Stroke Prediction Analysis

ROLLAND OSAGIE, OLGA POSTOLACHI, WILLIAM RICHARDSON, AND BRYAN WERTH; DECEMBER 2021

Predicting Strokes with Machine Learning

Purpose and Questions we hope to answer:

- Every year more than 795,000 people in the US suffers a stroke, contributing to 1 out of every 6 deaths due to cardiovascular disease in 2018.
- How likely is someone to suffer a stroke based on our variables: gender, age, hypertenstion, heart disease history, BMI(Body Mass Index), smoking status, avg glucose level, if they were ever married, what type of work they do, and where they reside- rural or urban?
- Which variable contributes the most to suffering a stroke?
- Which age bracket shows when strokes start affecting people based on the data?

Stroke information source: https://www.cdc.gov/stroke/facts.htm

Data Storage

Data Storage Challenges

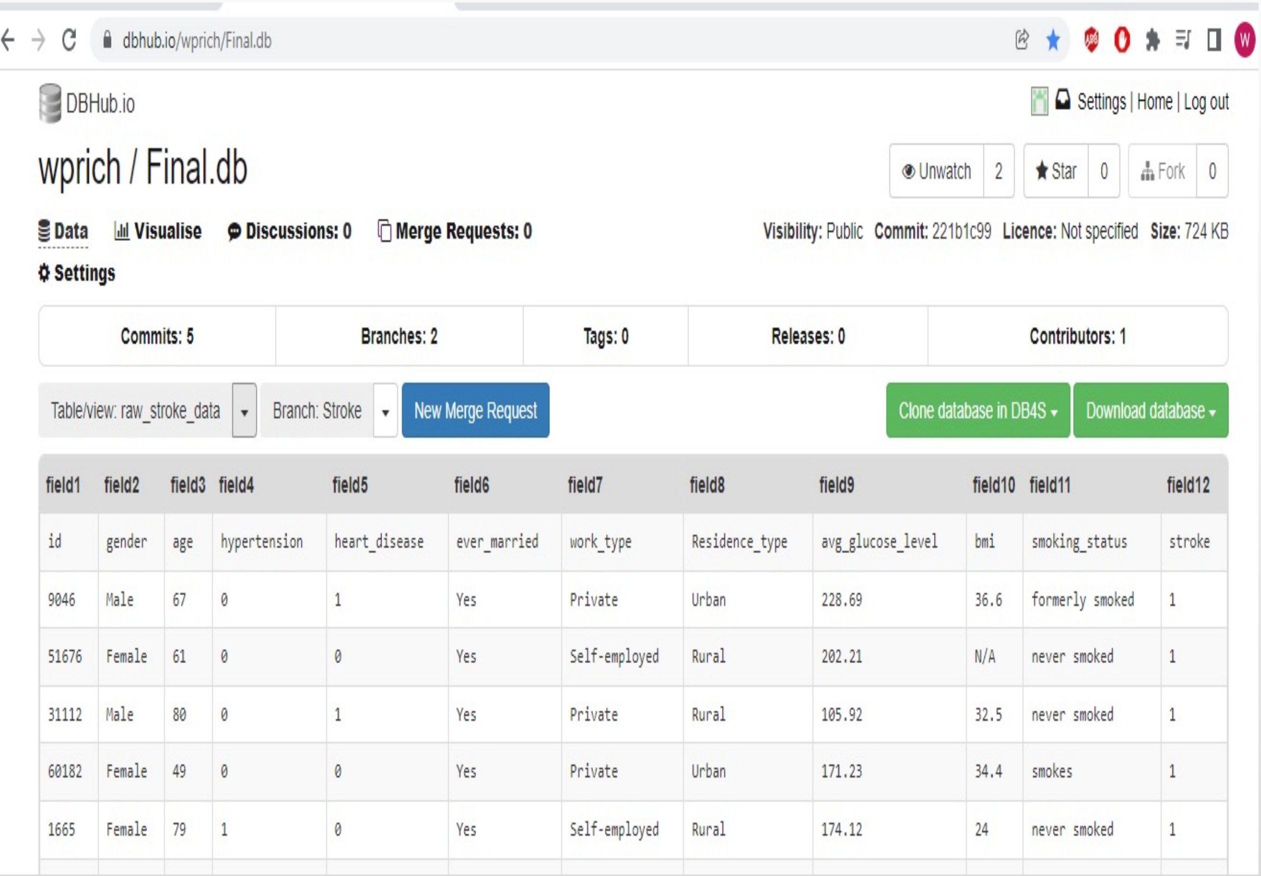
- ➢ Storage Location?
- ➢ How to share database with team members?

