MACHINE LEARNING USING PYTHON

TOPIC: PREDICTING HEART DISEASE

PROJECT REPORT

ABSTRACT

In this project, we were asked to experiment with a real world dataset and to explore how

Machine learning algorithms can be used to find patterns in data. We were asked to create ML models to predict the 'target' class and plot the confusion matrix for all classifier models and also to calculate the accuracy score for each. We were also instructed to build the following

Models:

Support Vector Machines (SVM)

K-Nearest Neighbor classifier (K-NN)

SVM with PCA

K-NN with PCA

Classification plot for all the four models, bar plot for count of male and female having heart

Disease, scatter plot between age and maximum heart were also asked to be done.

After performing the required tasks on the dataset, herein lies our final report.

TABLE OF CONTENTS

1.0 INTRODUCTION

Machine learning (ML) is a category of an algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available.

Machine learning is used for the following purposes:

- Prediction — Machine learning can also be used in the prediction systems. Considering the loan example, to compute the probability of a fault, the system will need to classify the available data in groups.
- Image recognition — Machine learning can be used for face detection in an image as well. There is a separate category for each person in a database of several people.
- Speech Recognition — It is the translation of spoken words into the text. It is used in voice searches and more. Voice user interfaces include voice dialing, call routing, and appliance control. It can also be used a simple data entry and the preparation of structured documents.
- Medical diagnoses — ML is trained to recognize cancerous tissues.
- Financial industry and trading — companies use ML in fraud investigations and credit checks.

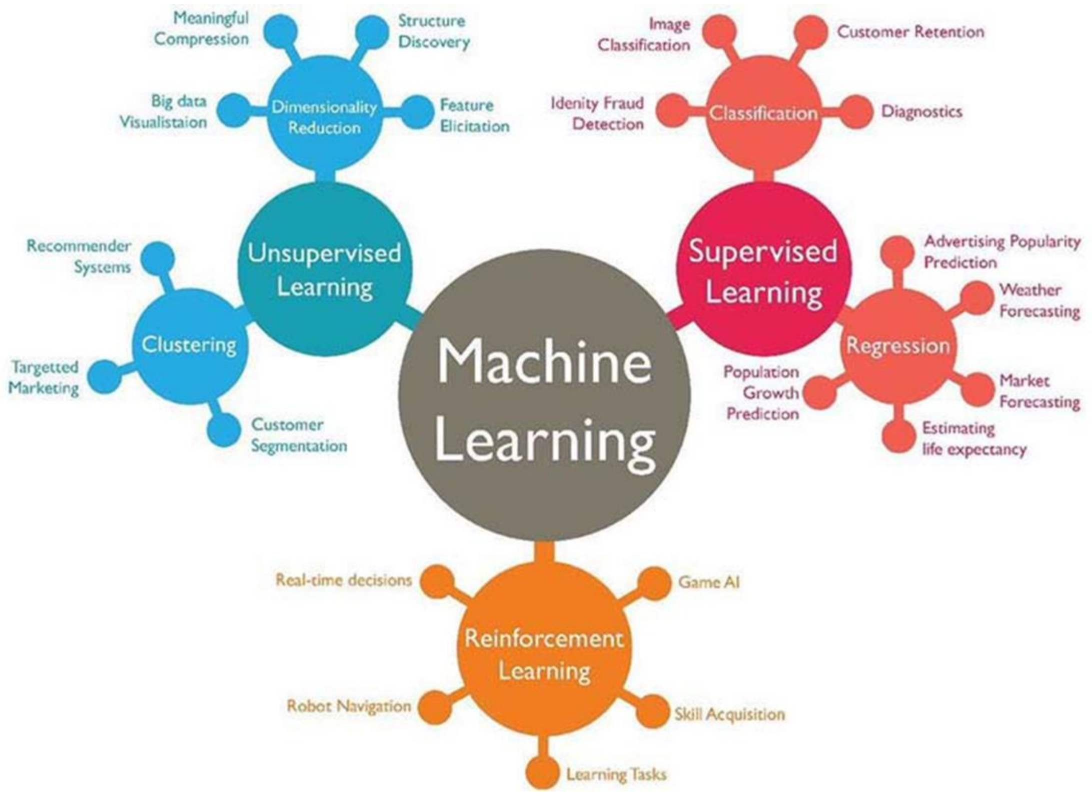
Types of Machine Learning?

Machine learning can be classified into 3 types of algorithms.

1.Supervised learning

2.Unsupervised learning

3.Reinforcement learning



Supervised Learning

In Supervised learning, an AI system is presented with data which is labeled, which means that each data tagged with the correct label.The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

Types of Supervised learning

- **Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Unsupervised Learning

In unsupervised learning, an AI system is presented with unlabeled, uncategorized data and the system's algorithms act on the data without prior training. The output is dependent upon the coded algorithms. Subjecting a system to unsupervised learning is one way of testing AI.

Types of Unsupervised learning

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Reinforcement Learning

A reinforcement learning algorithm, or agent, learns by interacting with its environment. The agent receives rewards by performing correctly and penalties for performing incorrectly. The agent learns without intervention from a human by maximizing its reward and minimizing its penalty. It is a type of dynamic programming that trains algorithms using a system of reward and punishment.

2.1 DESCRIPTION OF THE PROJECT

This project gives an idea about which parameters hold top priority in predicting heart disease.

