CAPSTONE FINAL REPORT

GROUP 1

JULY 31, 2024

Batch details	PGPDSE-FT BANGALORE JAN24
Team members	Shaik Shameer Bhushan Choudhary Purva Vaishnav Sam Wilkerson Shreya Bhatkande S.P Gajapathii
Domain of Project	Bank Marketing
Proposed project title	Predicting Term Deposit subscription
Group Number	1
Team Leader	Shaik Shameer
Mentor Name	Ms. Anjana Agrawal

Date: 31-07-2024

Signature of the Mentor

Signature of the Team Leader

CONTENTS

Industry Review

In the banking sector, direct marketing campaigns are a common practice used to promote products such as term deposits, loans, and credit cards. These campaigns typically involve contacting potential customers through various channels, such as phone calls, emails, and direct mail. The goal is to identify and convert leads into customers, thereby increasing the bank's revenue and customer base.

Literature Survey

Numerous studies and publications focus on the application of machine learning and statistical models to improve the effectiveness of marketing campaigns. Previous research has explored various algorithms for customer segmentation, predictive modeling, and response optimization. Key publications in this area include studies on logistic regression for response prediction, the use of decision trees and random forests for classification tasks, and the application of boosting algorithms like XGBoost for improving model accuracy.

Dataset Description:

Data Dictionary:

Here is the description of the dataset:

Column Name	Description
Age (numeric)	Age of client
job (categorical)	type of job ('admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
Marital (categorical)	marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)
Education (categorical)	' Basic.4y','basic.6y','basic.9y','high. school, 'illiterate', 'professional. course', 'university. Degree', 'unknown'
Default (categorical)	has credit in default? (categorical: 'no', 'yes', 'unknown')
Housing (categorical)	has housing loan? ('no', 'yes', 'unknown')
Loan (categorical)	has personal loan? ('no', 'yes', 'unknown')
Contact (categorical)	Communication type ('cellular', 'telephone')

Month (categorical)	Last contact month of the year (‘jan’ to ‘dec’)
Day_of_week (Categorical)	Last contact day of the week (‘mon’ to ‘fri’)
Campaign (numeric)	number of contacts performed during this campaign and for this client
Pdays (numeric)	number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
Previous (numeric)	number of contacts performed before this campaign and for this client
Poutcome (categorical)	outcome of the previous marketing campaign ('failure', 'nonexistent', ‘success')
emp.var.rate (numeric)	employment variation rate - quarterly indicator
Cons.price.idx (numeric)	consumer price index - monthly indicator
Cons.conf.idx (numeric)	consumer confidence index - monthly indicator
Euribor3m (numeric)	euribor 3-month rate - daily indicator
Nr.employed (numeric)	number of employees - quarterly indicator
Output Variable (Target)	
Y (categorical)	has the client subscribed a term deposit? (binary: 'yes', 'no')

Variable categorization (count of numeric and categorical)

11 categorical variables and 9 numerical variables (5 continuous-numeric and 4 discrete-numeric)

```
1  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         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  campaign        41188 non-null  int64
 11  pdays           41188 non-null  int64
 12  previous        41188 non-null  int64
 13  poutcome        41188 non-null  object
 14  emp.var.rate    41188 non-null  float64
 15  cons.price.idx  41188 non-null  float64
 16  cons.conf.idx   41188 non-null  float64
 17  euribor3m       41188 non-null  float64
 18  nr.employed     41188 non-null  float64
 19  y               41188 non-null  object
dtypes: float64(5), int64(4), object(11)
memory usage: 6.3+ MB
```

Preprocessing Data Analysis:

Pre-Processing Data Analysis (count of missing/ null values, redundant columns, etc.)

count of missing values - Initially, we had a no missing value

```
1  df.isnull().sum()
```

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       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
```

Unknown values:

We have seen that there are no null values in our dataset. But the search does not stop here as we've only cross-verified for our numeric columns. For the categorical columns, there are high chances of null values to be equivalent to 'unknown' so we shall convert 'unknown' to NA to finally count the total number of missing values in our categoric columns

```
df = df.replace(to_replace='unknown', value=np.nan)

