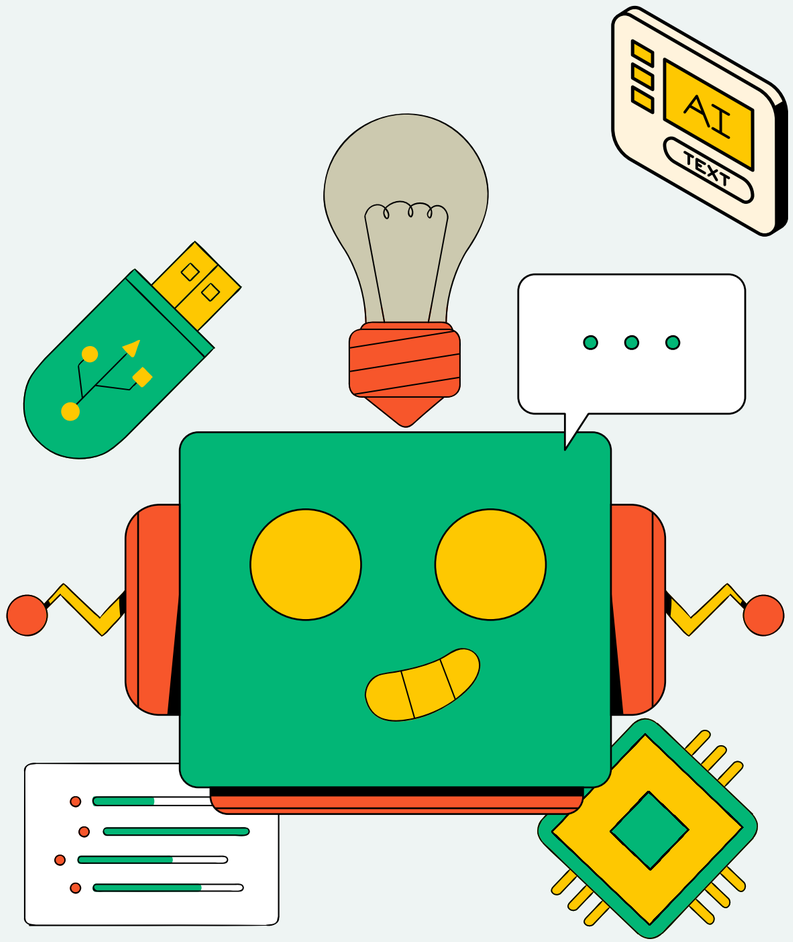
THYNK UNLIMITED

WE LEARN FOR THE FUTURE



BRAIN STROKE ANALYSIS & PREDICTION

PRESENTED BY:

MUDALIAR SAURABH (RA2111027010016)

K AKASH(RA2111027010015)

RITESH(RA2111027010014)

ARUNIMA(RA2111027010017)

PRESENTATION OUTLINE

- Introduction
- Importing Libraries
- About Dataset
- Basic Exploration
- Dataset Summary
- Data Preprocessing
- Custom Palette For Visualization

INTRODUCTION

- Stroke is a destructive illness that typically influences individuals over the age of 65 years age.
- Prediction of stroke is time-consuming and tedious for doctors.

- Therefore, the project mainly aims at predicting the chances of the occurrence of stroke using emerging Machine Learning techniques.

- Five different algorithms are used and a comparison is made for better accuracy.

- Stroke is a disease that affects the arteries leading to and within the brain. A stroke occurs when a blood vessel that carries oxygen and nutrients to the brain is either blocked by a clot or ruptures. According to the WHO, stroke is the 2nd leading cause of death worldwide.
- Early detection of stroke is a crucial step for efficient treatment to be able to do that, Machine Learning (ML) is an ultimate technology which can help health professionals make clinical decisions and predictions.

- Brain stroke is mainly classified into two categories:

1. Hemorrhage Stroke.
2. Ischemic Stroke
3. Transient ischemic attack (a warning or “mini-stroke”).

IMPORTING LIBRARIES

In [2]:
```
#importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

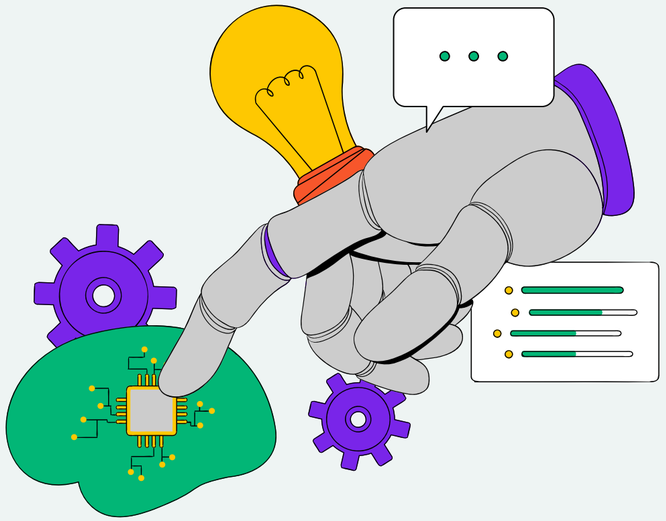
ABOUT DATASET

- **id**: Unique identifier
- **gender:** Gender of the patient (Male, Female, Other)
- **age:** Age of the patient
- **hypertension:** 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
- **heart_disease**: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
- **ever_married**: Yes if the patient is married, No if the patient is not married
- **work_type**: Profession of the patient (children, Govt_job, Never_worked, Private, Self-employed)
- **Residence_type**: Residence category of the patient (Rural, Urban)

ABOUT DATASET

- **avg_glucose_level:** Average glucose level in blood of the patient
- **bmi**: Body Mass Index of the patient
- **smoking_status**: Smoking status of the patient (formerly smoked, never smoked, smokes, Unknown). Unknown in smoking_status means that the information is unavailable for this patient
- **stroke**: 1 if the patient had a stroke or 0 if not

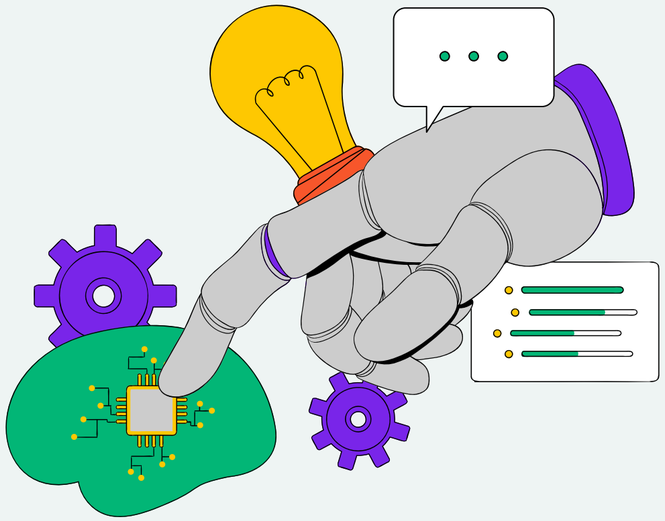
DATASET SUMMARY



Summary Of The Dataset :

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

BASIC EXPLORATION



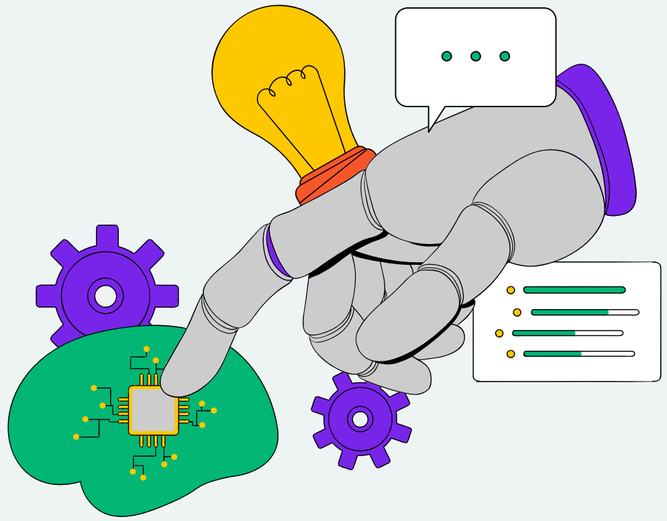
Let's have a glimpse of the dataset.

Shape Of The Dataset : (5110, 12)

	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

DATASET SUMMARY

