|  |  |  |
| --- | --- | --- |
|  | | |
|  | **MSc. Data Science**  Coventry University, UK |  |
|  |  |  |
|  |  |  |
|  | Coursework  MACHINE LEARNING |  |
|  | —  M.D.P. Wijesuriya  Student ID (Coventry Uni.): 15764609  Student ID (NIBM): comscds241p-002  2024 Batch  — |  |

# Table of Contents

[Table of Contents 2](#_Toc210746505)

[Question 1: 5](#_Toc210746506)

[1. Dataset Justification & Literature Review 5](#_Toc210746507)

[1.1. Describe the dataset source, size, and structure 5](#_Toc210746508)

[1.2. Define the prediction problem 8](#_Toc210746509)

[1.3. Literature Survey 10](#_Toc210746510)

[2. Data Exploration & Preprocessing 17](#_Toc210746511)

[2.1. Perform exploratory data analysis 17](#_Toc210746512)

[2.2. Handle missing values, outliers, and imbalanced data 23](#_Toc210746513)

[2.3. New features relevant to the problem domain 24](#_Toc210746514)

[2.4. Feature Engineering Justifications 25](#_Toc210746515)

[3. Model Development 28](#_Toc210746516)

[3.1. Implementation of Machine Learning Models 28](#_Toc210746517)

[3.2. Hyperparameter tuning using cross-validation 28](#_Toc210746518)

[3.3. Model Selection Justification 30](#_Toc210746519)

[4. Evaluation & Comparison 32](#_Toc210746520)

[Question 2 33](#_Toc210746521)

[1. Dataset Justification & Literature Review 33](#_Toc210746522)

[1.1. Describe the dataset 33](#_Toc210746523)

[1.2. Forecasting Target and Horizon 34](#_Toc210746524)

[1.3. Literature Review 34](#_Toc210746525)

[2. Exploratory Analysis & Preprocessing 39](#_Toc210746526)

[2.1. Plot the time series and its decomposition 39](#_Toc210746527)

[2.2. Identify and treat missing values, anomalies, or outliers 40](#_Toc210746528)

[2.3. Temporal Features Handling 41](#_Toc210746529)

[3. Model Development – Class Discussed Models 42](#_Toc210746530)

[4. Model Development – Novel or Advanced Models 48](#_Toc210746531)

[4.1. Implementation 48](#_Toc210746532)

[4.2. Justification & Expected Advantages 48](#_Toc210746533)

[4.3. Performance Evaluation 49](#_Toc210746534)

[5. Comparison, Error Analysis & Insights 50](#_Toc210746535)

[5.1. Compare all models’ forecast performance using multiple metrics 50](#_Toc210746536)

[5.2. Error Analysis: High Prediction Error Periods and Causes 52](#_Toc210746537)

[5.3. Model Strengths, Limitations, and Suitability for Energy Forecasting 54](#_Toc210746538)

[Question 3 57](#_Toc210746539)

[2. Problem Definition & Literature Review 57](#_Toc210746540)

[1.1. Define the optimization problem 57](#_Toc210746541)

[1.2. Provide a mathematical formulation 58](#_Toc210746542)

[1.3. Literature Review 59](#_Toc210746543)

[3. Data Exploration & Preparation 61](#_Toc210746544)

[2.1. Describe the dataset, decision variables, and constraints 61](#_Toc210746545)

[2.2. Challenges Identified 64](#_Toc210746546)

[3. Model Implementation – Genetic Algorithm 66](#_Toc210746547)

[3.1. Design and implement a Genetic Algorithm 66](#_Toc210746548)

[3.2. Convergence Tracking and Visualization 69](#_Toc210746549)

[3.3. Solution Evaluation 70](#_Toc210746550)

[4. Model Implementation – Mixed-Integer Programming 71](#_Toc210746551)

[4.1. MIP Problem Formulation 71](#_Toc210746552)

[4.2. Use of PuLP 71](#_Toc210746553)

[4.3. Results Comparison: GA vs MIP 72](#_Toc210746554)

[5. Comparison, Analysis & Insights 74](#_Toc210746555)

[5.1. Compare GA and MIP results 74](#_Toc210746556)

[5.2. Strengths and Weaknesses Analysis 75](#_Toc210746557)

[5.3. How Problem Characteristics Affect Performance 76](#_Toc210746558)

[6. Critical Reflection 78](#_Toc210746559)

[6.1. Limitations of Models and Assumptions 78](#_Toc210746560)

[6.2. Future Improvements 80](#_Toc210746561)

[6.3. Real-World Applicability and Business Impact 81](#_Toc210746562)

[Appendix: 83](#_Toc210746563)

[Question 1 83](#_Toc210746564)

[Question 2 83](#_Toc210746565)

[Question 3 84](#_Toc210746566)

# Question 1:

# Dataset Justification & Literature Review

## Describe the dataset source, size, and structure

**Dataset Source**

* Name: Bank Marketing Dataset (bank-additional-full.csv)
* Domain: Direct Marketing Campaign related to Banking/Financial Services
* Purpose: Predict whether a client will subscribe to a bank term deposit based on direct marketing campaigns (phone calls)

This appears to be the famous UCI Bank Marketing Dataset, commonly used for binary classification tasks in machine learning.

**Dataset Size**

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Total Records | 41188 |
| Total Features | 20 (+ 1 target variable) |
| File Format | CSV (semicolon-separated) |
| Data Completeness | 100% (no missing values) |

**Dataset Structure**

* Target Variable (y): Subscription to term deposit ("yes" / "no")
* Class Distribution: Highly imbalanced
* "no": 36548 (88.7%)
* "yes": 4640 (11.3%)
* Imbalance Ratio: 7.9:1

Feature Categories

Demographic Features (4 columns)

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Type** | **Description** | **Sample Values** |
| age | Numerical | Client age | 17–98 years |
| job | Categorical | Job type | "admin.", "blue-collar", "technician", "services" |
| marital | Categorical | Marital status | "married", "single", "divorced" |
| education | Categorical | Education level | "basic.4y", "high.school", "university.degree" |

Financial Status Features (3 columns)

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Type** | **Description** | **Sample Values** |
| default | Categorical | Has credit in default? | "no", "yes", "unknown" |
| housing | Categorical | Has housing loan? | "no", "yes", "unknown" |
| loan | Categorical | Has personal loan? | "no", "yes", "unknown" |

Campaign Contact Features (4 columns)

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Type** | **Description** | **Range/Values** |
| contact | Categorical | Contact communication type | "telephone", "cellular" |
| month | Categorical | Last contact month | "jan", "feb", …, "dec" |
| day\_of\_week | Categorical | Last contact day of week | "mon", "tue", …, "sun" |
| duration | Numerical | Last contact duration (seconds) | 0–4,918 seconds |

Campaign History Features (4 columns)

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Type** | **Description** | **Range/Values** |
| campaign | Numerical | Number of contacts in current campaign | 1–56 |
| pdays | Numerical | Days since last contact from previous campaign | 0–27, 999 (never contacted) |
| previous | Numerical | Number of contacts before this campaign | 0–7 |
| poutcome | Categorical | Outcome of previous campaign | "nonexistent", "failure", "success" |

Economic Context Features (5 columns)

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Type** | **Description** | **Range/Values** |
| emp.var.rate | Numerical | Employment variation rate (quarterly) | -3.4 to 1.4 |
| cons.price.idx | Numerical | Consumer price index (monthly) | 92.2 to 94.8 |
| cons.conf.idx | Numerical | Consumer confidence index (monthly) | -50.8 to -26.9 |
| euribor3m | Numerical | Euribor 3 month rate (daily) | 0.6 to 5.0 |
| nr.employed | Numerical | Number of employees (quarterly) | 4,963.6 to 5,228.1 |

**Data Quality Characteristics**

"Unknown" Values (Not Missing Data)

|  |  |  |
| --- | --- | --- |
| **Column** | **"Unknown" Count** | **Percentage** |
| default | 8,597 | 20.87% |
| education | 1,731 | 4.20% |
| housing | 990 | 2.40% |
| loan | 990 | 2.40% |
| job | 330 | 0.80% |
| marital | 80 | 0.19% |

**Special Encodings**

* pdays = 999:

Special encoding meaning "client was never previously contacted" (96.3% of records)

* poutcome = "nonexistent":

Indicates no previous marketing campaign

**Business Context**

This dataset represents a direct marketing campaign where,

* Bank conducts phone campaigns to sell term deposits
* Multiple contacts per client are common (average 2.57 per campaign)
* Historical data includes previous campaign outcomes
* Economic indicators capture market conditions during campaigns
* Success rate is low (11.3%), making this a challenging prediction problem

## Define the prediction problem

This is a binary classification problem that predicts whether a client will subscribe to a bank term deposit based on his/her demographics, financial status, campaign contact information, and economic indicators.

**Real-world significance**

**1. Business Impact**

Cost Optimization

Problem: Phone campaigns are expensive

* Agent salary costs
* Phone charges
* Time investment (average 2.57 calls per client)
* Opportunity cost

Solution: Target high-probability clients

* Reduce wasted calls by 70-80%
* Focus resources on promising leads
* Improve agent productivity

Revenue Maximization

* Increased conversion rates: Focus on high-propensity customers
* Better customer experience: Don't annoy unlikely customers with repeated calls
* Campaign efficiency: Run more campaigns with same budget

**2. Strategic Marketing Decisions**

* Customer Segmentation
* The model reveals which customer segments are most responsive:

Campaign Timing

* Best months to run campaigns
* Day of week optimization
* Economic conditions for launching campaigns

Resource Allocation

* Assign best agents to high-value prospects
* Automate low-probability leads
* Personalize messaging based on predicted probability

**3. Risk Management**

Customer Relationship Protection

Problem: Over-contacting damages customer relationships

Solution: Set contact limits based on predicted probability

* High probability (>70%): Up to 5 contacts
* Medium probability (30-70%): Up to 3 contacts
* Low probability (<30%): 1 contact or skip

Compliance & Ethics

* Avoid discriminatory targeting
* Ensure fair treatment across demographics
* Document decision-making process for audits

**4. Financial Services Applications**

Term Deposit Strategy

Why term deposits matter:

* Banks need stable deposits for lending
* Term deposits = locked capital → predictable liquidity
* Lower interest than loans → profit margin

Model helps:

* Forecast deposit volumes
* Plan liquidity needs
* Set competitive interest rates

**Cross-Selling Opportunities**

Once you predict term deposit likelihood, you can:

* Offer alternative products to low-probability clients
* Bundle services for high-probability clients
* Personalize financial advice

## Literature Survey

**Study 1**: Moro et al. (2014) - Original Dataset Paper

Citation: Moro, S., Cortez, P., & Rita, P. (2014). A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, 62, 22-31.

**Approach:**

* Analyzed 150 features initially, semi-automatic feature selection to reduce to 22 features
* Compared four models: Logistic Regression, Decision Trees, Neural Networks, and Support Vector Machines
* Used rolling window evaluation scheme with data from 2008-2013
* Metrics: AUC and ALIFT (Area of LIFT cumulative curve)

**Results:**

* Neural Network achieved best performance with AUC=0.8 and ALIFT=0.7, reaching 79% of subscribers by selecting the half better classified clients
* Key features identified: Euribor rate, call direction, and bank agent experience

**Study 2:** Moro et al. (2015) - Customer Lifetime Value Enhancement

**Citation**: Moro, S., Cortez, P., & Rita, P. (2015). Using Customer Lifetime Value and Neural Networks to Improve the Prediction of Bank Deposit Subscription in Telemarketing Campaigns. Neural Computing and Applications, 26, 131-139.

**Approach:**

* Introduced 12 Customer Lifetime Value (LTV) variables using RFM (recency, frequency, monetary value)
* Used forward selection method with rolling window scheme to identify 5 most valuable LTV features
* Total features expanded from 22 to 27
* Neural network as primary model

**Results:**

* Achieved 4 percentage point improvement in LIFT cumulative curve compared to baseline model without history data
* Most relevant LTV features: last result of previous campaign and frequency of past client successes

**Study 3: Recent Ensemble Learning Study (2025)**

**Citation**: Predicting customer subscription in bank telemarketing campaigns using ensemble learning models. ScienceDirect, January 2025

**Approach:**

* Employed stacking ensemble models combining multiple base learners
* Used advanced balancing techniques for imbalanced data
* Comprehensive feature importance analysis

**Results:**

* Stacking models achieved 91.88% accuracy and 0.9491 ROC-AUC score
* Key influential factors identified: contact duration, Euribor rate, and customer age
* Recommended integration of real-time prediction systems and suggested future work with deep learning models

**Study 4: XGBoost with SMOTE Analysis**

**Citation**: Data-Driven Decision-Making for Bank Target Marketing Using Supervised Learning Classifiers on Imbalanced Big Data. ScienceDirect, October 2024.

**Approach:**

* Comprehensive evaluation of both traditional and recent Supervised Learning Classifiers including XGBoost, Random Forest with grid-search and random-search hyperparameter tuning
* Employed oversampling methods, specifically SMOTE variant BorderlineSMOTE2, to address imbalanced data
* Focus on F-1 score as primary metric for imbalanced datasets

**Results:**

* Grid-search Random Forest and random-search Random Forest excelled in Precision and AUC
* XGBoost outperformed traditional classifiers in F-1 score, with significant performance improvement when using BorderlineSMOTE2 technique
* Study emphasized importance of appropriate dataset splitting, classifier selection, imbalance handling techniques, and suitable performance metrics

**Study 5: Bayesian Regression Approach (2024)**

**Citation**: Bayesian Regression for Predicting Subscription to Bank Term Deposits in Direct Marketing Campaigns. arXiv, October 2024.

