

# Data Management for Machine Learning - Assignment Repository

This repository contains the assignment for the subject **Data Management for Machine Learning**. The objective of this assignment is to demonstrate the skills and knowledge acquired during the course.

## Group Members

- Rishabh Setiya 2023AC05268
- Deepti Mahajan 2023AC05918
- Mariyam Bhoira 2023AC05328
- Narayan Saha 2023AC05494

## Video Recording

You can watch the video recording of our project presentation [here](#).

The link to Github repository is

<https://github.com/rishabhsetiya/DataMgmt4MLassignment.git>

## 1. Problem Formulation

### Business Problem

Customer churn occurs when an existing customer stops using a company's services or purchasing its products. Addressable churn, which can be mitigated through strategic interventions, leads to revenue losses, increased customer acquisition costs, and a negative impact on brand reputation. The primary goal is to predict customer churn and develop proactive strategies to enhance customer retention.

### Key Business Objectives

- Reduce customer churn by identifying at-risk customers early.
- Improve customer retention strategies through predictive insights.
- Minimize revenue loss by leveraging data-driven decision-making.
- Automate the data processing pipeline for scalability and efficiency.

## Expected Outputs from the Pipeline

1. *Clean datasets for Exploratory Data Analysis (EDA)*
  - o Remove missing values and duplicates
  - o Normalize and standardize features
  - o Handle categorical variables
2. *Transformed features for machine learning*
  - o Feature engineering (aggregated metrics, derived attributes)
  - o Feature selection (important predictors of churn)
  - o Encoding and scaling
3. *Deployable model for customer churn prediction*
  - o Train ML models (Logistic Regression, Random Forest, Neural Networks, etc.)
  - o Evaluate performance using key metrics
  - o Deploy the best-performing model

## Measurable Evaluation Metrics

- *Accuracy*: Measure the proportion of correct predictions.
- *Precision & Recall*: Balance false positives and false negatives.
- *F1 Score*: Harmonic mean of precision and recall for imbalanced datasets.
- *ROC-AUC Score*: Evaluate the discriminative power of the model.
- *Model Interpretability*: Feature importance analysis.

# The various phases of the data pipeline

## 1. Data Ingestion

In this phase, we got the data from two different sources :

