8/6/2023

**Project Report**

ICR - Identifying Age-Related Conditions [Kaggle Competition]

Github Link:  
https://github.com/TrAnurag/Kaggle\_IGC/tree/main

**TEAM:**

**Anurag Singh (Group Lead) [ 041056292]**

**Aikhoje Ikhuoso [041096330]**

**Ben Biju Jacob [ 041085514]**

**Gabriel Ntoweng [041099292]**

**Happiness Ndubisi [041075639]**

**ICR - Identifying Age-Related Conditions (Kaggle Competition)**

**Project Summary:**

**Background**

There is a significant effect of age on health. From heart disease and dementia to hearing loss and arthritis, aging is a risk factor for numerous diseases and complications. The growing field of bioinformatics includes research into interventions that can help slow and reverse biological aging and prevent major age-related ailments. Data science could have a role to play in developing new methods to solve problems with diverse data, even if the number of samples is small.

**Problem Statement**

To develop a predictive model capable of detecting the presence of three specific medical conditions based on measurements of health characteristics. The model will aid researchers in identifying potential patients who may have one or more of the specified medical conditions without requiring a long and intrusive process of data collection from patients. Currently, models like XGBoost and random forest are used to predict medical conditions, yet the models' performance is not good enough. Dealing with critical problems where lives are on the line, models need to make correct predictions reliably and consistently between different cases.

Object:

* Predictive Model Development: Build and train a predictive model capable of accurately classifying individuals into two classes: those with age-related medical conditions (Class 1) and those without any age-related conditions (Class 0).
* Privacy Preservation: Ensure the privacy of patients by utilizing anonymized health characteristics in the model. The use of key characteristics will enable the encoding of relevant patient details while protecting sensitive information, thus complying with data privacy regulations.

**Methodology:**

The methodology encompasses data preprocessing, model building, and evaluation stages:

1. **Data Preprocessing:**
   * Load the training dataset and the supplementary metadata dataset (greeks.csv) into Pandas Data Frames.
   * Handle missing values: Impute or remove missing data points as appropriate.
   * Address outliers: Apply outlier detection and treatment techniques to enhance the quality of the data.
   * Explore and analyze the distributions of features to gain insights into the data.
2. **Exploratory Data Analysis (EDA):**
   * Perform visualizations and statistical analysis to understand the relationships between the health characteristics and the target variable (Class).
   * Investigate the supplemental metadata (greeks.csv) to identify any patterns or correlations with the target variable.
3. **Feature Selection:**
   * If necessary, use feature selection techniques to identify the most relevant health characteristics that contribute significantly to the prediction of age-related conditions**.**
4. **Model Building and Training:**
   * Split the preprocessed data into training and testing sets.
   * Implement multiple machine learning models based on the project's objectives, **including:**
     + Logistic Regression
     + XGBoost
     + Random Forest
     + Ensembled Learning
5. **Model Evaluation:**
   * Evaluate the trained models using appropriate metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.
6. **Conclusion and Documentation:**
   * Summarize the results and conclusions drawn from the model evaluation**.**

**Data Preprocessing**

**Data Description:**

The dataset provided for this project comprises two main components: the training dataset and the greeks dataset.

**Training Dataset**

The training dataset consists of 617 observations, each containing a unique ID and fifty-six health characteristics that have been anonymized. These characteristics include fifty-five numerical features and one categorical feature. Alongside the health characteristics, the dataset also includes a binary target variable called "Class." The primary goal of this project is to predict the Class of each observation based on its respective features.

**Libraries Used**

**Pandas**: Pandas is a powerful data manipulation and analysis library. It provides data structures like dataframe and Series, which allow you to work with structured data easily. You can load, filter, transform, and analyze data using pandas.

**Numpy**: numpy is a fundamental library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

**Matplotlib**: Matplotlib is a popular plotting library in Python. It allows you to create various types of plots, such as line plots, scatter plots, histograms, and more, to visualize your data and results.

**%matplotlib inline**: This is a Jupyter Notebook magic command. It allows the matplotlib plots to be displayed directly within the notebook cells.

**Seaborn**: Seaborn is built on top of matplotlib and provides a higher-level interface for creating attractive statistical graphics. It simplifies the process of creating complex visualizations and can enhance the default matplotlib plots.

