

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)*
 - Remove missing values and duplicates
 - Normalize and standardize features
 - Handle categorical variables
2. *Transformed features for machine learning*

- Feature engineering (aggregated metrics, derived attributes)
 - Feature selection (important predictors of churn)
 - Encoding and scaling
3. *Deployable model for customer churn prediction*
 - Train ML models (Logistic Regression, Random Forest, Neural Networks, etc.)
 - Evaluate performance using key metrics
 - 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

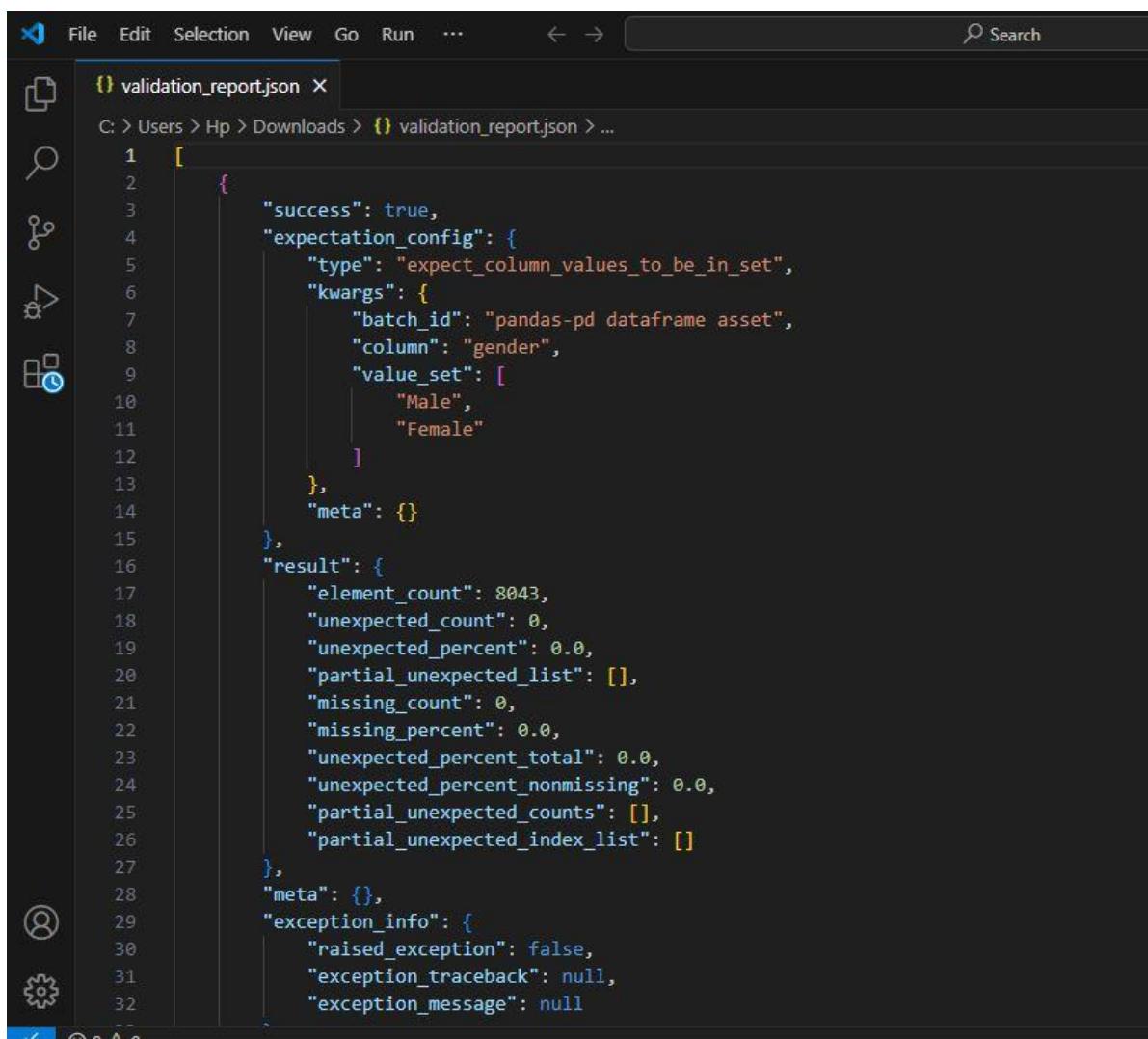
1. Data Ingestion

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

- (i) *Kaggle Dataset ([Telco Customer Churn](#))*
 - Customer demographics
 - Service subscription details
 - Monthly charges and tenure
 - Churn indicator
- (ii) *Custom API Endpoint* (Data extraction and feature engineering)
 - Aggregated customer activity data
 - Real-time interaction metrics
 - 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 :



The screenshot shows a code editor window with a dark theme. The file being viewed is "validation_report.json". The code content is as follows:

```
File Edit Selection View Go Run ... ← → Search
validation_report.json X
C: > Users > Hp > Downloads > validation_report.json > ...
1 [
2   {
3     "success": true,
4     "expectation_config": {
5       "type": "expect_column_values_to_be_in_set",
6       "kwargs": {
7         "batch_id": "pandas-pd_dataframe asset",
8         "column": "gender",
9         "value_set": [
10           "Male",
11           "Female"
12         ],
13       },
14       "meta": {}
15     },
16     "result": {
17       "element_count": 8043,
18       "unexpected_count": 0,
19       "unexpected_percent": 0.0,
20       "partial_unexpected_list": [],
21       "missing_count": 0,
22       "missing_percent": 0.0,
23       "unexpected_percent_total": 0.0,
24       "unexpected_percent_nonmissing": 0.0,
25       "partial_unexpected_counts": [],
26       "partial_unexpected_index_list": []
27     },
28     "meta": {},
29     "exception_info": {
30       "raised_exception": false,
31       "exception_traceback": null,
32       "exception_message": null
33     }
34   }
35 ]
```

