

STROKE PATIENT HEALTHCARE USING DEEP LEARNING

An overview of data preprocessing, visualization, and modeling.

By Athira Viju



Contents

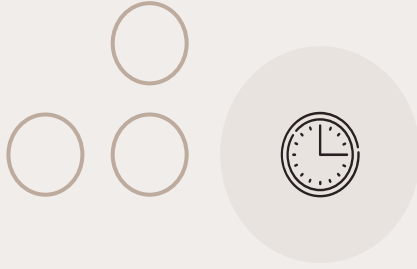
<u>Objective</u>	A summary of the project goals and aim
<u>Project Overview</u>	Define the outline of the project
<u>Milestone 1</u>	Details the initial steps, such as data collection, cleaning, and preprocessing.
<u>Milestone 2</u>	Covers data visualization used to understand patterns and trends.
<u>Milestone 3</u>	Covers encoding techniques
<u>Milestone 4</u>	Focuses on tuning, optimizing, and finalizing model performance metrics.
<u>Dataset Concusion</u>	Summarizes insights drawn from the dataset analysis
<u>Model Final Evaluation</u>	Highlights the final model's performance and weakness based on key metrics
<u>Final Insights</u>	Presents the overarching conclusions,
<u>Stroke Prediction WebApp</u>	Building a Stroke Prediction Model with Streamlit



Objective of the project:

The central aim of this project is to conduct a comprehensive analysis of healthcare data to uncover significant patterns and develop a high-accuracy predictive model to assess the likelihood of health conditions, such as stroke. By leveraging robust data exploration, insightful visualizations, and machine learning techniques, this project aims to derive actionable insights and reliable predictions.





Our aim

To analyze the dataset and build a machine learning model that predicts stroke occurrence based on patient health data.



The goal

To derive meaningful insights from the data through preprocessing and visualization, and to evaluate the performance of various models to identify the most effective one for accurate predictions.



Project Overview

The project focuses on analyzing healthcare data to uncover patterns and build a high-accuracy predictive model for assessing stroke risk. It encompasses four milestones that systematically progress from data exploration to model development.



01

Data Preparation

02

Data Visualization

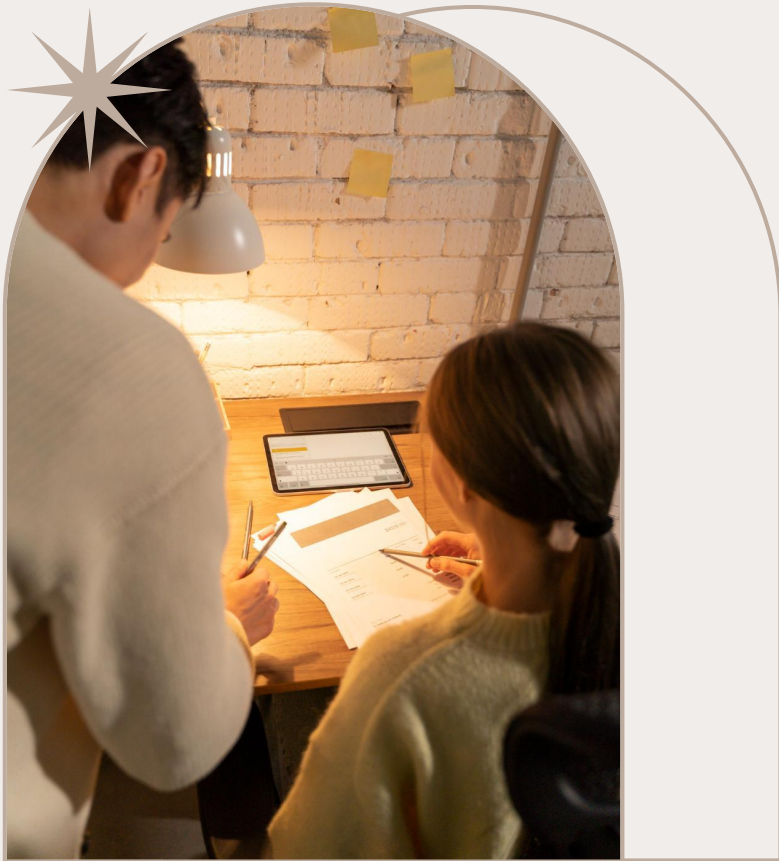
03

Data Encoding

04

Machine Learning Models





01


Data Preparation

Dataset Characteristics

- ★ The dataset consists of 5110 entries across 12 attributes, incorporating both numerical and categorical features.
- ★ Key columns include age, gender, hypertension, heart_disease, work_type, avg_glucose_level, bmi, and the target variable stroke.

Exploratory Analysis:

Analysis	Definition	Observation
<code>df.shape()</code>	Returns a tuple representing the number of rows and columns in the dataset.	The dataset contains 5110 rows and 12 columns, indicating sufficient data for analysis and modeling.
<code>df.info()</code>	Displays a concise summary of the DataFrame, including non-null counts and data types.	Identified bmi as the only column with missing values. Other columns are complete, with data types categorized as int, float, or object.