**Sklearn (Scikit-learn):** Scikit-learn is a powerful machine learning library in Python. It provides a wide range of tools for data preprocessing, model selection, and evaluation. It includes many classification, regression, clustering, and dimensionality reduction algorithms.

**Train\_test\_split**: This function from `sklearn.model\_selection` helps in splitting the data into training and testing sets for building and evaluating machine learning models.

**R2\_score`, `mean\_squared\_error`:** These are evaluation metrics from `sklearn.metrics` for regression tasks. R-squared (R2) measures the proportion of variance in the dependent variable that is predictable from the independent variables. Mean squared error (MSE) calculates the average squared difference between the predicted and actual values.

**Classification\_report**: This function from `sklearn.metrics` generates a text report showing the main classification metrics (precision, recall, F1-score, and support) for each class in a classification problem.

**Confusion\_matrix:** Also from `sklearn.metrics`, this function computes the confusion matrix to evaluate classification model performance.

**Precision\_score, recall\_score, f1\_score**: These functions from `sklearn.metrics` are used to compute precision, recall, and F1-score for binary or multiclass classification problems.

**Roc\_curve, auc**: These functions from `sklearn.metrics` are used for receiver operating characteristic (ROC) curve analysis to evaluate binary classification model performance. AUC (Area Under the Curve) is a metric that represents the area under the ROC curve.

**Precision\_recall\_curve, average\_precision\_score**: These functions from `sklearn.metrics` are used for precision-recall curve analysis, which is especially useful for imbalanced classification problems.

**Make\_scorer:** This function from `sklearn. metrics` allows creating custom scoring functions for use in model evaluation during cross-validation.

**Data Shape**  
  
The dataset used in this analysis contains a total of 617 observations (rows) and 58 features (columns).

Each row represents a unique individual with health-related characteristics, and each column corresponds to a specific attribute.

**Statistical Information**  
  
A table was made that presents the summary statistics for the numeric features of the dataset. Each row corresponds to a specific numeric feature, and the statistics include count, mean, standard deviation, minimum, 25th percentile (Q1), median (50th percentile or Q2), 75th percentile (Q3), and maximum values.

Highlight some key insights from the summary statistics:

AB: The feature "AB" has a mean value of approximately 0.48, indicating that, on average, it falls around 0.48 units. The minimum value is 0.081, and the maximum value is 6.162. This feature appears to have some variation.

AF: The "AF" feature represents health-related characteristics with a mean value of approximately 3502.01 and a standard deviation of 2300.32. The values range from 192.59 to 28688.19.

AH: The "AH" feature has a relatively low standard deviation compared to the mean, suggesting that the values tend to cluster around the mean of 118.62. The minimum value is 85.20, and the maximum value is 1910.12.

Class: The target variable "Class" is binary with values of 0 or 1. The mean of approximately 0.18 indicates that about 18% of the individuals have been diagnosed with age-related conditions (Class 1).

EJ\_B: The binary categorical feature "EJ\_B" has a mean value of approximately 0.64, suggesting that about 64% of the observations fall into this category.

**Overall**, the summary statistics provide valuable insights into the distribution and spread of the numeric features in the dataset. The mean, standard deviation, and quartile values offer an initial understanding of the data's central tendency and variability, which will be further explored during the data exploration phase.

**Datatypes and Nulls**  
  
Dtypes: The data types of the columns are primarily of three types:

float64: There are 55 columns with float64 data type. These columns contain numeric values, which are often continuous or discrete measurements of various health-related characteristics.

int64: There is one column with int64 data type. This column is the target variable "Class," which is binary (0 or 1) and represents whether the individual has been diagnosed with age-related conditions (Class 1) or not (Class 0).

Object: There are two columns with object data type. The column "Id" serves as a unique identifier for each observation, and the column "EJ" is a categorical feature representing a specific health characteristic.

Non-Null Count: The non-null count for each column represents the number of non-missing (non-null) values in that column. Missing values are indicated by null values.

Memory Usage: The memory usage for the dataset is approximately 279.7 KB. This provides an idea of the amount of memory used to store the dataset in its current format.

**Missing Values**:

Columns BQ, CB, CC, DU, EL, FC, FL, FS and GL have missing values (NaN) as indicated by non-null counts being less than 617 (the total number of entries). These missing values will need to be handled during the data preprocessing phase before building machine learning models.