Solution

- ➢ **SQLite** – is a high quality, visual, open-source tool to create, design, and edit database files compatible with **SQLite**.
- ➢ **DBHub.io** – a simple API server, used for querying databases remotely was used to share data amongst team members.

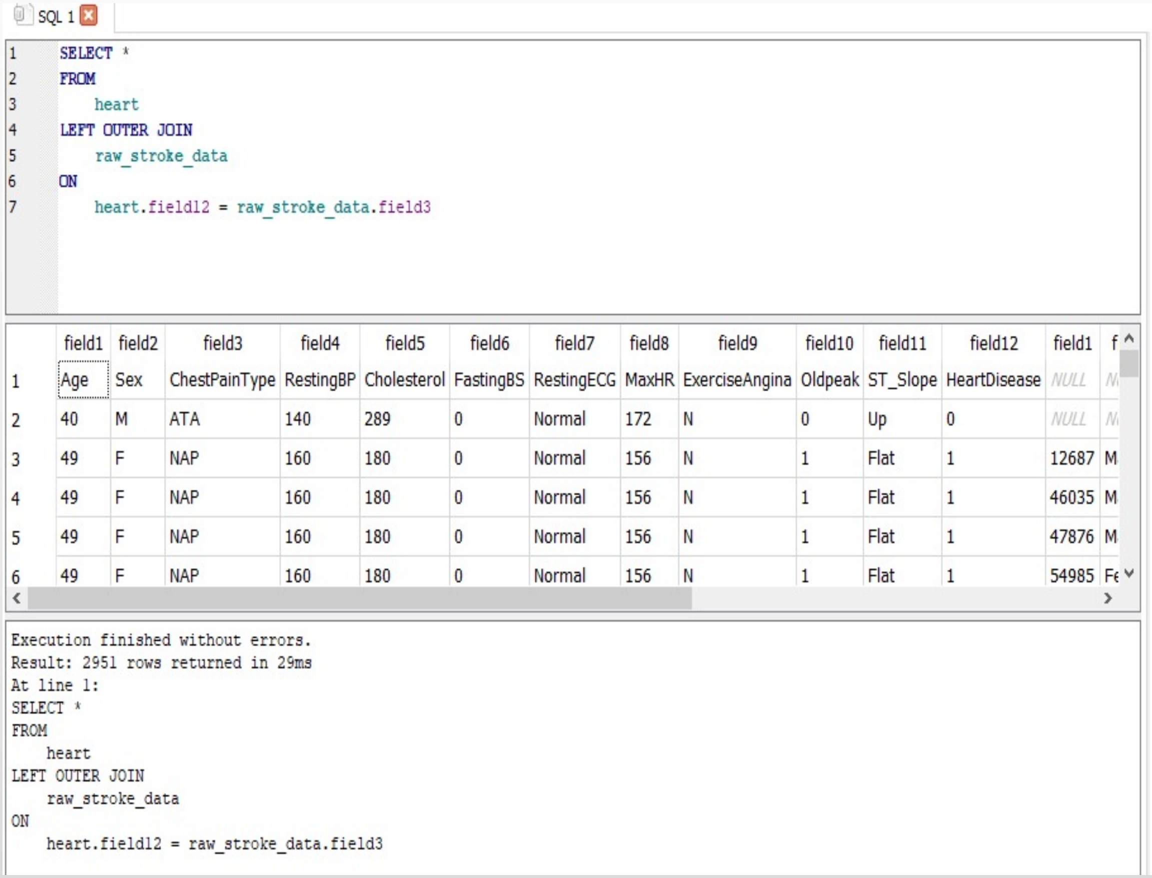
Setting Up Database and Potential Expansion

DBHub.io Website Database View



field1	field2	field3	field4	field5	field6	field7	field8	field9	field10	field11	field12
id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A	never smoked	1
31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1