The various parameters which are taken into account are as follows:

The parameters included are :

1. Age
2. Sex (0 – female and 1 – male)
3. cp: chest pain type
   -- Value 1: typical angina
   -- Value 2: atypical angina
   -- Value 3: non-anginal pain
   -- Value 4: asymptomatic
4. trestbps: resting blood pressure (in mm Hg on admission to the hospital)
5. 12 chol: serum cholesterol in mg/dl
6. fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. restecg: resting electrocardiographic results
   -- Value 0: normal
   -- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
   -- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
8. thalach: maximum heart rate achieved
9. exang: exercise induced angina (1 = yes; 0 = no)
10. oldpeak = ST depression induced by exercise relative to rest
11. slope: the slope of the peak exercise ST segment
    -- Value 1: upsloping
    -- Value 2: flat
    -- Value 3: downsloping
12. ca: number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
14. num: diagnosis of heart disease (angiographic disease status)
    -- Value 0: < 50% diameter narrowing
    -- Value 1: > 50% diameter narrowing (**Predicted value)**

In this project four ML models are built:SVM, KNN, SVM with PCA, KNN with PCA. Confusion matrix is plotted for all classifier models accuracy score is calculated for each. Also the classification plot is plotted for all four models,bar plot is plotted for count of male and female having heart disease and scatter plot between age and maximum heart rate is also plotted.

2.2 DESCRIPTION OF THE LIBRARIES

NumPy

NumPy is a python library used for working with arrays.It also has functions for working in domain of linear algebra, fourier transform, and matrices.NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.NumPy stands for Numerical Python. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

Install and import:

- Installing it using either of the following commands:
  $conda install numpy or $pip install numpy
- To import Numpy we use the following command:
  import numpy as np

Scikit-learn

Scikit-learn (formerly **scikit-learn** and also known as **sklearn**) is a free software machine learning library for the python programming language. It features various classification,regression and clustering algorithms including support vector machines,random forest regressor, gradient-boostiong, k-means and DBSCAN and is designed to interoperate with the Python numerical and scientific libraries Numpy and Scipy.

Install and import:

- Installing it using either of the following commands:
  $conda install scikit-learn or $pip install -U scikit-learn

- To import Numpy we use the following command:
  import sklearn

- **SciPy**
  SciPy is a free and open source Python library library used for scientific computing and technical computing. SciPy contains modules for optimization,linear algebra,integration, interpolation,special functions, FFT,signal and image processingprocessing,ODE solvers and other tasks common in science and engineering. SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib,Pandas,SciPy and an expanding set of scientific computing libraries.

Install and import:

- Installing it using either of the following commands:
  $ conda install -c anaconda scipy or $pip install scipy
- To import Numpy we use the following command:
  import scipy

- **Pandas**

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Install and import:

- Installing it using either of the following commands:
  $conda install pandas or $pip install pandas
- To import Numpy we use the following command:
  import pandas as pd

- **Matplotlib**
  Matplotlib is a plotting library used for 2D or 3D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits. Matploitlib is a Python Library used for plotting, this python library provides and objected-oriented APIs for integrating plots into applications.

Install and import:

- Installing it using either of the following commands:
  $conda install matplotlib or $pip install matplotlib
- To import Numpy we use the following command:
  import matplotlib.pyplot as plt

3.0 IMPLEMENTATION

3.1 Hardware used

- Laptop/Desktop
- Processor : Intel®Core™ i3 and above
- RAM : 4GB

3.2 Software used

- Operating System : Windows 10
- Programming Language: Python
- IDE :Jupyter notebook

PYTHON

Python is an interpreted,high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach approach aim to help programmers write clear, logical code for small and large-scale projects. Python is a multi-paradigm programming language. Object-oriented programming and structured-programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming. It is used for:

- web development (server-side),
- software development,
- system scripting.

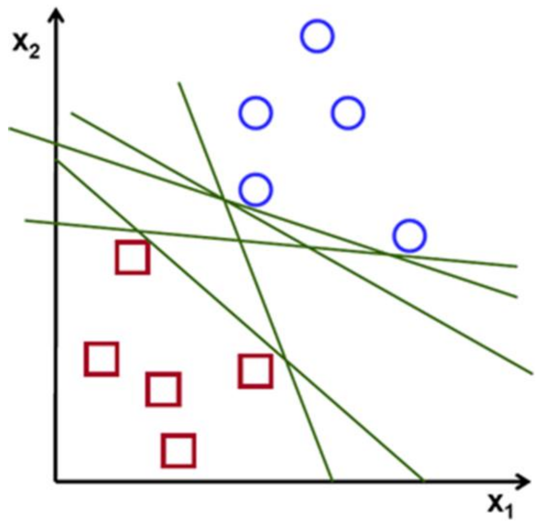
What can Python do?

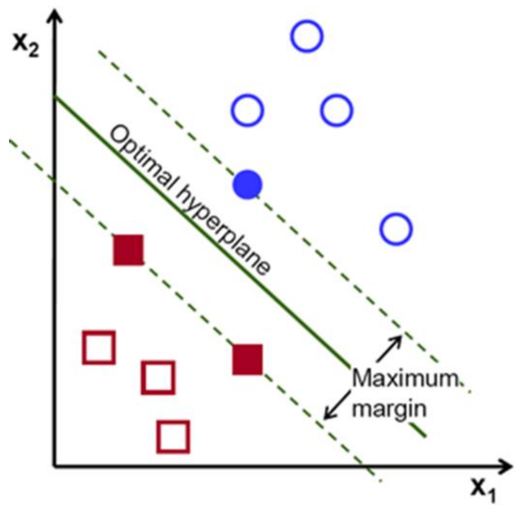
- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

4.1.1 SUPPORT VECTOR MACHINES(SVM)

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

So you’re working on a text classification problem. You’re refining your training data, and maybe you’ve even tried stuff out using Naive Bayes. But now you’re feeling confident in your dataset, and want to take it one step further. Enter Support Vector Machines (SVM): a fast and dependable classification algorithm that performs very well with a limited amount of data to analyze.