The dataset's information and data types provide crucial insights into the nature of the features, their data representations, and any missing data, which will be valuable during subsequent data preprocessing and modeling steps.

**Checking for Duplicated Values**  
  
To ensure data integrity and avoid potential biases in the analysis, it is essential to check for duplicated records in the dataset.

Duplicated records occur when two or more rows have the exact same values for all columns, indicating possible data entry errors or data collection issues.

Upon performing the check for duplicated values, we find that there are no duplicated rows in the dataset. The total number of duplicated rows is 0.

**Visualization and Feature Engineering**

We were able to get insights into the distribution of classes in the 'Class' column of the Data Frame. We saw how many occurrences there for each class and the corresponding percentage of each class are out of the total count.

Correlation:

The heatmap provides a clear visual representation of the correlation between different numerical variables in the dataset. Dark blue cells in the heatmap indicate strong positive correlations between the corresponding variables. This means that when one variable increases, the other variable tends to increase as well, and vice versa. Dark green cells in the heatmap indicate strong negative correlations between the corresponding variables.

This is evident by the correlation coefficient being 1.0 for these cells. It is a trivial correlation as each variable perfectly correlates with itself. The mask applied to the heatmap hides the lower triangular part, avoiding the replication of information due to the symmetry of the correlation matrix. This ensures that only the upper triangular part of the matrix is displayed in the heatmap.

**Hot Deck Imputation for Numerical Variables with Missing Values**

Hot deck imputation is a method used to fill in missing values in a dataset by borrowing values from "donor" observations that are like the observations with missing data. In this implementation, we identify the numerical variables that have missing values and perform hot deck imputation for each of them. The identified numerical variables with missing values are: 'EL', 'BQ', 'CB', 'CC', 'DU', 'EL', 'FC', 'FL', 'FS', and 'GL'.

Implementation Steps:

* For each variable with missing values, a "donor pool" is created using the rows that have non-null values for that variable.
* We identify numeric columns that can be used for similarity calculation between observations (numeric columns).
* For each observation with a missing value in the variable, we calculate a similarity measure (e.g., Euclidean distance) between the observation and the donors in the donor pool using the numeric columns.
* The observation receives the value from the donor with the closest match based on the calculated similarity measure.
* The missing values are imputed using the above process.

By imputing missing values, we ensured that the dataset is complete and ready for exploratory data analysis and machine learning modeling.

**Analyzing Target Variable**

The bar plot displays the distribution of the target variable "Class," which indicates whether an individual has been diagnosed with age-related conditions or not. Class 0 represents individuals without age-related conditions, while Class 1 represents individuals diagnosed with age-related conditions.

From the plot, we observe the following:

Class 0: The bar for Class 0 has a count of 509, which represents approximately 82.5% of the total dataset. This indicates that a majority (82.5%) of the individuals in the dataset do not have age-related conditions.

Class 1: The bar for Class 1 has a count of 108, which represents approximately 17.5% of the total dataset. This indicates that a minority (17.5%) of the individuals in the dataset have been diagnosed with age-related conditions.

The class distribution is not perfectly balanced, with Class 0 being the dominant class and Class 1 being the minority class. Imbalanced class distribution can have implications on model performance, especially in binary classification tasks. We decided to balance the target variable using sampling techniques.

**Normalization of features**

Here we used Min-Max scaler for the model as we wanted to maintain the pattern of the data frame.

Normalization of features is the process of scaling the values of different features of a dataset to a common scale, typically between 0 and 1 or -1 and 1. This is done to ensure that each feature contributes equally to the analysis and to prevent features with larger values from dominating the analysis.

Normalization is particularly important in machine learning algorithms that use distance-based metrics, such as k-nearest neighbors or clustering, because features with larger values will dominate the distance calculation. By normalizing the features, we can ensure that all features contribute equally to the distance calculation.

**Model Building and Training and Evaluation**

**Lenear Regression Model:**

This model performs reasonably well, with an accuracy of approximately 88.7%. It demonstrates a good balance between precision and recall, as indicated by the F1-score of 0.892. However, it may slightly favor the recall of the positive class (class 1) over the recall of the negative class (class 0) based on the higher sensitivity (recall) value for class 1. The specific performance of the model depends on the context and requirements of the classification task. The ROC curve and PR curve provide insights into the model's performance at different classification thresholds, while cross-validation helps assess the model's generalization ability on unseen data.

