Higher National Diploma in Data Science.

Machine Learning.

Credit Card Fraud Detection

COHNDDS231F

05/30/2023

Nethmi Sarandi (COHNDDS23.1F-008)

ACKNOWLEDGEMENT

I want to express my sincere gratitude to Mr.Chameera De Silva, who served as my lecturer, for his insightful advice and encouragement in helping me finish my project.

Also, I want to thank my family and friends for helping me out by giving me access to the resources and information I needed to complete my assignment.

Finally, I'd like to thank all of my colleagues and everyone else who helped make this study a success.

Table of Contents

Table of Figures

Introduction

Bank term deposits, referred to by the terms fixed deposits or certifications of deposit, are a common choice for investments for both consumers and businesses, and they play an important part in the worldwide financial environment. By providing an interest rate that is fixed for a set length of time, term deposits provide a secure and reliable means to grow savings, making them a desirable option for those who are wary of risk.

This study report’s goal is to look at the nuances of bank term deposit subscriptions and the variables that lead people and companies to choose this type of investment. This paper intends to offer beneficial insights for financial institutions, investors and policymakers by looking at the present trends, customer habits and market conditions around bank term deposit subscriptions.

Membership in bank term deposits has grown in importance among both people and businesses as a financial option. Depending on a number of variables, such as the rate of interest, market conditions, and personal goals in life, this form of investment might have distinct characteristics. Financial institutions that want to draw in new clients and hold onto current ones have to recognize these factors. Bank term deposits allow investors an easy way to make consistent profits on their investments. However, it is essential to take into account how inflation may affect the future value of these investments. The price spectrum for bank term deposits is shaped in part by policymakers' policies and incentives that support the banking sector's stability and expansion.

As a result, when making options concerning bank short-term deposit subscriptions, it's critical to be educated regarding current trends and client behavior. Businesses, financiers, and politicians can all profit from the possibilities offered by this kind of financial investment by doing this.

Table for Variables and Their Definitions

Variables	Definitions
Age	Table for Variables, Their Definitions
Job	The occupation or job category of the individual
Marital	The marital status of the individual, such as married, single, divorced, or in a registered partnership.
Education	The educational background or level of education of the individual, such as primary school, secondary school, tertiary education, or unknown.
Default	This binary attribute indicates whether the individual has credit in default (yes) or not (no).
Balance	The current balance (in euros) of the individual's bank account.
Housing	This binary attribute indicates whether the individual has a housing loan (yes) or not (no).
Loan	This binary attribute indicates whether the individual has a personal loan (yes) or not (no).
Contact	This describes the communication type used to contact the individual, either cellular (cellular) or by telephone (telephone).
Day	This represents the day of the month on which the individual was last contacted.
Month	This indicates the month in which the individual was last contacted.
Duration	This represents the duration (in seconds) of the last contact between the individual and the marketing team.
Campaign	This represents the number of contacts made during the current marketing campaign for this individual.
Pdays	This represents the number of days that passed by after the individual was last contacted from a previous campaign (999 means the individual was not previously contacted).
Previous	This represents the number of contacts made before the current marketing campaign for this individual.

Poutcome	This describes the outcome of the previous marketing campaign, which can be success, failure, nonexistent, or unknown.
Target	This is the target variable or the outcome variable. It indicates whether the individual subscribed to a term deposit (yes) or not (no) as a result of the marketing campaign.

Code Explanation

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import os
sb.set()
```

Figure 1: Importing libraries

Mounting Google Drive

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

Figure 2: Mounting google drive

Load data set

```
data=pd.read_csv("/content/drive/MyDrive/Machine learning/bank-full.csv")
```

Figure 3: Loading the data set

Head

```
data.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	Target
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

Figure 4: Printing first columns

View data set details

Shape

We have 45211 rows and 17 columns in our banking dataset.

```
data.shape
(45211, 17)
```

Figure 5: Data set shape

Data types of columns

There are 7 columns being of integer type and 10 columns being of object type.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  Target     45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

Figure 6: Data types

Find missing values

There are no null values in the data set.

```
data.isnull().sum()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
Target       0
dtype: int64
```

Figure 7: Find missing values

Descriptive summary in data set

```
data.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

Figure 8: Descriptive summary

Check unique values

```python
# printing unique value of categorical data
for col in data.columns:
    if data[col].dtype == 'object':
        unique_values = pd.unique(data[col])
        print(f'{col} : {unique_values}\n')
```

```
job : ['management' 'technician' 'entrepreneur' 'blue-collar' 'unknown'
 'retired' 'admin.' 'services' 'self-employed' 'unemployed' 'housemaid'
 'student']

marital : ['married' 'single' 'divorced']

education : ['tertiary' 'secondary' 'unknown' 'primary']

default : ['no' 'yes']

housing : ['yes' 'no']

loan : ['no' 'yes']

contact : ['unknown' 'cellular' 'telephone']

month : ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'jan' 'feb' 'mar' 'apr' 'sep']

poutcome : ['unknown' 'failure' 'other' 'success']

Target : ['no' 'yes']
```

Figure 9: Check unique values

Correlation analysis

```python
Correlation =data.corr()
Correlation
```

```
<ipython-input-8-d505ede0ce3d>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid col
  Correlation =data.corr()
```

	age	balance	day	duration	campaign	pdays	previous
age	1.000000	0.097783	-0.009120	-0.004648	0.004760	-0.023758	0.001288
balance	0.097783	1.000000	0.004503	0.021560	-0.014578	0.003435	0.016674
day	-0.009120	0.004503	1.000000	-0.030206	0.162490	-0.093044	-0.051710
duration	-0.004648	0.021560	-0.030206	1.000000	-0.084570	-0.001565	0.001203
campaign	0.004760	-0.014578	0.162490	-0.084570	1.000000	-0.088628	-0.032855
pdays	-0.023758	0.003435	-0.093044	-0.001565	-0.088628	1.000000	0.454820
previous	0.001288	0.016674	-0.051710	0.001203	-0.032855	0.454820	1.000000

Figure 10: Correlation analysis

```
sb.heatmap(Correlation,xticklabels=Correlation.columns , yticklabels=Correlation.columns , annot=True)
```

