DSCI 4411-01 Fall 2023

Project

Bank Marketing Data Analysis

Group members:

Hana Ahmed Abouhussein (900200000)

Joumanah Elkhouly (900201019)

Laila Ahmed Hamza (900201723)

Maya Hany Elshweikhy (900204233)

Dr. Seif Eldawlatly

10 December 2023

Table of Contents

1. Introduction

Financial institutions, such as banks, provide clients with the opportunity to make low-risk investments in the form of term deposits. Clients subscribe to a term deposit, also known as a certificate of deposit, by placing a certain amount of money in a sealed account, which can only be accessed at a predetermined maturity date at the risk of incurring penalties to their returns. This financial product offers its subscribers stable returns in the form of fixed interest, which can be given out in segments at regular intervals or as a lump sum at maturity. Through term deposits, clients are also able to preserve their capital's nominal value amidst fluctuating market conditions.

Banks often campaign for term deposits since they represent a stable funding source that allows them to manage their day-to-day operations and lending services. In addition to contributing to the bank's liquidity management, term deposits are also a means of retaining clients with the appeal of a secure, low-risk investment opportunity. Campaigns typically involve contacting potential subscribers and selling them on the bank's offered rates or its special promotions. These campaigns involve identifying clients who contribute to diversifying the bank's funding sources and are more likely to subscribe.

Therefore, being able to classify clients into discernable clusters, predict the likelihood of their subscribing to the deposits, as well as delineating any potential anomalies are all essential tools for efficient campaigning. This report aims to tackle these objectives, providing an in-depth analysis of relevant data on one such campaign.

2. Data Description

The Bank Marketing Data[1] represents a collection of features relevant to the direct marketing campaigns of a Portuguese banking institution. These campaigns involved phone calls made to clients in order to determine whether or not they would subscribe to a bank term deposit. The same client was occasionally contacted more than once. Below is a brief descriptive summary of the data.

Number of Observations: 11,162 (No missing values)

Number of Features: 20 + Label

	Feature Name	Type	Description	Units
	age	Integer	The client's age.	Years
	job	Categorical	The client's job.	-
	marital	Categorical	The client's marital status.	-
	education	Categorical	The client's highest educational level.	-
	default	Binary	Whether or not the client has defaulted credit.	-
	housing	Binary	Whether or not the client has taken out a housing loan.	-
	loan	Binary	Whether or not the client has taken out a personal loan.	-
	contact	Categorical	The type of phone communication.	-
	day	Integer	The last day of the week	-

[1] Moro, S., Cortez, P., & Rita, P. (2014, June). A Data-Driven Approach to Predict the Success of Bank Telemarketing. *Decision Support Systems, Elsevier*, *62*(22-31). 10.1016/j.dss.2014.03.001

			the client has been contacted.	
0	month	Categorical	The last month of the year the client has been contacted	-
1	duration[2]	Integer	The last contact duration.	Seconds
2	campaign	Integer	The number of contacts made to a client for this current campaign.	Count
3	pdays	Integer	The number of days since the last contact with a client from a previous campaign.	Count
4	previous	Integer	The number of contacts made to a client in previous campaigns.	Count
5	poutcome	Categorical	The outcome of the previous campaign.	-
6	emp.var.rate	Continuous numeric	employment variation rate - quarterly indicator	
7	cons.price.idx	Continuous numeric	consumer price index - monthly indicator	
8	cons.conf.idx	Continuous numeric	consumer confidence index - monthly indicator	
9	euribor3m	Continuous numeric	euribor 3 month rate - daily indicator	
0	nr.employed	Continuous numeric	number of employees - quarterly indicator	
	y	Categorical	Has the client subscribed to a term deposit?	

Possible values each feature can take:

[2] This feature is highly indicative of the label since a contact duration of 0 indicates a label of ‘no’.

	Feature	Unique Values/Range of Values
1	age	[18 : 95]
2	job	‘admin’, ‘technician’, ‘services’, ‘management’, ‘retired’, ‘blue-collar’, ‘unemployed’, ‘entrepreneur’, ‘housemaid’, ‘unknown’, ‘self-employed’, ‘student’
3	marital	‘married’, ‘single’, ‘divorced’[3]
4	education	‘primary’, ‘secondary’, ‘tertiary’, ‘unknown’
5	default	‘yes’, ‘no’
6	housing	‘yes’, ‘no’
7	loan	‘yes’, ‘no’
8	contact	‘unknown’, ‘cellular’, ‘telephone’
9	day	[1 : 31][4]
10	month	‘jan’, ‘feb’, ‘mar’, ‘apr’, ‘may’, ‘jun’, ‘jul’, ‘aug’, ‘sep’, ‘oct’, ‘nov’, ‘dec’
11	duration	[2 : 3881]
12	campaign	[1 : 63]
13	pdays	[-1 : 854][5]
14	previous	[0 : 58]
15	poutcome	‘unknown’, ‘other’, ‘failure’, ‘success’
16	emp.var.rate	[-3.4 : 1.4]
7	cons.price.idx	[92.2 : 94.7]
8	cons.conf.idx	[-50.8 : -26.9]
	euribor3m	[0.63 : 5.05]

[3] The ‘divorced’ category also includes widowers.
[4] Encoded?
[5] A value of ‘-1’ indicates no previous contact has been made to this client.

9		
0	nr.employed	[4963 : 5229]
21	y	‘no’, ‘yes’

3. Data Preprocessing

There are no missing values in our data set however there are a lot of unknown values in a few variables. One of the ways to handle missing values is to delete the observations but that would lead to reduction of data set so instead we will be imputing as many unknown variables as possible. The variables with unknown values are: *“Job”, “Education”, “Housing”, “Loan”, “Pdays”, “Poutcome”, “Marital”,* and *“Default”*. While imputing the values for job and education, we were cognizant of the fact that the correlations should make real world sense. If it didn't make real world sense, we didn't replace the missing values.

I. Data Cleaning:

a) **Contact:**

The decision to remove the variable "contact" from the dataset, which contained observations related to communication channels such as "cellular" or "telephone," was due to the variable being considered non-informative and not contributing significantly to the predictive power of the model. This was done to remove any unnecessary complexity and enhance computational efficiency.

b) **Job with Education & Age**:

Figures 3.1.1 and 3.1.2 show the distribution of the variables “Job” and “Education”.

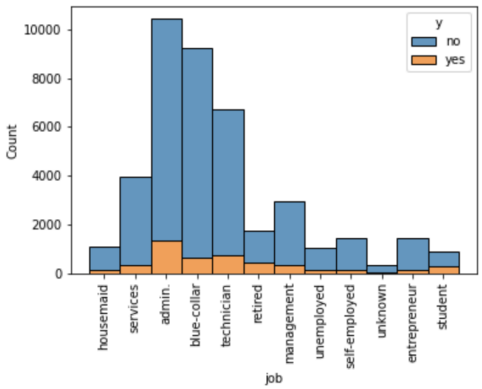


Figure 3.1.1

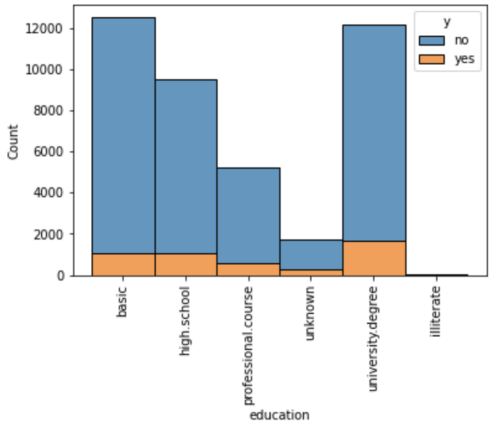


Figure 3.1.2

Figure 3.1.3 shows the percentage proportions (cross tab) between “Job” and “Education”.

education job	basic	high.school	illiterate	professional.course	university.degree	unknown
admin.	6.975628	31.942046	0.009595	3.483017	55.200537	2.389177
blue-collar	79.608818	9.487789	0.086449	4.895180	1.015777	4.905987
entrepreneur	28.708791	16.071429	0.137363	9.271978	41.895604	3.914835
housemaid	60.849057	16.415094	0.094340	5.566038	13.113208	3.962264
management	12.004104	10.191518	0.000000	3.043776	70.554036	4.206566
retired	47.500000	16.046512	0.174419	14.011628	16.569767	5.697674
self-employed	23.786066	8.304011	0.211119	11.822660	53.835327	2.040816
services	18.795666	67.573696	0.000000	5.492567	4.358781	3.779289
student	15.771429	40.800000	0.000000	4.914286	19.428571	19.085714
technician	7.845173	12.946760	0.000000	49.236245	26.827821	3.144001
unemployed	32.741617	25.542406	0.000000	14.003945	25.838264	1.873767
unknown	31.818182	11.212121	0.000000	3.636364	13.636364	39.696970

Figure 3.1.3

Inferring education from job:

- 70.6% of the people in the admin job have a university degree so any unknown values in education that have an admin job can be replaced with university degree.
- 79.6% of the people that work in blue-collar jobs have a basic degree (we combined the values 'basic.6y','basic.4y', 'basic.9y' into ‘basic’) so any unknown values in education that have a blue-collar job can be replaced with basic.
- 60.8% of the people that work as housemaids have a basic degree so any unknown values in education that have a housemaid job can be replaced with basic.
- 67.6% of the people that work in services have a highschool degree so any unknown values in education that have a services job can be replaced with highschool.

Inferring job from education:

- 49.2% of the people with a professional course certificate work as technicians so any unknown values in job that have a professional course degree can be replaced with technician.

Figure 3.1.4 shows the proportion of people above 60 that work in each job.

```
retired          74.505495
housemaid         5.934066
admin.            5.164835
technician        3.736264
management        3.296703
unknown           2.307692
blue-collar       2.197802
self-employed     0.989011
entrepreneur      0.879121
unemployed        0.769231
services          0.219780
```

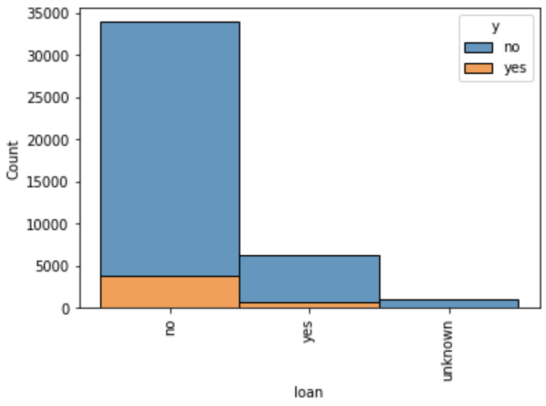
Figure 3.1.4

Inferring job from age:

- 74.5% of the people above 60 are retired so this allows us to replace any unknown job values for people above 60 with retired.

c) Job & Loan:

Figures 3.1.5 and 3.1.6 show the distribution of the variable "Loan" and the proportion of people (cross tab) in each job with the variable "Loan"



loan	no	unknown	yes
job			
admin.	81.414316	2.178085	16.407599
blue-collar	82.629610	2.576162	14.794228
entrepreneur	83.379121	2.472527	14.148352
housemaid	82.735849	2.735849	14.528302
management	82.558140	2.428181	15.013680
retired	83.400345	2.527283	14.072372
self-employed	83.462350	2.814919	13.722730
services	82.312925	2.544722	15.142353
student	81.142857	2.628571	16.228571
technician	83.148230	2.176810	14.674959
unemployed	82.642998	2.662722	14.694280
unknown	81.818182	2.020202	16.161616

