Cardiovascular Disease Prediction Algorithm

Abstract

Cardiovascular diseases are the main cause of death around the world. The most recent estimations shows that Cardiovascular will be responsible for the deaths of about 23 million people be the year 2030. Early diagnosis can save lives, and it will reduce the cost of treatment if disease is found in early stage. The practitioners equipped with additional tools like Artificial Intelligence prediction systems can speed up their diagnosis work and focus more on treatments or other necessary tasks. Proposed Artificial Intelligence systems are predicting cardiovascular disease by using three different classifiers: Random Forest, Logistic Regression and KNeighbours. It is also implementing advanced training data preparation for better model accuracy. The dataset used for this project was acquired from (1) Kaggle, it consists of 70k entries and 12 features. Dataset has 3 types of input: objective: factual information; Examination: results of medical examination; Subjective: information given by the patient.

Introduction

Cardiovascular disease is main reason for deaths around the world by (2) World Health Organization causes around 17.9 million lives each year. “The most important behavioral risk factors of heart disease and stroke are unhealthy diet, physical inactivity, tobacco use and harmful use of alcohol. Amongst environmental risk factors, air pollution is an important factor. The effects of behavioral risk factors may show up in individuals such as raised blood pressure, raised blood glucose, raised blood lipids, and overweight and obesity. These “intermediate risks factors” can be measured in primary care facilities and indicate an increased risk of heart attack, stroke, heart failure and other complications.” Early detection of disease can save lives and help doctors with decision making, reducing heart failures. (3) Study in field show that machine learning algorithms are extremely helpful as predictors and can enchant practitioner’s work. In Ireland (4) “Cardiovascular disease is estimated to cost the Irish state €1.7 billion per annum, almost half of which are direct health care costs.”. Artificial intelligence systems can significantly reduce these costs. In today world of big data and Internet of Things is much easier to monitor and collect patients’ data, than use it with Artificial intelligence to support healthcare in their tasks as Artificial Intelligence algorithms can go through big datasets faster than humans.

Related Work

Many researchers have used Machine Learning and Artificial Intelligence Algorithms to perform cardiovascular disease predictions. (5) In this research during data preparation outliers were removed from (6) dataset which is different

dataset. Multiple data mining was performed to find the best features, Random Forest was used to build model, and they got after employing several features selection methods 99% accuracy, where other tested models in their research got: SVM: 85%, KNN: 95%, Logistic regression: 86%. The method to find best features provides good results as other researchers on this dataset without features selection were getting 92% accuracy. In this research (7) few models were used for testing like Random Forest, CART, Extra Tree Classifier, and Extreme Gradient Boosting with best accuracy 91.9%. This was done on different dataset got it form Google Colab with features selection and best model performance choose. In this research (8) few models were used with accuracy like: Lasso-AdaBoost: 81%, FLR-L1-LR: 83%, FLR-RF: 82%, FLR-SVM: 83%. They performed Bayesian optimization and grid search to get the best hyperparameters. These prediction models were trained and tested on different dataset. In this research (9) 4 different models were tested: SVM with accuracy 82%, KNN with accuracy 86%, Logistic Regression with accuracy 87% and best Random Forest with accuracy 96%. For the testing feature selection was performed. In most cases the best results were met by performing features selections and picking the right hyperparameters.

