Faculty of Engineering and Technology

Electrical and computer engineering department

Machine Learning and Data Science - ENCS5341

Assignment #3

Instructor Name: Dr.Yazan Abu Farha

Student name: Nasralla Hassan

Student number: 1200134

Date :1/4/2024

Contents

Introduction

In this project, I developed a classification model to predict the occurrence of a cardiovascular disease. This is an important project because heart diseases are one of the main causes of death worldwide. Early and precise predictions can help improve healthcare management.

Task description

The main task is building a proper model using a set of health and personal features to predict cardiovascular disease in patients with high accuracy.

Models Tried

1- **Logistic regression**: is a simple and easy-to-understand binary classification model.
2- **K-Nearest Neighbors**: This technique splits the data points based on the majority class of its k neighbors.
3- **Random Forest Classifier**: Uses several decision trees to get better accuracy and avoid overfitting.

Evaluation Metrics

1- **Accuracy**: the percentage of accurately predicted cases compared to total predictions.
2- **Precision**: The ratio of correctly predicted positive to the total predicted positives. High precision relates to a low rate of false positives, which is critical in medical diagnostics.
3- **Recall**: Is the ratio of accurately predicted positive cases to all cases in the class. It is so important to get as small as possible false negatives in illness prediction.
4- **F1 Score:** Is a weighted average of Precision and Recall.
5- **Confusion Matrix:** Shows The true positive/ negative, and false positive/negative cases.

Data set

Description

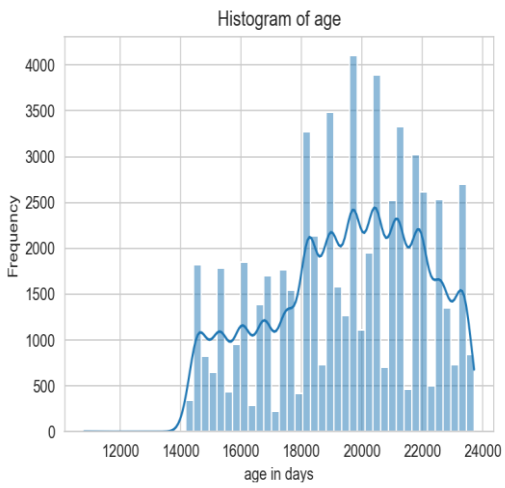
1- **Id:** unique identifier for each row.
2- **Age:** age of the individuals in days. Mean= 19465.
3- **Gender:** gender for individuals (1 (women) or 2(men)).
4- **Height:** height of individuals in centimeters. Mean= 164 cm.
5- **Weight:** weight of individuals in centimeters. Mean=74
6- **Ap_hi:** systolic blood pressure. Mean=129.
7- **Ap_lo:** diastolic blood pressure. Mean=96.
8- **Cholesterol:** cholesterol level (1(normal), 2(above normal), or 3(well above normal)). Mean=1.3.
9- **Gluc:** glucose level (1(normal), 2(above normal), or 3(well above normal)). Mean= 1.22.
10- **Smoke:** smoking status (0: non-smoker, 1: smoker). Mean= 0.08.
11- **Alco**: Alcohol intake status (0: non-drinker, 1: drinker). Mean=0.053349
12- **Active:** physical activity status (0: inactive, 1: active). Mean= 0.804183
13- **Cardio**: presence or absence of cardiovascular disease (0: No, 1; YES). Mean=0.49

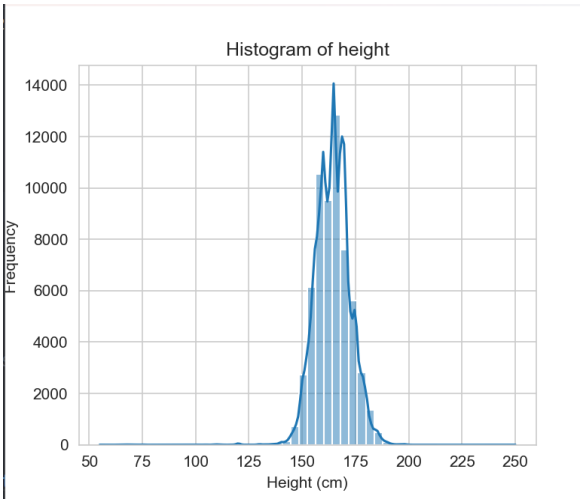
Link to data set: https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset

