Data Mining - Assignment 2 Report

Team: Kiran Venkatesh Kulkarni - 1001848434
Vijetha Shenoy Badiadka - 1001822855

Data Mining is used to extract useful information from datasets and display it easy visulaizations.

1) Decision Tree methods and Naïve Bayes Classifier:

Decision Tree methods: Decision tree methodology is used to establish classification systems based on multiple covariates or for developing prediction algorithms for a target variable. Recently, this has become more popular in the medical industry. It classifies a population into branch like segments that construct an inverted tree with a root node, internal nodes, and leaf nodes. It is a non-parametric algorithm which can deal with large, complicated datasets. When the sample size is big enough, the study data can be divided into training and validation datasets. The training data set can be used to build the decision tree model and validation dataset to decide on appropriate tree size needed to achieve the optimal final model. Decision trees are generally easier to understand because it shares internal decision-making logic. There are 4 types of Decision tree algorithms:
ID3, CART (Classification and Regression Trees), Chi-square and Reduction in variance.

ID3 decision tree algorithm uses Information Gain to decide the splitting points. In order to measure how much information we gain, we use entropy to calculate the homogeneity of the sample. (Information gain is the decrease or increase in the entropy value when the node is split).

Information Gain = 1- Entropy

Entropy = $- \sum pi \log2 pi$

CART (Classification and Regression Trees): This uses the Gini method to create split points including Gini Index (Gini impurity) and Gini Gain. (Gini Index is the probability of assigning a wrong label to a sample by picking the label randomly and is also used to measure feature importance in trees).

Gini Impurity = 1- Gini

Gini = $\sum pi^2$, where I ranges from 1 to n.

Chi - square: This method of splitting nodes in decision tree for datasets having categorical target values. It can make 2 or more than 2 splits. It works on the statistical differences of the differences between the parent and the child nodes.

Chi-square = squareroot[(actual -expected)^2/expected]

Here, expected is the expected value for a class in the child node based on the distribution of classes in the parent node, and actual is the actual value for a class in the child node.

Naïve Bayes Classifier: Naïve Bayes Classifier is used in text classifier that includes high dimensional training dataset. It is a supervised learning algorithm. It is a probabilistic classifier, meaning it predicts on the basis of probability of an object. Some popular examples are email/spam filtration, sentimental analysis, classifying articles.

There are 3 types of Naïve Bayes model:
1)**Gaussian**: This model assumes that features follow a normal distribution.
2) **Multinomial**: This model is used when the data is multinomial distributed.
3)**Bernoulli**: This model is similar to Multinomial model, but the predictor variables are independent variables.

We use **Gaussian Model** as the predictors take continuous values instead of discrete and then the model assumes that these values are sampled from the Gaussian distribution. The training dataset has continuous values. Thus we apply Gaussian on the training set.

Classification report: Precision, recall, f1-score and support

Precision: This is the ratio of correctly predicted observations to the total predicted positive observations.

Precision = TP/(TP+FP)
Where TP = True Positive
FP = False Positive

Recall: This is the ratio of correctly predicted positive values to all the observations in the actual class.

Recall = TP/(TP+FN)
Where TP = True Positive
FN =. False Negative

F1-score: This is the weighted average of precision and recall.

F1-score = 2*(recall*Precision)/(recall+ precision)

Support: Number of times a particular value has occurred.

Confidence: This refers to the amount of times a given rule turns out to be true in practice.

Preprocessing: We have preprocessed our data as follows:

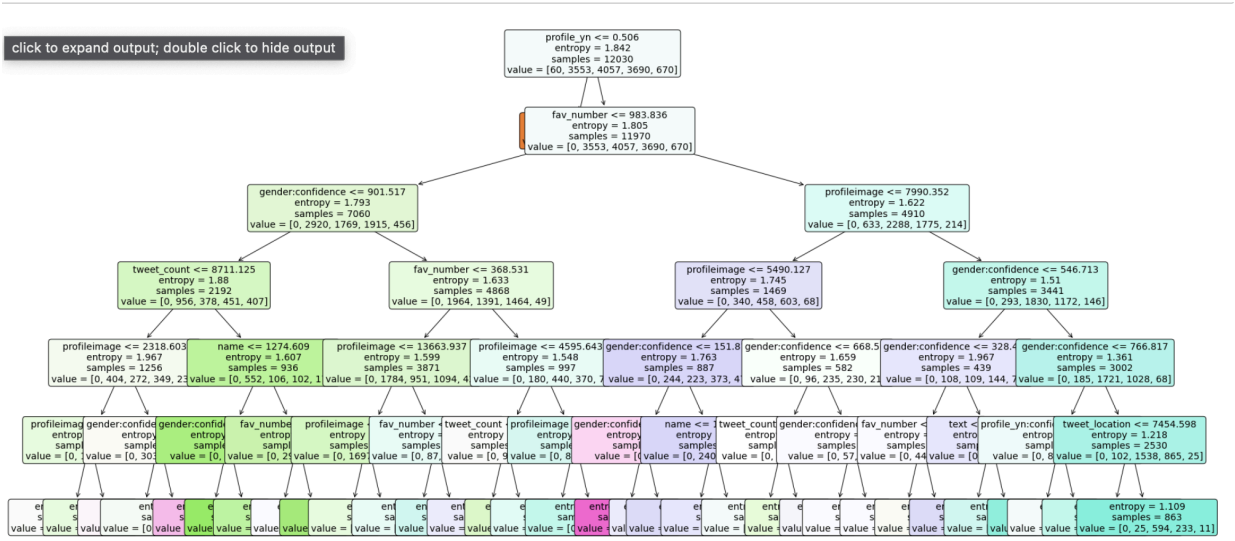
We have used Label Encoder to convert string attributes to numerical values. In the target variable, gender, there is redundant data like ‘brand’ and ‘unknown’. We remove those redundant data by dropping those rows.

2) Describing the dataset:

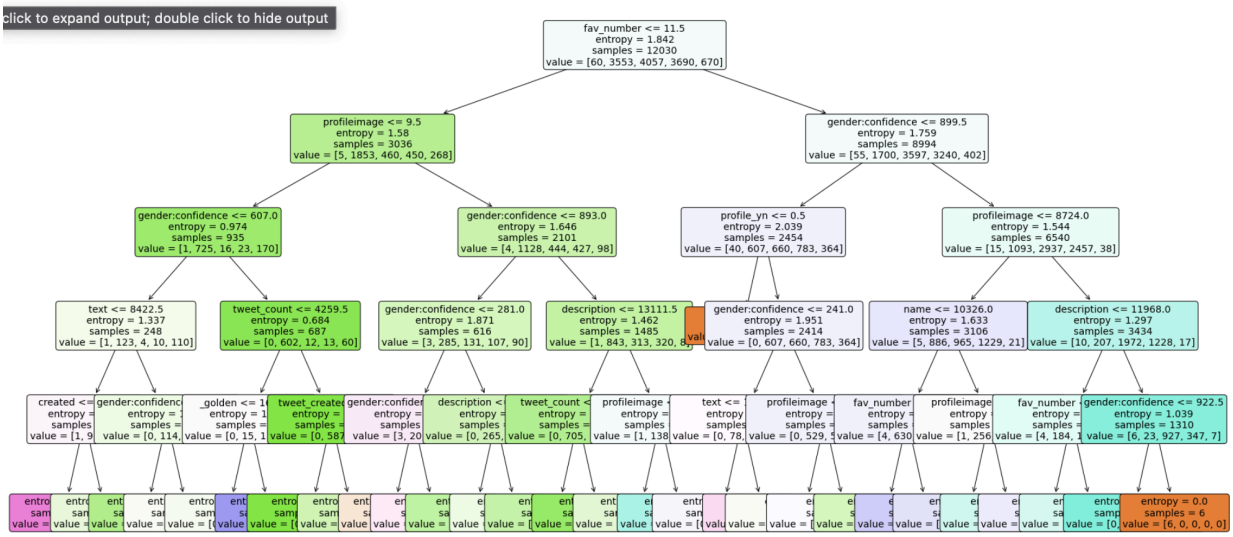
The training dataset was discrete and had around 26 attributes and about 20,0000 rows.

3) Visualization of the decision tree for Gini and entropy:

For Gini, the decision tree is as follows:



For entropy, the decision tree is as follows:



4) Interpret results comparing Gini and entropy:

When we compare Gini and entropy, we find that both Gini and entropy have similar values of accuracy.

