Bhavan's Vivekananda College
of Science, Humanities and Commerce

BANK MARKETING ANALYSIS

Using Machine Learning

Presented by Group 7
Priyanka Parthasarathy
Sai Phaneendra
Subhasish Choudhury
Rishi Raj Singh

ABSTRACT

The Bank Marketing Dataset contains data from Portuguese bank campaigns aimed at promoting term deposit subscriptions. This dataset supports predictive modeling for identifying clients likely to subscribe, optimizing marketing strategies. Each record reflects an outreach attempt, allowing analysis across algorithms like logistic regression and neural networks for applications in customer segmentation and response prediction in financial marketing.

OBJECTIVE

To develop predictive models that identify clients with a high likelihood of subscribing to a term deposit, using insights from demographic, behavioral, and campaign-specific data. By analyzing patterns in client responses, the objective is to enhance marketing effectiveness, improve targeting strategies, and optimize resource allocation in future campaigns. This analysis aims to maximize subscription rates while minimizing contact costs, providing the bank with actionable intelligence for data-driven marketing decisions.

CONTENT

Introduction

- **Bank marketing refers to the strategies and tactics used by financial institutions to promote their products and services. It involves targeted campaigns based on customer demographics, behavior, and preferences to drive engagement and conversions.**

- **Bank marketing helps increase customer acquisition, retention, and overall profitability by effectively reaching the right audience with personalized offers. It also enhances brand visibility and fosters trust in the bank's services.**

LITERATURE REVIEW

Literature

Review-1

Elsalamony

Elsalamony (2014) examines data mining techniques, including decision trees and neural networks, to improve customer targeting in bank marketing.

The study finds these methods effective for identifying potential customers, highlighting data mining’s role in enhancing campaign success and efficiency.

Literature

Review-2

D Wang

Wang (2020) explores machine learning techniques, such as decision trees, SVM, and neural networks, to improve customer targeting in bank marketing.

The study finds SVM and neural networks effective for predicting customer responses, enabling more personalized marketing. Emphasis on data preprocessing and feature selection is noted as key to enhancing model accuracy and campaign efficiency.

Literature

Review-3

AM Zaki
N Khodadadi
WH Lim

Zaki et al. (2024) explore machine learning in direct marketing to predict customer subscriptions to bank term deposits.
They compare models like logistic regression, decision trees, and gradient boosting, finding that ensemble models, especially gradient boosting, achieve the highest accuracy.

This study underscores machine learning's potential to enhance targeted marketing efficiency in the banking sector.

DATA PREPROCESSING

Data

Dataset: **Our Dataset consists of 21 variable and 41189 records**

Source: **https://archive.ics.uci.edu/dataset/222/bank+marketing**

- **Categorical Variable**

1. **Age**
2. **Duration**
3. **Campaign**
4. **pdays**
5. **Previous**
6. **Employment Variation Rate**
7. **Consumer Confidence Index**
8. **Consumer Confidence Index**
9. **Euro Interbank Offered Rate - 3 Month**
10. **Number of Employees**

- **Continuous Variable**

1. **Job**
2. **Martial**
3. **Education**
4. **Default**
5. **Housing**
6. **Loan**
7. **Contact**
8. **Month**
9. **Day of Week**
10. **Previous Marketing Campaign**
11. **Term Deposit**

Data Set

age	job	marital	education	default	housing	loan	contact	month	day_of_w	duration	campaign	pdays	previous	poutcome	emp.var.r	cons.price	cons.conf.	euribor3m	nr.employed	y
56	housemai	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
57	services	married	high.scho	unknown	no	no	telephone	may	mon	149	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
37	services	married	high.scho	no	yes	no	telephone	may	mon	226	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
56	services	married	high.scho	no	no	yes	telephone	may	mon	307	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
45	services	married	basic.9y	unknown	no	no	telephone	may	mon	198	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
59	admin.	married	professior	no	no	no	telephone	may	mon	139	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
41	blue-colla	married	unknown	unknown	no	no	telephone	may	mon	217	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
24	technician	single	professior	no	yes	no	telephone	may	mon	380	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
25	services	single	high.scho	no	yes	no	telephone	may	mon	50	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
41	blue-colla	married	unknown	unknown	no	no	telephone	may	mon	55	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
25	services	single	high.scho	no	yes	no	telephone	may	mon	222	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
29	blue-colla	single	high.scho	no	no	yes	telephone	may	mon	137	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
57	housemai	divorced	basic.4y	no	yes	no	telephone	may	mon	293	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
35	blue-colla	married	basic.6y	no	yes	no	telephone	may	mon	146	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
54	retired	married	basic.9y	unknown	yes	yes	telephone	may	mon	174	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
35	blue-colla	married	basic.6y	no	yes	no	telephone	may	mon	312	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
46	blue-colla	married	basic.6y	unknown	yes	yes	telephone	may	mon	440	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
50	blue-colla	married	basic.9y	no	yes	yes	telephone	may	mon	353	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
39	managem	single	basic.9y	unknown	no	no	telephone	may	mon	195	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
30	unemploy	married	high.scho	no	no	no	telephone	may	mon	38	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no
55	blue-colla	married	basic.4y	unknown	yes	no	telephone	may	mon	262	1	999	0	nonexiste	1.1	93.994	-36.4	4.857	5191	no

Data Cleaning

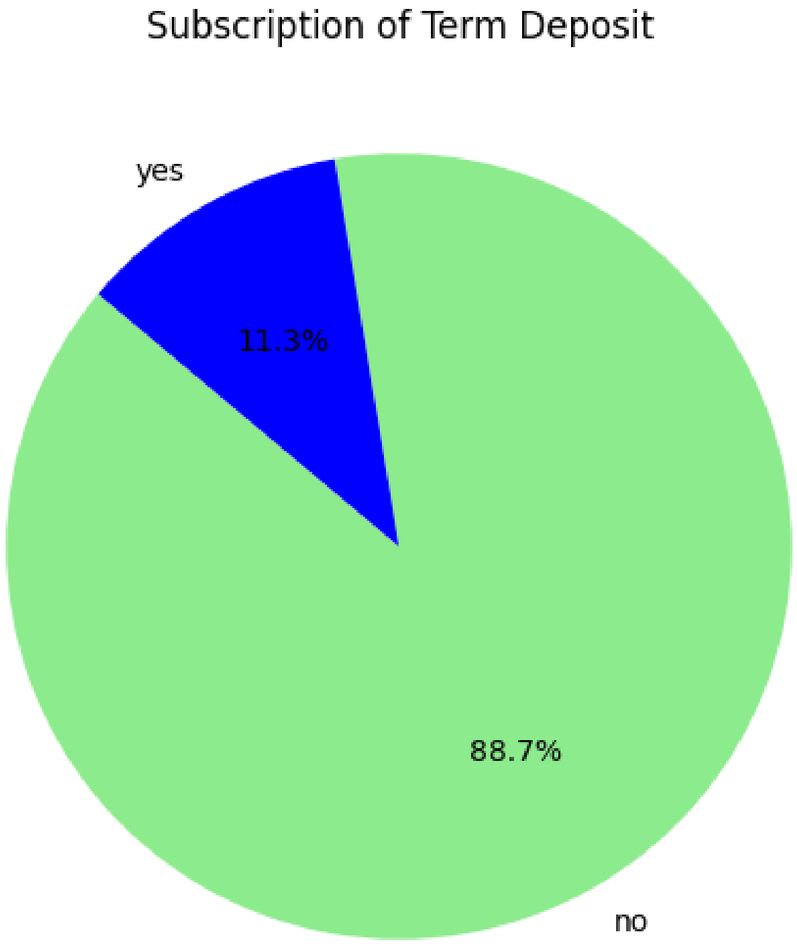
Removing Columns: To avoid confusion from duplicate timestamps on identical dates, we removed the 'Day of week' column.

Quality Control: Scanned for any missing data and verified the uniqueness of values for a clean dataset.

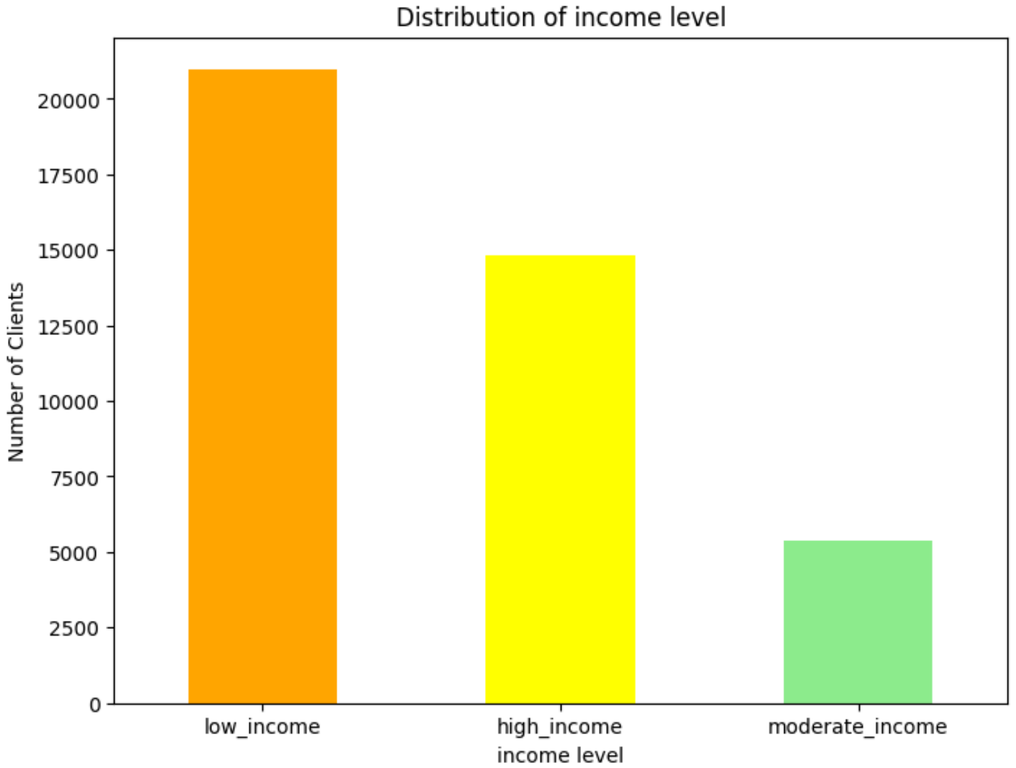
Enabled and Dummified the categorical variables: Converted the 'WeekStatus' variable to binary format, with '0' representing weekdays and '1' for weekends.

