Key Indicators of General Health

Karan Gupta 2021258
Kuber Budhija 2021260
Shantanu Prakash 2021285
Shivesh Gulati 2021286
Rahul Oberoi 2021555

1. Abstract

This study evaluates various machine learning models to classify general health based on demographic and lifestyle factors, using a dataset of 445,132 entries from Kaggle. After preprocessing, the dataset contained 39 features, with 34 categorical and 5 numerical attributes, and the target variable representing general health status. The models tested include Naive Bayes, Logistic Regression, Decision Trees, Random Forest, Support Vector Classifier (SVC), AdaBoost, and XGBoost, with hyperparameter tuning performed using Grid Search.

Initial results indicated that XGBoost outperformed other models, achieving the highest test accuracy of 47.25% and a Macro F1 score of 43.60%, demonstrating its robustness in handling high-dimensional and imbalanced data. Random Forest also showed strong performance but exhibited overfitting, while simpler models like Naive Bayes provided a baseline for comparison. Hyperparameter optimization further improved model performance, with XGBoost achieving an optimized test accuracy of 47.18% and a Macro F1 score of 43.07%.

The results highlight the trade-offs between model complexity, interpretability, and computational cost, with XGBoost emerging as the most balanced and effective model for this task. This analysis offers insights into the relative strengths of different machine learning approaches in modeling health indicators, providing a foundation for developing targeted health interventions.

2. Problem Statement

To analyze and assess an individual's general health by examining a range of parameters, including BMI, ethnicity, sleep time, age, gender, and lifestyle factors. The aim is to identify patterns that correlate with various levels of health risk, allowing for a comprehensive understanding of how different factors impact an individual's overall health.

This assessment can guide individuals and healthcare providers in making informed decisions to improve health and reduce potential risks. Enhance our understanding of how lifestyle choices and medical conditions contribute to an individual's health. Assist in developing targeted interventions by identifying high-risk groups.

3. Dataset Description

We took dataset from kaggle [can be found at this link: https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease] we have total 445132 enteries out of which 157 are duplicates, we have total 39 features out of which 34 are categorical and 5 are numerical features, target column is general health.

The dataset contains 902,665 missing values, indicating that certain data points are incomplete and may require preprocessing 157 duplicate entries, which need to be handled to ensure data integrity.

The dataset contains a variety of features categorized into categorical and numerical columns. The categorical columns include attributes such as ”State,” ”Sex,” ”Had Asthma,” ”Had Skin Cancer,” ”Last Checkup Time,” ”Had COPD,” ”Physical Activities,” ”Had Depressive Disorder,” ”Removed Teeth,” ”Had Heart Attack,” ”Had Angina,” ”Had Stroke,” ”Blind or Vision Difficulty,” ”Difficulty Concentrating,” ”Difficulty Walking,” ”Difficulty Dressing Bathing,” ”Difficulty Errands,” ”Deaf or Hard of Hearing,” ”Race/Ethnicity Category,” ”Age Category,” ”Alcohol Drinkers,” ”HIV Testing,” ”Had Kidney Disease,” ”Had Arthritis,” ”Had Diabetes,” ”Smoker Status,” ”E-Cigarette Usage,” ”FluVaxLast12,” ”PneumoVaxEver,” ”TetanusLast10Tdap,” ”Chest Scan,” ”HighRiskLastYear,” and ”CovidPos.” On the other hand, the numerical columns consist of ”Physical Health Days,” ”Mental Health Days,” ”Sleep Hours,” ”Height in Meters,” ”Weight in Kilograms,” and ”BMI.” These features encompass a comprehensive range of health-related, behavioral, and demographic factors, providing a rich dataset for analyzing and predicting health outcomes.

4. EDA

In exploratory data analysis (EDA) of categorical features using count plots and pie charts, both before and after preprocessing. While plotting bar graphs for features, we observed significant variation among its entries which was readjusted after preprocessing. Then for another feature “Race/Ethnicity” we plotted pie charts and observed ”White only, Non-Hispanic” forms the largest group at 74.3% followed by ”Hispanic” at 10%, and other smaller groups. During preprocessing some minor adjustments were made to it to improve accuracy during training models. For EDA of numerical columns, we plotted box plots, violin plots, pair plots, histograms against target column “General Health” and correlation heatmap. We observed in box plots, for the columns ‘SleepHours’ , ‘WeightInKilograms’ And ‘HeightInMeters’Vs ‘GeneralHealth’ we observed a high amount of outliers which we removed during preprocessing. In Histograms, ‘MentalHealthDays vs GeneralHealth’ And ‘PhysicalHealthDays vs GeneralHealth’ indicate distribution of the two feature columns is very dispersed. (Most of the values are zero). In the correlation heatmap, we found there was no strong correlation between columns except 'WeightInKilograms' and 'BMI' indicating independence between majority of our columns.