Failed Join Dataset



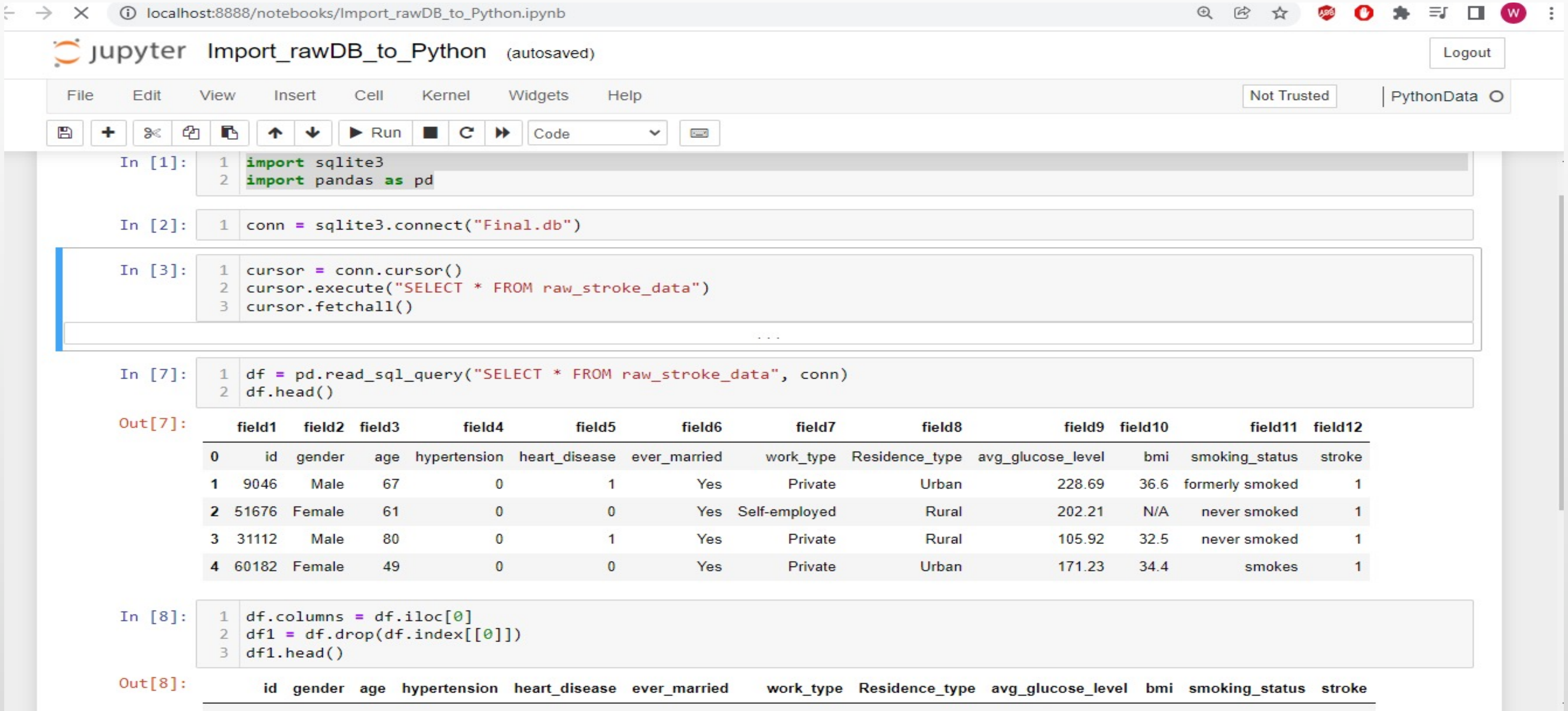

```
SELECT *
FROM
    heart
LEFT OUTER JOIN
    raw_stroke_data
ON
    heart.field12 = raw_stroke_data.field3
```

	field1	field2	field3	field4	field5	field6	field7	field8	field9	field10	field11	field12	field1	f
1	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease	NULL	N
2	40	M	ATA	140	289	0	Normal	172	N	0	Up	0	NULL	N
3	49	F	NAP	160	180	0	Normal	156	N	1	Flat	1	12687	M
4	49	F	NAP	160	180	0	Normal	156	N	1	Flat	1	46035	M
5	49	F	NAP	160	180	0	Normal	156	N	1	Flat	1	47876	M
6	49	F	NAP	160	180	0	Normal	156	N	1	Flat	1	54985	Fe

```
Execution finished without errors.
Result: 2951 rows returned in 29ms
At line 1:
SELECT *
FROM
    heart
LEFT OUTER JOIN
    raw_stroke_data
ON
    heart.field12 = raw_stroke_data.field3
```

