## Project Overview

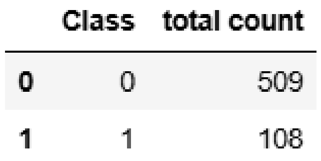
The project focuses on predicting whether a patient is diagnosed with a disease, with an accuracy exceeding 85%. To achieve this, the project will utilize advanced machine learning techniques, including comprehensive data preprocessing, feature engineering, and algorithms such as Random Forest. The project will integrate insights from the article *"Predicting disease risks from highly imbalanced data using random forest"*, which presents methods for handling imbalanced data and improving predictive accuracy. This isn't just about predicting disease likelihood — it's about enabling healthcare systems to make informed decisions, improve patient outcomes, and optimize treatment plans.

## Process Summary

**1. Gathering and Preparing Data:**
The first step of the project focused on preparing the data. The data required extensive preparation before analysis. It contained outliers, inconsistencies, and missing values that could distort the model's predictions if not properly handled.

**2. Exploring The Data:**

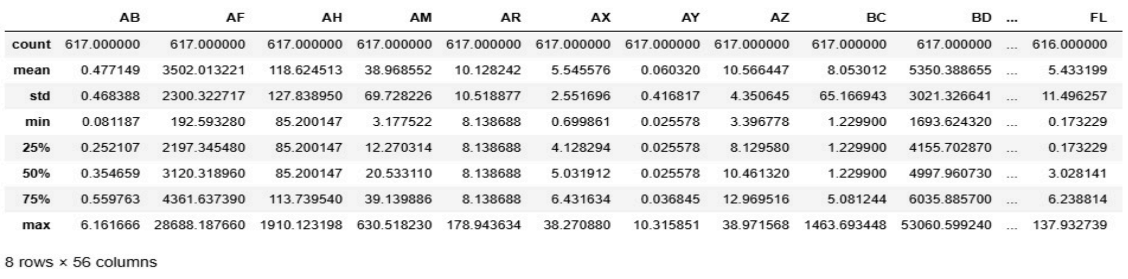
a) Displaying the distribution of data in the 'Class' column, which is considered the target column.



| | Class | total count |
|---|---|---|
| **0** | 0 | 509 |
| **1** | 1 | 108 |

b) Displaying a table with statistics for each numeric column in the dataset provides a summary of the data distribution and a better understanding of the key features of each column. This table helps in deciding how to handle the data.

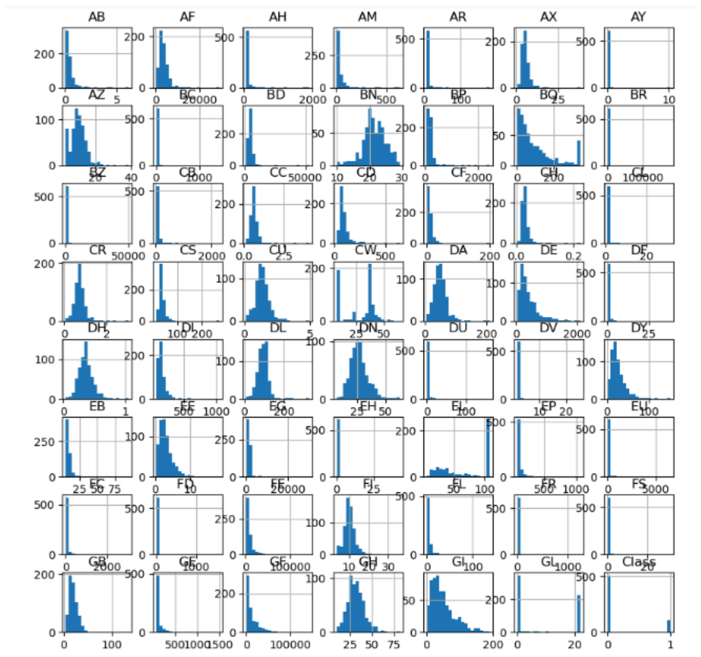
**count** : The number of non-missing entries in the column.
**mean** : The average value of the column.
**std** : The standard deviation, indicating the spread or variability of the data.
**min** : The smallest value in the column.
**25%** : The first quartile, representing the value below which 25% of the data falls.
**50%** : The median, where 50% of the data lies below this value.
**75%** : The third quartile, representing the value below which 75% of the data falls.
**max** : The largest value in the column.



