STROKE PREDICTION

STROKE PREDICTION

Stroke is a medical condition that can lead to the death of a person. It's a severe condition and if treated on time we can save one's life and treat them well. There can be n number of factors that can lead to strokes and in this project blog, we will try to analyze a few of them. we have taken the dataset from Kaggle. It has 11 variables and 5110 observations.

Dataset: Stroke Prediction Dataset

This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient. It has 11 features and 5110 observations.

1: Importing and Visualizing the Dataset

We download the stroke prediction dataset then read it into a Pandas Data Frame, enabling easy manipulation and analysis. making it suitable for working with structured data like CSV files. We visualize the first 100 lines of the dataset to gain an initial understanding of its structure and contents.

Out[242]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
95	2458	Female	78.0	0	0	Yes	Private	Rural	235.63	32.3	never smoked	1
96	35512	Female	70.0	0	0	Yes	Self-employed	Rural	76.34	24.4	formerly smoked	1
97	56841	Male	58.0	0	1	Yes	Private	Rural	240.59	31.4	smokes	1
98	8154	Male	57.0	1	0	Yes	Govt_job	Urban	78.92	27.7	formerly smoked	1
99	4639	Female	69.0	0	0	Yes	Govt_job	Urban	82.81	28.0	never smoked	1

100 rows × 12 columns

Table show the first 100 lines of the dataset

Information about dataset:

Feature name	Description	Range
Id	Unique identifier	Int
Gender	The gender of the participant	Male – Female - Other
Age	The age of the participant	Float
hypertension	0 doesn't have hypertension, 1 has hypertension	0 or 1
heart_disease	0 doesn't have heart disease, 1 has heart disease	0 or 1
ever_married	Yes, the participant is married No, the participant is not married	Yes or no
work_type	type of participant work	Govt_jov, never worked, Private or Self-employed
Residence_type	Type of participant residence	Rural or Urban
avg_glucose_level	average glucose level in blood	Float
bmi	body mass index	Float
smoking_status		formerly smoked, never smoked, smokes or Unknown
stroke	1 if the patient had a stroke 0 if not	0 or 1

2: Data type of each feature in the dataset:

We use the **info ()** method provided by the Pandas library. This method displays a summary of the Data Frame, including the number of non-null entries and the data types of each column

#	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

3: Handling Null values in the bmi feature:

We notice from the above table that the bmi feature has **null** values (4909 non null) and missing values within datasets can hinder analysis and modeling efforts, potentially leading to biased results or inaccurate predictions.

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

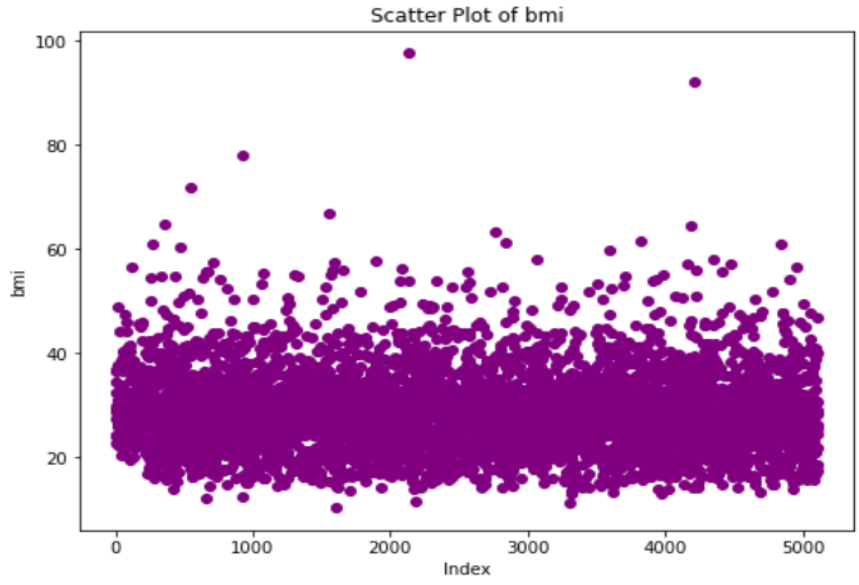
In this line we replace these null values with the mean value of the BMI feature. This approach helps maintain the integrity of the dataset by retaining valuable information while mitigating the impact of missing data on subsequent analyses.

```
 #   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                5110 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
```

Notice that there are no null values.

