Predicting Whether Clients Will Subscribe To A Term Deposit Based On Their Information.

Authors: Godsgift Braimah, Devon Vorster, Chikire Aku-Ibe, & Fu Hung Kwong

In [1]:
```
import pandas as pd
import numpy as np
import altair_ally as aly
import altair as alt
from deepchecks.tabular.checks import (
    OutlierSampleDetection,
    IsSingleValue,
    StringMismatch,
    ClassImbalance,
    FeatureLabelCorrelation,
    FeatureFeatureCorrelation
)
from deepchecks.tabular import Dataset
from sklearn.model_selection import train_test_split, RandomizedSearch
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder, Ordin
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from scipy.stats import loguniform, randint, uniform
from sklearn.impute import SimpleImputer
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import pandera.pandas as pa
aly.alt.data_transformers.enable('vegafusion')
```

```
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/s
ite-packages/deepchecks/core/serialization/dataframe/html.py:16: UserWa
rning:

pkg_resources is deprecated as an API. See https://setuptools.pypa.io/e
n/latest/pkg_resources.html. The pkg_resources package is slated for re
moval as early as 2025-11-30. Refrain from using this package or pin to
Setuptools<81.
```

Out[1]:
```
DataTransformerRegistry.enable('vegafusion')
```

Summary:

We built a classification model using both the Logistic Regression and Support Vector Classifier (SVC) which can use the information related to the client and the marketing contact to predict whether a client will subscribe to a term deposit.

Our final classifier, the SVC, performed well on an unseen test data set, achieving a Test Score (Accuracy) of **0.9075** (compared to **0.88875** for Logistic Regression model). Its Train Score indicated a good fit without significant overfitting. Although the simple accuracy score does not detail the balance of True Positives versus False Negatives, an accuracy exceeding **80%** suggests the model is highly effective.

Given our goal is to increase the subscription rate, the model's primary goal is to minimize the False Negative rate, avoiding the error of predicting a client will not subscribe when they would have. The current performance suggests that using this model for initial client prioritization could significantly improve resource allocation, making the model valuable for immediate business implementation. However, further analysis of precision and recall would be necessary to optimize its practical utility.

Introduction:

Direct marketing campaigns, particularly those relying on phone calls, are a significant investment for banking institutions. The success of these campaigns is measured by the client subscription rate to a product like a term deposit. Another dataset shows that the subscription rate of term deposit in a Portuguese banking is only around 11.70% Ngu Hui En 2024, optimizing the targeting strategy is crucial to maximize return on investment and minimize operational costs.

Here we ask if a machine learning algorithm can be used to predict whether a client will subscribe to a term deposit based on information related to the client, such as type of job, education level; and also the marketing contact, e.g. number of contacts during the campaign, number of days since last contact. Answering this question is important because term deposit campaigns often require multiple contacts to the same client, making the process labor-intensive and expensive. Thus, if a machine learning algorithm can accurately and effectively predict client subscription, this could allow the bank to prioritize clients who are most likely to convert, leading to more efficient resource allocation and a higher overall

subscription rate, improving the campaign results.

Methods:

Data:

The data we used was obtained the UCI Machine Learning Repository which can be found here, specifically the Bank Marketing dataset of a Portuguese bank institution . The dataset contains various features about bank customers and whether they subscribed to a term deposit, an investment product offered by the bank (variable y). Each row in the dataset contains details of customers which was used to predict if they would subscribe to the term deposit or not. The original dataset contains 45211 records with 16 features and one target (17 columns). For the purpose of this analysis, we sampled 4,000 records from the original dataset to speed up the EDA and model training process.

Analysis:

We started this analysis by perfoming an exploratory data analysis (EDA) to understand the nature of the variables and their relationships. We observed some missing values in the dataset. We also observed that the target variable (y) was imbalanced with a higher proportion of customers not subscribing to the term deposit.

Furthermore, distribution plots for variables previous, pday, campaign, duration, balance were highly right-skewed. This implies that most customers had low values for these varaibles and a few customers had high values. In addition, the correlation plots in Figure 3 showed that "previous" and "pday" had the highest positive correlation.

We decided to use both Logistic Regression and Support Vector Classifier (SVC) models for this analysis.The sklearn package was greatly used in these processes. We performed hyperparameter tuning using Python RandomizedSearchCV to find the best parameters for each model. We mapped the values of the target y, using 'yes': 1, 'no': 0. The data was split using 80% for the training set and 20% for the test set. In the preprocessing, we dropped 'day_of_week' and 'pdays' because we considered them not relevant for analysis. The 'poutcome' variable was also dropped since it had a high number of missing values. Categorical variables were one-hot encoded, ordinal data were handled

using ordinal encoding, numerical variables were scaled using StandardScaler and missing values for the selected features were imputed using SimpleImputer with the "most frequent" strategy.

The models were evaluated based on their accuracy on the test set.

