

Banking Dataset - Marketing Strategy

Materi
Presentasi
Final Project

(dipresentasikan setiap sesi mentoring)



1. Background

Permasalahan yang ingin diselesaikan

- Divisi Marketing Investment Bank saat ini memiliki **urgensi** untuk melakukan **pengurangan biaya marketing** untuk term deposit pada nasabah yang ada, dikarenakan keuangan bank terdampak akibat kondisi ekonomi yang belum pulih 100% akibat pandemi..
- Ini perlu dilakukan, sebab, dalam **campaign sebelumnya**, hanya sekitar **11%** dari total nasabah yang **setuju untuk berlangganan** sedangkan campaign telah dilakukan untuk **semua** nasabah.

Peran dalam Project

- Kami selaku **tim data Scientist** yang mensupport divisi marketing Bank, bertugas membuat **model machine learning yang bisa memprediksi pelanggan mana saja yang berpotensi berlangganan deposite** dengan tepat, karena filtering secara manual dianggap tidak memiliki akurasi yang bagus dengan banyaknya profil nasabah dengan kompleksitas tinggi.

1. Background

Goal yang Ingin Dicapai

1. Dapat **menentukan target pelanggan yang potensial untuk membuka deposito** dengan *Machine Learning*.
2. Dapat mengetahui **fitur-fitur penting apa saja yang mendasari keputusan berlangganan** dengan *Machine Learning*.
3. **Meningkatkan nilai conversion rate** pelanggan yang “berlangganan”
4. **Mengurangi biaya marketing** pada campaign deposito berjangka

Objektif yang Ingin Dicapai

1. **Mencari dan menentukan algoritma Machine Learning (ML) dengan nilai evaluation metric tertinggi** untuk melakukan prediksi dan rekomendasi pelanggan yang berpotensi berlangganan term-deposit
2. **Mengklasifikasi pelanggan yang diprediksi berlangganan** berdasarkan variabel penting dengan ML.
3. **Menghitung conversion rate pelanggan** yang berlangganan dibandingkan dengan jumlah pelanggan yang dihubungi dengan ML.
4. **Menghitung jumlah biaya yang berkurang jika melakukan marketing** kepada pelanggan yang direkomendasikan ML.

1. Background

Business Metrics untuk Mengukur Ketercapaian Objective

1. **Cost Reduction** = menghitung jumlah biaya yang dikeluarkan dan penghematan yang dilakukan setelah melakukan apa yang direkomendasikan machine learning.
2. **Conversion rate** = jumlah nasabah yang berlangganan deposito berjangka secara **aktual** / jumlah total nasabah yang diprediksi “berlangganan” oleh machine learning (**Output variable y = ‘yes’**) dan menerima panggilan telepon. **Semakin tinggi semakin baik.**

EXPLORATORY DATA ANALYSIS

Tentang Dataset

Terdapat 17 kolom (16 kolom fitur dan 1 kolom target)

	age	job	marital	education	default	balance	housing	loan
23547	32	management	married	tertiary	no	0	no	no
16662	40	blue-collar	married	secondary	no	3131	yes	no
11145	48	management	single	tertiary	no	0	no	no
24101	31	admin.	married	secondary	no	352	no	no
2632	52	admin.	divorced	secondary	no	26	yes	no

	contact	day	month	duration	campaign	pdays	previous	poutcome	y
	cellular	28	aug	15	13	-1	0	unknown	no
	cellular	24	jul	401	1	-1	0	unknown	no
	unknown	18	jun	96	3	-1	0	unknown	no
	telephone	28	oct	60	1	-1	0	unknown	no
	unknown	13	may	215	1	-1	0	unknown	no

