CardioPredict

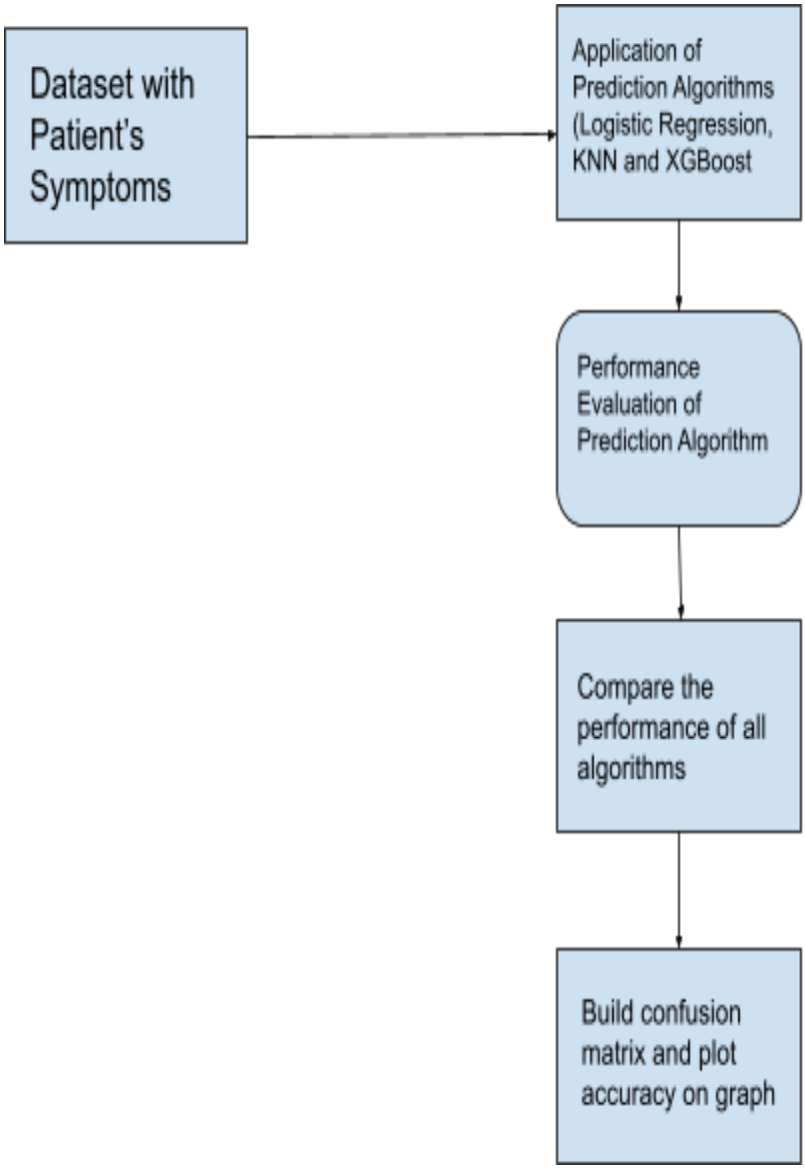
Aumkar Anant Ringe | Dharani Chandra | Meghana Sharath | Viraj Sanjay Kenekar

Abstract:

Considering cardiovascular diseases (CVDs) being a major cause of death worldwide, the objective of this project is to rely on machine learning techniques to develop a model for prediction for evaluating the risk of heart strokes.The project's dataset include individual clinical measurements, medical records, lifestyle factors, and demographic data. Preprocessing techniques were used to encode categorical variables and handle missing values in order to prepare the data for analysis. This guarantees that the format of the data is suitable for machine learning techniques.To further understand the dataset, exploratory data analysis (EDA) was then carried out. EDA involves evaluating the correlations and distribution of various features. Heatmaps, scatter plots, and histograms were a few of the visualizations used for examining the data and look for any trends or insights.Several machine learning methods were used to create predictive models. The dataset was used to train XGBoost, K-Nearest Neighbors (KNN), and logistic regression classifiers. The models were evaluated based on how accurately they predicted cardiac attacks. We may evaluate each model's performance to figure out which one is best at identifying individuals who are at risk. Furthermore, to address the issue of class imbalance in the dataset, oversampling techniques were employed. By focusing on the minority class representing individuals with heart strokes, these

techniques ensure that the models receive sufficient exposure to positive cases during training. This helps improve the models ability to accurately predict heart strokes and reduce bias towards the majority class.

