Cairo University

Faculty of Engineering
Cairo University

Machine Learning

#CMP4040

Project Report

Team 18

Submitted to:

Eng. Mohamed Shawky

Submitted By:

NAME	SEC	BN	ID
Sarah Elzayat	1	29	9202618
Abdelrahman Fathy	2	2	9202846
Yasmine Ashraf Ghanem	2	37	9203707
Yasmin Abdullah Nasser	2	38	9203717

Team Contribution

NAME	Contribution
Sarah Elzayat	Logistic Regression & SVM
Abdelrahman Fathy	Data Analysis & AdaBoost
Yasmine Ashraf Ghanem	Data Analysis & AdaBoost
Yasmin Abdullah Nasser	Logistic Regression & SVM

Selected Problem: Diabetes Health Indicators

- *Definition:*
  The dataset consists of health indicators collected from the Behavioral Risk Factor Surveillance System, focusing on diabetes. It includes a wide range of variables, such as health behaviors, health outcomes, and the use of preventive services. This dataset offers a comprehensive view of factors that may influence diabetes conditions in individuals.

- *Motivation:*
  Diabetes is a growing health concern worldwide, leading to various complications and affecting millions of people's quality of life. Understanding the factors that contribute to diabetes can help in early detection, prevention, and management strategies. By analyzing this dataset, we aim to uncover patterns and relationships between different health indicators and diabetes status. Insights derived from this analysis could inform public health policies, individual lifestyle choices, and medical interventions aimed at reducing the incidence of diabetes and improving the lives of those living with the condition. Specifically, we could identify high-risk groups based on demographics or behaviors, determine key factors contributing to diabetes, and suggest targeted interventions to mitigate these risks.

Evaluation Metrics

Selected metrics are:

1. *Accuracy:*
   The fraction of predictions our model got right. It's a good starting point but doesn't work well with imbalanced datasets.

2. *Precision and Recall:*
   Precision measures the accuracy of positive predictions, while recall measures the fraction of positives that were correctly identified.

3. *F1 Score:*
   The harmonic mean of precision and recall. It combines both metrics into one, balancing their contributions.

4. *ROC-AUC:*
   The area under the Receiver Operating Characteristic curve. It measures the model's ability to distinguish between classes.

Those metrics are subject to modification and change depending on the final analysis.

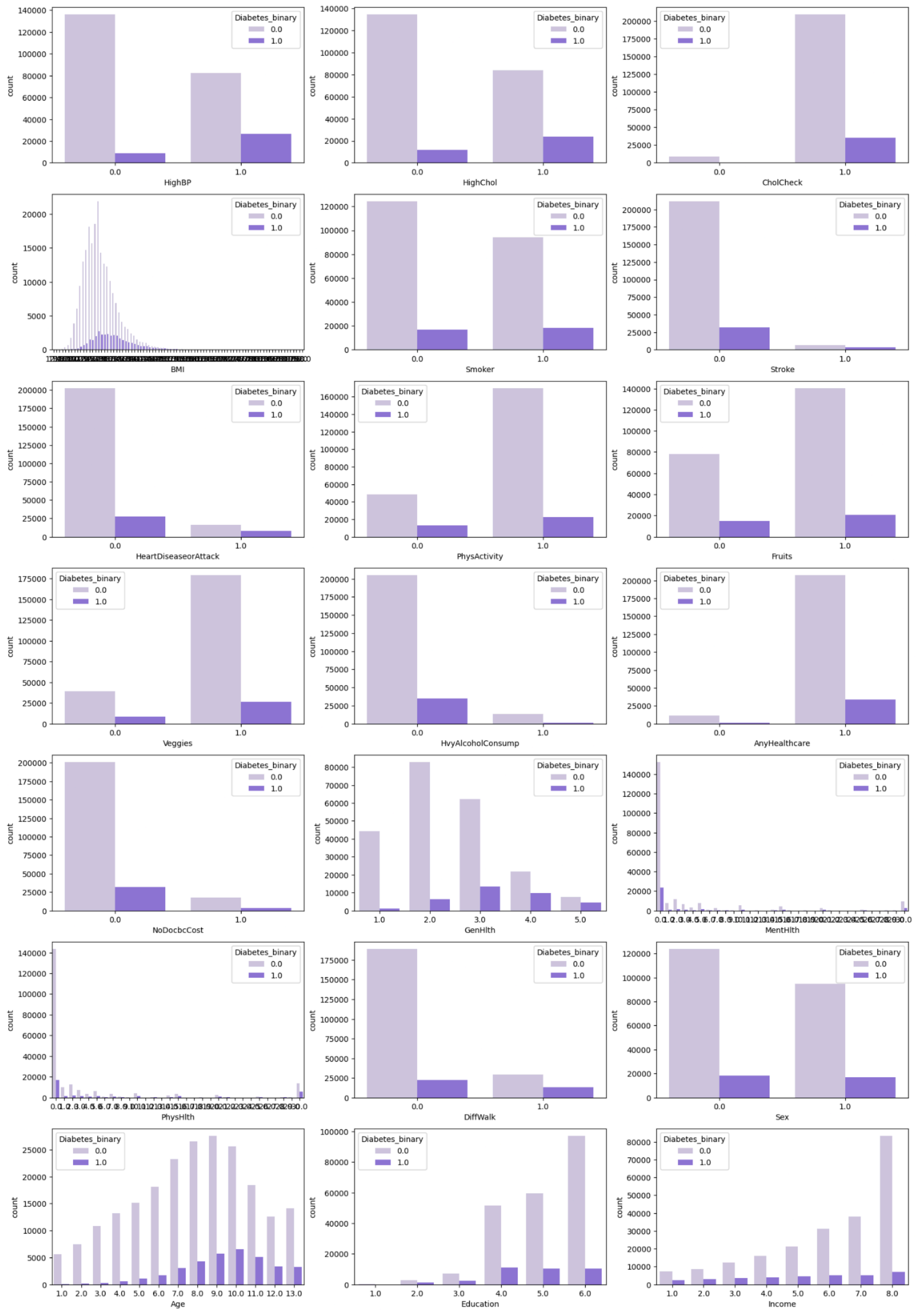
References

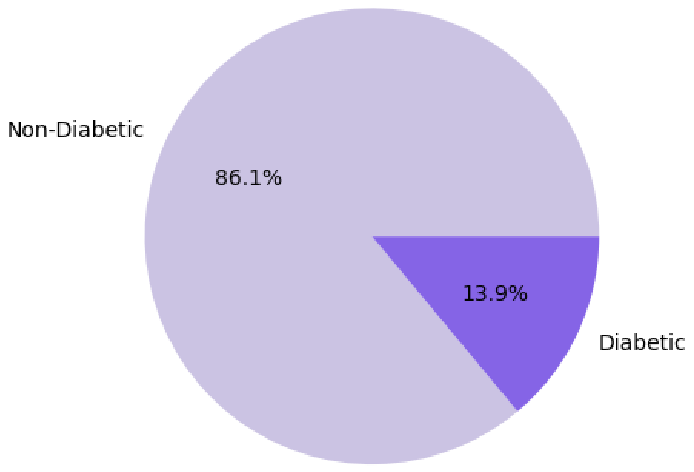
Dataset: *Diabetes Health Indicators Dataset (kaggle.com)*
CDC Diabetes Health Indicators - UCI Machine Learning Repository

