DiseaseX_ Detection

Project Description:

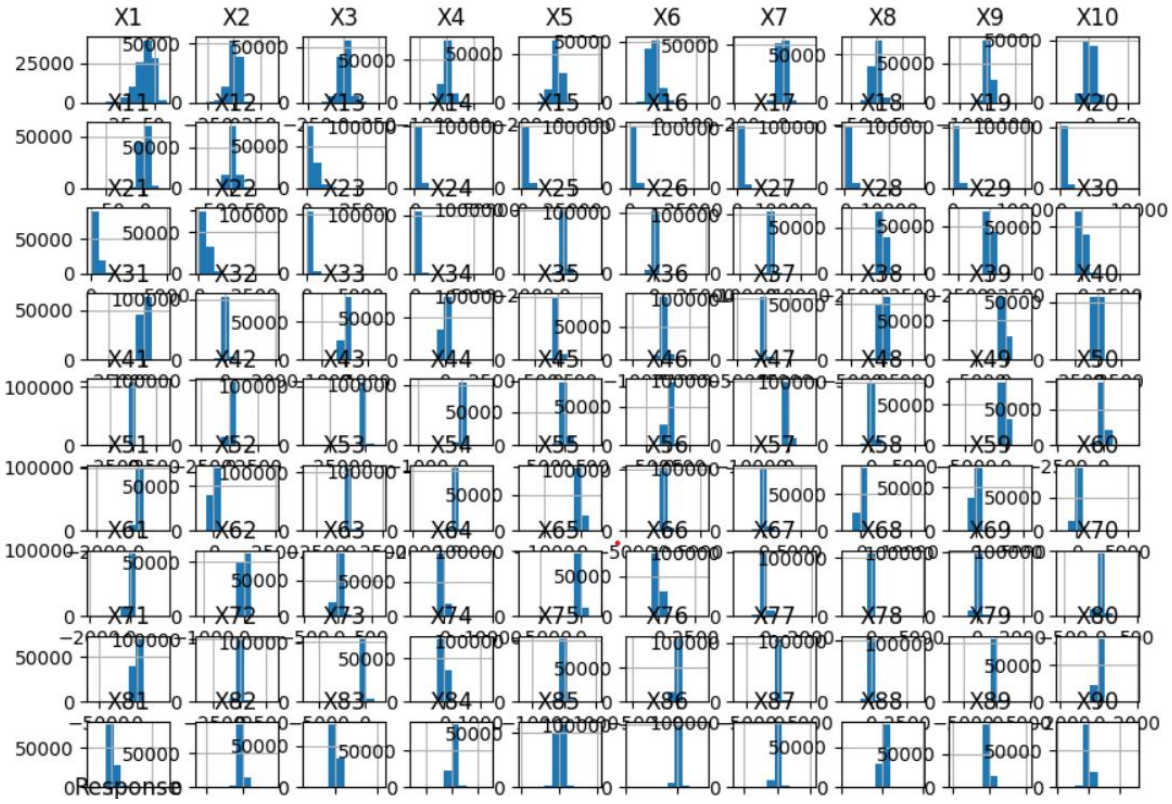
The project aims to develop a predictive model for disease detection based on physiological features presented in numerical form. The dataset comprises approximately 110,000 samples with 90 predictors. Each sample includes physiological measurements alongside the diagnosis of individuals as infected (class 1) or not (class 0). The objective is to create a reliable model that accurately classifies individuals into these two categories, aiding in early disease diagnosis and intervention.

The Predictors are named as [X1,X2….X90] and the Target column is named as ‘Response’.

