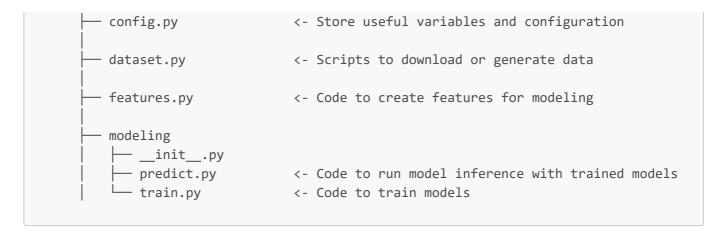
# Allianz Data Science Challenge



A short description of the project.

# **Project Organization**

```
— LICENSE
                      <- Open-source license if one is chosen
-- Makefile
                      <- Makefile with convenience commands like `make data` or
`make train`
- README.md
                     <- The top-level README for developers using this project.
 — data
    ── external <- Data from third party sources.</p>
    — interim
                     <- Intermediate data that has been transformed.
     — processed
                     <- The final, canonical data sets for modeling.
    L— raw
                      <- The original, immutable data dump.
                      <- A default mkdocs project; see www.mkdocs.org for details
 — docs
- models
                      <- Trained and serialized models, model predictions, or
model summaries
─ notebooks
               <- Jupyter notebooks. Naming convention is a number (for</p>
ordering),
                         the creator's initials, and a short `-` delimited
description, e.g.
                         `1.0-jqp-initial-data-exploration`.
 - pyproject.toml
                      <- Project configuration file with package metadata for
                         allianz_data_science_challenge and configuration for
tools like black
- references
                      <- Data dictionaries, manuals, and all other explanatory
materials.
                      <- Generated analysis as HTML, PDF, LaTeX, etc.
  reports
   └── figures
                     <- Generated graphics and figures to be used in reporting
requirements.txt <- The requirements file for reproducing the analysis
environment, e.g.
                         generated with `pip freeze > requirements.txt`
├── setup.cfg
                     <- Configuration file for flake8</p>
lianz_data_science_challenge <- Source code for use in this project.</pre>
      - __init__.py
                              <- Makes allianz_data_science_challenge a Python</pre>
module
```



# How the Project Works (App Execution Flow)

The heart of this project lies in the app.py script, which serves as the main entry point for training the model and making predictions. It leverages modular components from the allianz\_data\_science\_challenge package.

File: app.py

This file acts as a command-line interface (CLI) for:

- 1. Loading a sample customer profile.
- 2. Making a prediction using the current best model.
- 3. Automatically training the model if no valid model exists.
- 4. Printing a human-readable result.

# 

### 1. Input Preparation:

- A sample customer record is manually defined (you can replace it with dynamic input later).
- The record includes key features like age, job, contact, month, duration, and economic indicators.

#### 2. Prediction Attempt:

- The sample data is passed to the predict() function.
- $\circ$  If the model is already trained and valid, it returns a binary prediction (0 = No, 1 = Yes).

#### 3. Model Retraining if Needed:

- If prediction fails (e.g., missing or corrupted model file), the script falls back to train\_model() from the train.py module.
- After successful training, it retries the prediction with the freshly trained model.

#### 4. Output:

Displays whether the customer is predicted to subscribe to a term deposit.

• Handles errors gracefully and provides user-friendly feedback.

## Key Modules Used

Module	Purpose
train_model()	Trains and saves the best model.
predict(df)	Loads the model and predicts on new data.
config.MODELS_DIR	Manages file paths like best_model.pkl.

## Tuture Improvements

- Replace hardcoded input with CLI flags, web input, or batch prediction.
- Add API or Web UI on top (e.g., Flask/FastAPI).
- Include logging, monitoring, and metrics tracking for production usage.

## Example CLI Output

# **&** Problem Framing

The core business challenge is **optimizing the cost-efficiency** of a direct call campaign to sell term deposit products to existing bank customers.

- **Objective**: Reduce the number of unnecessary calls made to customers who are unlikely to subscribe, while retaining as much business value (conversions) as possible.
- Constraints:
  - A single call costs approximately 8€.
  - Each successful subscription yields ~80€ in profit.
  - The **only channel** for conversion is the phone call no call, no sale.
  - o If a customer does **not** subscribe during the call, they are considered permanently uninterested.
- Framed as a supervised classification problem:
  - o Input: Customer features (demographics, contact history, previous campaign outcome, etc.)
  - Output: Binary label 1 if the customer subscribed, ∅ otherwise.
- Business metric focus:
  - Maximize profit = (80€ × true positives) (8€ × total calls)

 Optimize for balanced F1 score and profit gain rather than raw accuracy, due to class imbalance (conversion rate is low).

# Key Drivers of Conversion (Based on Model Insights)

### 1. III Month of Contact

- Conversion rates were significantly higher in March, December, and October.
- Likely due to seasonal behaviors or financial planning cycles.

## 2. Dutcome of Previous Campaign (poutcome)

- Customers with a previous successful interaction had a much higher likelihood of converting again.
- Indicates strong behavioral momentum or positive prior engagement.

## 3. **L** Type of Contact

- Cellular contact showed better conversion than telephone (landline).
- Suggests that mobile outreach may reach customers at more convenient times or signals better accessibility.

### 4. 🖺 Job Type

- Students and retirees had notably higher conversion rates.
- Indicates they might be more receptive to long-term financial planning products.

### 5. State Education Level

- Customers with **tertiary education** were more likely to subscribe.
- Suggests education level correlates with product understanding or trust.

These findings help build a data-driven approach to select weekly cohorts of customers who are:

- Most likely to convert based on these predictive features.
- Actionable for real-time prioritization and cost-saving contact decisions.

### Business Metrics

• Profit if calling ALL customers: €8,896.00

• Profit using model predictions: €21,016.00

• Improvement from model: €12,120.00

• Cost savings from avoided calls: €56,040.00

• Total business value delivered: €77,056.00

• Maximum possible profit: €74,800.00

• Profit efficiency: 28.1%