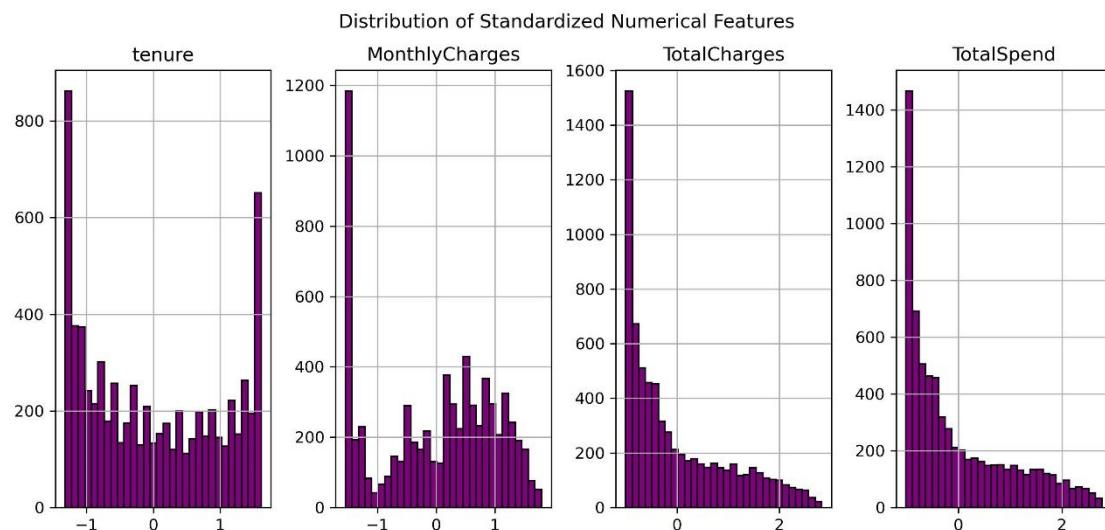
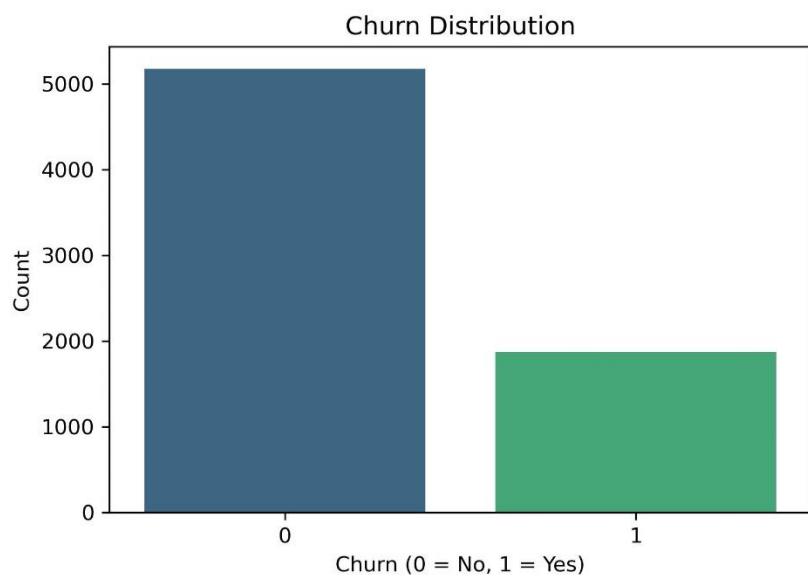
3. Preprocessing the Data

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

File Edit View Query Project Debug Tools Window Help

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 Messages

	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

File Edit View Query Project Debug Tools Window Help

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 Messages

	TenureCategory	Count
1	1	3109
2	0	2582
3	2	2352

(iii) Churn Distribution

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the TELCO_CHURN_DB database and its tables. The central pane displays a query titled '--Churn Distribution' which calculates the count and percentage of churn levels. The results show two rows: one for Churn level 0 with a count of 5708 and a percentage of 70.97%, and another for Churn level 1 with a count of 2335 and a percentage of 29.03%.

```
--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;
```

Churn	Count	Percentage
1	5708	70.97000000000000
2	2335	29.03000000000000

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

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the TELCO_CHURN_DB database and its tables. The central pane displays a query titled '--If particular payment method leads to highr spending' which calculates the average total spend for each payment method. The results show four rows: PaymentMethod 0 with AvgTotalSpend 0.31, PaymentMethod 1 with 0.3, PaymentMethod 2 with -0.08, and PaymentMethod 3 with -0.48.

```
--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;
```

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

The screenshot shows the Hopsworks Feature Store interface. The left sidebar contains navigation links: Home, Feature Store, Feature Groups (selected), Feature Views, Storage Connectors, Compute, Jupyter, Ingestions, Airflow, Data Science, Model Registry, and Deployments. The main content area is titled "CustomerChurn" and shows a search bar with "Find a Feature Group...". It displays "1 feature groups" and lists "customer_churn_features". This entry is described as "Customer churn features from SQL database". It shows a version of "1" and was last updated "3 minutes ago". A "feature loggi" button is visible on the right.