\* **Romani Nasrat Shawqi Gerges**

**\* Ahmed Mohamed Ali Abo El-Kassem**

**\* Zeyad Elsayed Abdel-Azim Ali Yousif**

**\* Sara Reda Moatamed Hassan**

**\* Reham Moustafa Ali Abdel-Moati**

**\* Rawan Abdel-Aziz Ahmed Mahmoud**

**\* Abdallah Mohammed Abdel-monnem**

**\* Mohannad Ayman salah Abdel-fattah**

**\* Mohammed Abdallah Abdel-salam Abdel-Dayem**

Ds Pro



**Problem Statement:**

The goal of this notebook is to develop a robust predictive model capable of accurately classifying individuals based on their risk of heart disease. The classification task is complex due to the intricate interplay of diverse risk factors encompassing lifestyle habits, medical history, and physiological indicators. This problem statement encapsulates the following key points:

**1. Classification Objective:**

- The primary objective is to classify individuals into two categories: those at risk of heart disease and those not at risk.

**2. Predictive Model Development:**

- The focus is on building a predictive model using machine learning techniques to automate the classification process.

**3. Accuracy Emphasis:**

- The emphasis is on achieving high accuracy in classification to ensure reliable identification of potential heart disease patients.

**4. Complexity of Risk Factors:**

- The classification task is complicated by the intricate interplay of various risk factors, including lifestyle habits (e.g., smoking, alcohol consumption), medical history (e.g., previous strokes, diabetes), and physiological indicators (e.g., BMI, physical health).

**5. Data-Driven Approach:**

- The approach involves leveraging available data containing information on diverse risk factors to train the predictive model.

**6. Reliability and Validity:**

**-** The ultimate goal is to develop a classifier that is reliable and valid, capable of making accurate predictions on unseen data.

**7. Practical Implications:**

**-** A successful predictive model can have significant practical implications, enabling timely intervention and personalized healthcare strategies for individuals at risk of heart disease.

**By addressing these aspects, the notebook aims to provide valuable insights and tools for healthcare professionals to identify and manage individuals at risk of heart disease effectively.**

|  |
| --- |
| **1.About Dataset:**  -The dataset originates from the CDC's Behavioral Risk Factor Surveillance System (BRFSS), which collects health status data through annual telephone surveys of U.S. residents.  -This dataset combines responses from the 2020 and 2022 surveys.  -It provides valuable insights into various health-related factors and conditions prevalent among the surveyed population |
| 2. **Size:** The dataset contains 764,927 entries and 18 columns |
| 3. **Attributes:**  - Sex: Gender of the respondent.  - GenHealth: Self-reported general health status.  -PhysicalHealth:Self-reported physical health status.  - MentalHealth:Self-reported mental health status.  - PhysicalActivity: Level of physical activity.  - SleepTime: Average sleep time in hours.  - Stroke: History of stroke.  - Asthma: History of asthma.  - SkinCancer: History of skin cancer.  - Diabetic:History of diabetes.  - BMI:Body Mass Index.  - AlcoholDrinking:Frequency of alcohol consumption.  - Race: Ethnicity or race of the respondent.  - AgeCategory:Age category of the respondent.  -HeartDisease:Presence of heart disease (target variable).  - KidneyDisease: History of kidney disease.  - Smoking: Smoking status.  - DiffWalking: Difficulty walking. |
| **4. Missing Values:**  - The dataset contains missing values in several columns.  - The columns with the highest percentage of missing values are **BMI, AlcoholDrinking, Smoking, DiffWalking, and Race.** |

**5.Objective:**

-The objective of using this dataset is to build a classifier that can accurately predict the presence of heart disease based on the given attributes.

-The classifier aims **to assist in early detection and intervention for individuals at risk of heart disease.**

**1-Exploring Dataset**

**1. Data Loading:**

- Loaded the dataset from a CSV file using pandas.

**2. Initial Inspection:**

- Checked the dimensions of the dataset using `df.shape` to know the number of rows and columns.

- Displayed the first few rows of the dataset using `df.head()` to get an overview of the data.

**3. Data Information:**

- Used `df.info()` to obtain information about the dataset, including column data types and non-null counts.

**4. Missing Values Analysis:**

- Identified missing values in the dataset using `df.isna().sum()` to count the number of null values in each column.