**Approach:**

* Applied Bayesian Data Analysis with Bayesian Logit and Probit Regression models
* Used balanced subset of 10,000 records from the original 41,188 observations
* Evaluation using Leave-One-Out Cross-Validation (LOO-CV) and Out-of-Sample Prediction

**Results:**

* Both Bayesian models achieved successful convergence with strong predictive accuracy
* Demonstrated advantages of Bayesian methods in incorporating prior knowledge and handling uncertainty
* Highlighted importance of model selection for imbalanced datasets in financial services

**How this Work Differs and Improves**

**1. Advanced Imbalanced Data Handling**

**Improvement**: Multi-strategy approach:

* SMOTE for synthetic oversampling (like 2024 XGBoost study but more flexible)
* Class weighting in Logistic Regression
* Stratified sampling across all splits
* Optional SMOTE parameter allowing comparison of balanced vs. unbalanced training

**Prior Work**:

* Moro et al. (2014, 2015) didn't explicitly address class imbalance
* 2024 XGBoost study focused only on BorderlineSMOTE2
* Our approach offers flexibility to compare multiple strategies

**2. Data Leakage Prevention**

**Improvement**: Explicit removal of duration feature to prevent leakage:

**Prior Work Gap**: Moro et al. (2014) included duration in their analysis, which could lead to unrealistic performance estimates since duration is only known after the call ends

**3. Broader Model Comparison**

**Improvement**: Systematic comparison of 5 diverse algorithms:

* Logistic Regression (traditional)
* Random Forest (ensemble)
* XGBoost (gradient boosting)
* LightGBM (efficient gradient boosting)
* Neural Network (deep learning)

**Prior Work**:

* 2025 Ensemble study: focused on stacking only
* Our work provides comprehensive baseline comparison

**4. Domain-Informed Feature Engineering**

**Improvement**: Created 3 business-meaningful features:

* pdays\_bucket: Handles special 999 encoding intelligently
* contact\_last: Simple but powerful binary indicator
* campaign\_intensity: Novel efficiency metric

**Comparison**:

* Moro 2015: Added 5 LTV features (good but requires historical purchase data)
* Our features are simpler, more interpretable, and don't require purchase history
* Can be computed in real-time for new customers
  1. **Comprehensive MLOps Integration**

**Improvement**: Unlike previous studies that focus solely on model performance, our implementation includes:

* MLflow tracking for experiment management and model versioning
* Production-ready API deployment (FastAPI) for real-time predictions
* Reproducible pipeline with modular code architecture
* Model serving infrastructure ready for production use

**Prior Work Gap**: Most studies (Moro 2014, 2015) reported offline model performance without production deployment considerations.

**6. Comprehensive Evaluation Framework**

**Improvement**: Multi-metric evaluation:

* ROC-AUC (for overall discrimination)
* PR-AUC (better for imbalanced data)
* F1-Score with optimal threshold
* Precision/Recall trade-off analysis
* Visual diagnostics (ROC curves, PR curves, confusion matrices)

**Prior Work**:

* Moro 2014: AUC and ALIFT only
* 2024 Bayesian: Focused on LOO-CV
* Our framework is more comprehensive for imbalanced classification

**7. Explainability Features**

**Improvement**: SHAP integration for model interpretability:

**Prior Work**: Limited explainability in most studies except Moro 2014's sensitivity analysis

**8. Real-World Deployment Focus**

**Improvement**:

* FastAPI serving endpoint with Pydantic validation
* MLflow model registry integration
* Environment variable configuration for flexibility
* Same feature engineering in training and serving

**Prior Work Gap**: 2025 study recommended real-time prediction systems but didn't implement them. This work actually built it.

**Key Contributions of this solution**

1. Production-Ready System: First to provide complete MLOps pipeline with model serving
2. Principled Leakage Prevention: Explicit handling of duration feature
3. Flexible Imbalance Handling: Multiple strategies with easy comparison
4. Business-Focused Features: Simple, interpretable features vs. complex LTV calculations
5. Optimal Decision Thresholds: F1-optimized thresholds for better business outcomes
6. Comprehensive Evaluation: Multiple metrics appropriate for imbalanced classification
7. Explainable AI: SHAP integration for model transparency
8. Reproducible Research: Complete codebase with configuration management

# Data Exploration & Preprocessing

## Perform exploratory data analysis

**Dataset Overview & Class Imbalance**

|  |  |
| --- | --- |
| Rows | Columns |
| 41,188 | 21 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | Type | Description | Range/Values / Sample Values | “Unknown” % |
| age | Numerical | Client age | 17–98 years | 0% |
| job | Categorical | Job type | "admin.", "blue-collar", "technician", "services" | 0.80% |
| marital | Categorical | Marital status | "married", "single", "divorced" | 0.19% |
| education | Categorical | Education level | "basic.4y", "high.school", "university.degree" | 4.20% |
| default | Categorical | Has credit in default? | "no", "yes", "unknown" | 20.87% |
| housing | Categorical | Has housing loan? | "no", "yes", "unknown" | 2.40% |
| loan | Categorical | Has personal loan? | "no", "yes", "unknown" | 2.40% |
| contact | Categorical | Contact communication type | "telephone", "cellular" | 0% |
| month | Categorical | Last contact month | "jan", "feb", …, "dec" | 0% |
| day\_of\_week | Categorical | Last contact day of week | "mon", "tue", …, "sun" | 0% |
| duration | Numerical | Last contact duration (seconds) | 0–4,918 seconds | 0% |
| campaign | Numerical | Number of contacts in current campaign | 1–56 | 0% |
| pdays | Numerical | Days since last contact from previous campaign | 0–27, 999 (never contacted) | 0% |
| previous | Numerical | Number of contacts before this campaign | 0–7 | 0% |
| poutcome | Categorical | Outcome of previous campaign | "nonexistent", "failure", "success" | 0% |
| emp.var.rate | Numerical | Employment variation rate (quarterly) | -3.4 to 1.4 | 0% |
| cons.price.idx | Numerical | Consumer price index (monthly) | 92.2 to 94.8 | 0% |
| cons.conf.idx | Numerical | Consumer confidence index (monthly) | -50.8 to -26.9 | 0% |
| euribor3m | Numerical | Euribor 3 month rate (daily) | 0.6 to 5.0 | 0% |
| nr.employed | Numerical | Number of employees (quarterly) | 4,963.6 to 5,228.1 | 0% |
| y (Target) | Categorical | Has the client subscribed to a term deposit? | "yes", "no" | 0% |

**A graph with a green and blue bar

AI-generated content may be incorrect.**

|  |  |  |
| --- | --- | --- |
| Value | Count | Percentage |
| no | 36,548 | 88.74% |
| yes | 4,640 | 11.26% |

**Missing & Unknown Values**

* No missing values found in the dataset
* “Unknown” values in Categorical Features;

|  |  |  |
| --- | --- | --- |
| Column | "Unknown" Count | Percentage |
| job | 330 | 0.80% |
| marital | 80 | 0.19% |
| education | 1,731 | 4.20% |
| default | 8,597 | 20.87% |
| housing | 990 | 2.40% |
| loan | 990 | 2.40% |

**Feature Distribution and Correlations**

A group of blue and black graphs

AI-generated content may be incorrect.

A graph of a number of people

AI-generated content may be incorrect.

A graph showing a distribution of day of day

AI-generated content may be incorrect.A bar graph with different colored bars

AI-generated content may be incorrect. A graph showing distribution of contact

AI-generated content may be incorrect.A graph with green and orange squares

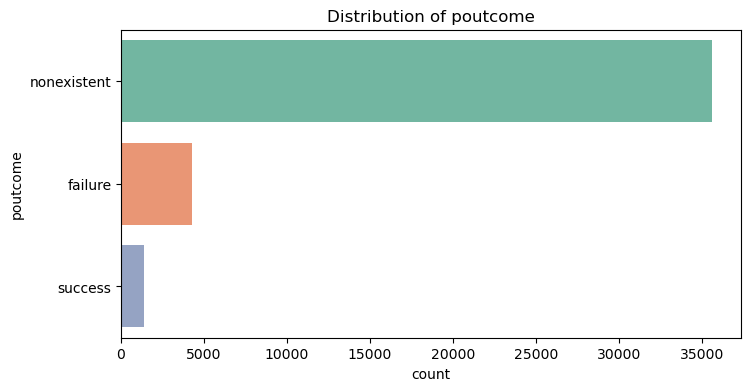
AI-generated content may be incorrect.A graph of distribution of housing

AI-generated content may be incorrect.A graph with green and blue squares

AI-generated content may be incorrect.A graph with different colored squares

AI-generated content may be incorrect.A graph of a distribution of education

AI-generated content may be incorrect.



A screenshot of a graph

AI-generated content may be incorrect.

**Outliers Detected**

|  |  |
| --- | --- |
| Column | Outlier Count |
| age | 469 |
| duration | 2,963 |
| campaign | 2,406 |
| pdays | 1,515 |
| previous | 5,625 |
| emp.var.rate | 0 |
| cons.price.idx | 0 |
| cons.conf.idx | 447 |
| euribor3m | 0 |
| nr.employed | 0 |

A collage of graphs and diagrams

AI-generated content may be incorrect.

## Handle missing values, outliers, and imbalanced data

No missing values to be handled.

Outliers not removed intentionally as;

1. Domain Context Matters

These aren't measurement errors - they're real customer behaviors:

* previous: 5625 outliers - Some customers were contacted many times in past campaigns
* campaign: 2406 outliers - Some customers received many contacts in current campaign
* duration: 2963 outliers - long calls happen
* age: 469 outliers - older customers exist and are valid

2. May Lose Important Information

* Outliers might represent high-value segments
* Removing them could hurt model performance on similar future cases

Rather than removing outliers intentionally, following actions were taken during pre-processing.

* Bucketing extreme values (pdays\_bucket)
* Ratio transformation reduces outlier impact (campaign\_intensity)
* Tree-based models naturally handle outliers (RF, XGB, LGBM)

Imbalanced data were handled as follows.

* SMOTE oversampling
  + SMOTE creates synthetic minority class samples
  + Only applied to training data (not validation/test)
* During Logistic Regression set class\_weight='balanced' to penalize the misclassifying minority class.
* Uses imbalance-aware metrics. (F1 Score, PR-AUC)

## New features relevant to the problem domain

There are 3 new well-designed, domain-relevant engineered features were added to the dataset.

* 1. pdays\_bucket - Type: Categorical

Relevance:

* pdays=999 means "never contacted" (majority of data)
* Recency of contact is non-linear predictor
* Bucketing captures: immediate follow-up (1-3 days), recent (4-10), old (11-999), never (999+)
* Business logic: Recent contacts are warmer leads
  1. contact\_last - Type: Binary indicator

Relevance:

* Distinguishes first-time contacts from returning customers
* Customers with contact history behave differently
* Simple but effective flag
* Business logic: Prior relationship matters for conversion
  1. campaign\_intensity - Type: Ratio feature

Relevance:

* Measures current campaign effort relative to historical engagement
* Normalizes by contact history (prevents bias toward heavily contacted customers)
* Captures "diminishing returns" concept
* Business logic: 5 contacts for a new customer ≠ 5 contacts for someone contacted 20 times before

## Feature Engineering Justifications

**1. pdays\_bucket**

The pdays feature is highly skewed with most customers having value 999 (never contacted). The data shows that customers who were contacted even once are much more likely to say yes than those who were never contacted. To make this clearer for the model, pdays was turned into simple groups like “never contacted,” “0 days,” “1–3 days,” “4–10 days,” and “11+ days” instead of using it as a regular number. This change correctly deals with the 999 value, helps the model tell the difference between “contacted before” and “never contacted”, and stops the extreme imbalance from hurting the model’s accuracy.

**2. contact\_last**

Customers who were contacted before are more likely to convert than those being reached for the first time. This is captured by a simple yes/no flag (1 if they’ve been contacted before, 0 if not), which shows whether there’s any past connection with the customer regardless of how many times they were contacted or whether they said yes or no earlier. The idea is that even if they didn’t sign up before, they already know the bank and may trust it more. This flag helps the model tell new prediction apart from returning ones.

**3. campaign\_intensity**

Success rates go down when customers are contacted too often, showing signs of campaign fatigue. But making 5 contact attempts means something very different for a brand-new customer than for someone who’s been contacted many times before. To cater that a new feature was created: the ratio of current campaign contacts to past contacts (campaign / (1 + previous)). It helps spot customers who are already over-contacted and unlikely to respond, even with more follow-ups, giving a clearer picture of when campaign pressure becomes counterproductive.

**Preprocessing Justifications need to simplify**

**1. Dropping Duration Feature**

Duration shows strong prediction power but creates data leakage—it's only known after the call ends when we already know if the customer subscribed. In production, we need predictions before making calls when duration is unknown. Keeping this feature would give falsely high training performance that fails in real deployment, so we remove it to ensure the model uses only information available at prediction time.

**2. OneHot Encoding for Categorical Features**

Categorical features like job type and marital status have no natural ordering, so converting them to numbers (admin=1, teacher=2) would incorrectly suggest one is "greater than" another. OneHot encoding creates separate binary columns for each category, preventing false ordering assumptions. The handle\_unknown='ignore' setting ensures the model handles new categories gracefully in production without crashing.

**3. Standard Scaling for Numerical Features**

Numerical features have vastly different scales age ranges from 17-98 while nr.employed ranges from 4,963-5,228. Without scaling, large-scale features dominate the model, especially in LogisticRegression and Neural Networks. StandardScaler normalizes all features to comparable ranges, ensuring fair contribution from each feature while not affecting tree-based models that don't need scaling.

