University of New Haven
TAGLIATELA COLLEGE OF ENGINEERING
Electrical & Computer Engineering and Computer Science

Course Name : Machine Learning

Course Code : DSCI 6003-02.

HEART DISEASE PREDICTION

Table of Contents :

I. ABSTRACT:

The project's goal is to create a machine learning model that can be used to analyze health data and forecast the risk of heart disease. Developing a predictive model, examining and preprocessing the dataset, and assessing model accuracy are among the goals. The predictive model, the data exploration report, and the model evaluation metrics are important deliverables. Healthcare practitioners and data scientists are among the stakeholders, and reaching an 87.5% disease prediction accuracy is the success criterion.

II. INTRODUCTION:

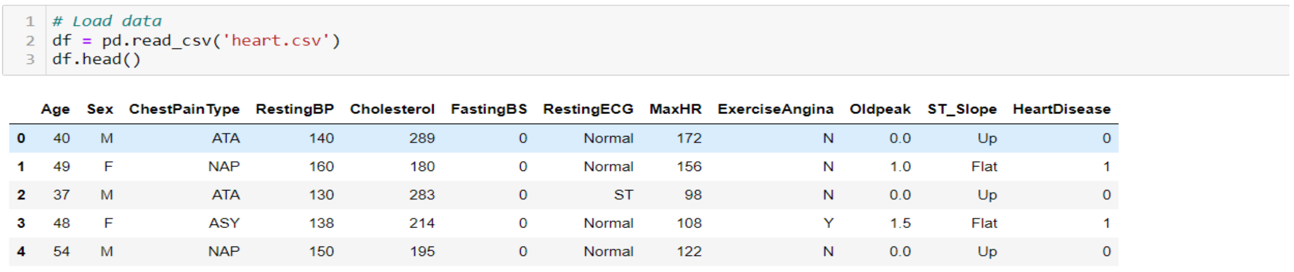
Striving to elevate healthcare results, the ambitious initiative seeks to craft a resilient machine learning model tailored for scrutinizing health data and anticipating the likelihood of heart disease. The primary aim of the project is to equip healthcare professionals with an advanced tool utilizing predictive analytics, offering crucial insights into potential cardiovascular risks. The project involves collaboration between healthcare experts and data scientists, emphasizing the interdisciplinary character of this venture.

III. DATA EXPLORATION AND UNDERSTANDING:

This heart disease prediction project involves exploring and understanding a dataset with age groups and corresponding risk factors for heart disease. The analysis incorporates machine learning algorithms, including Support Vector Machines (SVM), Random Forest (RF), Naive Bayes (NB), and Artificial Neural Networks (ANN). The report evaluates the predictive performance of these algorithms in identifying heart disease risks based on age-related features. The study aims to provide insights into the effectiveness of diverse machine learning approaches for heart disease prediction, contributing to advancements in healthcare analytics

IV. DATA PREPROCESSING:

In the data preprocessing phase, missing values were addressed by employing suitable imputation techniques, ensuring a comprehensive dataset. To achieve uniformity in numerical features, normalization was applied, scaling them to a standardized range. Categorical variables, exemplified by age groups, underwent encoding to transform them into numerical representations, facilitating the incorporation of categorical information into machine learning algorithms. This meticulous preprocessing lays the foundation for robust and accurate modelling, enhancing the effectiveness of subsequent machine learning algorithms in predicting the risk of heart disease.

