Semester Project

(CSE 602) Machine Learning-I

Muhammad Asad ur Rehman (29456)  
Murtaza Hussain (29449)

# Abstract

Class Imbalance problem is a very prevalent problem encountered in Machine Learning domain. In this document we will be trying to solve this problem and assess the performance of different methods to resolve the Class Imbalance Problem using multiple datasets to identify the effectiveness of these methods.

## Datasets

The setting for our attempt to solve these problems would be as follows. We will use the following Datasets:

1. **Credit Card Fraud:** This dataset contains information of multiple transactions which includes fraudulent transactions as well. The ratio of Fraudulent to Legitimate Transactions is 1:99
2. **Cerebral Stroke:** This dataset contains information regarding patients with similar conditions and identifies the patients that did in fact suffer from a Cerebral Stroke. The ratio of Cerebral Stroke to Normal Patients is 10:90
3. **Marketing Slot Bidding Data:** This dataset is for predicting that whether an advertiser should bid on the marketing slots given the relevant factors. The ratio of Conversion is 15:85

As can be seen for all the above datasets the datasets are highly imbalanced with one class dominating the other.

## Base Algorithms

We will use the following Algorithms to assess the performances of the above datasets using different CI Methods:

1. **Logistic Regression:** A Basic Algorithm, which uses linear regression as its base, is fast to train and can model linear problems quite well.
2. **Random Forest:** An ensemble model which uses decision tree as its base with bagging, it is slower to train but is good at modelling complex problems and can be helpful in solving CI problem.
3. **K-Nearest Neighbours:** A Grouping based Classifier, which can be helpful in identifying and grouping similar datapoints together. Is easy to train and can model problem better where a grouping is observed between classes.
4. **Support Vector Machines:** One of the more advanced models, uses the concept of Support Vectors to find the boundary plane for Classification. Is slow to train but models complex problems well especially in cases where there is a definite boundary plane between classes.
5. **Gaussian Naive-Bayes:** A simple Bayesian based model where it takes the assumption that all classes are independent of each other. Is easy to train and can be pretty accurate in many situations especially where the above assumption holds.

## Class Imbalance Methods

For this project we will be employing the following CI methods:

1. **Resampling Methods:** A method to resample the source data to create similar instances of the minority class and hence balancing the class distribution. In particular we will be using the following:
   1. **SMOTE:** Generates synthetic samples uniformly across the feature space for the minority class, hence balancing the Dataset.
   2. **ADASYN:** Generates synthetic samples of the minority class while placing more emphasis on areas where the classification is difficult, hence balancing the Dataset.
2. **Algorithmic Methods:** Using different methods to solve the CI problem which involves modifying the algorithmic settings or even using the algorithms as a selector for a different algorithm. The methods we will be exploring are as follows:
   1. **Class Weight:** Adjust Class Weight such that the algorithm puts more emphasis on learning the minority class as compared to the majority class.
   2. **Bagging:** Use Bootstrap Aggregating or Bagging as the main classifier using the Original Algorithms as the base Estimator.
   3. **Boosting:** Using AdaBoost as the main classifier using the Original Algorithms as the base Estimator.
3. **One Class Training Methods:** Using different methods to train the model on the majority class so that it identifies the minority class as an anomaly or an outlier. We achieved this using two ways:
   1. **Adjusted Algorithms:** By adjusting the original algorithms and training them on the majority dataset (with traces of minority instances) and outputting their prediction probability to set a threshold to identify the minority class.
   2. **Special Algorithms:** Using special algorithms designed to perform One Class Learning such as One Class SVM, Isolation Forest and Local Outlier Factor, to train on the majority class and predict the minority class as whether they belong to the same class or not.

Given the above conditions let’s begin discussing the experimentation and their results.

# Dataset 1: Credit Card Fraud Dataset

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 223 frauds out of 99,776 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.22% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, the original features and more background information about the data was not provided. Features V1 to V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

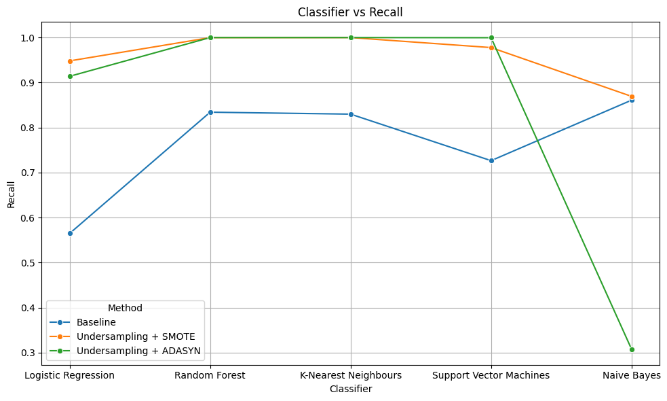
Given the class imbalance ratio, we cannot use accuracy to determine the classification results as a biased classifier would still have an accuracy of 99.78% on this dataset. The suggested metrics to assess the accuracy of the model would be the **Recall (TP / TP + FN)** and **Precision (TP/TP + FP)** for the minority class. As the data is highly sensitive, and an underestimation or missing out on detecting a fraudulent transaction could lead to monetary loss, hence it would be wise to prioritize Recall over Precision and we can afford wrongly classifying legitimate transactions but could not risk to misclassify a fraudulent transaction as legitimate.