Exporting Database to Jupyter Notebook

- Also utilizing the wonderful world of google search results, I found a way to connect our program running in python, to the database file.



localhost:8888/notebooks/Import_rawDB_to_Python.ipynb

jupyter Import_rawDB_to_Python (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted PythonData

Run Code

In [1]:
```python
import sqlite3
import pandas as pd
```

In [2]:
```python
conn = sqlite3.connect("Final.db")
```

In [3]:
```python
cursor = conn.cursor()
cursor.execute("SELECT * FROM raw_stroke_data")
cursor.fetchall()
```

In [7]:
```python
df = pd.read_sql_query("SELECT * FROM raw_stroke_data", conn)
df.head()
```

Out[7]:

	field1	field2	field3	field4	field5	field6	field7	field8	field9	field10	field11	field12
0	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
1	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
2	51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A	never smoked	1
3	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
4	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1

In [8]:
```python
df.columns = df.iloc[0]
df1 = df.drop(df.index[[0]])
df1.head()
```

Out[8]:

id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke

Preliminary Data Exploration

- After getting our data from Kaggle.com, we performed some initial clean up and data exploration techniques to describe its characteristics including size, quantity, and accuracy of the data.
- Using Python and Jupyter Notebook, we Extracted, Transformed, and Load our data into a Dataframe that we could analyze and manipulate.
- We went from 5110 rows to 5109 dropping one due to gender being “unknown”; we also replaced NaN values in our BMI column with the median value to avoid dropping a large number of our data points.

Data Source: https://www.kaggle.com/fedesoriano/stroke-prediction-dataset

```
# Check unique values
df2['gender'].unique()
```

```
array(['Male', 'Female', 'Other'], dtype=object)
```

```
# Count how many 'Other' values are in gender column
df2['gender'].value_counts()
```

```
Female    2994
Male      2115
Other        1
Name: gender, dtype: int64
```

```
# Drop the row that contains 'Other' value
other_value = df2[df2['gender'] =='Other'].index
df3 = df2.drop(other_value)
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5109 entries, 1 to 5110
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   gender             5109 non-null   object
 1   age                5109 non-null   float64
 2   hypertension       5109 non-null   int32
 3   heart_disease      5109 non-null   int32
 4   ever_married       5109 non-null   object
 5   work_type          5109 non-null   object
 6   Residence_type     5109 non-null   object
 7   avg_glucose_level  5109 non-null   float64
 8   bmi                5109 non-null   float64
 9   smoking_status     5109 non-null   object
 10  stroke             5109 non-null   int32
dtypes: float64(3), int32(3), object(5)
memory usage: 419.1+ KB
```

Since we dropped 1 row, now we have 5109 rows.

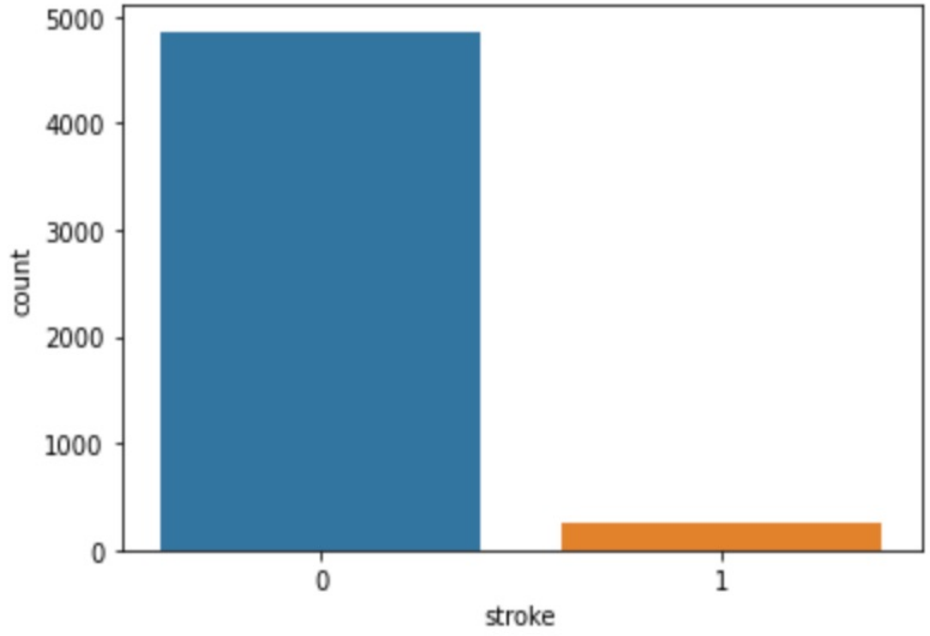
Data Analysis using Visualizations

Tableau Visualizations

Distribution of predicted values:

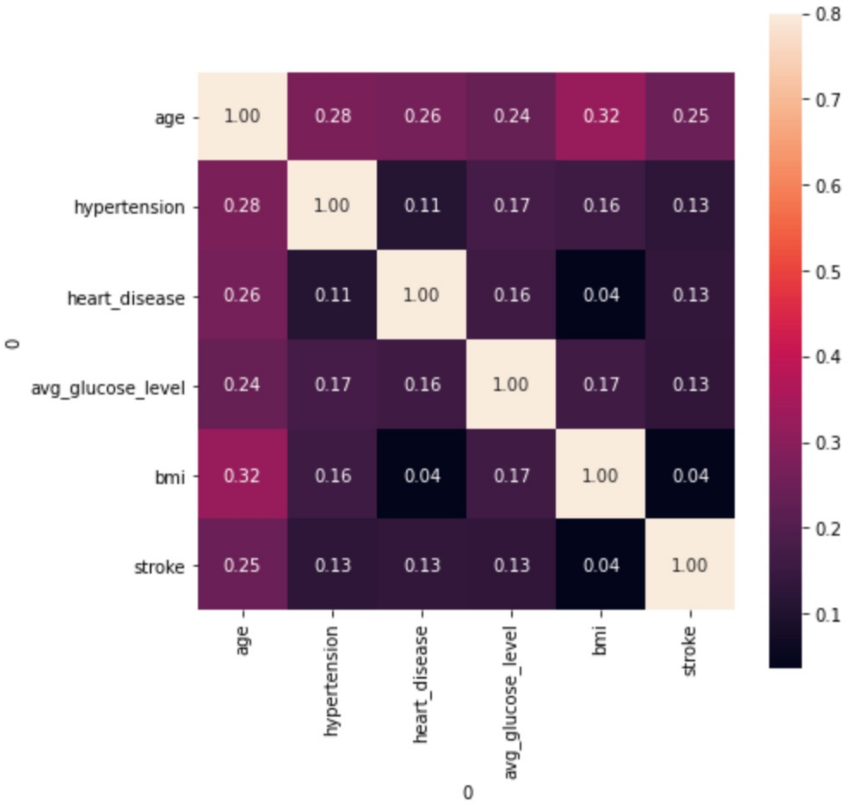
```
# Visualize distribution of predicted values
sns.countplot(x='stroke', data=df3)
df3.stroke.value_counts()
```

```
0    4860
1     249
Name: stroke, dtype: int64
```