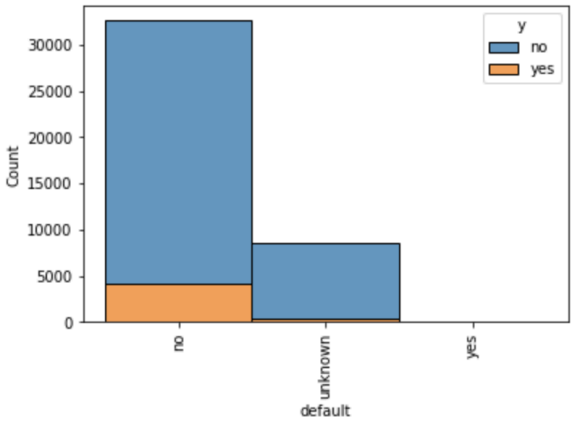
Figure 3.1.5 *Figure 3.1.6*

Inferring loan from job:

- We can see that in every job more than 80% of the people dont have loans so we can replace all unknown loan variables with ‘no’.

d) Job & Default:

Figures 3.1.7 and 3.1.8 show the distribution of the variable “Default” and the proportion of people (cross tab) in each job with the variable “Default”



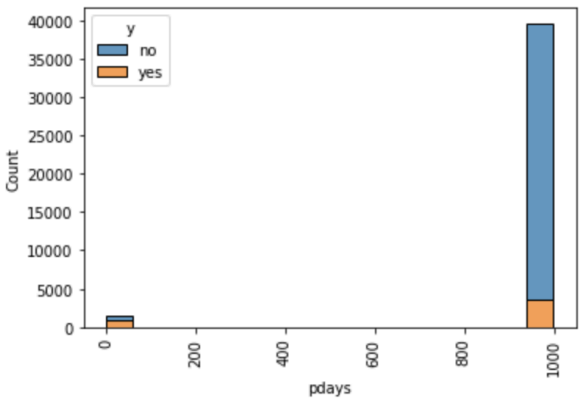
default	no	unknown	yes
job			
admin.	87.593552	12.406448	0.000000
blue-collar	65.718867	34.281133	0.000000
entrepreneur	79.326923	20.673077	0.000000
housemaid	69.905660	30.094340	0.000000
management	84.336525	15.663475	0.000000
retired	77.197013	22.802987	0.000000
self-employed	80.154821	19.845179	0.000000
services	76.921139	23.078861	0.000000
student	88.114286	11.885714	0.000000
technician	85.458315	14.512069	0.029616
unemployed	76.528600	23.372781	0.098619
unknown	47.979798	52.020202	0.000000

Figure 3.1.7 *Figure 3.1.8*

This shows that the whole variable is redundant as most of the observations are ‘no’ or ‘unknown’ values, only 3 observations in the whole data set have a value of ‘yes’ so we will be dropping this column as it adds no additional information.

e) Pdays:

Figures 3.1.9 and 3.1.10 show the distribution of the variable “Pdays” and the proportions in each value.



999	96.321744
3	1.065844
6	1.000291
4	0.286491
9	0.155385
2	0.148101
7	0.145673
12	0.140818
10	0.126250
5	0.111683
13	0.087404
11	0.067981
1	0.063125

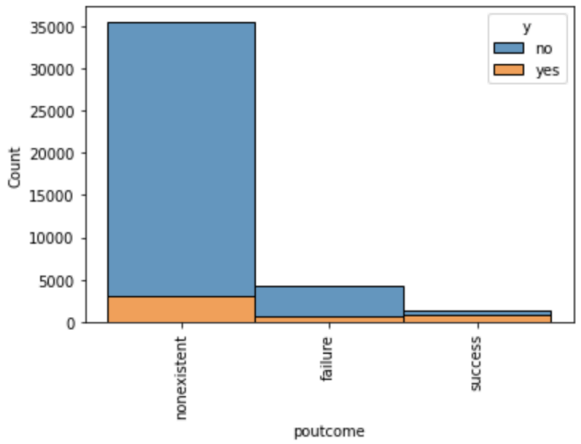
Figure 3.1.9

Figure 3.1.10

This variable tells you when the person was last contacted in terms of the number of days that have passed however 96.3% of the observations are '999' values which means the person has not been previously contacted. The rest of the values range from 1-27 days which also adds no additional information so I transformed the variable to tell us whether the person was previously contacted or not, the '999' values turned to 'no' values and numbers 1-27 turned to 'yes' values.

f) Poutcome:

Figures 3.1.11 and 3.1.12 show the distribution of the variable "Pdays" and the proportions in each value.



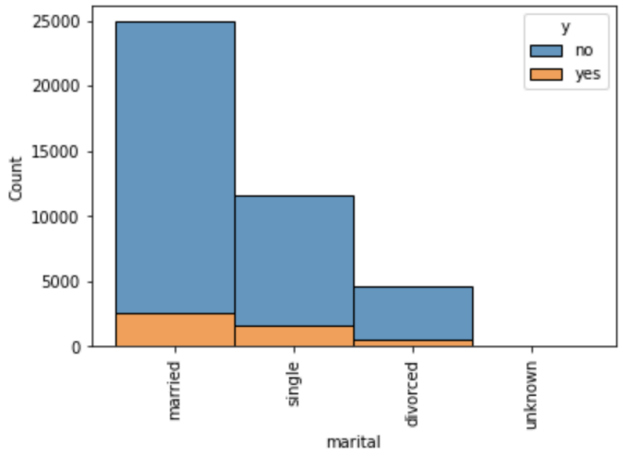

```
nonexistent    86.343110
failure        10.323395
success         3.333495
```

Figure 3.1.11 *Figure 3.1.12*

86.3% of the observations are 'nonexistent' values which is the same as 'unknown' so the entire variable is considered uninformative so we delete it.

g) Marital:

Figures 3.1.13 and 3.1.14 show the distribution of the variable "Marital" and the proportions in each value.




```
married      60.522482
single       28.085850
divorced     11.197436
unknown       0.194231
```

Figure 3.1.13 *Figure 3.1.14*

Only 0.19% of the values are 'unknown' which could be due to people not wanting to disclose that information, so we just replaced it with the value 'other'

II. Statistical Distributions:

a) Duration:

Figures 3.2.1 and 3.2.2 show the distribution of the variable “duration” before and after fitting it to the normal and exponential distributions.

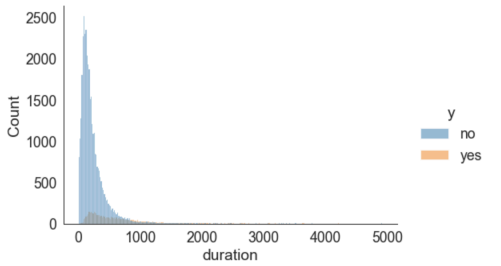


Figure 3.2.1

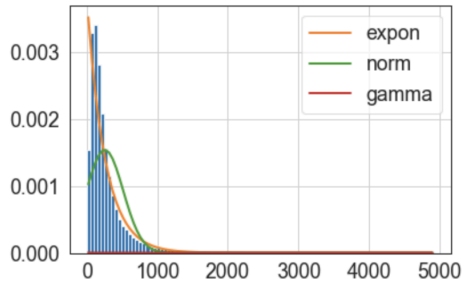


Figure 3.2.2

Upon fitting it to a few common distributions, it was determined that it best fits the exponential distribution, with a sum square error of 0.000006 (basically zero). It was also determined that it had a location parameter (mean) of 0 and a scale parameter (standard deviation) of 258.3

b) Age:

Figures 3.2.3 and 3.2.4 show the distribution of the variable “age” before and after fitting it to the normal and exponential distributions.

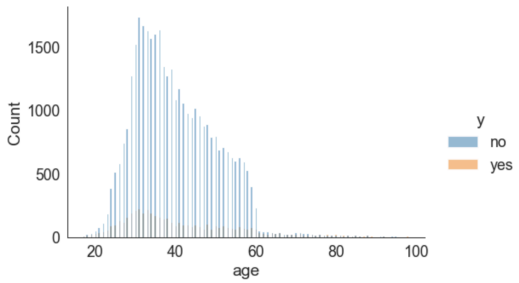


Figure 3.2.3

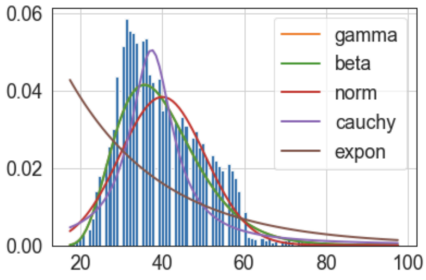


Figure 3.2.4

Upon fitting it to a few common distributions, it was determined that it best fits the gamma distribution, with a sum square error of 0.0096 (basically zero). It was also determined that it had a location parameter (mean) of 15.3 and a scale parameter (standard deviation) of 4.4 and an 'a' value of 5.61.

c) Campaign:

Figures 3.2.5 and 3.2.6 show the distribution of the variable "campaign" before and after fitting it to the normal and exponential distributions.

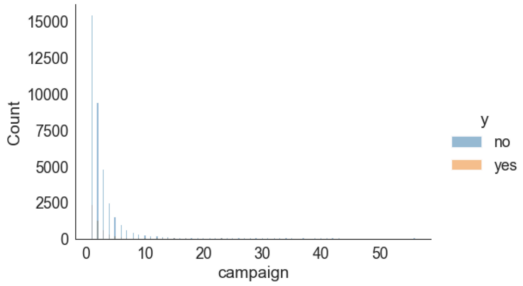


Figure 3.2.5

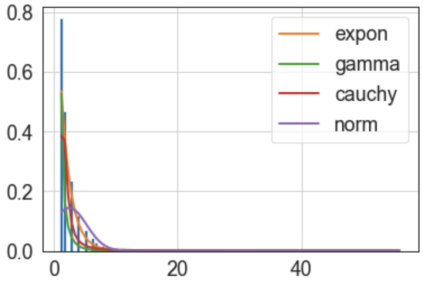


Figure 3.2.6

Upon fitting it to a few common distributions, it was determined that it best fits the exponential distribution, with a sum square error of 0.165 (basically zero). It was also determined that it had a location parameter (mean) of 1 and a scale parameter (standard deviation) of 1.568

d) Previous:

Figures 3.2.7 and 3.2.8 show the distribution of the variable "previous" before and after fitting it to the normal and exponential distributions.

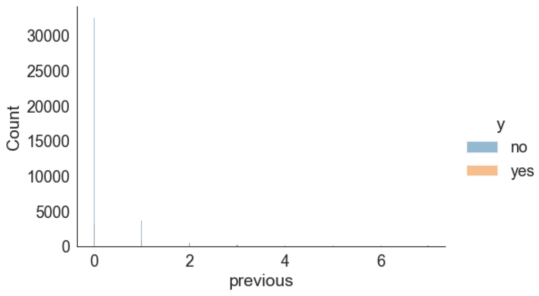


Figure 3.2.7

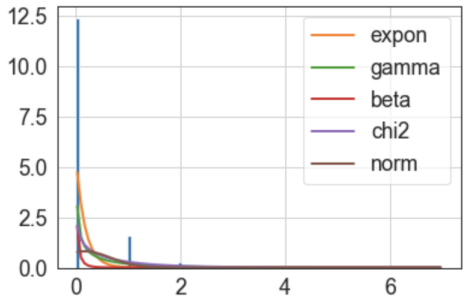


Figure 3.2.8

Upon fitting it to a few common distributions, it was determined that it best fits the exponential distribution, with a sum square error of 78.36. It was also determined that it had a location parameter (mean) of 0 and a scale parameter (standard deviation) of 0.173.

e) Emp.var.rate:

Figures 3.2.9 and 3.2.10 show the distribution of the variable "emp.var.rate" before and after fitting it to the normal and exponential distributions.

