Assignment Two

Semester 1 2024

Student Names: Thuan Nguyen, Ha Ly
Student IDs: 23194073, 23194064
PAPER NAME: Foundations of Data Science

PAPER CODE: COMP615

Due Date: 2nd Jun 2024 (midnight NZ time)

TOTAL MARKS: 100

INSTRUCTIONS:

1. **The following actions may be deemed to constitute a breach of the General Academic Regulations Part 7: Academic Discipline,**
   - Communicating with or collaborating with another person regarding the Assignment
   - Copying from any other student work for your Assignment
   - Copying from any third-party websites unless it is an open book Assignment
   - Uses any other unfair means
2. **Please email DCT.EXAM@AUT.AC.NZ if you have any technical issues with your Assessment/Assignment/Test submission on Canvas immediately**
3. **Attach your code for all the datasets in the appendix section**.

Table of Contents

Table of Figures

Part A: Predicting Bank Marketing Campaign Outcomes

a) Explain KNN and Naïve Bayes Algorithms

- KNN: K-Nearest Neighbour is a supervised machine learning algorithm with labelled training data. With every example as a data point in a space of dimension d, d is the number of attributes, D is the set of training of examples, k is the number of nearest neighbours, for each test instance x, the classifier would:
  1. Compute distance between x and every example (point) using Euclidean distance, Manhattan distance, or q norm distance
  2. Select a list Dx (from D) of k training examples that are closest to x
  3. Determine the class from the nearest neighbour list by considering the majority vote of class labels among the k-nearest neighbours
     - Vote is weighted according to distance: w = $1/d^2$

- Naïve Bayes: Naive Bayes methods is a set of supervised learning algorithms using Bayes theorem with naive assumptions which are the statistical independence between pairs of features. Given the dependent feature vector x_1 through $x_{n'}$ and class variable y:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Using the naive conditional independence assumption that:

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

We can use the following classification rule since P(x_1,..., x_n) is constant given the input:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Both P(y) and P(x_i|y) may be estimated using Maximum A Posteriori (MAP); the former represents then the relative frequency of class y in the training set.

b) Perform Exploratory Data Analysis (EDA)

The dataset is related to a direct marketing campaign with an aim to predict the outcome if a client would make a subscription to term deposit (variable y).

```
Bank data set dimensions : (4521, 17)
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
y            object
dtype: object
```

Figure 1: Bank Dataset Information

Based on Figure 1, the bank dataset consists of 4521 instances and 17 attributes which are:

- Features:
    - Numerical: age, balance (average yearly balance in euros), day (last contract day of the month), duration (last contact duration, in seconds), campaign (number of contacts performed during this campaign and for this client), pdays (number of days that passed by after the client was last contacted from a previous campaign), previous (number of contacts performed before this campaign and for this client)
    - Categorical:
        - job: type of job ("admin.", "unknown", "unemployed", "management", "housemaid", "entrepreneur", "student", "blue-collar", "self-employed", "retired", "technician", "services")
        - martial: status ("married", "divorced", "single"; note: "divorced" means divorced or widowed)
        - education: ("unknown", "secondary", "primary", "tertiary")
        - default: has credit in default? (binary: "yes", "no")
        - housing: has housing loan? (binary: "yes", "no")
        - loan: has personal loan? (binary: "yes", "no")
        - contact: contact communication type ("unknown", "telephone", "cellular")

  - month: last contact month of year ("jan", "feb", "mar", “apr”, “may”, “jun”, “jul”, “aug”, “sep”, “oct”, "nov", "dec")
  - poutcome: outcome of the previous marketing campaign ("unknown", "other", "failure", "success")
- Target: y, can be seen as categorical, is actually binary with 1 representing “yes,” and 0 representing “no”.

Then, data cleanliness is evaluated. Firstly, missing values of the dataset is examined.

```
Missing values in dataset:
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
--------------
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

Figure 2: Missing Values in Dataset

Based on Figure 2, there is no missing values or null values in the dataset. Therefore, the dataset is cleaned, and no handling needs to be done.

Afterwards, summary statistics of numerical features is presented as follows.

```
                age       balance          day     duration     campaign  \
count  4521.000000   4521.000000  4521.000000  4521.000000  4521.000000
mean     41.170095   1422.657819    15.915284   263.961292     2.793630
std      10.576211   3009.638142     8.247667   259.856633     3.109807
min      19.000000  -3313.000000     1.000000     4.000000     1.000000
25%      33.000000     69.000000     9.000000   104.000000     1.000000
50%      39.000000    444.000000    16.000000   185.000000     2.000000
75%      49.000000   1480.000000    21.000000   329.000000     3.000000
max      87.000000  71188.000000    31.000000  3025.000000    50.000000

