

# Stroke prediction model comparison

Alex, Himali, Swapna, Vy



“Globally, the World Health Organisation estimates that there are at least 62 million stroke survivors. In Australia, 50,000 people suffer a stroke each year, leaving them with physical and mental disabilities that create an enormous emotional, social, and financial burden on our community.”

- **Aim:**

Develop a model that can be used to predict stroke based on personal status and health condition

- **Objectives:**

- (1) Data import and pre-processing (PySpark, Pandas) by remove outlier, null value, non-informative features
- (2) Building predicting models (K-means, PCA, Neural Network...)
- (3) Compare and suggest the best model

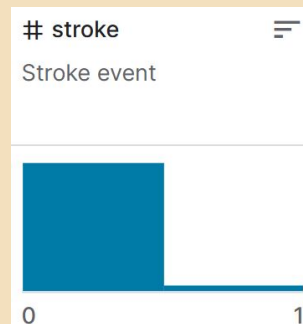
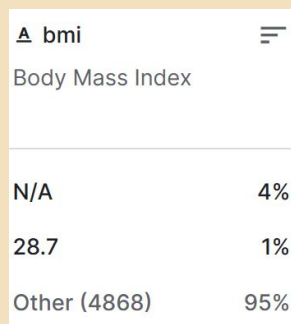
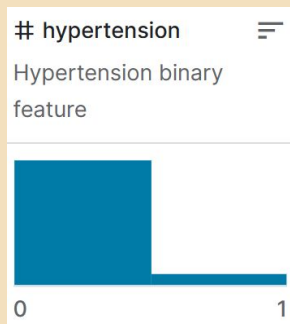
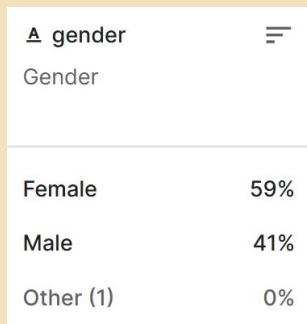
## **Objective 1 - Data import and pre-processing**

- Import data using Spark
- Inspect data

*Features:* Id, gender, age, hypertension, heart\_disease, ever\_married, work\_type, Residence\_type, avg\_glucose\_level, bmi, smoking\_status

*Label:* stroke

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
56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smoked	1
53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
10434	Female	69	0	0	No	Private	Urban	94.39	22.8	never smoked	1
27419	Female	59	0	0	Yes	Private	Rural	76.15	N/A	Unknown	1



## Data pre-processing

- Remove row with “Other” value from “gender” feature
- Remove non-informative feature “Id”
- Remove rows with “N/A” value in the “bmi”
- Remove missing value from “stroke”

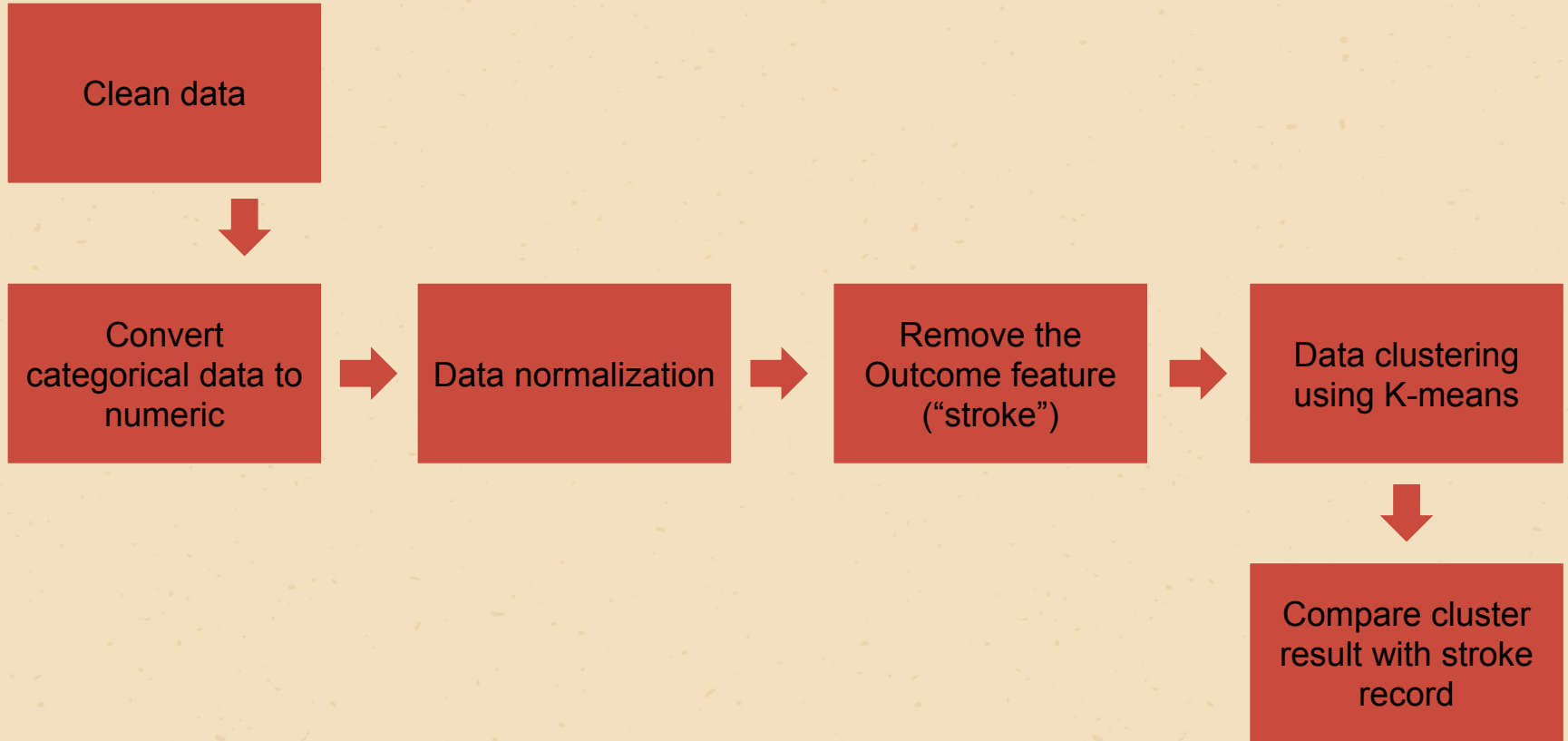
Number of Data point	
Female	2994
Male	2115
Other	1