Terdapat 2 jenis tipe data, yaitu **int64** dan **object**. Semua tipe data **sudah sesuai dengan kolom fitur**.

```
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  y           45211 non-null  object  
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

EXPLORATORY DATA ANALYSIS

Descriptive Statistic

	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

Kolom Numerical

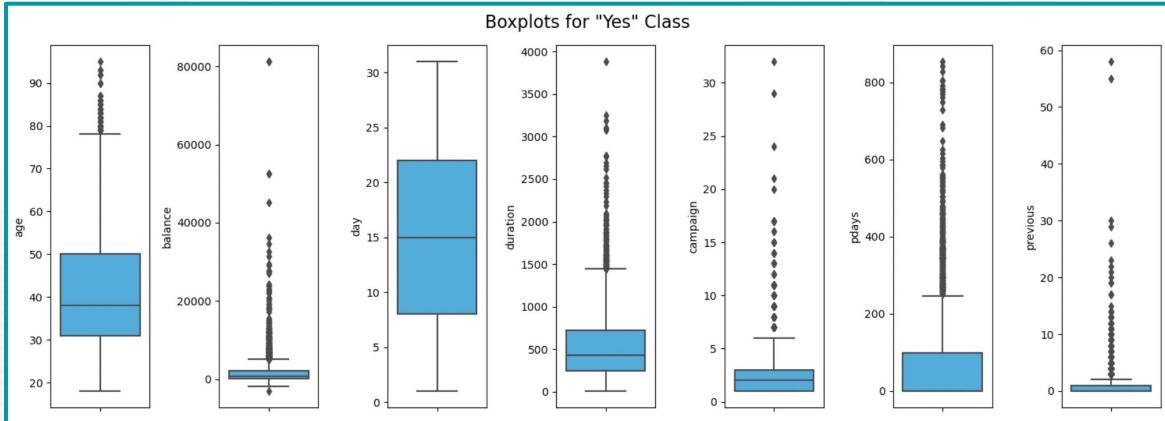
Nilai **mean** lebih besar nilainya bila dibandingkan dengan **median**, yang mengindikasikan bahwa kurva distribusi frekuensi **menceng kanan** atau **kemencengan positif**.

	job	marital	education	default	housing	loan	contact	month	poutcome	y
count	45211	45211	45211	45211	45211	45211	45211	45211	45211	45211
unique	12	3	4	2	2	2	3	12	4	2
top	blue-collar	married	secondary	no	yes	no	cellular	may	unknown	no
freq	9732	27214	23202	44396	25130	37967	29285	13766	36959	39922

Kolom Categorical

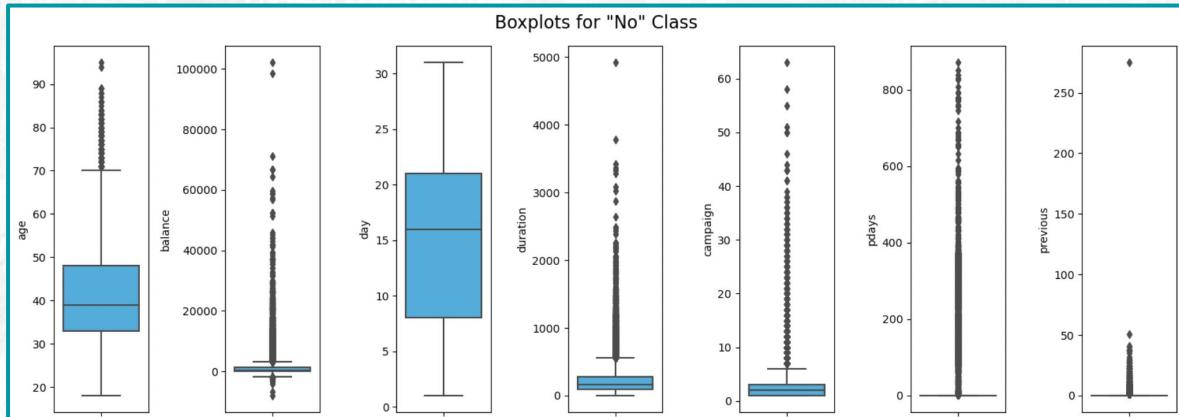
Sebagian besar dari para nasabah tersebut **tidak mendepositkan uang mereka pada bank sebelumnya**

Univariate Analysis



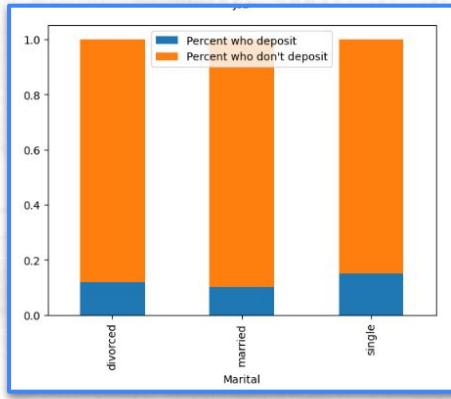
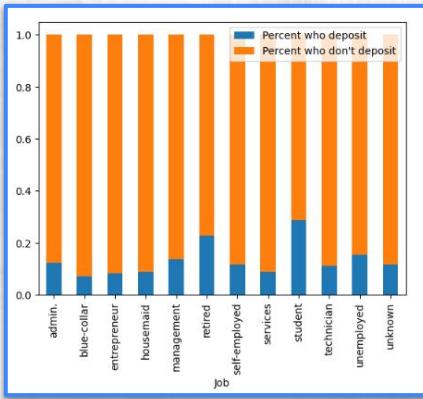
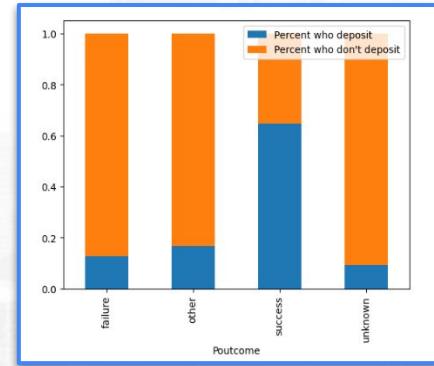
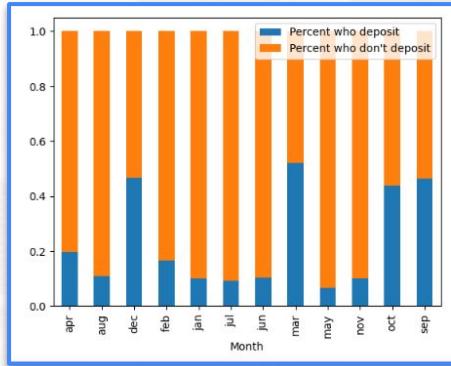
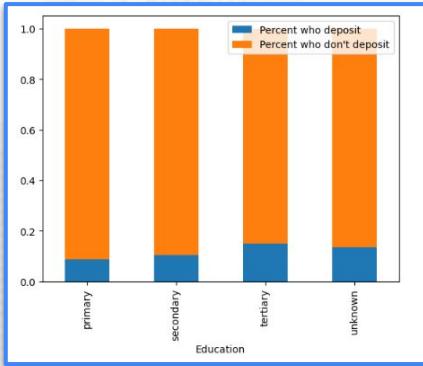
Beberapa variabel dalam dataset **memiliki outliers yang signifikan di sisi atas distribusi.**

Dengan adanya outliers, kami berencana menggunakan **robust scaler, penghapusan atau transformasi nilai-nilai outliers tersebut** pada tahap pre-processing nanti.



Multivariate Analysis

Stacked Bar Chart - Categorical Columns

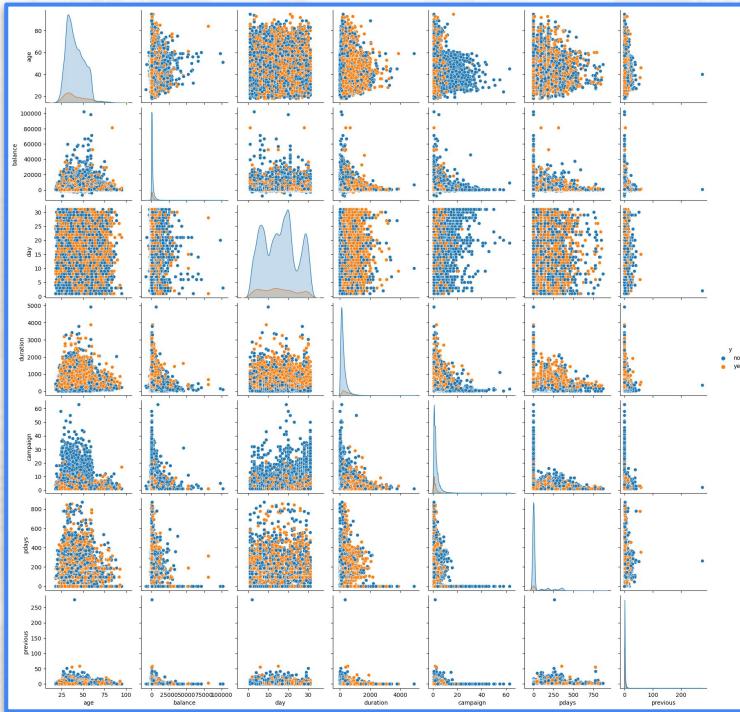


Nasabah yang pernah mendepositokan uangnya adalah **kelompok pelajar, nasabah single, yang memiliki pendidikan tinggi**, serta kampanye pada bulan-bulan seperti **Maret dapat meningkatkan peluang konversi nasabah menjadi pelanggan berlangganan depositeo berjangka**.

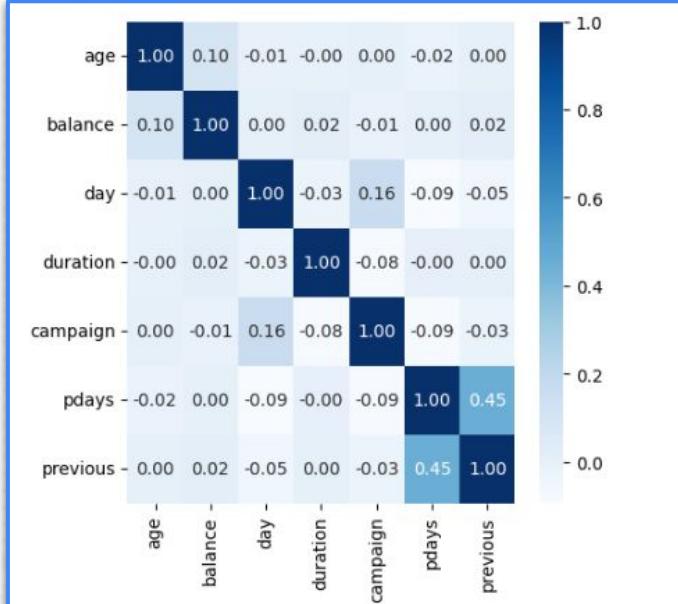
Multivariate Analysis

Heatmap & Pairplot - Numerical Columns

Scatter plot **tidak bisa menunjukkan pola yang membedakan** antara nasabah yang mendepositkan uangnya, dengan yang tidak.



Feature '**pdays**' dan '**previous**' memiliki korelasi yang masuk dalam kategori **moderate correlation (0.45)**, namun sisanya berkorelasi lemah atau sangat lemah.



Data Cleansing (Handle missing values)