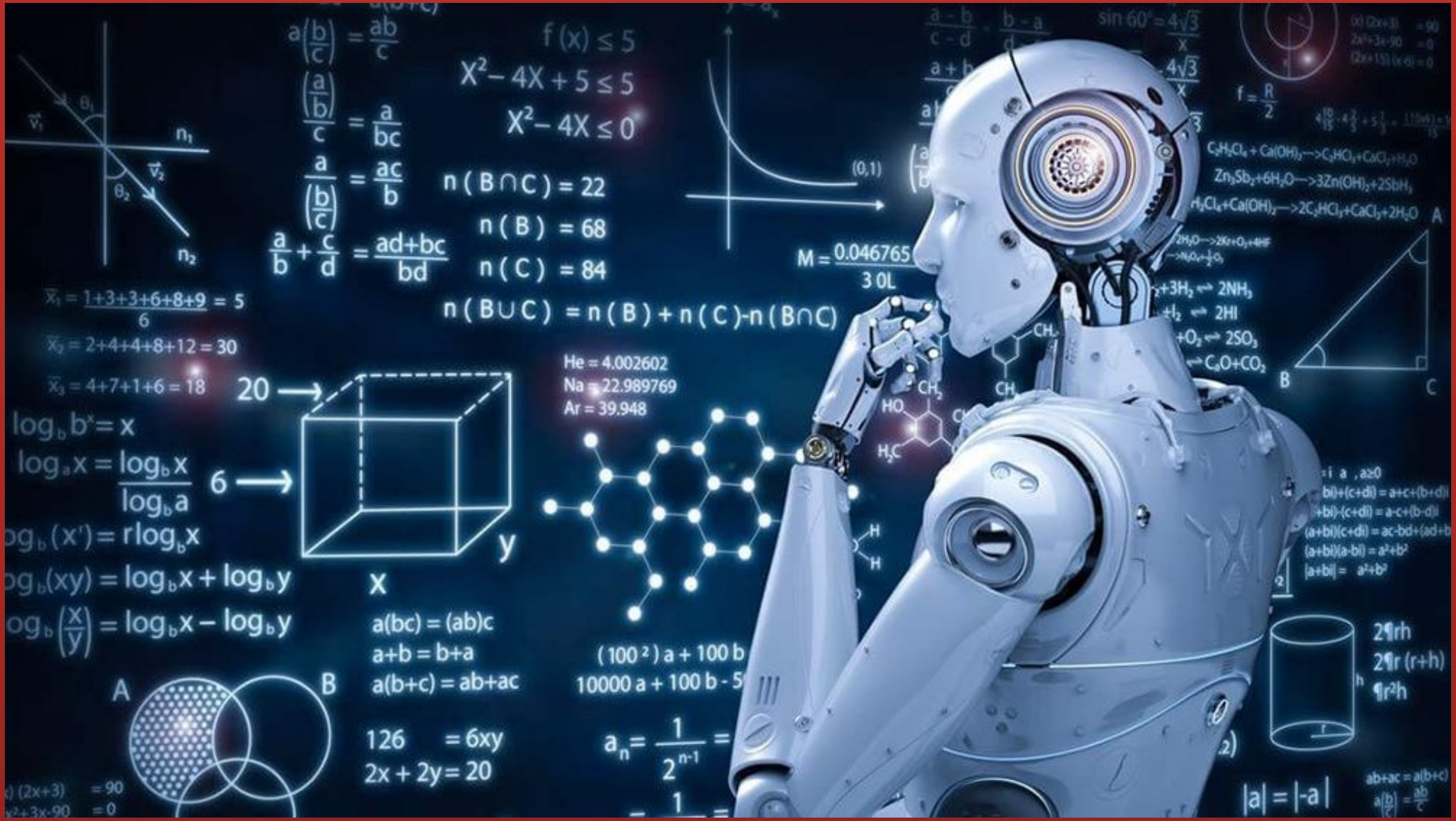


Correlation between our variables and stroke:

```
# Check what correlation can be found between the stroke and variables in the dataset
correlation = df3.corr()
fig, axes = plt.subplots(figsize=(7, 7))
sns.heatmap(correlation, vmax=.8, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10});
```



Machine learning



Data Preprocessing/ Cleaning

- Importing all necessary libraries
- Drop ID columns
- Handling missing data
- Convert categorical variables to numerical (object to int or float)
- Label encoding (get_dummies)

Preparing data for the model

- Defining features and target sets (X, y)
- Splitting data into Train (80%) and Test (20%) sets (X_train, X_test, y_train, y_test)
- **Scaling the data (StandardScaler())**

```
# Check the numbers of positive and negative predicted stroke in training set

from collections import Counter
Counter(y_train)
```

```
Counter({0: 3900, 1: 187})
```

- Oversample X and y training sets (SMOTE)

```
# Oversample X and y training sets
X_resampled, y_resampled = SMOTE(random_state=1).fit_resample(X_train, y_train)

Counter(y_resampled)
```

```
Counter({0: 3900, 1: 3900})
```