Number of Data point	
Female	2897
Male	2011

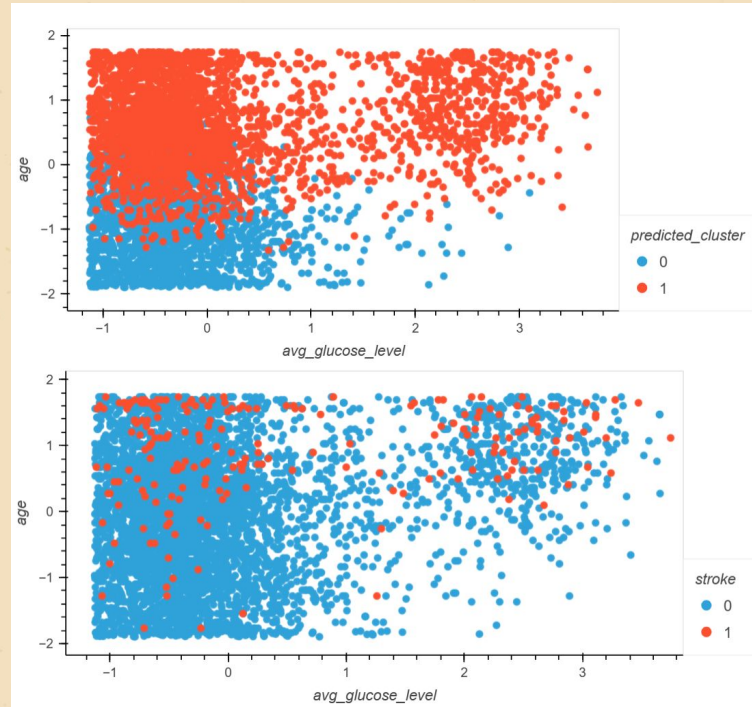
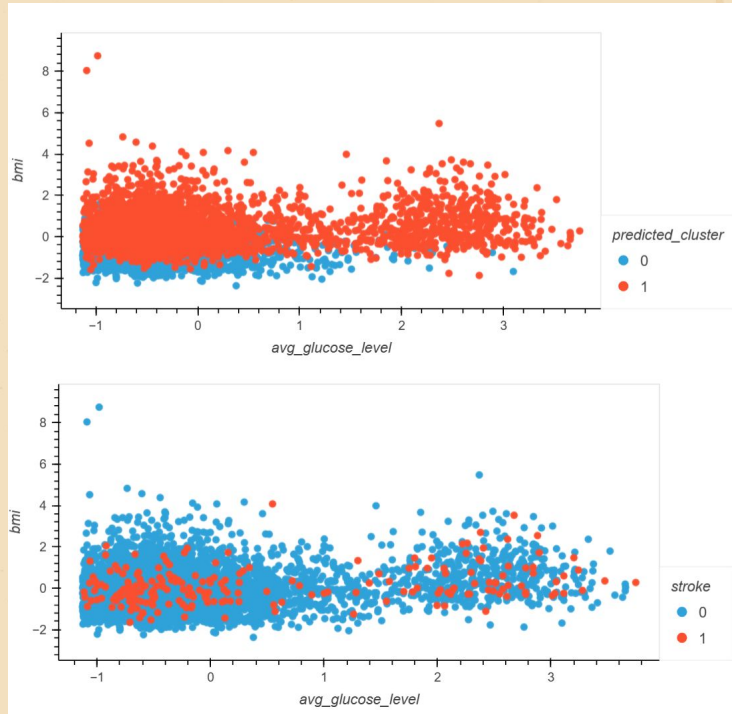
## **Objective 2 - Building predicting models**

