FALL
2021

ECEN 649 FINAL PROJECT

FARDEEN HASIB MOZUMDER

UIN: 730007771

PROJECT TITLE

Heart Disease Detection using attributes/features of patients using known Machine Learning models (SVM, KNN & DECISION TREE)

Main goal/Objective

Machine Learning is used across many spheres around the world. The healthcare industry is no exception. Machine Learning can play an essential role in predicting presence/absence of Locomotor disorders, heart diseases and more. Such information, if predicted well in advance, can provide important insights to doctors who can then adapt their diagnosis and treatment per patient basis.

In this project I worked on predicting potential heart diseases in people using Machine Learning algorithms. The algorithms included K-Neighbors Classifier, Support Vector Classifier and Decision Tree Classifier. The dataset has been taken from https://archive.ics.uci.edu/ml/datasets/heart+disease

The Dataset

Before we get into the details of the project implementation, let us observe our 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

A short description of all the features in the dataset is given below:

age: age in years

sex: (1 = male; 0 = female)

cp: chest pain type (1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic)
trestbps: resting blood pressure (in mm Hg on admission to the hospital)
chol: serum cholestoral in mg/dl
fbs: (fasting blood sugar > 120 mg/dl) (1: true; 0: false)
restecg: resting electrocardiographic results (0: normal, 1: having abnormality, 2 = showing ventricular hypertrophy)
thalach: maximum heart rate achieved
exang: exercise induced angina (1: yes, 0: no)
oldpeak: ST depression induced by exercise relative to rest
slope: the slope of the peak exercise ST segment (1: upsloping, 2: flat, 3: downsloping)
ca: number of major vessels (0-3) colored by flourosopy
thal: thalassemia(a blood disorder) (3 = normal; 6 = fixed defect; 7 = reversable defect)
target: heart disease (0: no, 1: yes)

Data preprocessing

Missing Values:

There were no missing values in our dataset. As a result, I did not have to clean the dataset. But, if there were any missing values in the dataset, generally one of these two steps are done:

- Replacing the missing values by the mean (or median) of that feature
- Delete the entire row where we find any missing values (easier step, but in cost of reducing the dataset)

One Hot Encoding:

There are several features in the dataset which are categorical in nature but are represented by decimal values. For example, CP (chest pain) is represented by decimal values 1-4 but they are actually the type of chest pain the patient experienced while admitting to the hospital, such as:

"1" represents the patient has "typical angina",
"2" represents the patient has "atypical angina",
"3" represents the patient has "non-anginal pain",
"4" represents the patient has "asymptomatic pain"

Before applying our models, we need to convert these decimal values to categorical values as CP_0, CP_1, CP_2, CP_3. That means, if a patient has typical angina chest pain he will fall under CP_0 category. If a patient has atypical angina chest pain he will fall under CP_1 category. If a patient has non-anginal chest

pain he will fall under CP_2 category and if a patient has asymptomatic chest pain he will fall under CP_3 category.

Similarly, we convert Thal (thalassemia blood disorder) to categorical Thal_1, Tha_2, Thal_3 and Slope (the slope of the peak exercise ST segment) to categorical Slope_0, Slope_1, Slope_2.

