Ensemble approach for heart disease

Name – Sonu Kumar

Registration Number – 11905982

Section – KM018

GitHub Link - https://github.com/sonuacc/Heart_Disease_Prediction

ABSTRACT

The objective of this project is to determine whether or not a patient has cardiac disease. The dataset is accessible on Kaggle and includes information about several patients. The collection has 303 records and 13 features. In this project, I have used a variety of methods to diagnose heart illness, including Logistic Regression, Search Vector Machine, and Decision Tree Classifier, and then I combined these three machine learning techniques to create an ensemble model. And out of all these models, Logistic Regression and Ensembled Model provided the highest accuracy of over 90%.

INTRODUCTION

According to the World Health Organization, 12 million people die each year because of heart disease. From the last several years, the burden of cardiovascular disease has been quickly increasing all over the world. Many studies have been carried out to define the most important factors in heart disease and to precisely forecast the overall risk. Heart disease is also referred to as a "silent killer" because it causes death without causing noticeable symptoms. Early detection of cardiac disease is critical for implementing lifestyle modifications in high-risk people and, as a result, reducing consequences. This project tries to predict future heart illness by evaluating patient data and using machine-learning algorithms to classify whether they have heart disease or not.

DATASET

The dataset is available on the Kaggle. It provides patient information which includes 303 records and 14 attributes which are : age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope,ca, thal & target. The dataset is in csv format.

Dataset:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

Attributes Information

Attribute Name	Information	Values
age	The person's age in years.	Continuous Value.
sex	The person's sex.	1 = male, 0 = female.
cp	The chest pain experienced.	Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic.
trestbps	The person's resting blood pressure.	Continuous Value in mm/Hg
chol	The person's cholesterol measurement.	Continuous Value in mg/dl.
fbs	The person's fasting blood sugar.	> 120 mg/dl ? 1 = true, 0 = false
restecg	Resting electrocardiographic measurement.	0 = normal, 1 = having ST-T wave abnormality, 2 = left ventricular hypertrophy.
thalach	The person's maximum heart rate achieved.	Continuous Value.
exang	Exercise induced angina.	0 – No, 1 - Yes
oldpeak	ST depression induced by exercise relative to rest.	Continuous Value.
slope	the slope of the peak exercise ST segment.	Value 1: upsloping, Value 2: flat, Value 3: down sloping.
ca	The number of major vessels.	0-3.
thal	A blood disorder called thalassemia.	3 = normal, 6 = fixed defect, 7 = reversable defect.
target	Heart Disease	0 – No, 1 - Yes

DATA PRE-PROCESSING

Dataset contains 13 features and 303 records with having 0 null records.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

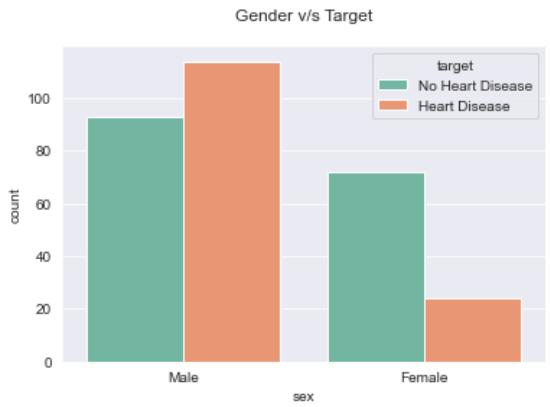
As we can see there are no null values in any of the records.

