Aston University
BIRMINGHAM UK

DATA MINING (CS4850)

Swetha M Kulkarni(Student id:230180644)

Executive Summary:

The basic objective of the analysis is to develop a predictive model that has the potential to identify the linear B-cell epitopes from the high-dimensional dataset. This model has been designed for early vaccine development and therapeutic intervention activities for Chagas disease caused by Trypanosoma cruzi, which has been one of the major health problems in South America. We operationalize over a dataset that spans over 1,200 feature columns of proteomics data, embedded using state-of-the-art techniques made publicly available by Aston University.

From this proteomics data from Aston University, we did exploratory data analysis, including issues of data quality and techniques used in normalization and PCA for feature reduction. To increase the chances of our accuracy without increasing the imbalance in datasets, we experimented with some models like Random Forests, Decision Tree, and even Logistic Regression. The final selected model, based on balanced accuracy, gives promising results, which will simply be the epitope candidate prioritization for laboratory validation. Finally, the work shown here forms a strong base for future epitope prediction research using computational techniques to address one of the critical health concerns in South America.

Introduction:

Linear B-cell epitopes are, therefore, critical in the immune response against infectious diseases, and these epitopes become prime targets for which vaccines for prevention and therapeutics for treatment may be pointed out.

Identification of these epitopes is quite important in diseases for biomedical research, more especially those caused by Trypanosoma cruzi, like Chagas disease, which still remains a huge public health burden in endemic regions.

Most of these epitope prediction methods have relied on traditional immunogenic assays, which are laborious and hardly carry comprehensive epitope recognition. In that matter, the method used for epitope prediction is very critical and may use computational approaches, which hasten the discovery process with less need for expensive experimental validation.

This study, therefore, predicted T cell epitopes based on datasets emanating from the Immune Epitope Database (IEDB) and the NCBI Protein. Our project mainly aims to develop a sturdy data mining pipeline for the effective prediction of linear B-cell epitopes within T. cruzi proteins, taking the challenges of dimensionality and observation dependencies into account.

The present dataset, therefore, explores the use of a supervised approach to develop a predictive model for informing Chagas' disease research by providing potential epitope targets for experimental validation. We would take corresponding data preprocessing steps before putting that model to use in the quest for a better mechanistic understanding of epitope-host interaction and immune response to T. cruzi through systematic data mining, including explorative data analysis and modelling.

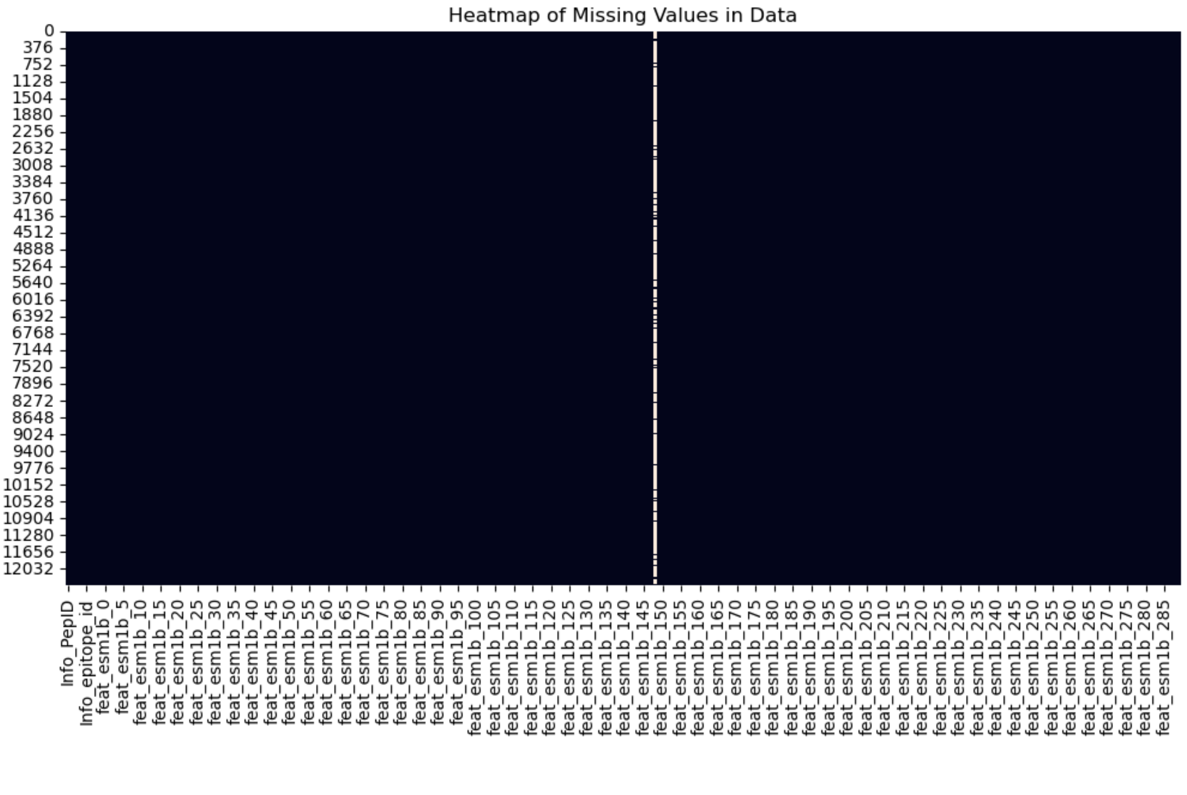
Details of our methodology and results are presented in the following sections. In so doing, we evidence a concrete outcome of the way our work shines light on the complexities of epitope prediction and translates insights into actionable strategies for advancing computational immunology to develop novel Chagas disease control strategies.