(i) *Kaggle Dataset ([Telco Customer Churn](#))*

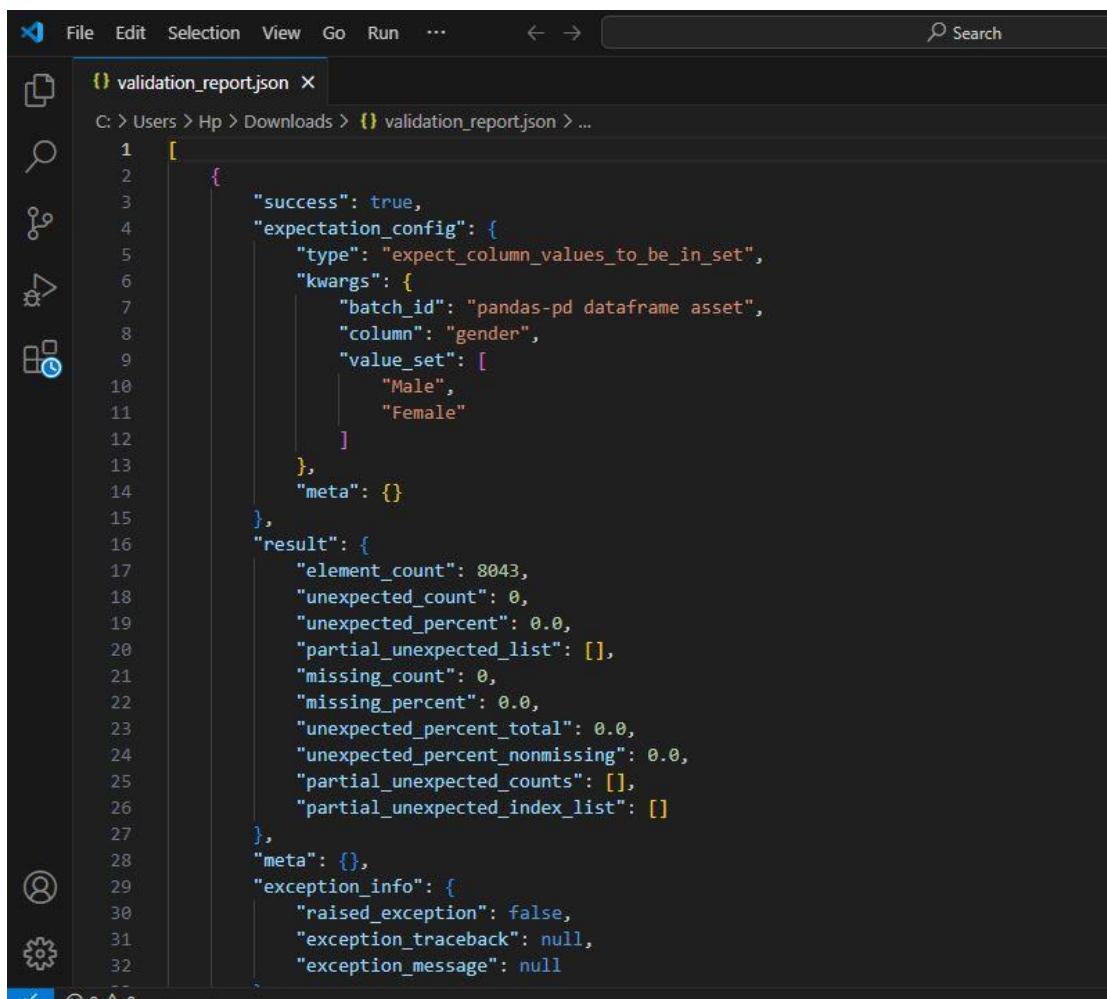
- o Customer demographics
- o Service subscription details
- o Monthly charges and tenure
- o Churn indicator

(ii) *Custom API Endpoint (Data extraction and feature engineering)*

- o Aggregated customer activity data
- o Real-time interaction metrics
- o Custom features derived from transactional and behavioral data

## 2. Data Validation

Data validation is applied using **Great Expectations** to ensure quality. The validation report is saved in validation.json file. The screenshot of the file :



A screenshot of a code editor showing a JSON file named "validation\_report.json". The file path is C:\Users\Hp\Downloads\validation\_report.json. The JSON content is as follows:

```
1 [ { 2   "success": true, 3   "expectation_config": { 4     "type": "expect_column_values_to_be_in_set", 5     "kwargs": { 6       "batch_id": "pandas-pd_dataframe asset", 7       "column": "gender", 8       "value_set": [ 9         "Male", 10        "Female" 11       ] 12     }, 13     "meta": {} 14   }, 15   "result": { 16     "element_count": 8043, 17     "unexpected_count": 0, 18     "unexpected_percent": 0.0, 19     "partial_unexpected_list": [], 20     "missing_count": 0, 21     "missing_percent": 0.0, 22     "unexpected_percent_total": 0.0, 23     "unexpected_percent_nonmissing": 0.0, 24     "partial_unexpected_counts": [], 25     "partial_unexpected_index_list": [] 26   }, 27   "meta": {}, 28   "exception_info": { 29     "raised_exception": false, 30     "exception_traceback": null, 31     "exception_message": null 32   } }
```