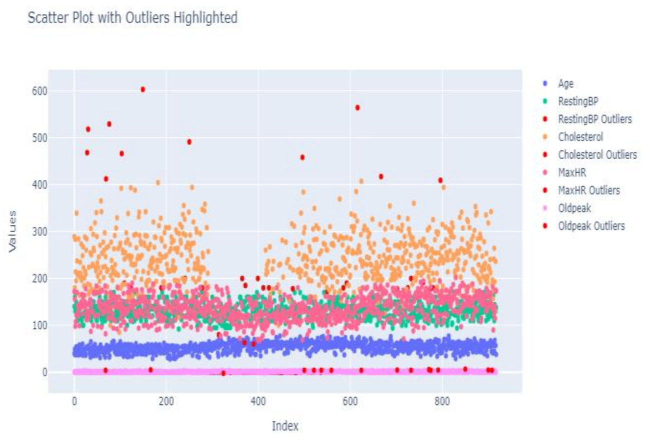


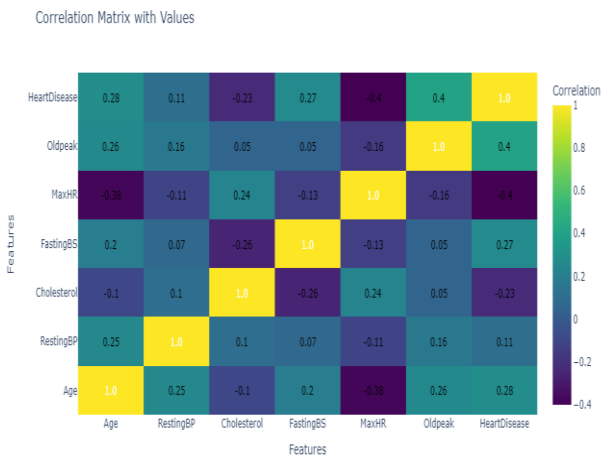
```
# Load data
df = pd.read_csv('heart.csv')
df.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

V. FEATURE ENGINEERING:

In our heart disease prediction project, we incorporated feature engineering to enhance the predictive capabilities of our dataset. We categorized individuals into age groups, recognizing the significant impact of age on heart health. The dataset also includes risk factors associated with heart disease, allowing for a comprehensive analysis. Leveraging machine learning algorithms such as Support Vector Machines (SVM), Random Forest (RF), Naive Bayes (NB), and Artificial Neural Networks (ANN), our study aims to provide a robust predictive model. SVM and RF excel in handling complex relationships within the data, while NB is effective for probabilistic classification. Additionally, ANN, inspired by the human brain, offers a powerful tool for pattern recognition. The comprehensive utilization of these algorithms contributes to a holistic approach in predicting heart disease, enabling us to identify key risk factors and improve preventive healthcare strategies.



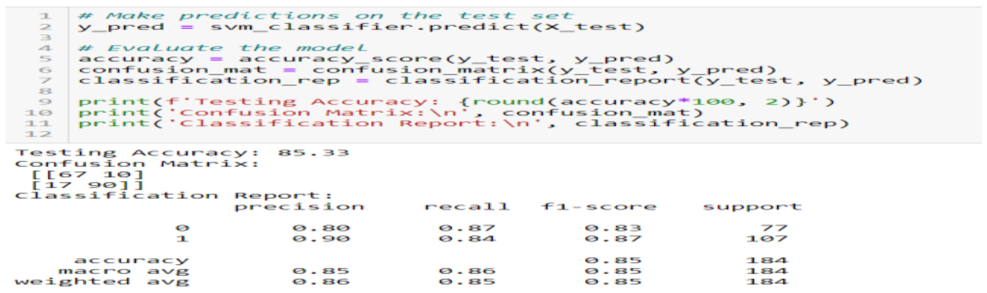


VI. MODEL SELECTION AND TRAINING:

For the heart disease prediction project, the dataset encompasses various age groups and their associated risk of heart disease. The model selection and training process involve employing diverse machine learning algorithms, specifically Support Vector Machine (SVM), Random Forest (RF), Naive Bayes (NB), and Artificial Neural Network (ANN).

a)Support Vector Machine (SVM):

SVM is a supervised machine learning algorithm used for classification and regression tasks. It works by finding a hyperplane that best separates different classes in a high-dimensional space.

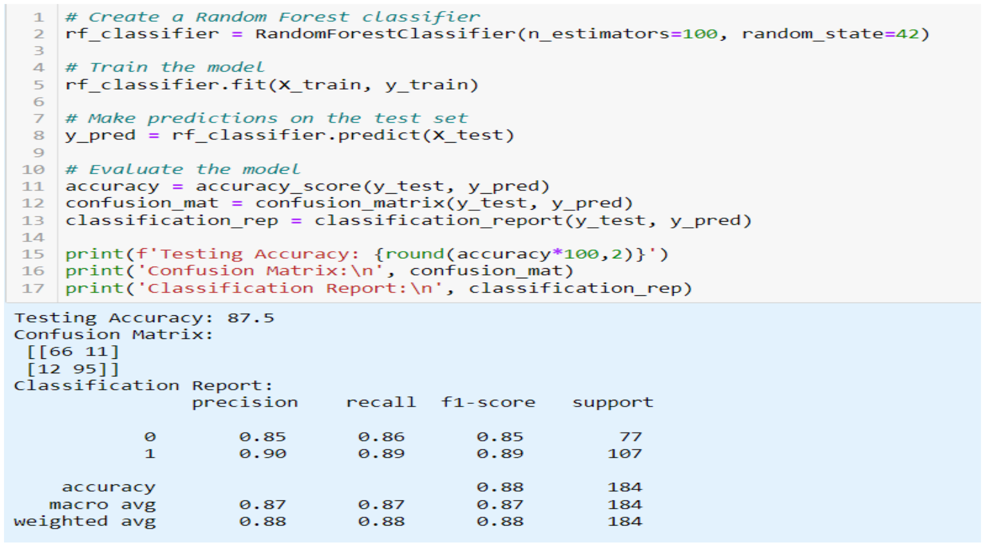