- *More information:*
  Features: *21*
  Instances: *253,680*

Data Analysis

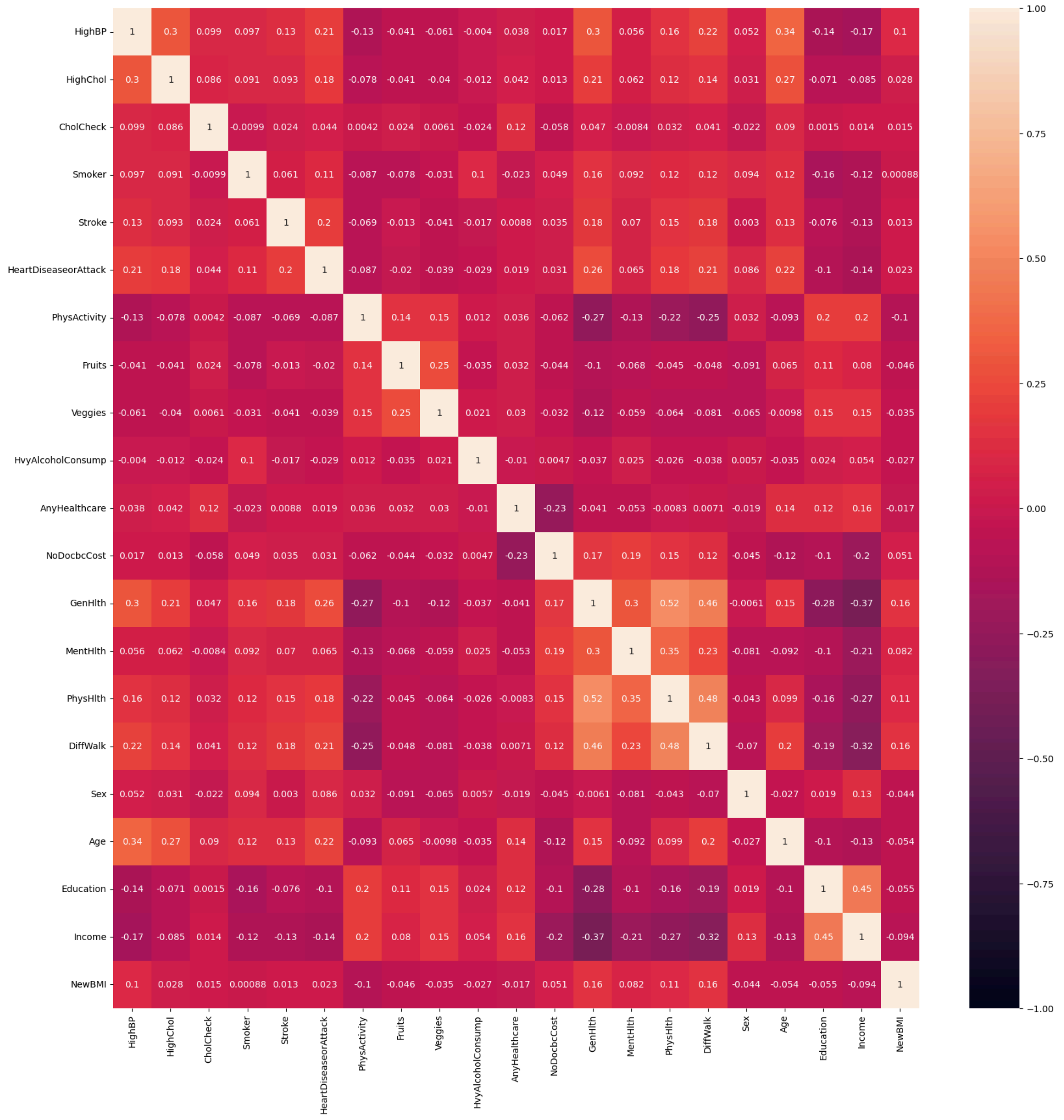
Data Distribution:





- The pie chart and count plots are to show the distribution of the data across the dataset to get a better understanding of what we are dealing with. The statistical analysis along with the graphs clearly show that we are dealing with an imbalanced dataset where the majority class is the 0 or non-diabetic class.
- After researching, we found that most applications that deal with data imbalance handle using either oversampling techniques, undersampling techniques, or adjust the class weights during model training.

Data Correlation:



- The correlation matrix is used to show the correlation between each feature and the other.
- The numbers show that the data is not strongly correlated as the maximum correlation in the dataset is 0.52
- The matrix doesn't show the correlation between the features and the actual classification (Diabetic). This can be shown in the histograms.

ZeroR Algorithm:

The ZeroR algorithm simply gives a baseline accuracy based on the maximum classified class in the original training set which is 86.1% for the non-diabetic class shown in the pie chart.

Handling Dataset Imbalance:

Since the dataset is imbalanced and the first round of training the classifiers resulted in an accuracy of 0.86 however the F1 Score was 0.23 so we researched how to handle imbalanced datasets which were quite common in medical diagnosis problems like this one. We chose the Synthetic Minority Oversampling TEchniques (SMOTE) which adds synthetic samples to the minority class. This was chosen over the undersampling techniques since undersampling could lead to loss of information. The results of SMOTE on the models used will be discussed later for each model.

Model Selection

Note: In this problem we are concerned more with accuracy and recall. This is because Recall is important when missing positive instances (false negatives) is costly, and we want to maximize the number of true positive predictions. Misdiagnosing someone with diabetes can cause severe health problem issues

1. Logistic Regression

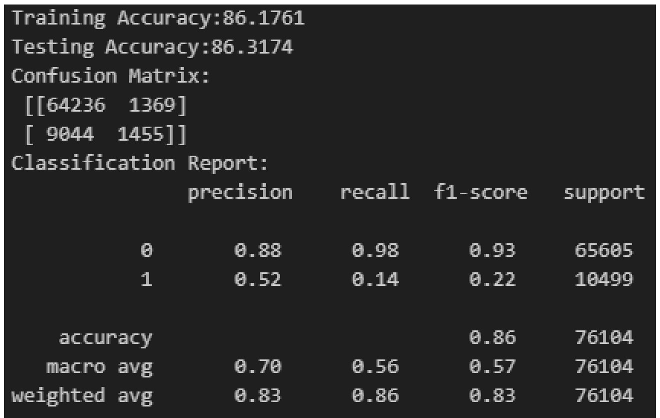
This Classifier is a statistical method used for binary classification. It uses the logistic function (sigmoid) to model the probability that a given input belongs to a particular class. This is done by calculating maximum likelihood.

Parameters:

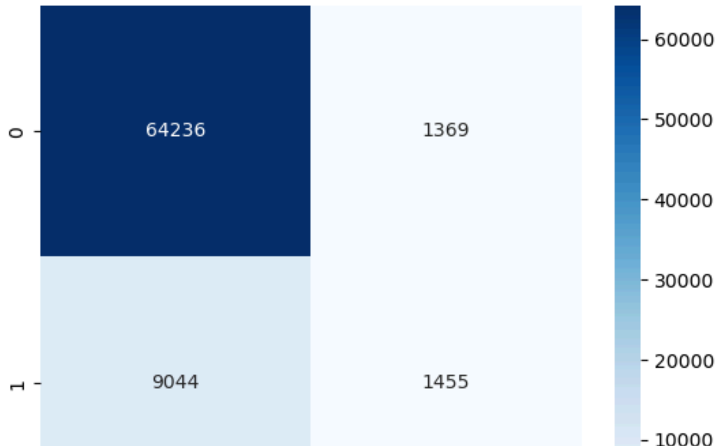
- solver
- max_iter
- penalty
- C
- class_weight

Preprocessing and Trials:

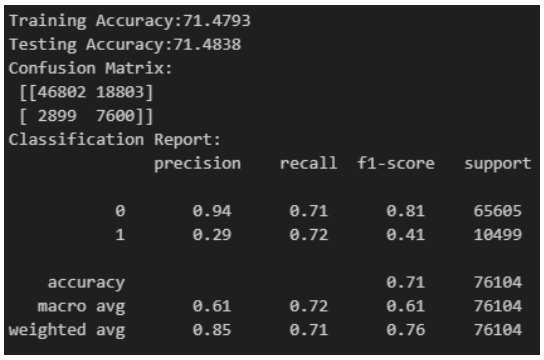
- The first trial was training using sklearn's Logistic Regression on the original dataset without SMOTE:



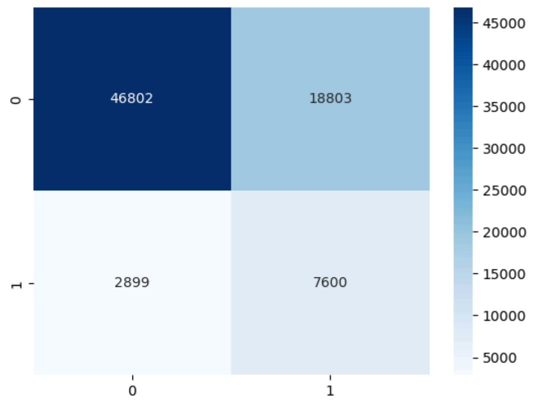
```
Training Accuracy:86.1761
Testing Accuracy:86.3174
Confusion Matrix:
 [[64236  1369]
 [ 9044  1455]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.98      0.93     65605
           1       0.52      0.14      0.22     10499

    accuracy                           0.86     76104
   macro avg       0.70      0.56      0.57     76104
weighted avg       0.83      0.86      0.83     76104
```



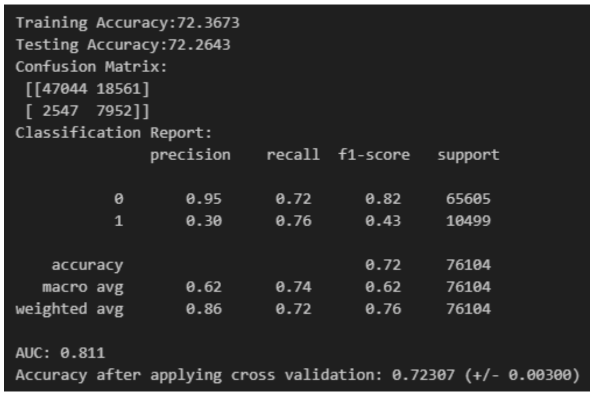
- The second trial was training using sklearn’s Logistic Regression on balanced dataset with SMOTE :



```
Training Accuracy:71.4793
Testing Accuracy:71.4838
Confusion Matrix:
 [[46802 18803]
 [ 2899  7600]]
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.71      0.81     65605
           1       0.29      0.72      0.41     10499

    accuracy                           0.71     76104
   macro avg       0.61      0.72      0.61     76104
weighted avg       0.85      0.71      0.76     76104
```



- Tuning class_weight parameters:
  We can see the accuracy dropped as previous because data is balanced but it has slightly higher accuracy. We can compromise the accuracy fr he drastic change of the recall.



```
Training Accuracy:72.3673
Testing Accuracy:72.2643
Confusion Matrix:
 [[47044 18561]
 [ 2547  7952]]
Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.72      0.82     65605
           1       0.30      0.76      0.43     10499

    accuracy                           0.72     76104
   macro avg       0.62      0.74      0.62     76104
weighted avg       0.86      0.72      0.76     76104

AUC: 0.811
Accuracy after applying cross validation: 0.72307 (+/- 0.00300)
```

- Tuning solver parameters:
  Saga and lbfgs are the best choices. So based on other factors as they are nearly the same, like computational efficiency and the presence of regularization penalties We chose lbfgs as it is usually chosen for large-scale problems like ours and slightly lower std.

solver	mean scores {mean(std)}
newton-cg	0.72309 (+/- 0.00302)
lbfgs	0.72307 (+/- 0.00299)
sag	0.72307 (+/- 0.00299)
saga	0.72309 (+/- 0.00302)

- Tuning penalty parameters:
  Only compatible regularization is the L2

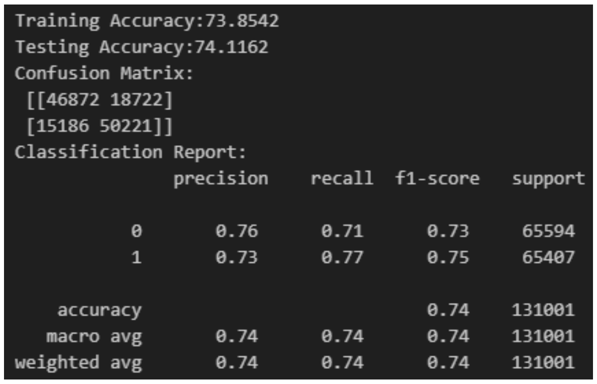
penalty	mean scores
l2	0.72307 (+/- 0.00299)

- Tuning C parameter:

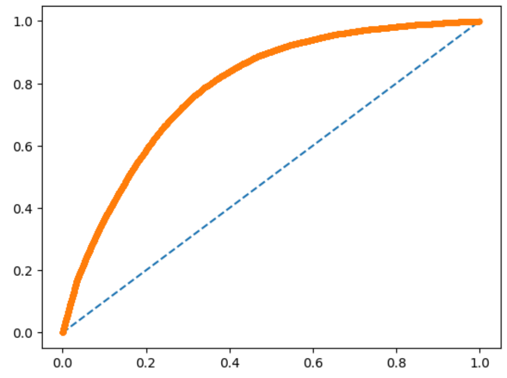
C	mean scores
0.1	0.72309 (+/- 0.00303)
0.2	0.72308 (+/- 0.00303)
0.3	0.72308 (+/- 0.00304)
0.4	0.72312 (+/- 0.00299)
0.5	0.72308 (+/- 0.00304)
0.6	0.72306 (+/- 0.00303)
0.7	0.72312 (+/- 0.00302)
0.8	0.72307 (+/- 0.00300)

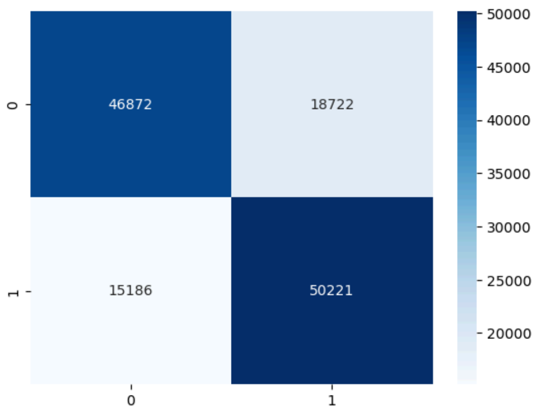
Final Model:

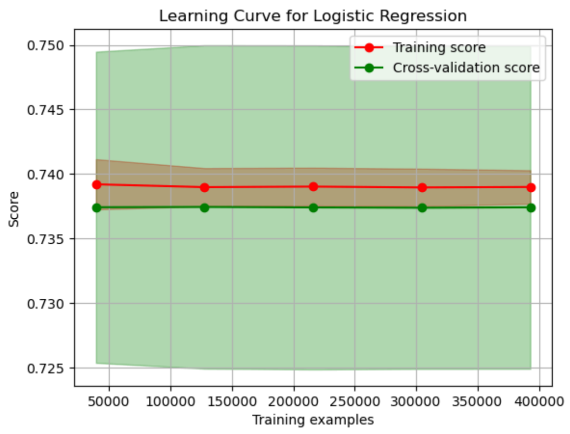
```
Accuracy after applying cross validation: 0.73906 (+/- 0.00238)
AUC: 0.7860
```



```
Training Accuracy:73.8542
Testing Accuracy:74.1162
Confusion Matrix:
 [[46872 18722]
 [15186 50221]]
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.71      0.73     65594
           1       0.73      0.77      0.75     65407

    accuracy                           0.74    131001
   macro avg       0.74      0.74      0.74    131001
weighted avg       0.74      0.74      0.74    131001
```







2. SVM

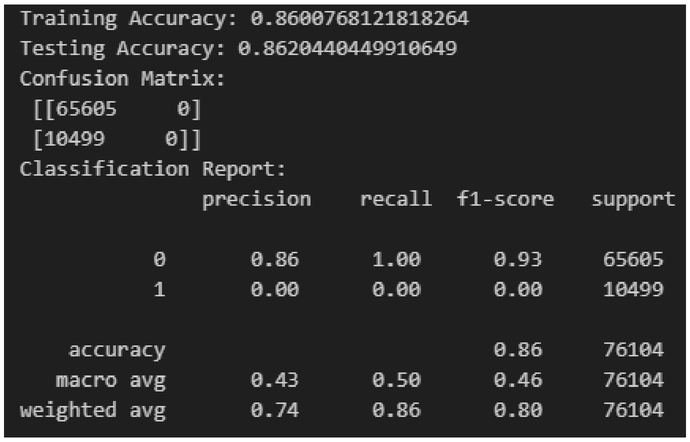
Support Vector Machines (SVMs) are powerful supervised learning models used for classification and regression tasks. They are used for imbalanced data as it's interested in support vectors that imply a small margin of error.
However, we came to learn that if the data is significantly imbalanced then it becomes an anomaly detection method.