Flowchart


Dataset with Patient's Symptoms
Application of Prediction Algorithms (Logistic Regression, KNN and XGBoost
Performance Evaluation of Prediction Algorithm
Compare the performance of all algorithms
Build confusion matrix and plot accuracy on graph

Objective

The objective of this project is to develop a machine learning model capable of accurately predicting the risk of heart strokes based on individual clinical measurements, medical records, lifestyle factors, and demographic data. Through thorough preprocessing techniques, exploratory data analysis, and the implementation of various machine learning algorithms such as XGBoost, K-Nearest Neighbors (KNN), and logistic regression, the goal is to identify the most effective predictive model. Additionally, addressing class imbalance in the dataset through oversampling techniques ensures a robust and unbiased approach towards identifying individuals at risk of cardiovascular diseases.

Description of Dataset

The dataset titled "Heart Disease Dataset" available on Kaggle contains data related to various attributes of individuals potentially associated with heart disease. The source of the dataset is not explicitly mentioned, but it appears to be a collection of data possibly obtained from medical records, surveys, or studies related to heart disease. It includes several features or attributes for each individual, along with a target variable indicating the presence or absence of heart disease, typically encoded as 0 for absence and 1 for presence. These features may include demographic information, lifestyle factors, and medical indicators such as age, sex, cholesterol levels, blood pressure, presence of symptoms (e.g., chest pain), exercise-induced angina, and

other relevant factors. The dataset is provided in a structured format, such as a CSV file, facilitating easy importation into data analysis tools for further exploration and analysis. It is intended for use in predictive modeling and analysis related to heart disease, allowing researchers, data scientists, and healthcare professionals to develop machine learning models to predict the likelihood of heart disease based on the provided attributes. The size of the dataset in terms of the number of records (individuals) and attributes (features) is not specified in the description, but it likely contains hundreds or thousands of instances with multiple attributes. Users interested in exploring this dataset can access it through the provided link on Kaggle, perform data preprocessing, visualization, and analysis to derive insights or build predictive models related to heart disease.

Implementation

Loading Dataset and Exploration:

The code first loads the dataset and handles missing values by filling them with appropriate strategies such as mean imputation. It also encodes categorical variables using label encoding.

Exploratory Data Analysis:

Several visualizations such as box plots, histograms, scatter plots, and kernel density estimation (KDE) plots are used to explore the relationships between different variables and their distributions. These visualizations help in understanding the data and identifying patterns.

Encoding the Data:

For preparing the data for modeling, variables that are categorical are encoded using Label Encoding.

Data Modeling:

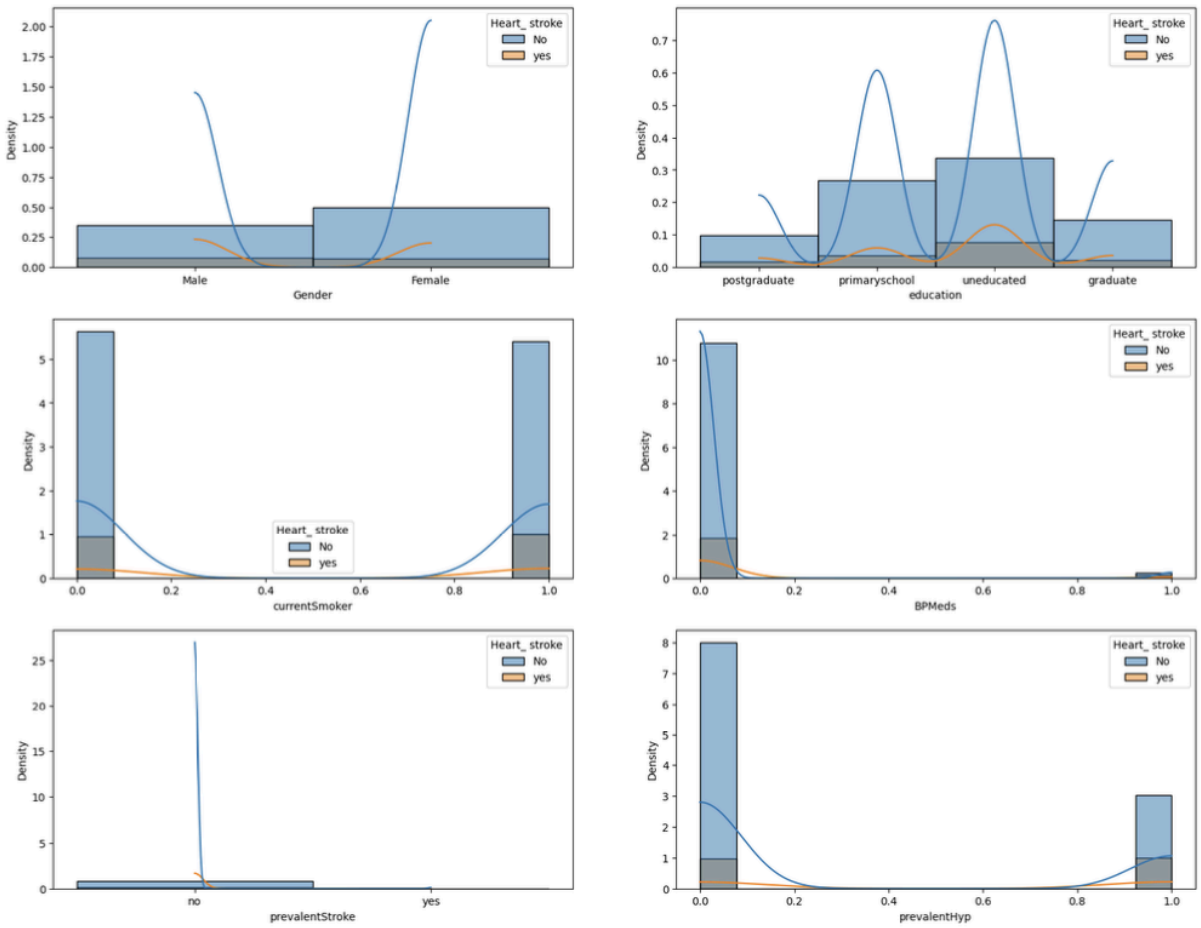
The code then splits the data into training and testing sets and builds machine learning models for heart disease prediction. It uses logistic regression, k-nearest neighbors (KNN), and XGBoost classifiers for this purpose.

Model Evaluation:

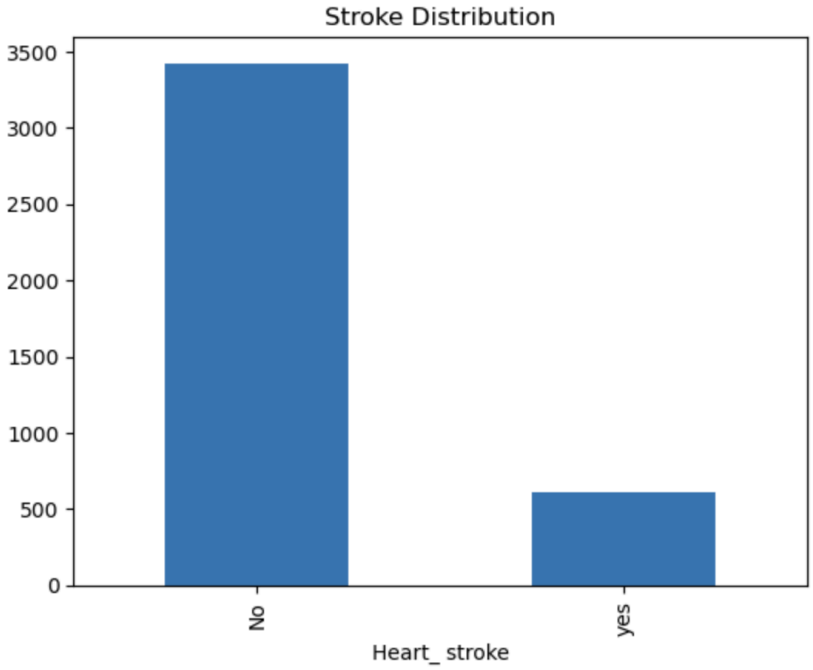
The performance of the models is evaluated using metrics such as accuracy and confusion matrix. Additionally, classification reports are generated to provide insights into precision, recall, and F1-score for each class.

