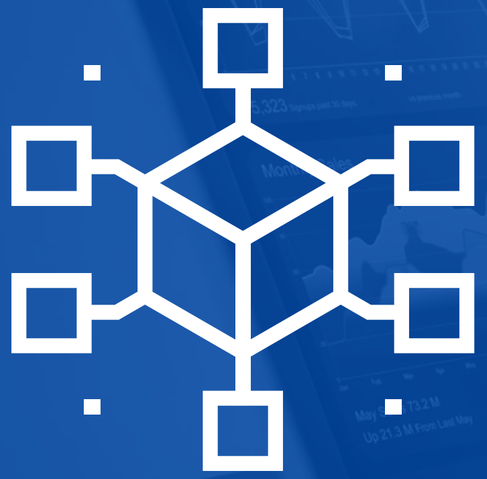
Brainstation, Data Science



Final Project: Stroke Prediction

Diane McEvoy, 2024

Stroke Prediction - Overview

01 Problem & Goal

02 Data at a Glance

03 Data Cleansing

04 EDA Visualizations

Exploring Algorithms 05

Final Model 06

Next Steps 07

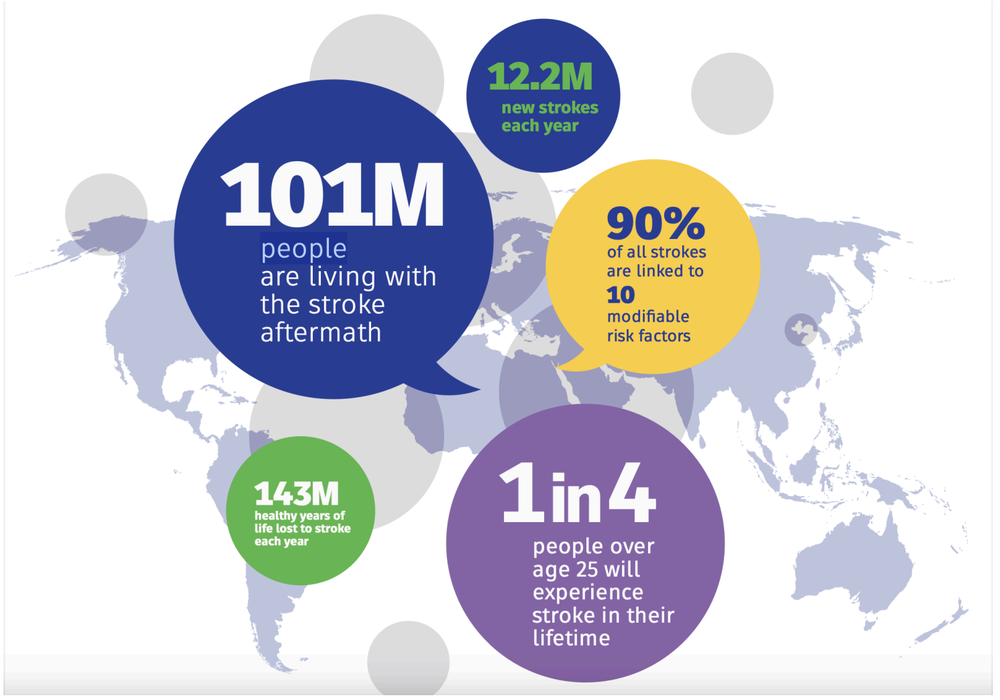
Questions? 08

The Problem

- Strokes are a leading cause of death and disability globally.
- Affects over 62,000 Canadians annually (Heart & Stroke Foundation).
- Early prediction empowers individuals to take preventive actions, reducing the risk of strokes and their long-term consequences like paralysis, speech difficulties, and cognitive impairment.