df.isnull().sum() / len(df) *100
```

```
age                0.000000
job                0.801204
marital            0.194231
education          4.202680
default           20.872584
housing            2.403613
loan               2.403613
contact            0.000000
month              0.000000
day_of_week        0.000000
campaign           0.000000
pdays              0.000000
previous           0.000000
poutcome           0.000000
emp.var.rate       0.000000
cons.price.idx     0.000000
cons.conf.idx      0.000000
euribor3m          0.000000
nr.employed        0.000000
y                  0.000000
dtype: float64
```

Alternate sources of data that can supplement the core dataset (at least 2-3 columns)

The data is enriched by the addition of five new social and economic features/attributes (national wide indicators from a ~10M population country), published by the Banco de Portugal and publicly available at: https://www.bportugal.pt/estatisticasweb.

This dataset is almost identical to the one used in [Moro et al., 2014] (it does not include all attributes due to privacy concerns). Using the rminer package and R tool (http://cran.r-project.org/web/packages/rminer/), we found that the addition of the five new social and economic attributes (made available here) lead to substantial improvement in the prediction of a success, even when the duration of the call is not included. Note: the file can be read in R using: d=read.table("bank-additional-full.csv",header=TRUE,sep=";")

Project Justification:

Problem Objective: The classification goal is to predict if the client will subscribe (yes/no) to the term deposit

Complexity Involved: Handling imbalanced data, feature selection, feature engineering, model building and evaluation.

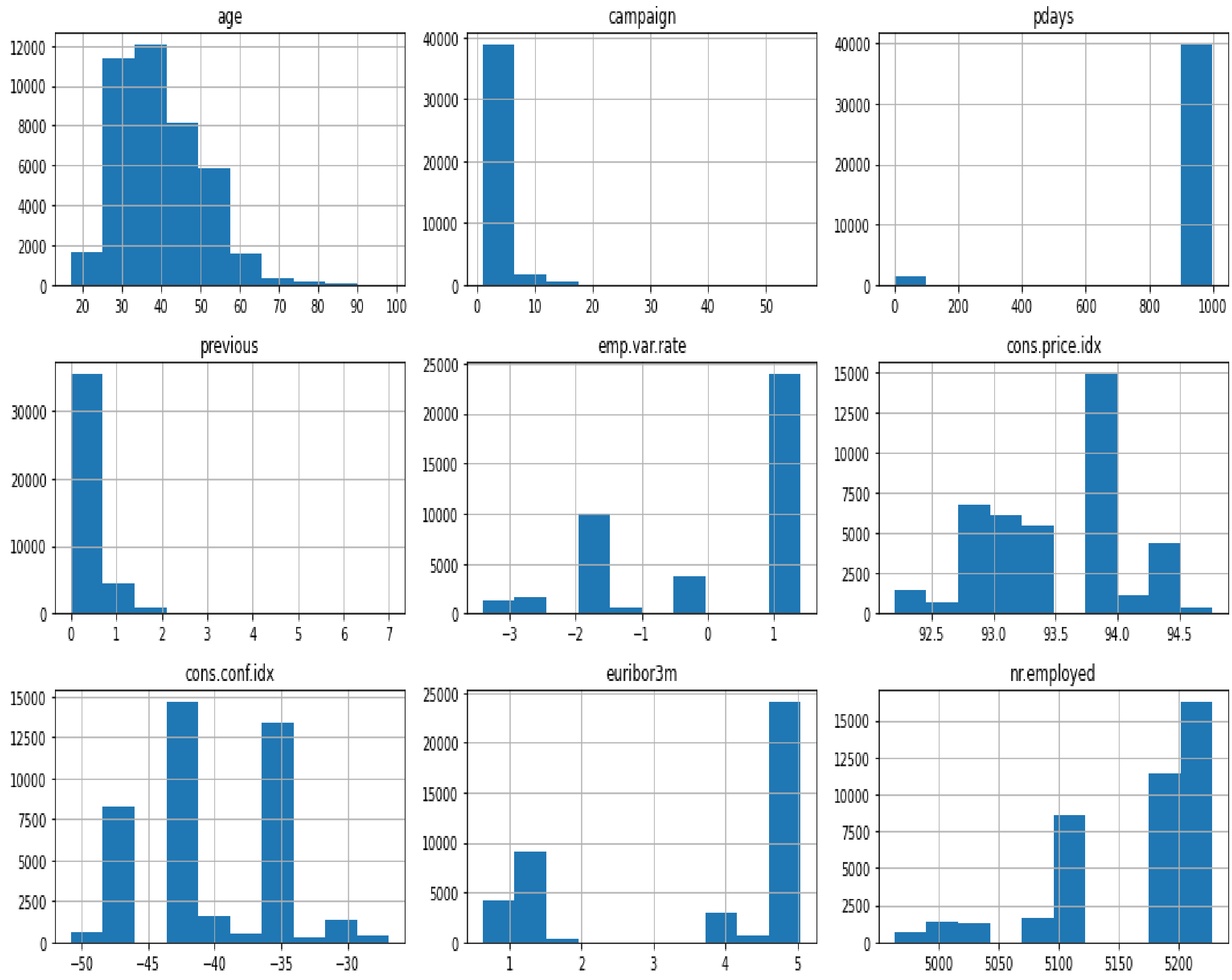
Project Outcome: The outcome will have commercial value by improving campaign effectiveness and customer conversion rates. Increases bank revenue by identifying potential subscribers, thus enabling focused marketing efforts. Utilize machine learning to analyse historical data, leading to data-driven decision-making.

Exploratory Data Analysis:

Univariate Analysis: (numerical variables)

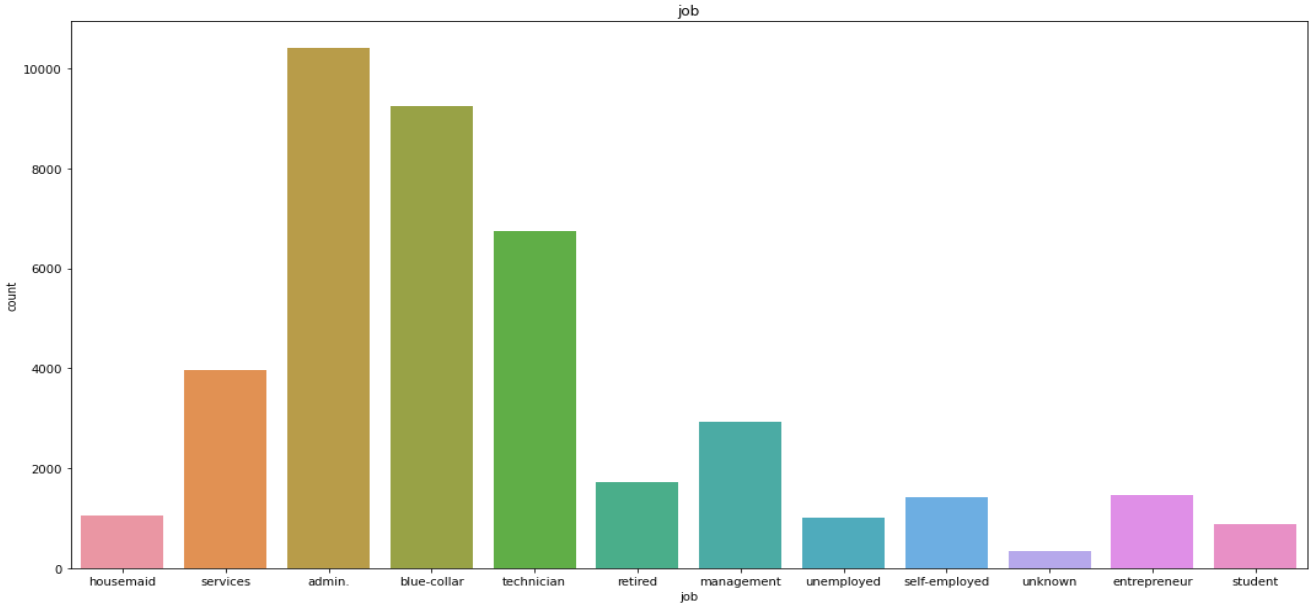
Distribution of numerical variables

We used histplot, for finding the distribution of numerical variables



Univariate Analysis: (Categorical variables)

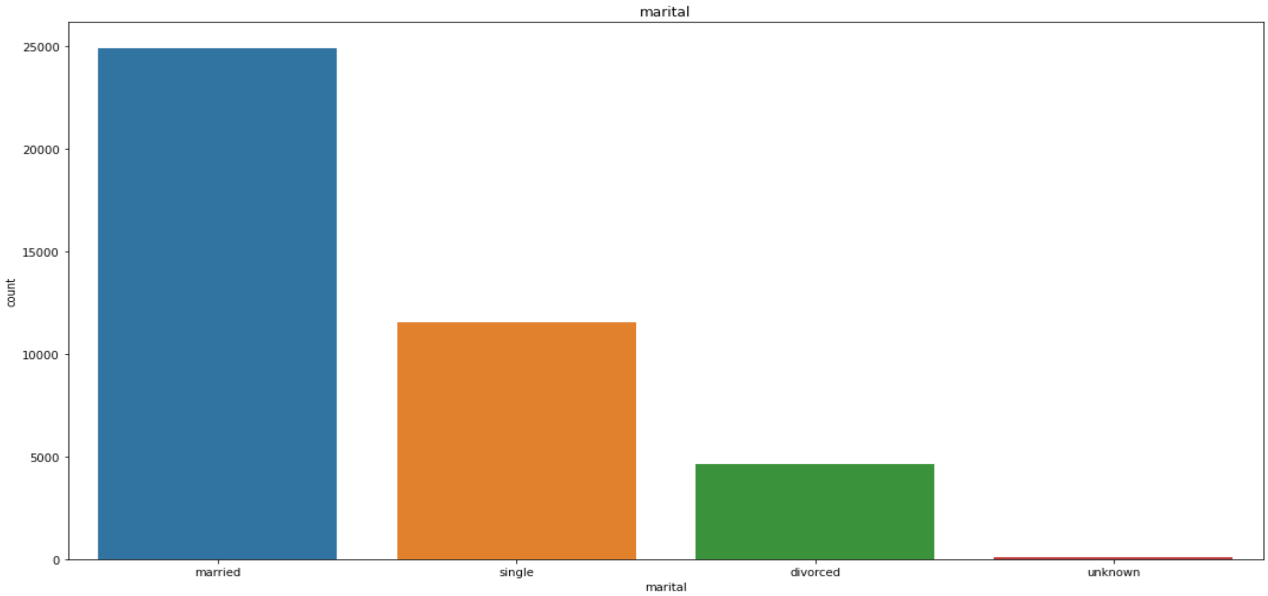
Job:



Inferences:

- Here Job Types, 'Blue-collar' and 'management' professions are the most common among clients
- Tailoring campaigns to the needs and interests of these job types can increase engagement and response rates

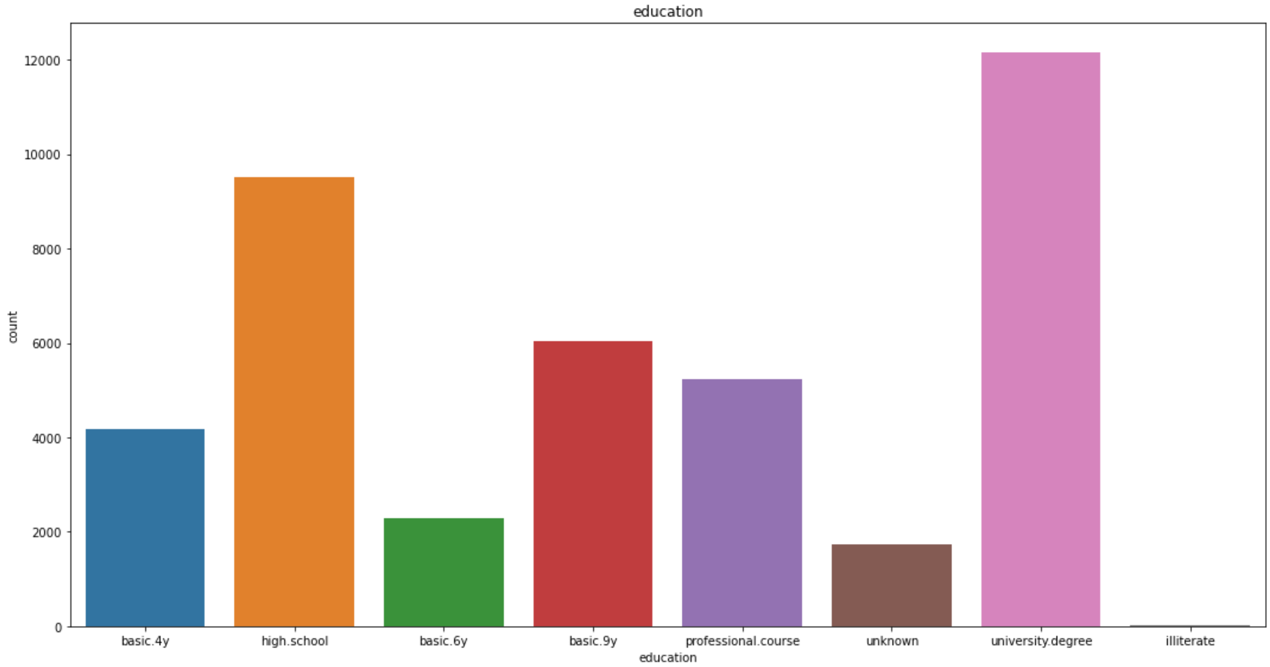
Marital:



Inferences:

- Married clients dominate the dataset
- Designing campaigns that cater to married individuals, such as family-oriented products or services, could enhance effectiveness and appeal

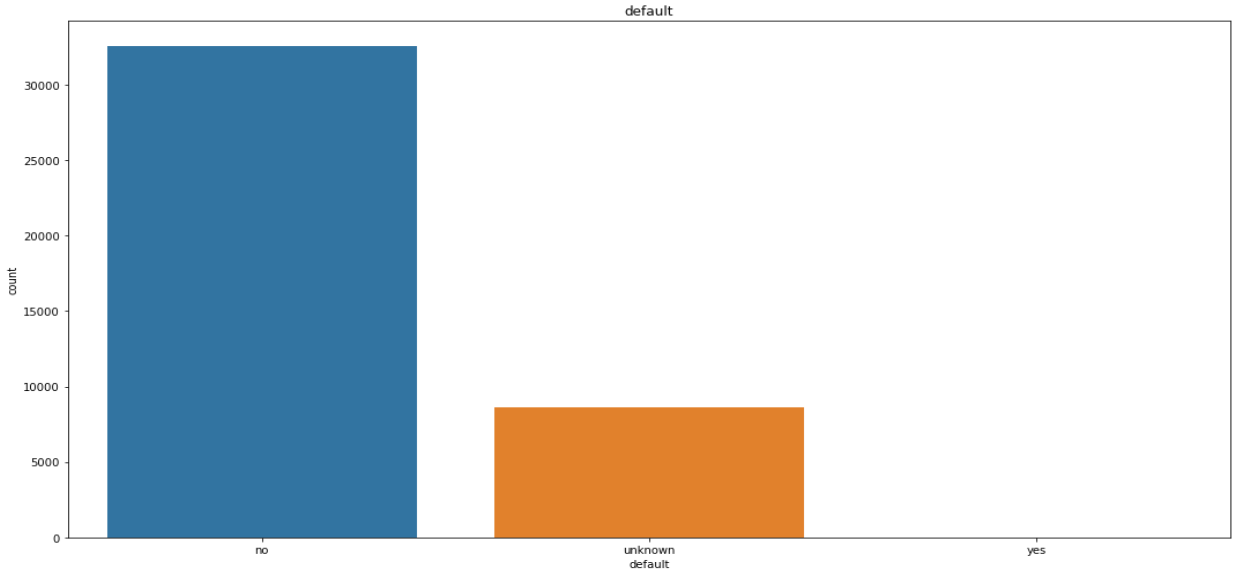
Education:



Inferences:

- Most clients have a 'university. Degree' or 'high. School' education
- Incorporating educational content and emphasizing product benefits aligned with their background can resonate well with these clients

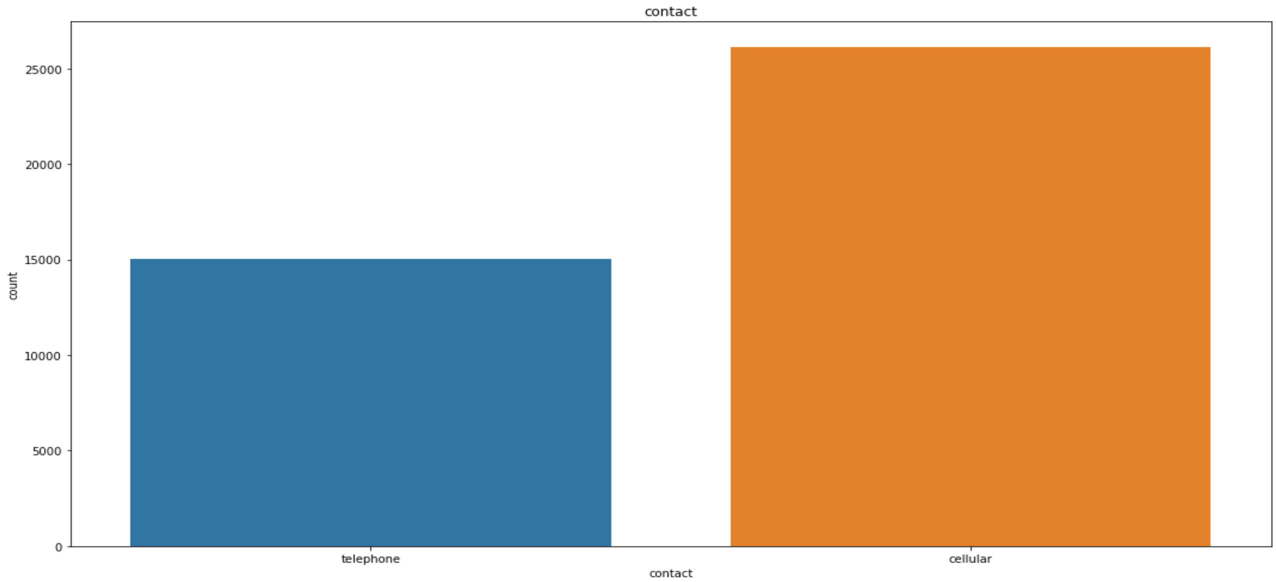
Default:



Inferences:

- The vast majority of clients do not have credit defaults, indicating financial stability
- Marketing strategies should focus on investment opportunities or premium offerings appealing to financially stable individuals

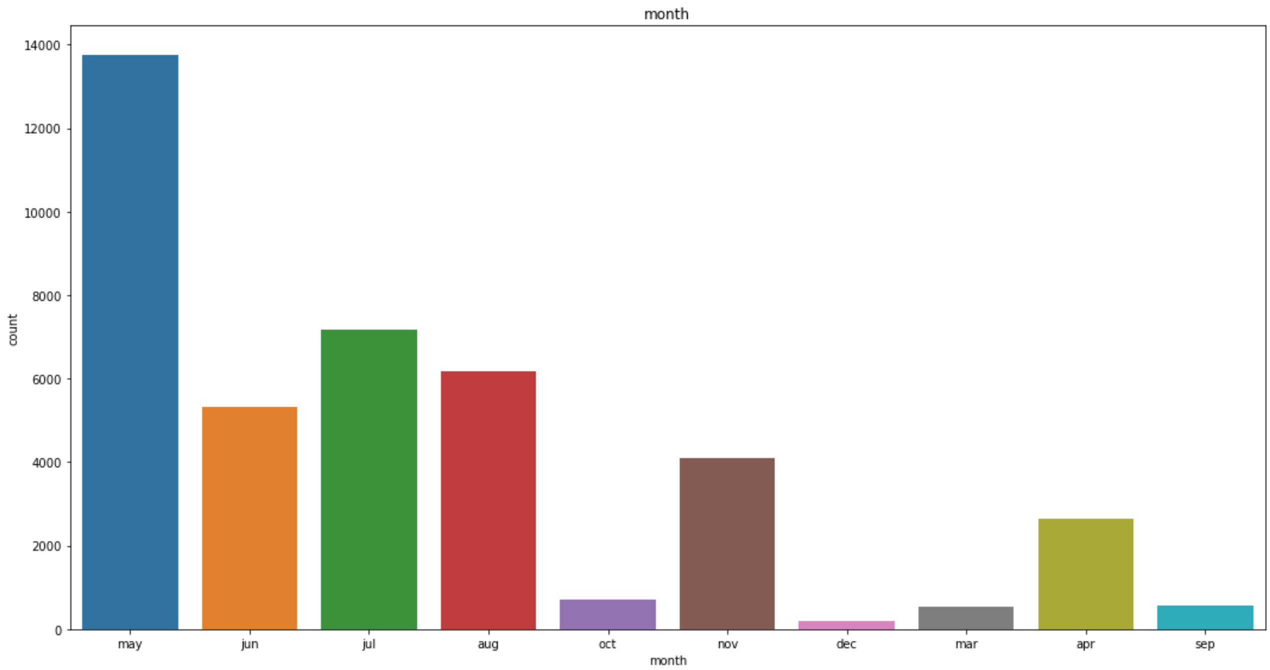
Contact mode:



Inferences:

- The 'cellular' contact method is more prevalent and effective than 'telephone' contact
- Prioritizing mobile outreach can enhance engagement and conversion rates

Month:

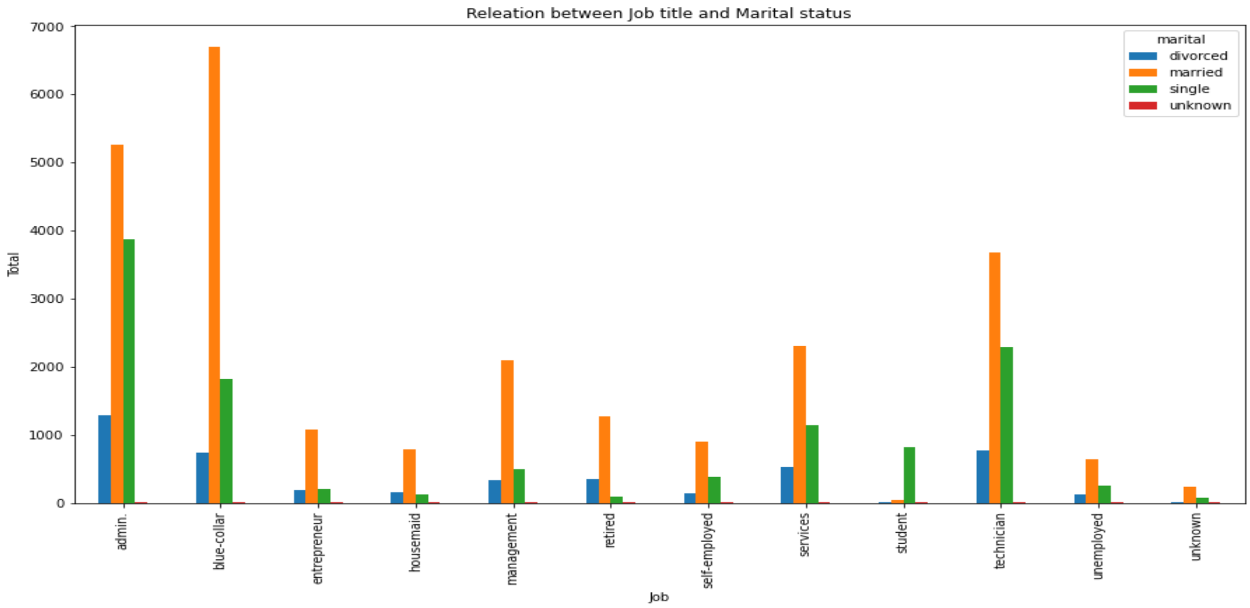


Inferences:

- May has the highest number of contacts, making it strategic for campaign launches
- Concentrating marketing efforts in May can capitalize on increased customer responsiveness

Bivariate Analysis: (cat vs cat)

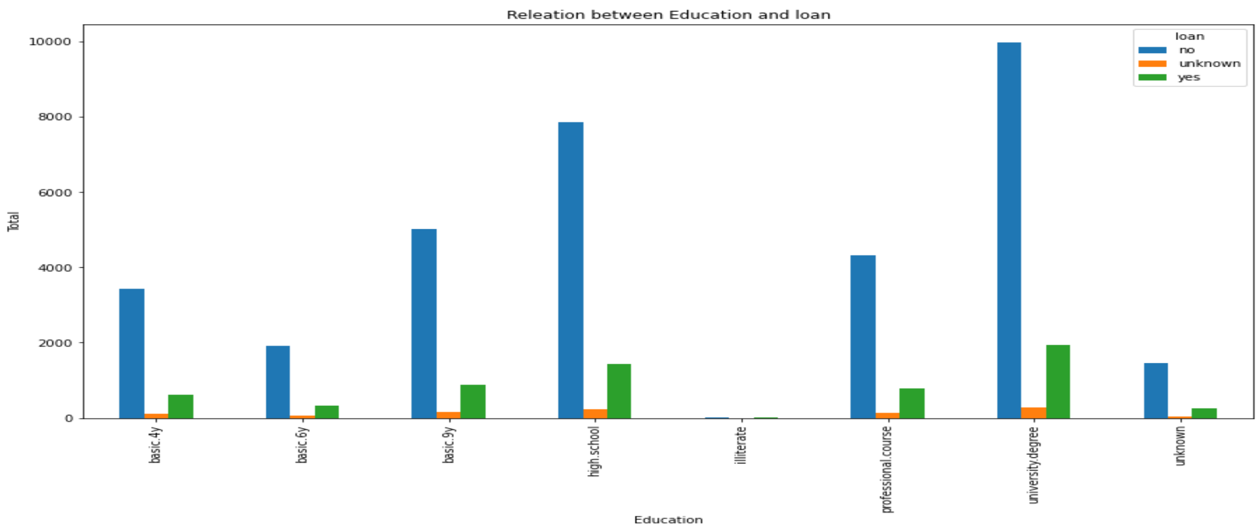
Job vs Marital:



Inferences:

- Married clients are prevalent across most job categories, especially 'blue-collar' and 'management'
- Marketing strategies should consider the dual-income and family-oriented needs of married clients

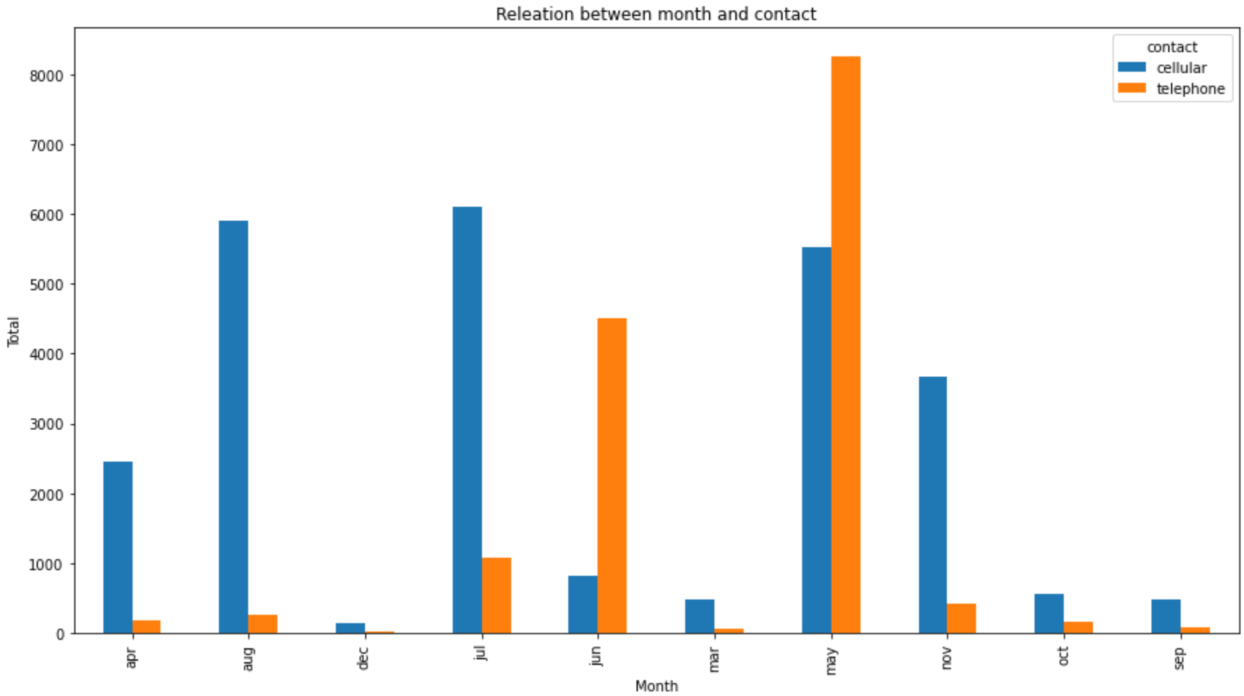
Education vs loan:



Inferences:

- Higher education levels correlate with fewer personal loans, suggesting financially prudent behaviors
- Financial products emphasizing savings and investments might appeal to educated clients

Month vs contact method:



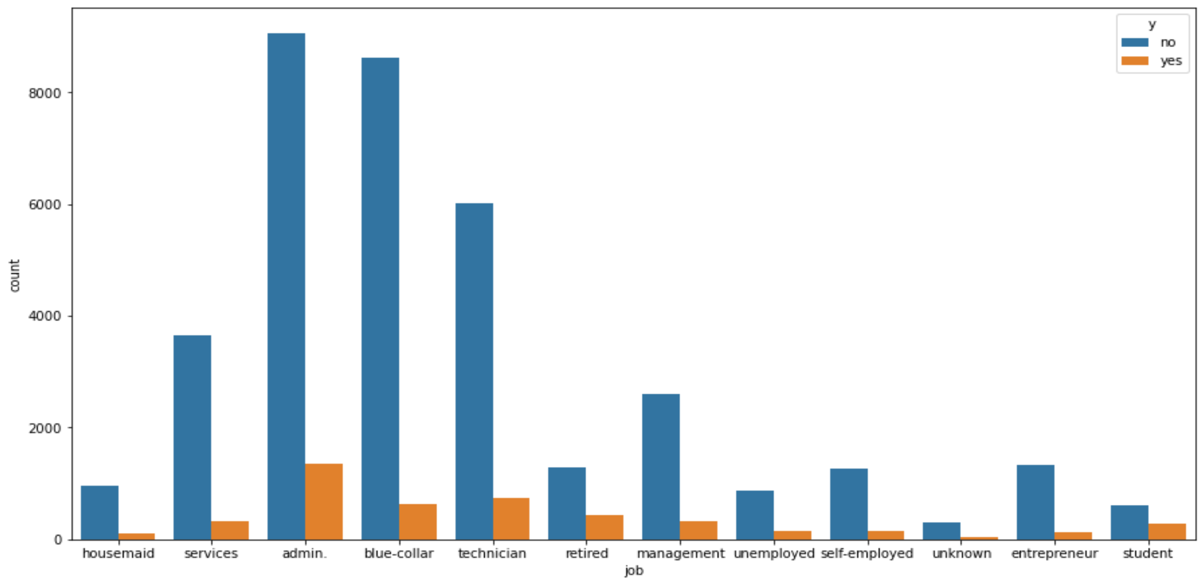
Inferences:

- 'Cellular' contact is more effective than 'telephone' across all months
- Prioritizing mobile outreach can improve campaign reach and effectiveness

Analysis with Target variable:

We have analysed some of the features with target variable. We got some efficient insights from that they are listed below:

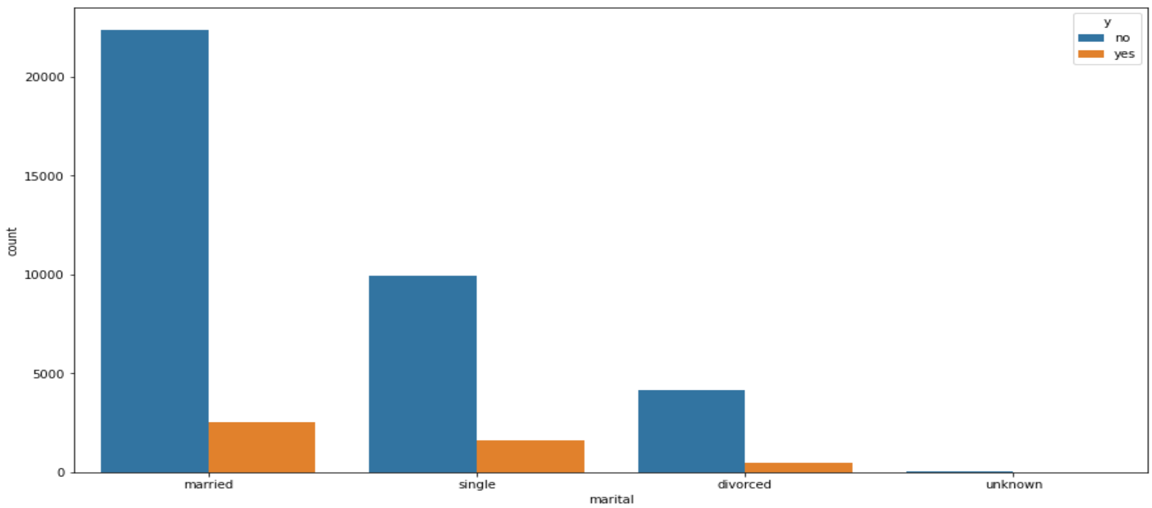
Job vs Target variable:



Inferences:

- Clients in 'admin.' and 'blue-collar' jobs show higher 'no' responses, while 'admin' and 'technicians' clients have more 'yes' responses
- Tailoring messages to address the specific concerns of 'admin.' and 'technicians' workers, while highlighting benefits relevant to 'blue-collar' and 'retired' clients, can optimize campaign success

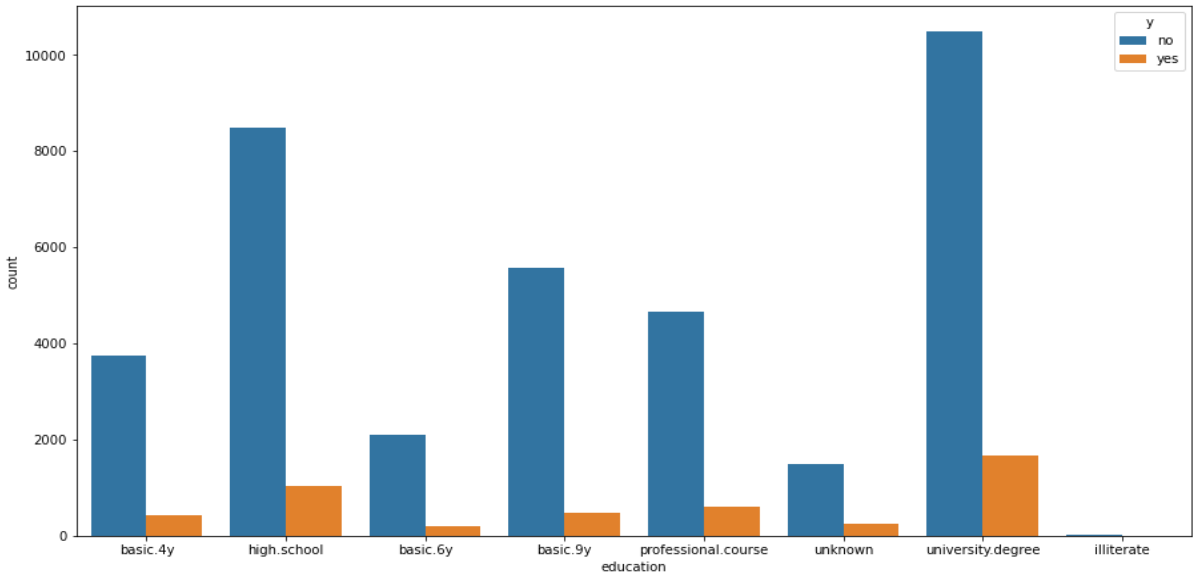
Marital status vs Target variable:



Inferences:

- Married clients have a higher number of 'no' responses, whereas single clients have a relatively higher proportion of 'yes' responses
- Campaigns designed to appeal to single clients' needs and lifestyles might achieve better results

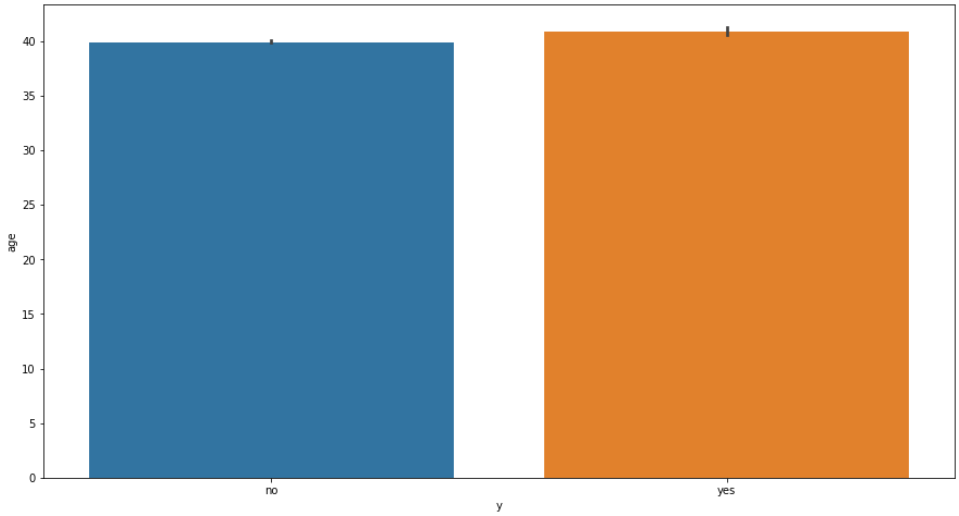
Education vs Target:



Inferences:

- Clients with a 'university.degree' show more 'yes' responses compared to those with only 'basic.4y' education
- Marketing efforts should emphasize educational and professional development benefits to attract this more responsive segment

Age vs Target:



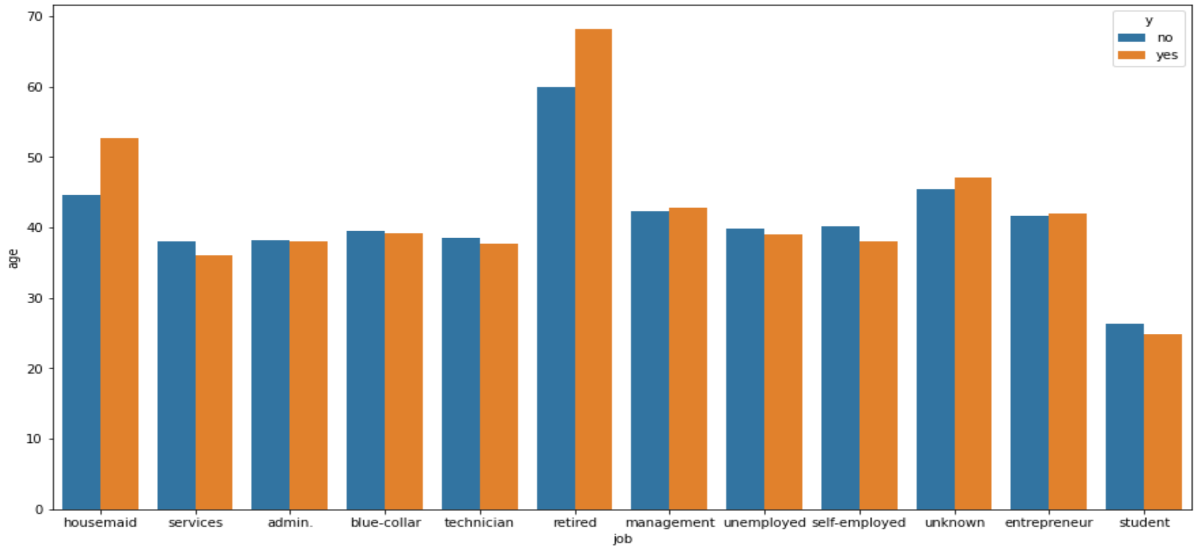
Inferences:

- Younger clients are less likely to subscribe to term deposits compared to older clients
- Targeting marketing strategies towards older demographics, who show higher interest, can increase subscription rates

Multivariate analysis:

Basically, multivariate analysis is analysing more than 2 variables. We have done multivariate analysis with some of the features with target variable. They are:

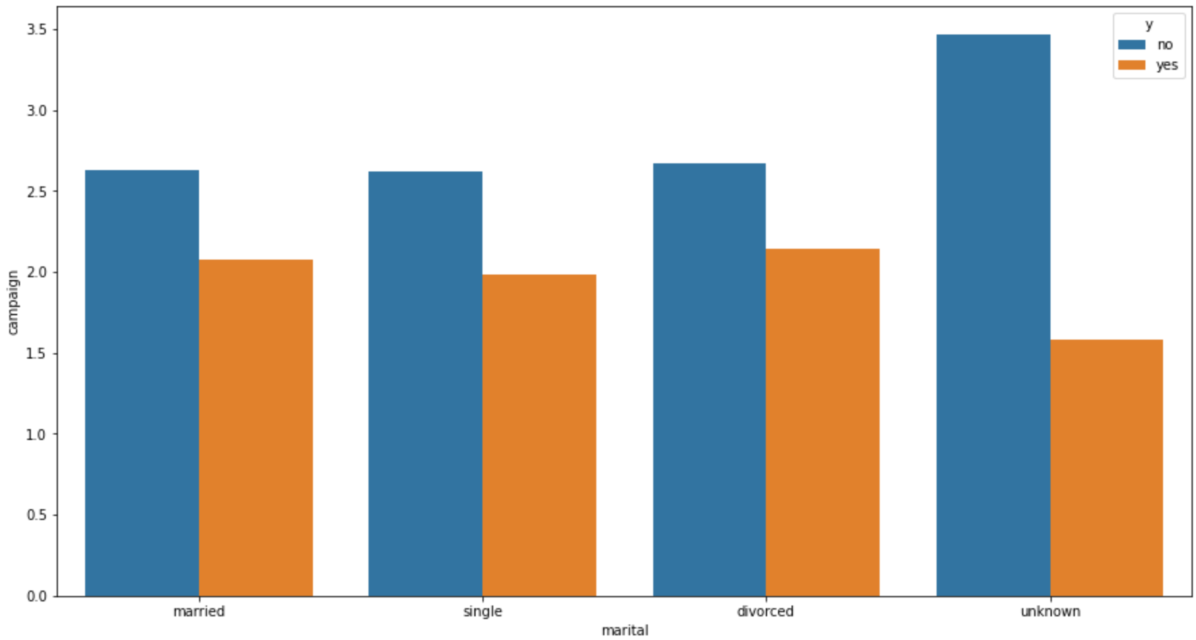
Job vs Age vs Target:



Inferences:

- 'Retired' and 'housemaid' clients, who tend to be older, have higher positive response rates
- Focusing on these segments can increase campaign success

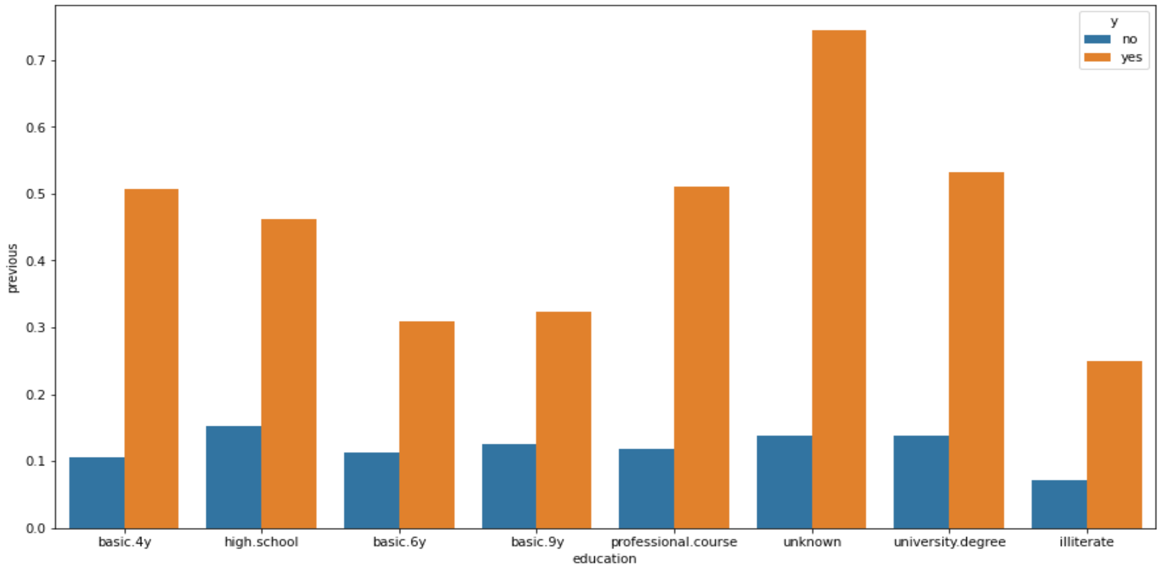
Marital status vs campaign vs Target:



Inferences:

- Single clients require fewer contacts for a positive response
- Strategies targeting single clients could be more cost-effective

Education vs previous contact vs Target:



Inferences:

- Clients with 'high school' show higher positive response rates and more previous contacts
- Prior successful engagement strategies should be replicated for educated clients

Correlation Matrix:

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. The value ranges between -1 and 1.

- 1 implies a perfect positive correlation
- -1 implies a perfect negative correlation
- 0 implies no correlation

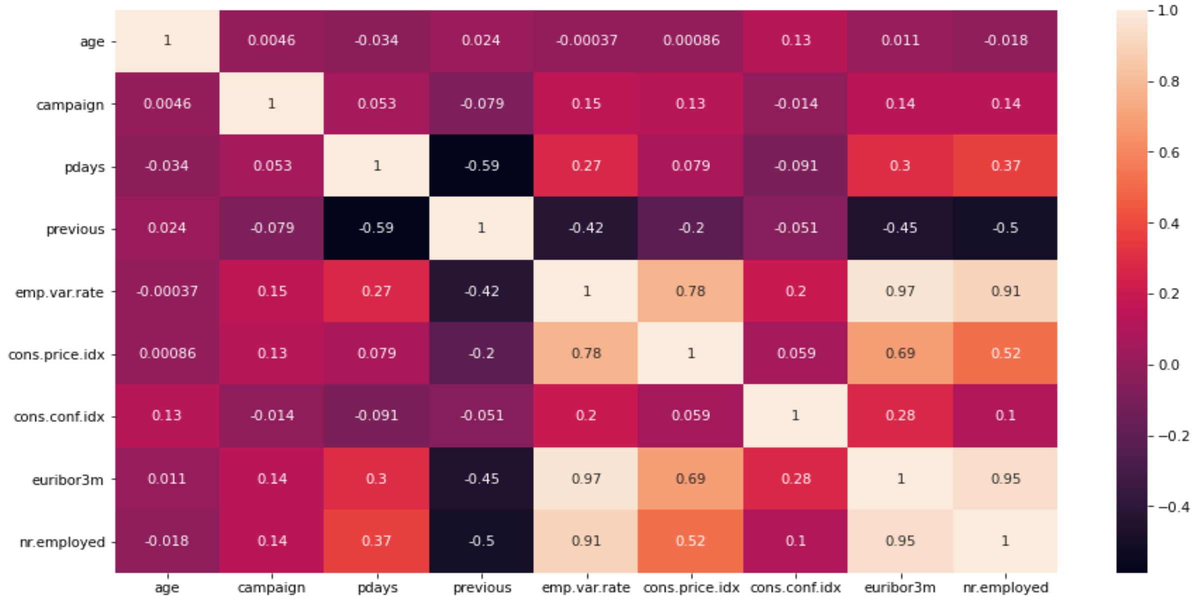
We got correlation matrix from that we noticed some of the variables are strongly correlated in both negative and positive directions.

