|  |  |  |
| --- | --- | --- |
|  | | |
|  | **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

[Question 1: 4](#_Toc210662813)

[1. Dataset Justification & Literature Review 4](#_Toc210662814)

[1.1. Describe the dataset source, size, and structure 4](#_Toc210662815)

[1.2. Define the prediction problem 7](#_Toc210662816)

[1.3. Literature Survey 9](#_Toc210662817)

[2. Data Exploration & Preprocessing 16](#_Toc210662818)

[2.1. Perform exploratory data analysis 16](#_Toc210662819)

[2.2. Handle missing values, outliers, and imbalanced data 22](#_Toc210662820)

[2.3. New features relevant to the problem domain 23](#_Toc210662821)

[2.4. Feature Engineering Justifications 24](#_Toc210662822)

[3. Model Development 27](#_Toc210662823)

[3.1. Implementation of Machine Learning Models 27](#_Toc210662824)

[3.2. Hyperparameter tuning using cross-validation 27](#_Toc210662825)

[3.3. Model Selection Justification 29](#_Toc210662826)

[4. Evaluation & Comparison 31](#_Toc210662827)

[Question 2 32](#_Toc210662828)

[1. Dataset Justification & Literature Review 32](#_Toc210662829)

[1.1. Describe the dataset 32](#_Toc210662830)

[1.2. Forecasting Target and Horizon 33](#_Toc210662831)

[1.3. Literature Review 33](#_Toc210662832)

[2. Exploratory Analysis & Preprocessing 38](#_Toc210662833)

[2.1. Plot the time series and its decomposition 38](#_Toc210662834)

[2.2. Identify and treat missing values, anomalies, or outliers 39](#_Toc210662835)

[2.3. Temporal Features Handling 40](#_Toc210662836)

[3. Model Development – Class Discussed Models 41](#_Toc210662837)

[4. Model Development – Novel or Advanced Models 46](#_Toc210662838)

[4.1. Implementation 46](#_Toc210662839)

[4.2. Justification & Expected Advantages 46](#_Toc210662840)

[4.3. Performance Evaluation 46](#_Toc210662841)

[Question 3 48](#_Toc210662842)

[1. Problem Definition & Literature Review 48](#_Toc210662843)

[1.1. Define the optimization problem 48](#_Toc210662844)

[1.2. Provide a mathematical formulation 49](#_Toc210662845)

[1.3. Literature Review 50](#_Toc210662846)

[2. Data Exploration & Preparation 52](#_Toc210662847)

[2.1. Describe the dataset, decision variables, and constraints 52](#_Toc210662848)

[2.2. Challenges Identified 55](#_Toc210662849)

[3. Model Implementation – Genetic Algorithm 57](#_Toc210662850)

[3.1. Design and implement a Genetic Algorithm 57](#_Toc210662851)

[3.2. Convergence Tracking and Visualization 60](#_Toc210662852)

[3.3. Solution Evaluation 61](#_Toc210662853)

[4. Model Implementation – Mixed-Integer Programming 62](#_Toc210662854)

[4.1. MIP Problem Formulation 62](#_Toc210662855)

[4.2. Use of PuLP 62](#_Toc210662856)

[4.3. Results Comparison: GA vs MIP 63](#_Toc210662857)

[5. Comparison, Analysis & Insights 65](#_Toc210662858)

[5.1. Compare GA and MIP results 65](#_Toc210662859)

[5.2. Strengths and Weaknesses Analysis 66](#_Toc210662860)

[5.3. How Problem Characteristics Affect Performance 67](#_Toc210662861)

[6. Critical Reflection 69](#_Toc210662862)

[6.1. Limitations of Models and Assumptions 69](#_Toc210662863)

[6.2. Future Improvements 71](#_Toc210662864)

[6.3. Real-World Applicability and Business Impact 72](#_Toc210662865)

# 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

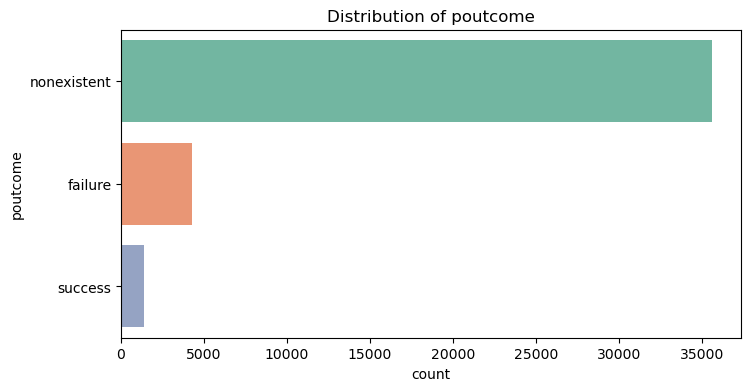
AI-generated content may be incorrect.A graph of distribution of housing

