Overview:

This part of the assignment involves in acquiring a classic machine learning paper, outline the work that was carried out, to replicate the experiment carried out and to lay down the results and compare between the two experiment.

Section 1. Introduction:

The paper[1] that has been used for this assignment aims in building an Intelligent Heart Disease Prediction System (IHDPS) using the following algorithms namely, Decision Trees, Naïve Bayes and Neural Network. It was coded in the .NET framework platform.

A challenge facing healthcare organizations is to provide quality services at affordable costs. Quality service can be thought of as diagnosing patients correctly and administering treatments that are effective and cheaper. When wrong clinical decisions are made, it can lead to a scenario of having disastrous consequences which are unacceptable. By making use of computer-based decision systems, many errors can be avoided.

The dataset used by this paper was the 'Heart Disease dataset' from the UCI Machine Learning repository. In particularly, the Cleveland Heart disease dataset was chosen. The classification to be done is to find if a patient has heart disease or not.

Section 2. Methodology:

The researchers have followed the CRISP-DM method to build the models. Typically, these are the phases involved in a CRISP-DM lifecycle: business understanding, data understanding, data preparation, modeling, evaluation and deployment.

In the Business understanding phase, the objectives and requirements from the business perspective was understood, to gain the knowledge and design a plan to achieve the objectives.

In the Data understanding phase, they have used the raw data and proceeded to understand the data, identify its quality, gain preliminary insights, and detect interesting subsets to form hypotheses for hidden information.

In the Data preparation phase, they have constructed the final dataset to be fed to the modeling tools. Attribute selection, data cleaning and transformation were performed in this phase.

In the modeling phase, the selection of models, application of various techniques, and calibration of the model parameters was performed.

In the evaluation phase, the model was evaluated to ensure that it achieved the business objectives.

Finally, in the deployment phase, the tasks that are needed to use the models were specified.

To maintain consistency, the researchers had converted all the variables to a categorical type. The data is already processed and doesn't have any NA values, missing values.

Data Mining Extension (DMX) query language was used for model creation, model training, model prediction and model content access. The trained models were evaluated against the test datasets for accuracy and effectiveness before they were deployed in IHDPS. The models were validated using Lift Chart and Classification Matrix.

Section 3. Validation and Evaluation:

The authors tried two ways of using the Lift Chart, one with predictable value (target column) and one without the predictable value. The Lift chart shows that if only 50% of population were trained for the model, Neural Network has the highest percentage of (49.34%), followed by Naïve Bayes (47.58%) and Decision Trees (41.85%). If the entire population is processed, Naïve Bayes model appears to perform better than the other two as it gives the highest number of correct predictions (86.12%) followed by Neural Network (85.68%) and Decision Trees (80.4%).

The Classification Matrix displays the frequency of correct and incorrect predictions. It compares the actual values in the test dataset with the predicted values in the trained model. Naïve Bayes appears to be most effective as it has the highest percentage of correct predictions (86.53%) for patients with heart disease, followed by Neural Network (with a difference of less than 1%) and Decision Trees. Decision Trees, however, appears to be most effective for predicting patients with no heart disease (89%) compared to the other two models.

Section 4. Results and Conclusion:

IHDPS can be used to serve as a training tool to train nurses and medical students to diagnose patients with heart disease. It can also help in supporting decisions to assist doctors, in making better clinical decisions. Having a larger dataset would definitely give better results. It is also necessary to test the system extensively with input from cardiologists, before it is ready to be deployed in hospitals.

The most effective model to predict patients with heart disease appears to be Naïve Bayes, followed by Neural Network and Decision Trees.

Experiment Replication:

Python Scikit-learn was used to replicate the experiment carried out by the researchers. Data was scaled using MinMaxScaler function. The training and testing dataset were split in a 50:50 ratio, which was done in the same way as the research paper.

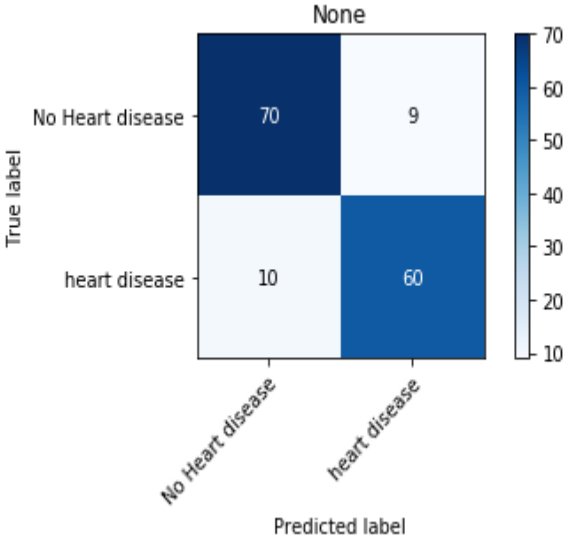
The classifiers Decision Tree, Naïve Bayes and Neural Network models were used to train on the training dataset.