5. Preprocessing

On the basis of our EDA, from box plots of 'SleepHours' , 'WeightInKilograms' and 'HeightInMeters' we detected outliers based on all the values lying outside the 1.5 InterQuantile Range and setting criterias based on our knowledge of columns. For data imputation, to replace NaN values, we used Mode of categorical feature columns and median in case of numerical feature columns. Also, this was necessary to be done before LOF (Local Outlier factor Detection) and using LOF Outlier detection, we set the contamination part = 0.1 and number of neighbours to 20 to filter out outliers based on above parameters. After the whole preprocessing, the dataset was reduced from 445,132 entries and 40 columns to 354862 entries and 40 columns.

6. Hypothesis Testing

To gain insight into the data and explore how various characteristics such as BMI, age category affect general health characteristics, we performed hypothesis tests. We conducted our hypothesis tests using random and stratified sampling techniques and then validated our hypothesis by performing the tests on the entire dataset. Table 1 summarizes the hypothesis tests that we have conducted on our dataset.

Hypothesis Test	**Null Hypothesis** (H_0)	**Alternate Hypothesis** (H_1)
χ^2 Test for Independence	Age Category does not affect General Health	Age Category affects General Health
Z-Test for Proportion	More than 40% of the population are overweight	At most 40% of the people of the population are overweight
One-Way ANOVA Test	Mean BMI for all the classes in the "GeneralHealth" column is the same	Mean BMI for all the classes in the "GeneralHealth" column is not the same.

Table 1. Hypothesis Tests and Hypotheses

6.1. χ^2 Test for Independence

We tested for the dependence between target column **GeneralHealth** and various dataset features.

6.1.1 Methodology

- We have used stratified sampling to create balanced subsamples where all of the target classes are equally represented. This will allows us to test for dependency without the skewed influence of the majority classes

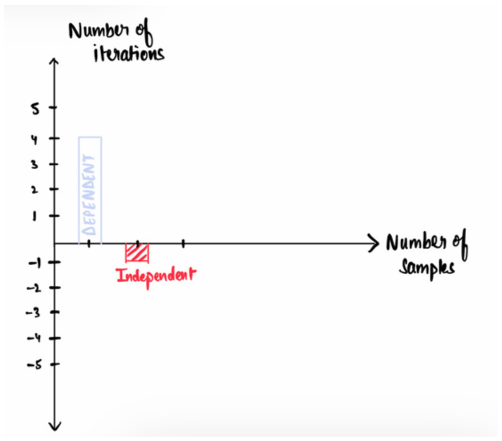


Figure 1. Graph of number of iterations versus the number of samples.

- We have taken multiple stratified samples of different sizes ranging from 100-1000 and 1000-10,000 for 5 iterations each, ensuring each sample has a similar proportion of all the classes.

- We then applied the chi-square test for independence on each of these stratified samples, where +1 represented dependence (rejection of null hypothesis) and -1 represented independence (fail to reject null hypothesis). This allowed us to verify if the hypothesis test is being satisfied consistently across the samples on an average.

6.1.2 Results Obtained

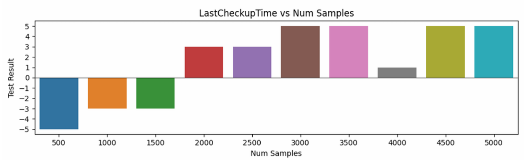


Figure 2. Graph of number of Samples versus the test result for LastCheckupTime (Random Sampling) (500-1000 samples)

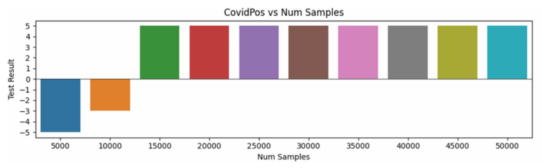


Figure 3. Graph of number of iterations versus test result for CovidPos (Random Sampling). (5000-50000 samples)

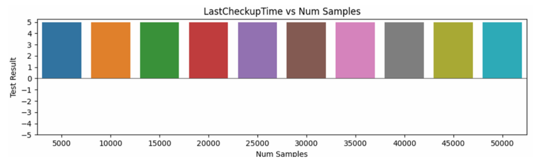


Figure 4. Graph of LastCheckupTime versus the number of samples.

