

# CT1 Individual Assignment (MLOps)

End to End MLOps Workflow for Bank Term Deposit Marketing

Submitted by: **AJEET KUMAR**

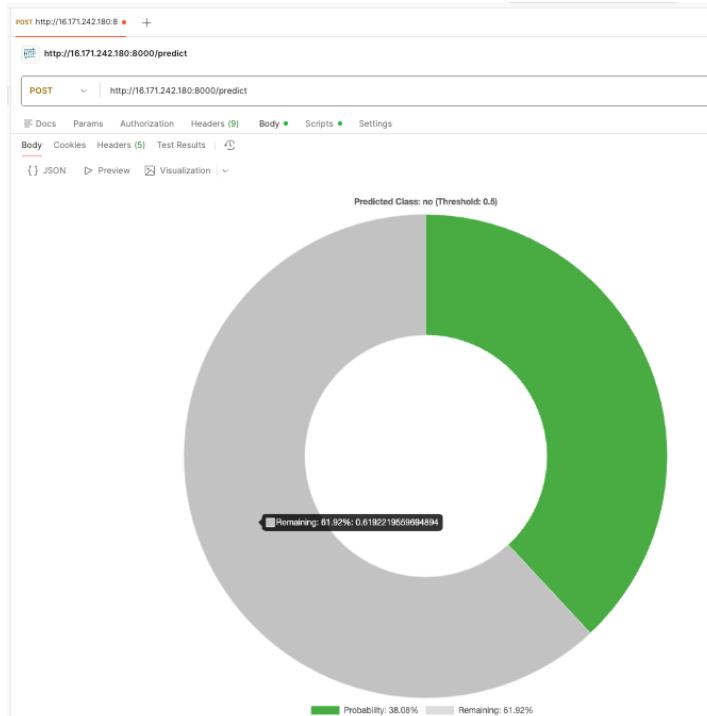
**12510040**  
AMPBA  
16<sup>th</sup> Dec 2025

## 1. Business Problem and Objective

- Briefly describe the bank's marketing context
  - The bank runs outbound campaigns to sell term deposits to customers.
  - Contacting every customer is costly. Only a small fraction accepts the offer.
- State the business objective clearly
  - Build a prediction model that identifies customers who are most likely to subscribe to a term deposit.
  - Use this model to prioritize outreach and improve campaign efficiency.

Target variable:

- $y = 1$  if customer subscribes to term deposit
- $y = 0$  otherwise



---

## 2. Dataset and Modelling Approach

### 2.1 Dataset Overview

- Mention dataset name and size
  - 4119 observations
  - 21 columns (20 input features plus target y)
- Feature groups
  - Client and demographic attributes (age, job, marital, education, etc.)
  - Campaign interaction variables (duration, campaign, previous, contact, month, day\_of\_week, poutcome)
  - Macroeconomic indicators (emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m, nr.employed)

### 2.2 Model and Preprocessing

- Algorithm
  - **Logistic Regression** for binary classification
- Preprocessing pipeline
  - Numeric features scaled using StandardScaler
  - Categorical features encoded using OneHotEncoder with handle\_unknown set to ignore
- Handling class imbalance
  - Used class\_weight balanced in Logistic Regression to give more weight to the minority positive class

### 2.3 Key Evaluation Metrics

- Report test set metrics from the notebook
  - ROC AUC
  - Precision, recall, F1 for both classes, with focus on the positive class (y = 1)
  - Confusion matrix to show trade off between true positives and false positives
- One short interpretation
  - For example, “The model achieves a ROC AUC of **0.9423**. This indicates good ranking ability between converters and non converters and makes the model suitable for prioritising leads in marketing campaigns.”

```
... ROC AUC: 0.9423

Classification report:
      precision    recall  f1-score   support
          0       0.977     0.880     0.926      734
          1       0.460     0.833     0.593       90

      accuracy                           0.875      824
     macro avg       0.719     0.857     0.760      824
  weighted avg       0.921     0.875     0.890      824

Confusion matrix:
[[646  88]
 [ 15  75]]
```

---

### 3. MLOps Pipeline Implementation

#### 3.1 Local Model Development

- Tools
  - Python and Jupyter notebook
  - scikit learn for modeling, joblib for saving the model
- Steps
  - Loaded bank-additional.csv
  - Performed basic EDA
  - Defined numeric and categorical feature lists
  - Built preprocessing plus Logistic Regression pipeline
  - Trained and evaluated the model
  - Saved the model as bank\_term\_deposit\_model.joblib

---

▶ # Train the model and evaluate

```
clf.fit(X_train, y_train)

y_proba = clf.predict_proba(X_test)[:, 1]
y_pred = (y_proba >= 0.5).astype(int)

print("ROC AUC: {:.4f}".format(roc_auc_score(y_test, y_proba)))

print("\nClassification report:")
print(classification_report(y_test, y_pred, digits=3))

print("\nConfusion matrix:")
print(confusion_matrix(y_test, y_pred))
```