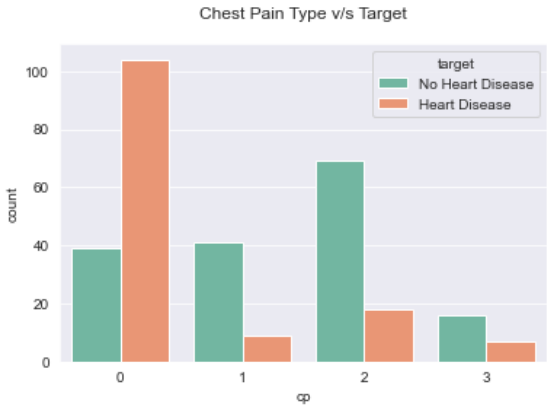
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

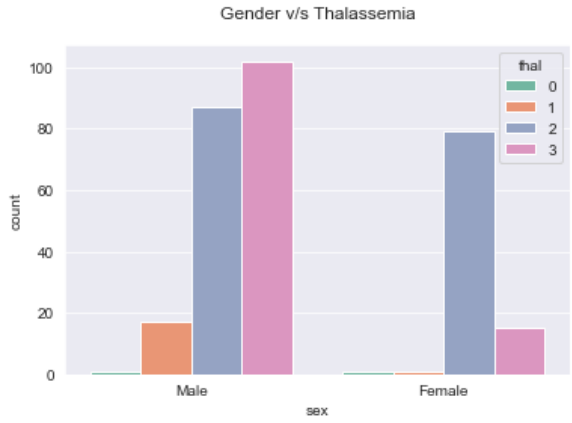
```
dataset.isnull().sum()
```

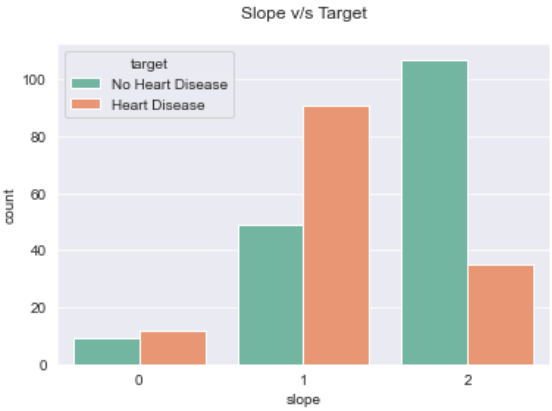
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Data Visualisation