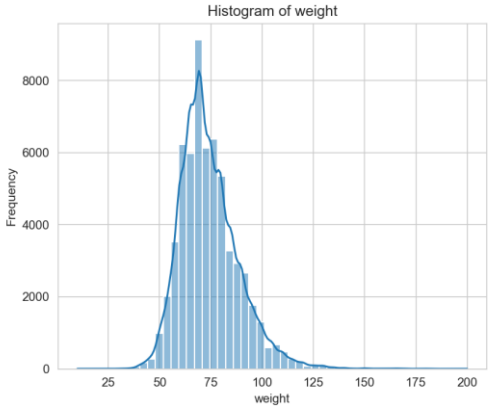
Statistics

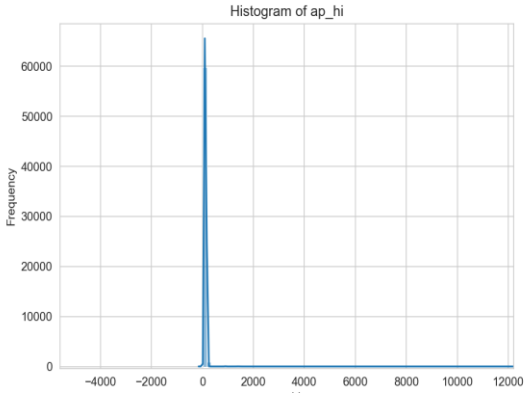
1- The data consists of 60863 entries.
2- Age: ranges from 29 to 65 years.
3- Height: ranges from 55 to 250 cm.
4- Weight: ranges from 10 to 200 kg.
5- Blood pressure: some values are normal, some of them are negative or extremely high (ranges from -150 to 16020).
6- Cholesterol and glucose levels: each is categorized into three levels.
7- Lifestyle factors (smoke, active, alco): binary values indicate the presence or the absence of each factor.
8- Cardiovascular disease(target): almost evenly distributed between presence (49%) and absence (51%) of the disease.