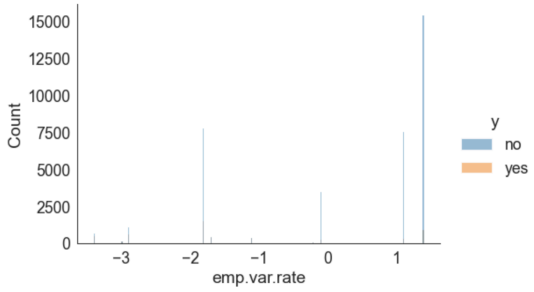


Figure 3.2.9

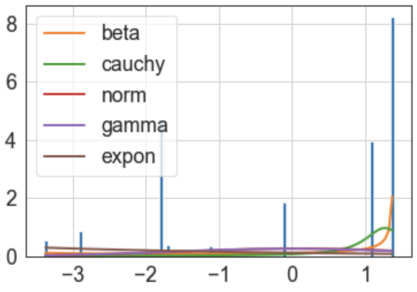


Figure 3.2.10

Upon fitting it to a few common distributions, it was determined that it best fits the beta distribution, with a sum square error of 78.7. It was also determined that it had a location parameter (mean) of -4.1 and a scale parameter (standard deviation) of 5.47 and an 'a' value of 0.451 and a 'b' value of 0.23.

f) Cons.price.idx:

Figures 3.2.11 and 3.2.12 show the distribution of the variable “cons.price.idx” before and after fitting it to the normal and exponential distributions.

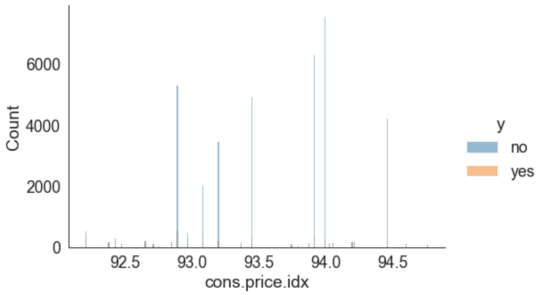


Figure 3.2.11

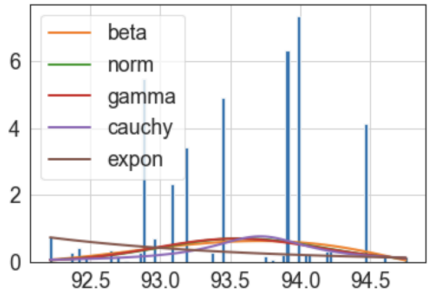


Figure 3.2.12

Upon fitting it to a few common distributions, it was determined that it best fits the beta distribution, with a sum square error of 165.9. It was also determined that it had a location parameter (mean) of 91.9 and a scale parameter (standard deviation) of 2.9 and an ‘a’ value of 3 and a ‘b’ value of 2.2.

g) Cons.conf.idx:

Figures 3.2.13 and 3.2.14 show the distribution of the variable “cons.conf.idx” before and after fitting it to the normal and exponential distributions.

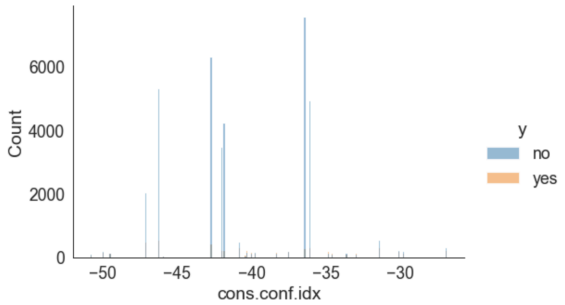


Figure 3.2.13

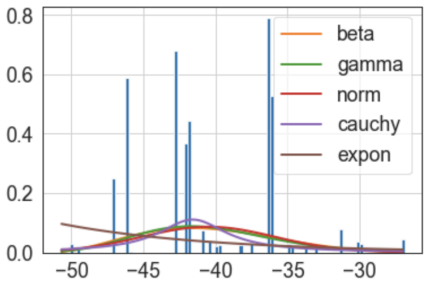


Figure 3.2.14

Upon fitting it to a few common distributions, it was determined that it best fits the beta distribution, with a sum square error of 1.87. It was also determined that it had a location parameter (mean) of -52 and a scale parameter (standard deviation) of 32.9 and an ‘a’ value of 3.7 and a ‘b’ value of 6.8.

h) Euribor3m:

Figures 3.2.15 and 3.2.16 show the distribution of the variable “euribor3m” before and after fitting it to the normal and exponential distributions.

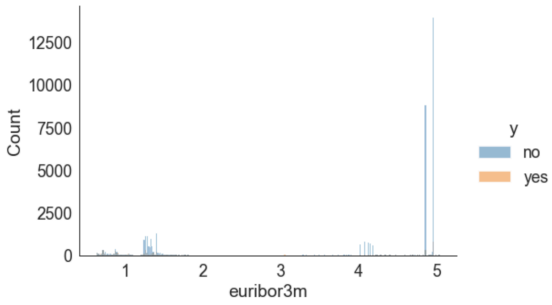


Figure 3.2.15

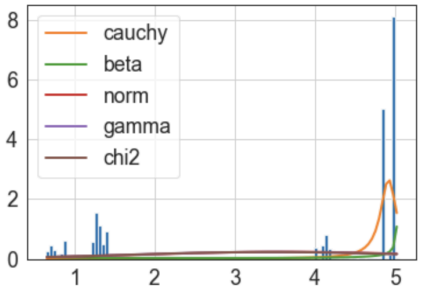


Figure 3.2.16

Upon fitting it to a few common distributions, it was determined that it best fits the cauchy distribution, with a sum square error of 70.2. It was also determined that it had a location parameter (mean) of 4.92 and a scale parameter (standard deviation) of 0.12

i) nr.employed:

Figures 3.2.17 and 3.2.18 show the distribution of the variable “nr.employed” before and after fitting it to the normal and exponential distributions.

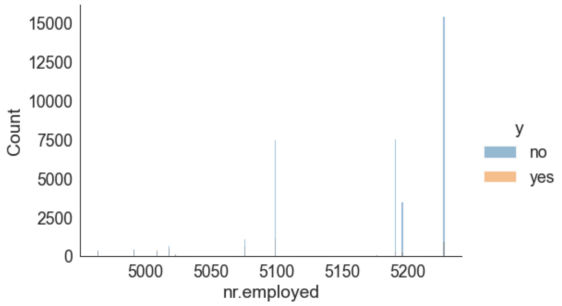


Figure 3.2.17

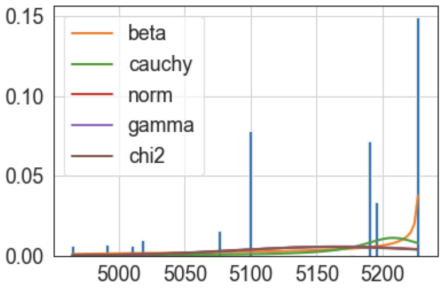


Figure 3.2.18

Upon fitting it to a few common distributions, it was determined that it best fits the beta distribution, with a sum square error of 0.025. It was also determined that it had a location parameter (mean) of 4953.3 and a scale parameter (standard deviation) of 274.8 and an 'a' value of 1.24 and a 'b' value of 0.42.

III. Exploratory Data Analysis (EDA):

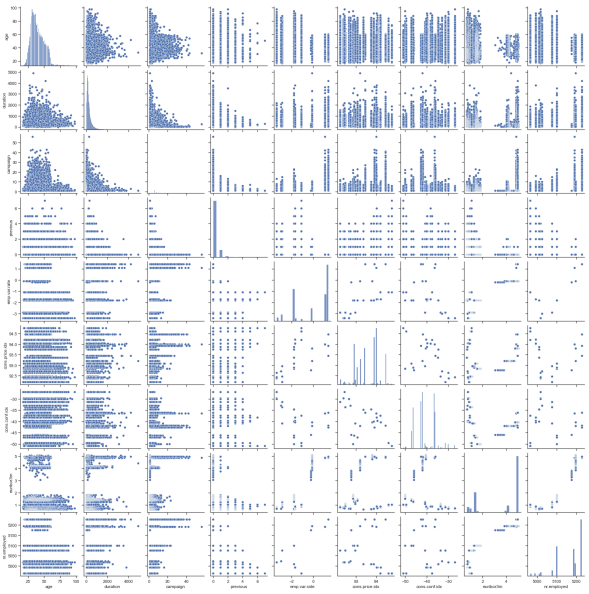


Figure 3.3.1

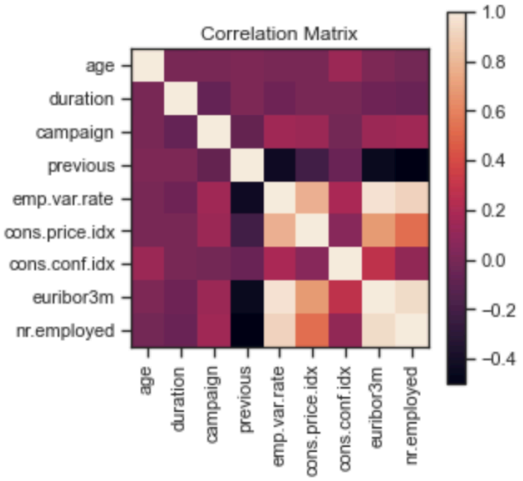


Figure 3.3.2

Figures 3.3.1 and 3.3.2 are paired plots illustrating the correlation between all quantitative variables, with the diagonal lines indicating the distribution of those variables.

Likewise, the graph on the right also displays the relationship between the quantitative variables. As we can see only a few variables exhibit a strong positive linear relationship, while several others demonstrate a weaker but still positive linear relationship or a weak negative linear relationship. The remaining data tend to be scattered either in one region or across the entire plot, suggesting that there's no clear linear relationship.

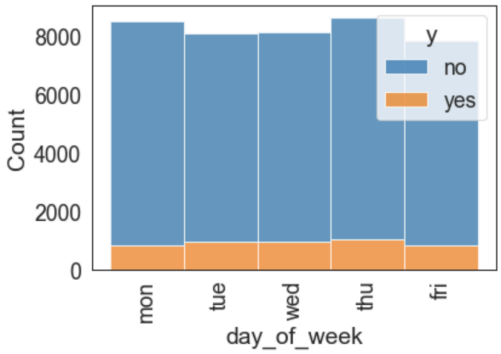


Figure 3.3.3

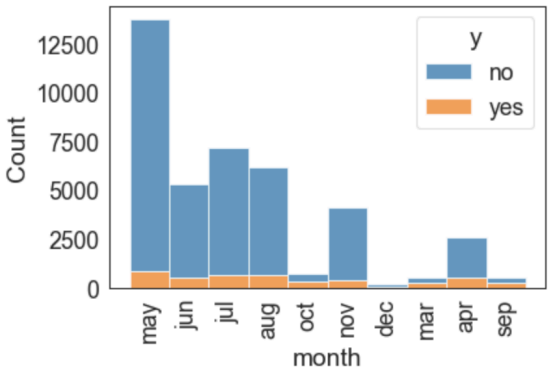


Figure 3.3.4

Figures 3.3.3 ad 3.3.4 show the distributions of the variables “day_of_week” and “month”, as we can see the proportions in day of week are almost identical so we will be dropping the variable as it adds no additional information.