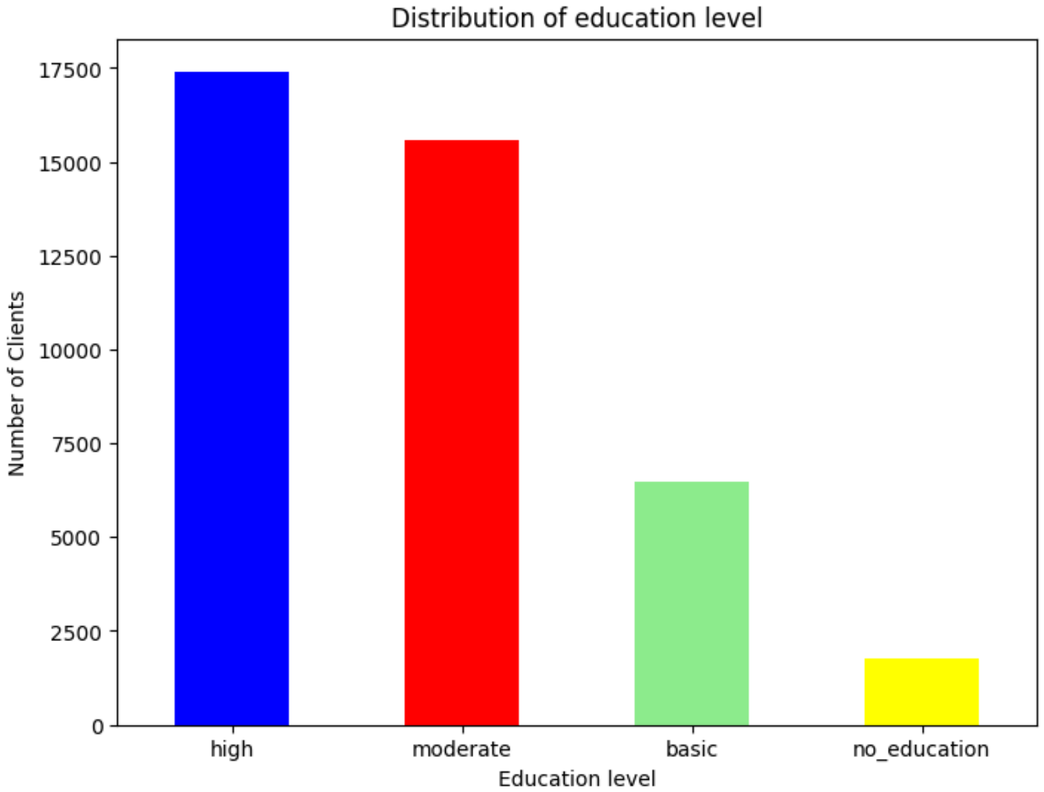
EXPLORATORY DATA ANALYSIS

Pie Chart

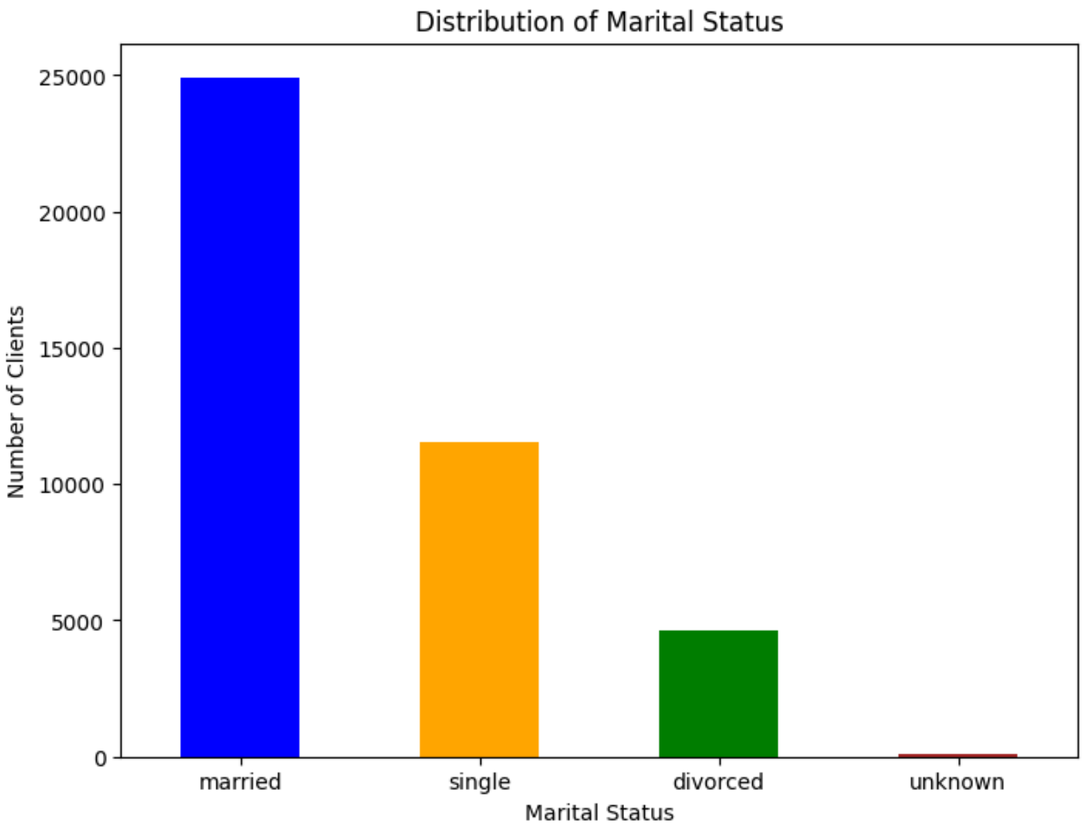


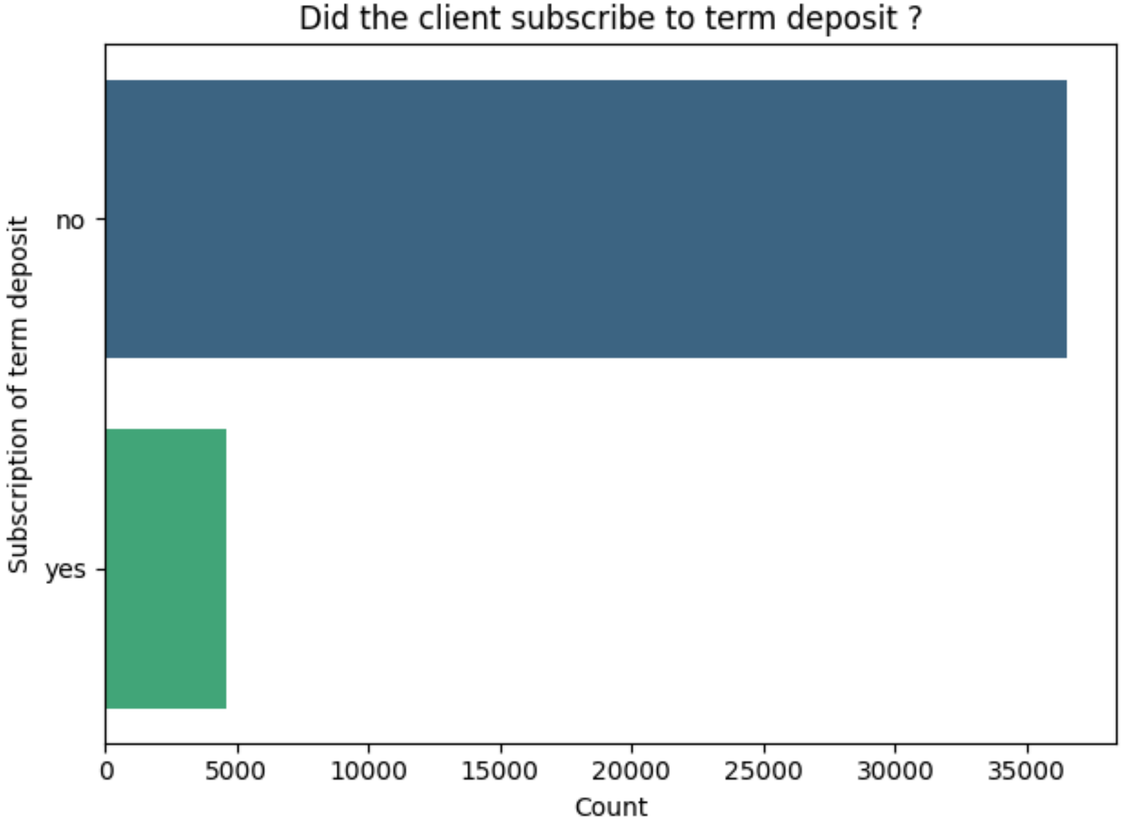
Bar Graph


Distribution of income level
Number of Clients
0
2500
5000
7500
10000
12500
15000
17500
20000
low_income
high_income
moderate_income
income level

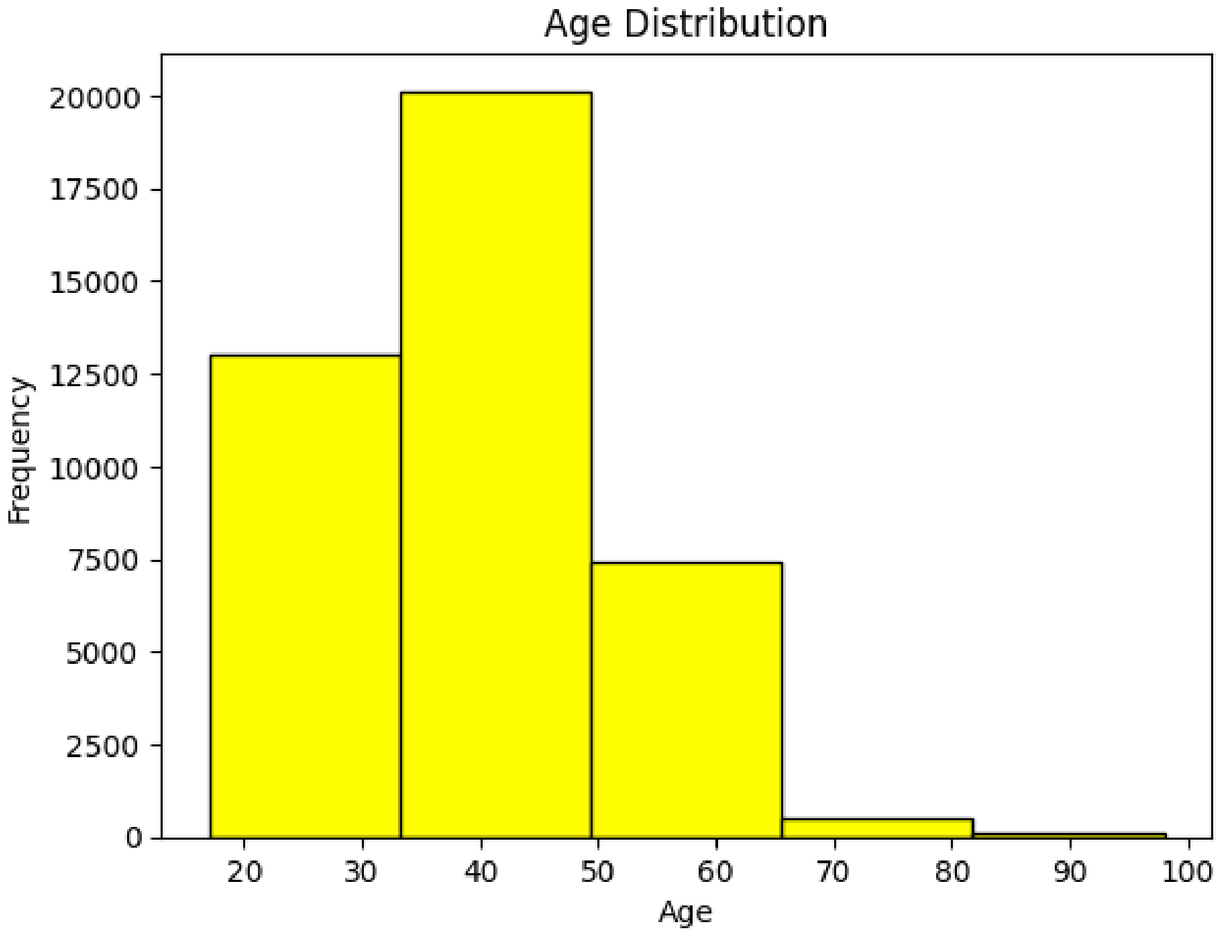

Distribution of education level
Number of Clients
0
2500
5000
7500
10000
12500
15000
17500
high
moderate
basic
no_education
Education level