Parameters:

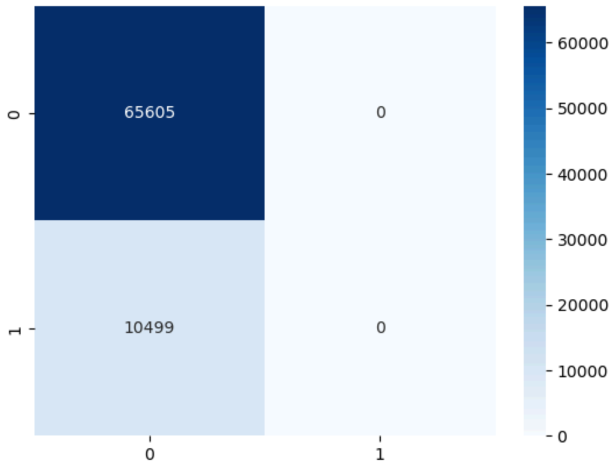
- C
- gamma
- kernel
- class_weight

Preprocessing and Trials:

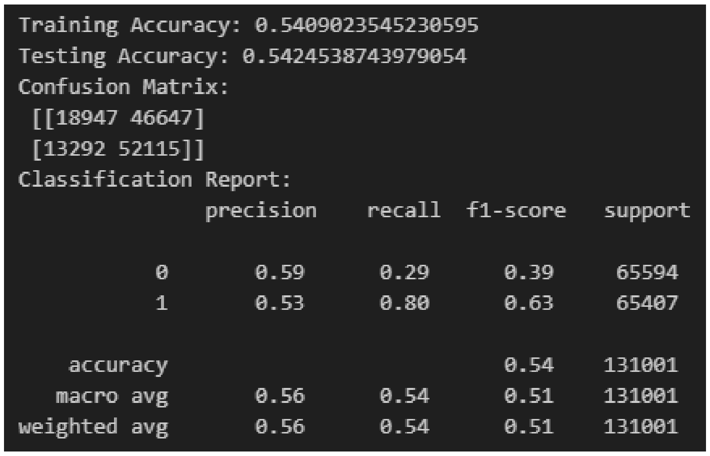
- The first trial was training using sklearn's SVC on the original dataset without SMOTE. The training took over 3 Hours. The model classified all results as 0 as this is the majority. When we searched for the reason why this was something common for SVM if the data is severely imbalanced.



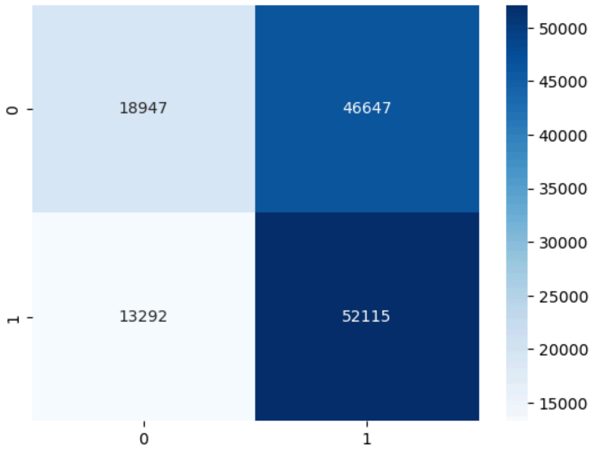
```
Training Accuracy: 0.8600768121818264
Testing Accuracy: 0.8620440449910649
Confusion Matrix:
 [[65605     0]
 [10499     0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.86      1.00      0.93     65605
           1       0.00      0.00      0.00     10499

    accuracy                           0.86     76104
   macro avg       0.43      0.50      0.46     76104
weighted avg       0.74      0.86      0.80     76104
```



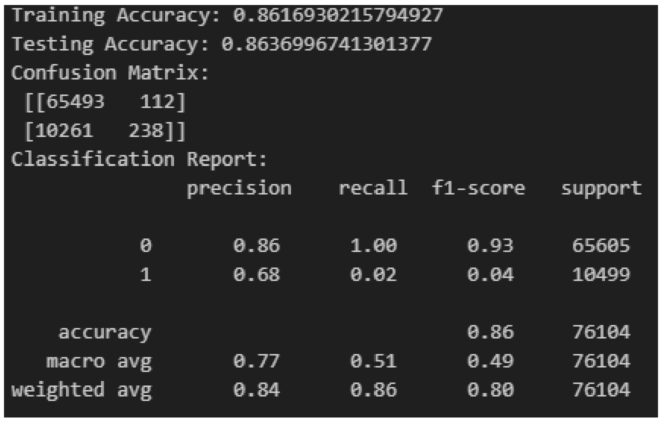
- Therefore for the second trial we trained on the balanced dataset by applying SMOTE and also having the parameter of class_weight = 'balanced'.



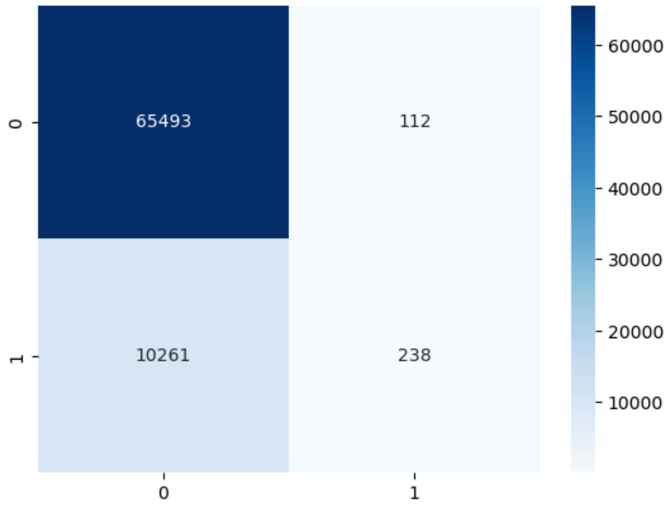
```
Training Accuracy: 0.5409023545230595
Testing Accuracy: 0.5424538743979054
Confusion Matrix:
 [[18947 46647]
 [13292 52115]]
Classification Report:
              precision    recall  f1-score   support

           0       0.59      0.29      0.39     65594
           1       0.53      0.80      0.63     65407

    accuracy                           0.54    131001
   macro avg       0.56      0.54      0.51    131001
weighted avg       0.56      0.54      0.51    131001
```



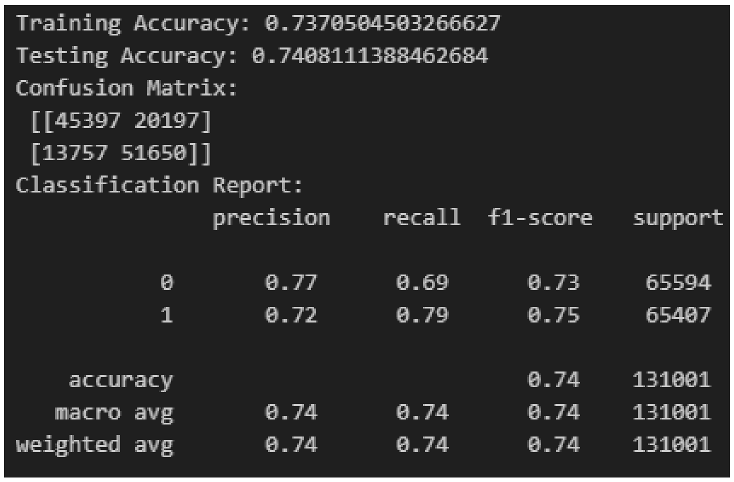
- The third trial was training using only 3 features by reducing them with PCA. But it had the same output of first trial
- The fourth trial was training using LinearSVC to be able to tune hyperparameters efficiently
    - A. Imbalanced Data