```
In [8]: #total number of null values present in the dataset
        data.isnull().sum()

Out[8]: 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
```



Insights:

- There are missing values in bmi. We will drop the rows that have missing bmi values.
- There is no duplicate values in this dataset.

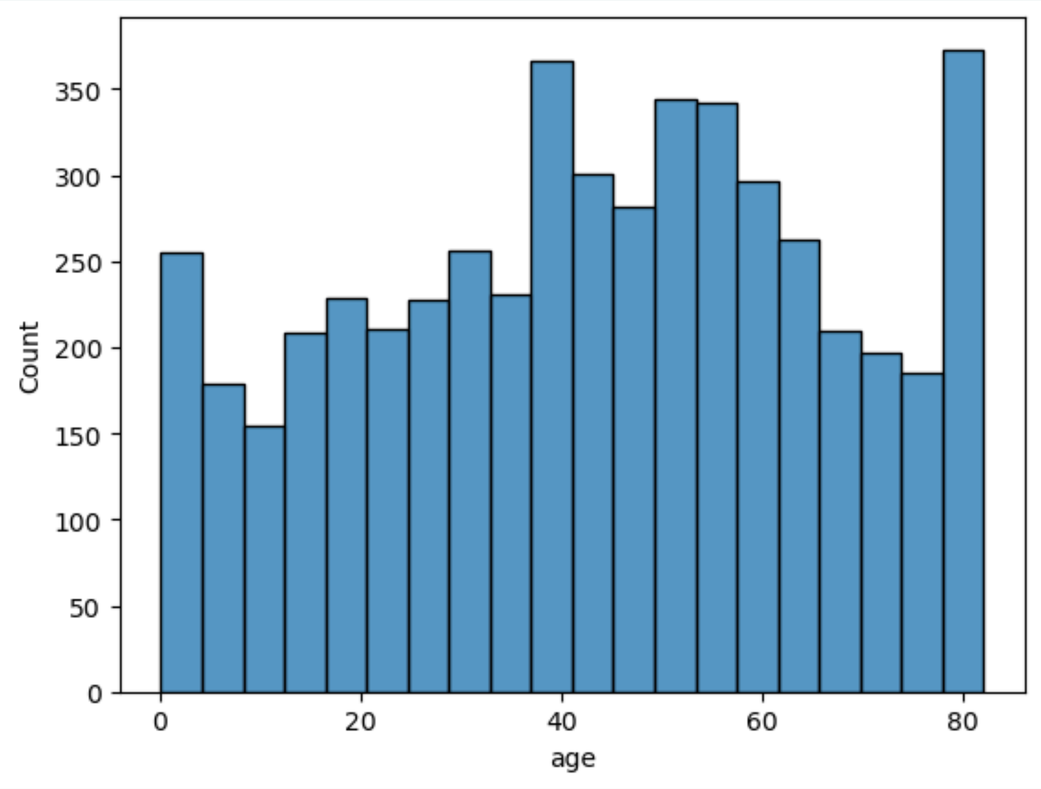
DATA PREPROCESSING

- After dropping null values, the shape of the Dataset is (4909, 12)
- After preprocessing, let's have a glimpse of the final dataset :

	Age	Gender	Marital Status	BMI	Occupation Type	Residence Type	Smoking Status	Hypertension	Heart Disease	Average Glucose Level	Stroke
0	67	Male	Married	36.600000	Private Job	Urban	Formerly Smoked	No	Yes	228.690000	Yes
2	80	Male	Married	32.500000	Private Job	Rural	Never Smoked	No	Yes	105.920000	Yes
3	49	Female	Married	34.400000	Private Job	Urban	Smokes	No	No	171.230000	Yes