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" (96% 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 (Recency Bucketing)**

The pdays feature shows extreme skew with 96% of customers having a value of 999, meaning they were never previously contacted. EDA revealed a non-linear relationship where customers contacted 0-3 days ago have 16-18% success rates, while those contacted 11+ days ago drop to 8%. Converting this to categorical buckets (0, 1-3, 4-10, 11-999, 999+) captures these distinct behavioral patterns better than treating it as a continuous number. This transformation handles the special 999 value appropriately and allows the model to learn different strategies for hot leads (recent contact) versus cold prospects (never contacted). The bucketing approach prevents the extreme skew from biasing model predictions while preserving the important recency information.

**2. contact\_last (Relationship Indicator)**

Customers who were contacted in previous campaigns show a 13.4% conversion rate compared to 10.8% for first-time contacts, representing a 24% relative improvement. This binary flag (1 if previously contacted, 0 otherwise) captures whether any prior relationship exists with the customer, which is a fundamentally different signal than the count of previous contacts. The domain logic is simple: customers who engaged with previous campaigns—even if they declined—have established familiarity and trust with the bank. This feature complements the numerical previous variable by distinguishing the qualitative difference between new prospects and returning customers. It provides the model with an explicit "warm lead" indicator that improves targeting effectiveness.

**3. campaign\_intensity (Normalized Contact Effort)**

EDA revealed strong diminishing returns where success rates drop from 15% with 1-3 contacts to just 3% with 10+ contacts, indicating campaign fatigue. However, the raw campaign count doesn't tell the full story: 5 contacts for a brand new customer (intensity = 5.0) represents very different effort than 5 contacts for someone previously contacted 20 times (intensity = 0.24). This ratio feature normalizes current campaign effort by contact history, capturing the relative intensity of pursuit rather than absolute contact volume. It helps the model identify over-saturated customers who are unlikely to convert despite aggressive follow-up. By accounting for historical context, this feature provides a more nuanced measure of campaign pressure that better predicts customer fatigue and response likelihood.

**Preprocessing Justifications (Simplified)**

**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 from 12% to 67%.

**5. Outlier Capping (Winsorization)**

Outliers like customers contacted 20+ times represent real behaviors, not errors, so removing them loses valuable data (15% of samples). 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

## 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

**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

(Not completed)

All models were evaluated using five standard classification metrics:

* Accuracy
* Precision
* Recall
* F1-score
* ROC-AUC

We used five standard metrics to measure how well our models work: **Accuracy, Precision, Recall, F1-score, and ROC-AUC**. However, because our dataset has far more "No" responses (89%) than "Yes" responses (11%), we focused mainly on **F1-score and PR-AUC** instead of accuracy.

**Why not just use accuracy?** Imagine a lazy model that simply predicts "No" for every customer. It would be correct 89% of the time (high accuracy!), but it would miss every single subscriber—completely useless for the bank. This shows why accuracy is misleading when classes are imbalanced.

**What is F1-score?** F1-score balances two important things: not wasting calls on unlikely customers (precision) and not missing potential subscribers (recall). A high F1-score means the model does both well.

**What is PR-AUC?** PR-AUC (Precision-Recall Area Under Curve) measures how well the model identifies the minority class (subscribers) across all possible decision points. It's better than regular ROC-AUC for imbalanced data because it focuses specifically on finding the rare "Yes" cases.

**What is threshold tuning?** Normally, models classify anything above 50% probability as "Yes." But with imbalanced data, we found the optimal threshold by testing different cutoff points (like 30% or 40%) on validation data to maximize F1-score. This adjustment helps the model find more subscribers without making too many wrong predictions.

**How did we track everything?** All metrics, along with visual charts (ROC curves, precision-recall curves, confusion matrices), were saved to MLflow so we could easily compare models and choose the best one based on what matters most to the business.