#### 3.2 API and Containerization with Docker

- API layer
  - Implemented a Flask app in app.py with two endpoints
    - GET /health to verify service status
    - POST /predict to accept JSON input and return prediction and probability
  - The API expects all 20 original input fields and returns “yes” or “no” along with probability
- Docker container
  - Defined Dockerfile based on python:3.11 slim
  - Installed dependencies from requirements.txt
  - Copied app.py and bank\_term\_deposit\_model.joblib into the container
  - Exposed port 8000 and used gunicorn to serve the Flask app

```

Amazon Linux 2023 Kernel Livepatch repository   266 kB/s |  29 kB    00:00
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:00:02 ago on Tue Dec 16 13:04:57 2025.
Dependencies resolved.
=====
Package          Arch    Version      Repository  Size
=====
Installing:
  docker        x86_64  25.0.13-1.amzn2023.0.2  amazonlinux  46 M
Installing dependencies:
  container-selinux  noarch  4:2.242.0-1.amzn2023            amazonlinux  58 k
  containerd       x86_64  2.1.5-1.amzn2023.0.1           amazonlinux  23 M
  iptables-libc   x86_64  1.8.8-3.amzn2023.0.2           amazonlinux 401 k
  iptables-nft    x86_64  1.8.8-3.amzn2023.0.2           amazonlinux 183 k
  libcgroup        x86_64  3.0-1.amzn2023.0.1           amazonlinux  75 k
  libnetfilter_conntrack x86_64  1.0.8-2.amzn2023.0.2           amazonlinux  58 k
  libnftnl         x86_64  1.0.1-19.amzn2023.0.2          amazonlinux  30 k
  libnftnl         x86_64  1.2.2-2.amzn2023.0.2          amazonlinux  84 k
  pigz             x86_64  2.5-1.amzn2023.0.3           amazonlinux  83 k
  runc             x86_64  1.3.3-2.amzn2023.0.1           amazonlinux  3.9 M
=====
Transaction Summary

```

### 3.3 AWS Deployment and Inference

- AWS setup
  - Launched an EC2 instance
  - Installed Docker on the instance
  - Copied code and model files to the instance
  - Built the Docker image and ran the container on port 8000
- Public endpoint testing
  - Used the EC2 public IP and port 8000
  - Verified health endpoint using curl or browser
  - Sent POST requests using curl or Postman to /predict with a sample JSON request
  - Confirmed that the prediction and probability received from the deployed model matched local predictions for the same record

i-00f05b3947e002fe (mlops-bank-api)  
PublicIPs: 16.171.242.180 PrivateIPs: 172.31.28.225

```

ec2-user@ip-172-31-28-225 app]$ curl http://localhost:8000/health
{"status": "ok"}
ec2-user@ip-172-31-28-225 app]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
21edba56462b       bank-ml-api      "python app.py"   44 seconds ago   Up 43 seconds   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   bank-ml-api-container
ec2-user@ip-172-31-28-225 app]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
21edba56462b       bank-ml-api      "python app.py"   50 seconds ago   Up 49 seconds   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   bank-ml-api-container
ec2-user@ip-172-31-28-225 app]$

```

- EC2 instance or SSH session showing the container running

- Postman or terminal showing a successful prediction response from the public endpoint

POST http://16.171.242.180:8 ● +

http://16.171.242.180:8000/predict

POST ▼ http://16.171.242.180:8000/predict

☰ Docs Params Authorization Headers (9) Body • Scripts • Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON** ▾

```
2 "age": 41,
3 "job": "blue-collar",
4 "marital": "married",
5 "education": "basic.6y",
6 "default": "no",
7 "housing": "no",
8 "loan": "no"
```

Body Cookies Headers (5) Test Results ⏱

{ } JSON ▾ ▷ Preview ☰ Visualization ▾

```
1
2   "details": {
3     "threshold": 0.5
4   },
5   "prediction": "no",
6   "prediction_proba": 0.3807780440305106
7
```

## 4. Conclusion

- **Summarised workflow**
    - Built a supervised classification model locally using the bank marketing dataset to predict term deposit subscription ( $y$ )
    - Performed preprocessing through a single reproducible pipeline (numeric scaling and categorical one hot encoding), trained a Logistic Regression model with class balancing, and evaluated performance using ROC AUC, classification report, and confusion matrix (ROC AUC = **0.9423**, accuracy = **0.875**)
    - Containerized the application using Docker (Dockerfile + requirements), validated it locally, and then deployed the same container to an AWS EC2 instance with port 8000 exposed for public access
  - **Business value**
    - The deployed model enables the bank to prioritize outreach toward customers most likely to subscribe, improving campaign efficiency by reducing unnecessary calls/emails while increasing conversion rates.
    - With strong ranking capability (ROC AUC **0.9423**) and high recall for the positive class (**0.833**), the solution is well-suited for marketing use cases where capturing potential subscribers is critical, and targeting thresholds can be tuned to match budget, call capacity, and cost per contact.