Final project report

DVAE26

David Öst, daviiddd99@gmail.com

February 27, 2025

KARLSTADS
UNIVERSITET

1 Problem definition and requirements

The goal is to develop application from a machine learning model that predicts strokes in humans based on tabular medical data. Based on the nature of the application and its objective the application should perhaps lean toward catching every single person in risk of having a stroke and reducing missed strokes.

2 Exploratory analysis

From simply observing the data one can see that there are alot of things that need to get adjusted before feeding the data to a model training algorithm. First of all the 'id' column can be eliminated to reduce bloat and alot of the data is in string format such as in what enviroment they live in, "urban" or "rural". The training algorithm wont be able to interpret data in that format so label encoding and one hot encoding will have to be used there.

There were also some datapoints that were outliers, not in the sense that the value was far from the average value instead there were some datavalues in a column that made up less than 5% of all the datapoints in a column. An example is the data value "other" in the column "gender" which there was only 1 datapoint with that value out of a total 5100. These outlier values should be removed instead of for this example add an entirely new column just for 1 datapoint.

When it comes to BMI, there were a total of 201 N/A values in that column out of 5110 datatypes. The easiest option would be to remove those 201 rows but the best would be to replace those rows with the average value. But because bmi is closely related to other features such as age, gender and glucose levels i think it would distort the data too much to simply replace those with the average. It would be best i think to simply remove those rows.

3 Feature engineering

Some columns had to be one-hot encoded, these columns were , "ever married" and "residence-type", "gender". But i thought that for the column smoking status it would be smarter to use label encoding since this feature is not nominal instead it has an inherent ranking in the context of the end purpose of our model which is to predict a stroke. Smoking has been linked with plenty of diseases such as stroke and the data for the smoking status column contain information of how much a person smokes, never smoked, formerly smoked,smoker.

4 Feature transformation

The features had to finally be transformed into a numbers format. Most of the features contained 0 or 1 but for the smoking status column the numbers ranged from 0 to 5. Finally the data had to be polished by removing outliers based on their numerical value. A separate function was written for this task called OutlierDetection.py which takes all of the columns whose values are not in categories and removes the datapoints containing values that are 3 standard deviations away from the overall mean value of the column.

5 Model training

A separate python file was written to test each model created called TestingModel.py. This file calculates the relevant metrics like accuracy, recall, precision and f1 score. It also as visualizes the confusion matrix aswell as the feature importances.

The first idea i had when choosing what metric for the model to prioritize was a metric that wants be balanced in correctly predicting positives and negatives which is why accuracy was chosen first.