```
[4] 1 # Missing values
2 missing_values = df.isna().sum()
3 missing_values

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

Tidak ada missing values yang terdapat pada dataset, sehingga tidak perlu untuk melakukan tindak lebih lanjut.

Data Cleansing (Handle duplicated data)

```
[5] 1 # Menghitung jumlah data yang duplikat
2 duplicate_count = df.duplicated().sum()
3 print("Jumlah data yang duplikat:", duplicate_count)
```

```
Jumlah data yang duplikat: 0
```

Tidak ada data yang duplikat pada dataset, sehingga tidak perlu untuk melakukan tindak lebih lanjut.

Data Cleansing (Feature transformation)

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
53	services	divorced	primary	no	-291	yes	yes	unknown	7	may	591	1	-1	0	unknown	yes
49	services	married	secondary	no	-8	yes	no	unknown	8	may	1119	1	-1	0	unknown	yes
43	blue-collar	married	primary	no	-192	yes	no	unknown	8	may	1120	2	-1	0	unknown	yes
32	blue-collar	married	secondary	yes	-1	yes	no	unknown	9	may	653	1	-1	0	unknown	yes
28	blue-collar	single	secondary	no	-197	yes	no	unknown	9	may	2016	2	-1	0	unknown	yes

Pemutusan untuk mengubah nilai **negatif** pada saldo menjadi nilai **absolut** dikarenakan terdapat beberapa pelanggan dengan nilai negatif yang memiliki hasil **target (y)** yang bernilai '**yes**'. Dalam hal ini mungkin terjadi **kesalahan input** dikarenakan pelanggan dengan saldo negatif tidak mungkin atau tidak memenuhi syarat untuk membuka term deposit.

Before

	age	balance	day	duration	campaign	pdays	previous
count	36954.000000	36954.000000	36954.000000	36954.000000	36954.000000	36954.0	36954.0
mean	40.932430	1318.788846	16.145424	257.726119	2.921957	-1.0	0.0
std	10.430218	3039.557077	8.372554	262.256406	3.325791	0.0	0.0
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.0	0.0
25%	33.000000	55.000000	9.000000	101.000000	1.000000	-1.0	0.0
50%	39.000000	414.000000	17.000000	177.000000	2.000000	-1.0	0.0
75%	49.000000	1358.000000	22.000000	318.000000	3.000000	-1.0	0.0
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	-1.0	0.0

After

	age	balance	day	duration	campaign	pdays	previous
count	36954.000000	36954.000000	36954.000000	36954.000000	36954.000000	36954.0	36954.0
mean	40.932430	1375.862451	16.145424	257.726119	2.921957	-1.0	0.0
std	10.430218	3014.151556	8.372554	262.256406	3.325791	0.0	0.0
min	18.000000	0.000000	1.000000	0.000000	1.000000	-1.0	0.0
25%	33.000000	123.000000	9.000000	101.000000	1.000000	-1.0	0.0
50%	39.000000	457.000000	17.000000	177.000000	2.000000	-1.0	0.0
75%	49.000000	1373.000000	22.000000	318.000000	3.000000	-1.0	0.0
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	-1.0	0.0

Data Cleansing (Feature encoding “Target”)

▼ Label Encode Variable Target (y)

```
[ ] #dataframe before label encoding 'y'
df.tail()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
45203	23	student	single	tertiary	no	113	no	no	cellular	17	nov	266	1	-1	0	unknown	yes
45205	25	technician	single	secondary	no	505	no	yes	cellular	17	nov	386	2	-1	0	unknown	yes
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown	yes
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown	yes
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown	no

```
▶ from sklearn.preprocessing import LabelEncoder

# Initialize LabelEncoder
encoder = LabelEncoder()

# Fit and transform the 'y' column
df['y'] = encoder.fit_transform(df['y'])

# Print the encoded DataFrame
df.tail()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
45203	23	student	single	tertiary	no	113	no	no	cellular	17	nov	266	1	-1	0	unknown	1
45205	25	technician	single	secondary	no	505	no	yes	cellular	17	nov	386	2	-1	0	unknown	1
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown	1
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown	1
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown	0

Untuk feature ‘y’, kami melakukan label encoding karena feature tersebut akan kami jadikan target, dimana untuk “yes” di encode 1 dan “no” di encode 0.