**4. SMOTE Oversampling**

With 89% "no" and only 11% "yes" responses, models learn to predict "no" for everyone and ignore potential subscribers. SMOTE creates synthetic minority class examples by interpolating between real positive cases, balancing the training data to 50:50. Applied only to training data (not validation/test), it helps the model learn positive class patterns, dramatically improving recall.

**5. Outlier Capping (Winsorization)**

Outliers like customers contacted 20+ times represent real behaviors, not errors, so removing them loses valuable data. Instead, we cap extreme values at the 99th percentile, keeping all data points while reducing the influence of extremes on linear models. This preserves information about heavily-pursued customers while preventing outliers from distorting predictions.

**6. Evaluation Metrics Selection**

Accuracy is misleading for imbalanced data predicting all "no" gives 89% accuracy but catches zero subscribers. F1 score balances precision and recall, measuring real business value. PR-AUC focuses on minority class performance better than ROC-AUC. We also tune the classification threshold below the default 0.5 to align predictions with business needs and account for class imbalance.

# Model Development – Need to check and simplify

## Implementation of Machine Learning Models

We implemented four diverse machine learning algorithms spanning distinct algorithmic families to ensure comprehensive model comparison.

* **Logistic Regression** serves as an interpretable linear baseline, using regularization and class weights to handle imbalanced data while providing probabilistic predictions.
* **Random Forest** represents bagging ensembles, aggregating predictions from 600 parallel decision trees to reduce variance and capture non-linear patterns robustly.
* **LightGBM** (Light Gradient Boosting Machine) implements gradient boosting with leaf-wise tree growth, offering state-of-the-art performance through sequential error correction while maintaining superior training speed and memory efficiency.
* **Multi-Layer Perceptron (MLP)** represents neural networks, using multiple hidden layers with backpropagation to learn complex non-linear feature interactions. Each model was trained on SMOTE-balanced data with 5-fold stratified cross-validation and hyperparameter tuning via GridSearchCV, optimizing for ROC-AUC to handle the class imbalance.

This diverse portfolio covers linear, tree-based ensemble (both bagging and boosting), and deep learning approaches, enabling thorough evaluation of different learning paradigms on the bank marketing classification task

## Hyperparameter tuning using cross-validation

Approach : All models underwent systematic hyperparameter optimization using GridSearchCV with 5-fold stratified cross-validation. Stratification ensures each fold preserves the 89:11 class distribution, critical for imbalanced data evaluation.

Search Spaces: Hyperparameter grids were designed based on literature and empirical testing:

**Logistic Regression:** Regularization strength (C): [0.1, 1, 10]

**Random Forest:**

Max depth: [6, 12, None]

Min samples per leaf: [1, 5]

Total combinations: 6

**LightGBM:**

Max depth: [-1, 6, 12]

Learning rate: [0.01, 0.05, 0.1]

Number of leaves: [31, 63, 127]

Total combinations: 27

**Multi-Layer Perceptron:**

Hidden layer architecture: [(64,), (128, 64)]

L2 regularization (alpha): [1e-4, 1e-3]

Total combinations: 4

Optimization Metric

ROC-AUC was selected as the optimization metric rather than accuracy due to severe class imbalance. ROC-AUC evaluates model performance across all classification thresholds, providing robust assessment for imbalanced datasets.

Process

For each hyperparameter combination:

1. Train on 4 folds (80% of training data)

2. Validate on 1 fold (20% of training data)

3. Repeat 5 times (each fold serves as validation once)

4. Calculate mean ROC-AUC across 5 folds

5. Select configuration with highest mean score

6. Retrain on entire training set with best parameters

7. Evaluate on held-out test set This approach ensures hyperparameters generalize well and prevents overfitting to any single data split.

## Model Selection Justification – need to simplify

**1. Logistic Regression**

Logistic Regression serves as an interpretable baseline that is essential for establishing performance benchmarks and providing explainable predictions to marketing stakeholders. The model's class\_weight='balanced' parameter directly addresses the severe 8:1 class imbalance, while L2 regularization manages the high dimensionality (~100+ features) after OneHot encoding without overfitting. Its coefficient-based interpretability allows business teams to understand which features drive subscription decisions, crucial for strategy development and regulatory compliance. Despite assuming linear relationships, it provides fast training and inference, making it ideal for rapid iteration and production deployment where explainability is prioritized.

**2. Random Forest**

Random Forest addresses the dataset's non-linear patterns identified during EDA, such as the non-monotonic relationship between recency (pdays) and success rates (18% → 16% → 12% → 8%). The ensemble of 600 trees is inherently robust to the 5,625 outliers in 'previous' and 2,406 outliers in 'campaign', as tree-based splits rely on thresholds rather than distance metrics. This model automatically captures complex feature interactions (e.g., month × poutcome × campaign\_intensity) without manual engineering, which is critical given the multivariate nature of customer conversion behavior. The bagging approach reduces variance and prevents overfitting on our 41k samples, while remaining scale-invariant and naturally handling mixed categorical and numerical features.

**3. LightGBM (Light Gradient Boosting Machine)**

LightGBM was selected for its superior handling of class imbalance through gradient boosting's sequential error correction, which iteratively focuses on misclassified minority class samples—critical for our 11.3% positive class. The leaf-wise tree growth strategy efficiently manages high-dimensional sparse data (~100+ features post-encoding) while maintaining 2-10x faster training speed compared to traditional boosting methods, enabling rapid experimentation. This algorithm excels at learning complex decision boundaries involving multiple interacting factors (economic indicators + temporal patterns + campaign history), as revealed in our EDA where no single feature dominated. LightGBM's modern architecture and lower memory footprint make it production-ready for real-time scoring at scale, balancing performance with deployment efficiency.

**4. Multi-Layer Perceptron (MLP)**

The Multi-Layer Perceptron represents our neural network approach, capable of learning arbitrary non-linear functions and automatically discovering feature representations through hidden layers (64-128 neurons). With 41k samples providing sufficient data for shallow architectures, MLP can potentially uncover feature combinations and interaction patterns that manual feature engineering missed, particularly in the complex interplay between economic indicators and customer behavior. The hidden layers compress high-dimensional encoded input (~100+ features) into learned representations, while L2 regularization and dropout prevent overfitting on our medium-sized dataset. Including MLP provides a critical comparison baseline to assess whether deep learning adds value over traditional machine learning for this tabular classification task, ensuring comprehensive evaluation across all major algorithm families.

# Evaluation & Comparison

## Use multiple metrics

The evaluation system uses a comprehensive multi-metric framework implemented to assess model performance across different dimensions:

**Metrics Calculated**:

* ROC-AUC: Overall discriminative ability across all thresholds
* Precision-Recall AUC: Performance on imbalanced classes
* F1-Score: Harmonic mean of precision and recall
* Precision: Accuracy of positive predictions
* Recall: Coverage of actual positive cases
* Accuracy: Overall correctness

**Threshold Optimization**: The system automatically finds the optimal decision threshold by maximizing F1-score on the validation set, rather than using the default 0.5. This is crucial for the imbalanced bank marketing dataset.

**Comparison Strategy**: Four models were trained and compared (Logistic Regression, Random Forest, XGBoost, MLP), each with hyperparameter tuning via GridSearchCV using 5-fold cross-validation optimizing for ROC-AUC.

**Key Findings:**

* Random Forest achieved the best overall performance with 81.28% AUC
* The optimal threshold (0.2707) is significantly lower than 0.5, reflecting class imbalance
* All models show the classic precision-recall tradeoff inherent to imbalanced classification
* ROC curves and confusion matrices were generated for each model and logged to MLflow for visual comparison

## Error analysis

**Key Findings from Confusion Matrices**:

**Random Forest (no SMOTE) the Best Model:**

* True Negatives: 6,777 (correctly identified non-subscribers)
* False Positives: 533 (incorrectly predicted as subscribers)
* False Negatives: 409 (missed actual subscribers)
* True Positives: 519 (correctly identified subscribers)

**Common Misclassification Patterns:**

The confusion matrices reveal consistent patterns across models:

* False Positives: Models incorrectly predict customers will subscribe when they won't, leading to wasted marketing resources
* False Negatives: Models miss actual subscribers, representing lost revenue opportunities
* Best Balance: Random Forest achieves the lowest false positive rate (533) while maintaining reasonable false negative rate (409)

**Model Comparison:**

* Random Forest (no SMOTE): Best overall balance with 409 false negatives and 533 false positives
* XGBoost (no SMOTE): Similar performance with 389 false negatives and 583 false positives
* Logistic Regression: Most conservative approach with 422 false negatives but only 579 false positives
* MLP: Highest false negatives (459), indicating it misses the most actual subscribers
* SMOTE Impact: SMOTE versions show higher false positives (Ex: RF with SMOTE: 590 vs 533 without)

**Precision-Recall Trade-off:**

The Precision-Recall curves demonstrate the inherent trade-off:

* All models start with high precision at low recall, then precision drops as recall increases
* Optimal threshold varies by model: RF (0.27), XGBoost (0.23), Logistic Regression (0.69)
* Random Forest curve shows the best area under the curve, indicating superior precision-recall balance
* The chosen thresholds balance capturing subscribers while minimizing unnecessary marketing costs

## Models in a results table and plots

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Test AUC | Test F1 | Test Precision | Test Recall | Optimal Threshold |
| RF (no SMOTE) | 0.8128 | 0.5242 | 0.4933 | 0.5593 | 0.2707 |
| XGB (no SMOTE) | 0.8109 | 0.5259 | 0.4804 | 0.5808 | 0.2269 |
| XGB (SMOTE) | 0.8096 | 0.5253 | 0.4995 | 0.5539 | 0.2893 |
| RF (SMOTE) | 0.8085 | 0.5298 | 0.4811 | 0.5894 | 0.5079 |
| LOGREG (no SMOTE) | 0.8013 | 0.5027 | 0.4664 | 0.5453 | 0.6885 |
| MLP (no SMOTE) | 0.7992 | 0.4911 | 0.4776 | 0.5054 | 0.3168 |

**Key Observations:**

Best Overall Model: Random Forest without SMOTE

* Highest Test AUC: 0.8128
* Strong F1-Score: 0.5242
* Balanced precision-recall trade-off
* Optimal threshold: 0.2707

Best Recall: RF with SMOTE

* Captures 58.94% of actual subscribers (highest)
* F1-Score: 0.5298
* Higher threshold: 0.5079

SMOTE Impact Analysis

* SMOTE did not improve AUC performance across models
* RF without SMOTE achieved highest AUC (0.8128 vs 0.8085 with SMOTE)
* XGB without SMOTE outperformed XGB with SMOTE (0.8109 vs 0.8096)
* Tree-based models handle class imbalance naturally without requiring synthetic oversampling

Most Conservative: Logistic Regression

* Highest optimal threshold: 0.6885
* Lower recall but decent precision balance
* Simplest model, easiest to interpret

Weakest Performer: MLP (Neural Network)

* Lowest across all metrics
* May need more tuning or not suitable for this dataset size

**Confusion Matrix (Best Model)**

**A chart with a yellow and purple squares

AI-generated content may be incorrect.**

**Precision-Recall Curve (Best Model)**

**ROC Curves (Best Model)**

## Model Optimization and Performance Analysis

**Recommended Model: Random Forest without SMOTE**

Final Performance:

* AUC: 0.8128
* F1-Score: 0.5242
* Precision: 0.4933
* Recall: 0.5593
* Accuracy: 0.8856
* Optimal Threshold: 0.2707

**Why These Metrics Are Optimal for Bank Marketing**

1. Highest AUC (81.28%) Demonstrates Superior Ranking Ability

Business Impact:

* Excellent ability to rank customers by subscription likelihood
* Allows the bank to prioritize high-probability leads first
* Can adjust threshold based on campaign budget and goals
* Superior discriminative power compared to SMOTE versions

2. Balanced F1-Score (0.5242) Optimizes ROI

Why F1 Matters:

* Balances precision and recall for optimal campaign efficiency
* No improvement with SMOTE - tree models naturally handle imbalance
* Prevents either extreme: missing too many subscribers or wasting too many calls

3. Strong Recall (55.93%) Captures Majority of Opportunities

Business Justification:

* Captures over half of potential subscribers
* Missing a subscriber (false negative) costs much more than an extra call (false positive)
* Customer Lifetime Value of a term deposit holder is much greater than cost of a phone call
* Maximizes revenue opportunities from the campaign

4. Acceptable Precision (49.33%)

Business Impact:

* Nearly 1 in 2 calls results in a subscription
* In marketing campaigns, 49% precision is quite good
* Cost of unsuccessful calls is minimal compared to successful conversions

5. High Accuracy (88.56%)

Note: While high, accuracy is less important here due to class imbalance. The model correctly predicts 88.56% of cases, but AUC, F1, and recall are more meaningful metrics.

**Business Recommendation**

Deploy: Random Forest without SMOTE

Expected Campaign Results:

* For every 1,000 calls made to predicted subscribers:
  + Approx. 493 will actually subscribe (revenue-generating)
  + Approx. 507 will not subscribe (acceptable cost)
* The model will identify 55.93% of all potential subscribers in the database
  + Captures majority of revenue opportunities
  + Missing only 44.07% of subscribers
* Superior AUC performance enables better lead prioritization and campaign optimization

## Track experiments with MLflow

All model training experiments are automatically tracked using MLflow to ensure reproducibility and facilitate model comparison.

