Heart Failure Prediction

Aseel Mustafa Ahmad Abdel-Qader
Computer Engineering Department
The University Of Jorden
Amman, Jordan
asy0195792@ju.edu.jo

***Abstract*— Cardiovascular diseases kill approximately 17 million people globally every year. and they mainly exhibit as heart failure. Heart failure occurs when the heart cannot pump enough blood to meet the needs of the body.so, people with cardiovascular disease or who are at high cardiovascular risk need early detection.**

***Keywords*—heart failure, model, data set, binary classification**

I. Introduction

The artificial intelligence algorithms are spired by the human brain that learns from a large amount of data, so AI helps the doctors detect the illness and analysis.

One of the significant benefits of machine learning in healthcare is the classification and analysis of illnesses, it made them more manageable as it was difficult to diagnose. Therefore, I select the Heart failure data set to predict the heart disease using classical techniques of binary classification.

Heart failure is a common event and it's the number one cause of death globally for this reason we need early detection and management, so we use the machine learning model to help us. In this binary classification problem, the model takes multiple integer features, and it uses this feature to predict the response feature "HeartDisease" in which if the output is 1; the person has heart disease else the person hasn't heart disease (normal).

II. Data Description

In this heart disease problem, I solved it using the binary classification, so I used the dataset that I found on the Kaggle website this dataset contains 12 columns and 918 rows, but the columns contain numerical and text data that's why I need to convert the text data to numerical data to solve this problem inefficient way [1].

I want to use many packages in python to solve this problem such that pandas, NumPy, and sci-kit learn. scikit learn is a library that is used in machine learning, and it focused on modeling the data it is open source and easy to use, so I will use this library to solve the heart disease problem in a simple and easy way.

Fig. 1. The Heart Disease

III. The Steps of Machine Learning

To solve any machine learning problem, we must follow these steps:

A. *Get The Data*

At first, I read the data set and got acquainted with it, it contains 12 columns including the response feature, and 918 rows each of these features means as follows [1]:

1) *Age: age of the patient*
2) *Sex: sex of the patient [Male or Female]*
3) *ChestPainType: chest pain type*
4) *RestingBP: resting blood pressure*
5) *Cholesterol: serum cholesterol*
6) *FastingBS: fasting blood sugar*
7) *RestingECG: resting electrocardiogram results*
8) *MaxHR: maximum heart rate achieved*
9) *ExerciseAngina: exercise-induced angina [Yes or No]*
10) *Oldpeak: oldpeak = ST [Numeric value measured in depression]*
11) *ST_Slope: the slope of the peak exercise ST segment*
12) *HeartDisease: output class [1: heart disease, 0: Normal]*

After that, I downloaded the dataset from the Kaggle website and then used the read_csv () function in the panda's package to read this dataset into the data frame file.

Then we have multiple methods in python that help us to take a quick look at the data structure I used two functions head () method and the info () method. So when I used the header method, which showed the first five rows of the data set, the info() method is useful to get a quick description of the data.

TABLE I.
THE INFORMATION ABOUT THE DATA SET

# Column	Non-Null Count	Dtype
Age	918 non-null	int64
Sex	918 non-null	object
ChestPainType	918 non-null	object
RestingBP	918 non-null	int64
Cholesterol	918 non-null	int64
FastingBS	918 non-null	int64
RestingECG	918 non-null	object
MaxHR	918 non-null	int64
ExerciseAngina	918 non-null	object
Oldpeak	918 non-null	float64
ST_Slope	918 non-null	object
HeartDisease	918 non-null	int64

dtypes: float64(1), int64(6), object (5)

Table 1 shows the output when I used the info methods, I concluded from this output the data set hasn't any missing features but it is mixed of the object, integer, and floating data types so I need to convert the object type to be integers.

B. *Discover the data to gain insights*

In this step, I will be focused on the correlation, so I computed the standard correlation coefficient between every pair of attributes using the efficient method in python which is the corr () method.