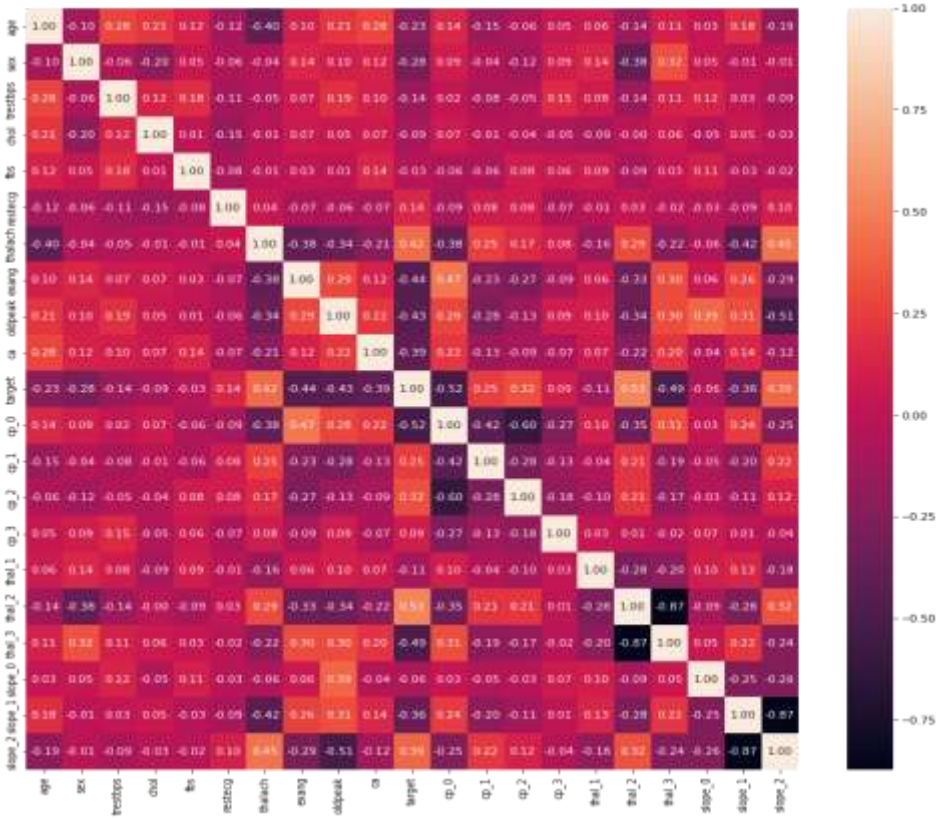
Normalization:

We normalize our training data by mapping non-categorical values to [0 to 1], before applying to our models as it helps in models' training process and increase the models' overall accuracies.

Data Inspection and Visualization

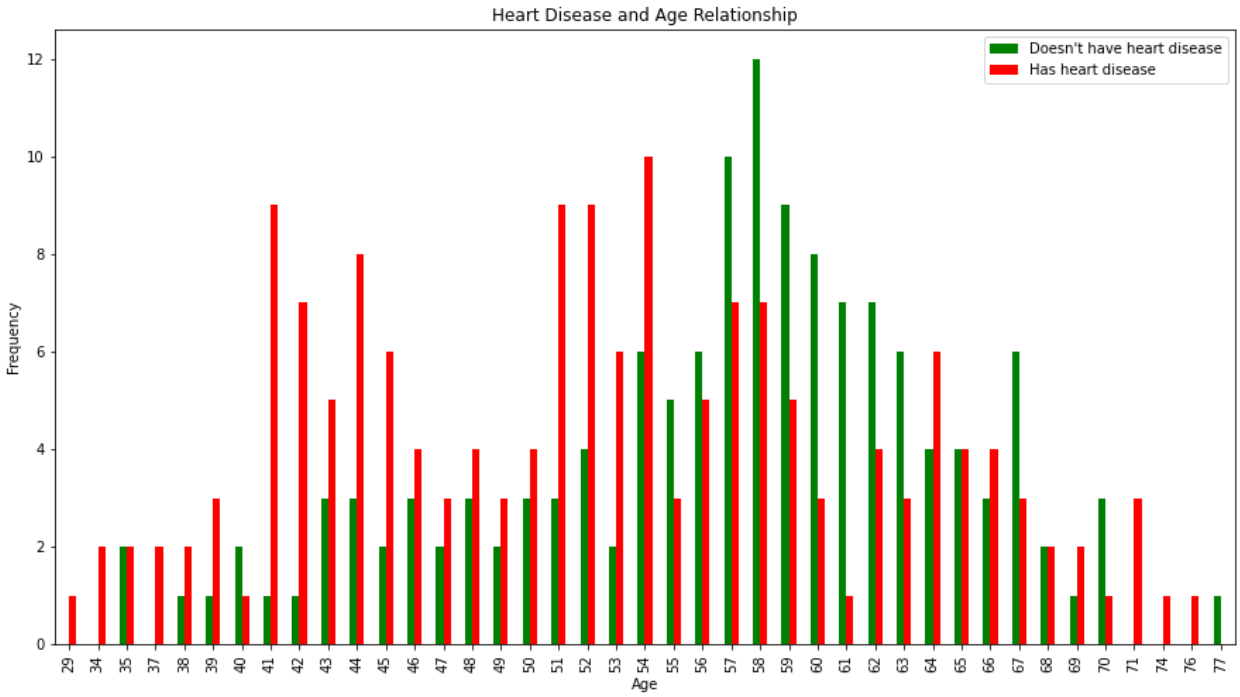
Correlation Heatmap:

Let us see the correlation matrix of features and try to analyze it

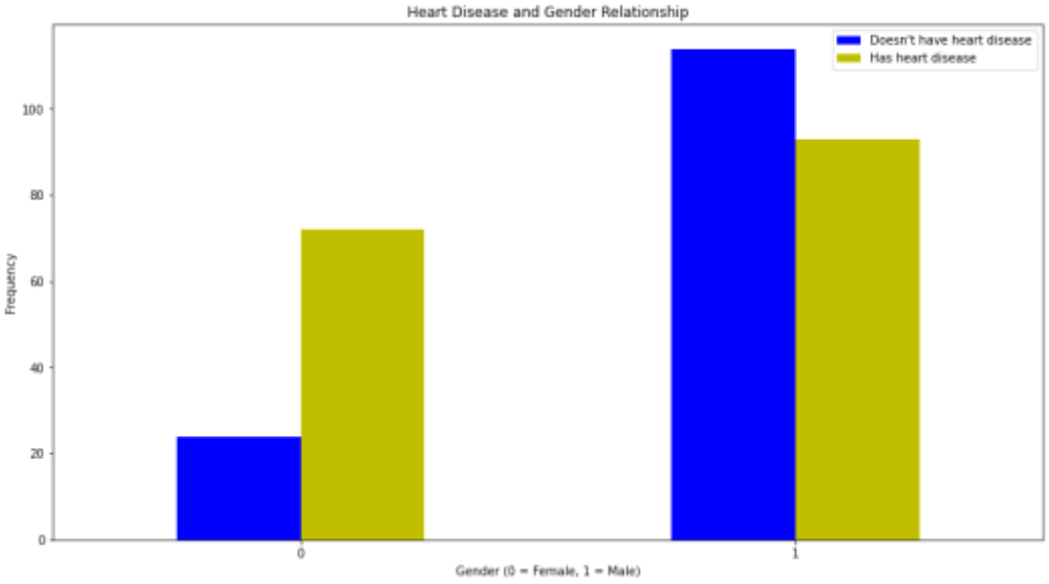


We can see that there is no single feature that has a very high correlation with our target value. Also, among all the features Thal_2 has the highest positive correlation and many features has negative correlation as well.

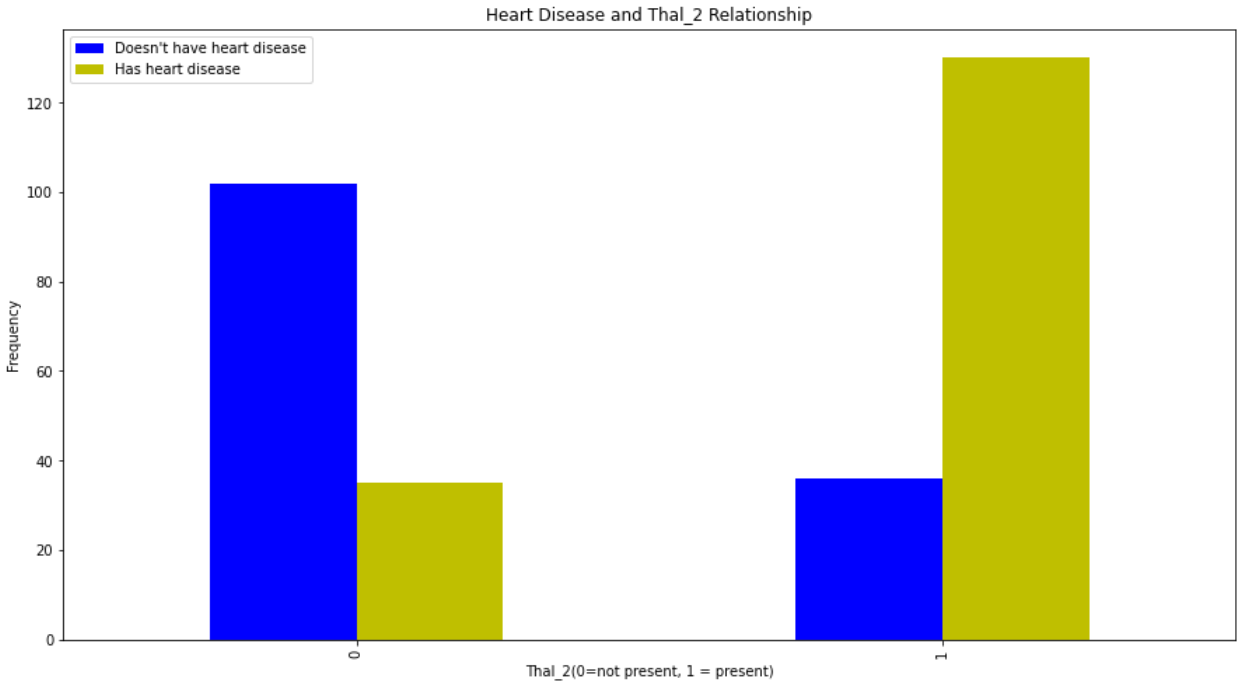
Let us see some histogram plots to see the relationship of that feature and the presence of heart disease in a patient.