```
<Axes: >
```

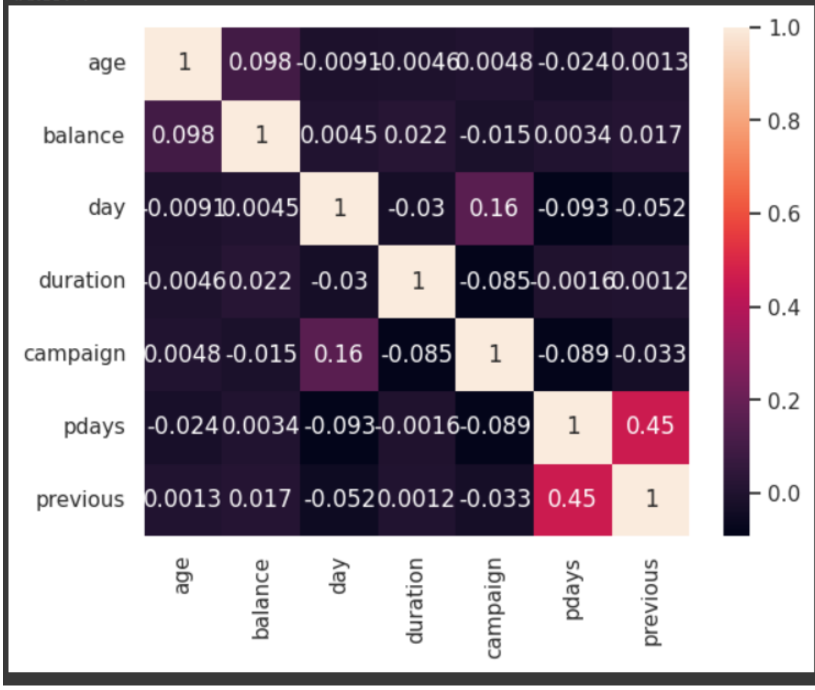

Figure 11: Heat map

Descriptive summary of age, balance and duration

01. Age

```
data["age"].describe()

count    45211.000000
mean        40.936210
std         10.618762
min         18.000000
25%         33.000000
50%         39.000000
75%         48.000000
max         95.000000
Name: age, dtype: float64
```

Figure 12: Descriptive summary of age

02. Balance

```
data["balance"].describe()
```

```
count     45211.000000
mean       1362.272058
std        3044.765829
min       -8019.000000
25%          72.000000
50%         448.000000
75%        1428.000000
max      102127.000000
Name: balance, dtype: float64
```

Figure 13: Descriptive summary of balance

03.Duration

```
data["duration"].describe()
```

```
count    45211.000000
mean       258.163080
std        257.527812
min          0.000000
25%        103.000000
50%        180.000000
75%        319.000000
max       4918.000000
Name: duration, dtype: float64
```

Figure 14: Descriptive summary of duration

Visualize the dataset

The count of each type of jobs

```
#the count of each type of jobs
job_count = data['job'].value_counts()
job_count
```

```
blue-collar      9732
management       9458
technician       7597
admin.           5171
services         4154
retired          2264
self-employed    1579
entrepreneur     1487
unemployed       1303
housemaid        1240
student           938
unknown           288
Name: job, dtype: int64
```

Figure 15: Value count of jobs

Count vs job plot

```
plt.figure(figsize = (8, 5))
job_count.plot(kind = "bar")
plt.title("Type of Jobs")
```

```
Text(0.5, 1.0, 'Type of Jobs')
```

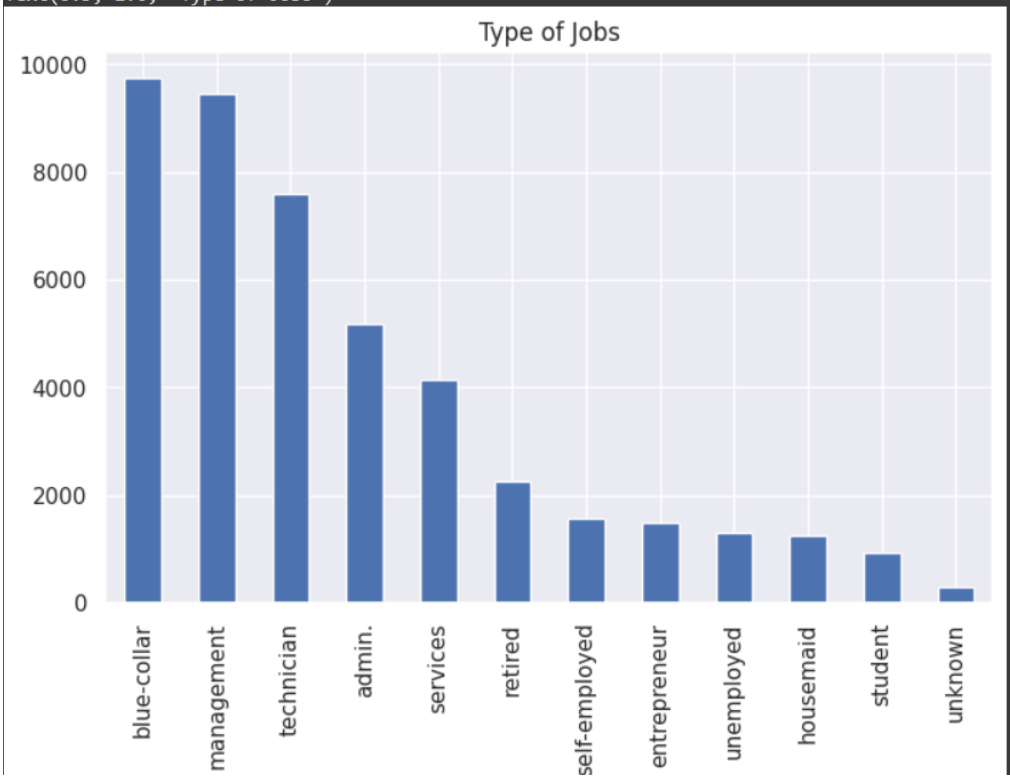

Figure 16: Count vs job plot

The count of default

```
#the count of default
default_count = data['default'].value_counts()
default_count
```

```
no     44396
yes      815
Name: default, dtype: int64
```

Figure 17: Value count of default

Count vs default

```
#count vs default plot
plt.figure(figsize = (8, 5))
default_count.plot(kind='bar').set(title='Default')
```

