Student ID Number:

16946615 & 22180242

Assignment Two

Semester 1 2024

Student Names: **Juchang Kim & Jinzhi Chen**

Student IDs: **22180242 & 16946615**

PAPER NAME: **Foundations of Data Science**

PAPER CODE: COMP615

Due Date: 2nd June 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.**

Contents

Part A. Predicting Bank Marketing Campaign Outcomes .... page. 5

a) Explain the KNN and Naïve Bayes Algorithms .... page. 5

b) Perform Exploratory Data Analysis (EDA) .... page. 8

- Figure A.b.1 The Summary of Dataset .... page. 8
- Figure A.b.2 Information of the Dataset page.9
- Figure A.b.3 Number of Missing Values page.10
- Figure A.b.4 Number of Null Values page.10
- Figure A.b.5 Number of Duplicated Rows page.11
- Figure A.b.6 Box plot with Numerical columns and Target Column page.12
- Figure A.b.7 Number of Outliers page.12
- Figure A.b.8 Distribution of each Categorical Column with Target Class by Box Plot page.14
- Figure A.b.9 Distribution of each Numerical Column with Target Class by Histogram page.18
- Figure A.b.10 Numerical Attributes Correlation Heatmap page.21
- Figure A.b.11 Distribution of the Target Class page.22

c) Feature Selection and Analysis page.24

- Figure A.c.1 Encoded Categorical Attributes without Target Class and Duration Attribute page.26
- Figure A.c.2 List of the P-value of all Features page.27
- Figure A.c.3 Correlation Heatmap With Selected Features page.28

- Figure A.c.4 Correlation Heatmap With Adjusted Selected Features page.30
- Figure A.c.5 The List of Adjusted Selected Features page.30
- Figure A.c.6 Histogram Distribution of Pday by Class page.32
- Figure A.c.7 Histogram Distribution of Campaign by Class page.33
- Figure A.c.8 Histogram Distribution of Housing by Class page.34
- Figure A.c.9 Histogram Distribution of Contact by Class page.35
- Figure A.c.10 Histogram Distribution of Loan by Class page.36

d) Independence Assumption in Naïve Bayes page.37

- Figure A.d.1 Correlation Heatmap with Selected Features page.38

e) Naive Bayes Model Building and Evaluation page.40

- Figure A.e.1 Accuracy, Text of Confusion Matrix, Classification Report of Naive Bayes model page.40
- Figure A.e.2 Confusion Matrix of The Naive Bayes Model page.41
- Figure A.e.3 K-Fold Cross Validation Scores in NB Model page.43
- Figure A.e.4 Graph of K-Fold Cross Validation Scores in NB Model page.44

f) KNN Model Building and Evaluation page.46

- Figure A.f.1 KNN Train Score, Test Score and Mean of CV Score for Different K Values page.47
- Figure A.f.2 KNN Accuracy for Different K Values page.48
- Figure A.f.3 KNN Classification Report page.50
- Figure A.f.4 Confusion Matrix KNN Classification Model when K is 2 .... page.50
- Figure A.f.5 List of K-Fold Cross Validation Accuracy Score when K is 2 ..... page.52

g) Model Comparison …………………………………………………………………………………. page.54

Part B. Exploring Artificial Neural Networks ……………………………………………………. page.57

a) Activation Function and Learning Rate in MLP …………………………………. page.57

- Figure B.a.1 An example of MLP …………………………………………………………………. page.57

b) Baseline Model with MLPClassifier ……………………………………………………. page.59

- Figure B.b.1. Confusion Matrix for 1 layer MLP ……………………………………………. page.60

c) Tracking Loss Value ………………………………………………………………………….. page.61

- Figure B.c.1 Loss Value Graph ……………………………………………………………………. page.61

d) Experimenting with Two Hidden Layers ……………………………………………. page.63

- Figure B.d.1 Results of testing different numbers of neurons in MLP ………… page.63
- Figure B.d.2 Confusion Matrix for 2 layers MLP ……………………………………………. page.65

e) Explaining Accuracy Variation ……………………………………………………………. page.67

f) Comparing MLP Classifier Performance ……………………………………………. page.69

- Figure B.f.1 Confusion Matrix of all models …………………………………………. page.70
- Figure B.f.2 MLP models using entire dataset ……………………………………… page.71

Table 1. Performance of Models ……………………………………………………………………. page.69

Reference List ………………………………………………………………………………………………. page.73

Part A (Predicting Bank Marketing Campaign Outcomes)

This part of the assignment is concerned with predicting the outcome of direct bank marketing campaigns (phone calls) of Portuguese banking. The dataset (Bank.zip) contains 17 attributes for which outcomes of subscribing to a term deposit (yes/no) are known. You are required to build models using the K-Nearest Neighbors (KNN) and Naïve Bayes (NB) algorithms.

a) Explain the KNN and Naïve Bayes Algorithms [10 marks]

In your own words, explain how each of the KNN and Naïve Bayes algorithms work.

K-Nearest Neighbors (KNN) Algorithm

The K-Nearest Neighbors (KNN) algorithm is a simple, non-parametric, and instance-based learning algorithm used for classification and regression tasks. Here's how KNN works in the context of predicting the outcome of direct bank marketing campaigns:

1. **Data Preparation**: The dataset consists of 16 input variables and one output variable (y), which indicates whether the client subscribed to a term deposit (yes/no). Before applying KNN, the dataset needs to be preprocessed to handle categorical variables, missing values (if any), and normalization of numerical features.
2. **Distance Measurement**: KNN relies on calculating the distance between data points. The most common distance metric used is Euclidean distance. For each test instance, the algorithm calculates the distance to all training instances.
3. **Selecting Neighbors**: The value of k is chosen, which represents the number of nearest neighbors to consider. For each test instance, the k training instances with the smallest distance to the test instance are selected as neighbors.
4. **Voting Mechanism**: For classification, KNN uses a majority voting mechanism. The class (yes/no) of the test instance is determined by the majority class among the k nearest neighbors. For example, if k=7 and 5 out of 7 neighbors have the class "yes", the test instance is classified as "yes".
5. **Prediction**: The predicted class for each test instance is based on the

majority vote from its nearest neighbors.

In the context of the bank marketing dataset, KNN can help identify patterns and similarities between clients who have similar characteristics (e.g., age, job, education) and predict whether a new client will subscribe to a term deposit based on these patterns.

Advantages of KNN:

- Simple and easy to understand.
- No assumptions about the data distribution.
- Effective for small to medium-sized datasets.

Disadvantages of KNN:

- Computationally expensive for large datasets.
- Sensitive to the choice of k and the distance metric.
- May be affected by irrelevant or redundant features.

Naïve Bayes (NB) Algorithm

Naïve Bayes (NB) is a probabilistic classification algorithm based on Bayes' theorem with the assumption of independence between the features. Here's how Naïve Bayes works in the context of predicting the outcome of direct bank marketing campaigns:

1. **Data Preparation**: Similar to KNN, the dataset needs to be preprocessed. However, Naïve Bayes can handle categorical data more naturally by using probabilities for each category.
2. **Bayes' Theorem**: NB uses Bayes' theorem to calculate the probability of a class given a set of features. The theorem is stated as:

$$P(y|X)=\frac{P(X|y)\cdot P(y)}{P(X)}$$

where:

- $P(y|X)$ is the posterior probability of class y given the features X.
- $P(X|y)$ is the likelihood of features X given class y.
- $P(y)$ is the prior probability of class y.
- $P(X)$ is the evidence or marginal likelihood of features X.

3. **Independence Assumption**: Naïve Bayes assumes that the features are conditionally independent given the class label. This simplifies the calculation of the likelihood $P(X|y)P(X|y)$ as the product of the individual feature probabilities:

$$P(X|y)=P(x1|y)\cdot P(x2|y)\cdot \ldots \cdot P(xn|y)P(X|y)=P(x1|y)\cdot P(x2|y)\cdot \ldots \cdot P(xn|y)$$

4. **Probability Calculation**: The algorithm calculates the posterior probability for each class and assigns the class with the highest probability to the test instance.

5. **Prediction**: For each test instance, the class with the highest posterior probability is predicted.

In the context of the bank marketing dataset, Naïve Bayes can help predict whether a client will subscribe to a term deposit by calculating the probabilities based on the client's characteristics and the outcome of previous marketing campaigns.

Advantages of Naïve Bayes:

- Simple and fast to train.
- Works well with high-dimensional data.
- Performs well even with a small amount of training data.

Disadvantages of Naïve Bayes:

- The independence assumption is often unrealistic in practice.
- May perform poorly with highly correlated features.
- Sensitive to zero probability issues, which can be mitigated using techniques like Laplace smoothing.

In summary, the K-Nearest Neighbors (KNN) algorithm makes predictions based on the majority class among the k nearest neighbors of a data point, considering its closest training examples (for this assignment, considering all 16 features after appropriate preprocessing). In contrast, the Naïve Bayes algorithm applies Bayes' theorem with the assumption of independence between features to calculate the probabilities of each class for a given data point, then selects the class with the highest probability. Both algorithms are simple yet powerful, each with unique strengths and weaknesses that make them suitable for different types of datasets.

b) Perform Exploratory Data Analysis (EDA) [10 marks]

Perform EDA and describe your dataset. Explain any pre-processing and data manipulation tasks you performed to prepare your dataset for building your models.

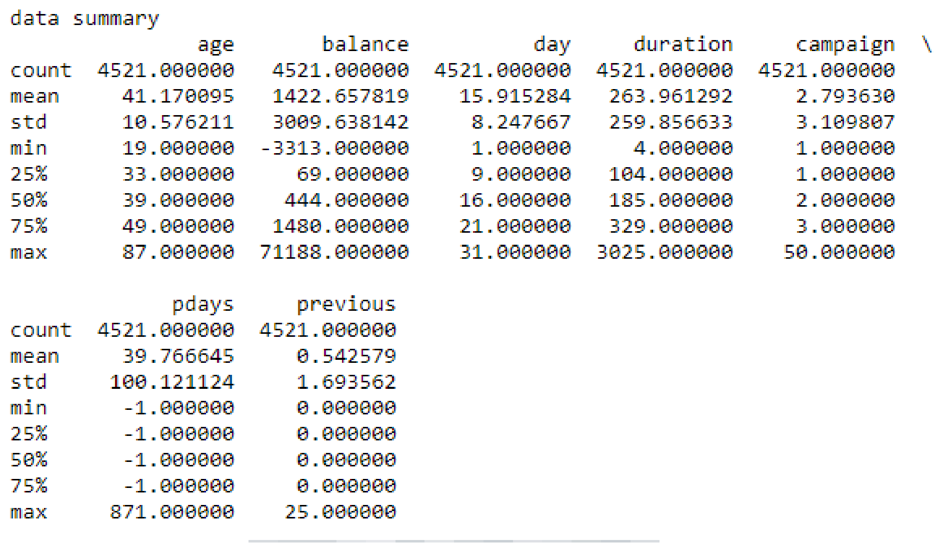
Exploratory Data Analysis (EDA) is an essential step in understanding the characteristics of the dataset before building machine learning models. Here's a step-by-step explanation of the data pre-processing and manipulation tasks typically performed during EDA:

1. **Loading the Dataset:**

   - Begin by loading the dataset into a pandas DataFrame using `pd.read_csv()` or similar functions. Ensure that the dataset is loaded correctly and that the column names and data types are as expected. The original dataset was separated by ";" (while python using "," as default). So we added a `sep=';'` to solve the problem.

2. **Understanding the Dataset:**

   - Use methods like `info()`, and `describe()` to get an overview of the dataset. These methods provide information about the data types, number of entries, missing values, and summary statistics of numerical columns.



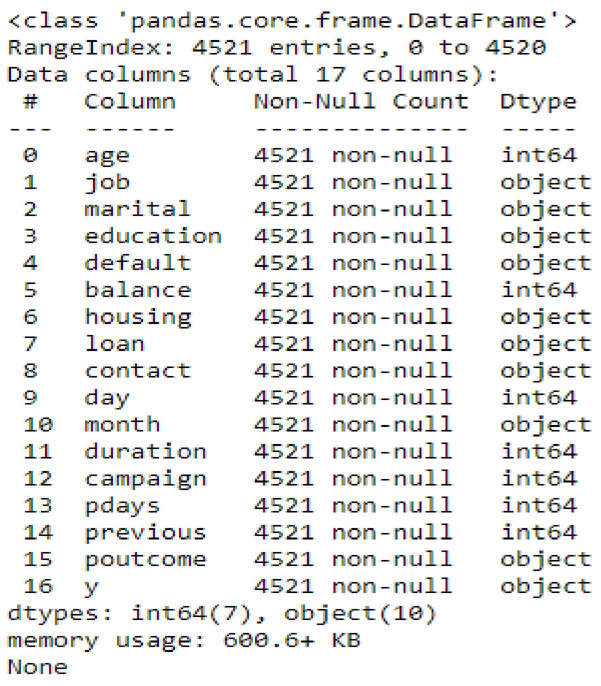
```
data summary
              age       balance          day     duration     campaign  \
count  4521.000000   4521.000000  4521.000000  4521.000000  4521.000000
mean     41.170095   1422.657819    15.915284   263.961292     2.793630
std      10.576211   3009.638142     8.247667   259.856633     3.109807
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 A.b.1 The Summary of Dataset

Explanation:

- In the above figure, there are 4521 rows and 7 numerical columns.

Categorical attributes can not be calculated so the categorical attributes are not showing in the summary.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        4521 non-null   int64 
 1   job        4521 non-null   object
 2   marital    4521 non-null   object
 3   education  4521 non-null   object
 4   default    4521 non-null   object
 5   balance    4521 non-null   int64 
 6   housing    4521 non-null   object
 7   loan       4521 non-null   object
 8   contact    4521 non-null   object
 9   day        4521 non-null   int64 
 10  month      4521 non-null   object
 11  duration   4521 non-null   int64 
 12  campaign   4521 non-null   int64 
 13  pdays      4521 non-null   int64 
 14  previous   4521 non-null   int64 
 15  poutcome   4521 non-null   object
 16  y          4521 non-null   object
dtypes: int64(7), object(10)
memory usage: 600.6+ KB
None
```

Figure A.b.2 Information of the Dataset

Explanation:

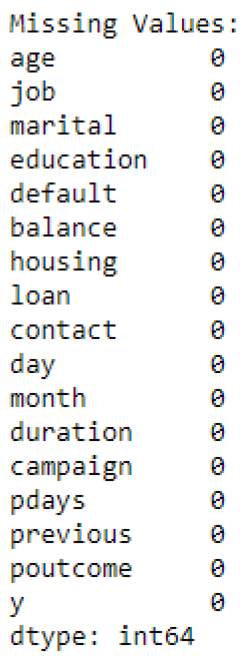
- There are 16 columns which are included in the target class('y'). Also, there are 7 numerical columns which are age, balance, day, duration, campaign, pdays, previous. There are 10 categorical columns which are job, marital, education, default, housing, loan, contact, month, poutcome, y.

3. **Handling Missing and Null Values:**

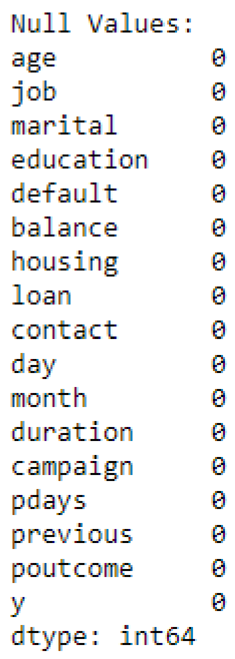
- Check for missing and null values in the dataset using methods like `isnull()` and `sum()`. If there are any missing and null values, decide on an appropriate strategy to handle them:

    - Imputation: Replace missing values or fill null values with a suitable statistic such as the mean, median, or mode of the column.

- Deletion: Remove rows or columns with missing and null values if they are negligible or cannot be imputed accurately.



```
Missing Values:
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
```



```
Null Values:
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
```

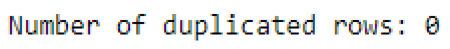
Figure A.b.3 Number of Missing Values   Figure A.b.4 Number of Null Values

Explanation:

- There are no null values and missing values. So, this dataset does not need to handle missing values and null values.

4. **Handling Duplicates:**

- Check for duplicate rows in the dataset using the `duplicated()` method. If duplicate rows are found, decide whether to remove them or keep one instance based on the context of the dataset.



```
Number of duplicated rows: 0
```

Figure A.b.5 Number of Duplicated Rows

Explanation:

There is no duplicated row. So, the dataset does not need to handle duplicate rows.

5. Outlier Detection and Treatment for numerical attributes:

- Identify outliers in numerical features using statistical methods or visualization techniques.

  Decide whether to remove outliers, cap/extreme values, or transform them using appropriate techniques.

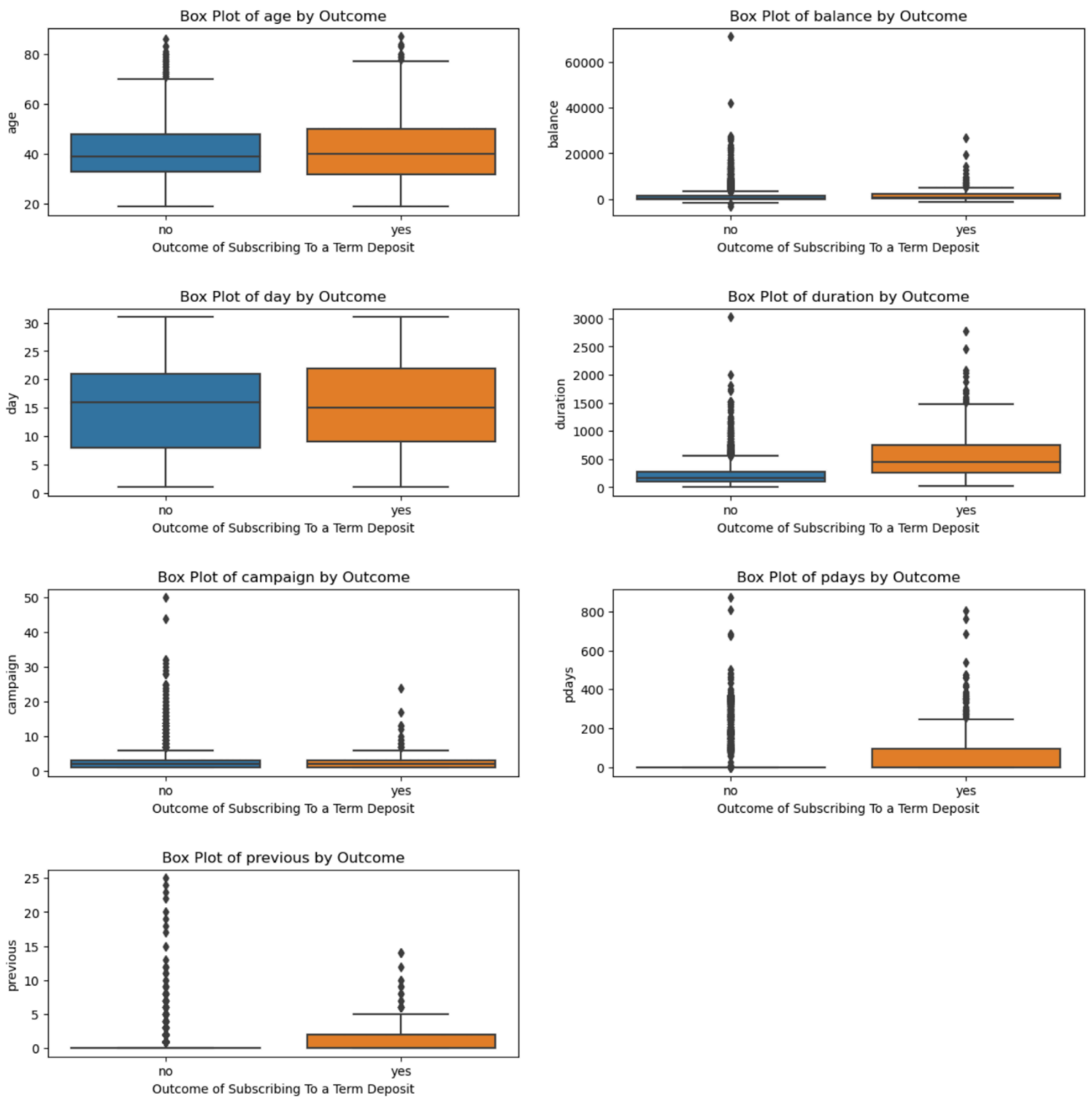
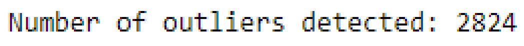


Figure A.b.6 Box plot with Numerical columns and Target Column



```
Number of outliers detected: 2824
```

Figure A.b.7 Number of Outliers

Explanation:

The outliers are detected by the IQR method which identifies the outliers when the data is more than IQR or less than IQR. The IQR is 1.5 times of 3rd quartiles minus 1st quartiles.

