

Stroke Prediction

Ayala Bouhnik-Gelbord
Maayan Sulimani

GitHub link: <https://github.com/AyalaBouhnik/Stroke-Prediction>

- This project has been done as part of 'Machine Learning' course, led by Prof. Lee-Ad Gottlieb

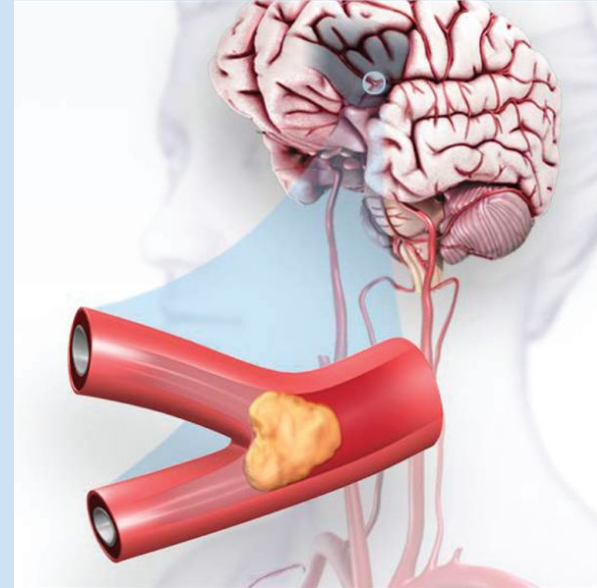


What is a Stroke?

Stroke is a **disease that affects the arteries leading to and within the brain**. It is the number 5 cause of death and a leading cause of disability in the United States. A stroke occurs when a blood vessel that carries oxygen and nutrients to the brain is either blocked by a clot or bursts (or ruptures).

What are the effects of stroke?

The brain is an extremely complex organ that controls various body functions. If a stroke occurs and blood flow can't reach the region that controls a particular body function, that part of the body won't work as it should.





Project Target-

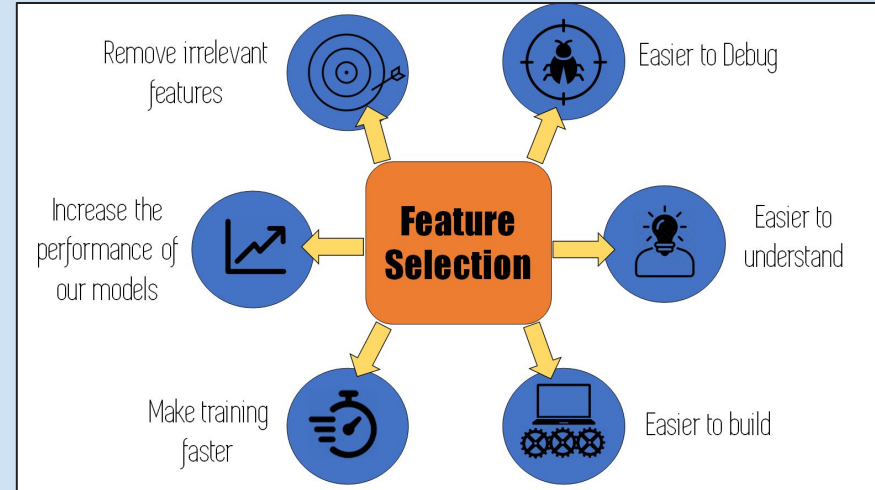
In order to try to reduce reduce stroke deaths, we will try to predict which people are more likely to have a stroke.

For this project we used Kaggle dataset, which includes 11 clinical features for predicting stroke events.











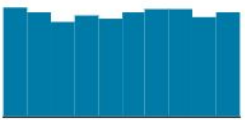














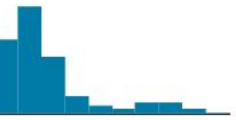


The Features-

1. Gender
2. Age
3. Hypertension binary feature
4. Heart disease binary feature
5. Has the patient ever been married?
6. Work type of the patient
7. Residence type of the patient
8. Average glucose level in blood
9. Body Mass Index
10. Smoking status of the patient
11. Stroke event



A Little Bit About Our Data..

 id  Unique id	 gender  Gender	 age  Age	 hypertension  Hypertension binary feature	 heart_disease  Heart disease binary feature
 67 72.9k	Female 59% Male 41% Other (1) 0%	 0.08 82	 0 1	 0 1
 ever_married  Has the patient ever been married?	 work_type  Work type of the patient	 Residence_type  Residence type of the patient	 avg_glucose_level  Average glucose level in blood	 bmi  Body Mass Index
 true 3353 66% false 1757 34%	Private 57% Self-employed 16% Other (1366) 27%	Urban 51% Rural 49%	 55.1 272	N/A 4% 28.7 1% Other (4868) 95%