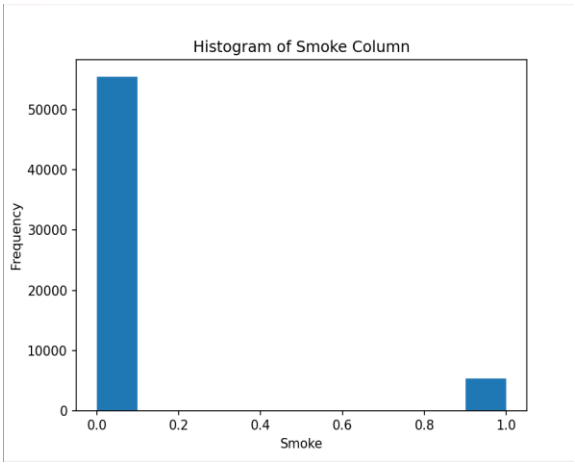
Visualizations

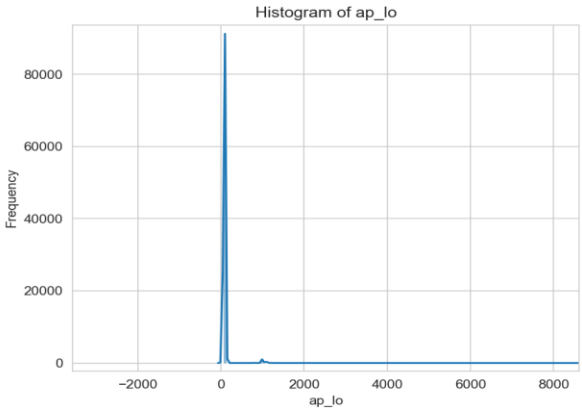


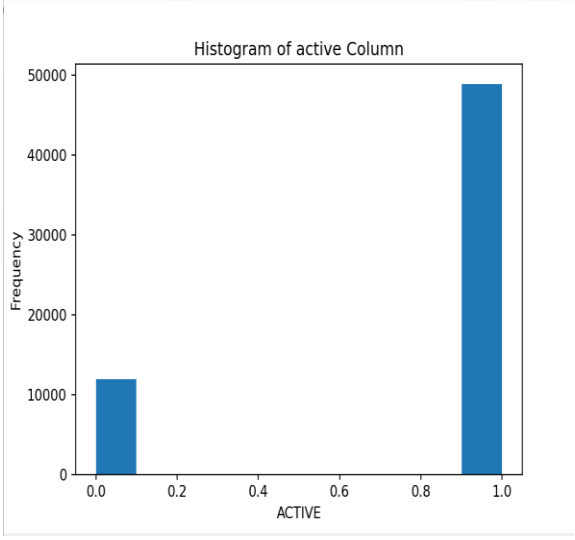


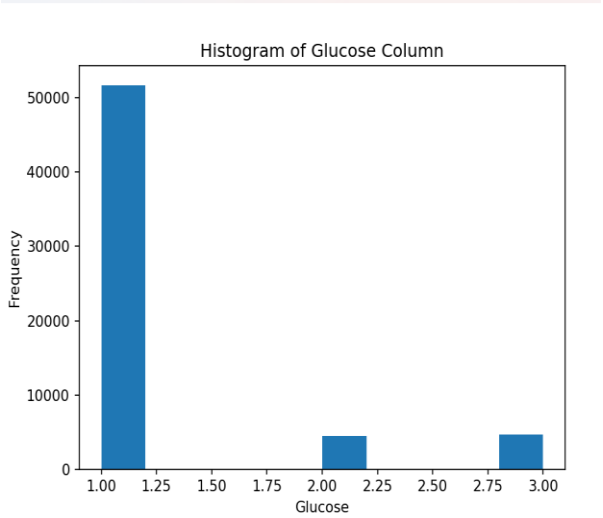


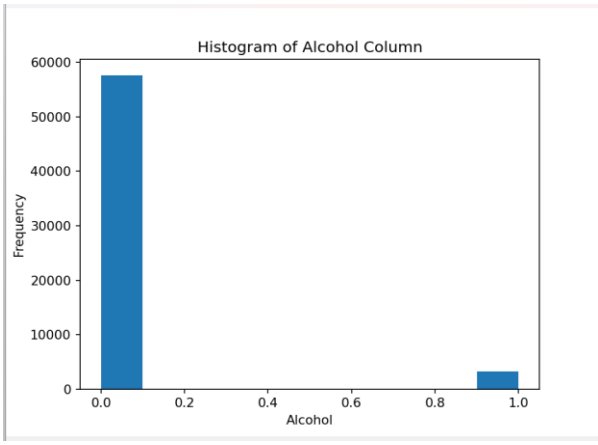


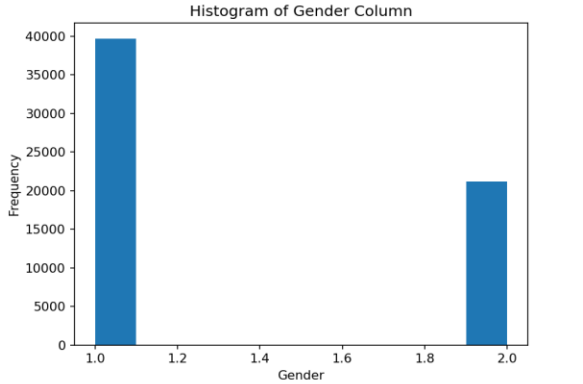


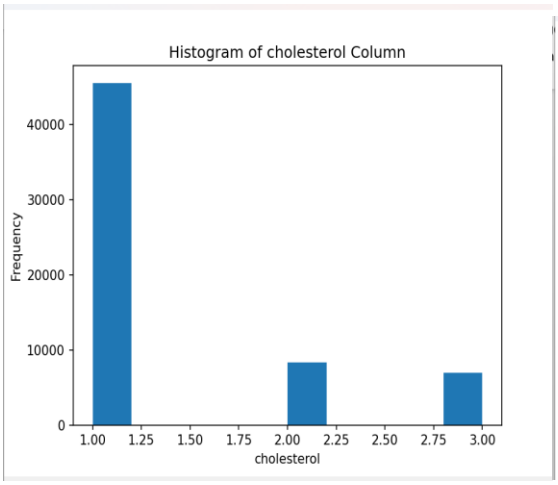


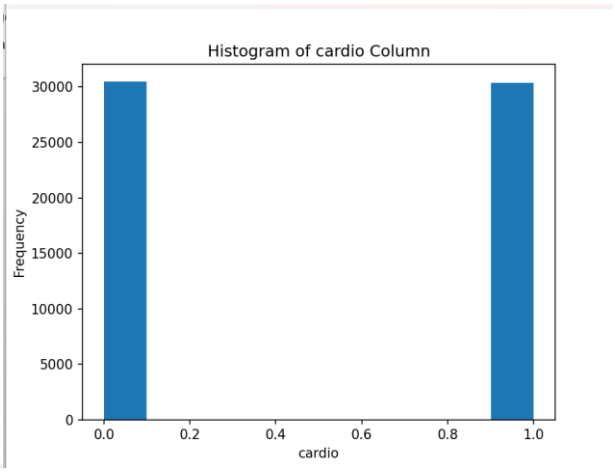

Histogram of active Column
Frequency
ACTIVE
0
10000
20000
30000
40000
50000
0.0
0.2
0.4
0.6
0.8
1.0