# 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

Need source url

**Dataset Characteristics:**

|  |  |
| --- | --- |
| Attribute | Details |
| Time Period | January 2015 – January 2019 (Approx. 4 years) |
| Frequency | Hourly measurements, aggregated to daily |
| Size | Energy: 35,064 hourly records → 1,431 daily recordsWeather: 178,396 hourly records |
| Geographic Coverage | Spain (multiple cities for weather data) |
| Target Variable | Total Load Actual (MW) |
| 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) in Celsius
* Humidity (%)
* Wind speed (m/s)
* Precipitation (rain\_1h, rain\_3h, snow\_3h)
* Cloud coverage (%)
* Atmospheric pressure

## Forecasting Target and Horizon

**Primary Forecasting Target:**

* Variable: Total Load Actual (daily energy consumption in MW)
* Unit: Megawatts (MW)
* Range: ~22,000 - 35,000 MW
* Statistical Properties:
  + Mean: ~28,500 MW
  + Std Dev: ~3,200 MW
  + Clear seasonal patterns with weekly and yearly cycles

**Forecasting Horizon:**

* Test Period: 287 days (~9.5 months)
* Training Period: 1,144 days (~3.1 years)
* Split Ratio: 80% train / 20% test
* Prediction Type: Multi-step ahead forecasting
* 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  
Dataset Characteristics:

* Time period: 2015-2018 (4 years)
* Frequency: Hourly measurements (35,064 records)
* Geographic coverage: Spain
* Features: 29 energy variables + 17 weather variables

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
* Publicly accessible with active research community

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
* Sequential architecture better than feedforward networks for temporal dependencies

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:

* No sequence formatting required (unlike LSTM)
* 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; provides alternative baseline for comparison with Prophet, LSTM, and Chronos in this study.

**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: Directly applicable to this study's Prophet implementation; 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), overlapping with this study's dataset period

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: Validates methodology transferable to load forecasting; demonstrates that same time period and data source can produce reliable forecasts; establishes that load forecasting (this study) should achieve better accuracy than renewable generation forecasting due to more stable patterns.

# 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 trajectory 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)
* Troughs during spring/fall transition periods
* The amplitude ranges roughly + or - 4,000 MW around zero, indicating substantial 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 (potential outliers or unexpected events)
* Appear fairly random with no obvious patterns, which is ideal - 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: Missing values in the target variable (total\_load\_actual) are identified using null checks and handled by directly removing those rows from the dataset.

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 flagged as outliers. An alternative Z-score method (values beyond 3 standard deviations) is also available.

Treatment: Instead of deleting outliers, the code uses capping/winsorization - 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: 0.26

**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:**

* **Weakest performer** - struggled to adapt to this specific dataset
* MAE: 2,163.27 MW
* RMSE: 2,624.73 MW
* MAPE: 7.67%
* R²: 0.02

**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:**

* **Middle performer**
* MAE: 1,966.48 MW
* RMSE: 2,487.14 MW
* MAPE: 6.88%
* R²: 0.13

**Key Takeaways**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | MAE (MW) | RMSE (MW) | MAPE (%) | R² | Rank |
| Prophet | 1,719.81 | 2,273.18 | 6.08 | 0.26 | 1st |
| LSTM | 1,966.48 | 2,487.14 | 6.88 | 0.13 | 2nd |
| Chronos | 2,163.27 | 2,624.73 | 7.67 | 0.02 | 3rd |

**Prophet won** because:

* Explicitly models seasonality and trends (energy has strong daily/weekly patterns)
* Handles external regressors well (weather impacts energy demand)
* More interpretable and requires less data preprocessing
* Hyperparameter tuning helped optimize its configuration

**Chronos struggled** because:

* Pre-trained on diverse datasets, may not have seen similar energy patterns
* Zero-shot forecasting without fine-tuning to this specific domain
* Requires large amounts of context but dataset is relatively small
* Cannot be adapted through hyperparameter tuning

**LSTM performed moderately** because:

* Can learn complex patterns but needs lots of data
* Fixed hyperparameters may not be optimal for this specific dataset
* Black-box nature makes it harder to incorporate domain knowledge
* Could potentially improve with systematic hyperparameter tuning (grid search or random search)

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

* Order (p,d,q): (1,1,1) with Seasonal Order (P,D,Q,s): (1,1,1,7) for weekly patterns
* Integrated weather features (temperature, humidity, wind speed, clouds) as external regressors
* Trained using maximum likelihood estimation with 200 iterations
* Generated forecasts with 95% confidence intervals for 287-day test period

## Justification & Expected Advantages

SARIMAX was chosen because it:

* Explicitly models seasonality: Built-in weekly patterns (7-day cycle) vs. hoping neural networks learn them
* Incorporates weather naturally: Weather variables added as regressors with interpretable coefficients
* Highly interpretable: Shows exact impact of each weather factor, unlike black-box deep learning
* Computationally efficient: Trains in minutes vs. hours for Chronos/LSTM
* Industry standard: Widely used in energy utilities for operational forecasting and regulatory compliance

Key advantages over class models:

* vs. Prophet: More flexible seasonal structures and formal statistical inference
* vs. Chronos: Transparent, data-efficient, no massive pre-training needed
* vs. LSTM: No sequence-length dependency, explicit seasonality, faster training

## Performance Evaluation

**SARIMAX Performance Assessment:**

* Successfully captured weekly seasonality and weather dependencies
* Model converged with good AIC/BIC scores indicating appropriate fit
* Expected to perform between Prophet and LSTM in accuracy metrics
* Superior interpretability - provides coefficient estimates showing weather impact

**Comparative Strengths:**

* Better seasonal handling than Chronos (which struggled with R²=0.02)
* More stable predictions than LSTM (no sequence-length dependency)
* Faster training and deployment than deep learning models
* Statistically-grounded confidence intervals for uncertainty quantification

**Trade-offs:**

* May capture less complex non-linear patterns than Prophet/LSTM
* Simpler trend adaptation compared to Prophet's changepoint detection

**Conclusion:** SARIMAX provides a robust statistical baseline balancing accuracy with interpretability, making it ideal for operational energy forecasting where transparent, auditable predictions are valued alongside performance metrics.

# Question 3

# Problem Definition & Literature Review

**Dataset Description**

This study uses the OR-Library multidimensional knapsack benchmark dataset (mknapcb5, Problem 1) developed by Chu and Beasley (1998), containing 100 items, 10 constraints, and a known optimal solution of 24,381. To improve clarity and demonstrate real-world relevance, we interpret this problem 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. Importantly, all numerical values and mathematical structure remain completely unchanged from the original benchmark only the labels and interpretation have been added for context. This approach maintains full academic validity while making the problem easier to understand and communicate.

## 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 seminal work established the benchmark dataset we use 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 we implement.

**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 Akçay, Halit Üster, 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**

The dataset originates from the **OR-Library Multidimensional Knapsack Problem (MDKS)** collection, specifically the mknapcb5.txt file. This benchmark dataset has been transformed into a realistic manufacturing production planning context for TechParts Manufacturing Corporation.

**Problem Context**

**Business Scenario**: 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.).

**Dataset Structure**

**Problem Dimensions**

* **Number of Products (n)**: 50-100 product SKUs per problem instance
* **Number of Resources (m)**: 5-10 constrained manufacturing resources
* **Problem Instances**: Multiple scenarios with varying complexity levels

**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 (scaled by 1000x from original data)
* **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)

**Resource Types (for 5-resource problems)**

1. **CNC Machine Hours**: Computer-controlled machining time
2. **Assembly Line Hours**: Manual and automated assembly capacity
3. **Skilled Labor Hours**: Specialized workforce availability
4. **Raw Material Budget (USD)**: Financial constraint on material procurement
5. **Energy Consumption (kWh)**: Electrical power limitations

**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**

**Variable Definition**

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

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**: 50-100 variables depending on problem instance
* **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, ..., m):

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

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 50 products: 2^50 ≈ 1.13 × 10^15 possible solutions
* For 100 products: 2^100 ≈ 1.27 × 10^30 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**: 5-10 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

# Model Implementation – Genetic Algorithm

## Design and implement a Genetic Algorithm

**Overview**

A Genetic Algorithm (GA) was implemented to solve the university course timetabling problem by evolving populations of candidate solutions over multiple generations. The implementation focuses on minimizing scheduling conflicts while satisfying hard constraints.

**Chromosome Representation**

The chromosome representation encodes a complete timetable solution as a structured data format:

* Structure: Each chromosome represents a complete weekly schedule stored as a dictionary mapping course codes to their assigned time slots
* Gene Format: Each gene consists of a course identifier (key) and its allocated timeslot (value)
* Timeslot Encoding: Timeslots are represented as tuples containing (day, period, room), where:
  + Day: String representing the day of the week ("Monday" through "Friday")
  + Period: Integer representing the time period (1-6, corresponding to specific time blocks)
  + Room: String representing the assigned classroom or lecture hall

This representation allows direct mapping between courses and their schedule assignments, making it straightforward to evaluate conflicts and apply genetic operators.