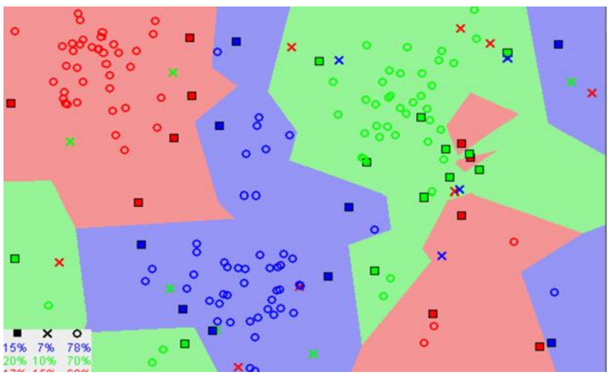


To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane.

4.1.2 K-NEAREST NEIGHBOUR(KNN)

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems**.**



Notice in the image above that most of the time, similar data points are close to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph.

The KNN Algorithm

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Advantages of KNN Algorithm:

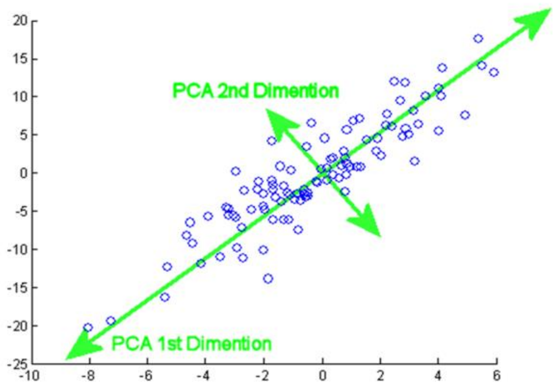
- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm**:**

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

4.1.3 PRINCIPAL COMPONENT ANALYSIS(PCA)

Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance. Finding such new variables, the principal components, reduces to solving an eigenvalue/eigenvector problem, and the new variables are defined by the dataset at hand, not *a priori*, hence making PCA an adaptive data analysis technique. It is adaptive in another sense too, since variants of the technique have been developed that are tailored to various different data types and structures.



Mathematics behind PCA

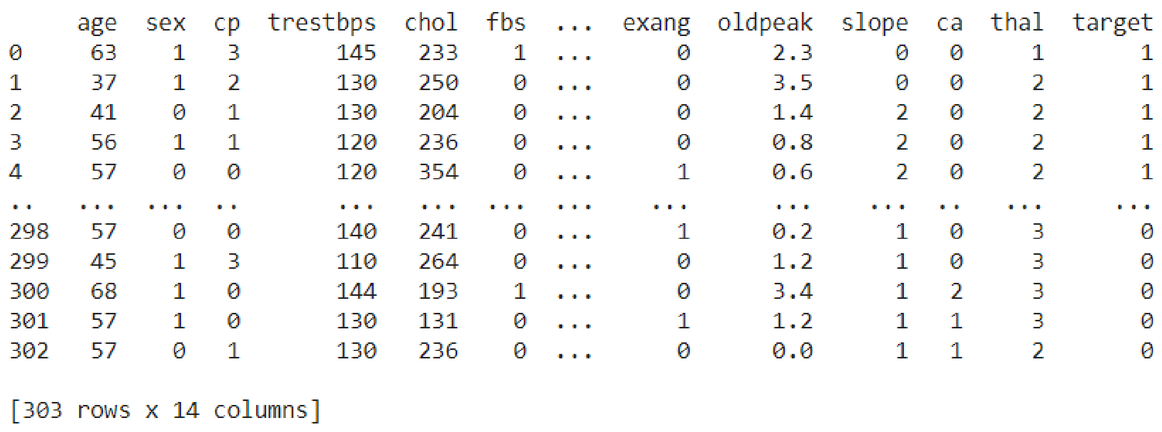
PCA can be thought of as an unsupervised learning problem. The whole process of obtaining principle components from a raw dataset can be simplified in six parts :

- Compute the *covariance matrix* of the whole dataset.
- Compute *eigenvectors* and the corresponding *eigenvalues*.
- Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a *d × k dimensional* matrix **W.**
- Use this *d × k eigenvector matrix* to transform the samples onto the new subspace.

CODES

5.1 SVM

#importing libraries and printing the dataset
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix
df=pd.read_csv('heart.csv')
print(df)
```



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

[303 rows x 14 columns]
```

#storing the whole dataset except last column in x and the last column in y
```
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

#scaling x column and storing it in x1
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x1=sc.fit_transform(x)
```

#training and testing x1 and y
```
x_tr,x_te,y_tr,y_te=train_test_split(x1,y,test_size=0.25,random_state=0)
```

