Predicting Marketing Campaign Response Using Logistic Regression

by Rabindranatah Duran Pons, Yasaman Eftekharypour, Valeria Siciliano, Rocco Lee

Summary

In this project, we developed a machine-learning pipeline using logistic regression to predict whether a customer will subscribe to a marketing campaign. The workflow combined a preprocessing stage (StandardScaler and OneHotEncoder) with a logistic regression classifier, followed by training and evaluation using a train/test split.

After applying class-weighting to address the dataset's imbalance, the model achieved an accuracy of approximately 85% and a ROC-AUC of ~0.91, indicating strong overall discrimination between the subscribed ("yes") and not-subscribed ("no") classes. Importantly, class-weighting significantly improved the model's ability to detect positive cases, giving the "yes" class a recall of 0.81. This shows that the weighted logistic regression approach is better suited for imbalanced marketing data, where correctly identifying potential subscribers is more valuable than simply maximizing accuracy.

Introduction

Background:

Financial institutions frequently rely on direct marketing campaigns to promote new products and services to potential clients. One common approach is telephone-based marketing, where bank representatives call customers to inform them about financial offerings, such as long-term deposits and attempt to persuade them to subscribe. Although this method can be effective, it is also resource-intensive calling uninterested or unlikely customers wastes time, labor, and operational costs. As a result, banks increasingly turn to data-driven decision-making to identify which individuals are most likely to respond positively to a campaign.

The Bank Marketing Dataset was created within this context. Compiled by researchers at the University of Minho in Portugal, it captures detailed information from a series of telephone marketing campaigns run by a Portuguese banking institution. The dataset includes demographic attributes (such as age, employment type, and marital status), financial indicators (such as credit defaults and loan status), and campaign-specific variables (including previous contact outcomes and call duration). The target variable indicates whether a customer ultimately subscribed to a term deposit, making the dataset a classic example of a binary classification problem.

The main question explored in this project is:

"Can we predict whether a bank customer will subscribe to a term deposit based on their demographic characteristics, financial information, and interactions with previous marketing campaigns?"

The dataset includes three main categories of features:

1. Client demographics and personal information

- age
- job
- marital
- education
- default (has credit in default)
- housing (has housing loan)
- loan (has personal loan)

2. Current campaign interaction

- contact — type of communication (cellular/telephone)
- day_of_week — day of contact
- month — month of campaign
- duration — call duration in seconds
- campaign — number of contacts during this campaign

3. Past campaign and historical interaction

- pdays — number of days since last contact
- previous — number of previous contacts
- poutcome — outcome of previous campaign

Methods

Data

The dataset used in this project is the Bank Marketing Dataset, created by Moro, Cortez, and Rita (2014) at the University of Minho in Portugal. The data was sourced from the UCI Machine Learning Repository, and can be accessed online at https://archive.ics.uci.edu/dataset/222/bank+marketing. Each row in the dataset represents a single customer contacted during a direct marketing phone campaign, and includes information such as demographic attributes, financial status, call details, previous campaign interactions, and the final outcome indicating whether the customer subscribed to a term deposit.

Analysis

A logistic regression model was used to predict whether a marketing campaign will be successful or not. All original variables from the dataset were included in the analysis. Before fitting, numerical features were standardized with a StandardScaler, and categorical variables were converted to binary indicators via OneHotEncoder. The dataset was split into 80% training and 20% testing, and class imbalance was addressed by balancing class weights during model training. The model's performance was evaluated using accuracy and ROC-AUC scores.

The code used to perform this analysis and generate the accompanying report can be found here:
https://github.com/Roccolee18/bank_marketing_group_24/blob/main/marketing_campain_pı

Results & Discussion

The logistic regression model developed for this analysis provides meaningful insight into the factors associated with customer subscription, but it also highlights the intrinsic challenges of modeling imbalanced marketing data. Our pipeline combined appropriate preprocessing steps—StandardScaler for numerical features and One-Hot Encoding for categorical variables—with a LogisticRegression classifier to ensure proper handling of the heterogeneous dataset while respecting the Golden Rule and avoiding data leakage.

