

# McMaster University

IBEHS 4QZ3

Modeling of Biological Systems

Fall 2022

---

## Analysis Project

A Comparative Analysis of Logistic Regression and Multilayer  
Perceptrons to Predict the Onset of Diabetes Mellitus

---

December 8, 2022

**Authors:**

Justin Prez,	400148692
Janek Wolos,	400131936
Fadia Cruz,	400135025

**Instructor:**

Taylor De Vet



# Table of Contents

<b>1 Introduction</b>	<b>3</b>
1.1 Diabetes Mellitus	3
1.2 Diabetes - Cures and Treatment	3
1.3 Predicting the Onset of Diabetes	3
<b>2 Methods</b>	<b>4</b>
2.1 Data Acquisition	4
2.2 Data Processing	5
2.2 Logistic Regression	5
2.3 Neural Network	6
2.4 Statistical Analysis	7
<b>3 Results</b>	<b>8</b>
3.1 Logistic Regression Results	8
Results on Test Set	10
Results on Training Set	11
3.2 Neural Network Results	12
5-Layer Neural Network - Results on Test Set	12
Results on Training Set	13
3.3 Summary of Results	14
<b>4 Discussion</b>	<b>15</b>
<b>5 Conclusion</b>	<b>15</b>
<b>References</b>	<b>16</b>
<b>Appendix</b>	<b>18</b>

# 1 Introduction

## 1.1 Diabetes Mellitus

Diabetes mellitus is a group of chronic medical conditions characterized by high glucose levels in the bloodstream. More than 10% of the population in Canada and the United States have diabetes, and 1 in 5 are unaware they have it [1,2]. Insulin, a hormone secreted by the pancreas, regulates glucose levels in the blood. When there is a defect in insulin production, insulin action, or both, blood glucose levels become too high, resulting in hyperglycemia. Over time, this can cause serious health complications, including heart disease, stroke, nerve damage, kidney impairment, vision loss, and foot problems [3].

There are three main types of diabetes mellitus. Type 1 is an autoimmune disease in which the body's immune system erroneously attacks the insulin-producing cells in the pancreas. Type 1 accounts for 5-10% of all diagnosed diabetes cases. Type 2 diabetes is a metabolic condition in which the body gradually becomes resistant to insulin, leading to higher blood glucose levels. Type 2 accounts for 90-95% of all diagnosed cases. Gestational diabetes is a condition characterized by hyperglycemia in women during pregnancy. It occurs in 2-10% of all pregnancies and can increase the risk of developing Type 2 diabetes following childbirth [1].

## 1.2 Diabetes - Cures and Treatment

There is currently no cure for diabetes. However, with early diagnosis and a proper treatment plan, diabetes is manageable, and people can continue living healthy and active lives. Treatment depends on the type of diabetes. For type 1 diabetes, the primary treatment is insulin therapy. Insulin is either injected or taken through an insulin pump. For type 2 diabetes, lifestyle changes, such as exercise and diet, are usually the first line of defence. Some individuals may require medication like insulin. For those with gestational diabetes, lifestyle changes are usually recommended, and medications such as insulin are used to help control blood sugar levels [4].

## 1.3 Predicting the Onset of Diabetes

In this project, we investigate the Pima Indians diabetes database, a collection of medical information gathered from the Pima Indians of Arizona, USA. This population has been under continuous study since 1965 by the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK) as they have been disproportionately affected by Type 2 diabetes [5,6,7]. The data includes measures of blood glucose, insulin, body mass index, age, and diabetes status for each patient. The Pima Indians dataset can provide insights into the relationship between the risk factors for diabetes and the prevalence of the disease in this population. This dataset provides a well-validated data resource that we can use to understand the onset of diabetes, leading to better outcomes in prevention and treatment in this population. Diagnosing diabetes early can reduce the risk of diabetes-related health complications and save on medical costs. We compare the use of two supervised machine learning algorithms, logistic regression and neural networks, to predict the onset of Type 2 diabetes within the Pima Indian sample population. The performance of each binary classification algorithm is analyzed to determine the one with the best accuracy, precision, recall and receiver operating characteristic (ROC) curves.

## 2 Methods

### 2.1 Data Acquisition

The Pima Indians Diabetes Database was acquired from Kaggle at [8]. It is available for use under the CC0: Public Domain license. The observations of this dataset were initially acquired through a series of clinical studies conducted by the NIDDK on the Pima Indians near Phoenix, Arizona. Several constraints were placed on the selection of cases from the original dataset to create a homogenous sample. In particular, all subjects are female, at least 21 years old, and of Pima Indian heritage. Only one examination is included per subject, meaning each observation is independent. The exact criteria used for case selection are detailed in [9]. This database contains 768 examinations. Each data instance corresponds to the results of a standardized examination, including an oral glucose tolerance test (GTT). Diabetes was diagnosed if plasma glucose concentration was greater than 200 mg/dl two hours after ingesting 75 g of a carbohydrate solution [9,10]. Eight medical predictor variables are provided for each data instance. [9] suggests that these variables are significant risk factors for diabetes among Pimas or other populations. A summary of the data is shown in Tables 1 and 2.

Variable Description	Data Type	Range
Number of Pregnancies	Numeric	[0, 17]
Plasma Glucose Concentration at 2 Hours in an Oral GTT	Numeric	[0, 199]
Diastolic Blood Pressure (mm Hg)	Numeric	[0, 122]
Triceps Skin Fold Thickness (mm)	Numeric	[0, 99]
2-Hour Serum Insulin ( $\mu$ U/mL)	Numeric	[0, 846]
Body Mass Index (Weight in kg / (Height in m <sup>2</sup> ))	Numeric	[0, 67.1]
Diabetes Pedigree Function - described in [9]	Numeric	[0.078, 2.42]
Age (years)	Numeric	[21, 81]
Outcome - Dichotomous response variable indicating Non-Diabetic / Diabetic	Binary	(0, 1)

*Table 1. Overview of the Pima Indian diabetes dataset variables.*

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

*Table 2. Statistical summary of the Pima Indians diabetes dataset.*

## 2.2 Data Processing

The input features were normalized to reduce the data variance and improve model accuracy. Normalization involves subtracting the mean from each feature and dividing by the standard deviation. The mean for each input parameter becomes 0, and the variance is rescaled to 1. Table 3 shows the statistical summary of the data after normalization. Normalizing the input features ensures the models are not overly influenced by a single feature with a higher magnitude and reduces the chances of overfitting.