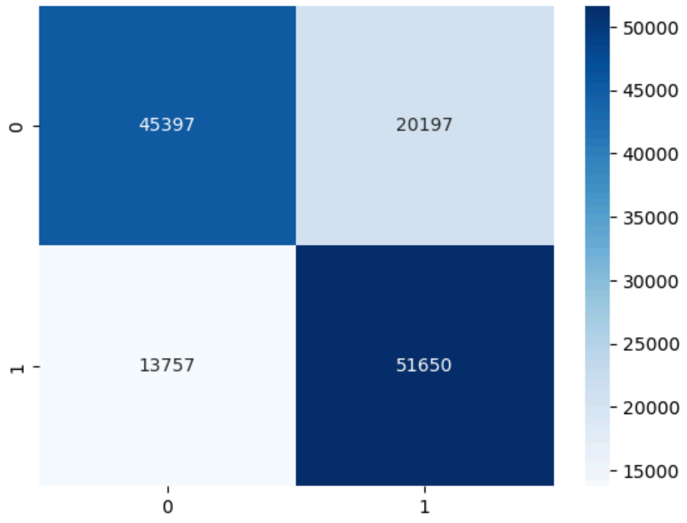
```
Training Accuracy: 0.8616930215794927
Testing Accuracy: 0.8636996741301377
Confusion Matrix:
 [[65493   112]
 [10261   238]]
Classification Report:
               precision    recall  f1-score   support

           0       0.86      1.00      0.93     65605
           1       0.68      0.02      0.04     10499

    accuracy                           0.86     76104
   macro avg       0.77      0.51      0.49     76104
weighted avg       0.84      0.86      0.80     76104
```



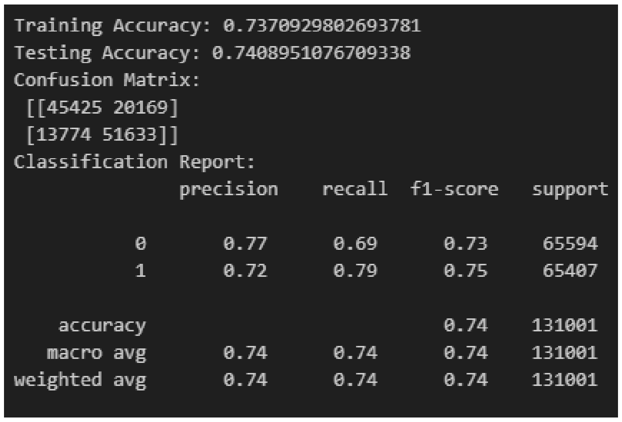
    - B. Data with SMOTE



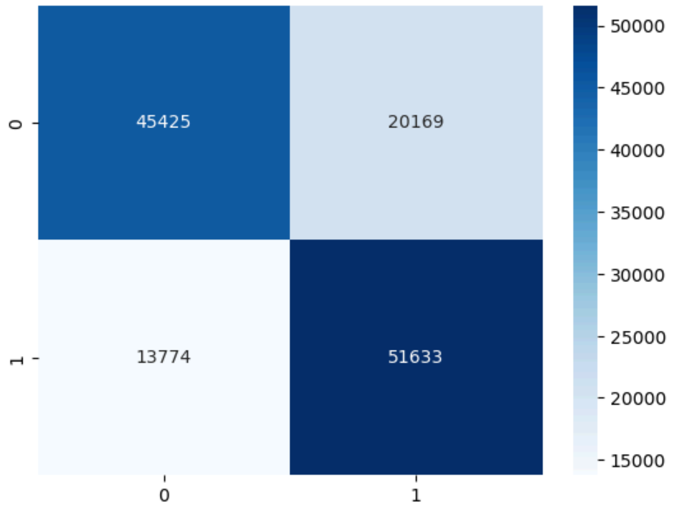
```
Training Accuracy: 0.7370504503266627
Testing Accuracy: 0.7408111388462684
Confusion Matrix:
 [[45397 20197]
 [13757 51650]]
Classification Report:
               precision    recall  f1-score   support

           0       0.77      0.69      0.73     65594
           1       0.72      0.79      0.75     65407

    accuracy                           0.74    131001
   macro avg       0.74      0.74      0.74    131001
weighted avg       0.74      0.74      0.74    131001
```



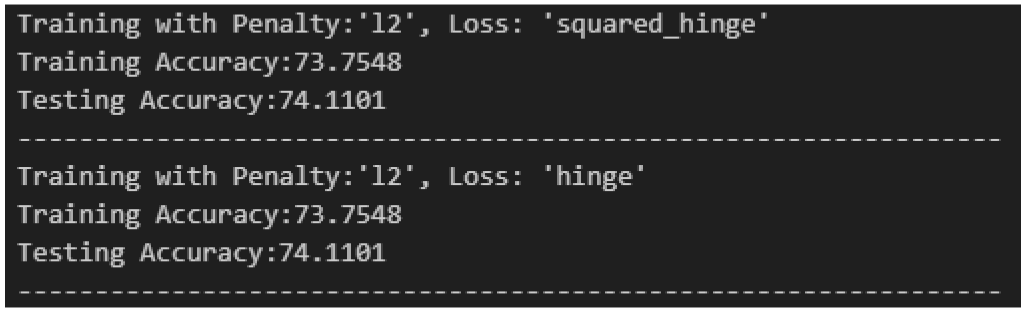
- Tuning with class_weight ='balanced':



```
Training Accuracy: 0.7370929802693781
Testing Accuracy: 0.7408951076709338
Confusion Matrix:
 [[45425 20169]
 [13774 51633]]
Classification Report:
               precision    recall  f1-score   support

           0       0.77      0.69      0.73     65594
           1       0.72      0.79      0.75     65407

    accuracy                           0.74    131001
   macro avg       0.74      0.74      0.74    131001
weighted avg       0.74      0.74      0.74    131001
```

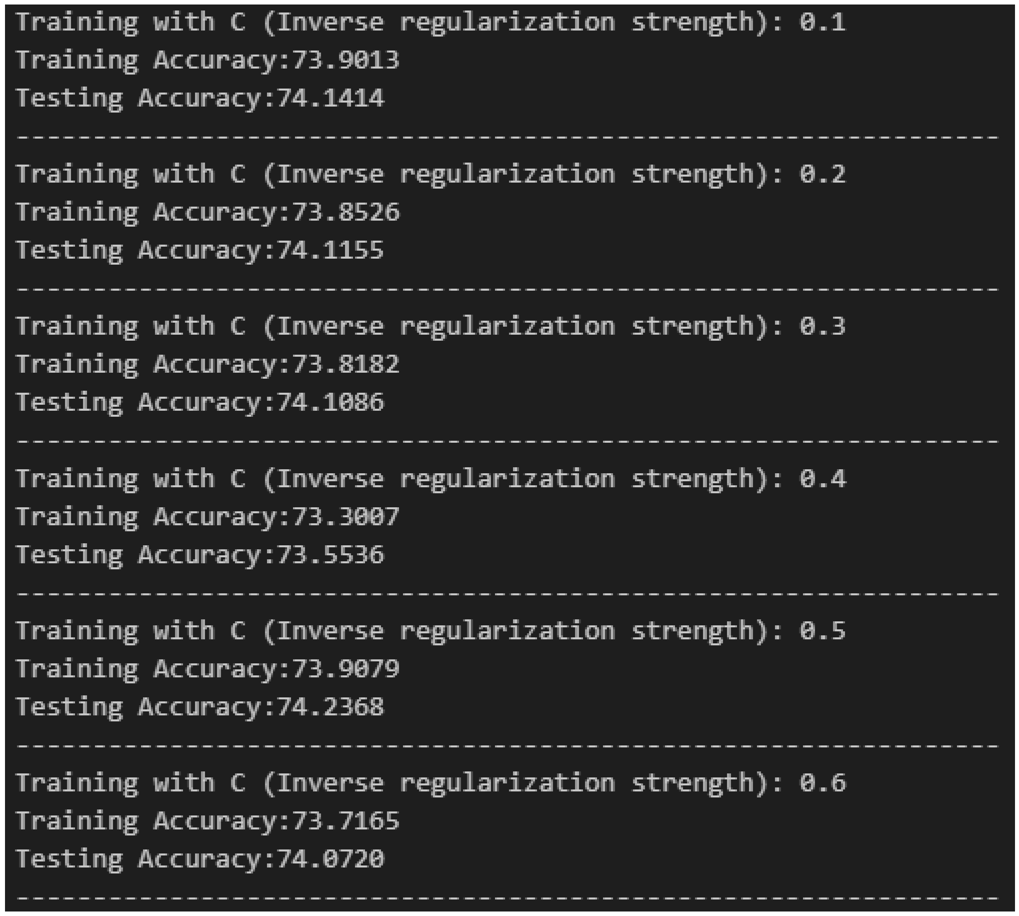


- Tuning penalty and loss combinations :
  L1 isn’t supported with loss functions here. We chose squared_hinge because The squared hinge loss penalizes margin violations more severely. It’s less sensitive to outliers and lead to more robust models
  It also results in a smoother and more continuously differentiable loss function compared to the hinge loss.