4: Exploring outliers in bmi feature:

We're looking at the bmi values in our dataset to see if there are any extreme values that could affect our analysis. We used a scatter plot to visualize the BMI values and check for outliers.



- Each dot on the plot stands for one person's bmi.
- There are dots on the scatter plot for the BMI feature that are far from the rest, they may indicate the presence of outliers.

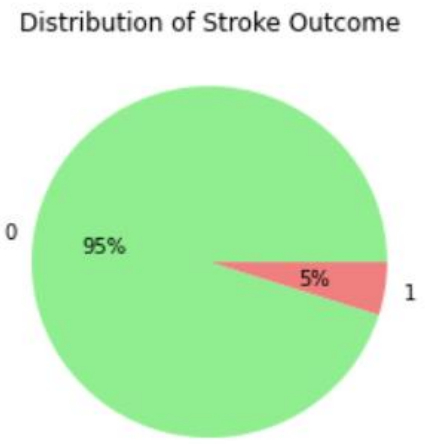
5: Handling outliers in the bmi feature:

Outliers are unusual data points that can mess up our analysis if we don't handle them properly. We fixed outliers in the BMI data to ensure our data is accurate, reliable, and unbiased, providing a clearer understanding of the overall trends in BMI within our dataset.

- First, we find the average (**mean**) BMI value in our dataset. By replacing outliers with the mean, we aim to maintain the overall distribution of the dataset while minimizing the impact of extreme values on our analysis. Additionally, using the **means** helps preserve the general characteristics and patterns present in the data.
- We set a range for outliers – BMI values that are too low (below 15) or too high (above 50).
- We look for BMI values in this range and replace them with the average BMI value we found earlier.

6: Distribution of stroke outcome:

We created a pie chart to visually represent the distribution of stroke outcomes in our dataset. The chart slices show the proportion of people who experienced strokes compared to those who didn't.

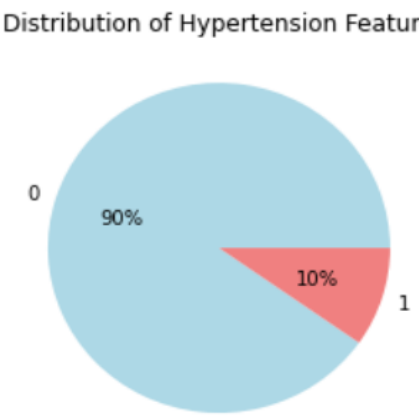


From the chart, we can see that only **5%** of the total individuals in our dataset are at risk of having a stroke. This indicates that most people in our dataset have not experienced a stroke, which is important for understanding the prevalence and risk factors associated with strokes in our population.

7: Distribution of Hypertension Feature:

We created a pie chart to show the proportion of individuals in our dataset who have hypertension, which is a condition associated with high blood pressure.

Understanding the prevalence of hypertension can help us better understand the health profile of our population and identify areas where health interventions may be needed.



The chart helps us see that out of every 100 people in our dataset, only 10 have hypertension. This suggests hypertension is not very common among the individuals in our dataset, with most people not affected by this condition.

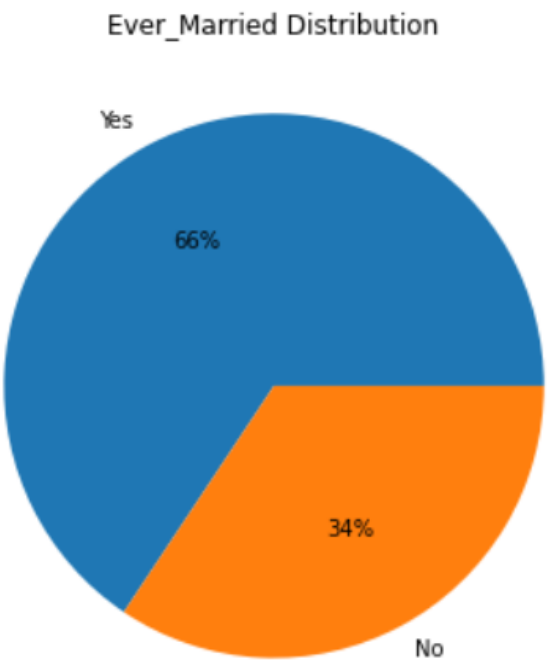
8: Distribution of Marital Status and Work Type:

We analyzed the marital status and work type distribution among individuals in our dataset.

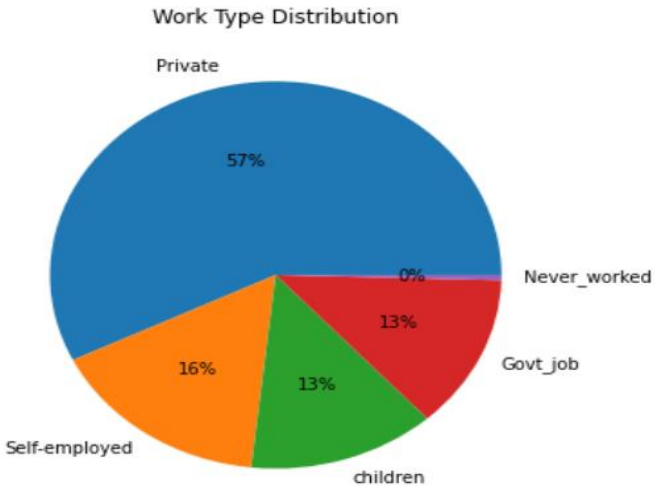
Understanding these distributions helps us gain insights into the demographics and employment patterns within our population.

We calculated the distribution of marital status using the **'ever_married'** column. And the distribution of work types using the **'work_type'** column.

We used pie charts to visualize the distribution of marital status and work type:



From the chart, we observed that over **65%** of individuals in our dataset are married. Which reveals that most individuals in our dataset are married, indicating the prevailing marital status among our population



From the chart, we found that **57%** of individuals work in the private sector Through the work type distribution chart, we identified that a substantial portion of individuals are employed in the private sector, highlighting employment preferences or opportunities.

9: Encoding Categorical Data Columns (Gender, Ever Married, Residence type):

- Converting categorical values in specific columns into numerical representations aids data analysis and modeling tasks.
- It ensures consistency and standardization in data handling and analysis processes.

- Numerical data are easier to manipulate and analyze, streamlining subsequent statistical operations and modeling tasks.
- Encoding categorical data into numerical values enhances the usability and interpretability of the dataset for analytical purposes.

We encode these columns **(Gender, Ever Married, Residence type)** with a straightforward numerical mapping as follow:

1- Replace the ***gender*** values by a numerical one Male => 0, Female =>1, other => 2.

2 - replace the ***ever_maried*** values No=>0, Yes=>1.

3 - Replace the ***residence_type*** values Rural=>0, Urban=>1

The data after converting **(Gender, Ever Married, Residence type)** columns to numerical values:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	0	67.0	0	1	1	Private	1	228.69	36.600000	formerly smoked	1
1	51676	1	61.0	0	0	1	Self-employed	0	202.21	28.893237	never smoked	1
2	31112	0	80.0	0	1	1	Private	0	105.92	32.500000	never smoked	1
3	60182	1	49.0	0	0	1	Private	1	171.23	34.400000	smokes	1
4	1665	1	79.0	1	0	1	Self-employed	0	174.12	24.000000	never smoked	1

10: Applying the one-hot encoding to the work_type feature:

We use one-hot encoding, which creates new binary columns to represent each category in 'work_type'. This process expands the dataset with new columns representing each category in 'work_type', ensuring compatibility with models while preserving the integrity of the original data.

Implementation:

1 - Extract the 'work_type' column from the dataset.

2 - Create an instance of the OneHotEncoder.

3 - Fit and transform the 'work_type' data using the encoder to generate binary columns.