Loading The Data

In [2]:
```
## Download Data

from ucimlrepo import fetch_ucirepo

# fetch dataset
bank_marketing = fetch_ucirepo(id=222)

# data (as pandas dataframes)
X = bank_marketing.data.features
y = bank_marketing.data.targets

# metadata
print(bank_marketing.metadata)

# variable information
print(bank_marketing.variables)

# create complete dataset
bank_marketing_data =X; bank_marketing_data['y'] = y
#bank_marketing_data.to_csv('data/bank_marketing.csv')

# take a sample from teh data set
bank_marketing_sample = bank_marketing_data.sample(4000, random_state=
#bank_marketing_sample.to_csv('data/bank_marketing_small.csv')
```

```
{'uci_id': 222, 'name': 'Bank Marketing', 'repository_url': 'https://ar
chive.ics.uci.edu/dataset/222/bank+marketing', 'data_url': 'https://arc
hive.ics.uci.edu/static/public/222/data.csv', 'abstract': 'The data is
related with direct marketing campaigns (phone calls) of a Portuguese b
anking institution. The classification goal is to predict if the client
will subscribe a term deposit (variable y).', 'area': 'Business', 'task
s': ['Classification'], 'characteristics': ['Multivariate'], 'num_insta
nces': 45211, 'num_features': 16, 'feature_types': ['Categorical', 'Int
eger'], 'demographics': ['Age', 'Occupation', 'Marital Status', 'Educat
ion Level'], 'target_col': ['y'], 'index_col': None, 'has_missing_value
s': 'yes', 'missing_values_symbol': 'NaN', 'year_of_dataset_creation':
2014, 'last_updated': 'Fri Aug 18 2023', 'dataset_doi': '10.24432/C5K30
6', 'creators': ['S. Moro', 'P. Rita', 'P. Cortez'], 'intro_paper': {'I
D': 277, 'type': 'NATIVE', 'title': 'A data-driven approach to predict
the success of bank telemarketing', 'authors': 'Sérgio Moro, P. Cortez,
P. Rita', 'venue': 'Decision Support Systems', 'year': 2014, 'journal':
None, 'DOI': '10.1016/j.dss.2014.03.001', 'URL': 'https://www.semantics
```

cholar.org/paper/cab86052882d126d43f72108c6cb41b295cc8a9e', 'sha': Non
e, 'corpus': None, 'arxiv': None, 'mag': None, 'acl': None, 'pmid': Non
e, 'pmcid': None}, 'additional_info': {'summary': "The data is related
with direct marketing campaigns of a Portuguese banking institution. Th
e marketing campaigns were based on phone calls. Often, more than one c
ontact to the same client was required, in order to access if the produ
ct (bank term deposit) would be ('yes') or not ('no') subscribed. \n\nT
here are four datasets: \n1) bank-additional-full.csv with all examples
(41188) and 20 inputs, ordered by date (from May 2008 to November 201
0), very close to the data analyzed in [Moro et al., 2014]\n2) bank-add
itional.csv with 10% of the examples (4119), randomly selected from 1),
and 20 inputs.\n3) bank-full.csv with all examples and 17 inputs, order
ed by date (older version of this dataset with less inputs). \n4) bank.
csv with 10% of the examples and 17 inputs, randomly selected from 3 (o
lder version of this dataset with less inputs). \nThe smallest datasets
are provided to test more computationally demanding machine learning al
gorithms (e.g., SVM). \n\nThe classification goal is to predict if the
client will subscribe (yes/no) a term deposit (variable y).", 'purpos
e': None, 'funded_by': None, 'instances_represent': None, 'recommended_
data_splits': None, 'sensitive_data': None, 'preprocessing_descriptio
n': None, 'variable_info': 'Input variables:\n   # bank client data:\n
1 - age (numeric)\n   2 - job : type of job (categorical: "admin.","unk
nown","unemployed","management","housemaid","entrepreneur","student",\n
"blue-collar","self-employed","retired","technician","services") \n   3
- marital : marital status (categorical: "married","divorced","single";
note: "divorced" means divorced or widowed)\n   4 - education (categori
cal: "unknown","secondary","primary","tertiary")\n   5 - default: has c
redit in default? (binary: "yes","no")\n   6 - balance: average yearly
balance, in euros (numeric) \n   7 - housing: has housing loan? (binar
y: "yes","no")\n   8 - loan: has personal loan? (binary: "yes","no")\n
related with the last contact of the current campaign:\n   9 - contac
t: contact communication type (categorical: "unknown","telephone","cell
ular") \n 10 - day: last contact day of the month (numeric)\n 11 - mo
nth: last contact month of year (categorical: "jan", "feb", "mar", ...,
"nov", "dec")\n 12 - duration: last contact duration, in seconds (nume
ric)\n   # other attributes:\n 13 - campaign: number of contacts perfo
rmed during this campaign and for this client (numeric, includes last c
ontact)\n 14 - pdays: number of days that passed by after the client w
as last contacted from a previous campaign (numeric, -1 means client wa
s not previously contacted)\n 15 - previous: number of contacts perfor
med before this campaign and for this client (numeric)\n 16 - poutcom
e: outcome of the previous marketing campaign (categorical: "unknown","
other","failure","success")\n\n Output variable (desired target):\n 1
7 - y - has the client subscribed a term deposit? (binary: "yes","no")\
n', 'citation': None}}