Data Cleansing (Feature encoding “predictor”)

Default, housing, loan adl fitur binary sehingga drop first column

```
encodee1 = ['default', 'housing', 'loan']

for e in encodee1:
    ohe = pd.get_dummies(df[e], prefix=e, drop_first=True)
    df = pd.concat([df, ohe], axis=1) # Concatenate the one-hot encoded columns

df = df.drop(encodee1, axis=1) # Drop the original categorical columns

print(df)
```

default_no	default_yes	housing_no	housing_yes	loan_no	lo
1	0	1	0	1	
1	0	1	0	0	
1	0	1	0	1	
1	0	1	0	1	
1	0	1	0	1	

Job, marital, education, contact, poutcome adl multiple class sehingga tidak drop first column

```
encodee2 = ['job', 'marital', 'education', 'contact', 'poutcome']

for e in encodee2:
    ohe = pd.get_dummies(df[e], prefix=e, drop_first=False) # Drop the first column
    df = pd.concat([df, ohe], axis=1) # Concatenate the one-hot encoded columns

df = df.drop(encodee2, axis=1) # Drop the original categorical columns

print(df)
```

contact_cellular	contact_telephone	contact_unknown	poutcome_failure	poutcome_other	poutcome_success	poutcome_unknown
1	0	0	0	0	0	1
1	0	0	0	0	0	1
1	0	0	0	0	1	0
0	1	0	0	0	0	1
1	0	0	0	1	0	0

Feature Engineering

Feature Engineering (Feature extraction)

month_quartal

```

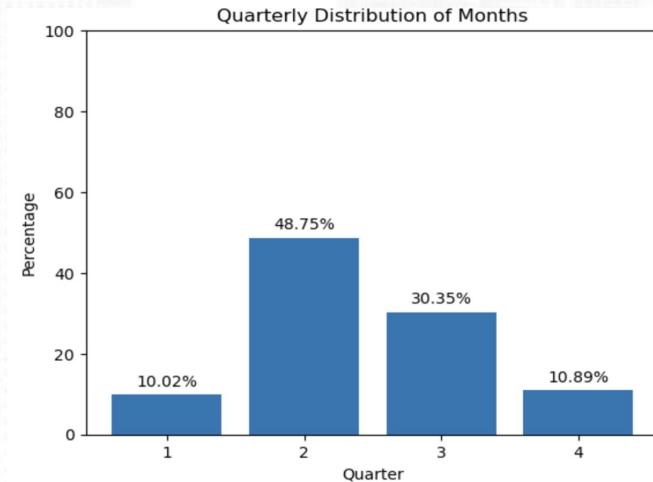
quartal_counts = df['month_quartal'].value_counts().sort_index()
total_respondents = len(df)
month_percentage = quartal_counts / total_respondents * 100

# Create the bar plot with percentage labels
plt.bar(month_percentage.index, month_percentage.values)
plt.xlabel('Quarter')
plt.ylabel('Percentage')
plt.title('Quarterly Distribution of Months')

# Adding percentage labels on top of the bars
for i, percentage in enumerate(month_percentage):
    plt.text(i + 1, percentage + 2, f'{percentage:.2f}%', ha='center')

plt.xticks(month_percentage.index)
plt.ylim(0, 100)
plt.show()

```



Fitur month_quartal diekstraksi dari fitur month, dengan pertimbangan untuk melihat banyaknya nasabah yang dihubungi pada quarter tertentu. Hasilnya, quarter terbanyak jatuh pada quarter 2, yang berarti nasabah paling banyak dihubungi pada bulan April, Mei, Juni.

Feature Engineering (Feature extraction)

duration_bin

```
# Make segment for duration feature
# Define the bin edges per 5 mins
bin_edges = [1, 300, 600, 900, 1200, 10000]

# Define the bin labels
bin_labels = ['0-5mins', '5-10mins', '10-15mins', '15-20mins', '>=20mins']

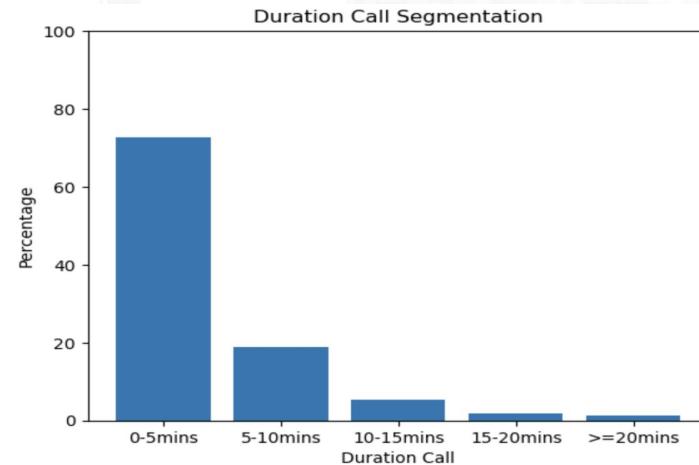
# Add a new column with bin categories
df['duration_bin'] = pd.cut(df['duration'], bins=bin_edges, labels=bin_labels, right=False)

print(df)

duration_counts = df['duration_bin'].value_counts().sort_index()
total_respondents = len(df)
duration_percentage = duration_counts / total_respondents * 100

# Create the bar plot with percentage labels
plt.bar(duration_percentage.index, duration_percentage.values)
plt.xlabel('Duration Call')
plt.ylabel('Percentage')
plt.title('Duration Call Segmentation')

plt.xticks(duration_percentage.index)
plt.ylim(0, 100)
```



Fitur duration_bin diekstraksi dari fitur duration, dengan pertimbangan untuk melihat durasi telepon/telemarketing. Hasilnya, angka durasi terbanyak berada pada kisaran 1 - 300 detik / 5 menit pertama.

Feature Engineering (Feature selection)

- Feature '**pdays**' dipertimbangkan untuk **tidak dimasukkan** dalam modelling karena memiliki korelasi tinggi dengan fitur 'previous' dan juga memiliki nilai **majoritas -1** dan **standar deviasi** yang sangat besar.
- Feature '**default**' tidak dimasukkan ke dalam modelling dikarenakan terdapat **class imbalance**: 98% value no dan 2% yes.
- Semua hasil OHE feature '**job**' tidak dimasukkan ke dalam modelling karena mengandung terlalu banyak data **unique**.
- Feature '**day**' tidak dimasukkan karena tidak memberikan info yang penting (hanya berisi tanggal)
- Feature '**campaign**' , semua hasil OHE feature **month** dan **contact** tidak dimasukkan karena secara konteks tidak relevan.
- Feature '**education_unknown**' tidak dimasukkan karena tidak berisi info yang jelas.
- Feature '*previous*' juga dipertimbangkan untuk tidak dimasukkan dikarenakan tidak begitu relevan,, namun melalui analisis **RFE**, jika tidak masuk dalam fitur terpenting, maka akan di drop.