Bar Graph

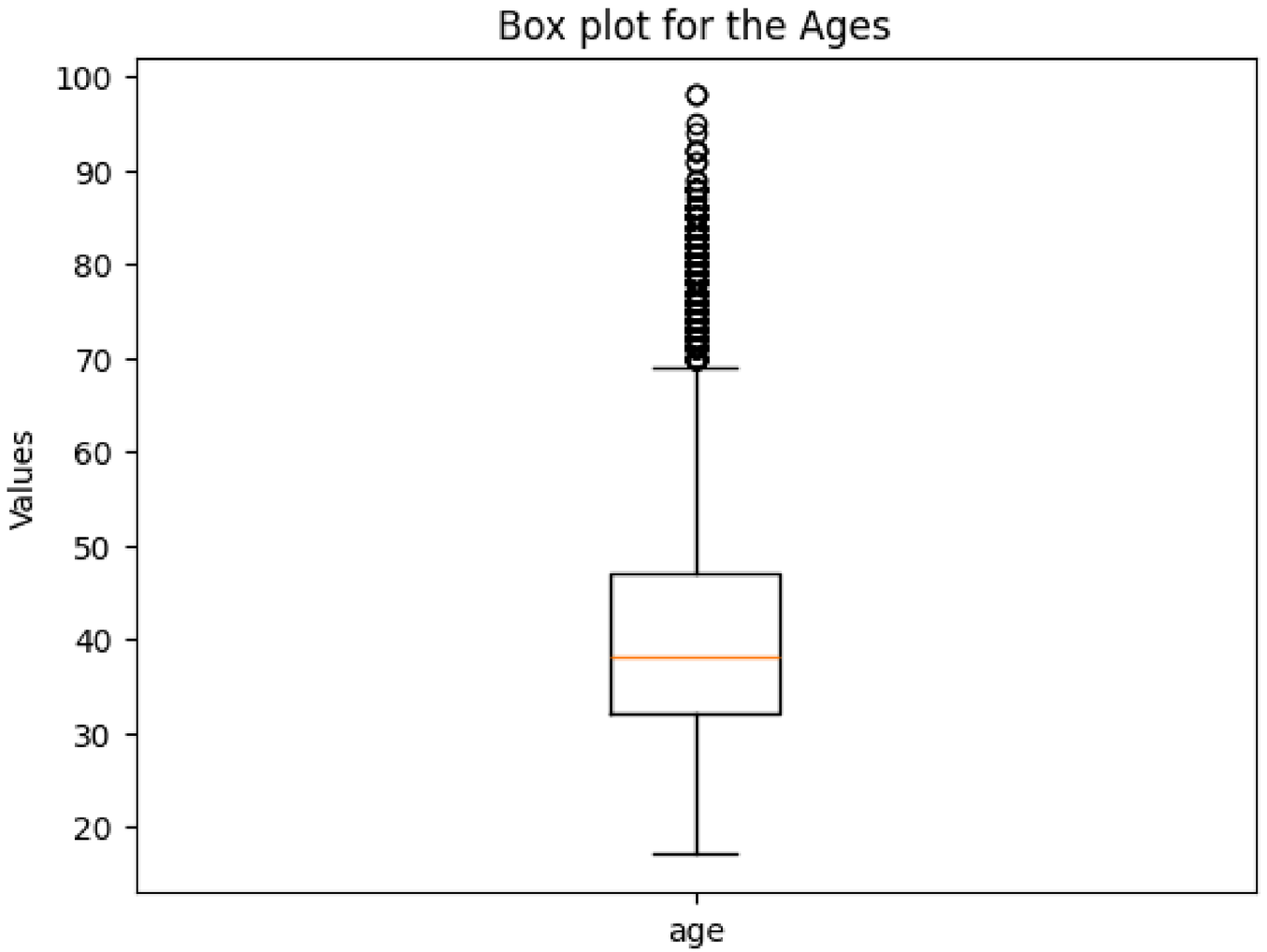

Distribution of Marital Status
Number of Clients
25000
20000
15000
10000
5000
0
married
single
divorced
unknown
Marital Status


Did the client subscribe to term deposit ?
Subscription of term deposit
no
yes
0
5000
10000
15000
20000
25000
30000
35000
Count

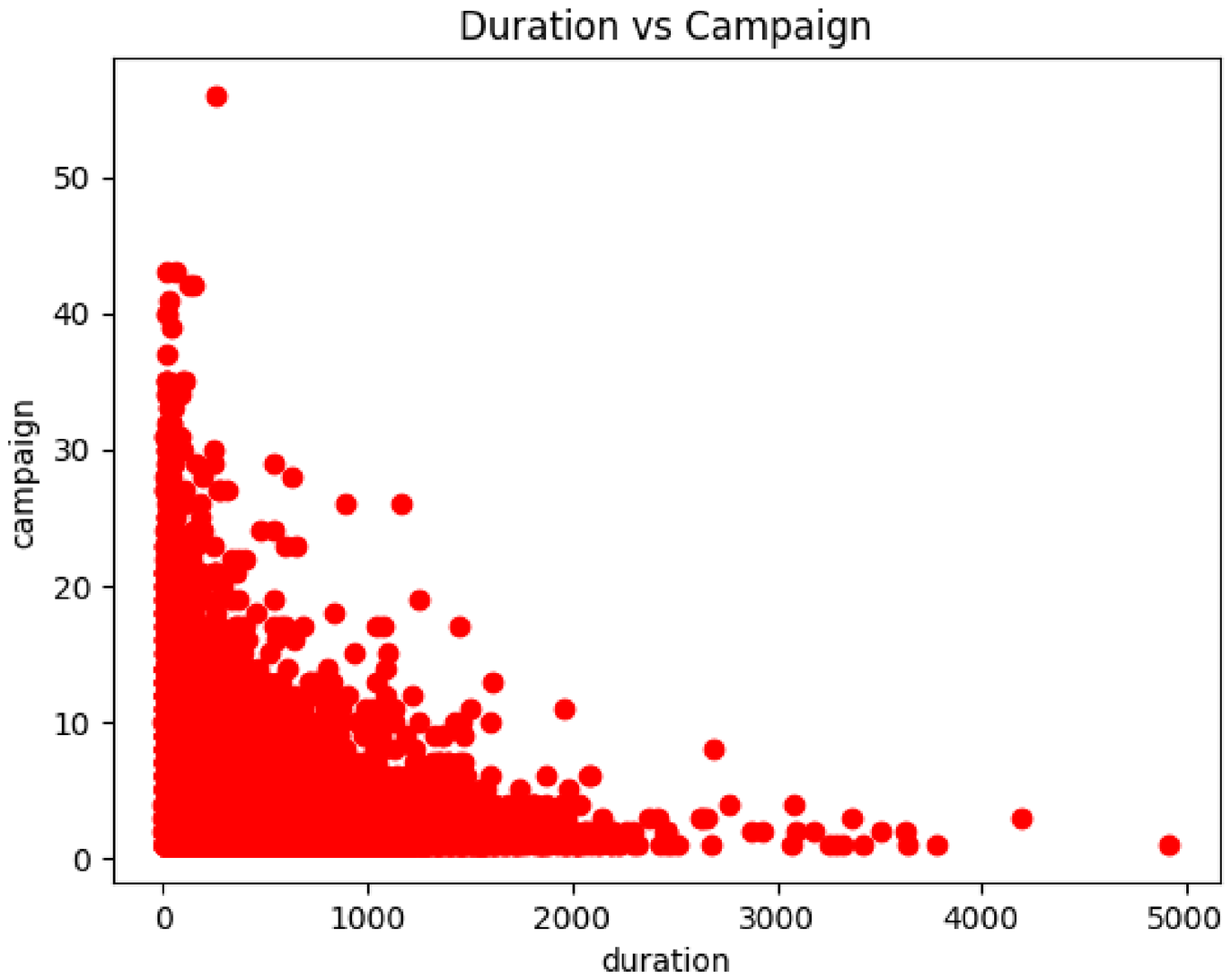
Histogram


Age Distribution
Frequency
20000
17500
15000
12500
10000
7500
5000
2500
0
20
30
40
50
60
70
80
90
100
Age