TABLE II.
CORRELATION BETWEEN THE FEATURE

The Attributes	The Correlation
HeartDisease	1.000000
Oldpeak	0.403951
Age	0.282039
FastingBS	0.267291
RestingBP	0.107589
Cholesterol	-0.232741
MaxHR	-0.400421

Name: HeartDisease, dtype: float64

Table 2 shows the output when I used the corr() method in the data set, the correlation is applied only to the float or integer data type and it makes the normalization of the data to be the output in the range 1 to -1, so if the output is negative values that mean the relationship between the feature and response is an inverse relationship, as is the Cholesterol and MaxHR features. so, in this output, I conclude that the highest correlation between the old peak feature with the response feature and the lowest correlation with the RestingBP feature.

Note that, the Oldpeak feature refers to a finding on an electrocardiogram.

C. *Prepare The Data For Machine Learning Algorithms*

This step has multiple steps to prepare the data before applying any machine learning algorithm

- Split the data set to train and test set:

  I split the data set into two separate data sets one for the train set and the other for the test set using the train_test_split method in sklearn.model_selection packages and I choose the test size to be 0.2 which means the train set contains 80% of the original data set.

- Separate the features from the response:

  I separate the feature from the response on the train and test set, so the shape of the feature in the train set becomes [734 rows x 11 columns].

- Handle text and categorical features:

  Most machine learning algorithms prefer to work with numbers therefore I convert the text in this data set to be the numeral feature using the OrdinalEncoder class.

```
ordinal_encoder = OrdinalEncoder()
x_enco=ordinal_encoder.fit_transform(x_cate)
ordinal_encoder.categories_
```

Fig. 2. The ordinal encoder code

Fig.2 shows the python code that started to make the object from the OrdinalEncoder class then I apply the fit_transform method on this object and this method returns the results in a single column of integers (0 to n_categories - 1) per feature.

The last line in fig.1 returns the list of the array which contains the integer value for each unique category.

- Build preparation pipeline:

  The purpose of the pipeline is to assemble several steps that can be cross-validated together, for this we can define a pipeline and apply it in the train set and test set and any new instance rather than write separate code to prepare the train or test set.

  Fig 3 shows the pipeline code First, I separate the attribute into two list category attributes and a normal attribute then I defined a pipeline using the ColumnTransformer class.

  The ColumnTransformer class take list of tuples and each tuple contains a name, a transformer, and a list of names of columns that the transformer should be applied to. so, I specify the numerical columns should be transformed using the StandardScaler class, and the categorical columns should be transformed using an OrdinalEncoder class.

  As I said before, I used the StandardScaler class to transform the numerical columns because many Machine Learning algorithms don't perform well when the input numerical attributes have very different scales. This is the case for HeartDisease data set.

  Finally, I applied this ColumnTransformer to the HeartDisease data: it applies each transformer to the appropriate columns and concatenates the outputs in one data frame. so, in the train set, I import the fit_transforme method to allow the pipeline to train the data and to get the overview of the data then to be able to transform it, but in the test set, I use the transform method.

```
cat_attribs = ["Sex", "ChestPainType", "RestingECG",
"ExerciseAngina", "ST_Slope”]

num_attribs = ["Age", "RestingBP", "Cholesterol",
"FastingBS", "MaxHR", "Oldpeak"]
full_pipline = ColumnTransformer ([("num",
StandardScaler(), num_attribs), ("cat",
OrdinalEncoder(), cat_attribs)]

data_prepared = full_pipline.fit_transform(x_train)
x_test_pre = full_pipline.transform(x_test)
```

Fig. 3. The Pipeline code

```
svc_model = SVC()
svc_model.get_params()
param_grid_svm={'C': [0.1,0.5,1,2,10],'gamma':['scale','auto'],'kernel': ['linear','rbf','poly','sigmoid']}
svc_grid_model= GridSearchCV(svc_model,param_grid_svm,cv=5)
svc_grid_model.fit(data_prepared,y_train)
print(svc_grid_model.best_params_)
svc_pred= svc_grid_model.predict(data_prepared)
accr = accuracy_score(y_train,svc_pred)
print(accr)
```

Fig. 4. The Support Vector Machine Code

D. *Select A Model And Train It*

In this step, I tried to solve this problem using multiple classical techniques and evaluate the best technics, I used the accuracy score to determine the best.

