Machine Learning: Imbalanced Stroke Dataset

2025-11-19

```python
import sys
assert sys.version_info >= (3, 7)
import sklearn #"scikit-learn"
assert sklearn.__version__ >= "1.2"
import numpy as np
import pandas as pd
import matplotlib as plt
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
#assert pd.__version__ >="1.5.0"

from sklearn.preprocessing import OneHotEncoder
import seaborn as sns
```

Decision Tree and Random Forest: Stroke or no Stroke?

Assessing the risk of a stroke from an imbalanced dataset

The "health-care-dataset-stroke-data.csv" from kaggle is a dataset from Kaggle that classifies patients who had a stroke from patients that didnet have a stroke and some other health conditions. This short report shows a model that will assess the risk of a stroke given health conditions that are present in the current database.

Such a model can be critical to infer patients health risks given, e.g. age or blood sugar to assess whether preventative measures should be advised and started.

Initial thoughts

What could be possible criteria for such a model? There are aspects in such a model that are more important than simple precision. Assessing a risk can be a bit like signal detection theory. The metrics from signal detection are usually presented in a confusion matrix with: 1. True positives (correctly identifies a positive case), 2. True negatives (correctly identifies a negative case), 3. False positives (incorrectly identifies a positive case) and 4. False negatives (incorrectly identifies a negative case).

However, fur the purpose of assessing a stroke risk and for a medical practicioner to make a meaningful decision some of the metrics from signal detection theory are more crucial than others. Correctly assessing a patient in time can help save the patients life. Thus, for our model to work, it is crucial it returns as little **false negatives** as possible. Missing a patients risk for a stroke can have to most severe consequences. **False positives** are less severe since they might indicate that risk factors are met, which not necessarily lead to a stroke. However, the biggest risk for that patient when applying preventative measures might be side effects from medication.

Thorough Analysis of the Data

Check the data and the various distributions of information for stroke vs. no-stroke patients in the dataset.

There is a strong imbalance in the dataset with only 250 stroke patients in comparison with 4861 patients, who haven't had a stroke (yet). That is why the data will have to be stratified for the train-test-split, ensuring that the same distribution of the various factors appears in the trainings and the test set. Due to their small number, their should be more weight on the data of the stroke patients.

These are the following column names and data types in the dataset:

```
## Spaltennamen Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
##        'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
##        'smoking_status', 'stroke'],
##       dtype='object')
```

Column Assessment 1. How useful are the columns for the purpose of this model? 2. What kind of information can their information provide? 3. How could the labels be interpreted?

Medical information

These seem to be information directly associated with medical conditions.

age numerical – useful information, - the likelihood of a stroke increases with age.

glucose level numerical – useful information - increased glucose levels harm the blood vessel, which can lead to clots and thus can induce a stroke.

hypertension categorical – useful information - medication that treats hypertension, can also decrease the risk for a stroke

heart disease categorical – sinnvolle Spalte für Fragestellung - most heart disease a connected with a risk for stroke → heart disease can lead to blood clots which can lead to a stroke

bmi numerical – useful for assessment of risk for a stroke - → correlation with diabetes and hypertension, which are established risk factors of a stroke. - lots of missing values for bmi: 201 missing from dataset that is 5110 in size → transpute missing values

Smoking Status categorical – use to assess risk for a stroke - arterial constriction might cause blood clots which then could lead to strokes

- There are a lot of children in the dataset and only 2 children with a stroke:
  - considersegmentation of the data should be considered: -→ maybe consider only data without the children

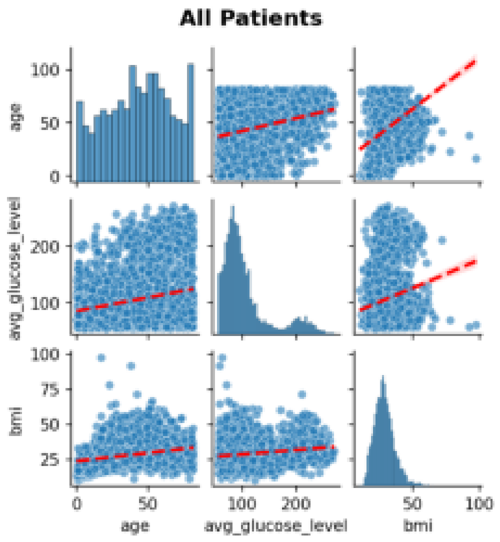
Stress factors

All the other columns appear to be related to some form of potential stress factors (ever married, gender, work type, residence type (urban or rural)). However, the labeling is unclear and the interpretation of the results are somewhat spurious. A Label like *ever married* is difficult to interpret since its meaning is far to broad and vague (does it mean: recently married, divorced, recently divorced, long-time divorced, long-time married?). The different statuses of *ever married* will have decidedly different effect on the stress levels.