<code>df.describe()</code>	Provides statistical summary (mean, std, min, max, etc.) for numerical columns.	Key numerical features like age, avg_glucose_level, and bmi were analyzed. The values show a wide range, with notable outliers in glucose levels and BMI.
<code>df.describe(include=object)</code>	Provides statistical summary for categorical columns (e.g., count, unique values, top category, frequency).	The dataset has higher representation of females (2,994 occurrences) and married individuals (3,353 occurrences). Private employment is the most common work type, urban areas are slightly more prevalent (2,596 occurrences), and the majority of participants have never smoked (1,892 occurrences).
<code>df.smoking_status.unique()</code>	Displays unique values in the smoking_status column.	The unique values are: ['formerly smoked', 'never smoked', 'smokes', 'Unknown']. These categories will need encoding for model compatibility.
<code>df.isnull().sum()</code>	Counts the number of missing values for each column.	Only bmi has 201 missing values. These were handled by imputing the median, ensuring no missing data remained.



(Q) Is it good to have null values in dataset ?

No, null values can hinder data analysis and model performance. They represent missing or incomplete information, which can lead to inaccuracies, biases, or errors if not properly handled. Addressing null values ensures the dataset is clean and reliable for analysis or model training.

Handling Null Values

```
df['bmi'] = df['bmi'].fillna(df['bmi'].median())
```

```
df.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         0
smoking_status 0
stroke      0
dtype: int64
```

```
df.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
```



02

Data Visualization

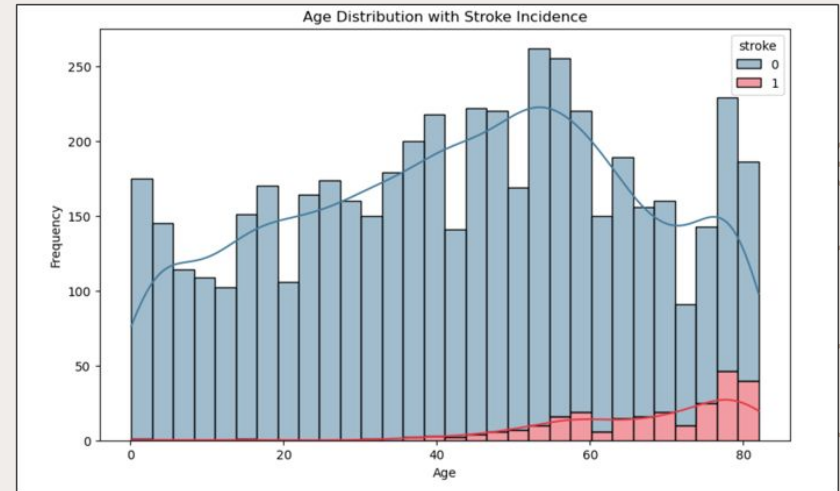
This part focuses on exploring the dataset visually to uncover meaningful patterns and relationships between variables. Through graphs and plots, we analyze trends and distributions, helping identify key insights related to stroke prediction. Additionally, data encoding ensures that categorical variables are transformed into a format suitable for machine learning models.

Age Distribution with Stroke Incidence

Visualization: A histogram with KDE overlay showing the distribution of age with stroke incidence.

Observation:

- ★ Stroke incidence increases significantly in older age groups.
- ★ Younger individuals have a relatively low frequency of strokes.

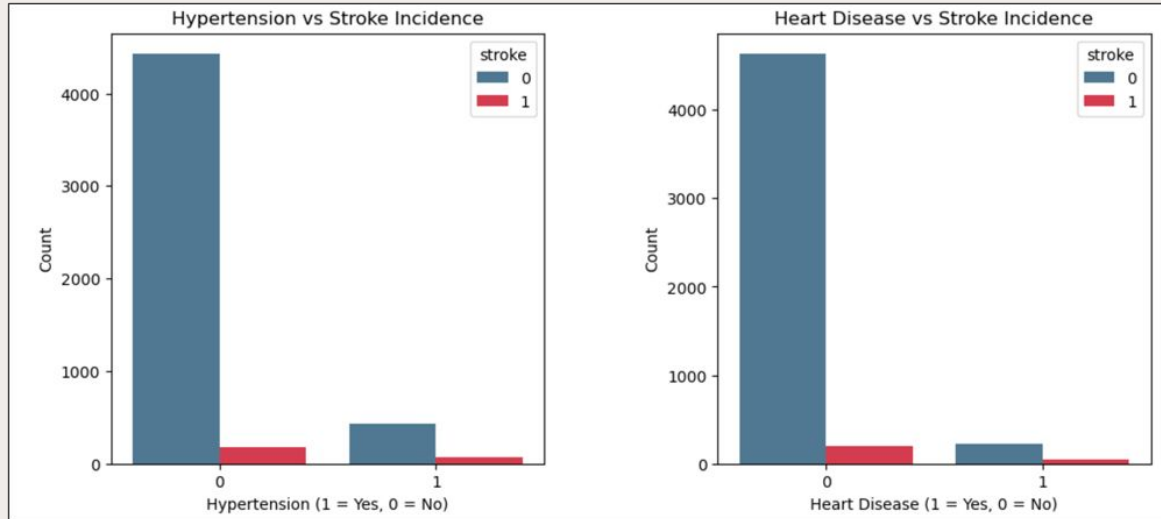


Hypertension and Heart Disease vs Stroke Incidence

Visualization: Two side-by-side bar plots depicting the correlation of hypertension and heart disease with stroke incidence.