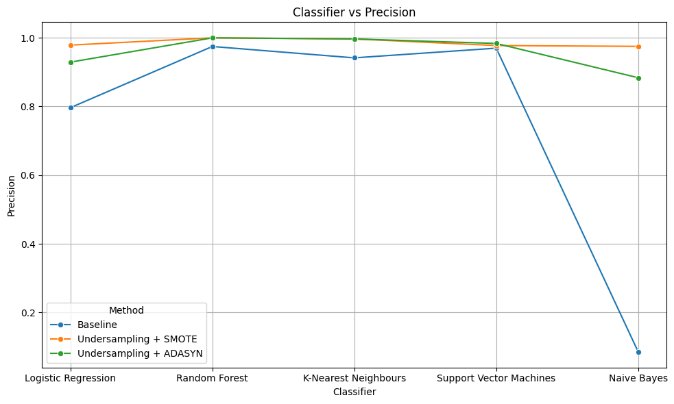
Dataset Link:[*https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud*](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

## Method 1: Sampling Method

### Results

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Classifier | Class 1 Recall | Class 1 Precision |
| Baseline | Logistic Regression | 0.564822 | 0.796502 |
| Baseline | Random Forest | 0.83419 | 0.974603 |
| Baseline | K-Nearest Neighbours | 0.829842 | 0.941499 |
| Baseline | Support Vector Machines | 0.726482 | 0.969722 |
| Baseline | Naive Bayes | 0.86087 | 0.085765 |
| Undersampling + SMOTE | Logistic Regression | 0.94814 | 0.978621 |
| Undersampling + SMOTE | Random Forest | 0.9999 | 0.99974 |
| Undersampling + SMOTE | K-Nearest Neighbours | 1 | 0.996552 |
| Undersampling + SMOTE | Support Vector Machines | 0.97776 | 0.977332 |
| Undersampling + SMOTE | Naive Bayes | 0.86936 | 0.97498 |
| Undersampling + ADASYN | Logistic Regression | 0.913934 | 0.929002 |
| Undersampling + ADASYN | Random Forest | 0.99998 | 0.99976 |
| Undersampling + ADASYN | K-Nearest Neighbours | 1 | 0.996472 |
| Undersampling + ADASYN | Support Vector Machines | 0.9997 | 0.983451 |
| Undersampling + ADASYN | Naive Bayes | 0.307513 | 0.883669 |





### Analysis

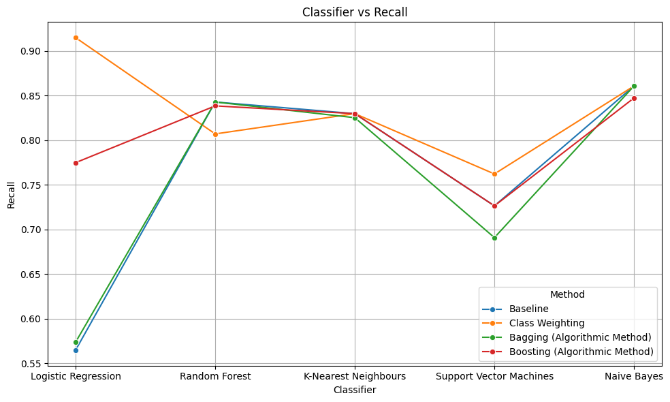
* **Baseline**: To handle class imbalance, the baseline models show varying performance. Random Forest, K-Nearest Neighbours, and SVM performed relatively well and had a better recall and precision for class 1. Naive Bayes, has high recall but very low precision, which indicates that it is underfitting and classifying a large chunk of Class 0 instances as Class 1.
* **Undersampling + SMOTE**: This technique, significantly improves the results, however, Random Forest and KNN have a perfect recall and precision indicating that the models have been over-fitted to the problem. A possible reason for overfitting of the model can be that the data generated through SMOTE created an artificial pattern in the sample which was not existent in the original dataset.
* **Undersampling + ADASYN**: This technique indicates an even better fit for most cases, although the improvement varies across classifiers. Random Forest and K-Nearest Neighbours again out perform other models and indicate a possible overfitting. A possible reason for overfitting can be a bias in the synthetic data produced by the ADASYN over sampler. Naive Bayes recall plummets low after applying this technique indicating that Naive Bayes might not be the right classifier to model this problem.

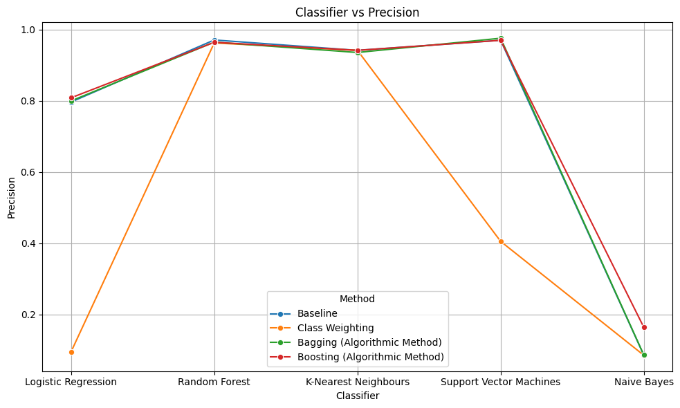
In conclusion, the best solution using Method 1 for this dataset would be Logistic Regression as it had a decent Recall and Precision for both the techniques. Secondary to this can be SVM as it has a decent Recall and Precision for both techniques. Random Forest and KNN seem to overfit the problem and Naive Bayes tends to underfit the Dataset and hence is struggling to perform well.