4	79	Female	Married	24.000000	Self Employed	Rural	Never Smoked	Yes	No	174.120000	Yes
5	81	Male	Married	29.000000	Private Job	Urban	Formerly Smoked	No	No	186.210000	Yes

STROKE PATIENT'S AGE

Let's have a look on the distribution of stroke patient's age :

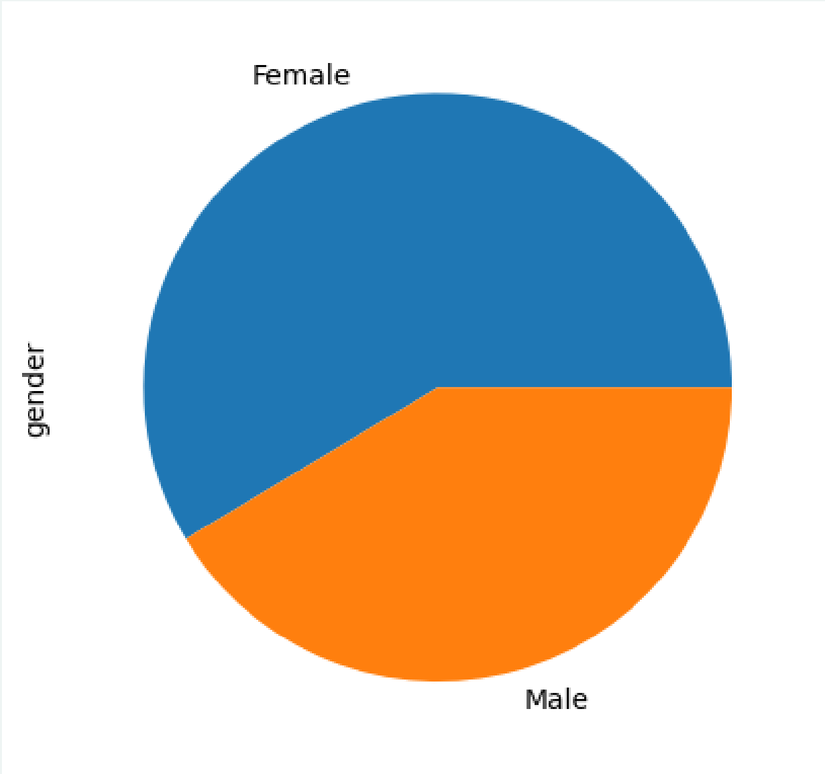


Let's have a look on the distribution of stroke patient's age :

- We can see the stroke patient's age distribution is left skewed. Most of the patients fall in between 40 years to 80 years.
- Also there are young and children female stroke patients too.

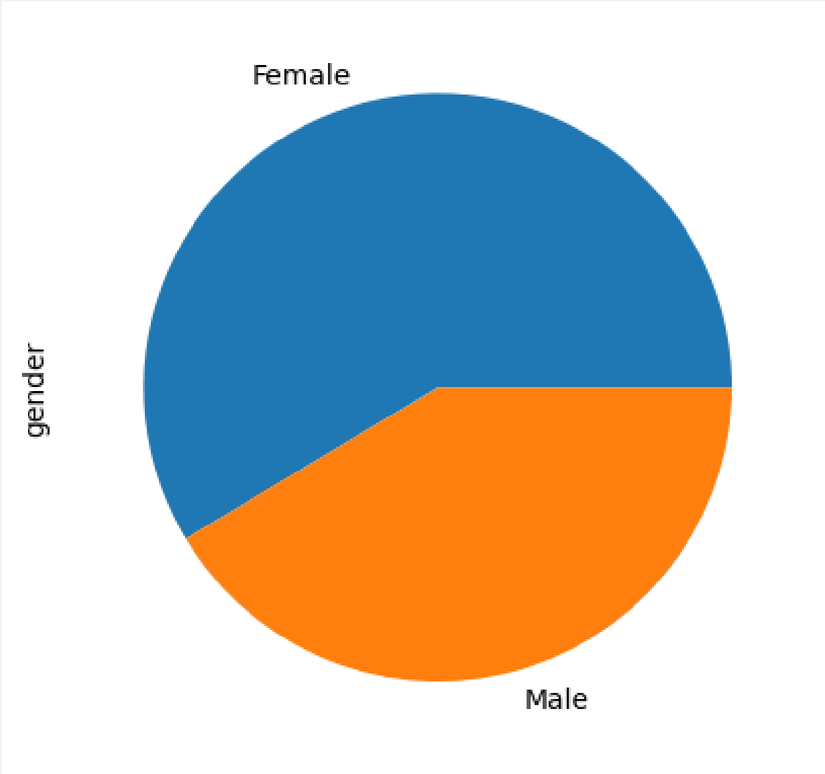
Let's have a look on the distribution of genderwise stroke patient's age :

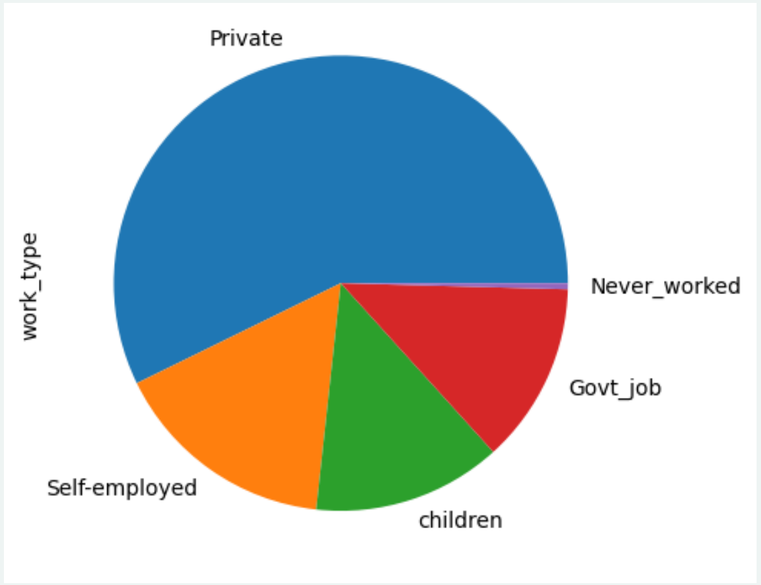
- We can see that there are more females than male



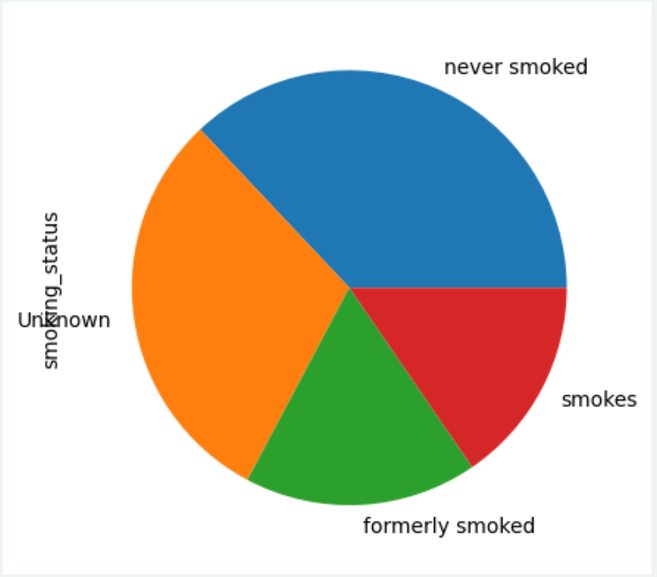
Let's have a look on the distribution of genderwise stroke patient's age :

- We can see that there are more females than male