Data visualization:

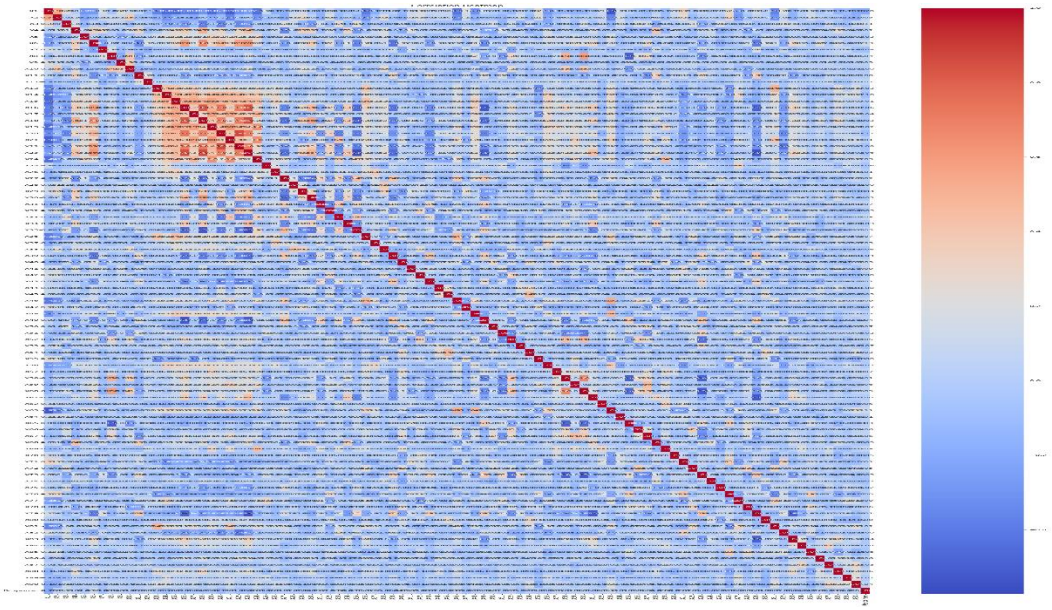
Box Plot for Each Feature:

A box plot is created to visualize the distribution of each predictor variable (X1 to X90). This plot provides insights into the spread of the data, including the median, quartiles, and potential outliers for each feature.



HeatMap:

- The correlation matrix of several variables, designated X1 through X90, as well as a response variable, are displayed in the heatmap.
- The heatmap's color scale shows how strongly and in which direction the variables are correlated. Colors like red in a heatmap typically indicate a positive association, blue a negative correlation, and white no correlation.



Data Pre-Processing and splitting Techniques:

The data preprocessing steps implemented in the code aim to ensure the cleanliness and uniformity of the dataset, laying a solid foundation for subsequent modelling and analysis tasks.

Data preprocessing involves two main steps:

1. Handling Missing Values: The code checks if the dataset contains any missing values using the .isnull().values.any() function. If missing values are detected, the code fills them using the median value of each column: df.fillna(df.median(), inplace=True). This approach ensures that missing values are replaced with a central tendency measure, maintaining the integrity of the dataset for further analysis and modeling.
2. Data Normalization/Standardization: After handling missing values, the code standardizes the numerical features using the StandardScaler() function from sklearn.preprocessing. This step ensures that all predictor variables are transformed to have a mean of zero and a standard deviation of one, thereby bringing them to a common scale. Standardization is essential for enhancing the performance of machine learning algorithms, as it prevents features with larger scales from dominating the model training process.

Splitting Technique:

The dataset is split into four main variables:

- **X_train**: Contains the predictor variables (features) for the training set.
- **X_test**: Contains the predictor variables (features) for the testing set.
- **y_train**: Contains the target variable (Response) for the training set.
- **y_test**: Contains the target variable (Response) for the testing set.

The **test_size** parameter is set to 0.3, indicating that 30% of the dataset will be allocated to the testing set, while the remaining 70% will be used for training. This split ratio strikes a balance between having sufficient data for model training and ensuring robust evaluation on unseen data.

Additionally, the **random_state** parameter is set to 42 to ensure reproducibility of the split. By fixing the random seed, the same data split will be obtained each time the code is executed, facilitating consistent evaluation of model performance.