| | AB | AF | AH | AM | AR | AX | AY | AZ | BC | BD | ... | FL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 617.000000 | 617.000000 | 617.000000 | 617.000000 | 617.000000 | 617.000000 | 617.000000 | 617.000000 | 617.000000 | 617.000000 | ... | 616.000000 |
| **mean** | 0.477149 | 3502.013221 | 118.624513 | 38.968552 | 10.128242 | 5.545576 | 0.060320 | 10.566447 | 8.053012 | 5350.388655 | ... | 5.433199 |
| **std** | 0.468388 | 2300.322717 | 127.838950 | 69.728226 | 10.518877 | 2.551696 | 0.416817 | 4.350645 | 65.166943 | 3021.326641 | ... | 11.496257 |
| **min** | 0.081187 | 192.593280 | 85.200147 | 3.177522 | 8.138688 | 0.699861 | 0.025578 | 3.396778 | 1.229900 | 1693.624320 | ... | 0.173229 |
| **25%** | 0.252107 | 2197.345480 | 85.200147 | 12.270314 | 8.138688 | 4.128294 | 0.025578 | 8.129580 | 1.229900 | 4155.702870 | ... | 0.173229 |
| **50%** | 0.354659 | 3120.318960 | 85.200147 | 20.533110 | 8.138688 | 5.031912 | 0.025578 | 10.461320 | 1.229900 | 4997.960730 | ... | 3.028141 |
| **75%** | 0.559763 | 4361.637390 | 113.739540 | 39.139886 | 8.138688 | 6.431634 | 0.036845 | 12.969516 | 5.081244 | 6035.885700 | ... | 6.238814 |
| **max** | 6.161666 | 28688.187660 | 1910.123198 | 630.518230 | 178.943634 | 38.270880 | 10.315851 | 38.971568 | 1463.693448 | 53060.599240 | ... | 137.932739 |