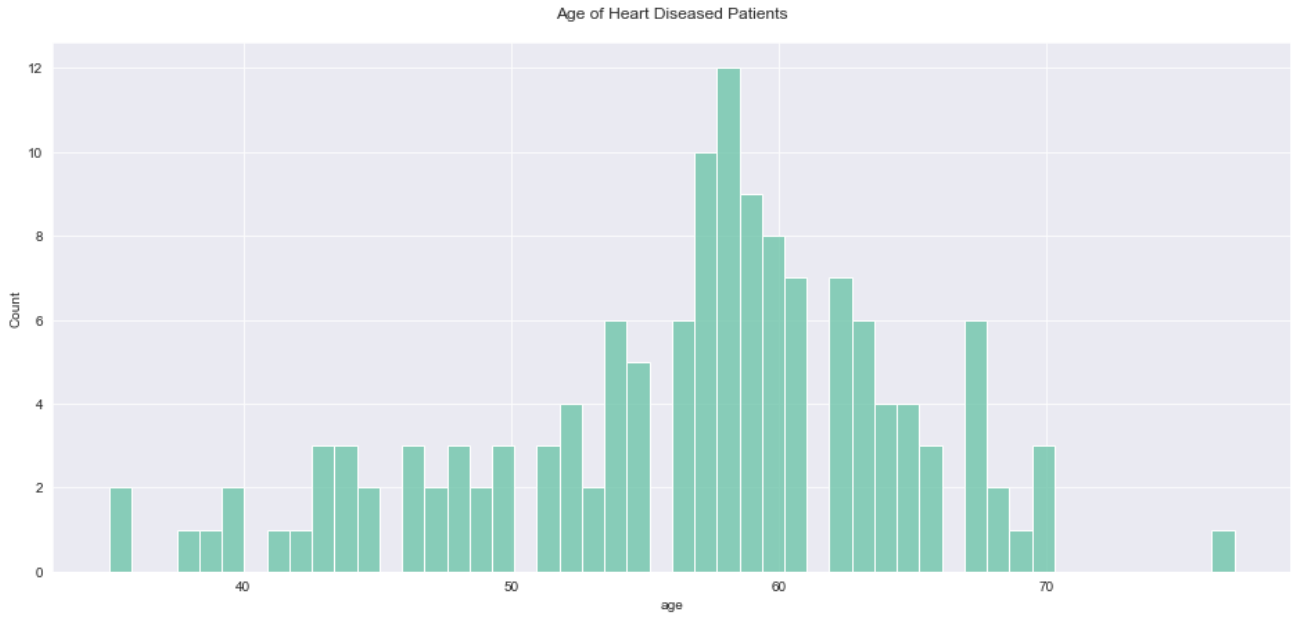








Age Of Heart Disease Patients

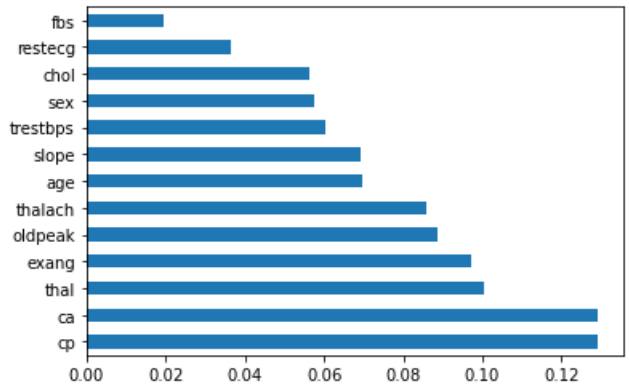


FEATURE SELECTION & EXTRACTION

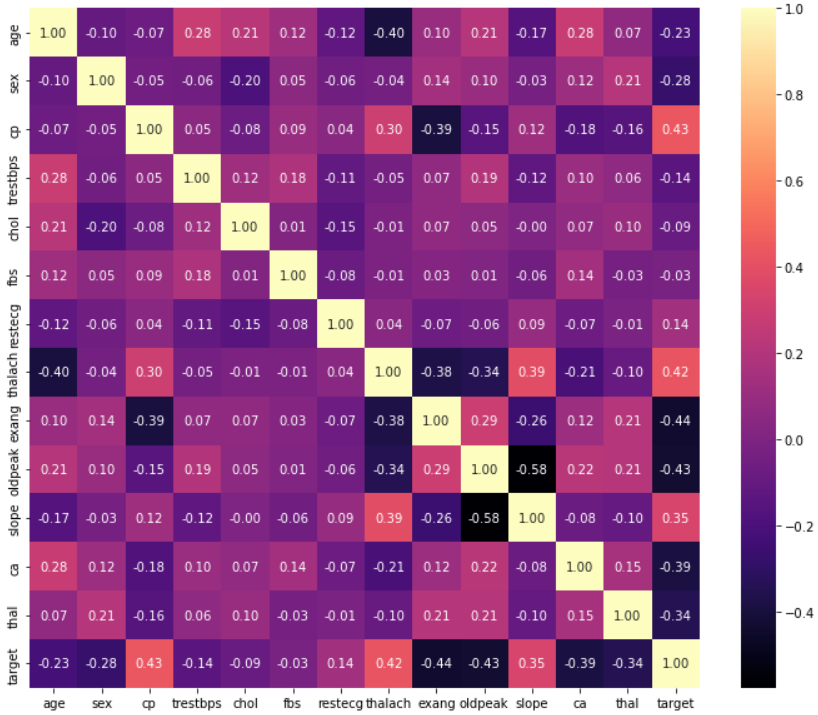
Feature Score

	Specs	Score
7	thalach	188.320472
9	oldpeak	72.644253
11	ca	66.440765
2	cp	62.598098
8	exang	38.914377
4	chol	23.936394
0	age	23.286624
3	trestbps	14.823925
10	slope	9.804095
1	sex	7.576835
12	thal	5.791853
6	restecg	2.978271