Here, we can see the patients of age group 40-45 and 50-55 years are more prone to be affected by heart disease.



Here, we can see male patients are more prone to have heart disease.



Here, we can clearly see that if a patient has Thalassemia of "fixed type" (Thal_2), then it is more likely that the patient has heart disease. We also verified this observation in correlation heatmap earlier.

The Models

SVM (Support Vector Machine):

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.

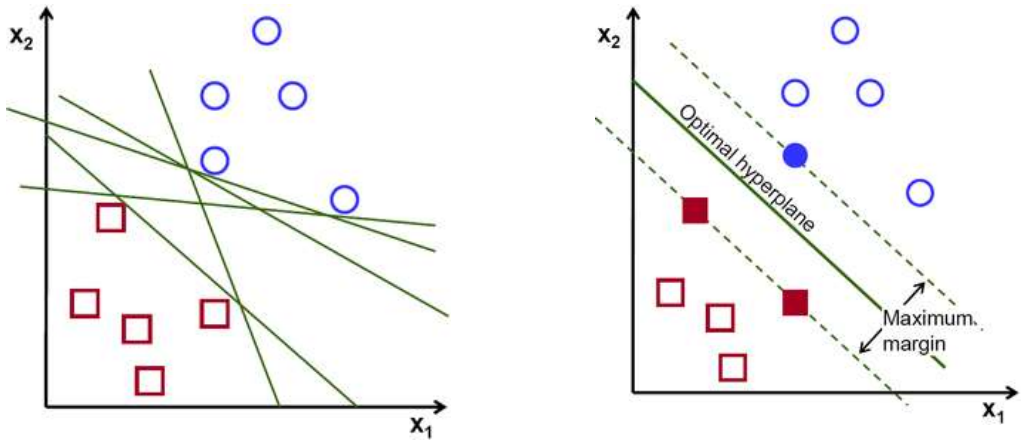


Figure: The data points are separated by maximum margin by optimal hyperplane using SVM

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.

K Nearest Neighbors (KNN):

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

"Birds of a feather flock together."