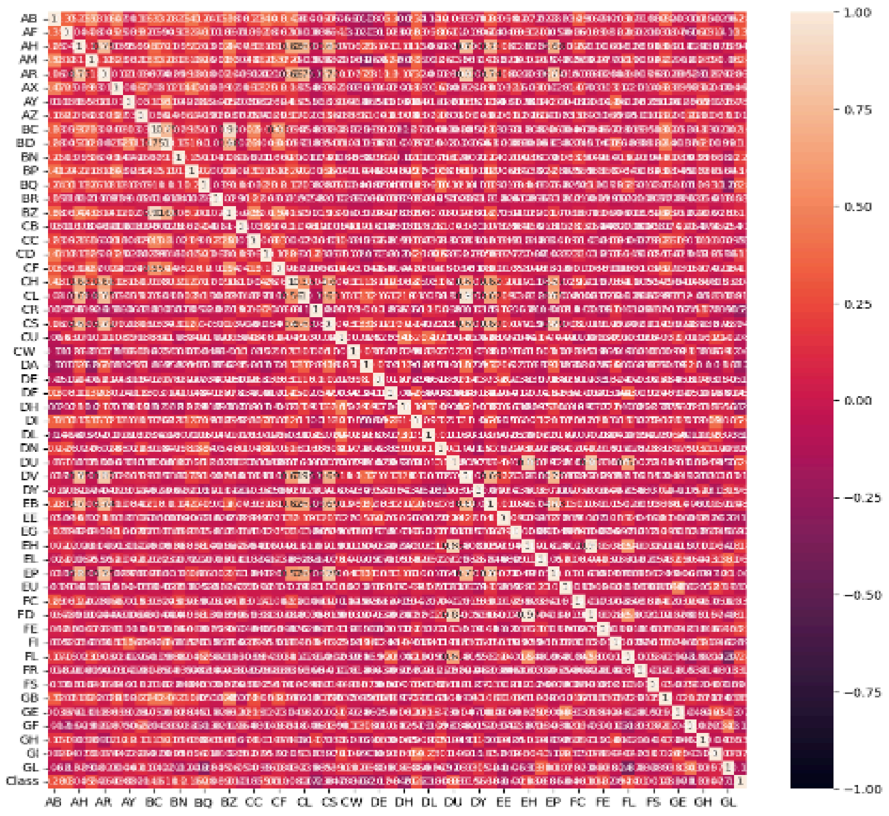
8 rows × 56 columns

## c) Histogram:



The histograms display the distributions of various variables in the dataset. Each subplot represents a specific variable, showing characteristics such as spread, skewness, and potential outliers. Some variables have wide ranges, while others are concentrated around lower values.

## d) Heatmap:



**Before Cleaning: The raw data.**

The heatmap illustrates the correlations between various variables in the dataset, with color intensity representing the strength and direction of the relationships.
Red and orange shades indicate **strong positive correlations** (closer to +1), where variables tend to increase together.
Purple shades represent strong **negative correlations** (closer to -1), where one variable increases as the other decreases.
White or neutral colors signify **weak or no correlation** (close to 0). The diagonal values, always +1, show perfect self-correlation of variables.
This visualization is useful for identifying significant positive or negative relationships and patterns within the dataset.