Observation:

- ★ Hypertension and heart disease are strong predictors of stroke.
- ★ Individuals with either condition have a higher likelihood of experiencing a stroke.

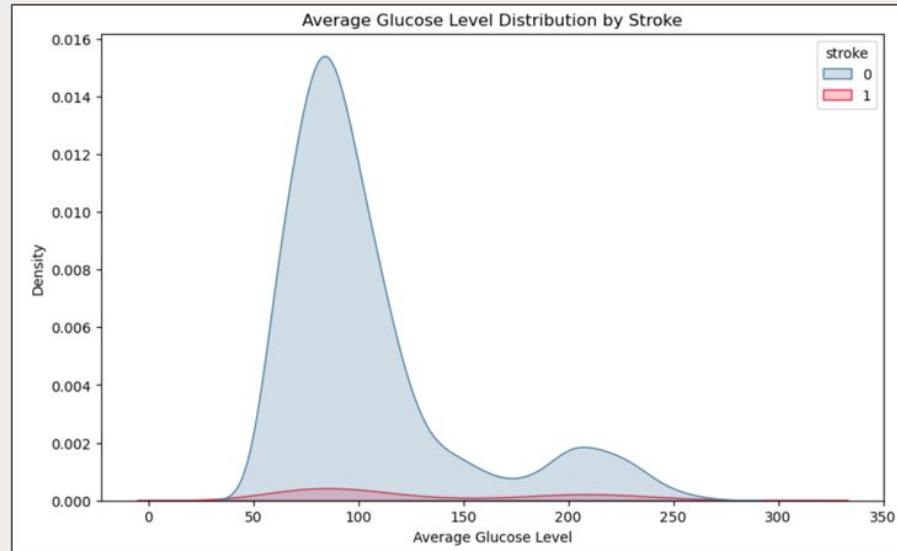


Average Glucose Level Distribution by Stroke

Visualization: A KDE plot illustrating the distribution of average glucose levels for stroke and non-stroke cases.

Observation:

- ★ Higher average glucose levels are associated with increased stroke incidence
- ★ A clear separation in glucose levels exists between stroke and non-stroke groups.

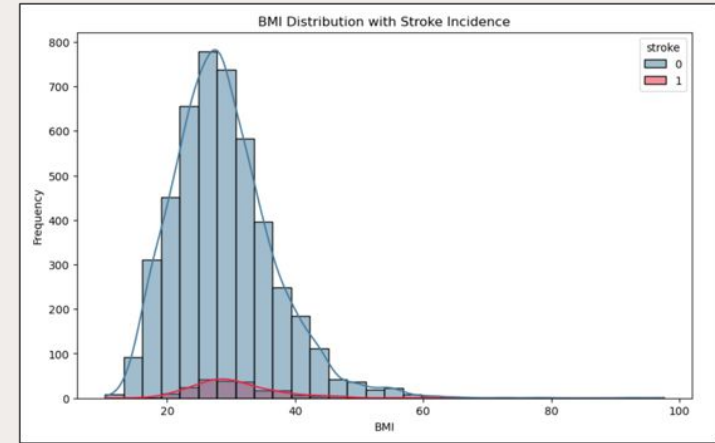


BMI Distribution with Stroke Incidence

Visualization: A histogram with KDE overlay depicting BMI distribution for stroke and non-stroke cases.

Observation:

- ★ Stroke cases are slightly more frequent among individuals with higher BMI.
- ★ BMI alone does not show a strong correlation with stroke.

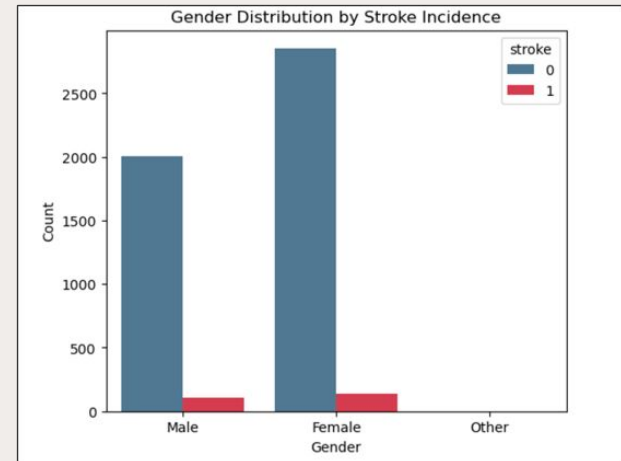


Gender Distribution by Stroke Incidence

Visualization: A count plot showing the distribution of stroke cases by gender.