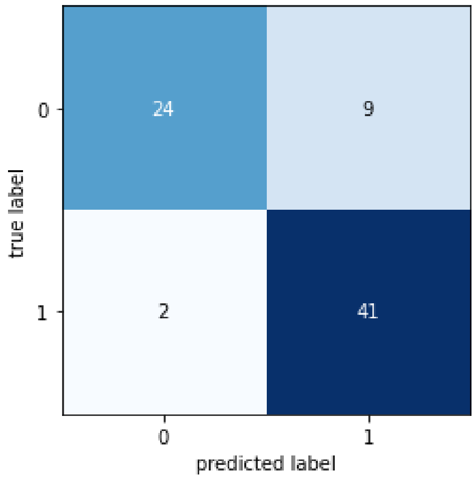
#fitting the model using SVM and predicting accuracy score and printing confusion matrix

```
classifier=SVC(kernel='linear',random_state=0)
classifier.fit(x_tr,y_tr)
y_pred=classifier.predict(x_te)
acc=accuracy_score(y_te,y_pred)
cm=confusion_matrix(y_te,y_pred)
print(acc)
print(cm)
```

```
0.8552631578947368
[[24  9]
 [ 2 41]]
```

#plotting confusion matrix

```
from mlxtend.plotting import plot_confusion_matrix,plot_decision_regions
plot_confusion_matrix(cm)
```

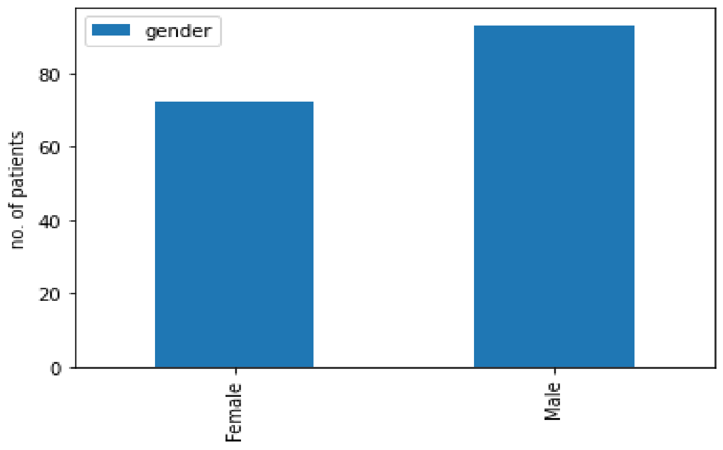


#counting the no. of males and females having heart disease and storing it in two lists

```
gender=[]
males=[]
females=[]
for i in range(0,302):
    if(x[i,1]==1):
        if(y[i]==1):
            males.append(x[i,1])
    elif(x[i,1]==0):
        if(y[i]==1):
            females.append(x[i,1])
plotdata = pd.DataFrame({"gender":[len(females),len(males)]},
    index=["Female","Male"])
```

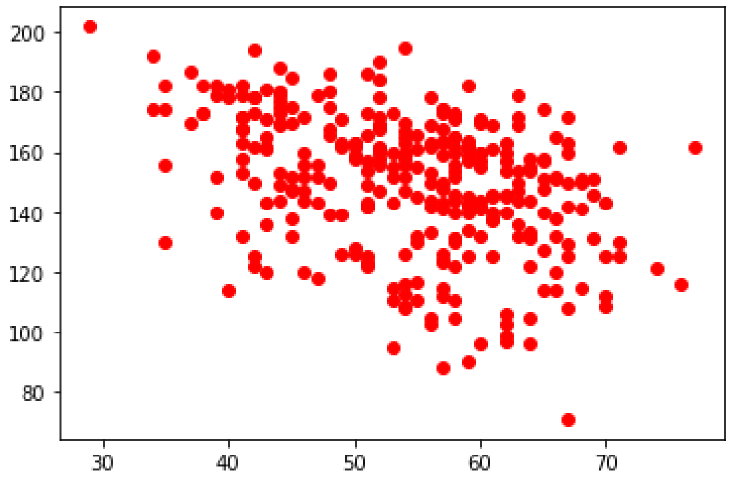
# Plot a bar chart

```
plotdata.plot(kind="bar")
plt.ylabel('no. of patients')
```



#storing the age column and the maximum heart rate column in x2 and y2 and performing scatterplot between them

```
x2=df.iloc[:,0]
y2=df.iloc[:,7]
plt.scatter(x2,y2,c='red')
```



5.1.1 SVM WITH PCA

#importing libraries and printing the dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix
df=pd.read_csv('heart.csv')
print(df)
```

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

[303 rows x 14 columns]
```

#storing the whole dataset except last column in x and the last column in y
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values

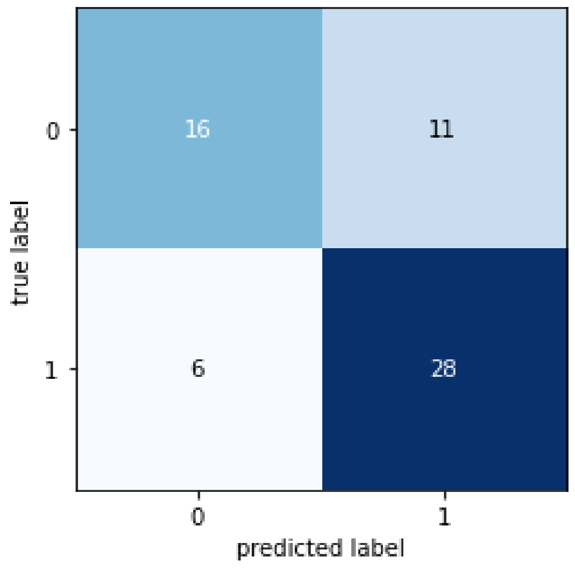
#scaling x column
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
#performing PCA on the dataset
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
x=pca.fit_transform(x)
PVE=pca.explained_variance_ratio_