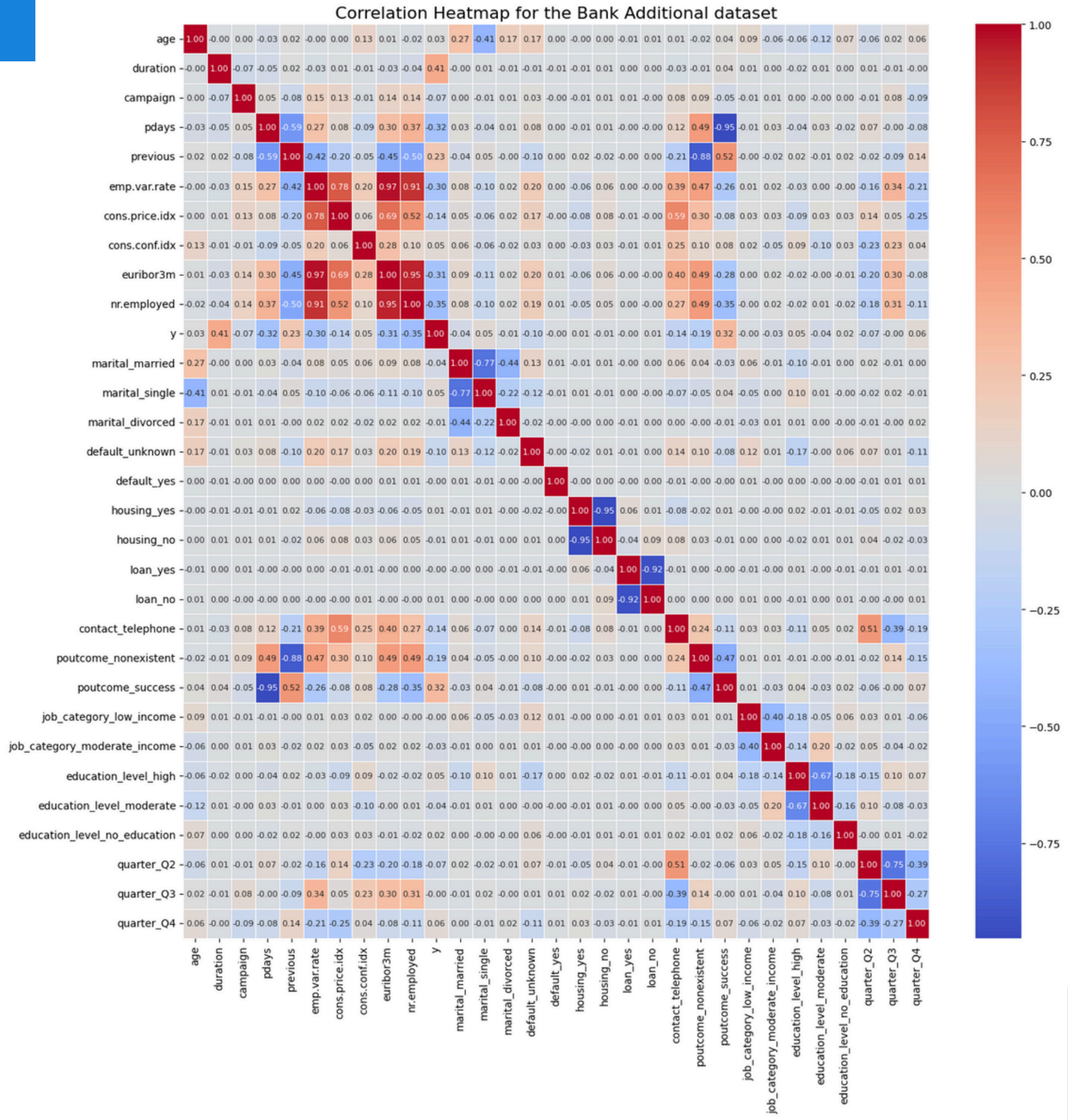
Box Plot



Scatter Plot



Correlation Matrix



Significance Test

OLS Regression Results

Dep. Variable:	y	R-squared (uncentered):	0.418
Model:	OLS	Adj. R-squared (uncentered):	0.418
Method:	Least Squares	F-statistic:	816.7
Date:	Sat, 09 Nov 2024	Prob (F-statistic):	0.00
Time:	13:19:11	Log-Likelihood:	-1892.9
No. Observations:	32950	AIC:	3844.
Df Residuals:	32921	BIC:	4087.
Df Model:	29		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
age	0.0004	0.000	2.369	0.018	6.38e-05	0.001
duration	0.0005	5.48e-06	83.458	0.000	0.000	0.000
campaign	0.0011	0.001	2.196	0.028	0.000	0.002
pdays	-0.0002	2.68e-05	-8.045	0.000	-0.000	-0.000
previous	-0.0122	0.007	-1.760	0.078	-0.026	0.001
emp.var.rate	-0.1086	0.007	-15.580	0.000	-0.122	-0.095
cons.price.idx	0.0875	0.003	29.501	0.000	0.082	0.093
cons.conf.idx	0.0017	0.000	3.786	0.000	0.001	0.003
euribor3m	0.0890	0.008	11.600	0.000	0.074	0.104
nr.employed	-0.0015	5.87e-05	-26.404	0.000	-0.002	-0.001
marital_married	-0.0163	0.032	-0.504	0.614	-0.080	0.047
marital_single	-0.0086	0.032	-0.264	0.791	-0.072	0.055
marital_divorced	-0.0169	0.033	-0.518	0.605	-0.081	0.047
default_unknown	-0.0133	0.004	-3.566	0.000	-0.021	-0.006
default_yes	0.0379	0.181	0.209	0.834	-0.318	0.393
housing_yes	0.0002	0.005	0.051	0.960	-0.009	0.010
housing_no	0.0016	0.005	0.322	0.748	-0.008	0.011
loan_yes	9.822e-05	0.005	0.018	0.985	-0.010	0.011
loan_no	0.0017	0.005	0.360	0.719	-0.008	0.011
contact_telephone	-0.0400	0.005	-7.624	0.000	-0.050	-0.030
poutcome_nonexistent	0.0325	0.009	3.500	0.000	0.014	0.051
poutcome_success	0.1338	0.026	5.059	0.000	0.082	0.186
job_category_low_income	-0.0039	0.003	-1.181	0.238	-0.010	0.003
job_category_moderate_income	-0.0117	0.005	-2.474	0.013	-0.021	-0.002
education_level_high	0.0124	0.005	2.695	0.007	0.003	0.021
education_level_moderate	-0.0002	0.005	-0.040	0.968	-0.009	0.009
education_level_no_education	0.0115	0.008	1.480	0.139	-0.004	0.027
quarter_Q2	-0.2690	0.013	-21.156	0.000	-0.294	-0.244
quarter_Q3	-0.1959	0.013	-14.877	0.000	-0.222	-0.170
quarter_Q4	-0.2707	0.014	-19.386	0.000	-0.298	-0.243

Omnibus:	8211.969	Durbin-Watson:	1.996
Prob(Omnibus):	0.000	Jarque-Bera (JB):	29378.802
Skew:	1.228	Prob(JB):	0.00
Kurtosis:	6.920	Cond. No.	1.03e+16

Significance Test

OLS Regression Results

Dep. Variable:	y	R-squared (uncentered):	0.383
Model:	OLS	Adj. R-squared (uncentered):	0.383
Method:	Least Squares	F-statistic:	1077.
Date:	Sat, 09 Nov 2024	Prob (F-statistic):	0.00
Time:	13:19:11	Log-Likelihood:	-2862.7
No. Observations:	32950	AIC:	5763.
Df Residuals:	32931	BIC:	5923.
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
duration	0.0005	5.45e-06	82.948	0.000	0.000	0.000
campaign	-0.0008	0.001	-1.492	0.136	-0.002	0.000
previous	-0.0007	0.004	-0.197	0.844	-0.008	0.007
emp.var.rate	-0.0543	0.001	-41.780	0.000	-0.057	-0.052
marital_single	0.0049	0.003	1.444	0.149	-0.002	0.011
marital_divorced	-0.0034	0.005	-0.712	0.476	-0.013	0.006
default_unknown	-0.0265	0.004	-7.167	0.000	-0.034	-0.019
default_yes	0.0042	0.187	0.022	0.982	-0.362	0.370
housing_no	-0.0033	0.003	-1.142	0.253	-0.009	0.002
loan_yes	-0.0073	0.004	-1.814	0.070	-0.015	0.001
contact_telephone	0.0234	0.004	5.913	0.000	0.016	0.031
poutcome_success	0.4055	0.010	42.391	0.000	0.387	0.424
job_category_low_income	-0.0180	0.003	-6.002	0.000	-0.024	-0.012
job_category_moderate_income	-0.0262	0.005	-5.570	0.000	-0.035	-0.017
education_level_high	-0.0184	0.004	-4.965	0.000	-0.026	-0.011
education_level_moderate	-0.0330	0.004	-8.776	0.000	-0.040	-0.026
education_level_no_education	0.0014	0.008	0.180	0.857	-0.014	0.016
quarter_Q3	0.0677	0.004	16.536	0.000	0.060	0.076
quarter_Q4	0.0116	0.005	2.436	0.015	0.002	0.021

Omnibus:	9336.777	Durbin-Watson:	1.995
Prob(Omnibus):	0.000	Jarque-Bera (JB):	31410.005
Skew:	1.426	Prob(JB):	0.00
Kurtosis:	6.839	Cond. No.	4.69e+04

MULTICOLLINEARITY CHECK

	variables	VIF
0	age	20.8
1	duration	2.0
2	campaign	1.9
3	pdays	341.0
4	previous	6.6
5	emp.var.rate	61.0
6	cons.price.idx	38919.2
7	cons.conf.idx	165.7
8	euribor3m	480.6
9	nr.employed	46498.3
10	marital_married	312.8
11	marital_single	145.8
12	marital_divorced	58.7
13	default_unknown	1.4
14	default_yes	1.0
15	housing_yes	inf
16	housing_no	inf
17	loan_yes	inf
18	loan_no	inf
19	contact_telephone	5.1
20	poutcome_nonexistent	37.3
21	poutcome_success	11.5
22	job_category_low_income	2.7
23	job_category_moderate_income	1.5
24	education_level_high	4.5
25	education_level_moderate	3.8
26	education_level_no_education	1.3
27	quarter_Q2	42.6
28	quarter_Q3	29.2
29	quarter_Q4	11.9

	variables	VIF
0	age	20.8
1	duration	2.0
2	campaign	1.9
3	pdays	346.8
4	previous	6.6
5	emp.var.rate	60.4
6	cons.price.idx	38593.3
7	cons.conf.idx	164.8
8	euribor3m	476.2
9	nr.employed	46123.6
10	marital_married	317.2
11	marital_single	148.6
12	marital_divorced	59.8
13	default_unknown	1.4
14	default_yes	1.0
15	housing_no	1.9
16	loan_yes	7.5
17	loan_no	36.8
18	contact_telephone	5.0
19	poutcome_nonexistent	37.5
20	poutcome_success	11.7
21	job_category_low_income	2.7
22	job_category_moderate_income	1.5
23	education_level_high	4.5
24	education_level_moderate	3.9
25	education_level_no_education	1.3
26	quarter_Q2	42.6
27	quarter_Q3	29.6
28	quarter_Q4	11.9

housing_yes was removed

MULTICOLLINEARITY CHECK

	variables	VIF
0	age	20.8
1	duration	2.0
2	campaign	1.9
3	pdays	345.2
4	previous	6.5
5	emp.var.rate	33.1
6	cons.price.idx	1495.1
7	cons.conf.idx	134.0
8	euribor3m	184.3
9	marital_married	316.4
10	marital_single	148.3
11	marital_divorced	59.6
12	default_unknown	1.4
13	default_yes	1.0
14	housing_no	1.9
15	loan_yes	7.5
16	loan_no	36.8
17	contact_telephone	4.2
18	poutcome_nonexistent	37.1
19	poutcome_success	11.7
20	job_category_low_income	2.7
21	job_category_moderate_income	1.5
22	education_level_high	4.5
23	education_level_moderate	3.9
24	education_level_no_education	1.3
25	quarter_Q2	42.6
26	quarter_Q3	29.5
27	quarter_Q4	11.6

nr.employed was removed

	variables	VIF
0	age	20.3
1	duration	2.0
2	campaign	1.9
3	pdays	259.9
4	previous	5.4
5	emp.var.rate	28.4
6	cons.conf.idx	107.5
7	euribor3m	156.8
8	marital_married	210.1
9	marital_single	97.7
10	marital_divorced	40.1
11	default_unknown	1.4
12	default_yes	1.0
13	housing_no	1.9
14	loan_yes	7.3
15	loan_no	35.9
16	contact_telephone	4.1
17	poutcome_nonexistent	33.1
18	poutcome_success	8.8
19	job_category_low_income	2.7
20	job_category_moderate_income	1.5
21	education_level_high	4.4
22	education_level_moderate	3.8
23	education_level_no_education	1.3
24	quarter_Q2	41.2
25	quarter_Q3	27.9
26	quarter_Q4	11.4

'cons.price.idx was removed

MULTICOLLINEARITY CHECK

	variables	VIF
0	age	20.0
1	duration	2.0
2	campaign	1.9
3	previous	5.2
4	emp.var.rate	26.1
5	cons.conf.idx	95.3
6	euribor3m	143.8
7	marital_married	175.4
8	marital_single	81.2
9	marital_divorced	33.7
10	default_unknown	1.4
11	default_yes	1.0
12	housing_no	1.9
13	loan_yes	7.3
14	loan_no	35.5
15	contact_telephone	4.1
16	poutcome_nonexistent	33.0
17	poutcome_success	1.5
18	job_category_low_income	2.7
19	job_category_moderate_income	1.5
20	education_level_high	4.4
21	education_level_moderate	3.8
22	education_level_no_education	1.3
23	quarter_Q2	40.4
24	quarter_Q3	27.2
25	quarter_Q4	11.4

pdays was removed

	variables	VIF
0	age	19.3
1	duration	2.0
2	campaign	1.9
3	previous	4.9
4	emp.var.rate	18.5
5	cons.conf.idx	64.2
6	euribor3m	104.3
7	marital_single	1.8
8	marital_divorced	1.2
9	default_unknown	1.4
10	default_yes	1.0
11	housing_no	1.9
12	loan_yes	7.1
13	loan_no	34.6
14	contact_telephone	4.0
15	poutcome_nonexistent	31.1
16	poutcome_success	1.5
17	job_category_low_income	2.7
18	job_category_moderate_income	1.5
19	education_level_high	4.4
20	education_level_moderate	3.8
21	education_level_no_education	1.3
22	quarter_Q2	38.7
23	quarter_Q3	25.2
24	quarter_Q4	11.2

marital_married was removed

MULTICOLLINEARITY CHECK

	variables	VIF
0	age	18.5
1	duration	2.0
2	campaign	1.9
3	previous	4.9
4	emp.var.rate	2.2
5	cons.conf.idx	62.9
6	marital_single	1.8
7	marital_divorced	1.2
8	default_unknown	1.4
9	default_yes	1.0
10	housing_no	1.9
11	loan_yes	6.8
12	loan_no	32.7
13	contact_telephone	3.8
14	poutcome_nonexistent	30.3
15	poutcome_success	1.5
16	job_category_low_income	2.7
17	job_category_moderate_income	1.4
18	education_level_high	4.3
19	education_level_moderate	3.8
20	education_level_no_education	1.3
21	quarter_Q2	32.2
22	quarter_Q3	19.9
23	quarter_Q4	7.5