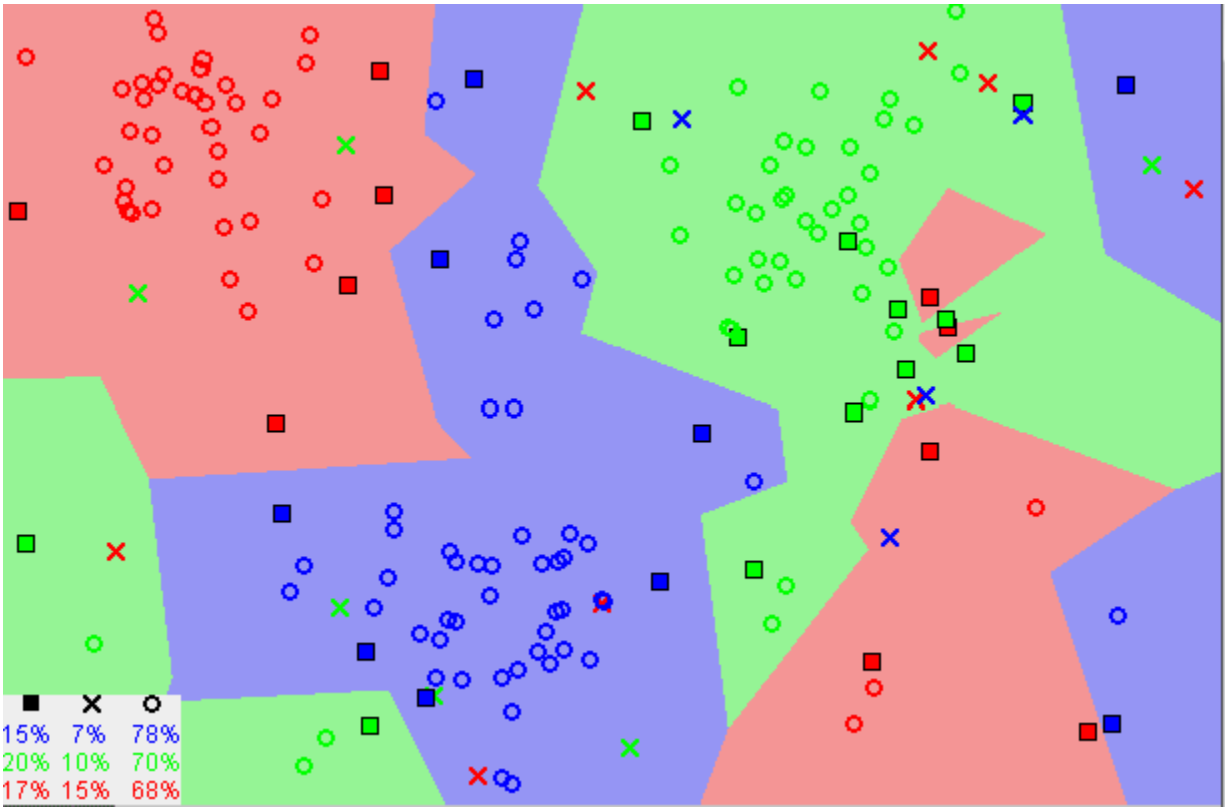


Figure: KNN clusters datapoints with similar attributes together

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.

Decision Tree:

A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

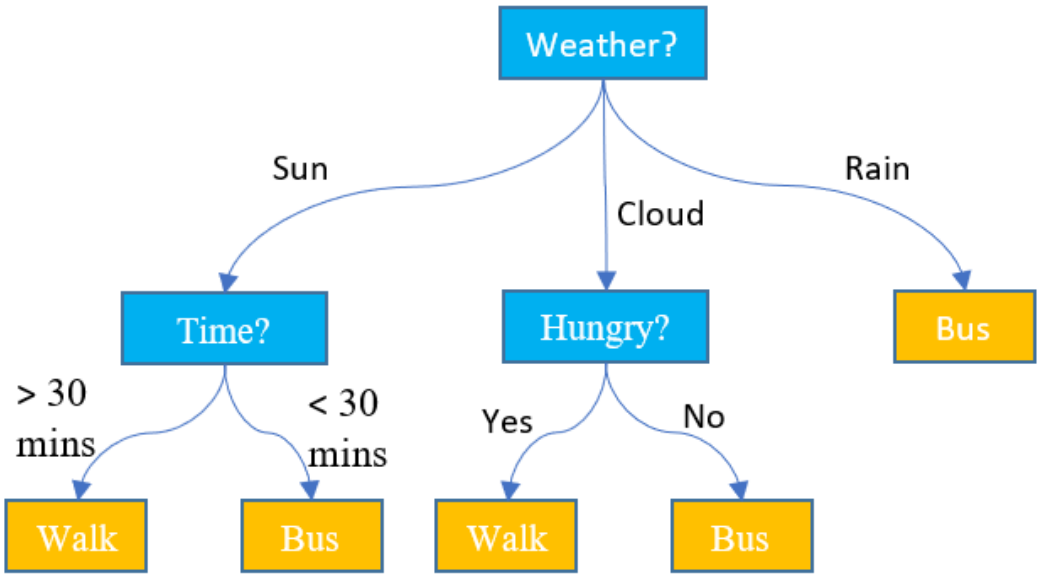


Figure: A simple decision tree to make a decision whether to walk or to take bus depending on weather, time in hand, hunger state.

Models and their Performances

Before we implement our models on the preprocessed dataset and evaluate their performance, let us see definitions and formulas of some performance metrics.

Recall Score:

Recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made. Unlike precision that only comments on the correct positive predictions out of all positive predictions, recall provides an indication of missed positive predictions

Since this is a medical problem, we need to avoid false negatives as much as possible. So, **recall** is really important for this situation and we will consider recall score along with accuracy to evaluate our model performances.