- Calculated the percentage of missing values for each feature to understand the extent of missingness.

1. **Visualization of Missing Values:**

- Plotted the percentage of missing values for each feature using a horizontal bar plot to visualize the distribution of missing data across columns.

**By following these steps, we gain a comprehensive understanding of the dataset's structure, content, and quality, which lays the foundation for further analysis and preprocessing.**

**2-Cleaning Steps:**

|  |  |  |
| --- | --- | --- |
| **1. Removing Missing Values:**  - Removed rows with missing values using `dropna()` method.  - Checked for any remaining missing values using `isna().sum()`.   |  |  | | --- | --- | | IMG_256 | IMG_256 |   **Before After** |
| **2. Removing Duplicates:**  - Checked for duplicate rows using `duplicated().any()`.  - Removed duplicate rows using `drop\_duplicates()`. |
| **3. Standardizing Categorical Attributes:**  - Standardized values in the 'Race' column by mapping certain categories to common names. |
| **4. Standardizing Values:**  - Standardized values in the 'AgeCategory' and 'Diabetic' columns by replacing certain values with their standardized equivalents. |

- After removing missing values and duplicates, there are no missing values or duplicate rows in the dataset.

- The 'Race' column has been standardized, with certain categories renamed for consistency.

- Values in the 'AgeCategory' and 'Diabetic' columns have been standardized for clarity and consistency.

These cleaning steps ensure that the dataset is free from missing values, duplicates, and inconsistencies, making it suitable for further analysis and modeling.

**3-Exploratory data analysis (EDA)**

|  |
| --- |
| **1. Splitting the Data:**  - The dataset is divided into training and testing sets using `train\_test\_split` from `sklearn.model\_selection`.  - The split is stratified based on the 'HeartDisease' column to maintain class distribution.  - Result:  - Training set shape: (501375, 18)  - Testing set shape: (125344, 18)  This step ensures that we have separate datasets for training and testing our model, and the split maintains the distribution of the target variable, which is essential for building a reliable classifier. |
| **2. Visualizing Class Imbalance:**  - A pie chart is plotted to visualize the distribution of the target variable ('HeartDisease').  - The imbalance in the target variable is evident from the chart.  IMG_256 |
| **3. Analyzing Heart Disease by Gender:**  - The count of individuals by gender is visualized using a pie chart and a countplot.  - Result:  IMG_256 |
| **4. Exploring Risk Factors by Gender:**  - Risk factors like physical activity, smoking, and alcohol drinking are visualized by gender using countplots. |
| **5. Analyzing Heart Disease Among Different Age Groups:**  - The age category distribution is visualized using a countplot.  IMG_256 |
| **6. Exploring Risk Factors Among Age Groups:**  - Risk factors' distribution among different age groups is visualized using countplots. |
| **7. Analyzing Diabetes, Smoking, Asthma, and Kidney Disease:**  - The count of individuals with diabetes, smoking habits, asthma, and kidney disease is visualized using countplots.   |  |  | | --- | --- | | IMG_256 | IMG_256 | | IMG_256 |  | |
| **8. Visualizing Numerical Variables:**  - Boxplots are used to visualize the distribution of numerical variables like physical and mental health, sleep time, BMI, etc., with respect to heart disease.  IMG_256  IMG_256 |
| **9. Handling Outliers:**  - Outliers in the 'SleepTime' variable are detected using the IQR method.  - Later, it's decided not to remove outliers as they were considered true values. |

**These steps provide a comprehensive exploration of the dataset, including class distribution, demographic factors, risk factors, and numerical variables, which are crucial for understanding the data before building a classifier.  
=================================================================**