	name	role	type	demographic	\
0	age	Feature	Integer	Age	
1	job	Feature	Categorical	Occupation	
2	marital	Feature	Categorical	Marital Status	
3	education	Feature	Categorical	Education Level	
4	default	Feature	Binary	None	
5	balance	Feature	Integer	None	

```
6       housing  Feature       Binary             None
7          loan  Feature       Binary             None
8       contact  Feature  Categorical             None
9   day_of_week  Feature         Date             None
10        month  Feature         Date             None
11     duration  Feature      Integer             None
12     campaign  Feature      Integer             None
13        pdays  Feature      Integer             None
14     previous  Feature      Integer             None
15     poutcome  Feature  Categorical             None
16            y   Target       Binary             None

                                                description  units missing_va
lues
0                                                     None   None
no
1   type of job (categorical: 'admin.','blue-colla...   None
no
2   marital status (categorical: 'divorced','marri...   None
no
3   (categorical: 'basic.4y','basic.6y','basic.9y'...   None
no
4                              has credit in default?   None
no
5                              average yearly balance  euros
no
6                                   has housing loan?   None
no
7                                  has personal loan?   None
no
8   contact communication type (categorical: 'cell...   None
yes
9                       last contact day of the week   None
no
10  last contact month of year (categorical: 'jan'...   None
no
11   last contact duration, in seconds (numeric). ...   None
no
12  number of contacts performed during this campa...   None
no
13  number of days that passed by after the client...   None
yes
14  number of contacts performed before this campa...   None
no
15  outcome of the previous marketing campaign (ca...   None
yes
16          has the client subscribed a term deposit?   None
no
```

Data Validation

During our data valuation, we checked the following:

- Correct data file format
- Correct column names
- No empty observations
- Missingness not beyond expected threshold
- Correct data types in each column
- No duplicate observations
- No outlier or anomalous values
- Correct category levels (i.e., no string mismatches or single values)
- Target/response variable follows expected distribution
- No anomalous correlations between target/response variable and features/explanatory variables
- No anomalous correlations between features/explanatory variables

After our checks, we observed that our data had the correct data file format, correct column names, no empty observations, no missingness beyond expected threshold, correct data types in each column, and no duplicate observations.

