**Ain Shams University**

**Faculty of Engineering**

**Computer Engineering and Software Systems Program**

**CSE385: Data Mining and Business Intelligence**

**Project**

Submitted by:

Ahmed Mohamed Ahmed Elqaffas 16P6012

Ahmed Mamdouh Mohammed 16P6020

Ahmed Elnokrashi Ibrahim 14P8144

Nada Tarek Ahmed Abdalla 16P6053

**Submitted to:**

Dr. Mahmoud Mounir Mahmoud Ali

Eng. Ahmed Hesham

# 1.0 DESCRIPTION

## 1.1 Preprocessing

We created a preprocessing class that contains multiple functions to clean and process the data before using our algorithms.

### 1.1.1 Split data

The split\_data method splits the data set into training set to train the model, and testing set to check the model accuracy. The splitting ratio is a parameter of the method.

### 1.1.2 Scaling

The scaling method applies standard scaler, min max scaler, max scaler, robust scaler or min scaler based on a parameter (scale\_type) using sklearn library.

### 1.1.3 Encoding

The encoding method encodes the categorical data to numerical data so that it can be used in any of the training algorithms. The “diamonds” and the “breast cancer” datasets had categorical data which needed to be encoded. In the “diamonds” dataset: the “cut”, “color” and “clarity” features must be encoded. In the “breast cancer” dataset: the “diagnosis” column (which is the dependent variable) must be encoded, malignant gets encoded to 1 and benign gets encoded to zero.

### 1.1.4 Missing Data

The “drop\_missing” method drops entries (rows) using pandas if the row contains a missing attribute.

We’ve also implemented a “replace\_missing” method that replaces the missing attribute with the mode of that attribute instead of dropping the whole row.

### 1.1.5 DataFrame to Numpy

In addition, we have a “dataframe\_to\_numpy” method that takes the dataset and splits it into two numpy arrays, one for the features and one for the output labels. Also, this method excludes unnecessary data from the numpy arrays like the “ID” feature in the “breast cancer” dataset or the index column. This method does not get applied on the “Iris” dataset because the “Iris” dataset is already loaded as a numpy array using the load\_iris function in sklearn.

## 1.2 Clustering

The clustering class is used to implement the K-means clustering algorithm. The class has two main methods. The class is used to cluster the iris data set as an example.

### 1.2.1 Cluster

The “cluster” method trains the model using the training set and then clusters (predicts) the testing set. The input parameters of the method are the training set, the test set and the number of clusters. The returned output is an array of predicted clusters for each test set entry.

### 1.2.2 Tune parameters

The “tune\_parameters” method is used to determine the ideal K (clusters number) for the K-means algorithm. The method runs the training algorithm with multiple iterations and chooses different values for the number of clusters on each iteration and calculates the inertia each time.

The parameter value of the model with the lowest inertia is then used in training the model.

## 1.3 Classifier

The classifier class contains multiple classification algorithms, the user’s input determines which algorithm to use. The classifier is used on the breast cancer dataset as an example.

### 1.3.1 classify

This function contains the classification algorithms that we implemented based on the user input it applies one of the following algorithms on the breast cancer dataset

* K Nearest Neighbors (KNN)
* Decision Trees
* Naïve Bayes classifier
* Random Forest
* Neural Network classification

We’ve studied all these classifiers in the course except the random forest and neural network, so, we are going to explain them.

**Random Forest Classifier:**

Random Forest Classifier is based on the decision tree classifier. The random forest is a classification algorithm consisting of many decision trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. In the Random Forests algorithm, each new data point goes through the same process, but now it visits all the different trees in the ensemble, which are were grown using random samples of both training data and features. Depending on the task at hand, the functions used for aggregation will differ. For Classification problems, it uses the mode or most frequent class predicted by the individual trees (also known as a majority vote).