The Goal

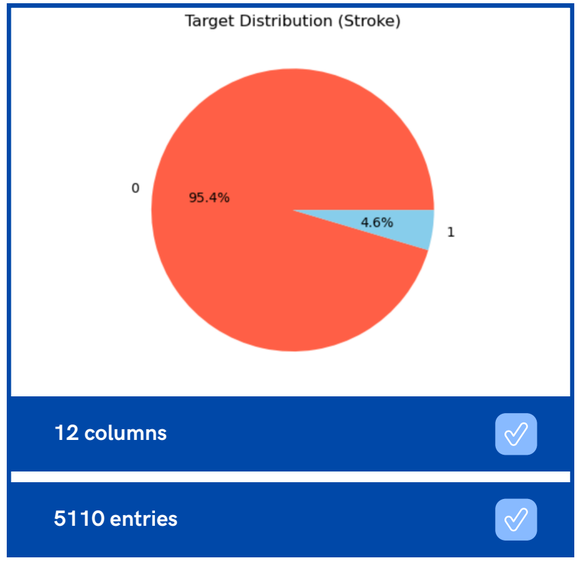
Build a predictive model to identify individuals at risk for stroke, based on key health metrics.



The Dataset

Source: https://www.kaggle.com/fedesoriano

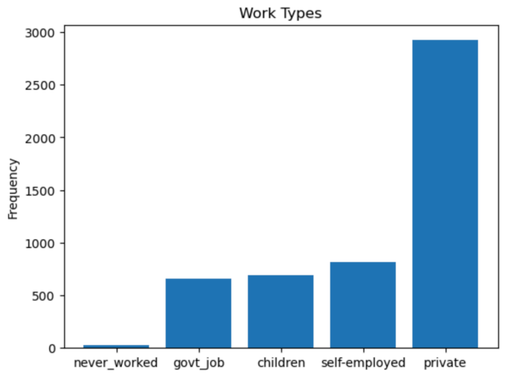
```
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
```



id	gender (male, female, other)	age	hypertension (0/1)	heart_disease (0/1)	ever_married (yes/no)	work_type (children, private, self-emp'd, government)	Residence_type (urban, rural)	avg_glucose_level	bmi
9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6
51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A
31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5
60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4