For Gini:

```
Using gini to measure the quality of the split
```

click to expand output; double click to hide output

```
Accuracy: 0.5145885286783043
Classification Report
              precision    recall  f1-score   support

           0       0.90      0.76      0.82        37
           1       0.57      0.66      0.61      2389
           2       0.58      0.51      0.55      2643
           3       0.41      0.46      0.43      2504
           4       0.32      0.06      0.10       447

    accuracy                           0.51      8020
   macro avg       0.56      0.49      0.50      8020
weighted avg       0.51      0.51      0.51      8020
```

For entropy:

```
Using Entropy to measure the quality of the split

Accuracy: 0.5145885286783043
Classification Report
              precision    recall  f1-score   support

           0       0.90      0.76      0.82        37
           1       0.57      0.66      0.61      2389
           2       0.58      0.51      0.55      2643
           3       0.41      0.46      0.43      2504
           4       0.32      0.06      0.10       447

    accuracy                           0.51      8020
   macro avg       0.56      0.49      0.50      8020
weighted avg       0.51      0.51      0.51      8020
```

5) Visualize the dataset for the target variable:

Snapshots of the code and the results are as follows:

Loading the data and printing the class labels, along with data preprocessing:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import math
from sklearn.tree import plot_tree
```

In [2]:
```
df=pd.read_csv('gender_Classifier.csv',encoding='latin1')
df.head(7)
```

Out[2]:

	_unit_id	_golden	_unit_state	_trusted_judgments	_last_judgment_at	gender	gender:confidence	profile_yn	profile_yn:confidence	created	...	
0	815719226	False	finalized	3	10/26/2015 23:24	male	1.0000	yes	1.0	12/5/2013 1:48	...	https://pb
1	815719227	False	finalized	3	10/26/2015 23:30	male	1.0000	yes	1.0	10/1/2012 13:51	...	https://pb
2	815719228	False	finalized	3	10/26/2015 23:33	male	0.6625	yes	1.0	11/28/2014 11:30	...	https://pb
3	815719229	False	finalized	3	10/26/2015 23:10	male	1.0000	yes	1.0	6/11/2009 22:39	...	https://pb
4	815719230	False	finalized	3	10/27/2015 1:15	female	1.0000	yes	1.0	4/16/2014 13:23	...	https://pb
5	815719231	False	finalized	3	10/27/2015 1:47	female	1.0000	yes	1.0	3/11/2010 18:14	...	https://pb
6	815719232	False	finalized	3	10/27/2015 1:57	brand	1.0000	yes	1.0	4/24/2008 13:03	...	https://pb

7 rows × 26 columns

In [3]:
```
df.tail(7)
```

Out[3]:

	_unit_id	_golden	_unit_state	_trusted_judgments	_last_judgment_at	gender	gender:confidence	profile_yn	profile_yn:confidence	created	...	
20043	815756700	True	golden	240	NaN	male	1.0000	yes	1.0	9/27/2011 0:19	...	https:/
20044	815756767	True	golden	227	NaN	female	1.0000	yes	1.0	6/11/2014 1:55	...	https:/
20045	815757572	True	golden	259	NaN	female	1.0000	yes	1.0	8/5/2015 21:16	...	https:/
20046	815757681	True	golden	248	NaN	male	1.0000	yes	1.0	8/15/2012 21:17	...	https:/
20047	815757830	True	golden	264	NaN	male	1.0000	yes	1.0	9/3/2012 1:17	...	https:/
20048	815757921	True	golden	250	NaN	female	0.8489	yes	1.0	11/6/2012 23:46	...	https:/
20049	815757985	True	golden	249	NaN	female	1.0000	yes	1.0	4/14/2014 17:22	...	https:/

7 rows × 26 columns

In [4]:
```
print('Class labels before cleaning the data are: ', df.gender.unique().tolist())
```

```
Class labels before cleaning the data are:  ['male', 'female', 'brand', 'unknown', 'Other']
```

In [5]:
```
# le = preprocessing.LabelEncoder()
label_encoder = preprocessing.LabelEncoder()
for col in df:
    df[col] = label_encoder.fit_transform(df[col])
```

In [6]:
```
X = df.drop(['gender'], axis = 1)
Y = df.gender
```

In [7]:
```
df.head()
```

Out[7]:

	_unit_id	_golden	_unit_state	_trusted_judgments	_last_judgment_at	gender	gender:confidence	profile_yn	profile_yn:confidence	created	...	profileimage
0	0	0	0	0	101	3	922	1	266	4986	...	2179
1	1	0	0	0	107	3	922	1	266	1347	...	4099
2	2	0	0	0	110	3	561	1	266	3705	...	14587
3	3	0	0	0	87	3	922	1	266	11907	...	913
4	4	0	0	0	212	2	922	1	266	8932	...	4705

5 rows × 26 columns

In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.4)
```

