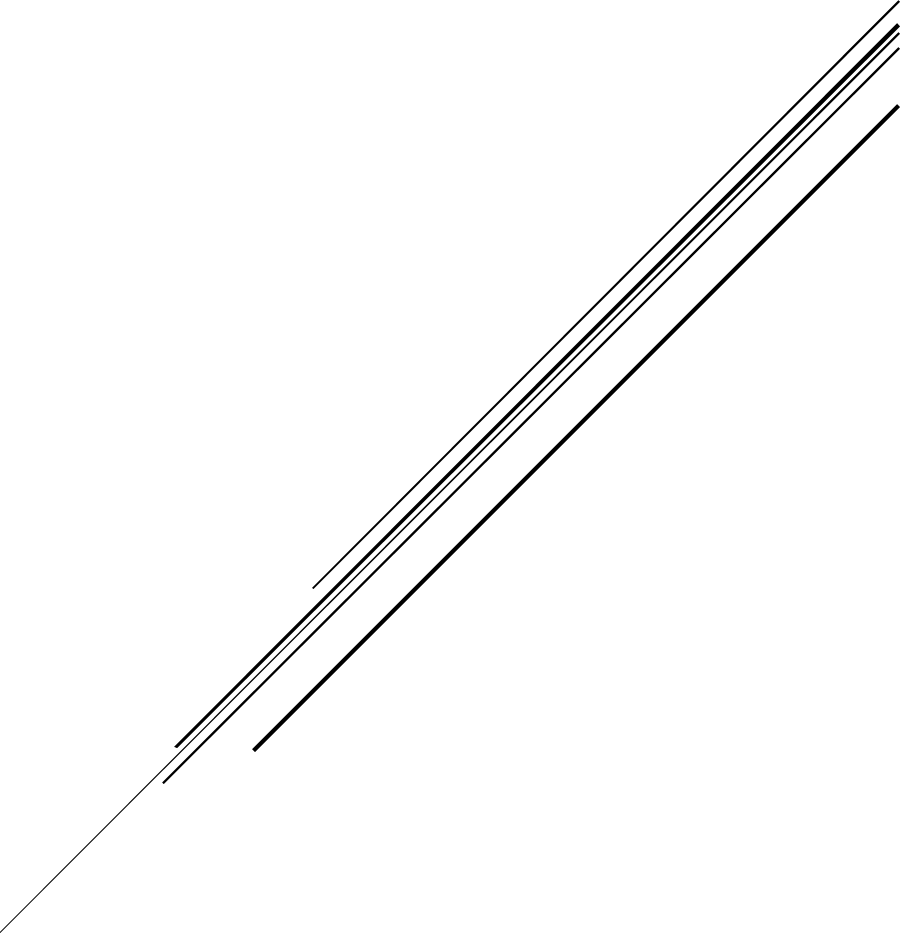
Stroke prediction report

Salma Deiaa Elgezawy

Bioinformatics major

Machine learning and data mining



**Introduction:**

In this project, we focus on leveraging machine learning algorithms to analyze a dataset pertaining to various attributes and risk factors associated with strokes. Our goal is to develop a predictive model that can accurately classify individuals as either having or not having a stroke based on their relevant health indicators.

**Dataset:**

The dataset contains information on various attributes and risk factors related to stroke and it was retrieved from Kaggle website. Each row represents an individual and includes the following features:

* Age: The age of the individual
* Hypertension: Presence of hypertension (0 = No, 1 = Yes)
* Heart disease: Presence of heart disease (0 = No, 1 = Yes)
* Avg glucose level: Average glucose level in the blood
* BMI: Body mass index
* Stroke: Target variable indicating the occurrence of a stroke (0 = No, 1 = Yes)
* Gender (Male, Other): Encoded as binary variables.
* Ever married (Yes): Encoded as binary variable.
* Work type (Never worked, Private, Self-employed, Children): Encoded as binary variables.
* Residence type (Urban): Encoded as binary variable.
* Smoking status (Formerly smoked, never smoked, Smokes): Encoded as binary variables.

