## Project Roadmap Project No 1

## Customer Churn Prediction

Objective:  
Predict whether a customer will leave (churn) based on their behavior and interaction history.

### Step 1: Problem Understanding

* Description:  
  Define the business goal — reduce churn rate by predicting at-risk customers using historical and behavioral data.

### Step 2: Data Collection

* Description:  
  Acquire relevant customer data including demographics, service usage patterns, billing information, and customer feedback.
* Sources:

https://www.kaggle.com/datasets/blastchar/telco-customer-churn/data

### Step 3: Data Preprocessing

* Description:  
  Clean the dataset by handling missing values, encoding categorical variables, managing outliers, and standardizing numerical features.
* Tools:  
  Python: Pandas, NumPy, Scikit-learn , matplotlib.pyplot ,seaborn

Reasons :

* (LabelEncoder, OneHotEncoder, SimpleImputer)

### Step 4: Exploratory Data Analysis (EDA)

* Description:  
  Explore data distributions, churn patterns, relationships between features, and key customer segments contributing to churn.
* Tools:  
  Python: Matplotlib, Seaborn

### Step 5: Feature Engineering

* Description:  
  Create new meaningful features such as average purchase value, tenure categories, complaint frequency, and interaction score.
* Tools:  
  Python: Pandas, NumPy, Scikit-learn

### Step 6: Model Selection & Training

* Description:  
  Choose appropriate classification models to predict churn, train them on preprocessed data, and compare performance.
* Candidate Models:  
  Logistic Regression, Decision Tree, Random Forest, XGBoost, LightGBM
* Tools:  
  Python: Scikit-learn, XGBoost, LightGBM

### Step 7: Model Evaluation

* Description:  
  Evaluate the model’s performance using metrics like Accuracy, Precision, Recall, F1-Score, and AUC-ROC to select the best model.
* Tools:  
  Scikit-learn: metrics module (classification\_report, confusion\_matrix, roc\_auc\_score)

### Step 8: Hyperparameter Tuning

* Description:  
  Optimize model parameters using Grid Search or Random Search techniques to improve performance.
* Tools:  
  Scikit-learn: GridSearchCV, RandomizedSearchCV

## 2. E-commerce Product Recommender

Objective:  
Build a recommendation system to suggest products to users based on their previous interactions and preferences.

### Step 1: Problem Understanding

* Description:  
  Define the project’s business objective — improve product engagement and sales by suggesting relevant products to users.

### Step 2: Data Collection

* Description:  
  Collect product interaction data: product views, purchases, ratings, reviews, and customer IDs.
* Sources:

https://www.kaggle.com/datasets/PromptCloudHQ/flipkart-products/data

* Tools:  
  Kaggle, Scrapy, BeautifulSoup, Python requests

### Step 3: Data Preprocessing

* Description:  
  Clean the data by removing duplicates, handling missing values, encoding categorical features, and preparing user-item matrices.
* Tools:  
  Python: Pandas, NumPy

### Step 4: Exploratory Data Analysis (EDA)

* Description:  
  Understand product popularity, user activity distribution, and co-purchase patterns.
* Tools:  
  Python: Seaborn, Matplotlib

### Step 5: Feature Engineering

* Description:  
  Create new features such as total user interactions, product categories, and average product ratings per user.
* Tools:  
  Python: Pandas, NumPy

### Step 6: Model Selection & Training

* Description:  
  Choose an appropriate recommendation approach:
  + Collaborative Filtering (Matrix Factorization, SVD)
  + Content-Based Filtering (TF-IDF on product descriptions)
  + Hybrid Recommender (if combining both)
* Tools:  
  Python: Surprise, Scikit-learn, SciPy

### Step 8: Hyperparameter Tuning

* Description:  
  Optimize key model parameters such as number of latent factors and regularization terms.
* Tools:  
  Surprise GridSearchCV, Scikit-learn

## 3. Demand Forecasting for Retail

Objective:  
Predict future product demand based on historical sales data to optimize inventory, reduce stockouts, and improve planning.

### Step 1: Problem Understanding

* Description:  
  Define forecasting goals — daily/weekly/monthly product demand per store or category to improve inventory decisions.

### Step 2: Data Collection

* Description:  
  Gather historical sales data, product attributes, promotions, holidays, and store details.
* Sources:

https://www.kaggle.com/datasets/aslanahmedov/walmart-sales-forecast

* Tools:  
  Kaggle

### Step 3: Data Preprocessing

* Description:  
  Handle missing values, convert date columns, manage outliers, and resample time-series data as needed.
* Tools:  
  Python: Pandas, NumPy

### Step 4: Exploratory Data Analysis (EDA)

* Description:  
  Visualize sales trends, identify seasonality, detect anomalies, and analyze demand patterns.
* Tools:  
  Python: Matplotlib, Seaborn

### Step 5: Feature Engineering

* Description:  
  Create time-based features like day of week, month, year, promotions, holidays, and lag/rolling window features.
* Tools:  
  Python: Pandas, NumPy

### Step 6: Model Selection & Training

* Description:  
  Choose suitable forecasting models:
  + Classical Time-Series Models: ARIMA, SARIMA
  + Machine Learning Models: Random Forest, XGBoost, LightGBM
  + Specialized Tools: Facebook Prophet, LSTM (for deep learning)
* Tools:  
  Statsmodels, Scikit-learn, XGBoost, LightGBM, Prophet, TensorFlow/Keras (for LSTM)

### Step 7: Model Evaluation

* Description:  
  Evaluate model performance using metrics: RMSE, MAE, MAPE.
* Tools:  
  Scikit-learn: mean\_squared\_error, mean\_absolute\_error

### Step 8: Hyperparameter Tuning

* Description:  
  Optimize model parameters like order values for ARIMA or depth and learning rate for tree-based models.
* Tools:  
  Scikit-learn: GridSearchCV / RandomizedSearchCV  
  Hyperopt (optional)

### Step 9: Forecast Visualization

* Description:  
  Visualize actual vs. forecasted sales along with confidence intervals.
* Tools:  
  Python: Matplotlib, Seaborn