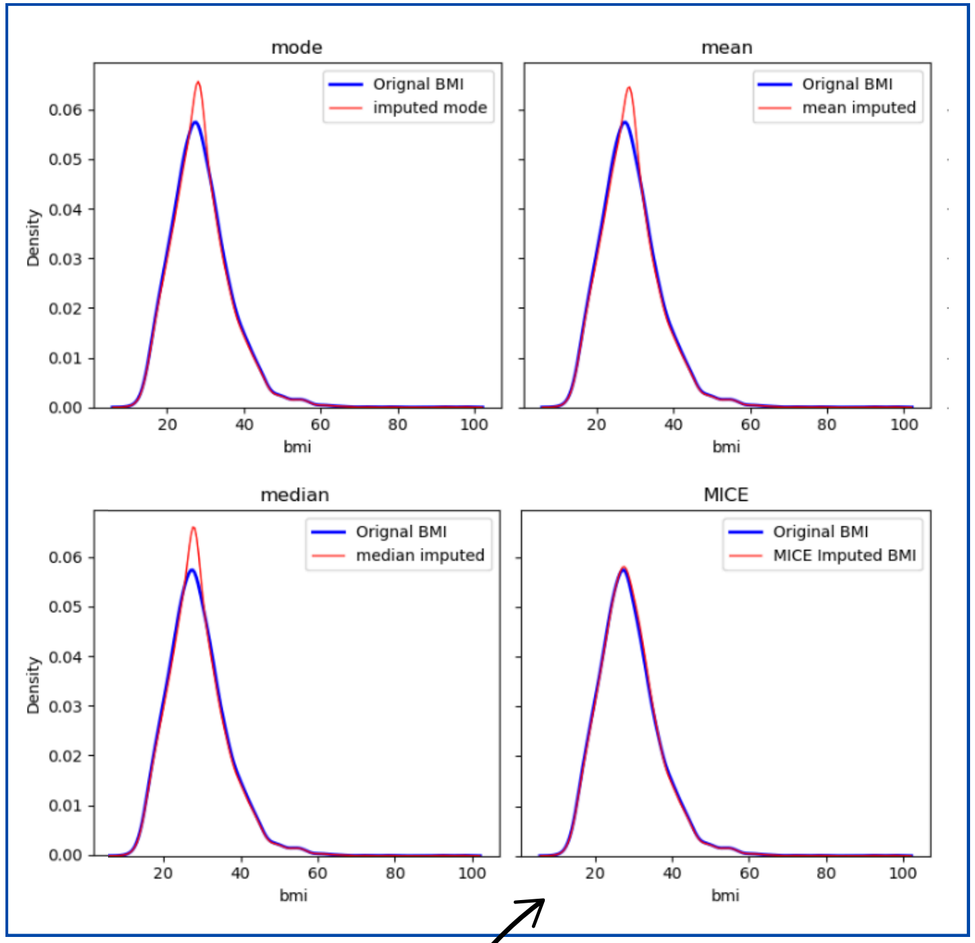
Data Cleansing

- Formatted data types (age, float to integer)
- Handled missing values: Imputed 201 missing values from BMI column (4%) - tried various methods, eventually using Multiple Imputation by Chained Equations (MICE).
- Feature engineering: Converted categorical variables into numerical - originally used mapping, but finally applied One Hot Encoding
- Reset the index with ID numbers



work_type_Never_worked	20
work_type_Private	2334
work_type_Self-employed	657
work_type_children	539

One Hot Encoding for categorical features



MICE imputation produced best results

Data Cleansing

- Data types all numerical or Boolean
- 23 columns after OHE

```
RangeIndex: 4088 entries, 0 to 4087
Data columns (total 23 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   age                             4088 non-null   int64
 1   avg_glucose_level               4088 non-null   float64
 2   gender_Female                   4088 non-null   bool
 3   gender_Male                     4088 non-null   bool
 4   hypertension_0                  4088 non-null   bool
 5   hypertension_1                  4088 non-null   bool
 6   heart_disease_0                 4088 non-null   bool
 7   heart_disease_1                 4088 non-null   bool
 8   ever_married_No                 4088 non-null   bool
 9   ever_married_Yes                4088 non-null   bool
 10  work_type_Govt_job              4088 non-null   bool
 11  work_type_Never_worked          4088 non-null   bool
 12  work_type_Private               4088 non-null   bool
 13  work_type_Self-employed         4088 non-null   bool
 14  work_type_children              4088 non-null   bool
 15  residence_type_Rural            4088 non-null   bool
 16  residence_type_Urban            4088 non-null   bool
 17  smoking_status_Unknown          4088 non-null   bool
 18  smoking_status_formerly smoked  4088 non-null   bool
 19  smoking_status_never smoked     4088 non-null   bool
 20  smoking_status_smokes           4088 non-null   bool
 21  stroke                          4088 non-null   int64
 22  bmi_bell_imputed                4088 non-null   float64
dtypes: bool(19), float64(2), int64(2)
```