```
[Text(0.5, 1.0, 'Default')]
```

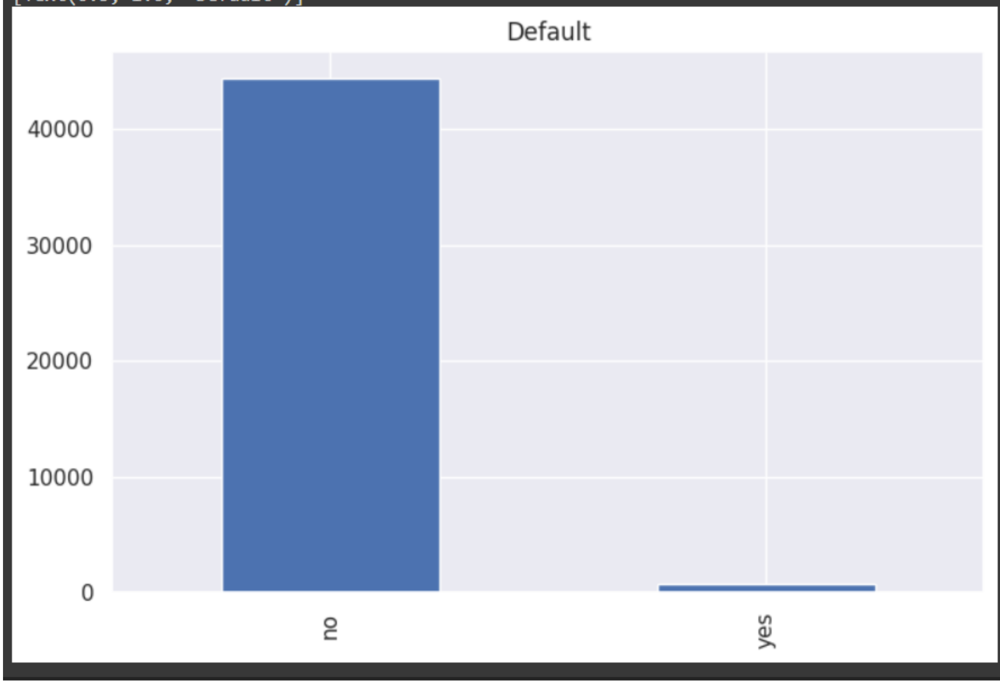

Figure 18: Count vs default plot

The count of martial

```
#the count of martial
marital_count = data['marital'].value_counts()
marital_count
```

```
married     27214
single      12790
divorced     5207
Name: marital, dtype: int64
```

Figure 19: Value count of martial

Martial vs Count

```
#count vs martial plot
plt.figure(figsize = (8, 5))
marital_count.plot(kind = "bar").set(title = "Mertial")
```

```
[Text(0.5, 1.0, 'Mertial')]
```

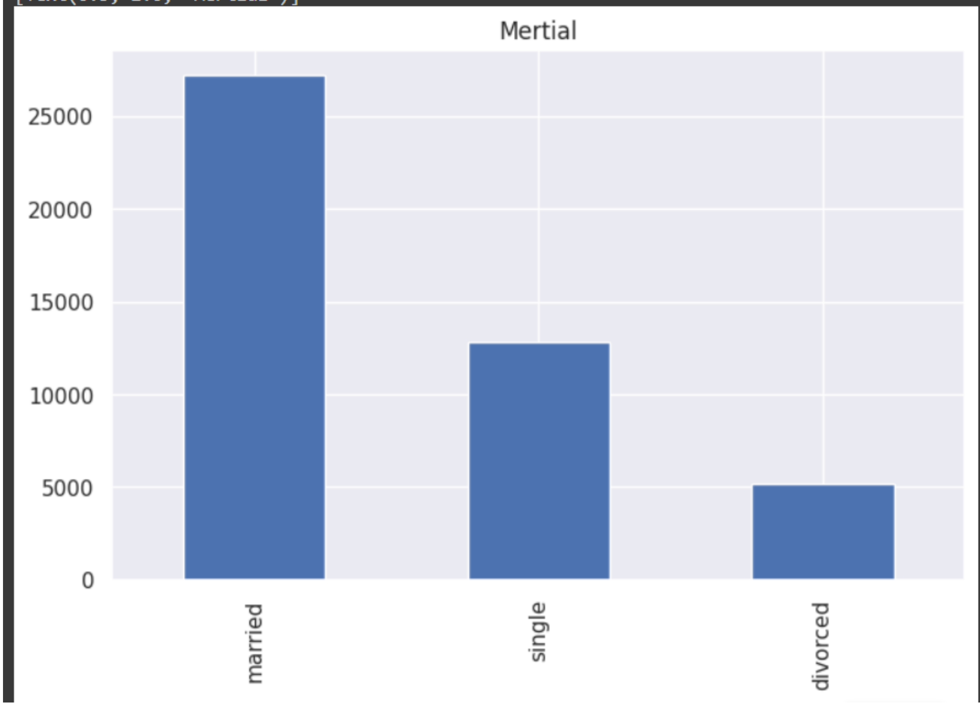

Figure 20: Count vs martial plot

The count customer has personal loan or not

```
#the count customer has personal loan or not
loan_count = data['loan'].value_counts()
loan_count
```

```
no     37967
yes     7244
Name: loan, dtype: int64
```

Figure 21: Value count of customer has personal loan or not

Plot customer has a personal loan or not

```
#Plot customer has a personal loan or not
plt.figure(figsize = (8, 5))
loan_count.plot(kind = "bar").set(title = "Loan")
```

```
[Text(0.5, 1.0, 'Loan')]
```

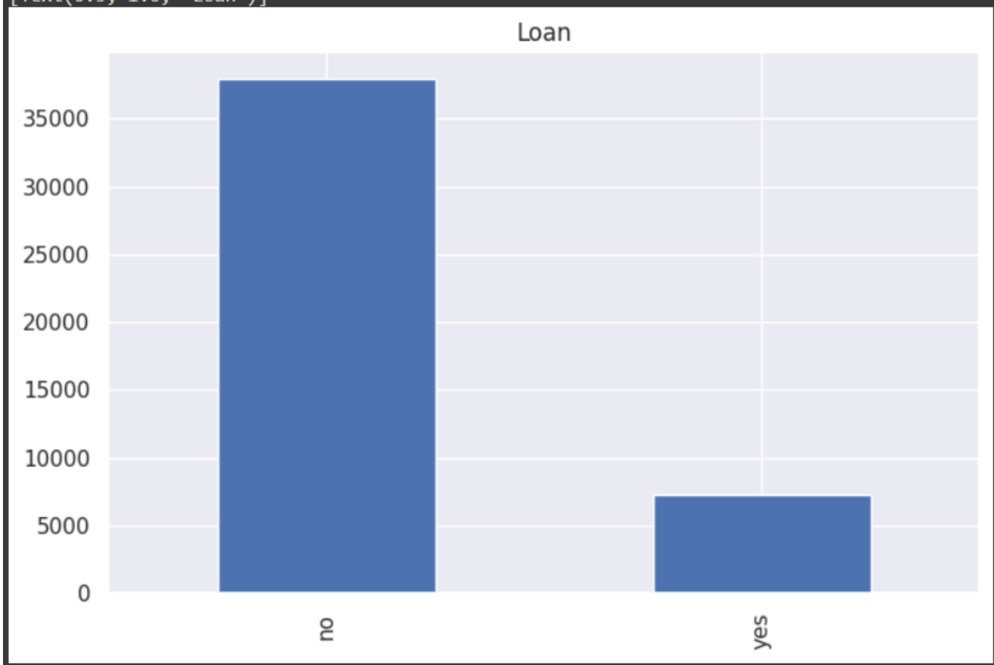

Figure 22: Count vs customer has a personal loan or not plot

The count client has housing loan or not