#training and testing the model
from sklearn.model_selection import train_test_split
x_tr,x_te,y_tr,y_te=train_test_split(x,y,test_size=0.25,random_state=0)

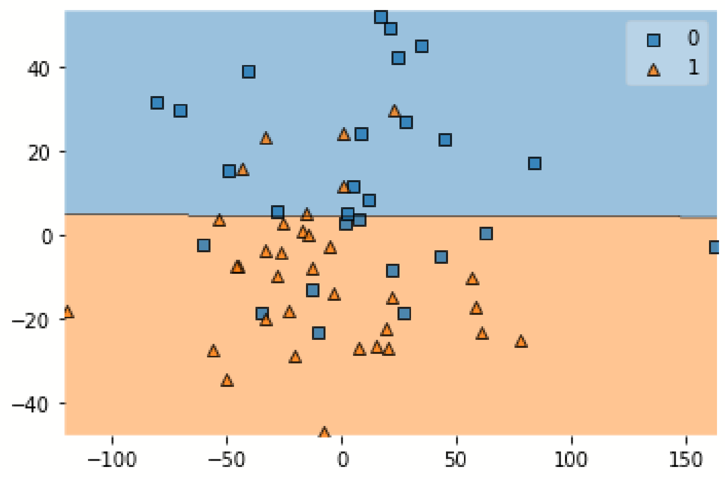
#fitting the model using SVM and predicting accuracy score and printing confusion matrix
from sklearn.svm import SVC
classifier=SVC(kernel='linear',random_state=0)
classifier.fit(x_tr,y_tr)
y_pred=classifier.predict(x_te)
acc=accuracy_score(y_te,y_pred)
cm=confusion_matrix(y_te,y_pred)
print(acc)
print(cm)

```
0.7213114754098361
[[16 11]
 [ 6 28]]
```

#plotting confusion matrix
from mlxtend.plotting import plot_confusion_matrix,plot_decision_regions
plot_confusion_matrix(cm)



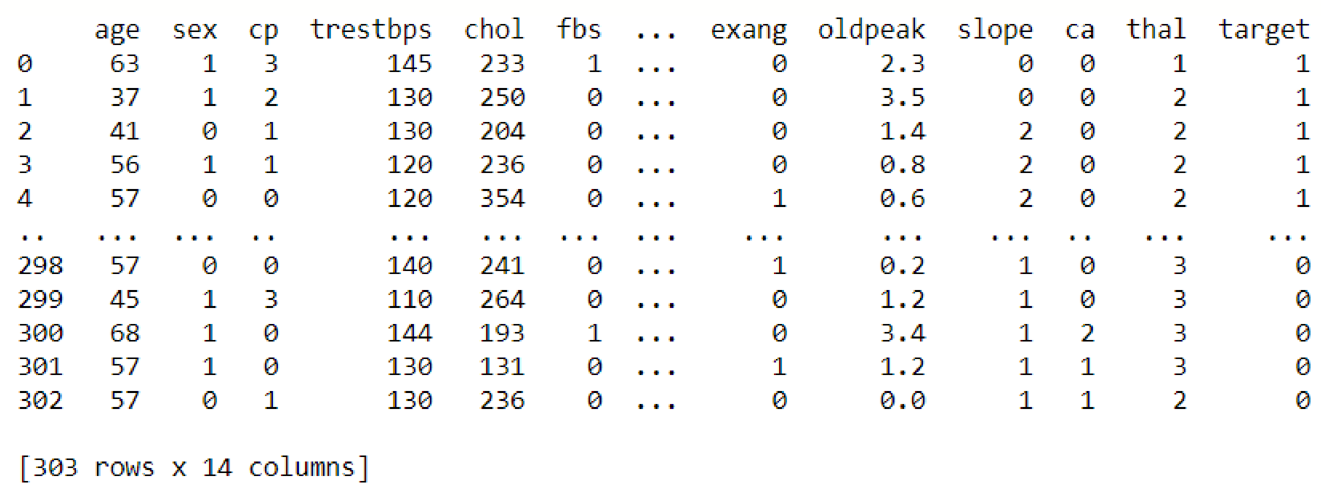
#plotting decision region
plot_decision_regions(X=x_te,y=y_te,clf=classifier)
plt.show()



5.2 KNN

#importing the libraries and dataset

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('heart.csv')
print(df)