Histogram of Glucose Column
Frequency
Glucose
0
10000
20000
30000
40000
50000
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00


Histogram of Alcohol Column
Frequency
Alcohol
0
10000
20000
30000
40000
50000
60000
0.0
0.2
0.4
0.6
0.8
1.0


Histogram of Gender Column
Frequency
Gender
0
5000
10000
15000
20000
25000
30000
35000
40000
1.0
1.2
1.4
1.6
1.8
2.0


Histogram of cholesterol Column
Frequency
cholesterol
0
10000
20000
30000
40000
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00


Histogram of cardio Column
Frequency
cardio
0
5000
10000
15000
20000
25000
30000
0.0
0.2
0.4
0.6
0.8
1.0

Feature	Visualization
age	Distribution is relatively uniform across the range.
Height & weight	Show a normal distribution, with some potential outliers
Blood pressure	Has a wide range, with some unrealistic values

To address this, I used a huge dataset originally containing 70,000 items to predict cardiovascular disease. Due to software limitations, I had to reduce it to 60,863 entries. Following a thorough cleanup, I eliminated the 'id' column, which was unnecessary for our prediction, leaving us with 58,654 entries and 12 important features. This dataset has key health features such as age, height, weight, blood pressure measurements (systolic and diastolic), cholesterol, and glucose levels, as well as smoking, alcohol consumption, and physical activity.

My primary objective? To put a higher priority on recall to reduce false negatives, which occur when a patient has an illness but is wrongly labeled as healthy. An accurate diagnosis is critical in this situation.

Here's what I've done so far to clean up the data:

Outliers Management: I have used boundaries (min and max values) for height, weight, and blood pressure (both systolic and diastolic) to detect and remove any outliers. The blood pressure ranges are 60 and 120 mmHg for diastolic and 90 and 200 mmHg for systolic. Weight and height range from 50 kg to 180 kg and 110 cm to 220 cm, respectively. Age Conversion and Rounding: The age data, originally given in days, made little sense for practical reasons. So, I converted it into years to make it more understandable.

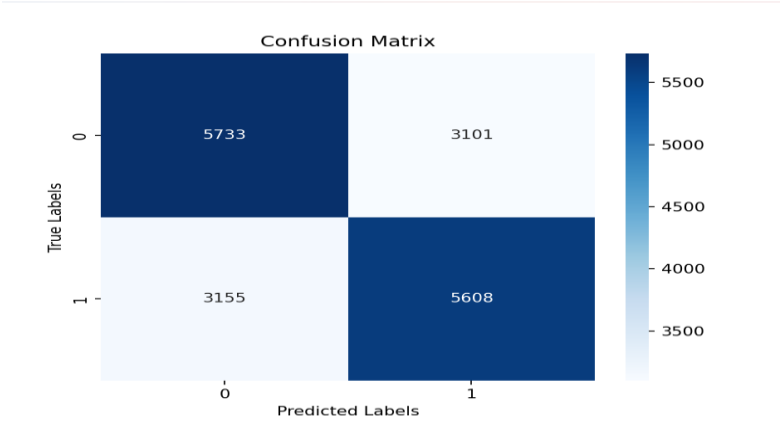
Experiments and Results

(KNN-Classifier)

1- When K =1. Indicates that the classification is based on the nearest neighbor. This may make the classifier sensitive to noise in the dataset.

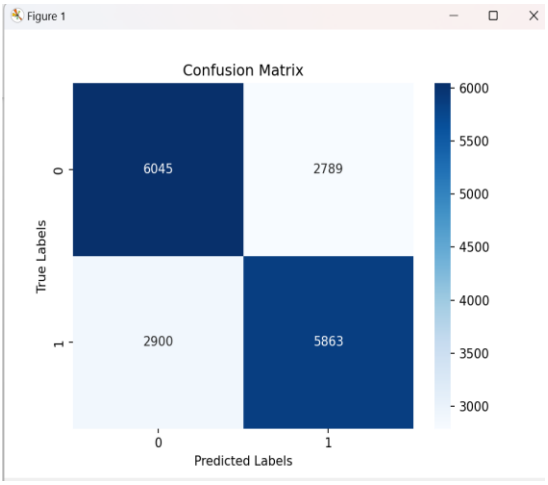
The performance of the base line was as following:

precision	recall	F1-score	accuracy
0.64	0.63	0.64	0.64



2-When K = 3.