Description of solution:

EDA

In this EDA, we analyzed the structure, summary statistics, and datatypes of the 'df_reduced.csv' dataset with the help of the Python pandas library. Visualizations of Seaborn and Matplotlib help to understand the patterns of missingness and correlations between 290 numerical features.

The data is split by 80% for data exploration and model training and 20% for validation, with intrinsic cluster groupings, maintained so that data integrity is not compromised. The plotting of the missing values against other features had to be noted, knowing that they can be pre-emptive for multicollinearity before training the model. More specific insights include center tendency and feature value dispersion analysis. The mean value analysis indicated good centring around zero, an advantage to some machine learning models. On the other hand, boxplots summarized feature distribution and important outliers that were important in preprocessing. The summary standard deviation plots and range provide both stability and measurement precision of the features. This finding further justifies the need to have a look not only at the distribution of features but also at the variance of features in the process of feature selection, normalization, and transformation as a prelude to predictive modeling. This feature places a roadmap in both data preparation and model selection, such that the wealth in the dataset is well revealed for the structure of predictive analytics.



DATA PREPROCESSING:

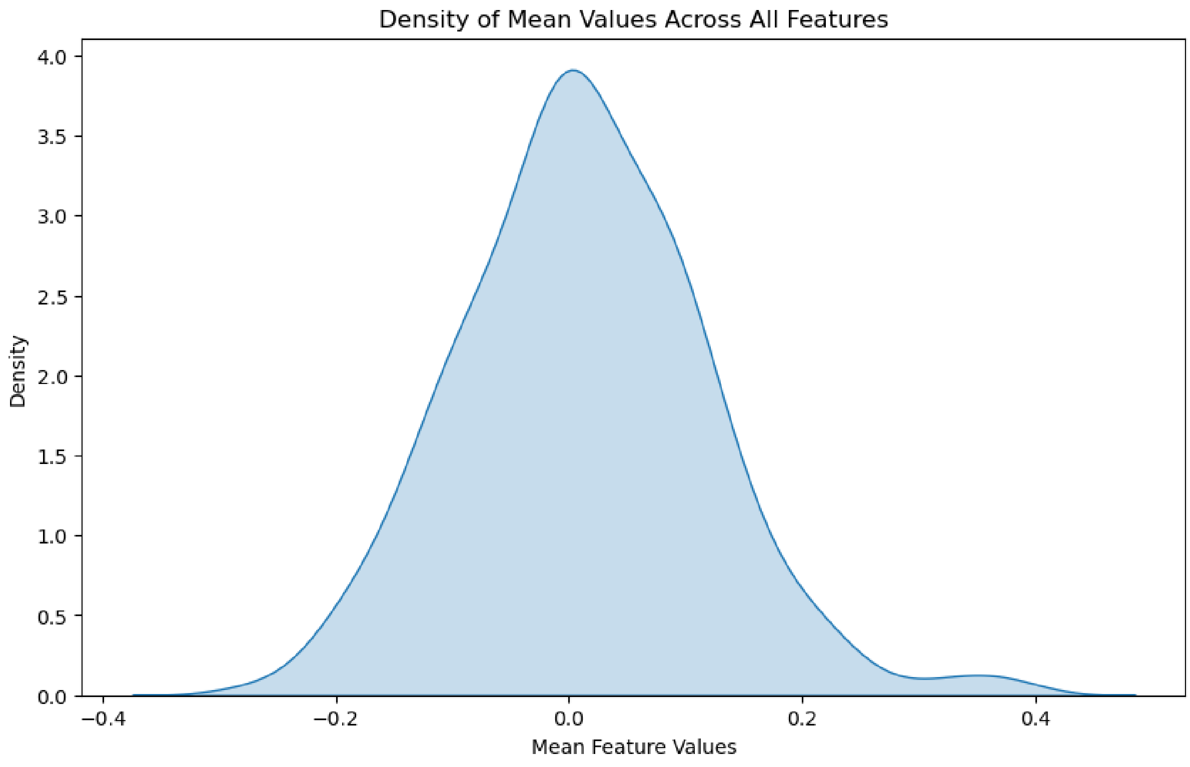
In the first pre-processing step, we took a pretty long journey to clean and refine the dataset for its best use with the machine learning models. First, we started addressing the issue of the missing values, so they wouldn't be a bother for us later.