1) Support Vector Machine (SVC)

This technique very powerful and versatile machine learning model, capable of performing linear and nonlinear classification, and the SVC model gives a large margin classification.

Fig 4 shows the support vector machine code that I used First, I make the object from the SVC () class and then train this object to the train data set that was prepared earlier, after that, I prepared the prediction data using the predict method. when I used the accuracy score the accuracy was 85.6 %.

Then, I tried to use the hyperparameter to get higher accuracy. so, I used the GridSearchCV class to search for the best hyperparameter. Only I need to detect which hyperparameters to experiment with, and what values to try out, and it will evaluate all the possible combinations of hyperparameter values, using cross-validation.

Therefore, as Fig 4 shows I prepared the param_grid which takes the direction type that includes the key and value so, I set different values for different keys C, kernel, and gamma parameters.

The grid search will explore 11 combinations of support vector machine hyperparameter values, and it will train each model five times because I used the cv parameter equal to 5, when it is done, I print the best hyperparameter using the best_params_ method and the output was {'C': 2, 'gamma': 'auto', 'kernel': 'rbf'} [2].

After that, I got higher accuracy 89% when I evaluate the train set and 86% when I evaluate the test set. I think this is a good solution.

2) Logistic Regression

Logistic Regression is commonly used for classification, this model computes a weighted sum of the input features, but instead of outputting the result directly as the Linear Regression model does, it outputs the logistic of this result.

I prepared the object of this model and trained this object in the train set using the fit method then I evaluate the model on the train set using the accuracy score and got 85.5%, but when I evaluate this model on the test set, I got 84.2% of the accuracy.

3) Decision Tree

Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks and even multioutput tasks, and it is very powerful algorithms, capable of fitting complex datasets.

I prepared the object of this model, and I tried this model in multiple values of max_depth hyperparameters then I found the best max_depth equal to 4, After that, I trained this object in the train set using the fit method then I evaluate the model on the train set using the accuracy score and got 87.5%, but when I evaluate this model on the test set, I got 86.9% of the accuracy.

4) Random Forests Classifier

Random Forests work by training many Decision Trees on random subsets of the features, then averaging out their predictions.

Therefore, I tried to use this model on the HeartDisease data set to achieve higher accuracy. so, I prepared the object using random forest class, then I trained this object on the train set and evaluate the accuracy score on the train set the accuracy equals 89.9% and on the test set accuracy equals 88.5%.

When I got this high accuracy, I tried to evaluate the performance of this binary classification using other performance measures which as the confusion matrix, to compute the confusion matrix, I needed a set of predictions that I prepared last then I used the confusion_matrix() function. just pass the target classes and the predicted classes [3].

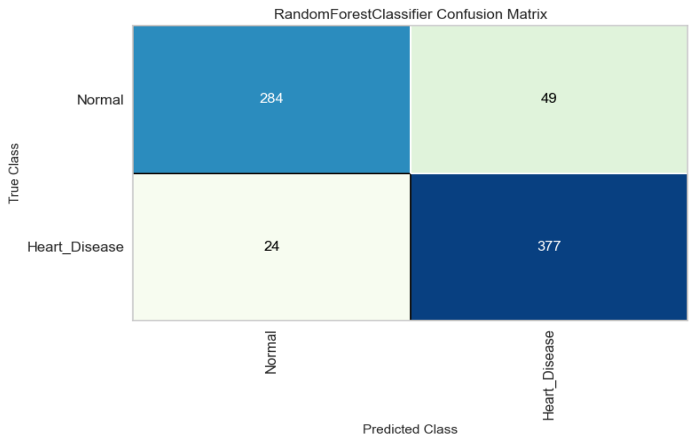

Fig. 5. Confusion Matrix

Fig 5 shows the output when I used the confusion_matrix function for predicting the train set, this confusion matrix gives a lot of information, but sometimes you may prefer a more concise metric. from this matrix I can compute the precision and recall, the recall is also called sensitivity or true positive rate this is the ratio of positive instances that are correctly detected by the classifier as equation (1) shows.