The sample population was manually split into two groups with equal stratification of diagnosed and undiagnosed individuals (using the `train_test_split()` in scikit-learn). 75% of the samples were used for training the predictive models. The remaining 25% of the samples were used for testing.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02
mean	2.544261e-17	3.614007e-18	-1.327244e-17	7.994184e-17	-3.556183e-17	2.295979e-16	2.462585e-16	1.857600e-16
std	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00
min	-1.141852e+00	-3.783654e+00	-3.572597e+00	-1.288212e+00	-6.928906e-01	-4.060474e+00	-1.189553e+00	-1.041549e+00
25%	-8.448851e-01	-6.852363e-01	-3.673367e-01	-1.288212e+00	-6.928906e-01	-5.955785e-01	-6.889685e-01	-7.862862e-01
50%	-2.509521e-01	-1.218877e-01	1.496408e-01	1.545332e-01	-4.280622e-01	9.419788e-04	-3.001282e-01	-3.608474e-01
75%	6.399473e-01	6.057709e-01	5.632228e-01	7.190857e-01	4.120079e-01	5.847705e-01	4.662269e-01	6.602056e-01
max	3.906578e+00	2.444478e+00	2.734528e+00	4.921866e+00	6.652839e+00	4.455807e+00	5.883565e+00	4.063716e+00

Table 3. Statistical summary of the Pima Indians diabetes dataset after normalizing input features.

## 2.2 Logistic Regression

Binary logistic regression estimates the probability of a dichotomous outcome given the values of the input parameters. In this approach, we develop a logistic regression model that estimates the probability that an individual will develop diabetes mellitus. Let  $w = (w_0, w_1, \dots, w_8)^T$  denote the vector of parameters estimated via training with  $w_0$  representing the bias or intercept. Additionally, let  $\bar{x} = (1, x_0, \dots, x_8)$  represent the feature vector for the eight input variables with an additional dummy feature. Logistic regression uses the logistic or sigmoid function to transform any real input into a value between 0 and 1. The logistic function for  $y$  is given by:

$$y(w^T \bar{x}) = \frac{1}{1 + e^{-w^T \bar{x}}}$$

Here,  $y$  represents the predicted probability that an individual has diabetes (Outcome variable is 1). Training in logistic regression solves for the minimization of the cost function  $C(w)$ , given as:

$$C(w) = \frac{1}{N} \sum_{i=1}^N \ell_{ce}(y^{(i)}, t^{(i)})$$

Where  $\ell_{ce}(y, t)$  is the logistic-cross-entropy loss function, and  $t \in (0,1)$  is the known outcome (ground truth). The loss function expanded is:

$$\ell_{ce}(y, t) = -t \log(y) - (1 - t) \log(1 - y)$$

To minimize  $C(w)$ , we solve  $\nabla C(w) = 0$  using the Gradient Descent method. With each iteration in gradient descent, the weights are updated for the  $j$ th iteration as:

$$w^{(j)} = w^{(j-1)} - \alpha \nabla C(w^{(j-1)}), \quad \text{where } \alpha > 0 \text{ is the learning rate}$$

The weights are updated on each iteration until the solution converges or the stop criterion is met. We can then use the updated vector of parameters  $w$  to make predictions with the test samples.

Logistic regression requires little to no multicollinearity among predictor variables. To assess this assumption, we will compute the pairwise correlation and pair plots between independent variables. Strongly influential outliers should also be accounted for in logistic regression. We assess the presence of outliers graphically using box plots.

## 2.3 Neural Network

A neural network (NN) is a machine learning algorithm that emulates how neurons work in the brain. A NN consists of a group of neuron-like processing units organized as nodes in a directed graph. Each neuron receives a set of inputs multiplied by weights, which are then passed through a non-linear activation function. Common activation functions include the sigmoid, hyperbolic tangent and rectified linear unit (ReLU) functions. The neuron's output is then passed to the next layer of neurons in the network. In this supervised learning approach, we employ a multi-layer perceptron (MLP). An MLP is a feedforward neural network (FNN) with no feedback connections and fully connected layers.

Training the NN for binary classification follows a similar approach to logistic regression. The setup and training are as follows:

1. Choose a loss function that evaluates the error between the correct label  $t$  and the model predicted output  $y$ . We use the same logistic-cross-entropy loss function  $\ell_{ce}(y, t)$ .
2. The cost of the average loss over all training samples is given as:

$$C(w) = \frac{1}{N} \sum_{i=1}^N \ell_{ce}(y^{(i)}, t^{(i)})$$

3. Define the activation function to use. We have opted for the ReLU function (the default recommendation for the MLPClassifier() in scikit-learn).
4. Define the number of layers and the size of each layer. We trained with 3-, 5-, and 7-layer NN models to assess the model for underfitting and overfitting. (3- and 7- were not included in the Results of this report due to space constraints but were ultimately ruled out as weaker fits).
5. The goal of training is to minimize  $C(w)$ , so we solve  $\nabla C(w) = 0$  using the Gradient Descent method. The weights and biases are updated on each iteration until the solution converges and/or the stop criterion is met.

Once the training step is completed, the updated matrix of weights and biases can be used to make predictions with the test samples. The neural networks do not make assumptions about the data being analyzed. Therefore this model is not validated against any assumptions.