The performance metrics indicate that the model performs reasonably well overall. The ROC-AUC score of approximately 0.91 suggests strong ability to distinguish between subscribers ("yes") and non-subscribers ("no"). Although overall accuracy is around 0.85, accuracy alone is not an appropriate metric for this imbalanced context, because the majority class dominates the dataset.

More importantly, the class-weighted logistic regression successfully shifts the model's focus toward the minority class. The recall for the "yes" class reaches 0.81, a substantial improvement compared to what a non-weighted model would typically achieve on an imbalanced dataset. This indicates that the model is able to identify most customers who eventually subscribe—an outcome that aligns with the core business objective, where failing to detect potential subscribers is far more costly than incorrectly flagging non-subscribers. The precision for the “yes” class is lower (0.42), which is an expected trade-off: by increasing recall and giving more weight to positive cases, the classifier becomes more permissive and produces more false positives. However, in a marketing context—where the cost of contacting an uninterested customer is low compared to the value of identifying a true potential subscriber—this trade-off is acceptable and strategically desirable.

The confusion matrix supports this interpretation. Out of 1,058 actual subscribers, the model correctly identifies 862 true positives while misclassifying 196 as non-subscribers. On the other hand, among the majority class, 6,786 non-subscribers are correctly classified, with 1,199 false positives. These numbers reflect a deliberate shift in the decision boundary due to class balancing: the model becomes more sensitive to the minority class at the expense of increasing false positives.

Overall, the balanced logistic regression model is appropriate for this business problem. Its ability to capture a large portion of true subscribers, even with lower precision, aligns with the strategic goal of maximizing successful marketing outreach. By prioritizing recall in the positive class, the model supports proactive customer engagement and provides a meaningful foundation for future marketing campaigns.

These results show that using a class-weighted logistic regression helps the model catch many more people who are likely to subscribe. This can be useful for marketing teams because it means they can focus their efforts on customers who are more likely to say “yes.” It also shows which factors—like the success of previous campaigns, the month of contact, or call duration—matter most, which can help improve how future campaigns are planned.

These results also bring up a number of future questions. For example, it is unclear whether another type of model, such as a tree-based method, could perform even better than logistic regression on this imbalanced data. Another question is whether the same patterns would appear if we ran this analysis on a different marketing campaign or a different time period. Finally, it would be useful to understand which types of customers the model tends to misclassify most often, and whether adding more customer information could help the model make more reliable predictions.

The following code reads the data programatically and saves it to the data folder:

In [1]:
```
from ucimlrepo import fetch_ucirepo
import pandas as pd
import altair as alt
import os
```

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    accuracy_score, classification_report,
    roc_auc_score, confusion_matrix
)
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import seaborn as sns

# Create the data folder if it doesn't exist
os.makedirs("data", exist_ok=True)

# fetch dataset
bank_marketing = fetch_ucirepo(id=222)

# Convert the data into a pd dataframe
X = bank_marketing.data.features
y = bank_marketing.data.targets

df = pd.concat([X, y], axis=1)

# Save combined dataset to data folder
df.to_csv("data/bank_marketing.csv", index=False)

print(df.shape)
print(df.head())
```

```
(45211, 17)
   age           job  marital  education default  balance housing loan  \
0   58    management  married   tertiary      no     2143     yes   no
1   44    technician   single  secondary      no       29     yes   no
2   33  entrepreneur  married  secondary      no        2     yes  yes
3   47   blue-collar  married        NaN      no     1506     yes   no
4   33           NaN   single        NaN      no        1      no   no

  contact  day_of_week month  duration  campaign  pdays  previous poutcome
y
0     NaN            5   may       261         1     -1         0      NaN
no
1     NaN            5   may       151         1     -1         0      NaN
no
2     NaN            5   may        76         1     -1         0      NaN
no
3     NaN            5   may        92         1     -1         0      NaN
no
4     NaN            5   may       198         1     -1         0      NaN
no
```

EDA

The following code performs some preliminary EDA:

In [2]:
```
df.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          44923 non-null  object
 2   marital      45211 non-null  object
 3   education    43354 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      32191 non-null  object
 9   day_of_week  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     8252 non-null   object
 16  y            45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