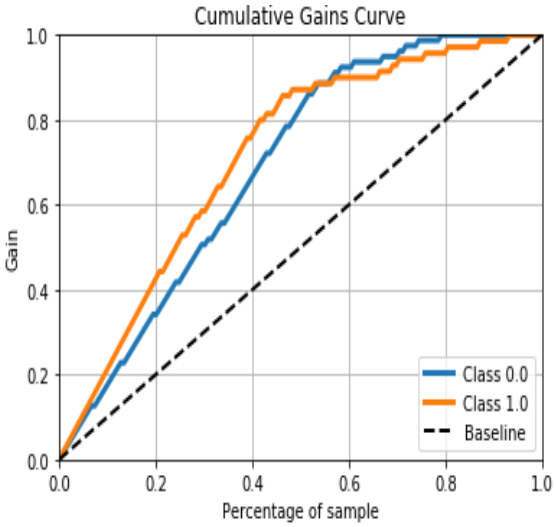
Results from Neural Network:

```
Neural Network : 0.87248322147651
             precision    recall  f1-score   support

        0.0       0.88      0.89      0.88        79
        1.0       0.87      0.86      0.86        70

avg / total       0.87      0.87      0.87       149
```





Neural Network model gives a good accuracy of 87%. Neural network model seems to outperform both Naïve Bayes and Decision tree and is the best model for this replication experiment. Precision of 'heartdisease' instance is almost the same as 'noheartdisease' instance.

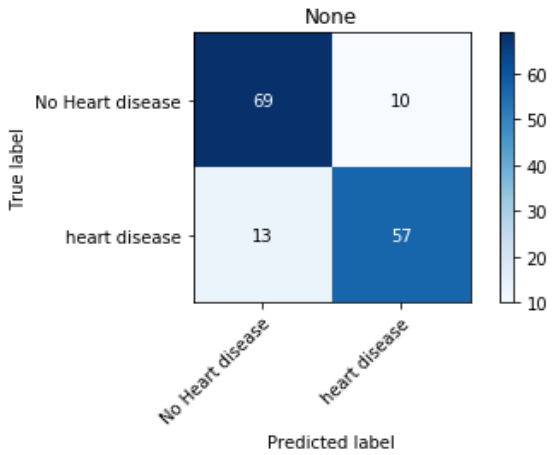
On comparing with the research carried out by the authors, Neural Network of this experiment is performing better than the best model of their paper, which is Naïve Bayes with an accuracy of 86%. On another note, the Neural Network model carried out by the authors gave an accuracy of only 85.6%.

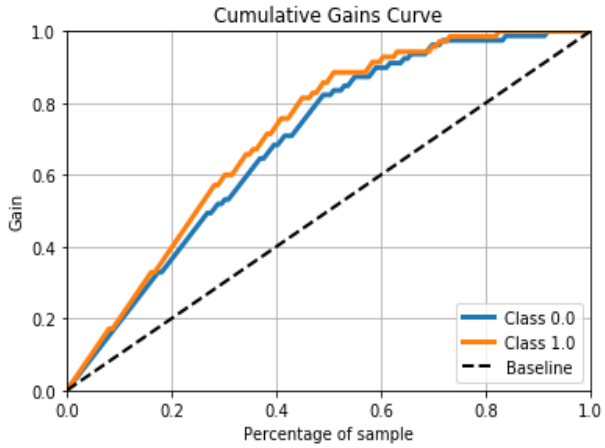
Results from Naïve Bayes:

```
Naive Bayes : 0.8456375838926175
              precision    recall  f1-score   support

         0.0       0.84      0.87      0.86        79
         1.0       0.85      0.81      0.83        70

avg / total       0.85      0.85      0.85       149
```





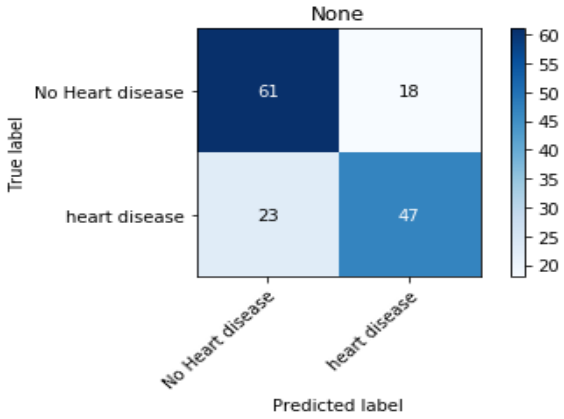
Naïve Bayes in this replication experiment, is the second best in terms of performance, with an accuracy of 84%. Whereas, Naïve Bayes was the best performing model in the research paper carried out by the authors, with an accuracy of 86.12%.