4 - Convert the encoded data into a DataFrame for easy manipulation.

5 - Concatenate the encoded DataFrame with the original DataFrame, dropping the original 'work_type' column.

The data after applying one hot encoding to “work_type” feature. Notice the new columns (***Work_Govt_job , work_Never_worked , work_Private , work_Self_employed , work_childred***

ever_married	Residence_type	avg_glucose_level	bmi	smoking_status	stroke	work_Govt_job	work_Never_worked	work_Private	work_Self-employed	work_children
1	1	228.69	36.600000	formerly smoked	1	0.0	0.0	1.0	0.0	0.0
1	0	202.21	28.893237	never smoked	1	0.0	0.0	0.0	1.0	0.0
1	0	105.92	32.500000	never smoked	1	0.0	0.0	1.0	0.0	0.0
1	1	171.23	34.400000	smokes	1	0.0	0.0	1.0	0.0	0.0
1	0	174.12	24.000000	never smoked	1	0.0	0.0	0.0	1.0	0.0

11: Applying the label encoding to the smoking status feature:

Label Encoding refers to converting the labels into a numeric form to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

Implementation:

1 - Create an instance of LabelEncoder, which is a tool provided by scikit-learn.

2 - Apply Label Encoding to the 'smoking_status' column using the fit_transform() method. This assigns a 3 - unique numerical label to each category in the column.

4 - Drop the original 'smoking_status' column from the DataFrame to replace it with the encoded version.

The data after applying label encoding to “smoking_status” feature. Notice ***smoking_status_encoded*** column

ed	Residence_type	avg_glucose_level	bmi	stroke	work_Govt_job	work_Never_worked	work_Private	work_Self-employed	work_children	smoking_status_encoded
1	1	228.69	36.600000	1	0.0	0.0	1.0	0.0	0.0	1
1	0	202.21	28.893237	1	0.0	0.0	0.0	1.0	0.0	2
1	0	105.92	32.500000	1	0.0	0.0	1.0	0.0	0.0	2
1	1	171.23	34.400000	1	0.0	0.0	1.0	0.0	0.0	3
1	0	174.12	24.000000	1	0.0	0.0	0.0	1.0	0.0	2

12: Correlation Matrix using Heatmap:

It is sometimes useful to measure the inter-feature correlation. If we find that a feature is highly correlated with the class (target), it could be an indication that this feature is informative about the class. In addition, if a feature is highly correlated with other features, we could remove one of them thus reducing the complexity while potentially improving the model's learning.

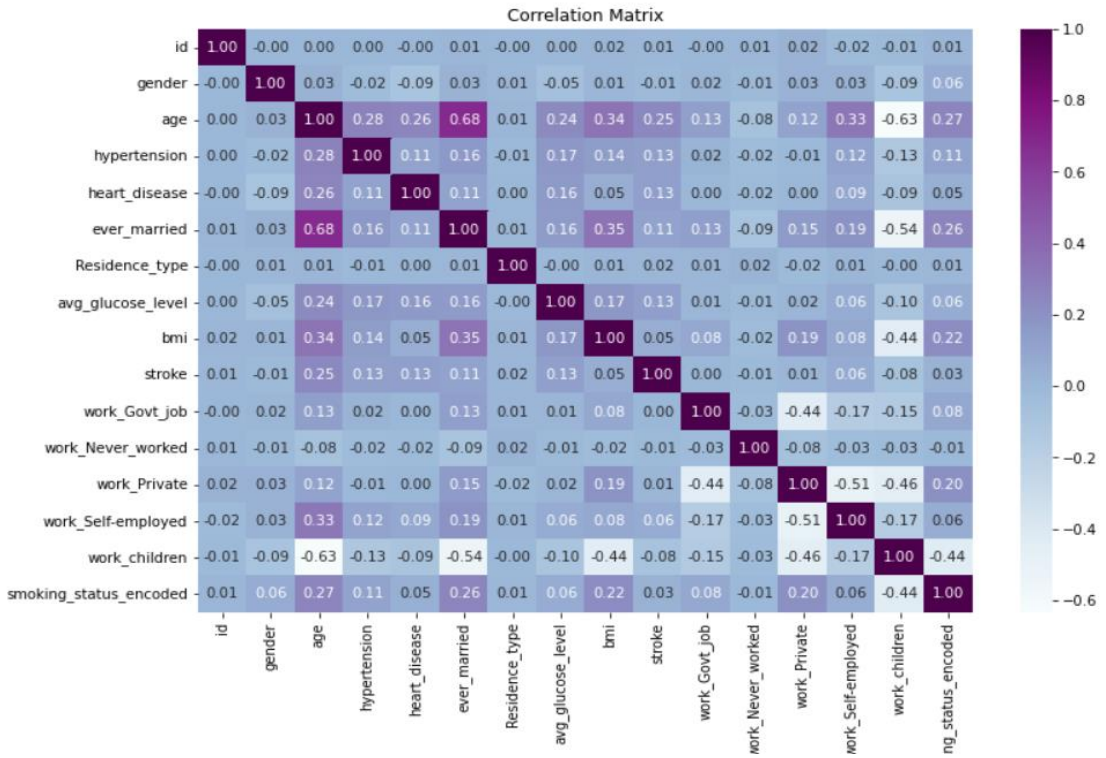
Recognize potential relationships between input and target variables, aiding feature selection in predictive modeling.