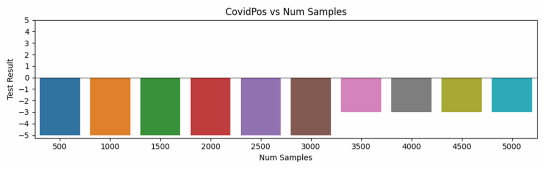


Figure 5. Graph of CovidPos versus the number of samples.

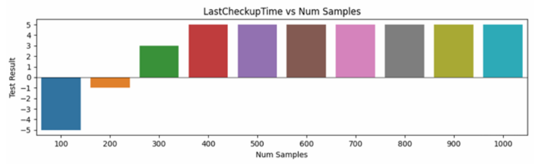


Figure 6. Graph of LastCheckupTime versus the number of samples.

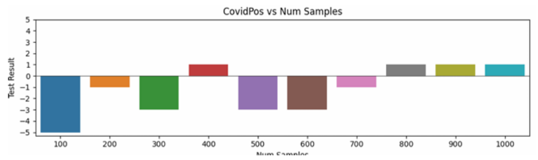


Figure 7. Graph of CovidPos versus the number of samples.

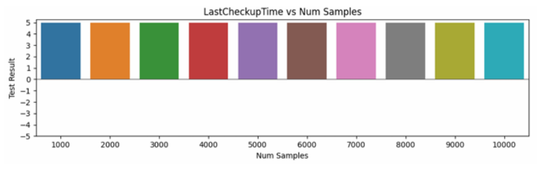


Figure 8. Graph of LastCheckupTime versus the number of samples.

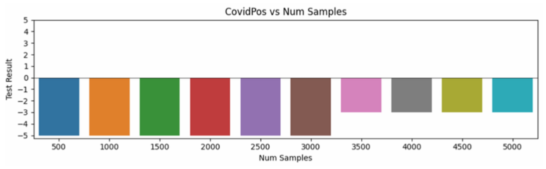


Figure 9. Graph of CovidPos versus the number of samples.

6.2. Z-Test for Proportion

6.2.1 Methodology

- According to the World Health Organization, 43% of people are classified as overweight (Source). Based on this statistic, we have used 40% as a reasonable estimate for the proportion of overweight individuals in our hypothesis.
- We have tested it for 10% (nearly 40,000 samples) of our data using stratified and random sampling.

6.2.2 Experiment Results

- Random Sampling
  - Z-Stat Value: 117.585
  - Critical Value: 1.644
  - Result: Fail to Reject H_0
- Stratified Sampling
  - Z-Stat Value: 118.810
  - Critical Value: 1.644
  - Result: Fail to Reject H_0

6.2.3 Validation

To validate our experiments, we conducted Z-Test for proportion on our entire dataset and we observed that more than 40 percent of the people are overweight. The median weight and height values that we received on our dataset are 81.19 kg and 1.7 m (170 cm). Calculating the BMI on these values we get 28 (Range of overweight is 25.0 - 29.9) which is in accordance with the result of the z-test.

6.2.4 Validation Results

Complete Dataset

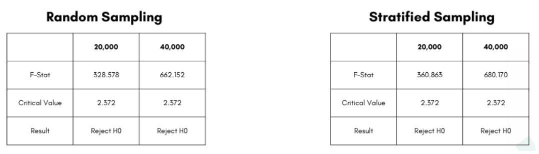
- Z-Stat Value: 350.398
- Critical Value: 1.644
- Result: Fail to Reject H_0

6.3. One-Way Anova Test

6.3.1 Methodology

- The motivation behind conducting this test was to analyze if there is variation in the mean BMI of the people that lie in various categories of health, for this we conducted the ANOVA test
- We tested that the mean BMI for all the classes (Poor, Fair, Good, Very Good, Excellent) in the "GeneralHealth" is the same for all of the classes i.e. mean of "poor" class = mean of "fair" class = mean of "good" class = mean of "very good" class = mean of "excellent" class.
- We have tested it for 5% (nearly 20,000) and 10% (nearly 40,000) data using stratified and random sampling.