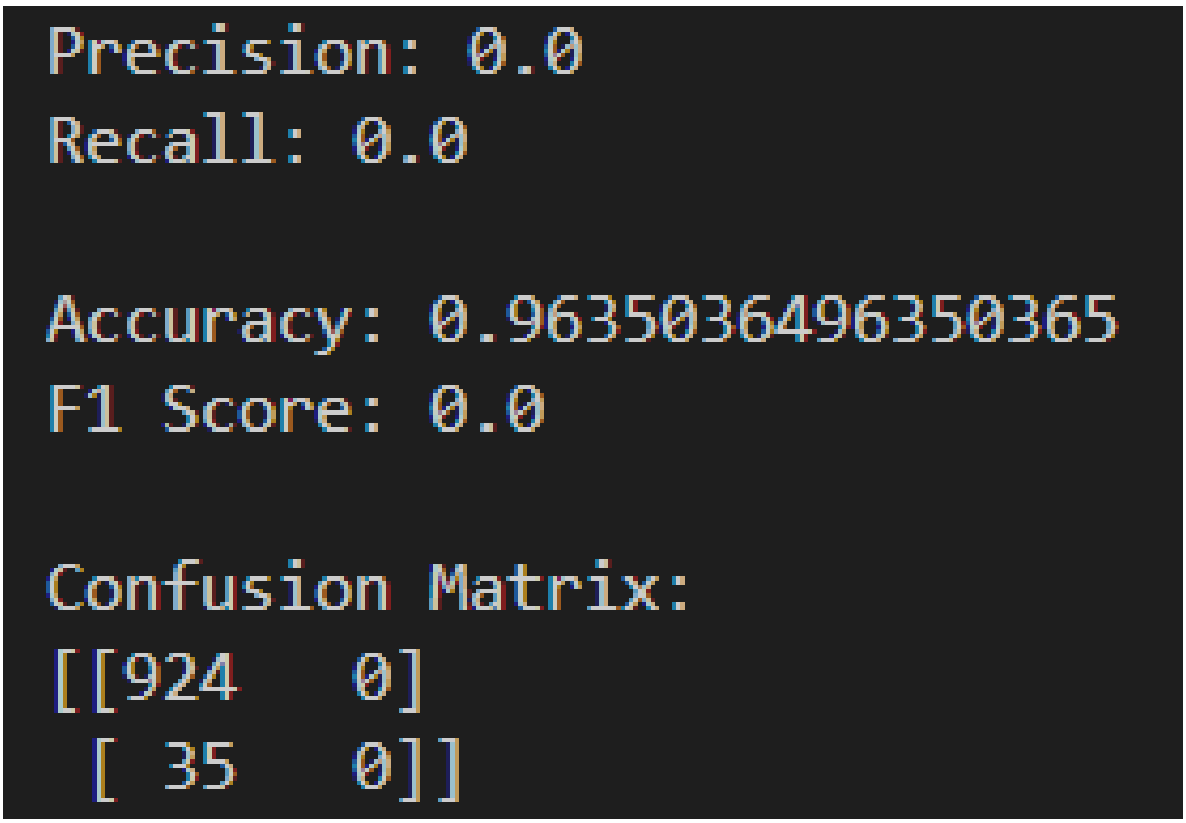

Figure 1: Performance of the accuracy model

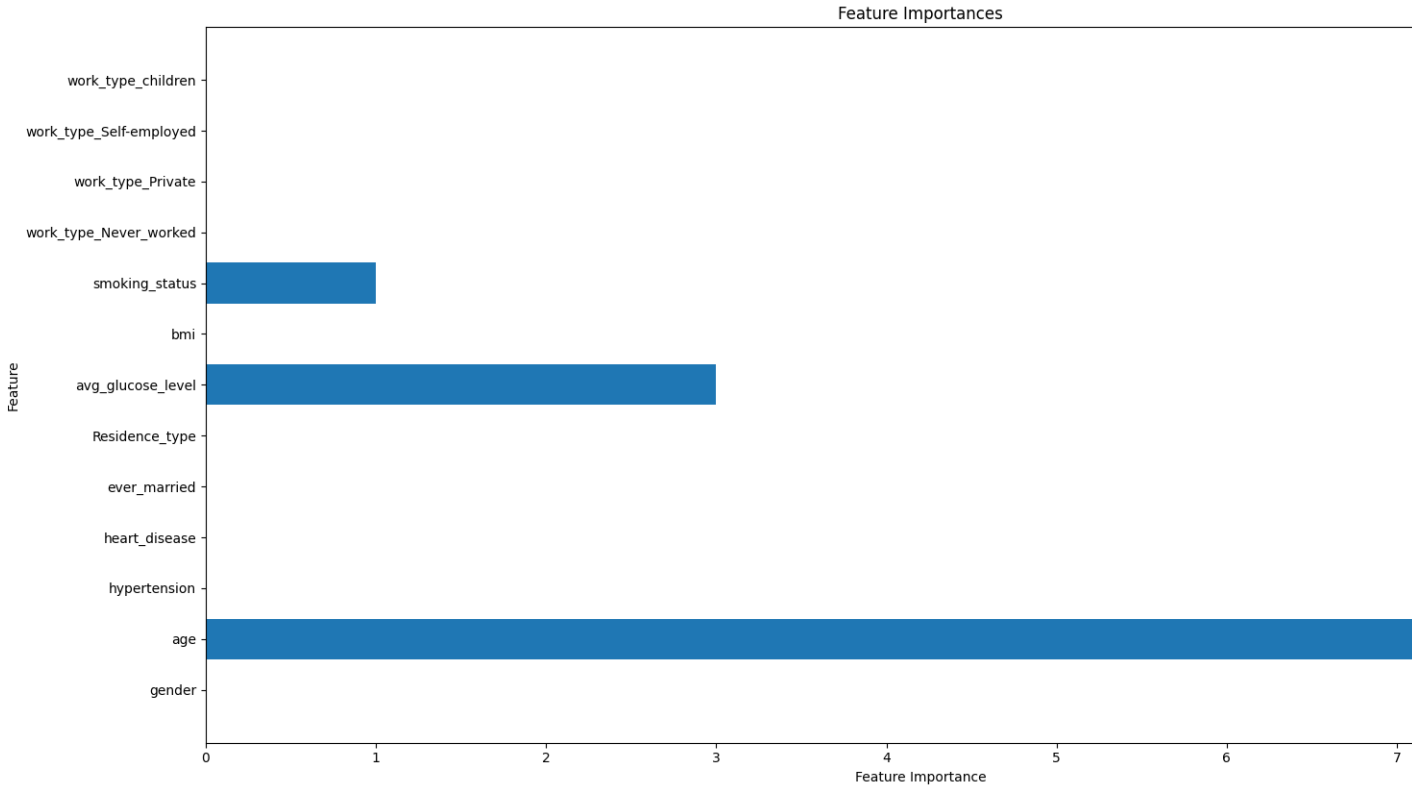

Figure 2: Feature importance of accuracy model

This model performance was not optimal at all considering it did not classify a single instance as positive at all and also failed to identify important risk factors for a stroke such as hypertension and bmi. All models were saved for versioning purposes in the Models folder with the log file which is produced during training along with the actual model as a pickle file.

The next model was trained with a different metric since it seemed accuracy performed poorly both in terms of performance but also feature importance. The next metric was chosen to be ROC AUC (Receiver Operating Characteristic Area Under the Curve).