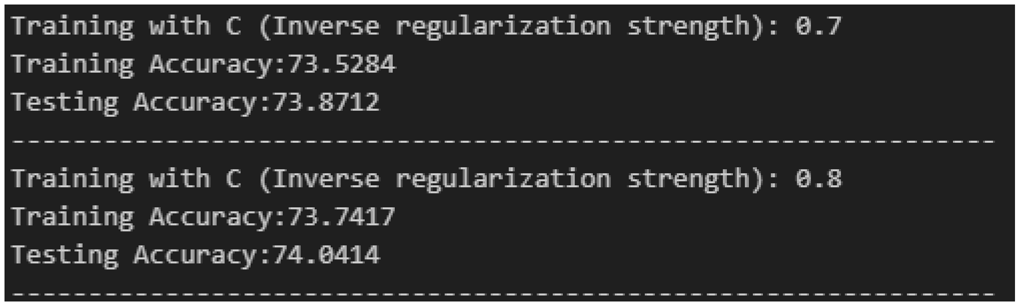


```
Training with Penalty:'l2', Loss: 'squared_hinge'
Training Accuracy:73.7548
Testing Accuracy:74.1101
----------------------------------------------------------------
Training with Penalty:'l2', Loss: 'hinge'
Training Accuracy:73.7548
Testing Accuracy:74.1101
----------------------------------------------------------------
```

- Tuning C parameter:



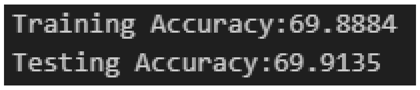
```
Training with C (Inverse regularization strength): 0.1
Training Accuracy:73.9013
Testing Accuracy:74.1414
----------------------------------------------------------------
Training with C (Inverse regularization strength): 0.2
Training Accuracy:73.8526
Testing Accuracy:74.1155
----------------------------------------------------------------
Training with C (Inverse regularization strength): 0.3
Training Accuracy:73.8182
Testing Accuracy:74.1086
----------------------------------------------------------------
Training with C (Inverse regularization strength): 0.4
Training Accuracy:73.3007
Testing Accuracy:73.5536
----------------------------------------------------------------
Training with C (Inverse regularization strength): 0.5
Training Accuracy:73.9079
Testing Accuracy:74.2368
----------------------------------------------------------------
Training with C (Inverse regularization strength): 0.6
Training Accuracy:73.7165
Testing Accuracy:74.0720
----------------------------------------------------------------
```



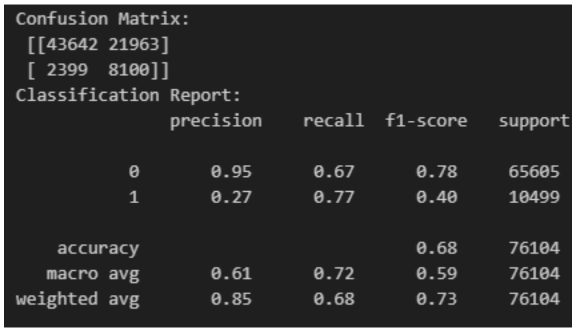
```
Training with C (Inverse regularization strength): 0.7
Training Accuracy:73.5284
Testing Accuracy:73.8712
----------------------------------------------------------------
Training with C (Inverse regularization strength): 0.8
Training Accuracy:73.7417
Testing Accuracy:74.0414
----------------------------------------------------------------
```

Final Model:

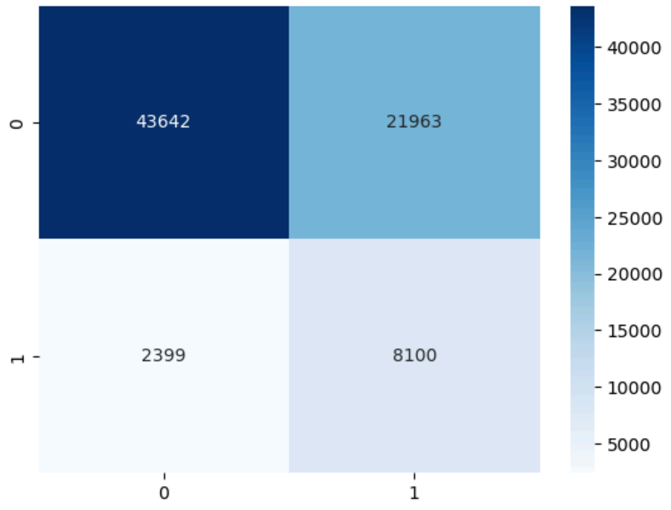
On normal Data:

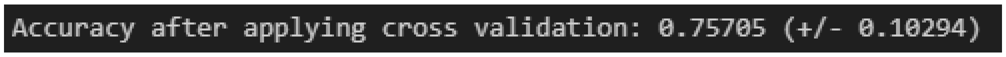


```
Training Accuracy:69.8884
Testing Accuracy:69.9135
```



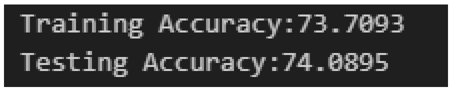
```
Confusion Matrix:
 [[43642 21963]
 [ 2399  8100]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.67      0.78     65605
           1       0.27      0.77      0.40     10499

    accuracy                           0.68     76104
   macro avg       0.61      0.72      0.59     76104
weighted avg       0.85      0.68      0.73     76104
```



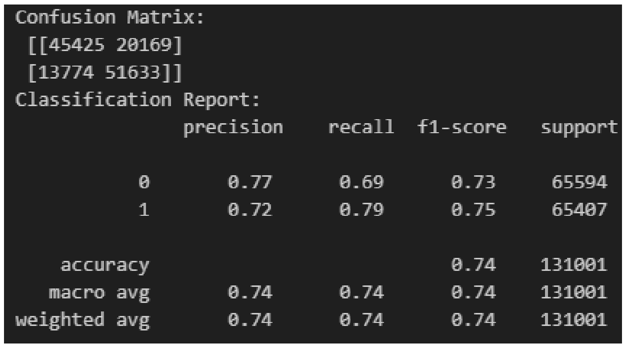


```
Accuracy after applying cross validation: 0.75705 (+/- 0.10294)
```

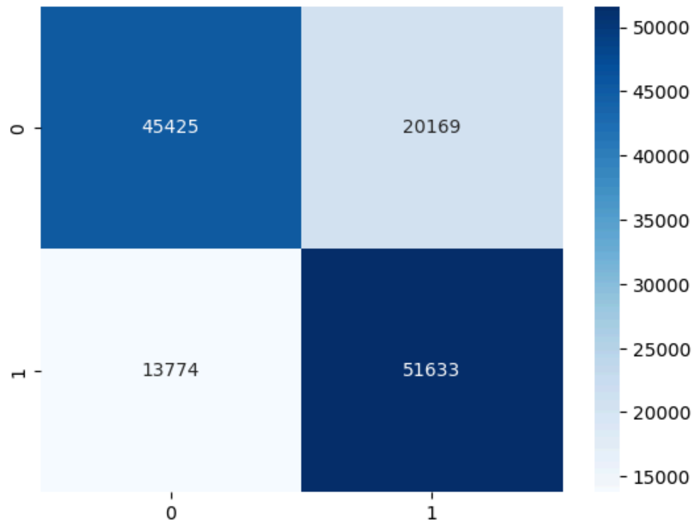
On balanced Data:



```
Training Accuracy:73.7093
Testing Accuracy:74.0895
```



```
Confusion Matrix:
 [[45425 20169]
 [13774 51633]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.69      0.73     65594
           1       0.72      0.79      0.75     65407

    accuracy                           0.74    131001
   macro avg       0.74      0.74      0.74    131001
weighted avg       0.74      0.74      0.74    131001
```



3. AdaBoost

a. Decision Tree Classifier

The AdaBoost Classifier is an ensemble algorithm that is created to turn multiple weak classifiers into a strong classifier. The main idea of the AdaBoost Classifier is to pass mistakes made by one classifier to the next so that it can learn from it by giving higher weight to the misclassified points. It also gives a weight for each classifier based on the accuracy and the learning rate.