Observation:

- ★ Both genders show comparable stroke incidence rates, though slight variations exist.

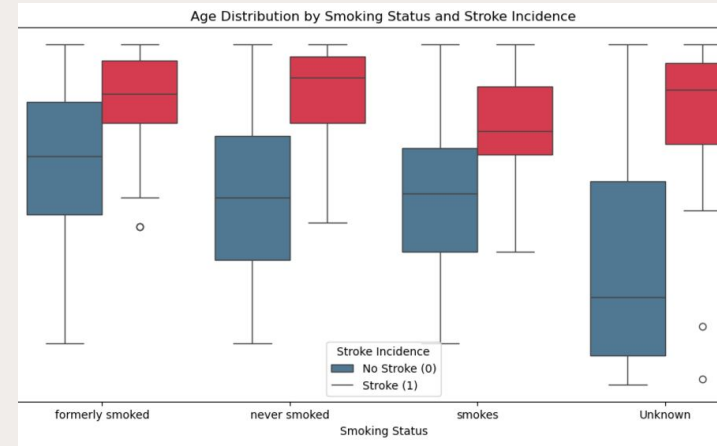


Age Distribution by Smoking Status and Stroke Incidence

Visualization: A boxplot showing age distribution across different smoking statuses with stroke incidence.

Observation:

- ★ Stroke is more prevalent among older individuals, regardless of smoking status.
- ★ Smoking does not directly correlate to stroke in younger age groups but shows an impact in older populations.

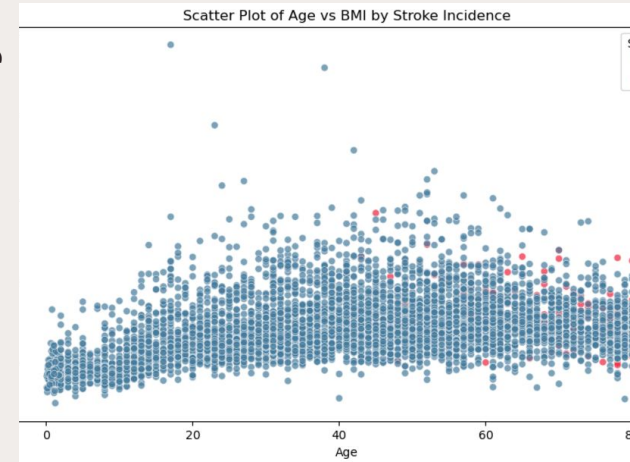


Scatter Plot of Age vs BMI by Stroke Incidence

Visualization: A scatter plot showing the relationship between age, BMI, and stroke incidence.

Observation:

- ★ Most stroke cases cluster in older individuals with a range of BMI values.
- ★ No distinct pattern emerges in BMI but age is a significant factor.