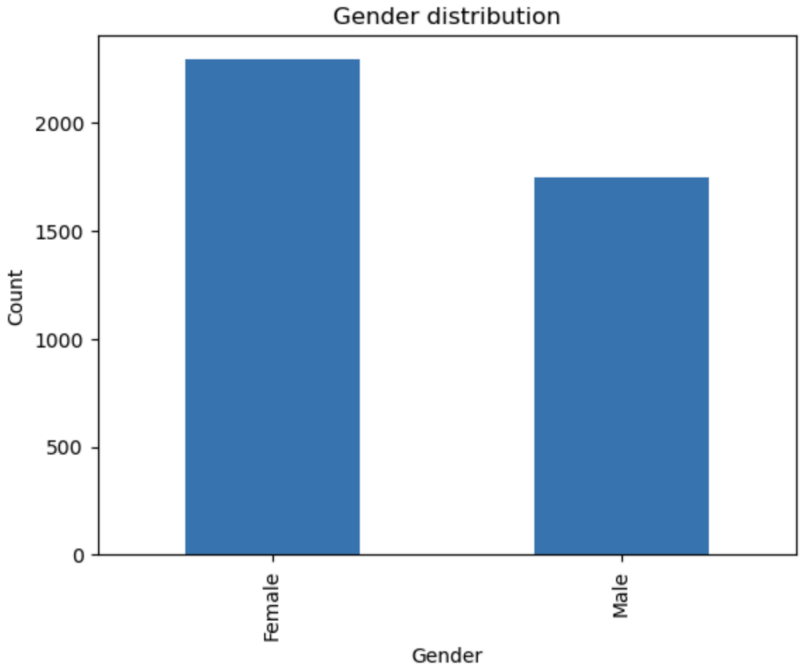
Feature Engineering

To address missing values in the dataset, appropriate values are substituted in place of null values using techniques such as mean, median, or mode for numerical variables (cigsPerDay, BPMeds, totChol, BMI, heartRate, glucose), while categorical variables (education) are assigned a default value. Categorical variables (Gender, education, prevalentStroke, Heart_stroke) are transformed into numerical format suitable for machine learning algorithms using label encoding.



The result will consist of multiple histograms that display the distribution of categorical variables in the dataset. These histograms will be divided based on the target variable 'Heart_ stroke', enabling to visually compare the distributions of each categorical variable across different 'Heart_ stroke' values.





The bar plot depicting the "Stroke Distribution" illustrates the occurrences of heart attacks categorized as "Yes" and "No" within the dataset. This distribution serves as a valuable insight into the prevalence of heart attacks among the sampled individuals. The height of the bars indicates the frequency of responses for each category, showcasing the number of occurrences

for "Yes" and "No" responses. A taller bar for "No" signifies a higher prevalence of individuals who have not experienced a heart attack compared to those who have.
The bar plot illustrates the distribution of genders in the dataset by showing the count for each gender category. It offers valuable information about the gender makeup of the dataset, typically consisting of two categories, possibly denoting males and females. The height of the bars indicates the frequency of each gender category. In comparison to category 1, a taller bar representing category 0 indicates that there are more examples of that gender in the dataset. These distributions help in understanding the balance or imbalance of categories within these variables in the dataset. It can be useful for identifying any potential biases or trends in the data related to gender distribution and occurrence of heart stroke.

Feature Encoding

For transforming categorical data into a numerical representation that works with machine learning algorithms, feature encoding is used. Since many machine learning methods rely on numerical input data, this is required. It utilizes Label Encoding as the specific method for feature encoding. Label Encoding involves assigning a unique numerical label to each category within a categorical variable. The implementation of Label Encoding in the code involves using the LabelEncoder class from the sklearn.preprocessing module. Initially, the 'education', 'Heart_ stroke', and 'Gender' columns are of type object (indicating categorical data). For example, 'education' column originally had categories like 'uneducated', 'primaryschool', 'graduate', and 'postgraduate'. After label encoding, these categories are represented by numerical values (0, 1, 2, 3).

```
<class 'pandas.core.frame.DataFrame'>
Index: 4031 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Gender           4031 non-null   object 
 1   age              4031 non-null   int64  
 2   education        4031 non-null   object 
 3   currentSmoker    4031 non-null   int64  
 4   cigsPerDay       4031 non-null   float64
 5   BPMeds           4031 non-null   float64
 6   prevalentStroke  4031 non-null   object 
 7   prevalentHyp     4031 non-null   int64  
 8   diabetes         4031 non-null   int64  
 9   totChol          4031 non-null   float64
 10  sysBP            4031 non-null   float64
 11  diaBP            4031 non-null   float64
 12  BMI              4031 non-null   float64
 13  heartRate        4031 non-null   float64
 14  glucose          4031 non-null   float64
 15  Heart_ stroke    4031 non-null   object 
dtypes: float64(8), int64(4), object(4)
memory usage: 664.4+ KB
```

```
education
3    1670
2    1226
0     671
1     464
Name: count, dtype: int64
```

```
prevalentStroke
0    4007
1      24
Name: count, dtype: int64
```

```
Gender
0    2287
1    1744
Name: count, dtype: int64
```