## Method 2: Algorithmic Method

### Results

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Classifier | Class 1 Recall | Class 1 Precision |
| Baseline | Logistic Regression | 0.564822 | 0.796502 |
| Baseline | Random Forest | 0.842885 | 0.970621 |
| Baseline | K-Nearest Neighbours | 0.829842 | 0.941499 |
| Baseline | Support Vector Machines | 0.726482 | 0.969722 |
| Baseline | Naive Bayes | 0.86087 | 0.085765 |
| Class Weighting | Logistic Regression | 0.91502 | 0.0953 |
| Class Weighting | Random Forest | 0.807115 | 0.962283 |
| Class Weighting | K-Nearest Neighbours | 0.829842 | 0.941499 |
| Class Weighting | Support Vector Machines | 0.762253 | 0.405465 |
| Class Weighting | Naive Bayes | 0.86087 | 0.085765 |
| Bagging (Algorithmic Method) | Logistic Regression | 0.573715 | 0.799597 |
| Bagging (Algorithmic Method) | Random Forest | 0.842885 | 0.964522 |
| Bagging (Algorithmic Method) | K-Nearest Neighbours | 0.825296 | 0.935467 |
| Bagging (Algorithmic Method) | Support Vector Machines | 0.690909 | 0.975556 |
| Bagging (Algorithmic Method) | Naive Bayes | 0.86087 | 0.086355 |
| Boosting (Algorithmic Method) | Logistic Regression | 0.774901 | 0.808315 |
| Boosting (Algorithmic Method) | Random Forest | 0.838538 | 0.964175 |
| Boosting (Algorithmic Method) | K-Nearest Neighbours | 0.829842 | 0.941499 |
| Boosting (Algorithmic Method) | Support Vector Machines | 0.726482 | 0.969722 |
| Boosting (Algorithmic Method) | Naive Bayes | 0.847431 | 0.164062 |





### Analysis

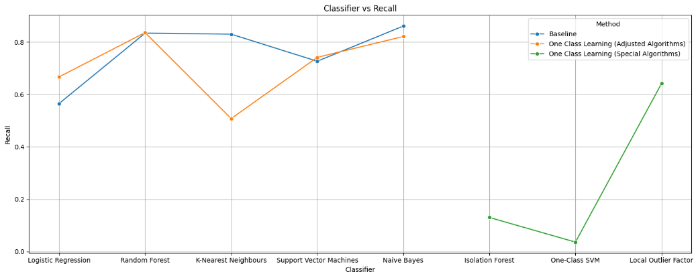
* **Baseline**: To handle class imbalance, the baseline models show varying performance. Random Forest, K-Nearest Neighbours, and SVM performed relatively well and had a better recall and precision for class 1. Naive Bayes, has high recall but very low precision, which indicates that it is underfitting and classifying a large chunk of Class 0 instances as Class 1.
* **Class Weighting:** Class Weighting method improved recall for Logistic Regression and SVM but the Precision dropped significantly hence these models became underfitted. For Naïve Bayes and Random Forest, no significant changes were observed.
* **Bagging:** In case of Bagging, no significant change was observed and all the models were similar to their Baseline performance. An exception to this would be SVM as Recall decreased from Baseline.
* **Boosting:** Boosting was also not much beneficial for the algorithms and the Recall and Precision did not significantly change from Baseline. An exception to this would be Logistic Regression where Recall increased significantly with no change in Precision.

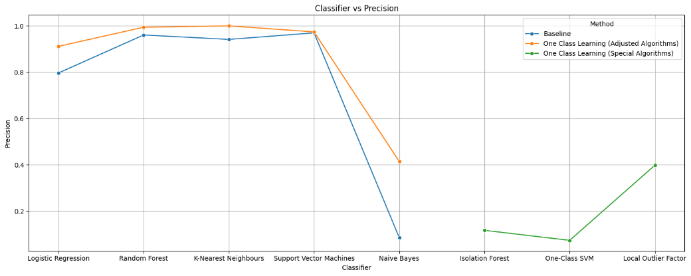
To conclude, it can be said that Baseline performance of Random Forest was best across all techniques. A secondary classifier to Random Forest would be KNN as it also had similar results across different techniques. Boosting, when applied to Logistic Regression did significantly improve the results, however, they were still not as good as Random Forest or KNN. SVM had a good Precision but a sub-standard recall which indicates that it might not be the most ideal classifier to model the problem. Same is the case with Naive Bayes which struggled consistently with lower precision, showing a high rate of false predictions.

## Method 3: (One-Class) Learning Method

### Results

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Classifier | Class 1 Recall | Class 1 Precision |
| Baseline | Logistic Regression | 0.564822 | 0.796502 |
| Baseline | Random Forest | 0.833794 | 0.960668 |
| Baseline | K-Nearest Neighbours | 0.829842 | 0.941499 |
| Baseline | Support Vector Machines | 0.726482 | 0.969722 |
| Baseline | Naive Bayes | 0.86087 | 0.085765 |
| One Class Learning (Adjusted Algorithms) | Logistic Regression | 0.666667 | 0.911565 |
| One Class Learning (Adjusted Algorithms) | Random Forest | 0.835821 | 0.994083 |
| One Class Learning (Adjusted Algorithms) | K-Nearest Neighbours | 0.507463 | 1 |
| One Class Learning (Adjusted Algorithms) | Support Vector Machines | 0.741294 | 0.973856 |
| One Class Learning (Adjusted Algorithms) | Naive Bayes | 0.820896 | 0.414573 |
| One Class Learning (Special Algorithms) | Isolation Forest | 0.130045 | 0.117409 |
| One Class Learning (Special Algorithms) | One-Class SVM | 0.035874 | 0.074074 |
| One Class Learning (Special Algorithms) | Local Outlier Factor | 0.641256 | 0.398329 |





### Analysis

* **Baseline**: To handle class imbalance, the baseline models show varying performance. Random Forest, K-Nearest Neighbours, and SVM performed relatively well and had a better recall and precision for class 1. Naive Bayes, has high recall but very low precision, which indicates that it is underfitting and classifying a large chunk of Class 0 instances as Class 1.
* **One Class Learning (Adjusted Algorithms):** The Adjusted Algorithms showed a better Precision as compared to their Baseline Counterparts in most case, and had a similar or better Recall as well with the exception of KNN whose Recall decreased significantly with the Adjustment.
* **One Class Learning (Special Algorithms):** Local Outlier Factor performed better as compared to Isolation Forest and One-Class SVM with a higher recall and precision. But even LOF had worse precision and recall than most of the Baseline models.