# K-means Clustering





# K-means Clustering



➤ K-mean clustering method appears to cluster the patients differently when compared to actual stroke record from patients. Therefore, using K-mean clustering might not be a good option for this dataset.

# K means clustering

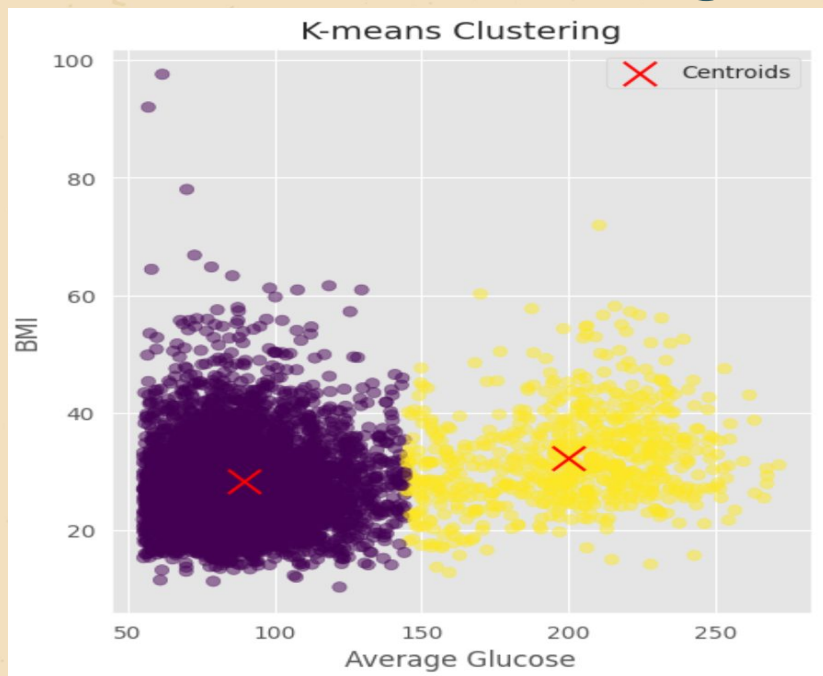


Figure 1: Visualisation of true datapoints K-means trained only against "BMI and average glucose levels"