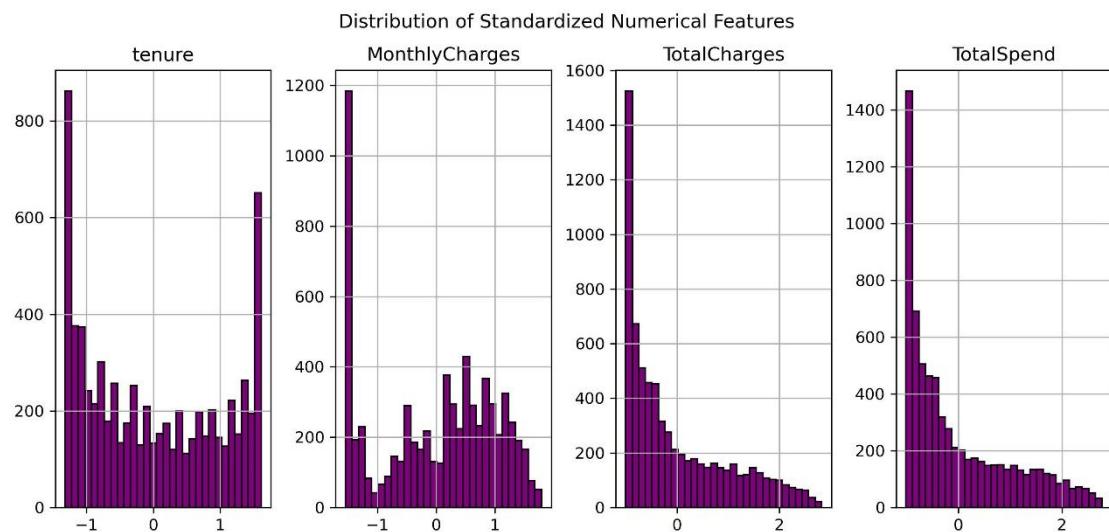
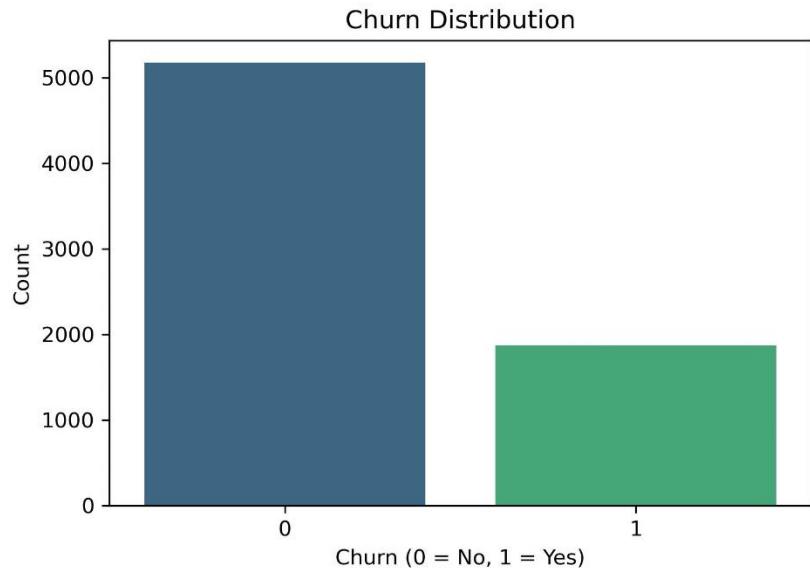
### **3. Preprocessing the Data**

We applied transformations to ensure the data is clean, structured, and suitable for machine learning models. The following transformations were applied on the dataset before saving it to database.

- “**TotalCharges**” is stored as an **object (string)** instead of a numeric value. So, we converted it to a numeric field.
- We are imputing missing values in Numerical columns with mean and missing values in Categorical columns with mode.
- Checked for duplicate rows and deleting if any exists.
- Converted Tenure to Categories since Tenure is numerical, but customer retention is often linked to different periods (short-term, medium-term, long-term customers).
- Created a feature “**TotalSpend**” to know customer spending behaviour
- Created a binary column “**HasInternet**” so that we can get useful information
- Created a column “**NumServices**” which counts for the number of services the user has opted for.
- Standardized the numerical features to bring them into same scale otherwise it might impact the ML model
- Encoded all the categorical columns with label encoder.