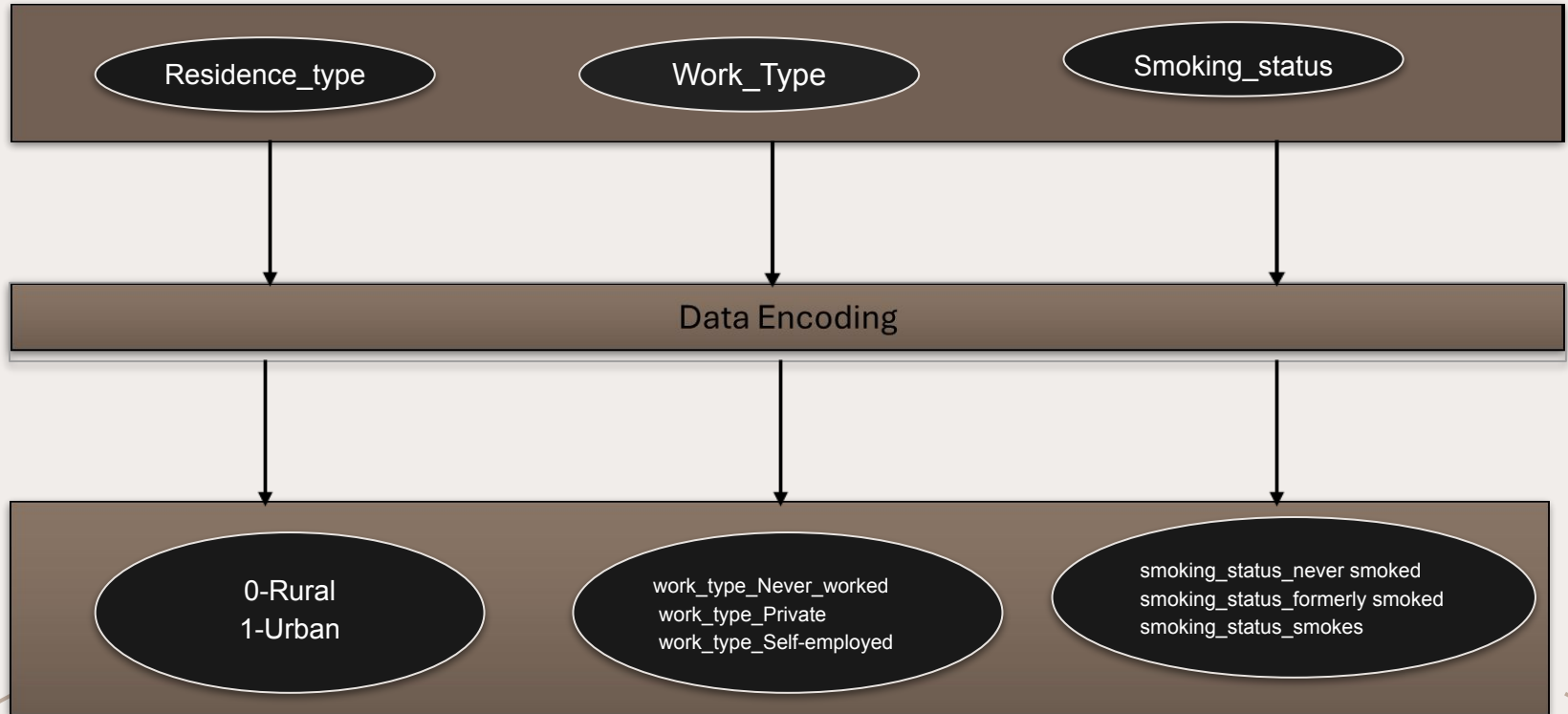


03

Data Encoding

Data Encoding


Data encoding is the process of converting categorical data into numerical formats so that machine learning algorithms can interpret it.






04

Machine Learning Models



Machine Learning (ML) models are algorithms that enable computers to learn patterns from data and make predictions or decisions without being explicitly programmed. Common models are :



Linear Regression

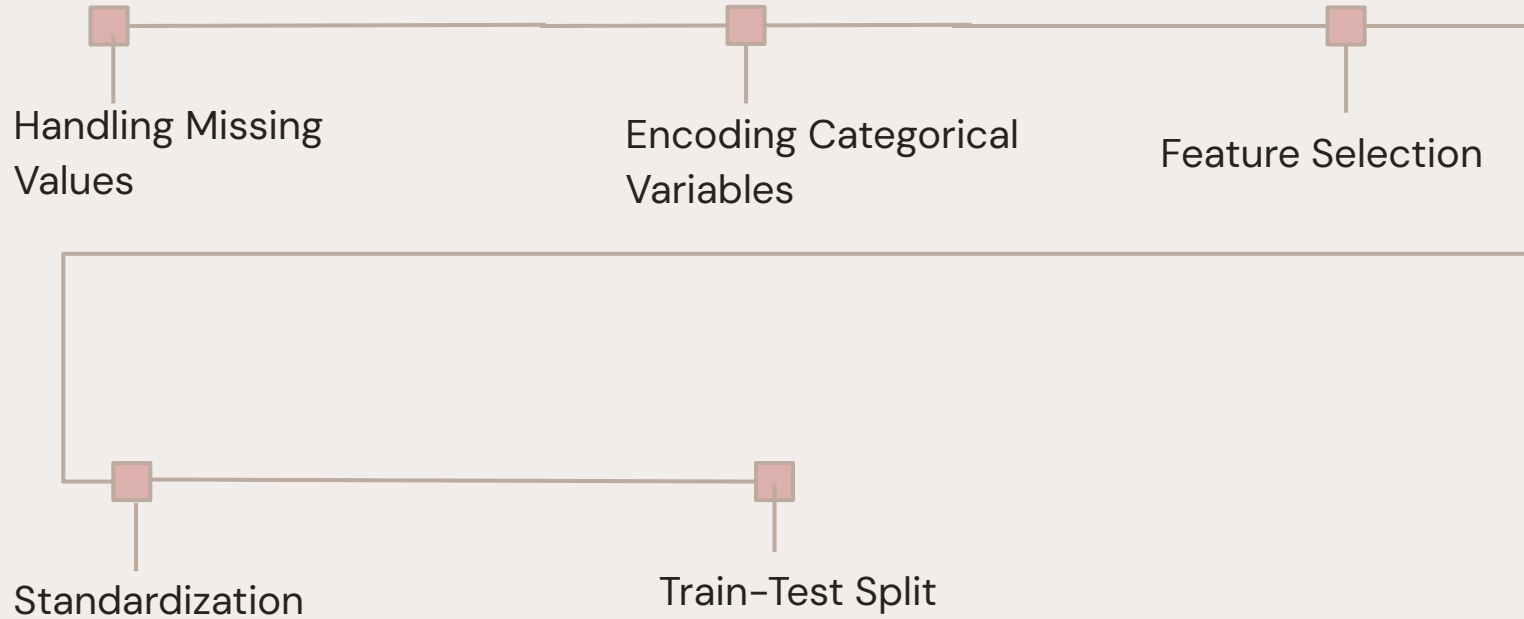
Ridge Regression

Logistic Regression

Lasso Regression



Preprocessing and Data Preparation for Regression Models



Maths Behind Ridge Regression Model

In a Ridge Regression model, the relationship between the dependent variable y and the independent variables $x_1, x_2, x_3, \dots, x_p$ is also assumed to be linear, similar to linear regression. However, Ridge Regression introduces a regularization term to prevent overfitting by penalizing large coefficients, which helps improve the model's generalization.