## 2.4 Statistical Analysis

Accuracy, precision, recall, F1 score, and receiver operator characteristic (ROC) curves are used to evaluate each model's predictive capabilities with the test set. We can obtain these metrics for the logistic regression and neural network models to compare their performance.

Accuracy is the number of correctly classified data instances over the total number of classifications. Precision, also known as the true positive rate, is the likelihood that a positive detection is a true positive. Recall, also known as sensitivity, is the model's ability to capture all positive tests. The F1 score is a harmonic mean of precision and recall. The formula for each of these binary classification statistics is included below. (TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative)

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \quad Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN} \quad F1 \text{ Score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The ROC curve is a graph that displays the performance of a binary prediction model by plotting the true positive rate vs. the false positive rate at all threshold values. The threshold is a value between 0 and 1 that sets the boundary for class allocation. The area under the ROC curve (AUC) is an aggregate performance metric for all the possible classification thresholds.

### 3 Results

#### 3.1 Logistic Regression Results

As shown in Figure 1, there are outliers in every feature, which may be due to underlying factors. We attempt to diminish the outliers' effect by transforming the data into a common scale via normalization. As the dataset is not very large, we avoid unnecessarily removing any data instances. The pairwise correlation ( $r$ ) and pair plots between independent variables are shown in Table 4 and Figure 2, respectively. We can see that the most closely correlated features include pregnancy and age ( $r > 0.5$ ). In this project, we opted to proceed with all input features present in both models for the comparative analysis, but it is noted that future models could be improved by removing highly correlated input features.

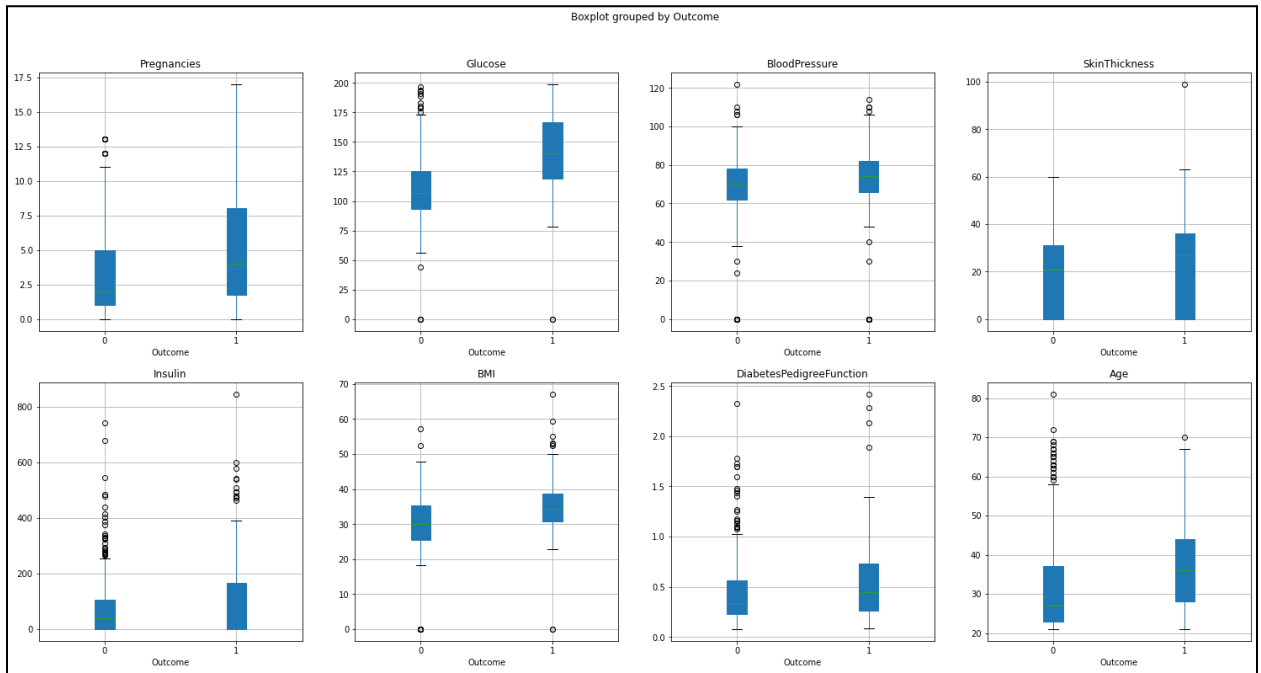
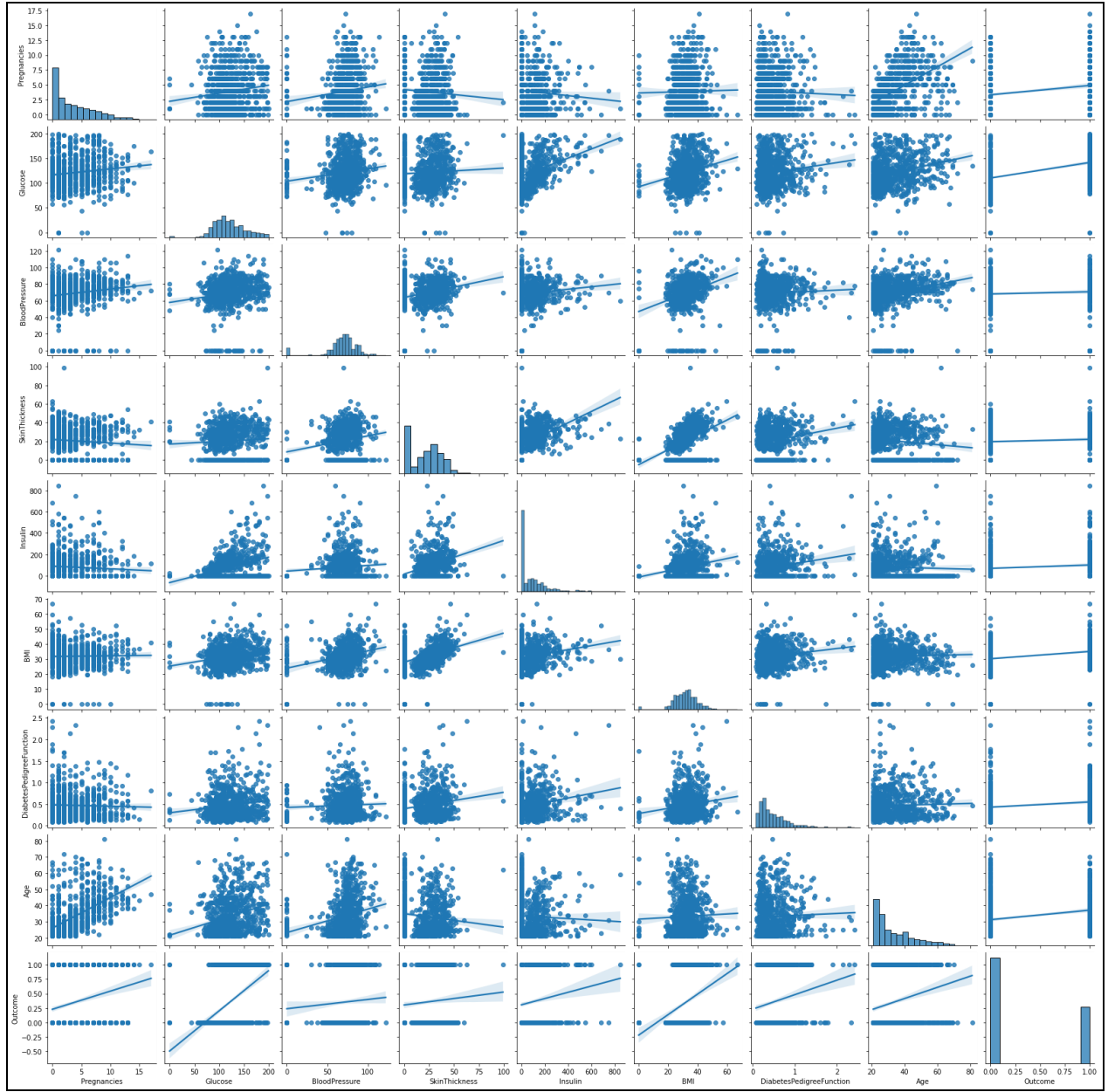