IV. Encoding & Scaling:

Each column was encoded based on the data type corresponding to its values. All binary categorical variables had the ‘yes’ and ‘no’ values changed to 1 and 0 respectively. Categorical variables with more than two categories were either one-hot encoded or label encoded, depending on whether or not they were ordinal (in which case, label encoding is more suitable).

Accordingly, one-hot encoding was implemented for ‘job’, ‘marital’, ‘education’, ‘housing’, and ‘month’. To avoid any multicollinearity issues resulting from this type of

encoding, an issue known as the 'Dummy Variable Trap', the features were one-hot encoded so that the lowest-level category was dropped. Essentially, this splits the data into *n-1* features, instead of *n*, thus preserving the linear independence assumption.

Standard scaling was used to scale and center the numerical features in the dataset in order to give each feature equal weights.

We will then split the dataset using stratified splitting into X_train, X_test, y_train, and y_test without target variable being "y" which is whether the client subscribed to a term deposit or not.

V. Synthetic Minority Over-sampling Technique (SMOTE):

In addressing the imbalance within the target class, we employed the Synthetic Minority Over-sampling Technique (SMOTE) using 1 nearest neighbor. The original dataset exhibited an imbalance with the ratio of 8:1. When using SMOTE, it artificially increases the minority class instances by creating synthetic samples that closely resemble the existing data. This decreases the risk of the model being biased towards the majority class. Figures 3.5.1 and 3.5.2 show the proportions in our target variable before and after SMOTE.

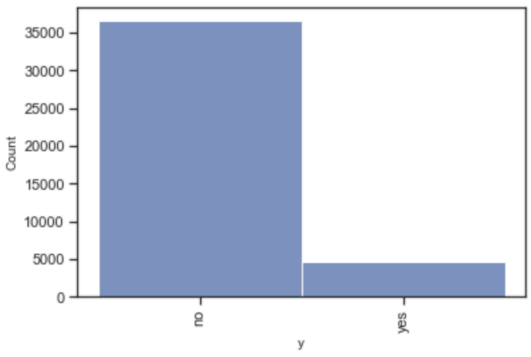


Figure 3.5.1

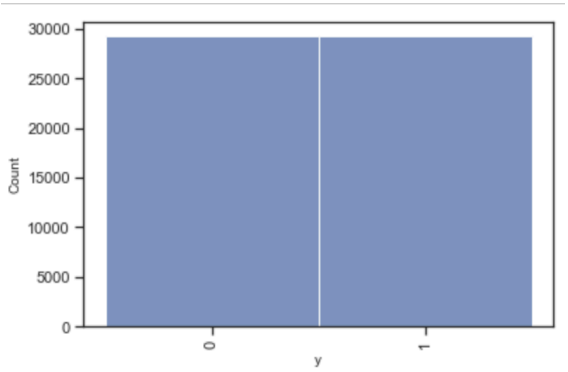


Figure 3.5.2

VI. Principal Component Analysis:

We applied PCA to our dataset to reduce its dimensions while still keeping 99% of the variability, the data set originally had 46 columns then was reduced to 31 (principal components) after PCA. Now, the dataset is a more compact yet informative representation of the original data. This enhances computational efficiency and ensures that our models are used on a dataset that captures the essential patterns and variability.

VII. Anomaly Detection:

The analysis aimed to detect anomalies in the Bank Market dataset using various approaches, including visual, distance-based, density-based, and clustering-based methods.

Visual Approaches for Anomaly Detection Using Box Plots

Box plots were employed to visually inspect the distribution of numerical features for both classes (yes and no) of the target variable. This graphical method allows for the identification of outliers and provides insights into the data's spread, center, and symmetry. As shown in Figure 3.7.1, the box plots provided a clear visual representation of the distribution of numerical features for different classes of the target variable. Notably, features like age, balance, duration, campaign, and previous were identified as having outliers, as evident from the extended whiskers in the box plots.

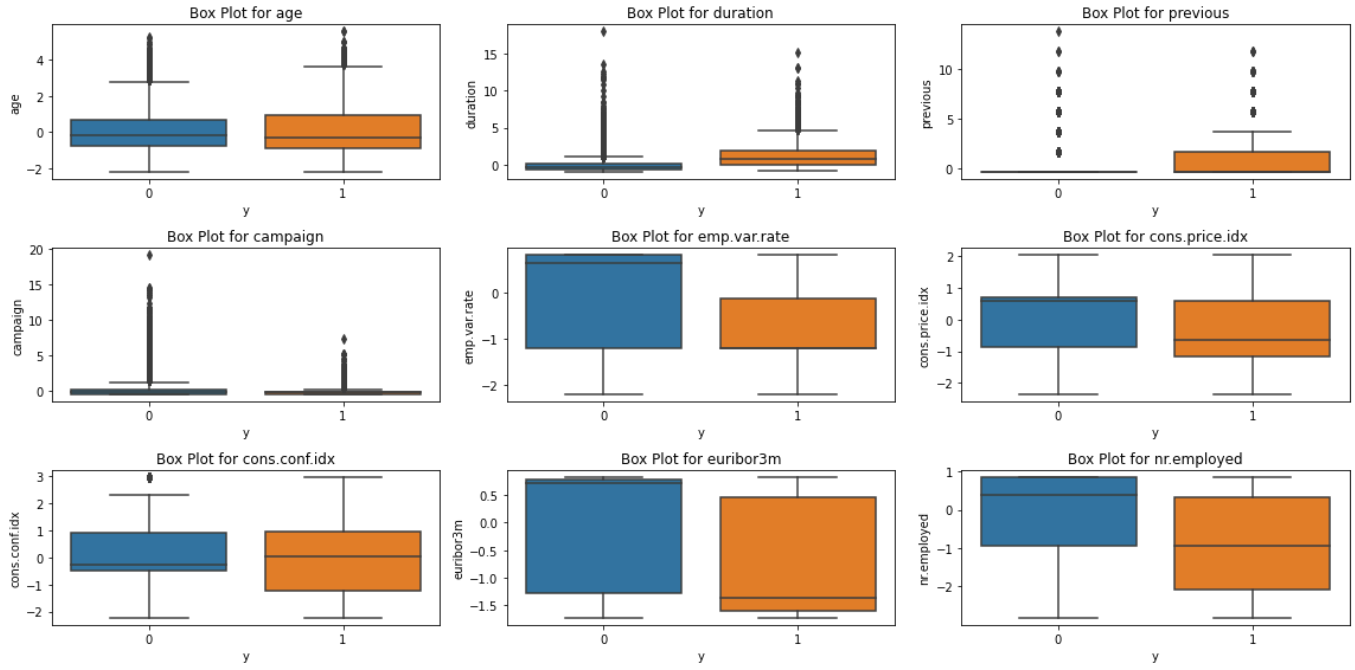


Figure 3.7.1: Box Plot for each variable for both classes (yes and no) of the target variable

Box Plot Outlier Detection by calculating IQR:

Following the visual inspection, a quantitative approach was implemented using box plots. Outliers were identified by calculating the Interquartile Range (IQR) and defining lower and upper bounds. The IQR was calculated for each attribute, and observations less than 3xIQR from the value at the 25% quartile and larger than 3xIQR from the value at the 75% quartile were considered outliers. This method revealed the presence of outliers in several numerical features.

Numerical Results:

Number of outliers: 10828

The average values of numerical variables for different Y values:

:

y	age	campaign	duration	previous
no	39.911185	2.633085	220.844807	0.132374
yes	40.913147	2.051724	553.191164	0.492672

Figure 3.7.2

```
df[['age', 'duration', 'previous', 'campaign']].describe()
```

	age	duration	previous	campaign
count	41188.00000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	0.172963	2.567593
std	10.42125	259.279249	0.494901	2.770014
min	17.00000	0.000000	0.000000	1.000000
25%	32.00000	102.000000	0.000000	1.000000
50%	38.00000	180.000000	0.000000	2.000000
75%	47.00000	319.000000	0.000000	3.000000
max	98.00000	4918.000000	7.000000	56.000000

Figure 3.7.3: Descriptive Statistics of the Variables Table

As shown in Figures 3.7.2 and 3.7.3, the indices of outliers are identified as outliers based on the Box Plot approach because these attributes deviate from the mean, either below or above the average value.

Distance-based Approach: KNN Outlier Detection

The Local Outlier Factor (LOF) algorithm, a distance-based approach, was utilized to assign outlier scores based on the distances to the kth nearest neighbour. Outliers were identified

by negative outlier scores, and their indices were recorded using different values of k. This approach focused on the local density of points to detect anomalies.

Number of Distance-based outliers: 4119

Density-based Approach: DBSCAN

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm was employed to identify outliers based on the density of data points. Objects in regions of low density were considered outliers. The results provided insights into areas with sparse data.

Number of Density-based outliers: 2670

In conclusion, the analysis employed a diverse range of anomaly detection methods to gain insights into the Bank Market dataset. Visual approaches, such as box plots, provided a clear representation of outlier distributions, complemented by numerical methods utilizing the interquartile range for quantitative anomaly identification. Distance-based techniques, like the KNN algorithm, incorporated local density information, while density-based approaches highlighted outliers in sparsely populated regions. The combination of visual, numerical, and distance-based methods proved effective for identifying outliers in features like age, balance, duration, campaign, and previous in the Bank Market dataset.

4. Association Analysis

In the “Association Analysis” notebook, we aimed to extract all possible association rules in this dataset. However, by first looking at the dataset, many numerical variables in the dataset are discrete and continuous. This would prove to be difficult when trying to extract the association rules due to computational complexity, and some values were very scarce in the

dataset so this would harden the search for meaningful rules. We deduced that the best way to solve this problem was to start binning the variables with continuous data. The number of bins and the type of binning, whether equal frequency or equal intervals, were decided on each variable separately depending on the distribution of the data. To bin the data using equal frequency data, we used quantile binning. Meaning if we set the bin number to 4 bins, then the first 25% of the data would be in the first bin, the second 25% in the second bin, etc. To decide the number of bins, we went through a trial and error system until we felt like the data was best represented and most equally distributed. This was also done for each variable individually. After binning the data, we dropped the original column as we created a new column that has the intervals, so the original column became redundant. Binning all the data made all of the variables categorical because of the variables that were already categorical, and the numerical variables that were binned became intervals, hence also categorical variables. So, we proceeded to create dummy variables of the whole dataset to be able to start extracting the association rules. To extract the rules out of the dataset we first generated the frequent itemsets that were larger than 1 itemset and used the apriori principle. After generating the frequent itemsets, we then extracted the association rules using a minimum confidence threshold.

<u>The steps for each variable binning:</u>

a) Age:

Figure 4.1 shows the distribution of the age variable. It can be seen that the data is very skewed to the left as there are many outliers in the data. It can be concluded that if we were to bin the data using equal intervals, the first and last bins would have very little data compared to

the bins in the middle. Hence for the age variable, we used equal frequency binning. We decided that the best number of bins for age was going to be 4. Another reason why 4 bins were chosen was because by choosing the 4 bins, we felt this best categorized the different age groups. Figure 4.2 shows the age variable after binning the data into 4 bins and the intervals chosen to satisfy the equal frequency binning.