RFE

```

import numpy as np
from sklearn.datasets import make_classification
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# Generate a sample dataset
X, y = make_classification(n_samples=100, n_features=20, random_state=42)

# Create a logistic regression model
model_glm = LogisticRegression()
model_rfo = RandomForestClassifier()
model_xgboost = XGBClassifier()

# Create the RFE model and specify the number of features to select (e.g., 10)
rfe_glm = RFE(model_glm, n_features_to_select=3)
model_rfo = RFE(model_rfo, n_features_to_select=3)
model_xgboost = RFE(model_xgboost, n_features_to_select=3)

# Fit the RFE model to the data
rfe_glm.fit(X, y)
model_rfo.fit(X, y)
model_xgboost.fit(X, y)

# Get the selected features
selected_features_glm = np.where(rfe_glm.support_)[0]
selected_features_rfo = np.where(model_rfo.support_)[0]
selected_features_xgboost = np.where(model_xgboost.support_)[0]

print("Selected Features with GLM :", selected_features_glm)
print("Selected Features with RF :", selected_features_rfo)
print("Selected Features with XGB :", selected_features_xgboost)

```

#	Column	Non-Null Count	Dtype
0	age	45211	int64
1	balance	45211	int64
2	previous	45211	int64
3	housing_yes	45211	int64
4	loan_yes	45211	int64
5	marital_divorced	45211	int64
6	marital_married	45211	int64
7	marital_single	45211	int64
8	education_primary	45211	int64
9	education_secondary	45211	int64
10	education_tertiary	45211	int64
11	poutcome_failure	45211	int64
12	poutcome_other	45211	int64
13	poutcome_success	45211	int64
14	poutcome_unknown	45211	int64
15	duration_bin_0-5mins	45211	int64
16	duration_bin_5-10mins	45211	int64
17	duration_bin_10-15mins	45211	int64
18	duration_bin_15-20mins	45211	int64
19	duration_bin_>=20mins	45211	int64

dtypes: int64(20)
memory usage: 6.9 MB

Selected Features with GLM : [1 10 15]
 Selected Features with RF : [1 10 15]
 Selected Features with XGB : [4 10 15]

Fitur **Previous** akan kami drop juga karena tidak menjadi 3 fitur terpenting menurut RFE.

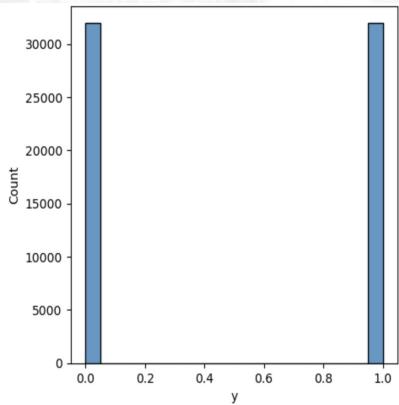
Handling Class Imbalance (y)

```
# x nya adalah atribut yang me
# y nya adalah target itu sena
X = df.drop(['y'], axis = 1)
y = df['y']

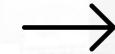
print(X.shape)
print(y.shape)
```

(45211, 23)
(45211,)

Menentukan variabel dataset



Hasil akhir menunjukkan jumlah yang seimbang



```
[ ] from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from collections import Counter

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply SMOTE to the training data
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Print class distribution before and after applying SMOTE
print("Class distribution before SMOTE:", Counter(y_train))
print("Class distribution after SMOTE:", Counter(y_train_resampled))
```

Pertama, kita mengimpor perpustakaan yang diperlukan untuk menggunakan SMOTE dan memeriksa distribusi kelas. Kemudian, kita memuat dataset dan membaginya menjadi subset train dan test. Penerapan SMOTE pada data train menggunakan metode fit_resample.



```
print(pd.Series(y_train_resampled).value_counts())
0    31970
1    31970
Name: y, dtype: int64
```

Hasil akhir menunjukkan jumlah yang sama antara kedua kelas, yaitu 'yes' dan 'no' dengan jumlah yang sama yaitu 26846. Jumlah total sampel dapat meningkat secara signifikan, karena SMOTE menghasilkan sampel sintetik dengan melakukan interpolasi antar sampel yang ada di kelas minoritas, dan dengan melakukan hal tersebut, berpotensi membuat sampel baru dalam jumlah besar.



Feature Engineering (Scaling Dataset)

Robust Scaler

```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()
columns_to_scale = ['age', 'balance']

scaler.fit(X_train_resampled[columns_to_scale])
for i, df in enumerate(dflist):

    data_to_scale = df[columns_to_scale]

    scaled_data = scaler.transform(data_to_scale)

    df[columns_to_scale] = scaled_data
```

Robust scaler perlu digunakan pada kolom 'age' dan 'balance' untuk mengatasi distribusi skewed. Dengan menggunakan median dan IQR, robust scaler dapat mengurangi pengaruh outlier pada proses penskalaan dan membantu menghasilkan data yang memiliki distribusi yang lebih terpusat dan terdistribusi secara merata.

Feature Engineering (Scaling Dataset)

Standard Scaler

```
from sklearn.preprocessing import StandardScaler

standard_scaler = StandardScaler()
columns_to_standard_scale = [i for i in df.columns.to_list() if not (i in columns_to_scale or i in encodeel or i in
standard_scaler.fit(X_train_resampled[columns_to_standard_scale])
for i, df in enumerate(dflist):
    data_to_standard_scale = df[columns_to_standard_scale]

    standard_scaled_data = standard_scaler.transform(data_to_standard_scale)

df[columns_to_standard_scale] = standard_scaled_data
```

	age	balance	housing_yes	loan_yes	marital_divorced	marital_married	marital_single	education_primary
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	0.172718	0.495253	0.341063	0.207687	0.147120	0.221982	0.075593	0.184208
std	0.624633	1.870249	1.018940	1.231688	1.203571	0.979170	1.039963	1.211495
min	-1.176471	-5.267199	-0.798786	-0.330313	-0.287100	-0.982081	-0.577591	-0.327773
25%	-0.294118	-0.297297	-0.798786	-0.330313	-0.287100	-0.982081	-0.577591	-0.327773
50%	0.058824	-0.066339	1.251900	-0.330313	-0.287100	1.018246	-0.577591	-0.327773
75%	0.588235	0.535627	1.251900	-0.330313	-0.287100	1.018246	1.731329	-0.327773
max	3.352941	62.390049	1.251900	3.027432	3.483111	1.018246	1.731329	3.050888

	poutcome_success	poutcome_unknown	duration_bin_0-5mins	duration_bin_5-10mins	duration_bin_10-15mins	duration_bin_15-20mins	duration_bin_>=20mins
	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
	-0.030864	0.361357	0.507913	0.098136	0.020939	0.030967	0.032019
	0.923567	0.806997	0.892014	1.085965	1.043033	1.115674	1.151152
	-0.202598	-1.346485	-0.948583	-0.425430	-0.225039	-0.124113	-0.097246
	-0.202598	0.742674	-0.948583	-0.425430	-0.225039	-0.124113	-0.097246
	-0.202598	0.742674	1.054204	-0.425430	-0.225039	-0.124113	-0.097246
	4.935884	0.742674	1.054204	2.350565	4.443677	8.057141	10.283218

Standard Scaler perlu digunakan pada kolom sisanya untuk mengubah skala data menjadi standar dan sesuai dengan asumsi distribusi normal. data akan diubah sehingga memiliki rata-rata nol dan simpangan baku satu, sesuai dengan distribusi normal standar.