Precision	recall	F1-score	accuracy
0.67	0.66	0.68	0.67



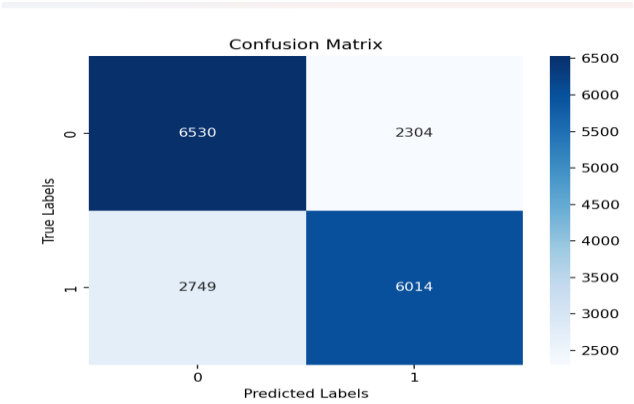
When we compare the confusion matrices for KNN classifiers with (k=1) and (k=3), we can see that the model's performance is getting better as (K) gets increased. Especially, with (k=3), the classifier performs better in detecting both negative and positive cases, as shown by a higher number of true negatives and true positives. Also, both false positives and false negatives are reduced, showing that the classifier with (k=3) is less likely to make classification errors. However there still some space for improvements on the performance, since the results aren't really that good.

For the K of my choice:

I have used **K-cross validation** (**15** cross validation) to find the best K. the range values of K were between **(1 - 30).** I have found that to get the minimum mean square error, K must be **28**, and to have the highest recall, K must be **11**. So, I have chosen (K) = 11 to get the best recall **(69).**

Overall Performance:

K-NN classifier when K = 11:



Classification Report for KNN:

	precision	recall	f1-score	support
0	0.70	0.74	0.72	8834
1	0.72	0.69	0.70	8763
accuracy			0.71	17597
macro avg	0.71	0.71	0.71	17597
weighted avg	0.71	0.71	0.71	17597

Analysis:

Precision: is good for both classes which means that when the model predicts a class, it is correct about 70-72% of the time.

Recall: The model is slightly better at identifying all relevant instances of Class 0 than Class 1. But it's still better than k=1, and k =3.

Comments:

The model looks to perform good in both classes (0 and 1), which is excellent because it does not appear to favor one over the other. The fact that the precision, recall, and F1-scores for all categories are quite similar helps this.

The confusion matrix indicates 2,304 false positives, which occurred when the model wrongly predicted the presence of the 'cardio' illness. This type of inaccuracy can cause stress for patients, further testing, and potentially unneeded treatment.

False negatives, on the other hand, account for 2,749 cases in which the model failed to detect the disease when it was there. This seems to be dangerous, since it may result in missed opportunities for early intervention, potentially leading to more serious health outcomes for those individuals.

There is a small bias toward more false positives than negatives, according to the confusion matrix. This could indicate that the model prefers to predict Class 0 unless it is very certain about Class 1. So, I need to find a better model, that focuses more on boosting the recall, K-NN could not be completely performing well.

Random forest:

For the random forest I have used stepwise feature selection algorithm to find the best features to build the model. And the result was:

When number of estimators = 10 => [‘ap_hi', ‘cholesterol’, ‘age’]

When number of estimators = 50 => [‘ap_hi', ‘gluc’, ‘age’]

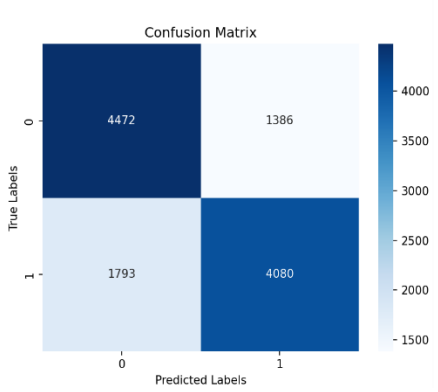
When number of estimators = 100 => [‘ap_hi', ‘gluc’, ‘age’]