In [9]:
```
gini = DecisionTreeClassifier(criterion='entropy', splitter = 'random', min_samples_leaf = 5, max_depth = 6)

# Train Decision Tree Classifer
clf_gini = gini.fit(X_train, y_train)

# Predict the response for test dataset
Y_pred_gini = clf_gini.predict(X_test)

accuracy_gini = metrics.accuracy_score(y_test, Y_pred_gini)
# Quantifying the Quality of the split using Classification Report
print("Using Gini Index to measure the quality of the split\n")
print("Accuracy:", accuracy_gini)
```

```
Using Gini Index to measure the quality of the split

Accuracy: 0.486284289276808
```

In [10]:
```
#Print the Accuracy of the split
print("Classification Report\n", metrics.classification_report(y_test, Y_pred_gini))

# USING ENTROPY TO MEASURE THE QUALITY OF SPLIT
# Create Decision Tree classifer object
entropy = DecisionTreeClassifier(criterion='entropy', random_state = 100, min_samples_leaf = 5, max_depth = 5)

# Train Decision Tree Classifer
clf_entropy = entropy.fit(X_train, y_train)

# Predict the response for test dataset
Y_pred_entropy = clf_entropy.predict(X_test)

accuracy_entropy = metrics.accuracy_score(y_test, Y_pred_entropy)
# Print the Accuracy of the split
print("Using Entropy to measure the quality of the split\n")
print("Accuracy:", accuracy_entropy)

# Quantifying the Quality of the split using Classification Report
print("Classification Report\n", metrics.classification_report(y_test, Y_pred_entropy))
```

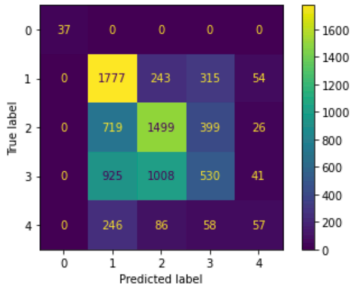
```
Classification Report
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        37
           1       0.48      0.74      0.59      2389
           2       0.53      0.57      0.55      2643
           3       0.41      0.21      0.28      2504
           4       0.32      0.13      0.18       447

    accuracy                           0.49      8020
   macro avg       0.55      0.53      0.52      8020
weighted avg       0.47      0.49      0.46      8020

Using Entropy to measure the quality of the split

Accuracy: 0.5145885286783043
```

Classification Report

	precision	recall	f1-score	support
0	0.90	0.76	0.82	37
1	0.57	0.66	0.61	2389
2	0.58	0.51	0.55	2643
3	0.41	0.46	0.43	2504
4	0.32	0.06	0.10	447
accuracy			0.51	8020
macro avg	0.56	0.49	0.50	8020
weighted avg	0.51	0.51	0.51	8020

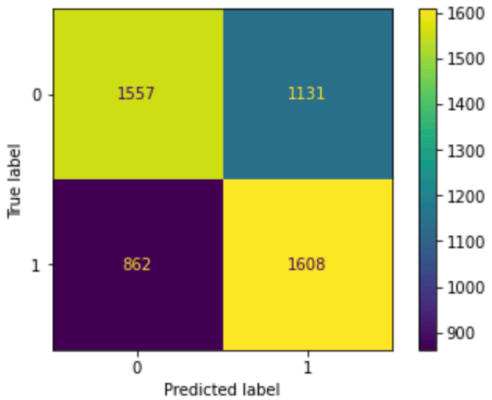
```
In [11]: con_matrix = confusion_matrix(y_test, Y_pred_gini, labels=clf_gini.classes_)
         disp = ConfusionMatrixDisplay(confusion_matrix=con_matrix, display_labels=clf_gini.classes_)
         disp.plot()
```

```
Out[11]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1cf0bf92f40>
```