```
df.select_dtypes(np.number).corr()
```

	age	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
age	1.000000	0.004594	-0.034369	0.024365	-0.000371	0.000857	0.129372	0.010767	-0.017725
campaign	0.004594	1.000000	0.052584	-0.079141	0.150754	0.127836	-0.013733	0.135133	0.144095
pdays	-0.034369	0.052584	1.000000	-0.587514	0.271004	0.078889	-0.091342	0.296899	0.372605
previous	0.024365	-0.079141	-0.587514	1.000000	-0.420489	-0.203130	-0.050936	-0.454494	-0.501333
emp.var.rate	-0.000371	0.150754	0.271004	-0.420489	1.000000	0.775334	0.196041	0.972245	0.906970
cons.price.idx	0.000857	0.127836	0.078889	-0.203130	0.775334	1.000000	0.058986	0.688230	0.522034
cons.conf.idx	0.129372	-0.013733	-0.091342	-0.050936	0.196041	0.058986	1.000000	0.277686	0.100513
euribor3m	0.010767	0.135133	0.296899	-0.454494	0.972245	0.688230	0.277686	1.000000	0.945154
nr.employed	-0.017725	0.144095	0.372605	-0.501333	0.906970	0.522034	0.100513	0.945154	1.000000

We also notice a few strong negative and strong positive correlations.

Graphical Representation: We used heatmap for better understanding how variables are correlated.



Identification of Duplicates:

Duplicate records can cause a model to overfit, meaning it performs very well on the training data but poorly on new, unseen data. The model might learn the noise or redundant patterns instead of the underlying trends. In statistical models, duplicates can skew the estimates of parameters, leading to biased or incorrect conclusions. Duplicates can inflate performance metrics like accuracy, precision, and recall, giving a false impression of the model's effectiveness. It can increase the computational complexity of the model training process, making it slower and more resource-intensive. Duplicates can affect the correlation matrix, as they artificially inflate the correlation coefficients, leading to misinterpretation of the relationships between variables.

Treatments: We removed duplicates from the dataset since it will affect our model.

Checking duplicates