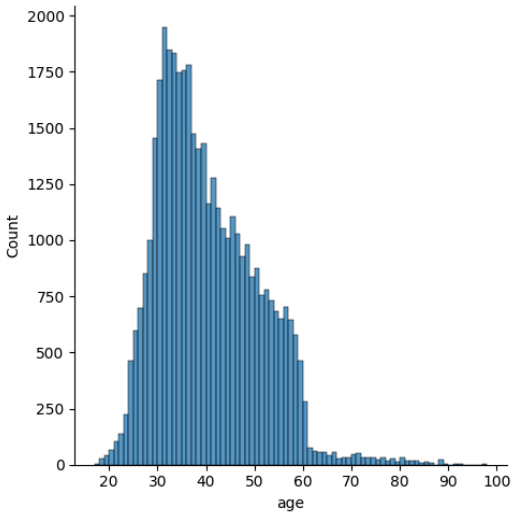


Figure 4.1

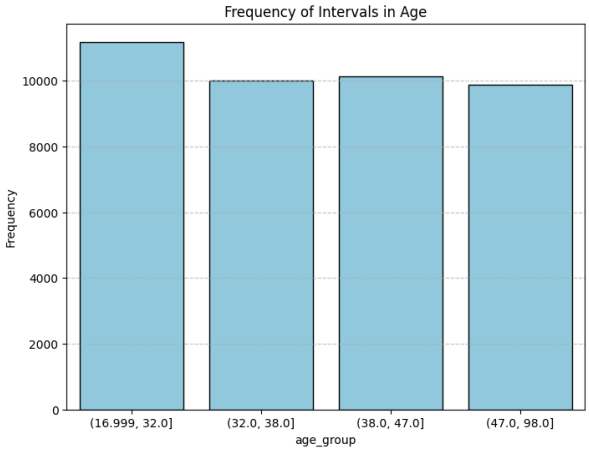


Figure 4.2

b) Duration:

Figure 4.3 shows the initial distribution of the duration variable. Again, it is incredibly skewed to the left with many outliers in the data. So, we decided to use equal frequency bins using quantile binning. We also used 4 bins because this was the optimal number for the equal distribution. Figure 4.4 shows the bins and intervals chosen for the duration variable.

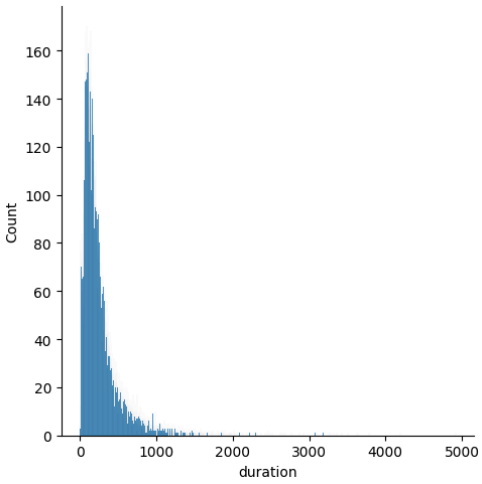


Figure 4.3

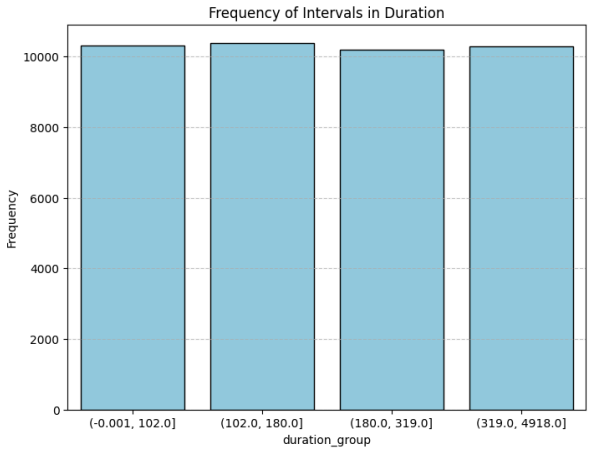


Figure 4.4

c) Campaign:

As can be seen by Figure 4.5, the variable campaign is decreasing exponentially. This made choosing the bins tough, but we concluded that we were going to join all the infrequent attributes into 1 bin, and the campaign values of (1) in another separate bin, this can be seen below in Figure 4.6.

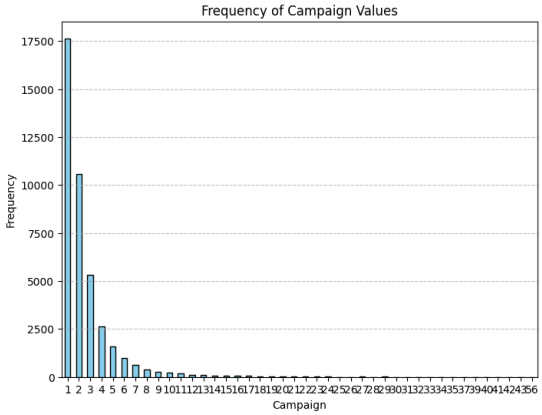


Figure 4.5

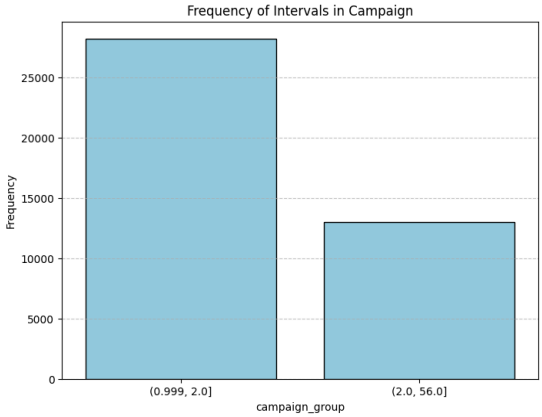


Figure 4.6

d) Emp.var.rate:

Figure 4.7 shows the distribution of emp.var.rate. It can be seen that it is a uniform distribution, discrete values. This variable may not be binned, but we decided to reduce the computational complexity of the data so we can reduce this from 10 to 2. In this variable, all the negative variables were very infrequent compared to the positive value and looking at the meaning of the emp.var.rate it made sense to combine all negative values into one 'fired' column and all the positive values into one hired column. The new column and its distribution can be seen in Figure 4.8.

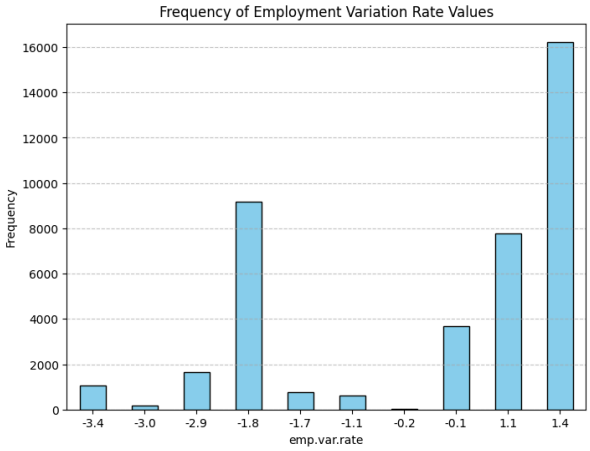


Figure 4.7

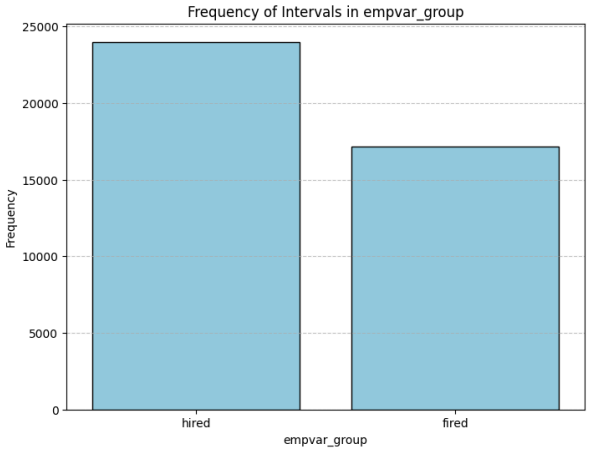


Figure 4.8

e) Cons.price.idx:

In Figure 4.9, it can be seen that the distribution of the data isn't equal, there are many infrequent data. Also, it's not very clear in the graph but there are many different categories in the data that are all between 92, 93, 94. The data is in the fourth decimal place. So instead of binning this variable, we decided to round down the data. So this in a way also creates 3 different bins, 92, 93, 94. It also concatenated a lot of the infrequent data together to make it represent more. No further binning was possible for this column.

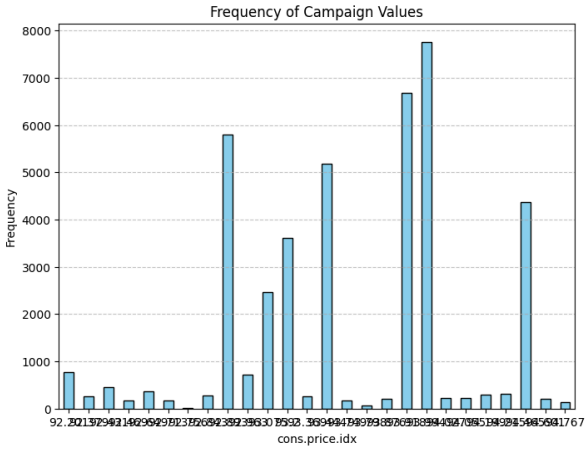


Figure 4.9

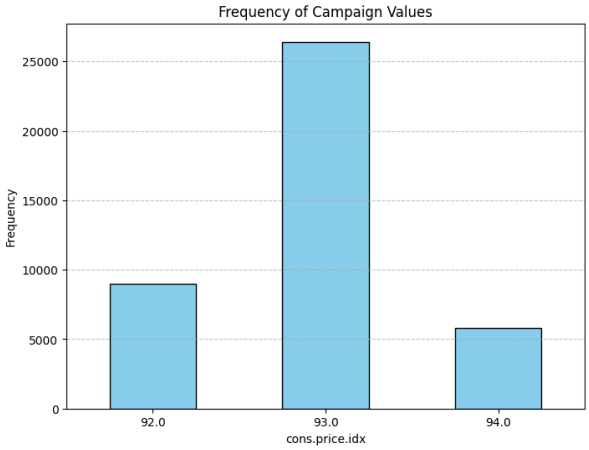


Figure 4.10

f) Cons.conf.idx:

For this variable, we binned using equal frequency bins (quantile binning) to be able to attempt to balance the huge imbalance in the data inside this attribute. We couldn't bin more bins than what's shown because the value for 34 is extremely infrequent in the dataset so having more bins would have to split this value in half, which isn't possible. Figure 4.11 shows the initial distribution, while Figure 4.12 shows after binning the attribute to 4 bins.