6.3.2 Experiment Results



Random Sampling

	20,000	40,000
F-Stat	328.578	662.152
Critical Value	2.372	2.372
Result	Reject H0	Reject H0

Stratified Sampling

	20,000	40,000
F-Stat	360.863	680.170
Critical Value	2.372	2.372
Result	Reject H0	Reject H0

6.3.3 Validation

- To validate our experiments, we performed the ANOVA test upon our entire dataset and we observed that the mean BMI of all the classes is NOT the same.

- This result is in accordance to the tests we conducted, indicating that the mean BMI of different categories of health ranging from poor to excellent cannot be the same

6.3.4 Validation Results

- F-Stat Value: 5849.724

- Critical Value: 2.371

- Result: Reject H_0

7. Models Used

We evaluated a range of machine learning models to classify general health based on demographic and lifestyle factors. The models included Naive Bayes, Logistic Regression, Decision Trees, Random Forest, Support Vector Classifier (SVC), AdaBoost, and XGBoost, each chosen for its specific strengths and suitability for different data characteristics.

7.1. Naive Bayes

Naive Bayes is a probabilistic model that assumes independence among features, making it computationally efficient and straightforward to implement. It served as a strong baseline for this project. The model achieved an accuracy of 32.57% and a Macro F1 score of 34.38%. While these metrics were the lowest among the models evaluated, they provided a benchmark to measure improvements from more complex algorithms. However, the assumption of feature independence limited its ability to capture complex relationships within the dataset, making it less relevant for our objective of modeling nuanced health indicators.

7.2. Decision Trees

Decision Trees are non-parametric models that split the data recursively based on feature thresholds, making them interpretable and effective for smaller datasets. The model achieved an accuracy of 35.64% and a Macro F1 score of 33.69%. Although the performance was limited due to the high dimensionality and imbalance of the dataset, Decision Trees provided valuable insights into feature importance and interactions within the data.

7.3. Random Forest

Random Forest is an ensemble model that combines multiple decision trees to improve predictive accuracy and reduce overfitting. The model achieved an accuracy of 45.33% and a Macro F1 score of 41.69%. Random Forest proved relevant for its ability to handle complex data and provide feature importance, aiding in understanding the dataset's structure.

7.4. Logistic Regression

Logistic Regression is a linear model widely used for binary and multi-class classification tasks. It is effective when the data is approximately linearly separable. It achieved an accuracy of 45.34% and a Macro F1 score of 37.17%. Logistic Regression's relevance lies in its interpretability and ability to serve as a benchmark for assessing the contributions of scaling and preprocessing techniques applied to the dataset.

7.5. AdaBoost

AdaBoost is an ensemble method that combines weak learners iteratively, focusing on misclassified samples to improve overall performance. The model achieved an accuracy of 45.71% and a Macro F1 score of 38.21%. AdaBoost's iterative approach made it particularly relevant for this dataset, where misclassification of minority classes required additional focus.

7.6. XGBoost

XGBoost, a gradient boosting algorithm with built-in regularization, emerged as the best-performing model. It achieved the highest accuracy of 47.25% and a Macro F1 score of 43.60%. XGBoost was highly relevant for this project due to its ability to optimize gradient boosting processes and balance model complexity with predictive accuracy, making it the most robust model for identifying key health indicators.

7.7. Support Vector Classifier

SVC leverages kernel functions to find non-linear decision boundaries, making it effective for high-dimensional datasets. The model achieved an accuracy of 45.02% and a Macro F1 score of 33.04%, with a substantial training time of approximately 14,529 seconds. While SVC offered insights into non-linear patterns, its computational expense rendered it less practical for this dataset, especially when compared to other models with comparable or superior performance.

7.8. Results Obtained

Results without Grid Search are listed down below:

8. Hyperparameter Tuning using Grid Search

We applied Grid Search with cross-validation to systematically optimize hyperparameters for selected models. This approach allowed us to fine-tune the models and identify configurations that maximized their performance.