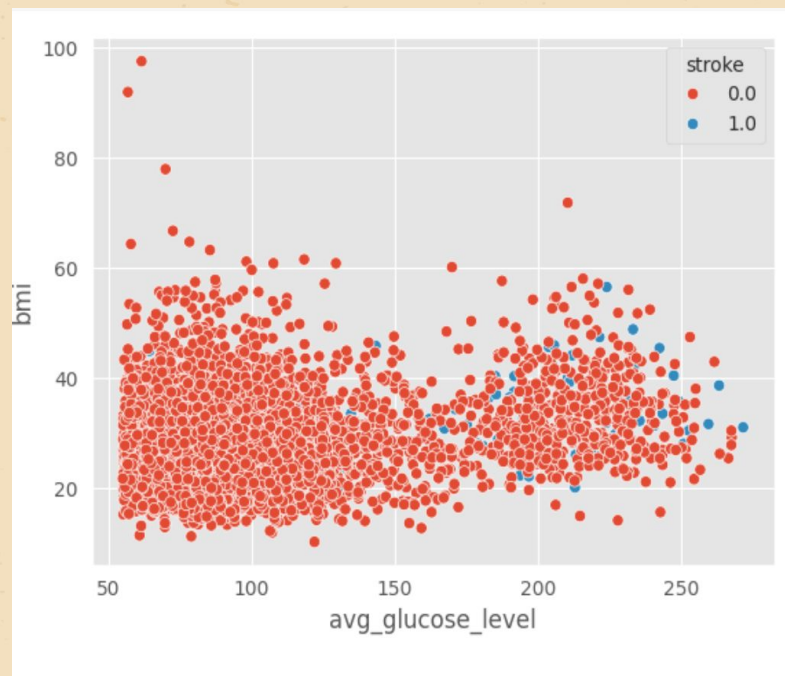
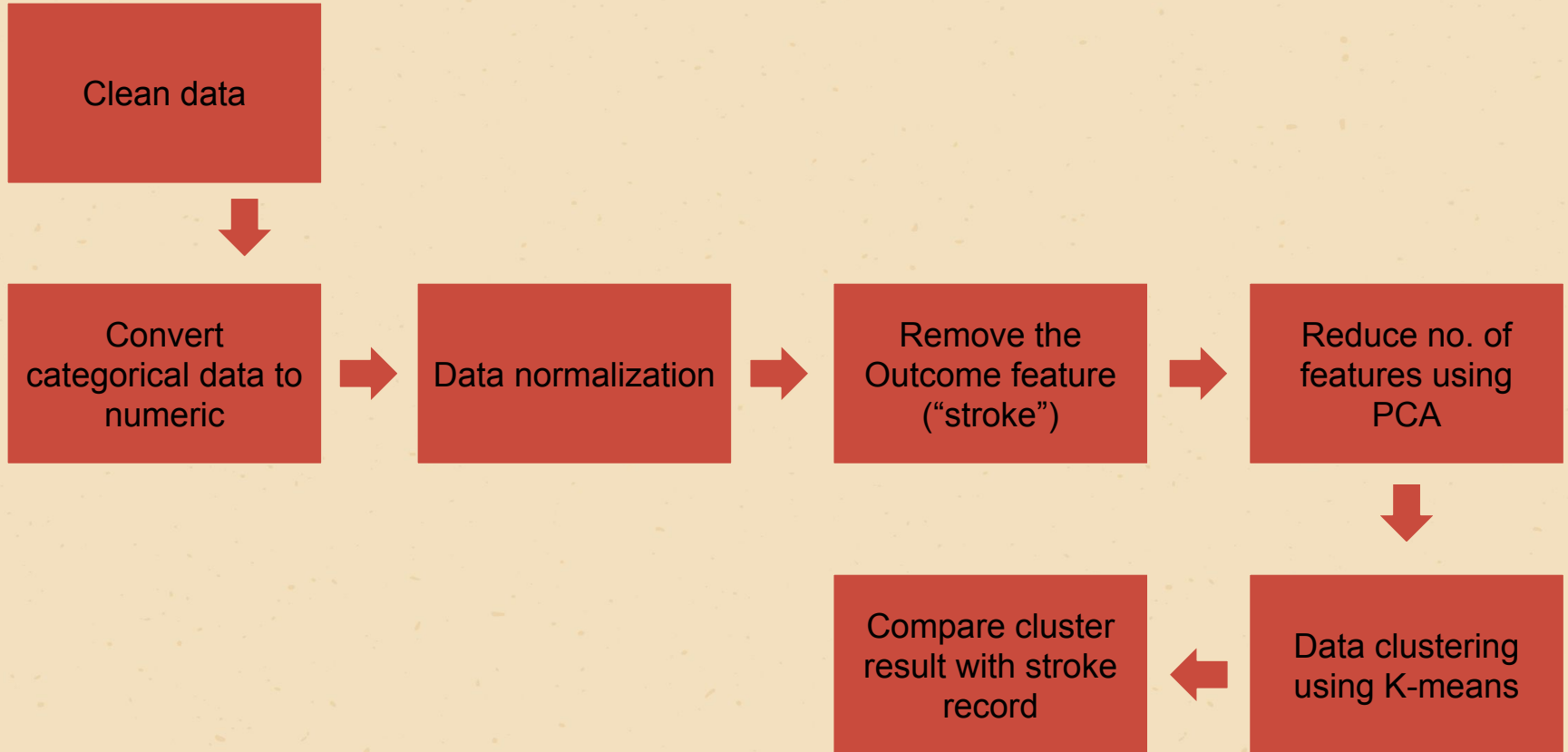