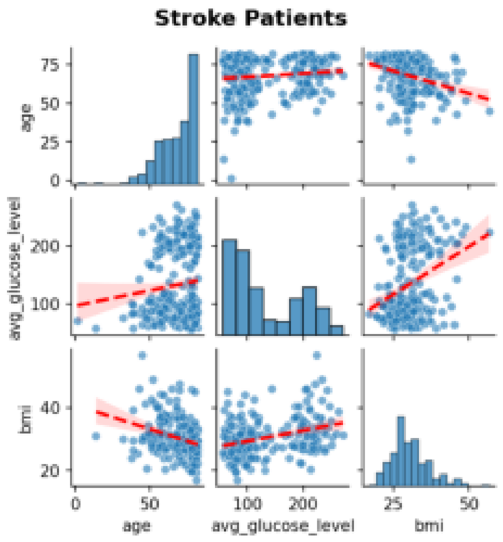
Separation of data into stroke and no-stroke patients

Correlation plots of the numerical data for all patients vs stroke patients

No Stroke Patients



Stroke Patients

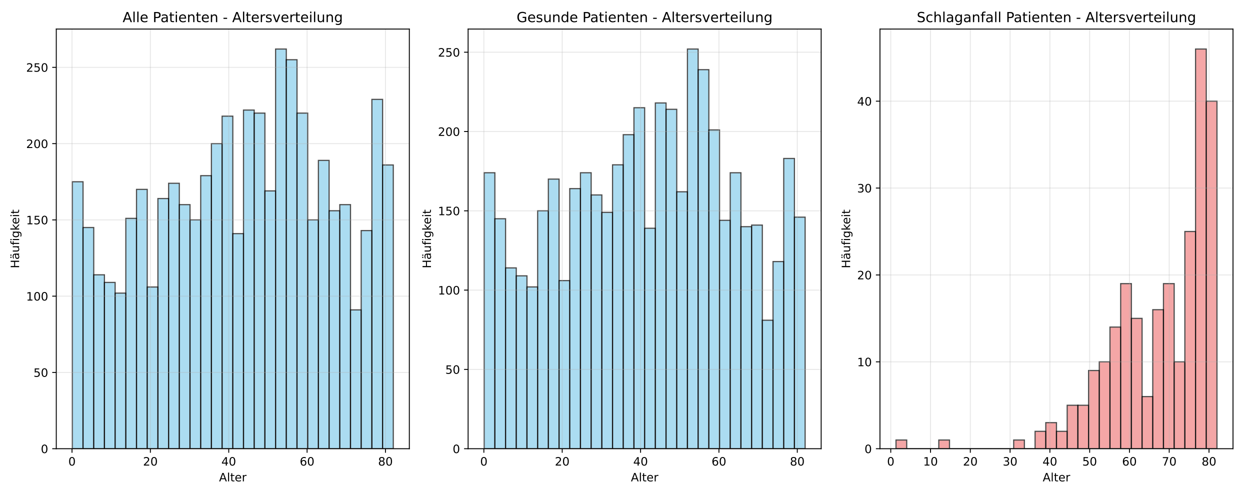


- side-by-side the plots show the big imbalance of the data between both groups (no-stroke vs stroke)
- all three columns show different patterns,
- glucose levels it is noteworthy that stroke patients seem to be separated into 2 groups → high ↔ low blood sugar
- age appears to be a huge risk factor for having a stroke:

Age distribution

Comparison

1. age distribution
2. no-stroke patients
3. stroke patiens



Observation children: - the data set contains 1025 children under the age of 20 → there are only 2 children in the stroke group - in addition there shouldnt be too many smokers in the children group

```
print("Allgmeine Info zu Smoking Status","\n",df["smoking_status"].value_counts(),"\n")
```

```
## Allgmeine Info zu Smoking Status
##  smoking_status
## never smoked       1892
## Unknown            1544
## formerly smoked     885
## smokes              789
## Name: count, dtype: int64
```

```
# print out the value counts for the smoking status column for people below 20
print("Info zum smoking bei Kindern unter 20:","\n",df[(df['age'].astype(np.int8) < 20)]["smoking_statu
```