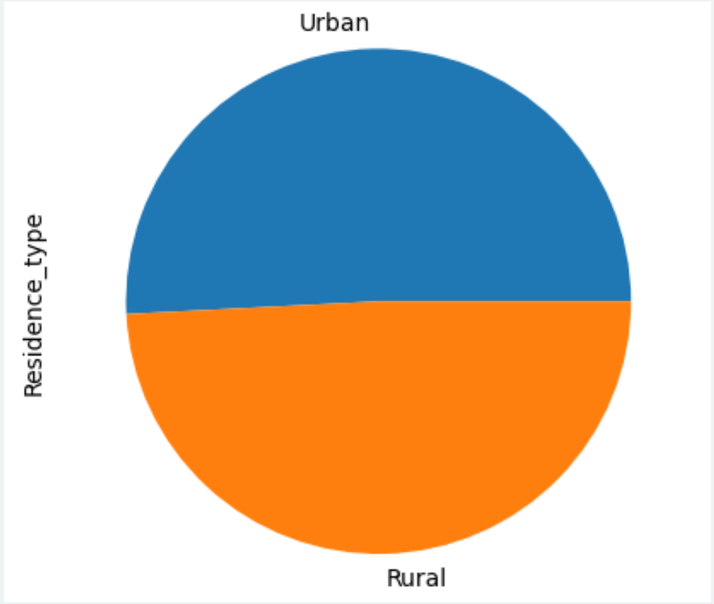




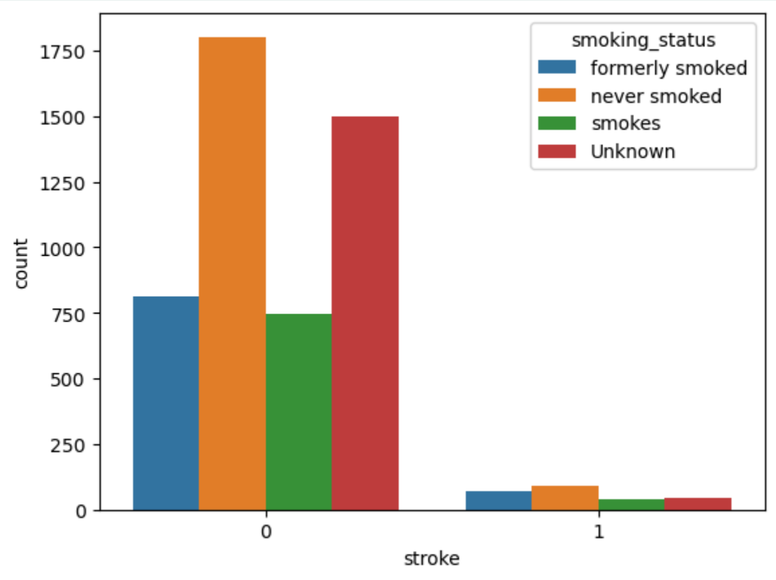
WORK TYPE ANALYSIS

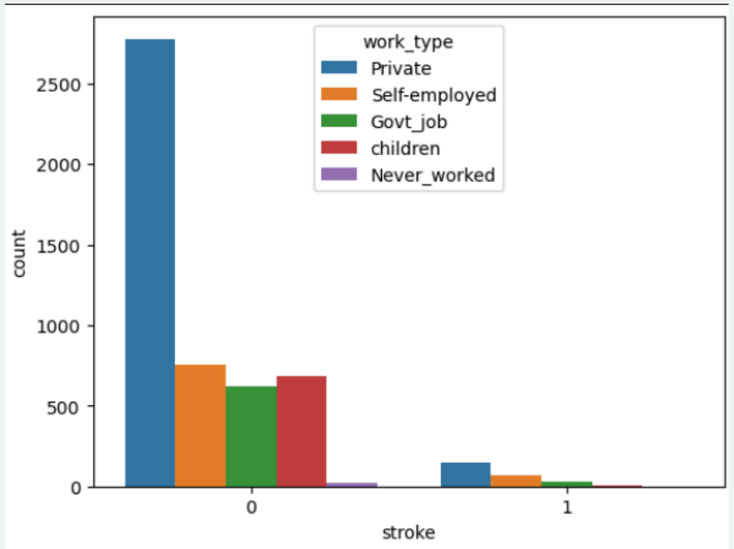


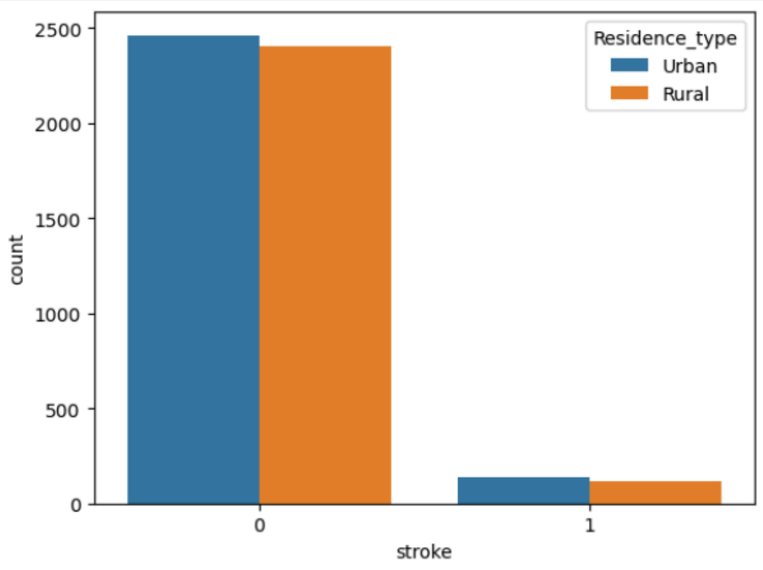
SMOKING STATUS ANALYSIS

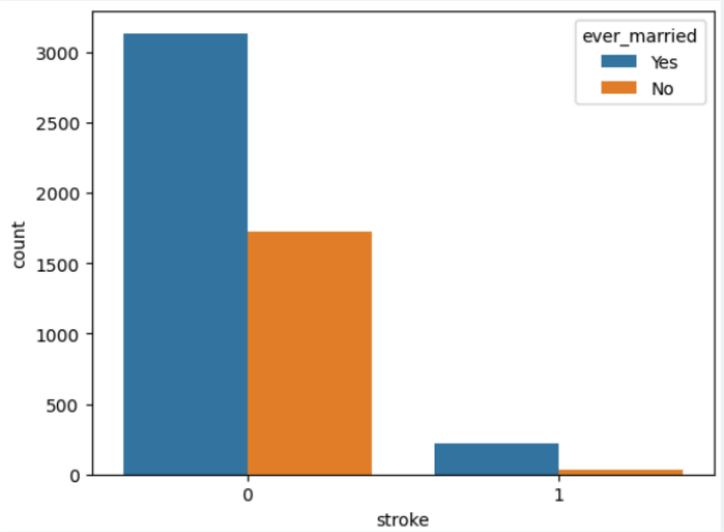


RESIDENCE TYPE ANALYSIS


smoking_status
formerly smoked
never smoked
smokes
Unknown
count
stroke
0
1
0
250
500
750
1000
1250
1500
1750


work_type
Private
Self-employed
Govt_job
children
Never_worked
count
stroke
0
1
0
500
1000
1500
2000
2500


Residence_type
Urban
Rural
count
stroke
0
1
0
500
1000
1500
2000
2500


ever_married
Yes
No
count
stroke
0
1
0
500
1000
1500
2000
2500
3000

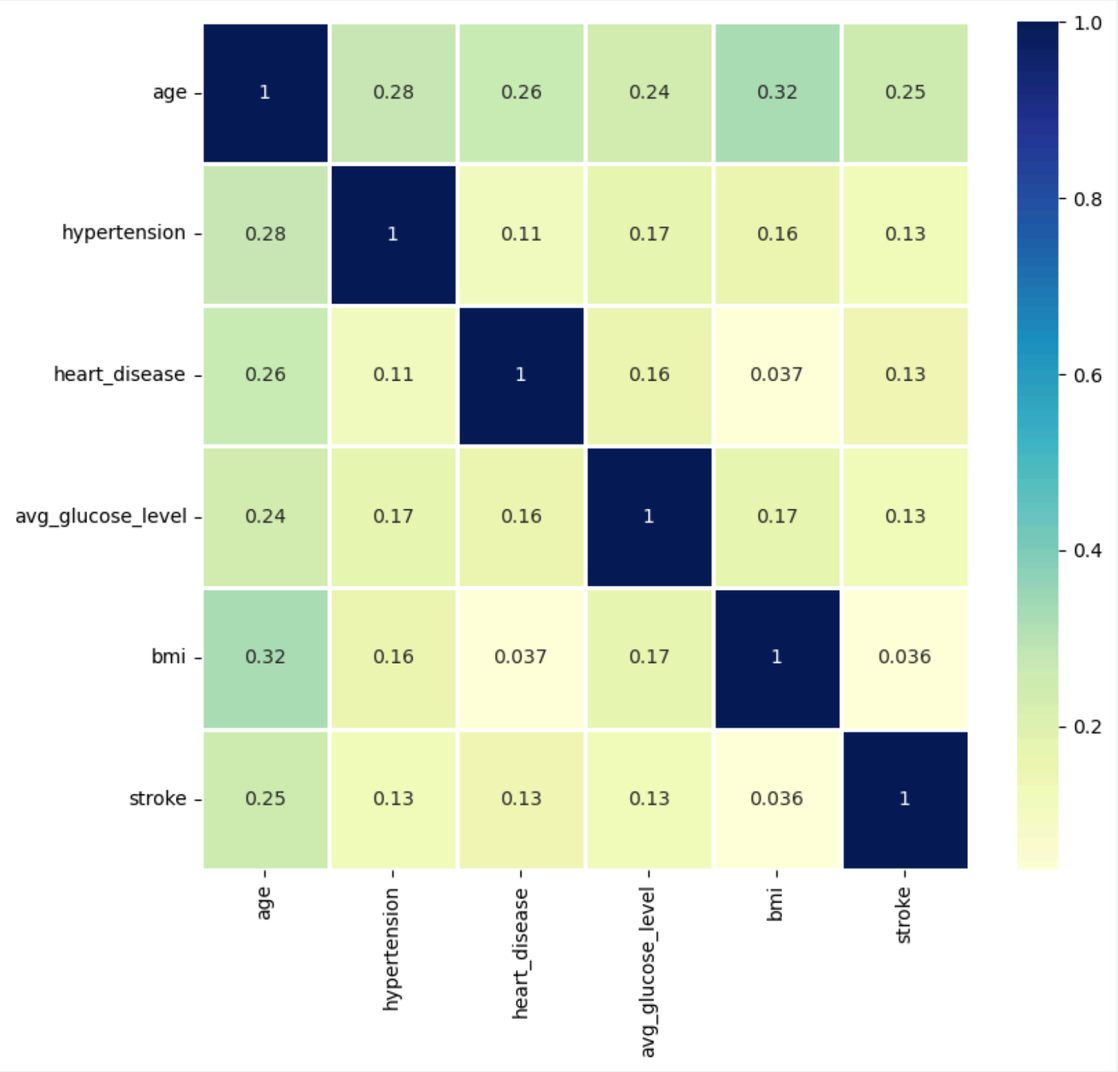
CORRELATION MATRIX

* There is a weak correlation between the attributes as per the plotted heatmap

* The highest correlation found was between age and bmi – 0.32

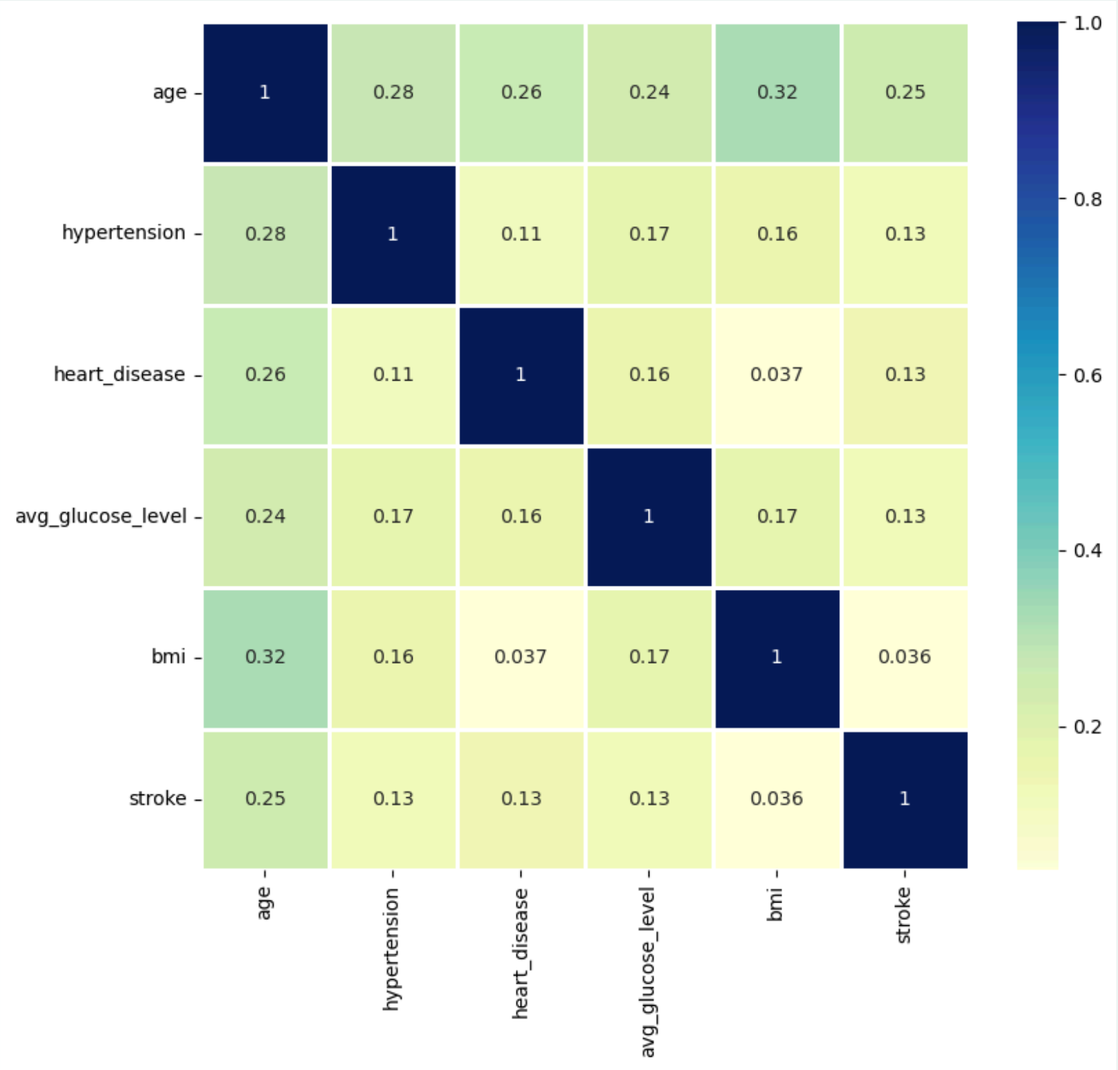