```
df.duplicated().sum()
```

```
2109
```

```
df = df.drop_duplicates()
```

```
df.shape
```

```
(39079, 20)
```

Note: After removing duplicates, we removed around 2000 record

Pdays Modification:

Here We have numeric attribute named pdays means that number of days that passed by after the client was last contacted from a previous campaign (There are 999 and other numeric values inside of this column. 999 means client was not previously contacted. We converted the ‘pdays’ column numeric to categorical which as two classes. If the value is equal to 999, we put the 0 instead of 999, otherwise we put 1. So we changed the pdays column.

```
In [27]:  df['pdays'] = [0 if i == 999 else 1 for i in df['pdays']]
          df.pdays

Out[27]: 0        1
         1        1
         2        1
         3        1
         4        1
                 ..
         41183    1
         41184    1
         41185    1
         41186    1
         41187    1
         Name: pdays, Length: 41188, dtype: int64

In [56]:  df.pdays.value_counts()

Out[56]: 0    37565
         1     1514
         Name: pdays, dtype: int64
```

Statistical significance of variables:

Assessing the statistical significance of variables in a dataset is a crucial step in understanding which variables have a meaningful impact on the target variable. Statistical significance is typically determined through hypothesis testing, which helps you understand whether the observed relationships in your data occurred by chance or reflect actual relationships in the population.

Pearson Correlation coefficient:

The Pearson correlation coefficient is a measure of the linear relationship between two variables. It ranges from -1 to 1, where:

- 1 indicates a perfect positive linear relationship,
- -1 indicates a perfect negative linear relationship,
- 0 indicates no linear relationship.

