IntroML

Yonatan Ashlag, Orel Zvi, Tal Dugma

October 2023

1 Introduction

In this project we attempt to predict stroke likeliness based on patients data. we will implement and analyse different machine learning methods,discuss how we used them and compare results.

1.1 The Data

Our data consists of 40910 samples taken from both people that had a stroke, and didn't. Each sample has 10 features: 'sex', 'age' ,'hypertension', 'heart disease', 'ever married', 'work type', 'Residence type', 'avg glucose level', 'bmi', 'smoking status', 'stroke'.

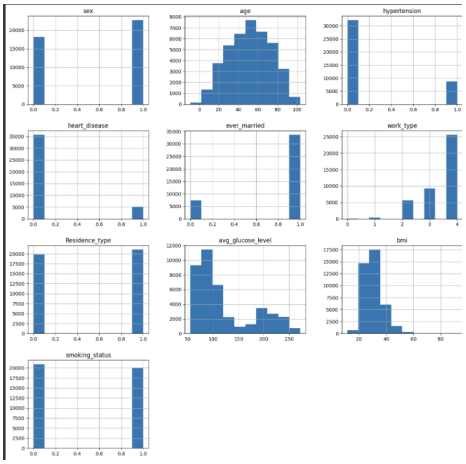


Figure 1: feature histogram

1.2 The Data Preparation Process

Normalization: since different features are measured with different measuring units, normalizing all the data to the same scale is mandatory in order to

improve model performances. in our project we used MinMax scaling, transforming all features to corresponding values between 1 and 0.

One hot encoding: the "work type" feature has 3 values: 'Govt job', 'Self-employed' and 'Private'. They were represented in a single column as 2,3,4. This is problematic since it creates a linear correlation between unrelated values. To combat this we used the "one hot encoding" technique transforming the column into 3 separate columns, each with binary values. This change improved our results.

Skewness: high skewness in data may harm results since outliers have heavy weight and can "confuse" the model. We solved this issue by using a quantile transformer function that shifts the data into a normal distribution.
Splitting the data: lastly, when training machine learning models it's common practice to split the data into a Train/Test/Validation split, in this project we did a 70/15/15 split.

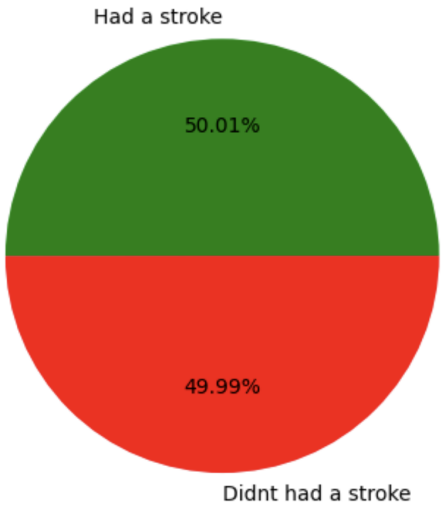


Figure 2: Positive Negative Split

2 Neural Network

2.1 The Network's Structure

We implemented a fully connected neural network with 3 hidden layers. Our input layer had 12 neurons. After some trial and error we found that the most effective structure for the hidden layers is 12/8/4 and one neuron output layer.