             pdays     previous
count  4521.000000  4521.000000
mean     39.766645     0.542579
std     100.121124     1.693562
min      -1.000000     0.000000
25%      -1.000000     0.000000
50%      -1.000000     0.000000
75%      -1.000000     0.000000
max     871.000000    25.000000
```

Figure 3: Summary Statistics of Dataset's Numerical Features

Based on Figure 3:

- age: clients' ages ranged from 19 years old to 87 years old, while the mean age of all clients is around 41 years old.
- balance: clients' average yearly balance ranged from –3,313.00 EUR to 71,188.00 EUR, while the mean balance of all clients is approximately 1,422.66 EUR
- day: clients' last contact day of the month ranged from 1st to 31st day of the month
- duration: clients' last contact duration ranged from 4 seconds to 3025 seconds, while the mean contacts duration is around 264 seconds.
- campaign: number of contacts during the campaign and for the clients ranged from 1 to 50 contacts, while the mean number of contacts is about 3 contacts.
- pdays: number of days have passed since last contact with a client from previous campaign ranged from –1 to 871 days, while the mean number of those days is about 40 days.
- previous: number of contacts performed before this campaign and for this client ranged from 0 to 25 contacts, while the mean number of those contacts is about 1 contact.

Then, count of values for the target variable of the dataset is shown:

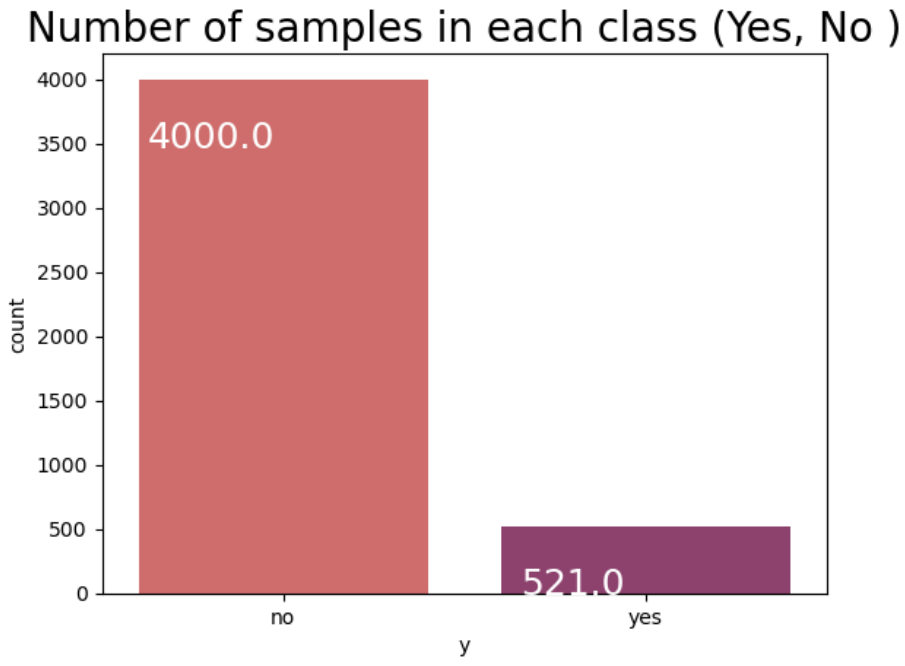


Figure 4: Values' Count of Dataset's Target

With the bar chart from Figure 4, it can clearly be seen that:

- 4000 out of 4521 instances, the client did not subscribe to the bank term deposit (the 'no' column).
- Whereas for a minority of 521 instances (, the client subscribed to the bank term deposit (the 'yes' column).

As a result, an imbalance of the target's values is indicated for this dataset.

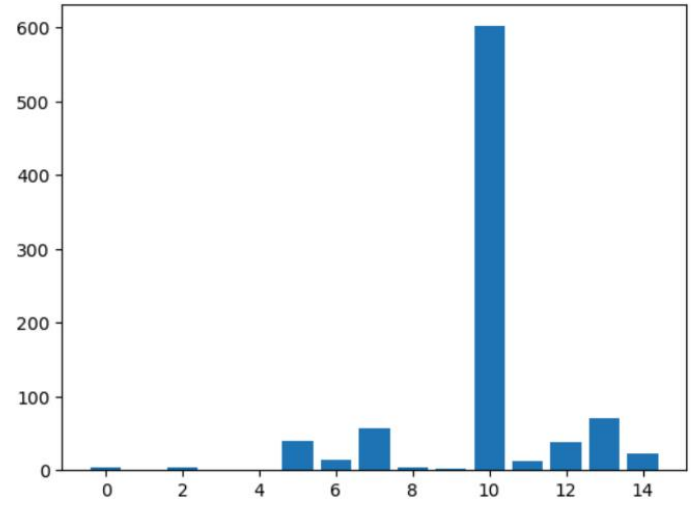
c) Feature Selection and Analysis

Firstly, these data preprocessing steps were taken to ensure efficient building and evaluation of machine learning models:

- Categorical Label encoding: is used to convert categorical data into numerical data which makes them appropriate for ANOVA.
- Scaling of features' values: would enable better performance of machine learning algorithms by handling features with same metric and similar scale.
- Train-test split: to split the data into training and testing sets, allowing for unbiased assessment of the models' performance, and prevent overfitting or underfitting.

Despite considering that Chi-square is commonly used for feature selection with categorical data, we chose to employ ANOVA after label encoding our categorical features. This approach was chosen to determine the most important predictors in our dataset by utilizing the numerical representation of our features to evaluate their variance. ANOVA offers a strong substitute for feature selection, guaranteeing that, even after transformation, we can still assess the significance of our categorical variables. Furthermore, the dataset also contains negative value, fundamentally conflict with the assumptions of the Chi-squared test for feature selection. Therefore, ANOVA might be the best choice for this dataset.

After that, we then proceeded to do features selection based on ANOVA (Analysis of Variance) Score.



	Score	Feature Name
0	601.815465	duration
1	70.061518	previous
2	57.439236	contact
3	40.320214	housing
4	37.537236	pdays
5	22.809634	poutcome
6	14.512146	loan
7	13.141424	campaign
8	4.683322	education
9	4.511153	job

Figure 5: Features' ANOVA Scores

From Figure 5, the five features which have the most vital positions in predicting the target based on the features scores are:

- The ‘duration’ feature the highest feature score of 601.82, indicating that it would be the most important feature for the models’ classifications.
- Following far behind are the features ‘previous’, ‘contact’, ‘housing’ , and ‘pdays’ with quite similar features scores at around 70.06, 57.44, 40.32, and 37.54, respectively. These features even though do not have the exceptionally high score of ‘duration’ feature, they would still have a considerable impact on the dataset’s target.

Therefore, with the features scores from ANOVA, the top five features for building our models are ‘duration’, ‘contact’, ‘previous’, ‘housing’, and ‘pdays’.

d) Independence Assumption in Naïve Bayes

After choosing the top five features for models' constructions, the independence assumption between these features needed to be identified. This would enable Naive Bayes algorithm to calculate posterior probability efficiently in a quick and scalable manner. Using features’ correlation heatmap, we could verify the assumption.

