# Machine Learning Project Pipeline

Nisha Chothe

# Index


- 1. Define a Business Goal
- 2. ML Problem Framing
- 3. Data Processing
- 4. Model Training and Deployment
- 5. Monitoring


## 1. Define a Business Goal

This stage focuses on clearly articulating the business problem and how a machine learning solution will provide value, establishing the scope and key success metrics.

| Component | Description | Reference |
| --- | --- | --- |
| Business Problem | Define the primary challenge or opportunity to be addressed. | File |
| Success Criteria | Quantitative metrics used to measure the business success (e.g., cost reduction, revenue increase, user engagement). | File |
| Stakeholders | List of primary stakeholders and their expectations. | File |

## 2. ML Problem Framing

This stage translates the defined business goal into a specific, actionable machine learning problem (e.g., classification, regression, clustering).

| Component | Description | Reference |
| --- | --- | --- |
| ML Task Type | The specific machine learning task (e.g., Binary Classification, | File |

| Component | Description | Reference |
| --- | --- | --- |
| | Multi-class Classification, Time Series Regression). | |
| Target Variable | The variable the model is intended to predict. | File |
| Target Metric | The technical evaluation metric for model performance (e.g., F1-Score, RMSE, AUC). | File |
| Initial Baseline | The performance of a simple, non-ML method or a simple model (e.g., predicting the mean, or majority class). | File |

## 3. Data Processing

This stage encompasses all steps required to collect, clean, transform, and prepare the raw data for model training and evaluation.

| Component | Description | Reference |
| --- | --- | --- |
| Data Source & Acquisition | The origin of the data (e.g., database, API) and the script for data extraction. | File |
| Data Cleaning | Methods for handling missing values, outliers, and errors (e.g., imputation strategy). | File |
| Feature Engineering | Creation of new variables from raw data to improve model predictive power. | File |
| Data Split | The strategy for splitting data into training, validation, and testing sets. | File |

# 4. Model training and Deployment

This stage involves selecting the final model, packaging it, and integrating it into the production environment for real-time predictions.

| Component | Description | Reference |
| --- | --- | --- |
| Final Model | The specific algorithm and hyperparameters chosen after validation. | File |
| Serving Environment | The platform used to host the model for inference (e.g., cloud platform, on-premise server). | File |
| API Specification | Definition of the model's prediction endpoint, including input and output schemas. | File |
| Rollout Plan | The strategy for releasing the model into production (e.g., A/B testing, Canary release). | File |

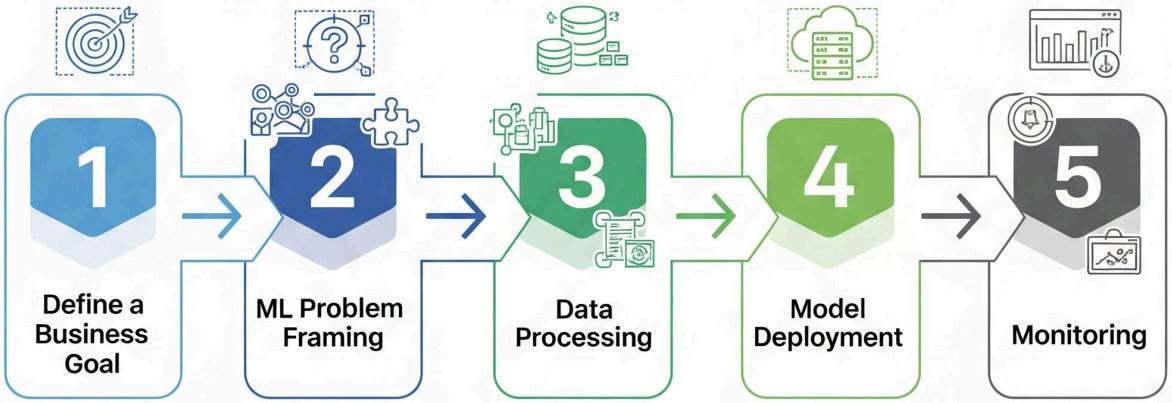
# 5. Monitoring

| Component | Description | Reference |
| --- | --- | --- |
| Performance Monitoring | Tracking the model's technical metrics (e.g., latency, throughput) in production. | File |
| Business Metric Monitoring | Tracking the impact of the model on the primary business success criteria. | File |

| Component | Description | Reference |
| --- | --- | --- |
| Data/Concept Drift Detection | Mechanisms to detect changes in the input data distribution or the relationship between inputs and the target. | File |
| Retraining Strategy | The plan for automatically or manually retraining and redeploying the model. | File |
| Review Meeting | Scheduled discussion to review live model performance and decide on maintenance actions. | Calendar event |

This final stage ensures that the deployed model maintains its performance and is retired or retrained when necessary.