It can be concluded that Random Forest performed consistently well showcasing the robustness in handling class imbalance. Logistic Regression shown improvement with adjusted algorithms, showing its adaptability to class imbalance handling techniques. Naive Bayes continues to struggle with precision across different methods, showing signs that it is not the right model for this problem. In Specialized one-class learning algorithms, Isolation Forest and One-Class SVM performed poorly on this dataset, indicating that they may not be suitable for this specific classification task.

# Dataset 2: Cerebral Stroke Dataset

Written code to solve the prevalent problem of imbalanced dataset, where one class dominates the dataset compared to the other. Such is the case for the following dataset related to Strokes. A stroke also known as a cerebrovascular accident (CVA) happens when part of the brain loses its blood supply and the part of the body that the blood-deprived brain cells control stops working, it is a medical emergency because strokes can lead to death or permanent disability. There are opportunities to treat strokes but those treatments need to be started in the first few hours after the signs of a stroke begin. Those signs and physical condition of the person are the features here and the target column which is imbalanced is of **‘stroke’** which tells if it’s true or false represented by 1s and 0s.

We have used the following methods to resolve Class Imbalance:

1. Random Under Sampling
2. Algorithmic Methods (Using Random Forest as well as modifying Class Weights)
3. Anomaly Detection Method

And we have used the following 5 Algorithms/models to draw a comparison between different methods:

1. Logistic Regression
2. K-Nearest Neighbours (KNN)
3. Random Forest
4. Support Vector Machines (SVM)
5. Naive Bayes (Gaussian)

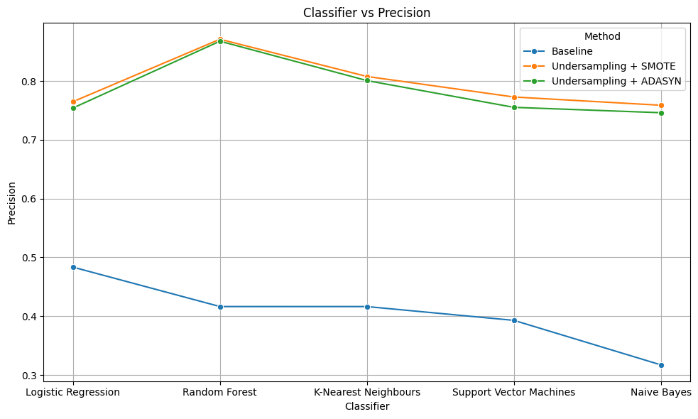
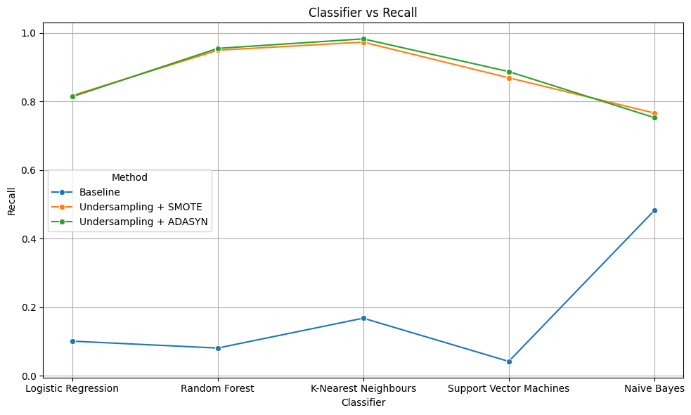
Dataset Link:*https://www.kaggle.com/datasets/shashwatwork/cerebral-stroke-predictionimbalaced-dataset*

## Sampling Method:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Classifier | Class 1 Recall | Class 1 Precision |
| Baseline | Logistic Regression | 0.101082 | 0.483481 |
| Baseline | Random Forest | 0.080913 | 0.416445 |
| Baseline | K-Nearest Neighbours | 0.168029 | 0.416405 |
| Baseline | Support Vector Machines | 0.041971 | 0.392949 |
| Baseline | Naive Bayes | 0.482067 | 0.31724 |
| Undersampling + SMOTE | Logistic Regression | 0.817176 | 0.764742 |
| Undersampling + SMOTE | Random Forest | 0.949427 | 0.870942 |
| Undersampling + SMOTE | K-Nearest Neighbours | 0.973092 | 0.807674 |
| Undersampling + SMOTE | Support Vector Machines | 0.869084 | 0.772736 |
| Undersampling + SMOTE | Naive Bayes | 0.766412 | 0.758717 |
| Undersampling + ADASYN | Logistic Regression | 0.813921 | 0.753992 |
| Undersampling + ADASYN | Random Forest | 0.954791 | 0.867632 |
| Undersampling + ADASYN | K-Nearest Neighbours | 0.982555 | 0.800644 |
| Undersampling + ADASYN | Support Vector Machines | 0.887263 | 0.755161 |
| Undersampling + ADASYN | Naive Bayes | 0.752954 | 0.746006 |

* **Baseline**: Performance varies across all classifiers, with relatively low recall and precision. Naive Bayes stands out with moderate recall and precision, indicating better performance compared to other baseline models, in this case.
* **Undersampling + SMOTE**: Significant improvement in both recall and precision across most classifiers compared to the baseline. Random Forest, K-Nearest Neighbours, and Support Vector Machines shows notable improvement with this technique.
* **Undersampling + ADASYN**: This also resulted in considerable improvement in both recall and precision across most classifiers compared to the baseline. Random Forest, K-Nearest Neighbours, and Support Vector Machines shows notable improvement with this technique.