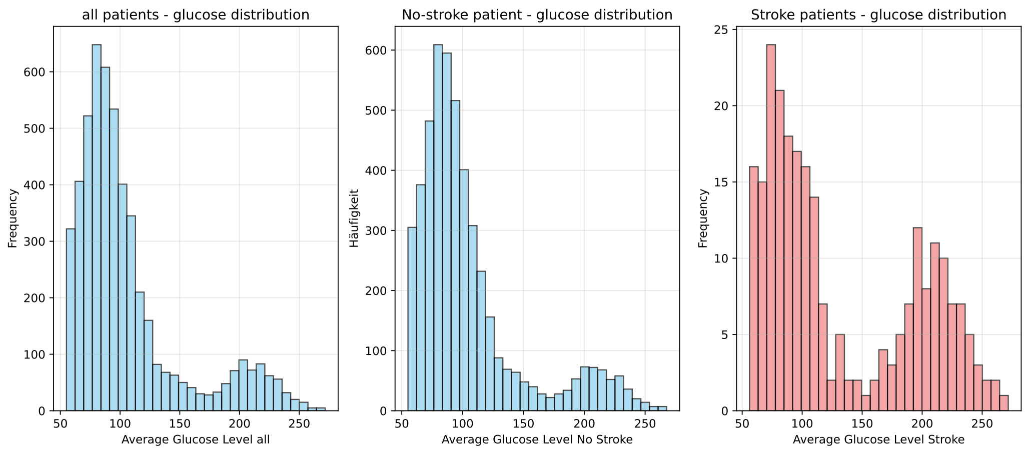
```
## Info zum smoking bei Kindern unter 20:
##  smoking_status
## Unknown            731
## never smoked       187
## formerly smoked     29
## smokes              19
## Name: count, dtype: int64
```

```
print("Work Type:","\n",df["work_type"].value_counts())
```

```
## Work Type:
##  work_type
## Private          2925
## Self-employed     819
## children          687
## Govt_job          657
## Never_worked       22
## Name: count, dtype: int64
```

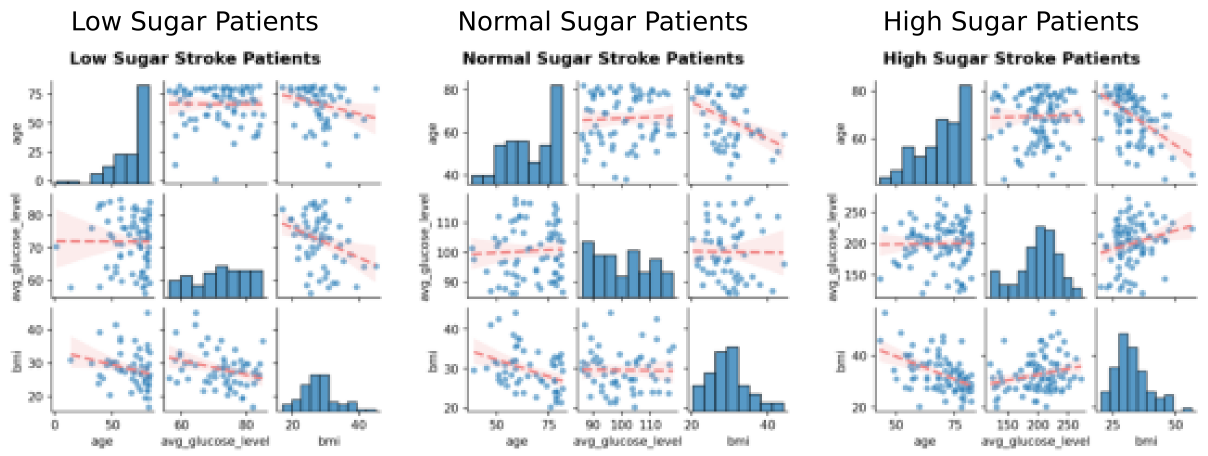
Level of glucose distribution

Comparison 1. all patients 2. no-stroke patients 3. stroke patients



There are spikes in glucose levels for low glucose and for high glucose in the stroke group. This might be not too surprising since there are two types of diabetes related with glucose levels. Diabetes associated with low glucose (hypoglycemia) and with high glucose (hyperglycemia) levels. This polynomial distribution is also visible in the plots. To further analyse, patients have been separated into three groups.