1. **Preprocessing Pipelines**

|  |
| --- |
| **1. Splitting the Data:**  **-** The dataset is split into training, validation, and test sets with proportions of 70%, 10%, and 20%, respectively, using `train\_test\_split` from `sklearn.model\_selection`.  - Stratification is applied based on the 'HeartDisease' column to ensure class distribution consistency.  - Result:  - Training set shape: (438,703, 18)  - Validation set shape: (62,672, 18)  - Test set shape: (125,344, 18)  - Proportions: 70.0% training, 10.0% validation, 20.0% test. |
| **2. Feature Engineering:**  **- Features are separated into numerical and categorical attributes.**  **- An ordinal encoder is implemented to transform categorical variables into ordinal representations.** |
| **3. Handling Skewed Numerical Attributes:**  - Skewness in numerical attributes is addressed using various transformations like logarithmic, square root, and power transformations.  - A power transformer is applied to handle skewness effectively. |
| **4. Pipeline Construction:**  - Separate preprocessing pipelines are built for numerical and ordinal categorical attributes.  - The pipelines include transformation steps such as scaling and ordinal encoding. |
| **5. Data Transformation:**  - Preprocessing pipelines are applied to transform the training, validation, and test sets.  - Result:  - Processed training set shape: (438,703, 16)  - Example of processed feature vector: [-1.02, 1.32, 1.36, -0.80, 0, 2, 1, 0, 0, 1, 0, 1, 10, 0, 2, 0] |
| **6. Setting Baseline Performance:**  - A logistic regression model is trained on the processed training set.  - Predictions are made on the validation set to evaluate baseline performance using metrics such as precision, recall, accuracy, and F1-score.  - Result:  - F1-score: 0.125  - Classification report shows precision, recall, F1-score, and support for each class.  - Accuracy is relatively high, but other metrics indicate poor performance, particularly in identifying instances of heart disease (class 1). |
| **7. Visualizing Confusion Matrix:**  - Confusion matrix is computed using `confusion\_matrix` from `sklearn.metrics`.  - The matrix is normalized to show percentages of true labels.  - A heatmap is plotted to visualize the confusion matrix.  - Result:  - The heatmap illustrates the distribution of true and predicted labels, providing insights into the model's performance across different classes.  - Each cell represents the percentage of instances belonging to a particular true label that were predicted as each class.  IMG_256 |

**Under-sampling** is a technique used to address class imbalance in a dataset, where one class (the majority class) heavily outweighs the other class(es) (the minority class or classes). In such cases, the model may become biased towards the majority class, leading to poor performance in predicting the minority class. Under-sampling aims to balance the class distribution by reducing the number of samples in the majority class.

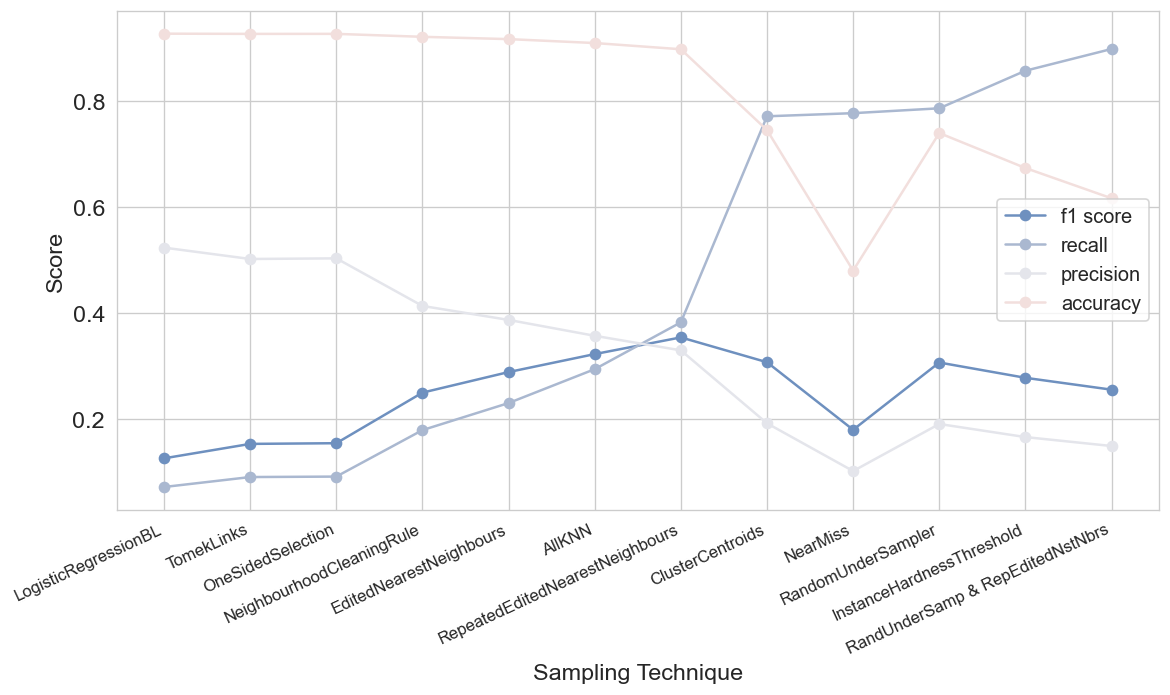
**In our code, under-sampling techniques are applied to address the class imbalance issue in the dataset**