# Machine Learning Project Pipeline



# AWS Spot Instance Risk Prediction -

## Dataset Pre-Processing

# Contents


1. Introduction
2. Dataset Acquisition
3. Data Cleaning and Preparation
    a. Handling Missing Values
    b. Outlier Detection and Treatment
    c. Data Type Conversion
4. Feature Engineering
    a. Creating Time-Based Features
    b. Encoding Categorical Variables
    c. Feature Scaling
5. Data Splitting (Train/Validation/Test)
6. Conclusion and Next Steps

# 1. Introduction

Dataset pre-processing is the foundational step for any successful machine learning project, acting as the critical link between raw data and a high-performing model. Without proper preparation, even the most sophisticated algorithms can yield misleading or inaccurate results, following the principle of "garbage in, garbage out." Pre-processing ensures the data is clean and in an optimal format for learning. This involves essential Data acquisition, Data Cleaning, Preparation, feature engineering and Dataset splitting. The meticulous pre-processing eliminates noise, improves data quality, and structures the features, directly leading to faster training times, better model convergence, and, most importantly, more accurate and reliable predictive performance.

For the AWS spot instance Termination risk prediction project, we have time series pricing data where it is even more critical to preserve the temporal features, analyze them and reveal patterns.
For many time series ML algorithms, it is important to preserve long term and short term trends, seasonality, upsampling (daily data -> hourly data) and downsampling (mean, sum or last value).

The AWS pricing data is taken for over 985 unique instance types across 17 regions. This results in 35000+ unique pools. This results in multivariate time series data. Synchronization makes sure that time series are recorded at the exact same time. Also we need to find the interdependencies and co-relation between two series. While leading with feature engineering, there is a need to create temporal features such as cyclic features and holiday indicators, lag features and rolling window features. Handling missing values and outliers is also an important part as transformers are sensitive to NaN values. After the necessary features are created, the dataset is then split into Train, Test and Validation dataset for training, hyperparameter tuning and testing on unseen data.

# 2. Data acquisition

**Data acquisition** in machine learning is the initial and crucial process of gathering raw, relevant data from defined sources. It is the starting point of the entire ML pipeline, as the quality and quantity of the acquired data directly determine the eventual performance and reliability of the predictive model. The key steps involve defining data requirements, data sourcing, data collection and extraction and the final step is data storage in required format.  It is important to find possible class imbalance, bias and fairness.

## What does AWS actually provide?

AWS exposes several APIs for spot-related decisions, but each one has restrictions that affect ML projects. The relevant signals/features that are useful in spot prediction are Spot placement score(SPS), Interruption frequency (IF), spot price, On demand price for required regions, and AZ.

**SPS scores** are the likelihood that your requested set of spot instances can be allocated immediately. It is not historical, predictive or stable across time. You can access them through API. AWS computes it *internally* based on capacity, demand, and allocation strategies, then returns only the final score. SPS is good for allocation decisions at the moment, not for predicting stability trends across pools. It hides underlying signals your model needs—interruptions, price volatility, demand trends.
**IF score** is posted by AWS per instance family per region. It is not available through API. Problem with IF score is that AWS does not expose time of interruptions per pool or per customer information.
**Spot instance price** history is an important feature for our prediction. You can access it through the API calls. Limitation with the API method is that AWS only stores 90 days of history and long term prediction is impossible. The large scale querying also leads to high costs.
**On demand pricing** data can be accessed through AWS pricing API of static JSON files provided by AWS. APIs can be slow and data extraction requires large JSON blobs.
API calls are costly and have large computational overhead. Also storing large files is expensive.

## Using Open source AWS instance pricing data

**Spot instance termination** data is not provided by AWS. It is usually difficult to catch the interruptions and can lead to unnecessary instance running costs. Also we need an extensive amount of interruption data across many instance pools. Hence it is not practical to run the instances in hopes of collecting the interruption data at the start of the project. Looking for an open source historical data is the practical way.
The paper  SpotLake: Diverse Spot Instance Dataset Archive Service, suggests that the factors affecting the AWS spot termination are widely dependent on market trends, seasonal trends and events. The paper aims to predict spot instance termination by using SPS values and termination rates. To achieve that they have collected the historical AWS data since 2021 for a variety of instance families and pools. Our aim is to use the on demand and spot pricing data

provided by them to capture the trends in pricing and use them to predict risk zones for particular instance pools.