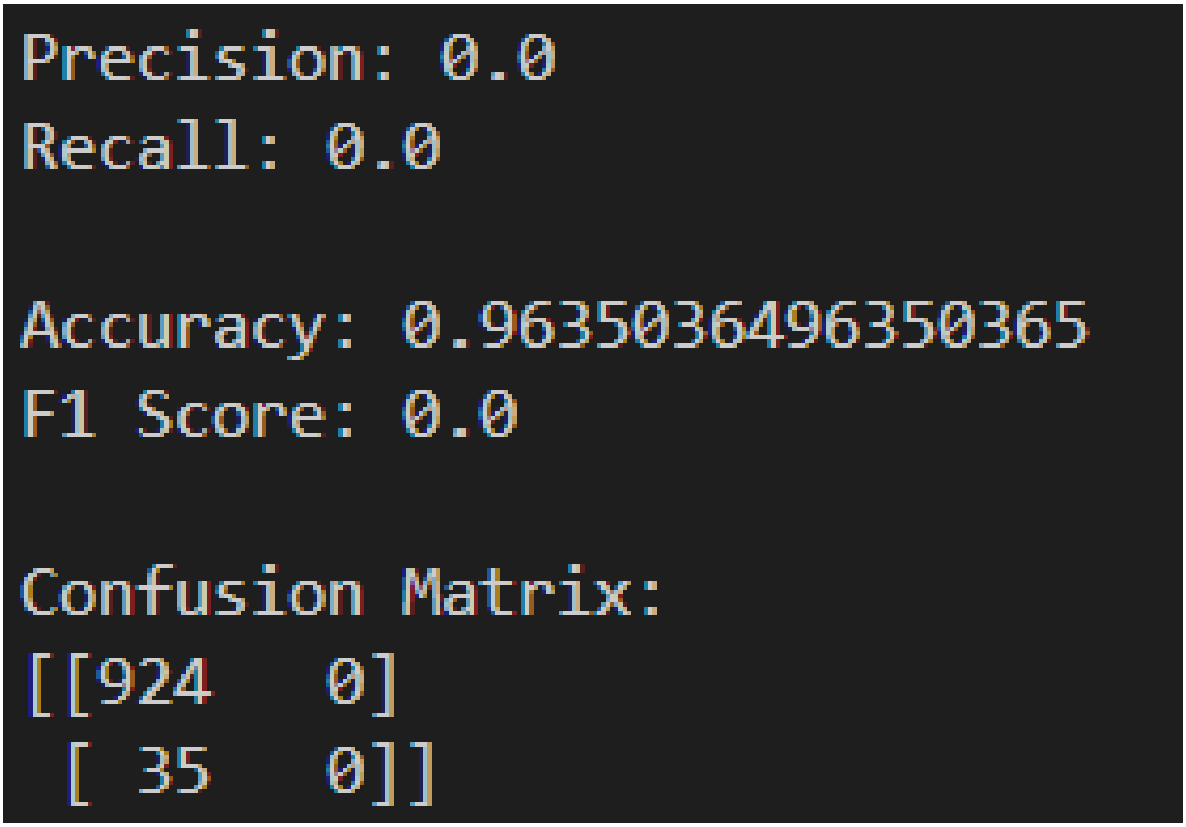

Figure 3: Performance of the Roc AUC model

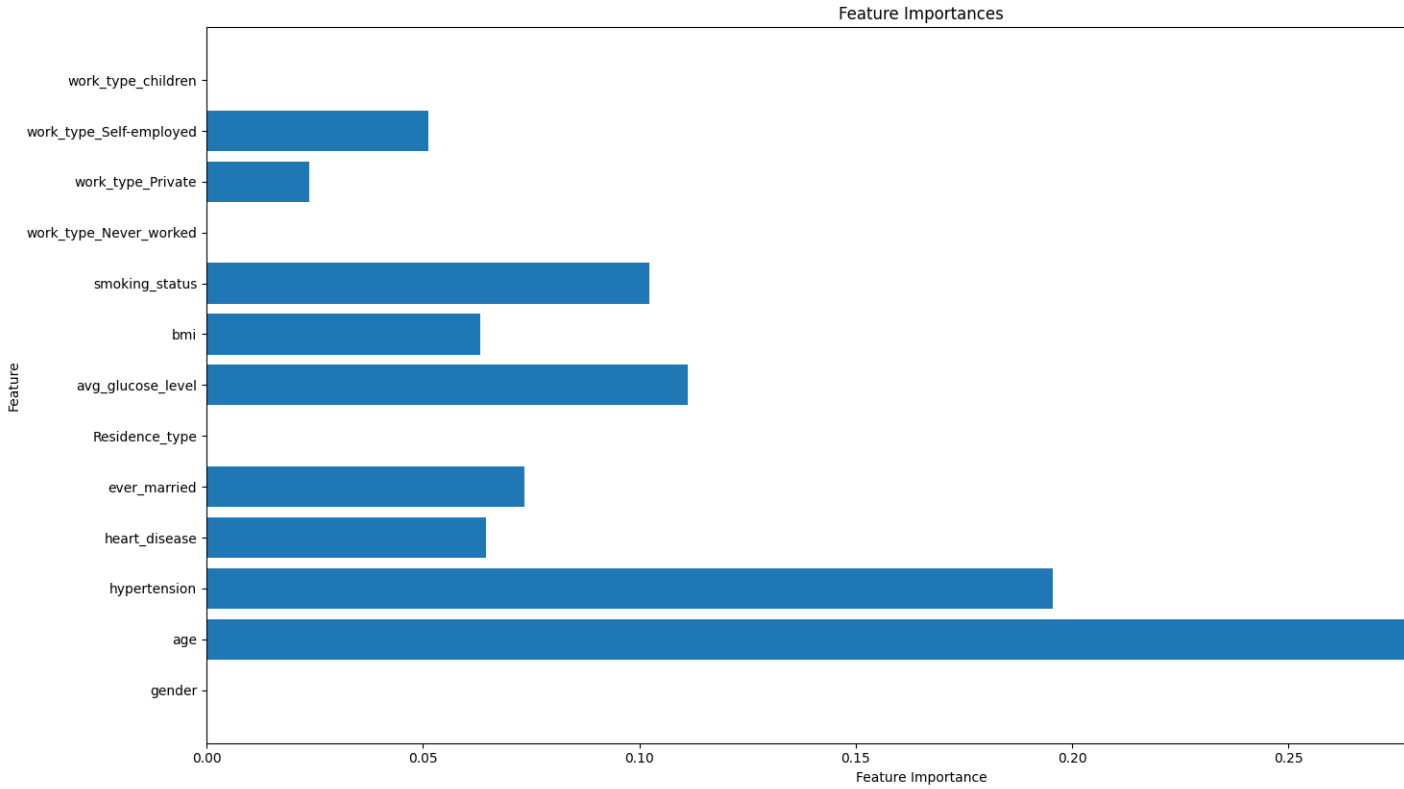

Figure 4: Feature importance of Roc AUC model

Unlike the accuracy model, this new model that maximizes the area under the ROC curve correctly identifies all of the features that are known risk factors for a stroke like bmi and smoking status. However it is worth noting that some of the identified features are false positives. For example, being self-employed is shown to have an almost equally significant impact on stroke risk as having heart disease. But despite these false positives it greatly outperforms the feature importances that the accuracy model displayed. When it comes to the overall performance of the model it shows the exact same bad performance as the accuracy model which is not acceptable so more testing had to be done to find a suitable model. For the next model the popular F1 metric was chosen.

```
Precision: 0.2608695652173913
Recall: 0.17142857142857143

Accuracy: 0.9520333680917622
F1 Score: 0.20689655172413793

Confusion Matrix:
[[907  17]
 [ 29   6]]
```