euribor3m was removed

	variables	VIF
0	age	18.0
1	duration	2.0
2	campaign	1.9
3	previous	4.5
4	emp.var.rate	2.2
5	marital_single	1.7
6	marital_divorced	1.2
7	default_unknown	1.4
8	default_yes	1.0
9	housing_no	1.9
10	loan_yes	6.1
11	loan_no	29.3
12	contact_telephone	3.4
13	poutcome_nonexistent	27.9
14	poutcome_success	1.5
15	job_category_low_income	2.7
16	job_category_moderate_income	1.4
17	education_level_high	4.2
18	education_level_moderate	3.7
19	education_level_no_education	1.3
20	quarter_Q2	24.8
21	quarter_Q3	17.5
22	quarter_Q4	6.4

cons.conf.idx was removed

MULTICOLLINEARITY CHECK

	variables	VIF
0	age	17.3
1	duration	2.0
2	campaign	1.9
3	previous	4.4
4	emp.var.rate	2.1
5	marital_single	1.7
6	marital_divorced	1.2
7	default_unknown	1.4
8	default_yes	1.0
9	housing_no	1.8
10	loan_yes	1.2
11	contact_telephone	3.4
12	poutcome_nonexistent	26.3
13	poutcome_success	1.5
14	job_category_low_income	2.6
15	job_category_moderate_income	1.4
16	education_level_high	4.1
17	education_level_moderate	3.6
18	education_level_no_education	1.3
19	quarter_Q2	20.4
20	quarter_Q3	14.6
21	quarter_Q4	5.4

loan_no was removed

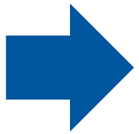
	variables	VIF
0	age	16.2
1	duration	2.0
2	campaign	1.9
3	previous	1.8
4	emp.var.rate	2.1
5	marital_single	1.7
6	marital_divorced	1.2
7	default_unknown	1.4
8	default_yes	1.0
9	housing_no	1.8
10	loan_yes	1.2
11	contact_telephone	3.3
12	poutcome_success	1.5
13	job_category_low_income	2.6
14	job_category_moderate_income	1.4
15	education_level_high	4.0
16	education_level_moderate	3.5
17	education_level_no_education	1.3
18	quarter_Q2	14.7
19	quarter_Q3	10.3
20	quarter_Q4	4.1

poutcome_nonexistent was removed

MULTICOLLINEARITY CHECK

	variables	VIF
0	duration	2.0
1	campaign	1.9
2	previous	1.8
3	emp.var.rate	2.1
4	marital_single	1.5
5	marital_divorced	1.2
6	default_unknown	1.4
7	default_yes	1.0
8	housing_no	1.8
9	loan_yes	1.2
10	contact_telephone	3.3
11	poutcome_success	1.4
12	job_category_low_income	2.6
13	job_category_moderate_income	1.4
14	education_level_high	4.0
15	education_level_moderate	3.5
16	education_level_no_education	1.3
17	quarter_Q2	8.7
18	quarter_Q3	5.4
19	quarter_Q4	2.4

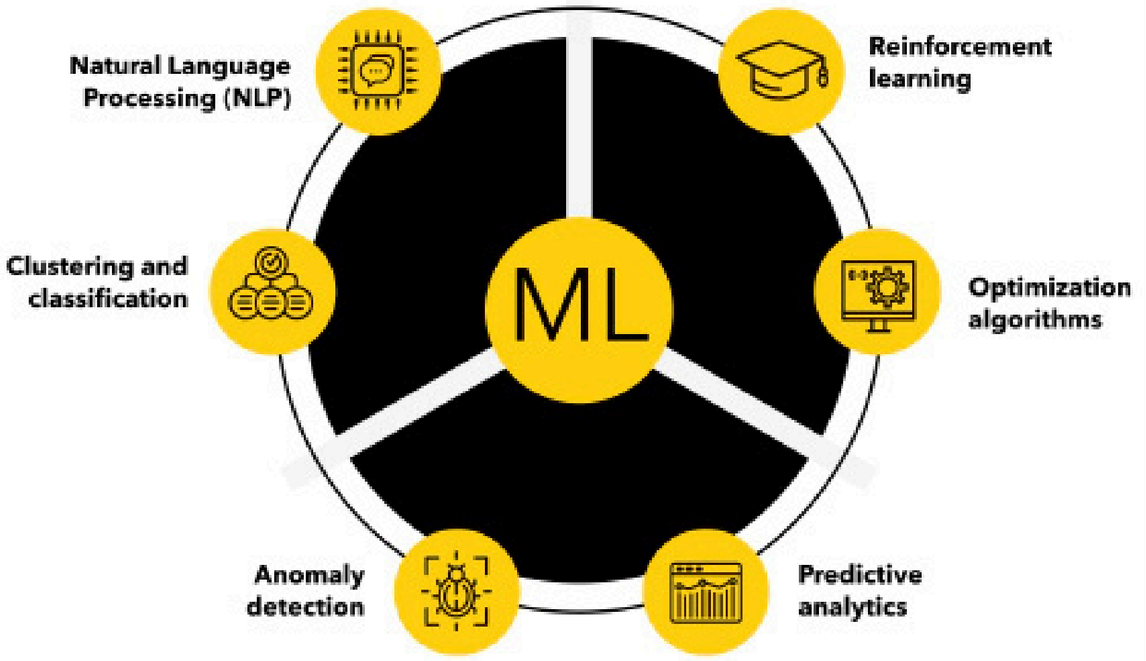
age was removed



	variables	VIF
0	duration	1.9
1	campaign	1.8
2	previous	1.7
3	emp.var.rate	2.0
4	marital_single	1.5
5	marital_divorced	1.2
6	default_unknown	1.3
7	default_yes	1.0
8	housing_no	1.8
9	loan_yes	1.2
10	contact_telephone	2.7
11	poutcome_success	1.4
12	job_category_low_income	2.2
13	job_category_moderate_income	1.4
14	education_level_high	2.8
15	education_level_moderate	2.5
16	education_level_no_education	1.2
17	quarter_Q3	2.7
18	quarter_Q4	1.3

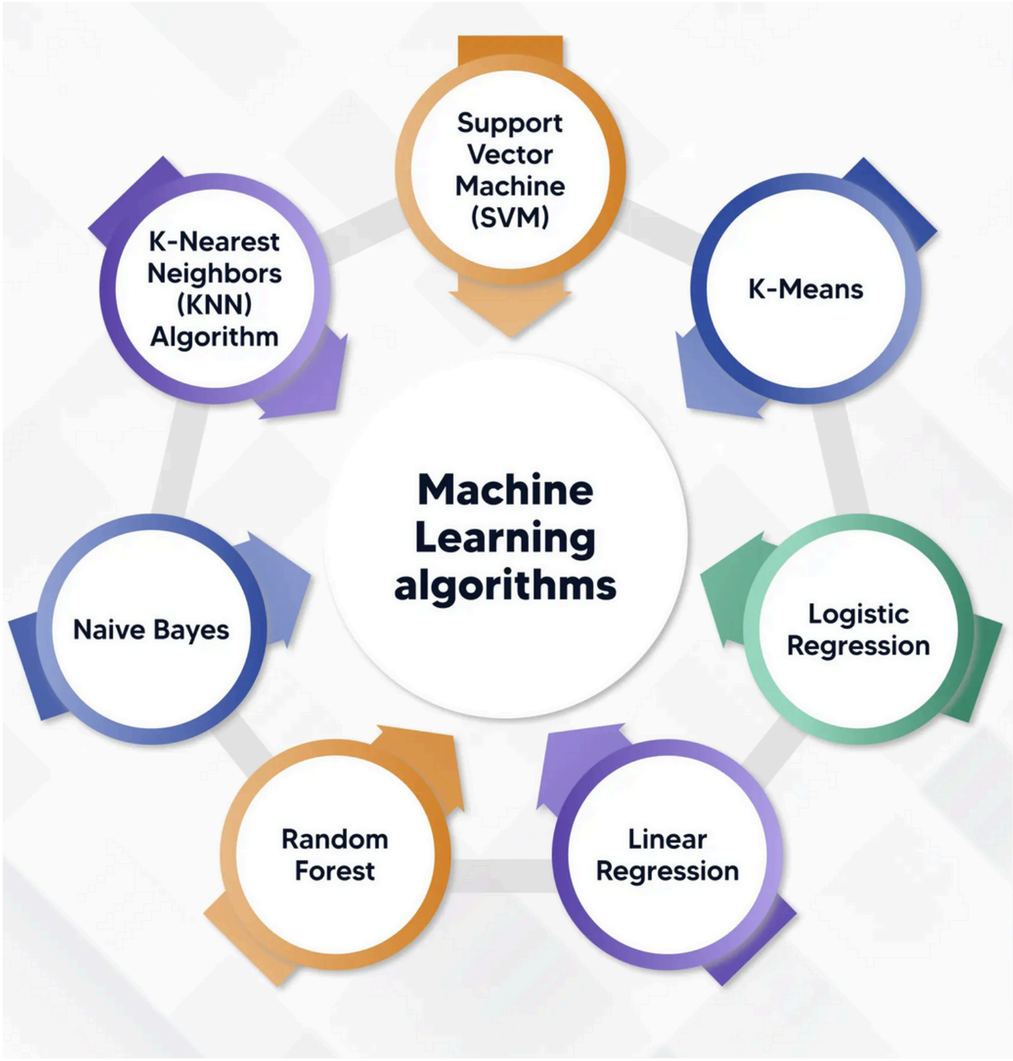
quarter_Q2 was removed

MACHINE LEARNING ALGORITHMS



ML ALGORITHMS

- Multiple Linear Regression
- K-Nearest Neighbors(KNN)
- Decision Tree
- Random Forest
- Bagging
- Boosting
- Support Vector Machine(SVM)



80-20 Train-Test Split

Algorithm	Model - 1 Accuracy	Model - 2 Accuracy
Logistics Regression	0.9100	0.9056
KNN	0.9090	0.8924
Decision Tree Regressor	0.8892	0.8775
Random Forest	0.9079	0.9000
XG Boosting	0.9119	0.9100
ADA Boosting	0.9100	0.9000
SVM	0.8959	0.8057

Model 1- Before VIF | Model 2- After VIF

75-25 Train-Test Split

Algorithm	Model - 1 Accuracy	Model - 2 Accuracy
Logistics Regression	0.9000	0.9063
KNN	0.9124	0.8938
Decision Tree Regressor	0.8924	0.8771
Random Forest	0.9100	0.9000
XG Boosting	0.9145	0.9075
ADA Boosting	0.9100	0.9100
SVM	0.8951	0.8079

Model 1- Before VIF | Model 2- After VIF

70-30 Train-Test Split

Algorithm	Model - 1 Accuracy	Model - 2 Accuracy
Logistics Regression	0.9100	0.9049
KNN	0.9117	0.8922
Decision Tree Regressor	0.8880	0.8765
Random Forest	0.9100	0.9000
XG Boosting	0.9151	0.9070
ADA Boosting	0.9100	0.9000
SVM	0.8965	0.8063

Model 1- Before VIF | Model 2- After VIF

60-40 Train-Test Split

Algorithm	Model - 1 Accuracy	Model - 2 Accuracy
Logistics Regression	0.9100	0.9038
KNN	0.9094	0.8924
Decision Tree Regressor	0.8892	0.8780
Random Forest	0.9100	0.9000
XG Boosting	0.9116	0.9071
ADA Boosting	0.9100	0.9000
SVM	0.8962	0.8026