Figure 1. Box plots showing the input distribution for each outcome.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

Table 4. Correlation matrix between independent variables.





*Figure 2. Correlation pair plots of all variables used.*

## Results on Test Set

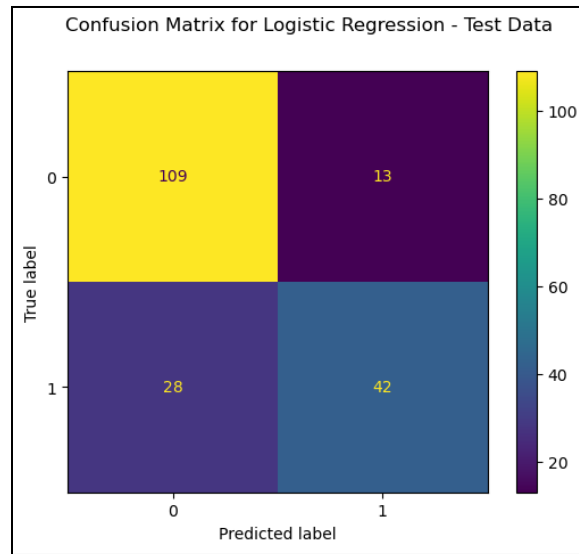


Figure 3. Confusion Matrix of Logistic Regression Model on Test Data.

The Confusion Matrix of the test data predictions using logistic regression gave the following values:

**True Negatives:** 109, **True Positives:** 42, **False Negatives:** 28, **False Positives:** 13

*****Classification Report On Test Set*****				
	precision	recall	f1-score	support
0	0.80	0.89	0.84	122
1	0.76	0.60	0.67	70
accuracy			0.79	192
macro avg	0.78	0.75	0.76	192
weighted avg	0.78	0.79	0.78	192

Figure 4. Classification Report of Logistic Regression on Test Data.

The Classification Report gives:

Average **Precision** of 0.78, an Average **Recall** of 0.79, and an Average **F1-Score** of 0.78

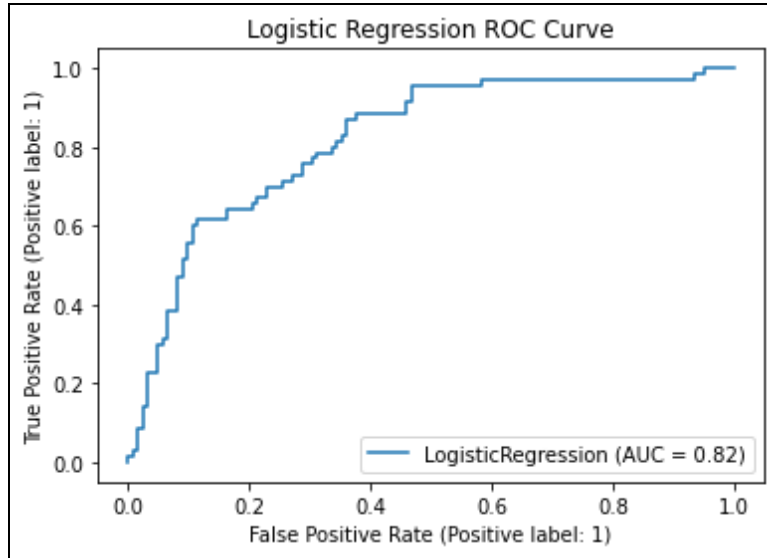


Figure 5: The ROC Curve of the Logistic Regression Classifier, giving an Area Under the Curve of 0.82.