|  |
| --- |
| * ****Generating techniques****   **ClusterCentroids:**  - It clusters the majority class samples and selects centroids as representatives.  - Result: After under-sampling, the dataset is balanced, and the F1 score is 0.307. |

|  |
| --- |
| * ****Selection techniques Controled****   **1. RandomUnderSampler:**  - This technique randomly selects samples from the majority class to balance the dataset.  - Result: After under-sampling, the number of instances in the majority and minority classes becomes more balanced, with an F1 score of 0.306. IMG_256 |
| **2. NearMiss:**  - It selects samples from the majority class that are close to the minority class.  - Result: The dataset is balanced, but the F1 score is relatively low at 0.179.  IMG_256 |

|  |
| --- |
| * ****Selection techniques Cleaning****   + Tomek-Links   + OneSidedSelection   + EditedNearestNeighbours   + RepeatedEditedNearestNeighbours   + AllKNN   + InstanceHardnessThreshold   + NeighbourhoodCleaningRule |
| **TomekLinks:**  - It removes samples from the majority class that are Tomek links with the minority class.  - Result: The dataset is balanced, but the F1 score is relatively low at 0.153. IMG_256 |
| **OneSidedSelection:**  - It selects samples from the majority class by considering Tomek links.  - Result: While the dataset is balanced, the F1 score is low at 0.154. IMG_256 |
| **EditedNearestNeighbours:**  - It removes samples from the majority class that are misclassified by their nearest neighbors.  - Result: The dataset is balanced, with an F1 score of 0.288.  IMG_256 |
| **RepeatedEditedNearestNeighbours:**  - Similar to EditedNearestNeighbours, but repeats the process multiple times.  - Result: The dataset is balanced, with the highest F1 score among the techniques at 0.354.  IMG_256 |
| **AllKNN:**  - It removes samples from the majority class based on the KNN algorithm.  - Result: The dataset is balanced, with an F1 score of 0.322.  IMG_256 |
| **InstanceHardnessThreshold:**  - It removes samples based on their hardness threshold calculated by a classifier.  - Result: The dataset is balanced, but the F1 score is relatively low at 0.277.  IMG_256 |
| **NeighbourhoodCleaningRule:**  - It removes samples based on the majority class that are in the neighborhood of the minority class.  - Result: The dataset is balanced, but the F1 score is relatively low at 0.249.  IMG_256 |

**Overall, RepeatedEditedNearestNeighbours performed the best among the under-sampling techniques, achieving the highest F1 score of 0.354.**



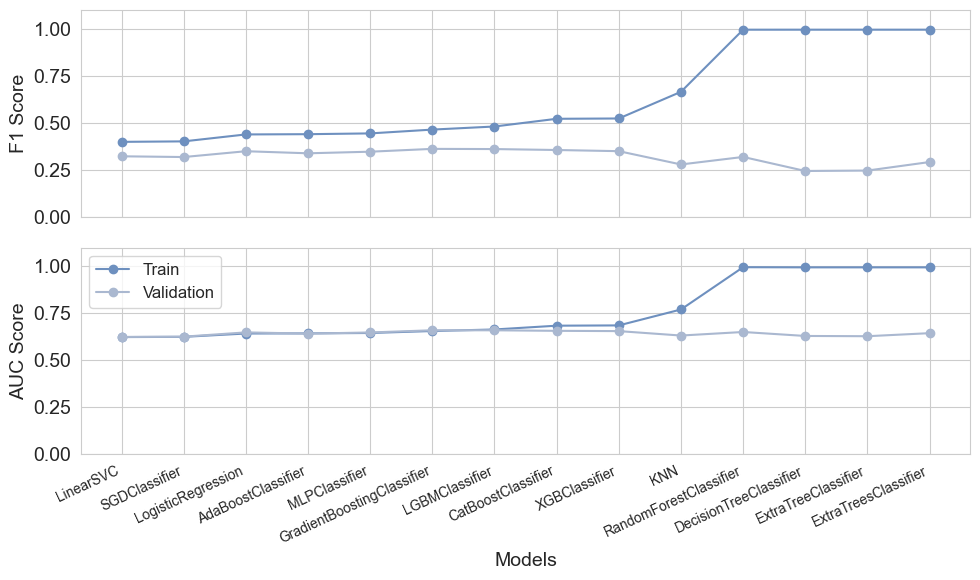
**Result: The dataset is now balanced, with instances reduced from 438,703 to 160,265**