There are so many outliers which are more than 2,500. The number of outliers is more than half of the whole dataset. If the outliers were removed, more than half of the dataset would be lost. It can cause inaccurate classification models then, so the outliers should not be removed.

6. **Handling “unknown” data in categorical columns.**

- As described in the dataset explanation text file, there are some “unknown” values in some categorical columns. It would impact on the result of the classification model and be ambiguous to explain the columns and what it is. So, those need to be handled and the “unknown” data should be changed to another name to understand easily in each column.

- The “unknown” data in job column:

  Those are changed to “no mention”. Because there are “unemployed” then, the marketing research, the customer would not mention their job so, the “unknown” data would be “no mention”.

- The “unknown” data in education column:

  Those can be changed to “no education”. The other data values are “primary”, “secondary”, “tertiary”. So, those are all types of education and then “no education” can be another data value.

- The “unknown” data in contact column:

  Those can be changed to “other”. In the column, there are two data values which are cellular and telephone. But there are some other kinds of contact available such as email, offline, visiting a bank and so on. So, the “unknown” data would be “other”.

- The “unknown” data in poutcome column:

  Those can be changed to “no record”. In the column, the content is asking the outcome of previous marketing campaigns. So, there are “success”, “fail”, “other”. So, the “unknown” data can be “no record” for new customers who have not got a marketing campaign.

7. **Exploring Relationships:**
   - Use visualization techniques such as count plots and bar plots to explore relationships between categorical, numerical, target class.

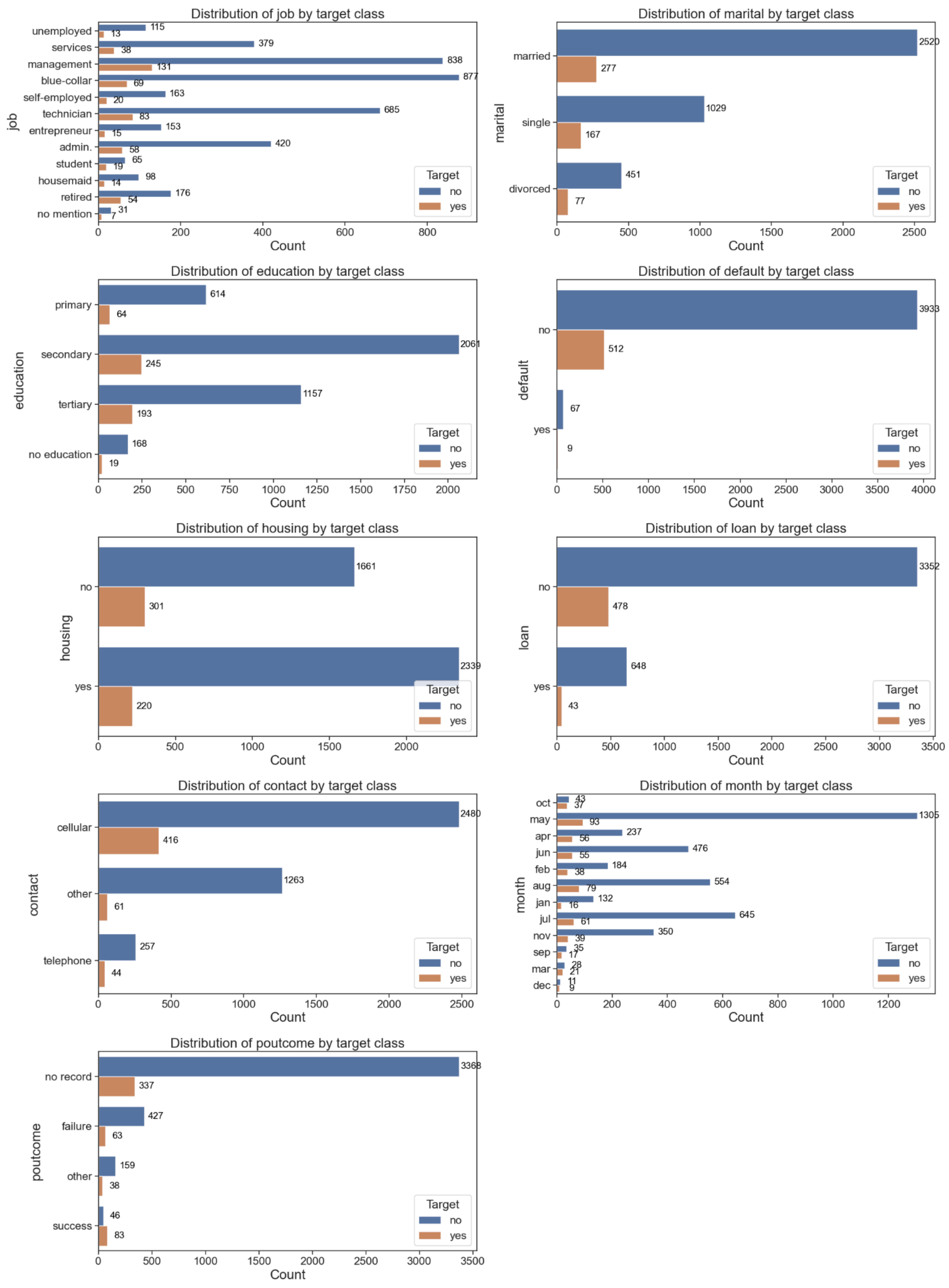


Figure A.b.8 Distribution of each Categorical Column with Target Class by Box Plot

Explanation:

- job column:

  Basically, the target value ‘yes’ is much less than the ‘no’ value in each job data type.

  The mode which is the most counted value is management in the job column. Also, the blue-collar job is high and closed to management data. The least data is “no mention” and it shows 38 data which are combined “yes” and “no” values.

- marital column:

  The marital has 3 types of value which are married, single, divorced. The most value type is married and it is around 2,800 which are combined “no” and “yes” value. The second largest is single value type and those are around 1,200 which are “no”, “yes” together. The least type of value is divorced; those are counted around 530. Generally, “no” value are much more than “yes” value.

- education column:

  There are 4 types of education which are primary, secondary, tertiary and no education. The most value type is “secondary” which are counted 2,300. Then, the next one is “tertiary”. That represents around 1,300. The third one is “primary” which are counted around 700. Then the last one is “no education which are counted around 200. Also, the “no” values are much more than “yes” value in each education value type.

- default column:

  There are only 2 types of default value which are “yes” and “no”. the question and is asking about having credit in default. So if the customer has the credit, respond “yes”, otherwise respond “no”.

  The default value “no” is counted quite more than the default value “yes”. So, the “yes” type of default value is very rare.

- housing column:

  This column shows the response from asking if the customer has a housing loan. The majority housing type value is “yes” and minor is “no” housing value type. In the distribution, Although the “yes” housing value has more “yes” in target class than the housing value type “no”, the housing value type “yes” has less “no” value type in target class than housing value type “no”. It means the customer who has a housing loan would be more successful to subscribe to a term deposit than the other.

- loan column:

  This column represents the response from asking if the customer has a personal loan. There are only 2 types of loan value which are “yes” and “no”. The most loan type value is “no” which is counted around 3,800 altogether. The number is much more than the “yes” type value of the loan. Also, the target value “no” is much more than “yes”.

- contact column:

  The column shows which type of communication with the customer.

  There are 3 types of contact value which are “cellular”, “other”, “telephone”. Among the three, the cellular is the most value type of contact which is counted around 2,900. It is more than half of the dataset then, it means more than half of customers communicated by cellular.

  Because the marketing campaign is based on phone call.

  The next one is “other” which can be email, visiting as offline and so on. that is counted around 1,300 altogether.

  The last one is “telephone” which represents around 300. it is less than 10% of the whole dataset. So, the customers who communicate by telephone are rare.

- month column:

  This distribution shows the last contact month with the customers. Commonly, the contact would be divided equally, However, the plot is much different with the expectation. Especially, in the may have lots of last contact with the customers, june, july, august as well. It can be guessed that it would be a busy season from May to August.

On the other hand, there would be a low season from September to December without November.

- poutcome column:

  The poutcome column represents the outcome of previous marketing campaign. There are 4 types of the value of poutcome which are “no record”, “failure”, “other”, “success”.

  The majority data type in the plot is “no record”. In this case, “no record” means the customers who have not got any marketing campaign. So, the customers without previous marketing campaign are most of the dataset. The second most type value in the poutcome column is “failure” which is counted around 500 altogether “yes” and “no” value in target class.

  The third one is “other” in the poutcome column which is counted around 200 combined “yes” and “no” value of target class.

  The last one is “success” value type in the poutcome column. that is a bit different with other data, “yes” value type in the poutcome and “yes” value type of the target class are more than the “no” value of the target. it mean if the customer was success to previous marketing campaign, subscribing to a term deposit would be more likely “yes” than “no”.

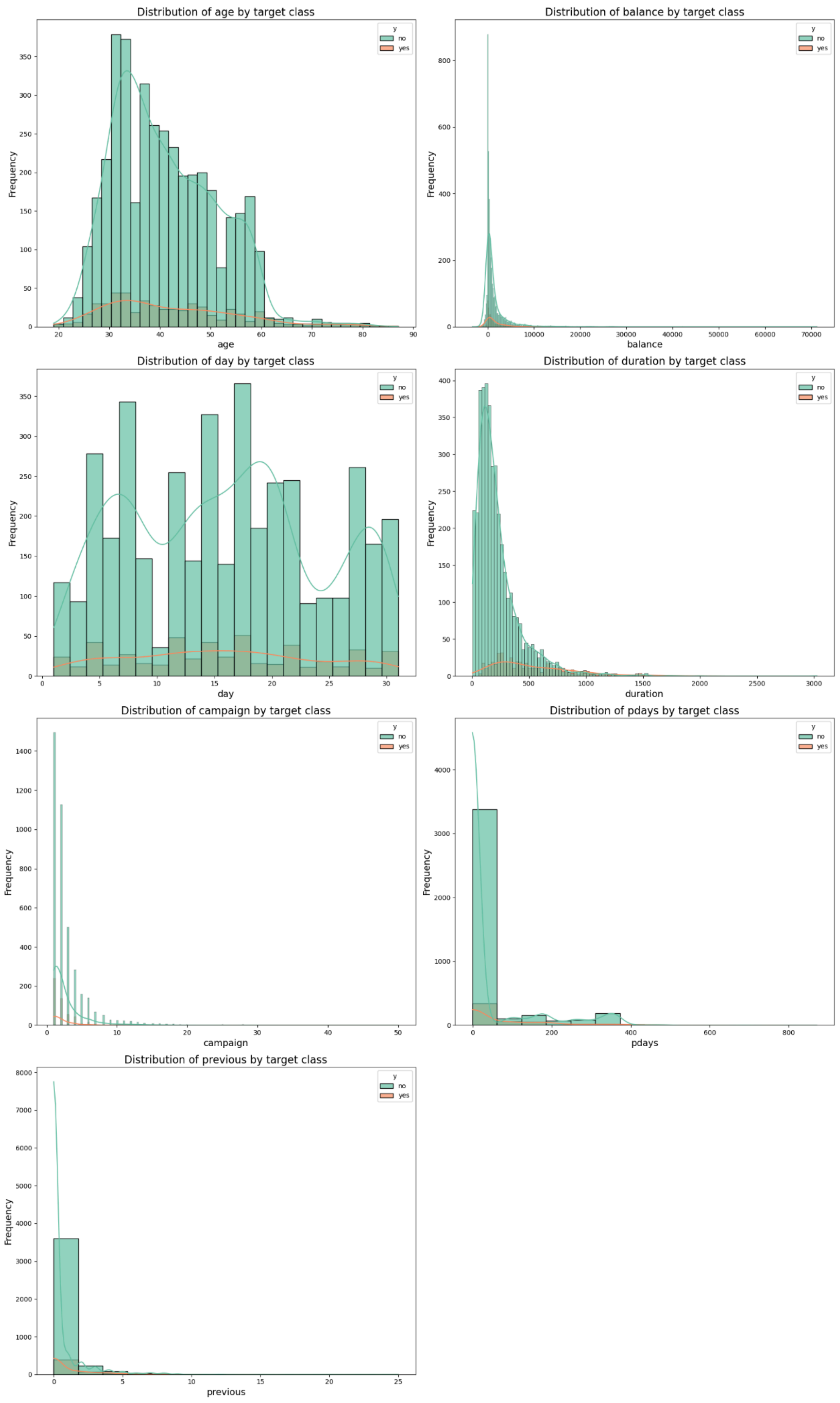


Figure A.b.9 Distribution of each Numerical Column with Target Class by Histogram

Explanation:

The above histogram plots show distribution with each numerical column and target column. Also, each target value which is yes and no represents each plot. As we mentioned, these plots show ‘no’ value is much more than ‘yes’ value as well.

- Age column:

    The distribution shape would be a right skewed shape which means the tail on the right side of the distribution extends further out than the tail on the left side. This indicates that there are more data points with higher values compared to lower values.

    Additionally, the mean (average) of the data is usually greater than the median (middle value) in a right-skewed distribution. This is because the mean is pulled towards the tail by the higher values.

- Balance column:

    The histogram leans to the left side, with a longer tail extending towards the right side (higher balance range). Imagine a bell curve tilted to the right, with most of the data concentrated on the left side and a few data points with very high balances on the right.

    The peak (highest point) of the distribution is located to the right of the center of the x-axis (balance range).

    A narrow bell shape suggests there's less variation in the data compared to a wider, more spread-out distribution also, the standard deviation of this distribution would be low.

- Day column:

    The distribution shape would not be the bell shape. The frequency would not be gathered to peak or center and it looks irregularly shaped.

    But the ‘yes’ values are distributed like bell shapes, however the values are much spread.

- Duration column:

  The duration distribution with the target column lean on the right side and the shape would be right skewed which has a long tail to right.

  The shape would be narrow which means the values are not spread much and gathered a lot, then the standard deviation would be low.

- Campaign column:

  The distribution is hard to define bell shaped distribution. Lowest value section of the campaign column is peak and the higher value of the campaign column, the lower frequency is represented.

- Pdays column:

  The distribution of Pdays with the target column is hard to determine as bell shape distribution. The lowest value section of the Pdays is the peak of this distribution and after the value section, the frequency decreases dramatically.

- Previous column:

  The distribution of the previous column with the target class looks similar to the distribution of the campaign column with the target column. The lowest value section of the previous column is peak and after the section, the frequency is decreasing.

- Calculate correlation coefficients to quantify the strength and direction of linear relationships between numerical features.

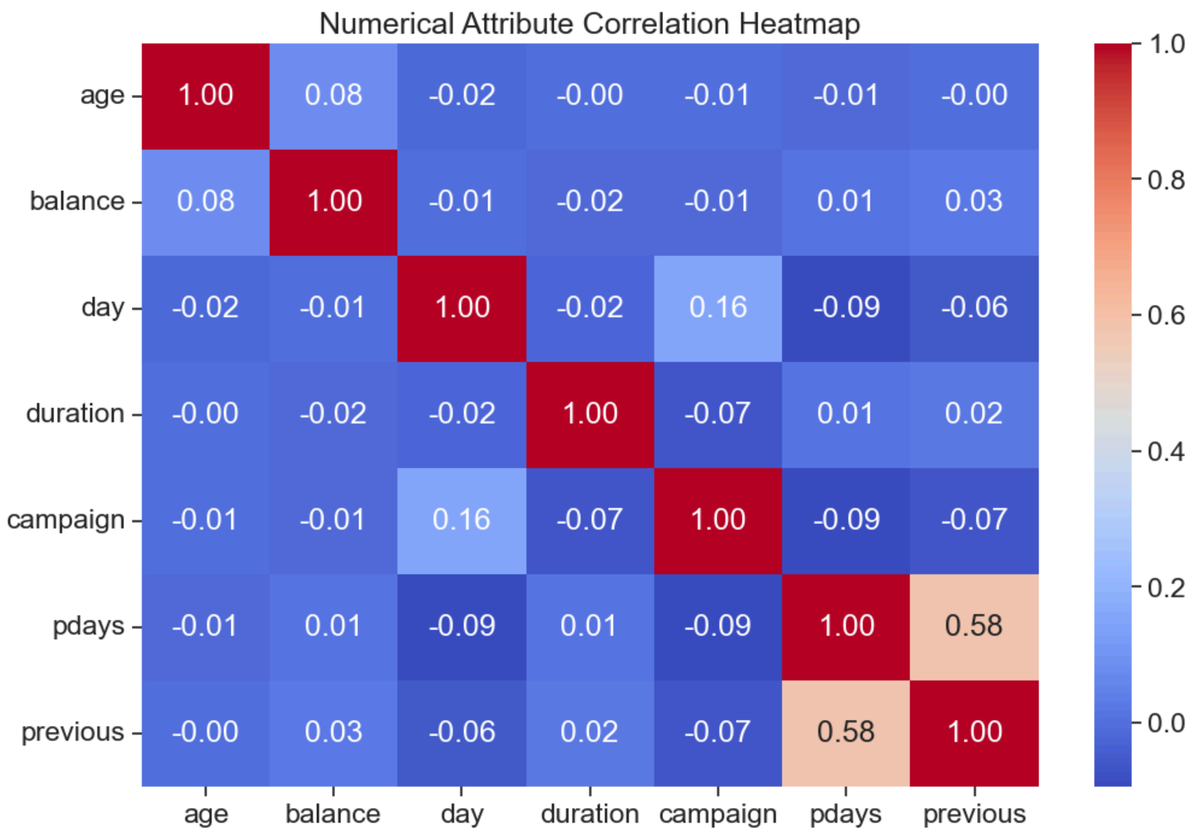


Figure A.b.10 Numerical Attributes Correlation Heatmap

Explanation:

- The above heatmap shows the relationship between each numerical column. Each number means correlation coefficient and closing 0 is less correlated to each other. Otherwise, it is highly correlated to each other.
- Generally, if the absolute value of the correlation coefficient under the 0.2, it would be a very weak correlation between each other. Then, almost of the correlation coefficient are close to 0 and the absolute value is under the 0.2
- However, there is only one is more than 0.2 which is (pdays, previous) correlation. the 0.58 would be said that it is moderately correlated to each other. it would impact further research in terms of accurate, robust classification models.

8. **Class Distribution:**
    - Investigate the distribution of the target variable (if it's a classification problem) to understand class imbalances and potential biases in the dataset.
    - Visualize the class distribution using techniques like bar plots or pie charts.