2.2 Hyper Parameters

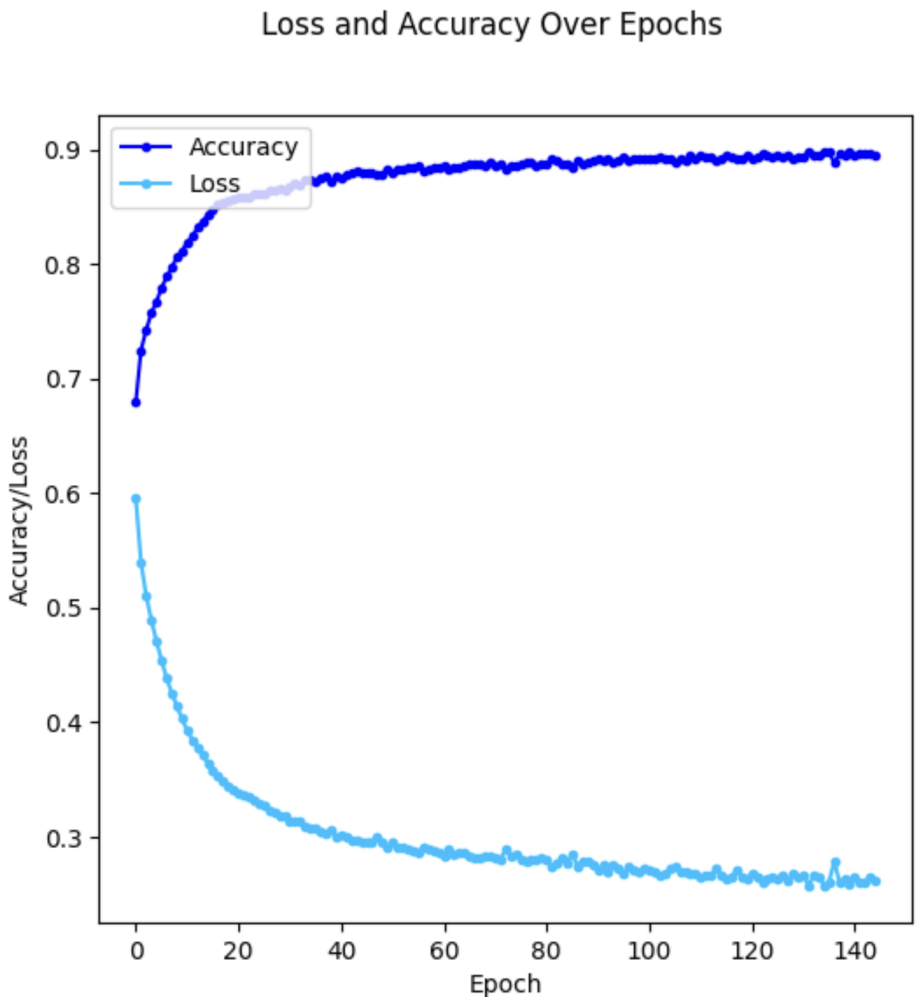
Loss Function: Our goal was to do binary classification. The most effective loss function for that is Binary Cross Entropy.
Optimizing Algorithm: We chose ADAM.
Activation Function: For the output layer we used the Sigmoid function since it creates separation and is great for classification. For the rest of the layers we used ReLu.

2.3 Overfitting

Overfitting is a common problem for learning algorithms where the model "focuses" to much on the variation in the data instead of learning the underlying truth. In our case we only have 12 features, which isn't very deep. This combining with the thousands of data points, we raised a major concern for overfitting. To mitigate that risk we implemented an early stopping mechanism that monitored the loss function progress over the epochs and stopped the learning when it noticed slow progress.

2.4 Performance

```
Nueral network accuracy: 88.6%
```



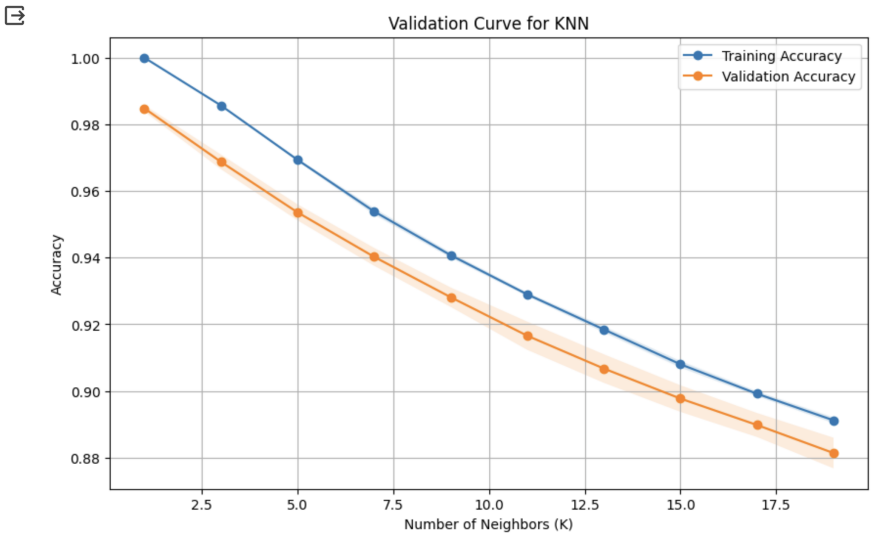
Note: We let the model run on hundreds of epochs but the early stopping mechanism stopped it after 150.