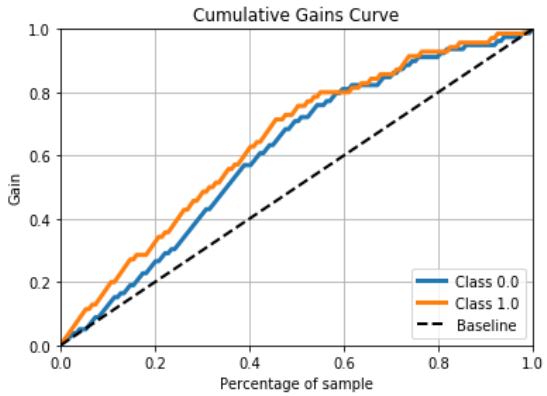
Result obtained by Decision Tree:

```
Decision Tree : 0.7248322147651006
              precision    recall  f1-score   support

         0.0       0.73      0.77      0.75        79
         1.0       0.72      0.67      0.70        70

avg / total       0.72      0.72      0.72       149
```





Decision Tree returns the least accuracy with 72.4% only. Decision tree is the least performing model of both this replication experiment and in the research carried out by the authors, where the decision tree done in the research paper performed better than this model, with an accuracy of 80.4%.

Discussion:

As we saw in the evaluation section, the results from this experiment differs in a small manner, from the work carried out in the research. The best performing model of this experiment is Neural Network, whereas the best performing model in the research paper is Naïve Bayes. The similarities found is that, the worst performing model is Decision Tree, in both the experiments.

In conclusion, it is safe to say that the result obtained in this replication experiment via Neural Network model (Accuracy = 87%) performs better than the best performing model of the research paper (Accuracy = 86.12%). It could also be because of all the advancements and contributions towards Scikit learn's packages, compared to the libraries available in .NET.

Appendix:

```
#Import statements
import pandas as pd
import numpy as np
import gc
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import scikitplot as skplt

#Read the data
heart = pd.read_csv('cleveland.data', sep=',', na_values='?')
heart[0:5]
heart=heart.dropna()
heart['num'].replace({2:1},inplace = True)
heart['num'].replace({3:1},inplace = True)
heart['num'].replace({4: 1},inplace = True)
heart['num'].replace({0.25:1},inplace = True)
heart['num'].replace({0.5:1},inplace = True)
heart['num'].replace({0.75:1},inplace = True)

#Scale the data
scaler = preprocessing.MinMaxScaler()
scaler.fit(heart)
heart=pd.DataFrame(scaler.transform(heart),index=heart.index,columns=heart.columns)

#Splitting dataset in a 50:50 ratio
heart_train, heart_test, target_train, target_test = train_test_split(heart.drop('num',axis=1), heart['num'],test_size=0.5)

#creating the definition for confusion matrix
def plot_confusion_matrix(y_true, y_pred, classes, normalize=False, title=None, cmap=plt.cm.Blues):
```

```
    cm = confusion_matrix(y_true, y_pred)
    fig, ax = plt.subplots()
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
    ax.set(xticks=np.arange(cm.shape[1]),       yticks=np.arange(cm.shape[0]),       xticklabels=classes,
yticklabels=classes, title=title, ylabel='True label',
           xlabel='Predicted label')
            plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
            rotation_mode="anchor")
            fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center",
                    color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    return ax
np.set_printoptions(precision=2)

#Data modelling
classifiers = [['Decision Tree :', DecisionTreeClassifier()],
               ['Naive Bayes :', GaussianNB()],
               ['Neural Network :', MLPClassifier()]]

predictions_df = pd.DataFrame()
predictions_df['actual_labels'] = target_test

for name, classifier in classifiers:
    classifier = classifier
    classifier.fit(heart_train, target_train)
    predictions = classifier.predict(heart_test)
    predicted_probas = classifier.predict_proba(heart_test)
    predictions_df[name.strip(" :")] = predictions
    print(name, accuracy_score(target_test, predictions))
    print(classification_report(target_test, predictions))
```

```
plot_confusion_matrix(target_test, predictions, ['No Heart disease','heart disease'])

skplt.metrics.plot_cumulative_gain(target_test, predicted_probas)

plt.show()
```

References:

[1] Palaniappan, S., & Awang, R. (2008, March). Intelligent heart disease prediction system using data mining techniques. In 2008 IEEE/ACS international conference on computer systems and applications (pp. 108-115). IEEE.