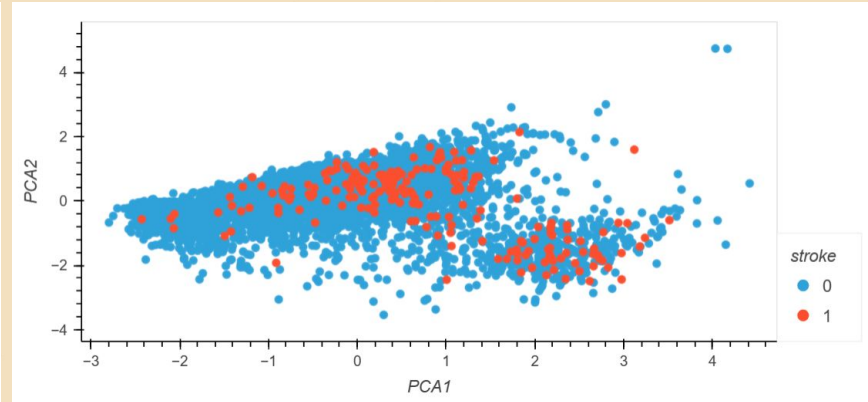
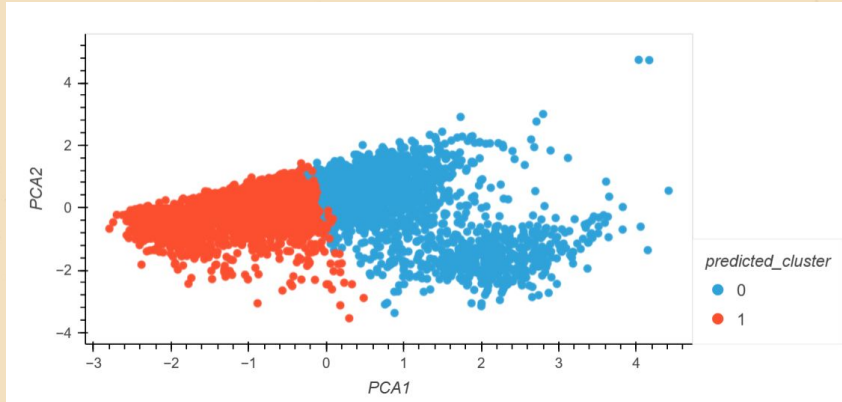
Figure 2 : Scatter plot displaying stroke status

- Yellow cluster seems to have more strokes however, due to the skewed nature of the dataset, strokes in the purple cluster lack visibility. This can be seen when compared to bottom left figure in previous slide.
- K means is not a suitable predicting model for this particular dataset.

# K-means with PCA



# K-means with PCA



- K-mean clustering with PCA method appears to cluster the patients differently when compared to actual stroke record from patients. Therefore, using K-mean clustering might not be a good option for this dataset.

# Random Forest model

- Convert data to fit in model
- All values converted in to binary values except age, bmi, avg\_glucose level.

ever_married	Residence_type	gender_Female	gender_Male	gender_Other	hypertension_0	...	heart_disease_1
1	1	0	1	0	1	...	1
1	0	1	0	0	1	...	0
1	0	0	1	0	1	...	1
1	1	1	0	0	1	...	0
1	0	1	0	0	0	...	0

Female 2994

Male 2115

Other 1

Yes 3353

No 1757

Name: ever\_married, dtype:  
int64

Private 2925

Self-employed 819

children 687

Govt\_job 657

Never\_worked 22

Name: work\_type, dtype: int64

# Creating and train random forest classifier

- X is features or independent variables
- Y is targeted variable

```
# Separate features (X) and target variable (y)
X = df_dummies.drop(columns=['stroke'])
y = df_dummies['stroke']
```

- Split data into training and testing set by 80%:20%

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True, random_state=42)
```

- Create random forest classifier and train

```
# Create a RandomForestClassifier instance
rf_classifier = RandomForestClassifier(n_estimators=500, random_state=48)