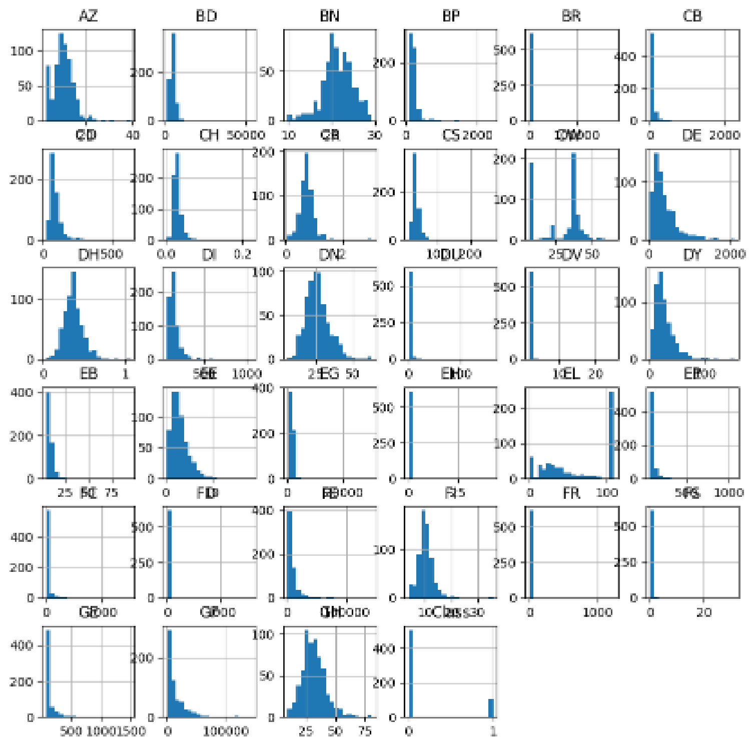
**3. Data Cleaning:** We cleaned the dataset by filling missing values, handling categorical values, dropping columns, removing features with high correlation- where correlation is defined to be between 0.2 and 0.8 and normalizing the data.

```
df_features = df_train.drop(columns=['Id', 'Class'])
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

**4. Exploring The Data After Cleaning:** Using the cleaned dataset, we proceeded to conduct Exploratory Data Analysis (EDA) to uncover underlying patterns and gain insights into the data. This step was essential for identifying the key factors influencing a patient's health status and diagnosing the likelihood of developing a disease, as well as understanding the overall structure of the dataset.

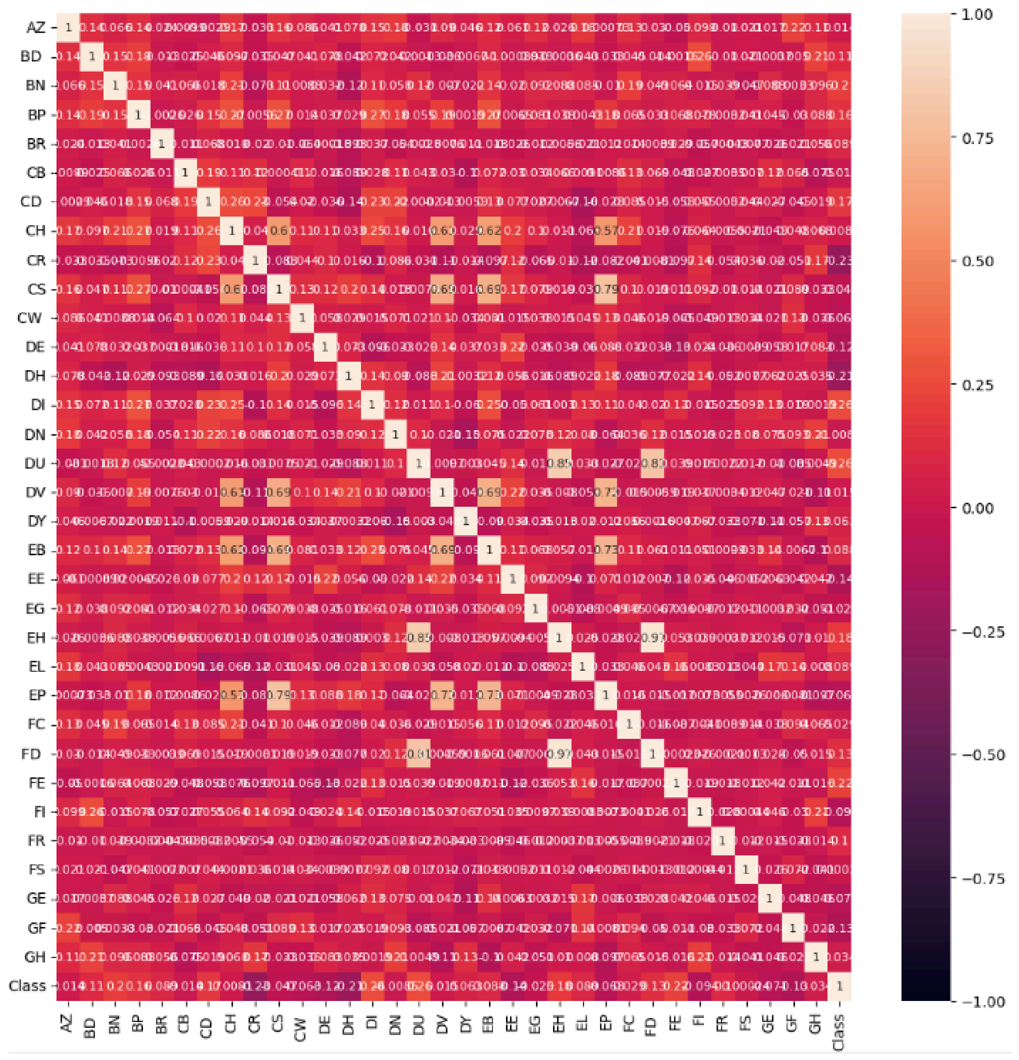
**a) Histogram:**



**After cleaning, the dataset has been significantly reduced and refined, ensuring it is fully prepared for analysis.**

b) **<u>Correlation Analysis by Heatmap</u>:** Through correlation analysis, we identified the variables most strongly linked to the likelihood of a patient developing a disease, based on their test results.