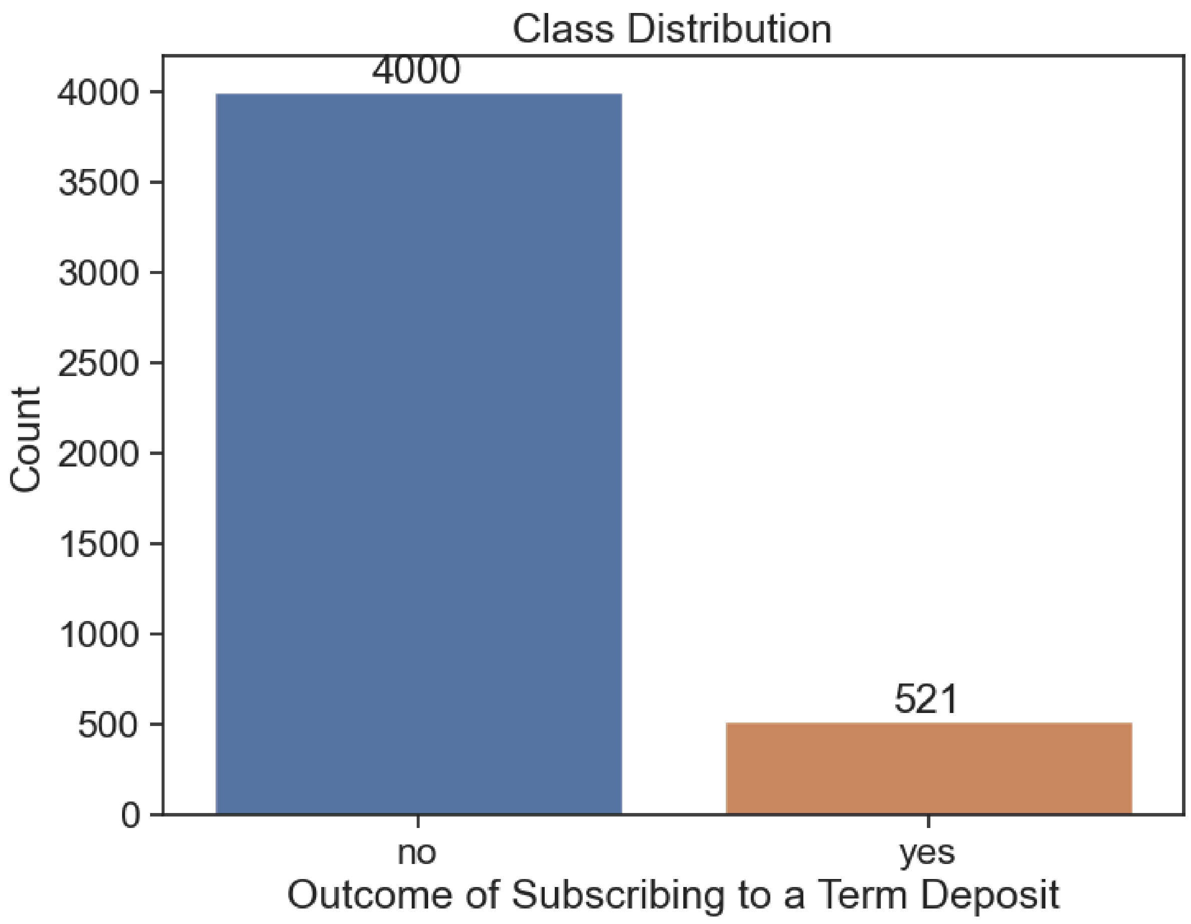


Figure A.b.11 Distribution of the Target Class

Explanation:

The above bar plot is the distribution of the Target class which is ‘y’. As it represents, the ‘no’ value is much more than ‘yes’ value. The difference between them is more than 6 times and it would be defined as an imbalanced target class.

So, when the dataset split to test and train set for further classification model, this dataset needs to be stratified by the target class and it would be biased classification model performance.

Summary Statistics and Insights:

- The dataset contains 4521 rows and 17 columns, with a mix of numerical and categorical features.
- There are no null values, but there are 'unknown' values in several categorical columns, which were replaced with more descriptive terms.
- There are no duplicate rows.
- Numerous outliers are present in the numerical features, but removing them would significantly reduce the dataset size.
- Most numerical features have weak correlations with each other, except for pdays and previous.
- The target variable ‘y’ is highly imbalanced, necessitating stratified sampling for building classification models.

c) Feature Selection and Analysis [10 marks]

Identify the most influential features in classifying this dataset using an appropriate method. Explain the process of the chosen feature selection method and use the top five features for building your models. Use a breakdown analysis for selected features by class and describe their distribution using appropriate plots.

Choosing the best feature selection method depends on the dataset characteristics and the specific machine learning algorithm you plan to use. Here's an explanation of each method and how it applies to the dataset, along with the process of feature selection for each:

Chi-Square Feature Selection

Explanation:

- The Chi-Square test is used for categorical data to test the independence of two variables. It measures how the observed counts of data points deviate from the expected counts under the null hypothesis of independence.
- Suitable for datasets with categorical features.

Process:

1. Encode the categorical variables.
2. Scale the data to ensure non-negative values.
3. Apply the Chi-Square test to rank features based on their Chi-Square statistic with the target variable.

ANOVA (Analysis of Variance)

Explanation:

- ANOVA is used for numerical data and tests whether there are significant differences between the means of two or more groups.
- Suitable for datasets with numerical features.

Process:

1. Encode the categorical variables.

2. Apply the ANOVA F-test to rank features based on the variance between groups.

Feature Selection Using ANOVA and Chi-Square Tests

1. Purpose of Feature Selection

Feature selection aims to identify the most important features in a dataset that contribute significantly to the prediction of the target variable. This helps improve model performance by:

- Reducing overfitting by eliminating irrelevant features.
- Enhancing model accuracy and robustness.
- Simplifying the model, making it easier to interpret and faster to train.

2. Statistical Tests for Feature Selection

ANOVA (Analysis of Variance)

- Purpose: ANOVA is used to determine if there are statistically significant differences between the means of two or more groups. For numerical features, it assesses whether different levels of the feature have significantly different values of the target variable.
- Usage in Feature Selection: When applied to numerical features, ANOVA helps identify features that have a strong influence on the target variable.
- P-Value Interpretation: A low p-value (typically ≤ 0.05) suggests that the feature significantly impacts the target variable and should be considered for inclusion in the model.

Chi-Square Test

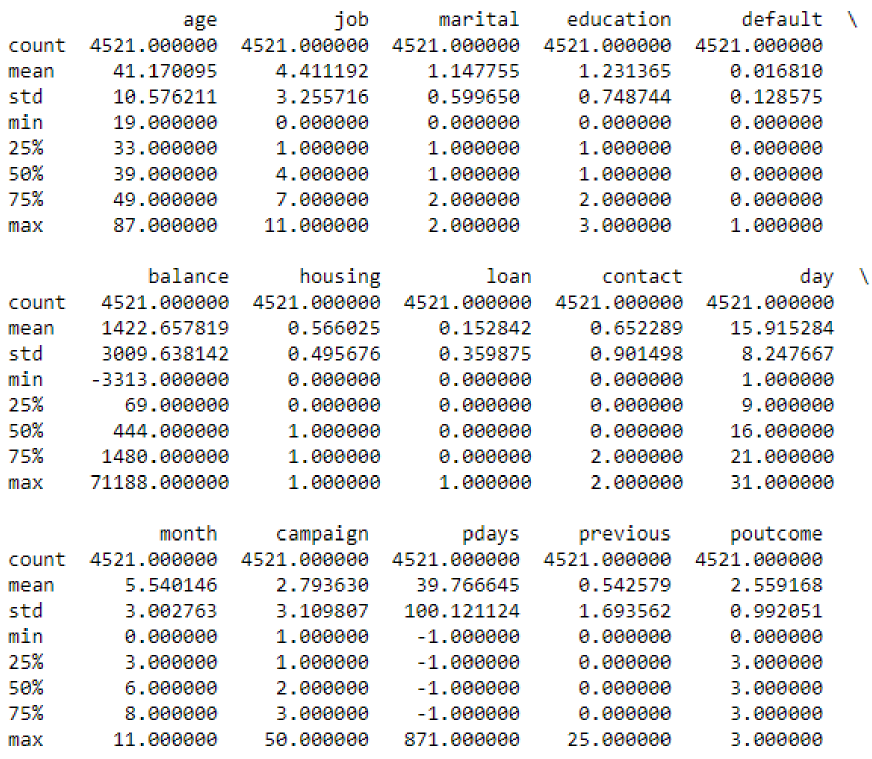
- Purpose: The Chi-Square test evaluates whether there is a significant association between two categorical variables. It compares the observed frequencies of data points against the expected frequencies if the variables were independent.
- Usage in Feature Selection: For categorical features, the Chi-Square test helps identify features that have a significant relationship with the target variable.

- P-Value Interpretation: A low p-value (typically ≤ 0.05) indicates that the feature is significantly associated with the target variable and should be considered for inclusion in the model.

3. Process of Feature Selection in Your Context

1. Loading and Preparing the Dataset:

    - Load the dataset and preprocess it by encoding categorical variables, and normalizing numerical features as needed.



```
               age          job      marital    education      default  \
count  4521.000000  4521.000000  4521.000000  4521.000000  4521.000000
mean     41.170095     4.411192     1.147755     1.231365     0.016810
std      10.576211     3.255716     0.599650     0.748744     0.128575
min      19.000000     0.000000     0.000000     0.000000     0.000000
25%      33.000000     1.000000     1.000000     1.000000     0.000000
50%      39.000000     4.000000     1.000000     1.000000     0.000000
75%      49.000000     7.000000     2.000000     2.000000     0.000000
max      87.000000    11.000000     2.000000     3.000000     1.000000

            balance      housing         loan      contact          day  \
count   4521.000000  4521.000000  4521.000000  4521.000000  4521.000000
mean    1422.657819     0.566025     0.152842     0.652289    15.915284
std     3009.638142     0.495676     0.359875     0.901498     8.247667
min    -3313.000000     0.000000     0.000000     0.000000     1.000000
25%       69.000000     0.000000     0.000000     0.000000     9.000000
50%      444.000000     1.000000     0.000000     0.000000    16.000000
75%     1480.000000     1.000000     0.000000     2.000000    21.000000
max    71188.000000     1.000000     1.000000     2.000000    31.000000

             month     campaign        pdays     previous     poutcome
count  4521.000000  4521.000000  4521.000000  4521.000000  4521.000000
mean      5.540146     2.793630    39.766645     0.542579     2.559168
std       3.002763     3.109807   100.121124     1.693562     0.992051
min       0.000000     1.000000    -1.000000     0.000000     0.000000
25%       3.000000     1.000000    -1.000000     0.000000     3.000000
50%       6.000000     2.000000    -1.000000     0.000000     3.000000
75%       8.000000     3.000000    -1.000000     0.000000     3.000000
max      11.000000    50.000000   871.000000    25.000000     3.000000
```

Figure A.c.1 Encoded Categorical Attributes without Target Class and Duration Attribute

Explanation:

Before these step, all the features are required to look at the description of each attribute first. In this case, The description of the duration attributes, it shows this attribute should be removed when the realistic predictive model. Because it affects the target class highly. So, in the feature selection process, the duration attribute is removed.

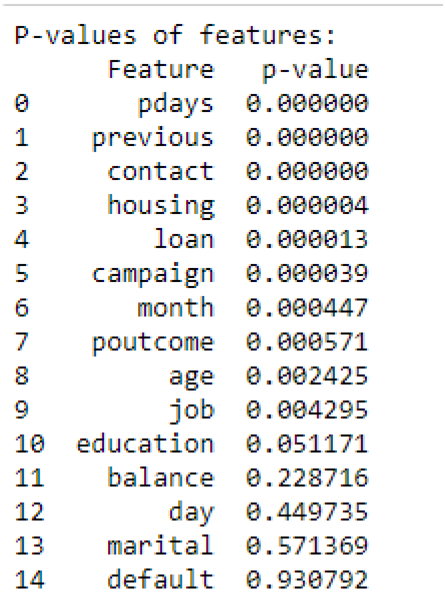
Then, the categorical attributes are encoded to numerical values to process the Chi-square test.

2. Applying Statistical Tests:

    - Numerical Features: Apply the ANOVA test to each numerical feature to calculate its p-value.
    - Categorical Features: Apply the Chi-Square test to each categorical feature to calculate its p-value.

3. Comparing P-Values:

    - Gather the p-values for all features (both numerical and categorical).
    - Select Features with the Lowest P-Values: Choose the features with the lowest p-values as they are the most statistically significant in predicting the target variable.



```
P-values of features:
      Feature   p-value
0       pdays  0.000000
1    previous  0.000000
2     contact  0.000000
3     housing  0.000004
4        loan  0.000013
5    campaign  0.000039
6       month  0.000447
7    poutcome  0.000571
8         age  0.002425
9         job  0.004295
10  education  0.051171
11    balance  0.228716
12        day  0.449735
13    marital  0.571369
14    default  0.930792
```

Figure a.c.2 List of the P-value of all Features

Explanation:

The above list is the p-value of all features. As it mentioned, low p-value which is less than 0.05 generally means that the feature significantly impacts the target variable and should be considered for inclusion in the model.

Even though most of the features have the p-value which is less than 0.05, the top 5 features have to be selected. pdays, previous, contact, housing, loan attributes are selected.

4. Evaluation Correlation Coefficient with Selected Attributes:

    - The previous step is check relationship with target class. Then, the selected attributes need to be check the correlation with each selected attribute.

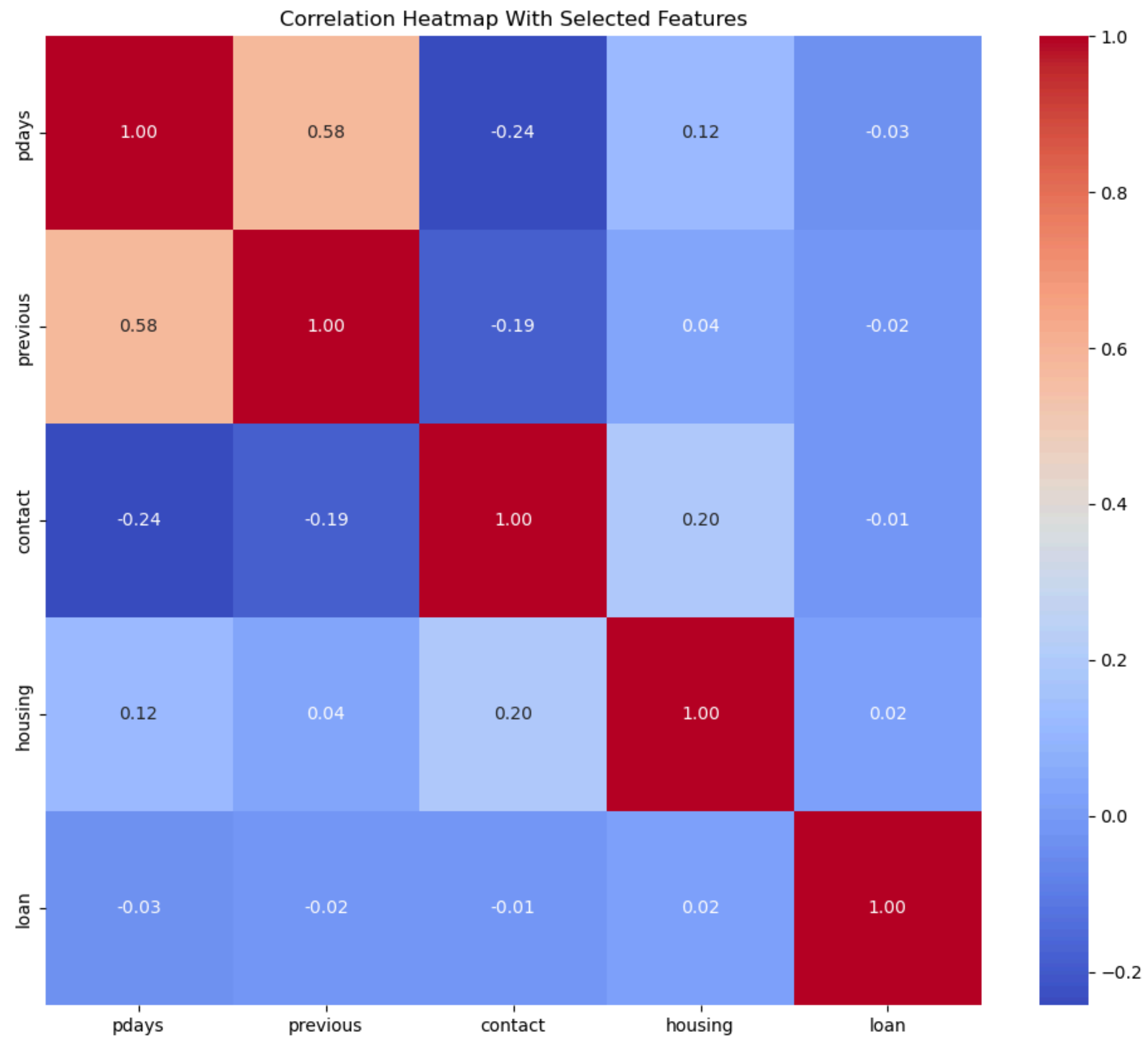


Figure A.c.3 Correlation Heatmap With Selected Features

Explanation:

The above heatmap represents correlation coefficient with each selected attribute.

By Pearson Correlation Coefficient, if the absolute value of correlation coefficient less than 0.2 would be very low correlated to each other.

Then, if the absolute value of correlation coefficient is between 0.2 to 0.39, it would be low correlated to each other.

Then, if the value is between 0.4 and 0.59, it would be moderately correlated to each other.

If the value is between 0.6 and 0.79, it would be highly correlated to each other.

If the value is more than 0.79, it would be very highly correlated to each other.

Generally, the absolute value of correlation coefficient which is close to 0 would be suitable for further steps and those features would not impact the other attribute and make a robust result.

5. Adjust Selected Features and Evaluation:

   In this case, most of them determine the absolute value of the correlation coefficient which is less than 0.2. It means most of them are very low correlated to each other.

   Then, two of the absolute values are between 0.2 and 0.39 such as (contact, housing), (contact, pdays). That means those are low correlated to each other.

   Also, there are some correlation coefficient which is the absolute value between 0.4 and 0.59 such as (pday, previous). It means those are moderately correlated to each other.

   there is one pair of selected attributes are correlated moderately then, it should be adjust by removing highly correlated features. In this case, the previous attribute is removed and add another attribute which campaign attribute because the campaign attribute has 6th lowest p-value.

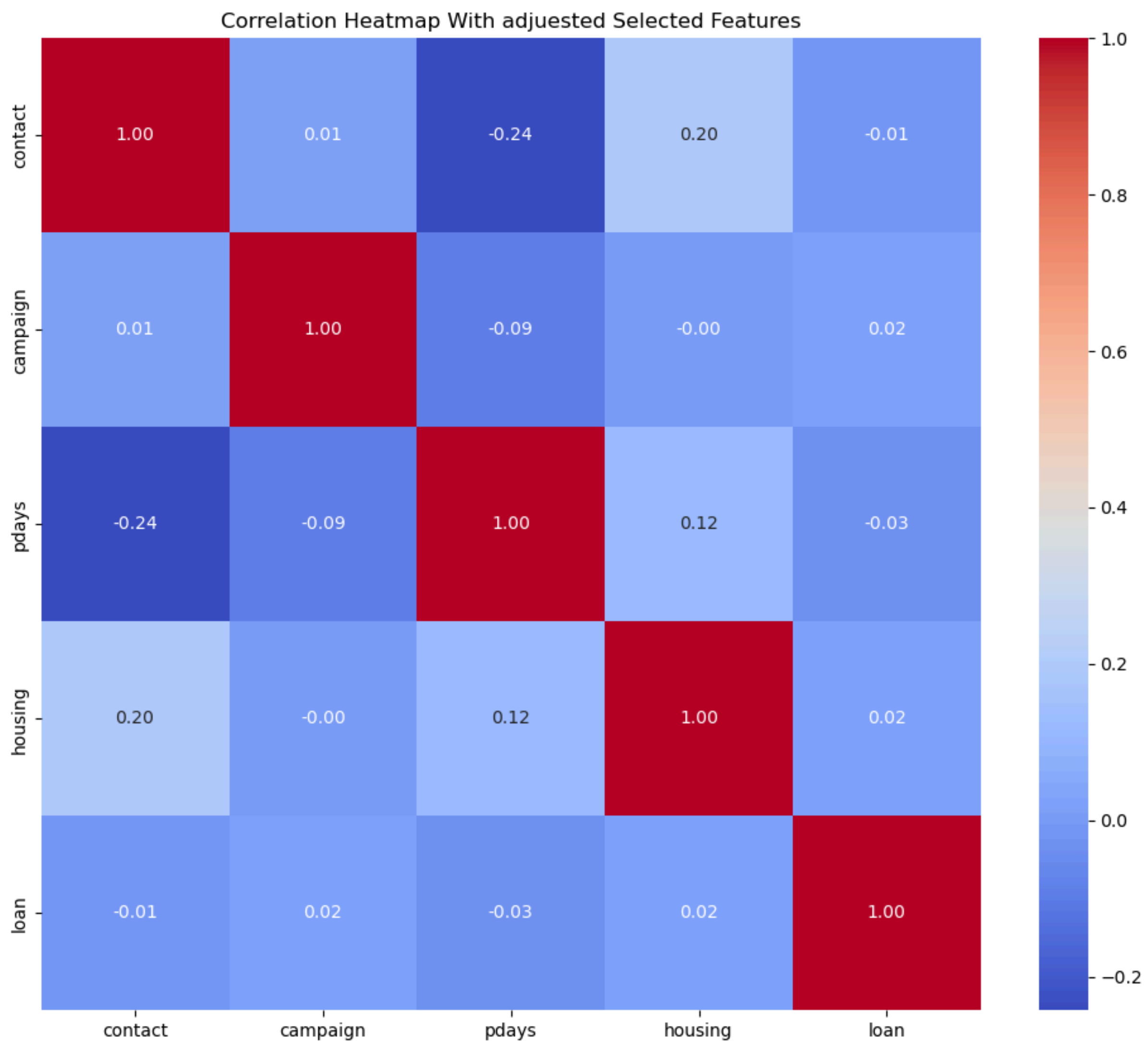


Figure A.c.4 Correlation Heatmap With Adjusted Selected Features

Explanation:

The selected features are adjusted and represent a heatmap of correlation coefficient. In this case, the correlation which is moderately correlated to each other is removed and all of the correlation coefficient are lower than 0.4 which means all correlation with each selected attribute are weak and each attribute would not impact other attributes.

```
The list of adjuested selected features:
['contact', 'campaign', 'pdays', 'housing', 'loan']
```

Figure A.c.5 The List of Adjusted Selected Features

So the final selected features are contact, campaign, pdays, housing, and loan.

