CSE 587 - Data Intensive Computing
Project 1
Phase 2 Report

Mrunmayee Vijay Rane UBIT : mrane
Steve Thomas UBIT : stevetho

Problem Statement:

- Analyze various NHANES datasets and predict people who are at risk of developing diabetes and depression.

Phase 2:

NHANES dataset for year 2015-16 and 2017 – 2020 from phase 1 included classes to be determined between diabetic and depressive people. Predicting diabetic people and depressive people individually is important to analyze people who are at risk of developing diabetes and depression.

In order to gain intelligence from the dataset in phase 1; we have used following Machine Learning Algorithms:

1. Gaussian Naive Bayes Classifier
2. Multi-class Logistic Regression
3. Random Forest Classifier/ Regressor
4. Gradient Boost Classifier
5. Xgboost Classifier/Regressor

Additionally, we explored and implemented other classification models such as ada boost classifier, Decision tree, SVM with radial basis function, Linear SVM and SVM with sigmoid function.

I. Gaussian Naive Bayes Classifier:

The Bayes Theorem is the foundation of Naive Bayes Classifiers. Strong independence assumptions between the characteristics are one presumption used. These classifiers make the erroneous assumption that the value of one feature is unrelated to the value of any other feature. Naive Bayes Classifiers are learned effectively in supervised learning environments. A modest amount of training data is required for naive bayes classifiers in order to estimate the classification parameters. Bayes' Theorem:

$P(h|d) = (P(d|h) * P(h)) / P(d)$

Where

- P(h|d) is the probability of hypothesis h given the data d. This is called the posterior probability.
- P(d|h) is the probability of data d given that the hypothesis h was true.
- P(h) is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
- P(d) is the probability of the data (regardless of the hypothesis).

Because the probabilities for each hypothesis are simplified to make their calculation tractable, it is also known as naïve Bayes or idiot Bayes. Instead of attempting to compute each attribute value individually, P(d1, d2, d3|h), they are believed to be conditionally independent given the goal value and calculated as P(d1|h) * P(d2|H), and so on.

Continuous valued features are supported by Gaussian Naive Bayes, which also models each as following a Gaussian (normal) distribution.

Assuming that the data is characterized by a Gaussian distribution with no covariance (independent dimensions) between dimensions is one method for building a straightforward model. Finding the mean and standard deviation of the points within each label, which is all that is required to establish such a distribution, will allow this model to be fit.

1. **<u>Explanation and Analysis</u>:**

   We tried implementing Gaussian Naive Bayes classifier for detecting the following four classes.

   1. Non depression - Non diabetes as Class 0 and labeled as 0.
   2. Non depression - Diabetes as Class 1 and labeled as 1.
   3. Depression - Non diabetes as Class 2 and labeled as 2.
   4. Depression - Diabetes as Class 3 and labeled as 3.

   We first segregated these class values in a column named DPDB_VAL as the output column or dependent variable y for classifying these four classes.

   Rest features are considered as independent variables contributing to dependent variables.

   First split the dataset as 75% (training): 25% (testing) and then implemented Gaussian navies bayes model using sklearn. Initial count of 4 classes were as follows:

```
In [24]:  1 finald["DPDB_VAL"].value_counts()

Out[24]: 0.0    3375
         2.0    1048
         1.0    1021
         3.0     466
         Name: DPDB_VAL, dtype: int64
```

1. Non depression - Non diabetes as Class 0 and labeled as 0. Count = 3375
2. Non depression - Diabetes as Class 1 and labeled as 1. Count = 1021
3. Depression - Non diabetes as Class 2 and labeled as 2. Count = 1048
4. Depression - Diabetes as Class 3 and labeled as 3. Count = 466

Out of which gaussian naive bayes has predicted as following:

```
In [40]: 1 from scipy.stats import itemfreq
         2 itemfreq(y_test)

<ipython-input-40-819e16e8029e>:2: DeprecationWarning: `itemfreq` is deprecated!
`itemfreq` is deprecated and will be removed in a future version. Use instead `np.unique(..., return_counts=True)`
  itemfreq(y_test)

Out[40]: array([[  0., 854.],
                [  1., 252.],
                [  2., 266.],
                [  3., 106.]])
```