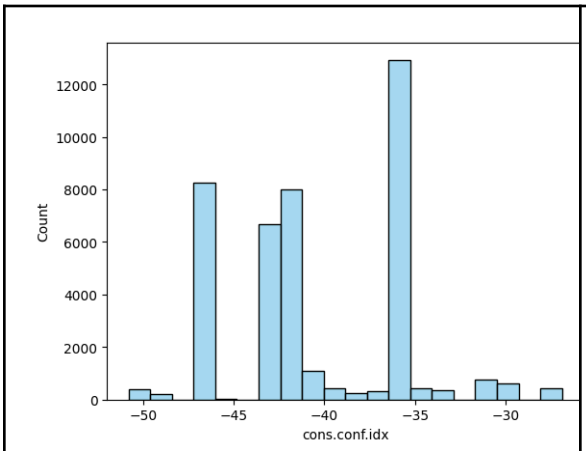


Figure 4.11

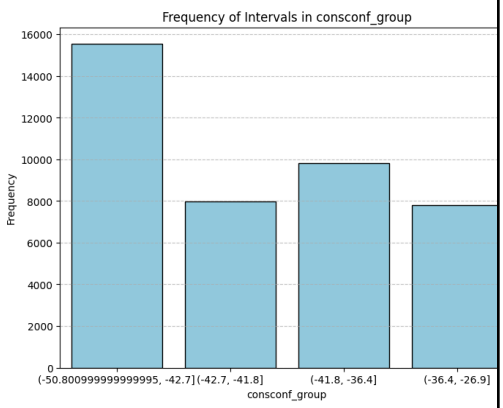


Figure 4.12

g) Euribor3m:

In Figure 4.13, it can be seen that there are many discrete categories in this attribute. Some are extremely frequent, some are the opposite. So we chose equal frequency (quantile) binning for this attribute to be able to represent the data equally.

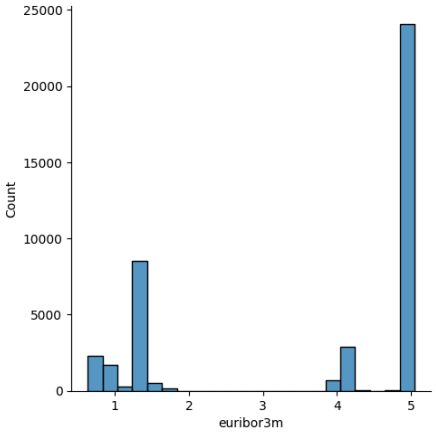


Figure 4.13

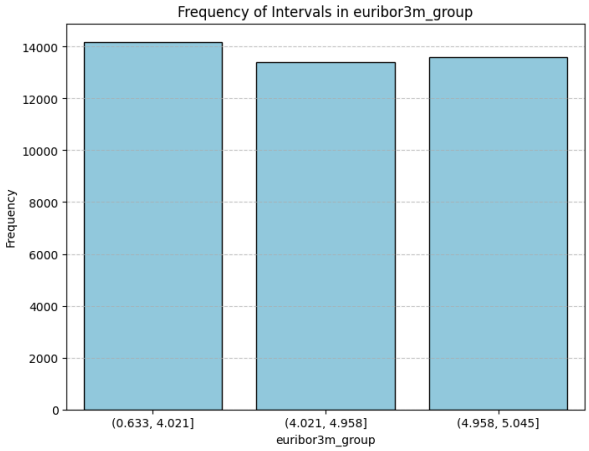


Figure 4.14

h) Nr.employed:

For this attribute, as can be seen from Figure 4.15, the only possible way to bin the data to represent the data equally was equal frequent binning in 2 bins. This transformation and the intervals can be seen clearly in Figure 4.16

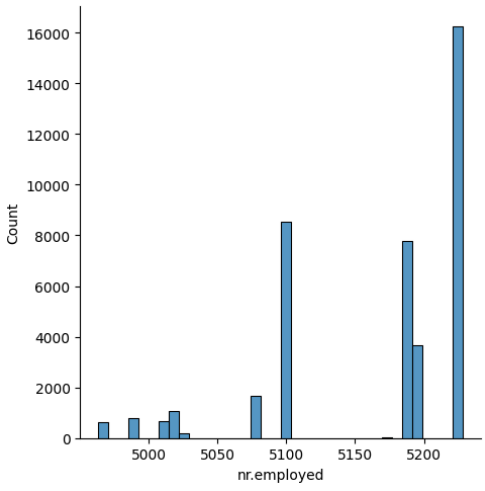


Figure 4.15

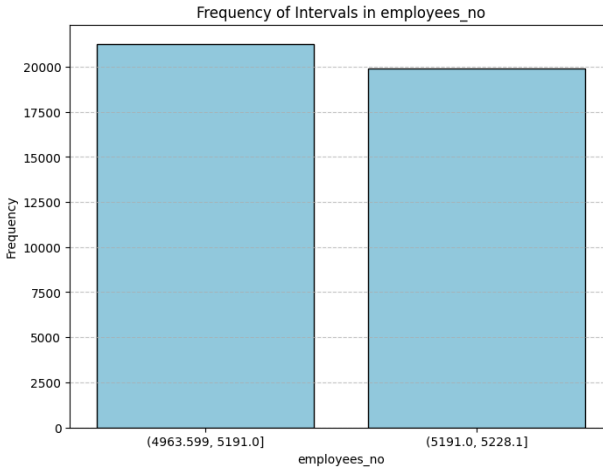


Figure 4.16

Extracting the association rules:

To start extracting the association rules in the dataset, we used the library mlxtend to first extract the frequent itemsets in the dataset. We added criteria of a minimum of 2 itemsets because any less it produced too many rules that weren't meaningful. We first created dummy variables to encode the whole dataset then used this new dataset to get the frequent itemsets. Using the apriori principle, and a minsup of 0.6 (obtained through trial and error clearly shown in the notebook), we were able to extract a total of 19 frequent itemsets. (58 if minsup=0.5, and 111 if minsup = 0.4). Then from these itemsets, we extracted the association rules using a minconf of 0.5 (obtained through trial and error clearly shown in the notebook), we were able to deduce a total of 6 association rules in the dataset. These can be further tuned if the confidence threshold is increased. However, most of the rules have a confidence higher than 76% so these are already high confidence. These rules can be seen in Figure 4.17.

	antecedents	consequents	support	confidence
0	(0_l, 0_pd)	(0_pr)	0.732422	0.896041
1	(no_y, 0_pd)	(0_l)	0.740895	0.847549
2	(no_y, 0_l)	(0_pr)	0.667525	0.887390
3	(no_y, 0_pd)	(0_pr)	0.787171	0.900486
4	(no_y, 0_pd)	(0_l, 0_pr)	0.667525	0.763616
5	(0_l, no_y, 0_pd)	(0_pr)	0.667525	0.900970

Figure 4.17

In summary what Figure 4.17 illustrates is the following:

1. (0 [loan], 0 [pdays]) → (0 [previous])
2. (no [y], 0 [pdays]) → (0 [loan])
3. (no [y], 0 [loan]) → (0 [previous])
4. (no [y], 0 [pdays]) → (0 [previous])
5. (no [y], 0 [pdays]) → (0 [loan], 0 [previous])

6. (0 [loan], no [y], 0 [pdays]) → (0 [previous])

Many of these rules can be further merged as taught in class to achieve a final association rule list of

1. (no [y], 0 [pdays]) → (0 [loan])
2. (no [y], 0 [pdays]) → (0 [loan], 0 [previous])
3. (0 [loan], no [y], 0 [pdays]) → (0 [previous])

This list can be very useful in our dataset because all the rules contain the label (y) so it can be easily used to fill in many unknowns (if the label is known, and other variables are NA) or used to predict if the label is unknown and the rule is present.

5. Prediction

In the "Prediction" Jupyter Notebook, we used a number of different methods to model the pre-processed dataset. We fitted the models twice, once to data that had been reduced (using PCA) and balanced (training set using SMOTE). Initially, these models were implemented using their default hyperparameters, which would likely need tuning to better suit our data. The classifiers implemented in this first stage were the following: Logistic Regression, Linear Discriminant Analysis, K-Nearest Neighbors, Decision Tree Classifier, Gaussian Naive Bayes, Support Vector Machines, Random Forest Decision Trees, and AdaBoost. Each classifier was fitted to our training data and cross-validated (using a 10-fold CV), producing an accuracy score that was used for a rudimentary comparison of the model performance. Table 6A shows the resulting sorted accuracy scores for each of the following models.

Classifier	Accuracy on PCA data & SMOTE	Accuracy on PCA data
Decision Tree	88.95%	92.71%

KNN	86.28%	92.55%
SVM	90.55%	90.55%
ADABoost	88.14%	88.13%
Random Forest	88.54%	88.40%
Logistic Regression	70.76%	86.43%
LDA	70.17%	84.12%
Gaussian	70.53%	76.71%

From the table above, we can see that the balanced data had slightly lower accuracy scores, but it is worth noting that the accuracy observed in the unbalanced data is not a meaningful measure since the majority class was overwhelmingly dominating the training process. It is worth noting that the performance scores retained their rankings in both groups, indicating that despite the figures being less meaningful and accurate in the second (unbalanced) group, they still offer insight as to which models are best suited for our data.

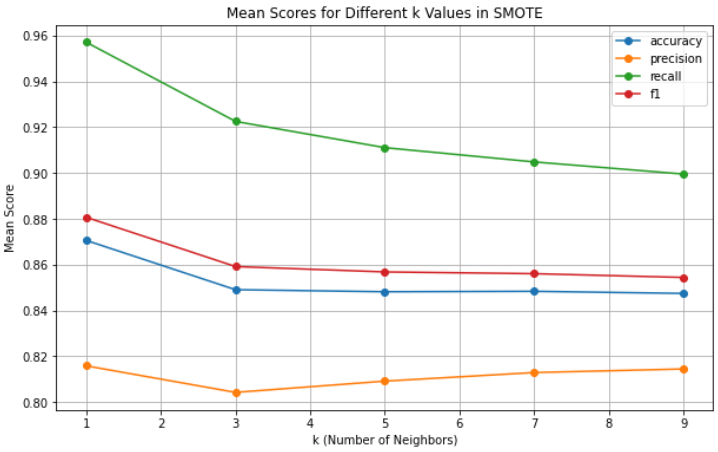
Refinement of Classifier Options:

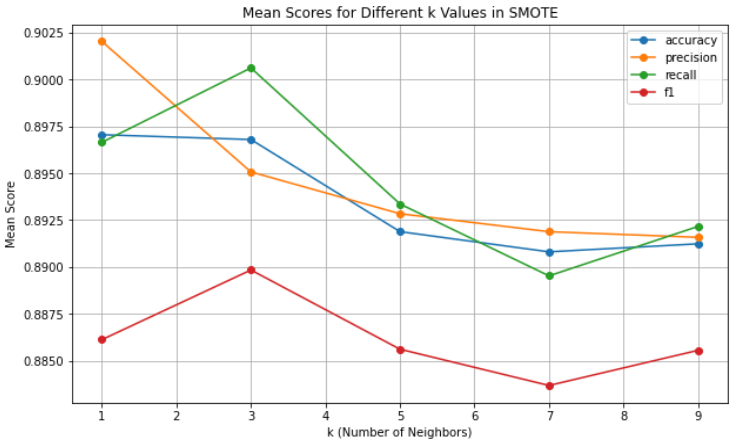
Based on the results of the previous table, the top two performing models (Decision Tree Classifier and KNN) were selected for hyperparameter tuning. At this stage, we were able to single out a feature, ‘duration’, that seemed to be indicative of the target variable. This feature refers to the amount of time, in seconds, spent on the phone with the bank clients during the current campaign. When this feature takes the value of ‘0’, the label would necessarily be ‘no’. This is because if the client was not contacted for the campaign, he or she would not have

deposited term deposits. Accordingly, it was dropped and the hyperparameter tuning was done on the resulting dataset.