In [3]:
```
df.describe()
```

Out[3]:

	age	balance	day_of_week	duration	campaign	
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	4521
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	4
std	10.618762	3044.765829	8.322476	257.527812	3.098021	10
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	87

In [4]:
```
df.value_counts()
```

Out[4]:

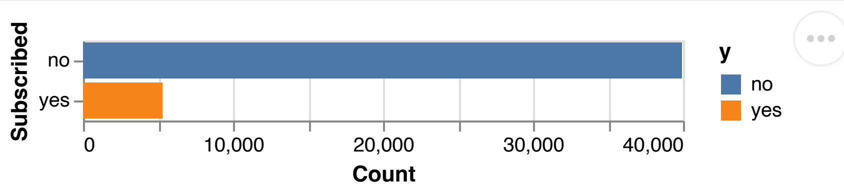
```
age  job          marital   education  default  balance  housing  loan  con
tact     day_of_week  month  duration  campaign  pdays  previous  poutcome
y
18   student      single    primary    no        608     no        no    cel
lular   13           nov    210        1         93     1         success
yes     1
44   blue-collar  married   secondary  no        6491    yes       no    cel
lular   19           nov    126        2         174    1         failure
no      1
                                                 3060    yes       no    cel
lular   20           apr    87         2         151    3         failure
no      1
                                                 2979    yes       no    cel
lular   18           may    18         5         178    3         other
no      1
                                                 1495    yes       no    cel
lular   1            oct    82         1         142    1         failure
no      1

..
34   management   single    tertiary   no        -444    yes       yes   cel
lular   17           apr    129        2         148    4         other
no      1
                  married   tertiary   no        8000    no        no    cel
lular   12           feb    291        1         260    2         failure
yes     1
                                                 5878    no        no    cel
lular   29           jan    159        7         98     3         success
no      1
                                                 4859    no        no    cel
lular   30           apr    113        1         37     1         other
no      1
89   retired      divorced  primary    no        1323    no        no    tel
ephone  29           dec    207        4         189    1         other
no      1
Name: count, Length: 7842, dtype: int64
```

The distribution of people who subscribed and didn't subscribe can be found below:

In [5]:

```
# Simplify working with large datasets in Altair
alt.data_transformers.enable('vegafusion')

alt.Chart(df).mark_bar().encode(
    y=alt.Y("y:N", title="Subscribed"),
    x=alt.X("count()", title="Count"),
    color="y:N"
)
```

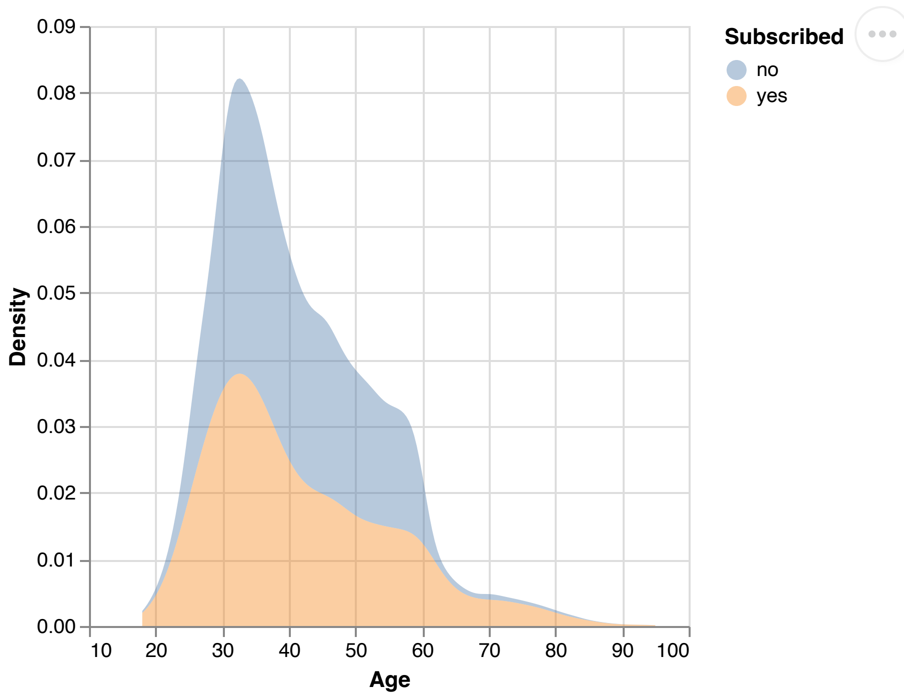
Out[5]:



A comparison of the distribution of subscribed people among age can be found below:

In [6]:
```
(
alt.Chart(df)
.transform_density(
    density="age",
    groupby=["y"],
    as_=["age", "density"]
)
.mark_area(opacity=0.4)
.encode(
    x=alt.X("age:Q", title="Age"),
    y=alt.Y("density:Q", title="Density"),
    color=alt.Color("y:N", title="Subscribed")
)
)
```

Out[6]:

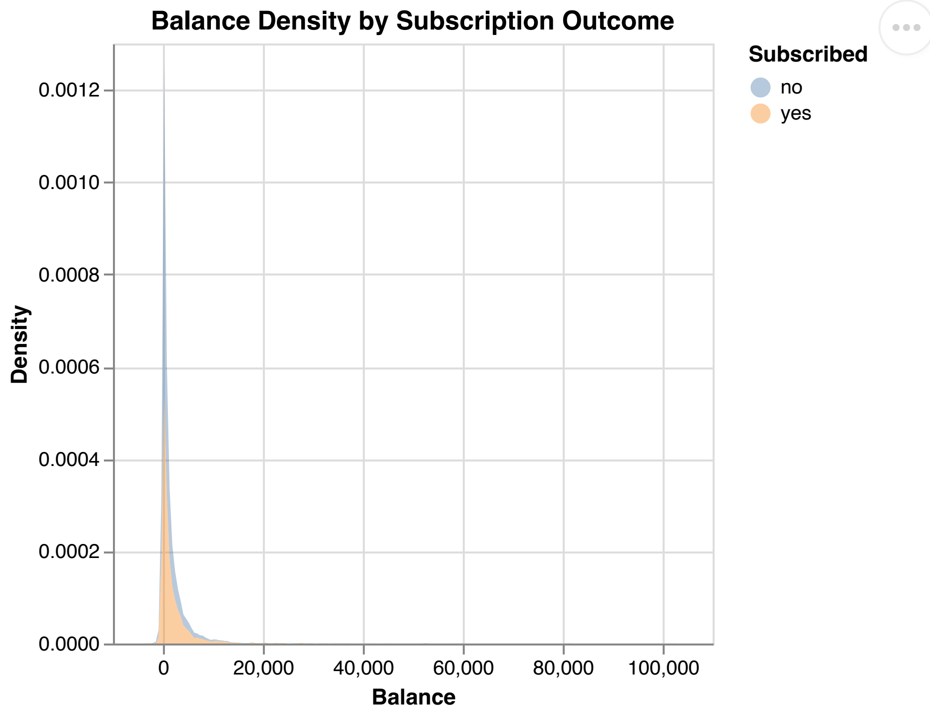


This shows that age can be a very good predictor for subsciption when used on its own.

