Arvato Customer Acquisition — Project Report

Machine Learning Engineer Nanodegree Capstone

This report addresses the five major project development stages: Definition, Analysis, Design, Implementation, and Results.

---

1. Definition

1.1 Project Overview

The project addresses the business question: **How can a mail-order company selling organic products acquire new customers more efficiently?** The client has demographic and lifestyle attributes for its existing customers and for the general population of Germany. The goal is to identify which people in the general population are most likely to become new customers, so that marketing campaigns can be targeted at high-potential individuals instead of the entire population. This reduces cost, increases return on marketing spend, and supports a shift from intuition-based to data-driven decision making (Arvato/Bertelsmann mission: *make decisions based on data instead of gut feel*).

1.2 Problem Statement

- **Inputs:** (1) Demographics and lifestyle attributes of established customers; (2) the same attribute structure for the general population of Germany; (3) a campaign dataset with response labels (responder vs. non-responder) for training; (4) a test campaign dataset for evaluation (Kaggle).
- **Output:** (1) Customer segments that over- or under-represent the customer base relative to the general population; (2) a model that predicts the probability of campaign response for each individual; (3) a Kaggle submission with response probabilities for the test set.
- **Success criteria:** The solution must be quantifiable (model and segment metrics), measurable (AUC-ROC, AUC-PR, silhouette score), and replicable (documented preprocessing and modeling pipeline).

1.3 Metrics

- **Supervised (response prediction):** AUC-ROC (ranking quality across thresholds; appropriate for imbalanced binary classification); Area Under the Precision-Recall Curve (AUC-PR), which is more informative when the positive class is rare; and the Kaggle competition metric (typically AUC-ROC).
- **Unsupervised (segmentation):** Silhouette score (cluster cohesion and separation); over/under-representation of customers by cluster (proportion of customers in each cluster vs. proportion in the general population).

1.4 Benchmark Model

Three baselines are used for comparison:

1. **Random targeting:** Assume a random subset of the population is targeted; expected response rate equals the overall response rate in the training data (no intelligence).
2. **Majority-class predictor:** A classifier that always predicts the majority class (non-responder); AUC-ROC = 0.5; establishes the gain from a real model.
3. **Simple rule-based heuristic:** Target individuals whose attributes most closely match the average customer profile (e.g., based on top PCA components or a few key demographics); provides an interpretable baseline before complex models.

The proposed solution (unsupervised segmentation + supervised response model) is evaluated against these benchmarks using the same metrics.

---

2. Analysis

2.1 Data Exploration

Four datasets from AZ Direct GmbH / Arvato (Udacity Bertelsmann Capstone) are used:

Dataset	Rows	Columns	Role
AZDIAS	~891,000	366	General population of Germany
CUSTOMERS	~192,000	369	Established customers (same schema + 3 metadata columns)
MAILOUT_TRAIN	~43,000	366 + RESPONSE	Campaign targets with labels
MAILOUT_TEST	~43,000	366	Test campaign (Kaggle)

Full filenames: `Udacity_AZDIAS_052018.csv`, `Udacity_CUSTOMERS_052018.csv`, `Udacity_MAILOUT_052018_TRAIN.csv`, `Udacity_MAILOUT_052018_TEST.csv`.

The CUSTOMERS file adds three columns (CUSTOMER_GROUP, ONLINE_PURCHASE, PRODUCT_GROUP) that are dropped for alignment with

AZDIAS and MAILOUT. All datasets share the same demographic attribute schema: demographics, financial behavior, lifestyle, household, and regional features. Attributes use DIAS-style coding with missing/unknown codes (-1, 0, -2, -9, 9), which must be replaced with NaN before analysis.

2.2 Data Characteristics

- **Missing values:** Many columns and rows contain a high proportion of missing values. Columns with more than 40% missing are dropped; rows with more than 30% missing are dropped to obtain a clean feature matrix without losing excessive information.
- **Categorical variables:** Some attributes are stored as object type and require label encoding. Unseen categories in the test or mailout data are mapped to a sentinel value (-1) to avoid pipeline failures.
- **Class imbalance:** The campaign response variable (RESPONSE) is highly imbalanced: the majority of individuals are non-responders. Accuracy is therefore misleading; AUC-ROC and AUC-PR are used instead.
- **Scale:** The full AZDIAS dataset has nearly 900k rows. For clustering, subsampling (e.g., 100k rows) or MiniBatchKMeans is used to keep runtime reasonable while preserving segment structure.