Methodology

For this project 3 models were used: Random Forest, KNeighbors and Logistic Regression. (10) Random Forest is an ensemble learning method for classification, regression and other tasks where many decision trees are built during training time, than for classification output of random forest is class that was chosen by most of the trees. For regression, the mean prediction of each tree is returned. KNeighbors is a simple machine learning algorithm that categorizes input by using its k(number of) nearest neighbors, it is providing combination of simplicity and effectiveness. Logistic Regression is great for situations where you need to classify between two categories. For these models 3 function was created:

```
# Random Forest
def RandomForest_fun():
    start_time = datetime.now()
    RandomForestModel = RandomForestClassifier(n_estimators=100, max_depth=10, min_samples_split=10, min_samples_leaf=1, random_
    RandomForestModel.fit(x_train, y_train)
    RandomForestModel_pred = RandomForestModel.predict(x_test)
    RandomForestModel_accuracy = metrics.accuracy_score(y_test, RandomForestModel_pred)*100
    end_time = datetime.now()
    cm = confusion_matrix(y_test, RandomForestModel_pred, labels= RandomForestModel.classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=RandomForestModel.classes_)
    disp.plot()
    print('Duration: {}'.format(end_time - start_time))
    print("R2 score: {0}".format(r2_score(RandomForestModel_pred, y_test)))
    return print(f"Accuracy Random Forest: {RandomForestModel_accuracy:.2f}")
```

```
# Logistic Regression
def LogisticRegresion_fun():
    start_time = datetime.now()
    LogisticRegresionModel = LogisticRegression(max_iter = 1000)
    LogisticRegresionModel.fit(x_train, y_train)
    LogisticRegresionModel_pred = LogisticRegresionModel.predict(x_test)
    LogisticRegresionModel_accuracy = metrics.accuracy_score(y_test, LogisticRegresionModel_pred)*100
    end_time = datetime.now()
    cm = confusion_matrix(y_test, LogisticRegresionModel_pred, labels= LogisticRegresionModel.classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=LogisticRegresionModel.classes_)
    disp.plot()
    print('Duration: {}'.format(end_time - start_time))
    print("R2 score: {0}".format(r2_score(LogisticRegresionModel_pred, y_test)))
    return print(f"Accuracy Logistic Regresion: {LogisticRegresionModel_accuracy:.2f}")
```

```
# KNeighbors
def KNeighbors_fun():
    start_time = datetime.now()
    KNeighborsModel = KNeighborsClassifier()
    KNeighborsModel.fit(x_train, y_train)
    KNeighborsModel_pred = KNeighborsModel.predict(x_test)
    KNeighborsModel_accuracy = metrics.accuracy_score(y_test, KNeighborsModel_pred)*100
    end_time = datetime.now()
    cm = confusion_matrix(y_test, KNeighborsModel_pred, labels= KNeighborsModel.classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=KNeighborsModel.classes_)
    disp.plot()
    print('Duration: {}'.format(end_time - start_time))
    print("R2 score: {0}".format(r2_score(KNeighborsModel_pred, y_test)))
    return print(f"Accuracy KNeighbors: {KNeighborsModel_accuracy:.2f}")
```

To prepare data for training and testing a few methods were used. First checking for missing data values was done:

```
# checking for missing values
print(dataset.isnull().sum())
```

```
id             0
age            0
gender         0
height         0
weight         0
ap_hi          0
ap_lo          0
cholesterol    0
gluc           0
smoke          0
alco           0
active         0
cardio         0
dtype: int64
```

Then column “Id” was dropped. After describing dataset was founded that max/low height, weight, ap_hi and ap_lo are looks like out of bounds, kind of unusual values:

```
dataset.describe()
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	a
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.00
mean	19468.865814	1.349571	164.359229	74.205690	128.817286	96.630414	1.366871	1.226457	0.088129	0.053771	0.80
std	2467.251667	0.476838	8.210126	14.395757	154.011419	188.472530	0.680250	0.572270	0.283484	0.225568	0.39
min	10798.000000	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000	0.000000	0.00
25%	17664.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.00
50%	19703.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.00
75%	21327.000000	2.000000	170.000000	82.000000	140.000000	90.000000	2.000000	1.000000	0.000000	0.000000	1.00
max	23713.000000	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000	1.000000	1.00

These outliers were removed by removing everything that is outside of range of 3% to 97%:

```
# Removing outliers, , we can see that max/low height, weight, ap_hi and ap_lo are looks like out of bounds, we will remove even
dataset.drop(dataset[(dataset['height'] > dataset['height'].quantile(0.97)) | (dataset['height'] < dataset['height'].quantile(0.(
dataset.drop(dataset[(dataset['weight'] > dataset['weight'].quantile(0.97)) | (dataset['weight'] < dataset['weight'].quantile(0.(
dataset.drop(dataset[(dataset['ap_hi'] > dataset['ap_hi'].quantile(0.97)) | (dataset['ap_hi'] < dataset['ap_hi'].quantile(0.03))]
dataset.drop(dataset[(dataset['ap_lo'] > dataset['ap_lo'].quantile(0.97)) | (dataset['ap_lo'] < dataset['ap_lo'].quantile(0.03))]
len(dataset)
```

By removing these outliers 57828 entries left in dataset and after running describe dataset looks ok:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	a
count	57828.000000	57828.000000	57828.000000	57828.000000	57828.000000	57828.000000	57828.000000	57828.000000	57828.000000	57828.000000	57828.00
mean	19491.173739	1.350194	164.620340	73.658834	126.236425	81.682334	1.354119	1.221605	0.085581	0.051947	0.80
std	2452.047137	0.477035	6.821789	11.426754	13.544854	7.505061	0.673139	0.569349	0.279747	0.221922	0.39
min	10859.000000	1.000000	150.000000	53.000000	100.000000	66.000000	1.000000	1.000000	0.000000	0.000000	0.00
25%	17732.000000	1.000000	160.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.00
50%	19716.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.00
75%	21333.000000	2.000000	170.000000	81.000000	140.000000	90.000000	1.000000	1.000000	0.000000	0.000000	1.00
max	23713.000000	2.000000	180.000000	105.000000	160.000000	100.000000	3.000000	3.000000	1.000000	1.000000	1.00

In next step converting age from days to years was done:

```
dataset['age'] = (dataset['age'] / 365).round().astype('int')

print(dataset.head())
```

```
   age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  \
0   50       2     168    62.0    110     80            1     1      0     0
1   55       1     156    85.0    140     90            3     1      0     0
2   52       1     165    64.0    130     70            3     1      0     0
3   48       2     169    82.0    150    100            1     1      0     0
5   60       1     151    67.0    120     80            2     2      0     0
```

Next categorizing age was performed, and that column was added to dataset:

```
age_edges = [30, 35, 40, 45, 50, 55, 60, 65]
age_labels = [0, 1, 2, 3, 4, 5, 6]

# bin in  5 years span
dataset['age_group'] = pd.cut(dataset['age'], bins=7, labels=range(7), include_lowest=True,right=True)
dataset.head()
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	age_group
0	50	2	168	62.0	110	80	1	1	0	0	1	0	3
1	55	1	156	85.0	140	90	3	1	0	0	1	1	4
2	52	1	165	64.0	130	70	3	1	0	0	0	1	4
3	48	2	169	82.0	150	100	1	1	0	0	1	1	3
5	60	1	151	67.0	120	80	2	2	0	0	0	0	5

Next step was adding additional column BMI which is Body Mass Index that is combination of weight and Hight and is more valuable feature than other 2 which may confuse model:

```
dataset['bmi'] = dataset['weight']/((dataset['height']/100)**2)
dataset.head()

bmiMin = int(dataset['bmi'].min())
bmiMax = int(dataset['bmi'].max())

print(bmiMin, bmiMax)

dataset['bmi'] = pd.cut(dataset['bmi'], bins=6, labels=range(6), right=True, include_lowest=True)
```

Next step was adding additional column MAP which is Mean Arterial Pressure, this way we can eliminate 2 columns 'ap_lo' and 'ap_hi':

```
dataset['map'] = ((2* dataset['ap_lo']) + dataset['ap_hi']) / 3

mapMin = int(dataset['map'].min())
mapMax = int(dataset['map'].max())

print(mapMin, mapMax)

dataset['map'] = pd.cut(dataset['map'], bins=6, labels=range(6), right=True, include_lowest=True)
```

Then removing unnecessary columns 'height','weight','ap_hi','ap_lo','age' which we replaced by the ones that were generated. Another step was doing label encoder:

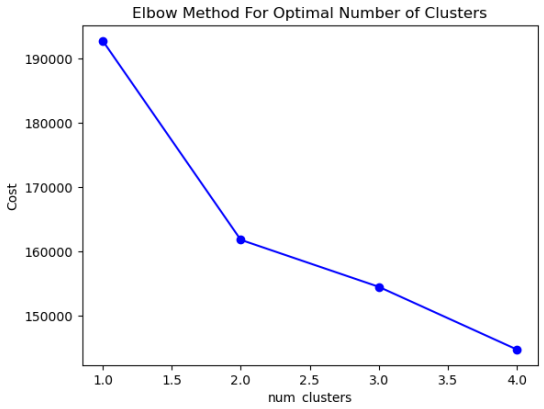
```
label_encoder = preprocessing.LabelEncoder()
dataset = dataset.apply(label_encoder.fit_transform)
```

