**🎯 Objective**

To build an end-to-end supervised machine learning solution that **predicts Customer\_Churn** using a structured classification dataset.

**📂 Input Files (available in root directory):**

1. synthetic\_classification\_dataset.csv – the dataset for modeling
2. synthetic\_classification\_data\_dictionary.docx – contains detailed field-level descriptions

**🔧 Step-by-Step Instructions**

**✅ Step 1: Understand the Dataset**

* Read and interpret synthetic\_classification\_data\_dictionary.docx
* Summarize the purpose of each feature and target variable (Customer\_Churn)
* Document the data understanding as a **brief summary (DOCX + PDF)** in the eda/ folder

**✅ Step 2: Import and Inspect the Data**

* Load synthetic\_classification\_dataset.csv
* Display the following:
  + Number of rows and columns
  + Column names, data types
  + Missing values (count and %)
  + Number of unique values per column
  + Basic statistics (mean, std, min, max for numerics)
* Save this **initial inspection report** to eda/data\_overview.docx and eda/data\_overview.pdf

**✅ Step 3: Exploratory Data Analysis (EDA)**

* Perform **univariate analysis**:
  + Histograms or bar plots for all variables
  + Summary tables (frequency/counts for categoricals, distribution for numericals)
* Perform **bivariate analysis**:
  + Churn vs other features (box plots, grouped bars, correlation, etc.)
* Clearly mark any skewed, imbalanced, or non-informative fields
* Save all **charts, code snippets, and insights** to eda/eda\_report.docx and eda/eda\_report.pdf

**✅ Step 4: Feature Engineering**

* Identify and process:
  + Missing values (drop/impute with strategy)
  + Outliers (detect and treat if needed)
  + Categorical features: encode (One-Hot or Label Encoding based on cardinality)
  + Numerical features: scale/normalize (if needed)
  + New feature creation (e.g., ratios, bins, interactions if relevant)
* Document all **steps, code, and transformed dataset summary** in feature-engineering/feature\_engineering.docx and feature-engineering/feature\_engineering.pdf

**✅ Step 5: Train-Test Split**

* Split data into **80% train and 20% test sets**
* Ensure stratified sampling for Customer\_Churn
* Save:
  + train.csv and test.csv in the model-data/ folder

**✅ Step 6: Logistic Regression Modeling**

* Build a logistic regression model using the training data
* Evaluate using:
  + Accuracy, Precision, Recall, F1-Score, ROC-AUC
  + Confusion Matrix and ROC Curve
* Save all outputs, including **model coefficients, feature importance**, and evaluation plots to:
  + model-performance/logistic\_regression\_report.docx
  + model-performance/logistic\_regression\_report.pdf

**✅ Step 7: XGBoost Modeling**

* Train an XGBoost classifier on the same training data
* Evaluate using the same metrics as Step 6
* Save outputs to:
  + model-performance/xgboost\_report.docx
  + model-performance/xgboost\_report.pdf

**✅ Step 8: Model Comparison**

* Compare Logistic Regression and XGBoost on:
  + Performance metrics (Accuracy, F1, AUC)
  + Fairness (bias across gender, city, etc.)
  + Explainability (SHAP/LIME if applicable)
* Summarize:
  + Strengths and weaknesses of each model
  + Recommendation on which model to use and why
* Save comparative summary to:
  + model-performance/model\_comparison\_summary.docx
  + model-performance/model\_comparison\_summary.pdf

**✅ Step 9: Save Final Model and Scoring Script**

* Save the best model (as .pkl or .joblib) to model/final\_model.pkl
* Write a Python **scoring script** (score.py) that:
  + Loads the model
  + Accepts input (single row or batch)
  + Outputs churn prediction
* Save to model/score.py

**📌 Output Folder Structure Summary**

├── eda/

│ ├── data\_overview.docx

│ ├── data\_overview.pdf

│ ├── eda\_report.docx

│ └── eda\_report.pdf

├── feature-engineering/

│ ├── feature\_engineering.docx

│ └── feature\_engineering.pdf

├── model-data/

│ ├── train.csv

│ └── test.csv

├── model-performance/

│ ├── logistic\_regression\_report.docx

│ ├── logistic\_regression\_report.pdf

│ ├── xgboost\_report.docx

│ ├── xgboost\_report.pdf

│ ├── model\_comparison\_summary.docx

│ └── model\_comparison\_summary.pdf

├── model/

│ ├── final\_model.pkl

│ └── score.py