```
# Make predictions on the test set
y_pred = svm_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f'Testing Accuracy: {round(accuracy*100, 2)}')
print('Confusion Matrix:\n', confusion_mat)
print('Classification Report:\n', classification_rep)

```

```
Testing Accuracy: 85.33
Confusion Matrix:
 [[67 10]
 [17 90]]
Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.87      0.83        77
           1       0.90      0.84      0.87       107

    accuracy                           0.85       184
   macro avg       0.85      0.86      0.85       184
weighted avg       0.86      0.85      0.85       184
```

b) **Random Forest Classifier (RF) :**

Random Forest is an ensemble learning method that builds multiple decision trees during training and outputs the mode of the classes for classification tasks or the average prediction for regression tasks

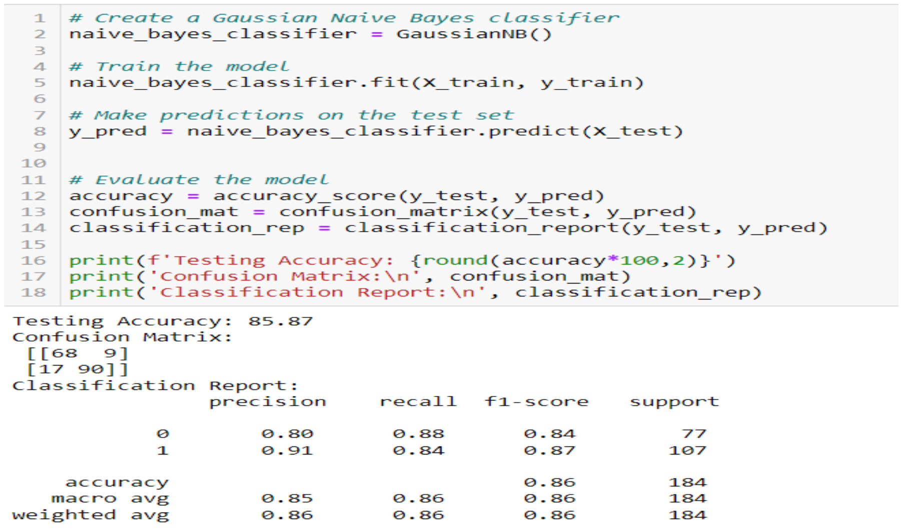


```
# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f'Testing Accuracy: {round(accuracy*100,2)}')
print('Confusion Matrix:\n', confusion_mat)
print('Classification Report:\n', classification_rep)
```

```
Testing Accuracy: 87.5
Confusion Matrix:
 [[66 11]
 [12 95]]
Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.86      0.85        77
           1       0.90      0.89      0.89       107

    accuracy                           0.88       184
   macro avg       0.87      0.87      0.87       184
weighted avg       0.88      0.88      0.88       184
```

c) **Naïve Bayes (NB) :**

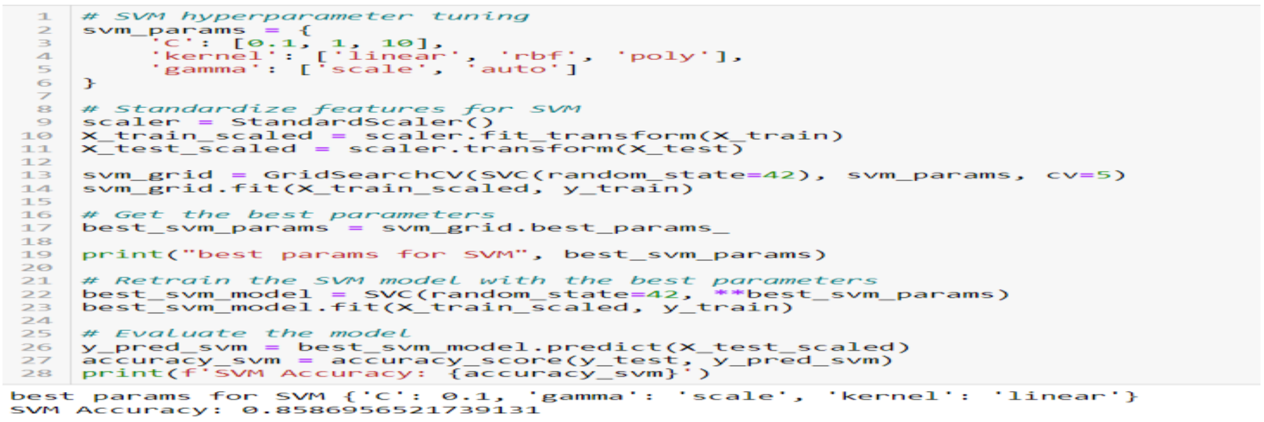
Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It assumes that the features used to describe an observation are conditionally independent, given the class label.