**Neural Network Classifier:**

Neural Network consists of an input layer, hidden layers and an output layer Each layer contains a certain number of blocks called neurons which make calculations and then the neurons forward the results to the next layer until the output layer is reached which outputs the predicted class. The leftmost layer, known as the input layer, consists of a set of neurons representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation, followed by a non-linear activation function like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values. This classifier produces the highest accuracy possible.

### 1.3.2 Accuracy

“get\_accuracy” method returns the accuracy of the model by applying the model on the test set. The method compares the predicted class with the actual class for each test set data entry and then calculates the number of correct classifications and divides it by the total number of entries in the test data set. The result is then multiplied by 100 so that it can be in percentage form before it gets returned.

### 1.3.3 Tune parameters

“tune\_NN\_parameters” method runs the neural network training algorithm with multiple iterations and chooses different values for the hidden layer sizes on each iteration and calculates the accuracy each time. The parameter value of the model with the highest accuracy is returned.

## 1.4 Regressor

Like the classifier, the regressor class contains multiple regression algorithms. They are applied on the “diamonds” dataset.

### 1.4.1 Predict

This function builds the model and uses it to predict the values of the testing set. The user chooses the algorithm that should be used, the algorithms are

* KNN Regression
* Decision Tress
* Linear Regression
* Polynomial Regression
* Random Forest Regression
* Neural Network Regression

### 1.4.2 Score

“get\_score” method returns the coefficient of determination of the model. One is the best possible score for any regression model so the closer the model score is to 1, the better.

## 1.5 Visualizer

The visualizer class visualizes the data given or produced so the inputs and results can be understood clearly. This class contains two methods, one for plotting the data of regression and classification and the other is for plotting data related to clustering.

### 1.5.1 Plotting Regression and Classification Results

“plot\_classifier\_regressor” method is used to plot the data related to the classification or regression operations. It’s used to visualize the classifier and the regressor outputs. In the case of classification, a histogram figure of the number of actual patients with malignant tumors and number of actual patients with benign tumors is plotted and then another histogram figure is plotted for the predicted number of patients with malignant tumors and predicted number of patients with benign tumors. In the case of regression, a scatter plot is plotted for the actual price of the diamond against the predicted price of the diamond. In a perfect regression model, there should be a line with the slope of 1 fitting the datapoints in the scatter plot.

### 1.5.2 Plotting Clustering Results

“plot\_clustering” method plots two diagrams of the iris dataset (one before and one after) clustering. Principle Component Analysis algorithm was used to reduce the dimensions of the Iris dataset into two dimensions so that the iris dataset becomes plottable. The number of clusters is assumed to be up to 4.

# 2.0 Flow of Code and Results

## 2.1 Flow of Code

1- The user chooses classification, regression or clustering.

2- The appropriate dataset is loaded for the chosen operation.

3- The user chooses the desired algorithm to apply on the dataset.

4- The rows with missing data are dropped.

5- Dataset is cleaned and encoded in the case of classification and regression.

6- Splitting the dataset into a training set and a testing set.

7- Scaling the dataset.

8- Applying the training algorithm and predicting the results.

9- Calculating the accuracy for classification, score for regression or inertia for clustering

10- Visualizing the actual data versus the results.

## 2.2 Results

## Some results of the program are shown in this section. The results are all applied on a testing set where the train/test split ratio was 0.19 .