```
#the count lient has housing loan or no
housing_count = data['housing'].value_counts()
housing_count
```

```
yes    25130
no     20081
Name: housing, dtype: int64
```

Figure 23: Value count of client has housing loan or not

Client has housing loan or not plot

```
#client has housing loan or not plot
plt.figure(figsize = (8, 5))
housing_count.plot(kind = "bar").set(title = "Housing Loan")
```

```
[Text(0.5, 1.0, 'Housing Loan')]
```

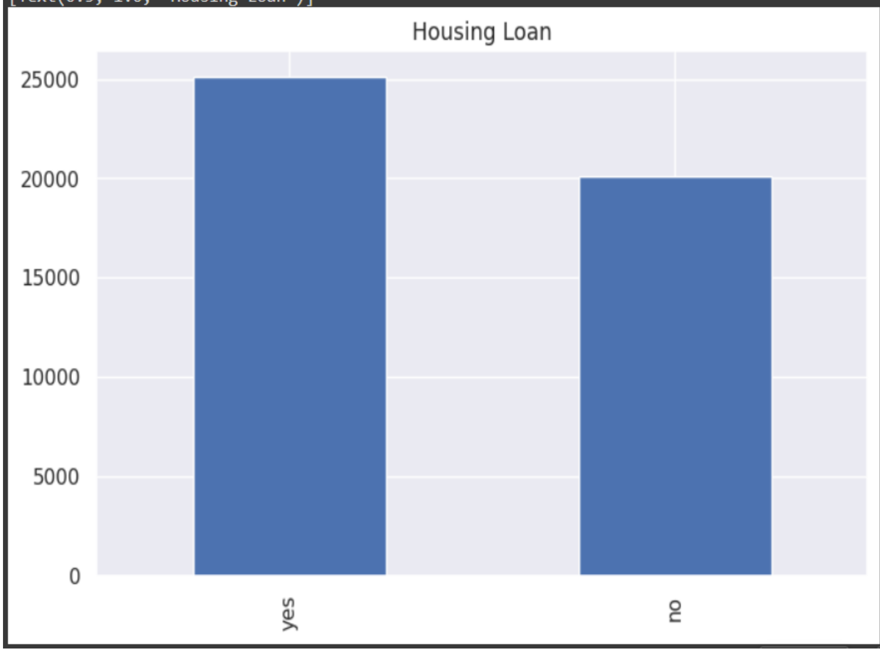

Figure 24: Count vs client has housing loan or not plot

The count of education

```
#the count of education
education_count = data['education'].value_counts()
education_count
```

```
secondary    23202
tertiary     13301
primary       6851
unknown       1857
Name: education, dtype: int64
```

Figure 25: Value count of education

Count vs education plot

```
#count vs education plot
plt.figure(figsize = (8, 5))
education_count.plot(kind = "bar").set(title = "Education Column Distribution")
```

```
[Text(0.5, 1.0, 'Education Column Distribution')]
```

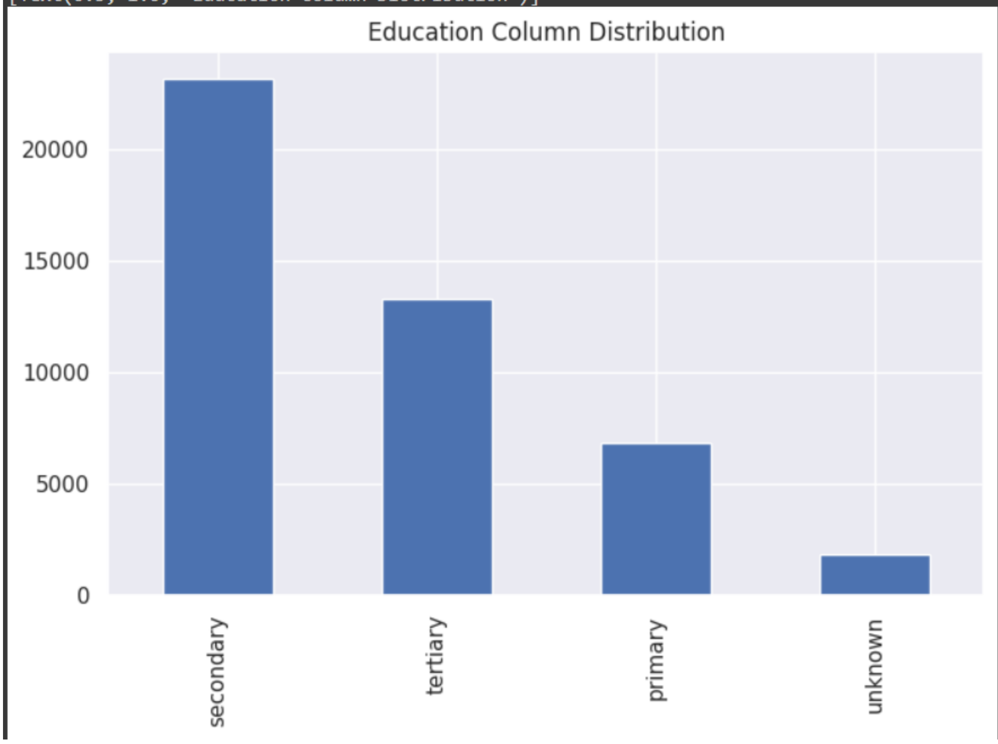

Figure 26: Count vs education plot

The count of contact