- Darker colors represent stronger correlations, while lighter colors indicate weaker correlations.
- Correlation coefficients range from [-1 to 1], where:

  1 => indicates a perfect positive correlation,

  -1 => indicates a perfect negative correlation, and

  0 => indicates no correlation.



13: 3D Scatter Plot for (bmi,stroke,age) columns:

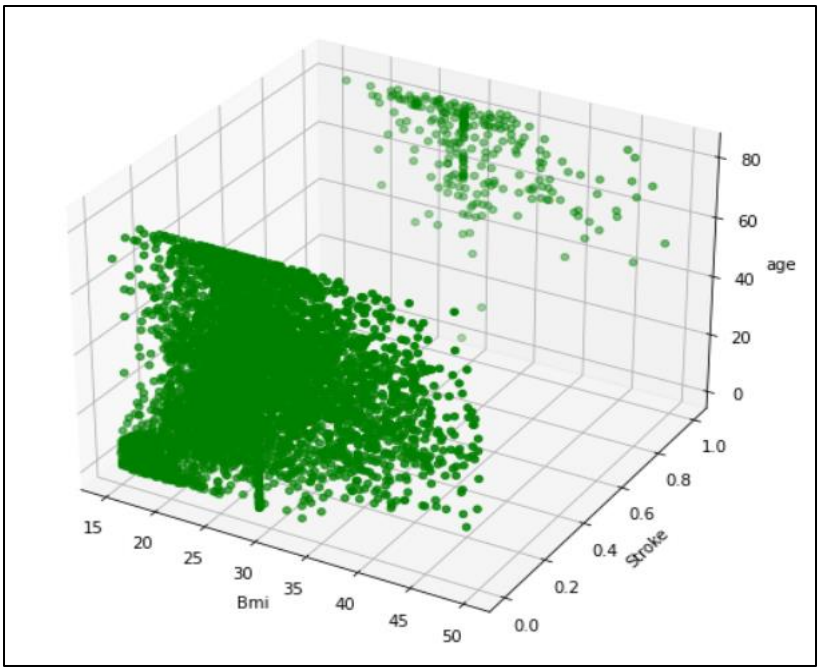
The 3D scatter plot visualizes the relationship between three variables: BMI, Stroke, and Age.

Each data point represents an individual from the dataset, with coordinates corresponding to their BMI, Stroke status, and Age.

Implementation:

1 – extract the data(bmi,stroke,age) from the Data Frame (df_encoding).

2 - Create the plot using Matplotlib's scatter () function in a 3D projection.



- We can see how BMI, Stroke status, and Age relate to each other.
- The way data points are grouped or spread out in the 3D space may suggest connections between BMI, Stroke, and Age.
- We can spot trends among people with different BMI levels, Stroke status, and Age ranges
- Understanding these patterns could help us learn more about what factors contribute to strokes and how to assess and prevent them effectively.