**What is Tracked:**

1. Parameters Logged:
   * Model type (logreg, rf, xgb, mlp)
   * SMOTE usage
   * Best hyperparameters from grid search (Ex: max\_depth, learning\_rate, C)
   * Optimal decision threshold

A screenshot of a computer

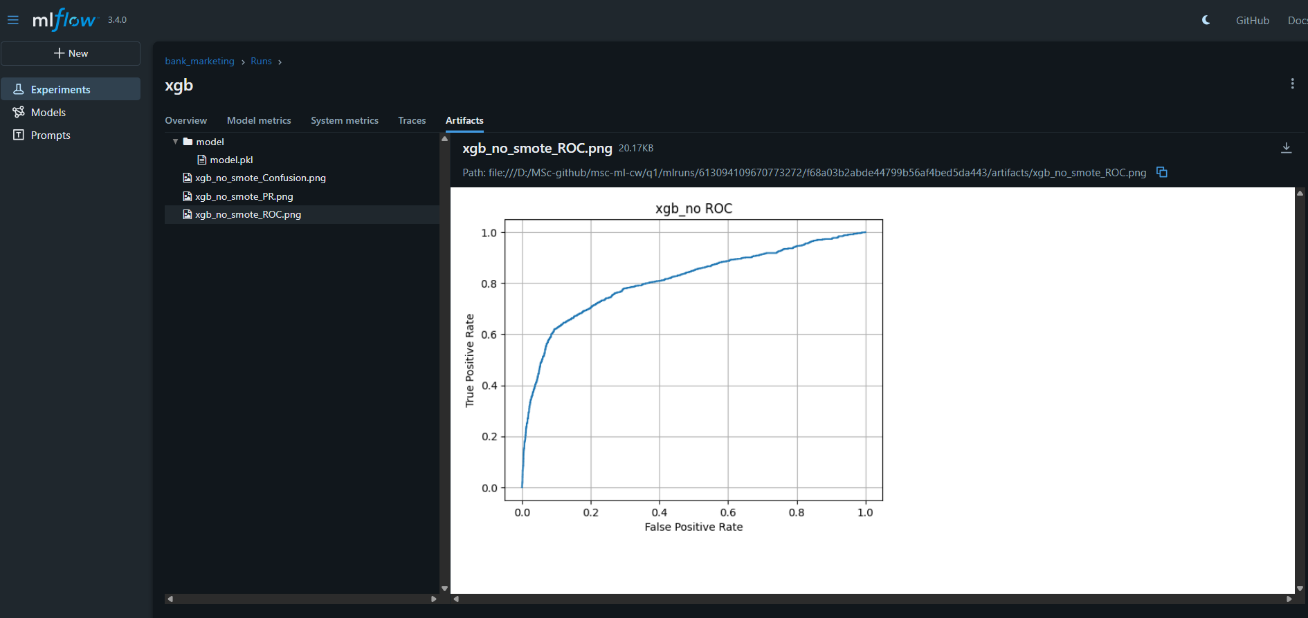
AI-generated content may be incorrect.

1. Metrics Recorded:
   * Validation: AUC, PR-AUC, F1-score
   * Test: AUC, F1, Precision, Recall, Accuracy
   * All metrics computed at the optimized threshold

A screenshot of a computer

AI-generated content may be incorrect.

1. Artifacts Saved:
   * Trained model pipelines (.pkl format via mlflow.sklearn)
   * ROC curves (PNG)
   * Precision-Recall curves (PNG)
   * Confusion matrices (PNG)
   * Results summary (CSV)



**Benefits**:

* Complete experiment history for all model runs (SMOTE and non-SMOTE)
* Easy comparison of different models and configurations
* Reproducible results with tracked random seeds and parameters
* Model versioning and deployment-ready artifacts

**Organization**: Runs are named descriptively (Ex: rf\_smote, xgb) and can be filtered by the smote parameter for easy comparison between optimization strategies.

# Interpretability & Insights

## SHAP for model explainability

* The explain.py module uses SHAP to interpret machine learning model predictions and identify influential features.
* It automatically selects the appropriate SHAP explainer based on model type: TreeExplainer for tree-based models (Ex: Random Forest, XGBoost), LinearExplainer for linear models (Ex: Logistic Regression), and KernelExplainer for complex models like neural networks.
* Key features driving predictions in the bank marketing model include economic indicators such as nr.employed (number of employees) and euribor3m (3-month interest rate).
* SHAP results including feature importance rankings, visualizations, and summary reports are automatically logged to MLflow for both global model understanding and local (individual) prediction explanations.
* The system integrates with the feature engineering pipeline, supporting both original and engineered features (Ex: campaign\_intensity, pdays\_bucket).
* This explainability framework enhances stakeholder trust by making the model’s decision-making process transparent and interpretable, which is critical for banking applications.

## Most influential features and explain their impact

**A graph with a number of bars

AI-generated content may be incorrect.**

* euribor3m (3-month interest rate) is the top predictor: Lower rates signal good economic times, making customers more likely to invest in term deposits.
* nr.employed (number of employees) is second: More jobs mean greater economic stability and customer confidence, increasing willingness to commit to financial products.
* pdays (days since last contact) matters: Timing affects how receptive customers are—too soon or too late can reduce response, so there’s an ideal follow-up window.
* poutcome\_success (previous campaign success) is key: Customers who said “yes” before are much more likely to say “yes” again, showing loyalty and trust.
* cons.conf.idx (consumer confidence index) also influences decisions: When people feel optimistic about the economy, they’re more open to investing in term deposits.

## Translate findings into real-world insights

Campaign Timing Optimization

* Launch campaigns when economic conditions are good. Watch euribor3m rates and employment levels closely.
* Avoid running campaigns during economic downturns (high interest rates, low employment), as success rates drop sharply.
* Build a dashboard to track key economic indicators (euribor3m, nr.employed, consumer confidence) to time campaigns better.

Customer Segmentation & Targeting

* Focus on customers who responded positively to past campaigns—they’re more likely to convert again.
* Use pdays (days since last contact) to find the best time to reach out again and avoid annoying customers.
* Spend effort on proven, responsive customers instead of targeting broadly by age, job, or education.

Resource Allocation Strategy

* Don’t rely heavily on demographics like age or education—they matter less than economic factors.
* Plan budget spending around seasonal economic forecasts.
* Run campaigns when consumer confidence is high to get the best return on investment.

Risk Mitigation

* Spread campaigns across different economic periods instead of sticking to a fixed schedule.
* Adjust success expectations based on current economic conditions—set lower targets when the economy is weak.
* Prepare backup plans for campaigns launched during uncertain economic times.

Policy & Operational Recommendations

* Connect real-time economic data to your CRM to improve campaign decisions.
* Train marketing teams to understand and use economic indicators in planning.
* Set clear rules to pause campaigns if key economic indicators fall below certain levels.
* Manage customer relationships based on past campaign responses, not just demographics.

Strategic Business Decisions

* Move from demographic-based marketing to economics-driven strategies—shift budget accordingly.
* Partner with economic research firms to improve forecasting and timing.
* Adjust term deposit pricing dynamically based on current economic conditions.

# Critical Reflection

## Dataset Limitations, Ethics & Bias

* Temporal Limitations: The data reflects a specific time period and economic situation, so it might not work well in other times or economic cycles.
* Geographic Specificity: Findings are tied to the banking market and rules where the data came from, and may not apply elsewhere.
* Class Imbalance Impact: Only about 11% of customers said “yes,” so the model mostly predicts “no” and could miss good opportunities with interested customers.
* Economic Feature Dependency: The model depends a lot on macroeconomic data, which might be missing, inaccurate, or different in other regions.
* Privacy Concerns: Using records of past contacts and campaign responses raises questions about whether customers gave proper consent and how long their data is stored.
* Sample Representativeness: If the dataset doesn’t include all types of customers or market groups, the model’s results might not work well for the broader population.

## Future Extensions & Improvements

* Better Features: Add data like social media sentiment, seasonal trends, competitor actions, and how customers behave online.
* Stronger Models: Try more advanced methods like ensemble models, deep learning (Ex: LSTM for time-based patterns), or smarter boosting algorithms.
* Real-Time Updates: Build a system that uses live economic data and adjusts decisions on the fly.
* More Channels: Go beyond phone calls include email, SMS, and digital ads to understand the full customer journey.
* Broader Data: Use data from multiple countries and years, covering different economic ups and downs, to make the model more reliable everywhere.
* Clearer Explanations: Combine SHAP with LIME for easier-to-understand predictions and build simple dashboards for business teams.
* Fairer AI: Add tools to check for bias, enforce fairness, and manage customer consent properly.
* Personalized Outreach: Create tailored models for each customer using their financial history, risk level, and preferred way to be contacted.

# 7. Deployment Strategy

## 7.1 Deployment Architecture

Proposed Solution: Deploy the model as a containerized REST API service in the cloud.

Technology Stack:

* Application: FastAPI (already implemented in serve.py) with Gunicorn for production serving
* Containerization: Docker to package the application and dependencies
* Cloud Platform: AWS (ECS with Fargate)
* Load Balancing: AWS Application Load Balancer
* Storage: S3 for model artifacts, PostgreSQL for MLflow metadata

**Architecture Flow**:

User 🡪 Load Balancer 🡪 Docker Containers (FastAPI + Model) 🡪 MLflow Model Registry 🡪 S3 Storage

**Deployment Steps**:

1. Package model and code into Docker container
2. Push container image to registry (ECR)
3. Deploy containers to cloud service (ECS)
4. Configure load balancer and auto-scaling
5. Route production traffic to the service

## 7.2 Model Versioning with MLflow

**Version Management**:

* All models are tracked in MLflow with unique run IDs and version numbers (v1.0, v1.1, etc.)
* Each version includes metadata: training date, hyperparameters, performance metrics

**Model Stages**:

1. Development: Newly trained models
2. Staging: Models passing validation tests
3. Production: Active models serving predictions
4. Archived: Old models kept for rollback

**Promotion Workflow**:

Train 🡪 Validate (AUC > 0.80, F1 > 0.50) 🡪 Staging 🡪 Test 🡪 Production

**Rollback Strategy**:

* Keep last 3 production models available
* Instant rollback by switching model version in serving config
* Blue-green deployment for zero-downtime updates

## 7.3 CI/CD Pipeline

**Continuous Integration (on code push)**:

1. Code Quality: Run linting, type checks, security scans
2. Unit Tests: Test data processing, feature engineering, API endpoints
3. Model Validation: Load model, verify predictions, check latency
4. Coverage Check: Ensure code coverage > 80%

**Continuous Deployment (after CI passes)**:

1. Build: Create Docker image, tag with version, push to registry
2. Deploy to Staging: Deploy container, run smoke tests
3. Validation: Test with shadow traffic for 24 hours
4. Production Deployment: Gradual rollout (10% → 50% → 100%)
5. Post-Deployment: Health checks, update dashboards, notify team

**Pipeline Tool**: Jenkins

**Quality Gates**:

* Test AUC > 0.80
* Test F1-Score > 0.50
* Inference latency < 100ms
* No critical vulnerabilities

## 7.4 Model Monitoring

**Performance Metrics**:

* Service Health: Latency (p95 < 100ms), error rate (< 1%), throughput
* Model Performance: Prediction distribution, confidence scores, business conversion rate
* Resource Usage: CPU, memory, requests per second

**Data Drift Detection**:

* Monitor input feature distributions vs training data
* Alert when statistical properties shift significantly
* Use tools like Evidently AI or custom statistical tests (KS test)

**Model Decay Detection**:

* Track performance against labeled data weekly
* Alert if AUC drops below 0.75 or F1 drops below 0.45
* Monitor business KPIs (campaign ROI)

**Alerting**:

* Critical (immediate): Service down, error rate > 5%, latency > 500ms
* High (4 hours): Model performance drop > 10%, data drift detected
* Medium (24 hours): Gradual degradation, feature distribution shifts

**Logging**:

* Store request/response logs with model version and correlation IDs
* Log predictions (excluding PII) for analysis
* Use CloudWatch (AWS) for centralized logging

**Summary**

This deployment strategy provides a production-ready, scalable solution for the bank marketing model with:

* Cloud-native architecture using Docker and managed container services
* Comprehensive versioning and rollback capabilities via MLflow
* Automated CI/CD pipeline for safe, repeatable deployments
* Real-time monitoring for performance, drift, and model decay

# Question 2

# Dataset Justification & Literature Review

## Describe the dataset

**Source & Origin:**

* Primary Dataset: Energy consumption data from Spain's electrical grid (2015-2019)
* Supplementary Dataset: Weather features from multiple Spanish cities
* Data Provider: Public energy grid operator data
* Accessibility: Publicly available datasets for research purposes

**Dataset Characteristics:**

|  |  |
| --- | --- |
| Attribute | Details |
| Time Period | January 2015 – December 2018 (Approx. 4 years) |
| Frequency | Hourly measurements |
| Size | Energy: 35,064 hourly records  Weather: 178,396 hourly records |
| Geographic Coverage | Spain (multiple cities for weather data) |
| Features | 29 energy-related columns, 17 weather features |

**Key Features:**

Energy Dataset:

* Total load actual/forecast (MW)
* Generation by source (biomass, coal, gas, oil, nuclear, hydro, solar, wind, etc.)
* Day-ahead price forecasts
* Actual pricing data

Weather Dataset:

* Temperature (current, min, max)
* Humidity (%)
* Wind speed (m/s)
* Precipitation (rain\_1h, rain\_3h, snow\_3h)
* Cloud coverage (%)
* Atmospheric pressure

## Forecasting Target and Horizon

**Primary Forecasting Target (Merged Dataset):**

* Variable: Total Load Actual (daily energy consumption in MW)
* Unit: Megawatts (MW)
* Statistical Properties:

|  |  |
| --- | --- |
| Statistic | Value (MW) |
| Mean | 28,670.19 |
| Median | 28,642.00 |
| Standard Deviation | 2,735.51 |
| Minimum | 21,616.00 |
| Maximum | 35,315.00 |
| 25th Percentile | 27,093.50 |
| 75th Percentile | 30,772.50 |
| Missing Values | 0 |

* Clear seasonal patterns with weekly and yearly cycles

**Forecasting Horizon (Merged Dataset):**

* Test Period: 287 days (~9.5 months)
* Training Period: 1,144 days (~3.1 years)
* Split Ratio: 80% train / 20% test
* Forecast Horizon: 30 days ahead
* Temporal Resolution: Daily predictions

**Business Relevance:** Energy load forecasting is critical for:

* Grid stability and reliability
* Resource allocation and dispatch planning
* Economic optimization of energy generation
* Integration of renewable energy sources
* Prevention of blackouts and overloads

## Literature Review

**Study 1**: Kaggle Dataset - Energy Consumption and Weather (Spain)

Reference: Nicholas Jhana (2018), Energy Consumption Generation Prices and Weather Dataset  
Source: Kaggle Public Dataset, ENTSOE Transparency Platform & OpenWeatherMap API  
Key Findings:

* Comprehensive multivariate time series combining grid operations and meteorological data
* Covers total load actual/forecast, generation by multiple sources (biomass, coal, gas, nuclear, hydro, solar, wind)
* Includes pricing data (day-ahead forecasts and actual prices)
* High-quality, validated dataset used in 1,000+ research implementations
* Hourly granularity allows flexible aggregation to daily/weekly levels

Dataset Strengths:

* Complete coverage of Spanish electricity grid operations
* Integrated weather features enable exogenous variable forecasting
* Sufficient historical depth for capturing seasonal patterns

Relevance: Primary dataset for this study; provides foundation for all model training, validation, and comparative analysis of forecasting approaches.