In [3]:
```
# validate data code adapted from Tiffany A. Timbers, Joel Ostblom & M
#Check for correct data file format:
def assert_csv_format(file_path):
    assert file_path.endswith(".csv"), "Error: File must be a CSV."
    return True

#Check for correct column names, correct data types in each column, no
schema = pa.DataFrameSchema(
    {
        "age": pa.Column(int, pa.Check.between(18, 91), nullable=True)
        "job": pa.Column(object, pa.Check.isin(['technician', 'blue-co
       'management', 'self-employed', 'retired', 'services', 'unemploy
       'housemaid', 'student']), nullable=True),
        "marital": pa.Column(object, pa.Check.isin(['married', 'single
        "education": pa.Column(object, pa.Check.isin(['tertiary', 'sec
        "default": pa.Column(object, pa.Check.isin(['no', 'yes']), nul
        "balance": pa.Column(int, pa.Check.between(-10000000, 10000000
        "housing": pa.Column(object, pa.Check.isin(['yes', 'no']), nul
        "loan": pa.Column(object, pa.Check.isin(['no', 'yes']), nullab
        "contact": pa.Column(object, pa.Check.isin(['cellular', 'telep
        "day_of_week": pa.Column(int, pa.Check.between(1, 31), nullabl
        "month": pa.Column(object, pa.Check.isin(['feb', 'nov', 'jul',
       'oct', 'sep', 'dec']), nullable=True),
        "duration": pa.Column(int, pa.Check.between(0, 4000), nullable
        "campaign": pa.Column(int, pa.Check.between(0, 40), nullable=T
        "pdays": pa.Column(int, pa.Check.between(-1, 800), nullable=Tr
        "previous": pa.Column(int, pa.Check.between(0, 30), nullable=T
        "poutcome": pa.Column(object, pa.Check.isin(['failure', 'succe
        "y": pa.Column(object, pa.Check.isin(['no', 'yes']), nullable=
    },
    #check for no duplicate observations, no empty observations and mi
```

```
    checks=[
        pa.Check(lambda df: ~df.duplicated().any(), error="Duplicate r
        pa.Check(lambda df: ~(df.isna().all(axis=1)).any(), error="Emp
        pa.Check(lambda df: (df.isna().mean() < 0.10).any(), error="Mi
    ]
)
assert_csv_format("data/data_marketing_sample.csv")
schema.validate(bank_marketing_sample, lazy=True)
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan
30577	42	technician	married	tertiary	no	1397	yes	no
30048	51	blue-collar	married	secondary	no	362	yes	yes
24147	23	admin.	single	secondary	no	40	no	no
25584	57	entrepreneur	married	tertiary	no	1828	yes	no
15822	47	management	divorced	tertiary	no	2105	no	yes
...	...	...	...	...	...	...	...	...
15271	41	management	single	tertiary	no	0	yes	no
22396	31	management	single	tertiary	no	10215	no	no
21330	60	self-employed	married	tertiary	no	3	no	no
36012	43	self-employed	married	tertiary	yes	310	yes	no
9839	54	entrepreneur	married	secondary	no	0	no	no

4000 rows × 17 columns

EDA

In [4]:
```
# split data
train_df, test_df = train_test_split(
    bank_marketing_sample, test_size=0.2,
    stratify=bank_marketing_sample['y'],
    random_state=522)
```

In [5]:
```
(train_df.isnull().sum() / train_df.shape[0]) * 100
```

Out[5]:
```
age             0.00000
job             0.84375
marital         0.00000
education       3.68750
default         0.00000
balance         0.00000
housing         0.00000
loan            0.00000
contact        30.28125
day_of_week     0.00000
month           0.00000
duration        0.00000
campaign        0.00000
pdays           0.00000
previous        0.00000
poutcome       83.50000
y               0.00000
dtype: float64
```

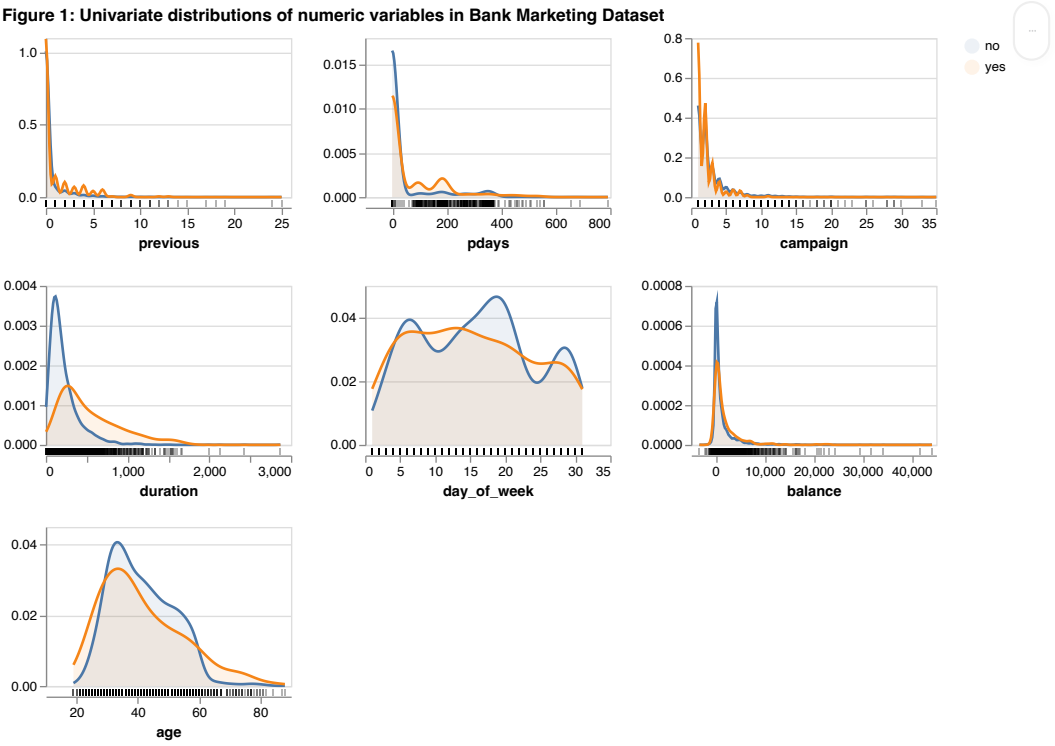
In [6]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3200 entries, 16093 to 8432
Data columns (total 17 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   age          3200 non-null   int64 
 1   job          3173 non-null   object
 2   marital      3200 non-null   object
 3   education    3082 non-null   object
 4   default      3200 non-null   object
 5   balance      3200 non-null   int64 
 6   housing      3200 non-null   object
 7   loan         3200 non-null   object
 8   contact      2231 non-null   object
 9   day_of_week  3200 non-null   int64 
 10  month        3200 non-null   object
 11  duration     3200 non-null   int64 
 12  campaign     3200 non-null   int64 
 13  pdays        3200 non-null   int64 
 14  previous     3200 non-null   int64 
 15  poutcome     528 non-null    object
 16  y            3200 non-null   object
dtypes: int64(7), object(10)
memory usage: 450.0+ KB
```

In [7]:
```
numeric_plot = aly.dist(train_df, color='y')

numeric_plot.properties(
    title="Figure 1: Univariate distributions of numeric variables in
)
```

Out[7]:



In [8]:

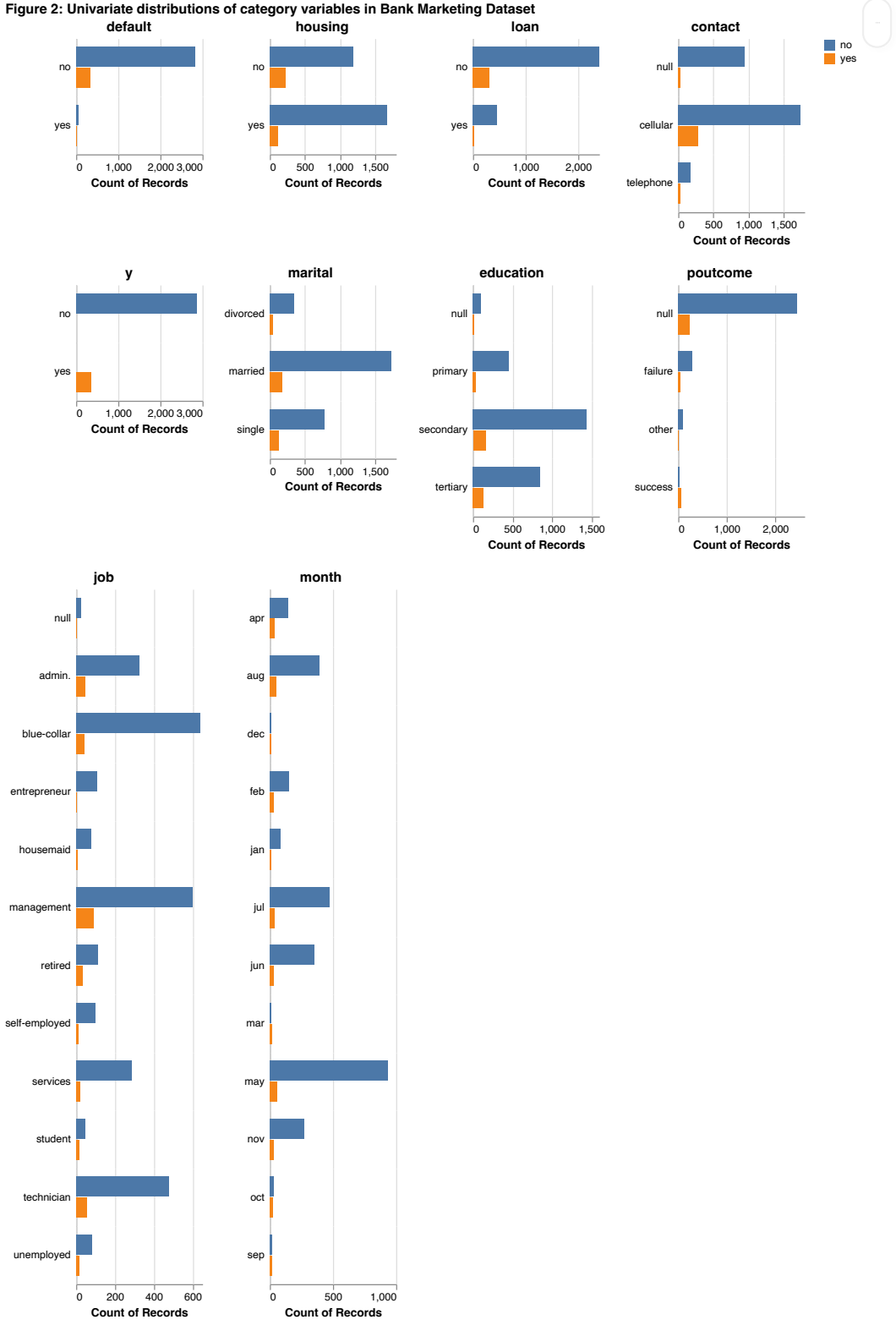
```
# univariate distrbutions (counts) for the categorical variables

categorical_plots = aly.dist(
    train_df, dtype='object', color='y')

categorical_plots.properties(
    title="Figure 2: Univariate distributions of category variables in
)
```

Out[8]:

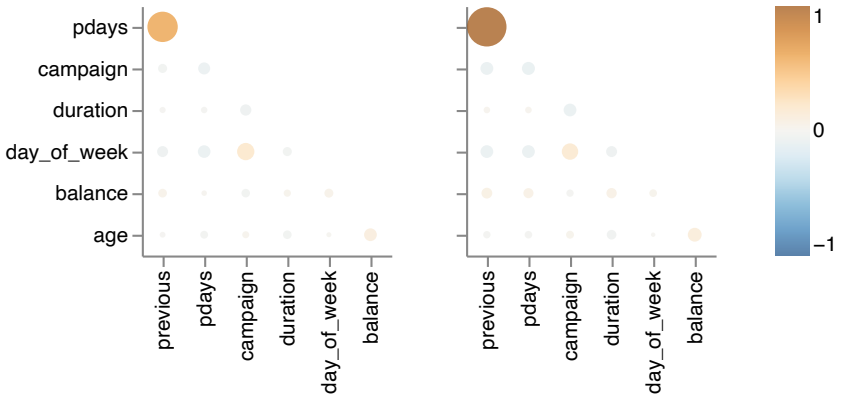
Figure 2: Univariate distributions of category variables in Bank Marketing Dataset