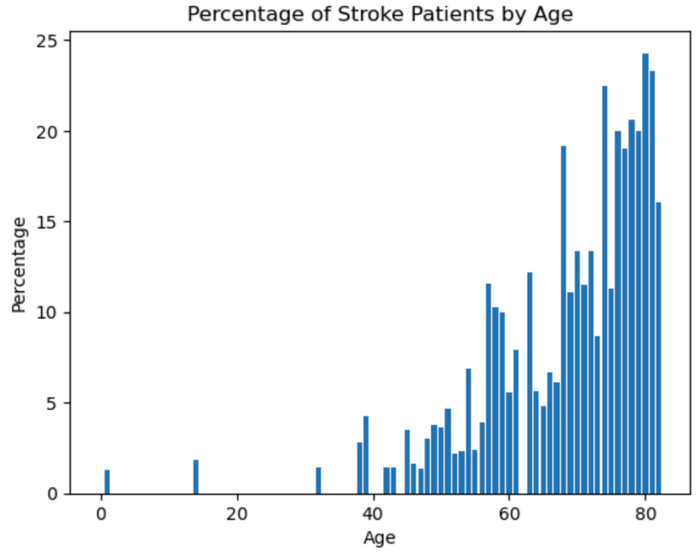
target feature

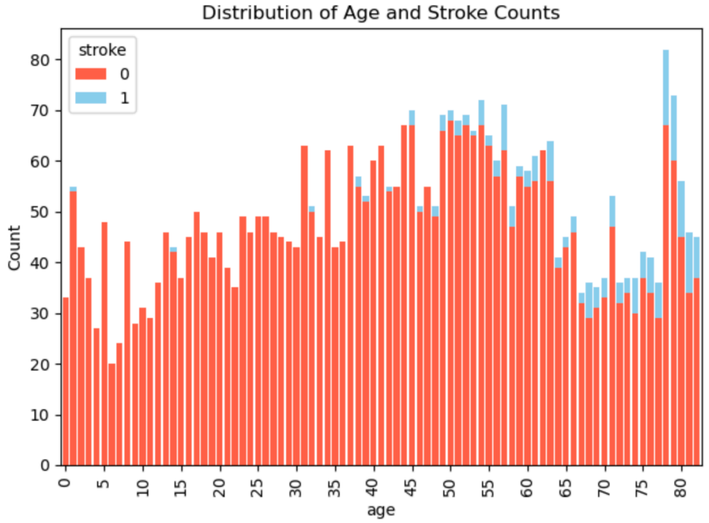
0 = No Stroke

1 = Stroke

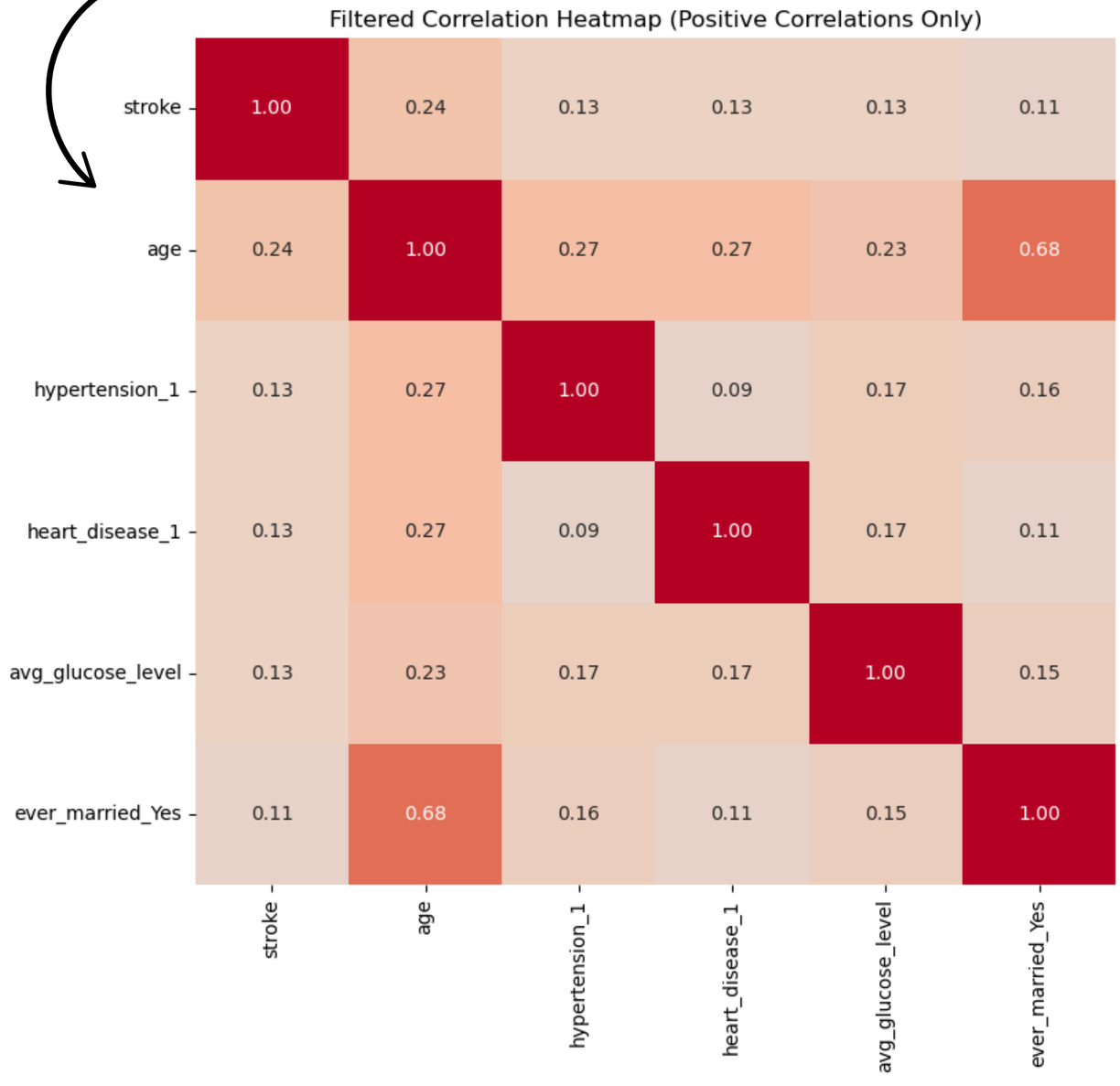
We want to predict the occurence of 1

EDA Visualizations





Best correlations, from highest to lowest



Exploring Algorithms

01 Problem type is a yes/no (stroke or no stroke). Falls into **Classification.**

02 Compared **Logistic Regression** and **Decision Trees** however…

03 **Data highly imbalanced**, not enough stroke data to train on.
F1 scores: 0.00 (LR), 0.08 (DT)

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
#Instantiate the model
logreg_model = LogisticRegression(solver='liblinear')
#Fit the model
logreg_model.fit(X, y)
#Prediction
logreg_model_pred = logreg_model.predict(X)
```

✓ 0.2s

```
# Generate and print the classification report
from sklearn.metrics import classification_report
print(classification_report(y, logreg_model_pred))

#f1-score 0?!
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.95	1.00	0.98	3889
1	0.00	0.00	0.00	199
accuracy			0.95	4088
macro avg	0.48	0.50	0.49	4088
weighted avg	0.91	0.95	0.93	4088

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
#Instantiate the model
dt_model = DecisionTreeClassifier(max_depth=5)
#Fit the model
dt_model.fit(X, y)
#Prediction
dt_model_pred = dt_model.predict(X)
```

✓ 0.1s

```
print(classification_report(y, dt_model_pred))
#f1-score 0.08
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.95	1.00	0.98	3889
1	1.00	0.04	0.08	199
accuracy			0.95	4088
macro avg	0.98	0.52	0.53	4088
weighted avg	0.96	0.95	0.93	4088

With more advanced DS techniques, this can be solved by creating synthetic data using SMOTE or ADASYN

Synthetic Minority Oversampling Technique. It's a technique used in machine learning to address imbalanced datasets.

Adaptive Synthetic algorithm, similar to SMOTE but generates different number of samples depending on an estimate of the local distribution