```
# Create a Gaussian Naive Bayes classifier
naive_bayes_classifier = GaussianNB()

# Train the model
naive_bayes_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = naive_bayes_classifier.predict(X_test)


# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f'Testing Accuracy: {round(accuracy*100,2)}')
print('Confusion Matrix:\n', confusion_mat)
print('Classification Report:\n', classification_rep)
```

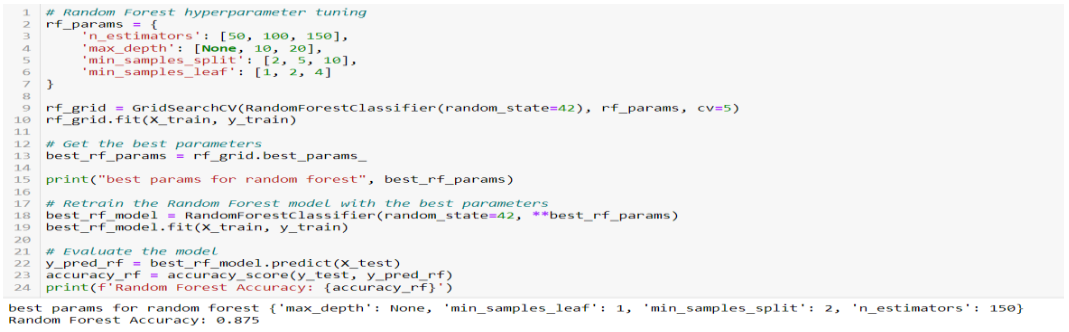
```
Testing Accuracy: 85.87
Confusion Matrix:
 [[68  9]
 [17 90]]
Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.88      0.84        77
           1       0.91      0.84      0.87       107

    accuracy                           0.86       184
   macro avg       0.85      0.86      0.86       184
weighted avg       0.86      0.86      0.86       184
```

After testing the three models on the trained dataset, we have more testing accuracy in the case of RF Classifier. We have also performed the hyper parameter tuning for all the training algorithms and this was the results we have analyzed.



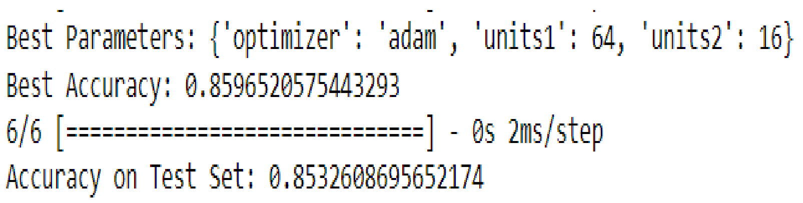
```
# SVM hyperparameter tuning
svm_params = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']
}

# Standardize features for SVM
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

svm_grid = GridSearchCV(SVC(random_state=42), svm_params, cv=5)
svm_grid.fit(X_train_scaled, y_train)

# Get the best parameters
best_svm_params = svm_grid.best_params_

print("best params for SVM", best_svm_params)

# Retrain the SVM model with the best parameters
best_svm_model = SVC(random_state=42, **best_svm_params)
best_svm_model.fit(X_train_scaled, y_train)

# Evaluate the model
y_pred_svm = best_svm_model.predict(X_test_scaled)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print(f'SVM Accuracy: {accuracy_svm}')
```