```python
import pandas as pd
import numpy as np
from scipy.stats import pearsonr
import statsmodels.api as sm


# Pearson Correlation for numerical feature vs numerical target

for i in list(df_num.columns):
    pearson_corr, p_value = pearsonr(df_num[i], df['Term_Deposit_en'])
    print(f"Pearson Correlation for {i}: {pearson_corr}, p-value: {p_value}")

    if p_value < 0.05 :
        print(f'{i} is significantly related to Target variable')
    else :
        print('Not significantly related to Target variable')
    print('')

# Null hypothesis - numerical feature is not significantly correlated with the numerical target.
# Alternative hypothesis - numerical feature is significantly correlated with the numerical target.

# A high correlation coefficient and a low p-value (typically < 0.05)
# suggest that the numerical feature is significantly correlated with the numerical target.
```

```
Pearson Correlation for age: 0.028049085291230482, p-value: 2.92660627735226e-08
age is significantly related to Target variable

Pearson Correlation for campaign: -0.07354603251440603, p-value: 5.185469310382746e-48
campaign is significantly related to Target variable

Pearson Correlation for pdays: 0.3247905744255473, p-value: 0.0
pdays is significantly related to Target variable

Pearson Correlation for previous: 0.22920900401436461, p-value: 0.0
previous is significantly related to Target variable

Pearson Correlation for emp.var.rate: -0.2978822933278055, p-value: 0.0
emp.var.rate is significantly related to Target variable

Pearson Correlation for cons.price.idx: -0.13683890105561797, p-value: 1.184260702622688e-162
cons.price.idx is significantly related to Target variable

Pearson Correlation for cons.conf.idx: 0.05818855652357087, p-value: 1.1433520976591828e-30
cons.conf.idx is significantly related to Target variable

Pearson Correlation for euribor3m: -0.3076909973482342, p-value: 0.0
euribor3m is significantly related to Target variable

Pearson Correlation for nr.employed: -0.3540505572527318, p-value: 0.0
nr.employed is significantly related to Target variable
```

We have done Pearson correlation test which is having null and alternative hypothesis as:

Null hypothesis - numerical feature is not significantly correlated with the numerical target.

Alternative hypothesis - numerical feature is significantly correlated with the numerical target.

From this test some of the features are significantly related to target variable they are age, campaign, pdays etc.

ANOVA: Analysis of Variance

Analysis of Variance (ANOVA) is a statistical method used to test the differences between the means of three or more groups. It helps to determine if at least one of the group means is statistically different from the others. ANOVA is useful when comparing multiple groups to see if they have different effects on a dependent variable.

```python
# ANOVA for categorical feature vs numerical target
from scipy.stats import pearsonr, chi2_contingency, f_oneway

for i in list(df_cat.columns):
    groups = [df[df[i] == category]['Term_Deposit_en'] for category in df_cat[i].unique()]
    anova_result = f_oneway(*groups)
    print(f"ANOVA for {i} : F-statistic: {anova_result.statistic}, p-value: {anova_result.pvalue}", sep='  ')
    if anova_result.pvalue < 0.05 :
        print('Reject null - means are different')
    else :
        print('Means are same')
    print('')

# Null Hypothesis - means of the target variable are same across all categories of the categorical feature.
# Alternative Hypothesis - means of the target variable are different across the categories of the categorical feature.

# A significant F-statistic and a low p-value indicate that
# the means of the target variable are significantly different across the categories of the categorical feature.
```

```
ANOVA for job : F-statistic: 92.90579658256864, p-value: 7.413366918260916e-191
Reject null - means are different

ANOVA for marital : F-statistic: 59.699090351547255, p-value: 1.2958192283654688e-26
Reject null - means are different

ANOVA for education : F-statistic: 32.28352561167793, p-value: 5.225884741410034e-39
Reject null - means are different

ANOVA for default : F-statistic: 0.3997649286853823, p-value: 0.5272143816364695
Means are same

ANOVA for housing : F-statistic: 5.759892057394752, p-value: 0.016400728342722028
Reject null - means are different

ANOVA for loan : F-statistic: 3.4337198022983078, p-value: 0.063885606832147
Means are same

ANOVA for contact : F-statistic: 929.147196000253, p-value: 1.052212050964353e-201
Reject null - means are different

ANOVA for month : F-statistic: 348.2741332368062, p-value: 0.0
Reject null - means are different

ANOVA for day_of_week : F-statistic: 7.263664262520789, p-value: 7.655977077850013e-06
Reject null - means are different

ANOVA for poutcome : F-statistic: 2231.6474340967216, p-value: 0.0
Reject null - means are different
```

We have done anova for numerical variable vs categorical variable with more than 2 classes. And we found some of the categories with more than 2 classes are having different means.

Chi- Square Test for Independence:

The Chi-Square Test for Independence is a statistical test used to determine if there is a significant association between two categorical variables. It tests the null hypothesis that the variables are independent.

```python
# Chi-Square contingency Test for categorical feature vs categorical target
for i in list(df_cat.columns):
    contingency_table = pd.crosstab(df[i], data['y'])
    chi2, p, dof, ex = chi2_contingency(contingency_table)
    print(f"Chi-Square for {i}: Chi2: {chi2}, p-value: {p}")
    if p < 0.05 :
        print('We reject null - Features are not associated ')
    else :
        print('Features are associated')
    print('')

# Null Hypothesis - proportions of Features are same
# Alternative Hypothesis - proportions of Features are different

# A significant Chi-Square statistic and a low p-value suggest a significant association between the categorical feature
# and the categorical target.

```

```
Chi-Square for job: Chi2: 907.7331707127846, p-value: 1.3790626203754682e-188
We reject null - Features are not associated

Chi-Square for marital: Chi2: 119.04360512386378, p-value: 1.4125687506963004e-26
We reject null - Features are not associated

Chi-Square for education: Chi2: 192.7801404754759, p-value: 6.521830896219545e-39
We reject null - Features are not associated

Chi-Square for default: Chi2: 0.06966652094052214, p-value: 0.791822945110442
Features are associated

Chi-Square for housing: Chi2: 5.683891954681657, p-value: 0.017121347927528913
We reject null - Features are not associated

Chi-Square for loan: Chi2: 3.354230303965751, p-value: 0.06703254095034286
Features are associated

Chi-Square for contact: Chi2: 906.6361419605003, p-value: 3.5416779713316803e-199
We reject null - Features are not associated

Chi-Square for month: Chi2: 2902.411858648708, p-value: 0.0
We reject null - Features are not associated

Chi-Square for day_of_week: Chi2: 29.036783770944787, p-value: 7.68403320528975e-06
We reject null - Features are not associated

Chi-Square for poutcome: Chi2: 4006.0616664132167, p-value: 0.0
We reject null - Features are not associated
```

The Chi-Square Test for Independence is a useful tool for determining if there is a significant relationship between two categorical variables. By comparing observed and expected frequencies, you can infer whether the variables are associated or independent. This test is widely used in fields like marketing, research, and social sciences to analyze survey data and experimental results. We found that some of the independent features which are categorical are not associated that much.

Inferences from Statistical Test: After performing ANOVA and Chi-Square Contingency test we found features **Loan and Default** as non-significant. As a result, we will be building models without these features to improve the accuracy and F1 score.

Class Imbalance:

Class imbalance occurs when the classes in a classification problem are not represented equally. This can be problematic because machine learning models might become biased towards the majority class, leading to poor performance on the minority class.

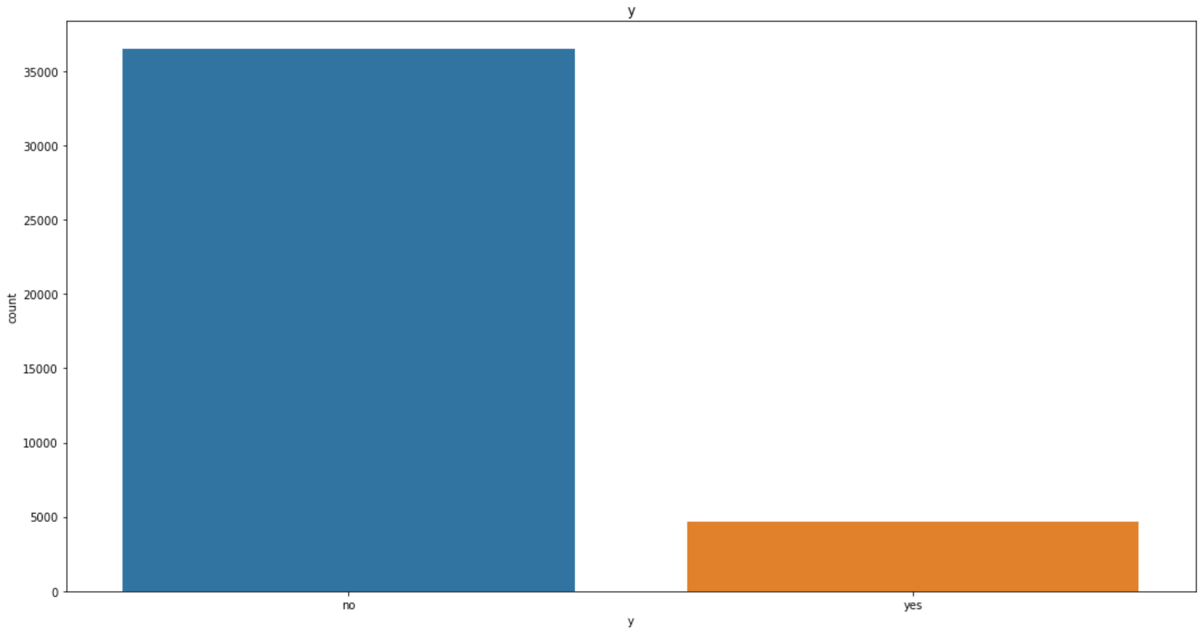
Issues Caused by Class Imbalance:

Bias Towards Majority Class: The model may perform well on the majority class but poorly on the minority class.

Misleading Accuracy: High accuracy might be misleading as the model could simply be predicting the majority class most of the time.

Poor Generalization: The model might not generalize well to unseen data, particularly for the minority class.

From our data we found that target variable is strongly imbalanced. No class is majority and yes class is minority.



Methods to Handle Class Imbalance:

Resampling Techniques:

Oversampling the Minority Class: Increase the number of instances in the minority class by duplicating them.

Under sampling the Majority Class: Reduce the number of instances in the majority class by randomly removing them.

Synthetic Data Generation (SMOTE): Create synthetic instances of the minority class.

Base Model Building

Encoding:

Encoding is the process of converting categorical variables into numerical values that can be used by machine learning algorithms. Different encoding techniques are used depending on the nature of the categorical data and the requirements of the model.

Common Encoding Techniques

Label Encoding

Label Encoding assigns a unique integer to each category. This method is straightforward but can introduce an ordinal relationship where none exists.

One-Hot Encoding

One-Hot Encoding creates a new binary column for each category, which has a value of 1 for the presence of the category and 0 otherwise. This method avoids the ordinal relationship issue.

Ordinal Encoding

Ordinal Encoding assigns ordinal values to categories based on a specific order. This is useful when the categories have a meaningful ranking.

Frequency Encoding

Frequency Encoding replaces categories with their frequency of occurrence. This method can be useful for high cardinality features.

Target Encoding

Target Encoding replaces categories with the mean of the target variable for each category. This is often used in categorical features with high cardinality.

Choosing the right encoding technique is crucial for the performance of machine learning models. While label encoding is simple and useful for ordinal data, one-hot encoding is generally preferred for nominal data. More advanced techniques like binary encoding, ordinal encoding, frequency encoding, and target encoding can be advantageous depending on the dataset and the specific problem. Proper encoding ensures that categorical variables are effectively utilized, leading to better model accuracy and performance.

We have used one-hot encoding technique for our categorical variables.