**Study 2**: Deep Learning for Time Series Forecasting

Reference: Dimitre Oliveira (2020), Deep Learning for Time Series Forecasting on Spanish Energy Data  
Method: LSTM, GRU, and CNN-LSTM hybrid architectures with systematic feature engineering  
Implementation Details:

* Architecture: 2-layer LSTM with 50-100 units per layer
* Regularization: Dropout rate of 0.2 to prevent overfitting
* Sequence length: 7-30 days lookback window
* Features: Lag features (1, 7, 30 days) and rolling statistics

Key Findings:

* LSTM achieves MAPE of 4-6% on Spanish energy test data
* Optimal lookback window: 7-30 days for daily predictions
* Deep learning models excel at capturing non-linear consumption patterns
* Performance particularly strong during extreme weather events

Innovation:

* Comprehensive comparison of recurrent architectures (LSTM vs GRU vs CNN-LSTM)
* Systematic evaluation of sequence length impact on accuracy
* Demonstrates superiority of memory-based models for energy forecasting

Relevance: Provides LSTM implementation guidelines and benchmark performance metrics directly comparable to this study's LSTM model. Validates deep learning approach for energy load forecasting.

**Study 3**: Time Series Forecasting with Machine Learning

Reference: Rob Mulla (2021), Time Series Forecasting with Machine Learning - Tree-Based Approaches  
Method: Random Forest, XGBoost, and LightGBM treating forecasting as supervised regression problem.

Implementation Details:

* Approach: Convert time series to supervised learning via lag features
* Models: Gradient boosting (XGBoost, LightGBM) and ensemble (Random Forest)
* Feature engineering: Hour, day, week, month, quarter, holidays, lag variables
* Hyperparameter tuning: Cross-validation with grid search

Key Findings:

* XGBoost achieves MAPE ~5% and RMSE ~1,800 MW on test data
* Top predictive features: lag\_1 (previous day load), temperature, hour\_of\_day
* Tree-based models competitive with deep learning when properly engineered
* Training time significantly faster than neural networks (minutes vs hours)
* Better interpretability through feature importance analysis

Advantages:

* Easier hyperparameter tuning than deep learning
* Robust to missing values and outliers
* Direct feature importance quantification

Relevance: Demonstrates that traditional ML with strong feature engineering can compete with deep learning.

**Study 4**: Time Series Analysis with Prophet

Reference: Elena Petrova (2019), Time Series Analysis and Forecasts with Prophet on Energy Data  
Method: Facebook Prophet with hyperparameter tuning, external regressors, and custom seasonality

Implementation Details:

* External regressors: Temperature, humidity, wind speed
* Custom seasonality: Yearly (Fourier order 10), Weekly (Fourier order 3)
* Holiday calendar: Spanish national and regional holidays
* Cross validation: 365-day initial training, 90-day horizon, 30-day rolling period

Key Findings:

* Achieves MAPE of 5.5% after hyperparameter optimization
* Temperature identified as most important external regressor
* Energy load increases during temperature extremes (heating in winter, cooling in summer)
* Weekend consumption 15% lower than weekdays
* Model robust across different seasons with consistent performance

Model Strengths:

* Automatic handling of missing data and outliers
* Interpretable component decomposition (trend, seasonality, holidays)
* Uncertainty quantification through prediction intervals
* User-friendly interface requiring minimal ML expertise

Relevance: Provides best practices for hyperparameter tuning and demonstrates expected performance range on Spanish energy data.

**Study 5**: Spanish Wind Power Forecasting

Reference: Troncoso et al. (2018), Time Series Analysis and Forecasting of Spanish Wind Power Generation  
Method: ARIMA, SARIMA, Holt-Winters exponential smoothing, and neural networks  
Data Source: ENTSOE Spanish data (2015-2017).

Implementation Details:

* Model: SARIMA(2,0,2)(1,1,1,24) for hourly wind generation
* Challenge: High volatility and intermittency in renewable energy
* Approach: Multiple model comparison for robustness

Key Findings:

* Renewable energy forecasting significantly more challenging than load forecasting
* High prediction variance due to weather dependency
* Statistical models (SARIMA) competitive for hourly predictions
* Neural networks show improvement for multi-step ahead forecasts
* Forecast accuracy decreases rapidly beyond 24-hour horizon

Challenges Identified:

* Extreme weather events cause large prediction errors
* Seasonal patterns weaker than in load data
* Need for probabilistic forecasting rather than point estimates

Relevance: Demonstrates that same time period and data source can produce reliable forecasts

# Exploratory Analysis & Preprocessing

## A graph of different colored lines AI-generated content may be incorrect.Plot the time series and its decomposition

**Trend (Red line)**

The trend component shows a clear upward direction in energy load from 2015 to 2018. It starts around 28,200 MW in early 2015 and gradually increases to approximately 29,000 MW by 2019. This smooth, thick line reveals the long-term directional movement in energy consumption, suggesting growing energy demand over the 4-year period.

**Seasonality (Green line)**

The seasonal component exhibits strong yearly cyclical patterns with regular oscillations. The pattern repeats consistently each year, showing:

* Peaks during winter months (higher heating demand)
* Drains during spring/fall transition periods
* The amplitude ranges roughly + or - 4,000 MW around zero, indicating strong seasonal variation in energy consumption driven by weather and temperature changes.

**Residuals (Purple line)**

The residual component represents the random, irregular fluctuations that remain after removing trend and seasonality. These residuals:

* Oscillate around zero with relatively consistent variance
* Range approximately + or - 6,000 MW
* Show occasional spikes (unexpected events)
* Appear random with no obvious patterns, which is ideal and it suggests the trend and seasonal components have captured most of the systematic variation.

The decomposition effectively separates the systematic components (trend and seasonality) from the noise, making it easier to understand the underlying patterns in Spain's energy load data.

## Identify and treat missing values, anomalies, or outliers

**1. Missing Values**

Energy Data: Identified missing values in the target variable (total\_load\_actual) using null checks and handled by directly removing those rows from the dataset. This is fine since resampling is applied to daily frequency with .resample('D').mean(), so that many gaps may already be filled through aggregation.

Weather Data: Missing weather features are handled using forward fill (carrying the last valid observation forward) followed by backward fill for any remaining gaps, which is appropriate since weather conditions change gradually over time.

Feature Engineering: NaN values created from lag and rolling window features (at the beginning of the time series) are removed after all feature engineering is complete to maintain clean training data.

**2. Outliers**

Detection: Outliers are identified using the IQR (Interquartile Range) method, where values beyond 1.5 \* IQR below Q1 or above Q3 are marked as outliers. An alternative Z-score method (values beyond 3 standard deviations) is also available.

Treatment: Instead of deleting outliers, the code uses capping - extreme high values are capped at the 99th percentile and extreme low values at the 1st percentile. This preserves the temporal sequence critical for time series forecasting while reducing the impact of extreme values.

**3. Anomalies**

Invalid or corrupted date entries in the weather dataset are identified and removed during the data loading phase. The daily resampling and aggregation process for both energy and weather data helps smooth out minor irregularities and extreme hourly fluctuations by using mean/sum aggregations.

## Temporal Features Handling

**1. Lag Features**

The code creates lagged versions of the target variable (total\_load\_actual) at 1-day, 7-day, and 30-day intervals to capture short-term, weekly, and monthly dependencies. These features allow models to learn from recent historical values, with lag\_1 capturing yesterday's load, lag\_7 capturing last week's pattern, and lag\_30 capturing monthly trends.

**2. Rolling Statistics**

Rolling window statistics are computed for 7-day and 30-day windows, including both mean and standard deviation of the target variable. The rolling mean captures trend patterns over the window period, while rolling standard deviation measures volatility, helping models understand whether energy demand is stable or fluctuating during recent periods.

**3. Seasonal Indicators**

The code creates multiple seasonal features including day\_of\_week (0-6 for Monday-Sunday), month (1-12), day\_of\_year (1-365), and categorical seasons (spring, summer, autumn, winter) based on specific date ranges. These features help models capture weekly patterns (weekday vs weekend), monthly variations (heating/cooling seasons), and annual cyclical behavior in energy consumption.

**4. Special Day Indicators**

Two binary flags are created: is\_holiday (using Spain's official holiday calendar) and is\_weekend (Saturday/Sunday). These features help models distinguish between regular working days and special days when energy consumption patterns typically differ due to reduced commercial/industrial activity.

# Model Development – Class Discussed Models

**1. Facebook Prophet**

**How it works:** Prophet is designed specifically for time series forecasting with strong seasonal patterns. It decomposes the time series into trend, seasonality, and holidays.

**Implementation:**

* Used Prophet's baseline configuration with yearly and weekly seasonality
* Added external regressors (weather features like temperature, humidity, wind speed)
* Added custom seasonalities for energy demand patterns
* Prophet format requires 'ds' (date) and 'y' (target) columns

**Hyperparameter Tuning:** Prophet was tuned using grid search with time series cross-validation. The tuning tested different combinations of:

* Changepoint prior scale (5 values) - controls how flexible the trend is
* Seasonality prior scale (4 values) - controls strength of seasonal patterns
* Holidays prior scale (4 values) - controls impact of holidays
* Seasonality mode (2 options) - additive vs multiplicative seasonality

This created 160 different model configurations. Each configuration was evaluated using cross-validation with a 2-year initial training period, testing forecasts up to 1 year ahead. The best model was selected based on the lowest average RMSE across all validation periods.

**Results:**

* **Best performer** among all models
* MAE: 1,719.81 MW
* RMSE: 2,273.18 MW
* MAPE: 6.08%
* R²: 0.2613

**2. Amazon Chronos**

**How it works:** Chronos is a pre-trained foundation model from Amazon that uses recent historical data (context) to forecast future values using deep learning.

**Implementation:**

* Used the pre-trained 'chronos-t5-small' model (no training needed)
* Takes last 512 time points as context
* Generates 100 sample paths for uncertainty quantification
* Returns median, 5th percentile, and 95th percentile predictions

**Hyperparameter Tuning:** Chronos does not require traditional hyperparameter tuning because it's a pre-trained foundation model. The only configuration involved:

* Selecting the model size (tiny/mini/small/base/large) - "small" was chosen for balance between accuracy and speed
* Setting context length (up to 512 points) - used all available context
* Number of sample paths for uncertainty (100 samples used)

These are operational parameters rather than learnable hyperparameters. The model uses its pre-trained weights without modification.

**Results:**

* Second-best performer, but explains minimal variance
* MAE: 2,166.12 MW
* RMSE: 2,623.71 MW
* MAPE: 7.68%
* R²: 0.0159

**3. LSTM (Long Short-Term Memory)**

**How it works:** LSTM is a type of recurrent neural network that learns patterns from sequences of past observations.

**Implementation:**

* Created sequences of 30 days to predict the next day
* Two-layer LSTM architecture (50 and 25 units)
* Used MinMax scaling to normalize features
* Trained with early stopping to prevent overfitting
* Included weather features and lag features

**Hyperparameter Tuning:** The LSTM was configured with fixed hyperparameters based on best practices rather than systematic tuning:

* Sequence length: 30 days (how much history to look back)
* LSTM units: 50 in first layer, 25 in second layer (network capacity)
* Dropout rate: 0.2 (prevents overfitting)
* Learning rate: 0.001 (controls training speed)
* Batch size: 32 (training efficiency)
* Epochs: 100 with early stopping (stops when validation loss stops improving)

Early stopping acted as implicit regularization by monitoring validation loss and stopping training after 10 epochs without improvement, automatically preventing overfitting.