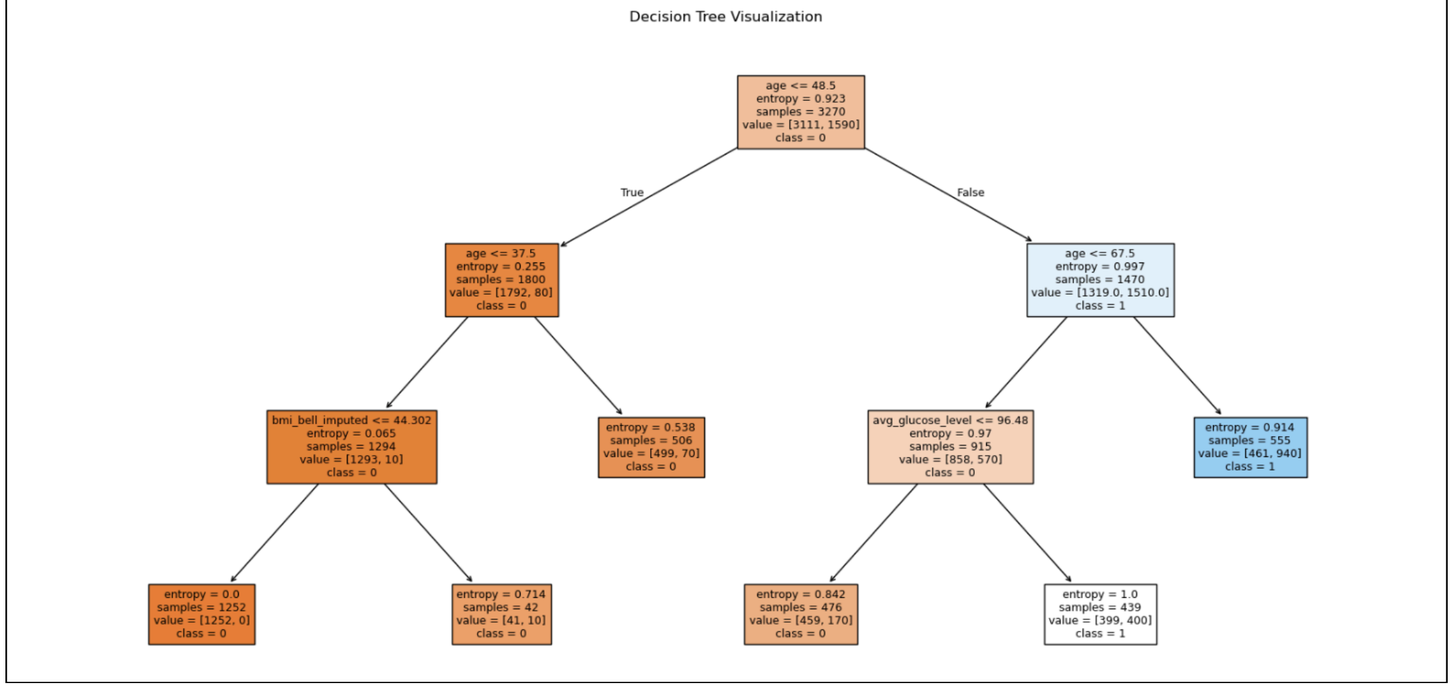
Final Model

- Used **Optuna** to assign weight to the 1 (stroke) class.
- Weight for ‘No Stroke’ around 1 and for ‘Stroke’ around 7 or 8.
- Fed new weightings into Decision Tree