Important Features



Correlation Matrix Heatmap



Converting Categorial Value To One-Hot Encoding

```
dataset = pd.get_dummies(dataset, columns = ["cp","restecg","slope","ca","thal"])
```

Numerical Columns V/S Categorial Columns

```
numerical_col = ["age","trestbps","chol","thalach","oldpeak"]
cat_col = list(set(dataset.columns)-set(numerical_col)-{"target"})
```

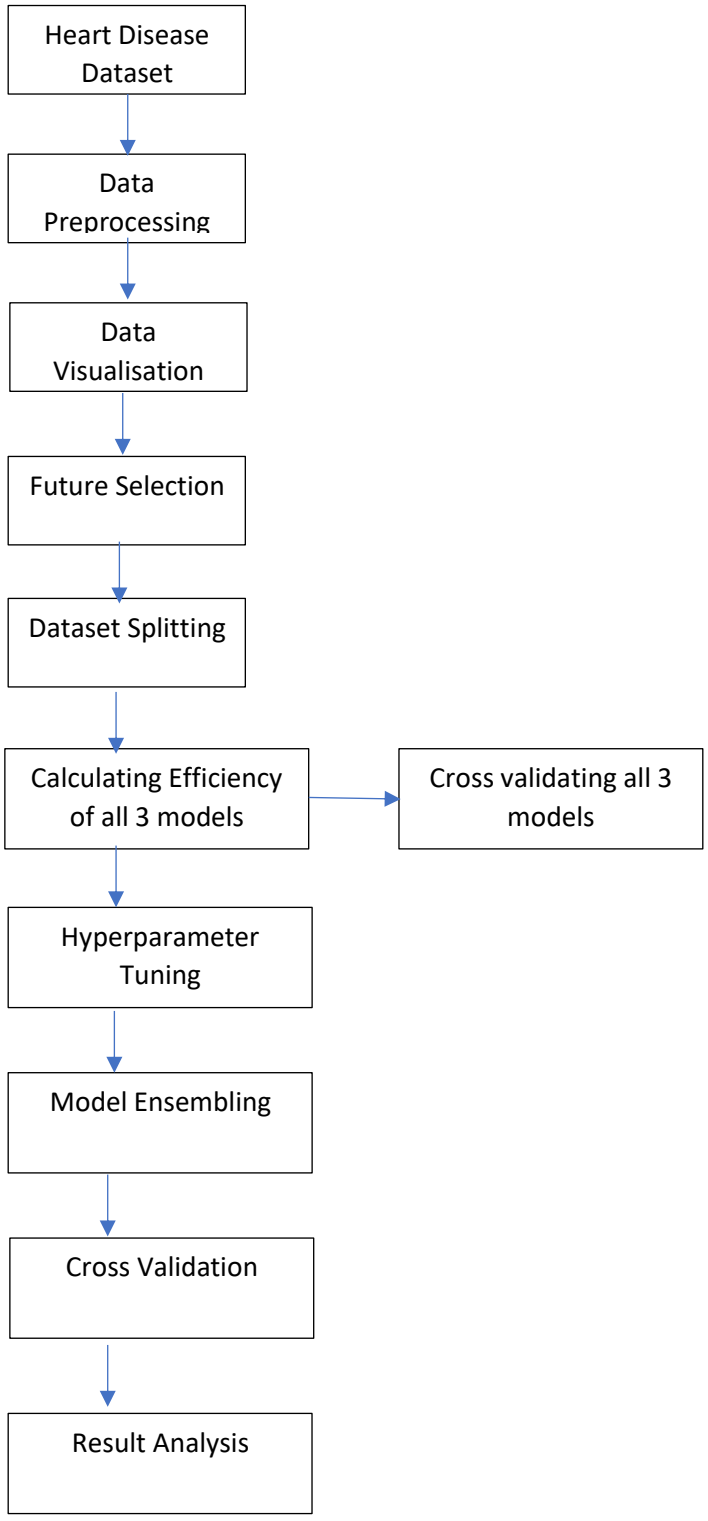
Training & Test Dataset Splitting

Finally, this resulting data split into 80% train and 20% test data, which was further passed to the Logistic Regression, SVM, DTC model to fit, predict and score the model.