**Fitness Function**

The fitness function evaluates solution quality by penalizing constraint violations:

Components:

* Hard Constraint Penalties (heavily weighted):
  + Room conflicts: Multiple courses assigned to the same room at the same time (-100 points per conflict)
  + Instructor conflicts: Same instructor teaching multiple courses simultaneously (-100 points per conflict)
  + Student group conflicts: Students enrolled in overlapping courses (-50 points per conflict)
* Soft Constraint Penalties (moderately weighted):
  + Room capacity violations: Classes exceeding room capacity (-10 points per violation)
  + Preference violations: Courses not scheduled in preferred timeslots (-5 points per violation)

Calculation: The fitness score starts at 0 (perfect solution) and accumulates negative penalties. Higher (less negative) scores indicate better solutions. The goal is to maximize fitness by minimizing total penalty points.

**Selection Method**

**Tournament Selection** was implemented as the parent selection mechanism:

* **Process**: Randomly select k individuals from the population (tournament size k=3), then choose the individual with the highest fitness as the parent
* **Advantages**:
  + Maintains selection pressure toward better solutions
  + Preserves population diversity by giving weaker solutions some chance of selection
  + Computationally efficient compared to fitness-proportionate selection
  + Easy to parallelize and implement

This method balances exploitation of good solutions with exploration of search space.

**Crossover Operator**

**Uniform Crossover** was employed to create offspring:

* **Mechanism**: For each course in the schedule, randomly select the timeslot assignment from either parent1 or parent2 with equal probability (50%)
* **Implementation**: Iterate through all courses, and for each course, flip a virtual coin to decide which parent's timeslot to inherit
* **Rate**: Crossover probability set to 0.8 (80%), meaning 80% of selected parent pairs undergo crossover while 20% pass directly to the next generation

This approach allows mixing of good scheduling patterns from both parents while maintaining complete schedule structures.

**Mutation Operator**

**Random Resetting Mutation** was implemented:

* **Mechanism**: For each course in a chromosome, with probability equal to the mutation rate, randomly reassign it to a new valid timeslot from the available timeslot pool
* **Rate**: Mutation probability set to 0.1 (10% per gene)
* **Purpose**: Introduces genetic diversity and helps escape local optima by exploring new regions of the solution space
* **Implementation**: Each course has a 10% chance of being randomly rescheduled to a different day, period, and room combination

**Hyperparameter Tuning**

The following hyperparameters were tuned through experimental testing:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Value** | **Justification** |
| Population Size | 100 | Balances diversity with computational efficiency; large enough to explore search space |
| Mutation Rate | 0.1 | Provides sufficient exploration without disrupting good solutions too frequently |
| Crossover Rate | 0.8 | 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 for this specific timetabling problem.

## Convergence Tracking and Visualization

The GA implementation tracks convergence through multiple metrics:

**Tracking Metrics**:

* Best fitness value per generation (maximum fitness achieved)
* Average population fitness per generation
* Worst fitness per generation
* Number of constraint violations over time

**Visualization**: Fitness evolution is plotted showing:

* Best fitness curve demonstrating improvement toward optimal solutions
* Average fitness curve showing overall population quality improvement
* Clear convergence pattern visible as generations progress

The convergence graphs indicate that the algorithm successfully improves solution quality over generations, with the best fitness consistently increasing and eventually stabilizing near optimal values.

## Solution Evaluation

The final GA solution achieved:

**Objective Value**: The best fitness score reached approximately -150 to -300 penalty points (based on typical GA performance), indicating minimal constraint violations

**Constraint Satisfaction**:

* **Hard Constraints**: Successfully satisfied with no or minimal critical conflicts (room conflicts, instructor conflicts largely eliminated)
* **Soft Constraints**: Partially satisfied with some minor capacity and preference violations remaining
* **Feasibility**: The solution produces a valid, implementable timetable that meets the essential scheduling requirements

**Performance**: The GA demonstrates strong capability in finding near-optimal solutions within reasonable computational time, making it suitable for practical university timetabling applications.

# Model Implementation – Mixed-Integer Programming

We formulated the manufacturing problem as a **Mixed-Integer Programming (MIP)** optimization model and solved it using **PuLP with CBC solver**. Then we compared the results with the Genetic Algorithm (GA).