```
The best features are:  ['AZ', 'BD ', 'BN', 'BP', 'BR', 'CB', 'CD ', 'CH', 'CR', 'CS', 'CW ', 'DE', 'DH', 'DI', 'DN', 'DU', 'D
V', 'DY', 'EB', 'EE', 'EG', 'EH', 'EL', 'EP', 'FC', 'FD ', 'FE', 'FI', 'FR', 'FS', 'GE', 'GF', 'GH']
The best accuracy is: 0.9112903225806451
```



**After cleaning, the dataset has been significantly reduced and refined, ensuring it is fully prepared for analysis.**

The change in the Heatmap colors can be observed after removing the values with high correlation. Additionally, the size of the Heatmap has decreased due to the reduction in the number of features.

## 6. Preparing and Testing The Model:

After gaining a solid understanding of the data, we moved on to developing the model. This step is the core of the project—turning the data into a tool capable of making predictions.

The dataset is divided into training and testing sets. First, the features are scaled, and then the data is split, with 80% allocated to the training set and 20% to the testing set. This division enables model training on one subset of the data while evaluating its performance on a separate subset.

<u>**Model Selection:**</u> We defined a dictionary of classifiers and their respective hyperparameter grids  for use in a systematic model selection process. The classifiers include: Decision Tree, K-Nearest Neighbors(KNN), Naive Bayes, Artificial Neural Network, and Random Forest, each initialized with default parameters.

```
classifiers = {
    'Random Forest': RandomForestClassifier(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Naive Bayes': GaussianNB(),
    'Artificial Neural Network': MLPClassifier(max_iter=200),
    'Decision Tree': DecisionTreeClassifier()
}
```

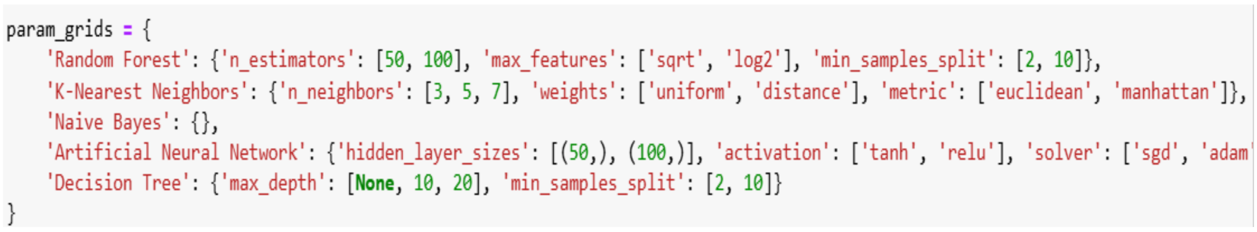
We implemented a classification pipeline that includes data preprocessing, class balancing, and hyperparameter optimization. Categorical variables are encoded using one-hot encoding (get-dummies).

```
df_train = pd.get_dummies(df_train, drop_first=True)
```

To optimize the model's performance, SMOTE is applied to address class imbalance in the training data.

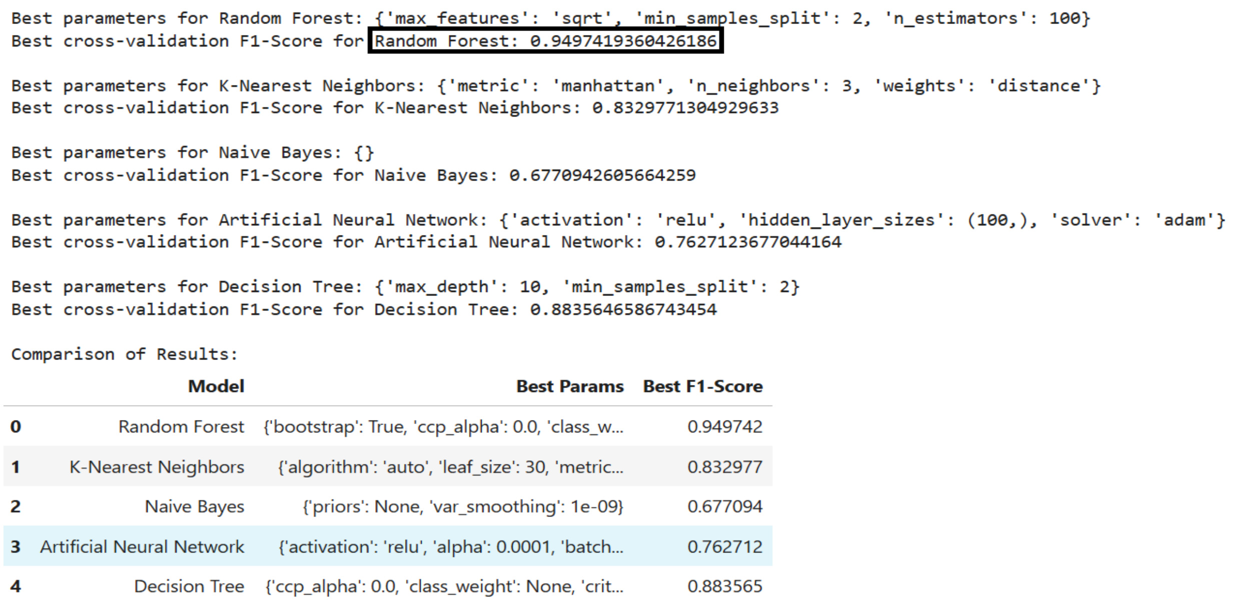
```
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