* Rest all correlations were less than 0.32

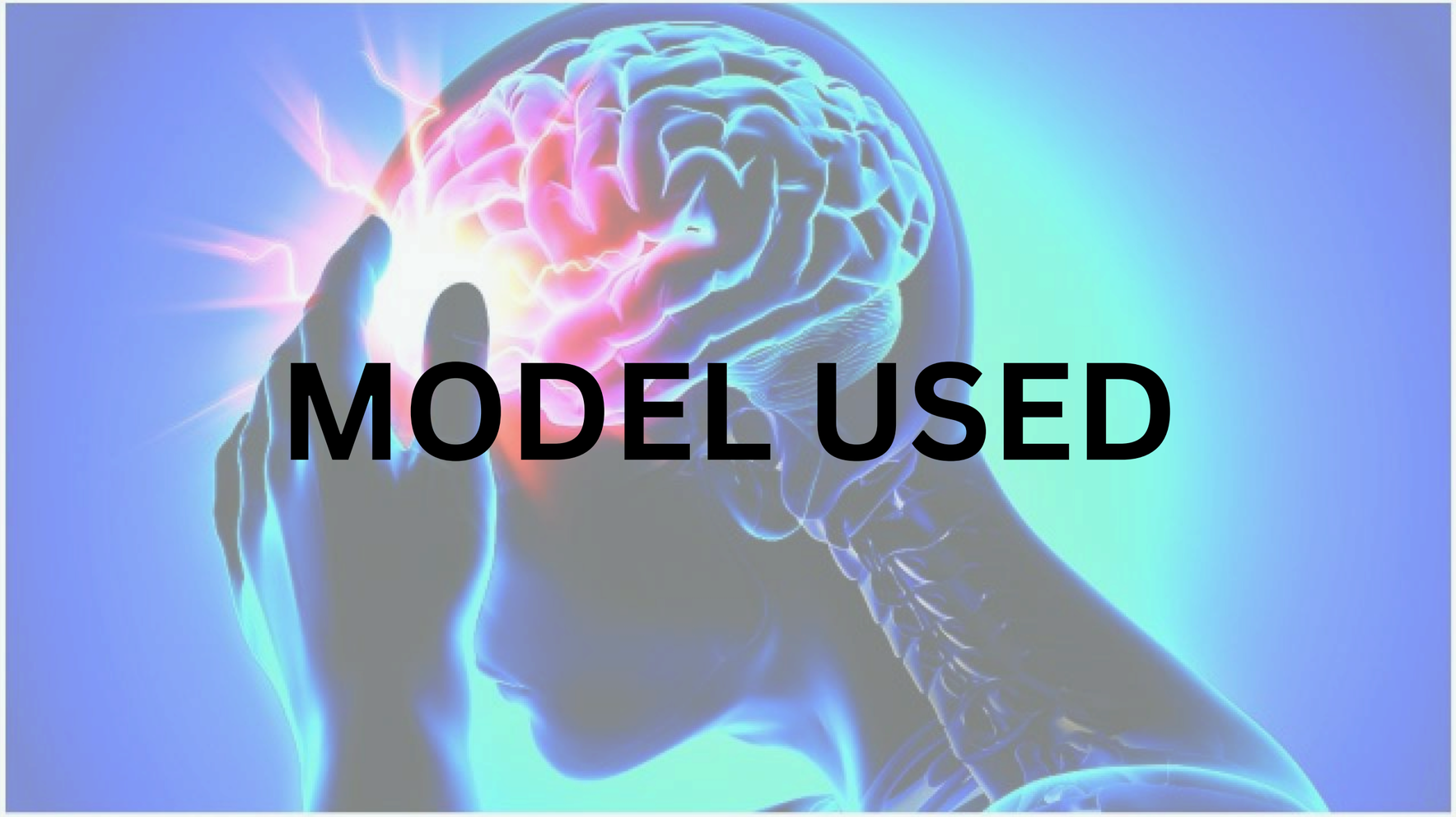
* We could not draw any statistical insight from heatmap



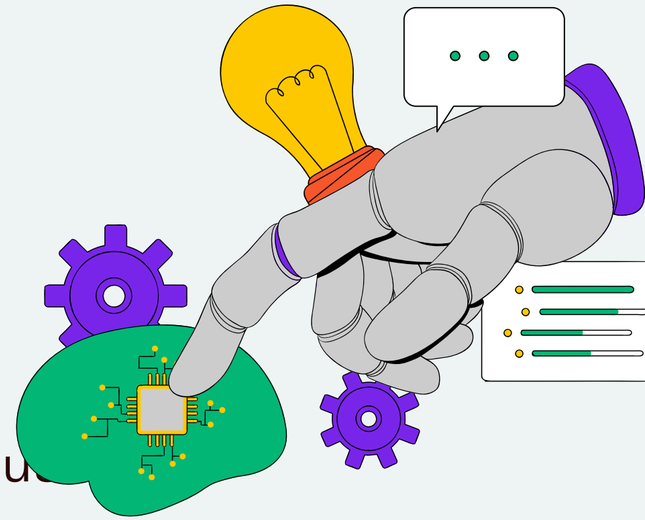
Insights:

- We can see there is not any high correlation between target feature and other features. Small positive correlation between target feature and Age, Hypertension, Heart Disease, Average Glucose Level.
- Small positive correlation between Age and Stroke, Hypertension, Heart Disease, Average Glucose Level, BMI.
- Small positive correlation between Smoking Status and Marital Status, Occupation Type and BMI.
- Medium positive correlation between Age and Occupation Type.
- Medium negative correlation between Age and Marital Status.




MODEL USED

K-NEAREST NEIGHBORS (KNN):

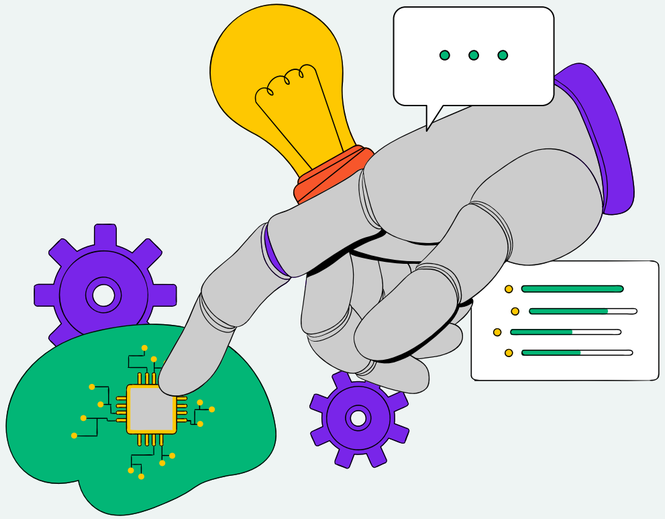


- K-Nearest Neighbors is a simple, intuitive, non-parametric algorithm u