```
#the count of contact
contact_count = data['contact'].value_counts()
contact_count
```

```
cellular     29285
unknown      13020
telephone     2906
Name: contact, dtype: int64
```

Figure 27: Value count of contact

Count vs contact plot

```
#count vs contact plot
plt.figure(figsize = (8, 5))
contact_count.plot(kind = "bar").set(title = "Contact")
```

```
[Text(0.5, 1.0, 'Contact')]
```

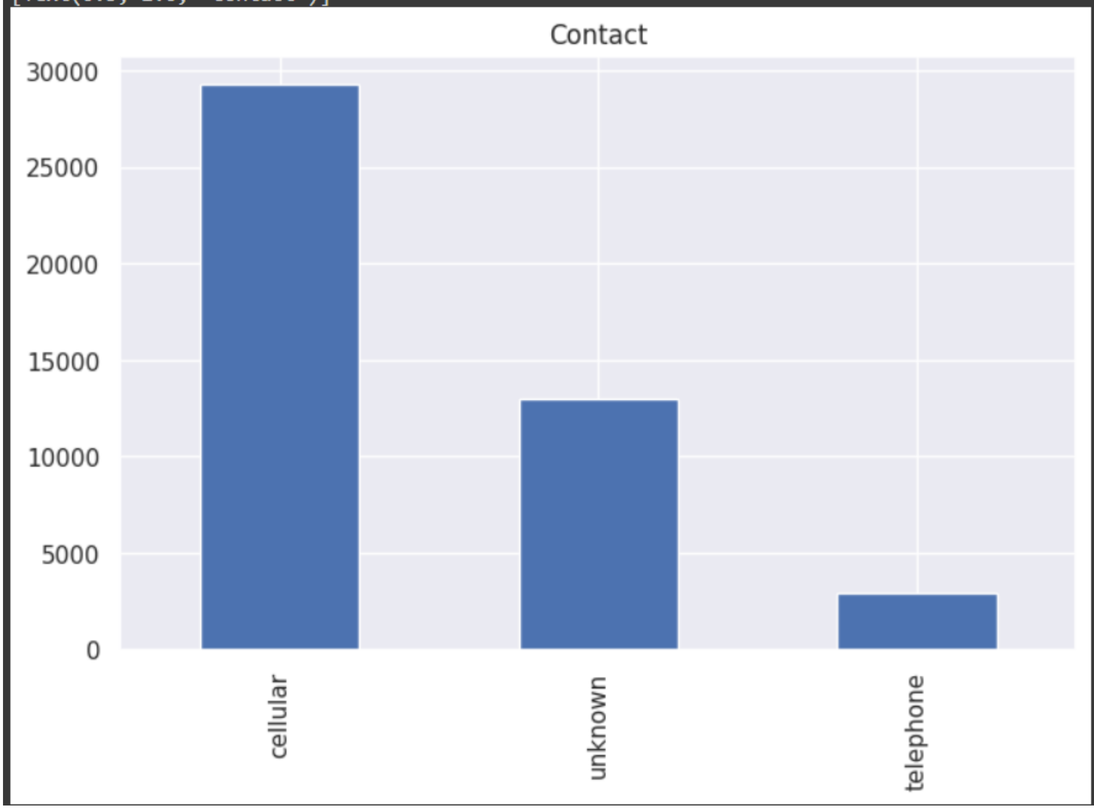

Figure 28: Count vs contact plot

The count of months

```
#count of month
month_count = data['month'].value_counts()
month_count
```

```
may    13766
jul     6895
aug     6247
jun     5341
nov     3970
apr     2932
feb     2649
jan     1403
oct      738
sep      579
mar      477
dec      214
Name: month, dtype: int64
```

Figure 29: Value count of month

Count vs month plot

```
#count vs month plot
plt.figure(figsize = (8, 5))
month_count.plot(kind = "bar").set(title = "Month")
```

```
[Text(0.5, 1.0, 'Month')]
```

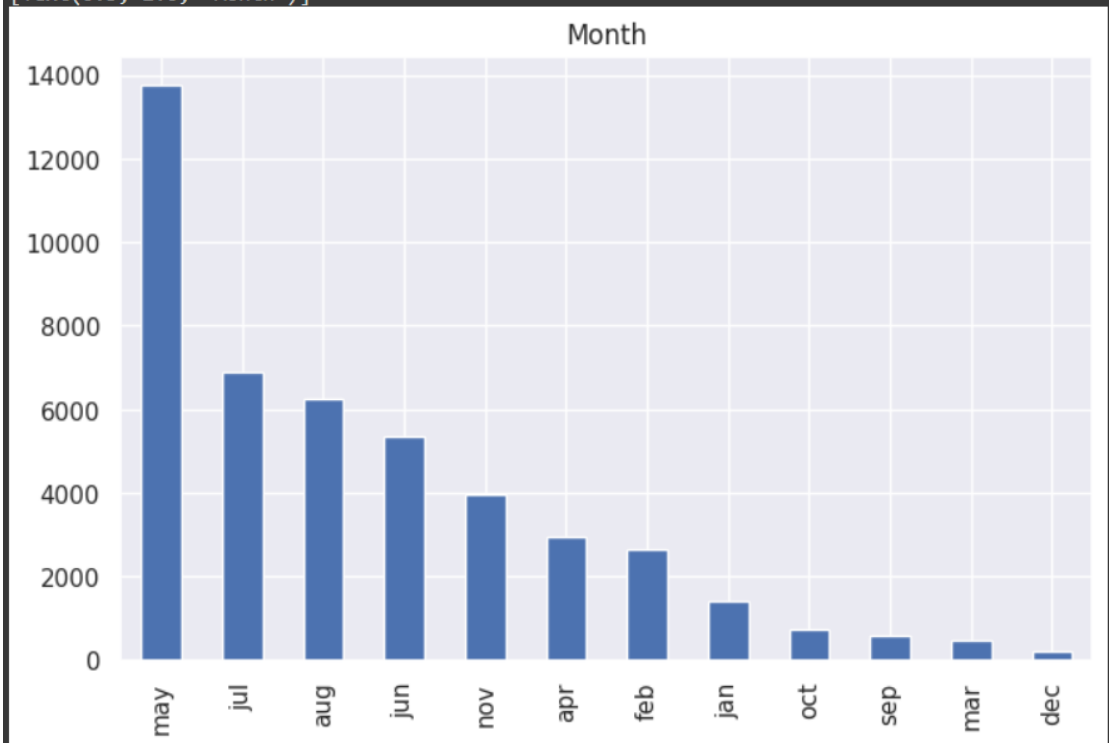

Figure 30: Count vs month plot

The count of target