Model 1- Before VIF | Model 2- After VIF

Algorithms Comparison

70:30 Train - Test Split

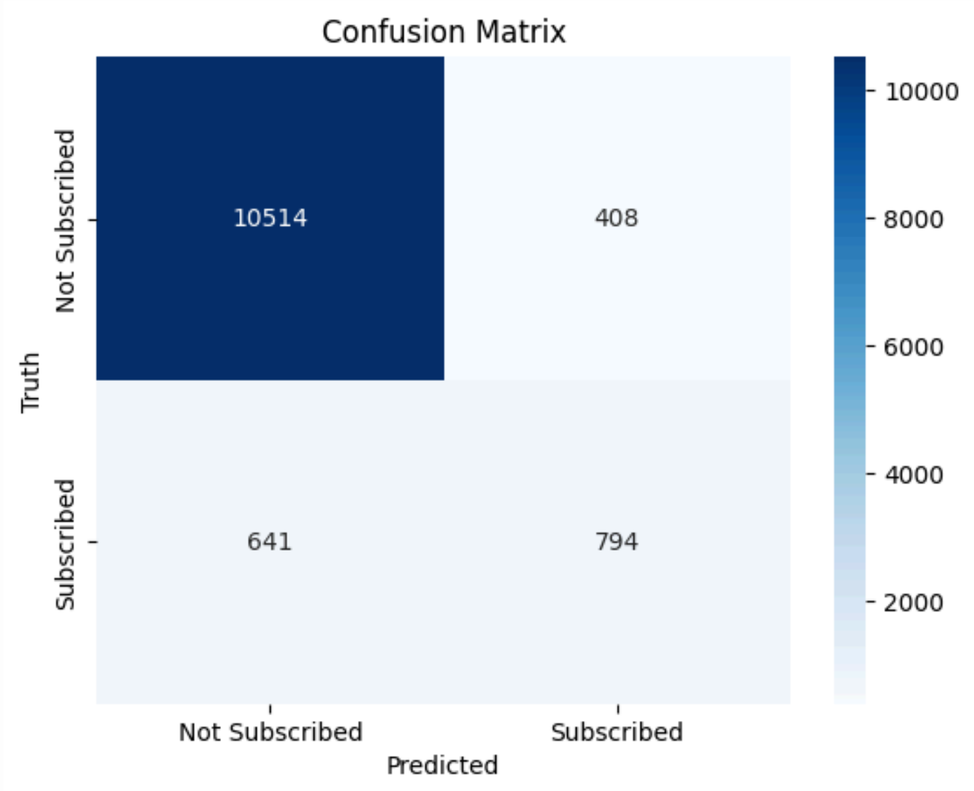
Algorithm	Model - 1 Accuracy Before VIF
Logistics Regression	0.9100
KNN	0.9117
Decision Tree Regressor	0.8880
Random Forest	0.9100
XG Boosting	0.9151
ADA Boosting	0.9100
SVM	0.8965

75:25 Train - Test Split

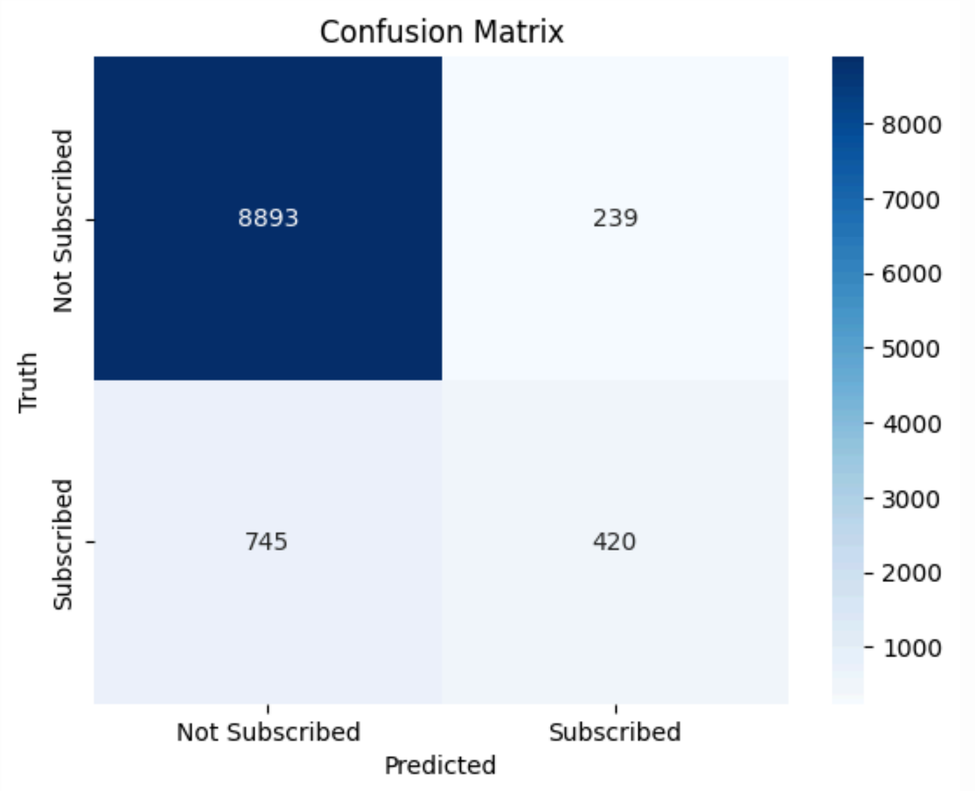
Algorithm	Model - 2 Accuracy After VIF
Logistics Regression	0.9063
KNN	0.8938
Decision Tree Regressor	0.8771
Random Forest	0.9000
XG Boosting	0.9075
ADA Boosting	0.9100
SVM	0.8079

Confusion Matrics

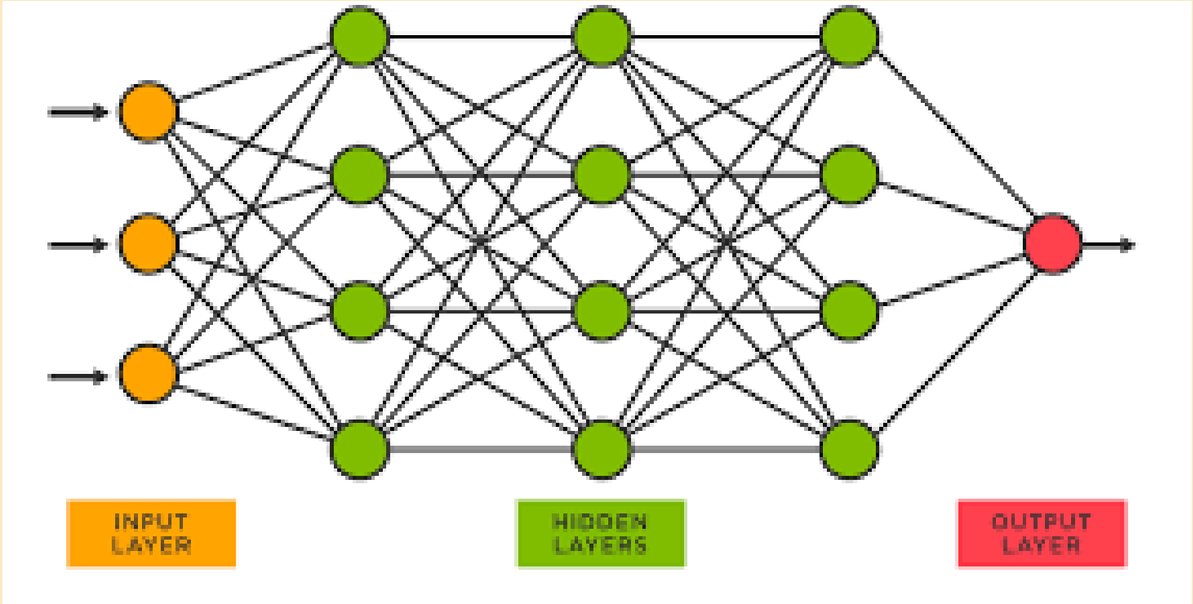
70-30 Train Test Split
XG Boosting Before VIF



75-25 Train Test Split
ADA Boosting After VIF



Architecture of Artificial Neural Network

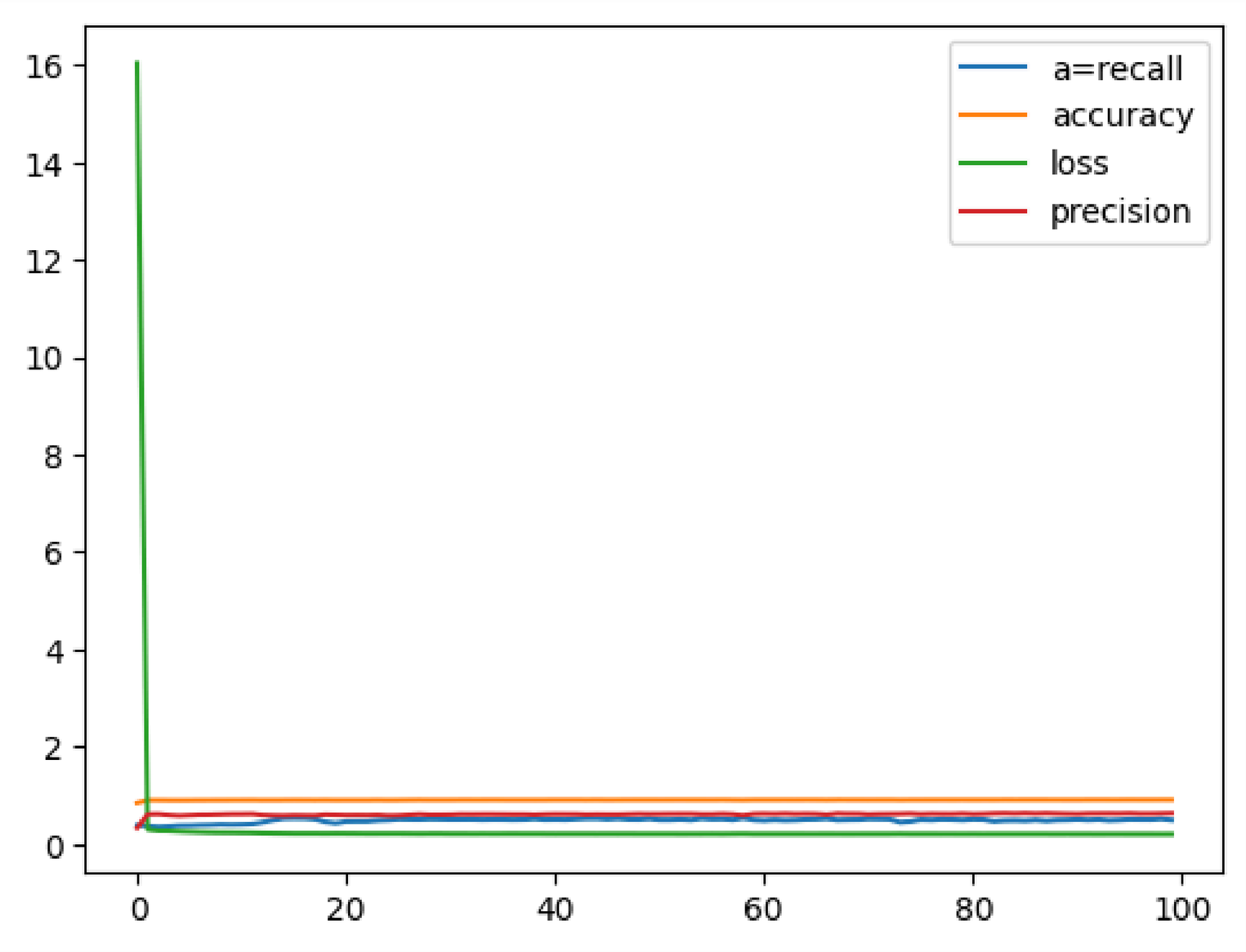


Before VIF

Train Test	Architecture	Optimizer	Epochs	Accuracy
80-20	10-7-5-1	Adam	100	0.9053
75-25	10-7-5-1	Adam	100	0.9014
70-30	10-7-5-1	Adam	100	0.8890
60-40	10-7-5-1	Adam	100	0.9132

After VIF

Train Test	Architecture	Optimizer	Epochs	Accuracy
80-20	10-7-5-1	Adam	100	0.9084
75-25	10-7-5-1	Adam	100	0.9041
70-30	10-7-5-1	Adam	100	0.9047
60-40	10-7-5-1	Adam	100	0.9021



Train-Test Split	**60-40**
Architecture	**10-7-5-1**
Optimizer	**Adam**
Epochs	**100**

SUMMARY

- This study aimed to determine the most effective machine learning approaches for forecasting term deposit subscriptions within the banking sector.