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

[303 rows x 14 columns]
```

#extracting x and y column

x = df.iloc[:,:-1].values

y = df.iloc[:,-1].values

#scaling the x column

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x1 = sc.fit_transform(x)

#training and testing x1 and y

from sklearn.model_selection import train_test_split

x_tr,x_te,y_tr,y_te = train_test_split(x1,y,test_size = 0.2,random_state =0)

#applying KNN model on the training dataset

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)

classifier.fit(x_tr,y_tr)

y_pred = classifier.predict(x_te)

#predicting accuracy score and printing confusion matrix

from sklearn.metrics import accuracy_score,confusion_matrix

acc = accuracy_score(y_te,y_pred)

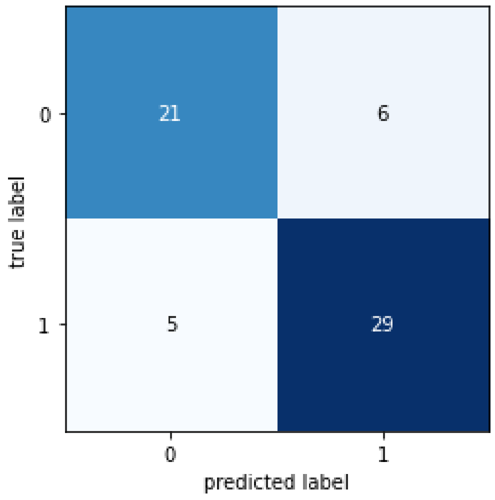
cm = confusion_matrix(y_te,y_pred)

print(acc)

print(cm)

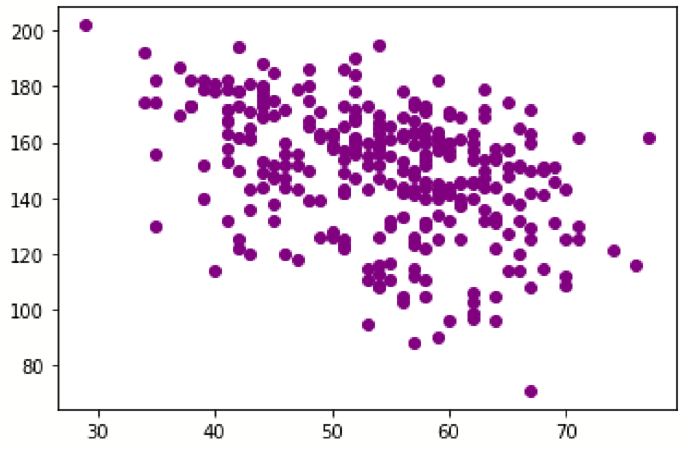
```
0.819672131147541
[[21  6]
 [ 5 29]]
```

#plotting confusion matrix



#storing the age column and the maximum heart rate column in x2 and y2 and performing scatterplot between them

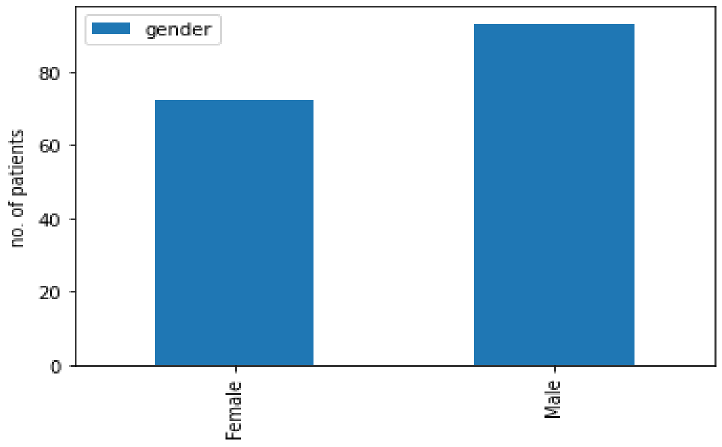
```
x2=df.iloc[:,0]
y2=df.iloc[:,7]
plt.scatter(x2,y2,c='purple')
```



#counting the no. of males and females having heart disease and storing it in two lists

```
gender=[]
males=[]
females=[]
for i in range(0,302):
    if(x[i,1]==1):
        if(y[i]==1):
            males.append(x[i,1])
    elif(x[i,1]==0):
        if(y[i]==1):
            females.append(x[i,1])
```

```
plotdata = pd.DataFrame({"gender":[len(females),len(males)]},
   index=["Female","Male"])
```
# Plot a bar chart
```
plotdata.plot(kind="bar")
plt.ylabel('no. of patients')
```



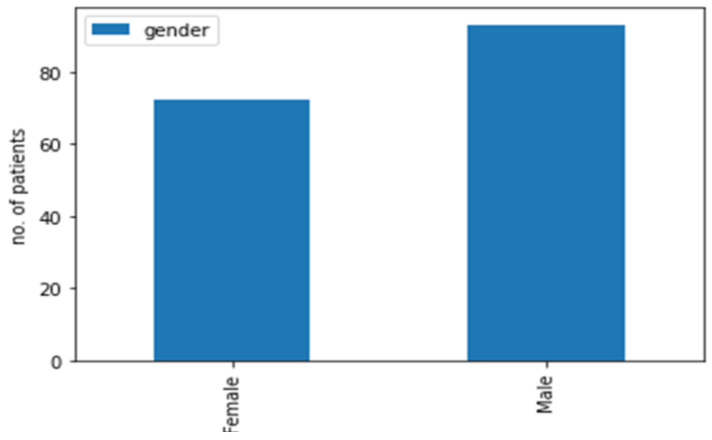
5.2.1 KNN WITH PCA

# importing the libraries

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.utils import shuffle

# importing the dataset

df=pd.read_csv('heart.csv')

print(df)
```