In conclusion, baseline models performed poorly, Naive Bayes was relatively better in performance compared to other baseline models. Undersampling with SMOTE and ADASYN effectively worked with class imbalance issues, leading to significant improvements in the performance of classifiers. Random Forest, K-Nearest Neighbours, and Support Vector Machines consistently showed notable improvements with both SMOTE and ADASYN techniques. Combining undersampling with techniques like SMOTE or ADASYN leads to noticeable improvements in the performance, it starts handling imbalanced datasets properly and identifies instances of the minority class with good precision.

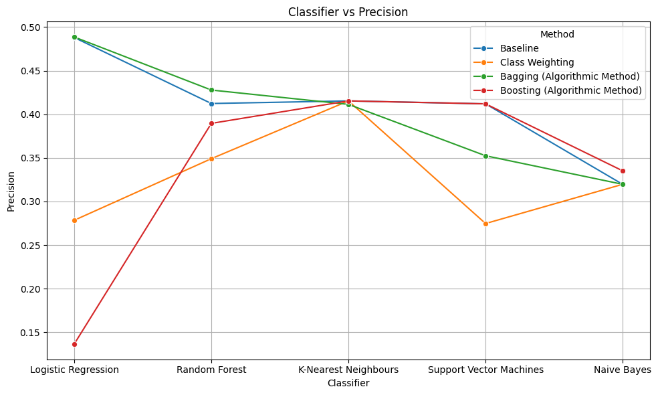
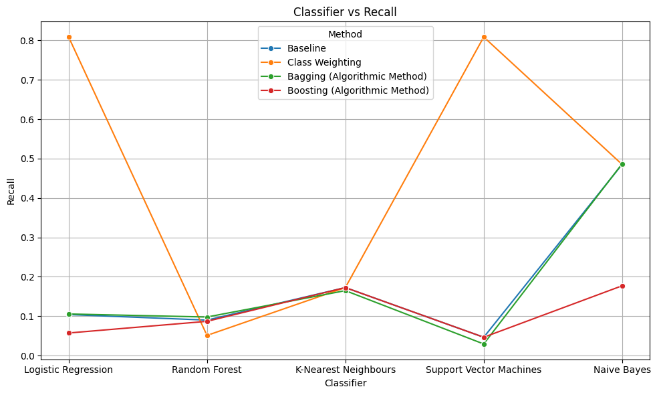


## Algorithmic Method:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Classifier | Class 1 Recall | Class 1 Precision |
| Baseline | Logistic Regression | 0.104191 | 0.488081 |
| Baseline | Random Forest | 0.090177 | 0.412368 |
| Baseline | K-Nearest Neighbours | 0.172735 | 0.41536 |
| Baseline | Support Vector Machines | 0.046645 | 0.412 |
| Baseline | Naive Bayes | 0.485308 | 0.319683 |
| Class Weighting | Logistic Regression | 0.808769 | 0.278446 |
| Class Weighting | Random Forest | 0.051296 | 0.349087 |
| Class Weighting | K-Nearest Neighbours | 0.172735 | 0.41536 |
| Class Weighting | Support Vector Machines | 0.808781 | 0.274694 |
| Class Weighting | Naive Bayes | 0.485308 | 0.319683 |
| Bagging (Algorithmic Method) | Logistic Regression | 0.105753 | 0.488664 |
| Bagging (Algorithmic Method) | Random Forest | 0.097965 | 0.427822 |
| Bagging (Algorithmic Method) | K-Nearest Neighbours | 0.164983 | 0.411372 |
| Bagging (Algorithmic Method) | Support Vector Machines | 0.029542 | 0.352448 |
| Bagging (Algorithmic Method) | Naive Bayes | 0.48687 | 0.319696 |
| Boosting (Algorithmic Method) | Logistic Regression | 0.057437 | 0.136534 |
| Boosting (Algorithmic Method) | Random Forest | 0.08704 | 0.389571 |
| Boosting (Algorithmic Method) | K-Nearest Neighbours | 0.172735 | 0.41536 |
| Boosting (Algorithmic Method) | Support Vector Machines | 0.046645 | 0.412 |
| Boosting (Algorithmic Method) | Naive Bayes | 0.177277 | 0.33516 |

* **Baseline**: Performance varies across all classifiers, with low recall and precision. Naive Bayes shows relatively better performance compared to other baseline models.
* **Class Weighting**: Logistic Regression and SVM show improvements in recall, Random Forest and K-Nearest Neighbours doesn’t show any improvement.
* **Bagging (Algorithmic Method)**: There is no significant improvement in recall or precision compared to the baseline, after using this technique.
* **Boosting (Algorithmic Method)**: There is no significant improvement in recall or precision compared to the baseline.

In conclusion, baseline models performed poorly, Naive Bayes shows relatively better performance compared to other baseline models. Class weighting, bagging, and boosting techniques did not lead to substantial improvements either. Logistic Regression and SVM classifiers showed some improvements in recall with class weighting. So, the chosen techniques did not effectively address the class imbalance issue on this dataset.