The model can be expressed mathematically as:

$$\min_{\beta} \left\{ \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$

- y_i : Actual value of the target variable for the i th observation
- \hat{y}_i : Predicted value for the i th observation
- n : Number of observations
- p : Number of features
- β_j : Coefficient for each feature
- $\sum (y_i - \hat{y}_i)^2$: Residual sum of squares
- $\sum \beta_j^2$: Penalty term



Maths Behind Ridge Regression

Root Mean Squared Error (RMSE): RMSE is a commonly used metric to evaluate the accuracy of a regression model. It measures the average magnitude of the error between predicted values and actual values.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

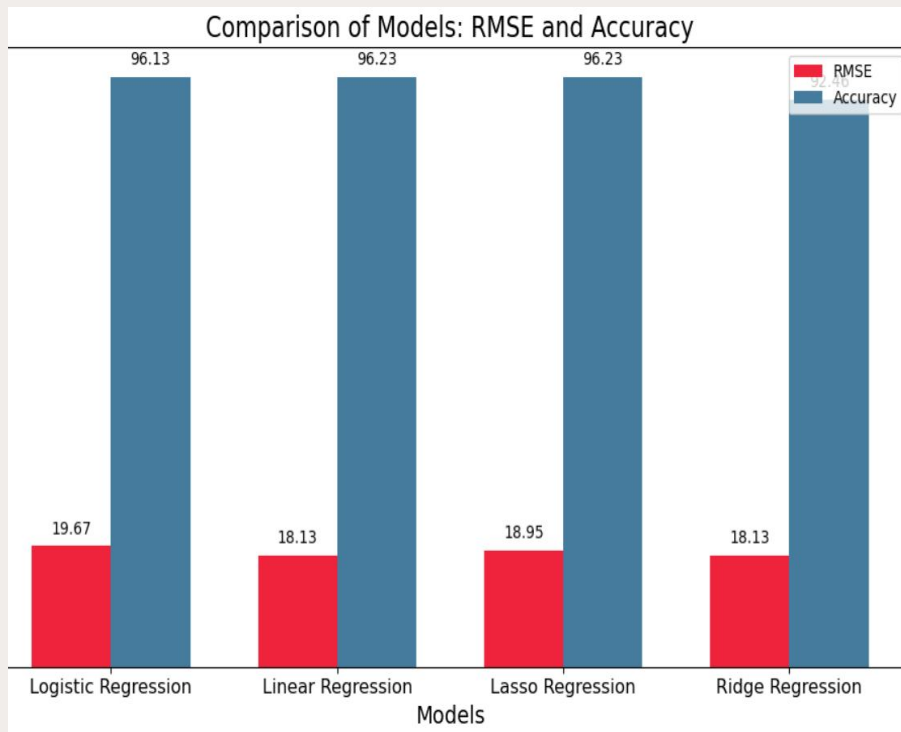
- n : is the number of data points
- y_i : is the actual value for the i -th data point
- \hat{y}_i : is the predicted value for the i -th data point.



Model Performance Summary: RMSE and Accuracy

Model	Accuracy	RSME
Ridge Regression	92.5	18.128
Logistic Regression	96.13	19.67
Linear Regression	96.23	18.12
Lasso Regression	96.23	18.95



Visualizing Different Models



Key observations

RMSE Insights: Linear and Ridge models have the lowest RMSE, indicating the most accurate predictions, while Lasso has slightly higher errors, and Logistic Regression performs the worst with the highest RMSE.

Accuracy Insights: All models show high accuracy (above 92%), with Linear and Lasso achieving 96.23%, Ridge at 92.46%, and Logistic Regression slightly behind at 96.13%.



Precision, Recall, F1 Score, and Accuracy Across Models

- ★ **Precision (P):** Measures how many predicted "Stroke" cases are correct.

$$P = TP / [TP + FP]$$

- ★ **Recall (R):** Measures how many actual "Stroke" cases are correctly identified.