2.3 Algorithms and Techniques Considered

- **Dimensionality reduction:** PCA is used to reduce the number of features, capture most of the variance, and reduce noise before clustering. The number of components (e.g., 50) is chosen so that cumulative explained variance reaches a target (e.g., 90%).
- **Clustering:** K-means (via MiniBatchKMeans for scalability) is applied in PCA space to segment the general population. The number of clusters (e.g., 8) is chosen with reference to the silhouette score and interpretability. Customers are mapped to the same cluster space to compute over/under-representation.
- **Classification:** Logistic regression (with class_weight='balanced'), Random Forest, and XGBoost are trained on the preprocessed MAILOUT_TRAIN data to predict RESPONSE. Logistic regression is used for the final Kaggle submission to avoid overfitting observed with tree-based models on the training set when no separate validation set is used.
- **Evaluation:** Silhouette score for clustering; AUC-ROC and AUC-PR for the response model; Kaggle leaderboard for the test set.

---

3. Design

3.1 Solution Architecture

The solution has two main stages:

Stage 1 — Customer segmentation (unsupervised):
Preprocess AZDIAS and CUSTOMERS (replace missing codes, drop high-missing columns/rows, encode categoricals, impute, standardize). Apply PCA to the general population sample, then MiniBatchKMeans to obtain cluster labels. Map customers to the same PCA/cluster space and compute the proportion of customers in each cluster vs. the proportion in the general population. Clusters with ratio > 1 are "customer-like"; clusters with ratio < 1 are under-represented.

Stage 2 — Response prediction (supervised):
Preprocess MAILOUT_TRAIN and MAILOUT_TEST using the same pipeline (fit on AZDIAS/CUSTOMERS where applicable: imputer, encoders, scaler). Train a classifier on MAILOUT_TRAIN (RESPONSE as target). Evaluate with AUC-ROC and AUC-PR; compare to benchmarks. Generate response probabilities for MAILOUT_TEST and submit to Kaggle.

3.2 Preprocessing Pipeline

1. **Load data:** Read all four CSVs (semicolon-separated). Optionally limit rows for AZDIAS and CUSTOMERS to speed up clustering.
2. **Replace missing codes:** Replace -1, 0, -2, -9, 9 with NaN in numeric columns.
3. **Drop high-missing columns:** Remove columns with missing rate $>$ 40%.
4. **Drop high-missing rows:** Remove rows with missing rate $> 30\%$.
5. **Align columns:** Use the intersection of columns across AZDIAS, CUSTOMERS, and MAILOUT (after dropping CUSTOMER_GROUP, ONLINE_PURCHASE, PRODUCT_GROUP from CUSTOMERS).
6. **Encode categoricals:** Label-encode object columns; map unseen labels to -1 when transforming mailout data.
7. **Impute:** Fit a median imputer on the cleaned AZDIAS data; apply to all datasets.
8. **Standardize:** Fit StandardScaler on the segmentation training data (AZDIAS sample); apply to customers and mailout train/test.

3.3 Model Selection

- **Segmentation:** PCA (50 components) + MiniBatchKMeans (8 clusters, batch_size=1000, n_init=3). Silhouette score and cluster proportion ratios guide interpretation.
- **Response prediction:** Logistic Regression (max_iter=500, class_weight='balanced') is used for the submitted predictions to favor generalization over training fit. Random Forest and XGBoost are trained and reported for comparison

but are not used for the final submission to avoid overfitting in the absence of a held-out validation set.

---

4. Implementation

4.1 Code Structure

- **`src/preprocessing.py`:** Defines `load_data()`, `replace_missing_codes()`, `clean_dataframe()`, `encode_and_impute()`, and `preprocess_pipeline()`. Handles missing codes, column/row filtering, categorical encoding, imputation, and optional subsampling for AZDIAS and CUSTOMERS.
- **`Arvato_Customer_Acquisition.ipynb`:** Jupyter notebook that runs the full workflow: load data, preprocess, PCA + K-means, cluster analysis, benchmark and model comparison, and Kaggle submission generation. Uses `nrows_azdias` and `nrows_customers` for faster runs.
- **`run_pipeline.py`:** Command-line script that runs the same pipeline (preprocessing, segmentation, benchmarks, models, submission) and writes `kaggle_submission.csv`. Supports a `--quick` flag for smaller subsets.