Description of the Model used for Analysis:

(a)Model Selection:

1. **Logistic Regression:**
    - **Justification:** Logistic Regression is a popular choice for binary classification problems due to its simplicity, interpretability, and efficiency. It provides probabilistic outputs that can be easily interpreted, making it suitable for medical diagnostics where understanding the likelihood of disease occurrence is essential.
2. **Decision Tree:**
    - **Justification:** Decision Trees are versatile models that can handle both numerical and categorical data, making them suitable for datasets with mixed data types. They are capable of capturing nonlinear relationships and interactions between features, which may exist in the Disease X Detection dataset.
3. **Random Forest:**.
    - **Justification:** Random Forest is chosen for its ability to handle high-dimensional datasets with many predictors, such as the Disease X Detection dataset with 90 predictors. By aggregating the predictions of multiple trees, Random Forest reduces variance and improves generalization, making it robust to noise and outliers. It also provides feature importance measures, which can help identify the most relevant predictors for disease detection.

(b) Resampling method(s) utilized during training:

The resampling method utilized during training is K-Fold cross-validation with stratification. Specifically, the **StratifiedKFold** class from the **sklearn.model_selection** module is used to split the dataset into K folds while preserving the percentage of samples for each class in the target variable.

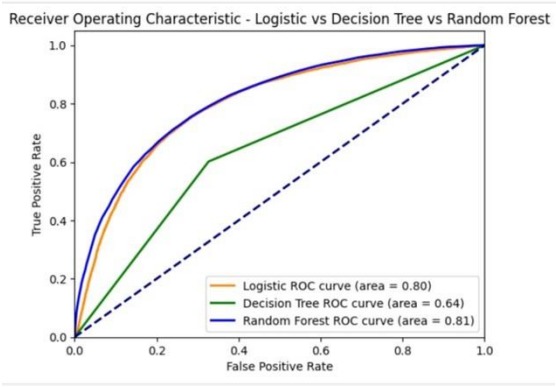
It partitions the dataset into K subsets (or folds), where each fold is used as a testing set exactly once, while the remaining K-1 folds are used for training. This process is repeated K times, with each fold serving as the testing set once.

(c) Performance metric used to gauge the prediction accuracy of the model:

1. Accuracy Score: This metric measures the proportion of correctly classified instances out of the total number of instances. It provides a general overview of the model's overall performance and is particularly useful when classes are balanced in the dataset.
2. Confusion Matrix: The confusion matrix is a tabular representation of actual versus predicted class labels. It provides insight into the model's performance by showing the number of true positives, true negatives, false positives, and false negatives. From the confusion matrix, other performance metrics such as precision, recall, and F1-score can be derived.
3. ROC Curve (Receiver Operating Characteristic Curve): The area under the ROC curve (AUC) is also calculated, with higher values indicating better discriminative power of the model.
4. Classification Report: This report provides a summary of various performance metrics including precision, recall, F1-score, and support (the number of occurrences of each class in the dataset)..

Visual Representation:

```
No missing values in the dataset.
Logistic Regression Cross-validated accuracy scores: [0.73298701 0.73363636 0.74246753 0.73337662 0.7371257
Logistic Regression Classification report:
              precision    recall  f1-score   support

           0       0.76      0.76      0.76     18029
           1       0.71      0.71      0.71     14971

    accuracy                           0.74     33000
   macro avg       0.73      0.73      0.73     33000
weighted avg       0.74      0.74      0.74     33000

Logistic Regression Confusion Matrix:
[[13698  4331]
 [ 4409 10562]]
Decision Tree Cross-validated accuracy scores: [0.63551948 0.6324026  0.63493506 0.6375974  0.63530099]
Decision Tree Classification report:
              precision    recall  f1-score   support

           0       0.67      0.67      0.67     18029
           1       0.60      0.60      0.60     14971

    accuracy                           0.64     33000
   macro avg       0.64      0.64      0.64     33000
weighted avg       0.64      0.64      0.64     33000
```



Conclusion:

After careful consideration of metrics and classification report we conclude the **Logistic Regression Model** is best fit for the data as it provides the Accuracy more compared to other model.

Deliverable 2:

1) In this updated deliverable, we used two new predictive models: Gradient Boosting Classifier (XGBoost) and Support Vector Machine (SVM).