We then imputed the median to the other features, which very few missing values, while maintaining the distribution of the values. After the integrity of the dataset, we then treated the issue of outliers, which otherwise might bring some sort of distortion for the predictive model. The numerical features were capped at the 5th and 95th percentiles in order to suppress extreme values. This was accomplished in such a way that the model was not very sensitive to some rare, though extreme, anomaly.

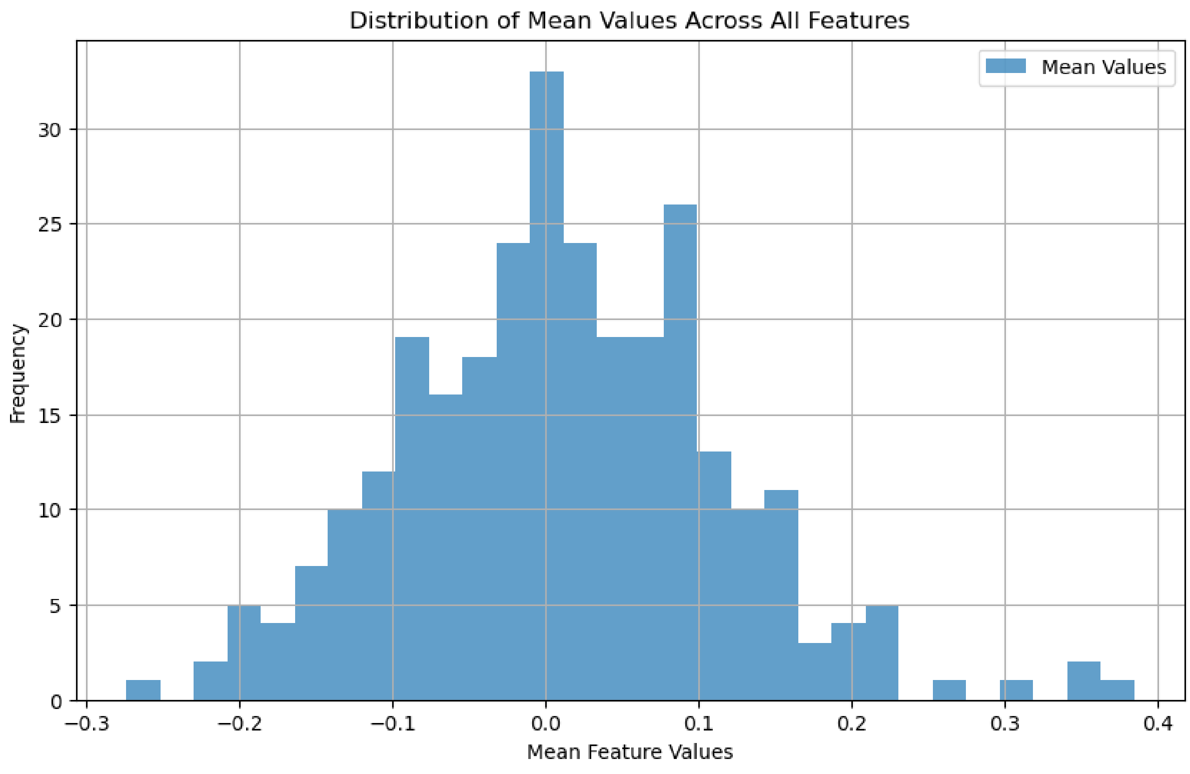
This was subject to standardization thereafter, with the use of StandardScaler, balancing the scales, thus ensuring no feature could outperform the rest and potentially dominate the model since its values had varying ranges. This standardizing is important for algorithms that work based on the size of input data, such as logistic regression, SVMs, and neural networks.