The ROC curve for this classifier gives an **Area Under the Curve** of 0.82.

**Results on Training Set**

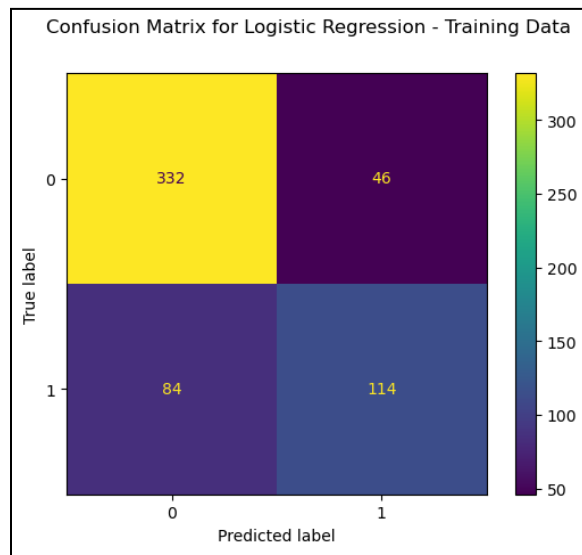


Figure 6. Confusion Matrix of Logistic Regression Model on Training Data.

The Confusion Matrix of the training data predictions using logistic regression gave:  
**True Negatives: 332, True Positives: 114, False Negatives: 84, False Positives: 46**

*****Classification Report On Training Set*****					
	precision	recall	f1-score	support	
0	0.80	0.88	0.84	378	
1	0.71	0.58	0.64	198	
accuracy			0.77	576	
macro avg	0.76	0.73	0.74	576	
weighted avg	0.77	0.77	0.77	576	

Figure 7. Classification Report of Logistic Regression on Training Data.

The Classification Report on the training set:

Average **Precision** of 0.77, an Average **Recall** of 0.77, and an Average **F1-Score** of 0.77

## 3.2 Neural Network Results

### Results on Test Set

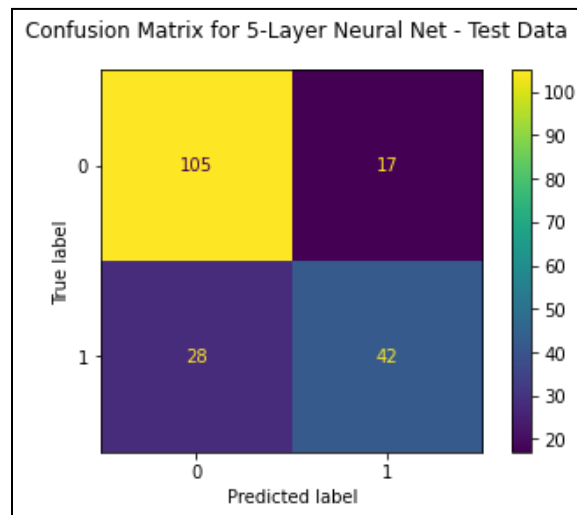


Figure 8. Confusion Matrix of 5-Layer Neural Network Classifier on Test Data.

The Confusion Matrix of the test data predictions using a 5-layer neural network gives:

**True Negatives:** 105, **True Positives:** 42, **False Negatives:** 28, **False Positives:** 17

*****Classification Report On Test Set*****					
	precision	recall	f1-score	support	
0	0.79	0.86	0.82	122	
1	0.71	0.60	0.65	70	
accuracy			0.77	192	
macro avg	0.75	0.73	0.74	192	
weighted avg	0.76	0.77	0.76	192	

Figure 9. Classification Report of a 5-Layer Neural Network Classifier on Test Data.

The Classification Report gives:

Average **Precision** of 0.76, an Average **Recall** of 0.77, and an Average **F1-Score** of 0.76

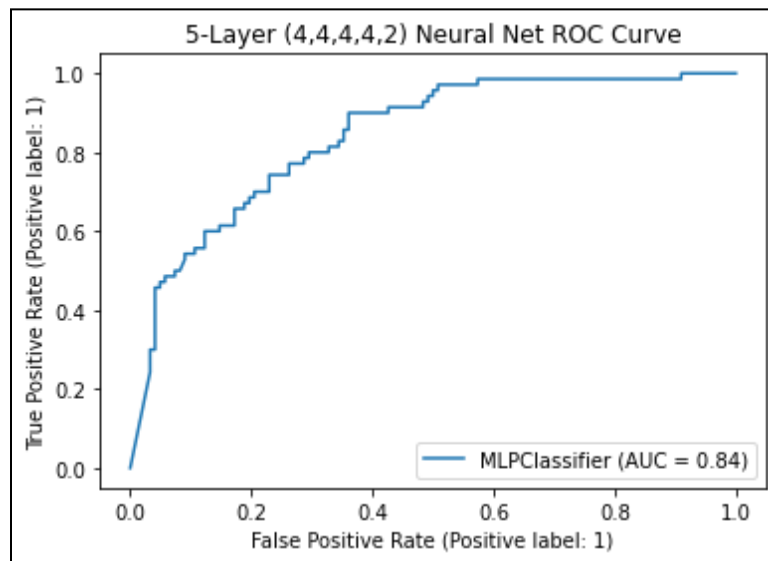


Figure 10. The ROC Curve of the 5-Layer Neural Network Classifier gives an AUC of 0.84.

The ROC curve for this classifier gives an **Area Under the Curve** of 0.84.