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("/content/healthcare-dataset-stroke-data.csv")
data.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1



Lets Explore Our Data-

```
<bound method DataFrame.info of
0    9046   Male   67.0 ...   36.6  formerly smoked    1
1    51676  Female  61.0 ...   NaN    never smoked    1
2    31112   Male   80.0 ...   32.5  never smoked    1
3    60182  Female  49.0 ...   34.4    smokes        1
4    1665   Female  79.0 ...   24.0  never smoked    1
...    ...    ...    ...    ...    ...    ...
5105  18234  Female  80.0 ...   NaN    never smoked    0
5106  44873  Female  81.0 ...   40.0  never smoked    0
5107  19723  Female  35.0 ...   30.6  never smoked    0
5108  37544   Male   51.0 ...   25.6  formerly smoked  0
5109  44679  Female  44.0 ...   26.2    Unknown      0

[5110 rows x 12 columns]>
```

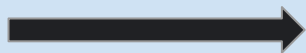
```
[34] data.isnull().sum()

id                0
gender            0
age              0
hypertension      0
heart_disease     0
ever_married      0
work_type         0
Residence_type    0
avg_glucose_level 0
bmi              201
smoking_status    0
stroke           0
dtype: int64
```

As we can see the 'bmi' column holds some missing values.

So we need to fill the null values:

```
data['bmi'].fillna(data['bmi'].mean(), inplace = True)
```



```
data.isnull().sum()

gender          0
age             0
hypertension    0
heart_disease   0
work_type       0
Residence_type  0
avg_glucose_level 0
bmi             0
smoking_status  0
stroke         0
dtype: int64
```



Removing an Unnecessary Column-

After we explored the data, we decided to remove one column- `id`, because the Id of the person is not a relevant information for our model.

So now we have 11 attributes.

✓
0s



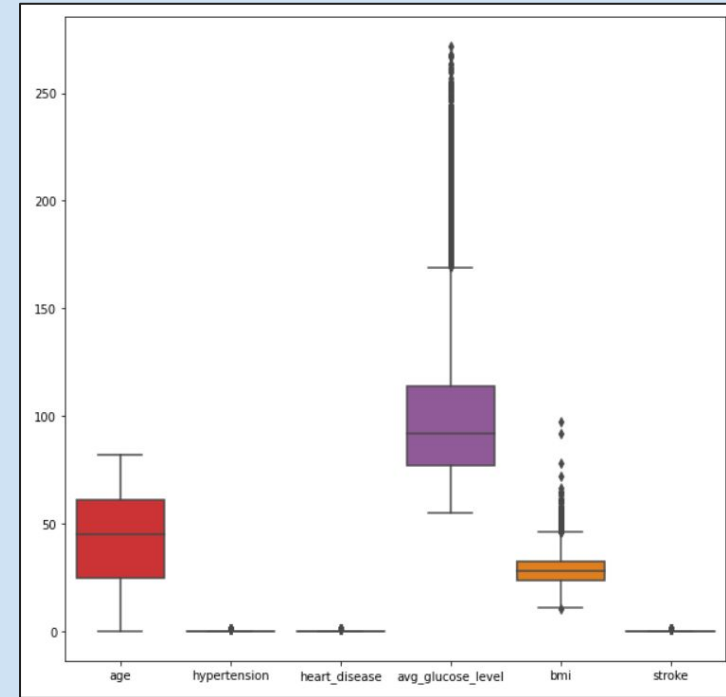
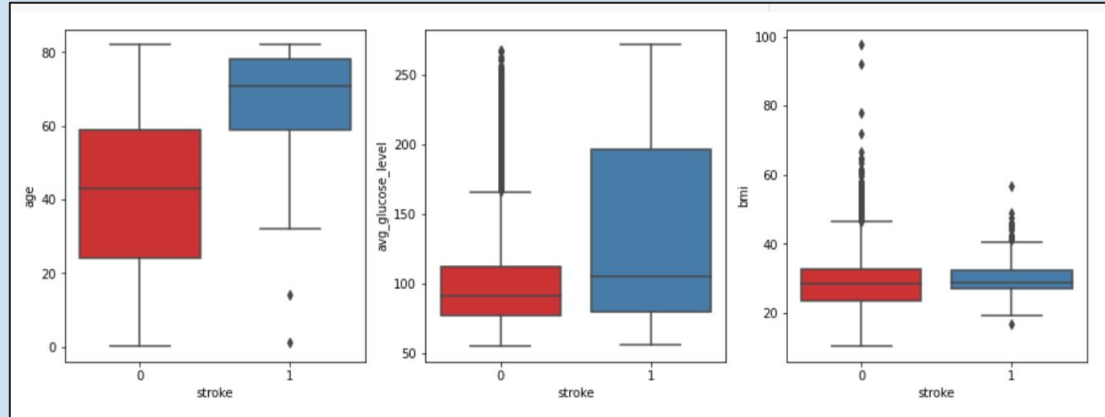
```
data.shape
```



```
(5110, 11)
```