PROPOSED METHODOLOGY

In this approach I have used Logistic Regression, Support Vector Machine and Decision Tree Classifier to predict the results. 80 % of the data I have used for training and 20% of the data I will use for testing. I will calculate the accuracy of all the models and their respective cross validation will be also considered followed by Hyperparameter tuning. After that I will use all this models to make an ensembled model using stacking followed by Cross Validation.

Flow chart of my Model will look like –



ALGORITHM USED & RESULT

The main purpose for designing this model is to predict whether the patient has heart disease or not. I have used Logistic regression, Support Vector Machine & Decision Tree Classifier as machine-learning algorithm to train my model.

1. Logistic Regression (1st Model)

Logistic regression is a classification technique used in machine learning. It's a supervised learning algorithm which is used to predict a binary outcome such as YES or NO,1 & 0. The results are based on prior observation of dataset. This types if model predicts a dependent data variable by analysing the relationships between one or more existing independent variables.

```
# Logistic Regression
lr = LogisticRegression(solver='lbfgs',penalty='l2')
lr.fit(x_train,y_train)
test_pred = lr.predict(x_test)
print("Mean Square Error:", mean_squared_error(y_test,test_pred)*100,"%")
print("Accuracy:",accuracy_score(y_test,test_pred)*100,"%")
```

```
Mean Square Error: 9.836065573770492 %
Accuracy: 90.1639344262295 %
```

Using Logistic regressing with parameters (solver = 'lbfgs' & penalty = 'l2') we achieved MSE of 8.83% and accuracy of 90.1639 %.

Cross Validation (Using K-Fold)

It is a model validation technique for checking how the results pf a statistical analysis will generalize to an independent dataset. K-Fold cross validation was used for Cross Validation and the result which we got is following

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
k = 5
kf=KFold(n_splits=k,shuffle=True)
result = cross_val_score (lr,x_train,y_train,cv=kf)
print("Accuracy: ",np.average(result)*100,"%")
```

```
Accuracy:  83.86054421768708 %
```

Accuracy achieved in cross validation is 83.8605 %.

2. Support Vector Machine (2nd Model)

Support Vector Machine is used for classification, regression and outliers detection. It's a supervised machine learning. Its goal is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in future. This boundary is known as hyperplane. It chooses the extreme points or vectors that help in creating the hyperplane. These extreme cases are known as Support Vector.

```
# Support Vector Machine
sc_clf = SVC()
sc_clf.fit(x_train,y_train)
svm_pred = sc_clf.predict(x_test)
print("Mean Square Error:", mean_squared_error(y_test,svm_pred)*100,"%")
print("Accuracy:",accuracy_score(y_test,svm_pred)*100,"%")
```

```
Mean Square Error: 11.475409836065573 %
Accuracy: 88.52459016393442 %
```

Using Support Vector Machine, we achieved MSE of 11.4754% and accuracy of 88.5245 %.

Cross Validation (Using K-Fold)

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
k = 5
kf=KFold(n_splits=k,shuffle=True)
result = cross_val_score (sc_clf,x_train,y_train,cv=kf)
print("Accuracy: ",np.average(result)*100,"%")
```

```
Accuracy:  82.23639455782313 %
```

Accuracy achieved in cross validation is 82.2363%.