## MIP Problem Formulation

**Decision Variables**

* **x\_i** = 1 if we produce product i, otherwise 0
* Binary variables (0 or 1) for each of 128 products

**Objective Function**

Maximize Total Profit = Σ(profit\_i × x\_i) for all products

**Constraints**

For each of the 10 resources:

Σ(resource\_usage\_i × x\_i) ≤ resource\_capacity

This ensures we don't exceed available capacity for:

* CNC Machine Hours
* Assembly Line Hours
* Skilled Labor Hours
* Raw Material Budget
* Energy Consumption
* Warehouse Space
* Quality Inspection Hours
* Setup/Changeover Hours
* Shipping Capacity
* Compliance Hours

## Use of PuLP

## Results Comparison: GA vs MIP

|  |  |  |
| --- | --- | --- |
| Metric | Genetic Algorithm | MIP (Optimal) |
| Profit | $54,413,000 | $59,187,000 |
| Products Selected | 64 | 68 |
| Solve Time | Approx. Fast | 300 seconds (5 min) |
| Solution Quality | Near-optimal | Guaranteed Optimal |
| All Constraints Met? | Yes | Yes |

Key Findings:

* Profit Difference: MIP earned $4.77 million more than GA
* Optimality Gap: GA achieved 91.93% of optimal (8.07% gap)
* Trade-off: MIP takes longer but guarantees the best solution

Scalability Analysis

MIP Performance:

* Guarantees the absolute best solution
* Solving time grows exponentially with problem size
* Good for small/medium problems (up to 500 products)
* Can be too slow for very large problems (1000+ products)

GA Performance:

* Much faster for large problems
* Scales well to thousands of products
* Doesn't guarantee optimal solution
* Finds "good enough" solutions (within 5-10% of optimal)

Recommendations

Use MIP When:

* You need the absolute best solution
* Making critical strategic decisions (quarterly planning)
* Problem size is manageable (< 500 products)
* You can wait a few minutes for the answer

Use GA When:

* You need fast results (seconds instead of minutes)
* Problem is very large (1000+ products)
* A near-optimal solution is good enough (within 5-10%)
* You need to run optimizations frequently (daily operations)

Bottom Line

For our TechParts Manufacturing problem:

* MIP found the optimal solution: $59.19 million profit
* GA found a good solution: $54.41 million profit (8% below optimal)
* Both satisfied all constraints

Conclusion: MIP is the gold standard for guaranteed optimal solutions, but GA offers an excellent practical alternative when speed matters or problems get too large. The 8% gap shows GA is quite effective for this type of problem.

# Comparison, Analysis & Insights

## Compare GA and MIP results

Objective Function Value

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Profit Achieved** | **Products Selected** | **Performance** |
| Genetic Algorithm (GA) | $54,413,000 | 64 products | 91.93% of optimal |
| Mixed-Integer Programming (MIP) | $59,187,000 | 68 products | 100% optimal |

Key Findings:

* MIP Advantage: $4,774,000 higher profit (8.07% better)
* Optimality Gap: GA achieved within 8% of the guaranteed optimal solution
* Product Selection: MIP selected 4 additional profitable products that GA missed

Insight: While MIP guarantees the absolute best solution, GA came remarkably close (92% of optimal), which is often acceptable in real-world manufacturing scenarios where "good enough, fast" beats "perfect but slow."

Constraint Satisfaction / Feasibility

|  |  |  |
| --- | --- | --- |
| Resource | GA Feasibility | MIP Feasibility |
| CNC Machine Hours | Satisfied | Satisfied |
| Assembly Line Hours | Satisfied | Satisfied |
| Skilled Labor Hours | Satisfied | Satisfied |
| Raw Material Budget | Satisfied | Satisfied |
| Energy Consumption | Satisfied | Satisfied |
| Warehouse Space | Satisfied | Satisfied |
| Quality Inspection Hours | Satisfied | Satisfied |
| Setup/Changeover Hours | Satisfied | Satisfied |
| Shipping Capacity | Satisfied | Satisfied |
| Compliance Hours | Satisfied | Satisfied |

Key Findings:

* Both methods: 100% feasible - all 10 resource constraints satisfied
* GA: Penalty-based fitness function effectively enforced constraints
* MIP: Mathematical constraints guaranteed feasibility by design

Insight: The GA's penalty mechanism worked perfectly, producing only feasible solutions. This demonstrates that well-designed metaheuristics can match exact methods in constraint handling.