In [7]:
```
(
alt.Chart(df)
.transform_density(
    density="balance",
    groupby=["y"],
    as_=["balance", "density"]
)
.mark_area(opacity=0.4)
.encode(
    x=alt.X("balance:Q", title="Balance"),
    y=alt.Y("density:Q", title="Density"),
    color=alt.Color("y:N", title="Subscribed")
)
```

```
.properties(title="Balance Density by Subscription Outcome")
)
```

Out[7]:



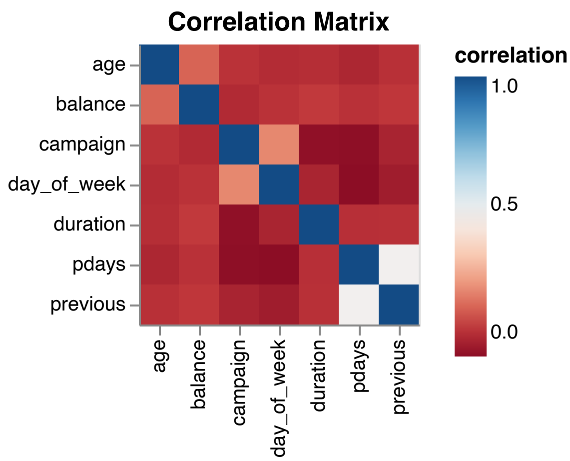
In the plot above we can see that the distribution of balance is right skwed and it might not be a good predictor on its own for subscription. However, when considering other variables, it might be useful.

The following shows a heatmap of the correlation among all the variables:

In [8]:

```
numeric_cols = df.select_dtypes(include='number').columns
corr_df = df[numeric_cols].corr().stack().reset_index()
corr_df.columns = ['var1', 'var2', 'correlation']

alt.Chart(corr_df).mark_rect().encode(
    x=alt.X('var1:N', title=""),
    y=alt.Y('var2:N', title=""),
    color=alt.Color('correlation:Q', scale=alt.Scale(scheme='redblue')),
    tooltip=['var1', 'var2', 'correlation']
).properties(
    title="Correlation Matrix"
)
```

Out[8]:



We can see that there is no multicolinearity among different variables.

In [9]:

```
print(df['education'].unique())
print(df['marital'].unique())
```

```
['tertiary' 'secondary' nan 'primary']
['married' 'single' 'divorced']
```

Preprocessing

In [10]:

```
df = df.dropna()
df = df[df['education'] != 'unknown']
df = df[df['job'] != 'unknown']
df = df[df['marital'] != 'unknown']

print(df.head())
print(df.info())
```

```
       age         job  marital  education default  balance housing loan  \
24060   33      admin.  married   tertiary      no      882      no   no
24062   42      admin.   single  secondary      no     -247     yes  yes
24064   33    services  married  secondary      no     3444     yes   no
24072   36  management  married   tertiary      no     2415     yes   no
24077   36  management  married   tertiary      no        0     yes   no

         contact  day_of_week month  duration  campaign  pdays  previous  \
24060  telephone           21   oct        39         1    151         3
24062  telephone           21   oct       519         1    166         1
24064  telephone           21   oct       144         1     91         4
24072  telephone           22   oct        73         1     86         4
24077  telephone           23   oct       140         1    143         3

      poutcome    y
24060  failure   no
24062    other  yes
24064  failure  yes
24072    other   no
24077  failure  yes
<class 'pandas.core.frame.DataFrame'>
Index: 7842 entries, 24060 to 45210
Data columns (total 17 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   age          7842 non-null   int64
 1   job          7842 non-null   object
 2   marital      7842 non-null   object
 3   education    7842 non-null   object
 4   default      7842 non-null   object
 5   balance      7842 non-null   int64
 6   housing      7842 non-null   object
 7   loan         7842 non-null   object
 8   contact      7842 non-null   object
 9   day_of_week  7842 non-null   int64
 10  month        7842 non-null   object
 11  duration     7842 non-null   int64
 12  campaign     7842 non-null   int64
 13  pdays        7842 non-null   int64
 14  previous     7842 non-null   int64
 15  poutcome     7842 non-null   object
 16  y            7842 non-null   object
dtypes: int64(7), object(10)
memory usage: 1.1+ MB
None
```