When number of estimators = 200 => [‘ap_hi', ‘gluc’, ‘age’]

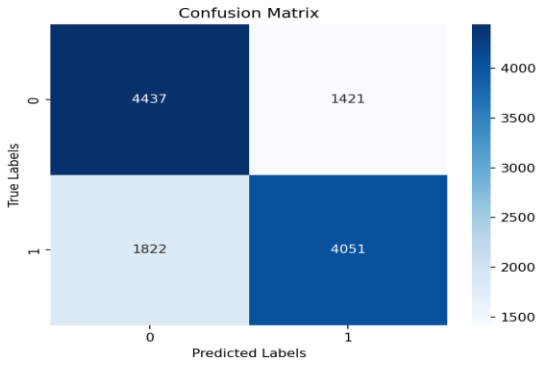
For tuning a parameter, I have **chosen** to tune the **‘**number of estimators**’**, which represents the number of trees in the forest.

# of estimators	**precision**	**recall**	**F1-score**	**accuracy**
10	0.74	0.69	0.73	0.73
50	0.74	0.68	0.73	0.72
100	0.75	0.693	0.73	0.73
200	0.74	0.689	0.73	0.72

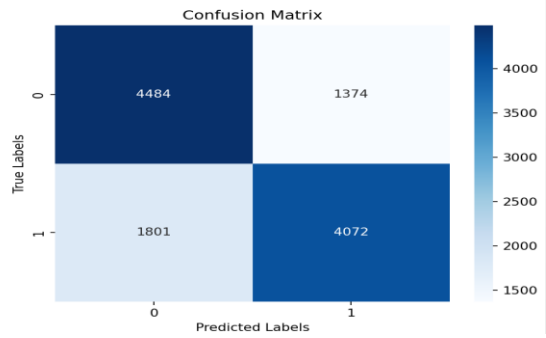
# of estimators = 10



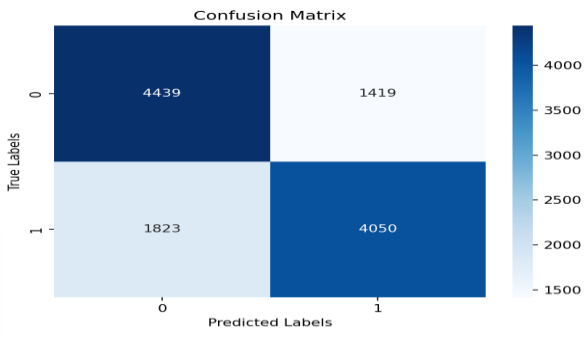
# of estimators = 50



# of estimators = 100



# of estimators = 200



Comments

Precision: It remained almost stable across different numbers of estimators, with a little increase when using 100 estimators.

Recall: It peaked at 100 estimators, showing that this setup was slightly better at recognizing true positives.

F1-score: It remained constant across all configurations, indicating a stable balance between precision and recall despite changes in the number of estimators.

Accuracy: decreases a little when the number of estimators increases from 10 to 50 and 100 to 200. However, the change is small.

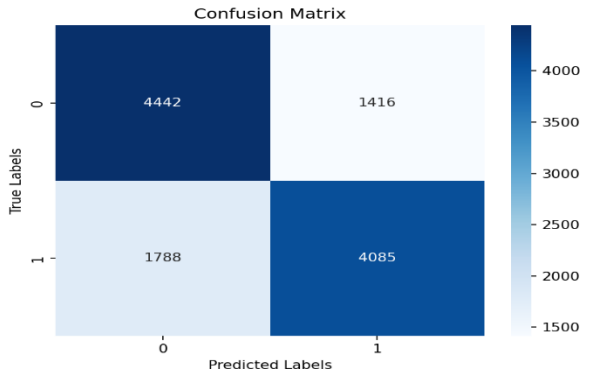
Discuss results:

I have found that changing the number of estimators in the model doesn't make a big difference. It's like turning the volume up or down a little – you don't really notice much change. The best results came with using 100 estimators, but the improvement was quite small.

Also, the model's accuracy and ability to identify correctly (precision and recall) stayed almost the same, no matter how many estimators we used. This means my data isn't very sensitive to the number of estimators, or maybe I was already in a range where the number doesn't change performance much.

Extra: I have tried a model **without deleting** any feature, using **1000** estimator, function **= 'gini'**,and max depth **500**, and the results was:

Precision: 0.76, **recall** = 0.70, **accuracy** = 0.73.