3 KNN

3.1 Hyper Parameters

In KNN hyper parameters play a huge roll in model performance, since there isn't a clear way to determine which are the best, we ran a grid search to determine the optimal amount of neighbors and the best weight metric.

3.2 Results




```
optimal parameters: n_neighbors=1,weights=distance
KNN accuracy: 98.9%
```

4 Random Forest

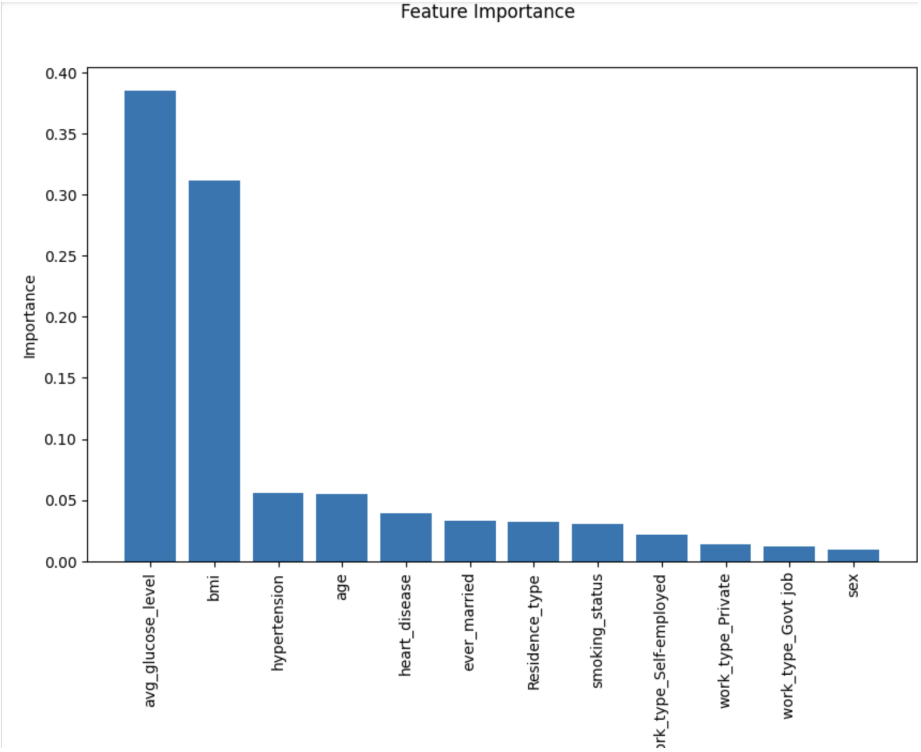
4.1 Results

since most of the features in our data are binary we had a theory that tree-based algorithms would work well. Indeed, the Random Forest algorithm, predicted the stroke with nearly a hundred percent certainty.

```
Random Forest Accuracy: 0.9965781326380968
```

4.2 Feature Importance

An interesting statistic to look at is feature importance; how important is each feature to the model's final output. The Random Forest model can give us an estimation for each feature importance by computing the mean and standard deviation of the accumulation of the impurity decrease within each tree.