As next step clustering was done to get better performance of our models, for this K-modes was used as clustering algorithm, clustering is a type of unsupervised learning, mean that we don't need label data for clustering algorithms:

```
cost = []
number_of_clusters = range(1,5) # 1 to 4
for i in list(number_of_clusters):
    kmode = KModes(n_clusters=i, init = "Huang", n_init = 5, verbose=0,random_state=1)
    kmode.fit_predict(dataset)
    cost.append(kmode.cost_)
```

To find best number of clusters for our dataset Elbow method was used:

```
plt.plot(number_of_clusters, cost, 'bo-')
plt.xlabel('num_clusters')
plt.ylabel('Cost')
plt.title('Elbow Method For Optimal Number of Clusters')
plt.show()
```



We can see that optimal is 2 clusters. Next step was built K-modes model and insert to our dataset:

```
km = KModes(n_clusters=2, init = "Huang", n_init = 5,random_state=1)
clusters = km.fit_predict(dataset)
clusters
```

```
array([1, 0, 1, ..., 1, 0, 1], dtype=uint16)
```

```
dataset.insert(0,"clusters",clusters,True)
```

After all this preparation our dataset looks like this:

	clusters	gender	cholesterol	gluc	smoke	alco	active	cardio	age_group	bmi	map
0	1	1	0	0	0	0	1	0	3	1	1
1	0	0	2	0	0	0	1	1	4	3	4
2	1	0	2	0	0	0	0	1	4	1	1
3	0	1	0	0	0	0	1	1	3	2	5
5	0	0	1	1	0	0	0	0	5	2	2

As next step was to split dataset to features and to result:

```
x = dataset.drop(['cardio'], axis=1)
y = dataset['cardio']
```

Next step was to split data into train and test data, for this we used 20% for test and random state 1:

```
x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.20, random_state=1)
```

For Random Forest model these parameters were used:

```
n_estimators=100, max_depth=10, min_samples_split=10, min_samples_leaf=1, random_state=1
```

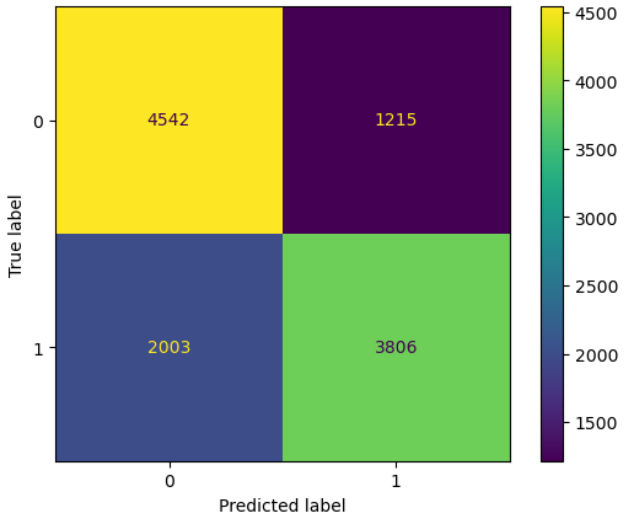
To check duration of models datatime function was implemented, that was checking time before and after model finish his task.

Evaluation

For testing 2 files were created on Jupiter Notebook: one with no preparation just removing outliers and doing label encoder; second one all methods implemented in Methodology part. We can see a huge difference in the performance of these approaches.