```
encoding_df= df.drop('Term_Deposit_en', axis=1)
encoding_df.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	poutcome	emp.var.rate	cons.pric
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	1	0	0	nonexistent	1.100000	93.99
1	57	services	married	high.school	no	no	no	telephone	may	mon	1	0	0	nonexistent	1.100000	93.99
2	37	services	married	high.school	no	yes	no	telephone	may	mon	1	0	0	nonexistent	1.100000	93.99
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	1	0	0	nonexistent	1.100000	93.99
4	56	services	married	high.school	no	no	yes	telephone	may	mon	1	0	0	nonexistent	1.100000	93.99

```
columns=encoding_df.select_dtypes(include=object).columns
df_en= pd.concat([encoding_df,pd.get_dummies(encoding_df[columns], drop_first=True)],axis=1)
```

```
df_en.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	campaign	pdays	previous	poutcome	emp.var.rate	cons.pric
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	1	0	0	nonexistent	1.100000	93.99
1	57	services	married	high.school	no	no	no	telephone	may	mon	1	0	0	nonexistent	1.100000	93.99
2	37	services	married	high.school	no	yes	no	telephone	may	mon	1	0	0	nonexistent	1.100000	93.99
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	1	0	0	nonexistent	1.100000	93.99
4	56	services	married	high.school	no	no	yes	telephone	may	mon	1	0	0	nonexistent	1.100000	93.99

Splitting the data:

Splitting data is a fundamental step in machine learning where you divide your dataset into training and testing sets. This allows you to train your model on one subset (training set) and evaluate its performance on another subset (testing set). Here's how you can split your data using Python:

Splitting Data Using train_test_split from sklearn.model_selection

The train_test_split function from sklearn.model_selection is commonly used for splitting datasets. It shuffles the data and splits it into two or more parts according to the specified proportions.

Here we divided the data into 2 parts which training parts is 80% of data and testing part is 20% of data.

Splitting the data

```
x=df_final.drop('Target', axis=1)
y= df_final['Target']
```

```
xtrain,xtest,ytrain,ytest= train_test_split(x,y, test_size=0.2, random_state=100)
```

We have splitted the whole dat into 80% for train and 20% for testing

```
xtrain.shape
```

```
(31263, 46)
```

```
ytrain.shape
```

```
(31263,)
```

```
xtest.shape
```

```
(7816, 46)
```

```
ytest.shape
```

```
(7816,)
```

Splitting your data correctly ensures that your model is trained on a diverse set of data and evaluated on unseen data, which helps in assessing its generalization performance.

As we are using all classification techniques, KNN model is a distance based model for which scaling is required. So we have also scaled the data and then again performed train test split.

Base Model Algorithm:

Based on the problem statement and our dataset it’s classification problem. We built base model using logistic regression algorithm.

Logistic Regression:

Logistic regression is a statistical model used for binary classification tasks, where the target variable (dependent variable) is categorical with two possible outcomes, often coded as 0 and 1. It's a type of regression analysis that estimates the probability of the outcome based on one or more predictor variables (independent variables).

```
lr= LogisticRegression(random_state=100)
lr.fit(xtrain,ytrain)
train_pred=lr.predict(xtrain)
test_pred=lr.predict(xtest)
```

Base model we have built using logistic regression

Evaluation Metric:

We used Classification Report for both training and testing data. As our target variable is imbalanced, we can't rely on accuracy so we are relying on f1_score and also plotted the confusion matrix.

Training data report:

From classification report we found out that training accuracy is 0.87

Training data report

```
print(classification_report(ytrain,train_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94     27597
           1       0.59      0.21      0.31      3666

    accuracy                           0.89     31263
   macro avg       0.75      0.59      0.62     31263
weighted avg       0.87      0.89      0.87     31263
```

Testing data report:

From classification report we found out that testing accuracy is also 0.87

Testing data report

```
print(classification_report(ytest,test_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94      6887
           1       0.61      0.21      0.32       929

    accuracy                           0.89      7816
   macro avg       0.76      0.60      0.63      7816
weighted avg       0.87      0.89      0.87      7816
```

Confusion Matrix:

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It allows visualization of the performance of an algorithm and provides insights into how well the model is performing in terms of predicting each class.

Components of a Confusion Matrix:

In a binary classification problem, the confusion matrix consists of four main metrics:

True Positive (TP): The number of correctly predicted positive instances (actual positive and predicted positive).

True Negative (TN): The number of correctly predicted negative instances (actual negative and predicted negative).

False Positive (FP): Type I error - the number of incorrectly predicted positive instances (actual negative but predicted positive).

False Negative (FN): Type II error - the number of incorrectly predicted negative instances (actual positive but predicted negative).

Interpretation:

True Positive (TP): The model correctly predicted the positive class.

True Negative (TN): The model correctly predicted the negative class.

False Positive (FP): The model incorrectly predicted the positive class (Type I error).

False Negative (FN): The model incorrectly predicted the negative class (Type II error).

We have calculated number of records in each segment and plotted as matrix. So, we checked the values of confusion matrix to see which records are under True positive, False positive, False negative and True Negative.

```python
confusion_matrix(ytest,test_pred)
```

```
array([[6761,  126],
       [ 731,  198]], dtype=int64)
```

```python
from matplotlib.colors import ListedColormap
```

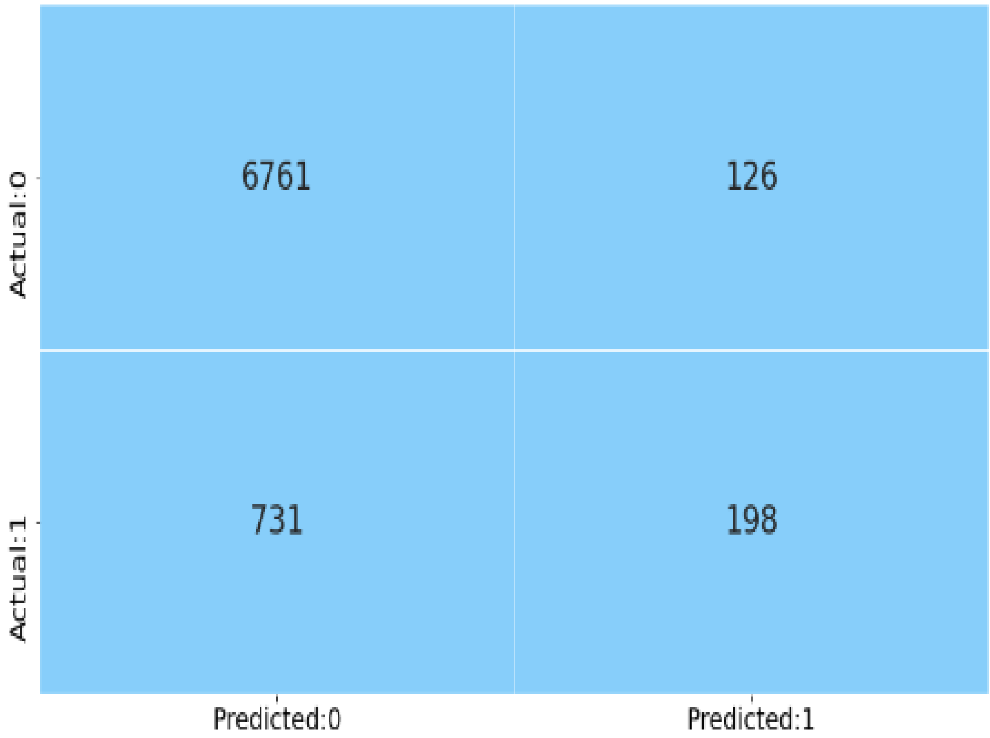
```python
# create a confusion matrix
# pass the actual and predicted target values to the confusion_matrix()
cm = confusion_matrix(ytest,test_pred)
conf_matrix = pd.DataFrame(data = cm,columns = ['Predicted:0','Predicted:1'], index = ['Actual:0','Actual:1'])

# plot a heatmap to visualize the confusion matrix
# 'annot' prints the value of each grid
# 'fmt = d' returns the integer value in each grid
# 'cmap' assigns color to each grid
# as we do not require different colors for each grid in the heatmap,
# use 'ListedColormap' to assign the specified color to the grid
# 'cbar = False' will not return the color bar to the right side of the heatmap
# 'linewidths' assigns the width to the line that divides each grid
# 'annot_kws = {'size':25})' assigns the font size of the annotated text
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = ListedColormap(['lightskyblue']), cbar = False,
            linewidths = 0.1, annot_kws = {'size':25})