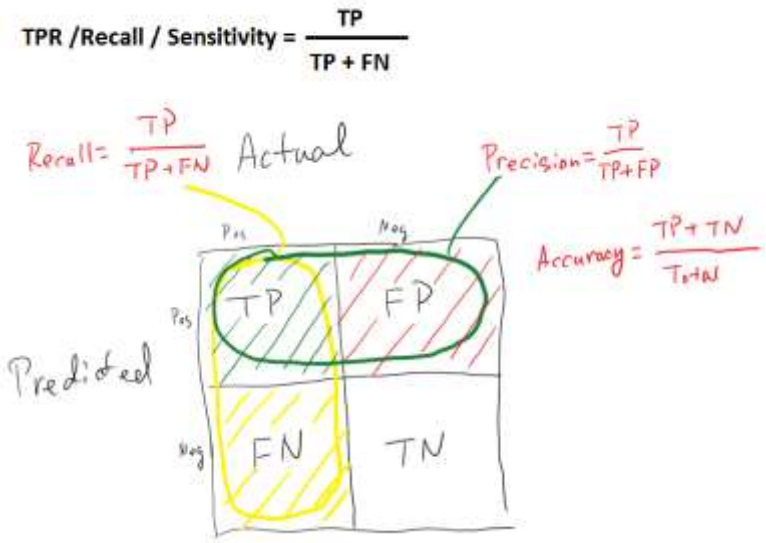


Figure: Definition of some performance metrics using confusion matrix diagram

Code implantation of the models in python using sci-kit learn

Support Vector Classifier:

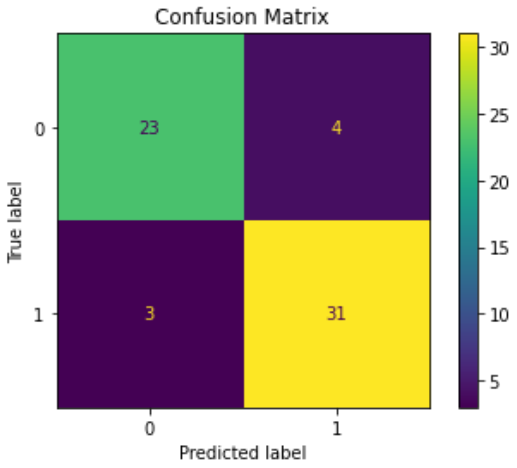
```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, recall_score
svc=SVC(kernel= 'rbf', gamma = 0.30 ,random_state=0)
svc.fit(X_train, y_train)
svm_pred = svc.predict(X_test)
svm_recall = round(recall_score(y_test,svm_pred,average='weighted'),3)
print (classification_report(y_test, svm_pred, labels=None, target_names=None, sample_weight=None, digits=3, output_dict=False))
acc = accuracy_score(y_test, svm_pred)
print ("Accuracy: %.3f" % acc)
```

```
              precision    recall  f1-score   support

           0      0.885     0.852     0.868        27
           1      0.886     0.912     0.899        34

    accuracy                          0.885        61
   macro avg      0.885     0.882     0.883        61
weighted avg      0.885     0.885     0.885        61

Accuracy: 0.885
```

We can see that the Support vector Classifier's accuracy is 88.5 % and recall score is also 88.5 %.

Plotting the test findings in confusion matrix:

```
from sklearn.metrics import plot_confusion_matrix

disp = plot_confusion_matrix(svc, X_test, y_test)

disp.ax_.set_title("Confusion Matrix")

plt.show()
```



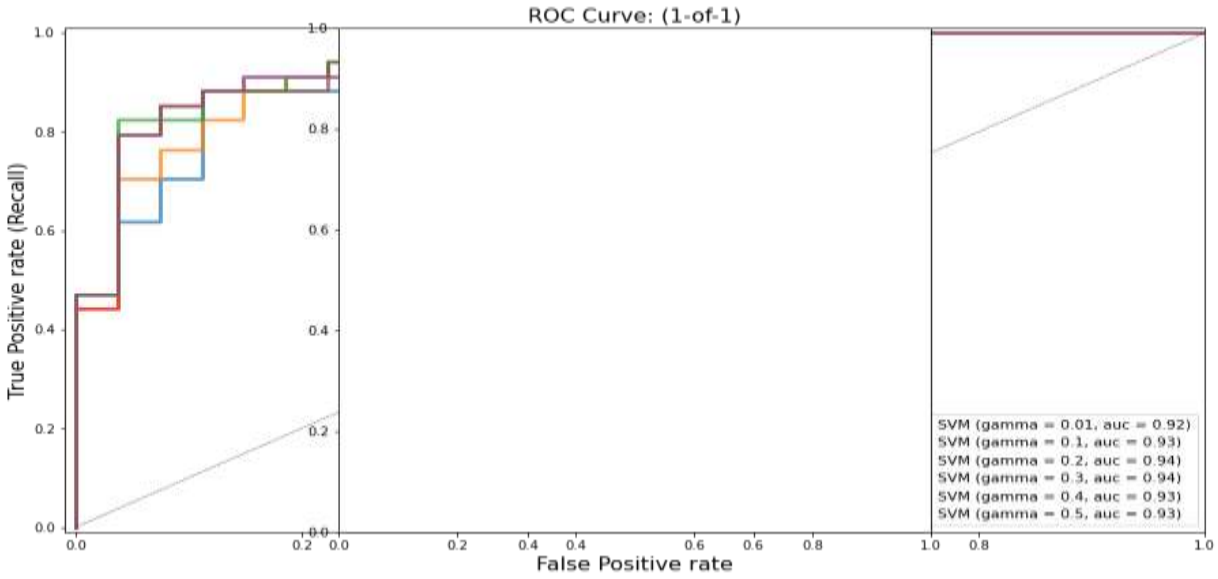
We can calculate **recall** manually looking at the confusion matrix. For recall: TP / (TP + FN) -> 23 / (23 + 3) = **0.885** which is same as what we found using sklearn's recall_score() method.