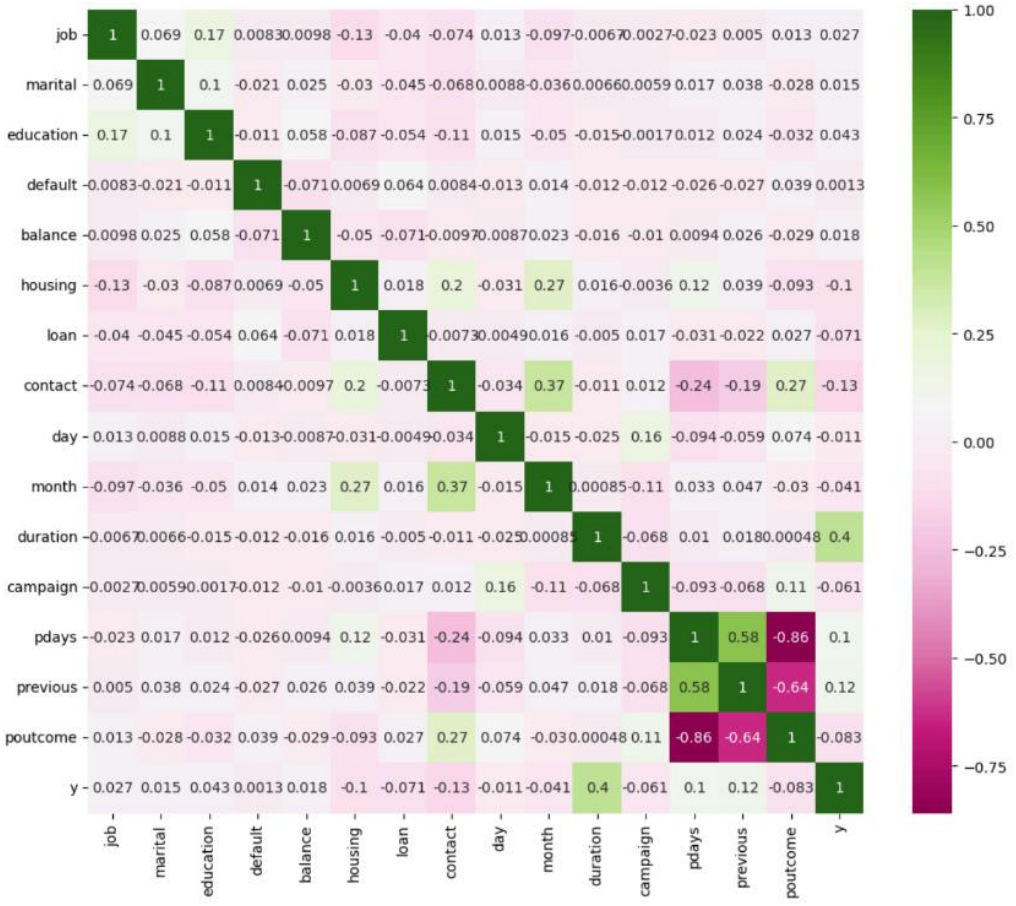


Figure 6: Correlation Heatmap of Dataset's Features

Based on Figure 6, coefficient of correlation between the five features are relatively low, ranging between -0.25 and 0.25 indicating the features' weak relationships. There is only one moderate relationship between features 'previous' and 'pdays' with a coefficient of correlation at 0.58. As a result, the five chosen features are statistically independent given the class label. Therefore, the independence assumption in Naïve Bayes is met, and the model building process can be implemented.

e) Naïve Bayes Model Building and Evaluation

The Gaussian Naïve Bayes implementation of the Naïve Bayes algorithm was applied on the chosen features. Following is the evaluation measures for the model:

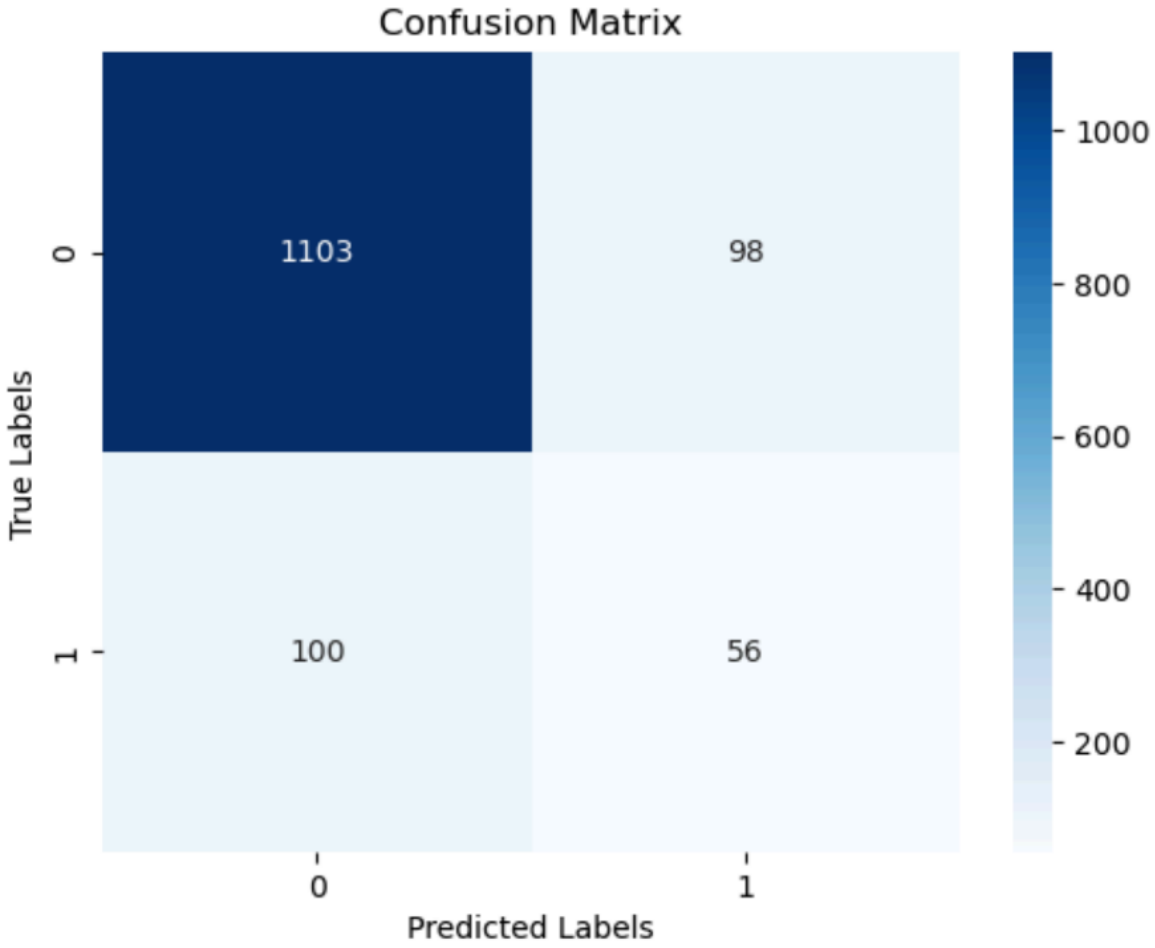


Figure 7: Confusion Matrix of Naïve Bayes Model

Firstly, the model's confusion matrix was generated. Based on this Figure 7:

- True Negative (TN): 1103 instances were 'no' (negative) and correctly classified 'no' (negative).
- False Positive (FP): 98 instances were actually 'no' (negative) and incorrectly classified as 'yes' (positive).
- False Negative (FN): 100 instances were actually 'yes' (positive) and incorrectly classified as 'no' (negative).

- True Positive (TP): 56 instances were 'yes' (positive) and correctly classified as 'yes' (positive).

Accuracy: 0.8540899042004422
Classification Report:

	precision	recall	f1-score	support
0	0.92	0.92	0.92	1201
1	0.36	0.36	0.36	156
accuracy			0.85	1357
macro avg	0.64	0.64	0.64	1357
weighted avg	0.85	0.85	0.85	1357

Figure 8: Classification Report of Naïve Bayes Model

Additionally, classification report (Figure 8) also gave us some insights into the Naïve Bayes model's performance:

- Accuracy = (TN+TP) / (TN+FP+TP+FN) = 0.85, implies that 85% of all instances were correctly classified by the model. However, because there is a class imbalance where 4000 "no" instances against 521 "yes" instances, this statistic could not be sufficient.
- Precision = TP / (TP+FP), implying the proportion of positive predictions which are correct.
    - Class 0 ('no'): 0.92
    - Class 1 ('yes'): 0.36
- Recall = TP / (TP+FN), implying the percentage of actual positives that the model was able to identify.
    - Class 0 ('no'): 0.92
    - Class 1 ('yes'): 0.36
- F1-Score = 2*Precision*Recall / (Precision + Recall), implying the harmonic mean of precision and recall, which create a balance between those two scores. Due to the imbalance of the class distribution, f1-score is particularly helpful in comparing models.
    - Class 0 ('no'): 0.92
    - Class 1 ('yes'): 0.36

Therefore, excellent performance can be seen for class 0 ('no') with all scores being 0.92 consistently, while this is not the case for class 1 ('yes'). Compared to its counterpart, class 1's model scores are relatively low with 0.36 for precision score, recall score, and f1-score. This indicates that while the model is efficient in predicting when a client would not subscribe to the bank term deposit, it struggles to identify instances where clients would subscribe to the term deposit.

f) KNN Model Building and Evaluation

The K-Nearest Neighbour machine learning algorithm was also applied to the dataset's selected features. To begin with, the suitable number of nearest neighbours or k need to be identified.

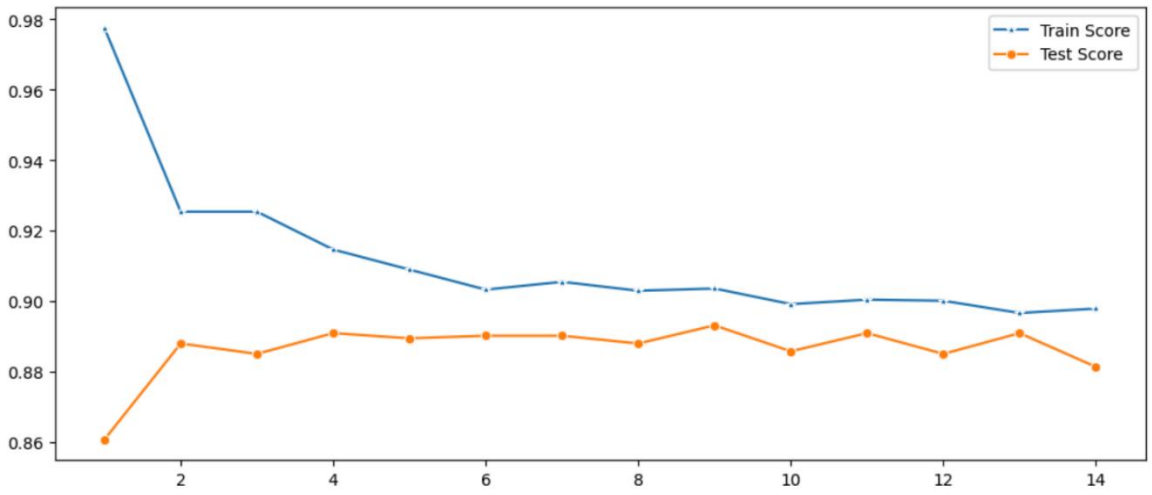


Figure 9: KNN Model for Range of K Values

From the figure above (Figure 9), the model train scores and test scores at different k point from 1 to 14 was produced. It can clearly be seen that at k = 13, the gap between train score and test score is the smallest. This indicates a good balance between bias and variance which would help the model to avoid overfitting or underfitting.

Then, the KNN model would be evaluated based on confusion matrix and classification report.

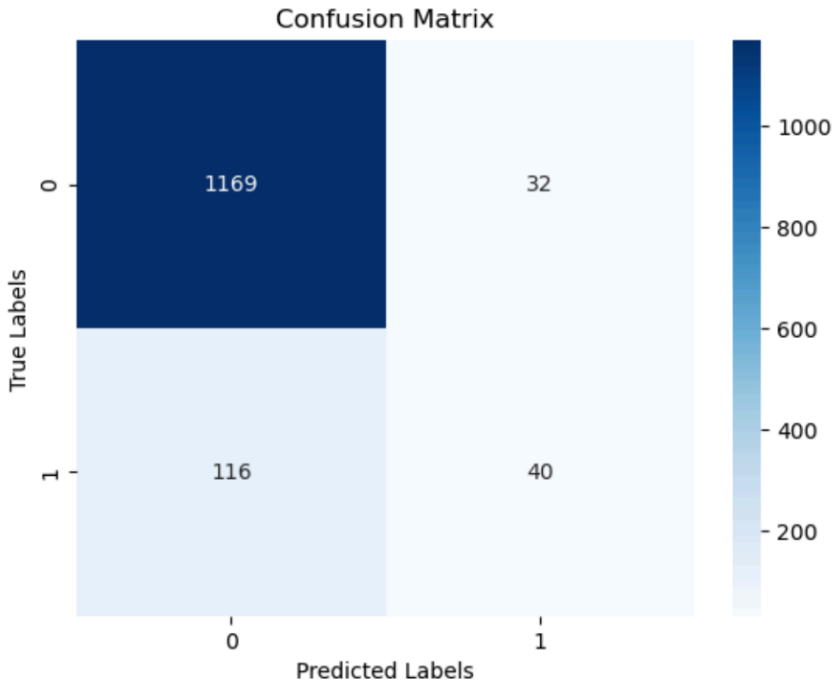


Figure 10: Confusion Matrix of KNN Model

Firstly, the model's confusion matrix can be seen from Figure 10.

- True Negative (TN): 1169 instances were 'no' (negative) and correctly classified 'no' (negative).
- False Positive (FP): 32 instances were actually 'no' (negative) and incorrectly classified as 'yes' (positive).
- False Negative (FN): 116 instances were actually 'yes' (positive) and incorrectly classified as 'no' (negative).
- True Positive (TP): 40 instances were 'yes' (positive) and correctly classified as 'yes' (positive).