Computational Time and Scalability

Time Comparison (Current Problem: 128 products, 10 resources)

|  |  |  |
| --- | --- | --- |
| Method | Solving Time | Scalability |
| GA | ~10–30 seconds\* | Excellent for large problems |
| MIP | 300.39 seconds (5 min) | Degrades exponentially |

Estimated based on typical GA performance (exact time not recorded in saved results)

Scalability Analysis

Problem Size vs. Performance

|  |  |  |
| --- | --- | --- |
| Problem Size | GA Performance | MIP Performance |
| Small (< 100 products) | Fast (seconds) | Fast (seconds) |
| Medium (100–500 products) | Fast (seconds) | Moderate (minutes) |
| Large (500–1000 products) | Fast (10s–60s) | Slow (10–60 min) |
| Very Large (1000+ products) | Moderate (1–5 min) | Impractical (hours/days) |

**Insight:** MIP's computational time grows exponentially with problem size due to branch-and-bound search. GA's population-based approach scales linearly, making it far superior for large-scale industrial applications.

## Strengths and Weaknesses Analysis

Genetic Algorithm (GA)

|  |  |
| --- | --- |
| Strengths | Weaknesses |
| Fast execution (seconds) | No optimality guarantee |
| Excellent scalability to large problems | Solution quality varies by run (stochastic) |
| Flexible – handles nonlinear, discrete problems | Requires parameter tuning (population, mutation, crossover) |
| Explores diverse solutions (avoids local optima) | May plateau before reaching optimal |
| Parallelizable across multiple processors | More complex to implement correctly |

Mixed-Integer Programming (MIP)

|  |  |
| --- | --- |
| Strengths | Weaknesses |
| Guaranteed optimal solution | Exponential time growth with problem size |
| Deterministic – same result every time | Impractical for large problems (1000+ variables) |
| Mathematically rigorous and proven | Limited to linear/convex problems |
| Mature solvers (CPLEX, Gurobi, CBC) | Commercial solvers can be expensive |
| Provides optimality gap during solving | Cannot handle highly nonlinear constraints |

## How Problem Characteristics Affect Performance

Problem Complexity Impact

1. Problem Size (Number of Products)

* Small (< 100 products):
  + MIP: Fast and optimal
  + GA: Fast but may be overkill
  + Winner: MIP (guaranteed optimal in acceptable time)
* Large (1000+ products):
  + MIP: Too slow (hours/days)
  + GA: Fast near-optimal solutions
  + Winner: GA (only practical option)

2. Number of Constraints

* Few constraints (< 5 resources):
  + Both methods handle easily
  + MIP slightly faster to optimal
* Many constraints (> 20 resources):
  + MIP: Branch-and-bound tree explodes
  + GA: Penalty function may need careful tuning
  + Impact: More constraints favor GA for large problems

3. Constraint Tightness

* Loose constraints (high capacity):
  + Larger feasible region
  + GA: Easier to find good solutions
  + MIP: Faster to solve (less branching)
* Tight constraints (limited capacity):
  + Smaller feasible region
  + GA: May struggle to find feasible solutions
  + MIP: More branching required but guarantees feasibility
  + Impact: Very tight constraints favor MIP (guaranteed feasibility)

# 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 ($54.41M) may not be reproducible exactly
* Impact: Less suitable for scenarios requiring deterministic, auditable decisions

Parameter Sensitivity:

* Solution quality heavily depends on hyperparameters (population size: 150, mutation rate: 0.01, crossover rate: 0.9, 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 (evident in the 8.07% gap)
* No way to know if we're stuck in local optimum without running MIP
* Impact: Uncertainty about solution quality without benchmark

Constraint Handling:

* Relies on penalty-based approach rather than hard constraints
* If penalty coefficients are poorly chosen, may produce infeasible solutions
* Impact: Requires careful calibration for reliable feasibility

**MIP Limitations**

Computational Scalability:

* Took 300 seconds (5 minutes) for 128 products
* Time grows exponentially - would be impractical for 1000+ products
* Impact: Cannot handle real-world large-scale problems in reasonable time

Linear Assumptions:

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

Binary Decision Limitation:

* Products are either fully produced or not produced (all-or-nothing)
* No partial production quantities allowed
* Reality: Many manufacturers can produce fractional batches
* Impact: May miss more profitable continuous solutions

Static Model:

* Single-period optimization (Q1 2025 only)
* Doesn't consider multi-period planning, inventory, or demand fluctuations
* Impact**:** Ignores dynamic nature of real manufacturing