  statistical classification and regression.
- It classifies a data point based on how its neighbors are classified. In regression, it predicts the value based on the average of the neighbors.

WORKING:

- KNN works by finding the k closest data points (neighbors) to a new, unlabelled point and predicting its label (classification) or value (regression) based on these neighbors.
- It uses different distance metrics like Euclidean, Manhattan, and Minkowski, which are critical in finding the nearest neighbors.

- Model Training and Evaluation:
  - Train the KNN model using historical patient data, including features such as age, blood pressure, cholesterol levels, and medical history.
  - Evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score.
  - Determine the optimal value of 'k' (number of neighbors) through hyperparameter tuning and cross-validation techniques.

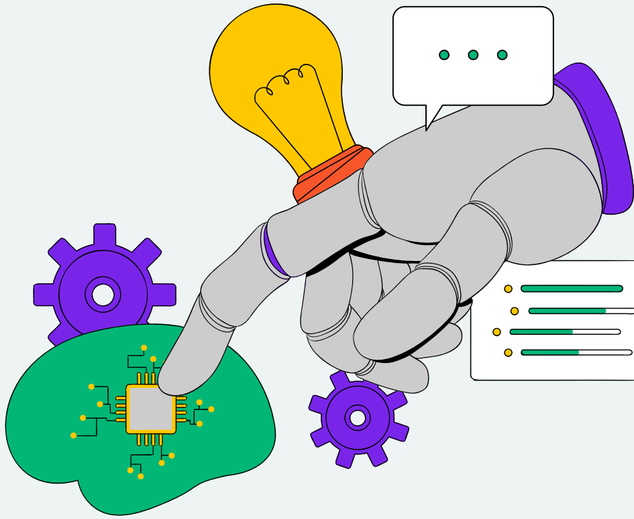