The best-performing model after Grid Search was XGBoost, which achieved an accuracy of 47.18% and a Macro F1 score of 43.07% on the test dataset. This performance demonstrates

Model	**Accuracy**	**Macro F1**
Naive Bayes	0.3257	0.3438
Logistic Regression	0.4554	0.3717
Decision Trees	0.3564	0.3569
Random Forest	0.4555	0.4169
Support Vector Classifier	0.4502	0.3304
AdaBoost	0.4571	0.3821
XGBoost	0.4725	0.4360

Table 2. Model performance metrics

its robustness in handling imbalanced and high-dimensional data. The training time, optimal hyperparameters, and evaluation scores for all models are summarized in the table below. For comparison, the scores prior to hyperparameter tuning are also presented, highlighting the improvements achieved through this process.

The best parameters, training times, train and test accuracies and the macro F1s on the train and test are listed in table 3.

8.1. Graphs Obtained

The graphs compare various models based on their computation time, test accuracy, and macro F1 scores for training and testing. SVM takes the longest time to compute, while Naive Bayes is the fastest. In terms of test accuracy and macro F1 scores, XGBoost and Random Forest perform the best, achieving similar scores while Naive Bayes shows the poorest performance. Random Forest exhibits significant overfitting, with a training macro F1 score close to 0.95, much higher than its test performance. Overall, XGBoost stands out as a balanced performer with good accuracy and F1 scores, while Random Forest requires attention to mitigate overfitting.

8.2. Comparison

- We observe that the lowest macro-f1 is observed in the case of dimensionality reduction using JL Lemma because it only preserves the distance between any two points while projecting into the lower dimensional space. It does not take into account the number of samples of that class and the class relationships between the points and just randomly projects the point into a lower dimension

- The highest macro-f1 among all the the dimensionality reduction techniques was observed when we used Coresets, because we used importance sampling to assign probability to each class sample, based upon the number of sample data points present of that particular class, which also preserves the class relationships while projecting in the lower dimension.