Outliers-

We used subplot function to find the outliers in our data.





Algorithms-

We want to know, based on the 11 features, what are the chances to of having a stroke.

We will use the following techniques:

- Logistic Regression
- SVM
- Decision Tree
- K- nearest neighbour



Train and Test-

We split our data into train and test,

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
```

After that, we used `fit_transform()` on the `x_train` variable and `transform` on the `x_test` variable.

Logistic Regression-

✓
0s

[382] #Logistic regression-

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

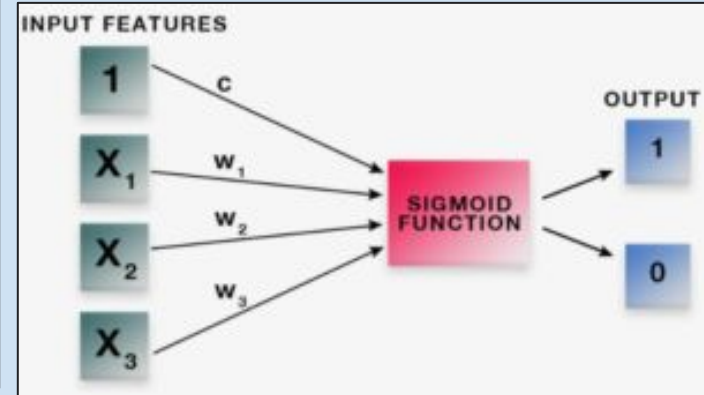
logistic_reg = LogisticRegression()

logistic_reg.fit(x_train_std, y_train)

y_pred_logistic_reg = logistic_reg.predict(x_test_std)

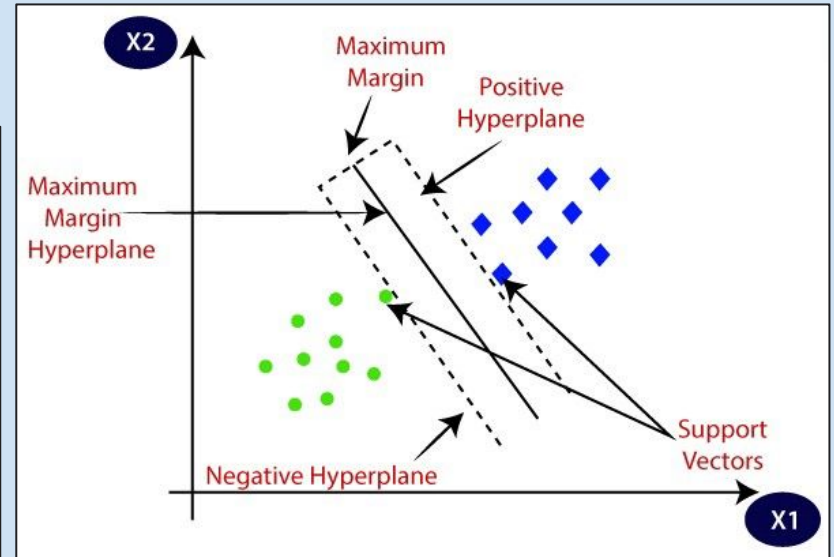
accuracy_logistic_reg = accuracy_score(y_test, y_pred_logistic_reg)
print("Logistic Regression accuracy = " + str(accuracy_logistic_reg*100) + "%")

Logistic Regression accuracy = 94.71624266144813%
```