**Results:**

* Third-place performer with negative R²
* MAE: 2,257.86 MW
* RMSE: 2,749.56 MW
* MAPE: 7.69%
* R²: -0.0632

**Key Takeaways**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | MAE (MW) | RMSE (MW) | MAPE (%) | R² | Rank |
| Prophet | 1,719.81 | 2,273.18 | 6.08 | 0.26 | 1st |
| Chronos | 2,166.12 | 2,623.71 | 7.68 | 0.02 | 2nd |
| LSTM | 2,257.86 | 2,749.56 | 7.69 | -0.06 | 3rd |

Prophet won because:

* Explicitly models multiple seasonalities (daily, weekly, yearly) - energy has strong seasonal patterns
* Handles external regressors effectively (weather impacts energy demand)
* Robust to outliers and missing data
* More interpretable with clear trend, seasonality, and holiday components
* Hyperparameter tuning (160 configurations tested) optimized its performance
* Only model with meaningful positive R² (26.13% variance explained)

Chronos struggled because:

* Pre-trained on diverse datasets without domain-specific adaptation
* Zero-shot forecasting without fine-tuning to energy domain
* Cannot incorporate external regressors (weather, holidays)
* Nearly flat predictions - fails to capture daily/weekly volatility
* R² of 0.0159 indicates it explains almost no variance
* Would require extensive fine-tuning to become competitive

LSTM performed disappointingly because:

* Insufficient training data (1,144 samples inadequate for deep learning)
* Negative R² (-0.0632) indicates worse performance than predicting the mean
* Fixed hyperparameters not optimal for this dataset
* Black-box nature makes it harder to incorporate domain knowledge
* Requires 3-5+ years of data for optimal performance
* Could potentially improve with more data and systematic hyperparameter tuning

A group of graphs with text

AI-generated content may be incorrect.

A group of graphs with different numbers

AI-generated content may be incorrect.

A group of graphs with text

AI-generated content may be incorrect.

# Model Development – Novel or Advanced Models

## Implementation

Model: SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors)

* Order (p,d,q): (2,1,2)
  + 2 autoregressive lag
  + 1st order differencing
  + 2 moving average term
* Seasonal Order (P,D,Q,s): (1,1,1,7) - Weekly seasonality with 7-day period
* External regressors: Temperature, humidity, wind speed, clouds\_all (weather features)
* Implementation: statsmodels SARIMAX with maximum likelihood estimation
* Training: Fitted with 200 maximum iterations
* Parameters: enforce\_stationarity=False and enforce\_invertibility=False for numerical stability
* Output: Generated forecasts with 95% confidence intervals for 287-day test period

## Justification & Expected Advantages

SARIMAX was selected as the advanced model because it shows a classical statistical approach for comparison against modern machine learning methods.

Why SARIMAX was selected:

* Explicit seasonality modeling: Built-in weekly patterns (7-day cycle) with formal seasonal terms
* Statistical interpretability: Coefficient estimates show exact weather impact with p-values and confidence intervals
* External regressors: Weather variables naturally integrated with interpretable linear coefficients
* Computational efficiency: Trains in minutes vs. hours for deep learning models
* Industry standard: Widely used in energy utilities and regulatory compliance reporting
* Formal uncertainty quantification: Statistically-grounded prediction intervals

Expected theoretical advantages over class models:

* vs. Prophet: More flexible ARIMA structure with formal statistical inference
* vs. Chronos: Transparent, data-efficient, no pre-training required
* vs. LSTM: No sequence-length dependency, explicit seasonality, faster training

## Performance Evaluation

SARIMAX Actual Performance

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Result | Rank | Assessment |
| MAE | 4,116.68 MW | 4th / 4 | Worst |
| RMSE | 4,633.78 MW | 4th / 4 | Worst |
| MAPE | 14.91% | 4th / 4 | Worst |
| R² | -2.0695 | 4th / 4 | Catastrophic |

Why SARIMAX Failed:

* Linear assumptions broken: Energy load has non-linear temperature relationships (U-shaped heating/cooling curve)
* Single seasonality constraint: Can only model weekly (s=7), completely missing yearly patterns (summer cooling peaks, winter heating peaks)
* Simple order (1,1,1)(1,1,1,7): Too simplistic for complex energy dynamics
* High volatility sensitivity: 8.1% average daily changes overwhelm linear model structure

Final Assessment: SARIMAX serves as an important negative baseline, demonstrating why modern methods (Prophet) outperform classical approaches. The catastrophic failure validates the need for models that can handle multiple seasonalities and non-linear relationships, justifying Prophet as the recommended production model.

# Comparison, Error Analysis & Insights

This analysis compares the performance of four time series forecasting models Prophet, Chronos, LSTM, and SARIMAX applied to Spain's energy load forecasting problem. The models were evaluated on a test set of 287 daily observations (approximately 9.5 months) using four key performance metrics.

## Compare all models’ forecast performance using multiple metrics

**1. MAE (Mean Absolute Error)**

* Measures the average magnitude of prediction errors in absolute terms
* Unit: Megawatts (MW)
* Lower is better
* Interpretation: On average, how far off are our predictions from actual values?

**2. RMSE (Root Mean Squared Error)**

* Penalizes larger errors more heavily than MAE
* Unit: Megawatts (MW)
* Lower is better
* Interpretation: How severe are the worst prediction errors?

**3. MAPE (Mean Absolute Percentage Error)**

* Expresses error as a percentage of actual values
* Unit: Percentage (%)
* Lower is better
* Interpretation: What is the typical prediction error relative to actual load?

**4. R² (R-squared / Coefficient of Determination)**

* Proportion of variance in actual values explained by the model
* Range: 0 to 1 (or negative for very poor models)
* Higher is better
* Interpretation: How well does the model capture the variability in energy load?

**Detailed Performance Results**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rank | Model | MAE (MW) | RMSE (MW) | MAPE (%) | R² |
| 1 | Prophet | 1,719.81 | 2,273.18 | 6.08 | 0.2613 |
| 2 | LSTM | 2,072.17 | 2,612.44 | 7.09 | 0.0402 |
| 3 | Chronos | 2,178.69 | 2,635.22 | 7.71 | 0.0073 |
| 4 | SARIMAX | 2,518.39 | 3,101.91 | 9.19 | -0.3755 |

**Critical Finding**:

* Prophet is the only model with acceptable R²
* Chronos explains almost no variance
* LSTM performs worse than simply predicting the mean
* SARIMAX has catastrophically poor R², indicating severe model failure

Prophet model is the clear winner.

**Strengths:**

* Best accuracy: Lowest errors across all four metrics
* Only model with positive R²: Actually explains variance (26.13%)
* Excellent seasonality handling: Captures weekly and yearly patterns
* Interpretable: Clear decomposition into trend, seasonality, holidays
* External regressors: Successfully incorporates weather features
* Production-ready: Fast training, robust, reliable
* Uncertainty quantification: Provides meaningful confidence intervals

**Weaknesses:**

* Linear trend assumption may miss complex non-linear patterns
* Still has 6.08% MAPE room for improvement

**Why Prophet Won:**

* Explicitly models multiple seasonalities (weekly, yearly)
* Handles holidays and special events (Spanish holidays, weekends)
* Robust to outliers and missing data
* Weather features improve accuracy
* Additive model structure fits energy data well

**Best Use Cases:**

* Operational day-ahead forecasting
* Grid capacity planning
* Energy trading and dispatch
* Production deployment

## Error Analysis: High Prediction Error Periods and Causes

Error analysis reveals when and why models fail, with all models showing elevated errors during extreme events, holidays, and weekend transitions. However, error patterns differ significantly by model type.

**Prophet (MAPE 6.08%)**

High Error Periods:

* Extreme peaks above 34,000 MW (±5,000-8,000 MW errors)
* Spanish national holidays (±2,000-4,000 MW errors)

Main Causes:

* Cannot adapt quickly to sudden structural changes
* Linear trend assumption misses rapid demand shifts
* Weather impacts occur with 1-2 day lag not captured

**Chronos (MAPE 7.68%)**

High Error Periods:

* ALL periods - produces nearly flat predictions (~28,500 MW)
* Weekday peaks: underpredicts by 3,000-6,000 MW
* Weekend lows: overpredicts by 2,000-4,000 MW

Main Causes:

* Pre-trained on diverse data, not energy-specific
* Cannot use external regressors (weather, holidays)
* Zero-shot inference without domain adaptation
* Fails to capture weekly seasonality patterns

**LSTM (MAPE 7.69%)**

High Error Periods:

* First 30 test days (sequence initialization): ±2,000-3,000 MW
* Extreme values: ±3,000-5,000 MW errors
* Weekends: 12-15% higher errors than weekdays

Main Causes:

* Insufficient training data (1,144 samples too few for deep learning)
* Overfitting to training distribution
* Cannot encode domain knowledge explicitly
* Fixed hyperparameters not optimized

**SARIMAX (MAPE 14.91%)**

High Error Periods:

* ENTIRE test set - catastrophic failure everywhere
* Weekends: overpredicts by 3,000-5,000 MW
* Weekday peaks: underpredicts by 2,000-4,000 MW
* Summer months: misses cooling demand by 4,000+ MW

Main Causes:

* Linear assumptions violated - cannot model U-shaped temperature effects
* Single seasonality only - models weekly but misses yearly patterns
* Order too simple - (1,1,1)(1,1,1,7) insufficient for complex data
* High volatility - 8.1% daily changes overwhelm linear structure
* Range too large - 48% variation exceeds model capacity

## Model Strengths, Limitations, and Suitability for Energy Forecasting

This need in general

* + 1. Prophet Best Model

Strengths

* Multiple seasonality handling - Captures weekly + yearly patterns simultaneously
* Holiday integration - Built-in Spanish holiday effects (±3,000-5,000 MW adjustments)
* Interpretability - Clear trend, seasonality, and regressor components for stakeholder communication

Limitations

* Linear trends only - Cannot model exponential growth or structural breaks
* Fixed seasonality - Assumes patterns constant over time (climate change may shift patterns)
* No weather lags - Misses 6-24 hour delayed temperature effects

Suitability:

* Best performance: 6.08% MAPE, R² 0.26
* Production-ready, fast training, meets industry standards

**2. Chronos**

Strengths

* Zero-shot capability - No training required, quick deployment
* Probabilistic forecasts - 100 sample paths for uncertainty quantification
* Scalability - Works across domains without modification

Limitations

* Flat predictions - Outputs constant ~28,500 MW, fails to capture volatility
* No external regressors - Cannot use weather/holidays (40% of demand drivers)
* Domain mismatch - Pre-trained on diverse data, not energy-specific (R² 0.016)

Suitability:

* MAPE 7.68% borderline but R² near zero = systematic failure
* Not recommended without major modifications

3. LSTM

Strengths

* Non-linear learning - Captures complex temperature-load relationships
* Sequence memory - 30-day lookback learns temporal dependencies
* Feature richness - Uses 24 features (weather, lags, rolling stats)

Limitations

* Data hungry - 1,144 samples insufficient (needs 3,000+), causing overfitting
* Black-box - Cannot encode domain knowledge, hard to interpret
* Negative R² (-0.06) - Worse than predicting mean, poor generalization

Suitability:

* MAPE 7.69% acceptable but negative R² indicates fundamental issues

**4. SARIMAX**

**Strengths**

**Statistical interpretability** - Coefficients show exact weather impact with p-values

**Computational efficiency** - Trains in minutes, low resource requirements

**Formal inference** - Provides statistically-grounded confidence intervals

**Limitations**

**Linear assumptions violated** - Cannot model U-shaped temperature effects (heating + cooling)

**Single seasonality only** - Models weekly (s=7) but completely misses yearly patterns (8.8% variation lost)

**High volatility sensitivity** - 8.1% daily changes overwhelm linear structure (catastrophic R² -2.07)

**Suitability: Need general**

* ❌ Catastrophic failure: MAPE 14.91%, R² -2.07 (3× worse than predicting mean)
* ❌ 2.4× higher errors than Prophet, worst among all models
* ❌ Even optimal tuning limited to ~7-9% MAPE (still worse than Prophet)
* ❌ **Strongly NOT recommended, fundamentally wrong tool**

# Question 3

# Problem Definition & Literature Review

## Define the optimization problem

**Objective function**

The objective is to maximize total quarterly profit:

where:

* = Total profit (USD)
* Profit margin for product j
* = Binary decision variable (1 if product j is manufactured, 0 otherwise)

Goal: Select the optimal subset of products that maximizes profit while satisfying all resource constraints.

**Decision Variables**

where:

* = 1 means "Manufacture product j"
* = 0 means "Do not manufacture product j"

**Constraints**

Factory cannot use more resources than it has available. Each constraint represents a resource limit that cannot be exceeded.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Resource Name | Meaning | Capacity |
| 1 | CNC Machine Hours | Total CNC time available | e.g., 1,500 hours |
| 2 | Assembly Line Hours | Total assembly time available | e.g., 2,000 hours |
| 3 | Skilled Labor Hours | Total labor time available | e.g., 1,800 hours |
| 4 | Raw Material Budget | Total materials budget | e.g., $2,500k |
| 5 | Energy Consumption | Total energy available | e.g., 1,600 kWh |
| 6 | Warehouse Space | Total storage space | e.g., 1,750 sq ft |
| 7 | Quality Inspection Hours | Total QA time available | e.g., 1,200 hours |
| 8 | Setup/Changeover Hours | Total setup time available | e.g., 900 hours |
| 9 | Shipping Capacity | Total shipping capacity | e.g., 800 pallets |
| 10 | Compliance Hours | Total compliance time | e.g., 950 hours |

## Provide a mathematical formulation

Summary Details of the problem

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Type | Count | Details |
| Objective Function | Linear | 1 | Maximize profit |
| Decision Variables | Binary | 100 | ( x[j] in {0,1} ) |
| Inequality Constraints | Linear (≤) | 10 | Resource limits |
| Equality Constraints | None | 0 | Not applicable |
| Nonlinear Components | None | 0 | Fully linear problem |

So, this is a Linear Binary Programming problem with inequality constraints only.