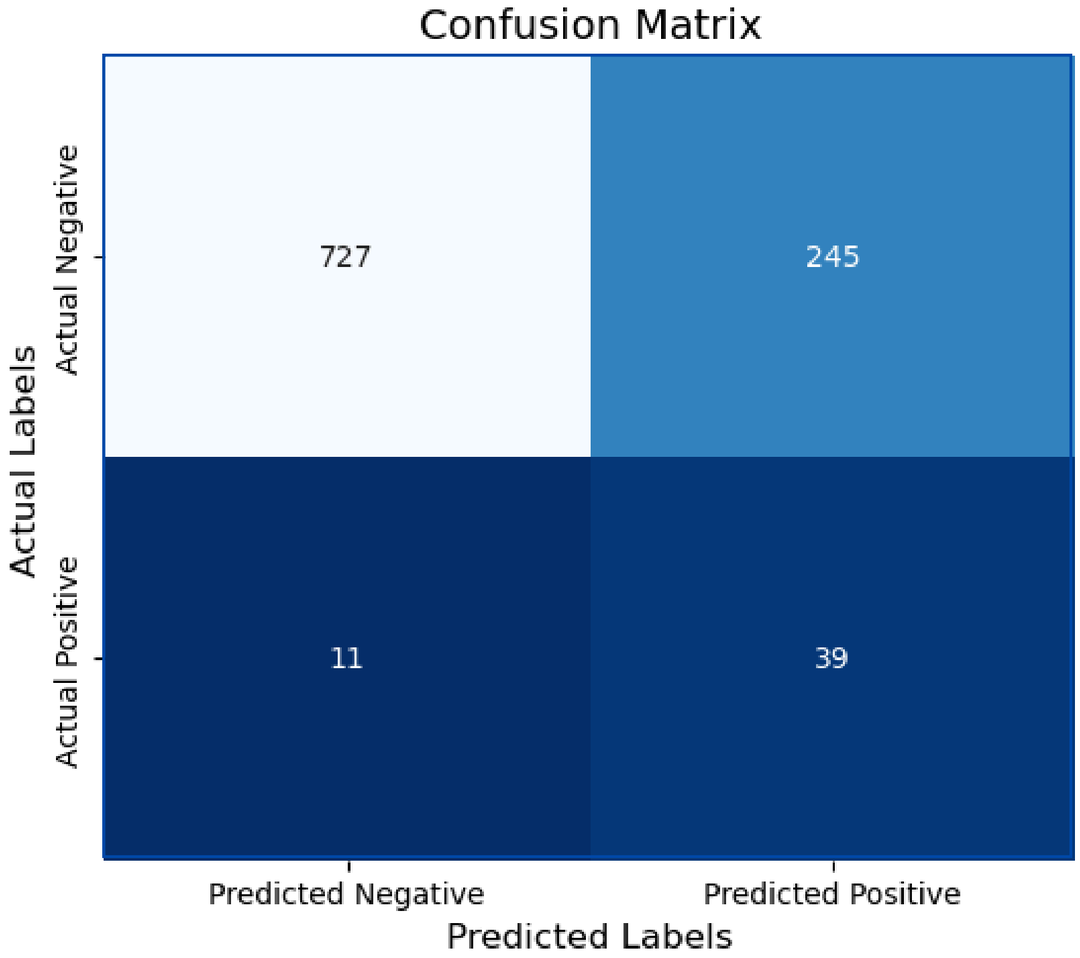
```python
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# Visualize the Decision Tree
plt.figure(figsize=(20, 10))
plot_tree(
    clf,                                # The trained decision tree classifier
    feature_names=X_train.columns,   # Feature names (columns of your dataset)
    class_names=clf.classes_.astype(str),  # Class names (if target variable is encoded)
    filled=True,                       # Fill nodes with colors based on class
    rounded=False,                      # Round the corners of nodes
    fontsize=9                     # Font size for readability
)
plt.title("Decision Tree Visualization", fontsize=12)
plt.show()
```

✓ 1.0s Python



Final Model - scores



	precision	recall	f1-score	support
0	0.99	0.75	0.85	972
1	0.14	0.78	0.23	50
accuracy			0.75	1022
macro avg	0.56	0.76	0.54	1022
weighted avg	0.94	0.75	0.82	1022

```
f1 = f1_score(y_test, y_pred_dt)

print("F1 Score:", f1)
```

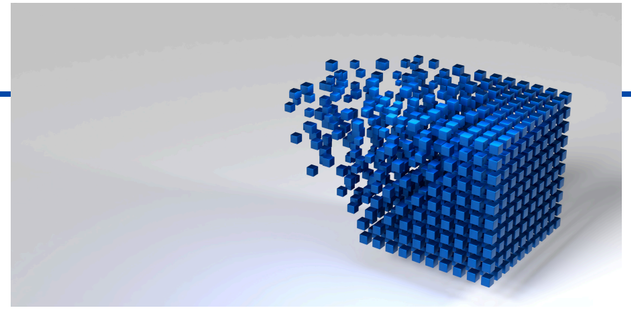
```
F1 Score: 0.23353293413173654
```

Better F1 score, but still quite low

Next Steps…

For this particular dataset:

- Data collection to improve balance (focus on Stroke (1) variable).
- Expand metrics (genetic markers) for more accuracy, more feature engineering.
- Apply advanced techniques such as SMOTE or ADASYN



For myself:

- Deepen understanding of statistics
- Continue to develop skills in Python
- Build portfolio and expand data skills, with focus in analysis.
- Become more active in data communities

Questions?

Thank you to Jasyot & Paras!