1. low glucose level
2. normal glucose level
3. high glucose level



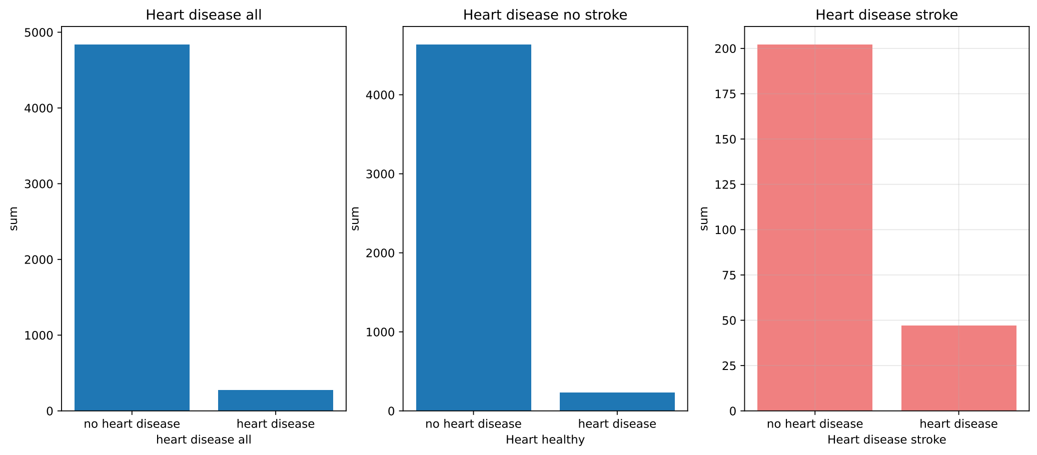
Noteworthy Findings

Stroke patients have been separated into 3 glucose level groups: low, normal and high. It is interesting to see that the correlation between glucose and the bmi is reversed: with the correlation between low_glucose x bmi being negative (the lower the glucose the higher the bmi) and high_glucose x bmi being positive (the higher the glucose the higher the bmi).

This information about the different effects from the glucose level must be integrated with the Machine Learning Analysis later.

Further group comparisons follow.