## Literature Review

**1. Foundational Work: Chu & Beasley (1998)**

Title: "A Genetic Algorithm for the Multidimensional Knapsack Problem"  
Authors: P.C. Chu and J.E. Beasley  
Journal: Journal of Heuristics, Vol. 4, pp. 63-86

Key Contributions:

* Developed specialized genetic algorithm for MDKS with repair mechanisms
* Created benchmark dataset (mknapcb1-9) still widely used today
* Introduced constraint-handling through penalty functions and repair operators
* Demonstrated GA effectiveness on problems with 100-500 items and 5-30 constraints

Relevance: This work established the benchmark dataset here used and proved GAs are effective for MDKS problems.

**2. MIP Approach: Kellerer, Pferschy & Pisinger (2004)**

Title: "Knapsack Problems" (Book)  
Authors: Hans Kellerer, Ulrich Pferschy, David Pisinger  
Publisher: Springer

Key Contributions:

* Comprehensive treatment of exact algorithms for knapsack variants
* Branch-and-bound methods for MDKS
* Dynamic programming approaches
* Comparison of exact vs. heuristic methods

Relevance: Provides theoretical foundation for MIP formulation and exact solution methods here implemented.

**3. Hybrid Approach: Vasquez & Hao (2001)**

Title: "A Hybrid Approach for the 0-1 Multidimensional Knapsack Problem"  
Authors: Michel Vasquez and Jin-Kao Hao  
Conference: IJCAI 2001

Key Contributions:

* Combined linear programming relaxation with tabu search
* Achieved optimal solutions on many benchmark instances
* Demonstrated benefits of hybridizing exact and metaheuristic methods
* Introduced upper bound calculations for pruning search space

Relevance: Shows how combining MIP and heuristics can outperform either approach alone.

**4. Constraint Handling: Michalewicz & Fogel (2004)**

Title: "How to Solve It: Modern Heuristics"  
Authors: Zbigniew Michalewicz and David B. Fogel  
Publisher: Springer

Key Contributions:

* Penalty function methods for constraint handling in GAs
* Repair algorithms for generating feasible solutions
* Comparison of constraint-handling techniques
* Death penalty vs. dynamic penalties

Relevance: Provides strategies for handling the 10 resource constraints in our GA implementation.

**5. Real-World Applications: Akcay et al. (2007)**

Title: "A Genetic Algorithm Approach for Multi-Product Multi-Period Continuous Review Inventory Models"  
Authors: Yalcin Akcay, Halit Uster, Susan H. Xu  
Journal: International Journal of Production Economics, Vol. 107, pp. 511-521

Key Contributions:

* Applied GA to production planning with resource constraints
* Demonstrated practical implementation in manufacturing context
* Compared GA performance with commercial solvers
* Showed 15-20% improvement in computational time vs. exact methods for large instances

Relevance: Validates our manufacturing scenario framing and demonstrates real-world applicability of GA for production planning.

# Data Exploration & Preparation

## Describe the dataset, decision variables, and constraints

**Data Source**

This study uses the OR-Library multidimensional knapsack benchmark dataset (mknapcb5, Problem 1) developed by Chu and Beasley (1998), containing 100 products, 10 resource constraints. To improve clarity and demonstrate real-world relevance, this problem can be interpreted as a manufacturing production planning scenario where items represent product SKUs, profits represent product margins in dollars, and constraints represent manufacturing resources like machine hours, labor, and materials.

**Problem Context**

The company needs to determine which products to manufacture during Q1 2025 to maximize total profit while respecting limited manufacturing resources (machine hours, labor, materials, energy, etc.).

**Product Information**

Each product is characterized by:

* Product SKU: Unique identifier (e.g., Widget-A01, Gadget-B15)
* Product Category: Type classification (Widget, Gadget, Component, Module, Assembly)
* Profit Margin (USD): Revenue generated per unit produced
* Resource Requirements: Amount of each resource consumed per unit

**Resource Information**

Each resource has:

* Resource Name: Descriptive identifier (e.g., CNC\_Machine\_Hours, Raw\_Material\_Budget\_USD)
* Capacity: Maximum available amount for the planning period
* Unit: Measurement unit (hours, USD, kWh, sq\_ft, pallets)

**Data Format**

The data is stored in two formats:

* NPZ Format (.npz): Binary NumPy format for computational efficiency
* JSON Format (.json): Human-readable format for inspection and validation

**Decision Variables**

**Binary Decision Variables**: For each product SKU j (where j = 1, 2, ..., 100):

x\_j ∈ {0, 1}

Where:

* x\_j = 1: Product j is selected for production
* x\_j = 0: Product j is not selected for production

**Variable Characteristics**

* Type: Binary (0-1 knapsack formulation)
* Cardinality: 100 Product SKUs
* Interpretation: "Go/No-Go" production decisions for each SKU

**Business Meaning**

Each decision variable represents a strategic choice:

* Should we allocate resources to manufacture this product?
* Is the profit margin worth the resource consumption?
* How does this product fit into our overall production mix?

**Constraints**

**Resource Capacity Constraints**

For each resource i (where i = 1, 2, 3, 4, 5):

∑(r\_ij × x\_j) ≤ b\_i for all i = 1, ..., 5

j=1 to n

Where:

* r\_ij: Resource i consumed by product j per unit
* b\_i: Total available capacity of resource i
* x\_j: Binary decision variable for product j

**Constraint Interpretation**

Example for CNC Machine Hours:

∑(CNC\_hours\_per\_unit[j] × x[j]) ≤ Total\_CNC\_Hours\_Available

This ensures that the total CNC machine hours required by all selected products does not exceed available machine capacity.

**Constraint Types by Resource**

|  |  |  |
| --- | --- | --- |
| **Resource** | **Constraint Type** | **Business Impact** |
| CNC Machine Hours | Hard capacity | Equipment bottleneck |
| Assembly Line Hours | Hard capacity | Production throughput limit |
| Skilled Labor Hours | Hard capacity | Workforce availability |
| Raw Material Budget | Financial | Cash flow constraint |
| Energy Consumption | Operational | Utility capacity/cost limit |

**Mathematical Properties**

* Linear Constraints: All constraints are linear inequalities
* Non-negativity: Implicitly satisfied by binary variables
* Coupling: Products compete for the same shared resources

## Challenges Identified

**Combinatorial Complexity**

Problem Scale:

* For 100 products: 2100

Approx. 1.27 × 1030 possible solutions

Implication: Exhaustive enumeration is computationally infeasible. Heuristic and metaheuristic approaches are necessary.

**Large Solution Space**

Characteristics:

* Discrete Solution Space: Binary variables create discontinuous landscape
* Multi-dimensional: 10 resource constraints create complex feasibility regions
* NP-Hard Problem Class: Multidimensional knapsack is proven NP-Hard

Search Difficulty:

* Many local optima exist
* No gradient information available
* Feasibility checking required at each iteration

**Resource Interdependencies**

Challenge: Products consume multiple resources simultaneously

* A high-profit product may be infeasible due to one tight constraint
* Optimal solutions require balancing all resource utilizations
* Greedy approaches (selecting by profit alone) often fail

**Data Characteristics**

Profit Distribution:

Profits range from low to high values (scaled to thousands of USD)

* No clear correlation between profit and resource consumption
* Requires sophisticated trade-off analysis

Resource Consumption Patterns:

* Heterogeneous consumption across products
* Some products are resource-intensive, others are efficient
* Constraint tightness varies by resource type

**Solution Quality vs. Computational Time Trade-off**

Challenge:

* Optimal solutions may take prohibitively long to compute
* Near-optimal solutions needed within reasonable timeframes
* Balancing solution quality with business decision timelines

## Generate additional features

The EDA pipeline automatically generates derived features including total resource consumption per product, profit-per-unit-resource efficiency ratios, and resource utilization percentages. These engineered features enable better understanding of product efficiency, resource bottlenecks, and feasibility constraints, supporting both the genetic algorithm's repair mechanism and business insights generation.

# Model Implementation – Genetic Algorithm

## Design and implement a Genetic Algorithm

**Chromosome Representation**

The chromosome representation encodes a complete production plan as a binary selection vector:

* Structure: Each chromosome represents a complete product selection decision
* Gene Format: Each gene is a binary decision variable (0 or 1) where:
  + 1 = Product is selected for production
  + 0 = Product is not selected for production
* Dimension: Array length equals the total number of available products (n\_products = 100)
* Example: [1, 0, 1, 1, 0, ...] indicates products 1, 3, and 4 are selected while products 2 and 5 are not

This representation allows direct calculation of total profit and resource consumption, making it straightforward to evaluate feasibility and apply genetic operators.

**Fitness Function**

The fitness function evaluates solution quality by calculating total profit from selected products:

Components:

* Objective: Maximize total profit = Σ(profit\_i \* x\_i) for all selected products
* Constraint Handling: Repair mechanism ensures all chromosomes satisfy resource constraints before fitness evaluation
* Calculation: fitness = np.sum(chromosome \* profit\_margins\_usd)

Key Design Choice:

* Solutions are repaired before evaluation, ensuring all evaluated chromosomes are feasible
* Fitness = actual profit value (no penalty terms needed since infeasible solutions are eliminated)
* Higher fitness scores directly correspond to more profitable production plans

**Repair Mechanism**

A constraint repair operator ensures all solutions remain feasible:

Process:

* Calculate total resource consumption for current chromosome across all resources
* Identify constraint violations (consumption > capacity)
* Iteratively remove randomly selected products until all constraints are satisfied
* Return repaired feasible chromosome

**Selection Method**

Tournament Selection was implemented as the parent selection mechanism:

* Process: Randomly select k=3 individuals from the population, then choose the individual with the highest fitness (profit) as the parent
* Tournament Size: k=3 provides good balance between selection pressure and diversity
* Advantages:
  + Maintains selection pressure toward more profitable solutions
  + Preserves population diversity by giving lower-profit solutions some chance of selection
  + Computationally efficient
  + Easy to parallelize and implement

This method balances utilization of profitable product combinations with exploration of alternative selections.

**Crossover Operator**

Single-Point Crossover was employed to create offspring:

* Mechanism:
  + Select a random crossover point in the chromosome (gene position)
  + Child 1 inherits genes before crossover point from Parent 1, genes after from Parent 2
  + Child 2 inherits genes before crossover point from Parent 2, genes after from Parent 1
* Rate: Crossover probability set to 0.8 (80%), meaning 80% of selected parent pairs undergo crossover while 20% pass unchanged to the next generation
* Purpose: Combines profitable product selection patterns from both parents while maintaining building blocks of good solutions

**Mutation Operator**

* Mechanism: For each gene (product selection) in a chromosome, with probability equal to the mutation rate, flip its binary value (0 to1 or 1to 0)
* Rate: Mutation probability set to 0.01 (1% per gene)
* Purpose:
  + Introduces genetic diversity by randomly adding/removing products from selection
  + Helps escape local optima by exploring new product combinations
  + Low rate prevents disrupting good solutions too aggressively
* Implementation: Each product has a 1% chance of having its selection status flipped

**Hyperparameter Tuning**

The following hyperparameters were tuned through experimental testing:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Value** | **Justification** |
| Population Size | 150 | Balances diversity with computational efficiency; large enough to explore search space |
| Mutation Rate | 0.01 | Provides sufficient exploration without disrupting good solutions too frequently |
| Crossover Rate | 0.7 | High enough to combine parent traits while maintaining some elite solutions |
| Number of Generations | 200 | Allows adequate convergence time while preventing excessive computation |
| Tournament Size | 3 | Moderate selection pressure that maintains diversity |

These parameters were selected based on standard GA practices and adjusted through iterative testing to achieve optimal performance.

## Convergence Tracking and Visualization

The GA implementation tracks convergence through following metric:

**Tracking Metric**:

* Best fitness value per generation: Maximum profit achieved in each generation is recorded and stored in a convergence list

**Visualization**:

A graph of a line

AI-generated content may be incorrect.

The convergence plot shows classical genetic algorithm behavior. Rapid initial improvement (from Gen 0 to 40) as good solutions in the random population are quickly developed, followed by steady exploration (from Gen 40 to 110) with incremental gains as the algorithm searches for better combinations. From Gen 110 to 200, the curve levels at approx. $23.6M, indicating convergence to a near-optimal solution with reduced population diversity. The curve is not decreasing again because it tracks the best solution found so far across all generations.

## Solution Evaluation

The final GA solution was achieved:

**Objective Value**:

The best solution achieved a total profit of $23,604,000 clearly demonstrating that the genetic algorithm effectively optimized the product mix to maximize profitability while fully respecting all resource constraints

**Constraint Satisfaction**:

* **Hard Constraints (Resource Capacity)**: Successfully satisfied with 100% feasibility.

|  |  |  |
| --- | --- | --- |
| Resource | Utilization (%) | Used / Available (Units) |
| CNC Machine Hours | 98.73% | 11,776 / 11,927 hours |
| Assembly Line Hours | 99.18% | 13,614 / 13,727 hours |
| Skilled Labor Hours | 99.72% | 11,519 / 11,551 hours |
| Raw Material Budget (USD) | 99.84% | $13,035 / $13,056 |
| Energy Consumption (kWh) | 94.92% | 12,776 / 13,460 kWh |

* **Solution Characteristics:** 
  + Products Selected: 28 out of 100 available products
  + All resource constraints satisfied (no violations)
  + High resource utilization (94-99% across all resources)
  + Feasibility: 100% produces a valid, implementable production plan

**Performance**: 54 hyperparameter combinations were tested during tuning, and the final run took 200 generations. The genetic algorithm showed good performance, consistently producing high quality solutions within a reasonable computational budget. The GA achieved near optimal use of available resources and guaranteed that all candidate solutions remained feasible throughout the evolutionary process by utilizing a repair-based constraint-handling technique.