Create/ Fit/ Predict

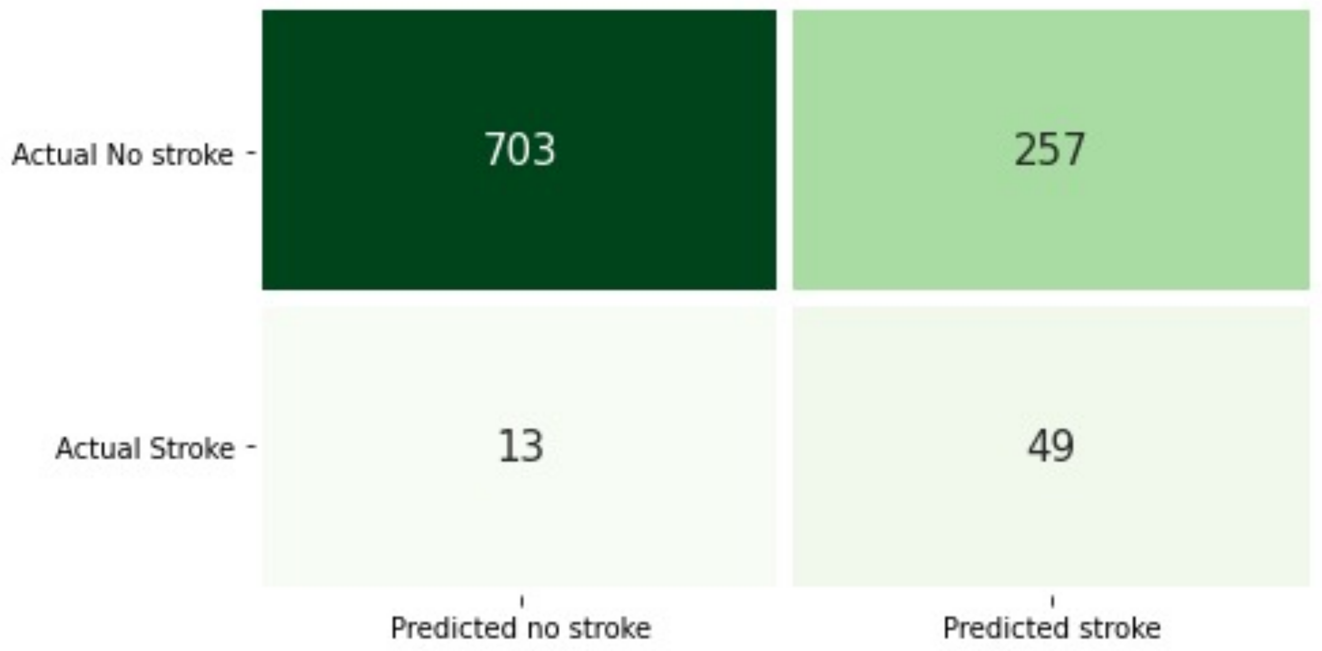
1. Support Vector Machine

```
# Create SVM model
SVM_model = SVC(kernel='linear')

# Fit the model using resampled data
SVM_model.fit(X_resampled, y_resampled)

# Create predictions
y_pred = SVM_model.predict(X_test)

# Calculated the balanced accuracy score
acc_score = balanced_accuracy_score(y_test, y_pred)
acc_score
```

```
0.761307123655913
```

Confusion Matrix

```
# Display the confusion matrix
plt.figure(figsize = (7, 4))
sns.heatmap(cm, cmap = 'Greens', annot = True, fmt = 'd', linewidths = 5
            yticklabels = ['Actual No stroke', 'Actual Stroke'], xtickla
plt.yticks(rotation = 0)
plt.show()
```



Classification report

	precision	recall	f1-score	support
0	0.98	0.73	0.84	960
1	0.16	0.79	0.27	62
accuracy			0.74	1022
macro avg	0.57	0.76	0.55	1022
weighted avg	0.93	0.74	0.80	1022

```
Accuracy Score: 0.735812133072407
--
F1 Score: 0.2663043478260869
--
Train score: 0.9542454529664988
--
Precision score: 0.16013071895424835
--
Recall Score: 0.7903225806451613
```

Comparison of Machine Learning models

	Models	Accuracy score	Recall
0	Random Forest	91%	13%
1	K Near Neighbour	81%	37%
2	Suport Vector machine	74%	79%
3	Logistic Regression	74%	79%
4	Naive Bias	30%	100%

The best Accuracy score has the Random forest classifier, but it also has the lowest Recall. On the other hand, Naive Bias has a 100% Recall, but the lowest Accuracy score. In conclusion we can say that the best performance showed SVM and Logistic regression with an Accuracy score of 74% and a Recall of 79%.

Recommendations for Future Analysis

- We could have more access to data from hospitals that represented the entire population of the U.S.
- Have access to more cloud database options that are easily accessed and more affordable.
- A geographical map of stroke data across the Country.
- Use the Neural Network model for optional results.
- Questions?