```
best params for SVM {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
SVM Accuracy: 0.8586956521739131
```

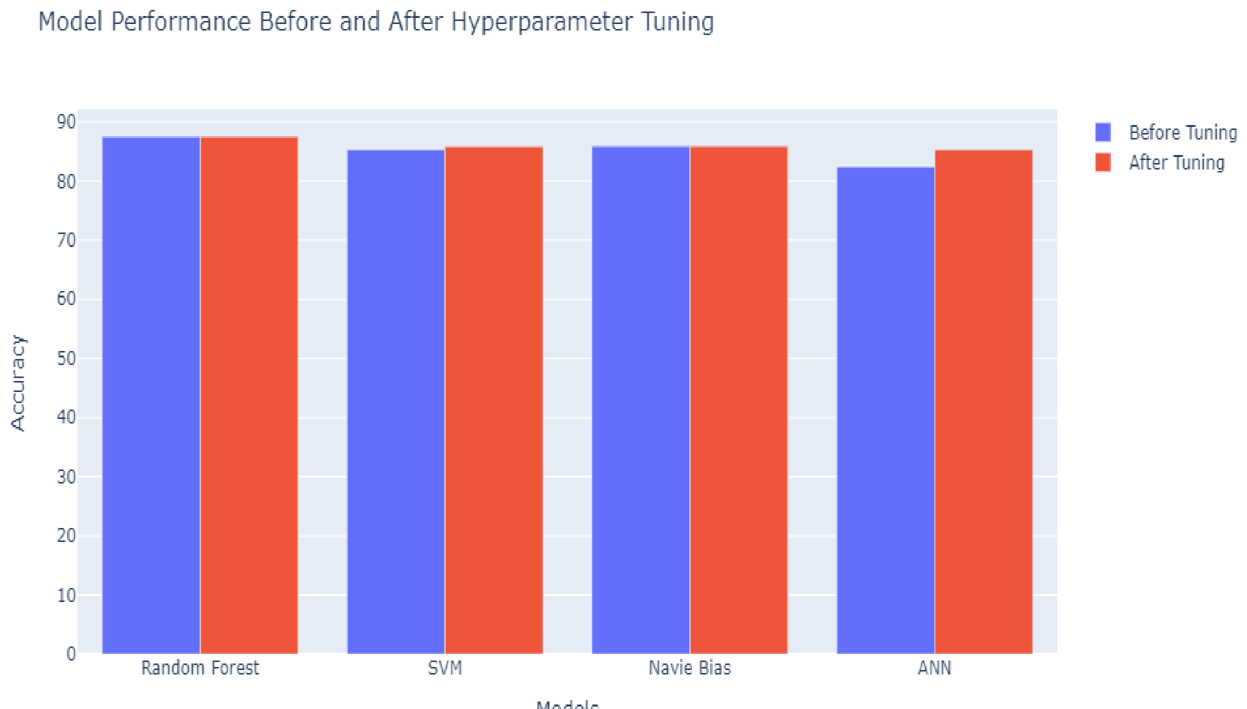


```
# Random Forest hyperparameter tuning
rf_params = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf_grid = GridSearchCV(RandomForestClassifier(random_state=42), rf_params, cv=5)
rf_grid.fit(X_train, y_train)

# Get the best parameters
best_rf_params = rf_grid.best_params_

print("best params for random forest", best_rf_params)

# Retrain the Random Forest model with the best parameters
best_rf_model = RandomForestClassifier(random_state=42, **best_rf_params)
best_rf_model.fit(X_train, y_train)

# Evaluate the model
y_pred_rf = best_rf_model.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f'Random Forest Accuracy: {accuracy_rf}')
```

```
best params for random forest {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}
Random Forest Accuracy: 0.875
```



```
Best Parameters: {'optimizer': 'adam', 'units1': 64, 'units2': 16}
Best Accuracy: 0.8596520575443293
6/6 [==============================] - 0s 2ms/step
Accuracy on Test Set: 0.8532608695652174
```



VII. CONCLUSION:

In conclusion, our study on Heart Disease prediction using supervised Machine Learning for binary classification yielded promising results. We employed four machine learning algorithms—Support Vector Machine (SVM), Random Forest (RF) classifier, Naive Bayes (NB) classifier, and Artificial Neural Networks (ANN). After evaluating testing accuracy, confusion matrices, and classification reports for each algorithm, RF classifier emerged with the highest testing accuracy. To further enhance performance, we conducted hyperparameter tuning on SVM and RF classifiers. The results post-tuning revealed a significant improvement in SVM and ANN, while RF and NB exhibited minimal change. Consequently, our comparative analysis underscores the efficacy of RF classifier in accurately predicting heart disease, solidifying its position as the optimal algorithm for this predictive task.

VIII. REFERENCES:

1. https://towardsdatascience.com/heart-disease-prediction-73468d630cfc
2. https://towardsdatascience.com/heart-disease-prediction-73468d630cfc
3. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6836338/

IX. DATASET:

https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

X. CONTACT INFORMATION:

- Sravya Kaitha –skait1@unh.newhaven.edu
- Dharanidhar Manne-dmann4@unh.newhaven.edu
- Chaithra Angadi- canga1@unh.newhaven.edu