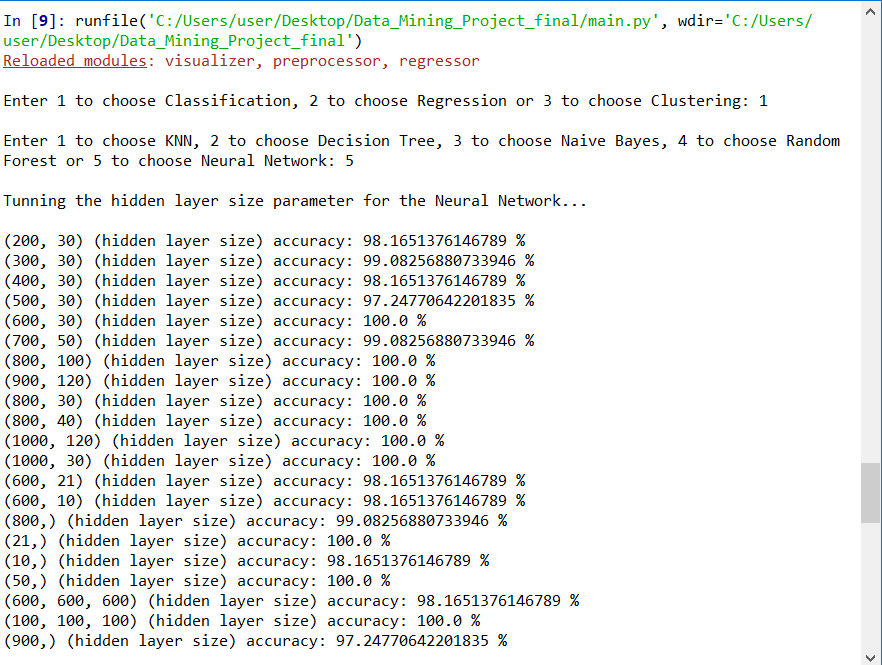
## 2.2.1 Classification Results

## We’ve decided to show the results of the “**Neural Network Classifier”** and the

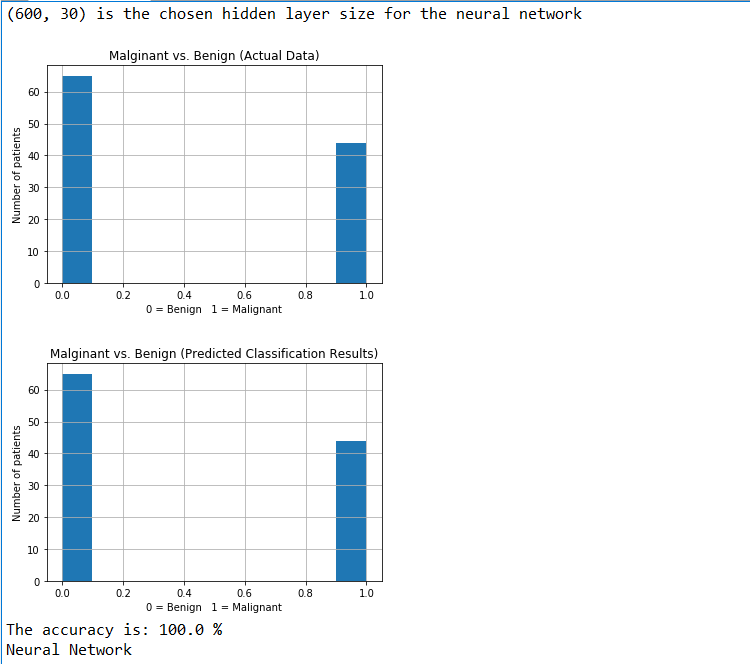
## **“Random Forest Classifier”** as they had the most impressive results.

**Neural Network Classifier Results:**

Tuning the “hidden layers sizes” parameter:

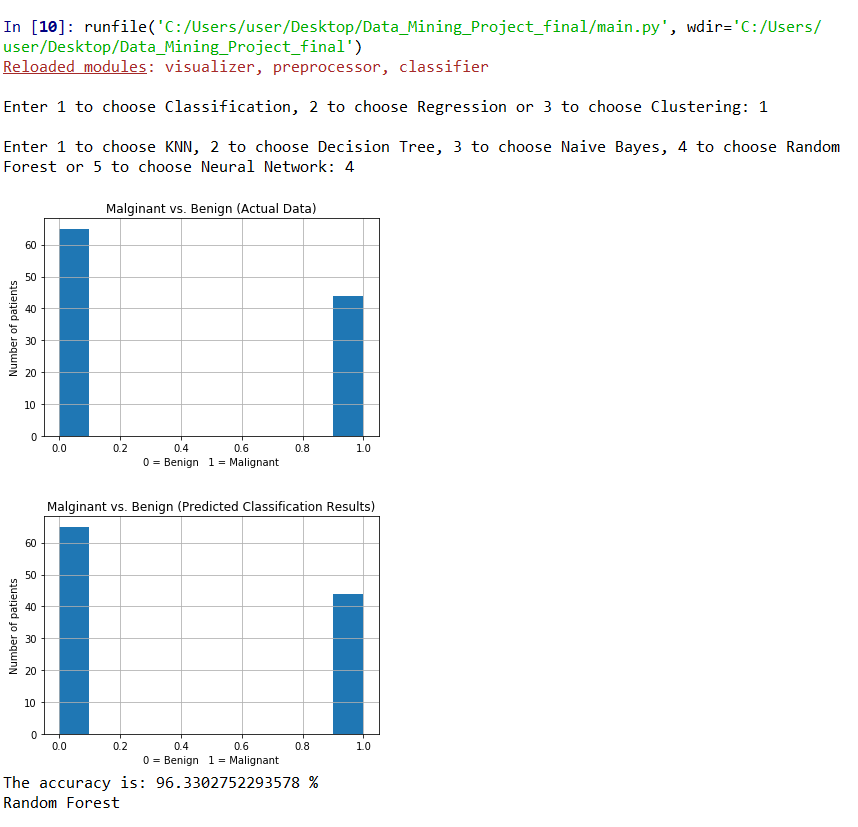


Predicted results:



As we can see, the Neural Network produces the highest accuracy possible (100 %).

**Random Forest Classifier Results:**

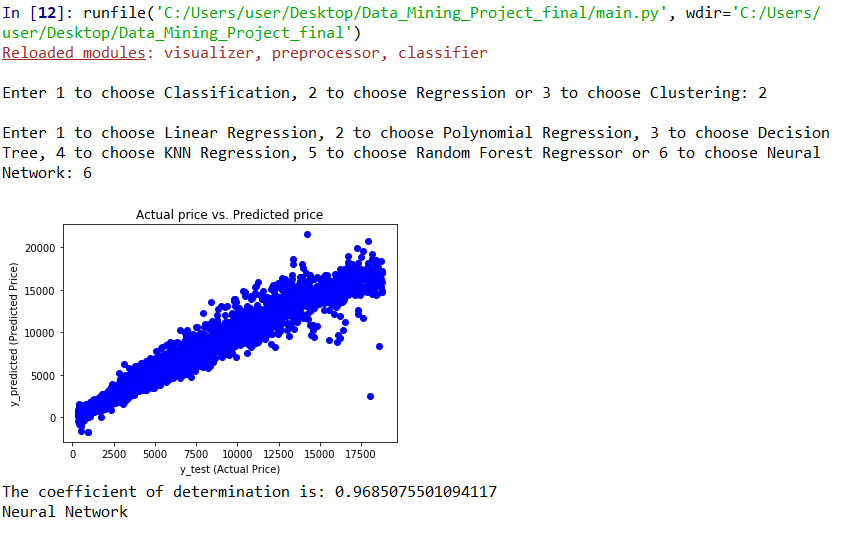


## 2.2.2 Regression Results

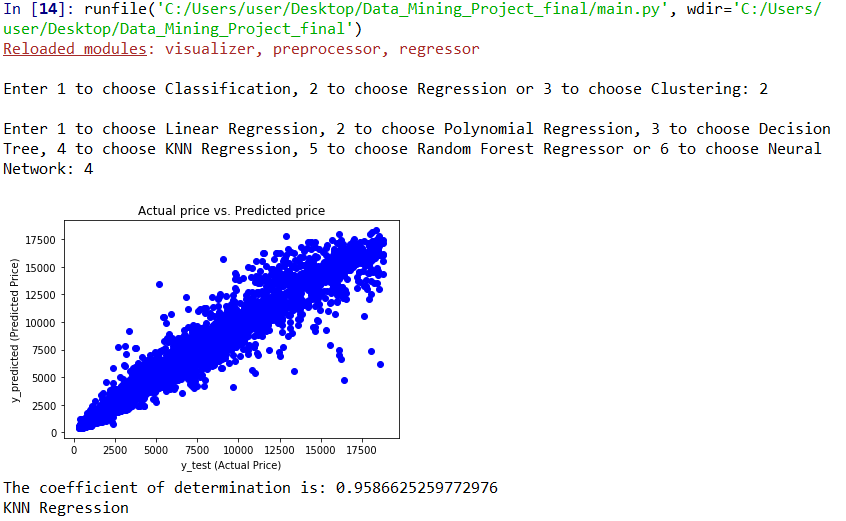
## We’ve decided to show the results of the “**Neural Network Regressor”** and the