Note, I used the “rbf” kernel with gamma set to 0.3 for my support vector classifier. We did cross validation and found setting “gamma” to 0.3 gives the best AUC (area under curve of ROC)/accuracy and hence selected that value.

Cross validation of “gamma” parameter of RBF kernel:

```
for g in [0.01, 0.1, 0.20, 0.30, 0.40, 0.50]:
    svm = SVC(gamma=g).fit(X_train, y_train)
    y_score_svm = svm.decision_function(X_test)
    fpr_svm, tpr_svm, _ = roc_curve(y_test, y_score_svm)
    roc_auc_svm = auc(fpr_svm, tpr_svm)
    accuracy_svm = svm.score(X_test, y_test)
```

gamma = 0.01 accuracy = 0.87 AUC = 0.92
gamma = 0.1 accuracy = 0.87 AUC = 0.93
gamma = 0.2 accuracy = 0.87 AUC = 0.94
gamma = 0.3 accuracy = 0.89 AUC = 0.94
gamma = 0.4 accuracy = 0.87 AUC = 0.93
gamma = 0.5 accuracy = 0.87 AUC = 0.93



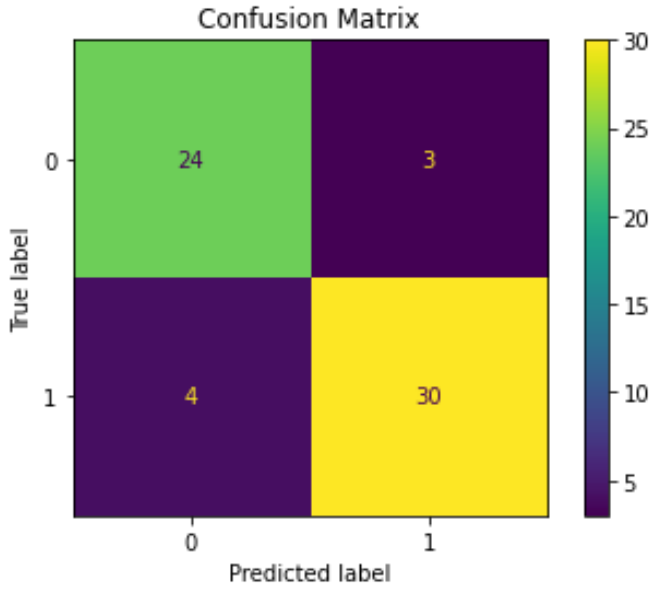
K- Nearest Neighbors (KNN):

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors= 7)

knn.fit(X_train, y_train)

knn_pred = knn.predict(X_test)


# storing recall_score for later comparision

knn_recall = round(recall_score(y_test,knn_pred,average='weighted'),3)


print (classification_report(y_test, knn_pred, labels=None, target_names=None, sample_weight=None, digits=3, output_dict=False))

print

print ("Accuracy: %.3f" % accuracy_score(y_test, knn_pred))


from sklearn.metrics import plot_confusion_matrix

disp = plot_confusion_matrix(knn, X_test, y_test)

disp.ax_.set_title("Confusion Matrix")

plt.show()
```

```
              precision    recall  f1-score   support

           0      0.857     0.889     0.873        27
           1      0.909     0.882     0.896        34

    accuracy                          0.885        61
   macro avg      0.883     0.886     0.884        61
weighted avg      0.886     0.885     0.885        61