```python
#importing the KNN Classifier module
from sklearn.neighbors import KNeighborsClassifier
# Libraries for calculating performance metrics
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
from sklearn.metrics import auc,roc_auc_score,roc_curve,precision_score,recall_score,f1_score

# Create the classifier object
# 2 neighbours because of the 2 classes
knn = KNeighborsClassifier(n_neighbors = 2)
# Training the classifier
knn.fit(X_train,y_train)
#predicting result using the test dataset
y_pred_knn = knn.predict(X_test)
y_pred_prob_knn = knn.predict_proba(X_test)[:, 1]

# Printing the accuracy and roc-auc score of the model
confusion_matrix(y_test, y_pred_knn)
print('Accuracy:',accuracy_score(y_test, y_pred_knn))
print('ROC AUC Score:', roc_auc_score(y_test, y_pred_prob_knn))
```

Python

```
Accuracy: 0.9722365038560411
```

LOGISTIC REGRESSION:



- Logistic Regression is a statistical method for modeling binary outcomes. In essence, it predicts the probability of the occurrence of an event by fitting data to a logistic curve.

- It is a type of supervised learning algorithm used primarily for classification tasks.

- Introduce the logistic (sigmoid) function,which outputs values between 0 and 1, making it suitable for modeling probabilities

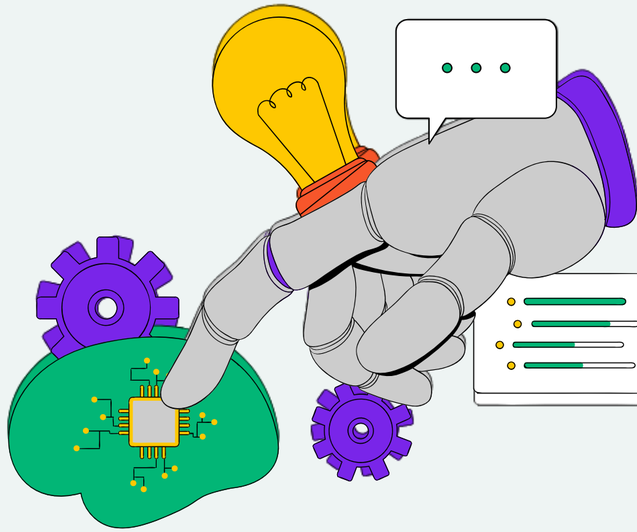
- Logistic Regression determines a decision boundary — a threshold over which the outcome changes from one class to another.