## **“K-Nearest Neighbors Regressor”** as they had the most impressive results. The results are all applied on a testing set where the train/test split ratio was 0.19

**Neural Network Regressor Results:**



**K-Nearest Neighbors Regressor Results:**



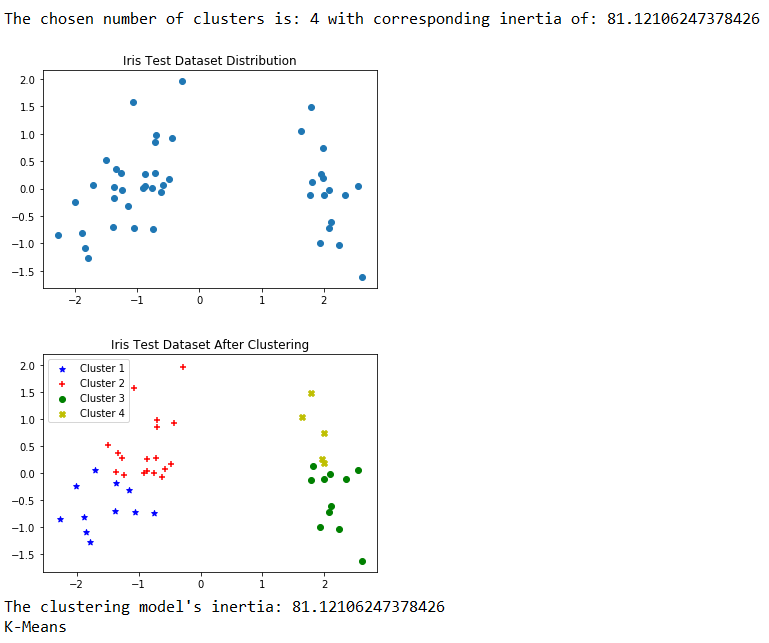
## 2.2.3 Clustering Results

## The **“K-Means”** algorithm results are shown in this section. The results are all applied on a testing set where the train/test split ratio was 0.19.

## **Tuning the “number of clusters” parameter:**

## 

**The predicted clusters:**



# 3.0 Source Code

## 3.1 “Main.py” File

import numpy as np

import pandas as pd

from visualizer import Visualizer

# ------------Loading the data-------------------

# 1 = classification, 2 = regression and any other number = clustering

method\_identifier = int(input('Enter 1 to choose Classification, 2 to choose Regression' +

' or 3 to choose Clustering: '))

if method\_identifier == 1:

dataset = pd.read\_csv('wisconsin\_breast\_cancer.csv')

elif method\_identifier == 2:

dataset = pd.read\_csv('diamonds.csv')

elif method\_identifier == 3:

from sklearn.datasets import load\_iris

X = load\_iris(return\_X\_y=False)['data']

iris = load\_iris()

# diamonds.csv on kaggle.com for regression

# wisonsin breast cancer.cvs on kaggle.com for classification

# iris.csv for clustering (drop class column)

# --------------Choosing the desired algorithm--------------------

# Choosing classification algorithm

if method\_identifier == 1:

identifier = int(input('Enter 1 to choose KNN, 2 to choose Decision Tree, 3 to choose Naive Bayes' +