1.Model Selection:

1. **Gradient Boosting Classifier (XGBoost)**:
    - **Justification**: XGBoost is a powerful ensemble learning algorithm known for its performance in various machine learning competitions. It's suitable for classification tasks, handles missing data well, and provides good accuracy without overfitting.
2. **Support Vector Machine (SVM)**:
    - **Justification**: SVM is a versatile algorithm for both classification and regression tasks. It's effective in high-dimensional spaces and performs well with clear margin separation. SVM can capture complex relationships in data and is robust against overfitting.

2.Resampling Method(s) Utilized During Training:

- **Stratified K-Fold Cross-Validation**: Used to ensure that each fold preserves the percentage of samples for each class. This method is suitable for imbalanced datasets, providing more reliable estimates of model performance.

3.Performance Metric Used:

Accuracy Score: Measures the ratio of correctly predicted instances to the total number of instances. It gives an overall idea of model performance.

Confusion Matrix: Provides a detailed breakdown of correct and incorrect predictions, enabling analysis of true positives, true negatives, false positives, and false negatives.

Sensitivity (Recall): Measures the proportion of actual positive cases correctly identified by the model.

Specificity: Measures the proportion of actual negative cases correctly identified by the model.

ROC Curve and AUC: Evaluates the trade-off between true positive rate and false positive rate across various thresholds, providing insights into model performance at different operating points.

2)Representation of Results:

- **Confusion Matrix**: Presented for both XGBoost and SVM models to provide insights into the types of errors made by each model.

```
XGBoost Confusion Matrix:
[[14431  3598]
 [ 5157  9814]]
```

```
SVM Confusion Matrix:
[[15003  3026]
 [ 4615 10356]]
```

- **Classification Report**: Displays precision, recall, F1-score, and support for each class, offering a comprehensive evaluation of model performance.

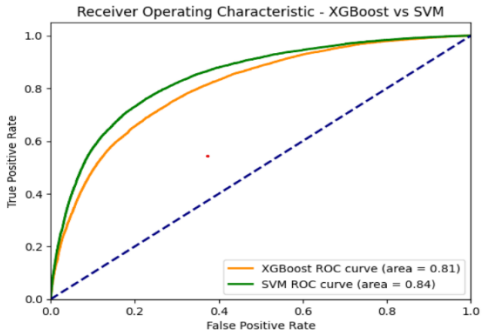
XGBoost Classification report:

	precision	recall	f1-score	support
0	0.74	0.80	0.77	18029
1	0.73	0.66	0.69	14971
accuracy			0.73	33000
macro avg	0.73	0.73	0.73	33000
weighted avg	0.73	0.73	0.73	33000

SVM Classification report:

	precision	recall	f1-score	support
0	0.76	0.83	0.80	18029
1	0.77	0.69	0.73	14971
accuracy			0.77	33000
macro avg	0.77	0.76	0.76	33000
weighted avg	0.77	0.77	0.77	33000

- **ROC Curve**: Plots for both models to visualize their discrimination ability across different thresholds, with AUC score provided for quantitative assessment.



- **Cross-validated Accuracy Scores**: Presented during model training to provide an estimate of each model's generalization performance.

```
XGBoost Cross-validated accuracy scores: [0.73402597 0.73266234 0.7388961  0.73006494 0.73563218]

SVM Cross-validated accuracy scores: [0.765      0.76525974 0.76883117 0.7611039  0.7685564 ]
```

3)Conclusion:

- **XGBoost vs. SVM Performance:** Both models demonstrated competitive performance, as evidenced by their accuracy scores, confusion matrices, and ROC curves. Although XGBoost exhibited slightly higher accuracy and AUC, SVM also performed admirably with consistently high accuracy scores across cross-validation folds.
- **Trade-offs and Considerations:** While XGBoost may offer slightly better predictive capability in this context, it's essential to consider factors such as computational efficiency, interpretability, and scalability when choosing the final model for deployment.
- Based on the evaluation results, both XGBoost and SVM show promise for DiseaseX detection. Considering the competitive performance and computational efficiency, SVM may be a suitable choice, especially if scalability and interpretability are significant concerns.