4. The Reason For This Approach?

- Statistical Significance: By using ANOVA and Chi-Square tests, you ensure that the features selected have a statistically significant relationship with the target variable.
- Robustness: This method helps in building a robust model by focusing on the most relevant features and excluding those that do not contribute significantly.
- Simplicity: Simplifying the model by selecting only the most important features reduces computational complexity and improves interpretability.

Analysis of The Feature Selection by Target Class

To understand the plots below, there are two types of bars which are blue and orange. Those are values of y (subscribing to a term deposit) which are ‘yes’ and ‘no’. As an explanation of class distribution, the no values are much more than yes values.

Pdays Feature:

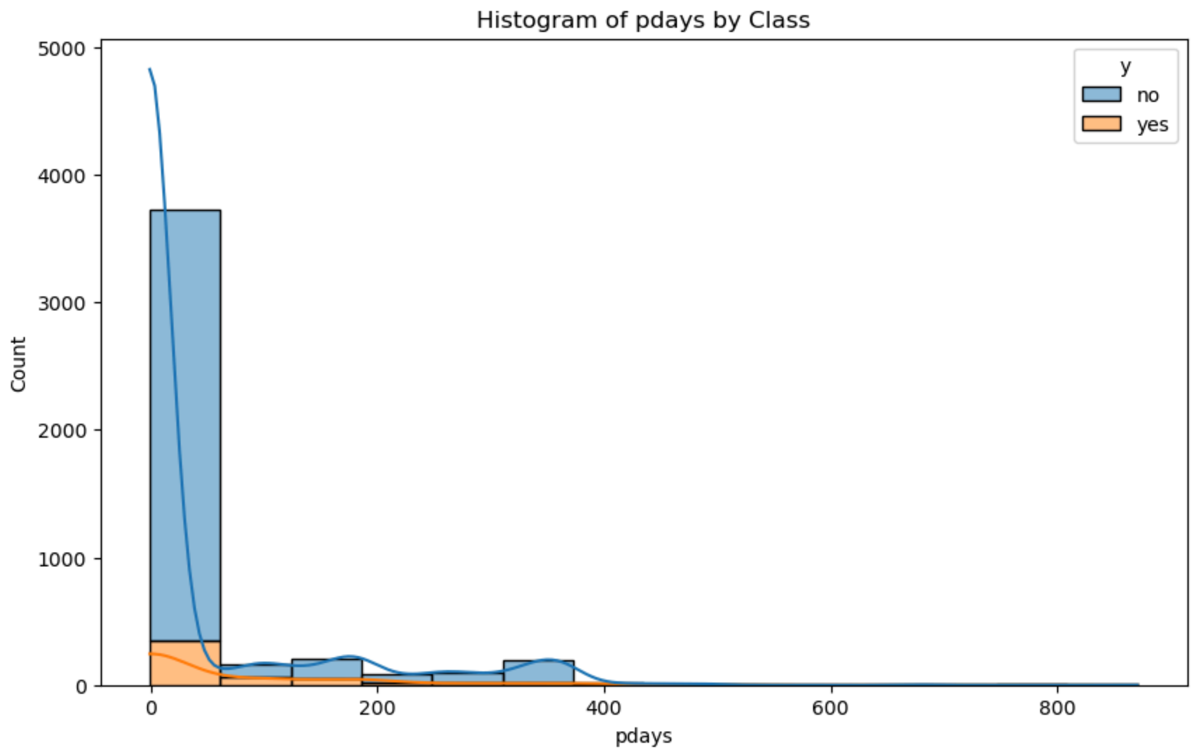


Figure A.c.6 Histogram Distribution of Pday by Class

Explanation:

The pdays means the number of days that passed by after the client was last contacted from a previous campaign So the bigger number shows longer time from last contact to previous campaign. Most of pdays are shorter than 200. It means after the campaign. The contact would be finished within 200 days typically.

Campaign Feature:

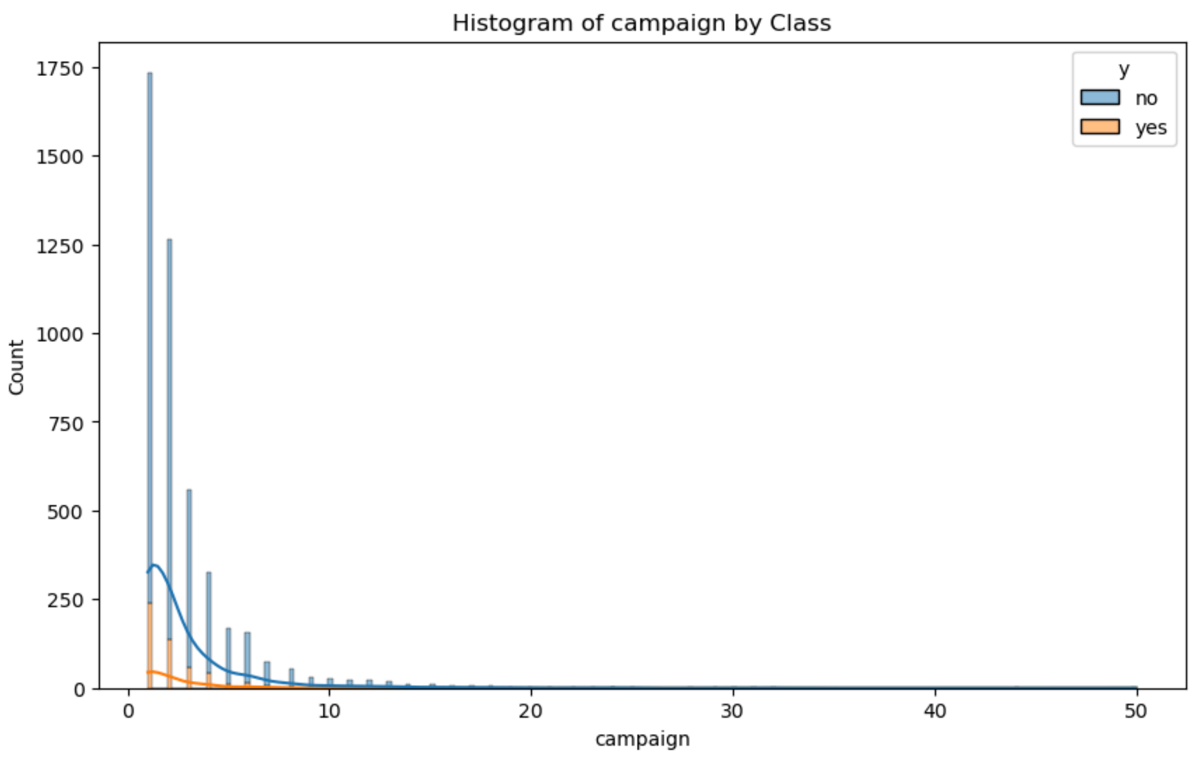


Figure A.c.7 Histogram Distribution of Campaign by Class

Explanation:

The campaign means number of contacts performed during this campaign and for this client. In this step, regardless of the y value, lowest value is the mode in the distribution and most of contact amount is less than 10. So, the client would contact to bank less than 10 time generally.

Housing Feature:

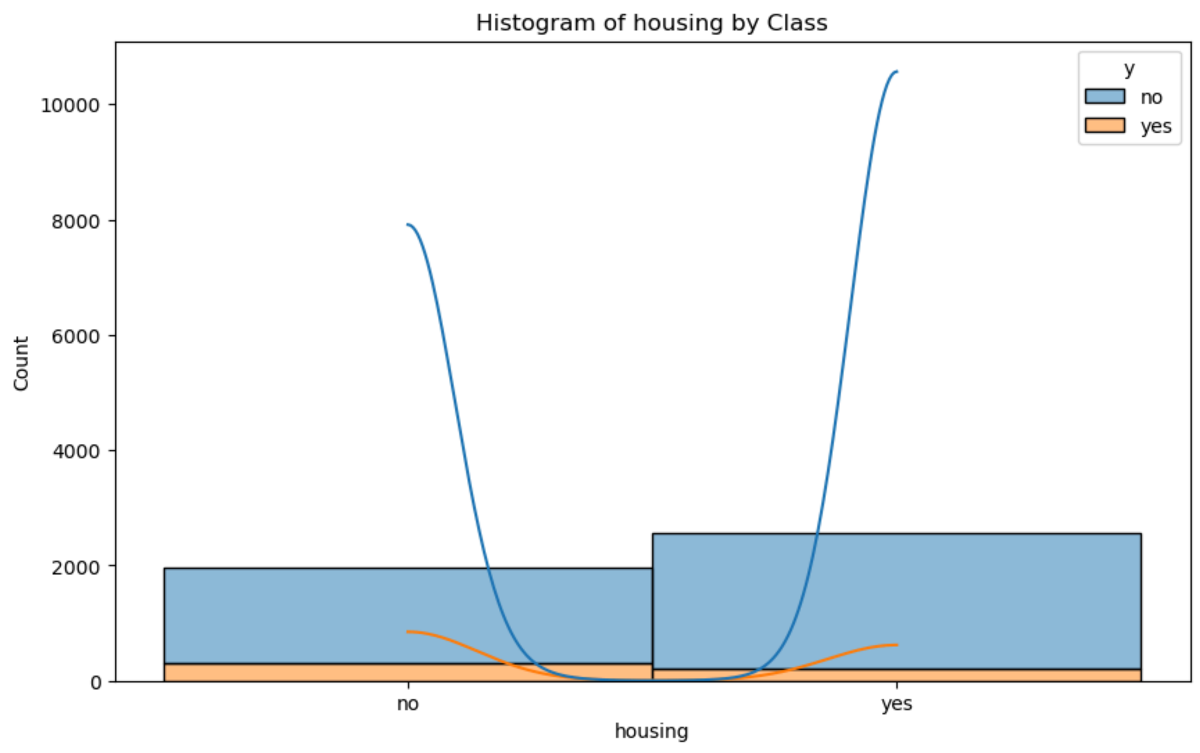


Figure A.c.8 Histogram Distribution of Housing by Class

Explanation:

The housing means if the client has a housing loan or not. In this distribution, the client who has a housing loan is more than the client who does not have a housing loan.

Contact Feature:

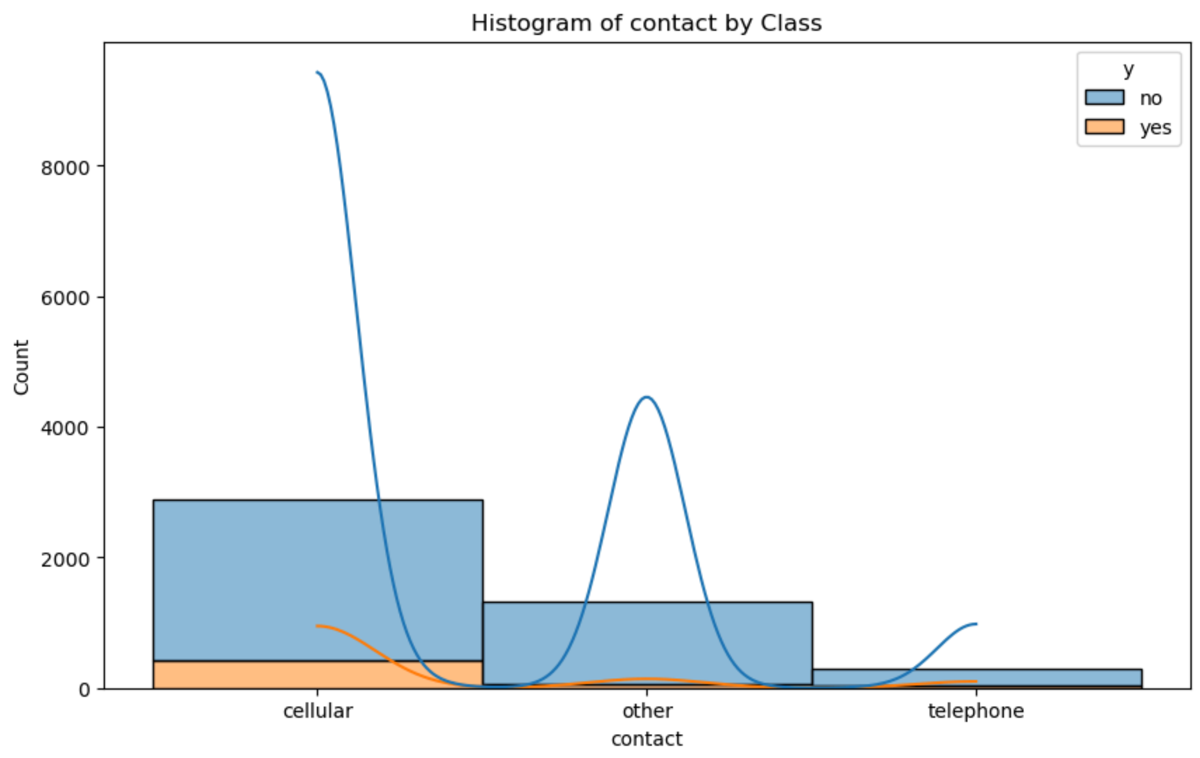


Figure a.c.9 Histogram Distribution of Contact by Class

Explanation:

In the EDA step, the contact feature is already described and the different point with the previous one is plot style. In the step, the count plot is represented and it has the line of the pattern it makes more visualize the trend of the distribution.

The cellular is most of the case which is communicating with the client regardless of y value then, the second one is the other. it could be another communication device or offline. The third one is the telephone which is the lowest count.

Loan Feature:

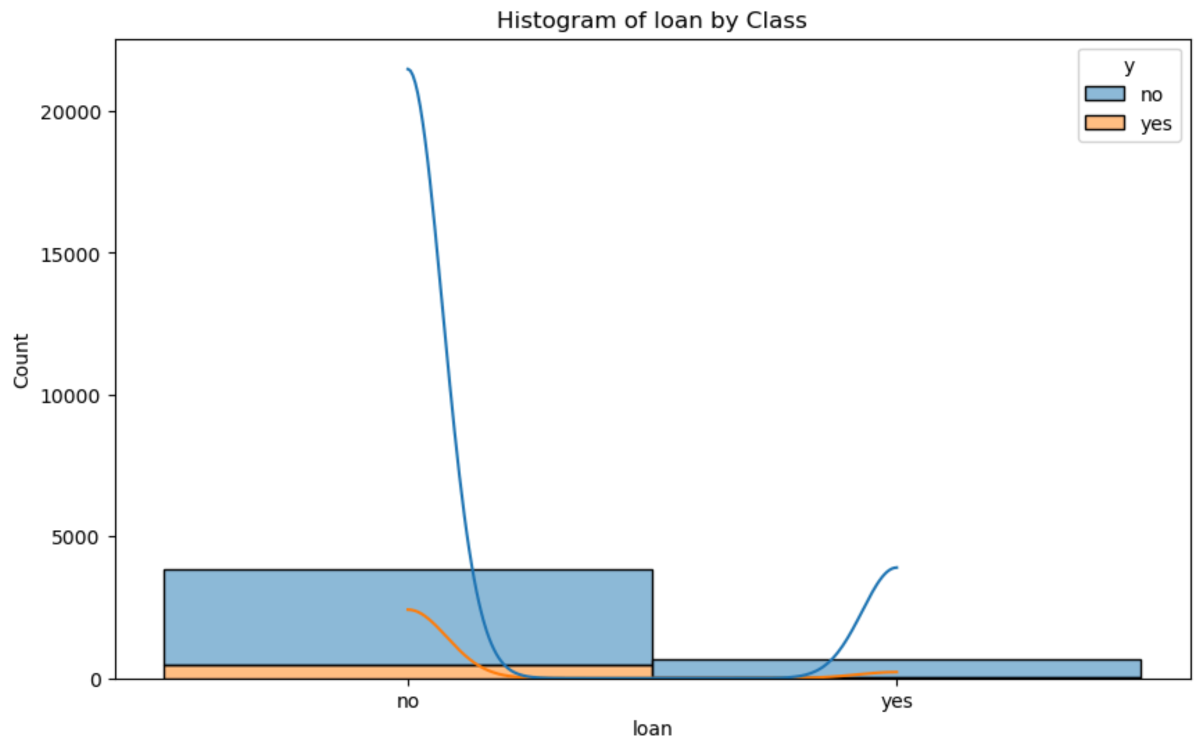


Figure A.c.10 Histogram Distribution of Loan by Class

Explanation:

In this plot, the loan shows if the client has a personal loan or not. Unlike housing, the 'no' value count in the loan attribute is much more than 'yes' value count. It looks like the opposite response and shape with housing feature distribution.

Summary:

The selected features distributions show totally different shape, numerical ranges. So, each feature would not be correlated to each other visually.

d) Independence Assumption in Naïve Bayes [5 marks]

Discuss the independence assumption between the features in the Naïve Bayes algorithm and support your answer with respect to the selected features.

Independence Assumption in Naïve Bayes

The Naïve Bayes algorithm is based on Bayes' Theorem and assumes that the features are conditionally independent given the class label. This means that the presence (or absence) of a particular feature is unrelated to the presence (or absence) of any other feature, given the class label.

Independence Assumption with Respect to the Selected Features

Let's examine the independence assumption in the context of the selected features from the previous step. We will use correlation to evaluate the degree of independence between features.

Although Naïve Bayes assumes independence, in practice, features may exhibit some degree of correlation. This analysis helps us understand how closely the real-world data aligns with the theoretical assumption.

1. Correlation Matrix

We will calculate and visualize the correlation matrix for the top 5 selected features to evaluate their pairwise correlations.

Interpretation of the Correlation Matrix

- Correlation Values: The correlation coefficient ranges from -1 to 1. A value closer to 0 implies weak correlation (independence), while values closer to -1 or 1 imply strong negative or positive correlation, respectively.

- Independence Assumption: In the context of Naïve Bayes, we prefer the features to be as uncorrelated as possible.