## Dataset

The AWS raw data is taken from over 17 regions and 134 unique instance families by using API calls. Data is fetched and logged for every 10 minutes for over 3 years. This results in massive data extraction with over 3 million data points. Storing this large amount of data requires massive data storage and computational resources.
Following table shows raw dataset summary -

| Metric | Count |
|---|---|
| Total Unique Regions | 17 |
| Total Unique AZs | 55 |
| Total Unique Instance Types | 985 |
| Instance Families | 134 |
| Total Unique Pools | 35114 |

| region | num_pools |
|---|---|
| ap-northeast-1 | 2353 |
| ap-northeast-2 | 1748 |
| ap-northeast-3 | 933 |
| ap-south-1 | 1670 |
| ap-southeast-1 | 2052 |
| ap-southeast-2 | 2108 |
| ca-central-1 | 1532 |
| eu-central-1 | 2444 |
| eu-north-1 | 1521 |
| eu-west-1 | 2366 |
| eu-west-2 | 1653 |
| eu-west-3 | 1278 |
| sa-east-1 | 1492 |
| us-east-1 | 4571 |
| us-east-2 | 2659 |
| us-west-1 | 1142 |
| us-west-2 | 3592 |

# 3. Data cleaning and Preparation

The datasets are created from raw data available according to the algorithm requirement. Instead of storing entire data, we extract the data from URLs and do further processing such as feature selection, Temporal feature creation and handling missing values.
As the dataset contains over 900 instance types and multiple AZ, we need to extract the features and data points for custom training requirements. Hence we have created sub datasets from raw data. Now various techniques are used to dataset preparation as follows -

- Handling Missing Values

  This component involves methods for addressing any gaps or missing observations within the dataset, such as an imputation strategy. For the time series data in this project, handling missing values is particularly important because the document notes that transformers are sensitive to NaN values. Proper handling ensures the data streams are complete for accurate time-based analysis.

  The 2024 Mumbai region data does not contain any missing values. In case of 2023 data, there are 20K missing availability zones (AZ). As knowing AZ is important for instance migration, it is important to remove the datapoints with missing values. As these data points are only 0.03 % of total data points, removing them will not affect the prediction.

- Outlier Detection and Treatment

  Outliers are data points that significantly deviate from other observations. The project includes methods for outlier detection and treatment as part of the overall data cleaning process. Identifying and mitigating the impact of these extreme values is vital because they can skew statistical analysis and model training, especially in time series data where they might represent market anomalies or data logging errors.

  In the both 2023 and 2024 data, there are multiple instances where the spot price is higher than on demand price which is impossible in practice. Also there are many instances where spot prices are not available at the moment. Hence they are labelled -1 instead of 0 or NaN. They are removed as they are distributed over multiple instances.

  In 2023 data, the instances below have negative savings values. As for instances in the U-6tb1 family, they are used for a niche, specific purpose: large in-memory databases like SAP HANA. Hence removing them will not cause any problem. For instance t2.nano, need to pay attention to its prediction.

  t2.nano: 6819 occurrences,

  u-6tb1.112xlarge: 6819 occurrences

  u-6tb1.56xlarge: 6819 occurrences

  Also the number of unique pools for 2023 are 1357 and for 2024 data are 1577 as each year AWS infrastructure and monitoring instance types increases. Hence it is important to

make sure to take pools with maximum number of data points to visualise the trends well.

- Data Type Conversion

  This process ensures that all variables in the dataset are in the correct format for the machine learning algorithms. For the AWS pricing data, which includes information like instance types, regions, and pricing figures, Data Type Conversion ensures that numerical data is treated numerically (e.g., for calculations) and categorical data is correctly encoded (e.g., for feature engineering steps like encoding categorical variables).

  Total columns: 8

  Column names: ['Time', 'InstanceType', 'Region', 'AZ', 'OndemandPrice', 'SpotPrice', 'Savings', 'timestamp']

  Data types:

| Column Name | Data Type |
|---|---|
| Time | object |
| InstanceType | object |
| Region | object |
| AZ | object |
| OndemandPrice | float64 |
| SpotPrice | float64 |
| Savings | int64 |
| timestamp | object |