# Train the classifier on the training data
rf_classifier.fit(X_train, y_train)
```

# Random Forest Model

- Accuracy: The proportion of correctly classified instances out of the total instances is 93%-94%
- Precision: proportion of true positive predictions out of all positive predictions made by the model is - For class 0 (94%).
- Recall (Sensitivity): The proportion of true positive predictions out of all actual positive instances in the dataset.1%
- F1 Score: The harmonic mean of precision and recall, providing a balance between the two metrics is Based on the achieved confusion matrix and classification report: 97% .

Confusion Matrix:

	Predicted 0	Predicted 1
Actual 0	960	0
Actual 1	62	0

Accuracy: 0.9393346379647749

Mean Squared Error: 0.060665362035225046

Classification Report:

	precision	recall	f1-score	support
0	0.94	1.00	0.97	960
1	0.00	0.00	0.00	62

Overall presented data set contains more no strokes then strokes



# Performance of a Random Forest Model

## Iterative Model Training

- Used 11 ITERATIONS to train and predict data ,each time creating a new instance of a Random Forest Classifier with a different random state and different shuffled data from given set
- Accuracy is 93%-94%

Iteration	Accuracy	Precision	Recall	F1-score
1	0.938356	0	0	0
2	0.938356	0	0	0
3	0.938356	0	0	0
4	0.939335	0.5	0.016129	0.03125
5	0.938356	0	0	0
6	0.940313	0.666667	0.0322581	0.0615385