5 SVM

5.1 Hyper Parameters

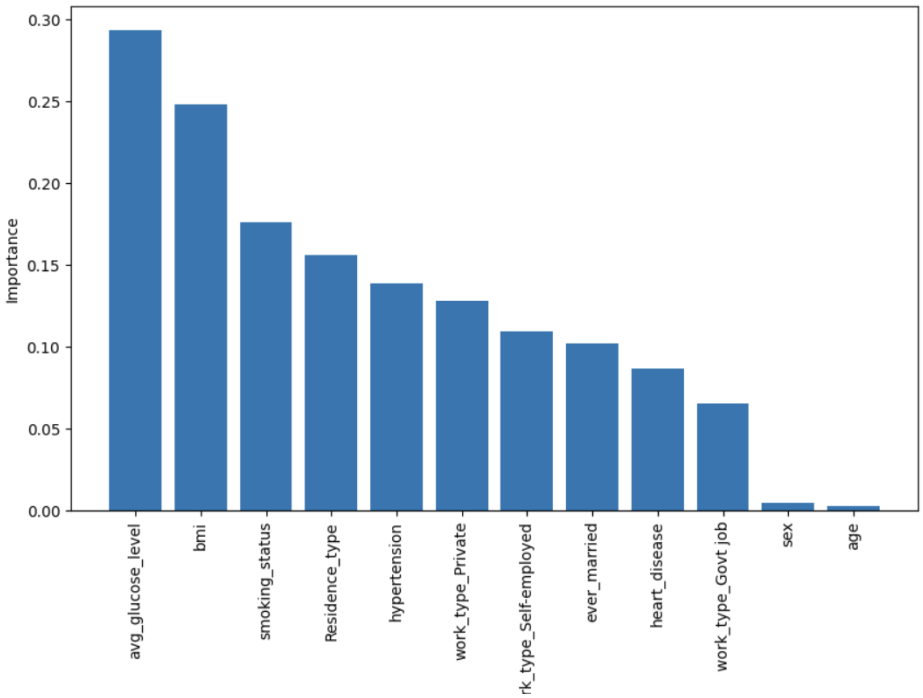
Kernel: we immediately saw that a linear kernel didn't work so we used rbf
Gamma Value: we ran a grid search to find the optimal value

5.2 Results

```
SVM Accuracy: 0.8978328173374613
```

5.3 Feature Importance

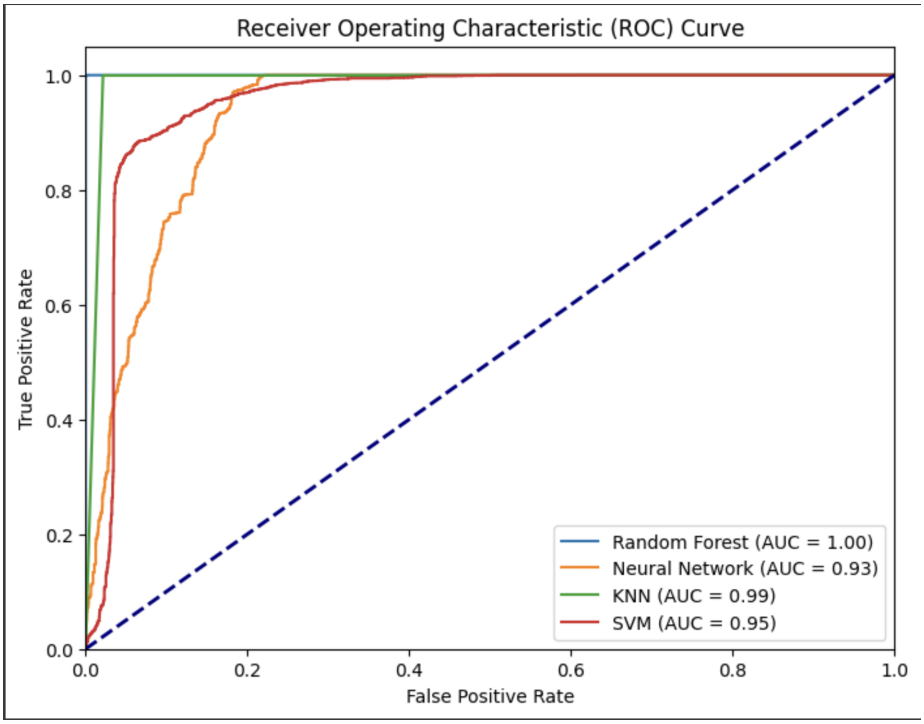
We compute feature importance from our SVM model by using permutation importance, permuting the values of each feature, and measuring the decrease in the model's performance.



Note: notice that both models agree that the two most important metrics are bmi and avg glucose level.

6 Conclusions

Here's a comparison between all 4 models:



6.1 Final Conclusion:

The two best performing models were Random Forest and KNN, boasting an 99+ percent accuracy and AUC scores.

6.2 Explaining The Results:

Logically thinking, we noticed that some of the features in the data such as: "work type", "ever married" and "residence type" are barley related to whether a person would have stroke or not. We believe that this group of redundant features hurt the performance of the supervised learning models, causing them to slightly overfit to the data.