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

[303 rows x 14 columns]
```

```
# extract x and y from the dataset
x=df.iloc[ : , :-1].values
y=df.iloc[ : ,-1].values
#Bar plot for Count of male and female having heart disease
gender=[]
males=[]
females=[]
count=[]
c=0
c2=0
for i in range(0,302):
   if(x[i,1]==1):
      if(y[i]==1):
         c=c+1
         males.append(x[i,1])
   elif(x[i,1]==0):
      if(y[i]==1):
         c2=c2+1
         females.append(x[i,1])
plotdata=pd.DataFrame({"gender":[len(females),len(males)]},index=["Females","Males"])
plotdata.plot(kind="bar")
plt.ylabel('no of patients')
```

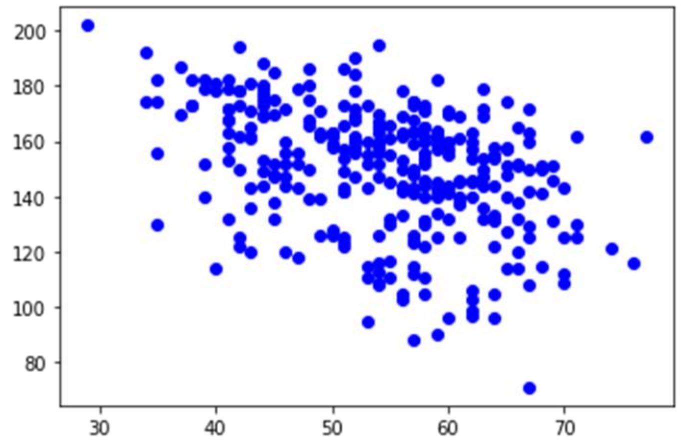


#Scatter Plot

x1=df.iloc[:,0]

y1=df.iloc[:,7]

plt.scatter(x1,y1,c='blue')



#scaling x column

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

x=sc.fit_transform(x)

#performing PCA

from sklearn.decomposition import PCA

pca=PCA(n_components=2)

x=pca.fit_transform(x)

#testing and training the columns

from sklearn.model_selection import train_test_split

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=0)

#implementing the knn

from sklearn.neighbors import KNeighborsClassifier

classifier=KNeighborsClassifier(n_neighbors=7,metric='minkowski',p=2)

classifier.fit(xtrain,ytrain)

ypred=classifier.predict(xtest)

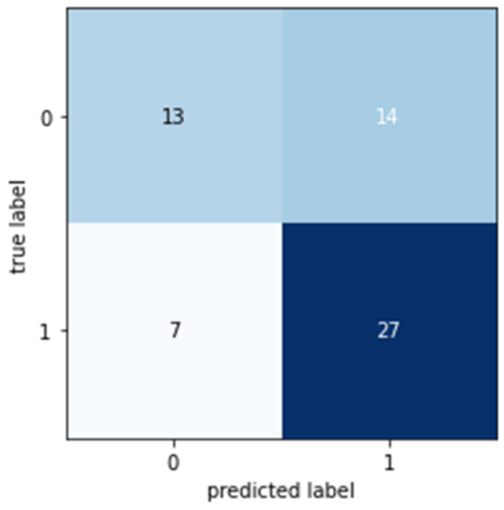
#predicting accuracy score and printing the confusion matrix

```
from sklearn.metrics import accuracy_score,confusion_matrix

acc=accuracy_score(ytest,ypred)

cm=confusion_matrix(ytest,ypred)

print(acc)

print(cm)
```

```
0.6557377049180327
 [[13 14]
[ 7 27]]
```

#plotting confusion matrix

```
from mlxtend.plotting import plot_confusion_matrix,plot_decision_regions

plot_confusion_matrix(cm)