In [11]:
```python
# Target variable: y = "yes" or "no"
df["y"] = df["y"].map({"yes": 1, "no": 0})
df["housing"] = df["housing"].map({"yes": 1, "no": 0})
df["loan"] = df["loan"].map({"yes": 1, "no": 0})
```

In [12]:
```python
# 4. Split features and target


# Identify numerical and categorical columns
numerical_cols = X.select_dtypes(include=["int64", "float64"]).columns
```

```
categorical_cols = X.select_dtypes(include=["object"]).columns

print(numerical_cols)
print(categorical_cols)
```

```
Index(['age', 'balance', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous'],
      dtype='object')
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contac
t',
       'month', 'poutcome'],
      dtype='object')
```

In [13]:
```
# 5. Preprocessing pipeline
numeric_transformer = Pipeline(
    steps=[
    ("scaler", StandardScaler())
])

categorical_transformer = Pipeline(
    steps=[
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])

preprocessor = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, numerical_cols),
        ("cat", categorical_transformer, categorical_cols)
    ]
)
```

Fitting the model and making predictions

In [14]:
```
# 6. Build model pipeline
model = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("classifier", LogisticRegression(max_iter=1000, class_weight ="balancec
])
```

In [15]:
```
# 7. Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

In [16]:
```
# 8. Train model
model.fit(X_train, y_train)
```

```
/Users/rabin/miniforge3/envs/522/lib/python3.12/site-packages/sklearn/utils/
validation.py:1406: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ), for
example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[16]:

▸ **Pipeline**

▸ **preprocessor: ColumnTransformer**

▸ **num** ▸ **cat**

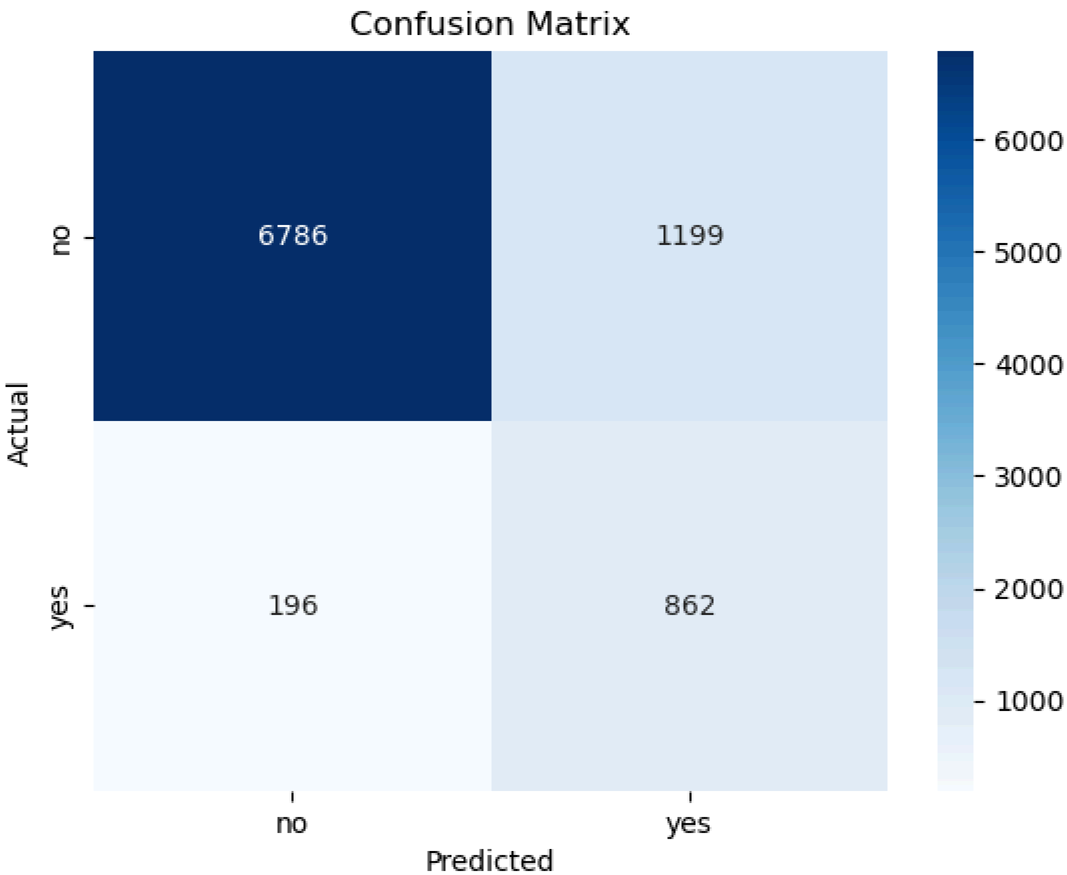
▸ StandardScaler ▸ OneHotEncoder

▸ LogisticRegression

In [17]:

```python
# 9. Predictions and evaluation
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]

print("Accuracy:", accuracy_score(y_test, y_pred))
print("ROC-AUC:", roc_auc_score(y_test, y_prob))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm,
            annot=True,
            fmt="d",
            cmap="Blues",
            xticklabels=["no", "yes"],
            yticklabels=["no", "yes"]
            )
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

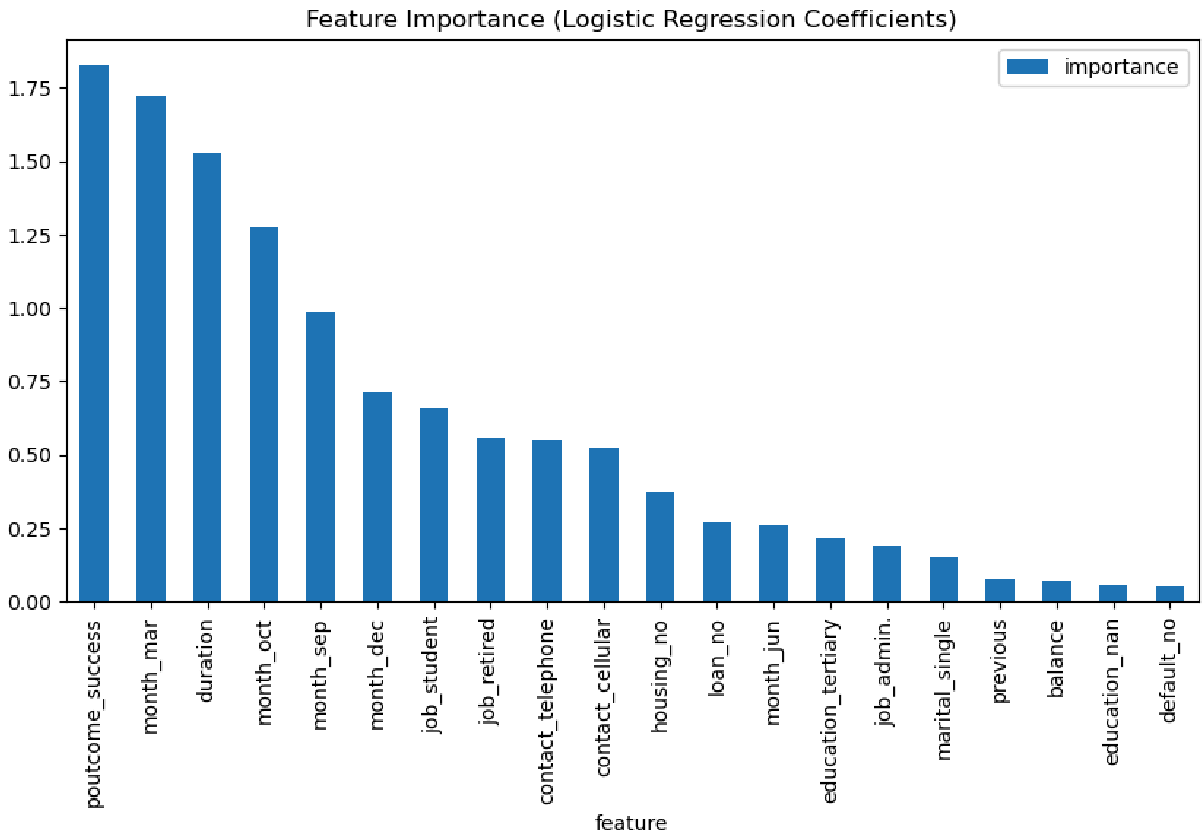
```
Accuracy: 0.8457370341700763
ROC-AUC: 0.9079218714674134