```
Accuracy: 0.8909358879882093
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.97      0.94      1201
           1       0.56      0.26      0.35       156

    accuracy                           0.89      1357
   macro avg       0.73      0.61      0.65      1357
weighted avg       0.87      0.89      0.87      1357
```

Figure 11: Classification Report of KNN Model

Additionally, classification report was presented on the figure above (Figure 11).

- Accuracy = (TN+TP) / (TN+FP+TP+FN) = 0.89, implies that 89% of all instances were correctly classified by the model.
- Precision = TP / (TP+FP), implying the proportion of positive predictions which are correct.
    - Class 0 ('no'): 0.91
    - Class 1 ('yes'): 0.56
- Recall = TP / (TP+FN), implying the percentage of actual positives that the model was able to identify.
    - Class 0 ('no'): 0.97
    - Class 1 ('yes'): 0.26
- F1-Score = 2*Precision*Recall / (Precision + Recall), implying the harmonic mean of precision and recall, which create a balance between those two scores.
    - Class 0 ('no'): 0.94
    - Class 1 ('yes'): 0.35

As a result, with a precision score of 0.91, a recall score of 0.97, and a f1-score of 0.94, class 0 ('no') exhibits great performance; in contrast, class 1 ('yes') does not. Class 1's model scores are comparatively low with 0.56 precision, 0.26 recall, and 0.35 f1-score. This suggests that although the model is effective in classifying instances when a client

would not subscribe to the bank term deposit, it finds it challenging to classify the occasions in which a client would subscribe to the term deposit.

g) Model Comparison

After gathering all the metrics for both models, these are the comparisons between Naïve Bayes (NB) and K-Nearest Neighbour (KNN) models:

- Accuracy: KNN model has a noticeably higher overall accuracy (0.89) compared to that of NB model (0.85)
- Precision: For the majority class 0, KNN and NB models have precision that is almost similar (0.91 and 0.92), but for the minority class 1, KNN model has a significantly greater precision (0.56 > 0.36)
- Recall: Compared to NB model, KNN model has much lower recall for the minority class 1 (0.26 < 0.36) but higher recall for the majority class 0 (0.97 > 0.92)
- F1-Score: Compared to NB model, KNN model has a little higher F1-score for the majority class 0 (0.94 > 0.92) and a quite similar F1-score for the minority class 1 (0.35 compared to 0.36).

Overall, the scores' differences can be interpreted as:

- Class 0 ('no'): Both models deliver excellent results. For this class, KNN model performs slightly better when considering recall and F1-score.
- Class 1 ('yes'): Both models struggled with class 1. The KNN model has a lower recall (more false negatives) but a greater precision (fewer false positives). It is therefore more cautious in its 'yes' classification but misses a greater number of genuine 'yes' instances. On the other hand, NB model has lower precision (more false positives) but higher recall (fewer false negatives). Thus, it generates more inaccurate "yes" predictions even while it classifies more real "yes" instances.

In conclusion, the NB model may be more helpful if capturing instances of the minority class ('yes') is crucial despite its lower precision since it has greater overall accuracy and recall for this class. The KNN model classified 'yes' with a more conservative bias since it has more evenly distributed accuracy and recall for the minority class 1, which may result in fewer false positives. Even though NB has less precision, it might be chosen if the main objective is to collect the greatest number of 'yes' instances (greater recall). If classifying 'yes' requires fewer false positives, KNN may be a better fit.

Part B: Exploring Artificial Neural Networks

a) Activation Function and Learning Rate in MLP

Both Activation Function and Learning Rate have a critical importance for Multi-Layer Perceptrons model constructing.

- Activation Function is the sigmoid function that is commonly used to model a situation that is as similar as human neurons system. The neurons in the hidden layer are enabled only when the input they receive is above a certain threshold value which can represented with the following formula:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

- Learning Rate specifies how large the "steps taken" in the process of adjusting weight throughout the iterative optimization process. By taking greater steps, a higher learning rate speeds up the learning process, but it also increases the chance of exceeding ideal weight values, which might result in less accuracy.

b) Baseline Model with MLPClassifier

We used a single hidden layer with 25 neurons, with the iteration is 200 to create the baseline model. Using 10-fold cross-validation, we found the best number of iterations that produced the highest accuracy.