### ***Results on Training Set***

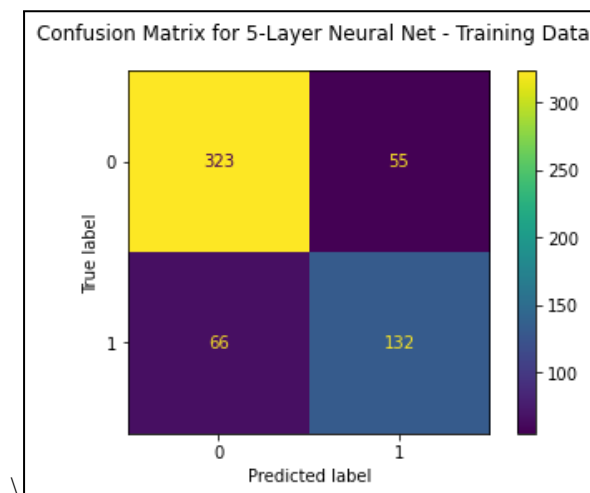


Figure 11. Confusion Matrix of a 5-Layer Neural Network Classifier on Training Data.

The Confusion Matrix of the training data predictions using logistic regression gave:

**True Negatives: 323, True Positives: 132, False Negatives: 66, False Positives: 55**

*****Classification Report On Training Set*****				
	precision	recall	f1-score	support
0	0.83	0.85	0.84	378
1	0.71	0.67	0.69	198
accuracy			0.79	576
macro avg	0.77	0.76	0.76	576
weighted avg	0.79	0.79	0.79	576

Figure 12. Classification Report of a 5-Layer Neural Network Classifier on Training Data

The Classification Report on the training set:  
Average **Precision** of 0.79, an Average **Recall** of 0.79, and an Average **F1-Score** of 0.79

### 3.3 Summary of Results

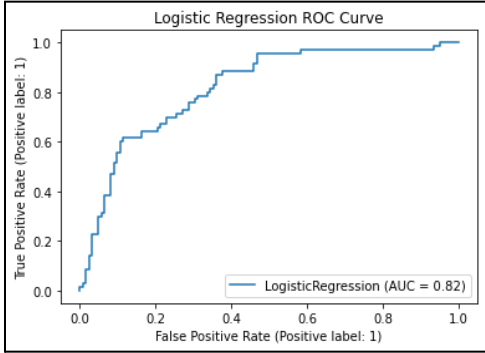
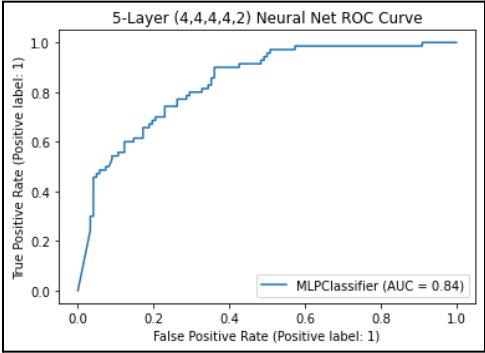
Metric	Logistic Regression		5-Layer Neural Net	
	Test	Training	Test	Training
Precision	0.78	0.77	0.76	0.79
Recall	0.79	0.77	0.77	0.79
Accuracy	0.79	0.77	0.77	0.79
F1 Score	0.78	0.77	0.76	0.79
ROC				

Table 5. Summary of classification performance for each model.

## 4 Discussion

The summary Table 5 shows that both methods performed well, with high precision, recall, accuracy, and F1 scores, all above 75%. Recall, which measures how well the model can identify all positive cases, is essential for a medical diagnosis classifier like ours [11], trying to detect the presence of diabetes, giving high confidence in both models.

The logistic regression model performed slightly better on the test set than on the training set, indicating that the training was appropriately controlled and did not indicate overfitting. In the case of the neural network, the training error was slightly lower than the validation error, suggesting some overfitting. An attempt was made to reduce the number of layers and nodes. However, the resultant drop in accuracy on the test model caused us to abandon that approach.

The Logistic Regression Classifier and 5 Layer Neural Network were also evaluated using the ROC curve, which shows the effectiveness of a binary classifier system as its classification threshold is varied. In predicting the onset of diabetes in a test subject, a ROC curve with a high AUC value is desirable, as it indicates that the classifier can accurately distinguish between subjects who are likely to develop diabetes and those who are not [12]. The Logistic Regression Classifier had an AUC of 0.82, and the 5-layer neural network had an AUC of 0.84.

## 5 Conclusion

The Logistic Regression Model has a higher recall, accuracy, and F1 score on the test set, meaning it may be more effective at making correct predictions overall. However, the Neural Network Model has a higher area under the curve on the ROC, suggesting that it may be better at distinguishing between the two classes, even if it is not as accurate overall [12]. More data would be required to determine which model performs better for diabetes onset prediction. Until then, the performance metrics are too close to confirm which method is superior. However, it can be said that both Logistic Regression and Neural Network Classifiers performed well in diagnosing diabetes in Pima Indians.

As a future step, a larger and more representative dataset would be collected to improve the accuracy of both models [13]. As an additional step, feature selection techniques, such as Principal Component Analysis and removing correlated features, can be used to remove insignificant predictors and improve model performance [14].