In [9]:
```
correlation_plots = aly.corr(bank_marketing_sample)
correlation_plots.properties(
    title="Figure 3: Correlation Plots For Numeric variables in Bank M
)
```

Out[9]:

Figure 3: Correlation Plots For Numeric variables in Bank Marketing Dataset



Data Preprocessing, Model Building and Evaluation

In [10]:
```
train_df.head()
```

Out[10]:

	age	job	marital	education	default	balance	housing	loan
16093	27	blue-collar	married	primary	no	237	yes	no
14526	29	entrepreneur	married	secondary	no	926	yes	no
19761	51	blue-collar	married	primary	no	507	yes	no
20123	34	management	divorced	tertiary	no	4231	yes	no
17548	59	unemployed	married	primary	no	46	no	no

In [11]:
```
# preprocessing

# map the target variable to numeric
train_df['y'] = train_df['y'].map({'yes': 1, 'no': 0})
test_df['y'] = test_df['y'].map({'yes': 1, 'no': 0})


# feature engineering on 'pdays' column into categorical determining i
train_df['pdays_contacted'] = train_df['pdays'].apply(lambda x: 'never
test_df['pdays_contacted'] = test_df['pdays'].apply(lambda x: 'never'


# Drop columns that are not needed from EDA: poutcome has 83% missing
drop_cols = ['day_of_week', 'pdays', 'poutcome']
train_df = train_df.drop(columns=drop_cols)
test_df = test_df.drop(columns=drop_cols)

# split data
X_train = train_df.drop(columns='y')
X_test = test_df.drop(columns='y')
y_train = train_df['y']
```

```
y_test = test_df['y']
```

In [12]:
```
# Rename target column for Deepchecks
train_df.rename(columns={'y': 'target'}, inplace=True)
test_df.rename(columns={'y': 'target'}, inplace=True)

# create Deepchecks Dataset
X_train_ds = Dataset(train_df, label="target", cat_features=['job', 'm
       'month', 'pdays_contacted'])

# Outlier Detection
check_outliers = OutlierSampleDetection()
check_outliers.add_condition_outlier_ratio_less_or_equal(0.05)
result_outliers = check_outliers.run(X_train_ds)

# Single Value Check
check_single_val = IsSingleValue()
result_single_val = check_single_val.run(X_train_ds)

# String Mismatch Check
check_string_mismatch = StringMismatch()
result_string_mismatch = check_string_mismatch.run(X_train_ds)

# Class Imbalance Check
check_imbalance = ClassImbalance()
check_imbalance.add_condition_class_ratio_less_than(0.99)
result_imbalance = check_imbalance.run(X_train_ds)

result_checks = {
              'Outliers': result_outliers,
              'Single Value': result_single_val,
              'String Mismatch': result_string_mismatch,
              'Class Imbalance': result_imbalance
       }

for name, result in result_checks.items():
       if not result.passed_conditions():
              raise ValueError(f"Check '{name}' failed!!")
       else:
              print(f"Check '{name}' passed.")
```

```
Check 'Outliers' passed.
Check 'Single Value' passed.
Check 'String Mismatch' passed.
Check 'Class Imbalance' passed.
```

In [13]:
```
# create Deepchecks Dataset for Test Data
X_test_ds = Dataset(test_df, label="target", cat_features=['job', 'mar
       'month', 'pdays_contacted'])

# Outlier Detection
check_outliers = OutlierSampleDetection()
```

```
check_outliers.add_condition_outlier_ratio_less_or_equal(0.05)
result_outliers = check_outliers.run(X_test_ds)

# Single Value Check
check_single_val = IsSingleValue()
result_single_val = check_single_val.run(X_test_ds)

# String Mismatch Check
check_string_mismatch = StringMismatch()
result_string_mismatch = check_string_mismatch.run(X_test_ds)

# Class Imbalance Check
check_imbalance = ClassImbalance()
check_imbalance.add_condition_class_ratio_less_than(0.99)
result_imbalance = check_imbalance.run(X_test_ds)

result_checks_test = {
            'Outliers': result_outliers,
            'Single Value': result_single_val,
            'String Mismatch': result_string_mismatch,
            'Class Imbalance': result_imbalance
      }

for name, result in result_checks_test.items():
      if not result.passed_conditions():
          raise ValueError(f"Check '{name}' failed!!")
      else:
          print(f"Check '{name}' passed.")
```

```
Check 'Outliers' passed.
Check 'Single Value' passed.
Check 'String Mismatch' passed.
Check 'Class Imbalance' passed.
```

In [14]:
```
# separating columns by type of transformation required

# One-hot encoding
categorical_cols = ['job', 'marital', 'default', 'housing', 'loan', 'c
# Ordinal encoding
ordinal_cols = ['education']
# Standard scaling
numerical_cols = ['age', 'balance', 'duration', 'campaign', 'previous'
```

We decided to train both a Logistic Regression and Support Vector Classifier (SVC) to determine which was more efficient in predicting if a customer would subscribed to the banks offering of term investments. We performed hyperparameter tuning using RandomizedSearchCV to find the best parameters for each model. The models were evaluated based on their accuracy on the test set.