**RANDOM FOREST CLASSIFIER:**This model performs excellently with an accuracy of approximately 98.0%. It demonstrates high precision and recall for both classes, indicating that it is effective in correctly classifying both positive and negative instances. The F1-score confirms a good balance between precision and recall. The model also has a high specificity, indicating its ability to correctly identify negative instances. Overall, it is a strong binary classification model with impressive performance on the given dataset.

**XGBoost model:**  
This model performs with an accuracy of approximately 99.0%. It demonstrates high precision and recall for both classes, indicating that it is effective in correctly classifying both positive and negative instances. The F1-score confirms a good balance between precision and recall. The model also has a high specificity, indicating its ability to correctly identify negative instances. Overall, it is a strong binary classification model with impressive performance on the given dataset.

**Working Of XGBoost Model:**

The XGBoost model is also called Extreme Gradient Boosting. It is a powerful and widely used machine learning algorithm that belongs to the ensemble learning category. It combines the predictions of multiple individual models to create a stronger, more accurate model.

XGBoost specifically focuses on boosting, which is a technique that iteratively improves the model's performance by training new models to correct the errors of previous models.

Here's a more detailed explanation of XGBoost and its key components:

1. **Gradient Boosting Framework**: XGBoost is built upon the gradient boosting framework. Gradient boosting involves creating an ensemble of weak learners (usually decision trees) sequentially, with each new model attempting to correct the errors of the previous models. The final prediction is the sum of predictions from all individual models.

**2. Decision Trees as Base Learners**: In XGBoost, decision trees are commonly used as base learners. These trees are often shallow and restricted in depth to prevent overfitting. Each tree focuses on capturing specific patterns in the data.

**3. Objective Function**: XGBoost uses a combination of loss functions and regularization terms in its objective function. The objective is to minimize the sum of the loss function and a penalty term, which helps prevent overfitting.

**4. Regularization**: XGBoost provides L1 (Lasso) and L2 (Ridge) regularization terms in its objective function. These terms help control the complexity of the model and prevent it from fitting noise in the data.

**5. Tree Pruning**: XGBoost employs a technique called tree pruning to control the depth of individual trees. Pruning helps prevent trees from becoming too complex and overfitting the training data.

**6. Learning Rate**: The learning rate is a hyperparameter that controls the contribution of each individual model to the final prediction. A smaller learning rate makes the model more robust to overfitting, but it requires more boosting rounds.

**7. Feature Importance**: XGBoost provides a measure of feature importance, which helps identify the features that have the most impact on the model's predictions. This information is valuable for feature selection and understanding the data.

**8. Handling Missing Values**: XGBoost has the ability to handle missing values in the dataset, reducing the need for preprocessing.

**9. Cross-Validation**: XGBoost supports cross-validation to help tune hyperparameters and estimate model performance on unseen data.

When using XGBoost, it's important to carefully tune hyperparameters and monitor model performance to avoid overfitting

**Voting Classifier:**

A Voting Classifier is an ensemble learning technique in machine learning where multiple individual models (classifiers) are combined to make a final prediction.

**Cross Validation**

RandomForestClassifier was a bit consistent across the five folds with an average accuracy of 96.08%. This indicates that the model is capable of generalizing well to unseen data and is suitable for the classification task at hand.

XGBClassifier performed well with high accuracy across the folds, with an average accuracy of around 95.59%. The model has good performance, similar to the RandomForestClassifier**.**

LogisticRegression model had the least performace compared to other models with an average accuracy of 80.76%. Though it still shows decent performance, it is outperformed other models used**.**

Model: VotingClassifier Accuracy scores for each fold: 0.98529412 0.97058824 0.96568627 0.95073892 0.96059113 Average accuracy: 0.9665797353424128

**Summary**

To conclude, Voting Classifier,the RandomForestClassifier and XGBClassifier models had the best performance with the highest average accuracy compared to other models used. My team recommended them for this classification task due to their high performance in terms of their average accuracy. However, the choice of the final model also depends on other factors, such as interpretability, computational complexity, and the specific requirements of the application. Further analysis and fine-tuning will be conducted using Gridsearch to select the best model.

The LogisticRegression model, although showing decent accuracy, might benefit from further optimization or feature engineering to improve its performance.