- Among the tested classification algorithms, the XG Boosting demonstrated outstanding performance with an accuracy of 91.5%, outperforming other methods in predicting term deposit subscriptions.

- In another model evaluation, ADA Boosting achieved the highest accuracy in its group with a score of 91%, which, while slightly less than the XG Boosting still proved highly effective.

FUTURE SCOPES

- Effective data preprocessing is essential for converting raw financial data (e.g., loans, savings, and deposits) into actionable insights.
- These insights enable banks to enhance personalization and better understand customer preferences.
- Refined customer segmentation helps in crafting more targeted and impactful marketing campaigns.
- Banks can leverage these strategies to innovate financial products and stay ahead in a competitive market.
- A focus on data-driven decision-making supports long-term growth and improved customer satisfaction.

WORK DISTRIBUTION

Team member 1 Priyanka Parthasarathy	Collect information about Steel Industry & Literature Review
Team member 2 Rishi Raj Singh	Data Preprocessing
Team member 3 Subhasish Choudhury	Exploratory Data Analysis
Team member 4 Sai Phaneendra	Implement Machine Learning Algorithms

Collab Notebook

THANK YOU

Priyanka Parthasarathy

Rishi Raj Singh

Subhasish Choudhury

Sai Phaneendra

APPENDIX

CONTENT

	Age	Job	Martial	Education	Default	Housing	Loan	\
0	37	admin.	married	university.degree	no	yes	no	
1	45	technician	married	professional.course	no	yes	no	
2	34	blue-collar	married	basic.9y	no	yes	no	
3	38	blue-collar	married	unknown	no	yes	no	
4	33	admin.	married	high.school	no	yes	no	

	Contact	Month	Day_of_week	...	Campaign	pdays	Previous	poutcome	\
0	cellular	jul	wed	...	1	999	0	nonexistent	
1	cellular	jul	wed	...	1	999	0	nonexistent	
2	cellular	jul	wed	...	1	999	0	nonexistent	
3	cellular	jul	wed	...	1	999	0	nonexistent	
4	cellular	jul	wed	...	1	999	0	nonexistent	

	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_employed	Y
0	1.4	93.918	-42.7	4.962	5228.1	no
1	1.4	93.918	-42.7	4.962	5228.1	no
2	1.4	93.918	-42.7	4.962	5228.1	no
3	1.4	93.918	-42.7	4.962	5228.1	no
4	1.4	93.918	-42.7	4.962	5228.1	no

[5 rows x 21 columns]

NO.OF COLUMNS

```
data.columns #no. of columns
```

```
Index(['Age', 'Job', 'Martial', 'Education', 'Default', 'Housing', 'Loan',
       'Contact', 'Month', 'Day_of_week', 'Duration', 'Campaign', 'pdays',
       'Previous', 'poutcome', 'emp_var_rate', 'cons_price_idx',
       'cons_conf_idx', 'euribor3m', 'nr_employed', 'Y'],
      dtype='object')
```

DATA FRAME STRUCTURE AND DATA TYPES

```
data.info() #summary of DataFrame structure and data types
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             41188 non-null  int64
 1   Job             41188 non-null  object
 2   Martial         41188 non-null  object
 3   Education       41188 non-null  object
 4   Default         41188 non-null  object
 5   Housing         41188 non-null  object
 6   Loan            41188 non-null  object
 7   Contact         41188 non-null  object
 8   Month           41188 non-null  object
 9   Day_of_week     41188 non-null  object
 10  Duration        41188 non-null  int64
 11  Campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  Previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp_var_rate    41188 non-null  float64
 16  cons_price_idx  41188 non-null  float64
 17  cons_conf_idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr_employed     41188 non-null  float64
 20  Y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

NULL VALUES

```
data.isna().sum() #no. of null values
```

	0
Age	0
Job	0
Martial	0
Education	0
Default	0
Housing	0
Loan	0
Contact	0
Month	0
Duration	0
Campaign	0
pdays	0
Previous	0
poutcome	0
emp_var_rate	0
cons_price_idx	0
cons_conf_idx	0
euribor3m	0
nr_employed	0
Y	0

dtype: int64

Categorization

```python
# Define a function to categorize the job types
def categorize_job(job):
    if job in ['management', 'entrepreneur', 'admin.']:
        return 'high_earning'
    elif job in ['technician','blue-collar', 'services','self-emp
        return 'moderate_earning'
    elif job in [ 'housemaid', 'retired', 'unemployed', 'student'
        return 'low_earning'


# Apply the categorization to the 'Job' column
data['Job_category'] = data['Job'].apply(categorize_job)

# Display the first few rows with the new 'Job_category' column
data[['Job', 'Job_category']].head()
```

	Job	Job_category
0	admin.	high_earning
1	technician	moderate_earning
2	blue-collar	moderate_earning
3	blue-collar	moderate_earning
4	admin.	high_earning

```python
# Define a function to categorize the education types
def categorize_education(education):
    if education in ['unknown', 'illiterate']:
        return 'no_edu'
    elif education in ['basic.4y', 'basic.6y']:
        return 'basic_edu'
    elif education in ['high.school', 'basic.9y']:
        return 'med_edu'
    elif education in ['professional.course', 'university.degree
        return 'high_edu'


# Apply the categorization to the 'Education' column
data['Education_category'] = data['Education'].apply(categorize_

# Display the first few rows with the new 'Education_category' c
data[['Education', 'Education_category']].head()
```

	Education	Education_category
0	university.degree	high_edu
1	professional.course	high_edu
2	basic.9y	med_edu
3	unknown	no_edu
4	high.school	med_edu

```python
# Define a function to categorize the months
def categorize_month(month):
    if month in ['jan', 'feb', 'mar']:
        return 'Q1'
    elif month in ['apr', 'may', 'jun']:
        return 'Q2'
    elif month in ['jul', 'aug', 'sep']:
        return 'Q3'
    elif month in ['oct', 'nov', 'dec']:
        return 'Q4'


# Apply the categorization to the 'Month' column
data['Month_category'] = data['Month'].apply(categorize_mo

# Display the first few rows with the new 'Month_category'
data[['Month', 'Month_category']].head()
```

	Month	Month_category
0	jul	Q3
1	jul	Q3
2	jul	Q3
3	jul	Q3
4	jul	Q3

DATA ENCODING

```
data_org.columns
```

```
Index(['Age', 'Martial', 'Default', 'Housing', 'Loan', 'Contact', 'Duration',
       'Campaign', 'pdays', 'Previous', 'poutcome', 'emp_var_rate',
       'cons_price_idx', 'cons_conf_idx', 'euribor3m', 'nr_employed', 'Y',
       'Job_category', 'Education_category', 'Month_category'],
      dtype='object')
```

```
# Reorder the categorical columns to ensure 'unknown' is first
data_org['Loan'] = pd.Categorical(data_org['Loan'], categories=['unknown', 'yes', 'no'], ordered=True)
data_org['Martial'] = pd.Categorical(data_org['Martial'], categories=['unknown', 'married', 'single', 'divorced'], ordered=True)
data_org['Housing'] = pd.Categorical(data_org['Housing'], categories=['unknown', 'yes', 'no'], ordered=True)

# List of categorical columns including your custom categories
categorical_columns = [
    'Martial', 'Default', 'Housing',
    'Loan', 'Contact', 'poutcome', 'Job_category', 'Education_category', 'Month_category'
]

# Apply dummy variable encoding (one-hot encoding with drop_first=True)
data_encoded = pd.get_dummies(data_org, columns=categorical_columns, drop_first=True)

# Convert 'yes' and 'no' to 1 and 0 respectively for remaining columns
data_encoded = data_encoded.replace({'yes': 1, 'no': 0})


# Display the first few rows of the dummy variable encoded dataset
print(data_encoded.head())
```

```
data_encoded.columns
```

```
Index(['Age', 'Duration', 'Campaign', 'pdays', 'Previous', 'emp_var_rate',
       'cons_price_idx', 'cons_conf_idx', 'euribor3m', 'nr_employed', 'Y',
       'Martial_married', 'Martial_single', 'Martial_divorced',
       'Default_unknown', 'Default_yes', 'Housing_yes', 'Housing_no',
       'Loan_yes', 'Loan_no', 'Contact_telephone', 'poutcome_nonexistent',
       'poutcome_success', 'Job_category_low_earning',
       'Job_category_moderate_earning', 'Education_category_high_edu',
       'Education_category_med_edu', 'Education_category_no_edu',
       'Month_category_Q2', 'Month_category_Q3', 'Month_category_Q4'],
      dtype='object')
```

```
# Identify boolean columns
boolean_columns = data_encoded.select_dtypes(include='bool').columns

# Convert only boolean columns to 0/1 (int)
data_encoded[boolean_columns] = data_encoded[boolean_columns].astype(int)

# Display the first few rows and check the dtypes to confirm only boolean columns are convert
print(data_encoded.dtypes)
print(data_encoded.head())
```

LOGISTIC REGRESSION

Logistic regression

```
#70 - 30

# fit a logistic regression model and store the class predictions
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1e9)

logreg.fit(X_train5, y_train5)
predictions5 = logreg.predict(X_test5)
print(predictions5)
```

```
[0 0 0 ... 0 0 0]
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
  n_iter_i = _check_optimize_result(
```

```
from sklearn.metrics import confusion_matrix
```

```
z=confusion_matrix(y_test5, predictions5)
z
```

```
array([[10698,   267],
       [  929,   463]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test5,predictions5))
```

	precision	recall	f1-score	support
0	0.92	0.98	0.95	10965
1	0.63	0.33	0.44	1392
accuracy			0.90	12357
macro avg	0.78	0.65	0.69	12357
weighted avg	0.89	0.90	0.89	12357

K Nearest Neighbors before VIF

```
from sklearn.neighbors import KNeighborsClassifier
```

70-30 Ratio

```
model=KNeighborsClassifier(n_neighbors=25)
model.fit(X_train3,y_train3)
```

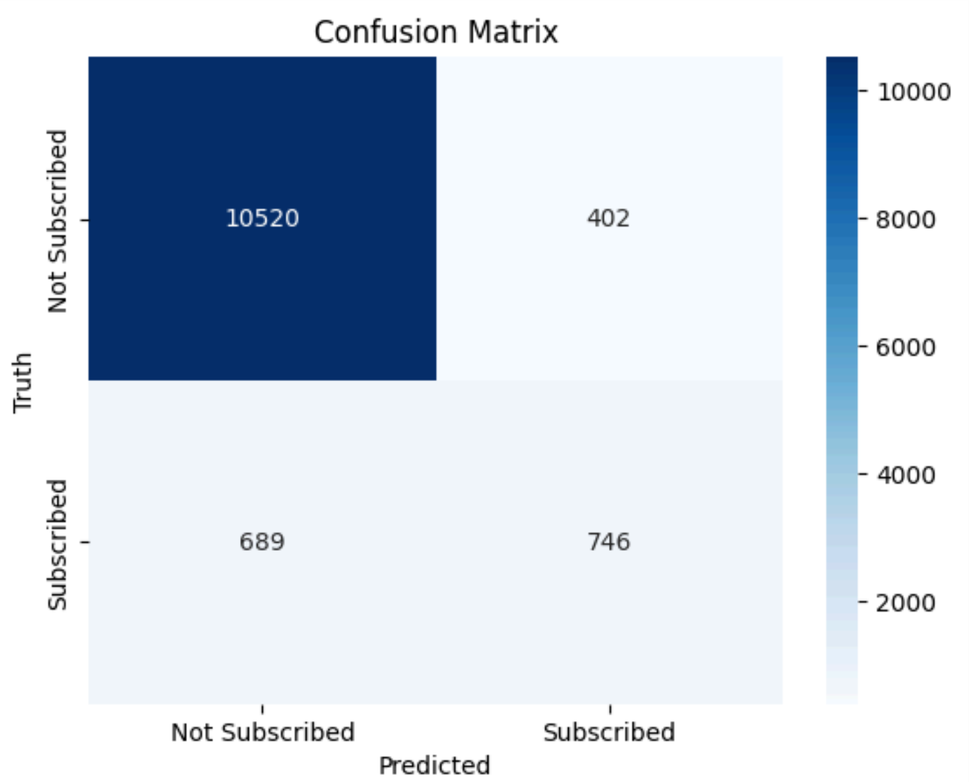
```
KNeighborsClassifier(n_neighbors=25)
```