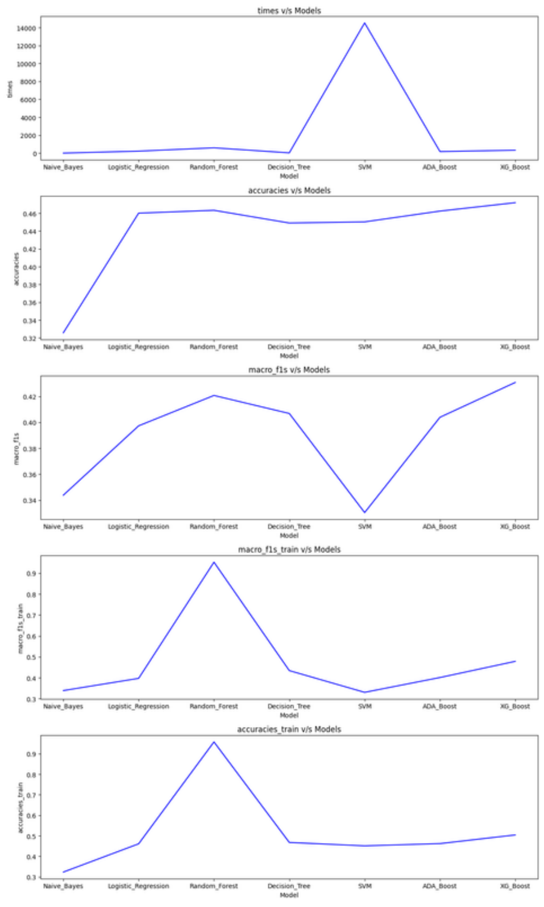


Figure 10. Comparison between models

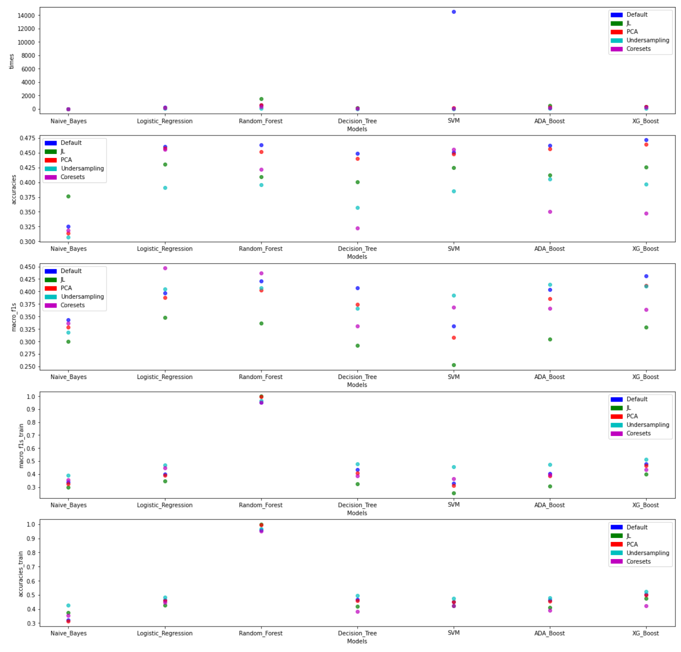
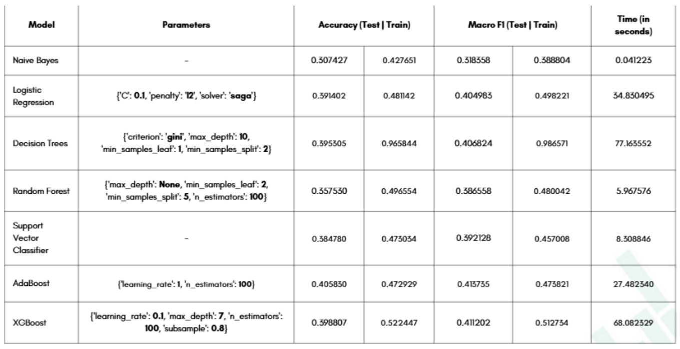


Figure 11. Comparison between different scaling techniques

Model	**Parameters**	**Accuracy (Test — Train)**	**Macro F1 (Test — Train)**	**Time (seconds)**
Naive Bayes	-	0.3257 — 0.3231	0.4718 — 0.3390	0.2980
Logistic Regression	{’C’: 0.1, ’penalty’: ’l2’, ’solver’: ’saga’}	0.4600 — 0.4609	0.3973 — 0.3973	216.5667
Decision Trees	{’criterion’: ’gini’, ’max_depth’: 10, ’min_samples_leaf’: 1, ’min_samples_split’: 2}	0.4490 — 0.4676	0.4207 — 0.4344	25.924320
Random Forest	{’max_depth’: None, ’min_samples_leaf’: 2, ’min_samples_split’: 5, ’n_estimators’: 100}	0.4632 — 0.9577	0.4068 — 0.9523	588.541620
Support Vector Classifier	-	0.4502 — 0.4511	0.3304 — 0.3306	14529.0550
AdaBoost	{’learning_rate’: 1, ’n_estimators’: 100}	0.4623 — 0.4620	0.4038 — 0.4013	170.2865
XGBoost	{’learning_rate’: 0.1, ’max_depth’: 7, ’n_estimators’: 100, ’subsample’: 0.8}	0.4718 — 0.5042	0.4307 — 0.4786	321.5975

Table 3. Model Performance After Grid Search



Model	**Parameters**	**Accuracy (Test | Train)**		**Macro F1 (Test | Train)**		**Time (in seconds)**
Naive Bayes	-	0.307427	0.427651	0.318358	0.388804	0.041223
Logistic Regression	{'C': **0.1**, 'penalty': '**l2**', 'solver': '**saga**'}	0.391402	0.481142	0.404983	0.498221	34.830495
Decision Trees	{'criterion': '**gini**', 'max_depth': **10**, 'min_samples_leaf': **1**, 'min_samples_split': **2**}	0.395305	0.965844	0.406824	0.986571	77.163552
Random Forest	{'max_depth': **None**, 'min_samples_leaf': **2**, 'min_samples_split': **5**, 'n_estimators': **100**}	0.357530	0.496554	0.386558	0.480042	5.967576
Support Vector Classifier	-	0.384780	0.473034	0.392128	0.457008	8.308846
AdaBoost	{'learning_rate': **1**, 'n_estimators': **100**}	0.405830	0.472929	0.413735	0.473821	27.482340
XGBoost	{'learning_rate': **0.1**, 'max_depth': **7**, 'n_estimators': **100**, 'subsample': **0.8**}	0.398807	0.522447	0.411202	0.512734	68.082329