Finally, we corrected class imbalances in the data using Synthetic Minority Over-sampling Technique (SMOTE), hence equating representation between majority and minority class instances. This precludes the scope of any overfitting of the model to the majority class, rather enforcing a less class-represented sensitivity.

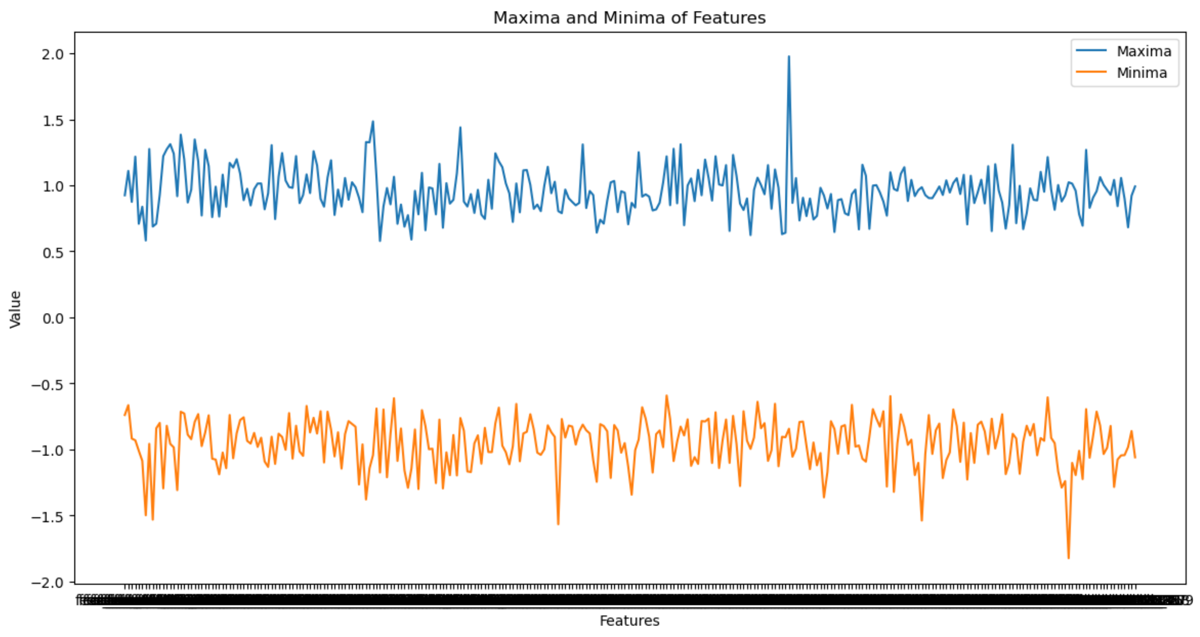
All along these steps, we encoded only those columns relevant even after capping, ensuring that the target variable is not included as a part of the feature transformations. This care, taken in advance, ensured that there was no such leakage of data, and ideally, the steps of preprocessing would be useful for training the model instead of accidentally giving bias.



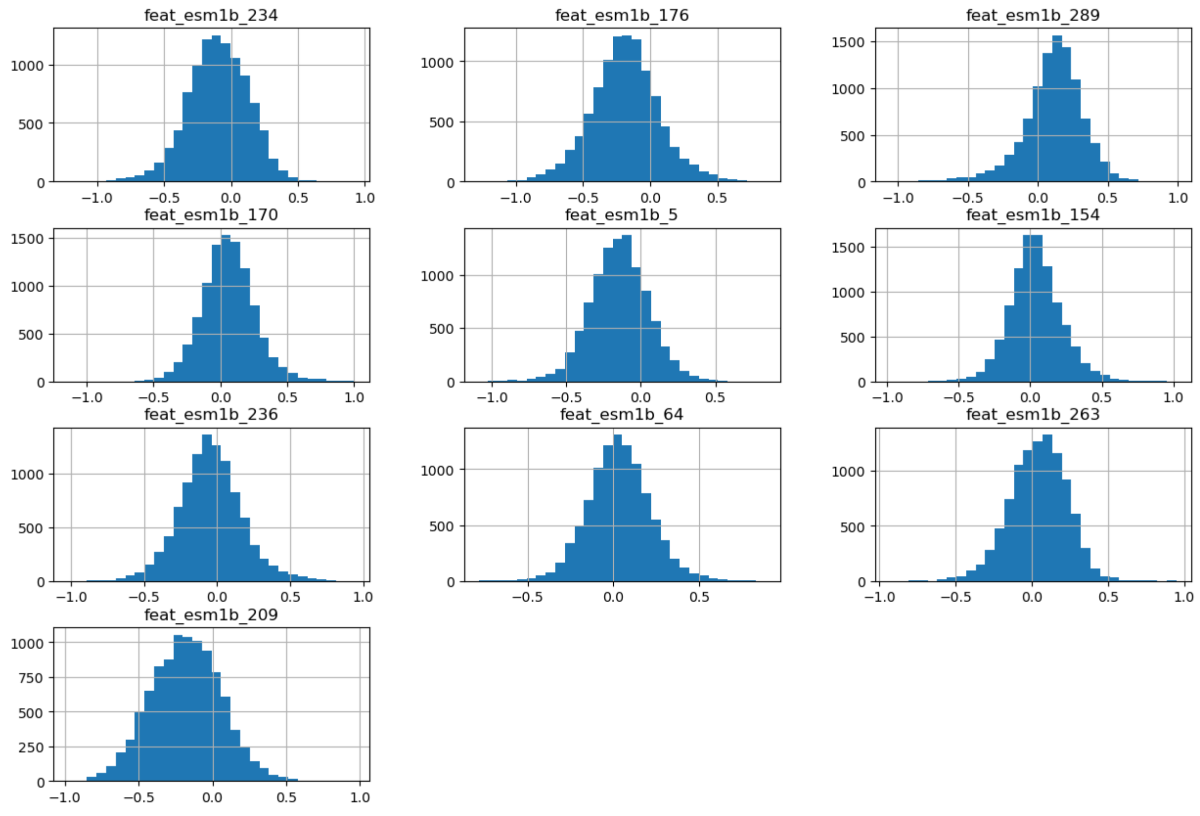
Density of Mean Values Across All Features
Density
Mean Feature Values