Model Development

Three distinct classifiers are trained on the preprocessed data: XGBoost, K-Nearest Neighbors (KNN), and Logistic regression. Using a train-test split, which divides the

data into training and testing sets with a portion of the data set reserve for model evaluation, the trained models are evaluated. Accuracy is used as a performance measure to evaluate each classifier once it has been trained on the training set of data. The testing data is used to make predictions, and accuracy ratings are calculated to determine how accurately each model predicts heart attacks.

Due to the presence of class imbalance in the target variable (heart stroke), where one class (no stroke) is more prevalent in the dataset, oversampling is employed to address this issue. RandomOverSampler from the imbalanced-learn library is utilized to randomly duplicate instances of the minority class (heart stroke) until a balance is achieved between both classes.

After performing oversampling, the XGBoost classifier is retrained using the oversampled data. This is done to assess the performance of the XGBoost classifier after addressing the class imbalance.

Model Evaluation

To evaluate the performance of each model in predicting true positives, false positives, true negatives, and false negatives, confusion matrices are generated. These matrices provide a visual representation of the model's predictive performance.

Additionally, classification reports are generated, which provide various metrics such as precision, recall, and F1-score for each class, along with the overall accuracy. These evaluation metrics are crucial in assessing the effectiveness of each model in predicting heart strokes and gaining insights into their strengths and weaknesses.

Models

Logistic Regression:

Logistic regression is utilized for the purpose of binary classification. The LogisticRegression class is utilized for implementing the logistic regression model, along with optional parameters like max_iter, which indicates the maximum amount of iterations, and random_state, which ensures consistent outputs.

The fit() method is used on the training data (X_train, y_train) to train the model. The model adjusts its coefficients in order to reduce the logistic loss function throughout the training process. The predict() method is then used to make predictions on the test data (X_test), with a frequently utilized probability limit of 0.5.

Since logistic regression is easy to understand, straightforward, and useful for binary classification problems like heart disease prediction, it is used as a baseline model in the code. Important information about the features' impact on the risk of heart disease can be obtained through looking at their coefficients.

K-Nearest Neighbors (KNN):

K-Nearest Neighbors (KNN) classifier is used for the purpose of classification.The KNeighborsClassifier class is used to implement the KNN classifier. To select the distance metric to be used, metric is set, and parameters like n_neighbors are supplied to specify how many neighbors to consider.

Fit() is used to train the model in a way akin to logistic regression using the training data (X_train, y_train). During the prediction process, the model calculates the distance between the new data point and each other point in the training set. It then chooses the largest class among the k-nearest neighbors to get the predicted class for the new data point.

The reason for using the KNN classifier in the code is that it serves as another baseline model, providing a different perspective on classification without making any assumptions about the underlying data distribution. KNN can capture intricate decision boundaries and is particularly useful for understanding local structures in the data.

XGBoost Classifier:

Classification is accomplished by means of the XGBoost classifier. The XGBClassifier class is used for implementing the XGBoost classifier. Here, parameters like learning_rate and n_estimators are set to control the step size reducing and the number of boosting rounds, respectively.

A gradient boosting approach called XGBoost constructs an ensemble of decision trees one at a time. Every tree learns from the mistakes made by the ones before it. The ensemble of decision trees is built during the model's training using the fit() method on the training data (X_train, y_train). The predict() method is then used to make predictions.