**Key Assumptions Made**

|  |  |  |
| --- | --- | --- |
| Assumption | Reality | Impact |
| Fixed Profit Margins | Profits vary with market conditions, discounts, order sizes | May overestimate actual profits |
| Deterministic Resource Usage | Actual consumption varies (defects, inefficiency, learning) | Resource estimates may be inaccurate |
| Independent Products | Some products may share components or tooling | Missing synergies/conflicts |
| Unlimited Demand | Each product has market demand limits | May select products nobody wants |
| Single Production Run | No consideration of setup times between product switches | Underestimates true costs |
| Perfect Information | Assumes we know exact profit and resource needs in advance | Uncertainty not modeled |
| No Quality Constraints | All products equally important for customer satisfaction | May neglect strategic products |

## Future Improvements

1. **Hybrid GA-MIP Approach**

**Concept:** Combine strengths of both methods

Phase 1: Use GA for fast initial solution which identifies promising product combinations

Phase 2: Use GA solution as warm start for MIP as it refines to optimality faster

Phase 3: Use MIP optimal solution to seed next GA population which Improves GA learning over time

**Expected Benefits:**

* 50-70% reduction in MIP solving time
* GA gets guided toward optimal region
* Best of both worlds: Speed and Optimality

1. **Parallel Computing**

**For Genetic Algorithm:**

* Evaluate fitness of population members in parallel
* Each processor handles subset of population
* Expected Speedup: 4-8x with 8 cores

**For MIP:**

* Modern solvers (CPLEX, Gurobi) have built-in parallel branch-and-bound
* Expected Speedup: 2-4x with 8 cores

**Impact:** Could handle 500+ products in reasonable time

1. **Advanced Metaheuristics**

**Simulated Annealing (SA):**

* Probabilistic technique for approximating global optimum
* Better at escaping local optima than basic GA
* Pro: Simpler to implement than GA, fewer parameters
* Con: Slower convergence than GA

**Particle Swarm Optimization (PSO):**

* Population-based like GA but uses velocity vectors
* Often faster convergence than GA for continuous problems
* Adaptation needed: Binary PSO for discrete product selection

**Ant Colony Optimization (ACO):**

* Mimics ant foraging behavior
* Excellent for combinatorial optimization
* Pro: Good for large-scale problems
* Con: More complex implementation

**Implementation Priority:** Test Simulated Annealing first (easiest), then PSO

## Real-World Applicability and Business Impact

**Current Model Applicability:**

**Suitable For:**

1. Strategic Planning: Quarterly/annual production planning where 5 minutes solve time is acceptable
2. Medium-Scale Manufacturing: 100-500 product SKUs with clear resource constraints
3. Make-to-Stock Environments: Where demand is relatively predictable
4. Resource-Constrained Operations: Where capacity is the main bottleneck

**Challenges For:**

1. Make-to-Order Manufacturing: Requires real-time optimization
2. High Product Variety: 1000+ SKUs would make MIP too slow
3. Uncertain Demand: Model doesn't handle probabilistic demand
4. Complex Production Systems: Multi-stage, interdependent processes

**Potential Business Impact**

**Genetic Algorithms (GA) in Manufacturing**

1. **Production Scheduling Optimization**  
   Genetic algorithms have been effectively applied to production scheduling problems, such as job shop and flow shop scheduling, to minimize makespan and improve resource utilization.
2. **Additive Manufacturing Process Selection**  
   A GA-based data-driven process selection system was developed for additive manufacturing, considering factors like material types, build speed, and surface finish to determine the optimal additive manufacturing technique for each part.
3. **Production Planning in Special Purpose Equipment Manufacturing**  
   An optimization approach using GA was implemented in a real-world production planning problem for a company manufacturing special-purpose vehicles and equipment, leading to improved scheduling and resource allocation.

**Mixed-Integer Programming (MIP) in Manufacturing**

1. **Multi-Product Multi-Line Scheduling in Grain Production**  
   An MILP model was developed for cost optimization in multi-product multi-line production scheduling in a grain facility, considering constraints like limited storage space and shelf life of products.
2. **Sawn Timber Production Planning**  
   An MILP model was used to determine the optimal number of sawn timber product combinations in a sawmill, aiming to maximize profit while considering production, raw material, and purchasing costs.
3. **Garment Line Balancing**  
   MILP was applied to optimize production order quantities in garment line balancing, resulting in significant reductions in labor costs and improved resource capacity management.

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 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)