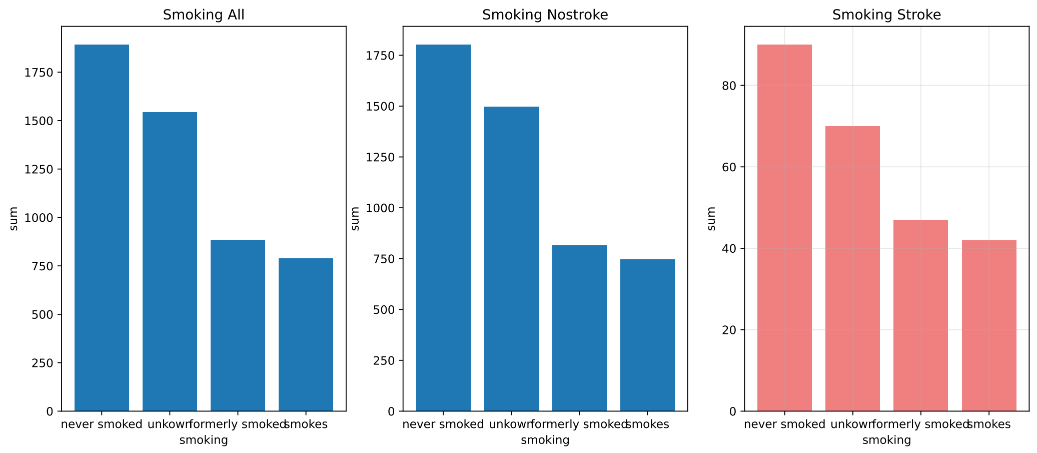
Barplots Heart Disease



Barplots Smoking Status

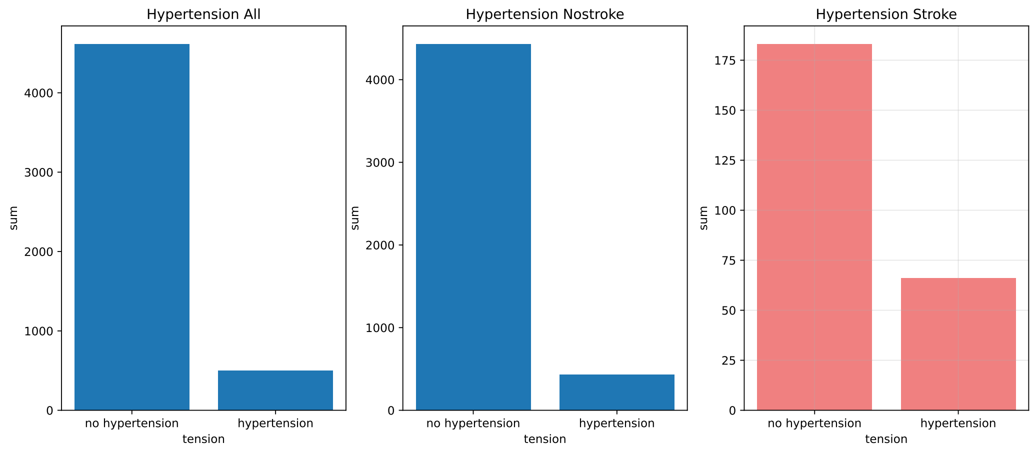
```
smoke_stroke = df_stroke["smoking_status"].value_counts()
smoke_Nostroke = df_Nostroke["smoking_status"].value_counts()
smoke_all = df["smoking_status"].value_counts()

fig, (ax1, ax2,ax3) = plt.subplots(1, 3, figsize=(15, 6))

ax1.bar(["never smoked","unkown","formerly smoked","smokes"],smoke_all)
ax1.set_xlabel('smoking')
ax1.set_ylabel('sum')
ax1.set_title('Smoking All')


ax2.bar(["never smoked","unkown","formerly smoked","smokes"],smoke_Nostroke)
ax2.set_xlabel('smoking')
ax2.set_ylabel('sum')
ax2.set_title('Smoking Nostroke')

ax3.bar(["never smoked","unkown","formerly smoked","smokes"],smoke_stroke,color='lightcoral')
ax3.set_xlabel('smoking')
```

```
ax3.set_ylabel('sum')
ax3.set_title('Smoking Stroke')
ax3.grid(True, alpha=0.3)
plt.show()
```



Barplots Hypertension

```
tension_stroke = df_stroke["hypertension"].value_counts()
tension_Nostroke = df_Nostroke["hypertension"].value_counts()
tension_all = df["hypertension"].value_counts()

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 6))

ax1.bar(["no hypertension","hypertension"],tension_all)
ax1.set_xlabel('tension')
ax1.set_ylabel('sum')
ax1.set_title('Hypertension All')

ax2.bar(["no hypertension","hypertension"],tension_Nostroke)
ax2.set_xlabel('tension')
ax2.set_ylabel('sum')
ax2.set_title('Hypertension Nostroke')

ax3.bar(["no hypertension","hypertension"],tension_stroke,color='lightcoral')
ax3.set_xlabel('tension')
ax3.set_ylabel('sum')
ax3.set_title('Hypertension Stroke')
ax3.grid(True, alpha=0.3)
plt.show()
```



Decision Tree for Analysis

Advantages

- + filter of relevant columnsrelevante Spalten lassen sich herausfiltern
- + for classification task
- + overfitting can be controlled

Nachteil - - risk of bias when data is not homogeneous → since data is very imbalanced, this risk must be considered when assessing the results

I've used Decision Tree with a Gridsearch, stratifying the data in preparation. Additionally, glucose levels were separated into 3 groups (low: below 85, medium: between 85 and 120, high: above 120). Using one-hot encoders, glucose levels were categorized.

```
# train-test split method
# split data into trainings and test set using stratification to make sure that the target (troke patie
# 20 % of the data for testset and 80% of the data for training

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Define the order for ordinal encoding
category_orders = [
    ["Male", "Female"],
    ["No", "Yes"],
    ['children', 'Govt_job', 'Private', 'Self-employed', 'Never_worked'], # ordered by estimated stress
    ["Rural", "Urban"],
    ["never smoked", "formerly smoked", "smokes"],
]

# column transformer for categorical columns
transformer_category_columns = ColumnTransformer(

transformers=[
    (
        "ordinal_encoder",
        OrdinalEncoder(
            categories=category_orders,
            handle_unknown="use_encoded_value",
            unknown_value=np.nan,
        ),
        [0, 4, 5, 6, 9],  # Indices of categorical columns to encode
    ),
    (
        "onehot_encoder",
        OneHotEncoder(
            categories="auto",
            handle_unknown="ignore",  # recommended to avoid errors at inference  # optional: returns d
        ),
        [10, 11, 12], # one hot encoder for additional 3 columns -- three different glucose levels
    ),
],
remainder="passthrough",
)

# complete data preparation pipeline
data_preparation_pipeline = Pipeline(
    steps=[
        ("transformer_category_columns", transformer_category_columns),
        ("str_to_float", FunctionTransformer(lambda X: np.where(X == 'N/A', 'nan', X).astype(float))),
        ("scaler", MinMaxScaler()),
        ("imputer", KNNImputer(n_neighbors=10)),
    ]
)

# prepare the training and test data
X_train_prepared = data_preparation_pipeline.fit_transform(X_train)
X_test_prepared = data_preparation_pipeline.transform(X_test)
```