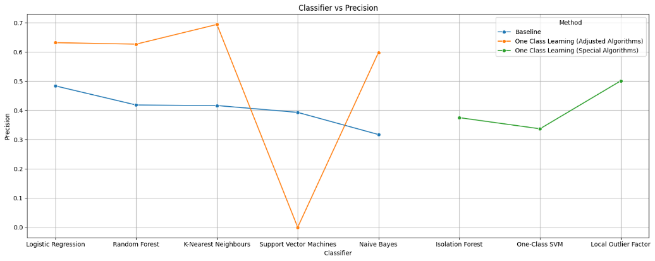
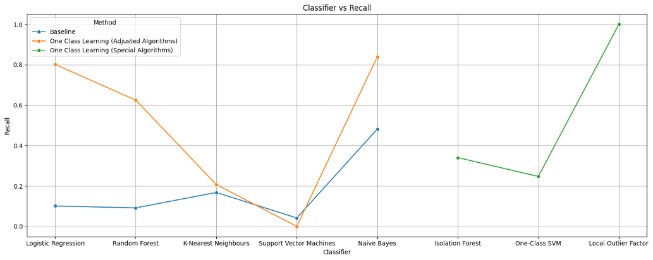


## Anomaly (One-Class) Method:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Classifier | Class 1 Recall | Class 1 Precision |
| Baseline | Logistic Regression | 0.101082 | 0.483481 |
| Baseline | Random Forest | 0.091803 | 0.418171 |
| Baseline | K-Nearest Neighbours | 0.168029 | 0.416405 |
| Baseline | Support Vector Machines | 0.041971 | 0.392949 |
| Baseline | Naive Bayes | 0.482067 | 0.31724 |
| One Class Learning (Adjusted Algorithms) | Logistic Regression | 0.801382 | 0.631293 |
| One Class Learning (Adjusted Algorithms) | Random Forest | 0.625216 | 0.626298 |
| One Class Learning (Adjusted Algorithms) | K-Nearest Neighbours | 0.207254 | 0.693642 |
| One Class Learning (Adjusted Algorithms) | Support Vector Machines | 0 | 0 |
| One Class Learning (Adjusted Algorithms) | Naive Bayes | 0.839378 | 0.597786 |
| One Class Learning (Special Algorithms) | Isolation Forest | 0.340591 | 0.375 |
| One Class Learning (Special Algorithms) | One-Class SVM | 0.247278 | 0.336864 |
| One Class Learning (Special Algorithms) | Local Outlier Factor | 1 | 0.500779 |

* **Baseline:** Performance was poor across all classifiers, Naive Bayes relatively performed better compared to other classifiers.
* **One Class Learning (Adjusted Algorithms):** Logistic Regression, Random Forest, K-Nearest Neighbours and Naive Bayes show improvements in both recall and precision compared to the baseline but Support Vector Machines show 0 which might be due to nature of data.
* **One Class Learning (Special Algorithms):** Isolation Forest, One-Class SVM, and Local Outlier Factor demonstrate improvements in either recall or precision compared to the baseline. Local Outlier Factor stands out with perfect recall but moderate precision, it can be either overfitted or due to nature of the data.

In conclusion, the baseline models show poor performance, Naive Bayes shows relatively better precision, while K-Nearest Neighbours has higher recall but lower precision. One Class Learning with adjusted algorithms shown promising improvements in both recall and precision, particularly with Logistic Regression, Random Forest, and Naive Bayes classifiers. And specialized algorithms, like Local Outlier Factor can effectively identify outliers in imbalanced datasets, leading to perfect recall in this scenario with moderate precision, which can be due to nature of data. So, techniques like One Class Learning with adjusted algorithms or specialized algorithms can lead to significant improvements in the performance of classifiers.



# Dataset 3: Bidding Dataset

Written code to solve the prevalent problem of imbalanced dataset, where one class dominates the dataset compared to the other. Such is the case for the following dataset for Bidding. This is real-time bidding data which is used to predict if an advertiser should bid for a marketing slot or not. Explanatory variables are things like browser, operation system or time of the day the user is online, marketplace, identifiers were traded on earlier, etc. The column **‘convert’** is the target in which 1 is when the person clicked on the ad and 0 if person did not. We have used the following methods to resolve Class Imbalance:

1. Random Under Sampling
2. Algorithmic Methods (Using Random Forest as well as modifying Class Weights)
3. Anomaly Detection Method

And we have used the following 5 Algorithms/models to draw a comparison between different methods:

1. Logistic Regression
2. K-Nearest Neighbours (KNN)
3. Random Forest
4. Support Vector Machines (SVM)
5. Naive Bayes (Gaussian)