In this HeartDisease data set, I train the classifier to detect if the patient has heart failure or not, therefore we must get a higher recall which means we need to minimize false-negative numbers because we do not want to predict that someone does not have heart failure and they turn out they do. So, to compute the recall we can use equation number (1). In this equation, the FN refers to the number of false negatives and TP refers to the number of true positives.

$$Recall = \frac{TP}{TP+FN} \qquad (1)$$

5) Votting Classification

When I tried the previous model, I achieve approximately similar accuracy. Therefore, I tried to use the hard votting classifier that takes multiple classifiers, aggregating each classifier's predictions and predicting the class that gets the most votes. So, the voting classifier often achieves higher accuracy than the best classifier in the ensemble.

Therefore, I create the votting classification composed of three diverse classifiers, the Decision Tree classifier, random forest classifier, and support vector classifier. Then I noticed that the voting classifier slightly outperforms all the individual classifiers, it achieved an accuracy score equal to 89% when I evaluate the test set.

E. Give The Result

After I Tried multiple classification techniques the classification accuracies reported in the random forest and vottind classification are significantly higher than in previous work. Fig 6 compares the classification result of all classical techniques when I evaluated using the test set.

Depending on the result shown in Fig 6 the best accuracy that I was able to get is 89% when I used the votting classification and 88.5% when I used the random forest classifications.

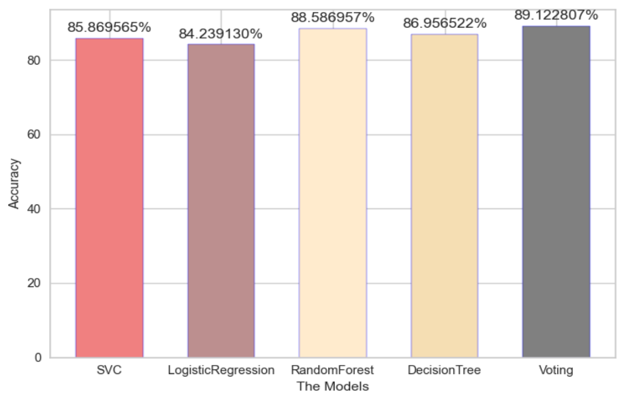

Fig. 6. The Final Result

IV. Conclusion

In this work, I used a machine learning approach to classify Heart Failure. After I read the data set and got acquainted with it, I split the data set into two data set train set which have 80% of the original data, and the test set which has 20% of the original data set, then I prepared the train set by defined a pipelined which handling the numerical feature and the text feature separately. on the numerical feature, I used the stander scalar class to scale the data and on the text feature, I used the one encoder class to convert this feature to the number feature even it is easy to deal with.

After I prepared the train set, I defined multiple models and I tried to use hyperparameter to get higher accuracy. So, I used GridSearch class to find the best hyperparameter that can take higher accuracy.

Finally, I plotted the result as Fig 6 shows and the highest accuracy that I was able to get is 89% when I used the votting classification and 88.5% when I used the random forest classifications. Also, if an additional external dataset with the same features from a different geographical region had been available, I would have used it as a validation cohort to verify our findings.

Regarding future developments, I think if I used the neural network with keras, I will get a high accuracy than the accuracy that I got when using the classical techniques.

V. References

[1] "Heart Failure Prediction Dataset," [Online]. Available: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction. [Accessed 22 5 2022].

[2] "Heart Failure Prediction," [Online]. Available: https://www.kaggle.com/code/tonyerhire/heart-failure-prediction-svm-89percent-accuracy. [Accessed 25 5 2022].

[3] "Heart Failure Prediction, RandomForest," [Online]. Available: https://www.kaggle.com/code/yaribol/heart-failure-prediction-randomforest-0-89.