Code for Decision Tree Implementation

```
###################################
# Decision tree with Gridsearch
######################################

tree_para = {'criterion':['entropy'],'splitter':['best'],'max_depth':[16,18,20,25],"min_samples_leaf":[5
clf = GridSearchCV(DecisionTreeClassifier(), tree_para, cv=5)

tree = DecisionTreeClassifier(max_depth=20,min_samples_leaf=4) # to handle the imbalanced dataset
tree.fit(X_train_prepared, y_train)
```

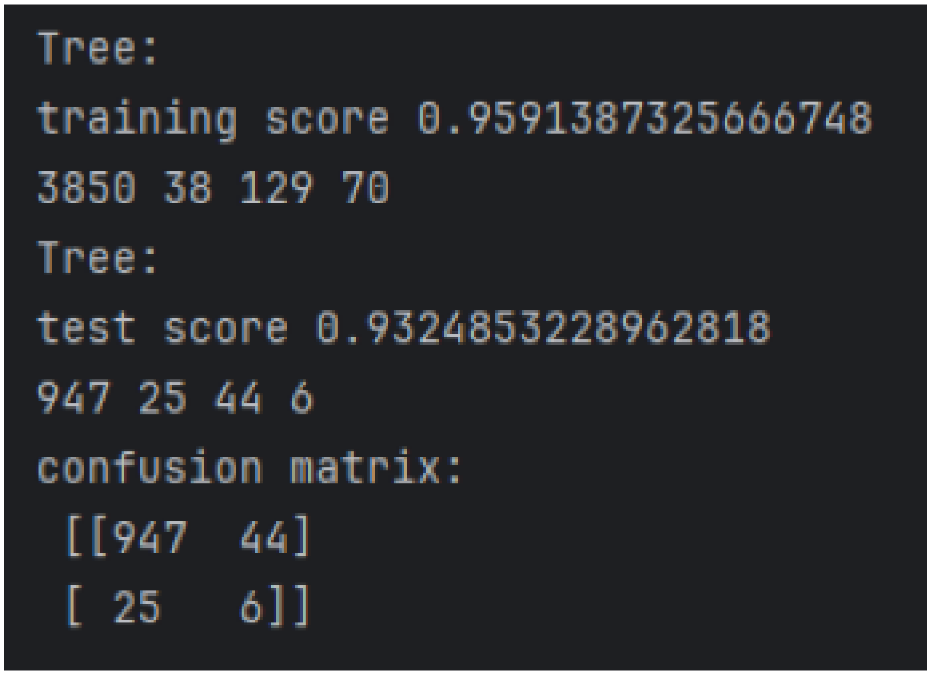
**Result of Analysis with Decision Tree

```
import cv2 as cv


dtg = cv.imread("DecisionTreeGridsearch.png")

# Convert the image from BGR to RGB
```

```
cut_dtg = cv.cvtColor(dtg, cv.COLOR_BGR2RGB)

# Display the image using Matplotlib
plt.imshow(cut_dtg)
plt.axis('off')  # Hide axes
```

```
## (np.float64(-0.5), np.float64(290.5), np.float64(209.5), np.float64(-0.5))
```

```
plt.show()
```



```
Tree:
training score 0.9591387325666748
3850 38 129 70
Tree:
test score 0.9324853228962818
947 25 44 6
confusion matrix:
 [[947  44]
 [ 25   6]]
```

The results are not good fit for our purpose. Even though training and testing score are not too bad, the relevant measures in the confusion matrix are not good enough. If there are only 6 patients correctly identified as patients with a risk for a stroke and in comparison 44 risk patients are incorrectly classified with no risk that is not fit for purpose and a huge risk factor for application in real-life health assessment. As somehow expected, the decision tree falls short when dealing with unbalanced data sets.