**Model Selection**

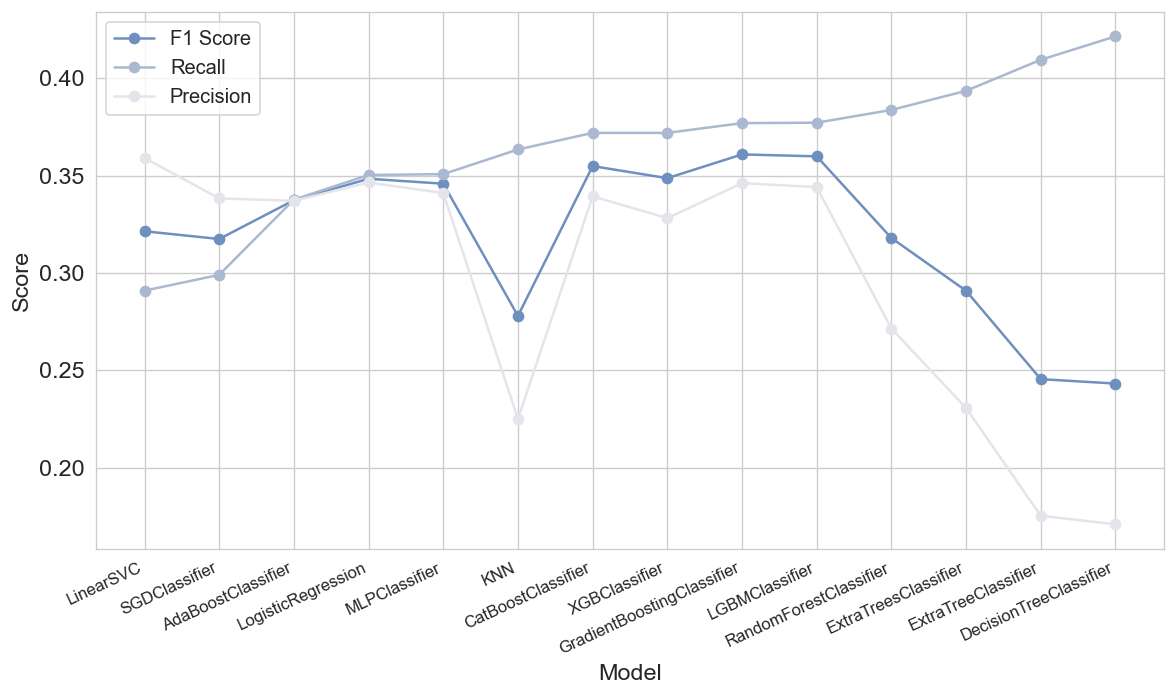
- Thirteen different classifiers are selected from various categories such as KNN, linear models, support vector machines, tree-based models, ensemble models, and others.

- Each classifier is trained on the resampled training data (X\_train\_resampled, y\_train\_resampled).

- Performance metrics including recall, precision, accuracy, F1-score, and AUC score are computed for both the training and validation sets.

- Result: F1 scores and other metrics for each classifier on both the training and validation sets are recorded and compared.  