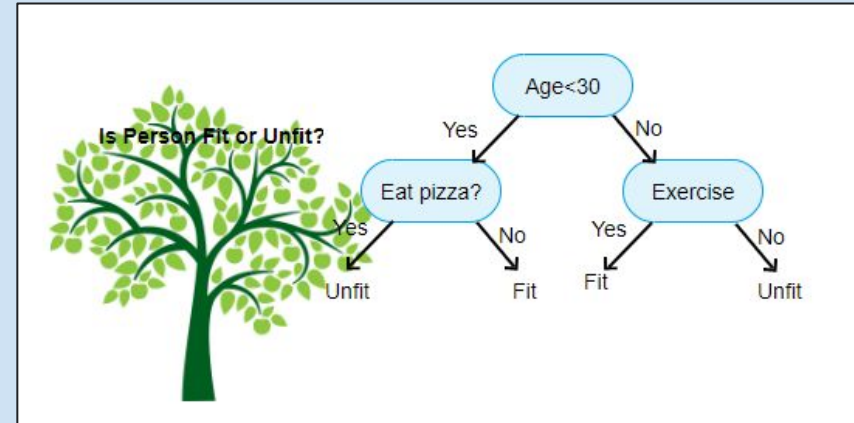
SVM-

```
#SVM-  
  
from sklearn.svm import SVC  
svm = SVC()  
  
svm.fit(x_train_std, y_train)  
  
y_pred_svm = svm.predict(x_test_std)  
  
ac_svm = accuracy_score(y_test, y_pred_svm)  
print("SVM accuracy = " + str(ac_svm*100) + "%")  
  
SVM accuracy = 94.71624266144813%
```



Decision Tree-

```
[390] #Decision tree-  
from sklearn.tree import DecisionTreeClassifier  
decision_tree = DecisionTreeClassifier()  
  
decision_tree.fit(x_train_std, y_train)  
  
decision_tree.feature_importances_  
  
array([0.04505728, 0.20077799, 0.03420359, 0.02100343, 0.00581771,  
       0.03533076, 0.02881657, 0.29332933, 0.26733965, 0.06832369])  
  
[386] x_train.columns  
  
Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',  
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',  
       'smoking_status'],  
      dtype='object')  
  
[387] from sklearn.metrics import accuracy_score  
  
y_pred_decision_tree = decision_tree.predict(x_test_std)  
  
accuracy_decision_tree = accuracy_score(y_test, y_pred_decision_tree)  
print("decision tree accuracy = " + str(accuracy_decision_tree*100) + "%")  
  
decision tree accuracy = 91.6829745596869%
```



K- Nearest Neighbour-

```
#K-Nearest-Neighbour

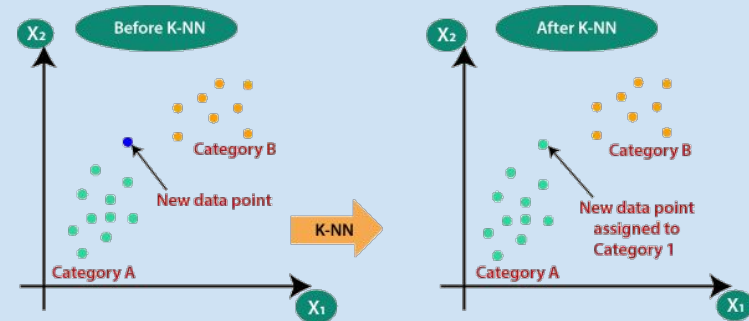
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()

knn.fit(x_train_std, y_train)

y_pred_knn = knn.predict(x_test_std)

accuracy_knn = accuracy_score(y_test, y_pred_knn)
print("K-nearest neighbour accuracy = " + str(accuracy_knn*100) + "%")
```

K-nearest neighbour accuracy = 94.71624266144813%





Results-

Algorithms-	Logistic regression	SVM	Decision Tree	K- Nearest Neighbour
Accuracy-	94.71624266144813%	94.71624266144813%	91.6829745596869%	94.71624266144813%



Outliers-

Could we run the algorithms without taking into consideration the outliers?

Yes.

In our data, the results would be quite close to the results with the outliers. It changes only the decision tree results (with the outliers the accuracy is slightly better, but there is not a significant difference).

The outliers were not as significant as we thought they would be.



Without Outliers Results-

Algorithms-	Logistic regression	SVM	Decision Tree	K- Nearest Neighbour
Accuracy-	94.71624266144813%	94.71624266144813%	91.58512720156556%	94.71624266144813%

Could We Use Other Algorithms?

Of course.

We could use other algorithms and make the necessary adjustments, and then choose the algorithm with the best accuracy to predict the chances of having a stroke.

Sources and reference materials-

- <https://www.kaggle.com/fedesoriano/stroke-prediction-dataset>
- <https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02>
- <https://github.com/riddhi-jain>
- <https://machinelearningmastery.com/how-to-use-statistics-to-identify-outliers-in-data>