**Data Info:**

1. The **df.dtypes** function was used to know the data types of the dataset. The output reveals that the dataset has 4 integers, 3 floats, and 5 objects.
2. The **`df.info()`** function was used to provide an overview of the dataset's structure and data types. The output reveals that the dataset consists of 5,110 entries and 12 columns. The data types of the features are either integer, float, or object. The integer columns are `id`, `hypertension`, `heart\_disease`, and `stroke`. The float columns are `age`, `avg\_glucose\_level`, and `bmi`. The object (categorical) columns are `gender`, `ever\_married`, `work\_type`, `Residence\_type`, and `smoking\_status`. Notably, the `bmi` column has some missing values, with 4,909 non-null entries out of the total 5,110. This information is crucial for understanding the nature of the data and selecting appropriate data preprocessing and modeling techniques.
3. The dataset was examined for missing values using the `**isnull().sum()**` function. The result showed that there are no missing values in most of the columns except for the `bmi` column, which has 201 missing values. This indicates that while most of the dataset is complete and ready for further analysis and modeling, the `bmi` column will require data imputation or other handling methods to address the missing values before proceeding with the analysis.

**Data Cleaning:**

1. The dataset was preprocessed by replacing "N/A" values with `**pd.NA**` and using mean imputation for missing values in the `bmi` column. Categorical variables were encoded using one-hot encoding, and the original categorical columns were dropped from the dataset. This preprocessing ensures the dataset is ready for modeling by handling missing values and converting categorical variables into a format suitable for machine learning algorithms.
2. `**isnull().sum()**` was used again to make sure that there are no missing values left and it showed 0 missing values in all features.
3. The **`df.info()`** function was used again to check the data types after the one hot encoder, and it showed all features are scaled to be floats and integers.
4. Then I dropped the 'id' column from my DataFrame df along the specified axis (in this case, axis 1, which refers to columns). By setting inplace=True, I've ensured that the change is made directly to the original DataFrame df.
5. I cleaned the dataset by removing rows with invalid age values. I identified rows where the age was less than 1 and dropped those rows from the dataset. This ensures that only valid age entries are retained for further analysis or modeling.

**Exploratory data analysis (EDA):**

1. The **lmplot** from the Seaborn library is utilized to explore the relationship between age and average glucose level in the dataset, while also considering the occurrence of strokes. By examining the scatter plot and the trend line, we can assess whether there's a linear correlation between age and average glucose level. Additionally, the color distinction based on stroke occurrence enables us to evaluate any potential differences in the relationship between the two variables among individuals who have had a stroke and those who haven't.
2. Then the **Implot** from Seaborn is used again to see the relationship between age and average glucose levels but this time for only people with strokes.
3. Seaborn **histplot** visualization is used to focus on the distribution of ages among individuals who have experienced a stroke. By examining the histogram and the kernel density estimation curve, we gain insights into the age profile of individuals who have suffered a stroke. Understanding the distribution of ages post-stroke is crucial for identifying age-related risk factors and tailoring intervention strategies accordingly.
4. Seaborn **histplot** visualization examines the distribution of average glucose levels within the dataset, supplemented by a kernel density estimation curve for a smooth representation of the distribution.
5. The distribution of glucose level categories is explored with respect to stroke occurrence. Glucose levels are categorized into 'low', 'normal', and 'high' based on quartile ranges, and each category's distribution is depicted using a count plot.
6. Seaborn **lmplot** visualization is then used to explore the relationship between body mass index (BMI) and average glucose level among individuals who have experienced a stroke.
7. Seaborn **lmplot** visualization is used to examine the relationship between body mass index (BMI) and age among individuals who have experienced a stroke. This visualization contributes to a deeper understanding of the relationship between BMI and age among individuals’ post-stroke, offering insights that may inform personalized interventions and treatment strategies.
8. Seaborn **boxplot** visualization is made to provide insights into the distribution of age, average glucose level, and body mass index (BMI) within the dataset. By examining the box plots, insights can be gained into the central tendency, spread, and presence of outliers for each variable.
9. The **MinMaxScaler** from Scikit-learn is applied to scale the numerical features 'age', 'avg\_glucose\_level', and 'bmi' to a common range, typically between 0 and 1. Scaling the features ensures that they are on a similar scale, preventing features with larger magnitudes from dominating the analysis.
10. A correlation matrix is computed for numerical features in the DataFrame **df**, and a heatmap visualization is generated to depict the strength and direction of correlations between these features.