Modeling

```
[ ] from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

[ ] models = {
    "logistic": LogisticRegression(solver='sag', max_iter=250),
    "RandomForest" :RandomForestClassifier(random_state=42),
    "XGBoost" : XGBClassifier(),
}
```

Kami memutuskan untuk menerapkan tiga jenis model yang berbeda dalam uji coba kami, dan nantinya kami akan melakukan evaluasi ulang terhadap ketiga model tersebut, dengan metrics yang telah ditentukan. Tiga jenis model yang akan kami gunakan adalah **Logistic Regression, Random Forest, dan XGBoost**.

Modeling (Metrics)

```
[ ] models_precision = {}
models_cm = {}
models_cv_precision_test = {}
models_cv_precision_train = {}

[ ] from sklearn.metrics import confusion_matrix, recall_score, precision_score
from sklearn.model_selection import cross_validate, StratifiedKFold

def evaluate_classifier(model_name, model):
    Y_pred = model.predict(X_test)
    models_precision[model_name] = precision_score(y_test, Y_pred)

    kfold = StratifiedKFold(n_splits=5, shuffle=True)
    precision_cv = cross_validate(model, X_train_resampled, y_train_resampled, cv=kfold, scoring='precision', return_train_score=True)
    models_cv_precision_train[model_name] = str(precision_cv['train_score'].mean())
    models_cv_precision_test[model_name] = str(precision_cv['test_score'].mean())

    models_cm[model_name] = confusion_matrix(y_test, Y_pred)
```

- **Confusion matrix** memberikan gambaran rinci tentang klasifikasi yang benar dan kesalahan yang dibuat oleh model.
- **Precision** score akan kami jadikan **metric utama**, dimana ini mengukur kemampuan model dalam mengidentifikasi semua kasus positif dengan benar, metric ini berfokus untuk **meminimalisir** adanya nilai **false positive**.



Dengan menggunakan **precision**, diharapkan false positive akan berkurang, sehingga perusahaan tidak perlu mengeluarkan **biaya lebih besar** (misal biaya telepon, campaign, karyawan) pada nasabah yang tidak tertarik pada deposit berjangka.

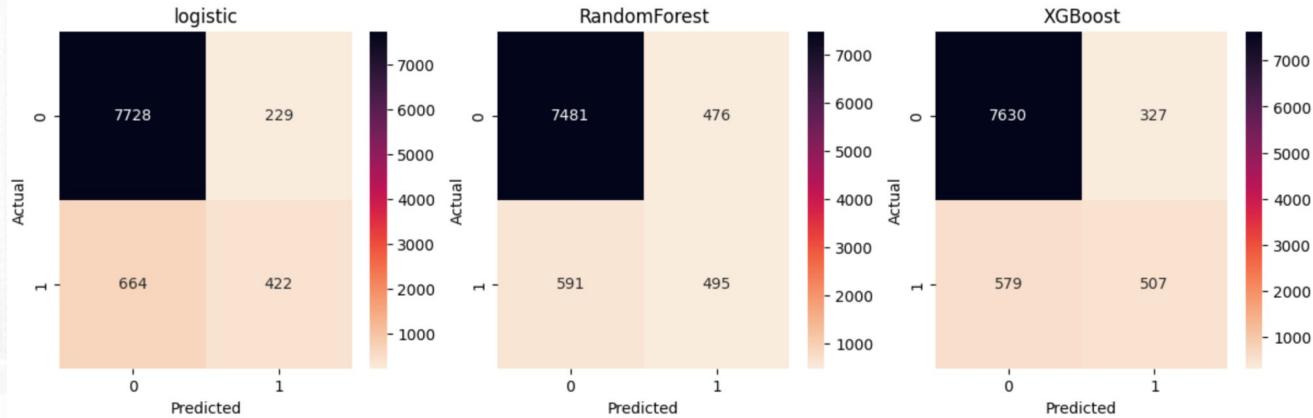
Kami **mengabaikan false negative** karena kami berasumsi bahwa nasabah yang tertarik akan langsung menghubungi kami tanpa kami harus melakukan campaign.

Modeling (Model Evaluation)

Model	Precision Score	Precision Cross-Val (Train)	Precision Cross-Val (Test)
logistic	0.648233	0.962009	0.961817
RandomForest	0.509784	0.998379	0.934569
XGBoost	0.607914	0.968101	0.948997

Dari ketiga algoritma yang telah dievaluasi, terdapat beberapa observasi penting yang perlu dipertimbangkan. Pertama, dalam hal precision untuk cross-validation (CV), **semua algoritma tidak menunjukkan tanda-tanda overfitting**, yang mengindikasikan bahwa model-model ini konsisten dalam kinerja mereka, khususnya, **Logistic Regression**. Dengan mempertimbangkan nilai precision score yang mencerminkan kemampuan model dalam menggambarkan kasus positif ('y'), **Logistic Regression** mengungguli dua algoritma lainnya, dengan nilai **0.648**.

Modeling (Model Evaluation)



Model logistic regression **memiliki kasus true negative cukup tinggi** daripada kedua model yang lain, walaupun ia **memprediksi true positif yang lebih rendah**. Hal tersebut dikarenakan Logistic Regression seringkali memiliki ambang batas yang lebih konservatif, yang berarti ia lebih cenderung untuk mengklasifikasikan pengamatan sebagai negatif dan berhati-hati saat membuat prediksi positif.

Namun, dalam hal **false positive**, logistic memiliki nilai paling rendah dari semua algoritma, sehingga cocok untuk minimalisir kesalahan prediksi benar.

Modeling (Hyperparameter Tuning)

Model	Precision Score	Precision Cross-Val (Train)	Precision Cross-Val (Test)
logistic	0.648233	0.962009	0.961817
RandomForest	0.509784	0.998379	0.934569
XGBoost	0.607914	0.968101	0.948997
Tuned_LogisticRegressor	0.643815	0.959408	0.959432

Setelah melakukan hyperparameter tuning, nilai precision mengalami sedikit **peningkatan namun tidak signifikan**. Oleh karena itu, kami mempertimbangkan untuk kembali menggunakan **model awal** dalam hal precision.