7	0.938356	0	0	0
8	0.938356	0	0	0
9	0.940313	0.666667	0.0322581	0.0615385
10	0.940313	0.666667	0.0322581	0.0615385
11	0.939335	0.5	0.016129	0.03125

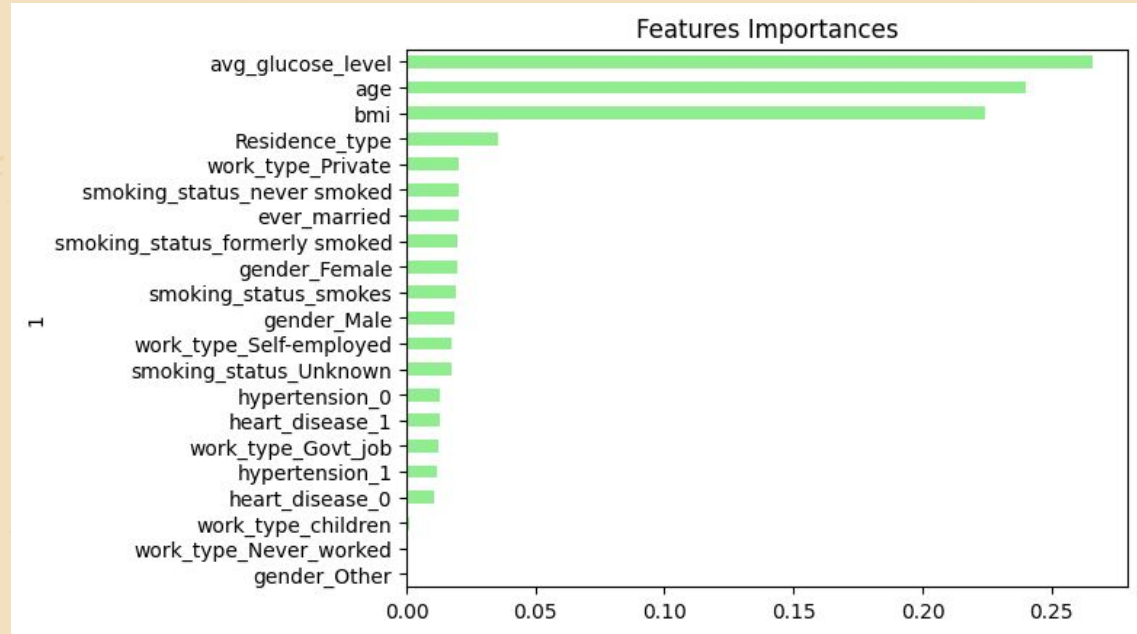
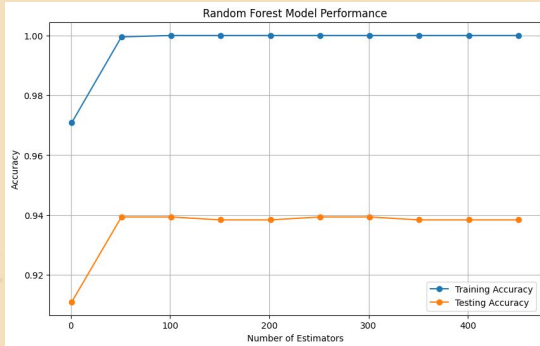


# Features importances

Top Features :

- Age
- Average glucose level,
- BMI

Used to predict stroke data

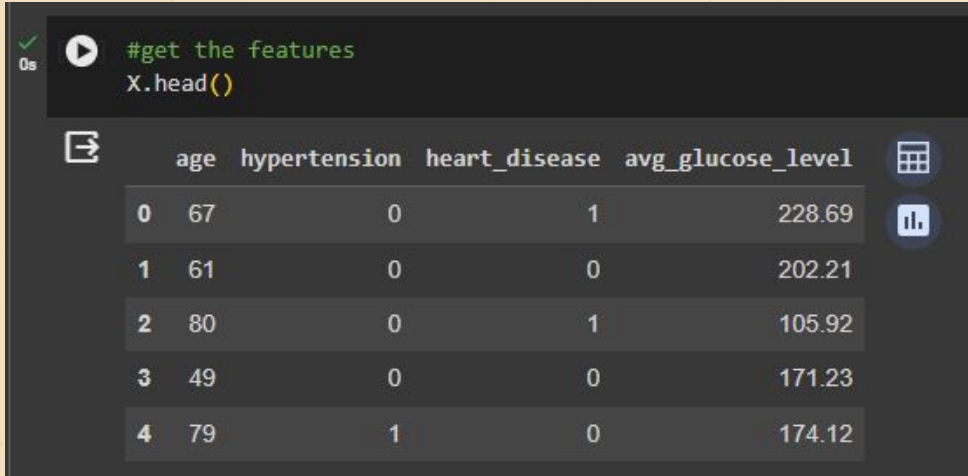


# Logistic Regression

Build Model using numeric features

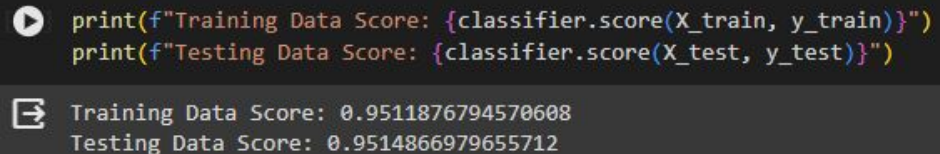
- Age
- Hypertension
- Heart Disease (True or False)
- Average Glucose Level

High accuracy score with given features to predict stroke status.



A screenshot of a Jupyter Notebook interface. At the top, a code cell is executed, showing a green checkmark and '0s' time. The code is `#get the features` and `X.head()`. Below the code, a table icon is visible. The table displays the first five rows of the feature matrix X. The columns are 'age', 'hypertension', 'heart\_disease', and 'avg\_glucose\_level'. The rows are indexed 0 to 4.

	age	hypertension	heart_disease	avg_glucose_level
0	67	0	1	228.69
1	61	0	0	202.21
2	80	0	1	105.92
3	49	0	0	171.23
4	79	1	0	174.12



A screenshot of a Jupyter Notebook interface showing the output of model evaluation. A code cell is executed, displaying the training and testing data scores. The output shows a high accuracy score for both training and testing data.