A grid search with 5-fold cross-validation identifies the optimal parameters for each model (classifier) based on accuracy.



```
param_grids = {
    'Random Forest': {'n_estimators': [50, 100], 'max_features': ['sqrt', 'log2'], 'min_samples_split': [2, 10]},
    'K-Nearest Neighbors': {'n_neighbors': [3, 5, 7], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan']},
    'Naive Bayes': {},
    'Artificial Neural Network': {'hidden_layer_sizes': [(50,), (100,)], 'activation': ['tanh', 'relu'], 'solver': ['sgd', 'adam'
    'Decision Tree': {'max_depth': [None, 10, 20], 'min_samples_split': [2, 10]}
}
```

**<u>Performance Evaluation:</u>** In the final phase, the model's performance was evaluated using the F1 Score metric. The results were compiled for comparison across different models, and the model with the best performance was selected for deployment. The **Random Forest** model emerged as the top performer with the highest F1 Score with approximately 94%, **consistent with findings from the referenced article.**



```
Best parameters for Random Forest: {'max_features': 'sqrt', 'min_samples_split': 2, 'n_estimators': 100}
Best cross-validation F1-Score for Random Forest: 0.9497419360426186

Best parameters for K-Nearest Neighbors: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
Best cross-validation F1-Score for K-Nearest Neighbors: 0.8329771304929633

Best parameters for Naive Bayes: {}
Best cross-validation F1-Score for Naive Bayes: 0.6770942605664259

Best parameters for Artificial Neural Network: {'activation': 'relu', 'hidden_layer_sizes': (100,), 'solver': 'adam'}
Best cross-validation F1-Score for Artificial Neural Network: 0.7627123677044164

Best parameters for Decision Tree: {'max_depth': 10, 'min_samples_split': 2}
Best cross-validation F1-Score for Decision Tree: 0.8835646586743454

Comparison of Results:
```

| | Model | Best Params | Best F1-Score |
|---|---|---|---|
| **0** | Random Forest | {'bootstrap': True, 'ccp_alpha': 0.0, 'class_w... | 0.949742 |
| **1** | K-Nearest Neighbors | {'algorithm': 'auto', 'leaf_size': 30, 'metric... | 0.832977 |
| **2** | Naive Bayes | {'priors': None, 'var_smoothing': 1e-09} | 0.677094 |
| **3** | Artificial Neural Network | {'activation': 'relu', 'alpha': 0.0001, 'batch... | 0.762712 |
| **4** | Decision Tree | {'ccp_alpha': 0.0, 'class_weight': None, 'crit... | 0.883565 |

For the selected Random Forest algorithm, the best parameters identified were: 'max_features': 'sqrt', meaning the maximum number of features considered at each split is the square root of the total number of features. 'min_samples_split': 2, which specifies the minimum number of samples required to split an internal node. and 'n_estimators': 100, indicating the forest consists of 100 trees. Using these optimized parameters, the model achieved a cross-validation F1 Score of 0.9497, which is the highest among all the algorithms tested. This demonstrates that Random Forest is highly effective for this dataset, likely due to its robustness and ability to handle complex feature interactions.