Classification Report:
               precision    recall  f1-score   support

          no       0.97      0.85      0.91      7985
         yes       0.42      0.81      0.55      1058

    accuracy                           0.85      9043
   macro avg       0.70      0.83      0.73      9043
weighted avg       0.91      0.85      0.87      9043
```



In [18]:

```
# 10. Feature Importance (for logistic regression)
# This is a bit tricky with pipelines — we extract processed feature names
ohe = model.named_steps["preprocessor"].named_transformers_["cat"]["onehot"]
cat_feature_names = ohe.get_feature_names_out(categorical_cols)
feature_names = np.concatenate([numerical_cols, cat_feature_names])

# Get coefficients
coeffs = model.named_steps["classifier"].coef_[0]

feat_imp = pd.DataFrame({
    "feature": feature_names,
    "importance": coeffs
}).sort_values(by="importance", ascending=False)

print(feat_imp.head(10))
feat_imp.head(20).plot(kind="bar", x="feature", y="importance", figsize=(10,
plt.title("Feature Importance (Logistic Regression Coefficients)")
plt.show()
```

	feature	importance
49	poutcome_success	1.824477
42	month_mar	1.722014
3	duration	1.525751
45	month_oct	1.276304
46	month_sep	0.984794
37	month_dec	0.713720
15	job_student	0.657311
12	job_retired	0.556467
33	contact_telephone	0.546549
32	contact_cellular	0.522132



References

A data modeling approach for classification problems: application to bank telemarketing prediction Stéphane Cédric KOUMETIO TEKOUABOU, Walid Cherif, H. Silkan Published in International Conferences on... 27 March 2019 Computer Science, Business https://www.semanticscholar.org/paper/A-data-modeling-approach-for-classification-to-bank-TEKOUABOU-Cherif/241d6ca92c4bc65ac3ee903e4732f70bff5c5e9f

Predicting the Accuracy for Telemarketing Process in Banks Using Data Mining F. Alsolami, Farrukh Saleem, A. Al-malaise, AL-Ghamdi, Published 2020 Business, Computer Science https://www.semanticscholar.org/paper/Predicting-the-Accuracy-for-Telemarketing-Process-Alsolami-Saleem/6391b7edcdd3c443bb57624b153bf9a8cca027db

Using Logistic Regression Model to Predict the Success of Bank Telemarketing Y. Jiang, Published 21 June 2018 Business, Computer Science, Journal of data science https://www.semanticscholar.org/paper/Using-Logistic-Regression-Model-to-Predict-the-of-Jiang/11ea58c843d0e745716d624b03067235dc285c30

Prediction of Term Deposit in Bank: Using Logistic Model Enjing Jiang, Zihao Wang, Jiaying Zhao, Published in BCP Business & Management 14 December 2022 Business, Computer Science https://www.semanticscholar.org/paper/Prediction-of-Term-Deposit-in-Bank%3A-Using-Logistic-Jiang-Wang/e36cafceaad636e9b2b558166c16be31a913ad0d