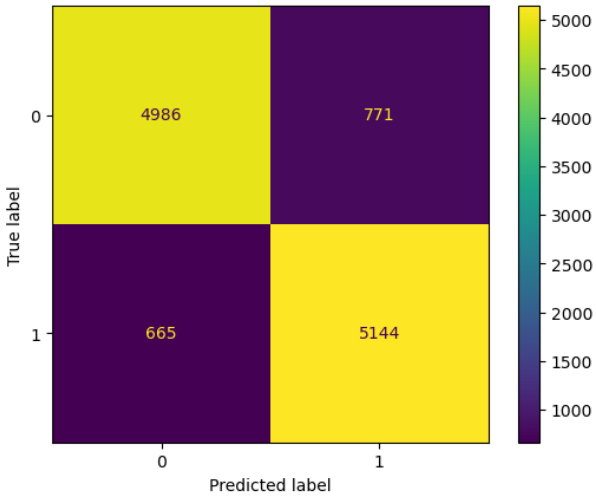
Random Forest in first approach got:

```
Sklearn Precision =  0.758016331408086
Duration: 0:00:01.402186
R2 score: -0.132581218469898
Accuracy Random Forest: 72.18
```



Where on second approach Random Forest got:

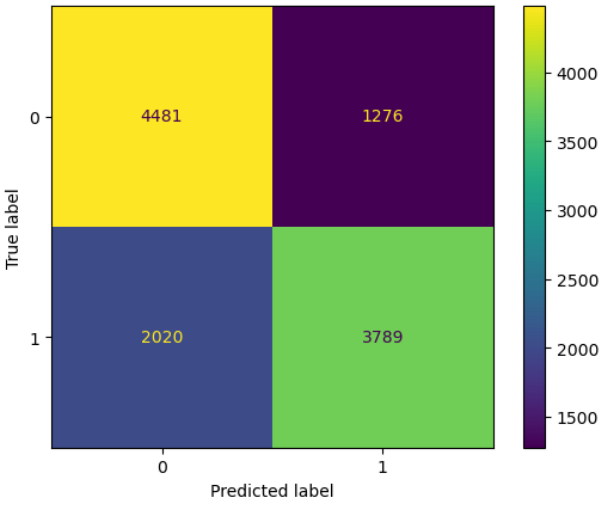
```
Sklearn Precision =  0.8696534234995773
Duration: 0:00:00.883841
R2 score: 0.5031130719463623
Accuracy Random Forest: 87.58
```



We see huge accuracy increase from 72.18% to 87.58%, precision increase from 75.8% to 87%, duration time to perform training and prediction drop from 1.40 seconds to 0.88 seconds and R2 (which is coefficient measure) score increase from -0.13 to 0.50.

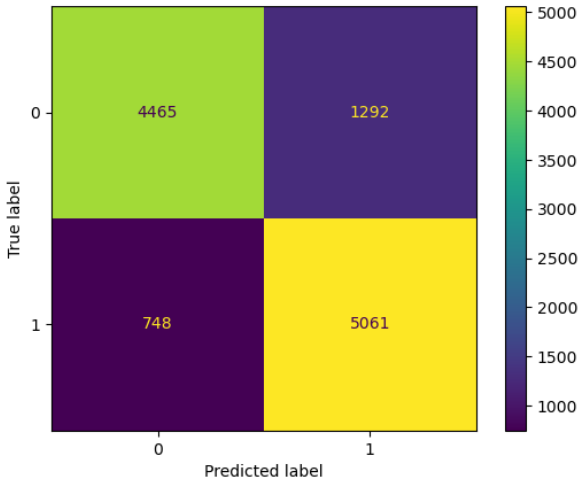
Logistic Regression on first approach got:

```
Sklearn Precision =  0.7480750246791708
Duration: 0:00:00.221280
R2 score: -0.157739298365974
Accuracy Logistic Regresion: 71.50
```



Where on second approach Logistic Regression got:

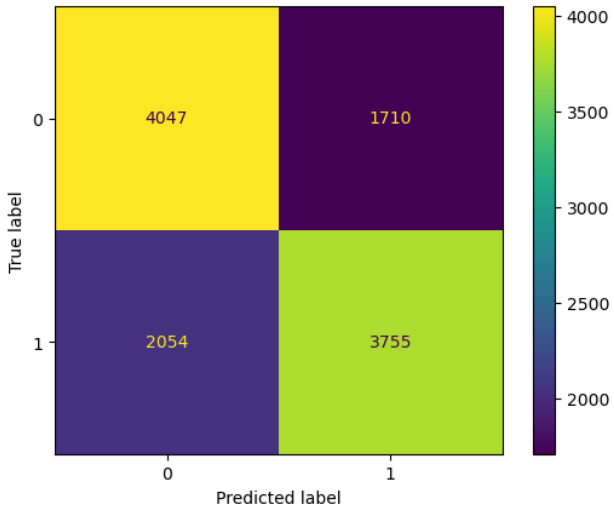
```
Sklearn Precision =  0.7966315126711789
Duration: 0:00:00.044430
R2 score: 0.2875624932269095
Accuracy Logistic Regresion: 82.36
```



For Logistic Regression we see accuracy increase from 71.5% to 82.36%, precision increase from 74.8% to 79.6%, duration time to perform training and prediction drop from 0.221 seconds to 0.044 seconds and R2 (which is coefficient measure) score increase from -0.16 to 0.29.

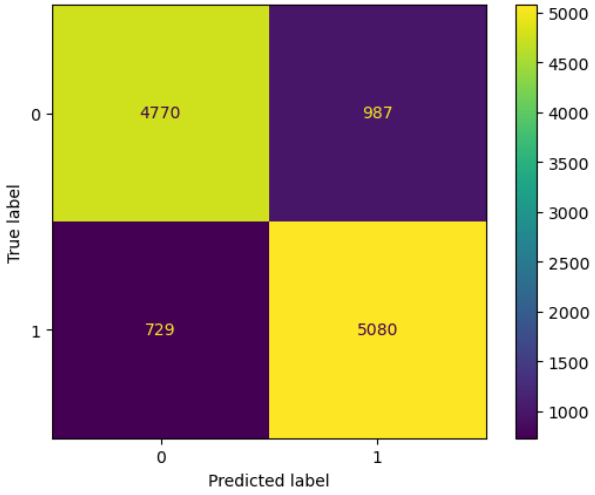
KNeighbors on first approach got:

```
Sklearn Precision =  0.687099725526075
Duration: 0:00:00.461355
R2 score: -0.3056946103806415
Accuracy KNeighbors: 67.46
```



Where on second approach KNeighbors got:

```
Sklearn Precision =  0.8373166309543432
Duration: 0:00:00.655591
R2 score: 0.40510166030157346
Accuracy KNeighbors: 85.16
```



For KNeighbors we see accuracy increase from 67,46% to 85.16%, precision increase from 68.7% to 83.73%, duration time to perform training and prediction increase from 0.461 seconds to 0.655 seconds and R2 (which is coefficient measure) score increase from -0.30 to 0.40.

From testing we saw that the biggest impact on accuracy and precision had clustering method, it provide around 10% increase for all models. Looks like clustering perform good classification on this type of datasets and help models perform better. Removing outliers add little bit to overall

performance. We can see that adding few methods to preparation of dataset for models give huge impact on performance and can help achieve better results.

Conclusion and Future work

This project shows that different approaches can provide better or worst performance results. Having more time will be nice to do more tests on various methods and see their performances changes. Spend more time to find best hyperparameters and see how they affect performance. Try more classifier types or use genetic algorithms and see how this type performs. In this project on this dataset highest accuracy was 87.6% and will be nice to see this increase more when additional technics will be applied. After all proposed model display quite high accuracy and looks promising tool for early cardiovascular detection.

REFERENCES

1. https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset
2. https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1
3. https://www.frontiersin.org/journals/medicine/articles/10.3389/fmed.2023.1150933/full
4. https://irishheart.ie/wp-content/uploads/2023/11/IHF_CVD_PP_Summary_SCREEN.pdf
5. https://www.sciencedirect.com/science/article/pii/S2352914823001624
6. https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset/data
7. https://www.researchgate.net/publication/372521974_Cardiovascular_Disease_Prediction_Using_Machine_Learning_Approaches
8. https://www.frontiersin.org/journals/cardiovascular-medicine/articles/10.3389/fcvm.2022.854287/full
9. https://www.sciencedirect.com/science/article/pii/S2352914823001624
10. https://en.wikipedia.org/wiki/Random_forest