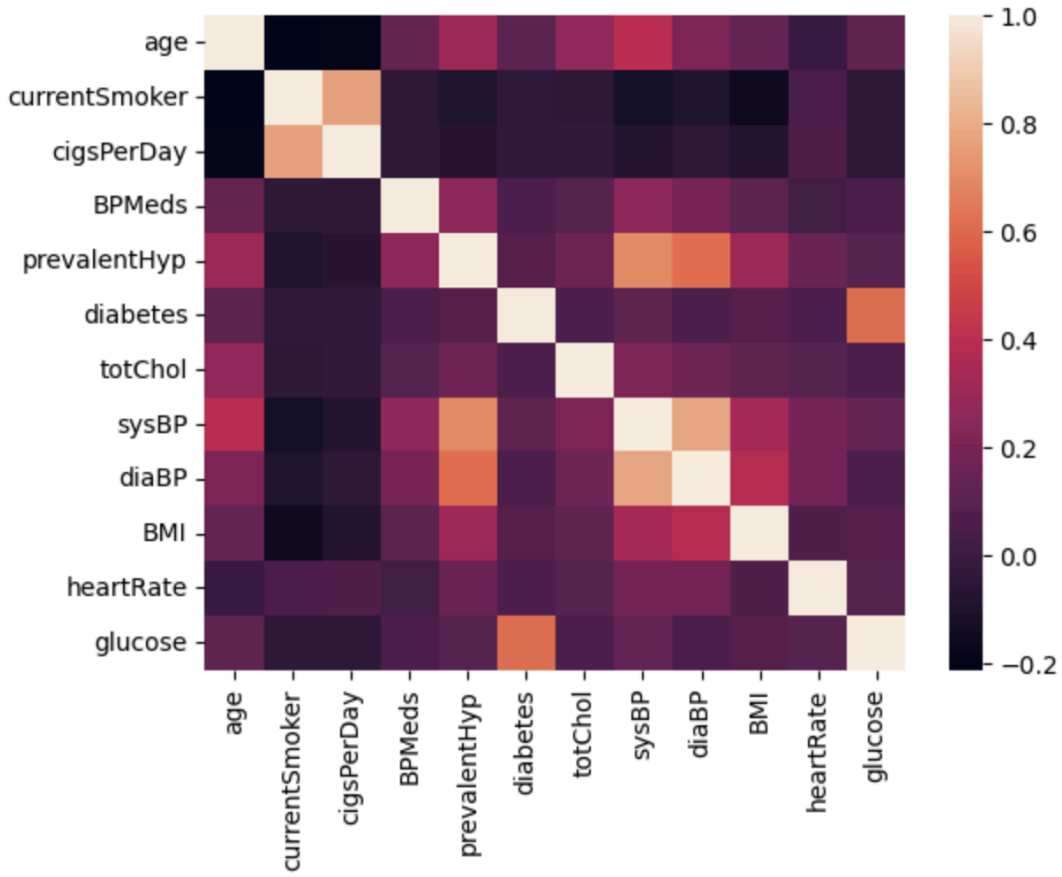
The reason for using the XGBoost classifier in the code is because it offers high performance, efficiency, and the ability to handle complex relationships within the data. XGBoost automatically handles missing values, and it provides feature importance scores, which can aid in feature selection and enhance understanding of the data.

XGBoost Classifier OverSampler:

To solve the class imbalance in the dataset—where positive cases of heart disease are much fewer than negative ones—XGBoost with oversampling has been implemented. Oversampling methods like Random Oversampling are used to achieve this. These methods involve multiplying the number of positive examples, or occurrences of the minority class, inside the dataset. This rebalances the dataset and guarantees that the model sees a suitable amount of positive cases during training.

The XGBoost model is subsequently trained using the resampled data, which consists of the oversampled instances (X_train_resampled, y_train_resampled).Class imbalance might result in biased models that favor the majority class, which is why the code uses oversampling with XGBoost. We guarantee that the model is trained on a more balanced dataset, which improves classification performance and lowers bias, by oversampling the minority class.

Experiment Results



We can identify which factors have stronger correlations with one another by examining the heatmap. For instance, strong correlations between risk factors and the incidence of heart disease, such as cholesterol levels (totChol), systolic blood pressure (sysBP), or diastolic blood pressure (diaBP), indicate that these factors could be important indicators or predictors of heart disease.

Logistic Regression

```
0.828996282527881
```

The result 0.828996282527881 represents the accuracy of the logistic regression model on the testing set. This accuracy score indicates the proportion of correctly predicted instances out of all instances in the testing set. An accuracy of 0.828996282527881 means that approximately 82.90% of the instances in the testing set were correctly classified by the logistic regression model. In summary, the result demonstrates how well the logistic regression model performs in predicting the target variable 'Heart_ stroke' on unseen data, as evaluated on the testing set.

