**Customer Churn Prediction - Project Documentation**

**Overview**

The **Customer Churn Prediction** project leverages machine learning models to predict whether a customer will churn (i.e., leave a service) based on features like customer demographics, usage patterns, and payment information. The project includes steps for data preprocessing, exploratory data analysis (EDA), model training, model evaluation, and deployment.

This project is organized into several Python modules, each focusing on a specific aspect of the customer churn prediction process. The overall flow is from data loading and preprocessing, through training and evaluation, to model deployment.

**Project Structure**

The project follows a modular structure, where each aspect of the code is contained in separate files for better maintainability and readability. Here’s an overview of the directory structure:

plaintext

Copy code

customer-churn-prediction/

│

├── src/ # Source code for the project

│ ├── \_\_init\_\_.py # Initializes the src package, imports key classes and functions

│ ├── data\_preprocessing.py # Data preprocessing logic

│ ├── exploratory\_data\_analysis.py # Exploratory Data Analysis (EDA) logic

│ ├── model\_training.py # Model training logic

│ ├── model\_evaluation.py # Model evaluation logic

│ ├── predictive\_system.py # Logic for deploying the model and making predictions

│ ├── utils.py # Helper functions for logging, plotting, and saving models

│

├── logs/ # Logs directory for storing logs

│ └── process.log # Log file for tracking process execution

│

├── models/ # Saved models and encoders

│ ├── customer\_churn\_model.pkl # The trained machine learning model

│ └── encoders.pkl # Label encoders for categorical features

│

├── requirements.txt # Python dependencies for the project

├── Dockerfile # Docker configuration for containerizing the project

└── README.txt # Project overview and setup instructions

**Module Descriptions**

**1. \_\_init\_\_.py (Inside src/ directory)**

This file initializes the src package and makes key classes and functions available for easy import. The purpose of the \_\_init\_\_.py file is to streamline the usage of the package and provide an organized entry point for key functionalities.

python

Copy code

# src/\_\_init\_\_.py

# Import key classes and functions for easier access when using the package

from .data\_preprocessing import DataPreprocessing

from .exploratory\_data\_analysis import EDA

from .model\_training import ModelTraining

from .model\_evaluation import ModelEvaluation

from .predictive\_system import PredictiveSystem

from .utils import plot\_histogram, plot\_boxplot, plot\_correlation\_heatmap, log\_model\_performance, save\_model

# Optional: Initialize package-level variables or configurations

import logging

# Configure logging for the entire package

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(message)s')

# Log when the package is initialized (optional)

logging.info("Package 'src' initialized successfully.")

**2. data\_preprocessing.py**

This module contains the code for preprocessing the raw data. It handles tasks such as loading the dataset, cleaning the data, handling missing values, and encoding categorical variables.

**Key Functions**:

* load\_and\_preprocess\_data(): Loads data, cleans it, and prepares it for modeling.

**3. exploratory\_data\_analysis.py**

This module focuses on performing exploratory data analysis (EDA). It includes visualizations and summary statistics to better understand the distribution and relationships of features in the dataset.

**Key Functions**:

* plot\_histogram(): Plots histograms with KDE for a given numerical column.
* plot\_boxplot(): Plots boxplots to visualize the spread and outliers of a numerical column.
* plot\_correlation\_heatmap(): Plots a correlation heatmap for numerical features.

**4. model\_training.py**

This module is responsible for training machine learning models. It includes the code for model selection, training with cross-validation, and hyperparameter tuning.

**Key Classes and Functions**:

* ModelTraining: Class that handles model selection, training, and saving the model.

**5. model\_evaluation.py**

Once the model is trained, this module evaluates its performance on unseen test data using metrics like accuracy, confusion matrix, and classification report.

**Key Functions**:

* evaluate\_model(): Evaluates the model and logs performance metrics such as accuracy, confusion matrix, and classification report.

**6. predictive\_system.py**

This module provides the functionality to load a saved model and make predictions on new customer data.

**Key Functions**:

* load\_and\_predict(): Loads the model and encoders, processes the input data, and makes a prediction.

**7. utils.py**

This module contains utility functions used throughout the project. It includes functions for plotting, logging, saving models, and handling exceptions.

**Key Functions**:

* plot\_histogram(), plot\_boxplot(), plot\_correlation\_heatmap(): Visualization functions.
* log\_model\_performance(): Logs the model’s performance metrics.
* save\_model(): Saves the trained model to a file.

**Installation Instructions**

**1. Clone the repository:**

bash

Copy code

git clone https://github.com/rathi/customer-churn-prediction.git

cd customer-churn-prediction

**2. Install the required dependencies:**

The dependencies required to run the project are listed in requirements.txt.

bash

Copy code

pip install -r requirements.txt

**3. (Optional) Set up Docker:**

If you want to run the project in a Docker container, use the following steps.

**Build the Docker image:**

bash

Copy code

docker build -t customer-churn-prediction .

**Run the Docker container:**

bash

Copy code

docker run -p 5000:5000 customer-churn-prediction

**How to Use**

**Training the Model**

1. Preprocess the data using the data\_preprocessing.py module.
2. Train the model using model\_training.py (default models include Decision Tree, Random Forest, and XGBoost).
3. Evaluate the model using model\_evaluation.py.
4. Save the trained model and encoders for future use.

python

Copy code

from src import ModelTraining

model\_trainer = ModelTraining(X\_train, y\_train)

model\_trainer.train\_and\_evaluate()

**Making Predictions**

Once the model is trained and saved, you can use the predictive\_system.py module to load the model and make predictions on new customer data.

python

Copy code

from src import PredictiveSystem

input\_data = {

'gender': 'Female',

'SeniorCitizen': 0,

'Partner': 'Yes',

'Dependents': 'No',

'tenure': 1,

'PhoneService': 'No',

'MultipleLines': 'No phone service',

'InternetService': 'DSL',

'OnlineSecurity': 'No',

'OnlineBackup': 'Yes',

'DeviceProtection': 'No',

'TechSupport': 'No',

'StreamingTV': 'No',

'StreamingMovies': 'No',

'Contract': 'Month-to-month',

'PaperlessBilling': 'Yes',

'PaymentMethod': 'Electronic check',

'MonthlyCharges': 29.85,

'TotalCharges': 29.85

}

predictive\_system = PredictiveSystem(model\_path='models/customer\_churn\_model.pkl')

prediction = predictive\_system.predict(input\_data)

print(prediction)

**Logging**

Logs of the project execution are stored in the logs/ directory, specifically in process.log. The logs will contain information about the data processing steps, model training, and evaluation results.

**Log Example:**

plaintext

Copy code

2024-12-30 12:00:00 - Package 'src' initialized successfully.

2024-12-30 12:01:00 - Histogram of tenure plotted successfully.

**Requirements**

Here are the Python dependencies required for this project, listed in the requirements.txt file:

plaintext

Copy code

pandas

numpy

scikit-learn

xgboost

matplotlib

seaborn

imbalanced-learn

pickle5

**Conclusion**

This project serves as a comprehensive framework for predicting customer churn. By following this documentation, you can preprocess data, train machine learning models, evaluate their performance, and deploy the model for real-time predictions.

If you encounter any issues, please refer to the logs for more details or contact the project maintainers.