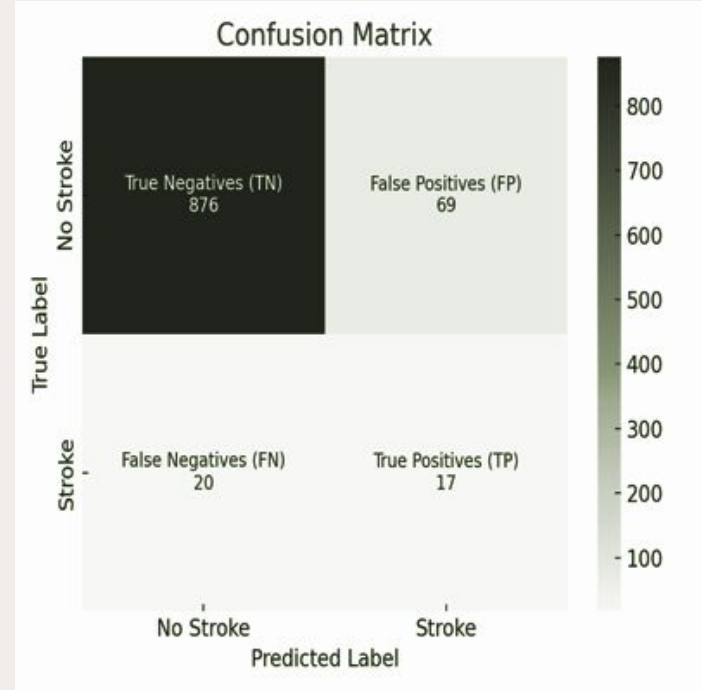
$$R = TP / [TP + FN]$$

- ★ **F1 Score:** Harmonic mean of Precision and Recall, providing a balance between both.

$$F1 \text{ Score} = 2 * (P \times R / P + R)$$

- ★ **Accuracy:** Overall correctness of the predictions.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$



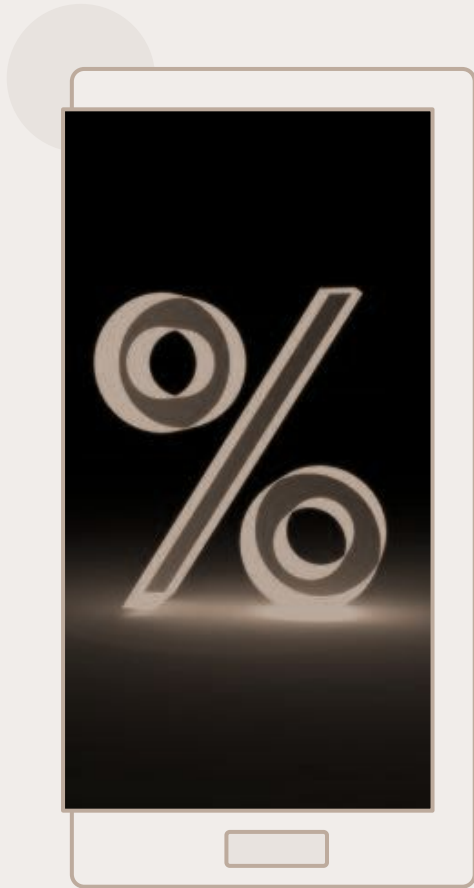
Observations

Model	Precision	Recall	F1 Score	Accuracy
Ridge Regression	0.253	0.513	0.339	0.92
Linear Regression	0.333	0.054	0.093	0.96
Lasso Regression	0.197	0.459	0.276	0.90
Logistic Regression	0.333	0.0270	0.05	0.96

Ridge Regression: Best recall with balanced performance overall.

Linear Regression: High accuracy but poor recall, struggles with the minority class.

Lasso & Logistic Regression: High accuracy but very low recall and F1 scores, indicating poor minority class detection.

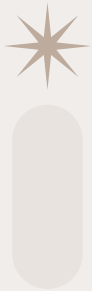


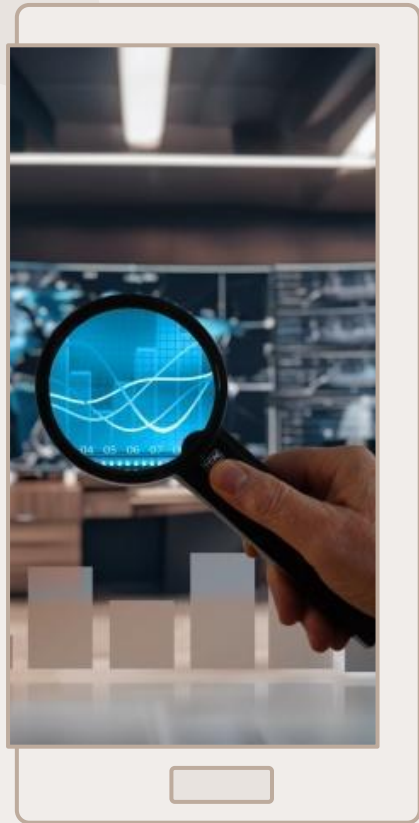
Is accuracy a factor for model performance?

Accuracy alone is not a reliable metric for evaluating model performance, especially in cases of class imbalance. While high accuracy values (e.g., Logistic Regression: 96.13%, Linear Regression: 96.02%) might seem impressive, they can be misleading when the minority class is poorly identified.

For instance:

- Ridge Regression, despite having lower accuracy (92.46%), achieves the highest recall (51.35%), making it more effective at detecting the minority class.
- Logistic Regression has the highest accuracy (96.13%) but extremely low recall (2.7%), failing to identify most minority cases.

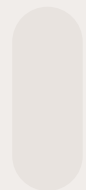




Is the dataset biased?