When compared to the previous results where the number of estimators varied from 10 to 200, the recall has improved from a maximum of 0.693 to 0.70. This indicates a better ability to identify positive cases. However, the precision increased a little while accuracy has remained the same, showing that increasing the number of estimators and the max depth doesn't extremely improve the model performance.

In conclusion, the increase in recall with small change in precision indicates that the model has improved a little bit such as getting less errors in classifying the negative class as positive. However, the lack of huge improvement in accuracy and precision shows that just adding additional estimators and depth may not be the best way to get better performance.

Increasing the number of estimators from 10 to 50 results in a small improvement but increasing the number of estimators more didn't really cause a major impact on the model's performance. This means I need to find other models.

Logistic regression [Best Model]

In this model I am using logistic regression to train my model. The parameters that can be tuned in this model are: 1- solver = [‘saga’, ‘liblinear’,’lbfgs’,’newton-cg’], 2- class_weight, 3- penalty=[‘l1’, ‘l2’], 4- C, 5-max_iter.

C: represent the regularization parameter. L: represents the penalty (type of regularization). Class_weight: represents the weight of each class. Solver: represents the optimization algorithm. Max_iter: maximum number of iterations.

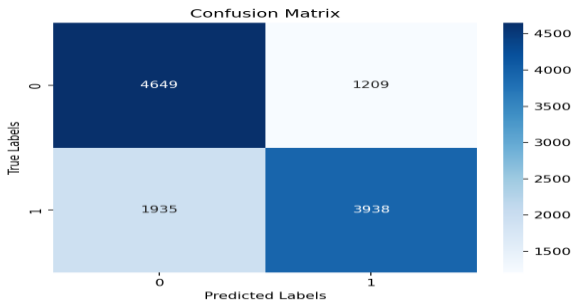
The parameter that I have decided to tune is: **Class weight.** This parameter could be assigned to ‘balanced’, or can be assigned to a specific weight for each class.

Parameters model I am using: 1- solver: ‘saga’. 2- penalty=’l2’ to prevent overfitting. 3- C=10. Max_iter=3000.

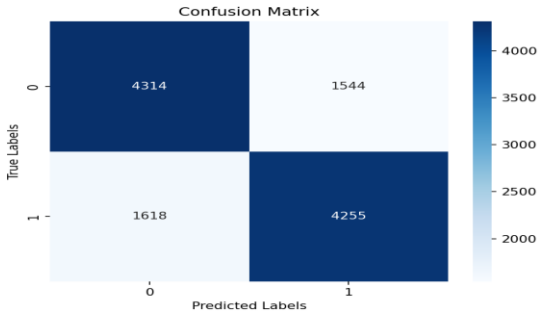
The results I gain:

Class_weight	precision	recall	F1-score	accuracy
Balanced	0.765	0.67	0.71	0.73
1:1.2	0.733	0.724	0.72	0.73
1:1.5	0.68	0.8	0.74	0.71
1:2	0.629	0.87	0.73	0.67

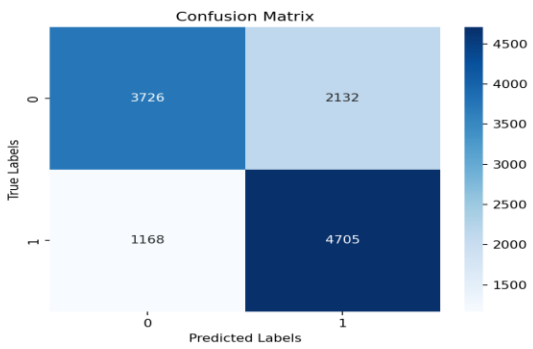
Class_weight=’balanced’



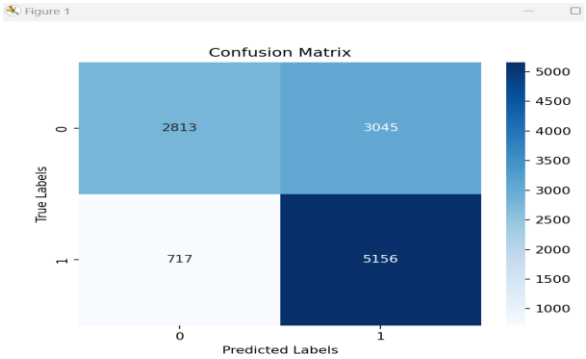
class_weight=’1:1.2’



Class_weight=’1:1.5’



Class_weight=’1:2’



Conclusion

The configuration with a class_weight of 1:1.5 seems to offer the best compromise, achieving a high recall of 0.8 while maintaining a reasonable balance with precision and accuracy.