# set the font size of x-axis ticks using 'fontsize'
plt.xticks(fontsize = 20)

# set the font size of y-axis ticks using 'fontsize'
plt.yticks(fontsize = 20)

# display the plot
plt.show()
```



we can say that TN and FN records are high from confusion matrix TP and and FP counts are low

Model Selection:

List of Algorithms: Logistic Regression, Decision Trees, Random Forests, Bagging Classifier, Ada-Boost, Gradient Boost, XG Boost.

We performed basic model building from this list of models and also performed their hyperparameter tuning. And finally, we got 13 models. The summary of every model is shown below.

	Model	Accuracy	Recall	Precision	F1 Score	AUC_Score
0	LogisticReg-Base	0.890993	0.221744	0.614925	0.325949	0.601506
1	LogisticReg-Scaled	0.622825	0.688913	0.193998	0.302744	0.651412
2	Decision Tree-Gini	0.823951	0.328310	0.288553	0.307150	0.609559
3	Decision Tree-Entropy	0.831372	0.348762	0.312440	0.329603	0.622617
4	Decision Tree-Tuned	0.895343	0.278794	0.636364	0.387725	0.628652
5	Random Forest	0.884084	0.284177	0.522772	0.368201	0.624592
6	Random Forest-Tuned	0.894575	0.182992	0.723404	0.292096	0.586777
7	Bagging Classifier-dt	0.878454	0.270183	0.479924	0.345730	0.615344
8	Bagging Classifier-knn	0.886387	0.264801	0.545455	0.356522	0.617517
9	Bagging Classifier-Log	0.891377	0.199139	0.637931	0.303527	0.591946
10	AdaBoost-dt	0.852738	0.286329	0.352785	0.316102	0.607736
11	Gradient Boosting	0.897262	0.278794	0.660714	0.392127	0.629741
12	XGB	0.896238	0.232508	0.687898	0.347546	0.609139

After analyzing the accuracy value, f1 score and AUC score of all models we found two models **"Decision Tree Tuned "and "Gradient Boosting" model** to have the best values.

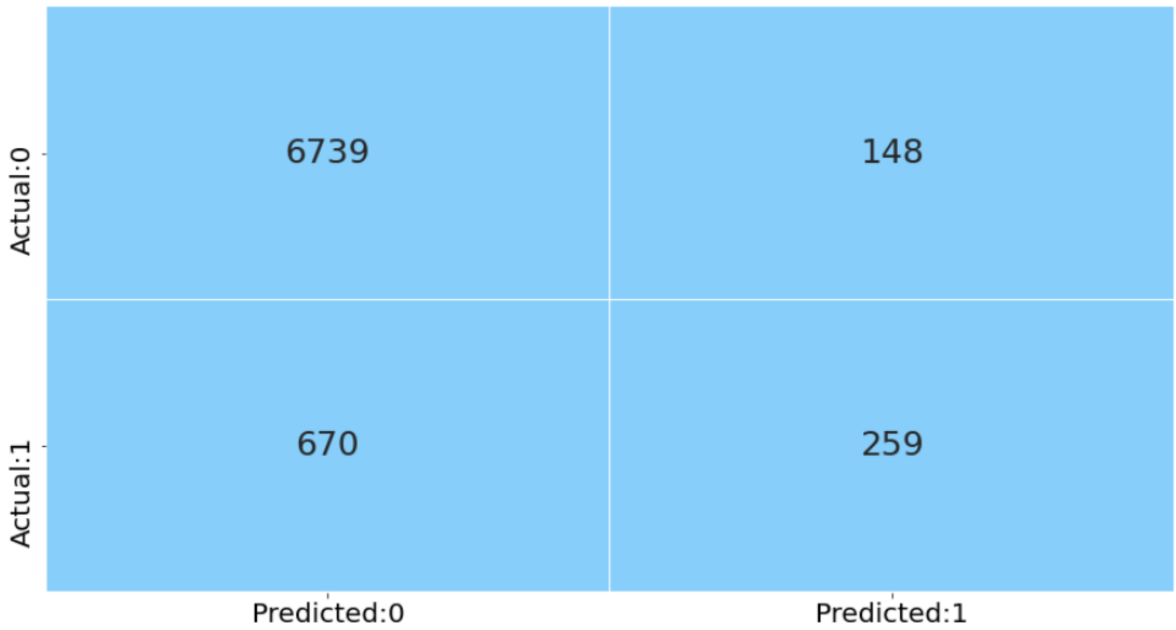
As mentioned, we also performed model building after feature selection i.e., after removing Loan and Default features to check and achieve higher accuracy and F1-score. The table shown below contains the results.

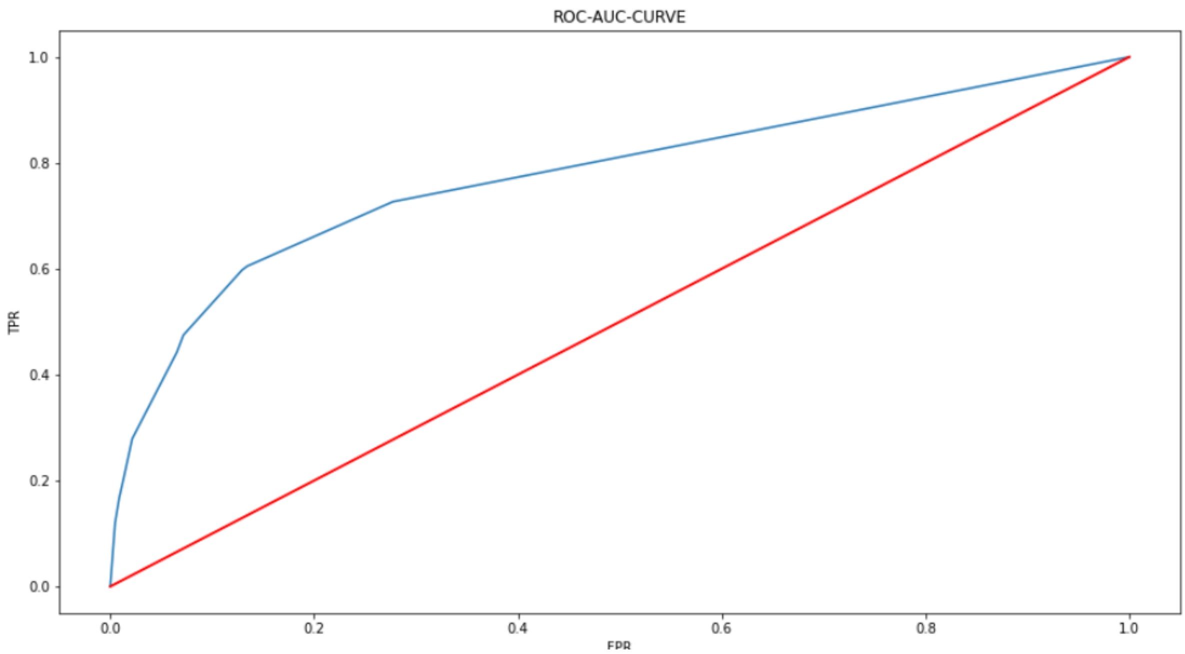
	Model	Accuracy	Recall	Precision	F1 Score	AUC_Score
0	Logistic-feature_select	0.890993	0.221744	0.614925	0.325949	0.601506
1	Decision Tree-Gini_select	0.823951	0.328310	0.288553	0.307150	0.609559
2	Decision Tree-Entropy_select	0.827533	0.328310	0.296404	0.311542	0.611592
3	Decision Tree-Tuned_select	0.892528	0.274489	0.605701	0.377778	0.625193
4	Random Forest_select	0.885107	0.286329	0.530938	0.372028	0.626104
5	Random Forest-Tuned_select	0.894191	0.179763	0.719828	0.287683	0.585163

From this approach we can see that there is no significant change in the values of accuracy and F1 Score.

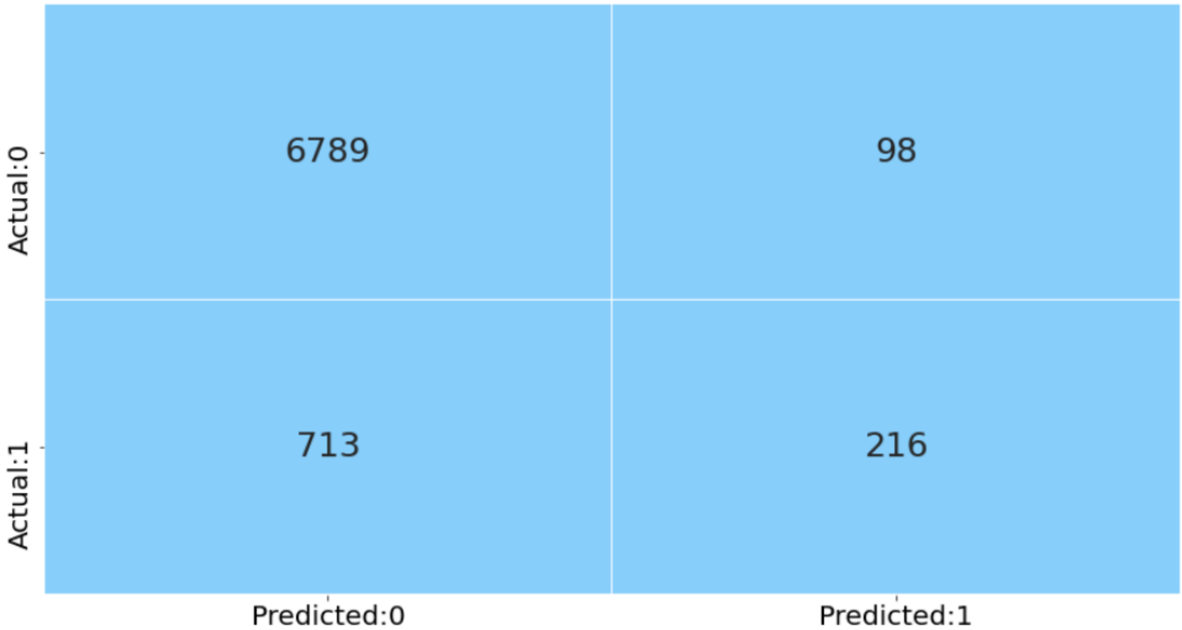
Here are the plots of confusion matrix and ROC curve for the selected best models.

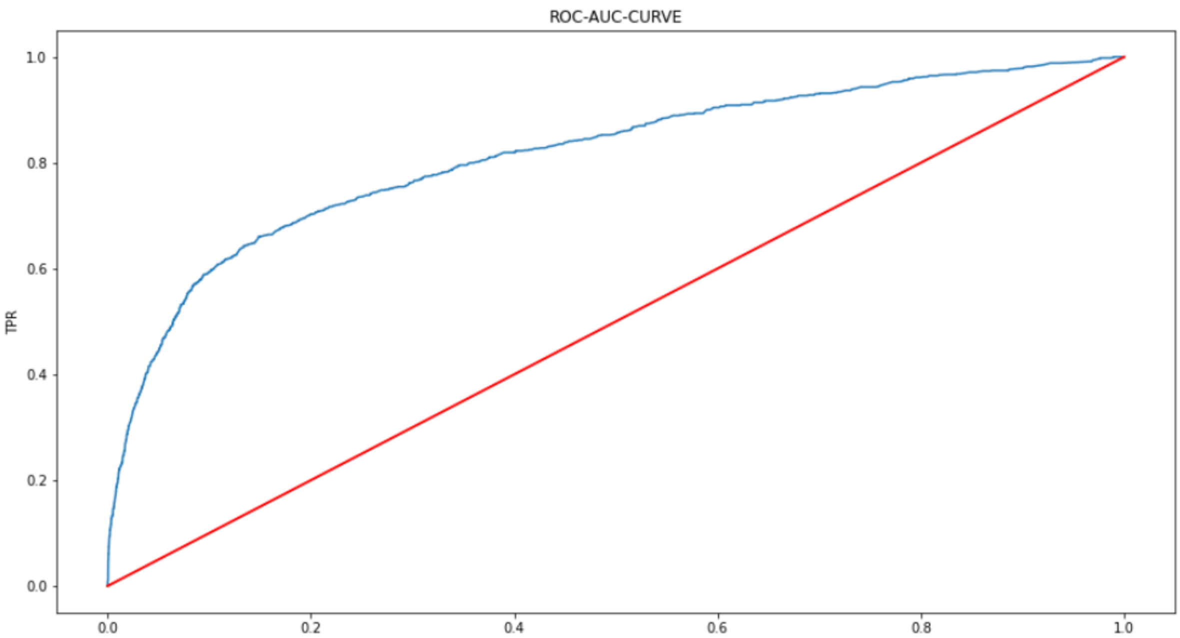
Decision Tree Tuned





Gradient Boosting





Conclusion and Outcome:

The primary objective of this project was to predict if a client would subscribe to a term deposit, utilizing historical data to inform future marketing strategies and improve campaign effectiveness. The key challenges included selecting relevant features, engineering new features, building robust models, and evaluating their performance.

Outcome and Commercial Value:

- The project successfully identified potential subscribers, allowing for more focused and effective marketing efforts.
- By improving customer conversion rates, the project is expected to increase bank revenue and enhance the efficiency of marketing campaigns.
- The insights and models developed from this project enable data-driven decision-making, optimizing future campaigns and improving customer engagement strategies.

Overall, the project achieved its objective by leveraging machine learning techniques to analyze historical data and predict client subscription to term deposits, thereby providing significant commercial value to the bank.