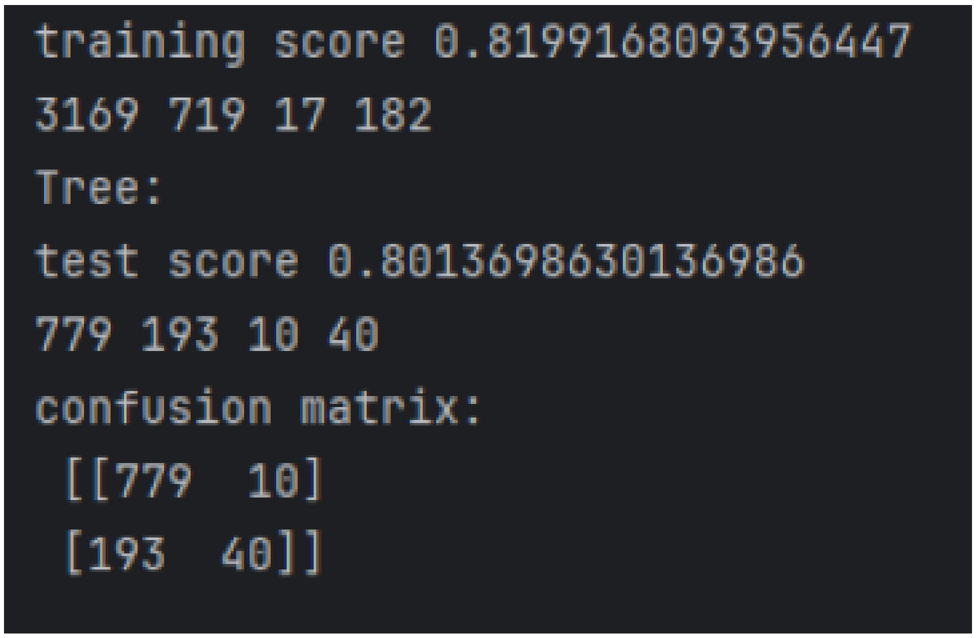
A next step would be the use of Ensemble methods that can deal with imbalanced data sets like the random forest classifier. Ensembling 2000 decision trees, the Gridsearch can be improved and the imbalance compensated. The following code shows the implementation of random fores with 2000 decision trees, weight (balanced, max_depths=20 and minimal sample leaves of 20)

Code Random Forest

```
X, y = make_classification(n_samples=2000, n_features=13)
clf = RandomForestClassifier(max_depth=20, random_state=42,min_samples_leaf=20,class_weight='balanced')
clf.fit(X_train_prepared, y_train)
```

```
print("\nTree:\ntraining score" , clf.score(X_train_prepared, y_train))
y_train_pred = clf.predict(X_train_prepared)

tn_train, fp_train, fn_train, tp_train = confusion_matrix(y_train,y_train_pred).ravel().tolist()
print("TN:", tn_train,"FP:", fp_train,"FN:", fn_train,"TP:", tp_train)

print("Tree: \ntest score" , clf.score(X_test_prepared, y_test))
y_test_pred = clf.predict(X_test_prepared)
tn, fp, fn, tp = confusion_matrix(y_test, y_test_pred).ravel().tolist()
print("TN:", tn,"FP:", fp,"FN:", fn,"TP:",tp)
# print("recall: ", recall_score(y_test, y_test_pred))

print("confusion matrix: \n", confusion_matrix(y_test_pred, y_test))
```

Confusion Matrix Random Forest



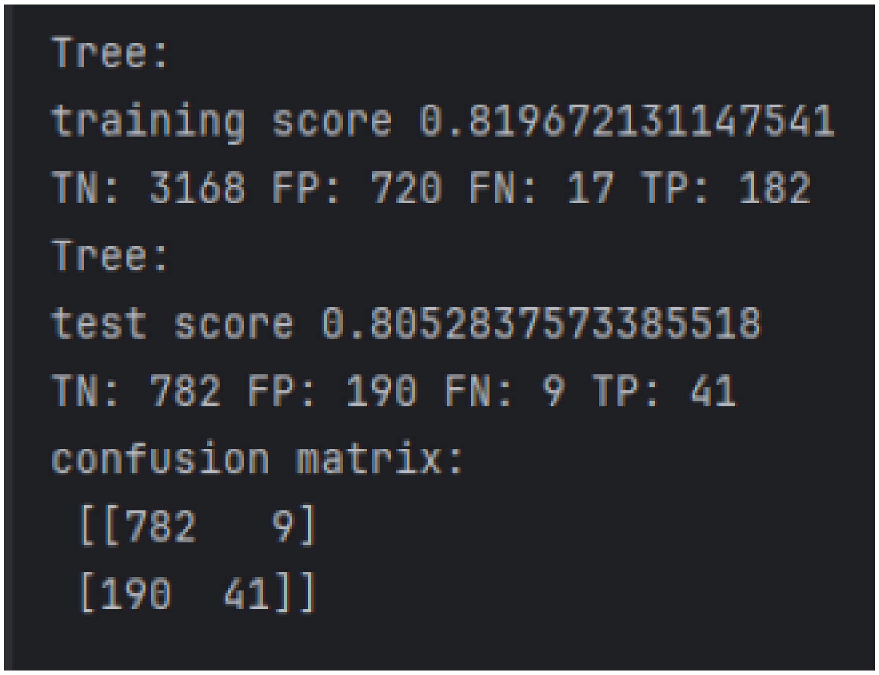
```
training score 0.8199168093956447
3169 719 17 182
Tree:
test score 0.8013698630136986
779 193 10 40
confusion matrix:
 [[779  10]
 [193  40]]
```

Tests show that only 10 peope were incorrectly categorised as healthy, while 40 people were correctly classified as risk patients for a stroke. Even though 193 were incorrectly classified as risk patients, their risk of misclassifictations were relatively minor in comparison to the False Negatives. These results showed a significant improvement to the Decision Tree analysis.