Figure 5: Performance of the F1 model

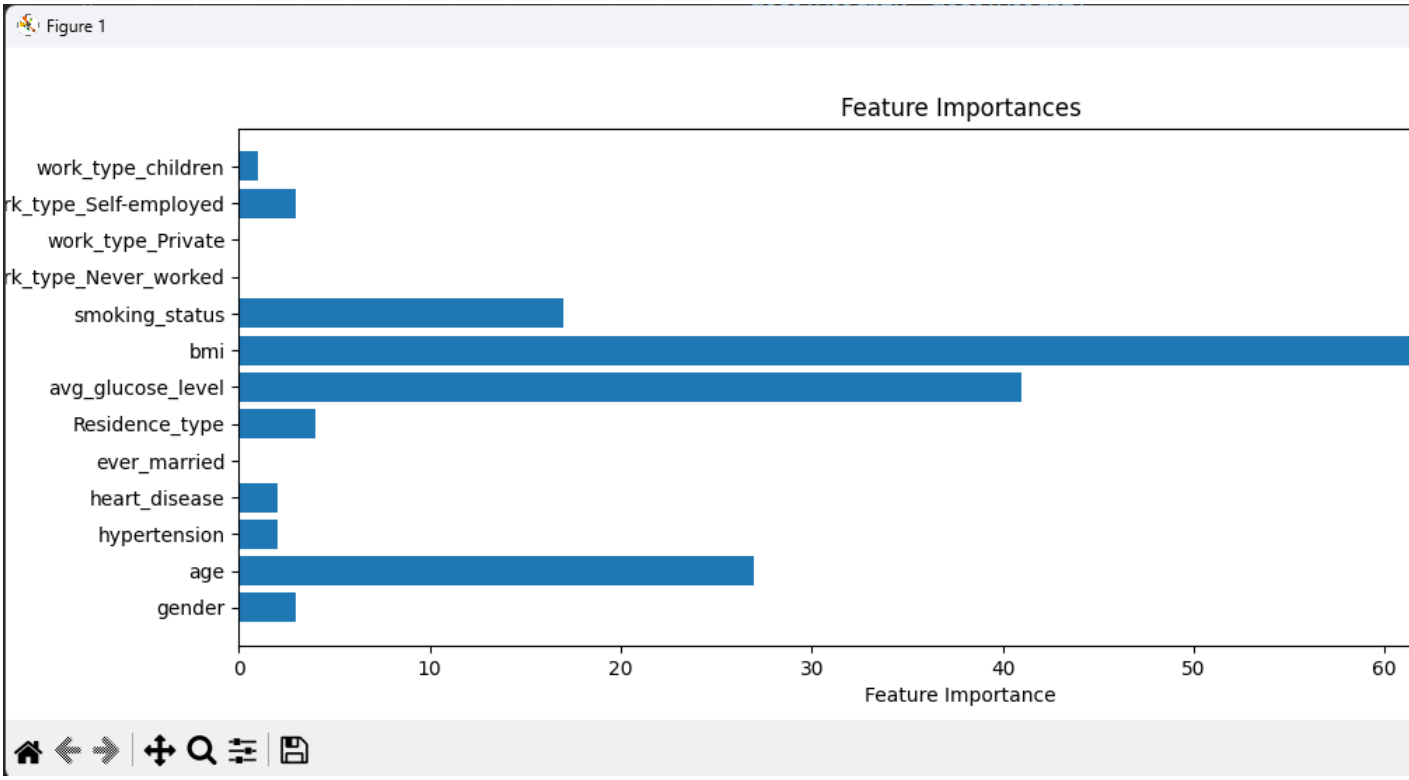

Figure 6: Feature importance of F1 model

Important note is that the confusion matrix i create using a function from the scikit-learn library.

```
cm = confusion_matrix(y_test, y_pred)
```

The confusion matrix that gets created is inverted meaning instead of having (1,1) in the second, quadrant it has (0,0).

```
               Predicted
               Negative    Positive
Actual Negative    TN          FP
       Positive    FN          TP
```

It seems this model performed for the first time displayed some balance in both performance and feature importance. The F1 model was the first model that could identify positive cases with a recall value of 17% meaning it correctly identified 17% of the positive cases. (6/35).

Even though this was an improvement in the recall value it was not sufficiently large, it is not really acceptable to only stop 17% of people from suffering a stroke and letting 29 out of 35 people potentially die as a result. In the scenario for this assignment, if you compare the potential harm of false positives and false negatives you could assume that false negatives is the one you want to prevent reduce more.

Seeking to improve the recall value i read online that you can apply class weights for a classification model using the sample weight paramter in the AutoML fit function.

```
automl.fit(X_train, y_train, task="classification", sample_weight=sample_weightArr, time_budget=60, *
```

Seeing as only 17 out of 924 cases of the 0 class get incorrectly predicted, the 0 class does not need any help so the weight labeled in the code as "weight0" was set to 1 for all the training runs while "weight1" was tweaked alot in between runs. After training multiple models with different values to weight1 i discovered that the overall performance of the models started to significantly decline when weight1 exceeded 17 so that is what the weight was finally set to which seemed to deliver the best overall results out of all the models.