**Model building and Evaluating:**

1. The "Categorize\_glucose" column was removed from the data frame.
2. We split the data into features (**X**) and the target variable (**y**). The features (**X**) are all the columns in the dataset except for the 'stroke' column, which is our target variable (**y**). This separation allows us to train our machine learning model on the features to predict the target variable.
3. The dataset is divided into two subsets: training and testing sets. The training set (denoted as **X\_train** and **y\_train**) is used to train the machine learning model, allowing it to learn patterns and relationships within the data. The testing set (**X\_test** and **y\_test**) is kept separate and used to evaluate the model's performance and generalization ability on unseen data. The **test\_size** parameter, set to 0.30, specifies that 30% of the data will be allocated to the testing set, while the remaining 70% will be used for training. The **random\_state** parameter ensures reproducibility by fixing the random seed used for shuffling the data before splitting.
4. Feature standardization is performed using the **StandardScaler** from the scikit-learn library. Standardization is a preprocessing technique that rescales the features to have a mean of 0 and a standard deviation of 1, making the features more comparable and ensuring that they have similar scales.
5. The implementation and evaluation of a logistic regression model for binary classification tasks using scikit-learn is made. Beginning with model training on the training dataset, the classifier is then used to predict outcomes on unseen test data. Evaluation metrics including precision, recall, and F1-score are computed via a classification report, offering insights into the model's predictive performance. Additionally, a confusion matrix is generated to visualize the distribution of true positive, true negative, false positive, and false negative predictions.
6. Random Forest classifier for binary classification tasks is made. Following model initialization and training on the provided dataset, the classifier is employed to predict outcomes on unseen test data. Evaluation metrics, including precision, recall, and F1-score, are then computed through a classification report, offering a comprehensive understanding of the model's predictive performance. Furthermore, a confusion matrix is generated and visually represented, facilitating the interpretation of true positive, true negative, false positive, and false negative predictions.
7. Support Vector Machine (SVM) classifier for binary classification using scikit-learn is made. The SVM model, trained with a linear kernel, is used to predict outcomes on test data. Key evaluation metrics, including precision, recall, and F1-score, are computed, and presented in a classification report, offering a detailed view of the model's performance. Additionally, a confusion matrix is generated to visualize the distribution of true and false predictions.
8. The trained machine learning models are evaluated solely based on their accuracy scores. Each model's accuracy in correctly classifying the test data is reported, providing a simplified overview of their performance.
9. After collecting the accuracy scores, a horizontal bar plot is generated using Matplotlib to visually compare the accuracies of different models. Each bar represents the accuracy of a specific model. We then can indicate that the logistic regression model and the SVM model have the same accuracy of 0.94609.

**Model deployment:**

1. The **pickle.dump()** function is utilized to serialize the trained models object and write it to a binary file named 'healthcare-dataset-stroke-data.sav'. Serializing the model in this manner allows for easy storage and later retrieval. The saved model file can be loaded and used for making predictions on new data without the need to retrain the model each time.
2. Streamlit, a Python library for creating web applications, is imported as **st**.
3. The title of the application is set to 'Stroke Prediction app', providing a clear indication of its purpose.
4. An informational message is displayed using **st.info()**, informing users that the application is for Stroke prediction.
5. A sidebar is added to the application interface using **st.sidebar.header()**, allowing users to select features for prediction or perform other interactive actions.
6. The Streamlit application is extended to include text input fields for users to input various features related to heart disease prediction. These input fields allow users to input their personal health information or clinical test results, which will be used as features for Stroke prediction.
7. A Pandas DataFrame is constructed using the **pd.DataFrame()** function, where the data consists of user-inputted values for various features related to Stroke prediction.
8. The **pickle.load()** function is used to deserialize the trained model object from the specified file path 'D:\Education\university\6 term\machine learning\healthcare-dataset-stroke-data.sav'. The model is loaded in binary mode (**'rb'**).
9. A button named 'confirm' is added to the sidebar using Streamlit's **st.sidebar.button()** function. This button serves as a trigger for initiating the prediction process.
10. When the 'confirm' button is clicked (Con is True), the user-inputted data stored in the DataFrame df is passed to the loaded model's **predict()** function. The predicted result is obtained from the model and displayed using Streamlit's **st.write()** function.

- The link to the explanatory video: [Meeting with Salma Deiaa Elgezawy-20240531\_152858-Meeting Recording.mp4](https://nileuniversity-my.sharepoint.com/:v:/g/personal/s_deiaa2110_nu_edu_eg/EY4fxK5ElixIiUaHgOYPytYBRMONLKpStVPnqqcmBdL2FQ?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1XZWJBcHAiLCJyZWZlcnJhbFZpZXciOiJTaGFyZURpYWxvZy1MaW5rIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXcifX0%3D&e=zNCRJe)