```python
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

y_pred_lr = classifier.predict(X_test)

confusion_matrix(y_test, y_pred_lr)
print('Accuracy:', accuracy_score(y_test, y_pred_lr))
```

Python

```
Accuracy: 0.770694087403599
```

DECISION TREE:



- A decision tree is a flowchart-like tree structure where each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.
- Useful for both categorical and continuous input and output variables.

WORKING:

- Begins with the dataset at the root node and splits the data down the tree in accordance with feature values that result in the largest information gain (IG) or the greatest reduction in impurity (Gini impurity, entropy).

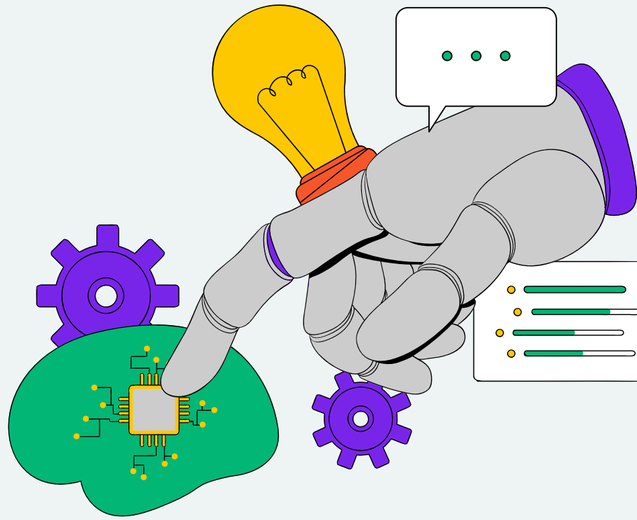
- At each node, the algorithm chooses the best split at that level using a greedy approach to optimize a certain criterion (e.g., Gini, entropy).
- This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same values of the target variable, or when splitting no longer adds value.

```python
#importing the Decision Tree Classifier module
from sklearn.tree import DecisionTreeClassifier
# Libraries for calculating performance metrics
from sklearn import metrics
from sklearn.metrics import auc,roc_auc_score,roc_curve,precision_score,recall_score,f1_score

# Create the classifier object
clf = DecisionTreeClassifier()

# Training the classifier
clf = clf.fit(X_train,y_train)

#predicting result using the test dataset
y_pred = clf.predict(X_test)

# Printing the accuracyof the model
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

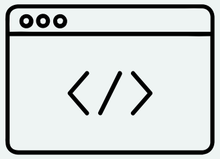
Python

```
Accuracy: 0.9794344473007712
```

XGBOOST(EXTREME GRADIENT BOOSTING):



- XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable.
- It implements machine learning algorithms under the Gradient Boosting framework, providing a scalable, efficient, and effective solution for both regression and classification problems.
- XGBoost is faster than other implementations of gradient boosting due to its core algorithm and system optimization.
- Built-in handling of missing data, which allows it to handle sparse datasets effectively.
- Uses hardware optimization techniques like parallel processing and cache awareness to speed up computations.



CODE:

```
#importing the XGBoost Classifier module
from xgboost  import XGBClassifier

# Create the classifier object
xgb = XGBClassifier()
# Training the classifier
xgb.fit(X_train,y_train)
#predicting result using the test dataset
y_pred_xgb = xgb.predict(X_test)
y_pred_prob_xgb = xgb.predict_proba(X_test)[:, 1]

# Printing the accuracy and roc-auc score of the model
print('Accuracy:', accuracy_score(y_test, y_pred_xgb))
print('ROC AUC Score:', roc_auc_score(y_test, y_pred_prob_xgb))

# plots of roc_auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_xgb)

plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, linewidth=2, color= 'teal')
plt.plot([0,1], [0,1], 'r--' )
plt.title('ROC Curve of XGBOOST')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

plt.show()
```

```
Accuracy: 0.9773778920308483
```

REFERENCE:

https://www.kaggle.com/datasets/zzettrkalpakbal/full-filled-brain-stroke-dataset

[1] Singh, M.S., Choudhary, P., Thongam, K.: A comparative analysis for various stroke prediction techniques. In: Springer, Singapore (2020)
[2] Pradeepa, S., Manjula, K. R., Vimal, S., Khan, M. S., Chilamkurti, N., &Luhach, A. K.: DRFS: Detecting Risk Factor of Stroke Disease from Social Media Using Machine Learning Techniques. In Springer (2020).
[3] Dataset named „Stroke Prediction Dataset" from Kaggle: https://www.kaggle.com/fedesoriano/strokeprediction-dataset [4] Wang, S., Li, Y., Tian, J., Peng, X., Yi, L., Du, C., Feng, C., Liang, X. (2020). A randomized controlled trial of brain and heart health manager-led mHealth secondary stroke prevention. Cardiovascular Diagnosis and Therapy, 10(5): 1192-1199. https://doi.org/10.21037/cdt-20-423
[5] Wilkinson, D.A., Daou, B.J., Nadel, J.L., Chaudhary, N., Gemmete, J.J., Thompson, B.G., Pandey, A.S. (2020). Abdominal aortic aneurysm is associated with subarachnoid hemorrhage. Journal of Neuro Interventional Surgery. https://doi.org/10.1136/neurintsurg2020-016757
[6] Verma, A., Jaiswal, S., Sheikh, W.R. (2020). Acute thrombotic occlusion of subclavian artery presenting as a stroke mimic. Journal of the American College of Emergency Physicians Open, 1(5): 932-934. https://doi.org/10.1002/emp2.12085
[7] Yu, J., Park, S., Lee, H., Pyo, C.S., Lee, Y.S. (2020). An elderly health monitoring system using machine learning and in-depth analysis techniques on the NIH stroke scale. Mathematics, 8(7): 1-16. https://doi.org/10.3390/math8071115