Class 0: 854

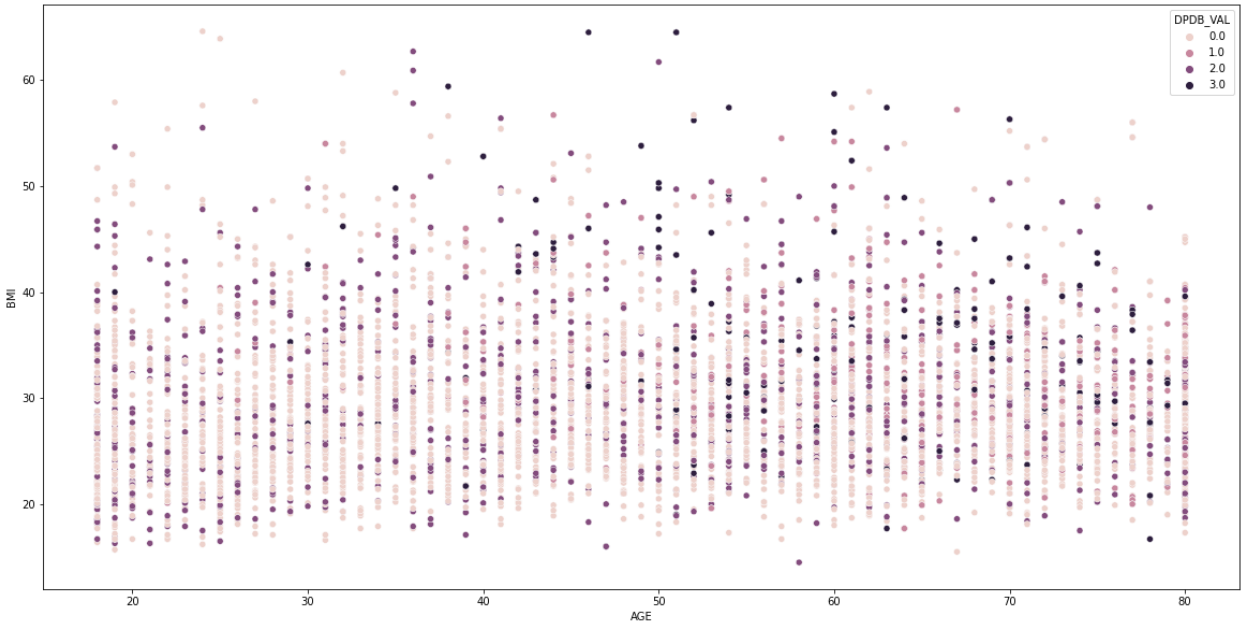
Class 1: 252

Class 2: 266

Class 3: 106

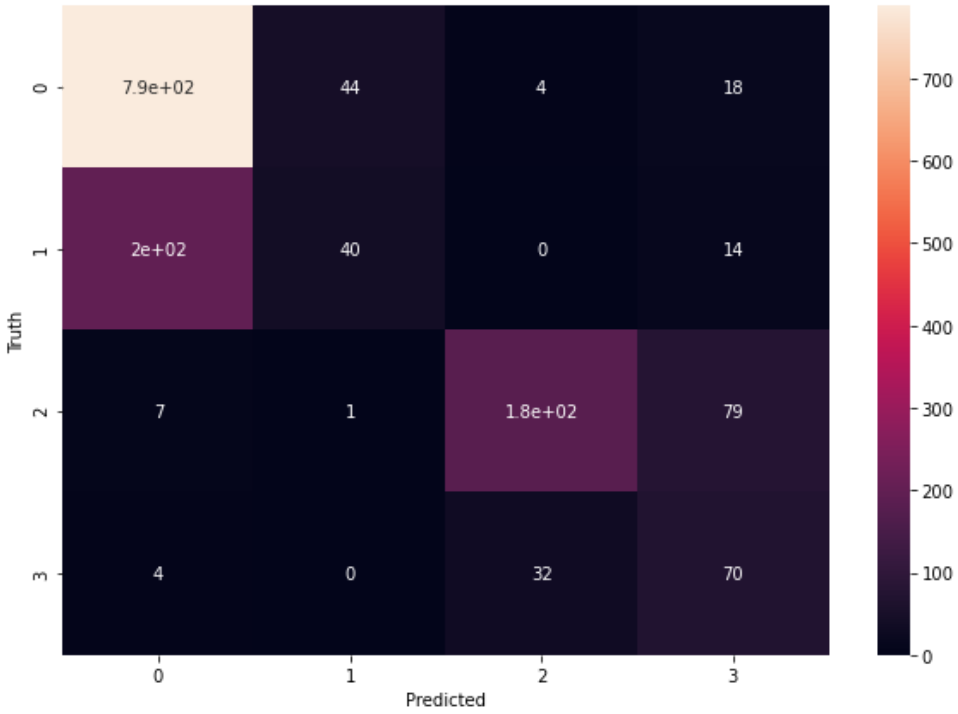
2. **Model Insights:**

Gained 72.86% accuracy for naive bayes model. We visualized a scatter plot of age and BMI in the dataset with these 4 classes.



3. **Visualization**:

Confusion Matrix Plot for Naives Bayes:



Classification Report:

```
In [36]:  1 print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0.0	0.79	0.92	0.85	854
1.0	0.47	0.16	0.24	252
2.0	0.83	0.67	0.74	266
3.0	0.39	0.66	0.49	106
accuracy			0.73	1478
macro avg	0.62	0.60	0.58	1478
weighted avg	0.71	0.73	0.70	1478

Precision, Recall, F1-score:

```
Confusion matrix :
[[788  44   4  18]
 [198  40   0  14]
 [  7   1 179  79]
 [  4   0  32  70]]
Accuracy score: 0.7286874154262517
Precision Score :  0.6200644549151836
Recall Score :  0.6036891189206133
F1 score:  0.580227001232518
```

II. Multi-class - Multinomial Logistic Regression:

A variation of logistic regression that includes native support for multi-class classification issues is known as multinomial logistic regression.

By default, logistic regression can only be used to solve two-class classification issues. Although they necessitate that the classification problem be initially turned into numerous binary classification problems, some extensions, such as one-vs-rest, can enable logistic regression to be utilized for multi-class classification problems.

1. **Explanation and Analysis:**

   We tried to implement multinomial logistic regression for classification of above-mentioned classes such as non-depression - non diabetes, non-depression - diabetes, depression – non diabetes, depression - diabetes. This model gave 73.95% accuracy on predicting these 4 classes.

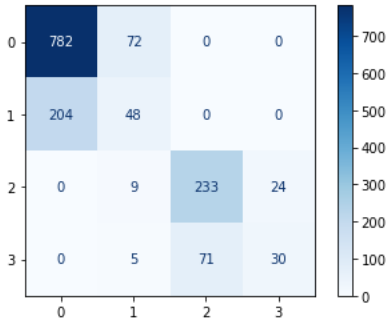
   We later tried to predict individual probabilities such as prediction of diabetes as dependent variable and rest features as independent variables gaining accuracy of 76.65%. Additionally, we predicted depression as dependent variable and rest features as independent variable with accuracy of 100%.

2. **Model Insights:**

   It accurately (100%) predicted labels for depression given rest features such as age, BMI, diabetes, audiometry data, ethnicity, and blood pressure. However, for predicting diabetes it was only 76.65% accurate. Implicitly indicating the relationship between diabetes or the rest parameters and depression.

3. **Visualization for classification problem**:

   Confusion Matrix Plot for Multinomial Logistic Regression :



Classification Report:

```
In [87]: 1 print(classification_report(y_test, y_pred))
```

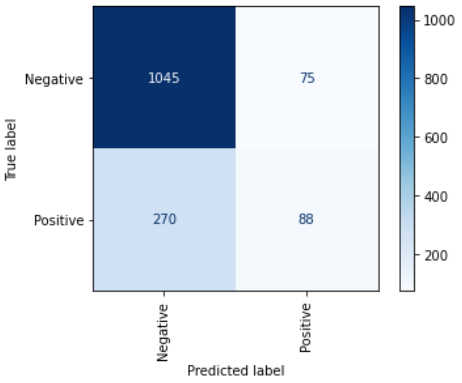
	precision	recall	f1-score	support
0.0	0.79	0.92	0.85	854
1.0	0.36	0.19	0.25	252
2.0	0.77	0.88	0.82	266
3.0	0.56	0.28	0.38	106
accuracy			0.74	1478
macro avg	0.62	0.57	0.57	1478
weighted avg	0.70	0.74	0.71	1478

Precision, Recall, F1-score:

```
Confusion matrix :
[[782  72   0   0]
 [204  48   0   0]
 [  0   9 233  24]
 [  0   5  71  30]]
Accuracy score: 0.7395128552097429
Precision Score :  0.6183288318690877
Recall Score :  0.5662814436338295
F1 score:  0.5728121307153895
```

<u>Visualization for prediction of Diabetes</u>:

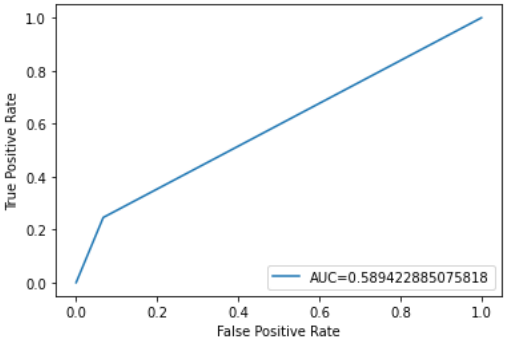
Confusion Matrix:



Precision, Recall, F1-score:

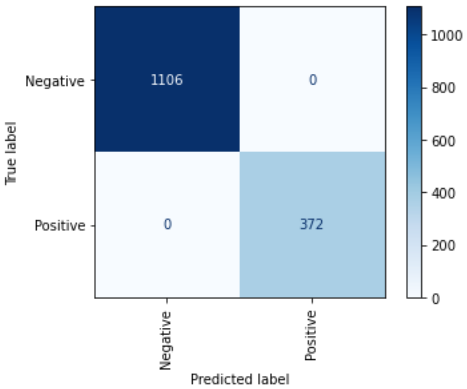
```
Confusion matrix :
[[1045   75]
 [ 270   88]]
Accuracy score: 0.7665764546684709
Precision Score :  0.6672770533485736
Recall Score :  0.5894228850758181
F1 score:  0.5980640609789262
```

AUC Curve:



Visualization for prediction of Depression:

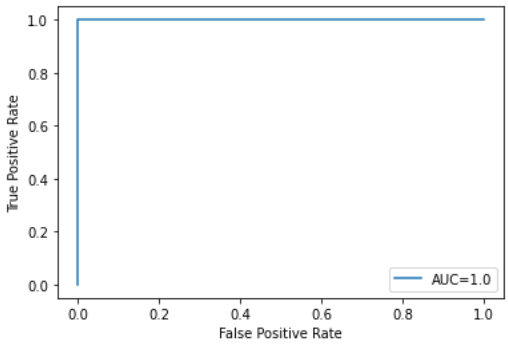
Confusion Matrix:



Precision, Recall, F1score:

```
Confusion matrix :
[[1106    0]
 [   0  372]]
Accuracy score: 1.0
Precision Score :  1.0
Recall Score :  1.0
F1 score:  1.0
```

AUC Curve:



III. <u>Random Forest Classifier/ Regressor</u>:

A supervised machine learning technique known as the Random Forest or Random Decision Forest is used for classification, regression, and other tasks using decision trees.

A randomly chosen portion of the training data is used by the Random forest classifier to generate a collection of decision trees. It simply consists of a collection of decision trees (DT) from a randomly chosen subset of the training set, which are subsequently used to decide the final prediction.

The wisdom of crowds is the basic idea behind random forest. The random forest model performs better than any of its component models when it is a committee of many generally uncorrelated models (trees), according to data science theory.

1. <u>Explanation and Analysis</u>:

We implemented random forest classifier for classification of above-mentioned classes such as non-depression - non diabetes, non-depression - diabetes, depression – non diabetes, depression - diabetes. This model gave 78.8% accuracy on predicting these 4 classes.

We later predicted individual probabilities such as prediction of diabetes as dependent variable and rest features as independent variable using random forest regressor,

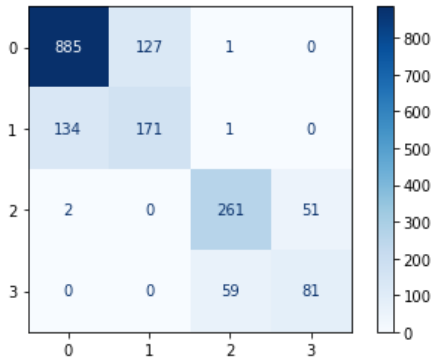
gaining accuracy of 74.8%. Additionally, we predicted depression as dependent variable and rest features as independent variable with accuracy of 100%.

2. Model Insights:

It accurately (100%) predicted labels for depression given rest features such as age, BMI, diabetes, Audiometry data, ethnicity, and blood pressure. However, for predicting diabetes, it was only 74.8% accurate. Implicitly indicating the relationship between diabetes and depression.

3. Visualization for Classification:

Confusion Matrix:



Classification Report:

```
In [50]: 1 print(classification_report(y_test, y_pred))
```

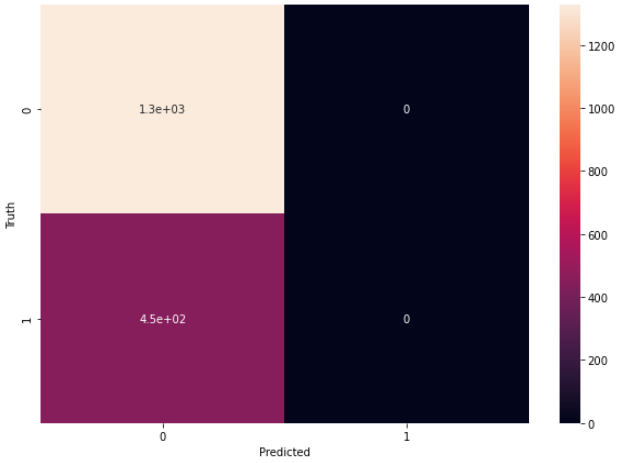
	precision	recall	f1-score	support
0	0.87	0.87	0.87	1013
1	0.57	0.56	0.57	306
2	0.81	0.83	0.82	314
3	0.61	0.58	0.60	140
accuracy			0.79	1773
macro avg	0.72	0.71	0.71	1773
weighted avg	0.79	0.79	0.79	1773

Precision, Recall, F1-score:

```
Confusion matrix :
[[885 127   1   0]
 [134 171   1   0]
 [  2   0 261  51]
 [  0   0  59  81]]
Accuracy score: 0.7884940778341794
Precision Score :  0.7162045326984615
Recall Score :  0.7105619486682759
F1 score:  0.71319365187842
```

Visualization for Regression – predicting diabetes:

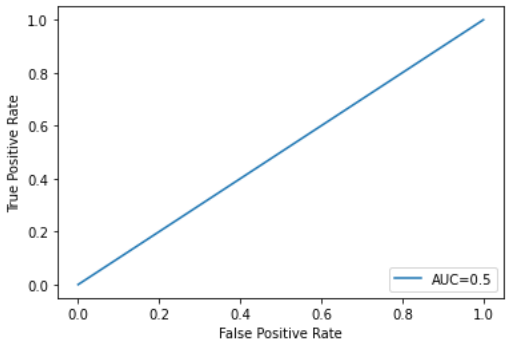
Confusion Matrix:



Precision, Recall, F1score:

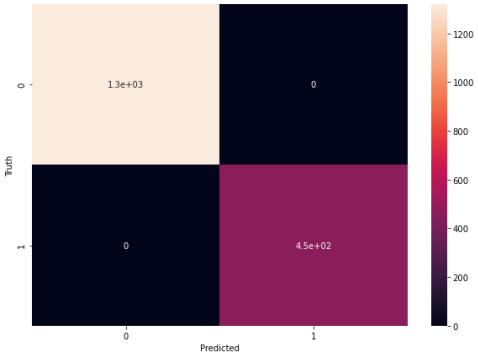
```
Confusion matrix :
[[1327    0]
 [ 446    0]]
Accuracy score: 0.748448956570784
Precision Score :  0.374224478285392
Recall Score :  0.5
F1 score:  0.42806451612903224
```

AUC Curve:



Visualization for Regression – predicting depression:

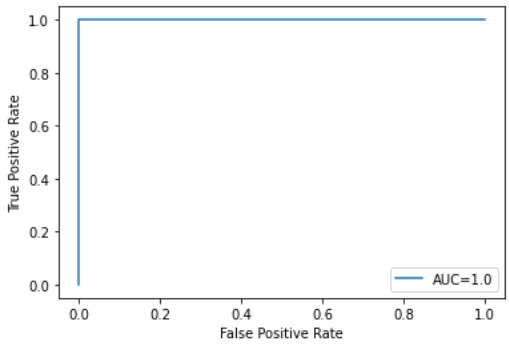
Confusion Matrix:



Precision, Recall, F1-score:

```
Confusion matrix :
[[1319    0]
 [   0  454]]
Accuracy score: 1.0
Precision Score :  1.0
Recall Score :  1.0
F1 score:  1.0
```

AUC Curve:



IV. Gradient Boost Classifier:

One of the most effective methods for creating predictive models is gradient boosting, often known as a Generalization of AdaBoost. Gradient Boost is a gradient descent optimization approach to add weak learners with the primary goal of minimizing the

loss function. As a result of the generalization, the technique was able to support regression, multi-class classification, and other tasks in addition to binary classification difficulties.

Three components of Gradient Boost:

Loss Function: The loss function's purpose is to determine how well the model performs predictions using the available data. Depending on the type of the issue, this might change.

Weak Learner: A weak learner is one who, when compared to random guessing, classifies the data so badly. Although various models can be employed in GBM, decision trees tend to be the weak learners.

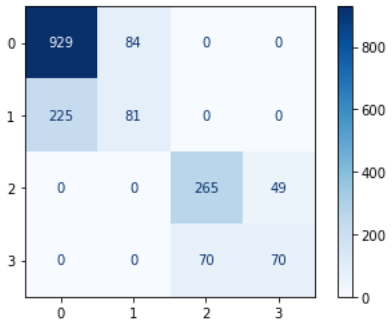
Additive Model: The decision trees are added one step at a time through an iterative and sequential procedure. The value of the loss function should decrease with each repetition. When the loss reaches a tolerable level or the training no longer improves the performance on an external validation dataset, a fixed number of trees are added.

1. **Explanation and Analysis:**

   We implemented gradient boost for classification of above-mentioned classes such as non-depression - non diabetes, non-depression - diabetes, depression – non diabetes, depression - diabetes. This model gave 75.9% accuracy on predicting these 4 classes.

2. **Visualization for Classification**:

   Confusion Matrix:



Classification Report:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.92      0.86      1013
           1       0.49      0.26      0.34       306
           2       0.79      0.84      0.82       314
           3       0.59      0.50      0.54       140

    accuracy                           0.76      1773
   macro avg       0.67      0.63      0.64      1773
weighted avg       0.73      0.76      0.74      1773
```

Precision, Recall, F1 score:

```
Confusion matrix :
[[929  84   0   0]
 [225  81   0   0]
 [  0   0 265  49]
 [  0   0  70  70]]
Accuracy score: 0.7586012408347433
Precision Score :  0.6688037894199841
Recall Score :  0.6314332282796482
F1 score:  0.639634281024266
```

V. <u>XGboost Classifier and Regressor</u>:

Extreme Gradient Boosting is the abbreviation for XGBoost. XGBoost is a distributed gradient boosting library that has been developed to be very effective, adaptable, and portable. It uses the Gradient Boosting framework to implement machine learning algorithms. It offers a parallel tree boosting to address a variety of data science challenges quickly and accurately.

Boosting is an ensemble learning strategy that creates a strong classifier out of a number of successively weak classifiers. In order to address the bias-variance trade-off, boosting techniques are essential. Boosting algorithms regulate all the components of a model—bias and variance—and are thought to be more effective than bagging algorithms, which solely account for excessive variance.

1. <u>Explanation and Analysis</u>:

We implemented Xgboost classifier for classification of above-mentioned classes such as non-depression - non diabetes, non-depression - diabetes, depression – non diabetes, depression - diabetes. This model gave the best accuracy of 81.8% on predicting these 4 classes.

We later predicted individual probabilities such as prediction of diabetes as dependent variable and rest features as independent variable using Xgboost regressor gaining R2

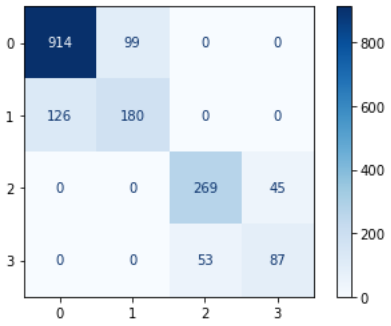
score of 0.22 and mean square error regression loss of 0.146. Additionally, we predicted depression as dependent variable and rest features as independent variable with R2 score of 0.998 and mean square error regression loss of 0.0002.

2. Model Insights:

It classifies all the four classes with 81.8% accuracy. It accurately predicted labels for depression given rest features such as age, BMI, diabetes, Audiometry data, ethnicity, and blood pressure. However, for predicting diabetes, it was only 74.8% accurate. Implicitly indicating the relationship between diabetes and depression.

3. Visualization for Classification:

Confusion Matrix:



Classification Report:

```
In [38]:  1 print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.88	0.90	0.89	1013
1	0.65	0.59	0.62	306
2	0.84	0.86	0.85	314
3	0.66	0.62	0.64	140
accuracy			0.82	1773
macro avg	0.75	0.74	0.75	1773
weighted avg	0.81	0.82	0.82	1773

Precision, Recall, F1-score:

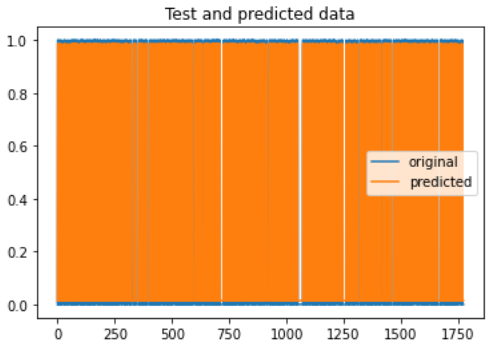
```
Confusion matrix :
[[914  99   0   0]
 [126 180   0   0]
 [  0   0 269  45]
 [  0   0  53  87]]
Accuracy score: 0.8178228990411731
Precision Score :  0.7546255199919295
Recall Score :  0.7421555618367844
F1 score:  0.7478516834583057
```

Visualization for Regression – predicting depression:

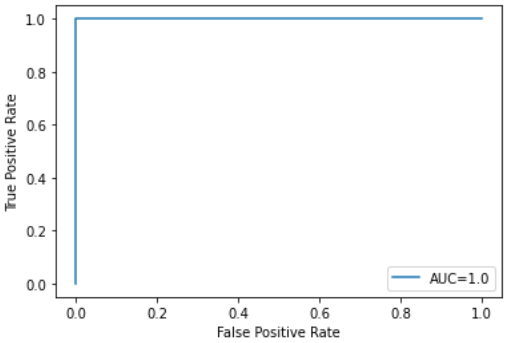
Metrics:

```
Maximum Residual Error:  0.014180958271026611
R2 Regression Score:  0.9989484806724465
Mean Squared Error Regression Loss:  0.00020030898803273493
Mean Absolute Error Regression Loss:  0.00020030898803273493
```

Plot of Test and Predicted data:



AUC Curve:

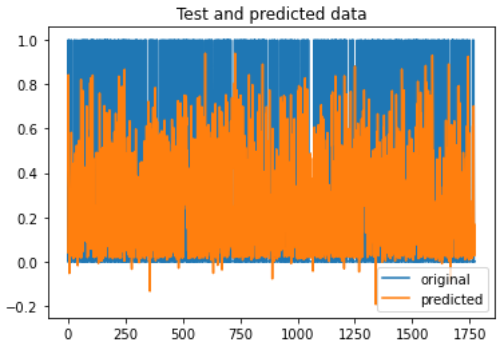


Visualization for Regression – predicting diabetes:

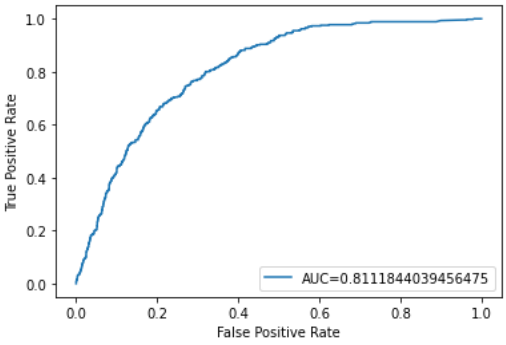
Metrics:

```
Maximum Residual Error:  0.9977018348872662
R2 Regression Score:  0.22044757172353857
Mean Squared Error Regression Loss:  0.1467687647405185
Mean Absolute Error Regression Loss:  0.1467687647405185
```

Plot of Test and Predicted data:



AUC Curve:



VI. Decision Trees:

A decision tree classifier is a non-parametric supervised machine learning algorithm that can be used for classification and regression. A decision tree has a structure that is similar to a tree with nodes with each internal node representing a test on an attribute, each branch in the tree represent the outcome of the test, and each leaf represents a class label. The decision trees implicitly perform feature selection and can handle multi-label output. The decision tree can be used to perform classification without requiring much computation power.

1. Explanation and Analysis

We implemented decision tree classifier for classification of above-mentioned classes such as non-depression - non diabetes, non-depression - diabetes, depression – non diabetes, depression - diabetes. This model gave the accuracy of 70.6% on predicting these 4 classes.

2. Model Insights

The decision model gave a good accuracy, shows that classification of the feature can easily be achieved and this also implies other ensemble models that uses decision tree might be able to achieve even better accuracy.

3. Visualizations

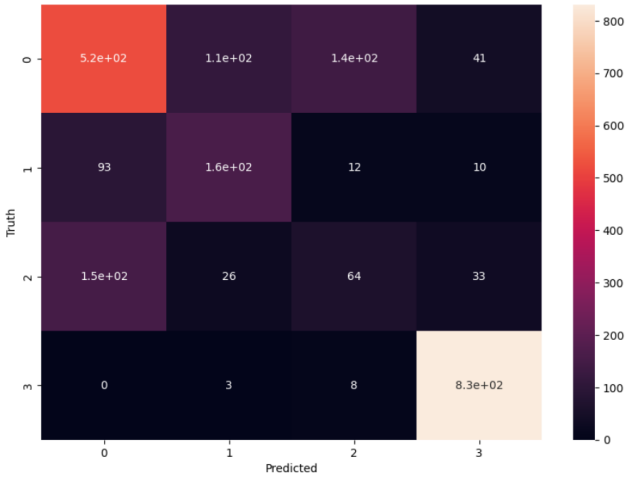
Metrics:

	precision	recall	f1-score	support
0	0.68	0.64	0.66	814
1	0.53	0.58	0.55	275
2	0.29	0.23	0.26	274
3	0.91	0.99	0.95	842
accuracy			0.71	2205
macro avg	0.60	0.61	0.60	2205
weighted avg	0.70	0.71	0.71	2205

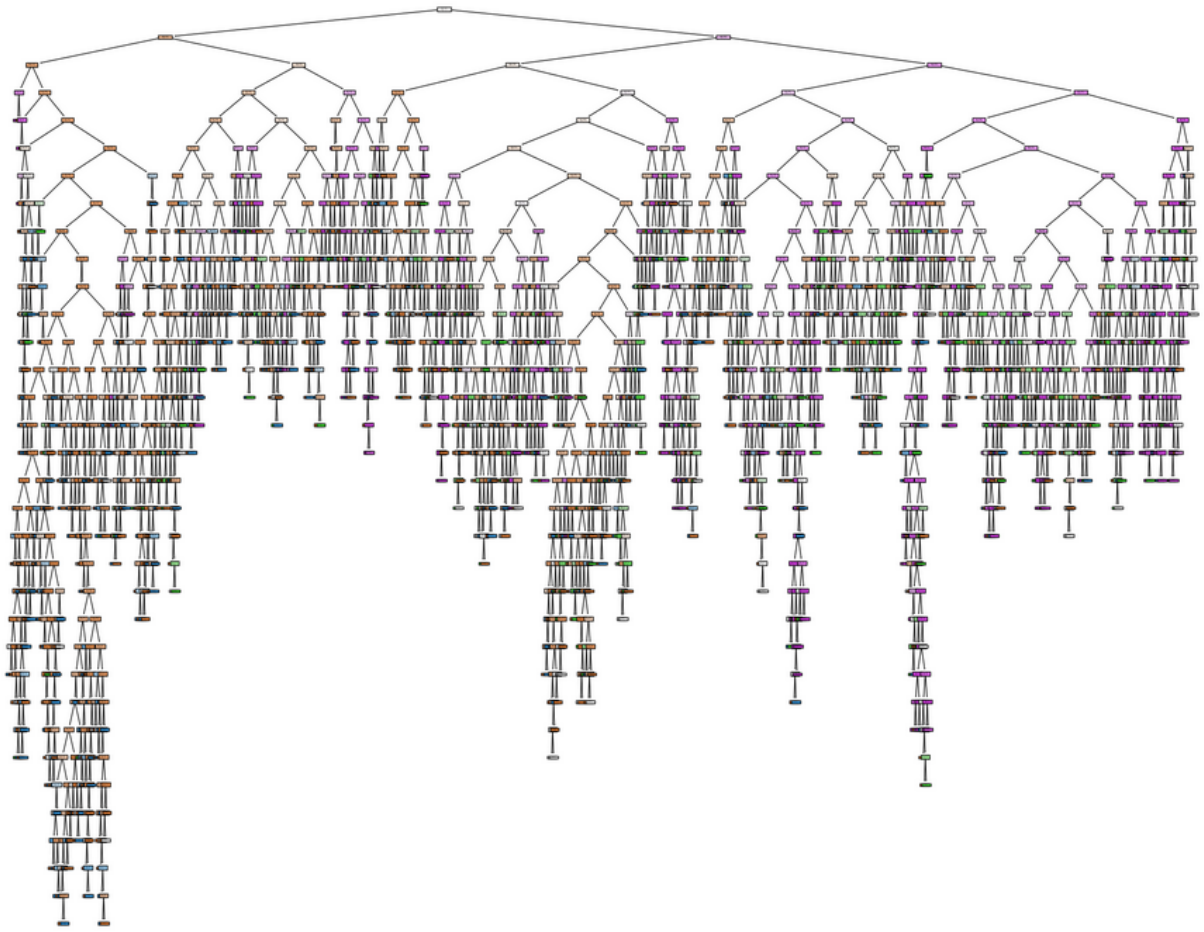
Precision, Recall and F1 score:

```
Confusion matrix :
[[521 113 139  41]
 [ 93 160  12  10]
 [151  26  64  33]
 [  0   3   8 831]]
Accuracy score: 0.7147392290249434
Precision Score :  0.6015098282859602
Recall Score :  0.6105949577966152
F1 score:  0.6044949730873876
```

Confusion matrix:



Tree Structure:



VII Support Vector Machine

1. Explanation and Analysis

Support Vector Machine is a supervised learning model that aims to find the optimal decision boundary (hyperplane) that distinctly classifies the data points. The hyperplane can be linear or nonlinear. In order to find nonlinear hyperplanes, we use kernels which applies a transformation to the features and then try to find the hyperplane. The polynomial kernel applies a polynomial transformation. The radial basis function (RBF) kernel tries to find new features by finding the distance between the feature points and centers. SVM works really well in situations where there is a clear margin of separation and there are high dimensional spaces. Here, we have tried linear, polynomial and radial basis function SVM. The best accuracy was achieved by polynomial SVM with a score of 59.96%

2. Model Insights

The accuracies for SVM models ranged from 54.8% (linear) to 59.96%. This showed that there was no clear margin of separations between the feature points and there is no clear hyperplane separating the feature points.

3. Visualizations

Linear SVM:

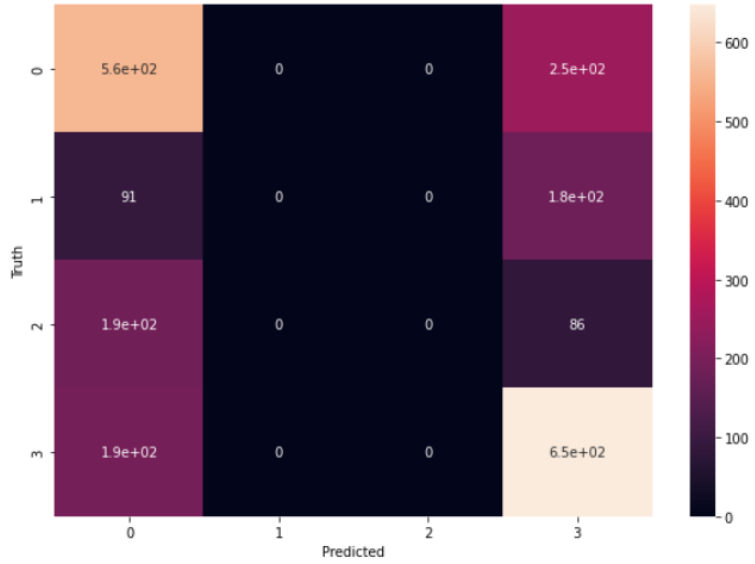
Metrics:

	precision	recall	f1-score	support
0.0	0.54	0.69	0.61	814
1.0	0.00	0.00	0.00	275
2.0	0.00	0.00	0.00	274
3.0	0.55	0.77	0.64	842
accuracy			0.55	2205
macro avg	0.27	0.36	0.31	2205
weighted avg	0.41	0.55	0.47	2205

Precision, Recall and F1 score:

```
Confusion matrix :
[[561   0   0 253]
 [ 91   0   0 184]
 [188   0   0  86]
 [194   0   0 648]]
Accuracy score: 0.5482993197278911
Precision Score :  0.27398159420026524
Recall Score :  0.36469634717853244
F1 score:  0.3127395145837769
```

Confusion Matrix:



Polynomial SVM

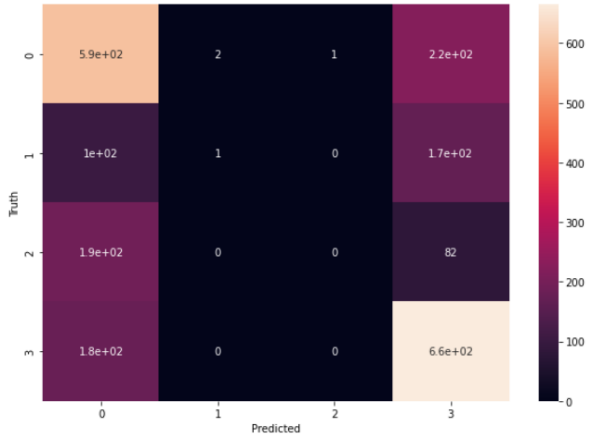
Metrics:

	precision	recall	f1-score	support
0.0	0.56	0.72	0.63	814
1.0	0.33	0.00	0.01	275
2.0	0.00	0.00	0.00	274
3.0	0.58	0.79	0.67	842
accuracy			0.57	2205
macro avg	0.37	0.38	0.33	2205
weighted avg	0.47	0.57	0.49	2205

Precision, Recall and F1 score:

```
Confusion matrix :
[[590   2   1 221]
 [103   1   0 171]
 [192   0   0  82]
 [177   0   0 665]]
Accuracy score: 0.5696145124716553
Precision Score :  0.36818359184469807
Recall Score :  0.37955957793249956
F1 score:  0.3268925510702365
```

Confusion Matrix:



RBF SVM:

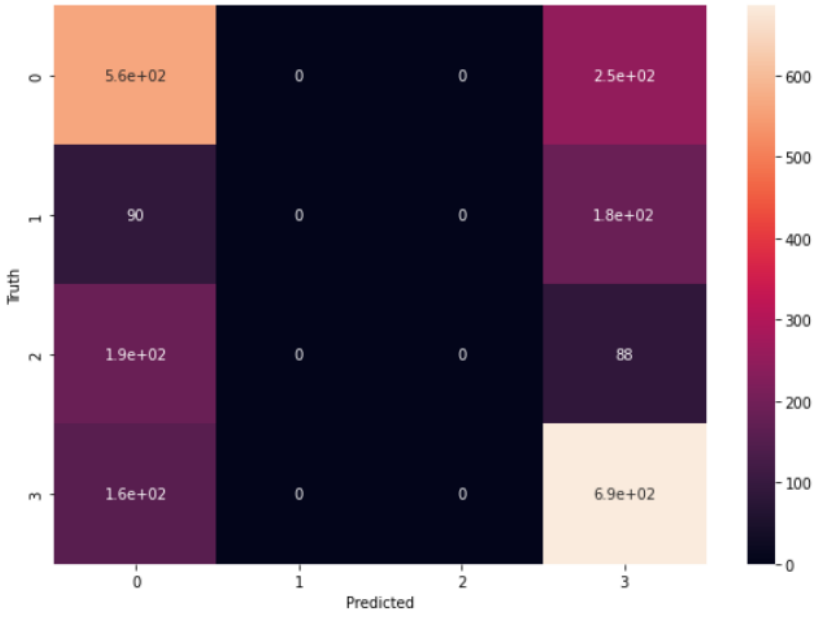
Metrics:

	precision	recall	f1-score	support
0.0	0.57	0.69	0.62	814
1.0	0.00	0.00	0.00	275
2.0	0.00	0.00	0.00	274
3.0	0.57	0.81	0.67	842
accuracy			0.57	2205
macro avg	0.28	0.38	0.32	2205
weighted avg	0.43	0.57	0.49	2205

Precision, Recall and F1 score:

```
Confusion matrix :
[[563   0   0 251]
 [ 90   0   0 185]
 [186   0   0  88]
 [156   0   0 686]]
Accuracy score: 0.5664399092970521
Precision Score :  0.2831928236222434
Recall Score :  0.3765932581253246
F1 score:  0.32276483081667684
```

Confusion Matrix:



ML models Tables:

Sr. No.	Model	Accuracy
1.	Gaussian Naive Bayes Classifier	72.86%
2.	Multinomial Logistic Regression	73.95%
3.	Logistic Regression for predicting diabetes Logistic Regression for predicting depression	76.65% 100%
4.	Random Forest Classifier	78.8%
5.	Random Forest Regressor for predicting diabetes Random Forest Regressor for predicting depression	74.8% 100%
6.	Gradient Boost Classifier	75.9%
7.	XgBoost Classifier	81.8%

8.	XGBoost Regressor for predicting diabetes XGBoost Regressor for predicting depression	R2 score: 0.22 R2 score: 0.99
9.	Ada Boost Classifier	75.9%
10	Decision Tree	71.56%
11	Linear SVM	54.82%
	RBF SVM	56.96%
	Polynomial SVM	56.64%

REFERENCES:

1. https://machinelearningmastery.com/blog/
2. https://www.mygreatlearning.com/blog/xgboost-algorithm/
3. https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496

4. https://scikit-learn.org/stable/modules/svm.html

5. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47