Tuning SMOTE For Selected Classifier Models:

After selecting the models for the hyperparameter tuning, we decided to also ensure that the 'k' parameter involved in the SMOTE imbalance handling technique was optimized to each one of those models. This was done by cross-validating (5-fold CV) a range of possible k values and evaluating their contributions based on the resulting accuracy, precision, recall, and f1 scores. The results were plotted and can be seen in Figures 6.1 and 6.2 below.





Figures 6.1 and 6.2 display the average scores (four predetermined metrics: accuracy, precision, recall, and f1) for 9 different k-values (1 through 9). Our dataset prioritizes precision as an evaluation criterion since a false positive is more detrimental than a false negative. In other words, a model that constantly misclassifies clients as deposit subscribers would affect the bank's operations (budgets, investments, forecasts, etc.) more than a model that does the

opposite. Investing in a client predicted to deposit but does not actually do so is a waste of time, resources, and opportunity.

From the plots and their corresponding scores, we can see that both the KNN and the Decision Tree Classifiers operate best (from a precision perspective) on data that has had its labels balanced using SMOTE, k = 1. The fact that both gave the same k is beneficial in that it maintains uniformity by removing the need to balance the data twice according to two different parameters.

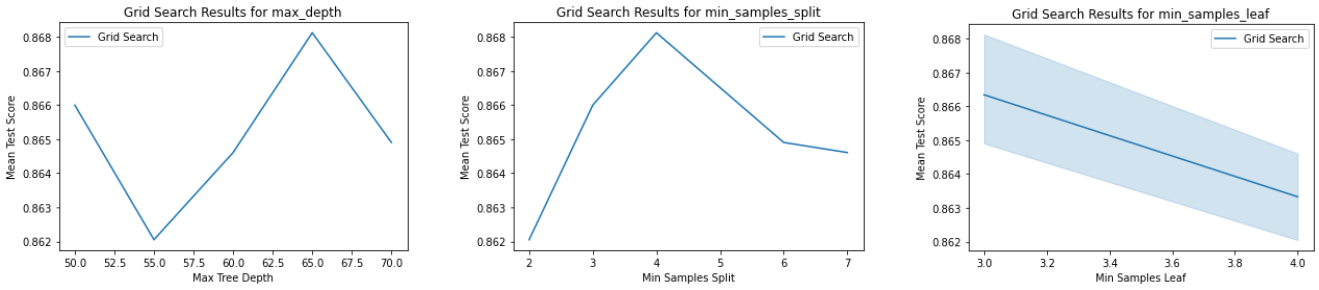
Decision Tree Hyper-Parameter Tuning:

For the decision tree model, we decided to focus on four main parameters for hypertuning: *max_depth*, *min_samples_split*, *min_samples_leaf,* and *criterion*. Instead of using GridSearchCV which proved itself too computationally expensive since it cross-validates every possible combination of hyperparameters, we made use of the RandomizedSearchCV function available in sklearn's model selection package which selects a random subset of hyperparameter options to evaluate. In addition, only a subset of the training set was used for the tuning process. This decision was made to reduce computation time during the tuning processes, but the model implementation itself was executed normally on the full data.

For the Decision Tree tuning, three runs of the RandomizedSearchCV were executed, each run iterating the 5-fold CV five times. There is no stopping criteria per se, as it is fundamentally a subjective process. Hence, time limitations and the fine-tuning of the parameter options were both key in determining when to accept the results as a decent estimation of the optimized parameters. These results are shown below.

max_depth	65
min_samples_split	4
min_samples_leaf	3
criterion	entropy

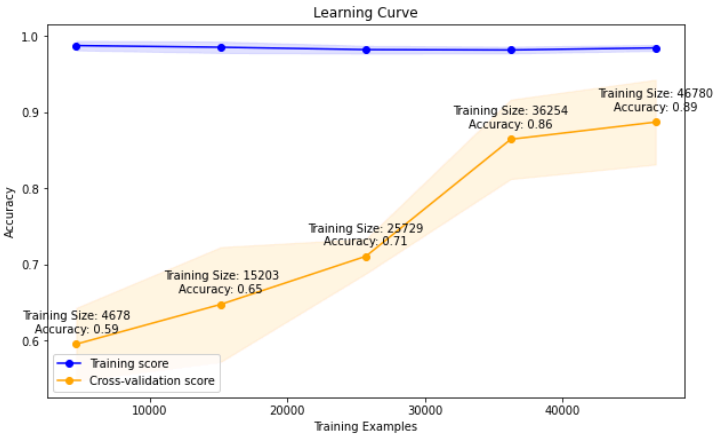
We also plotted these selected hyperparameters against the other values within the specified range so that we could further visualize the extent to which they were improving the model's performance. These figures are shown below, labeled as Figures 6.2, 6.3, and 6.4.



From the figures above, we can see a visual representation of the results presented in the optimal parameter table. While the uncertainty range shown in Figure 5.4 is made to seem unreasonably large due to the scaling, the results are actually showing minimal variability. The graph for the criterion parameter (not shown) reveals that both gini and entropy are almost identical in performance, with only a hundredth of a difference separating them. Hence, using either will not significantly impact the model.

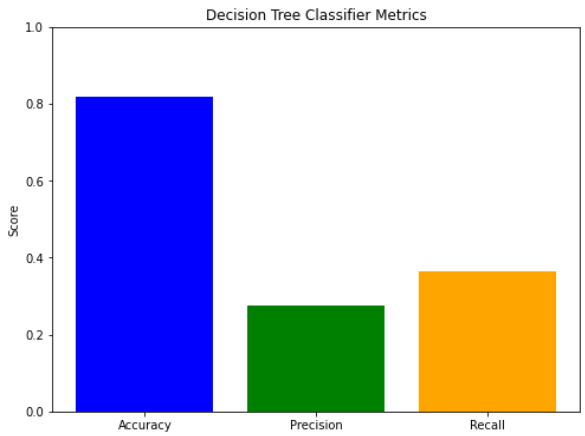
Decision-Tree Model Evaluation

The next step is evaluating the model's performance when set to the hyper-parameters reached in the previous section. First, we wanted to analyze the model's fit in terms of whether the model overfits the training data. This could be done by analyzing the model's corresponding Learning Curve, which also depicts the bias-variance tradeoff (validation set vs. training set) as well as the optimal training size of the data. The data we are currently using has 58,476 observations in the training set (after pre-processing and SMOTE). The Learning Curve will show the performance of the training set by cross-validating it with the validation (5-fold CV) for every possible training set size. Figure 5.5 shows the resulting plot, with the highlighted regions depicting the uncertainty or standard deviation.



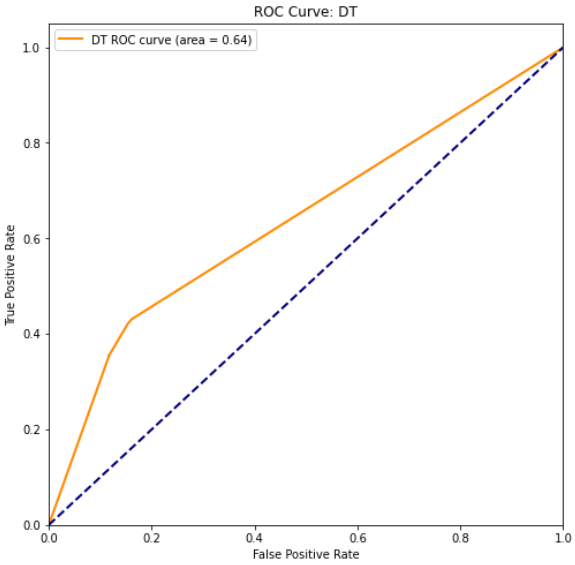
The graph indicates that when the training set size was 36,254, it had an accuracy of 86%, compared to an accuracy of 89% when it was 46,780 observations (almost 10,000 additional observations). We can conclude that this minor difference is not worth using the larger training set. Since the overall CV score (orange line) appears to converge, the model can be assumed to not be overfitting the training set (at least not to a considerable degree warranting major adjustments).

After having completed the training set evaluation, the next step is evaluating the model's performance on the testing set. Below is Figure 5.6 which shows the precision, recall, and accuracy of the optimized DT model on the full testing data.



Accuracy is much higher than precision and recall, indicating that the model makes more correct predictions overall, instead of correct positive predictions as a proportion of the predicted positives (precision) or actual positives (recall). In our case, we are more concerned with recall and, specifically, precision. The recall can be seen to be slightly higher than the precision of the model, but both are relatively low and unpromising. One possible explanation is that by using SMOTE to adjust such a drastic class imbalance, the minority class and its distribution were not accurately represented. Furthermore, since SMOTE had to add so many synthetic data points to balance the classes, it could potentially have created inaccurate patterns and relationships that affected the model's precision and recall ability.

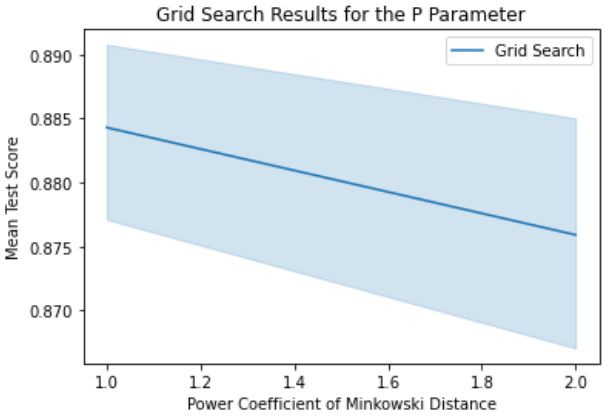
Finally, the decision tree model was also evaluated through the analysis of its ROC Curve (i.e., its AUC score). Figure 5.7 below shows that the model had an AUC score of 0.64, which indicates that the model's discriminatory power is moderate, only 14% better than the model's predictive power being completely random (at 0.5). Conventionally, an AUC between 0.6 and 0.7 is considered 'fair', suggesting that the model is not too reliable when making predictions but can still offer information (both room & potential for improvement).