In [15]:
```
# defining the preprocessor

data_preprocessor = make_column_transformer(
    (
        make_pipeline(SimpleImputer(strategy='most_frequent'),
                      OneHotEncoder(handle_unknown='ignore')),
                      categorical_cols
    ), (
        make_pipeline(SimpleImputer(strategy='most_frequent'),
                      OrdinalEncoder(categories=[['unknown', 'primary'
                      ordinal_cols
    ), (
        StandardScaler(), numerical_cols
        )
    )
```

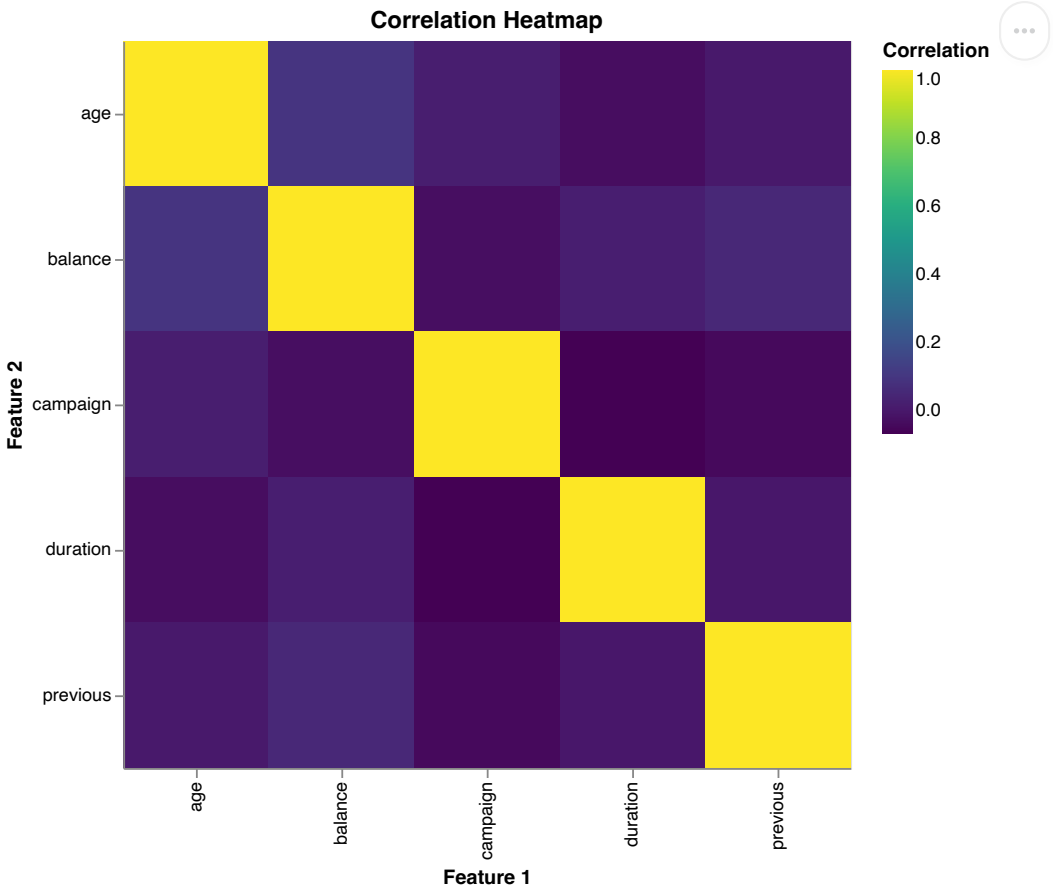
In [16]:
```
# Code adapted from from Tiffany A. Timbers, Joel Ostblom & Melissa Le
data_preprocessor.fit(X_train)
scaled_X_train = data_preprocessor.transform(X_train)

col_names = data_preprocessor.named_transformers_['pipeline-1'].get_fe
scaled_X_train_df = pd.DataFrame(scaled_X_train, columns=col_names)


correlation_matrix = scaled_X_train_df[numerical_cols].corr()
correlation_long = correlation_matrix.reset_index().melt(id_vars='inde
correlation_long.columns = ['Feature 1', 'Feature 2', 'Correlation']

alt.Chart(correlation_long).mark_rect().encode(
    x='Feature 1:O',
    y='Feature 2:O',
    color=alt.Color('Correlation:Q', scale=alt.Scale(scheme='viridis')
    tooltip=['Feature 1', 'Feature 2', 'Correlation']
).properties(
    width=400,
    height=400,
    title="Correlation Heatmap"
)
```

Out[16]:



In [17]:
```
# Code adapted from from Tiffany A. Timbers, Joel Ostblom & Melissa Le
# Data validation checks: feature-target and feature-feature correlati

# Check feature-label correlations
check_feat_lab_corr = FeatureLabelCorrelation().add_condition_feature_
check_feat_lab_corr_result = check_feat_lab_corr.run(dataset=X_train_d

# Check feature-feature correlations
check_feat_feat_corr = FeatureFeatureCorrelation().add_condition_max_n
check_feat_feat_corr_result = check_feat_feat_corr.run(dataset=X_train

if not check_feat_lab_corr_result.passed_conditions():
    raise ValueError("Feature-Label correlation exceeds the maximum ac

if not check_feat_feat_corr_result.passed_conditions():
    raise ValueError("Feature-feature correlation exceeds the maximum
```

In [18]:
```
# Logistic Regression cross-validation with RandomizedSearchCV
lr_pipe = make_pipeline(data_preprocessor, LogisticRegression(random_s
param_dist1 = {"logisticregression__C": loguniform(1e-4, 1e3)}
random_lr = RandomizedSearchCV(lr_pipe, param_distributions=param_dist
                               n_iter=100, n_jobs=-1, return_train_sc
```

In [19]:
```
# Fit model
random_lr.fit(X_train, y_train)
print(f'Train Score: {random_lr.score(X_train, y_train)}')
```

```
print(f'Test Score: {random_lr.score(X_test, y_test)}')
```

```
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/m
ultiprocessing/queues.py:122: UserWarning: pkg_resources is deprecated
as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html.
The pkg_resources package is slated for removal as early as 2025-11-30.
Refrain from using this package or pin to Setuptools<81.
  return _ForkingPickler.loads(res)
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/m
ultiprocessing/queues.py:122: UserWarning: pkg_resources is deprecated
as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html.
The pkg_resources package is slated for removal as early as 2025-11-30.
Refrain from using this package or pin to Setuptools<81.
  return _ForkingPickler.loads(res)
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/m
ultiprocessing/queues.py:122: UserWarning: pkg_resources is deprecated
as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html.
The pkg_resources package is slated for removal as early as 2025-11-30.
Refrain from using this package or pin to Setuptools<81.
  return _ForkingPickler.loads(res)
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/m
ultiprocessing/queues.py:122: UserWarning: pkg_resources is deprecated
as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html.
The pkg_resources package is slated for removal as early as 2025-11-30.
Refrain from using this package or pin to Setuptools<81.
  return _ForkingPickler.loads(res)
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/m
ultiprocessing/queues.py:122: UserWarning: pkg_resources is deprecated
as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html.
The pkg_resources package is slated for removal as early as 2025-11-30.
Refrain from using this package or pin to Setuptools<81.
  return _ForkingPickler.loads(res)
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/m
ultiprocessing/queues.py:122: UserWarning: pkg_resources is deprecated
as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html.
The pkg_resources package is slated for removal as early as 2025-11-30.
Refrain from using this package or pin to Setuptools<81.
  return _ForkingPickler.loads(res)
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/m
ultiprocessing/queues.py:122: UserWarning: pkg_resources is deprecated
as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html.
The pkg_resources package is slated for removal as early as 2025-11-30.
Refrain from using this package or pin to Setuptools<81.
  return _ForkingPickler.loads(res)
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/m
ultiprocessing/queues.py:122: UserWarning: pkg_resources is deprecated
as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html.
The pkg_resources package is slated for removal as early as 2025-11-30.
Refrain from using this package or pin to Setuptools<81.
  return _ForkingPickler.loads(res)
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/m
ultiprocessing/queues.py:122: UserWarning: pkg_resources is deprecated
as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html.
```

```
The pkg_resources package is slated for removal as early as 2025-11-30.
Refrain from using this package or pin to Setuptools<81.
  return _ForkingPickler.loads(res)
/Users/chikire/miniforge3/envs/term-deposit-classifier/lib/python3.11/m
ultiprocessing/queues.py:122: UserWarning: pkg_resources is deprecated
as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html.
The pkg_resources package is slated for removal as early as 2025-11-30.
Refrain from using this package or pin to Setuptools<81.
  return _ForkingPickler.loads(res)
Train Score: 0.9103125
Test Score: 0.88875
```

In [20]:
```
# SVC cross-validation with RandomizedSearchCV
svc_pipe = make_pipeline(data_preprocessor, SVC(random_state=42))
param_dist = { "svc__C": loguniform(1e-2, 1e3), "svc__gamma": logunifo
random_svc = RandomizedSearchCV(svc_pipe, param_distributions=param_di
                                n_iter=100, n_jobs=-1, return_train_sc
```

In [21]:
```
# Fit model
#The random_svc.fit() takes a while to run.
random_svc.fit(X_train, y_train)
print(f'Train Score: {random_svc.score(X_train, y_train)}')
print(f'Test Score: {random_svc.score(X_test, y_test)}')
```

```
Train Score: 0.9490625
Test Score: 0.9075
```

In [22]:
```
#Confusion Matrix for Logistic Regression model
ConfusionMatrixDisplay.from_estimator(
    random_lr,
    X_test,
    y_test,
    values_format="d",
);
plt.title("Figure 4: Confusion Matrix for Logistic Regression model")
plt.show()
```