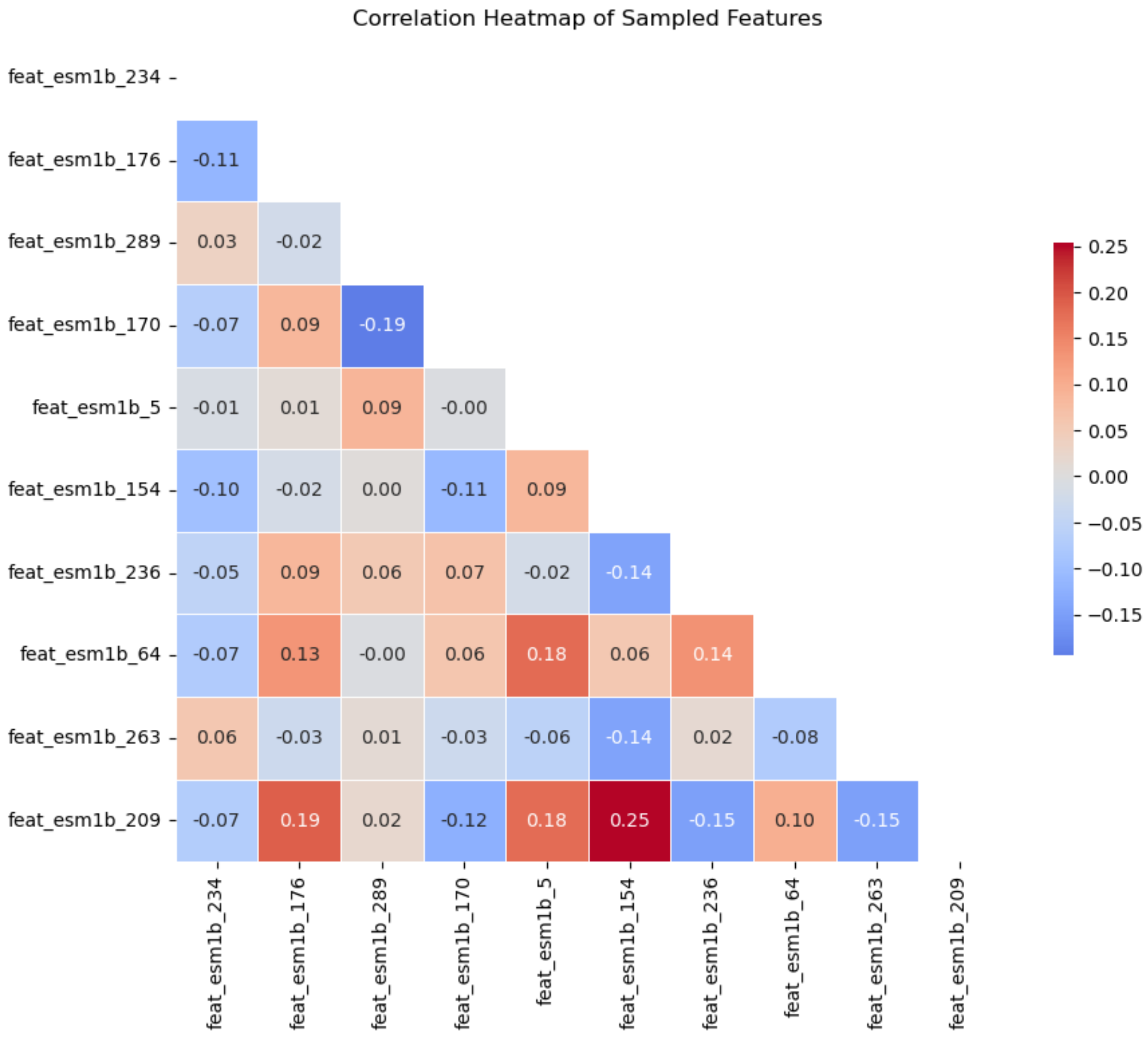


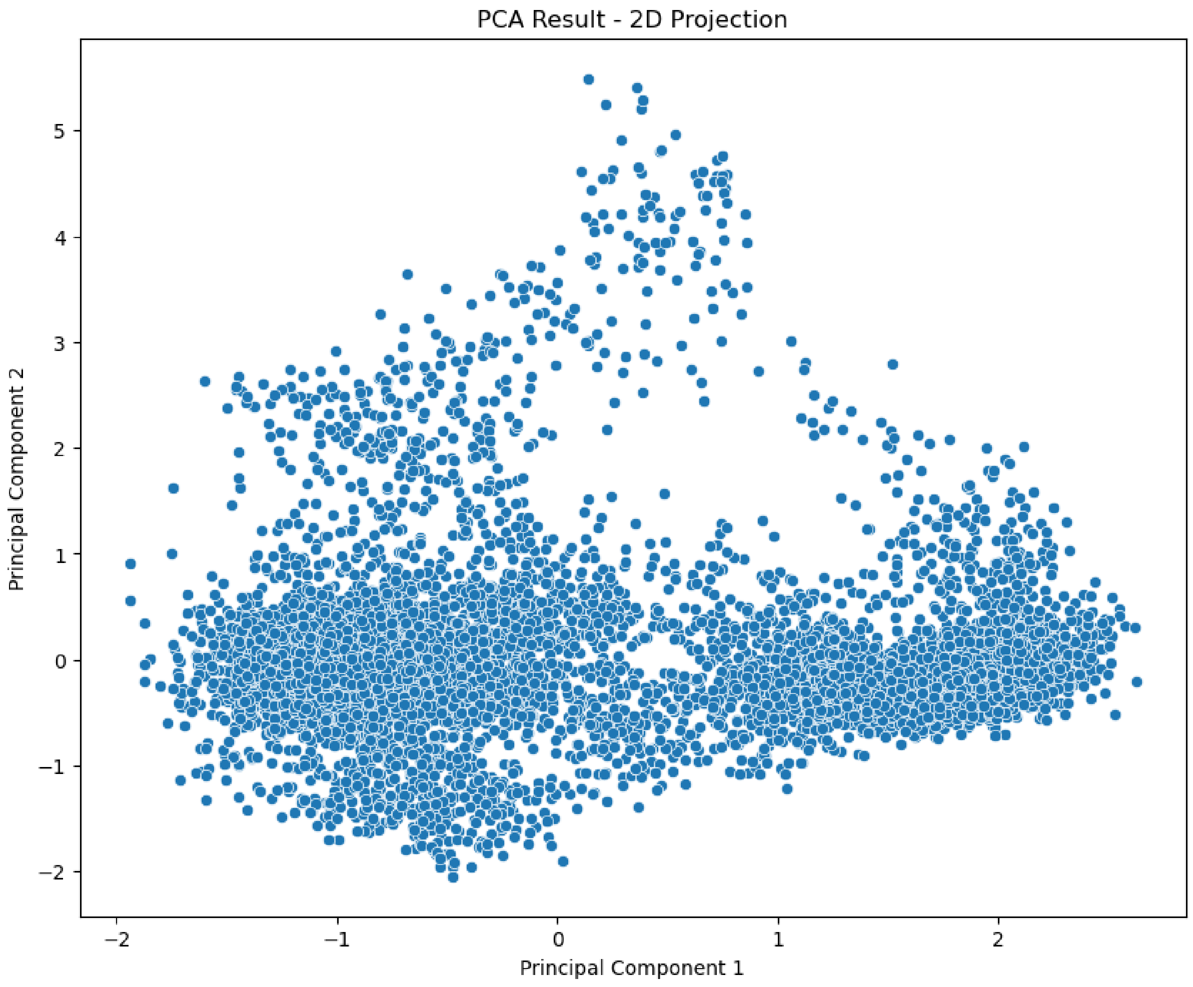
Distribution of Mean Values Across All Features
Mean Values
Frequency
Mean Feature Values