```
>0.890
score_test: 0.874
>0.889
score_test: 0.883
>0.893
score_test: 0.889
>0.888
score_test: 0.903
>0.885
score_test: 0.885
>0.887
score_test: 0.883
>0.888
score_test: 0.892
>0.888
score_test: 0.874
>0.893
score_test: 0.869
>0.892
score_test: 0.889
MLP Accuracy: 88.41%
Mean Testing set Accuracy: 0.889 (0.000)
Mean Train set Accuracy: 0.892 (0.000)
```

Figure 12: MLP Scores with 10-fold Cross-validation

After using 10-Fold Cross-validation (Figure 12), basically splitting the dataset into 10 small train set and test set, it is easily to observe that the best accuracy score that the model can achieve is 0.903 (approximately 90.3%) and the lowest accuracy is 0.869 (approximately 86.9%). The difference between the highest accuracy and the lowest accuracy is only 0.034, which is inconsiderable. Furthermore, the average accuracy of the model after ten times testing is 88.41%, suggesting that the model is good (industry standard is between 70% and 90%).

To determine the best number of iterations that gives the highest accuracy, we need to base on loss value curve which will be shown below.

c) Tracking Loss Value

Moreover, must track the loss as a function of the number of iterations and enable the loss value to be shown on the training section.

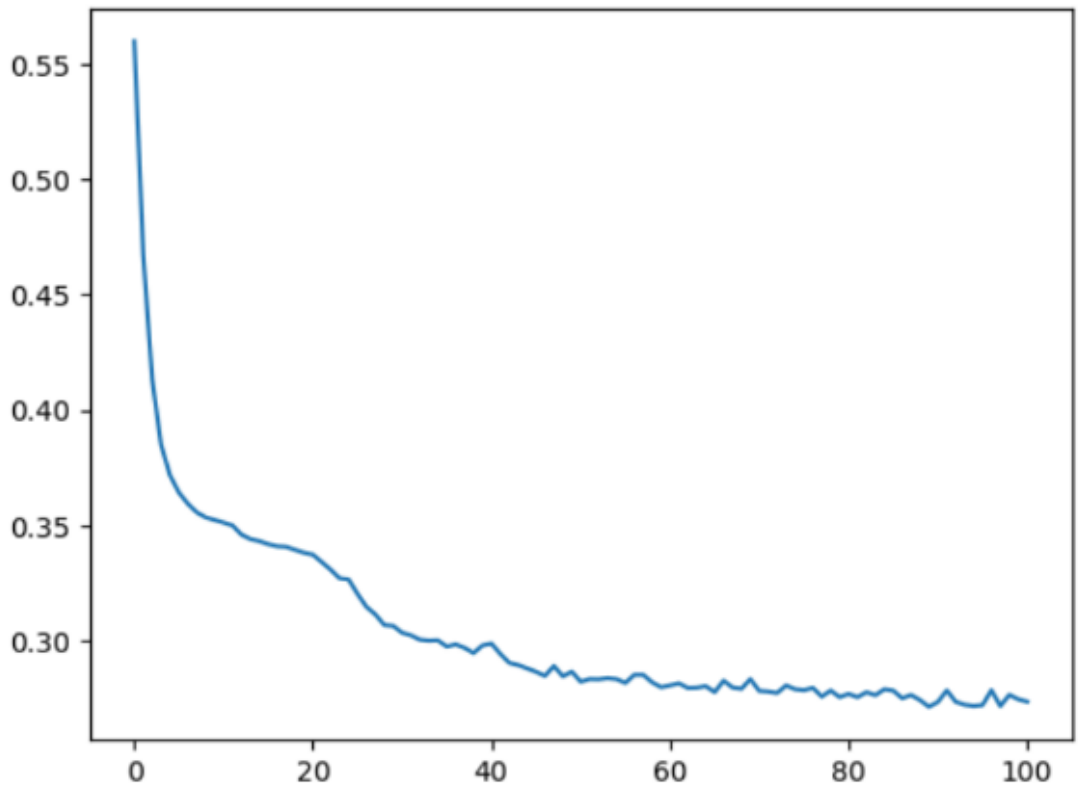


Figure 13: MLP model's Loss Curve

From Figure 13, it can clearly be seen that the loss curve declines from the first iteration until iteration 100, which can be understand that the model keep learning and decrease the loss value. After iteration 100, there is no further decline, indicates that the learning process has stalled or there has been no significant progress, implying no improvement in accuracy. Therefore, the best number of iterations is confirmed to be around 100, with a stable accuracy score of approximately 88%.

Apply the best number of iterations to the final to model of k-neuron = 25, with iteration = 100, the MLP model provide baseline Confusion Matrix and Classification report that can be used for later comparisons.