To get a prediction model for cardiovascular disease, I have used logistic regression to maximize recall while minimizing false negatives. The model employs a 'saga' solver, which is optimized for large datasets, and utilizes an L2 penalty to prevent overfitting, with a regularization strength (C) of 10 and a maximum iteration threshold set at 3000.

Balanced Class Weight: it provides an appropriate trade-off between precision and recall, but it does not entirely meet the wanted recall for medical diagnostic applications.

1:1.2 Class Weight: it improves recall, but it decreases precision. leaning a little towards the less common group helps the model without making it less good at diagnosing.

1:1.5 Class Weight: In this case, the model gets much better at finding what it should (recall is now 0.8), especially in spotting the right cases. But this makes it a bit less accurate overall. Still, this trade-off might be okay because it's important to correctly identify diseases.

1:2 Class Weight: the model is best at detecting cases (recall is 0.87), really focusing on finding them. However, the decrease in precision and accuracy indicates a high number of false positives, which could potentially lead to unnecessary and additional testing for patients.

In summary, the performance of the model changes with the change of the parameter. A balanced setting is okay but not great for medical needs where finding every case is crucial. A 1:1.5 setting really helps in spotting diseases, even though it's not as precise. The best detection is with a 1:2 setting, but it's not as accurate and might give too many false alarms. Overall, a 1:1.5 setting is the best choice for medical tests because it finds cases well and still stays fairly accurate.

Performance Analysis

Performance Analysis Summary: The logistic regression model made mistakes in 4974 cases when tested. Here's what I learned from these mistakes:

Age and Gender: Most errors were with middle-aged to older people, averaging around 54.5 years old, and 64% were in cases involving women.

Body Size: The average height and weight were 164.78 cm and 74.94 kg, which are normal.

Blood Pressure: The average blood pressure readings were also normal (124.64/80.82 mmHg).

Health Levels: Cholesterol and glucose were close to normal, with averages of 1.37 and 1.26.

Lifestyle: Only a few of the wrong cases were smokers (8.12%) or drank alcohol (4.72%). But a large part (79.12%) was from active people.

Model's Mistakes: About 58.6% of the errors were actual heart disease cases marked as no disease, and 41.4% were the opposite.

Note: shown in the code how I reached these statistics.

The model has problems finding heart disease in certain people. This includes women who are middle-aged or older, as well as people with normal blood pressure, cholesterol, and sugar. It also doesn't do well in checking the heart health of those who exercise a lot. Interestingly, smoking and drinking don't seem to cause these errors much. This means there are other reasons why the model isn't always right. I tried tunning other parameters such as C, number of iterations or even the solver, but it wasn’t that big jump in performance.

In conclusion
Trying to balance two things—recall and precision - is hard. When I make one better, the other gets worse. When I change the settings for balancing these two, it's still tough. Even when I find a middle ground, it's not perfect for medical use. Focusing more on the less common cases helps a bit, but then the overall accuracy drops. The system isn't as good at predicting issues for middle-aged and older women, and for people who are very active. It also struggles when someone's blood pressure, cholesterol, and sugar levels are normal. This might be because it's biased or lacks certain information for these groups. The system does okay with people who smoke or drink, showing these habits don't affect its predictions much.

Conclusion

In this project, I worked on making a model to predict heart disease. I tried different methods like logistic regression, K-Nearest Neighbors (KNN), and Random Forest. Out of these, logistic regression worked the best.

For the evaluation matrices, I have used Accuracy, precision, recall, and F1-score. accuracy (how often the model is right), Precision (how good the model is at identifying actual cases), Recall (how well the model finds all the cases), F1-Score (a balance between Precision and Recall), and the Confusion Matrix (a table showing the model's correct and wrong guesses). These methods helped understand how well each model worked in predicting the disease.

The focus was on making sure the predictions avoided as many false negatives as possible. Logistic Regression was the top performer. I found that adjusting the class weight ratio to 1:1.5 in Logistic Regression gave the best results. While trying to balance recall and precision was hard.

Also, I found some key challenges. Each Model has its drawbacks, like KNN's sensitivity to noisy data and Random Forest’s limited performance improvement by tunning the number of estimators.

In the end, I learned a lot from this project. The logistic regression model did a good job, especially in finding heart disease cases, but there's still room to make it better. This kind of work is important because it helps doctors catch heart disease early, which can save lives.