', 4 to choose Random Forest or 5 to choose Neural Network: '))

if identifier == 1:

algorithm\_name = 'KNN'

elif identifier == 2:

algorithm\_name = 'Decision Tree'

elif identifier == 3:

algorithm\_name = 'Naive Bayes'

elif identifier == 4:

algorithm\_name = 'Random Forest'

else:

algorithm\_name = 'Neural Network'

# Choosing Regression algorithm

elif method\_identifier == 2:

identifier = int(input('Enter 1 to choose Linear Regression, 2 to choose Polynomial Regression,' +

' 3 to choose Decision Tree, 4 to choose KNN Regression, 5 to choose' +

' Random Forest Regressor or 6 to choose Neural Network: '))

if identifier == 1:

algorithm\_name = 'Linear Regression'

elif identifier == 2:

algorithm\_name = 'Polynomial Regression'

elif identifier == 3:

algorithm\_name = 'Decision Tree'

elif identifier == 4:

algorithm\_name = 'KNN Regression'

elif identifier == 5:

algorithm\_name = 'Random Forest'

else:

algorithm\_name = 'Neural Network'

# Choosing Clustering algorithm

else:

algorithm\_name = 'K-Means'

# ---------------------Preprocessing the data---------------------------------

from preprocessor import Preprocessor

preprocess = Preprocessor()

# Cleaning the data

if method\_identifier == 3: # Drop missing rows in iris dataset

X = preprocess.drop\_missing(X)

else: # Drop missing rows in diamond or wisconsin breast cancer datasets

dataset = preprocess.drop\_missing(dataset)

if method\_identifier == 1:

X, y = preprocess.dataframe\_to\_numpy(dataset, 'breast cancer')

X, y = preprocess.encoding(X, y, 'breast cancer')

elif method\_identifier == 2:

X, y = preprocess.dataframe\_to\_numpy(dataset, 'diamonds')

X, y = preprocess.encoding(X, y, 'diamonds')

# Splitting the data into train and test sets

if method\_identifier == 1 or method\_identifier == 2:

X\_train, X\_test, y\_train, y\_test = preprocess.split\_data(X, 0.19, y)

elif method\_identifier == 3:

X\_train, X\_test = preprocess.split\_data(X, test\_ratio=0.3)

# Scaling the data

X\_train, X\_test = preprocess.scaling(X\_train, X\_test, scale\_type='Standard Scaler')

# ----------------------------Classifying the data----------------------------

if method\_identifier == 1:

from classifier import Classifier

classifier = Classifier(algorithm\_name)

y\_predicted = classifier.classify(X\_train, y\_train, X\_test, y\_test)

classifier\_accuracy = classifier.get\_accuracy(y\_test, y\_predicted)

# Visualizing the results

visualizer = Visualizer()

visualizer.plot\_classifier\_regressor(y\_test, y\_predicted, method\_identifier)

print('The accuracy is: ' + str(classifier\_accuracy) + ' %')

print(algorithm\_name)

# ---------------------Applying Regression to the data--------------------------

elif method\_identifier == 2:

from regressor import Regressor

regressor = Regressor(algorithm\_name)

y\_predicted = regressor.predict(X\_train, y\_train, X\_test)

regressor\_score = regressor.get\_score(y\_test, y\_predicted)

# Visualizing the results

visualizer = Visualizer()

visualizer.plot\_classifier\_regressor(y\_test, y\_predicted, method\_identifier)

print('The coefficient of determination is: ' + str(regressor\_score))

print(algorithm\_name)

# ---------------------Clustering the data------------------------------------

elif method\_identifier == 3:

from clustering import Clustering

clustering = Clustering(algorithm\_name)

n\_clusters, inertia = clustering.tune\_parameters(X\_train)

clusters = clustering.cluster(X\_train, X\_test, n\_clusters)

# Visualizing the results

visualizer = Visualizer()

visualizer.plot\_clustering(X\_test, clusters)

print("The clustering model's inertia: " + str(inertia))

print(str(algorithm\_name))

## 3.2 “Preprocessor.py” File