## References

- [1] "Diabetes basics," *Centers for Disease Control and Prevention*, 25-Oct-2022. [Online]. Available: <https://www.cdc.gov/diabetes/basics/>. [Accessed: 13-Dec-2022].
- [2] "About diabetes," *Diabetes Canada Association*. [Online]. Available: <https://www.diabetes.ca/en-CA/about-diabetes>. [Accessed: 13-Dec-2022].
- [3] American Diabetes Association, "Diagnosis and classification of diabetes mellitus," *Diabetes Care*, vol. 33, no. Supplement\_1, pp. S62–S69, Dec. 2010, doi: 10.2337/dc10-s062.
- [4] "Diabetes Overview," *Diabetes Symptoms, Causes, & Treatment*. [Online]. Available: <https://diabetes.org/diabetes>. [Accessed: 13-Dec-2022].
- [5] P. H. Bennett, T. A. Burch, and M. Miller, "Diabetes mellitus in American (PIMA) Indians," *The Lancet*, vol. 298, no. 7716, pp. 125–128, 1971, doi: 10.1016/s0140-6736(71)92303-8.
- [6] Knowler, W.C., P.H. Bennett, R.F. Hamman, and M. Milier, "Diabetes incidence and prevalence in Pima Indians: A 19-fold greater incidence than in Rochester, Minnesota," *American Journal of Epidemiology*, vol. 108, no. 6, pp. 497–505, 1978, doi: 10.1093/oxfordjournals.aje.a112648.
- [7] Knowler, W.C., D.J. Pettitt, P.J. Savage, and P.H. Bennett, "Diabetes incidence in Pima Indians: Contributions of obesity and Parental Diabetes1," *American Journal of Epidemiology*, vol. 113, no. 2, pp. 144–156, 1981, doi: 10.1093/oxfordjournals.aje.a113079.
- [8] "Pima Indians Diabetes Database," *Kaggle*, 06-Oct-2016. [Online]. Available: <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>. [Accessed: 13-Dec-2022].
- [9] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus," *Proceedings of the Annual Symposium on Computer Applications in Medical Care*, pp. 261–265, Nov. 1988.
- [10] World Health Organization, "Diabetes mellitus: Report of a WHO study group.," *Annals of Internal Medicine*, vol. 104, no. 4, p. 597, 1986, doi: 10.7326/0003-4819-104-4-597\_2.
- [11] R. A. Deeks, D. G. Altman, and D. J. Giri, "The importance of the recall measure in evaluating diagnostic accuracy," *Medical Decision Making*, vol. 15, no. 4, pp. 431–436, 1995.
- [12] Martin J. McNeil, David B. Peck, and L. Thomas Fry, "Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach," *Biometrics*, vol. 42, no. 3, pp. 887–898, 1986, doi: 10.2307/2531595.



[13] J. M. Luque and J. Derrac, "The relationship between sample size and the accuracy of a machine learning model," *Knowledge-Based Systems*, vol. 138, pp. 70–78, 2018.

[14] A. Zheng and A. Casari, *Feature Engineering for Machine Learning: Principles and techniques for Data scientists*. Sebastopol, CA: O'Reilly, 2018.

## Appendix

```
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
import numpy as np
import pandas as pd
from numpy.linalg import inv
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
from collections import defaultdict
from random import randint
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import plot_confusion_matrix
import warnings
warnings.filterwarnings("ignore")
randSeed = 1936

dataset = pd.read_csv('diabetes.csv', header = 'infer')
X = dataset.iloc[:, :-1].values
xdf = pd.DataFrame(X)
t = dataset.iloc[:, -1].values
x_train, x_test, y_train, y_test = train_test_split(X, t, test_size=0.25,
random_state=randSeed)
dataset.describe()

import seaborn as sns
x_pd = pd.read_csv('diabetes.csv')
sns.pairplot(x_pd, kind='reg', diag_kind='hist')
correlations= x_pd.corr()
correlations

x_pd.boxplot(figsize=(15,15))
fig, ax = plt.subplots(2, 4, figsize=(25, 12.5))
x_pd.boxplot('Pregnancies', 'Outcome', ax=ax[0][0], patch_artist=True)
x_pd.boxplot('Glucose', 'Outcome', ax=ax[0][1], patch_artist=True)
x_pd.boxplot('BloodPressure', 'Outcome', ax=ax[0][2], patch_artist=True)
x_pd.boxplot('SkinThickness', 'Outcome', ax=ax[0][3], patch_artist=True)
x_pd.boxplot('Insulin', 'Outcome', ax=ax[1][0], patch_artist=True)
x_pd.boxplot('BMI', 'Outcome', ax=ax[1][1], patch_artist=True)
x_pd.boxplot('DiabetesPedigreeFunction', 'Outcome', ax=ax[1][2], patch_artist=True)
x_pd.boxplot('Age', 'Outcome', ax=ax[1][3], patch_artist=True)
```

```

####Scale X Data to be in Range [0,1]####
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
x_pd_scale = sc.fit_transform(x_pd)
x_pd_df = pd.DataFrame(x_pd_scale)
x_pd_df = x_pd_df.rename(columns={0 : 'Pregnancies',1:'Glucose', 2:'BloodPressure',
3:'SkinThickness', 4:'Insulin', 5:'BMI', 6:'DiabetesPedigreeFunction', 7:'Age', 8:
'Outcome'})
x_pd_df.describe()
print(t.sum())

```

```

####Log Reg####
logReg = LogisticRegression()
logReg.fit(x_train, y_train)
logTrain = logReg.predict(x_train)
logPred = logReg.predict(x_test)
print("*****Logistic Regression*****")
print("*****Confusion Matrix On Training Set*****")
print(confusion_matrix(y_train,logTrain))
print("*****Classification Report On Training Set*****")
print(classification_report(y_train,logTrain))
print("*****Confusion Matrix On Test Set*****")
print(confusion_matrix(y_test,logPred))
print("*****Classification Report On Test Set*****")
print(classification_report(y_test,logPred))
RocCurveDisplay.from_estimator(logReg, x_test, y_test)
plt.title("Logistic Regression ROC Curve")
plt.show()

```

```

####Log Reg ####
logRegCV = LogisticRegressionCV(solver='sag', max_iter=10)
logRegCV.fit(x_train, y_train)
logCVTrain = logRegCV.predict(x_train)
logCVPred = logRegCV.predict(x_test)
print("*****Logistic Regression*****")
print("*****Confusion Matrix On Training Set*****")
print(confusion_matrix(y_train,logCVTrain))
print("*****Classification Report On Training Set*****")
print(classification_report(y_train,logCVTrain))
print("*****Confusion Matrix On Test Set*****")
print(confusion_matrix(y_test,logCVPred))
print("*****Classification Report On Test Set*****")

```

```

print(classification_report(y_test, logCVPred))
RocCurveDisplay.from_estimator(logRegCV, x_test, y_test)
plt.title("Logistic Regression ROC Curve")
plt.show()