The main problem with the AdaBoost Classifier was the imbalanced data since the recall for the positive class was 0.15. In this case this could be worse than having a low recall for the negative class since mis-diagnosing someone is more severe.

Parameters:

- estimator: the base (weak) classifier.
- n_estimators: how many weak classifiers to use in the training phase.
- learning _rate: the weight applied to each classifier in each iteration.
- class_weight

Preprocessing and Trials:

- The first trial was training using sklearn's AdaBoostClassifier on the original dataset without SMOTE:
    - First we try multiple number of estimators ranging from [50-1000]

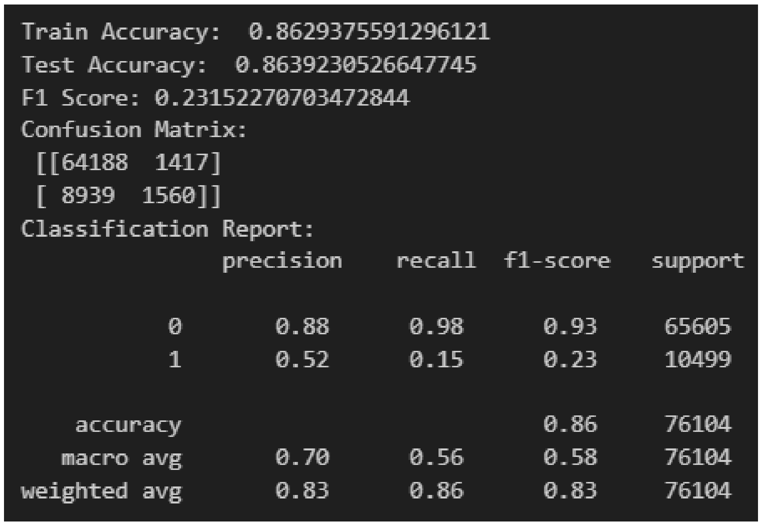
Number of Estimators	Test Accuracy	F1 Score
50	0.8639	0.237
100	0.8639	0.231
200	0.8639	0.233
500	0.864	0.234
1000	0.864	0.234

- The we try different learning rates

Learning Rate	Test Accuracy	F1 Score
0.01	0.862	0.0
0.05	0.864	0.19
0.1	0.864	0.22
0.5	0.864	0.23
1	0.864	0.23

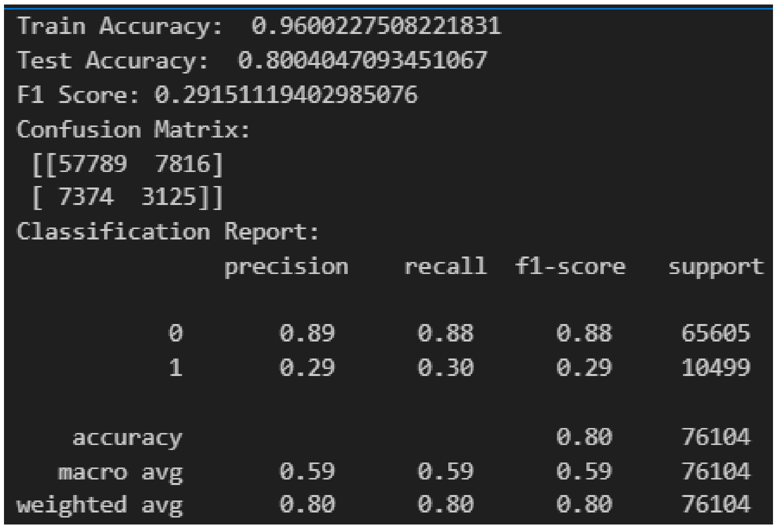
This shows the data imbalance holds a great influence on the evaluation metrics. So we focus on trying to handle this issue using SMOTE and class weight adjustments.

- The second trial we kept the n_estimators and the learning_rate constants and observed the effect of the weights:
  - Without any tuning



```
Train Accuracy:  0.8629375591296121
Test Accuracy:  0.8639230526647745
F1 Score: 0.23152270703472844
Confusion Matrix:
 [[64188  1417]
 [ 8939  1560]]
Classification Report:
               precision    recall  f1-score   support

           0       0.88      0.98      0.93     65605
           1       0.52      0.15      0.23     10499

    accuracy                           0.86     76104
   macro avg       0.70      0.56      0.58     76104
weighted avg       0.83      0.86      0.83     76104
```

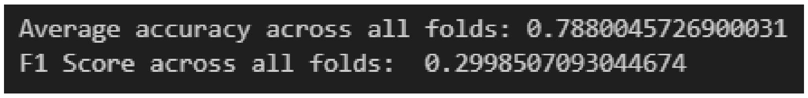
- Assign ‘balanced’ to class_weight parameter in the DecisionTreeClassifier
  The class_weight=’balanced’ parameter will adjust the weights based on the number of samples in each class which is made to deal with data imbalance. This is done on the original dataset without any resampling.



```
Train Accuracy:  0.9600227508221831
Test Accuracy:  0.8004047093451067
F1 Score: 0.29151119402985076
Confusion Matrix:
 [[57789  7816]
 [ 7374  3125]]
Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.88      0.88     65605
           1       0.29      0.30      0.29     10499

    accuracy                           0.80     76104
   macro avg       0.59      0.59      0.59     76104
weighted avg       0.80      0.80      0.80     76104
```

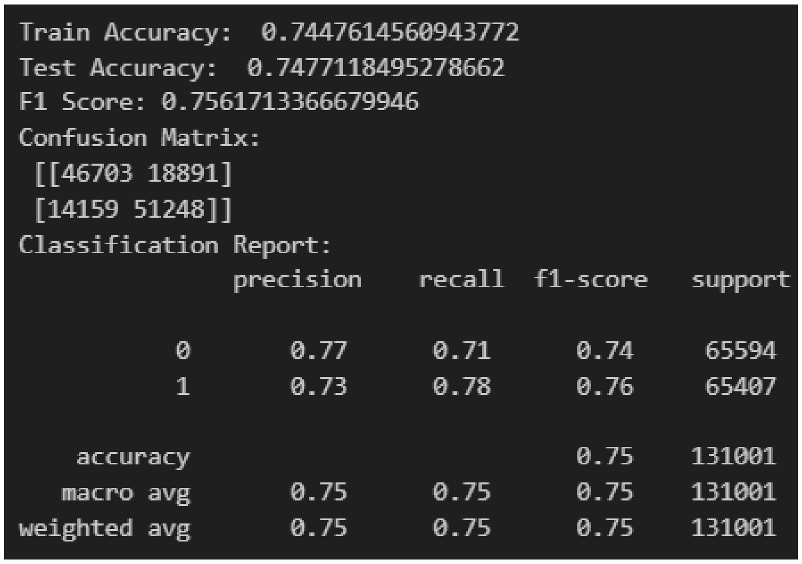
f

- In this case we apply cross validation to avoid overfitting of the data since there's a considerable gap between the training accuracy and the test accuracy.
- We used the StratifiedKFold with the number of splits 5
- The results show that the test accuracy and the f1 scores from the cross validation are close to that of the previous one.



```
Average accuracy across all folds: 0.7880045726900031
F1 Score across all folds:  0.2998507093044674
```