```
Text(50.722222222222214, 0.5, 'True Labels')
```

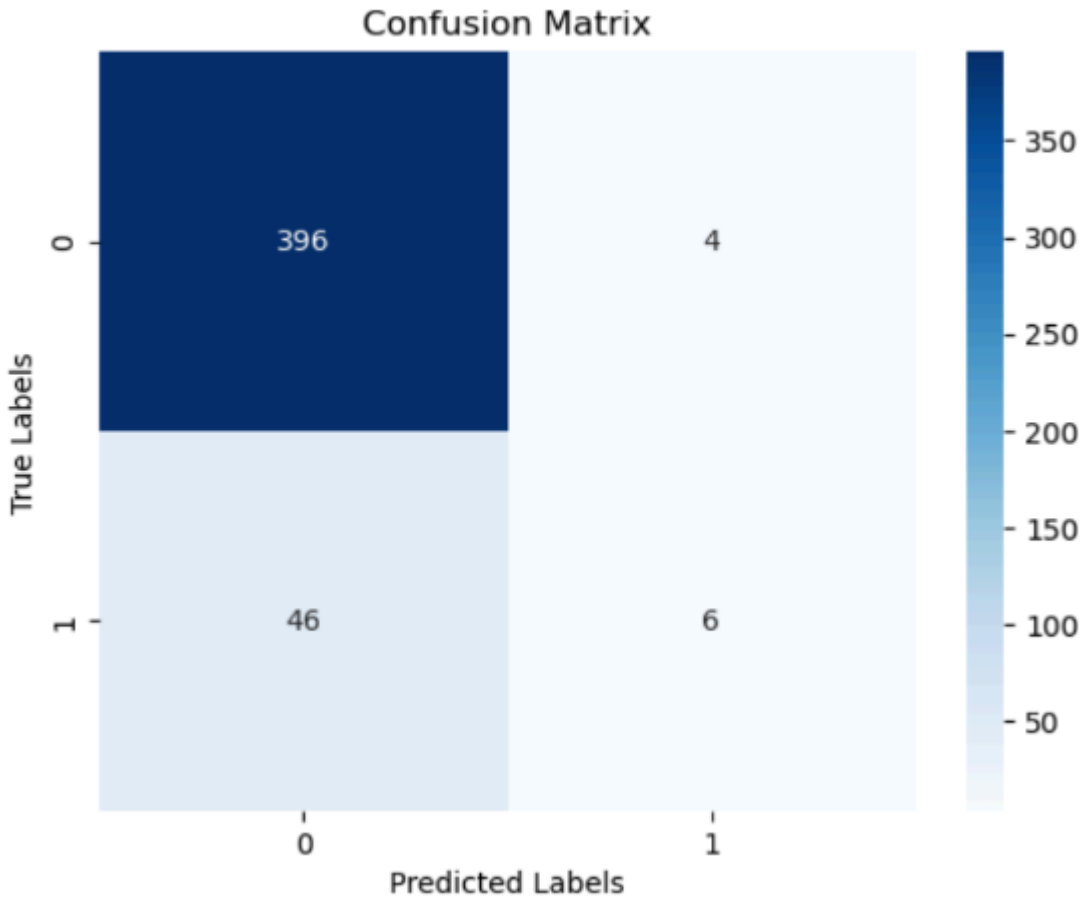


Figure 14: Confusion Matrix of MLP with Single layer model

This Figure 14 shows the confusion matrix of MLP single layer model.

- True Negative (TN): 396 instances were 'no' (negative) and correctly classified 'no' (negative).
- False Positive (FP): 4 instances were actually 'no' (negative) and incorrectly classified as 'yes' (positive).
- False Negative (FN): 46 instances were actually 'yes' (positive) and incorrectly classified as 'no' (negative).
- True Positive (TP): 6 instances were 'yes' (positive) and correctly classified as 'yes' (positive).