4.2 Key Implementation Details

- **Reproducibility:** Random seeds (e.g., 42) are set for sampling, PCA, K-means, and classifiers. The same preprocessing (imputer, encoders, scaler) is fit once and applied consistently to all datasets.
- **Scalability:** MiniBatchKMeans and optional row limits keep clustering feasible on large data. Full MAILOUT_TRAIN and MAILOUT_TEST are always used for supervised learning and submission.
- **Class imbalance:** All classifiers use `class_weight='balanced'` (or equivalent) so that the minority class (responders) is weighted appropriately during training.

4.3 Refinement and Validation

- Preprocessing thresholds (40% column missing, 30% row missing) were chosen to balance retention of data and quality of the feature matrix. Alternative values can be tuned if needed.
- The number of PCA components and clusters was chosen empirically (50 components, 8 clusters); the notebook includes visualizations (e.g., cumulative explained variance, cluster proportion bar charts) to support these choices.
- Cross-validation was not implemented in the initial pipeline; the report recommends adding it (e.g., stratified K-fold AUC-ROC) for robust model selection and hyperparameter tuning in a production setting.

---

5. Results

5.1 Segmentation Results

- **PCA:** Applying PCA with 50 components to the preprocessed general population captures a large share of the variance (e.g., >90% cumulative). The first components can be interpreted as broad demographic/lifestyle axes.
- **Clustering:** MiniBatchKMeans with 8 clusters yields a moderate silhouette score (e.g., ~0.04–0.05 on PCA space), reflecting the fact that demographic segments are not always well-separated. Nevertheless, cluster membership is useful for business interpretation.
- **Over/under-representation:** Comparing the proportion of customers in each cluster to the proportion in the general population identifies clusters that are over-represented among customers (ratio > 1) and thus "customer-like," and clusters that are under-represented (ratio < 1). These segments can guide targeting and messaging (e.g., prioritize individuals in over-represented clusters for campaigns).

5.2 Response Prediction Results

- **Benchmarks:** Majority-class predictor yields AUC-ROC = 0.5; random predictor yields AUC-ROC approximately 0.5. Any useful model must exceed these baselines.
- **Models:** Logistic Regression achieves a reasonable train AUC-ROC (e.g., ~0.77 in runs with subsampled data) and is used for the Kaggle submission. Random Forest and XGBoost can achieve higher training AUC but risk overfitting; they are reported for comparison.
- **Kaggle submission:** The pipeline produces `kaggle_submission.csv` with columns LNR and RESPONSE (probability). Submission to the competition provides the primary external evaluation of the response model.

5.3 Reflection and Improvement

- **Strengths:** The pipeline is modular (preprocessing, segmentation, supervised), documented (README, notebook, report), and reproducible (requirements.txt, fixed seeds). It addresses both unsupervised (segmentation) and supervised (response prediction) components of the capstone and produces a valid Kaggle submission.
- **Limitations:** (1) No cross-validation for model selection; (2) clustering quality (silhouette) is modest, suggesting segments could be refined (e.g., different k, different features); (3) full AZDIAS/CUSTOMERS are not always used for clustering in default runs to save time.
- **Improvements:** (1) Add stratified K-fold cross-validation for AUC-ROC and use it to select and tune the final classifier; (2) experiment with more sophisticated imputation (e.g., iterative imputer) and feature selection; (3)

use the full population for clustering when compute allows, or tune the number of clusters and PCA components systematically.

5.4 Conclusion

The project delivers a complete workflow for Arvato-style customer acquisition: segmenting the general population and existing customers with PCA and K-means, and predicting campaign response with a supervised classifier. The solution is quantifiable, measurable, and replicable, and meets the capstone requirements for code, documentation, and reporting. The generated Kaggle submission and cluster analysis provide a practical basis for data-driven targeting by the mail-order company.