Figure 12. Results Under Sampling

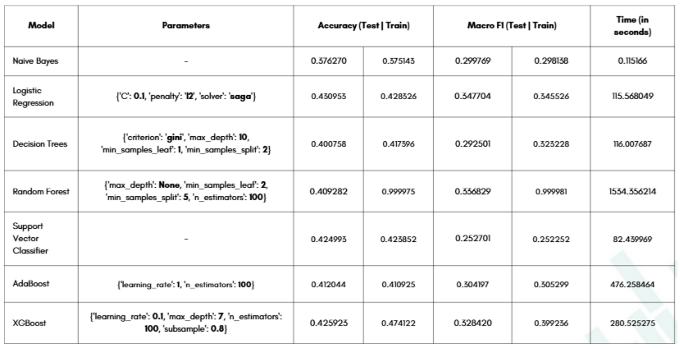
9. Scaling Techniques

9.1. Undersampling

9.2. JL (Johnson-Lindenstrauss Lemma)

Overview: The JL Lemma is a dimensionality reduction technique that guarantees the preservation of pairwise distances between points in a high-dimensional space when mapped to a lower-dimensional space. This is particularly useful for data with high dimensionality.
Dimensionality Reduction: Reduces the number of features while maintaining the geometric structure of the data.
Error Bound: The JL Lemma ensures that the distance between any two points is preserved up to a small error with high probability.

For our implementation we have JL dimentions as dX20
d=39 (number of features in our dataset)
Initial Train Dataset shape: 283889 x 39
Final Train Dataset Shape: 283889 x 20



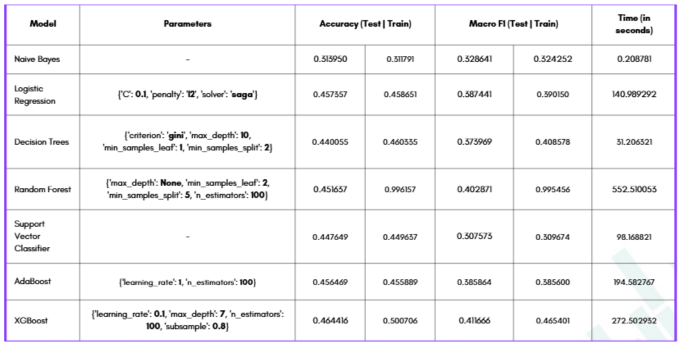
Model	**Parameters**	**Accuracy (Test | Train)**		**Macro F1 (Test | Train)**		**Time (in seconds)**
Naive Bayes	-	0.376270	0.375143	0.299769	0.298138	0.115166
Logistic Regression	{'C': **0.1**, 'penalty': '**l2**', 'solver': '**saga**'}	0.430953	0.428326	0.347704	0.345526	115.568049
Decision Trees	{'criterion': '**gini**', 'max_depth': **10**, 'min_samples_leaf': **1**, 'min_samples_split': **2**}	0.400758	0.417396	0.292501	0.323228	116.007687
Random Forest	{'max_depth': **None**, 'min_samples_leaf': **2**, 'min_samples_split': **5**, 'n_estimators': **100**}	0.409282	0.999975	0.336829	0.999981	1534.356214
Support Vector Classifier	-	0.424993	0.423852	0.252701	0.252252	82.439969
AdaBoost	{'learning_rate': **1**, 'n_estimators': **100**}	0.412044	0.410925	0.304197	0.305299	476.258464
XGBoost	{'learning_rate': **0.1**, 'max_depth': **7**, 'n_estimators': **100**, 'subsample': **0.8**}	0.425923	0.474122	0.328420	0.399236	280.525275

Figure 13. Results of JL lemma

9.3. PCA (Principal Component Analysis)

Dimensionality Reduction: By selecting only the top principal components, PCA reduces the dataset’s dimensions without losing significant information.
Variance Maximization: PCA identifies the directions (principal components) where the data varies the most.