Modeling (Adjusting Threshold)

```

import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve

# Assuming you have already trained and evaluated your model
# Make predictions on the test data
y_pred_proba = model["logistic"].predict_proba(X_test)

# Adjusted threshold (choose the threshold that worked best for you)
adjusted_threshold = target_thres # Modify this threshold as needed

# Calculate precision, recall, and thresholds for the adjusted threshold
precision, recall, thresholds = precision_recall_curve(y_test, y_pred_proba[:, 1])

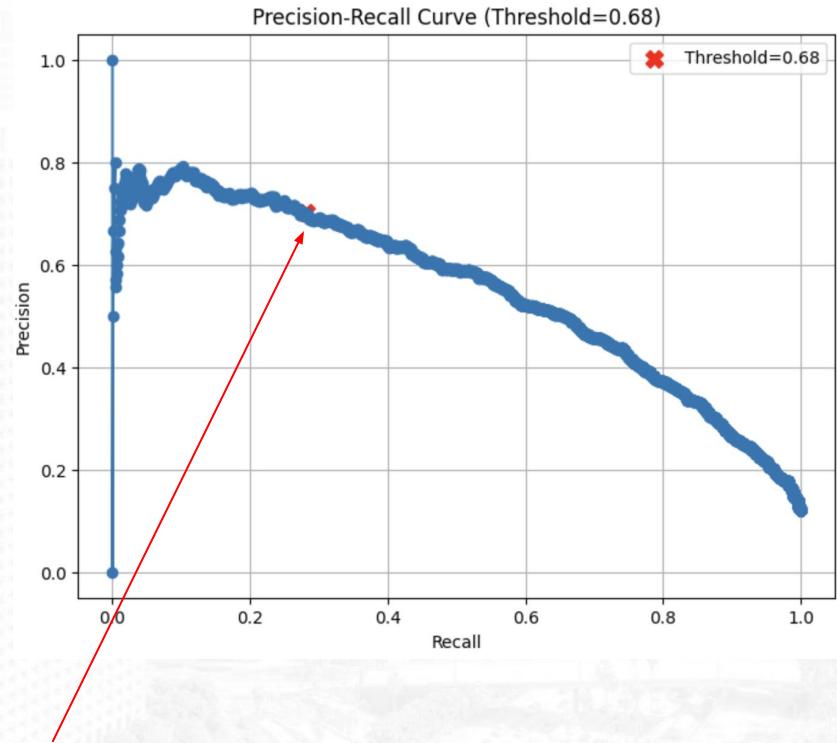
# Find the index of the threshold that is closest to the adjusted threshold
closest_threshold_index = (abs(thresholds - adjusted_threshold)).argmin()

# Plot the Precision-Recall Curve with the adjusted threshold
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, marker='o', linestyle='-' )
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title(f'Precision-Recall Curve (Threshold={adjusted_threshold:.2f})')

# Highlight the point on the curve corresponding to the adjusted threshold
plt.scatter(recall[closest_threshold_index], precision[closest_threshold_index], c='red', marker='x', s=100, label=f'Threshold={adjusted_threshold:.2f}')
plt.legend()
plt.grid(True)
plt.show()

```

Dalam rangka memaksimalkan precision, kami menggunakan **precision-recall curves**. Dimana kami **meningkatkan precision** dan menurunkan nilai recall dengan cara menyesuaikan nilai **threshold**.



Kami menggunakan Threshold dengan nilai **0.68**

Modeling (Adjusting Threshold)

	Precision	Recall	Threshold
8434	0.701149	0.280847	0.676661
8435	0.700461	0.279926	0.677893
8438	0.700696	0.278085	0.679010
8439	0.700000	0.277164	0.684032
8442	0.700234	0.275322	0.692501
...
8850	0.736842	0.012891	0.992239
8851	0.722222	0.011971	0.992279
8852	0.705882	0.011050	0.993717
8864	0.800000	0.003683	0.999512
8865	0.750000	0.002762	0.999771

412 rows × 3 columns

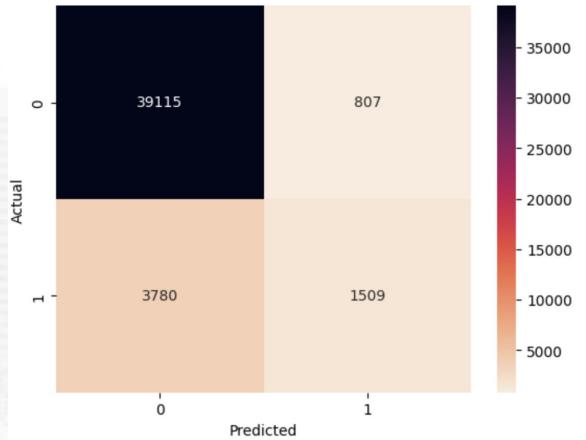
```
#pred_8 = np.where(Y_pred>0.676661, 1, 0)
pred_8 = np.where(Y_pred>target_thres, 1, 0)
precision_8 = precision_score(y_test, pred_8)
precision_8
```

0.7004608294930875

Dengan mencari nilai **precision 0.70**, kami menyesuaikan nilai **threshold sebesar 0.67661**. (yang tadinya default 0.5) Alhasil, nilai **precision naik menjadi 0.7004**.

Modeling Interpretation

Result Interpretation (Without Machine Learning)



INITIAL CONVERSION RATE

Conversion rate = (Jumlah Nasabah yang Berlangganan Secara Aktual / Jumlah Nasabah yang Dihubungi) * 100

$$\text{Conversion rate} = (5289 / 45211) * 100 = \sim 12\%$$

Hanya **12%** yang benar-benar berlangganan deposito berjangka.

AFTER ML CONVERSION RATE

Conversion rate = (Jumlah Nasabah yang **aktual** Berlangganan) / Jumlah Nasabah yang **diprediksi** berlangganan) * 100

$$\text{Conversion rate} = (1509 / 807+1509) * 100 = 0.6515 \sim 65\%$$

Berdasarkan prediksi model, sebanyak **65%** dari nasabah yang diprediksi benar-benar berlangganan deposito berjangka.

Result Interpretation (Cost efficiency)

Dari 45211 nasabah, total campaign atau contact yang dilakukan adalah sebanyak **124.956**. Jika asumsi menggunakan “lead cost”, maka biaya per contact adalah **25 pounds**. Kami mengasumsikan rata-rata pelanggan akan tertarik untuk deposito berjangka setelah dilakukan **2x campaign**.