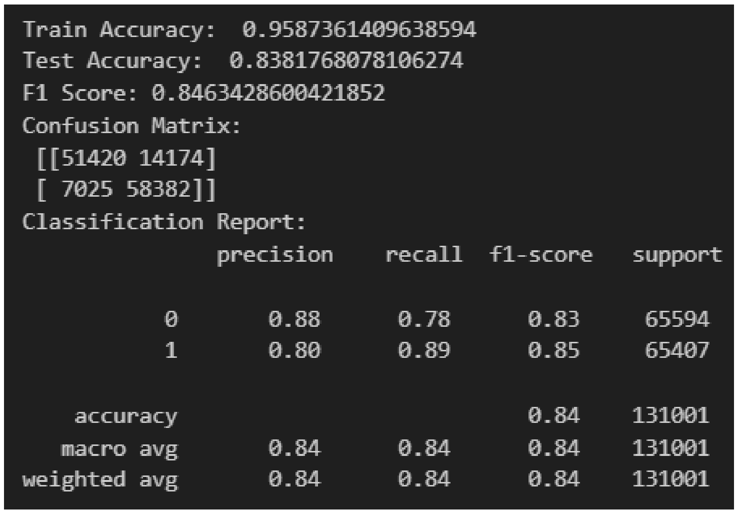
- Then we use the resampled dataset using SMOTE without adjusting any weight parameters.



```
Train Accuracy:  0.7447614560943772
Test Accuracy:  0.7477118495278662
F1 Score: 0.7561713366679946
Confusion Matrix:
 [[46703 18891]
 [14159 51248]]
Classification Report:
               precision    recall  f1-score   support

           0       0.77      0.71      0.74     65594
           1       0.73      0.78      0.76     65407

    accuracy                           0.75    131001
   macro avg       0.75      0.75      0.75    131001
weighted avg       0.75      0.75      0.75    131001
```

Using the resampled data the overall accuracy decreased however the F1 Score increased significantly which in our case could be a good thing since the correct diagnosis of diabetic patients increased.

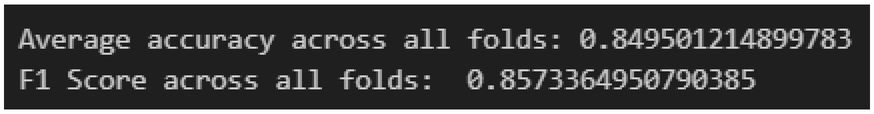
- Lastly we tried the class_weight='balanced' with the resampled data.



```
Train Accuracy:  0.9587361409638594
Test Accuracy:  0.8381768078106274
F1 Score: 0.8463428600421852
Confusion Matrix:
 [[51420 14174]
 [ 7025 58382]]
Classification Report:
               precision    recall  f1-score   support

           0       0.88      0.78      0.83     65594
           1       0.80      0.89      0.85     65407

    accuracy                           0.84    131001
   macro avg       0.84      0.84      0.84    131001
weighted avg       0.84      0.84      0.84    131001
```

The idea to resample the data could result in model overfitting as the samples so we apply cross validation.

  - So we applied the StratifiedKFold cross validation technique to ensure that the test avoids overfitting. The accuracy and the f1 scores for all folds are slightly higher than without cross validation but still with a considerable gap between the test accuracy and the train accuracy.



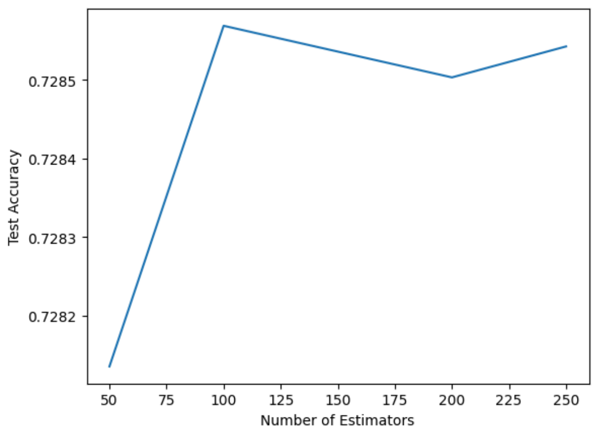
```
Average accuracy across all folds: 0.849501214899783
F1 Score across all folds:  0.8573364950790385
```

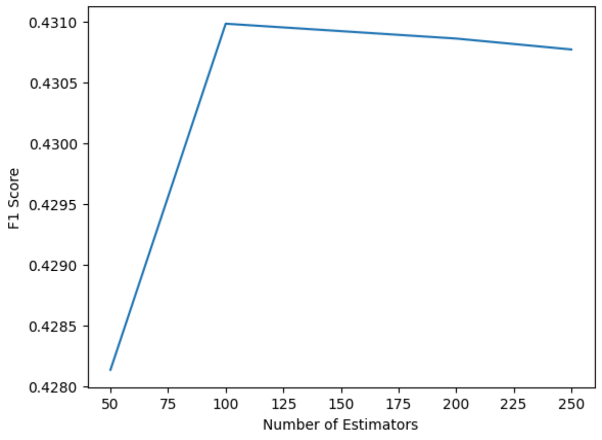
- Failed Attempt:
  - Since the AdaBoost works by assigning weights to each sample in the dataset based on if they were correctly or wrongly classified, we thought to increase the weight assigned to misclassified samples to give higher priority in each classifier. The trial failed since the accuracy along with the F1 Score kept decreasing when we increased the weight multiplier.
  - This was done by implementing a custom AdaBoost classifier and adjusting weight calculations during the fitting phase.
  - However it gave the same results using the resampled data.

b. Logistic Regression

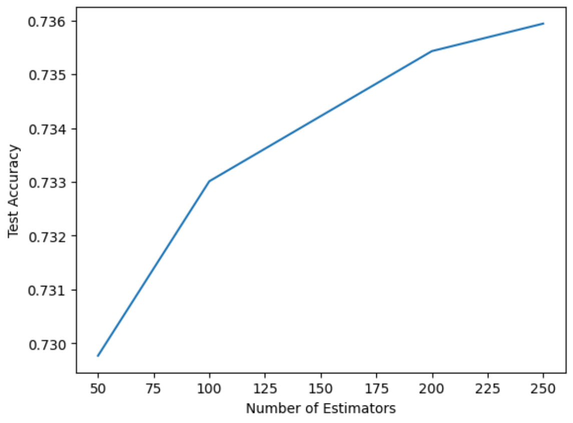
After tuning the Logistic Regression as a stand alone model we integrated the tuned model to the AdaBoost Classifier Base (Weak Model). Same as before we tried it on the original and the resampled datasets. The results were not very promising for both balanced and imbalanced data.

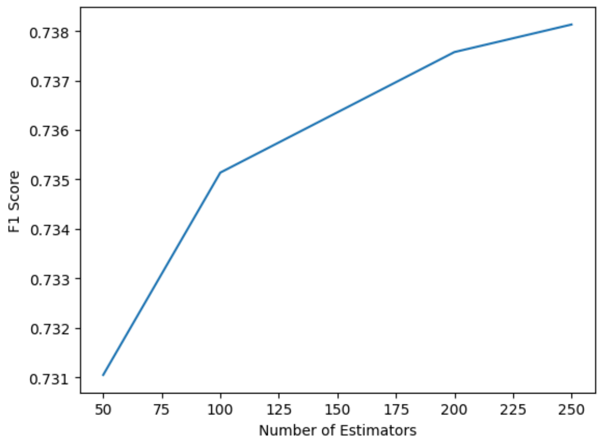
- Imbalanced Dataset





- Resampled Dataset Using SMOTE





Conclusion

The imbalancement in the data has a huge effect on the model evaluation metrics and resampling the data could lead to overfitting which could be handled using regularization and validation techniques. The greatest effect on the evaluation of the models was the adjustments of the class weight parameters which are used to handle imbalanced data. However, the other hyperparameters didn't have a significant effect on the accuracy of the model. The model that scored the highest across all evaluation metrics was the AdaBoost Classifier with the Decision Tree Classifier as the weak classifier as it can handle imbalanced data. Certainly there are other classifiers that can be used that handle imbalanced data better than the ones covered in the course and they can be used to train and evaluate our chosen problem.