```
print(f"Training Data Score: {classifier.score(X_train, y_train)}")  
print(f"Testing Data Score: {classifier.score(X_test, y_test)}")
```

Training Data Score: 0.9511876794570608  
Testing Data Score: 0.9514866979655712

# Logistic Regression - Optimisation

- Conversion of additional features to int data types
- Feature Coefficient Analysis
- Retraining the model including only values with
- No increase to performance / accuracy score

```
#add and convert additional features
X = pandas_df.drop(['stroke', 'id', 'work_type', 'Residence_type', 'smoking_status', 'bmi'], axis=1)
X['ever_married'].fillna('No', inplace=True)
X['ever_married'] = X['ever_married'].map({'Yes': 1, 'No': 0})
X['gender'] = X['gender'].map({'Male': 1, 'Female': 0})
X.head()
```

	gender	age	hypertension	heart_disease	ever_married	avg_glucose_level
0	1	67	0	1	1	228.69
1	0	61	0	0	1	202.21
2	1	80	0	1	1	105.92
3	0	49	0	0	1	171.23
4	0	79	1	0	1	174.12

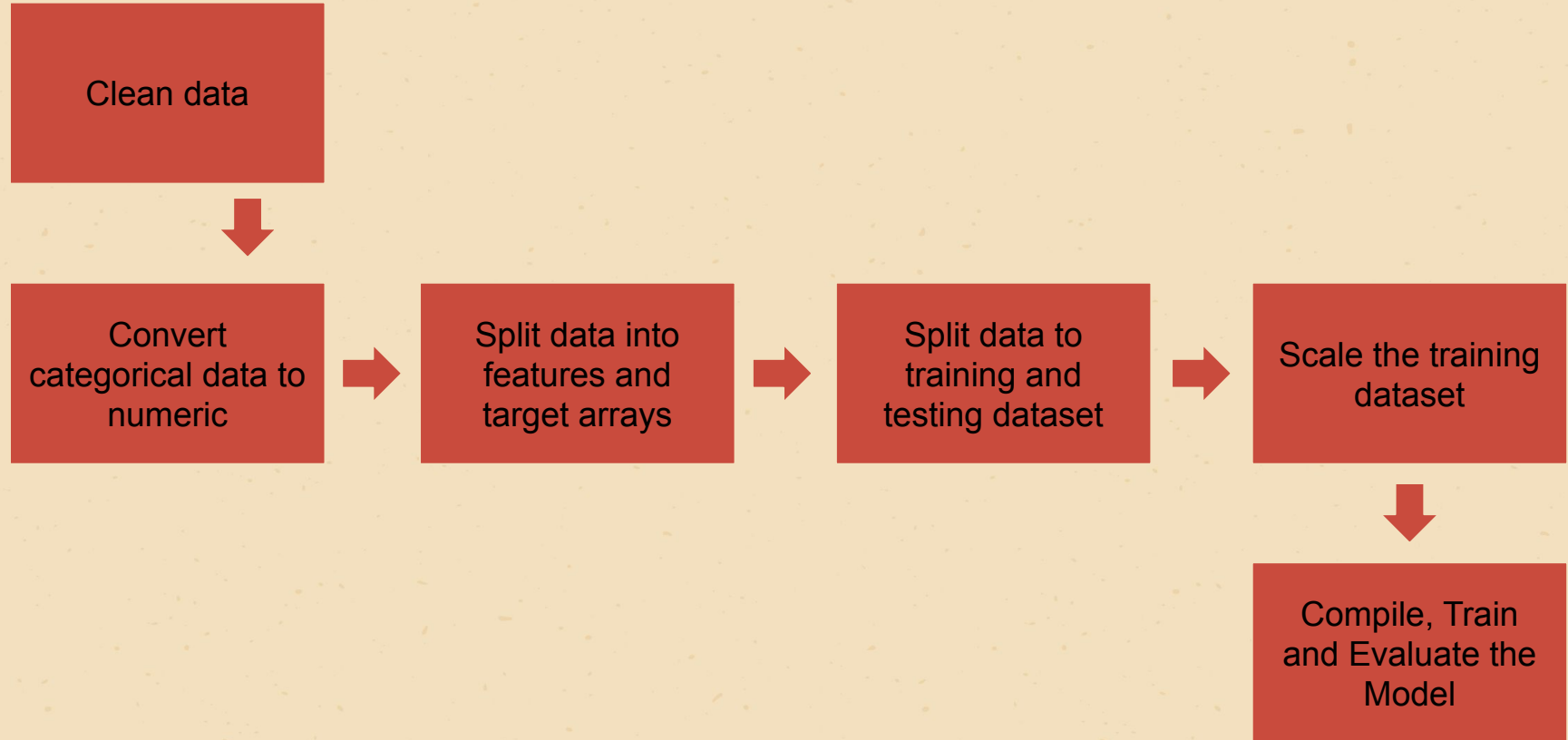
## Feature Importances:

gender: 0.0021709465205578316  
age: 0.07153149848800829  
hypertension: 0.008266308556091801  
heart\_disease: 0.008543232209240057  
avg\_glucose\_level: -0.004505799643233578  
bmi: 0.004575498287225877

```
print(f"Training Data Score: {classifier.score(X_train, y_train)}")
print(f"Testing Data Score: {classifier.score(X_test, y_test)}")
```

Training Data Score: 0.9511876794570608  
Testing Data Score: 0.9514866979655712

# Neural network - Keras model



# Neural network - Keras model - 1st attempt

## ➔ Model structure

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 4)	84
dense_1 (Dense)	(None, 4)	20
dense_2 (Dense)	(None, 1)	5
Total params: 109 (436.00 Byte)		
Trainable params: 109 (436.00 Byte)		
Non-trainable params: 0 (0.00 Byte)		

## ➔ Model evaluation on test dataset

39/39 - 0s - loss: 0.1421 - accuracy: 0.9617 - 220ms/epoch - 6ms/step  
Loss: 0.14209918677806854, Accuracy: 0.9616951942443848

# Neural network - Keras model - 2nd attempt

## ➔ Model structure

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 4)	84
dense_4 (Dense)	(None, 6)	30
dense_5 (Dense)	(None, 8)	56
dense_6 (Dense)	(None, 1)	9

=====  
Total params: 179 (716.00 Byte)  
Trainable params: 179 (716.00 Byte)  
Non-trainable params: 0 (0.00 Byte)  
=====

- Loss and accuracy is pretty good!
- Changing hyperparameters seem to not further improve the model.

## ➔ Model evaluation on test dataset

39/39 - 0s - loss: 0.1464 - accuracy: 0.9617 - 192ms/epoch - 5ms/step  
Loss: 0.14636258780956268, Accuracy: 0.9616951942443848

# Conclusion

- More data on people with strokes required to increase support / balance of result
- Best model - > Neural network

What we learnt from this data

- BMI, Age & Glucose levels are the main features contributing to strokes
- Not all the features are useful for analysis

# Questions