## Exploratory Data Analysis

Some of the images of the visualizations that are saved during the analysis are as follows:



#### 4. Storing the data to SQL Server

The data is saved to SQL Server. Sample queries to retrieve transformed data

##### (i) Summary

SQLQuery2.sql - DESKTOP-1JVEQTF.TELCO\_CHURN\_DB (sa (57)) - Microsoft SQL Server Management Studio

Object Explorer

SQLQuery2.sql - D...CHURN\_DB (sa (57))

```
--Summary
SELECT
    ROUND(MIN(TotalCharges),4) AS MinTotalCharges,
    ROUND(MAX(TotalCharges),4) AS MaxTotalCharges,
    ROUND(MIN(TotalSpend),4) AS MinTotalSpend,
    ROUND(MAX(TotalSpend),4) AS MaxTotalSpend
FROM telco_churn_table;
```

Results

	MinTotalCharges	MaxTotalCharges	MinTotalSpend	MaxTotalSpend
1	-1.746	2.8265	-1.3717	3.2238

##### (ii) Distribution by Tenure Category

SQLQuery2.sql - DESKTOP-1JVEQTF.TELCO\_CHURN\_DB (sa (57)) - Microsoft SQL Server Management Studio

Object Explorer

SQLQuery2.sql - D...CHURN\_DB (sa (57))

```
--Distribution by Tenure category
SELECT TenureCategory, COUNT(*) AS Count
FROM telco_churn_table
GROUP BY TenureCategory
ORDER BY Count DESC;
```

Results

TenureCategory	Count
1	3109
0	2582
2	2352

### (iii) Churn Distribution

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery2.sql - DESKTOP-1JVEQTF.TELCO\_CHURN\_DB (sa (57)) - Microsoft SQL Server Management Studio". The Object Explorer pane on the left shows the database structure for "DESKTOP-1JVEQTF (SQL Server 12.0.200)". The "Tables" node under "TELCO\_CHURN\_DB" is expanded, showing "System Tables", "FileTables", and "dbo.telco\_churn\_table". The "dbo.telco\_churn\_table" node is selected. The main query editor window contains the following SQL code:

```
--Churn Distribution
SELECT Churn, COUNT(*) AS Count,
       ROUND(COUNT(*)) * 100.0 / (SELECT COUNT(*) FROM telco_churn_table), 2) AS Percentage
FROM telco_churn_table
GROUP BY Churn;
```

The results grid shows the following data:

Churn	Count	Percentage
1	5708	70.970000000000
2	2335	29.030000000000

### (iv) To check if a particular payment method leads to higher spending

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery2.sql - DESKTOP-1JVEQTF.TELCO\_CHURN\_DB (sa (57)) - Microsoft SQL Server Management Studio". The Object Explorer pane on the left shows the database structure for "DESKTOP-1JVEQTF (SQL Server 12.0.200)". The "Tables" node under "TELCO\_CHURN\_DB" is expanded, showing "System Tables", "FileTables", and "dbo.telco\_churn\_table". The "dbo.telco\_churn\_table" node is selected. The main query editor window contains the following SQL code:

```
--If particular payment method leads to highr spending
SELECT PaymentMethod,
       ROUND(AVG(TotalSpend), 2) AS AvgTotalSpend
FROM telco_churn_table
GROUP BY PaymentMethod
ORDER BY AvgTotalSpend DESC;
```

The results grid shows the following data:

PaymentMethod	AvgTotalSpend
0	0.31
1	0.3
2	-0.08
3	-0.48

## 5. Feature Store

To ensure efficient feature management and reuse, we utilized the **Hopsworks Feature Store** as a centralized repository for storing the features. Screenshot of the feature store is as follows