Cost for Campaign **before ML**

$$124956 * \text{£}25 = \text{£} 3,123,900$$

Cost for Campaign **after ML**

$$(2316 \times 2) * \text{£}25 = \text{£} 115,800$$

Cost Reduction = **£ 3,008,100**

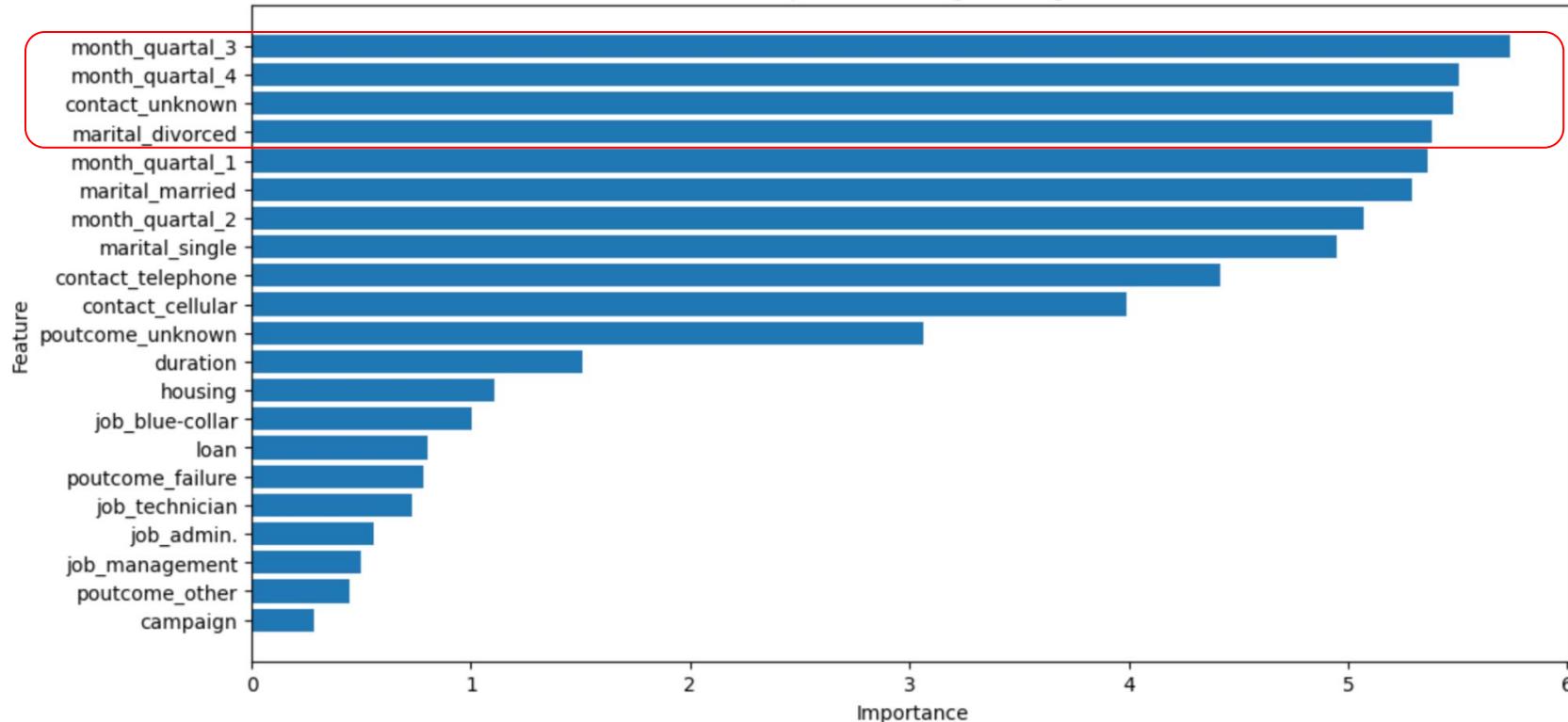
$$\text{Cost Reduction Percentage} = (\text{£}3,008,100 / \text{£}3,123,900) * 100 \approx 96.29\% \approx 96\%$$

Pengurangan biaya/Penghematan yang dicapai dengan menerapkan **Machine Learning** dalam campaign tersebut adalah sekitar **96%**. Hal ini menunjukkan pengurangan biaya yang signifikan dibandingkan dengan kampanye yang dilakukan tanpa bantuan model tersebut.

Feature Importance

Feature Importance

Feature Importances (Logistic Regression)



Insight

Month quartal 3 & 4 (semester 2)

- Semester 2 menjadi waktu yang **krusial dalam keputusan melakukan deposit**, hal ini bisa jadi karena semua pembayaran banyak selesai dilakukan di semester 1, sehingga semester 2 menyisakan uang yang biasanya **lebih senggang**
- Pada waktu ini juga, bank biasanya melakukan **penyesuaian** suku bunga dan juga banyak perusahaan merencanakan anggaran untuk tahun depan.

Contact Unknown

- Metode kontak atau campaign **diluar telemarketing** menjadi **krusial** untuk menentukan nasabah melakukan deposit atau tidak, mungkin karena **telemarketing** dinilai metode yang sudah **tradisional**

Marital Divorced

- Biasanya kategori ini merujuk kepada nasabah yang berusia **cukup dewasa** yang **telah melewati masa pernikahan**.
- Nasabah tipe ini memiliki keleluasaan dana karena banyak dari mereka yang hidup **sendiri**, sehingga bisa lebih baik spend untuk **invest perencanaan masa depan**.

Recommendation

Month quartal 3 & 4 (semester 2)

- Bank perlu melakukan **campaign masing-masing 1 kali** tiap nasabah pada **quartal 3 dan 4**
- Dalam campaign bisa menyesuaikan **promo liburan** (return lebih tinggi) dengan **pencairan** menjelang **akhir tahun** di quartal 4.

Contact Unknown

- Bank perlu mencoba metode lain seperti **email direct marketing** pada percobaan campaign yang **kedua** dan sebaiknya diinfokan dahulu di telemarketing percobaan pertama
- Bank bisa memberikan **promo khusus** berupa **minimum** nilai deposit yang lebih rendah jika **login ke website atau follow media sosial** bank yang dibagikan linknya melalui email marketing

Marital Divorced

- Selain promo liburan, bank juga perlu memberikan **program khusus** untuk **tabungan pendidikan anak dan hari tua**, karena mungkin banyak nasabah dari kategori ini yang merupakan janda memiliki anak dan mungkin janda tanpa tanggungan yang biasanya butuh tabungan pensiun untuk dirinya sendiri.

Thanks