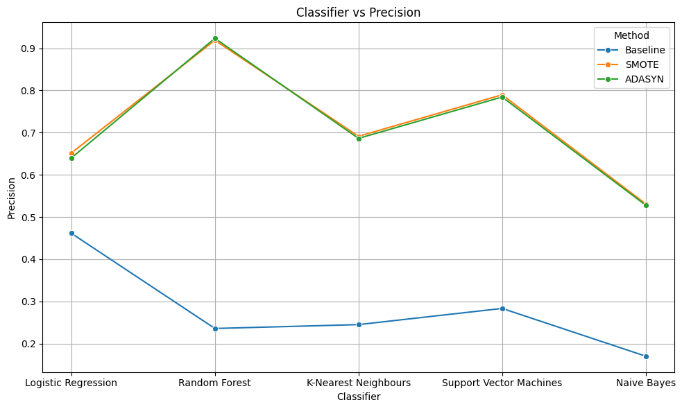
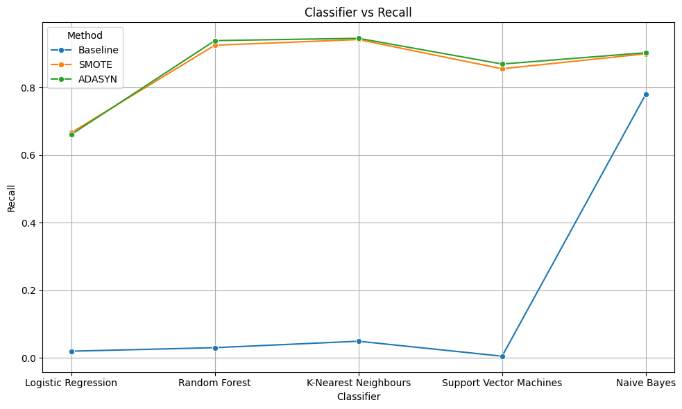
Dataset Link:*https://www.kaggle.com/datasets/zurfer/rtb*

## Sampling Method:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Classifier | Class 1 Recall | Class 1 Precision |
| Baseline | Logistic Regression | 0.020255 | 0.461523 |
| Baseline | Random Forest | 0.030696 | 0.235837 |
| Baseline | K-Nearest Neighbours | 0.04972 | 0.244985 |
| Baseline | Support Vector Machines | 0.005521 | 0.283333 |
| Baseline | Naive Bayes | 0.779991 | 0.170173 |
| SMOTE | Logistic Regression | 0.666361 | 0.651435 |
| SMOTE | Random Forest | 0.924843 | 0.918829 |
| SMOTE | K-Nearest Neighbours | 0.941499 | 0.691293 |
| SMOTE | Support Vector Machines | 0.855272 | 0.789622 |
| SMOTE | Naive Bayes | 0.89945 | 0.531015 |
| ADASYN | Logistic Regression | 0.660819 | 0.639713 |
| ADASYN | Random Forest | 0.93845 | 0.923488 |
| ADASYN | K-Nearest Neighbours | 0.945245 | 0.686362 |
| ADASYN | Support Vector Machines | 0.869195 | 0.784353 |
| ADASYN | Naive Bayes | 0.902655 | 0.527931 |

* **Baseline:** The performance is generally poor across all classifiers, with very low recall and precision. Naive Bayes stands out with high recall but low precision.
* **SMOTE:** Most classifiers in SMOTE significantly improves both recall and precision compared to the baseline. Random Forest and K-Nearest Neighbours particularly show high performance after implementing SMOTE.
* **ADASYN:** In ADASYN, classifiers improved both recall and precision. Random Forest and K-Nearest Neighbours again show notable improvement.

In Conclusion, the baseline models show poor performance which indicates a significant imbalance in the dataset that needs to be addressed. Both SMOTE and ADASYN shown improvement in handling class imbalance. Random Forest and K-Nearest Neighbours consistently performed well after applying these techniques, they correctly identified instances of the minority class while maintaining high precision. Naive Bayes shows high recall but low precision across all methods.

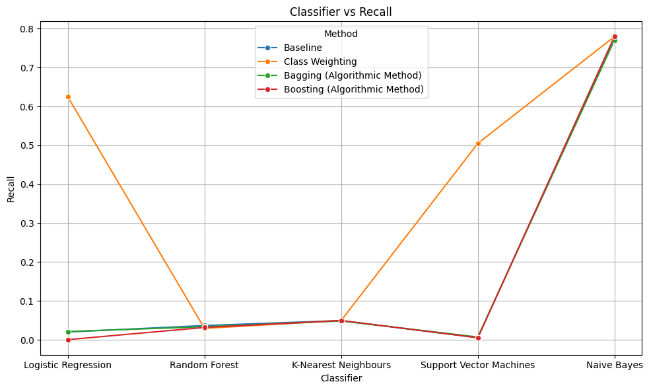


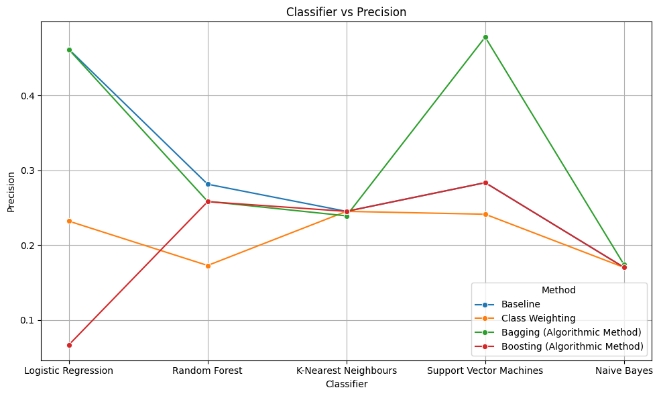
## Algorithmic Method:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Classifier | Class 1 Recall | Class 1 Precision |
| Baseline | Logistic Regression | 0.020255 | 0.461523 |
| Baseline | Random Forest | 0.036832 | 0.281264 |
| Baseline | K-Nearest Neighbours | 0.04972 | 0.244985 |
| Baseline | Support Vector Machines | 0.005521 | 0.283333 |
| Baseline | Naive Bayes | 0.779991 | 0.170173 |
| Class Weighting | Logistic Regression | 0.62495 | 0.231936 |
| Class Weighting | Random Forest | 0.029465 | 0.172701 |
| Class Weighting | K-Nearest Neighbours | 0.04972 | 0.244985 |
| Class Weighting | Support Vector Machines | 0.505231 | 0.241152 |
| Class Weighting | Naive Bayes | 0.779991 | 0.170173 |
| Bagging (Algorithmic Method) | Logistic Regression | 0.021482 | 0.461232 |
| Bagging (Algorithmic Method) | Random Forest | 0.033763 | 0.258352 |
| Bagging (Algorithmic Method) | K-Nearest Neighbours | 0.048498 | 0.238963 |
| Bagging (Algorithmic Method) | Support Vector Machines | 0.007366 | 0.477778 |
| Bagging (Algorithmic Method) | Naive Bayes | 0.770864 | 0.173883 |
| Boosting (Algorithmic Method) | Logistic Regression | 0.000613 | 0.066667 |
| Boosting (Algorithmic Method) | Random Forest | 0.031919 | 0.257922 |
| Boosting (Algorithmic Method) | K-Nearest Neighbours | 0.04972 | 0.244985 |
| Boosting (Algorithmic Method) | Support Vector Machines | 0.005521 | 0.283333 |
| Boosting (Algorithmic Method) | Naive Bayes | 0.779991 | 0.170173 |

