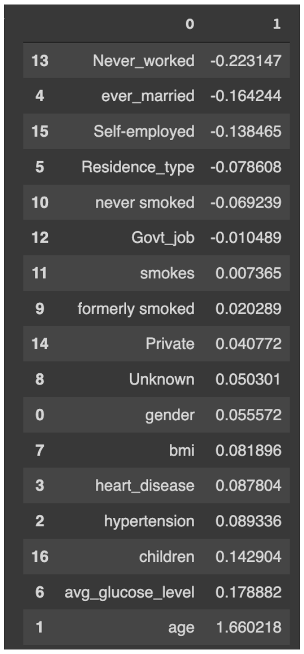
Logistic	SVM	Neural Network
0.9413	0.9439	0.9413

Table 10: Best Test Results Across Models

The dataset is extremely linearly separable and the relationship between the healthcare features and stroke is clear. Multiple observations strongly indicate this. The polynomial kernel SVM performed the best when the polynomial degree was 1 and 2. With part of the motivation for the polynomial kernel transformation as transforming data such that they become linearly separable, finding the optimal results at degree 1 simply indicates that the data itself is already linearly separable. Indication of this is also observed from the cross validation process in the logistic model with regularization penalty. On multiple instances, the best C value was 0.0001, or lambda = 10000. Usually a large lambda increases the risk of underfitting, but the results clearly are void of this problem. The high lambda value means the model is simple, and that the correlation between the features and the prediction target is strong.

Across all three models and the various approaches used with the models, the performance of the models were very similar. The convergence at around 0.94 test accuracy, when random shuffling was used when the data was split, indicates that the accuracy ‘ceiling’ across different approaches can perhaps be attributed to a result of outliers or features unaccounted for in the dataset. Also there is natural variability and uncertainty in the health status of individuals.

The biggest contributors for the output of the best logistic regression model and the linear SVM model were also investigated by looking at the weights of the model. The linear SVM model was used because using the model with a polynomial kernel changes the features and the weights will not show the importance of the original features. Investigation of the neural networks’ feature importance was omitted as it requires complicated methods. The results are shown in the figures below.



	0	1
13	Never_worked	-0.223147
4	ever_married	-0.164244
15	Self-employed	-0.138465
5	Residence_type	-0.078608
10	never smoked	-0.069239
12	Govt_job	-0.010489
11	smokes	0.007365
9	formerly smoked	0.020289
14	Private	0.040772
8	Unknown	0.050301
0	gender	0.055572
7	bmi	0.081896
3	heart_disease	0.087804
2	hypertension	0.089336
16	children	0.142904
6	avg_glucose_level	0.178882
1	age	1.660218

Figure 5: Logistic Regression Weights

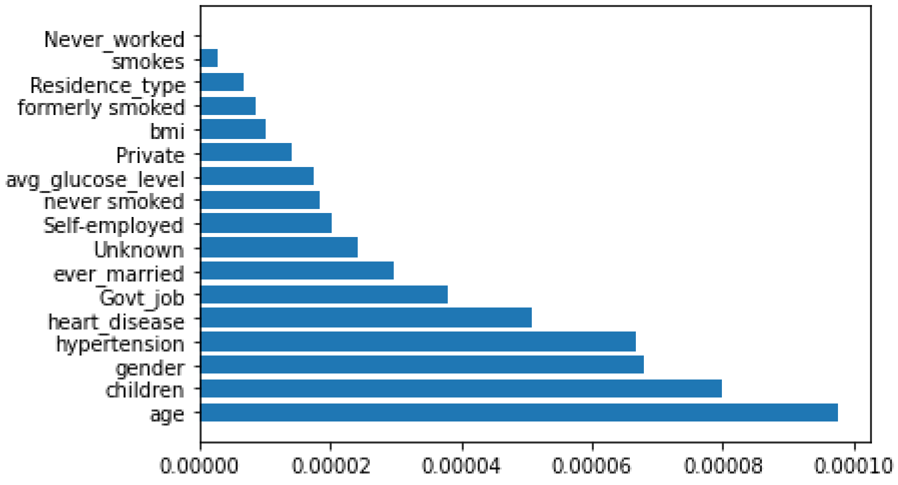


Figure 6: SVM weights

The logistic regression model's weights suggest that the 3 biggest contributors to having a larger stroke possibility are age, average glucose level, and having children. The linear SVM model's weights suggest that the 3 biggest contributors to having a larger stroke possibility are age, having children, and gender.