The Correlation Matrix with Selected Features

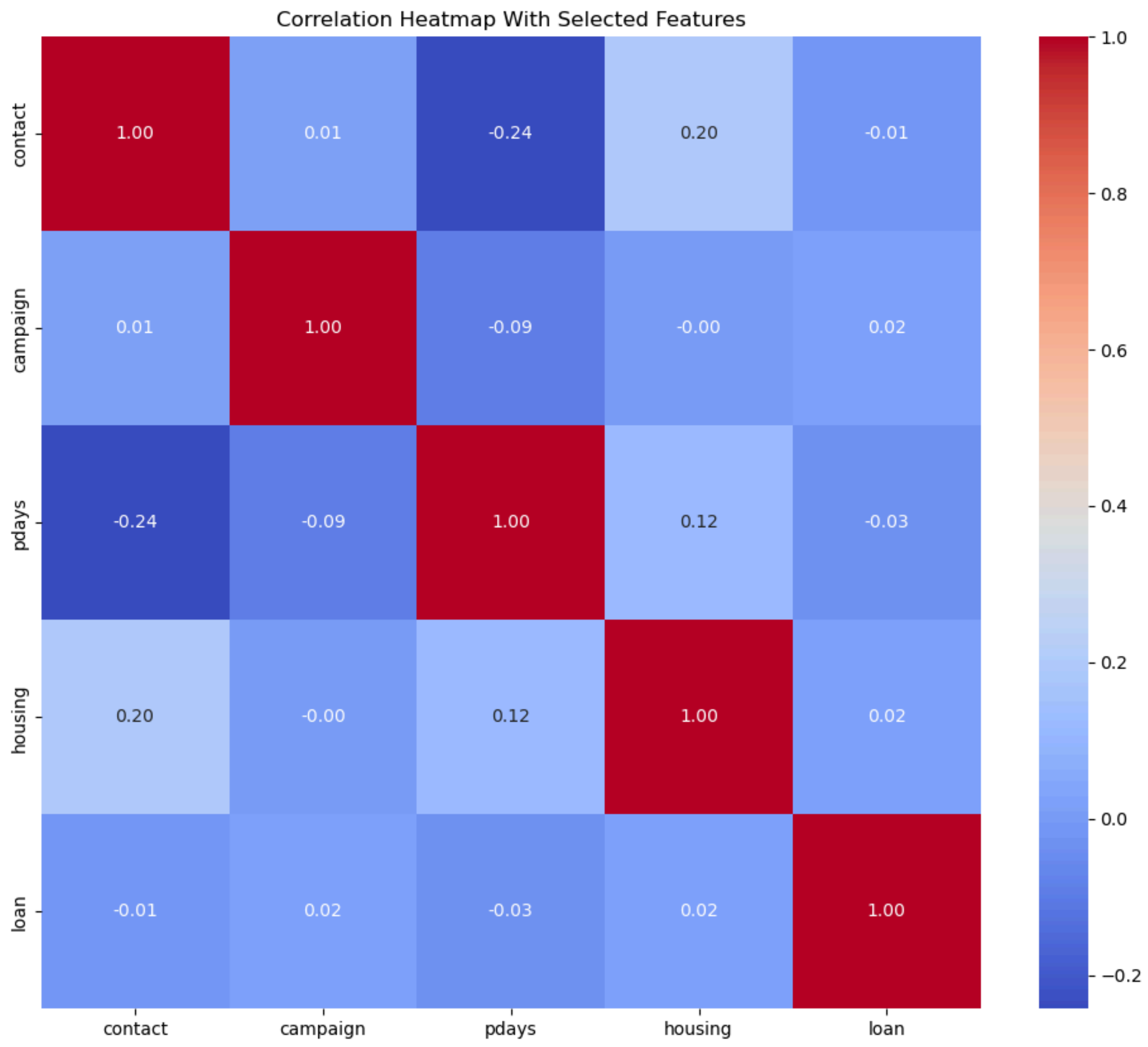


Figure A.d.1 Correlation Heatmap with Selected Features

Analysis of Correlations

- **Low Correlation**: Most of the correlation values are close to 0, suggesting that the features are approximately independent. This aligns well with the Naïve Bayes assumption.

- **High Correlation**: There are no strong correlations (values close to 1 or -1) between features, which is favorable for Naïve Bayes. However, some weak correlations exist, but they are not significant enough to violate the independence assumption.

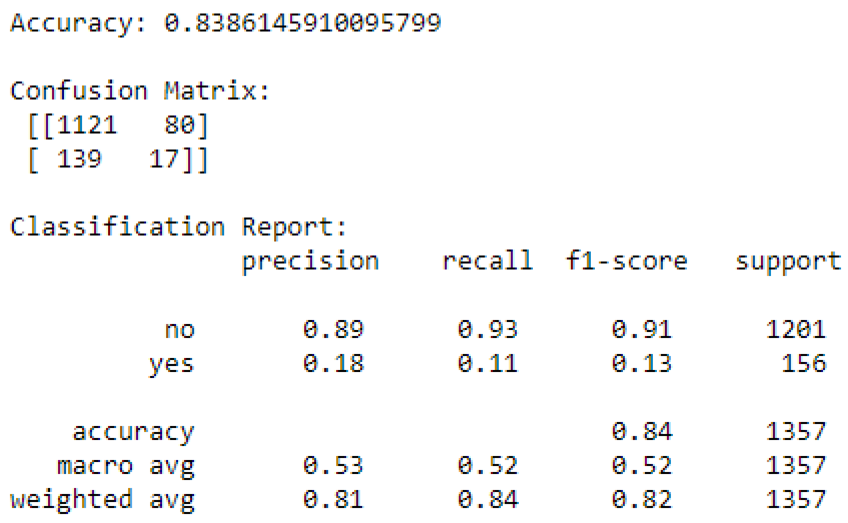
Conclusion

In summary, most of the selected features exhibit very weak correlations, supporting the Naïve Bayes assumption of independence. There is one correlated coefficient which is -0.24 which means weak correlation with each other, but it would be acceptable to independence for the Naive Bayes classification model.

e) Naive Bayes Model Building and Evaluation [10 marks]

Run the Naive Bayes algorithm with the GaussianNB implementation for the selected features. Provide evaluation metrics, including the confusion matrix, showing the performance of the NB model. Discuss the results.

Confusion Matrix and Classification Report of Naïve Bayes Model with Selected Features



```
Accuracy: 0.8386145910095799

Confusion Matrix:
 [[1121   80]
 [ 139   17]]

Classification Report:
              precision    recall  f1-score   support

          no       0.89      0.93      0.91      1201
         yes       0.18      0.11      0.13       156

    accuracy                           0.84      1357
   macro avg       0.53      0.52      0.52      1357
weighted avg       0.81      0.84      0.82      1357
```

Figure A.e.1 Accuracy, Text of Confusion Matrix, Classification Report of Naive Bayes model

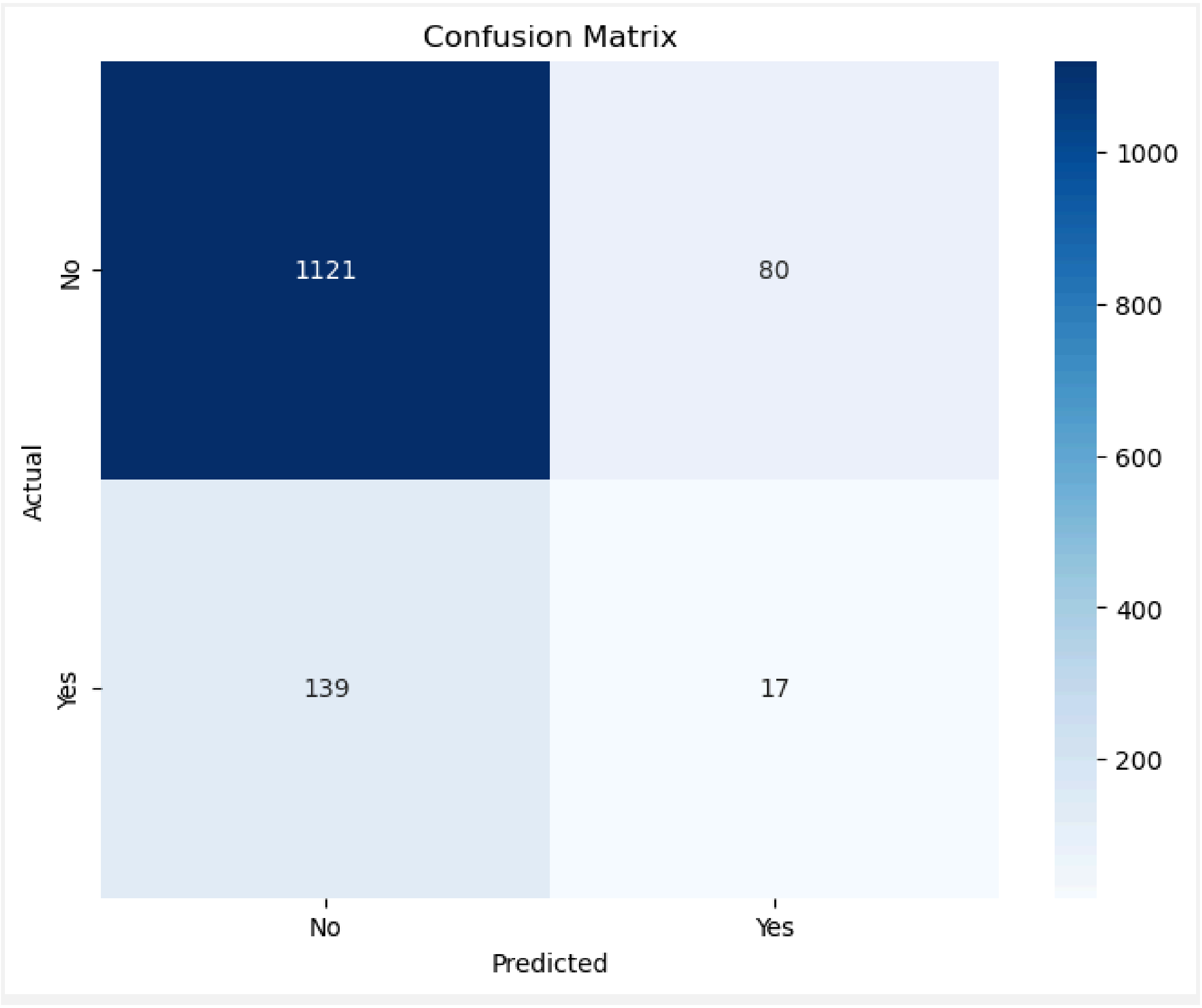


Figure A.e.2 Confusion Matrix of The Naive Bayes Model

Discussion of Results

1. Accuracy:
    - The overall accuracy of the model is approximately 0.84 (84%).
    - This means that 84% of the predictions made by the model were correct.
    - Formula: Accuracy = (True Positives + True Negatives) / (Total Predictions)

2. Confusion Matrix:

The confusion matrix provides a breakdown of the model’s predictions:

- True Positives (TP): 17 (actual ‘Yes’, predicted ‘Yes’)
- True Negatives (TN): 1121 (actual ‘No’, predicted ‘No’)
- False Positives (FP): 80 (actual ‘No’, predicted ‘Yes’)
- False Negatives (FN): 139 (actual ‘Yes’, predicted ‘No’)

3. Classification Report:
    - Precision: This metric indicates the proportion of positive identifications that were actually correct. High precision means that the classifier returned substantially more relevant results than irrelevant ones.
    - Formula: Precision = (True Positives) / (True Positives + False Positives)
    - Recall: This metric indicates the proportion of actual positives that were identified correctly. High recall means that the classifier returned most of the relevant results.
    - Formula: Recall = (True Positives) / (True Positives + False Negatives)
    - F1-Score: The F1-score is the harmonic mean of precision and recall and provides a single metric that balances both concerns. It is particularly useful when you need to balance precision and recall.
    - Formula: F1-Score = 2 * (Precision * Recall) / (Precision + Recall)
4. Class Label ‘No’:
    - Precision: 0.89 (out of all predicted ‘No’, 89% were actually ‘No’)
    - Recall: 0.93 (out of all actual ‘No’, 93% were correctly predicted)
    - F1-score: 0.91 (harmonic mean of precision and recall)

5. Class Label ‘Yes’:
    - Precision: 0.18 (out of all predicted ‘Yes’, 18% were actually ‘Yes’)
    - Recall: 0.11 (out of all actual ‘Yes’, 11% were correctly predicted)

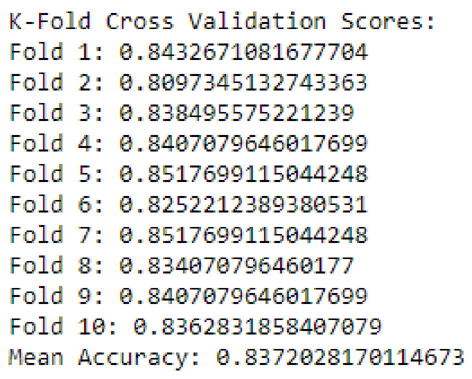
- F1-score: 0.13

6. Interpretation:
    - The model performs well in predicting the ‘No’ class (high precision and recall).
    - However, it struggles with the ‘Yes’ class (low precision and recall).
    - The F1-score balances precision and recall, and it’s lower for the ‘Yes’ class.
    - Consider whether improving precision or recall is more important based on your specific use case.

Evaluation with K-fold Cross Validation

Explanation of K-Fold Cross Validation Scores

- K-Fold Cross Validation provides an average performance metric by splitting the data into *k* subsets (folds) and training/testing the model *k* times, each time with a different fold as the test set and the remaining folds as the training set. The individual scores represent the accuracy for each fold, indicating how well the model generalizes to unseen data.



```
K-Fold Cross Validation Scores:
Fold 1: 0.8432671081677704
Fold 2: 0.8097345132743363
Fold 3: 0.838495575221239
Fold 4: 0.8407079646017699
Fold 5: 0.8517699115044248
Fold 6: 0.8252212389380531
Fold 7: 0.8517699115044248
Fold 8: 0.834070796460177
Fold 9: 0.8407079646017699
Fold 10: 0.8362831858407079
Mean Accuracy: 0.8372028170114673
```

Figure A.e.3 K-Fold Cross Validation Scores in NB Model

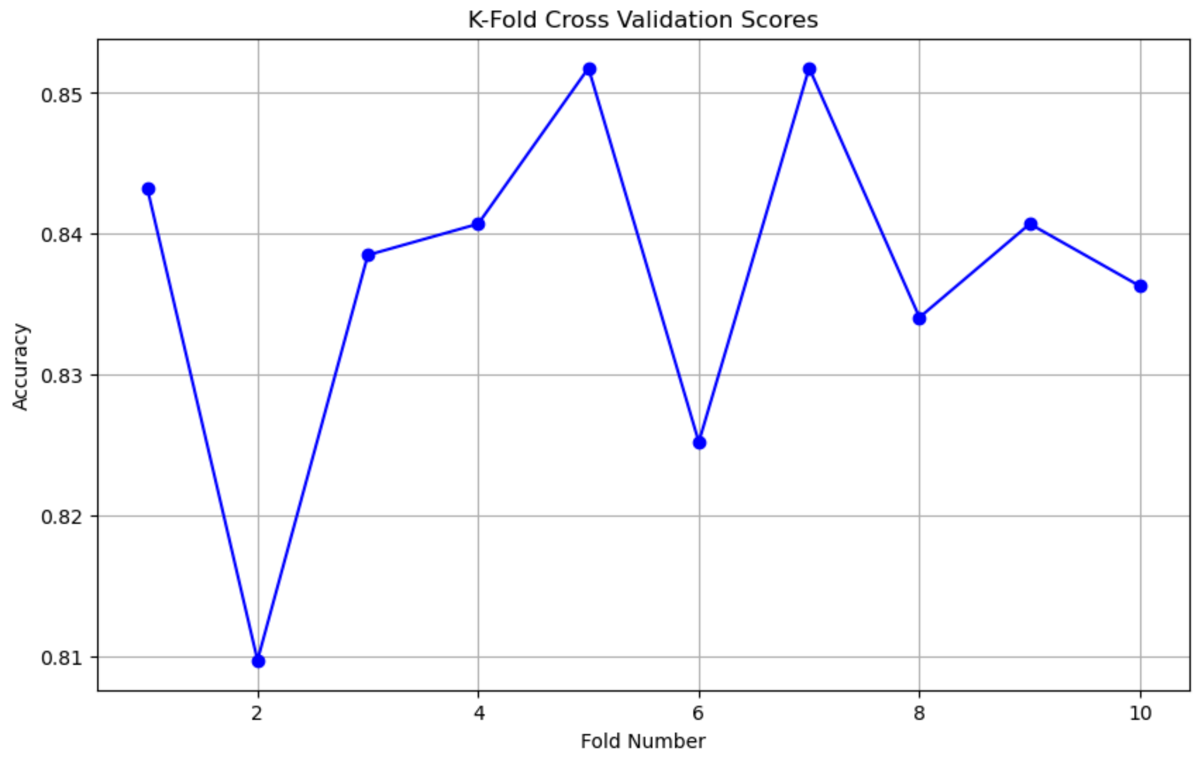


Figure A.e.4 Graph of K-Fold Cross Validation Scores in NB Model

Explanation and Interpretation

1. **Consistency and Stability:**

   The scores across the 10 folds are quite consistent, ranging from approximately 0.81 to 0.86. This consistency indicates that the Naive Bayes model is stable and performs reliably across different subsets of the dataset. There are no significant drops or peaks, which suggests that the model is not overly sensitive to the specific data points in each fold.

2. **High Mean Accuracy:**

   The mean accuracy of approximately 83.72% indicates that the Naive Bayes classifier is performing well on this dataset. This high accuracy suggests that the model is effective at distinguishing between the classes.

3. **Interpreting the Range of Scores:**

   The lowest accuracy observed is 80.97%, and the highest is 85.18%. This small range (approximately 4.21%) demonstrates that the model is quite robust, meaning it is not overly affected by the partitioning of the data.

4. **Performance Evaluation:**

The mean accuracy is a good metric, but it is also important to consider the precision, recall, and F1-score for a more comprehensive evaluation of the model. These metrics can help understand how well the model performs in terms of true positives, false positives, and false negatives.

Why the K-Fold Cross Validation Accuracy and Model

Accuracy Differ

The slight difference between the K-Fold cross-validation accuracy (mean 83.72%) and the model accuracy on the test set (83.86%) is expected due to the following reasons:

1. **Different Data Splits**:

   The K-Fold cross-validation uses multiple splits of the data, providing a more robust estimate of model performance. The test set accuracy is based on a single split and might slightly differ from the cross-validation mean.

2. **Variability in Data**:

   Randomness in the splitting of data for cross-validation can lead to slight variations in accuracy. Each fold might have slightly different distributions of classes, affecting the model's performance.

3. **Model Generalization**:

   Cross-validation helps in assessing how well the model generalizes to unseen data. A single test set might not capture all aspects of the model's performance, leading to a slight difference in observed accuracy.

Overall, the K-Fold cross-validation results indicate a strong and stable performance of the Naive Bayes classifier on this dataset. However, addressing class imbalance through techniques like resampling, cost-sensitive learning, or using alternative evaluation metrics could further improve the model's performance, especially for the minority class.

f) KNN Model Building and Evaluation [10 marks]

Fit a KNN model for a range of K values. Provide and examine the confusion matrix.

Generate and provide a classification report showing precision, recall, F1 score, and overall accuracy to evaluate your model performance. Discuss the results.

Process to Find Best K Value and Build KNN model

1. Range of K Values: The range of K values is defined from 1 to 14.
2. Initialize Lists: Lists are initialized to store training and test scores for each K value.
3. Loop through K Values: The KNN model is trained and evaluated for each K value, and the accuracy scores are stored.
4. Find Max Scores: The maximum train and test scores and their corresponding K values are found and printed.
5. Plot Accuracy Scores: The training and test accuracy scores are plotted against the K values.
6. Select Best K: The best K value is selected based on the highest test accuracy.
7. Train and Evaluate Best KNN Model: The KNN model is retrained with the best K value, and its performance is evaluated.
8. Print Evaluation Metrics: The accuracy, confusion matrix, and classification report for the best K value are printed.
9. Plot Confusion Matrix: The confusion matrix for the best K value is plotted.

Line Graph and List of KNN Model Accuracy with K Value

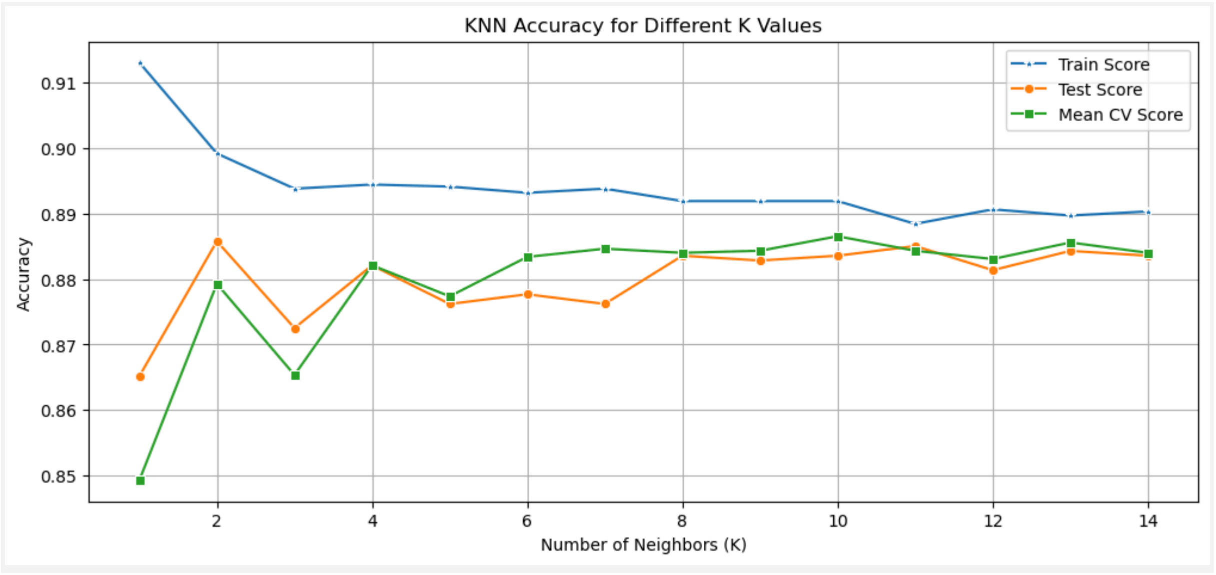


Figure A.f.1 KNN Train Score, Test Score and Mean of CV Score for Different K Values

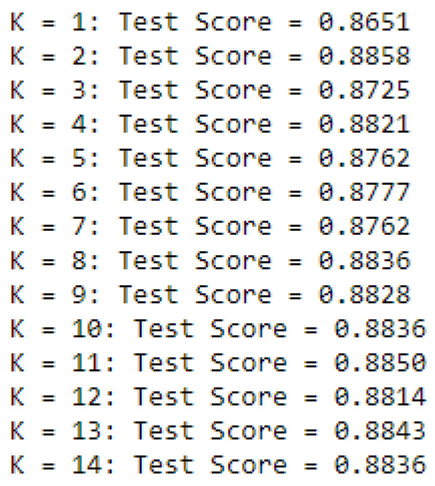
Explanation:

The graph shows the accuracy score of a KNN model for different values of K (number of neighbors).