```

#### ####3 Layer Neural Net####

```

nNet = MLPClassifier(random_state=randSeed, hidden_layer_sizes=(8,8), max_iter=400)
nNet.fit(x_train, y_train)
nNetTrain = nNet.predict(x_train)
nNetPred = nNet.predict(x_test)
print("*****Neural Net 3-Layer*****")
print("*****Confusion Matrix On Training Set*****")
print(confusion_matrix(y_train, nNetTrain))
print("*****Classification Report On Training Set*****")
print(classification_report(y_train, nNetTrain))
print("*****Confusion Matrix On Test Set*****")
print(confusion_matrix(y_test, nNetPred))
print("*****Classification Report On Test Set*****")
print(classification_report(y_test, nNetPred))
RocCurveDisplay.from_estimator(nNet, x_test, y_test)
plt.title("3-Layer (8,8,2) Neural Net ROC Curve")
plt.show()

```

#### ####5 Layer Neural Net####

```

nNetBig = MLPClassifier(random_state=randSeed, hidden_layer_sizes=(4,4,4,4),
max_iter=400)
nNetBig.fit(x_train, y_train)
nNetBigTrain = nNetBig.predict(x_train)
nNetBigPred = nNetBig.predict(x_test)
print("*****Neural Net 5-Layer*****")
print("*****Confusion Matrix On Training Set*****")
print(confusion_matrix(y_train, nNetBigTrain))
print("*****Classification Report On Training Set*****")
print(classification_report(y_train, nNetBigTrain))
print("*****Confusion Matrix On Test Set*****")
print(confusion_matrix(y_test, nNetBigPred))
print("*****Classification Report On Test Set*****")
print(classification_report(y_test, nNetBigPred))
RocCurveDisplay.from_estimator(nNetBig, x_test, y_test)
plt.title("5-Layer (4,4,4,4,2) Neural Net ROC Curve")
plt.show()

```

#### ####7 Layer Neural Net####

```

nNetMassive = MLPClassifier(random_state=randSeed,
hidden_layer_sizes=(40,40,40,40,40,40), max_iter=1000)

```

```

nNetMassive.fit(x_train, y_train)
nNetMassiveTrain = nNetMassive.predict(x_train)
nNetMassivePred = nNetMassive.predict(x_test)
print("*****Neural Net 7-Layer*****")
print("*****Confusion Matrix On Training Set*****")
print(confusion_matrix(y_train,nNetMassiveTrain))
print("*****Classification Report On Training Set*****")
print(classification_report(y_train,nNetMassiveTrain))
print("*****Confusion Matrix On Test Set*****")
print(confusion_matrix(y_test,nNetMassivePred))
print("*****Classification Report On Test Set*****")
print(classification_report(y_test,nNetMassivePred))
RocCurveDisplay.from_estimator(nNetMassive, x_test, y_test)
plt.title("7-Layer (40,40,40,40,40,2) Neural Net ROC Curve")
plt.show()

```

#### ####7 Layer Neural Net with Dropout####

```

nNetMassive = MLPClassifier(random_state=randSeed,
hidden_layer_sizes=(40,40,40,40,40,40), max_iter=1000, early_stopping = True)
nNetMassive.fit(x_train, y_train)
nNetMassiveTrain = nNetMassive.predict(x_train)
nNetMassivePred = nNetMassive.predict(x_test)
print("*****Neural Net 7-Layer with Dropout*****")
print("*****Confusion Matrix On Training Set*****")
print(confusion_matrix(y_train,nNetMassiveTrain))
print("*****Classification Report On Training Set*****")
print(classification_report(y_train,nNetMassiveTrain))
print("*****Confusion Matrix On Test Set*****")
print(confusion_matrix(y_test,nNetMassivePred))
print("*****Classification Report On Test Set*****")
print(classification_report(y_test,nNetMassivePred))
RocCurveDisplay.from_estimator(nNetMassive, x_test, y_test)
plt.title("7-Layer (40,40,40,40,40,2) Neural Net ROC Curve With Early Stopping")
plt.show()

```

#### ####Default Neural Net####

```

nNetDef = MLPClassifier(random_state=randSeed, max_iter=400)
nNetDef.fit(x_train, y_train)
nNetDefTrain = nNetDef.predict(x_train)
nNetDefPred = nNetDef.predict(x_test)
print("*****Neural Net 5-Layer*****")
print("*****Confusion Matrix On Training Set*****")
print(confusion_matrix(y_train,nNetDefTrain))
print("*****Classification Report On Training Set*****")
print(classification_report(y_train,nNetDefTrain))

```

```
print("*****Confusion Matrix On Test Set*****")
print(confusion_matrix(y_test,nNetDefPred))
print("*****Classification Report On Test Set*****")
print(classification_report(y_test,nNetDefPred))
RocCurveDisplay.from_estimator(nNetDef, x_test, y_test)
plt.title("Default Neural Net ROC Curve")
plt.show()
```

```
fig = plot_confusion_matrix(logReg, x_test, y_test, display_labels=nNet.classes_)
fig.figure_.suptitle("Confusion Matrix for Logistic Regression")
plt.show()
fig = plot_confusion_matrix(nNet, x_test, y_test, display_labels=nNet.classes_)
fig.figure_.suptitle("Confusion Matrix for 3-Layer Neural Net")
plt.show()
fig = plot_confusion_matrix(nNetBig, x_test, y_test, display_labels=nNet.classes_)
fig.figure_.suptitle("Confusion Matrix for 5-Layer Neural Net")
plt.show()
fig = plot_confusion_matrix(nNetMassive, x_test, y_test,
display_labels=nNet.classes_)
fig.figure_.suptitle("Confusion Matrix for 7-Layer Neural Net")
plt.show()
```

```
fig = plot_confusion_matrix(nNetBig, x_train, y_train,
display_labels=nNet.classes_)
fig.figure_.suptitle("Confusion Matrix for 5-Layer Neural Net - Training Data")
plt.show()
```