However, in the initial analyses the dataset was separated into 3 levels of glucose (low, medium, high). Interestingly, the correlations between glucose level and BMI completely reverse between high and low glucose groups. Glucose x BMI had a negative correlation while high glucose x BMI had a positive correlation. Thus, the role of the glucose level has been incorporated in a finel Random Forest Analysis.

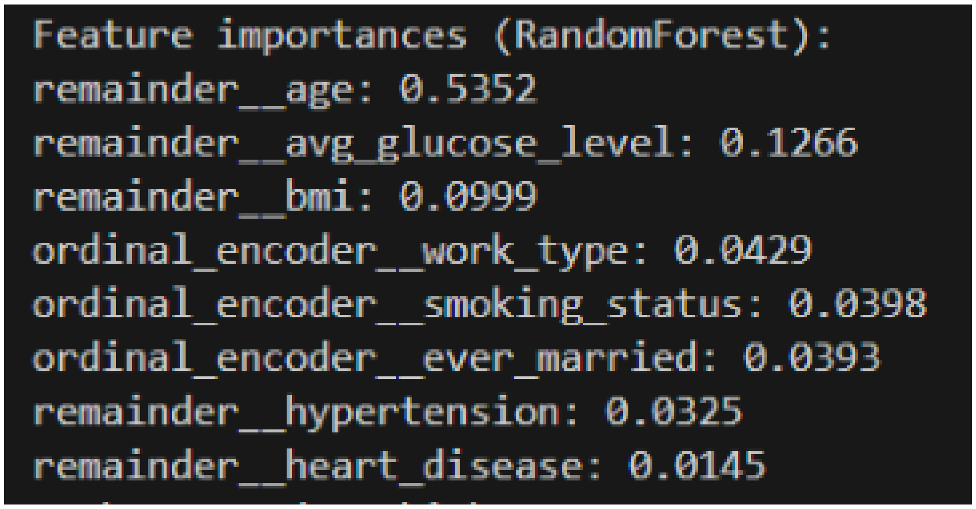
Bei der Analyse hat sich gezeigt, dass sich die Korrelation zwischen Glucose und BMI bei den Stroke Patienten umdrehen. Glucose level x BMI hatte eine negative Korrelation in der low glucose Gruppe, keine Korrelatin bei der normalen Gruppe, aber dann eine positive Korrelation bei der high glucose Gruppe. Damit diese Information beim Lernprozess berücksichtigt werden kann, aber ich eine One Hot Kodierung der 3 Gruppen der Glucosewerte durchgeführt und diese 3 Spalten an den bestehenden Datensatz angehangen. Darauf habe ich denselben RandomForestClassifier mit denselben Parametern angewandt und eine minimale Verbesserung bei der Erkennung erreicht.

While the Random Forest Classification showed a significant improvement, a final analysis incorporated that additional glucose level classification.



```
Tree:
training score 0.819672131147541
TN: 3168 FP: 720 FN: 17 TP: 182
Tree:
test score 0.8052837573385518
TN: 782 FP: 190 FN: 9 TP: 41
confusion matrix:
 [[782   9]
 [190  41]]
```

There has been a small, but crucial improvement in the results. 5 participants more have been correctly classified reducing the number of false negatives from 10 to 9. The number of false positives has been reduced by 3 people and there was 1 more patient that has been correctly identified as a risk patient for a stroke.



```
Feature importances (RandomForest):
remainder__age: 0.5352
remainder__avg_glucose_level: 0.1266
remainder__bmi: 0.0999
ordinal_encoder__work_type: 0.0429
ordinal_encoder__smoking_status: 0.0398
ordinal_encoder__ever_married: 0.0393
remainder__hypertension: 0.0325
remainder__heart_disease: 0.0145
```

The feature importance unsurprisingly shows that age is by far the biggest risk factor for a stroke followed by glucose levels and BMIs.

These results from our Machine Learning analysis are far from perfect and given the high risk of false negatives for patients, this model is also not fit for purpose. However with a highly imbalanced Dataset, the Random Forest Classifier with the glucose level classification returns the best results in this machine learning project.