- **Class 0 (No Stroke):** 95.74%
- **Class 1 (Stroke):** 4.26%

This severe imbalance skews the model toward predicting the majority class, leading to poor detection of the minority class.
- **Steps to mitigate bias:**
 - Changing random state
 - Threshold tuning
 - Adjusting class weights
 - Resampling



Ridge Regression Model Evaluation

Strengths:

- **High Recall (51.35%):** Ridge effectively identifies positive cases (true positives).
- **Good Accuracy (92.46%):** Performs well overall in correct predictions.

Weaknesses:

- **Low Precision (25.33%):** Many false positives, reducing precision.
- **Moderate F1 Score (33.93%):** Indicates an imbalance between precision and recall.

Conclusion:

Ridge Regression excels in recall but requires improvement in precision to balance performance.

Final Insights

Class Imbalance: Dataset is heavily skewed (95% "No Stroke", 5% "Stroke"), causing bias towards the majority class.

Impact on Performance: High accuracy but poor detection of strokes due to the imbalance.

Evaluation Metrics: Metrics like recall and F1-score are more reliable than accuracy for evaluating minority class detection.

Analysis Conducted: Data preprocessing, visualizations, and modeling plays significant role in this project.



Building a Stroke Prediction Model with Streamlit

About

This Streamlit app is designed to predict the likelihood of a stroke based on several health factors using a deep learning model. The app takes input features such as age, gender, hypertension, heart disease, marital status, work type, and smoking status, and provides a prediction on the probability of a stroke. The deep learning model has been trained on a dataset of health records and aims to assist healthcare professionals in identifying individuals at risk of stroke.

The model is based on deep learning algorithms, which analyze patterns in the data to make predictions.



Stroke Prediction Web Application

Leverage Deep Learning to Predict Stroke Risk Based on Key Health Factors

Deploy

Gender

Male

Age

30

Hypertension

No

Heart Disease

No

Ever Married

No

Work Type

Private

Residence Type

Building a Stroke Prediction Model with Streamlit

It displays the prediction after the user provides the input and clicks the predict button.

The image displays two screenshots of a Streamlit application interface for stroke prediction. The interface is dark-themed and features a header image of a person's face.

Top Screenshot (Stroke Detected):

- Prediction:** A Stroke is likely. Please consult a healthcare professional for further evaluation.
- Inputs:**
 - Age: 60.00
 - Body Mass Index: 100.00
 - Smoking Status: never smoked
- Buttons:** Predict (highlighted in red), Reset
- Label:** Stroke Detected

Bottom Screenshot (No Stroke detected):

- Prediction:** No Stroke detected. Keep maintaining a healthy lifestyle!
- Inputs:**
 - Age: 60.00
 - Body Mass Index: 25.00
 - Smoking Status: never smoked
- Buttons:** Predict (highlighted in red), Reset
- Label:** No Stroke

```

from sklearn.feature_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
import numpy as np

# Step 1: Data Preparation
X = df.drop(columns='stroke') # Drop the 'stroke' column
y = df['stroke'].values # Target variable

# Handle missing values (if any)
df = df.dropna() # Drop rows with missing values (if any)
X = df.drop(columns='stroke') # Re-define X after dropping
y = df['stroke'].values # Re-define y after dropping NaNs

# Step 2: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,

# Step 3: Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 4: Define and train the Ridge Regression model
ridge_model = Ridge(alpha=0.5) # Set alpha (regularization)
ridge_model.fit(X_train_scaled, y_train)

# Step 5: Make predictions
y_pred = ridge_model.predict(X_test_scaled)

```

```

for column in df.columns:
    # Check if column is categorical
    if df[column].dtype == 'object':
        # If it has only two unique values, use binary encoding
        if df[column].nunique() == 2:
            # Map the values directly to 0 and 1
            df[column] = df[column].map({df[column].unique()[0]: 1, df[column].unique()[1]: 0})
        # If it has more than two unique values, use one-hot encoding
        else:
            df = pd.get_dummies(df, columns=[column], drop_first=True)

# Convert any boolean columns to integers (True/False to 1/0)
df = df.applymap(lambda x: int(x) if isinstance(x, bool) else x)

# Check the transformed DataFrame
print("\nEncoded DataFrame:")
display(df.head())

/ 0.0s

```

```

# Calculate metrics
precision = precision_score(y_test, y_pred_class)
recall = recall_score(y_test, y_pred_class)
f1 = f1_score(y_test, y_pred_class)
accuracy = accuracy_score(y_test, y_pred_class)

# Print results
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Accuracy:", accuracy)

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred_class)

# Plot confusion matrix

```

Code

The background is a light beige color with several decorative elements. In the top left corner, there are two overlapping circles. To the left of the text, there are two solid light-gray circles. Below the text, there is another solid light-gray circle. To the right of the text, there is a solid light-gray circle with a small, empty square outline positioned just above its top-right corner. Below the text, there is a small, dark brown eight-pointed star. In the bottom right corner, there is a small, empty outline of a rounded rectangle.

Thanks!