```
df=pd.DataFrame({'Predicted':y_pred3,'Actual':y_test3})
print(df)
```

```
       Predicted  Actual
13159          0       0
3048           0       0
5167           0       0
3266           0       0
18189          0       0
...          ...     ...
4197           0       0
7470           0       1
20338          0       0
12832          0       0
9781           0       0

[12357 rows x 2 columns]
```

KNN



SVM

Support Vector Machine before VIF

```
[ ]  from sklearn.svm import SVC
```

70-30 Ratio

```
[ ]  model = SVC(kernel='sigmoid')
     model.fit(X_train3, y_train3)
```

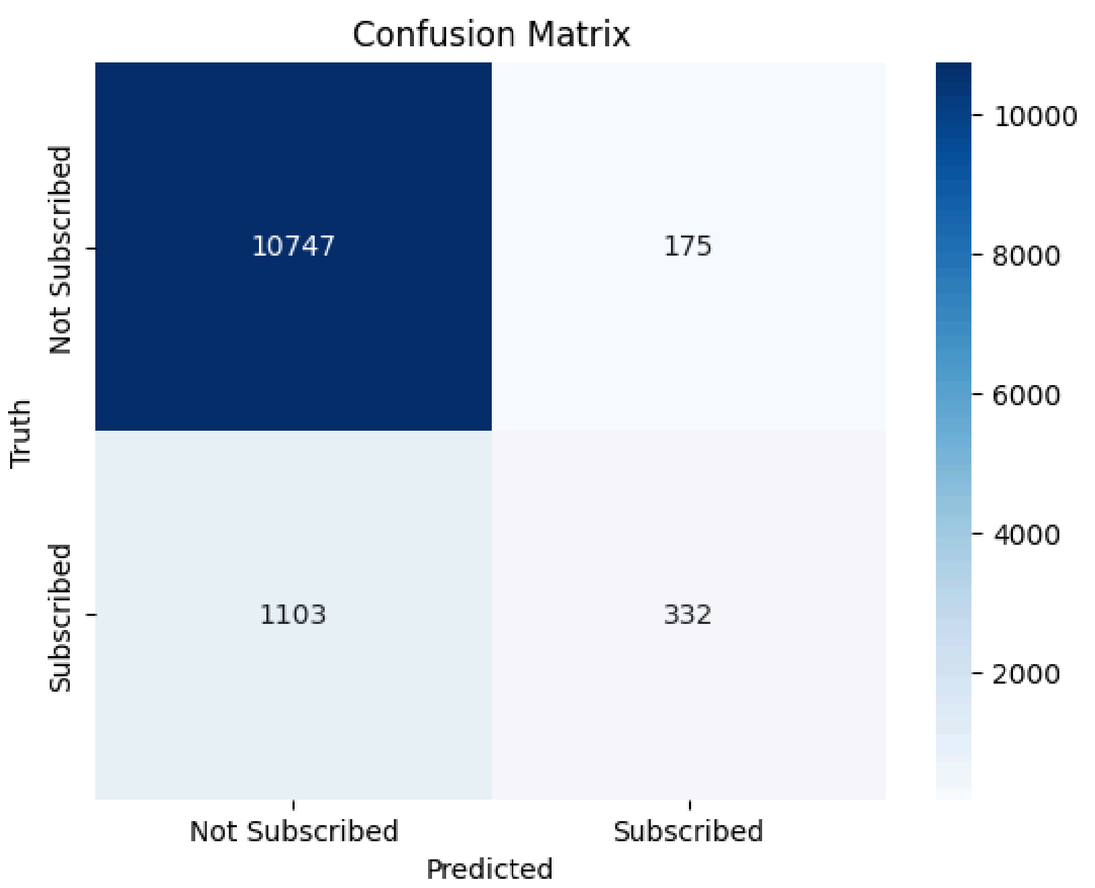
```
SVC(kernel='sigmoid')
```

```
[ ]  y_pred3 = model.predict(X_test3)
     print(y_pred3)
```

```
[0 0 0 ... 0 0 0]
```

```
[ ]  from sklearn.metrics import accuracy_score
     accuracy_score(y_test3,y_pred3)
```

```
0.8965768390386016
```



Decision Tree Classifier

Decision Tree Classifier before VIF

```
[ ]  import pandas as pd
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.model_selection import train_test_split
     from sklearn import metrics
```

70-30 Ratio

```
[ ]  clf = DecisionTreeClassifier()
     clf3 = clf.fit(X_train3,y_train3)
     print(clf3)
```

```
DecisionTreeClassifier()
```

```
[ ]  y_pred3 = clf.predict(X_test3)
     print(y_pred3)
```

```
[0 0 0 ... 0 0 0]
```

```
[ ]  print("Accuracy:",metrics.accuracy_score(y_test3, y_pred3))
```

```
Accuracy: 0.8880796309783928
```

ADA Boosting

ADA Boosting before VIF

```
[ ] import pandas as pd
    import seaborn as sns
    from sklearn.tree import DecisionTreeClassifier
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.ensemble import AdaBoostClassifier
    from sklearn.metrics import accuracy_score, classification_report
```

70-30 Ratio

```
[ ] model = AdaBoostClassifier(n_estimators=50, random_state=42)
    model.fit(X_train3, y_train3)
```

```
AdaBoostClassifier
AdaBoostClassifier(random_state=42)
```

```
[ ] y_pred3 = model.predict(X_test3)
    print(y_pred3)
```

```
[0 0 0 ... 0 0 0]
```

```
accuracy = accuracy_score(y_test3, y_pred3)
print(f"Accuracy: {accuracy:.2f}")
print(classification_report(y_test3, y_pred3))
```

```
Accuracy: 0.91
              precision    recall  f1-score   support

           0       0.93      0.97      0.95     10922
           1       0.67      0.40      0.50      1435

    accuracy                           0.91     12357
   macro avg       0.80      0.69      0.72     12357
weighted avg       0.90      0.91      0.90     12357
```

XG Boosting

XG Boosting before VIF

```
import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

```
[ ] print(classification_report(y_test3, y_pred3))
```

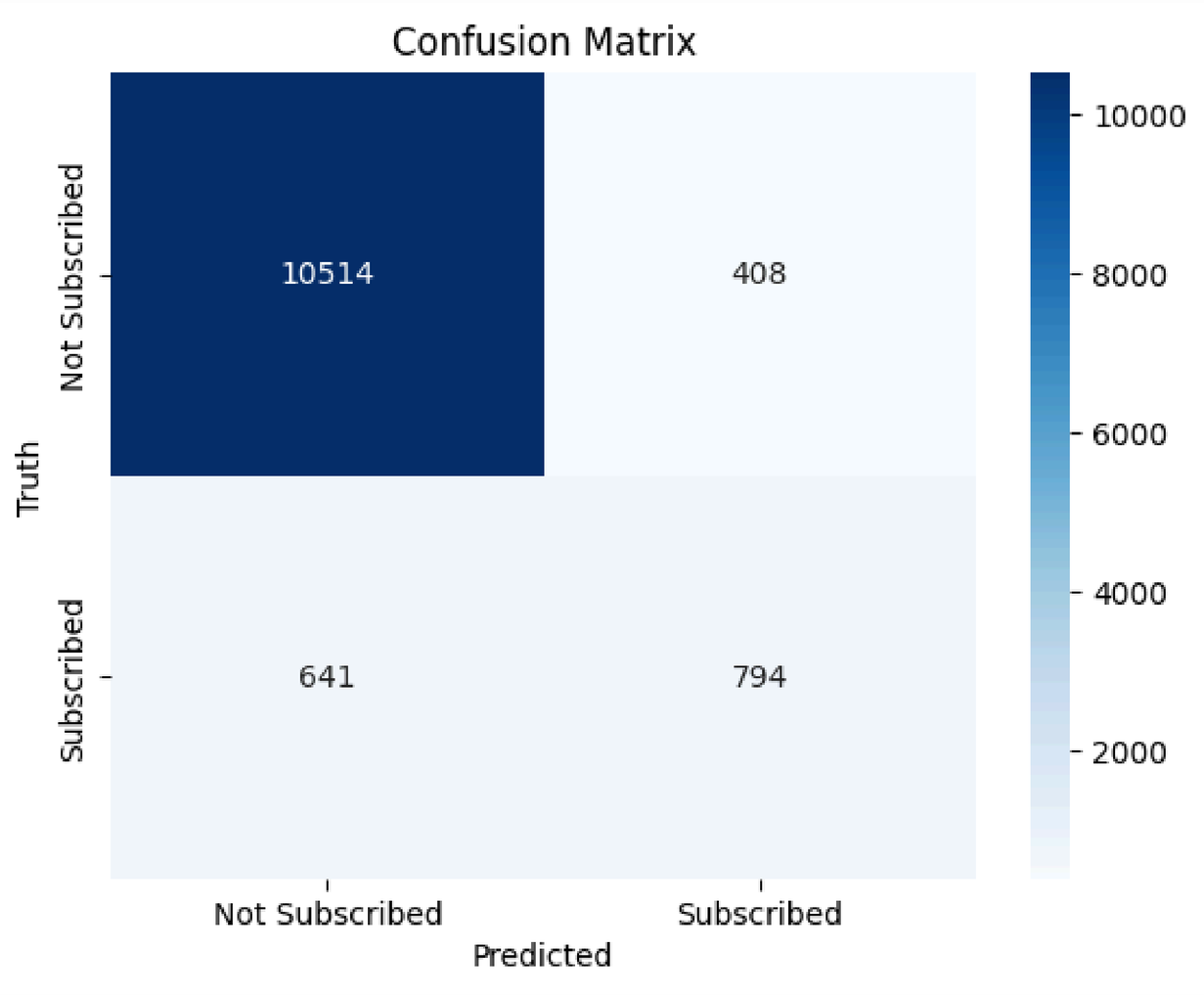
```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     10922
           1       0.66      0.55      0.60      1435

    accuracy                           0.92     12357
   macro avg       0.80      0.76      0.78     12357
weighted avg       0.91      0.92      0.91     12357
```

```
[ ] print("Accuracy:",metrics.accuracy_score(y_test3, y_pred3))
```

```
Accuracy: 0.9151088451889617
```



Artificial Neural Network

60-40 Ratio

```
tf.random.set_seed(42)

# STEP1: Creating the model

model= tf.keras.Sequential([
                            tf.keras.layers.Dense(10, activation='relu'),
                            tf.keras.layers.Dense(7, activation='relu'),
                            tf.keras.layers.Dense(5, activation='relu'),
                            tf.keras.layers.Dense(1, activation='sigmoid')
])

# STEP2: Compiling the model

model.compile(loss= tf.keras.losses.binary_crossentropy,
              optimizer= tf.keras.optimizers.Adam(learning_rate=0.001),
              metrics= [tf.keras.metrics.BinaryAccuracy(name='accuracy'),
                        tf.keras.metrics.Precision(name='precision'),
                        tf.keras.metrics.Recall(name='a=recall')
              ]
              )

# STEP1: Fit the model

history= model.fit(X_train4, y_train4, epochs= 100)
```