K-Nearest Neighbors (KNN)

```
y_knn_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
knn_accuracy
```

```
0.8302354399008675
```

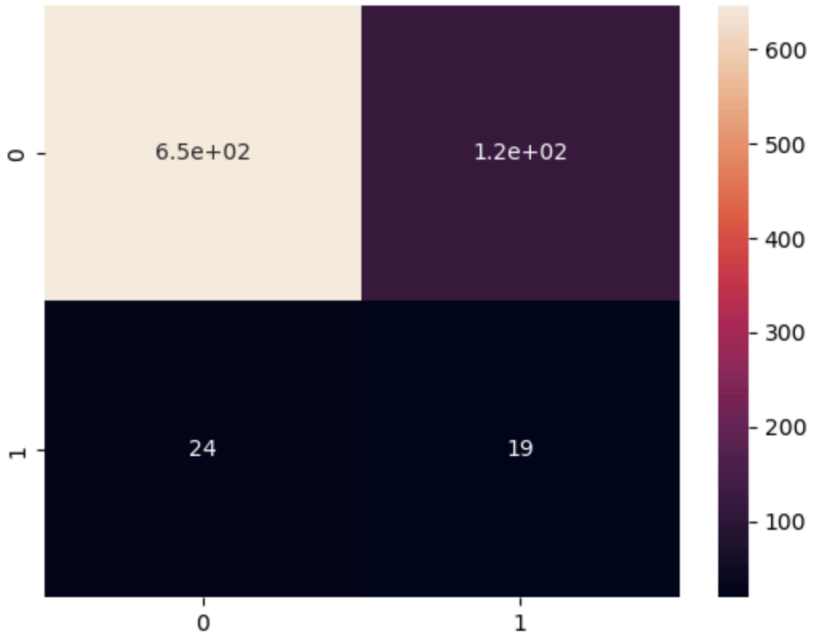
The target variable 'Heart_stroke' is predicted to have values in the y_knn_pred array for the test set using the k-nearest neighbors (KNN) classifier. These forecasts are shown as an array of binary numbers (0 or 1), where 1 denotes the occurrence of a heart attack and 0 usually indicates its absence.

The accuracy score of the KNN classifier on the test set, or about 0.8302354399008675, is stored in the knn_accuracy variable. According to this accuracy score, the KNN classifier correctly identified 83.02% of the test set's instances.

XGBoost Classifer

```
XGBoost's prediction accuracy is: 82.40
```

	precision	recall	f1-score	support
0	0.85	0.96	0.90	670
1	0.44	0.14	0.21	137
accuracy			0.82	807
macro avg	0.64	0.55	0.56	807
weighted avg	0.78	0.82	0.78	807

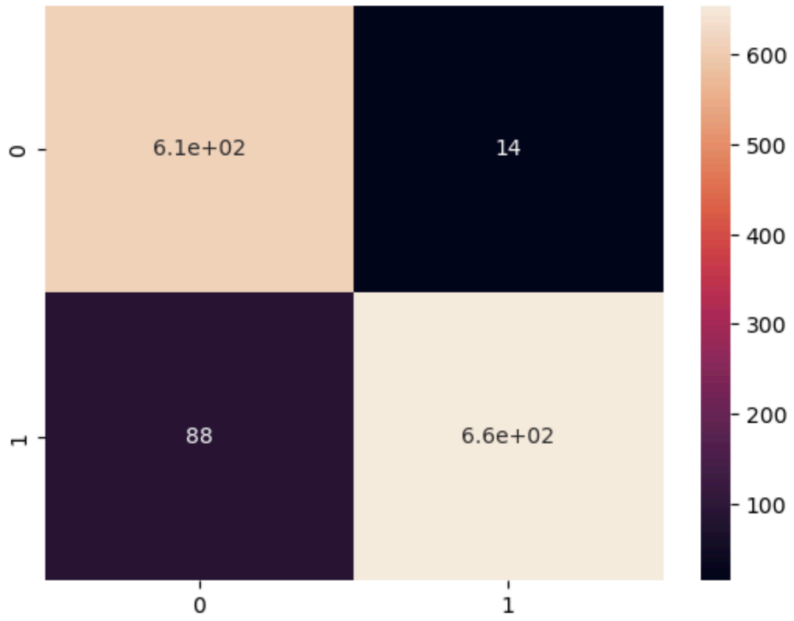


The outcome shows the XGBoost classifier's prediction accuracy. The performance of the classifier in predicting the target variable 'Heart_stroke' on the test set is indicated by this. The accuracy in this instance is roughly 82.40%. This shows that roughly 82.40% of the test set's cases were successfully identified by the XGBoost classifier.