```
#the count of target
target_count = data['Target'].value_counts()
target_count
```

```
no     39922
yes     5289
Name: Target, dtype: int64
```

Figure 31: Value count of target

Count vs target plot

```
#target vs count plot
plt.figure(figsize = (8, 5))
target_count.plot(kind = "bar").set(title = "Target")
```

```
[Text(0.5, 1.0, 'Target')]
```

Figure 32: Count vs target plot

Histogram of age

```
#histogram of age
plt.hist(data['age'])
plt.title("Age")
```

```
Text(0.5, 1.0, 'Age')
```

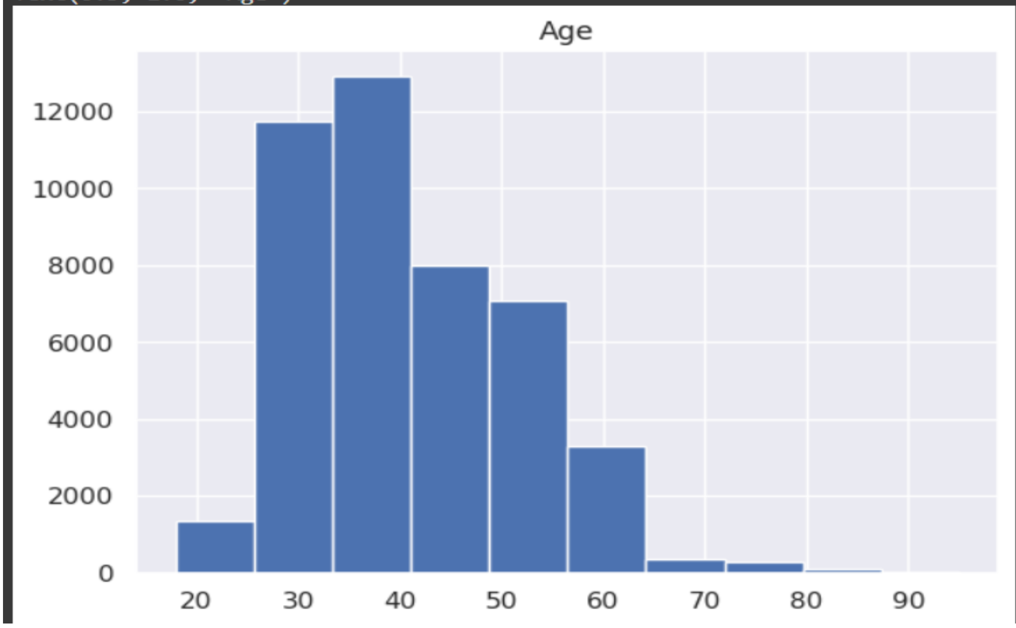

Figure 33: Histogram of age

Distribution of age

```
sb.displot(data['age'],kde=True,color="green")
plt.title("Age Distribution",fontsize=20)
plt.show()
```

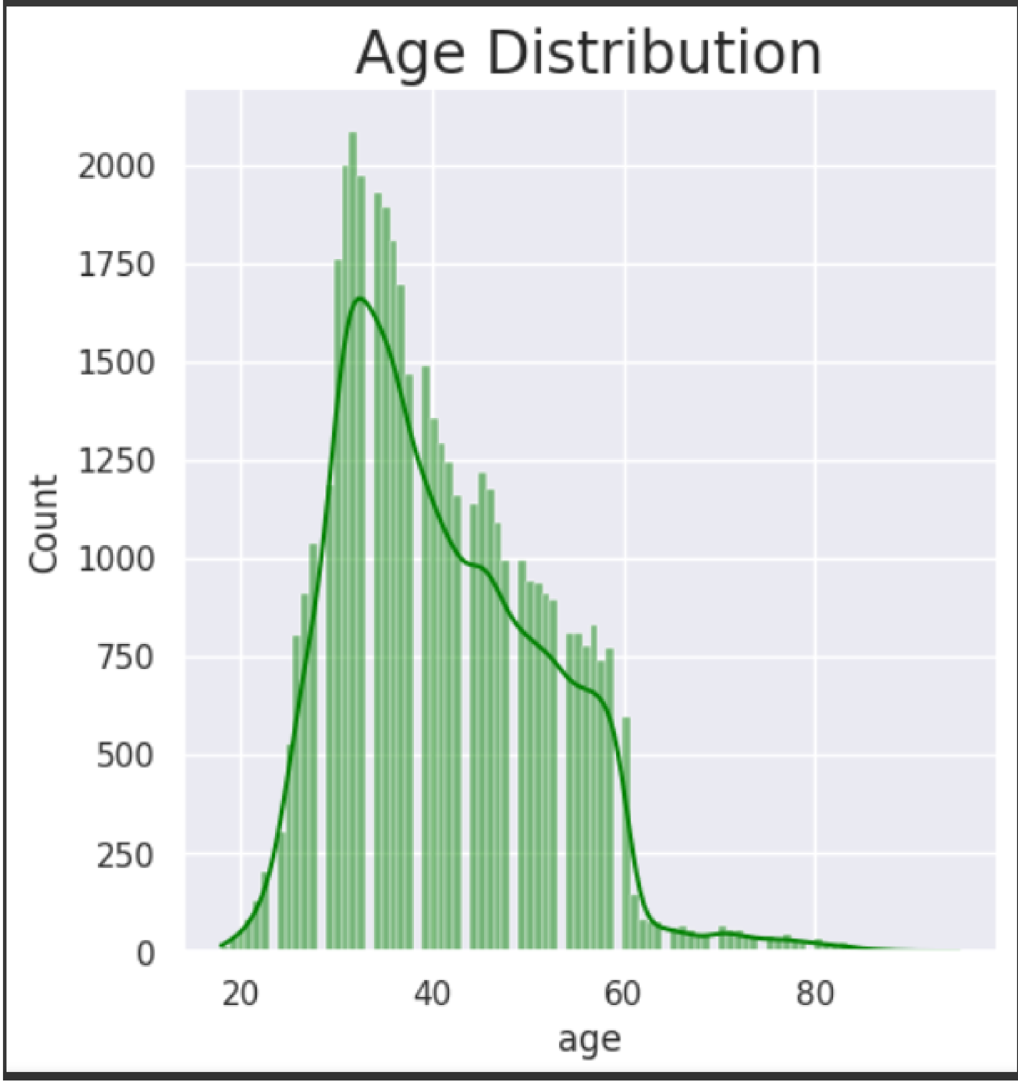

Figure 34: Distribution of age

Job vs target catplot