3. Decision Tree Classifier (3rd Model)

Decision Tree Classifier is a supervised machine learning algorithm used for both classification and regression problem. The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

```
# Decision Tree Classifier
dc_clf = DecisionTreeClassifier()
dc_clf.fit(x_train,y_train)
dlf_pred = dc_clf.predict(x_test)
print("Mean Square Error:", mean_squared_error(y_test,dlf_pred)*100,"%")
print("Accuracy:",accuracy_score(y_test,dlf_pred)*100,"%")
```

```
Mean Square Error: 18.0327868852459 %
Accuracy: 81.9672131147541 %
```

Using Decision Tree Classifier, we achieved MSE of 18.0327% and accuracy of 81.9672 %.

Cross Validation (Using K-Fold)

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
k = 5
kf=KFold(n_splits=k,shuffle=True)
result = cross_val_score (dc_clf,x_train,y_train,cv=kf)
print("Accuracy: ",np.average(result)*100,"%")
```

```
Accuracy:  73.9625850340136 %
```

Accuracy achieved in cross validation is 73.9625 %.

RESULTs

Model Name	MSE	Accuracy	Cross Validation
Logistic Regression	9.8360%	90.1639%	83.8605%
Support Vector Machine	11.4754%	88.5245%	82.2363%
Decision Tree Classifier	18.0327%	81.9672%	73.9625%

We can clearly see Efficiency (Logistic Regression) > Efficiency (Support Vector Machine) > Efficiency (Decision Tree Classifier).

HYPERPARAMETER TUNING

A hyperparameter is a parameter whose value is used to control the learning process. Hyperparameter tuning is a process of choosing a set of optimal hyperparameters for a learning algorithm. It can be used to see which parameter is giving the ideal performance so that those parameters can be used to increase the efficiency of a machine learning model.

Hyperparameter tuning applied on LR, SVM, DTC

- Parameters for Logistic Regression, Support Vector Machine & Decision Tree Classifier.

```
parameters = {
    'Logistic Regression' : {
        'model': LogisticRegression(solver='lbfgs'),
        'params': {
            'C': [1,5,10],
        }
    },
    'Support Vector Machine': {
        'model': SVC(gamma='auto'),
        'params' : {
            'C':[0.1,1,100,1000],
            'kernel':['rbf','poly','sigmoid','linear'],'degree':[1,2,3,4,5,6]
        }
    },
    'Decision Tree Classifier': {
        'model': DecisionTreeClassifier(),
        'params' : {
            'max_depth': [2, 3, 5, 10, 20],
            'min_samples_leaf': [5, 10, 20, 50, 100],
            'criterion': ["gini", "entropy"]
        }
    },

}
```

- Checking accuracy score after tuning and storing the value in accuracy_score.

```
accuracy_scores = []

for model_name,name in parameters.items():
    clf =  GridSearchCV(name['model'], name['params'], cv=5, return_train_score=False)
    clf.fit(x_train,y_train)
    accuracy_scores.append({
        'Model': model_name,
        'Best Accuracy': clf.best_score_,
        'Best Parameter': clf.best_params_
    })

acc_data = pd.DataFrame(accuracy_scores,columns=['Model','Best Accuracy','Best Parameter'])
acc_data
```

- Outcomes after hyperparameter tuning

	Model	Best Accuracy	Best Parameter
0	Logistic Regression	0.842857	{'C': 5}
1	Support Vector Machine	0.834524	{'C': 100, 'degree': 1, 'kernel': 'linear'}
2	Decision Tree Classifier	0.789456	{'criterion': 'gini', 'max_depth': 20, 'min_sa...

We can clearly see Logistic Regression is performing the best with accuracy of 84.28% followed by Support Vector Machine with accuracy of 83.45% and Decision Tree Classifier with accuracy of 78.94%.

ENSEMBLED MODEL

Ensemble methods use multiple number of algorithms in the background to obtain a better performance than any of the constituent algorithm alone. The models which I have used to develop this ensemble model are Support Vector Machine, Logistic Regression, Decision Tree Classifier & the type of ensemble is Stacking.