14: PCA to reduce the number of features to only 2 features

- PCA simplifies data analysis by reducing the number of variables in a dataset, making it easier to handle and analyze.
- PCA helps in selecting the most important variables from a dataset, particularly useful in machine learning scenarios with numerous variables
- PCA enables visualizing high-dimensional data in two or three dimensions, aiding interpretation and understanding.

- PCA compresses data by representing it with a smaller set of principal components, reducing storage requirements and processing time.

Implementation:

1. Separates the features and the target variable.
2. Applies PCA with two components.
3. Transforms the features using PCA.
4. Creates a new Data Frame with reduced features.
5. Adds the target variable to the new Data Frame.

The data Frame with 2 features:

	Pca1	Pca2	stroke
0	27471.828937	125.004457	1
1	-15158.170939	97.574904	1
2	5405.829195	5.555966	1
3	-23664.170850	65.269876	1
4	34852.829082	72.604677	1

15: K-means on the data frame with number of clusters = 3:

- K-means is straightforward to understand and implement, making it a popular choice for clustering tasks.
- It is computationally efficient and can handle large datasets with many variables.
- K-means scales well with the size of the dataset, making it suitable for clustering large datasets.
- K-means can be applied to a variety of data types and is effective in identifying clusters with spherical shapes.

Implementation:

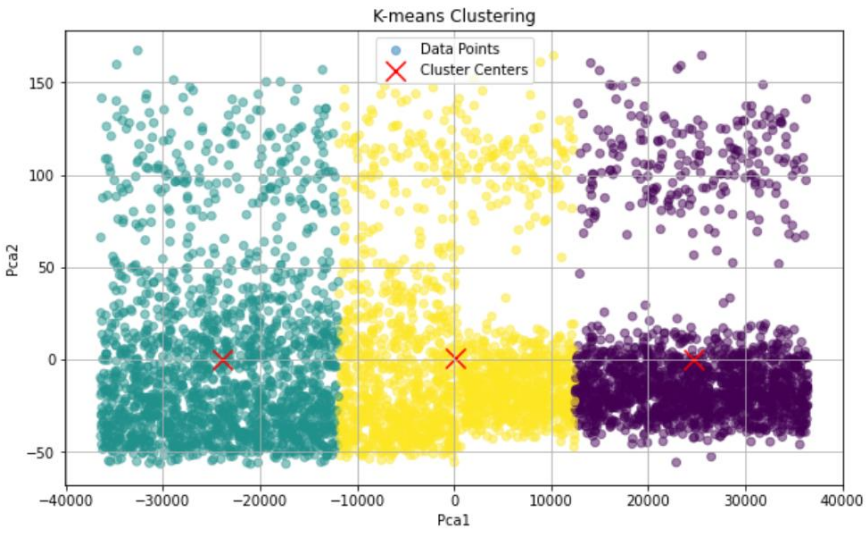
1. Separates the features (X) and target variable (y).
2. Applies K-means clustering with 3 clusters.
3. Adds cluster labels to the DataFrame.

The data Frame with K-means:

	Pca1	Pca2	stroke	cluster
0	27471.828937	125.004457	1	0
1	-15158.170939	97.574904	1	1
2	5405.829195	5.555966	1	2
3	-23664.170850	65.269876	1	1
4	34852.829082	72.604677	1	0

16: Scatter Plot to data and K-means clusters:

1. The scatter plot is created to visualize the data points in a two-dimensional space using the 'Pca1' and 'Pca2' features.
2. Each data point is colored according to its assigned cluster label, which is represented by the 'cluster' column in the Data Frame.
3. The cluster centers computed by K-means are plotted as 'x' markers in red color



interpretation of this graph:

- The scatter plot displays the data points in a two-dimensional space, where each point represents a combination of the reduced features obtained through PCA.
- Cluster Identification: The data points are color-coded according to the cluster they belong to, as determined by the K-means algorithm. Each color represents a different cluster.
- The red 'X' markers indicate the centroids or center points of each cluster, calculated by K-means.