```
Accuracy: 0.8893805309734514
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       400
           1       0.60      0.12      0.19        52

    accuracy                           0.89       452
   macro avg       0.75      0.55      0.57       452
weighted avg       0.86      0.89      0.85       452
```

Figure 15: Classification Report of MLP with Single layer model

Moreover, classification report of this model can be seen from the above figure (Figure 15):

- Accuracy = (TN+TP) / (TN+FP+TP+FN) = 0.89, implies that 89% of all instances were correctly classified by the model.
- Precision = TP / (TP+FP), implying the proportion of positive predictions which are correct.
    - Class 0 (‘no’): 0.90
    - Class 1 (‘yes’): 0.60
- Recall = TP / (TP+FN), implying the percentage of actual positives that the model was able to identify.
    - Class 0 (‘no’): 0.99
    - Class 1 (‘yes’): 0.12
- F1-Score = 2*Precision*Recall / (Precision + Recall), implying the harmonic mean of precision and recall, which create a balance between those two scores.
    - Class 0 (‘no’): 0.94
    - Class 1 (‘yes’): 0.19

d) Experimenting with Two Hidden Layers

Then, we try two hidden layers and split the total number of neurons across the two layers to identify the split that provides the best classification accuracy. Next, we get a 25 by 2 table showing the classification accuracy and the combination of neurons applied. The second column provides the classification accuracy.

	Neuron Split (L1, L2)	Accuracy
0	24, 1	0.884956
1	23, 2	0.884956
2	22, 3	0.884956
3	21, 4	0.887168
4	20, 5	0.882743
5	19, 6	0.867257
6	18, 7	0.876106
7	17, 8	0.887168
8	16, 9	0.884956
9	15, 10	0.873894
10	14, 11	0.871681
11	13, 12	0.882743
12	12, 13	0.884956
13	11, 14	0.853982
14	10, 15	0.876106
15	9, 16	0.873894
16	8, 17	0.838496
17	7, 18	0.867257
18	6, 19	0.889381
19	5, 20	0.880531
20	4, 21	0.896018
21	3, 22	0.884956
22	2, 23	0.884956
23	1, 24	0.884956

Figure 16: Table of MLP with Two layers model's Classification Accuracy

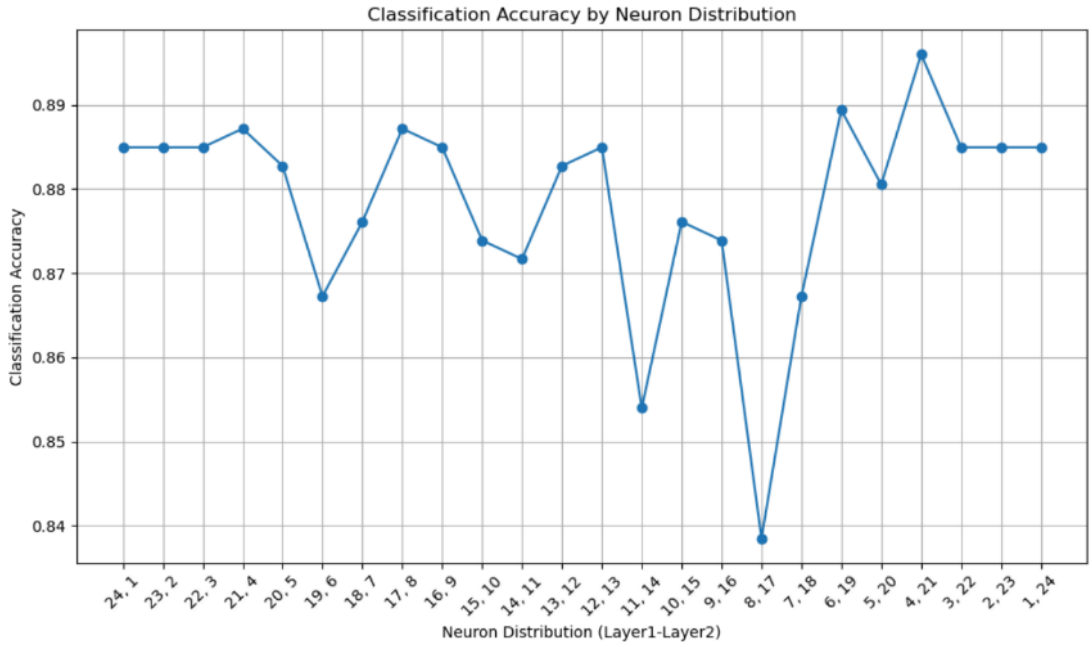


Figure 17: Classification Accuracy by Neuron Distribution

Based on these figures (Figure 16 and Figure 17), the accuracy for all neuron split combinations ranges from almost 0.84 to over 0.89, representing a considerable gap between these accuracy rates. Although, the neuron split across two hidden layers was used in experiments to show that the combination of (4, 21) produced the best classification accuracy at approximately 0.89. This shows that a larger second hidden layer is more effective in detecting the underlying patterns in the data for this dataset.

Then, to evaluate this experiment with two hidden layers, confusion matrix and classification report were presented.

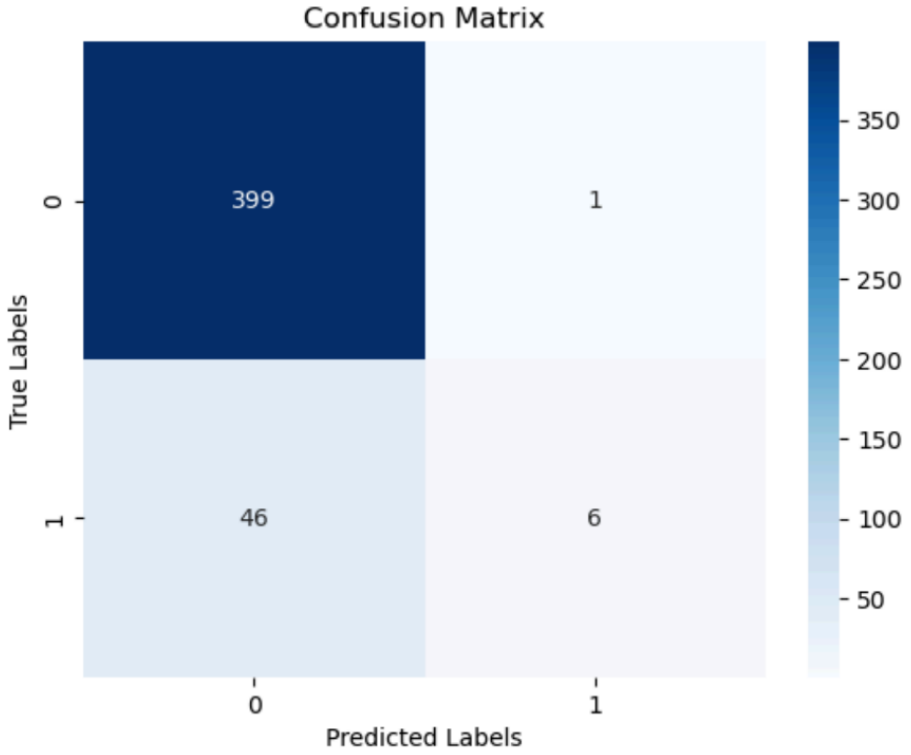


Figure 18: Confusion Matrix of MLP with Two layers model

Based on this confusion matrix (Figure 18), it can clearly be seen that:

- True Negative (TN): 399 instances were 'no' (negative) and correctly classified 'no' (negative).
- False Positive (FP): 1 instance were actually 'no' (negative) and incorrectly classified as 'yes' (positive).
- False Negative (FN): 46 instances were actually 'yes' (positive) and incorrectly classified as 'no' (negative).
- True Positive (TP): 6 instances were 'yes' (positive) and correctly classified as 'yes' (positive).