```
In [28]: from sklearn.ensemble import VotingClassifier
         Model1 = LogisticRegression()
         Model2 = DecisionTreeClassifier()
         Model3 = SVC()
         Model = VotingClassifier([('LR',Model1),('DTR',Model2),('SVC',Model3)])
         Model.fit(x_train,y_train)
         ans = Model.score(x_test,y_test)
         Model_pred = Model.predict(x_test)
         print("Mean Square Error:", mean_squared_error(y_test,Model_pred)*100,"%")
         print("Accuracy:",accuracy_score(y_test,Model_pred)*100,"%")

         Mean Square Error: 9.836065573770492 %
         Accuracy: 90.1639344262295 %
```

We can clearly see Using Ensembled Model, we achieved MSE of 9.8360% and accuracy of 90.1639 % which is better than Support Vector Machine & Decision Tree Classifier alone, but its accuracy is equal to the accuracy of Logistic Regression.

Cross Validation (Using K-Fold)

```
In [29]: from sklearn.model_selection import KFold
         from sklearn.model_selection import cross_val_score

         k = 5
         kf=KFold(n_splits=k,shuffle=True)
         result = cross_val_score (Model,x_train,y_train,cv=kf)
         print("Accuracy: ",np.average(result)*100,"%")

         Accuracy:  81.82823129251699 %
```

Accuracy achieved in cross validation is 81.8282 %.

RESULTs

Model Name	MSE	Accuracy	Cross Validation
Logistic Regression	9.8360%	90.1639%	83.8605%
Support Vector Machine	11.4754%	88.5245%	82.2363%
Decision Tree Classifier	18.0327%	81.9672%	73.9625%
Ensembled Model	9.8360%	90.1639%	81.8282%

We can clearly see Efficiency (Logistic Regression) = Efficiency (Ensembled Model) > Efficiency (Support Vector Machine) > Efficiency (Decision Tree Classifier).

Conclusion

Early detection of cardiovascular disease can help in making lifestyle adjustments in high-risk individuals, reducing consequences and perhaps saving lives, which might be a major breakthrough in medicine. This project effectively predicted heart disease with 90 percent accuracy by resolving feature selection behind the models. Logistic regression, Support Vector Machine & Decision Tree Classifier was employed as the model. Later we tried to make an ensembled model using all three models which gave us the accuracy of 90 percent same as Logistic Regression. In order to improve it, we may train on models and anticipate the sorts of cardiovascular problems that users would face, as well as employ more advanced models.

Reference

1. Soni, Jyoti, et al. "Predictive data mining for medical diagnosis: An overview of heart disease prediction." International Journal of Computer Applications 17.8 (2011).
2. Dangare, Chaitrali S., and Sulabha S. Apte. "Improved study of heart disease prediction system using data mining classification techniques." International Journal of Computer Applications 47.10 (2012).
3. Uyar, Kaan, and Ahmet İlhan. "Diagnosis of heart disease using genetic algorithm based trained recurrent fuzzy neural networks." Procedia computer science 120 (2017).
4. Kim, Jae Kwon, and Sanggil Kang. "Neural network-based coronary heart disease risk prediction using feature correlation analysis." Journal of healthcare engineering 2017 (2017).
5. Baccouche, Asma, et al. "Ensemble Deep Learning Models for Heart Disease Classification: A Case Study from Mexico." Information 11.4 (2020).
6. https://archive.ics.uci.edu/ml/datasets/Heart+Disease
7. https://www.kaggle.com/ronitf/heart-disease-uci
8. https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf
9. https://nthu-datalab.github.io/ml/labs/03_Decision-Trees_RandomForest/03_Decision-Tree_Random-Forest.html
10. https://www.kaggle.com/jprakashds/confusion-matrix-in-python-binaryclass.
11. scikit-learn, pandas and matplotlib.