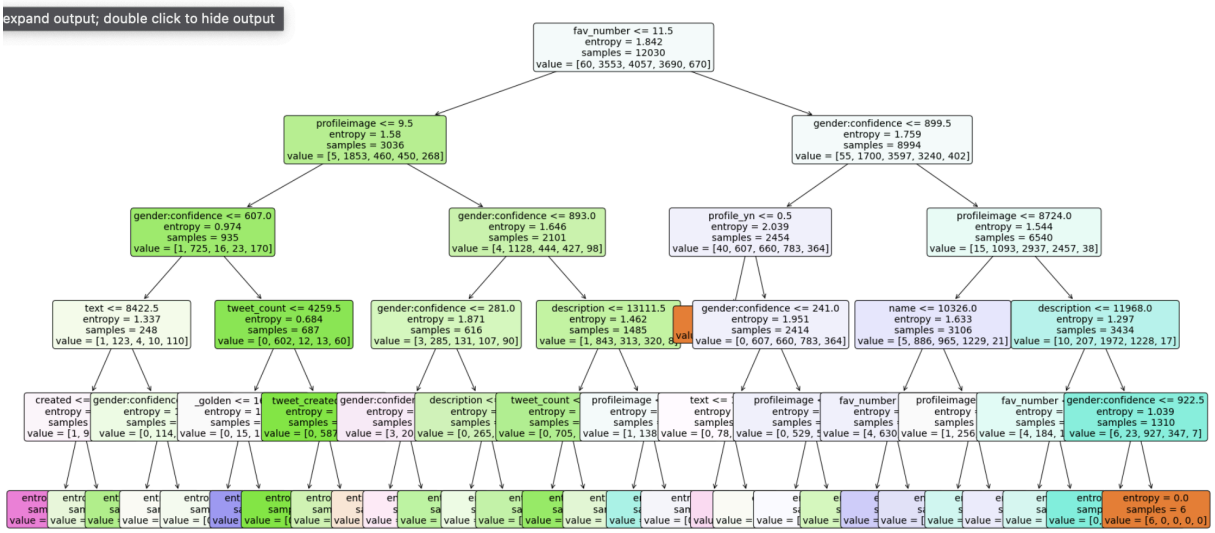


Or even like below:

```
Out[12]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2911b113ee0>
```



In [12]:
```
#4) Print the decision tree visualization. [10 points]

# Using Figure class instance from matplotlib to define axes
plt.figure(figsize=(30,15))

# Plot a Entropy Decision Tree using plot_tree from sklearn
decision_tree = plot_tree(clf_entropy,
                feature_names= list(df.columns[1:25]),
                filled=True,
                rounded=True,
                fontsize=14)
```

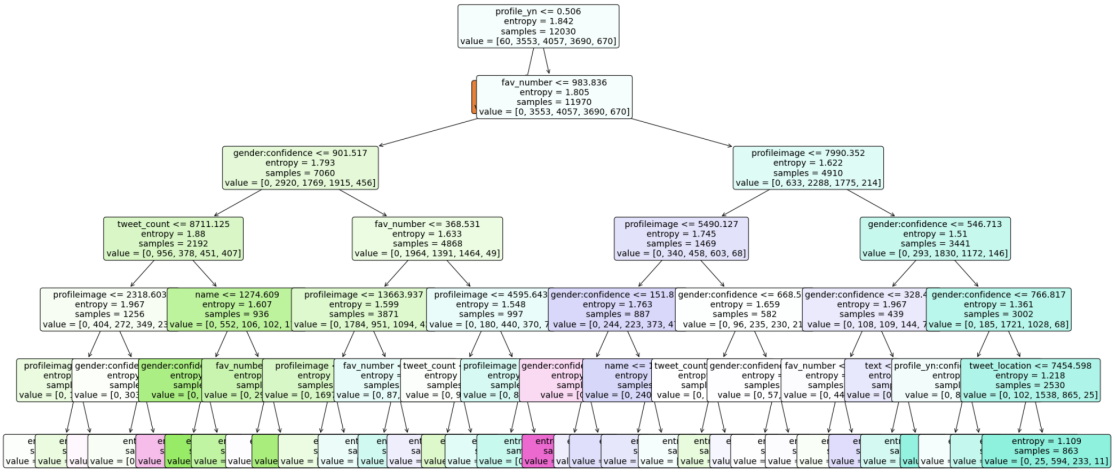


Decision Tree:

In [14]:
```
gini = DecisionTreeClassifier(criterion='gini', random_state = 100, min_samples_leaf = 5, max_depth = 5)

# Train Decision Tree Classifer
clf_gini = gini.fit(X_train, y_train)

# Predict the response for test dataset
Y_pred_gini = clf_gini.predict(X_test)

accuracy_gini = metrics.accuracy_score(y_test, Y_pred_entropy)
# Print the Accuracy of the split
print("Using gini to measure the quality of the split\n")
print("Accuracy:", accuracy_gini)

# Quantifying the Quality of the split using Classification Report
print("Classification Report\n", metrics.classification_report(y_test, Y_pred_entropy))
```

```
Using gini to measure the quality of the split

Accuracy: 0.5145885286783043
Classification Report
                precision    recall  f1-score   support

           0       0.90      0.76      0.82        37
           1       0.57      0.66      0.61      2389
           2       0.58      0.51      0.55      2643
           3       0.41      0.46      0.43      2504
           4       0.32      0.06      0.10       447

    accuracy                           0.51      8020
   macro avg       0.56      0.49      0.50      8020
weighted avg       0.51      0.51      0.51      8020
```

In [13]:
```
# Using Figure class instance from matplotlib to define axes
plt.figure(figsize=(30,15))
# Plot a Gini Decision Tree using plot_tree from sklearn
decision_tree1 = plot_tree(clf_gini,
                           feature_names= list(df.columns[1:25]),
                           filled=True,
                           rounded=True,
                           fontsize=14)
```



Naïve Bayes:

Naive Bayes (10 Points)

1) Use all types of Naive bayes classifier present in the sklearn to predict the test data. If you are not able to implement any of the classifier explain in detail why it can't be done.[5 point]

2) Use comments to explain your code and variable names[1 point]

3) Calculate and print the confusion matrix (use graphics instead showing a 2D array), and the classification Report (includes: precision, recall, f1-score, and support)for all the NB Classifier. show the best classifier. Explain the classification report in your own words. Do not copy from the internet. [4 points]

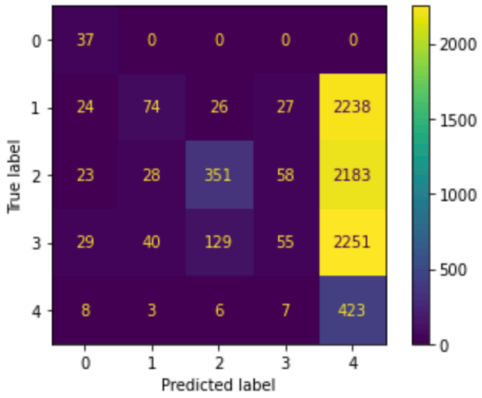
In [15]:
```
# Create Gaussian Naive Bayes classifer object
gaussian = GaussianNB()

# Train Gaussian Naive Bayes Tree Classifer
clf_gaussian = gaussian.fit(X_train, y_train)

# Predict the response for test dataset
Y_pred = clf_gaussian.predict(X_test)

# Print the Accuracy of the split
print("Accuracy:",metrics.accuracy_score(y_test, Y_pred))
```

```
Accuracy: 0.1172069825436409
```

In [16]:
```
# evaluate the accuracy of a classification using confusion matrix and displaying the matrix
con_matrix = confusion_matrix(y_test, Y_pred, labels=clf_gaussian.classes_)
display = ConfusionMatrixDisplay(confusion_matrix=con_matrix, display_labels=clf_gaussian.classes_)
display.plot()
```

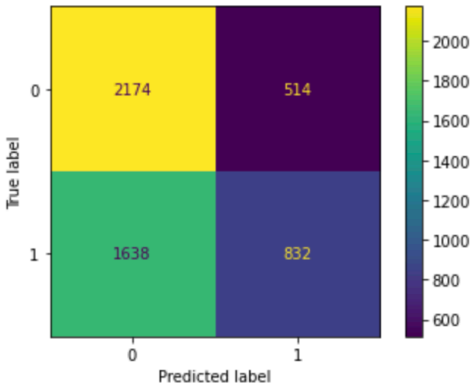
The confusion matrix is as follows:

```
Out[16]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1cf0ec7f1c0>
```



Or for simpler terms, confusion matrix can be like,

```
Out[20]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2911b525250>
```



6) References:

https://www.geeksforgeeks.org/decision-tree-implementation-python/
https://www.vebuso.com/2020/01/decision-tree-intuition-from-concept-to-application/
https://quantdare.com/decision-trees-gini-vs-entropy/
https://www.analyticsvidhya.com/blog/2020/06/4-ways-split-decision-tree/
http://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
https://www.javatpoint.com/machine-learning-naive-bayes-classifier
https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/