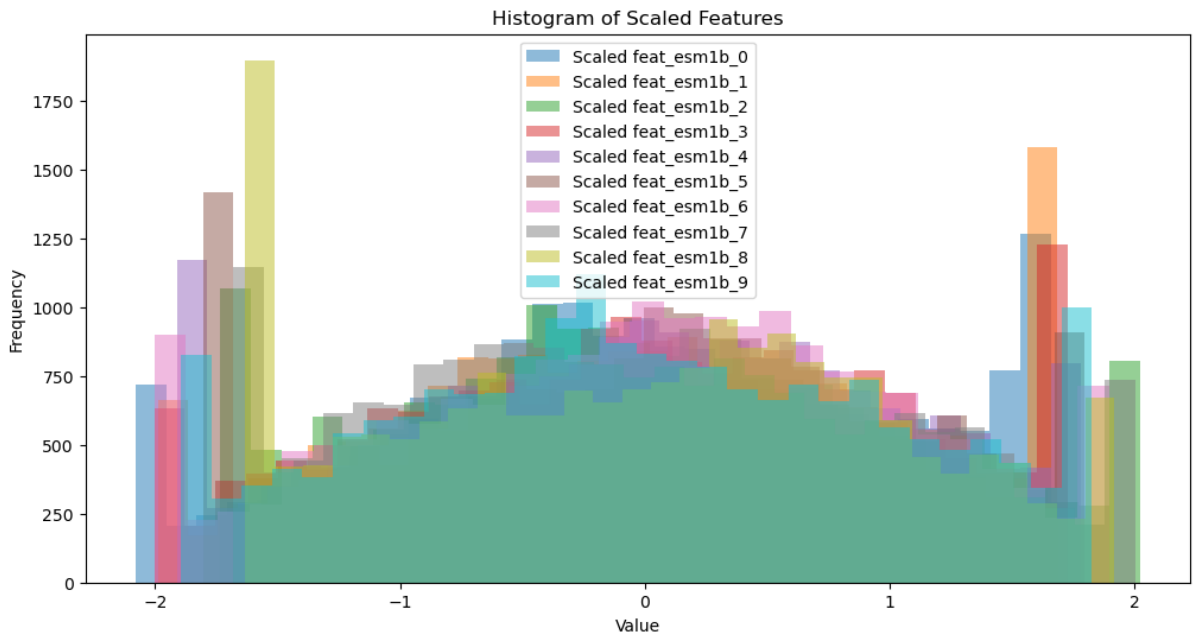


Feature Distributions




Correlation Heatmap of Sampled Features
feat_esm1b_234
feat_esm1b_176
feat_esm1b_289
feat_esm1b_170
feat_esm1b_5
feat_esm1b_154
feat_esm1b_236
feat_esm1b_64
feat_esm1b_263
feat_esm1b_209
-0.11
0.03
-0.02
-0.07
0.09
-0.19
-0.01
0.01
0.09
-0.00
-0.10
-0.02
0.00
-0.11
0.09
-0.05
0.09
0.06
0.07
-0.02
-0.14
-0.07
0.13
-0.00
0.06
0.18
0.06
0.14
0.06
-0.03
0.01
-0.03
-0.06
-0.14
0.02
-0.08
-0.07
0.19
0.02
-0.12
0.18
0.25
-0.15
0.10
-0.15
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15


PCA Result - 2D Projection
Principal Component 2
Principal Component 1
5
4
3
2
1
0
−1
−2
−2
−1
0
1
2


Histogram of Scaled Features
Scaled feat_esm1b_0
Scaled feat_esm1b_1
Scaled feat_esm1b_2
Scaled feat_esm1b_3
Scaled feat_esm1b_4
Scaled feat_esm1b_5
Scaled feat_esm1b_6
Scaled feat_esm1b_7
Scaled feat_esm1b_8
Scaled feat_esm1b_9
Frequency
Value
1750
1500
1250
1000
750
500
250
0
−2
−1
0
1
2

FEATURE SELECTION:

Feature selection in this phase of the analysis discussed some key steps in the feature engineering process to clean and finally reduce the dataset for further modeling purposes. This began with the extraction of scaled features from the dataset while ensuring that the target variable ('Class') is not undefined. Criteria for feature selection were based on the use of Mutual Information using the SelectKBest approach with a mutual_info_classif score, allowing to keep the top 10 features considered most informative for further analysis. Notably, these features included 'feat_esm1b_12', 'feat_esm1b_16', 'feat_esm1b_22', 'feat_esm1b_28', 'feat_esm1b_38', 'feat_esm1b_115', 'feat_esm1b_195', 'feat_esm1b_202', 'feat_esm1b_236', and 'feat_esm1b_285'. After the selection of features, a new dataset has been formed with a shape of (19628, 10) containing only the selected features along with the target variable, which is in turn of shape (19628,). Last but not least, the data was divided using GroupKFold for later modeling. This would allow efficient handling of data dependencies and permit the creation of training and testing sets. This process yielded training features with a shape of (15702, 10) and training labels with a shape of (15702,). On the other hand, testing features and testing labels have yielded shapes (3926, 10) and (3926,) respectively, again confirming that partitioning of the data was successful for further analysis.

PRILIMINARY MODEL ANALYSIS:

Preliminary Modeling Summary

The Logistic Regression model proved to be the best among all three classifiers used in this study, having yielded the highest scores of testing dataset accuracy and balanced accuracy. Besides, its precision, recall, F1-score metrics both reflected balanced performance among the two classes, as well as having an example of a perfect candidate to polish and take through evaluation to the next level of analysis.

Hyper-Parameter Tuning:

Random Forest Classifier:

To optimize the hyperparameters for the Random Forest classifier, we used GridSearchCV with a parameter grid exploring all different combinations of 'n_estimators', 'max_depth', 'min_samples_split', 'min_samples_leaf.

The best parameters identified for optimum performance included: 'bootstrap': False, 'max_depth': 30, 'min_samples_leaf': 1, 'min_samples_split': 2, and 'n_estimators': 100. Correspondingly, the best cross-validation score attained was at an exemplary 0.99, signaling the model fit to the training data was at very high levels.

Logistic Regression

Further, we used GridSearchCV to tune hyperparameters of the Logistic Regression model in a humanly manner. The parameter grid included options for C (strength of regularization), solver (optimization algorithm), and class_weight (weights related to classes). This provided the best parameter: 'C' set to 1, 'class_weight' to None, and 'solver' to 'liblinear', where with this best parameter at a cross-validation score of 0.80 in indicating the fit of training data. So, it did help the hyperparameter tuning performance of both models. The Random Forest model almost had perfect performance on training data, and the Logistic Regression model showed a huge improvement in the cross-validation score. "We have now completed this process of optimization, and the optimized models will be further evaluated using an unseen test dataset to assess the generalization performance.

PIPELINE:

Randomized Search for Logistic Regression Parameters:

At first, the model hyperparameters were tuned using the RandomizedSearchCV within the pipeline. To search for the best hyperparameters of the Logistic Regression model, a RandomizedSearchCV was used. With a parameter distribution of 'C' (the regularization strength) and 'solver' options of 'liblinear' and 'saga', it covered 100 combinations from the RandomizedSearchCV through 5-fold cross-validation. Best parameters resulted and were extracted as 'C_best' and 'solver_best'.

Pipeline Construction:

Implement a pipeline that incorporates all preprocessing steps, SMOTE for class balance, and a Logistic Regression classifier. Preprocessing numeric columns involved missing value imputation with median imputation and standardizing features. Class imbalance was handled using the SMOTE technique. The Logistic Regression classifier was used with certain adjusted parameters, such as 'solver' ('lbfgs'), 'max_iter' (1000), and 'class_weight' ('balanced').

Pipeline Evaluation:

The confusion matrix reveals true positive, false positive, true negative, and false negative predictions. The classification report presented both classes' (-1 and 1) precision and recall metrics along with the F1-score and accuracy. Moreover, the ROC AUC score measures the model's ability in discriminating classes.

```
Confusion Matrix:
 [[1473  275]
 [ 940 1238]]
Classification Report:
              precision    recall  f1-score   support

          -1       0.61      0.84      0.71      1748
           1       0.82      0.57      0.67      2178

    accuracy                           0.69      3926
   macro avg       0.71      0.71      0.69      3926
weighted avg       0.73      0.69      0.69      3926

ROC AUC Score: 0.7979049387152154
Balanced Accuracy Score: 0.7055443660654811
```

Result/ Prediction summary:

The predictions are saved into 'holdout_predictions.csv'.

When running the trained pipeline on the holdout dataset, we obtained a more balanced distribution of the predicted classes. Class -1 contained 52.10% of the samples, while class 1 held 47.90% of the samples. This more even distribution would seem to evidence a strong presence of both classes in the holdout data.