Also, there is a K-fold Cross Validation line. Each k for KNN, the K-fold Cross Validation mean accuracy is represented.

1. **Train Score (Blue Line):**
    - Represents the accuracy of the training data.
    - When K is very small (e.g., 1), the model fits the training data perfectly (overfitting).
    - As K increases, the train score decreases slightly.
2. **Test Score (Orange Line):**
    - Represents the accuracy on unseen test data.
    - When the k is 1, the test score is lowest in this model.
    - After K reaches an optimal range (around 8-10), the test score remains relatively stable.

3. **Mean CV Score (Green Line):**
    - Represents the mean accuracy from cross-validation (CV) folds.
    - Cross-validation helps estimate how well the model will perform on unseen data.
    - Similar to the test score and those are not much different.
4. **Overfitting and Underfitting:**
    - Small K (e.g., 1) leads to overfitting (high train score, low test score).
    - Large K (e.g., 14) may lead to underfitting (similar train and test scores, but lower overall accuracy).



```
K = 1: Test Score = 0.8651
K = 2: Test Score = 0.8858
K = 3: Test Score = 0.8725
K = 4: Test Score = 0.8821
K = 5: Test Score = 0.8762
K = 6: Test Score = 0.8777
K = 7: Test Score = 0.8762
K = 8: Test Score = 0.8836
K = 9: Test Score = 0.8828
K = 10: Test Score = 0.8836
K = 11: Test Score = 0.8850
K = 12: Test Score = 0.8814
K = 13: Test Score = 0.8843
K = 14: Test Score = 0.8836
```

Figure A.f.2 KNN Accuracy for Different K Values

Explanation:

In the above list, when the k value is 2, the accuracy would be the highest test score. So, the best k value is 2 then, it is used for KNN classification model.

However, there are some considerations to keep in mind regarding the choice of an even or odd K value:

1. Tie Situations:

- When K is an even number, there is a possibility of ties when voting for the majority class among the neighbors. For example, if $K=4$ and the 4 nearest neighbors consist of 2 instances of class A and 2 instances of class B, the algorithm may encounter a tie. Some implementations of KNN handle ties by using different strategies, such as choosing the class of the nearest neighbor or randomly selecting one of the tied classes. It's important to check how the specific KNN implementation in your library handles ties.

2. Odd K Value:

    - Using an odd K value can help avoid ties in the majority voting, as there will always be a majority class when K is odd. For example, if $K=5$, and the neighbors consist of 3 instances of class A and 2 instances of class B, class A will be chosen as the majority class.

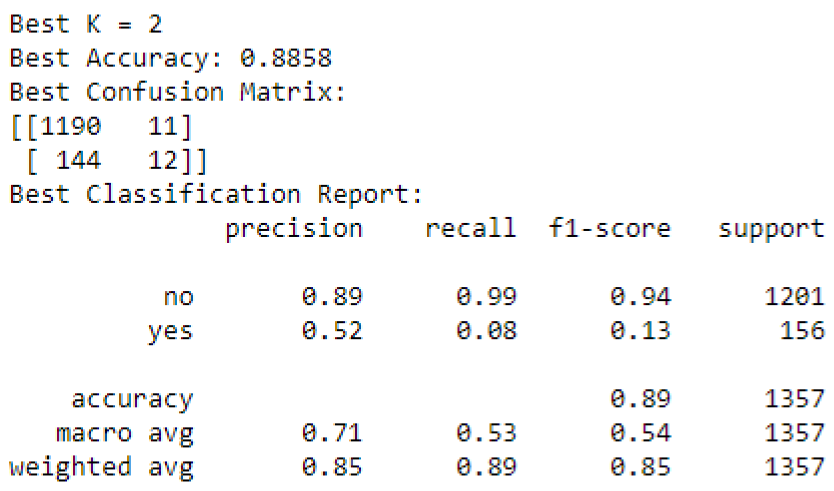
3. Empirical Performance:

    - The choice between even and odd K values should also be guided by empirical performance. As your results indicate, $K=2$ provided the best performance based on cross-validation and test accuracy. This suggests that for your specific dataset and problem, $K=2$ is a suitable choice, regardless of the potential for ties.

In summary, while using an even K value can introduce the possibility of ties, it is not inherently problematic as long as the implementation of KNN handles ties appropriately. If $K=2$ gives the best performance for your model, it is perfectly acceptable to use it.

Discussion of Results

Classification Report and Confusion Matrix of KNN Classification Model



```
Best K = 2
Best Accuracy: 0.8858
Best Confusion Matrix:
[[1190   11]
 [ 144   12]]
Best Classification Report:
              precision    recall  f1-score   support

          no       0.89      0.99      0.94      1201
         yes       0.52      0.08      0.13       156

    accuracy                           0.89      1357
   macro avg       0.71      0.53      0.54      1357
weighted avg       0.85      0.89      0.85      1357
```

Figure A.f.3 KNN Classification Report

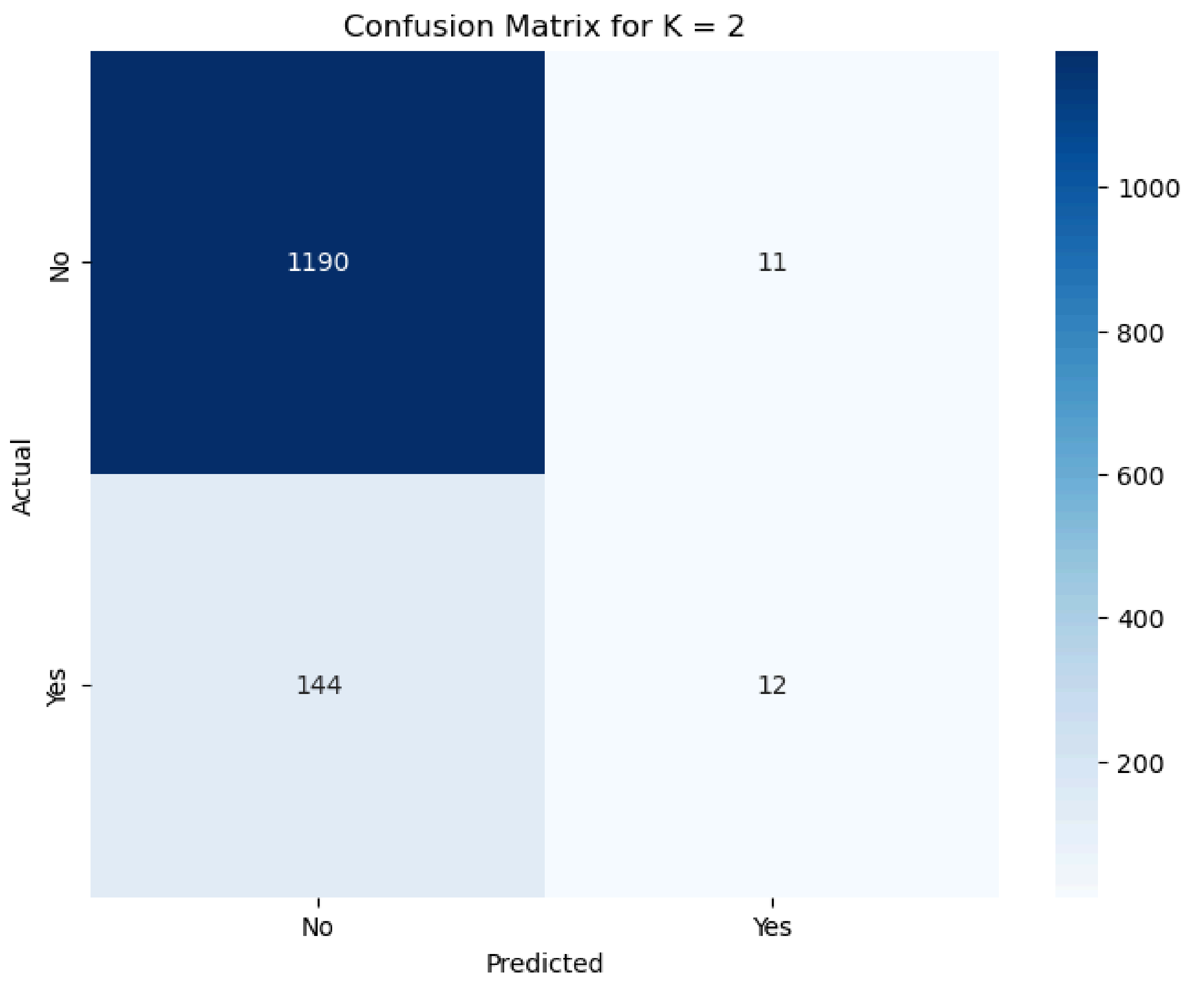


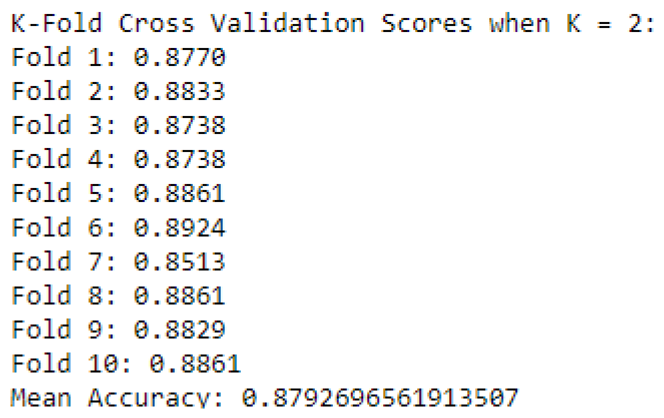
Figure A.f.4 Confusion Matrix KNN Classification Model when K is 2.

Explanation:

1. **Accuracy:**
    - Accuracy measures the overall correctness of the model's predictions across all classes.
    - Formula: Accuracy = (True Positives + True Negatives) / (Total Predictions)
    - Accuracy = (1190 + 12) / 1357 ≈ 89%
2. **Precision:**
    - Precision measures the proportion of true positive predictions among all positive predictions made by the model.
    - Formula: Precision = (True Positives) / (True Positives + False Positives)
    - For the class 'no': Precision = 1190 / (1190 + 144) ≈ 89%
    - For the class 'yes': Precision = 12 / (12 + 11) ≈ 52%
3. **Recall:**
    - Recall measures the proportion of true positive predictions among all actual positive instances in the dataset.
    - Formula: Recall = (True Positives) / (True Positives + False Negatives)
    - For the class 'no': Recall = 1190 / (1190 + 11) ≈ 99%
    - For the class 'yes': Recall = 12 / (12 + 144) ≈ 8%
4. **F1-Score:**
    - F1-Score is the harmonic mean of precision and recall, providing a balance between the above measurements.
    - Formula: F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

- For the class 'no': F1-Score = 2 * (0.89 * 0.99) / (0.89 + 0.99) ≈ 94%
- For the class 'yes': F1-Score = 2 * (0.52 * 0.08) / (0.52 + 0.08) ≈ 13%

Evaluation with K-fold Cross Validation



```
K-Fold Cross Validation Scores when K = 2:
Fold 1: 0.8770
Fold 2: 0.8833
Fold 3: 0.8738
Fold 4: 0.8738
Fold 5: 0.8861
Fold 6: 0.8924
Fold 7: 0.8513
Fold 8: 0.8861
Fold 9: 0.8829
Fold 10: 0.8861
Mean Accuracy: 0.8792696561913507
```

Figure A.f.5 List of K-Fold Cross Validation Accuracy Score when K is 2.

Explanation:

- **K-Fold Cross-Validation Scores:**
    - The K-fold cross-validation scores represent the accuracy of the KNN model on different subsets of the data. Each fold represents a different split of the data into training and validation sets. The mean accuracy provides an overall assessment of the model's performance across all folds.
    - Mean Accuracy: 87.92%

Comparison and Interpretation:

- The K-fold cross-validation scores provide an average accuracy of 87.92% across different subsets of the data, indicating that the KNN model generalizes well to unseen data.
- The KNN classification report shows that the model achieves relatively high precision, recall, and F1-score for the class 'no', indicating good performance in predicting instances of the majority class. However, the performance metrics for the class 'yes' are lower, suggesting that the model struggles to accurately classify instances of the minority class.

- The confusion matrix reveals that the model correctly identifies a large number of instances belonging to the majority class ('no') but performs less well in identifying instances of the minority class ('yes'). This is reflected in the lower precision, recall, and F1-score for the class 'yes' compared to the class 'no'.

Overall, while the model demonstrates good overall accuracy and performs well for the majority class, there is room for improvement in correctly classifying instances of the minority class. Further tuning of the model or addressing class imbalance may help improve performance for the minority class.

g) Model Comparison [5 marks]

Compare the performance of your KNN and NB models.

Discuss your findings

Overall Accuracy

- **Naive Bayes Model:**

Best Accuracy: 83.86%

- **KNN Model:**

Accuracy: 88.58%

Analysis:

The KNN model shows a slightly higher accuracy compared to the Naive Bayes model.

- Confusion Matrix:

  Naive Bayes:

  - True Negatives (TN): 1121
  - False Positives (FP): 80
  - False Negatives (FN): 139
  - True Positives (TP): 17

  KNN:

  - True Negatives (TN): 1190
  - False Positives (FP): 11
  - False Negatives (FN): 144
  - True Positives (TP): 12

KNN significantly reduces the number of false positives (11 vs. 80) compared to Naive Bayes. However, it slightly increases the number of false negatives (144 vs. 139).

- Precision, Recall, and F1-score:

  Class 'no':

  - Both models have the same precision (0.89) for class 'no'.
  - KNN has a higher recall (0.99) compared to Naive Bayes (0.93), indicating it correctly identifies more true 'no' instances.
  - Consequently, KNN has a higher F1-score (0.94) for class 'no' compared to Naive Bayes (0.91).

  Class 'yes':

  - KNN has a higher precision (0.52) compared to Naive Bayes (0.18), meaning it is better at correctly identifying instances that are truly 'yes'.
  - Both models have low recall for class 'yes' (KNN: 0.08, Naive Bayes: 0.11), indicating poor performance in identifying all 'yes' instances.
  - Both models have a similar F1-score for class 'yes' (0.13), indicating overall poor performance in predicting 'yes' instances.

Computational Efficiency and Practical Considerations

- **KNN**: KNN can be computationally intensive, especially with large datasets, because it requires storing all training data and computing distances for each prediction. However, it is a non-parametric model that can capture complex relationships.

- **Naive Bayes**: Naive Bayes is computationally efficient and fast, making it suitable for large datasets. It is a parametric model that works well with a large number of features and is easy to implement and interpret.

Conclusion:

- KNN: Performs better overall, especially for the majority class 'no', with higher accuracy and better precision and recall. However, it still struggles with the minority class 'yes', similar to Naive Bayes.

- Naive Bayes: While having lower accuracy, it still performs reasonably well for the majority class but struggles more with the minority class compared to KNN.

Given the better overall performance metrics, KNN appears to be the more effective model for this dataset, particularly if the focus is on maximizing overall accuracy and performance for the majority class. However, both models exhibit significant challenges with the minority class 'yes', suggesting that further techniques, such as data balancing or different modeling approaches, may be needed to improve minority class prediction.

Part B: Exploring Artificial Neural Networks

a) Activation Function and Learning Rate in MLP

Explain the role of an activation function and learning rate in building a Multilayer Perceptron (MLP).

First of all, what's Artificial Neural Networks and MLP: Ann is short for artificial neural network (ANN) which is a computational model in machine learning inspired by the intricate arrangement and functionality of neurons within the human brain. It comprises interconnected units termed artificial neurons, structured into layers to mimic biological neural networks.

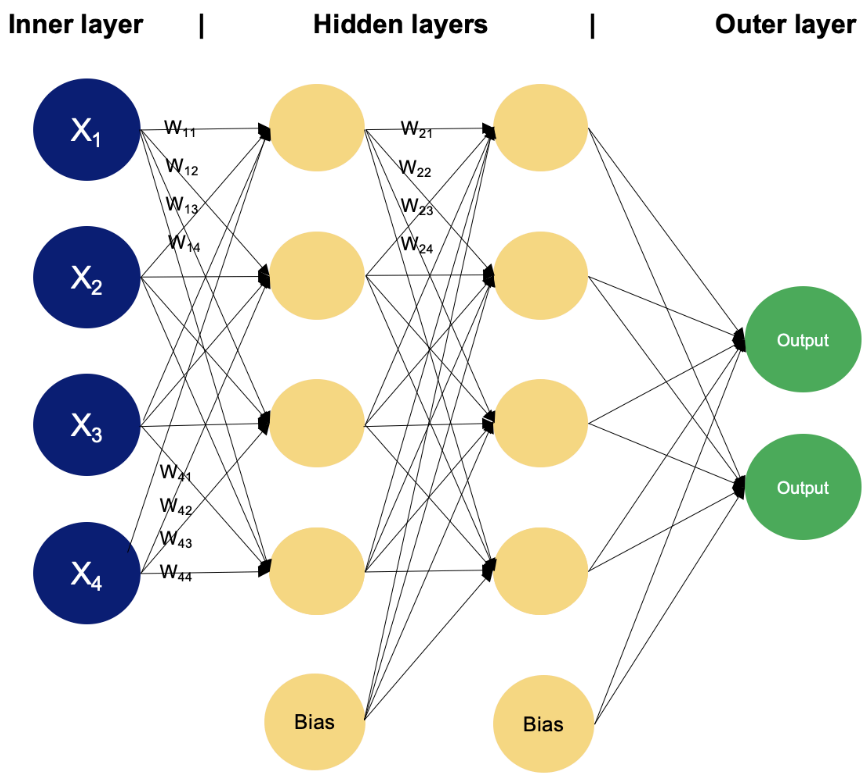


Figure B.a.1 An example of MLP

Multi-layer perceptron (MLP) is a special type of ANN. It denotes an artificial neural network architecture comprising numerous layers of interconnected neurons. Within the MLP framework, neurons customarily employ nonlinear activation functions, facilitating the network's capacity to discern intricate patterns within datasets (Sejal

Jaiswal, 2024). Here’s a visualized example of the MLP model shown above.

Then in building a Multilayer Perceptron (MLP), the activation function and the learning rate controls the learning result of the neural network and how well the Multilayer Perceptron can perform.

According to Goodfellow, Bengio, Courville and Bengio (2016), activation function is a mathematical function that provides a non-linearial method for the network’s classification.

Without an activation function, an MLP would only be able to perform linear transformations, even with multiple hidden layers, and thus would fail to learn complex patterns. The activation function will allow the neural network to be able to model complicated relationships from the dataset, especially those having non-linear mappings.

Apart from non-linearity, the activation function can regularize the output range. It compresses the output to a specific range (Depends on the specific function used but usually 0 to 1 or -1 to 1). This helps prevent excessively large or small output values, maintaining numerical stability (Sejal, 2024).

Learning rate is a hyperparameter that dictates the scale of adjustments made to a neural network's weights throughout the training process. It regulates the size of weight modifications in each cycle of the optimization algorithm.

The learning rate determines the step size for weight updates. While a higher learning rate promotes quicker convergence, it also heightens the risk of overfitting by repeating the same training data. Conversely, a lower learning rate can make the training process more stable but may take longer to converge.

The learning rate is also needed to balance exploring new solutions and exploiting the current solution. An appropriate learning rate helps the model effectively explore the solution space while fine-tuning with the existing information. A good model will need to find a balanced learning rate.