XGBoost Classifier OverSampler

```
XGBoost's prediction accuracy is: 92.55
```

	precision	recall	f1-score	support
0	0.98	0.87	0.92	701
1	0.88	0.98	0.93	669
accuracy			0.93	1370
macro avg	0.93	0.93	0.93	1370
weighted avg	0.93	0.93	0.93	1370



The results demonstrate that after applying oversampling using the RandomOverSampler technique, the XGBoost classifier achieved an accuracy of 92.55%. This improvement indicates a better balance in predicting both classes, with precision, recall, and F1-score all around 0.92. The classification report and confusion matrix further confirm the balanced performance, with high values for both classes.

Overall, oversampling effectively addressed class imbalance, resulting in a more accurate and balanced classifier.

Conclusion

The CardioPredict project effectively tackles the pressing requirement of precise forecasting of cardiovascular illnesses (CVDs), which continue to be the primary cause of death worldwide. By leveraging machine learning techniques, the project aims to develop a predictive model capable of assessing the risk of heart strokes based on various individual factors including clinical measurements, medical history, lifestyle factors, and demographic information.

The project begins with extensive data preprocessing to handle missing values and encode categorical variables, ensuring the data is suitable for analysis. Exploratory data analysis (EDA) is then conducted to understand the relationships between different variables and identify potential patterns or trends within the dataset.

The prediction performance of several machine learning techniques, such as logistic regression, k-nearest neighbors (KNN), and XGBoost classifiers, is evaluated during training. Metrics including accuracy, confusion matrices, and classification reports are used to evaluate the models. To address class imbalance in the dataset, oversampling techniques are also used, guaranteeing that the models are trained on a balanced dataset and reducing bias towards the majority class.

The results indicate promising performance across all models, with accuracies ranging from approximately 82% to 92.55% after addressing class imbalance using oversampling. The XGBoost classifier with oversampling demonstrates the highest

accuracy of 92.55%, indicating significant improvement in predicting both classes with balanced precision, recall, and F1-scores.

Future Work

Here are some things that can be done to improve the model and extend it to real time use -

- **Feature Selection:** Implement feature selection methods to identify the most relevant features for prediction. Techniques like Recursive Feature Elimination (RFE), feature importance from tree-based models, or L1 regularization can help in selecting the most informative features.
- **Model Tuning:** Perform hyperparameter tuning to optimize the performance of the AdaBoost classifier. Grid search or random search techniques can be used to find the best combination of hyperparameters for improved accuracy.
- **Ensemble Methods:** Experiment with other ensemble methods such as Random Forests, Gradient Boosting Machines (GBM), or XGBoost. These methods often provide better predictive performance and can capture more complex patterns in the data.

Individual Work:

Viraj Sanjay Kenekar - XGBoost
Meghana Sharath - Logistic Regression
Dharani Chandra - KNN
Aumkar Anant Ringe - Oversampling

Reference:

XGBoost Classifier - (How does this work)
https://medium.com/@prathameshsonawane/xgboost-how-does-this-work-e1cae7c5b6cb#:~:text=XGBoost%20(eXtreme%20Gradient%20Boosting)%20is,machine%20learning%20competitions%20and%20tasks.

XGBoost and imbalanced datasets: Strategies for handling class imbalance
https://medium.com/@rithpansanga/xgboost-and-imbalanced-datasets-strategies-for-handling-class-imbalance-cdd810b3905c

Understanding Logistic Regression
https://medium.com/analytics-vidhya/understanding-logistic-regression-b3c672deac04

K-Nearest Neighbor
https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

Oversampling—Handling Imbalanced Data
https://medium.com/@abdallahashraf90x/oversampling-for-better-machine-learning-with-imbalanced-data-68f9b5ac2696#:~:text=Oversampling%20is%20a%20data%20augmentation,to%20the%20under%2Drepresented%20class.

Git Link:

https://github.com/aumkarringe/ML-Project/blob/main/ML-Project.ipynb