```
#job vs target catplot
sb.catplot(data=data,x='job',kind='count',hue='Target')
plt.xticks(rotation=55)
plt.show()
```

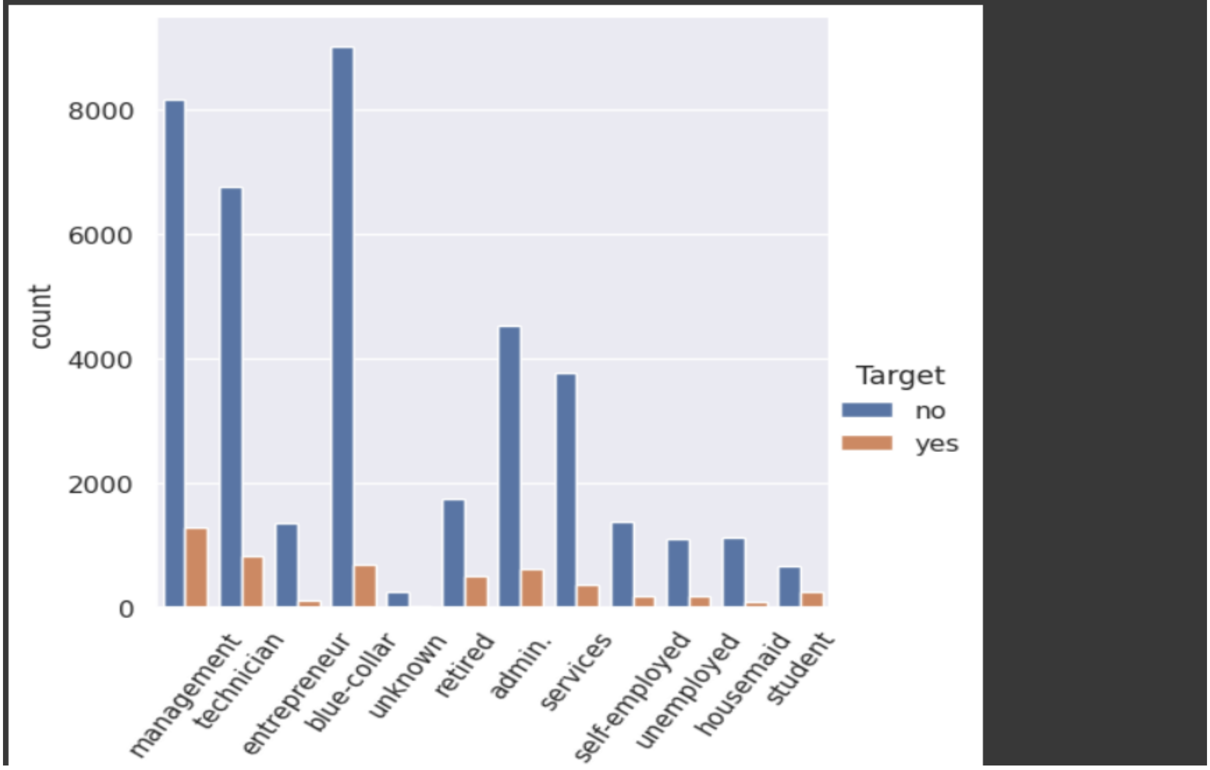

Figure 35: job vs target catplot

Marital vs target catplot

```
sb.catplot(data=data,x='marital',kind='count',hue='Target')
plt.xticks(rotation=55)
plt.show()
```

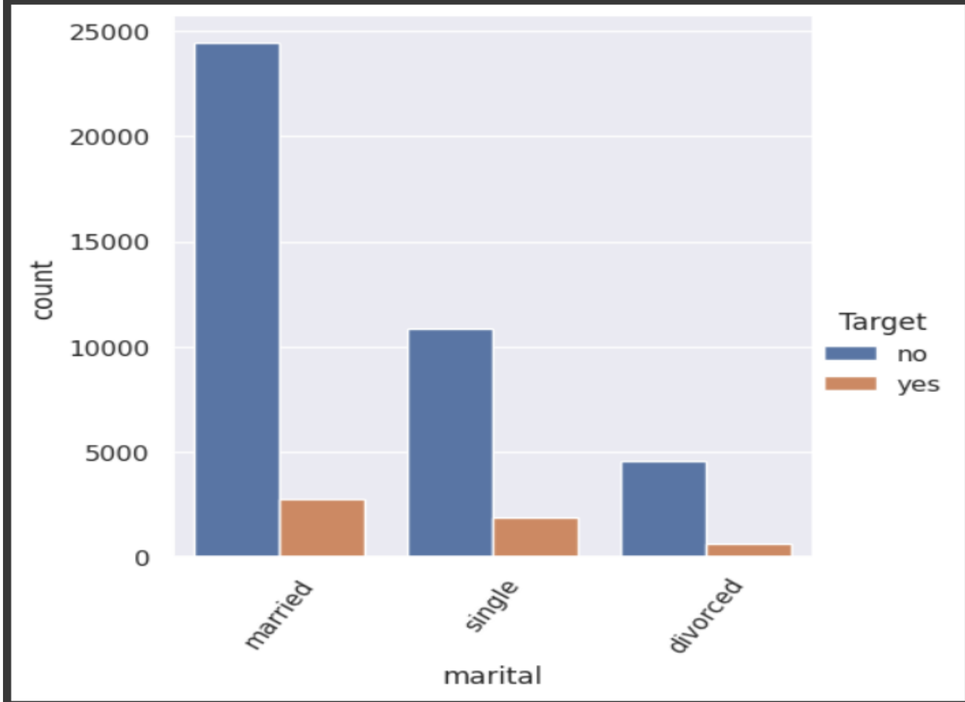

Figure 36: Martial vs target catplot

Education vs target catplot