|  |
| --- |
| **1. K-Nearest Neighbors (KNN):**  - Simple and intuitive algorithm.  - Calculates similarity between data points based on features.  - Classifies new instances by majority vote of neighbors.  **2. Linear Models (Logistic Regression, SGDClassifier, LinearSVC):**  - Computationally efficient.  - Logistic Regression: widely used for binary classification.  - SGDClassifier: offers flexibility in optimization techniques.  - LinearSVC: efficient for large-scale datasets.  **3. Support Vector Machine (LinearSVC):**  - Powerful classifier.  - Handles high-dimensional data and nonlinear relationships.  - Efficient for large-scale datasets.  **4. Tree-Based Models (DecisionTreeClassifier, ExtraTreeClassifier):**  - Robust and capable of capturing complex relationships.  - Interpretable.  - Handle nonlinearity well.  **5. Ensemble Models (GradientBoostingClassifier, RandomForestClassifier, ExtraTreesClassifier, CatBoostClassifier, XGBClassifier, LGBMClassifier):**  - Combine multiple weak learners to create strong learner.  - Outperform individual models.  - Robust to overfitting.  - Widely used ensemble techniques.  **6. Other Models (AdaBoostClassifier, MLPClassifier):**  - AdaBoost: Adaptive boosting algorithm focusing on hard-to-classify instances.  - MLPClassifier: Neural network capable of learning complex patterns. |

These models are selected based on their distinct characteristics and capabilities to address the heart disease classification problem effectively. They are trained on resampled data to mitigate class imbalance, and performance metrics are computed to evaluate their effectiveness in predicting heart disease.  
  
===============

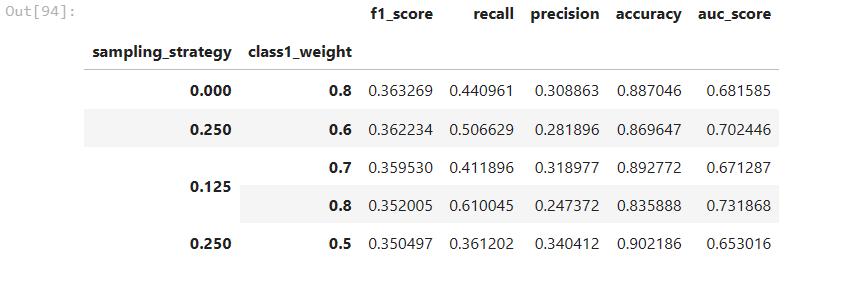
**Hyper-Parameter Tuning:**

- Grid search is performed to find the optimal combination of hyperparameters for the sampling strategy and class weights.

- The parameter grid includes different sampling strategies (0, 0.125, 0.25, 0.5, 1.0) and class weights (0.5, 0.6, 0.7, 0.8, 0.9).

- The average F1 score is computed over three stratified splits of the training data for each combination of parameters.

- Result: The grid search results in various combinations of sampling strategies and class weights along with their average F1 scores.



**Final Pipeline:**

- The best performing combination of hyperparameters (sampling strategy: 0.25, class weights: {0: 0.4, 1: 0.6}) is selected.

- A final pipeline is constructed, comprising the preprocessing steps and the LightGBM classifier with the selected hyperparameters.

- The pipeline is trained on the entire resampled training dataset.

- Result: The final pipeline is ready for evaluation and deployment.

**Model Evaluation:**

- The trained pipeline is evaluated on the validation set to assess its performance.

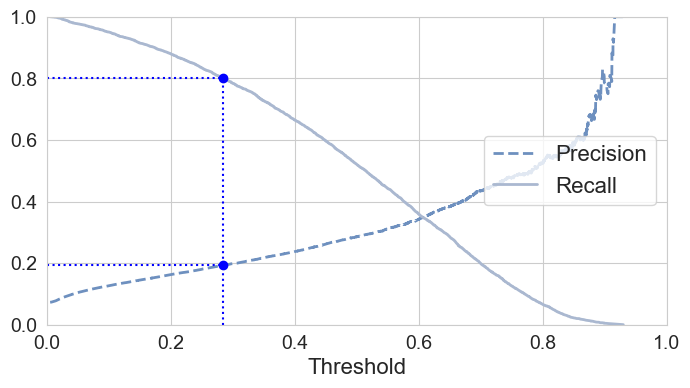
- Classification report and confusion matrix are generated to evaluate precision, recall, F1-score, and accuracy.

- Result: The model achieves an F1-score of 0.37 on the validation set.

**Threshold Adjustment:**

- Precision-recall curves are plotted to visualize the trade-off between precision and recall at different thresholds.

- A threshold is selected to achieve a desired recall level (80% in this case).

- Result: The threshold adjustment improves recall at the cost of precision.  


**Final Model Deployment:**

- The final pipeline, including preprocessing and the trained LightGBM classifier, is saved to a file using joblib.

- Result: The model is ready for deployment in production environments.