In this report, the activation function used will be `ReLU` (Rectified Linear Unit) which is the default of `MLPClassifier`. The learning rate will be using the default setting which is controlled by `Adam` in `MLPClassifier`. The scaling method used for MPL is `StandardScaler()` which is different from `MinMaxScaler()` in Part A due to the significant amount of outliers and after testing the best accuracy is slightly higher when using `StandardScaler()`.

b) Baseline Model with MLPClassifier

Use the sklearn.MLPClassifier with default parameter values and a single hidden layer with k neurons ($k \leq 25$). Determine and report the best number of iterations that give the highest accuracy. Use this classification accuracy as a baseline for comparison in later parts of this question.

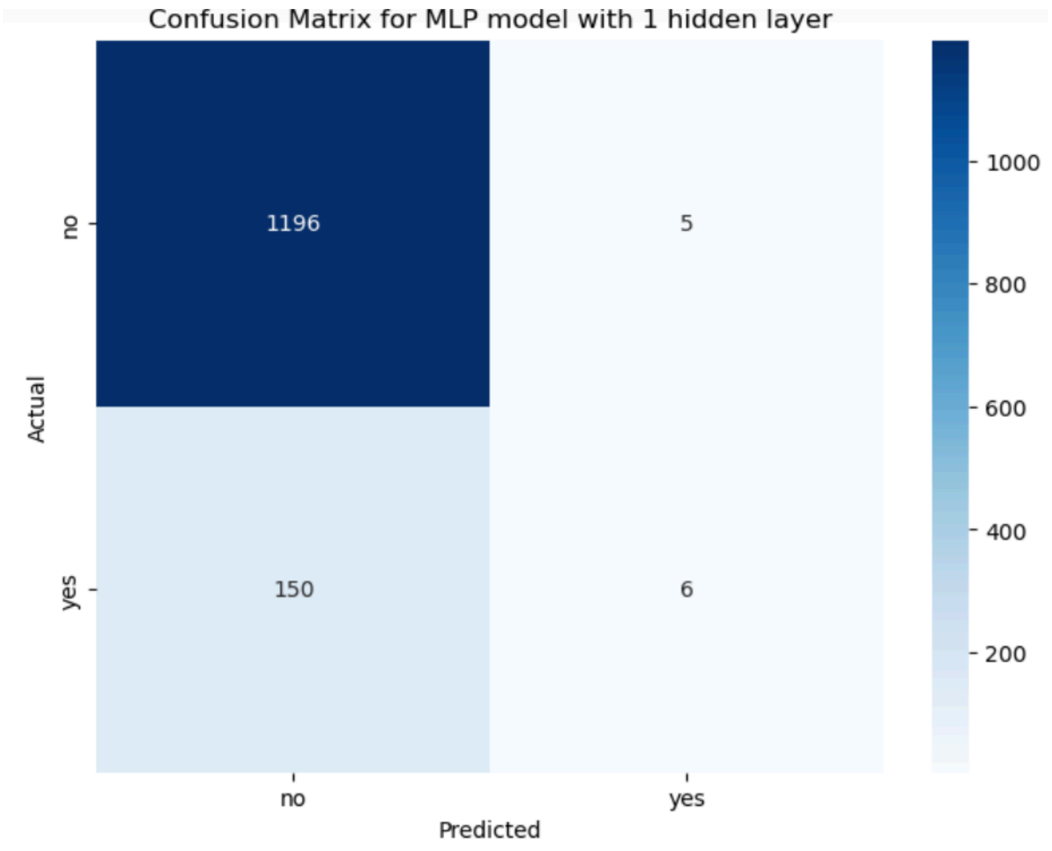
A baseline model with an MLP (Multi-Layer Perceptron) classifier typically refers to a simple implementation of an MLP classifier that serves as a starting point for more complex model development and comparison. This baseline model often involves minimal feature engineering and hyperparameter tuning, providing a reference point for evaluating the performance of more balanced models (Preethi, 2023).

There are several steps for creating a baseline model:

1. Data Preparation: Pre-processing feature selection of dataset has been done in Part A. The dataset were separated into the training part and testing part.
2. Define the structure: use a single hidden layer in this case.
3. Model compilation: The default method for MLPclassifier in python is cross-entropy and Adam method.
4. Model Training: The training data were used to train the MLP model, then cross validation were used to compare the performance for models with different k values.
5. Model Comparison and Evaluation: The best k were found and the accuracy were printed out.

In summary, to find the number of neurons (k) in the single hidden layer with highest accuracy, the cross-validation can be used to measure the performances of models with different values of k. So, each k within the provided range 1~25 will be tested using a for loop to run the cross_val_score method. The best result found out is when k = 13, which got the corresponding highest accuracy 88.58%. This suggests that, for this specific problem and dataset, a hidden layer with 13 neurons is the optimal choice for the model's performance.

Here's the confusion matrix printed out below showing the performance of this MPL with 13 neurons. From the matrix we can see that the model does not perform well in general. It has a high accuracy but extremely low on precision score, recall score and F1 score. The reason for that is the model is not performing well on “yes”. The reason for that is the unbalanced dataset and MLP model is weak on predicting unbalanced data.



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

          no       0.89      1.00      0.94      1201
         yes       0.55      0.04      0.07       156

    accuracy                           0.89      1357
   macro avg       0.72      0.52      0.51      1357
weighted avg       0.85      0.89      0.84      1357


Accuracy Score:  0.8857774502579219
Precision: 0.5454545454545454
Recall: 0.038461538461538464
F1 score: 0.0718562874251497
```

Figure B.b.1. Confusion Matrix for 1 layer MLP

However, this result can serve as a starting point for further improvements and optimizations of the model. Other hyperparameters such as learning rate, regularization parameters, etc., can be adjusted to further enhance the model's performance. Apart from that, exploring other types of models or ensemble methods may also be considered to improve predictive performance.

c) Tracking Loss Value

Enable the loss value to be shown on the training segment and track the loss as a function of the iteration count. Explain any observed discrepancies between loss value and error value over consecutive iterations.

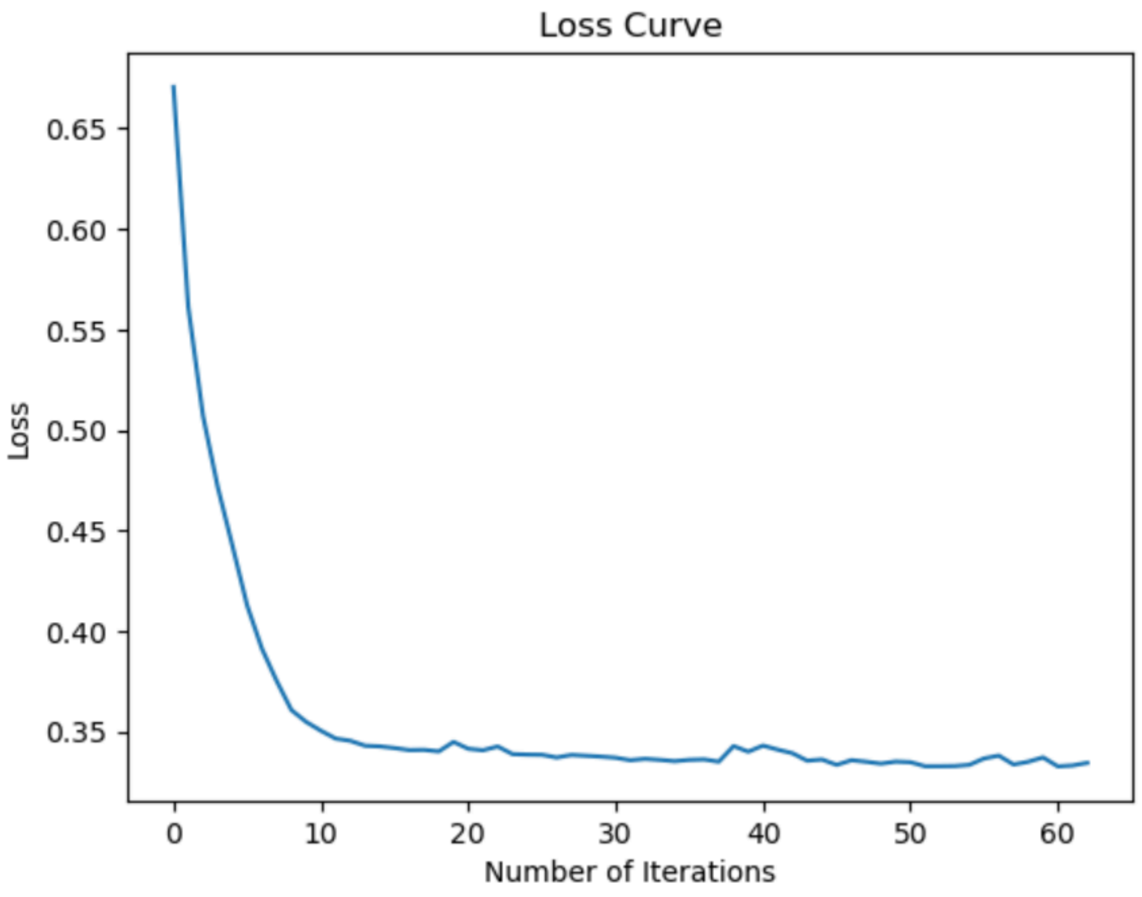


Figure B.c.1 Loss Value Graph

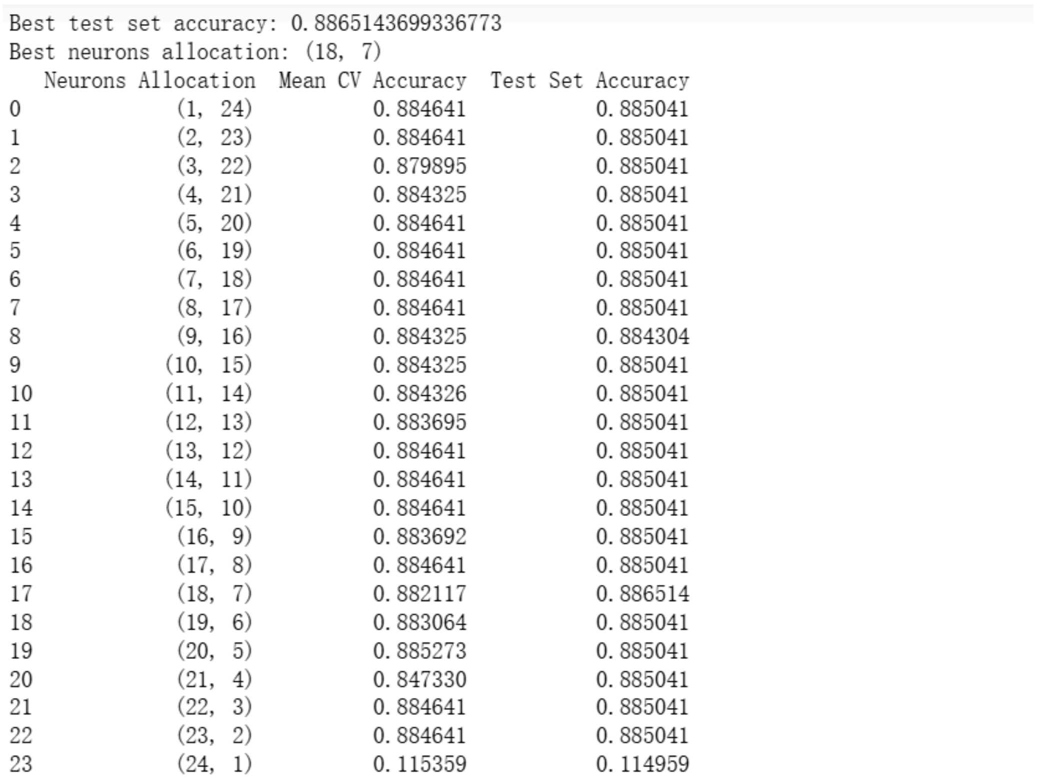
The Loss Value is regarded as a metric that encapsulates the cumulative errors made by our model. It serves as an indicator of the model's performance, reflecting how effectively (or ineffectively) the model is performing. When errors are abundant, the loss value increases, indicating suboptimal performance. Conversely, a lower loss value signifies improved performance, indicating that the model is making fewer errors and performing more effectively (Martin, 2024).

In the chart above showing the loss value, the X axis represents how many times the model has been iterated, the Y axis represents the loss value. The shape of the line shown in the chart is stable and does not have any variations at the beginning. Then it has had some rise and fall since 19 iterations. It indicates that the dataset does have some noises and the model is a bit unstable in general. Then from the information showing in the figure, we can see that the higher number of times it has been iterated, the less loss value will occur until 19 iterations. After that the number of loss values varies and it keeps around the same number. It shows the model's best performance achieved before 19.

The term "error value" typically refers to performance metrics of a model on a validation or test dataset, such as accuracy, precision, recall, etc., which reflect the model's performance on unseen data, i.e., its generalization ability. From the plotted loss curve above, it is not able to directly observe information related to error value. While both loss value and error value are metrics for assessing model performance, they are not always aligned. Specifically, the loss value is a continuous, differentiable function that measures the discrepancy between model predictions and actual labels, whereas error value is often a discrete metric, such as classification accuracy, that evaluates the model's classification accuracy.

d) Experimenting with Two Hidden Layers

Experiment with two hidden layers and experimentally determine the split of the number of neurons across each of the two layers that gives the highest classification accuracy. In part 1, you had all k neurons in a single layer, in this part you will transfer neurons from the first hidden layer to the second iteratively in step size of 1. Thus, for example in the first iteration, the first hidden layer will have k-1 neurons whilst the second layer will have 1, in the second iteration k-2 neurons will be in the first layer with 2 in the second and so on. Summarizing your classification accuracy results in a 25 by 2 table with the first column specifying the combination of neurons used (e.g., 12, 13) and the second column specifying the classification accuracy.



```
Best test set accuracy: 0.8865143699336773
Best neurons allocation: (18, 7)
   Neurons Allocation  Mean CV Accuracy  Test Set Accuracy
0             (1, 24)          0.884641           0.885041
1             (2, 23)          0.884641           0.885041
2             (3, 22)          0.879895           0.885041
3             (4, 21)          0.884325           0.885041
4             (5, 20)          0.884641           0.885041
5             (6, 19)          0.884641           0.885041
6             (7, 18)          0.884641           0.885041
7             (8, 17)          0.884641           0.885041
8             (9, 16)          0.884325           0.884304
9            (10, 15)          0.884325           0.885041
10           (11, 14)          0.884326           0.885041
11           (12, 13)          0.883695           0.885041
12           (13, 12)          0.884641           0.885041
13           (14, 11)          0.884641           0.885041
14           (15, 10)          0.884641           0.885041
15            (16, 9)          0.883692           0.885041
16            (17, 8)          0.884641           0.885041
17            (18, 7)          0.882117           0.886514
18            (19, 6)          0.883064           0.885041
19            (20, 5)          0.885273           0.885041
20            (21, 4)          0.847330           0.885041
21            (22, 3)          0.884641           0.885041
22            (23, 2)          0.884641           0.885041
23            (24, 1)          0.115359           0.114959
```

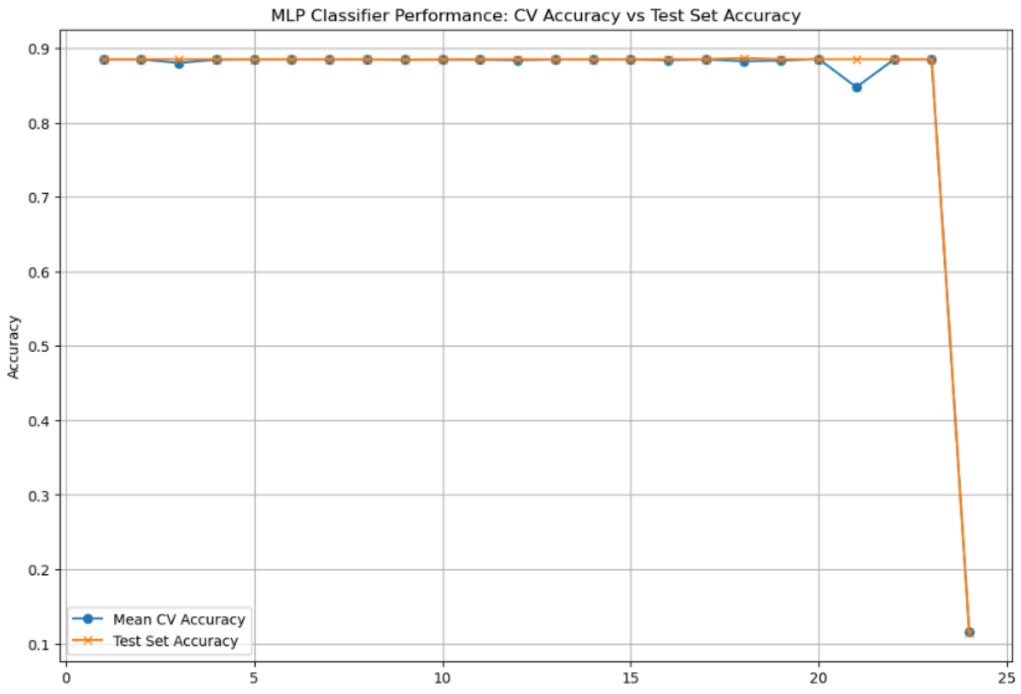
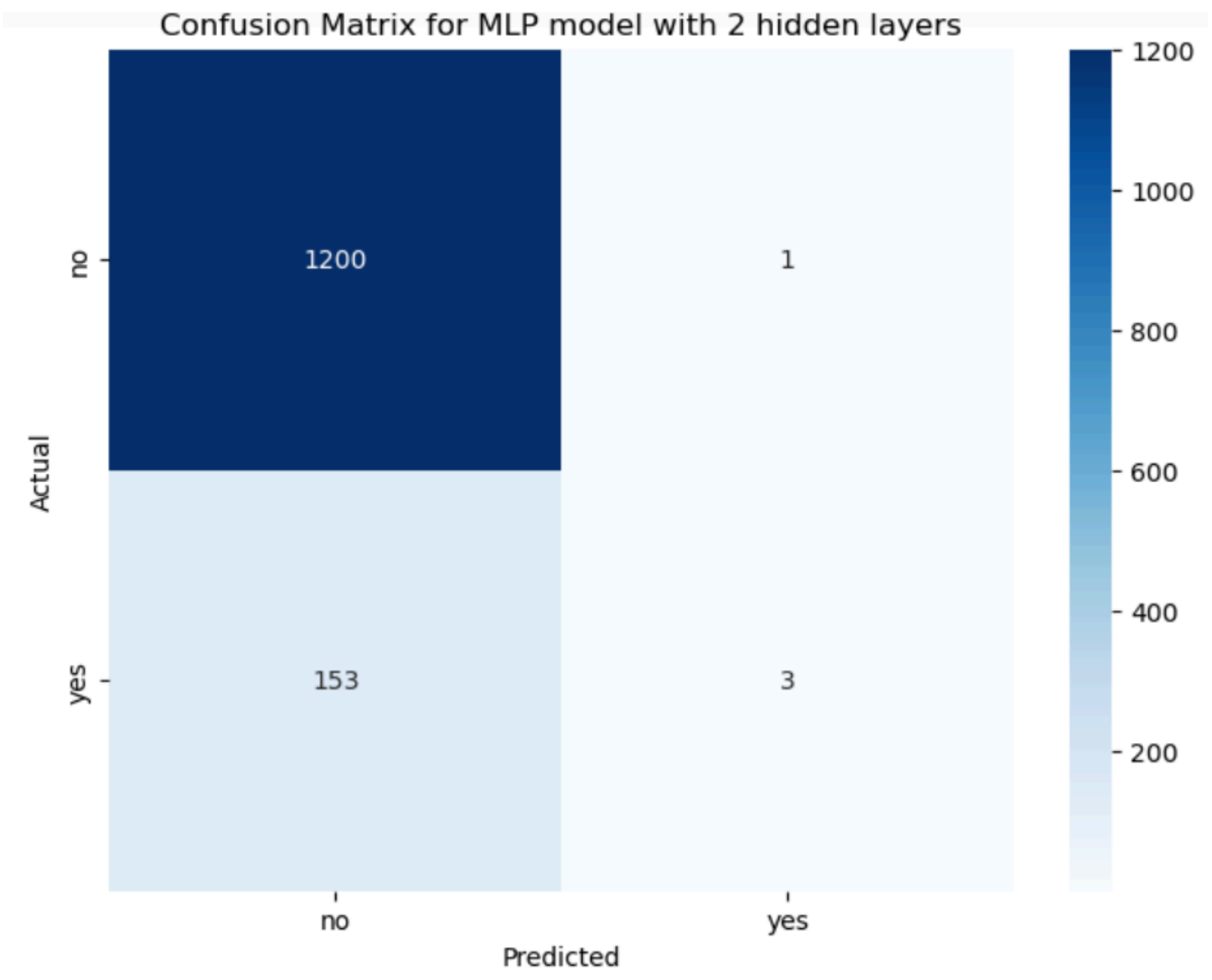


Figure B.d.1. Results of testing different numbers of neurons in MLP

This problem is to experimentally determine the optimal neuron allocation between the two hidden layers to improve classification accuracy. Specifically, we will gradually move neurons from the first hidden layer to the second hidden layer, one neuron at a time. For instance, in the first iteration, the first hidden layer will have k−1 neurons, and the second hidden layer will have 1 neuron; in the second iteration, the first hidden layer will have k−2 neurons, and the second hidden layer will have 2 neurons; and so on. This way, we can try different allocations of neurons between the two hidden layers and experimentally determine which distribution provides the best classification accuracy.

Here's the results of the process finding out the best distribution of the first hidden layer and second hidden layer showing above. The results table shows all different neuron allocation schemes and their corresponding classification accuracy. For example, when there is 1 neuron in the first hidden layer and 24 neurons in the second hidden layer, the classification accuracy is 0.8846. It can be observed that the classification accuracy varies with different neuron allocation schemes. Apart from (24, 1), all the sets got high scores and the scores are similar to each other. The best distribution will be splitting 18 neurons into the first hidden layer and 7 neurons into the second hidden layer. The accuracy for that will be around 88.65%.

The classification accuracy fluctuates across different neuron allocation schemes but remains relatively high in most cases. Accuracy from all results showing above are only slightly floating within a small range but mostly are the same number: 0.885. It reflects that the model could not get better performance by increasing the number of hidden layers or the number of neurons. So the two layer MLP is not working well with this dataset. Here's the confusion matrix of the model showing below which will indicates more details:



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

          no       0.89      1.00      0.94      1201
         yes       0.75      0.02      0.04       156

    accuracy                           0.89      1357
   macro avg       0.82      0.51      0.49      1357
weighted avg       0.87      0.89      0.84      1357

Accuracy: 0.8865143699336773
Precision: 0.75
Recall: 0.019230769230769232
F1 score: 0.0375
```

Figure B.d.2. Confusion Matrix for 2 layers MLP

Here's the confusion matrix of the MLP model with 18 neurons in the first hidden layer and 7 neurons in the second hidden layer shown above. The matrix indicates that the model has a bad performance on predicting "yes" which is the same issue with the MLP mode with hidden 1 layer. The Recall score and F1 score is lower than the single layer MLP model but the accuracy score, precision score are higher. From the confusion matrix, we can see that the 2 layers MLP model is predicting more cases into a single class and it indicates a worse performance. To sum up, the performance of the MLP model with single hidden layers is slightly better than the MLP model with 2 layers.