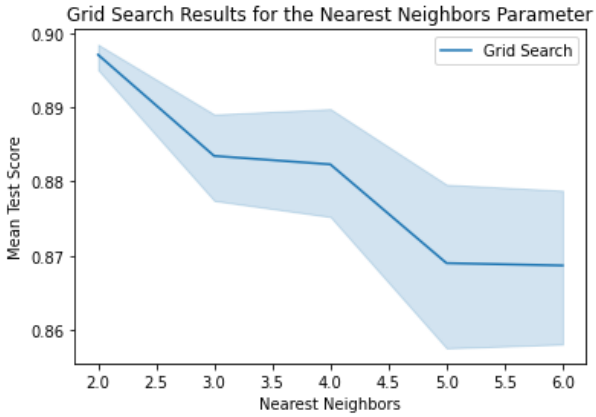


KNN Hyper-Parameter Tuning

The same concepts applied above in the DT hyper tuning were implemented for the KNN Classifier. However, GridSearchCV was used this time (on the same training data subsets created for the DT hyperparameter tuning) and it was validated 10-fold instead of 5. This was run twice, meaning we fine-tuned the original results once (for limited computational power & time constraints). The optimized parameters are shown in the table below, and their performance compared to other options is shown in Figures 6.8, 6.9, and 6.10. These parameters indicate that the number of nearest neighbors should be 2, the observations should be weighted depending on their distances, and the p-parameter should be set to 1. The p-parameter is the value of the p in the Minkowski Distance Coefficient. When p=1, the distance metric is the Manhattan distance.

n_neighbors	2
weights	distance
p	1





KNN Model Evaluation

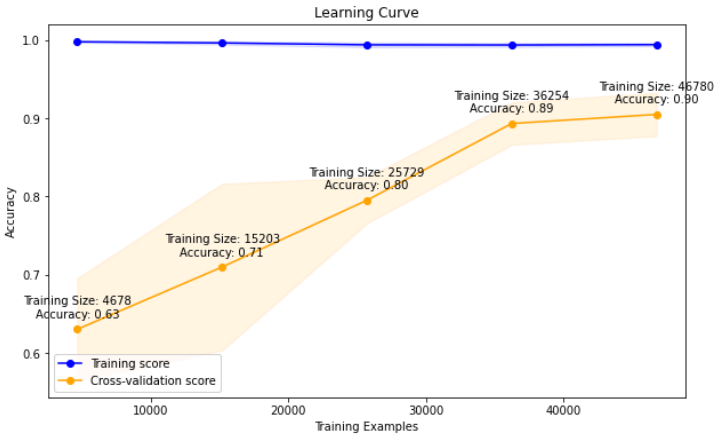
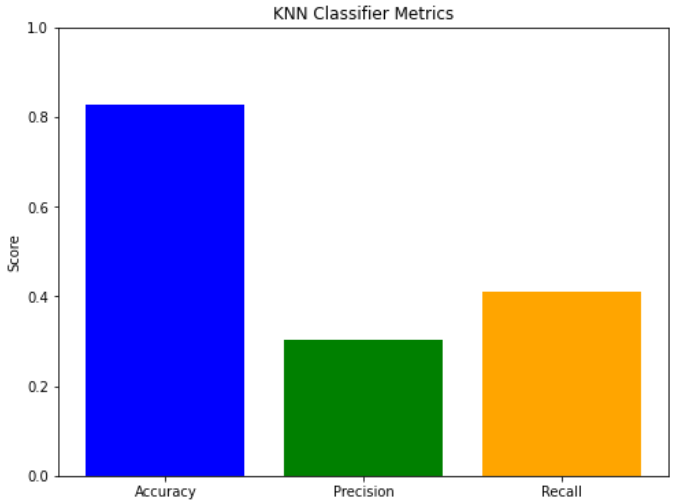
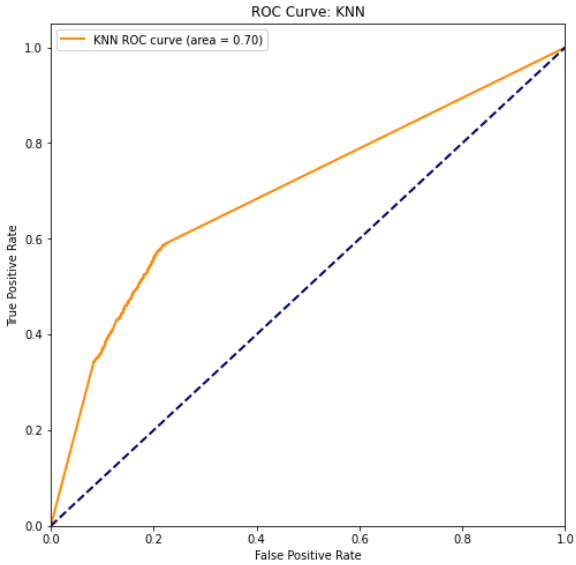


Figure 5.11 to the right shows the learning curve for the training set when using the KNN model with the selected parameters. The training score consistently being almost exactly 1 might suggest the possibility of overfitting, but since the CV score appears to converge so this might not necessarily be the case. If it were though, this could be attributed again to the decision of using SMOTE to balance the data, especially since it was done at k=1 (based on the optimal value found above).

The model was fitted to the test set and the following metrics were computed: precision, recall, and accuracy. Figure 5.12 shows this. The result is almost identical to the one produced during the Decision Tree section. A ROC curve was also plotted, shown in Figure 5.13,

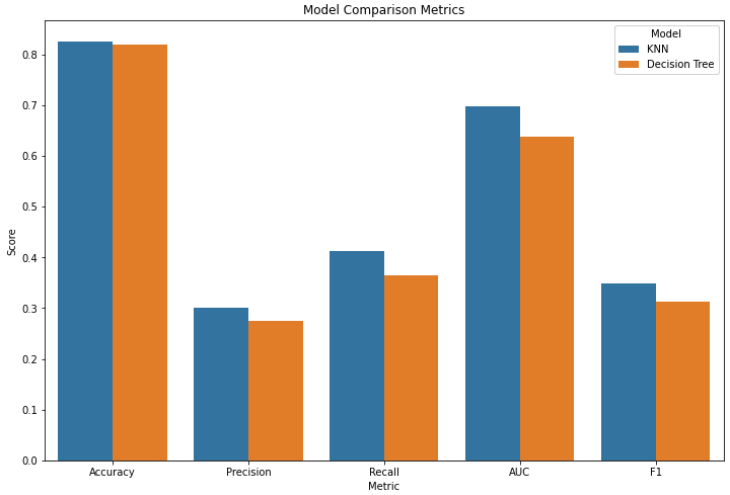
indicating an AUC score of 0.7 (slightly higher than the DT score of 0.64). This indicates that KNN's discriminatory power is slightly better than that of the DT classifier.



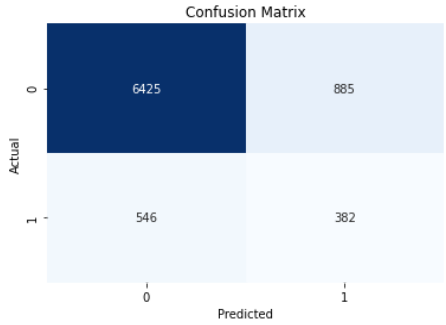


Finally, the models were compared using their respective accuracy, precision, recall, and AUC score. The table below provides this information (Figure 5.13). The scores were also visualized using a barplot (Figure 5.14).

Model	Metric	Score
KNN	Accuracy	0.8262927895120175
KNN	Precision	0.3014996053670087
KNN	Recall	0.41163793103448276
KNN	AUC	0.6976795780461343
KNN	F1	0.34806378132118454
Decision Tree	Accuracy	0.8194950230638505
Decision Tree	Precision	0.2740501212611156
Decision Tree	Recall	0.36530172413793105
Decision Tree	AUC	0.6382352204113402
Decision Tree	F1	0.31316397228637416

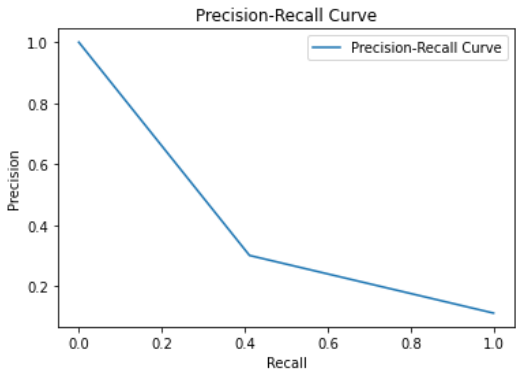


Based on the results, KNN was selected as our predictive model. Below is its confusion matrix, Figure 5.15, and classification report.



	precision	recall	f1-score	support
0	0.92	0.88	0.90	7310
1	0.30	0.41	0.35	928
accuracy			0.83	8238
macro avg	0.61	0.65	0.62	8238
weighted avg	0.85	0.83	0.84	8238

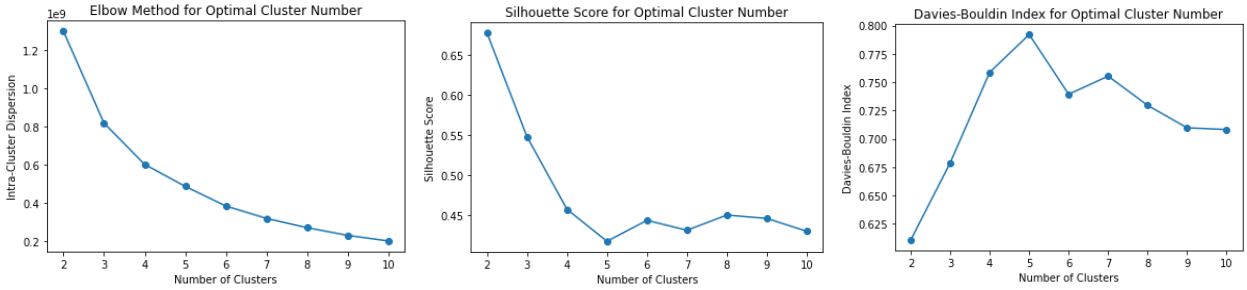
The low precision and recall were further analyzed by plotting a prediction-recall trade-off curve shown in Figure 5.16.



Supplementary: Clustering

Since the dataset is already labeled, there is no need for the implementation of clustering techniques. Nevertheless, we decided to observe the effectiveness of two clustering methods in correctly classifying the points given in the dataset to their correct label. In other words, this brief analysis serves the purpose of comparing the clustering techniques rather than attempting to actually group the observations.

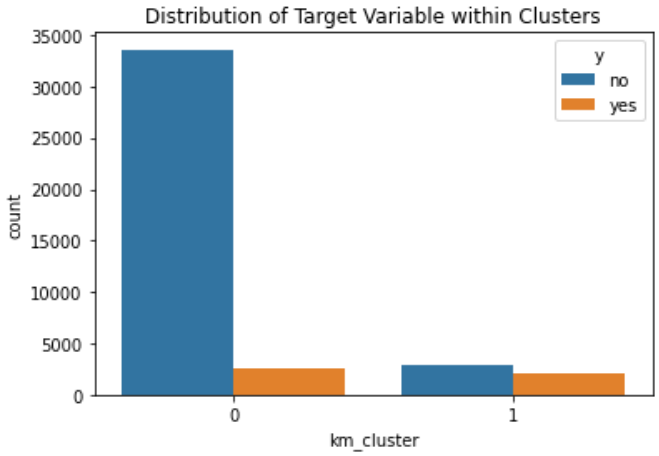
In order to choose the number of clusters with which the clustering algorithms should run, we made use of three methods: the Elbow Plot, the Silhouette Score, and the Davies-Bouldin Index. Below are the three resulting figures:



From the three plots shown above, conflicting conclusions can be made about the number of clusters with which we should separate our data. Of course, since we know that the target variable is binary, we can see that both the Silhouette Score (we take the cluster number associated with the highest score) and the Davies-Bouldin Index (we take the lowest here) provide the correct response of two. However, the elbow method seems to have failed to procure this correct response, instead indicating that the number of clusters should be four (this is, of course, a subjective interpretation). A potential reason for this error could be that the elbow plot

is very reliant on the initial cluster formations, which would undoubtedly favor the majority class.

The first clustering method implemented was KMeans, which relies on re-assigning cluster centers until convergence (no reassignments made). It was set to 2 clusters, and the resulting clusters (grouped by predicted label, colored by actual label) are shown in the figure below:



From the plot above, we can observe the effects of using K-means to cluster the unbalanced version of this data. The first cluster shows the number of occurrences the algorithm classified as having ‘no’ for their label. We can see that this represents the vast majority of occurrences with an actual label of ‘no’. However, it is fairly obvious that the technique was not as successful in correctly identifying the minority class, since it incorrectly classified more often than it did correctly.