plt.show()
```



#plotting decision region

```
plot_decision_regions(X=xtest,y=ytest,clf=classifier)
```

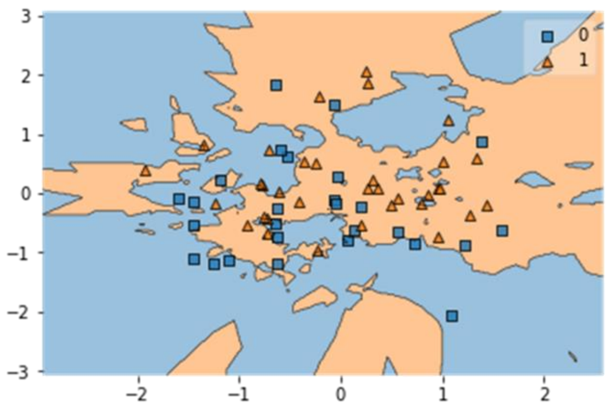
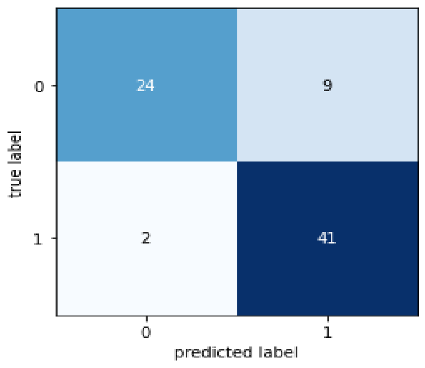
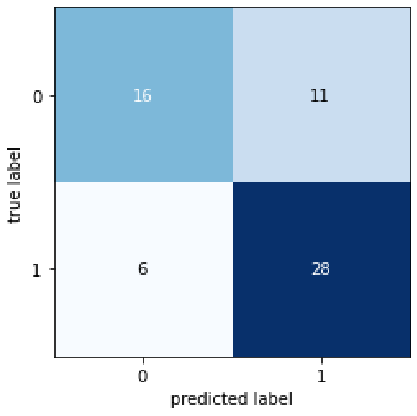
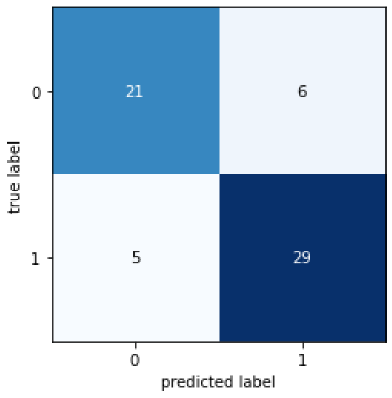
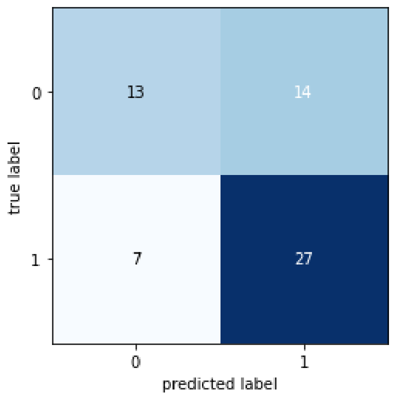
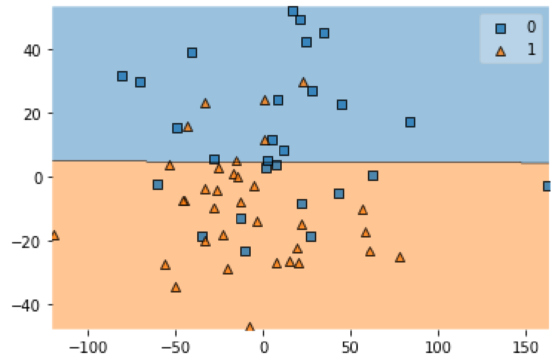
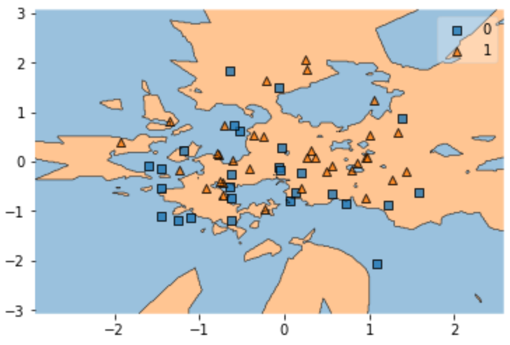


TABLE OF FIGURES/GRAPHS

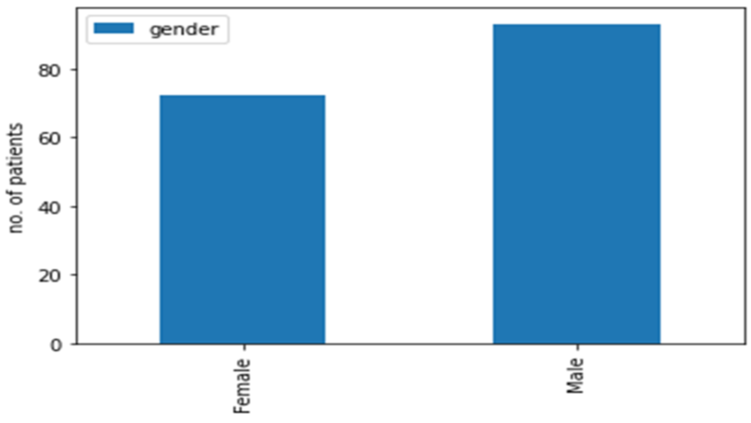
CONFUSION MATRIX

MODELS	CONFUSION MATRIX
Support Vector Machines (SVM)	
K-Nearest Neighbor classifier (K-NN)	
SVM with PCA	
KNN with PCA	

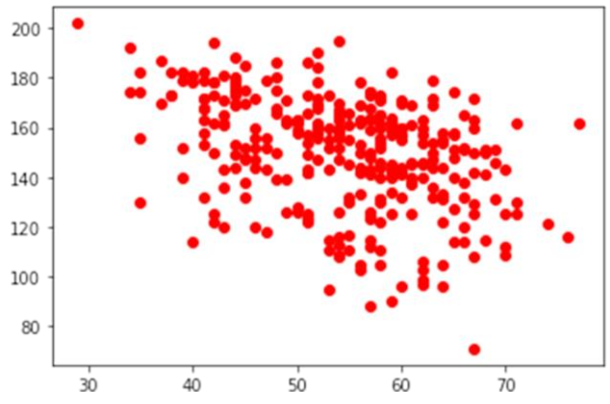
DECISION REGIONS

MODELS	DECISION REGION
SVM WITH PCA	
KNN WITH PCA	

Bar plot for Count of male and female having heart disease



Scatterplot between age and maximum heart rate



6. CONCLUSION

Thus we have coded solutions and have found out the accuracies for the algorithms are as follows:

Algorithm	Accuracies
Support Vector Machines (SVM)	85.52%
K-Nearest Neighbor classifier (K-NN)	81.96%
SVM with PCA	72.13%
KNN with PCA	65.57%

Thus as we get maximum accuracy in Support Vector Machines(SVM). So we choose SVM as our final algorithm to find possibilities of heart related diseases. If for same problem statement higher accuracies are required then following things can be done

- We need more data to train our model in multitude to train cases. With the given data we can get only above accuracies.
- There are limitations of Machine Learning such as Machine learning cannot handle multidimensional of arrays. To improve our chances for better prediction we need to implement another special application of Machine Learning which is also known as Deep Learning.
- Along with PCA we can also use Select KNN as best Algorithm which gives us accuracy of 85.38% (with n=7 features) The reasons why cases which use PCA along with them have relatively lower accuracies is because PCA takes only those features into consideration which contribute for a significant change in outputs. In cases where data provided is less this can lead to significant loss of information which can further increase accuracy.

7. ACKNOWLEDGEMENT

In the accomplishment of this project successfully, primarily We would like to thank god for providing us this opportunity and our Machine Learning teacher Mr.Guruvansh Singh whose valuable guidance has proved to be very helpful to us. His suggestions and his instruction have served as a major contributor towards completion of this project. Last but not least I will like to thank my team mates who have made it possible for the successful competition of this project.