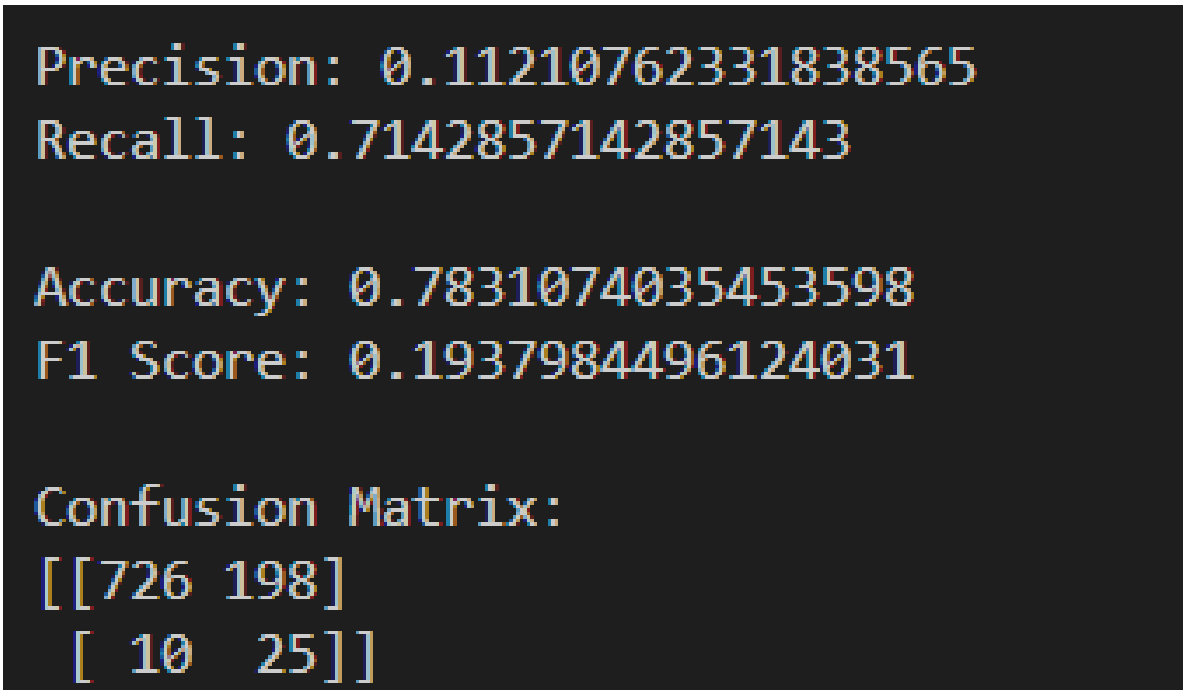

Figure 7: Performance of the weighted F1 model

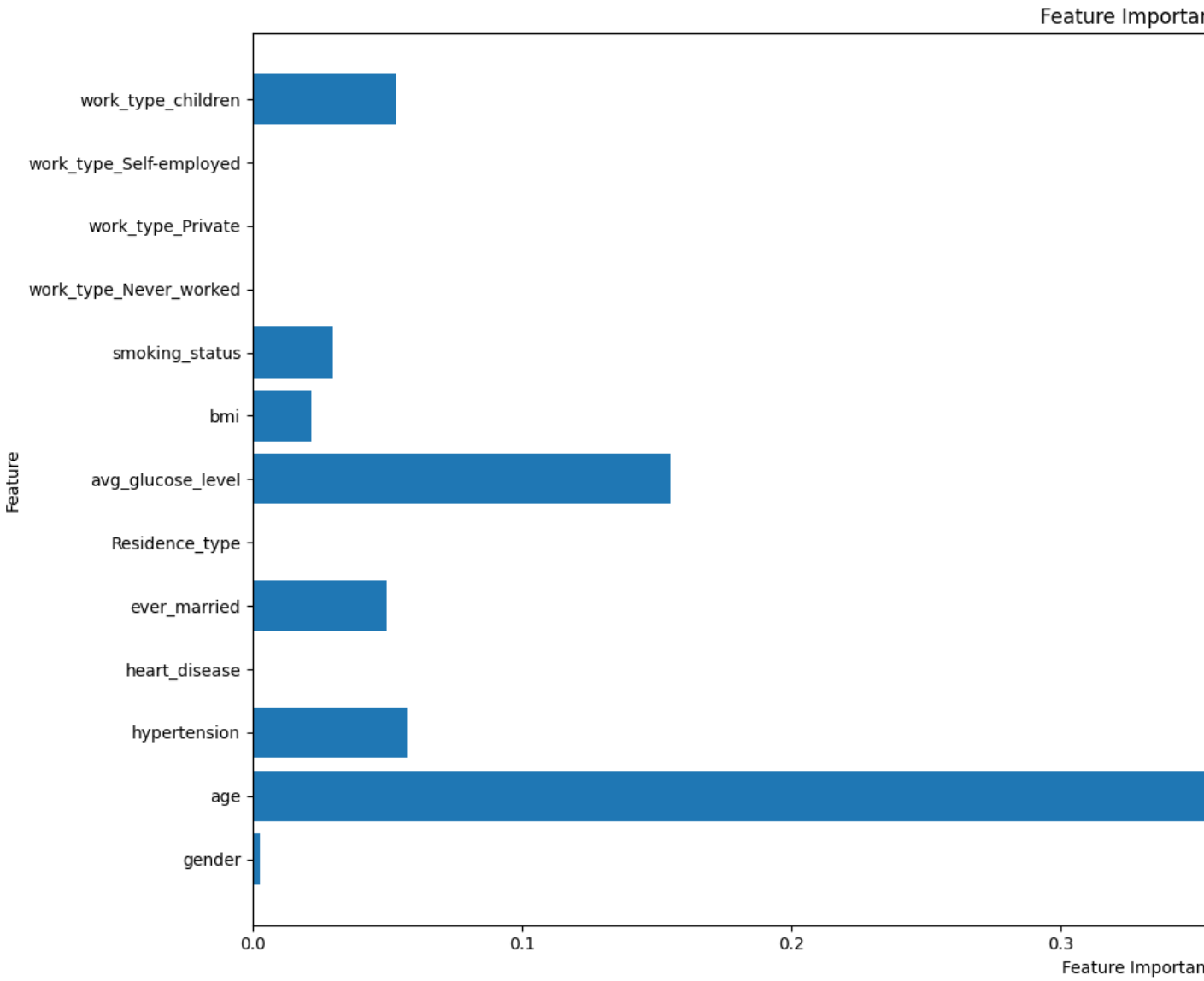

Figure 8: Feature importance of the weighted F1 model

Applying weights to our F1 model achieved almost the desired outcome perfectly. We significantly increased the recall value from 17% to 71% at the cost of increasing our false positives which like i said is likely the least damaging negative performance metric considering the use case of our model.

When it comes to feature importance the f1 model doesnt quite outcompete the roc auc model in its balance in risk factors, one could maybe argue that some features should be valued a little bit more and that the importance of age should be tuned down. Despite this the important thing is that it identifies nearly all the known risk factors but undervalues them instead of not identifying them at all, like the accuracy model did.

6 Model deployment

Feeling satisfied with "F1WeightedModel" i decided to create a streamlined upload functionality file called "UploadModels.py" that logs in to the Hugging face repository using my token i generated. In order to execute this python function you need the token generated which is in the file token.txt in the git repository.
This function takes a pickle filename and simply pushes it to the repository "FinalProjectRepo" https://huggingface.co/David991199/FinalProjectRepo/tree/main

This repository is in turn utilized by the Hugging Face space i have setup called "FinalProjectSpace" on my Hugging face account David991199. https://huggingface.co/spaces/David991199/FinalProjectSpace/tree/main