In this experiment, it's evident that a more complex model, indicated by a higher number of neurons, doesn't necessarily lead to a better performance. The optimal model, characterized by its highest accuracy, doesn't correspond to the one with the maximum number of neurons. Thus, there's a need to strike a balance between model complexity and performance to avoid overfitting and enhance generalization ability.

e) Explaining Accuracy Variation

From the table created in part d, you will observe the accuracy variation with the split of neurons across the two layers. Give explanations for some possible reasons for this variation.

When the model only has a single hidden layer containing 13 neurons, the model can capture some of the main features in the data, achieving an accuracy of 0.8858. The features of a single-layer network are simpler, train faster, and have a lower risk of overfitting. However, their representation capability is limited, which might prevent them from capturing complex patterns in the data.

According to Mayank (2023), when the model is using two hidden layers or more, it’s called a deep ANN. Here it has 2 hidden layers with 18 neurons in the first layer and 7 neurons in the second layer, a MLP model with more hidden layers and neurons allowing the model to capture and process more complex features. However, the more complex the model is, the higher the risk of overfitting. Here the accuracy of the 2 layers MLP model is lower than the single layer MLP model proved.

Generally, MPL with 2 layers can better allocate tasks: the first layer extracts low-level features, and the second layer combines these features for classification. It can better learn deep patterns in the data, improving the model's ability to generalize to unseen data and thus enhancing classification accuracy. Compared to single-layer networks, this division allows for more efficient use of information.

However, the two hidden layer MLP model is more complex, with more parameters and a greater capacity to represent complex functions. For linear or nearly linear relationships, this additional complexity is unnecessary and can even lead to instability or difficulty in convergence during training. Apart from that, more complex models are more prone to overfitting the training data, especially when the dataset is limited. Simpler models (like the single hidden layer model) are likely to generalize better to unseen data.

Then the single layer networks are simpler, thus having a lower risk of overfitting. Even their representation capability is limited, hindering the capture of complex patterns, but due to that a single layer network works better when the data is linearly separable or nearly linearly separable. A single hidden layer MLP model is essentially a universal function approximator capable of approximating any continuous function, including linear functions.

The results showing the MPL model with a single hidden layer are performing well in general. However, it doesn't have much differences with the MPL model with only a single hidden layer: there's only 3.44% variance. Here are some possible reasons to explain what might cause the small variance:

1. Dataset Characteristics: The dataset might be relatively simple, with less complex relationships between features. Following that, a single-layer neural network might already adequately capture the data patterns without the need for additional layers to perform complex feature abstractions and combinations.
2. Shape of dataset: The single layer MPL model is fitting better with linearly shaped dataset while MPL models with more layers are fitting better with non-linearly shaped models.
3. Feature Selection: If the features used are already capable of accurately describing the data patterns, then adding extra hidden layers will provide limited performance improvement. Even if a two-layer hidden model can extract higher-level features, if these features do not significantly contribute to the classification task, the performance improvement will not be substantial.
4. Model Capacity: "Entities should not be multiplied unnecessarily." The single-layer neural network may already have sufficient model capacity to fit the training data. In this case, increasing the depth of the network may not result in a significant performance improvement.
5. Risk of Overfitting: The increased complexity of the two-layer neural network may increase the risk of overfitting, especially with relatively significant noise.
6. Optimization Algorithm and Hyperparameter Tuning: For the small amount of variance in this scale, there might exist multiple suitable model configurations and hyperparameter settings. Even with increased network depth, without thorough hyperparameter tuning and optimization, there might be limitations to the performance improvement.
7. Local Optima: The single-layer neural network might have already converged to a local optimum, and increasing the network depth might not lead to better performance. In such cases, changing it to another model or using ensemble learning will be helpful to get a better performance.

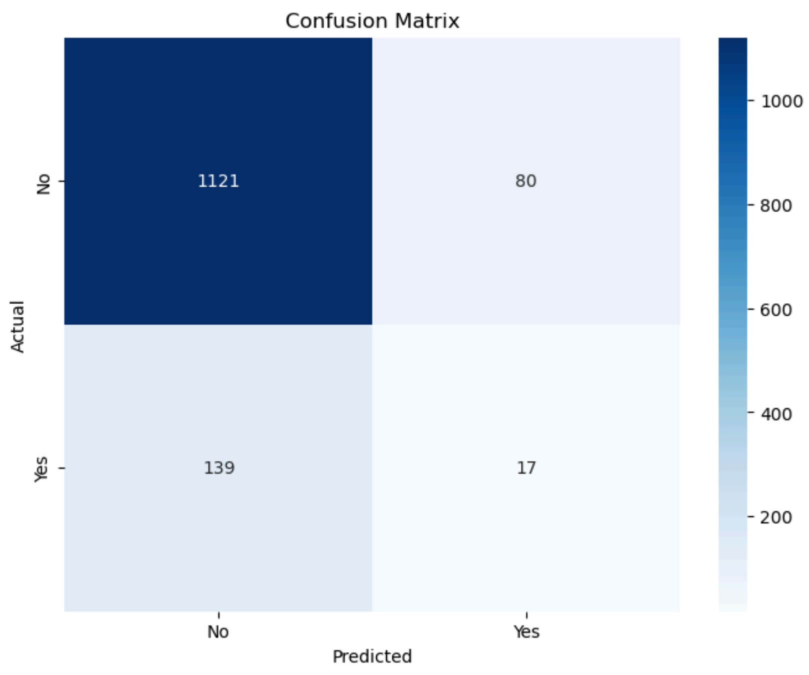
f) Comparing MLP Classifier Performance

Compare the performance of the MLP Classifier with other classifiers on your dataset in part A. Choose the best-performing model and explain why you chose it. Discuss your findings from the experiments and provide your opinion on these classifiers.

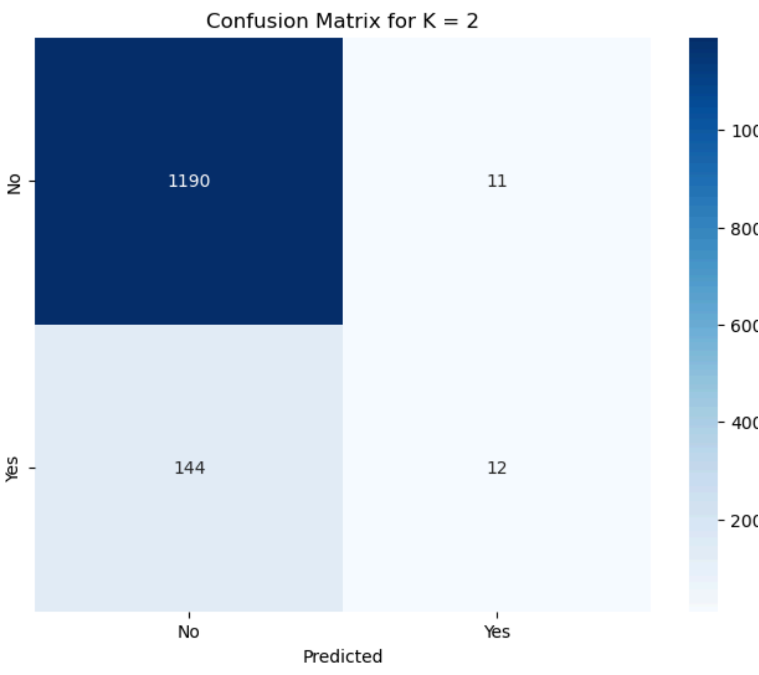
To summarize, the KNN model has an 88.58% on accuracy and 13.41% on its F1 score, while Naive Bayes model has accuracy of 83.86% and %13.44 of F1 score. Then the MPL model with 2 hidden layers has 88.65% accuracy, 3.75% on F1 score. The MPL with 1 layer has 88.58% accuracy and 7.19% F1 score. The numbers of scores and the confusion matrix of these 4 models are shown below.

Model	Accuracy	F1 score
Naive Bayes	83.86%	13.44%
KNN	88.58%	13.41%
MPL 1 layer	88.58%	7.19%
MPL 2 layers	88.65%	3.75%

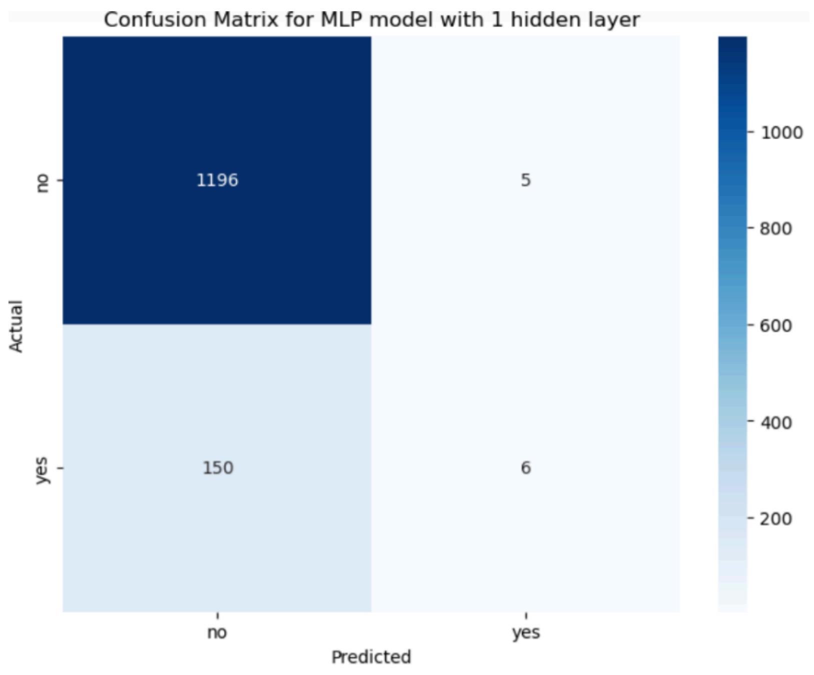
Table 1. Performance of Models



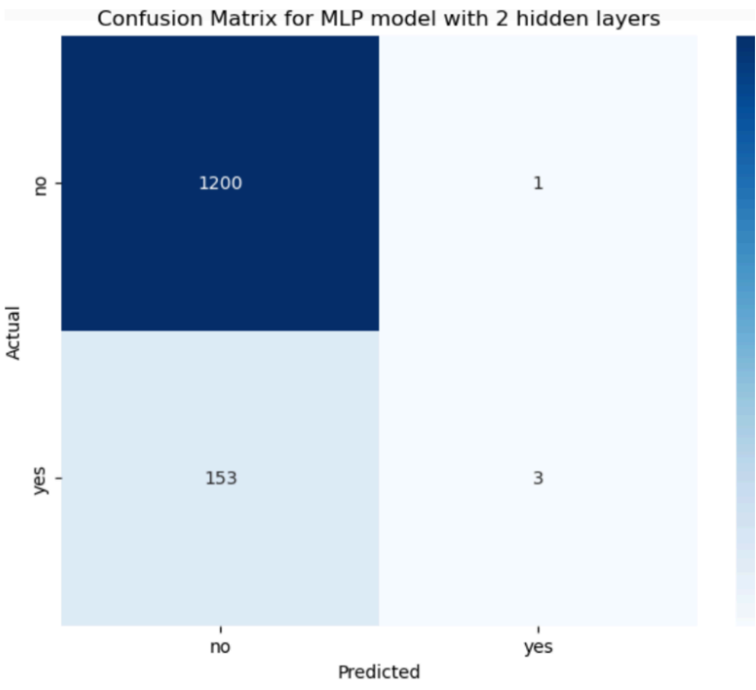
Naive Bayes



KNN



MLP with 1 layer



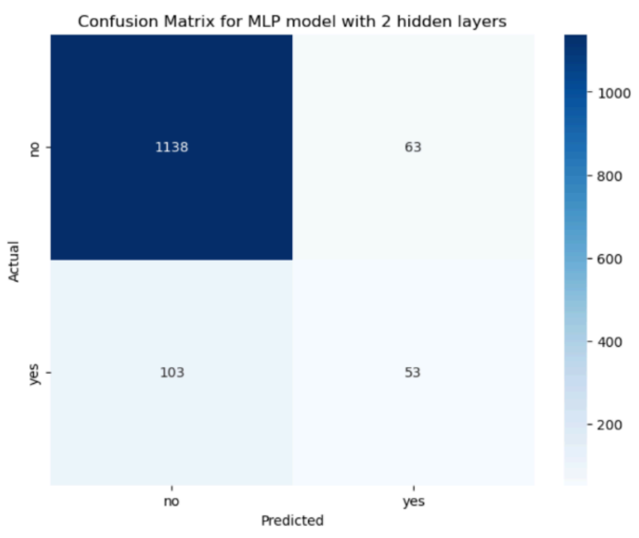
MLP with 2 layers

Figure B.f.1 Confusion Matrix of all models

During all these models, the MLP model takes the longest time for processing. Then it’s Naive Bayes model. The KNN model does cost the shortest time to process. However, the dataset is not too large that the time training a model will be a concern. So this won’t be involved in the consideration here.

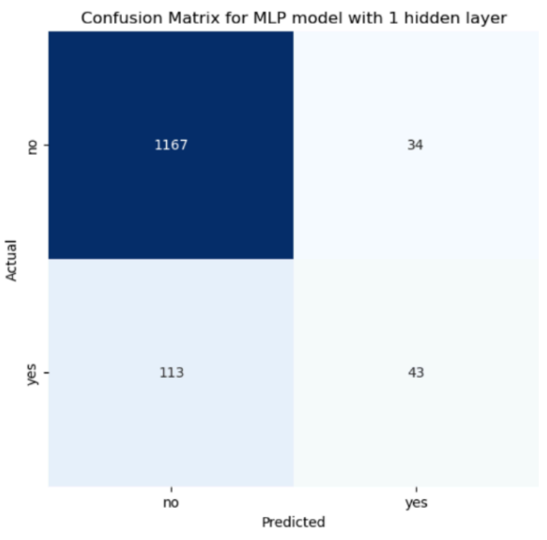
Comparing these four models, the MPL model with two hidden layers does have the highest accuracy. However, the variance of accuracy between each model is very tiny. The Naive Bayes model has the highest F1 score with the highest number predicting “yes” correctly. The MLP model with 2 layers does have the highest number of predicting class “No” correctly but lowest number of predicting class “yes” correctly. All models are not performing well on predicting class “yes” correctly but MLP models are much worse than other two models on that.

If, when using the same feature selection method, KNN and Naive Bayes can function properly while MLP fails to make effective predictions, it might be due to MLP's heightened sensitivity to nonlinear relationships within the feature space. Selecting only a small number of features may not capture enough complexity for MLP to operate effectively. In such a scenario, if it’s not using feature selection for MLP and instead utilizing all available features. This would provide MLP with more information to better learn the intricate relationships within the data. The confusion matrix for MLP models without feature selection are showing below:



```
accuracy: 0.8776713338246132
precision: 0.45689655172413796
recall: 0.33974358974358976
F1 score: 0.3897058823529412
```

MLP with 2 layers (entire dataset)



```
accuracy: 0.8916728076639646
precision: 0.5584415584415584
recall: 0.27564102564102566
F1 score: 0.36909871244635195
```

MLP with 1 layer (entire dataset)

Figure B.f.2 MLP models using entire dataset

Following that, exploring alternative feature selection methods or adjusting MLP model parameters could be beneficial in enhancing its performance. Or feature selection might not be the best option for MLP models, especially when only a limited number of features are selected. Hence, in some imbalanced dataset like this, employing all features for training MLP models could lead to improved performance.

As to efficiently compare these 4 models, F1 score will be a good measure. Accuracy is a good metric when the classes in the dataset are balanced, meaning there are roughly equal numbers of instances in each class. The F1 score is especially useful when the classes are imbalanced and when you want to find a balance between precision and recall. In this dataset, one class is much more frequent than others, and a model that predicts the majority class can achieve high accuracy simply by ignoring the minority class (Shung, 2018). In such a case, F1 score, by balancing precision and recall, offers a more comprehensive assessment of a model's performance, especially when correctly predicting minority classes is important. Therefore, the F1 score is a more reliable indicator of model performance.

KNN and Naive Bayes have much higher F1 scores compared to MLP models. So the best model fitting this dataset will be one of them. KNN and Naive Bayes have only 0.03% variance on F1 score but 4.72% on accuracy rate. Then as discussed in Part A comparison, KNN is the more effective model for this dataset, especially when the primary goal is to maximize overall accuracy and performance for the prediction.

Reference:

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep Learning. Retrieved in May 2024 from http://neuralnetworksanddeeplearning.com/about.html

Jaiswal, Sejal. (2024). Multilayer Perceptrons in Machine Learning: A Comprehensive Guide. Retrieved May 2024 from https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning

Prakash, Preethi. (2023). Understanding Baseline Models in Machine Learning. Retrieved May 2024 from https://medium.com/@preethi_prakash/understanding-baseline-models-in-machine-learning-3ed94f03d645

Riva, Martin. (2024). Interpretation of Loss and Accuracy for a Machine Learning Model. Retrieved May 2024 from https://www.baeldung.com/cs/ml-loss-accuracy#:~:text=Loss%20is%20a%20value%20that%20represents%20the%20summation,lower%20it%20is%2C%20the%20better%20our%20model%20works

Mayank, B. (2023). An Overview on Multilayer Perceptron (MLP). Retrieved May 2024 from https://www.simplilearn.com/tutorials/deep-learning-tutorial/multilayer-perceptron#:~:text=A%20fully%20connected%20multi%2Dlayer,a%20feedforward%20artificial%20neural%20network.

Shung, Koo Ping. (2018). Accuracy, Precision, Recall or F1? Retrieved May 2024 from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9