```
Accuracy: 0.8960176991150443
Classification Report:
              precision    recall  f1-score   support

           0       0.90      1.00      0.94       400
           1       0.86      0.12      0.20        52

    accuracy                           0.90       452
   macro avg       0.88      0.56      0.57       452
weighted avg       0.89      0.90      0.86       452
```

Figure 19: Classification Report of MLP with Two layers model

Moreover, from the above classification report (Figure 19), these scores are shown:

- Accuracy = (TN+TP) / (TN+FP+TP+FN) = 0.90, implies that 90% of all instances were correctly classified by the model.
- Precision = TP / (TP+FP), implying the proportion of positive predictions which are correct.
    - Class 0 (‘no’): 0.90
    - Class 1 (‘yes’): 0.86
- Recall = TP / (TP+FN), implying the percentage of actual positives that the model was able to identify.
    - Class 0 (‘no’): 1.00
    - Class 1 (‘yes’): 0.12
- F1-Score = 2*Precision*Recall / (Precision + Recall), implying the harmonic mean of precision and recall, which create a balance between those two scores.
    - Class 0 (‘no’): 0.94
    - Class 1 (‘yes’): 0.20

e) Explaining Accuracy Variation

From the table in figure, accuracy varied between different splits of neurons across the two hidden layers. There are a few potential causes for this variation:

- There is no fixed standard for model construction:
    - The performance of the model can vary depending on the neuron split used due to the flexibility of neural networks.
    - Testing several configurations is necessary to find the optimum one for the dataset.
- Parameters are sensitive, and it influenced how well the network learns from the data.
- We don't know how many k-neuron for each layer that can help the model achieve the best accuracy, so we test, and it cause the variation in accuracy for each test times.

f) Comparing MLP Classifier

Firstly, we need to compare the MLP classifier with a single hidden layer (MLP1) with the MLP classifier with two hidden layers (MLP2). Confusion matrix and classification report was used to determine the accuracy of both models. These are the scores differences between the two models:

- Accuracy: MLP1 and MLP2 models have similar accuracy at 0.89 and 0.90 respectively.
- Precision: MLP2 performs much better (0.86) than MLP1 (0.60) for class 1 ('yes'). While, MLP2 has a similar accuracy with MLP1 for class 0 ('no') at 0.90.

- Recall: For class 0 (‘no’), the recall values of the two models are almost identical same with MLP1 scoring 0.99 and MLP2 scoring a perfect 1.00, whereas class 1 ('yes') has relatively poor recall of 0.12 for both models.
- F1-Score: For class 1 ('yes'), MLP2's F1-score of 0.20 is slightly greater than MLP1's of 0.19. For class 0 ('no'), both models have the same F1-score of 0.94.

Comparing MLP1 and MLP2, it can be shown that despite both models have equivalent overall accuracy, MLP2 has a slight lead (0.90 compared to 0.89). For class 1 ('yes') MLP2 exhibits a notable increase in accuracy, suggesting improved ability in accurately recognizing positive instances. On the other hand, recall for class 1 ('yes') is a problem shared by both models, indicating the class imbalance of the dataset and the complexity of the 'yes' class in general. Although the improvement is small, the F1-score for class 1 is slightly higher for MLP2, confirming that MLP2 is a little more balanced in handling false positives and false negatives. In terms of all scores, both models function equally well for class 0 ('no').

It is justified to compare MLP1 to KNN and NB models since it shows almost the same overall accuracy as MLP2, maintaining objectivity and concentrating the comparison on meaningful changes in model architecture rather than performance. Moreover, contrasting MLP1, which has a more straightforward design, may provide insights on whether neural networks with more complexity have real advantages over more conventional models like KNN and NB.

The scores between KNN, NB, and MLP1 models would be compared as:

- Accuracy: When compared to NB with 0.85 of accuracy, KNN and MLP1 both exhibit greater accuracy at 0.89.
- Precision: MLP1 (0.60) surpasses KNN (0.56) and NB (0.36) for class 1 ('yes'). However, NB (0.92) performs slightly greater than KNN (0.91) and MLP1 (0.90) for class 0 ('no').
- Recall: At 0.99, MLP1 has the highest recall for class 0 ('no,') compared to 0.97 for KNN and 0.92 for NB. In contrast, NB and KNN both exhibit greater recall for class 1 ('yes') than MLP1 (0.12).
- F1-Score: For class 0 ('no'), MLP1 and KNN have the same F1-score of 0.94, although NB has a bit lower F1-score of 0.92. Both NB and KNN have close F1-scores for class 1 ('yes'), 0.36 and 0.35, respectively, which are both better than MLP1's 0.19.

In conclusion, it is clear from the comparative study of accuracy, precision, recall, and F1-score that each model of the three models KNN, NB, and MLP1 has advantages and disadvantages for certain scores and classes. MLP1 exhibits better precision for class 1 ('yes'); nonetheless, KNN performs better overall and balances better across various

parameters than both NB and MLP1. KNN obtains comparable recall rates for both classes and impressive accuracy for class 0 ('no'). Its performance is balanced over a range of assessment criteria, making it the most consistent of the three types. MLP1's overall performance is compromised by its relatively poor performance in other measures, even if it excels in precision for class 1 and recall for class 0. KNN therefore proves to be the best option for this dataset due to its balanced performance in all evaluated metrics.