For our implementation we have taken pca components as 4(for numerical features only)
Initial Train Dataset shape: 283889 x 39
Final Train Dataset Shape: 283889 x 37

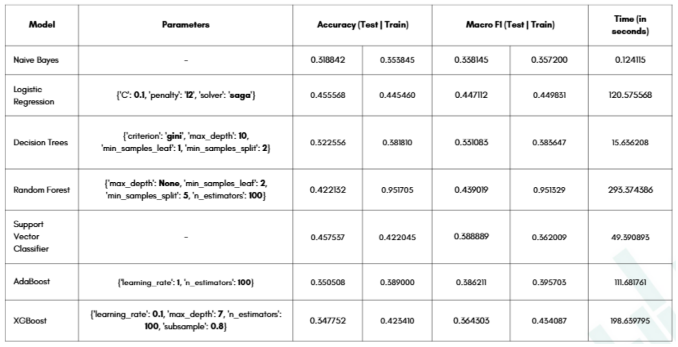


Model	Parameters	Accuracy (Test | Train)		Macro F1 (Test | Train)		Time (in seconds)
Naive Bayes	-	0.313950	0.311791	0.328641	0.324252	0.208781
Logistic Regression	{'C': **0.1**, 'penalty': '**l2**', 'solver': '**saga**'}	0.457357	0.458651	0.387441	0.390150	140.989292
Decision Trees	{'criterion': '**gini**', 'max_depth': **10**, 'min_samples_leaf': **1**, 'min_samples_split': **2**}	0.440055	0.460335	0.373969	0.408578	31.206321
Random Forest	{'max_depth': **None**, 'min_samples_leaf': **2**, 'min_samples_split': **5**, 'n_estimators': **100**}	0.451637	0.996157	0.402871	0.995456	552.510053
Support Vector Classifier	-	0.447649	0.449637	0.307573	0.309674	98.168821
AdaBoost	{'learning_rate': **1**, 'n_estimators': **100**}	0.456469	0.455889	0.385864	0.385600	194.582767
XGBoost	{'learning_rate': **0.1**, 'max_depth': **7**, 'n_estimators': **100**, 'subsample': **0.8**}	0.464416	0.500706	0.411666	0.465401	272.502932

Figure 14. Results of PCA

9.4. CORESETS

Coreset Selection: Selects a representative subset of data points that approximates the original dataset in terms of class distribution and data structure.
Importance Weights: Ensures underrepresented classes are sampled more frequently, maintaining class balance.
Probabilistic Sampling: Uses computed weights to create a probability distribution for selecting samples, reducing bias.
For our implementation
sample_weights= class imbalance
Initial Train Dataset shape: 283889 x 39
Final Train Dataset Shape: 200000 x 39



Model	Parameters	Accuracy (Test | Train)		Macro F1 (Test | Train)		Time (in seconds)
Naive Bayes	-	0.318842	0.353845	0.338145	0.357200	0.124115
Logistic Regression	{'C': **0.1**, 'penalty': '**l2**', 'solver': '**saga**'}	0.455568	0.445460	0.447112	0.449831	120.575568
Decision Trees	{'criterion': '**gini**', 'max_depth': **10**, 'min_samples_leaf': **1**, 'min_samples_split': **2**}	0.322556	0.381810	0.331083	0.383647	15.636208
Random Forest	{'max_depth': **None**, 'min_samples_leaf': **2**, 'min_samples_split': **5**, 'n_estimators': **100**}	0.422132	0.951705	0.439019	0.951329	293.374386
Support Vector Classifier	-	0.457537	0.422045	0.388889	0.362009	49.390893
AdaBoost	{'learning_rate': **1**, 'n_estimators': **100**}	0.350508	0.389000	0.386211	0.395703	111.681761
XGBoost	{'learning_rate': **0.1**, 'max_depth': **7**, 'n_estimators': **100**, 'subsample': **0.8**}	0.347752	0.423410	0.364303	0.434087	198.639795

Figure 15. Results of Coresets

10. CONCLUSION AND FUTURE WORK

We applied seven machine learning models, including Naive Bayes, Logistic Regression, Decision Trees, Random Forest, SVC, AdaBoost, and XGBoost, alongside scaling techniques such as PCA, JL Lemma, coresets, and randomized undersampling. XGBoost emerged as the best-performing model, achieving the highest accuracy while maintaining reasonable computational efficiency. The scaling methods effectively reduced training time with minimal impact on model performance, demonstrating their value for large-scale data.

Future work will focus on further optimizing the models through advanced hyperparameter tuning and experimenting with additional scaling techniques. We also plan to explore DL models and domain-specific feature engineering to enhance overall performance.

11. Github Link

Github Link