Accuracy: 0.885
```

We can see that the KNN Classifier’s accuracy and recall score is 88.5 %.

Plotting the test findings in confusion matrix:



Decision Tree Classifier:

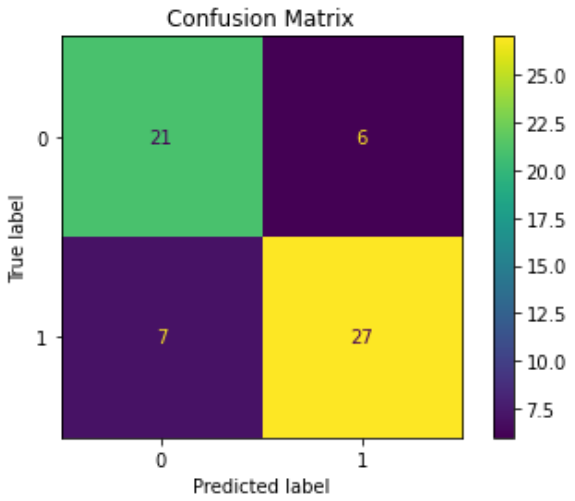
```
from sklearn import tree
tree = tree.DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)


# predict the target on the test dataset
tree_predict_test = tree.predict(X_test)


tree_recall = round(recall_score(y_test, tree_predict_test, average='weighted'),3)


print (classification_report(y_test, tree_predict_test, labels=None, target_names=None,
sample_weight=None, digits=3, output_dict=False))
```

	precision	recall	f1-score	support
0	0.750	0.778	0.764	27
1	0.818	0.794	0.806	34
accuracy			0.787	61
macro avg	0.784	0.786	0.785	61
weighted avg	0.788	0.787	0.787	61

We can see that the accuracy and recall score for the decision tree classifier is 78.7%.

Plotting the confusion matrix for the decision tree classifier:

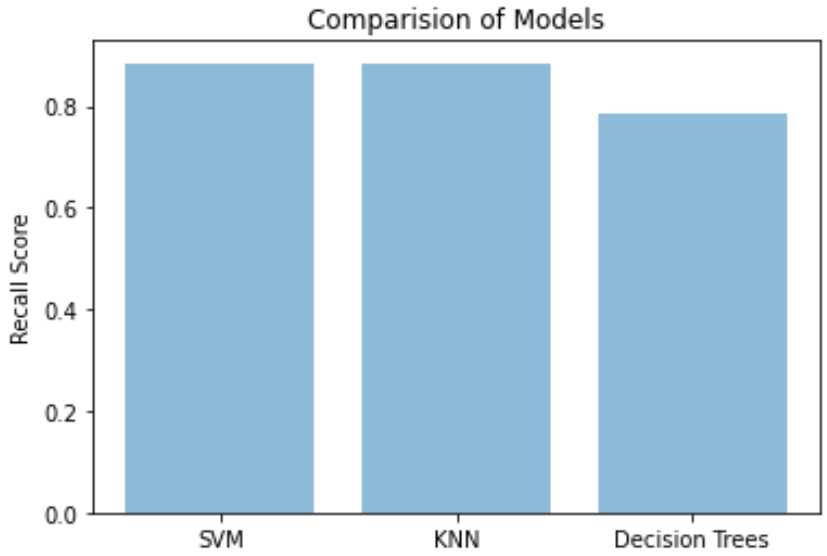
```
disp = plot_confusion_matrix(tree, X_test, y_test)
disp.ax_.set_title("Confusion Matrix")
plt.show()
```



Results

Here is the final result of our models in terms of recall scores:

	SVM	KNN	Decision Trees
Recall Score	0.885	0.885	0.787



We can see SVM and KNN classifiers' performances are similar and they both performed better the decision tree. The reason behind decision tree's underperformance may be because of the fact that, decision trees run the risk of being overfitted to the training data if not pruned properly. As we did not prune our decision tree, it is quite understandable that our decision tree could not perform well as other two models.

References

I found the following tutorials quite useful to complete my project,

For SVM: https://www.youtube.com/watch?v=8A7L0GsBiLQ

For Decision tree: https://www.youtube.com/watch?v=q90UDEgYqeI

For KNN: https://www.youtube.com/watch?v=4HKqjENq9OU