```
sb.catplot(data=data,x='education',kind='count',hue='Target')
plt.xticks(rotation=55)
plt.show()
```

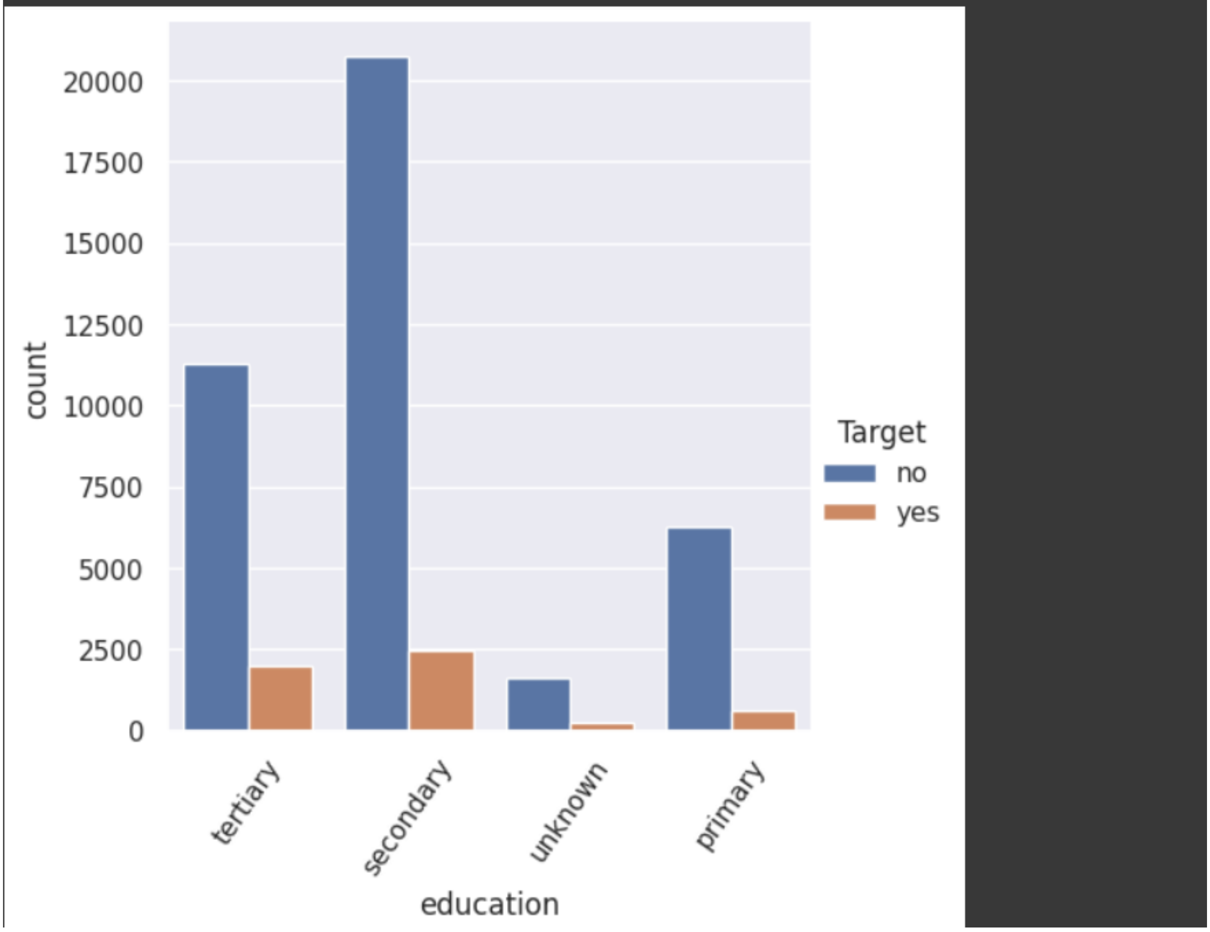

Figure 37: Education vs target catplot

Logistic Regression

Converting the categorical variable to numerical variable

```
from sklearn.preprocessing import LabelEncoder
#creating an encoder
le=LabelEncoder()
```

```
def object_to_int(dataframe_series):
    if dataframe_seri[illegible]'object':
        dataframe_series = LabelEncoder().fit_transform(dataframe_series)
    return dataframe_series
```

```
df = data.apply(lambda x: object_to_int(x))
```

Figure 38: Converting the categorical variable to numerical variable

```
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	Target
0	58	4	1	2	0	2143	1	0	2	5	8	261	1	-1	0	3	0
1	44	9	2	1	0	29	1	0	2	5	8	151	1	-1	0	3	0
2	33	2	1	1	0	2	1	1	2	5	8	76	1	-1	0	3	0
3	47	1	1	3	0	1506	1	0	2	5	8	92	1	-1	0	3	0
4	33	11	2	3	0	1	0	0	2	5	8	198	1	-1	0	3	0

Figure 39: After converting the categorical variable to numerical variable

Divide features and target into train and test data

```
#Divide features and target into train and test data
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, random_state = 0,stratify=y)
```

Figure 40: Divide features and target into train and test data

View the shape of x-train, x_test

```
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.metrics import recall_score, confusion_matrix, precision_score, f1_score, accuracy_score, classification_report
```

```
(9043, 16)
```

Figure 41: View the shape of x_train , x_test

Create a logistic regression model

```
#Create a LogisticRegression model
model=LogisticRegression()
```

```
model.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
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-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

Figure 42: Create a logistic model

Make predictions on the test data

```
# Make predictions on the test data
y_pred = model.predict(x_test)
model.coef_
model.intercept_
y_pred=model.predict(x_test)
y_pred
confusion_matrix(y_test,y_pred)
```

```
array([[7827,  158],
       [ 874,  184]])
```

Figure 43: Make predictions of the test data

Accuracy of the model

```
#accuracy of the model
accuracy=(np.diag(confusion_matrix(y_test,y_pred)).sum())/len(y_test)*100/100
accuracy
```

```
0.8858785801172178
```

Figure 44: Accuracy of the model

Conclusion

- Deposits have been used by people between the ages of 30 and 40.
- The vast majority of these people work in the blue-collar sector.
- Deposits are used more frequently by married persons than by single and divorced people.
- Most of these people have accepted housing loans rather than personal loans.
- Most of them have only completed their secondary schooling.
- The model's accuracy is 0.88, which translates to 88% when expressed as a percentage.

Summary

Data from a Portuguese banking institution's direct marketing efforts are being used.

On phone conversations, the marketing campaigns were based. In order to determine if the product (bank term deposit) would be subscribed to (or not), it was frequently necessary to make multiple contacts with the same client. I did some visualization part and create a logistic model.