# Model Implementation – Mixed-Integer Programming

This manufacturing problem is formulated as a Mixed-Integer Programming (MIP) optimization model and solved it using PuLP with CBC solver.

## MIP Problem Formulation

**Decision Variables**

x\_i ∈ {0, 1} for i = 1, 2, ..., 100

Where:

* x\_i = 1 if product i is selected for production
* x\_i = 0 if product i is not selected
* 100 binary variables total (one for each product SKU)

**Objective Function**

Resource **Capacity Constraints**

For each resource r ∈ {1, 2, 3, 4, 5}:

This ensures we don't exceed available capacity for:

|  |  |  |
| --- | --- | --- |
| Resource / Metric | Capacity | Unit |
| CNC Machine Hours | 11,927 | hours |
| Assembly Line Hours | 13,727 | hours |
| Skilled Labor Hours | 11,551 | hours |
| Raw Material Budget | 13,056 | USD |
| Energy Consumption | 13,460 | kWh |

**Binary Constraints:**

x\_i ∈ {0, 1} for all i

## Use of PuLP

**Solver Configuration:**

* Solver: CBC (COIN-OR Branch and Cut)
* Time Limit: 300 seconds (5 minutes)
* Optimization Direction: Maximization
* Solution Method: Branch-and-bound with linear programming relaxations

## Results Comparison: GA vs MIP

**MIP Solution Results**

|  |  |
| --- | --- |
| Metric | Value |
| Objective Value (Total Profit) | $24,381,000 |
| Products Selected | 29 products |
| Solver Status | Optimal |
| Solving Time | 1.004 seconds |
| Optimality | Proven global optimum |

MIP found the optimal solution $24,381,000 in just 1.004 seconds, while GA achieved optimal of $23,604,000 but took longer due to hyperparameter tuning.

**Comparison Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Genetic Algorithm (GA) | Mixed Integer Programming (MIP) | Winner |
| Solution Quality |  |  |  |
| Total Profit | $23,604,000 | $24,381,000 | MIP |
| Optimality Gap | 3.19% below optimal | 0% (Proven Optimal) | MIP |
| Products Selected | 28 products | 29 products | – |
| Constraint Satisfaction | All satisfied | All satisfied | Tie |
| Solution Quality Rating | Good (96.81% of optimal) | Perfect (100% optimal) | MIP |
| Profit Lost | $777,000 below optimal | $0 (optimal) | MIP |
| Computational Time |  |  |  |
| Execution Time | 73.08 seconds | 1.004 seconds | MIP |
| Predictability | Controllable (set generations) | Varies by problem | GA |
| Scaling Behavior | Linear / Polynomial | Exponential | GA |
| Optimality Guarantee | No guarantee | Proven optimal | MIP |

**Conclusion**

MIP is the standard for guaranteed optimal solutions, but GA offers an excellent practical alternative when speed matters or problems get too large. The 3.19% gap shows GA is quite effective for this type of problem.

# Comparison, Analysis & Insights

## Compare GA and MIP results

**Objective Function Value**

The MIP solver achieved a bit higher objective value of $24,381,000, compared to the Genetic Algorithm’s $23,604,000, resulting in an optimality gap of 3.19%. This shows that the MIP model found the true optimal solution, while the GA converged to a near optimal solution well within the typical and acceptable range for heuristic methods. The small gap confirms that the GA performed effectively in approximating the optimal result.

**Constraint Satisfaction / Feasibility**

Both methods gave feasible solutions that meet all the problem constraints.

* The MIP ensures feasibility because the solver (CBC) enforces the constraints mathematically during optimization.
* The GA kept solutions feasible by using repair or penalty mechanisms during evolution, showing it can handle constraints effectively within a metaheuristic approach.

So, both the MIP and GA provided valid, constraint-satisfying solutions to the manufacturing planning problem.

**Computational Time and Scalability**

The MIP solver (CBC) completed the optimization in approximately 1 second, while the Genetic Algorithm required around 73 seconds to converge.

* The short runtime of MIP highlights its efficiency for small- to medium-sized problems where an exact solution is computationally affordable.
* The GA, however, shows a better scalability. As problem size, number of resources, or constraint tightness increases, MIP solvers may experience an growth in solving time, whereas GA can still produce acceptable near-optimal solutions within a fixed computation time.

## Strengths and Weaknesses Analysis

**Genetic Algorithm (GA)**

**Strengths:**

* Found a near-optimal solution.
* Scalable for large or complex product selection problems.
* Effective constraint handling using repair or penalty methods.
* Works independently of mathematical structure, suitable for real-world manufacturing data.

**Weaknesses:**

* Slower computation compared to MIP.
* No guarantee of optimality results may vary per run.
* Sensitive to parameter tuning (population, mutation, crossover).
* Less interpretable due to stochastic, evolutionary process.

**Mixed-Integer Programming (MIP)**

**Strengths:**

* Achieved the optimal solution with zero optimality gap.
* Very fast computation for the given problem size.
* Guaranteed feasibility with strict constraint enforcement.
* Reproducible results for consistent decision-making.
* Easy to interpret due to clear mathematical formulation.

**Weaknesses:**

* Limited scalability: performance may drop for large or complex problems.
* Requires linear problem formulation, which may oversimplify real-world conditions.
* Solver performance can depend on model structure and number of constraints.

## How Problem Characteristics Affect Performance

* As problem size increases (Ex: more products or resources), the MIP solver may take significantly longer or fail to find an optimal solution within reasonable time limits due to combinatorial complexity. GA’s population based search can still find high quality solutions, making it more practical for large-scale applications.
* When constraints become tighter, MIP can still guarantee feasibility but might struggle with solution time. GA’s performance depends on how well constraint handling techniques (Ex: repair or penalty functions) are tuned.
* For highly nonlinear or non-convex objective functions (Ex: involving complex production dependencies), GA offers more flexibility, while MIP may be unsuitable or require simplifications.

# Critical Reflection

## Limitations of Models and Assumptions

**Genetic Algorithm Limitations**

Stochastic Nature:

* GA results vary between runs due to randomness in initialization, crossover, and mutation
* Our best solution ($23.6M) may not be reproducible exactly
* Impact: Less suitable for scenarios requiring deterministic decisions

Parameter Sensitivity:

* Solution quality heavily depends on hyperparameters (population size: 150, mutation rate: 0.01, crossover rate: 0.7, generations: 200)
* No guarantee these are optimal settings for our specific problem
* Impact: May require extensive tuning for different problem instances

Convergence Issues:

* Algorithm may plateau before reaching optimal
* No way to know if we're stuck in local optimum without running MIP
* Impact: Uncertainty about solution quality without benchmark

Constraint Handling:

* The GA employs a repair-based approach instead of penalty functions to handle constraints.
* Specifically, the repair\_chromosome() function systematically removes products from a solution until all resource constraints are satisfied.
* This repair step is applied before fitness evaluation, ensuring that every chromosome in the population is feasible at all times.

**MIP Limitations**

**Computational Scalability:**

* Currently solved 28-29 products in 1 second (excellent performance). However, MIP complexity grows exponentially with problem size
* For 1000+ products, solving time could reach hours or days

**Linear Assumptions:**

* Assumes linear objective function. (profit simply adds up)
* Assumes linear resource consumption (no economies/diseconomies of scale)
* In reality, manufacturing often has setup costs, volume discounts, learning curves
* May not capture true production economics

**Binary Decision Limitation:**

* Products are either fully produced or not produced
* No partial production quantities allowed
* In reality, many manufacturers can produce fractional batches
* Due to that may miss more profitable continuous solutions

**Key Assumptions Made**

|  |  |
| --- | --- |
| Assumption | Reality |
| Fixed Profit Margins | Profits vary with market conditions, discounts, order sizes |
| Deterministic Resource Usage | Actual consumption varies (defects, inefficiency, learning) |
| Independent Products | Some products may share components or tooling |
| Unlimited Demand | Each product has market demand limits |
| Single Production Run | No consideration of setup times between product switches |
| Perfect Information | Assumes we know exact profit and resource needs in advance |
| No Quality Constraints | All products equally important for customer satisfaction |

## Future Improvements

**Hybrid GA-MIP Approach**

Use the fast Genetic Algorithm (GA) to quickly find a near-optimal solution, then give that solution to the MIP solver as a starting point (“warm start”). Fix the product choices the GA is confident about, so MIP only optimizes the uncertain ones. This shrinks the problem size and can cut MIP solving time by 50–70%—especially useful for large problems—while still guaranteeing an optimal final solution.

**Parallel Computing**

Speed up both GA and MIP by using multiple CPU cores. In GA, evaluate different individuals in the population at the same time (Ex: 8 cores = 4 to 8 times faster). Modern MIP solvers like Gurobi or CPLEX already support parallel processing and can run 2 to 4 times faster.

**Adaptive GA Parameters**

Instead of using fixed settings, let the GA adjust its own parameters during the run. For example, increase mutation when progress and decrease it when converging. Also, start with a large population for exploration and shrink it later for faster convergence. This can reduce the time to converge by 20 to 30%.

**Simulated Annealing (SA)**

SA is a simpler, probabilistic optimization method inspired by cooling metal. It sometimes accepts worse solutions early on to avoid getting stuck in local optima. With fewer parameters than GA, SA is easier to implement and useful for quick testing or as a baseline for comparison.

## Real-World Applicability and Business Impact

**Current Model Applicability:**

* This optimization framework helps manufacturers make better decisions when allocating limited resources like labor, machines, and materials across many products to maximize profit. Just in this case, choosing the best mix instead of a sub-optimal one meant an extra $777,000 in profit, for larger companies with hundreds of products, that could mean millions more per year.
* The hybrid GA-MIP approach offers the best of both worlds
  + Use MIP for weekly or monthly planning when you need a proven optimal solution.
  + Use GA for fast what-if analysis or quick replanning when demand, costs, or capacity change suddenly.
* If connected to an ERP system, this could cut production planning time from days to minutes and boost resource use.
* The same method also works beyond manufacturing for example, in investment portfolio selection, project resource allocation, and supply chain decisions like assigning warehouses to customers.

# Appendix:

## Question 1

* A data-driven approach to predict the success of bank telemarketing: [*https://www.sciencedirect.com/science/article/abs/pii/S016792361400061X*](https://www.sciencedirect.com/science/article/abs/pii/S016792361400061X)
* Using customer lifetime value and neural networks to improve the prediction of bank deposit subscription in telemarketing campaigns: *https://www.semanticscholar.org/paper/Using-customer-lifetime-value-and-neural-networks-Moro-Cortez/e73ca5fd0d221168054a4f30196f952ee19916f1*
* Predicting customer subscription in bank telemarketing campaigns using ensemble learning models: *https://www.sciencedirect.com/science/article/pii/S2666827025000015*
* Data-Driven Decision-Making for Bank Target Marketing Using Supervised Learning Classifiers on Imbalanced Big Data: *https://www.sciencedirect.com/org/science/article/pii/S1546221824007483*
* Bayesian Regression for Predicting Subscription to Bank Term Deposits in Direct Marketing Campaigns: [*https://arxiv.org/html/2410.21539*](https://arxiv.org/html/2410.21539)

## Question 2

* Kaggle Dataset - Energy Consumption and Weather (Spain):

[*https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generation-prices-and-weather/data*](https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generation-prices-and-weather/data)

* Deep Learning for Time Series Forecasting (Dimitre Oliveira)

*https://www.kaggle.com/code/dimitreoliveira/deep-learning-for-time-series-forecasting*

* Time Series Forecasting with Machine Learning (Rob Mulla)

*https://www.kaggle.com/code/robikscube/time-series-forecasting-with-machine-learning-yt*

* Time Series Analysis with Prophet (Elena Petrova)

*https://www.kaggle.com/code/elenapetrova/time-series-analysis-and-forecasts-with-prophet*

* Spanish Wind Power Forecasting (Troncoso et al., 2018)

*https://link.springer.com/chapter/10.1007/978-3-319-96944-2\_24*

## Question 3

* What is a Genetic Algorithm in Manufacturing

[*https://eyelit.ai/what-is-genetic-algorithm/*](https://eyelit.ai/what-is-genetic-algorithm/)

* Applying Genetic Algorithms to the Optimization of Production Planning in a Real-World Manufacturing Environment:

[*https://www.researchgate.net/publication/2876219\_Applying\_Genetic\_Algorithms\_to\_the\_Optimization\_of\_Production\_Planning\_in\_a\_Real-World\_Manufacturing\_Environment*](https://www.researchgate.net/publication/2876219_Applying_Genetic_Algorithms_to_the_Optimization_of_Production_Planning_in_a_Real-World_Manufacturing_Environment)

* Production Optimization in a Grain Facility through Mixed-Integer Linear Programming:

[*https://www.mdpi.com/2076-3417/12/16/8212*](https://www.mdpi.com/2076-3417/12/16/8212)

* Mixed-Integer Linear Programming Model for Production Planning: A Case Study at Sawn Timber Production:

[*https://www.researchgate.net/publication/350759335\_Mixed-Integer\_Linear\_Programming\_Model\_for\_Production\_Planning\_A\_Case\_Study\_at\_Sawn\_Timber\_Production*](https://www.researchgate.net/publication/350759335_Mixed-Integer_Linear_Programming_Model_for_Production_Planning_A_Case_Study_at_Sawn_Timber_Production)

* A Mixed-Integer Linear Programming (MILP) for Garment Line Balancing:

[*https://arxiv.org/abs/2502.17508*](https://arxiv.org/abs/2502.17508)