* **Baseline:** Performance is poor across all classifiers, with very low recall and precision. Naive Bayes stands out with high recall but low precision.
* **Class Weighting:** It shows mixed results across classifiers. Logistic Regression and SVM show improvements in recall, Random Forest and K-Nearest Neighbours show further decline in recall and precision.
* **Bagging (Algorithmic Method):** In Bagging there is no significant improvement in recall or precision compared to the baseline.
* **Boosting (Algorithmic Method):** In Boosting there is no significant improvement in recall or precision compared to the baseline.

In conclusion, baseline models show poor performance, and Class weighting, bagging and boosting techniques did not show any significant improvement in the performance either. Naive Bayes consistently shows high recall but low precision across all methods. The chosen techniques; class weighting, bagging, and boosting does not address the class imbalance issue properly on this dataset.



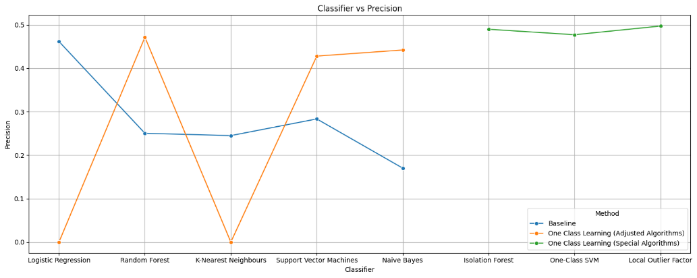
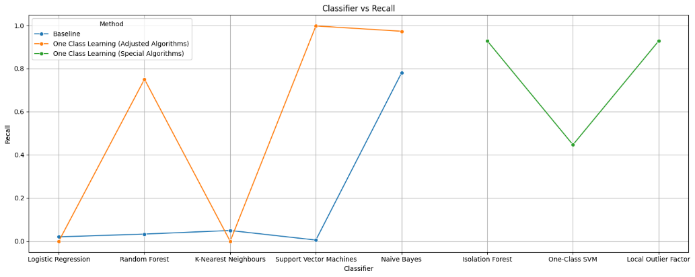


## Anomaly (One-Class) Method:

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Classifier | Class 1 Recall | Class 1 Precision |
| Baseline | Logistic Regression | 0.020255 | 0.461523 |
| Baseline | Random Forest | 0.033146 | 0.25035 |
| Baseline | K-Nearest Neighbours | 0.04972 | 0.244985 |
| Baseline | Support Vector Machines | 0.005521 | 0.283333 |
| Baseline | Naive Bayes | 0.779991 | 0.170173 |
| One Class Learning (Adjusted Algorithms) | Logistic Regression | 0 | 0 |
| One Class Learning (Adjusted Algorithms) | Random Forest | 0.750511 | 0.471318 |
| One Class Learning (Adjusted Algorithms) | K-Nearest Neighbours | 0 | 0 |
| One Class Learning (Adjusted Algorithms) | Support Vector Machines | 0.997273 | 0.427903 |
| One Class Learning (Adjusted Algorithms) | Naive Bayes | 0.972733 | 0.441932 |
| One Class Learning (Special Algorithms) | Isolation Forest | 0.928177 | 0.489637 |
| One Class Learning (Special Algorithms) | One-Class SVM | 0.4469 | 0.477064 |
| One Class Learning (Special Algorithms) | Local Outlier Factor | 0.927563 | 0.497039 |

* **Baseline:** Performance is generally poor across all classifiers, with very low recall and precision. Naive Bayes stands out with high recall but low precision.
* **One Class Learning (Adjusted Algorithms):** Random Forest and SVM shows improvement in both recall and precision compared to the baseline. Logistic Regression and K-Nearest Neighbours show no improvement, likely due to something related to the nature of data.
* **One Class Learning (Special Algorithms):** Isolation Forest, One-Class SVM, and Local Outlier Factor shows improvement in both recall and precision compared to the baseline.

In conclusion, baseline models performed poorly, in terms of recall and precision. One Class Learning, adjusted algorithms shown improvements in both recall and precision, especially Random Forest and SVM classifiers. Specialized algorithms like Isolation Forest, One-Class SVM, and Local Outlier Factor also performed well on this dataset, indicating their effectiveness in identifying anomalies or outliers on this imbalanced dataset. Such significant improvements in the performance of classifiers will particularly lead to correct identification of instances of the minority class while maintaining high precision.

****