VI. Appendix A. Source Code

This appendix shows the sklearn code in python which include all processing that I defined and used on the data set:

```
df = pd.read_csv("D:/ML-Project/dataset/heart.csv")

x = df.drop(['HeartDisease'], axis = 1)
y = df.loc[:,'HeartDisease'].values

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
print(corr_mat['HeartDisease']. sort_values(ascending=False))


#Full_Piplined
cat_attribs =["Sex", "ChestPainType", "RestingECG", "ExerciseAngina", "ST_Slope" ]
num_attribs = [ "Age", "RestingBP", "Cholesterol", "FastingBS", "MaxHR", "Oldpeak" ]
full_pipline = ColumnTransformer([("num", StandardScaler(), num_attribs), ("cat", OrdinalEncoder(),
cat_attribs)])
data_prepared = full_pipline.fit_transform(x_train)
x_test_pre= full_pipline.transform(x_test)

#Traning Models
svc = SVC()
svc.fit(data_prepared,y_train)
predic = svc.predict(x_test_pre)
accu = accuracy_score(y_test,predic)
print("SVC : ",accu)

svc_model = SVC()
svc_model.get_params()
param_grid_svm={'C': [0.1,0.5,1,2,10],'gamma':['scale','auto'],'kernel': ['linear','rbf','poly','sigmoid']}
svc_grid_model= GridSearchCV(svc_model,param_grid_svm,cv=5)
svc_grid_model.fit(data_prepared,y_train)
print(svc_grid_model.best_params_)
svc_pred= svc_grid_model.predict(x_test_pre)
accr = accuracy_score(y_test,svc_pred)
print("SVC with hyperparameter : ",accr)

lr = LogisticRegression()
lr.fit(data_prepared,y_train)
predi = lr.predict(x_test_pre)
accll = accuracy_score(y_test,predi)
print("accuraccy Logis :", accll)

r_forest = RandomForestClassifier(criterion = 'gini',n_estimators=100,max_depth=5,random_state=33)
r_forest.fit(data_prepared,y_train)
predicted = r_forest.predict(x_test_pre)
acc = accuracy_score(y_test,predicted)
print("accuraccyRA :",acc )

DecisionTreeClassifierModel = DecisionTreeClassifier(criterion='gini',max_depth=4,random_state=33)
DecisionTreeClassifierModel.fit(data_prepared, y_train)
pppp= DecisionTreeClassifierModel.score(x_test_pre, y_test)
print("Dec_tree",pppp)

#The Confusion Matrix
classes = ['Normal', 'Heart_Disease']
r_forest_cm = ConfusionMatrix(r_forest, classes=classes, cmap='GnBu')

r_forest_cm.fit(data_prepared, y_train)
r_forest_cm.score(data_prepared, y_train)
r_forest_cm.show()

# Votting Classifier
voting = VotingClassifier(estimators=[('lr',DecisionTreeClassifierModel), ('rf', r_forest),('svs',
svc_model)],voting='hard')
voting.fit(data_prepared,y_train)
vott = cross_val_score(estimator=voting, X=x_test_pre, y=y_test, cv =10)

poiuh = voting.predict(data_prepared)
acc3 = accuracy_score(y_train,poiuh)
print("trgfy  ",acc3)
vott_mean= vott.mean()
print(f"Accuracy of Voting Classifier : {vott.mean()}\n")
```

```
#The Final Result
x_Feature=["SVC", "LogisticRegression", "RandomForest", "DecisionTree", "Voting"]
y_Feature=[accr*100,accll*100,acc*100,pppp*100,vott_mean*100]
labels=[accr,accll,acc,pppp,vott_mean]
c=["lightcoral","rosybrown","blanchedalmond","wheat","gray"]
ax=plt.bar(x_Feature,y_Feature,width=0.6,color=c)
plt.xlabel("The Models")
plt.ylabel("Accuracy")

for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    plt.annotate(f'{height/100:%}', (x + width/2, y + height*1.02), ha='center')

plt.show()
```

Fig. 7. The source codes