PROJECT REPORT

1. PROBLEM STATEMENT

EliteBank our partner aims to leverage machine learning to predict long-term investor prospects while integrating robust MLOps practices. This plan emphasizes automated model tracking with MLflow, version control continuous integration using GitHub, and an interactive deployment interface via Streamlit. The objective is to streamline model development, ensure reproducibility, and deliver an engaging user interface for business stakeholders.

2. Environment Setup

Created a dedicated conda environment for the project using the following command:

```
conda create -p venv python==3.10 -y
```

Activate

Conda activate venv/

3. FILE AND FOLDER PARTITION

```
Bank_LongTerm_Investment_prediction/
├── artifacts/
        ├──Bank_LongTerm_Investment.csv
        ├──model.pkl
        ├──shap_summary.png
        ├──test_data.csv
        ├──train_data.csv
├──Logs
        ├──bank_LongTerm_investment.logs
├──Notebook
        ├──bank_longterm_investment.ipynb
├──src
        ├──components
                ├──data_ingestion.py
                ├──data_transformation.py
                ├──model_training.py
        ├──explain
                ├──shap_explain.py
        ├──pipeline
                ├──pipeline.py
        ├──utils
```

```
            ├──config.py
        ├──exception.py
        ├──logging.py
├──app.py
├──requirements.txt
```

- **`Artifacts/`**

Purpose: This folder is used to store all the intermediate and final outputs of the pipeline.
Contents:
Raw Data Files: e.g., CSV files produced from the ingestion step.
Transformed Data: Cleaned and feature-engineered datasets.
Train/Test Splits: Data used for model training and evaluation.
Model Artifacts: The final best model saved as a pickle file (`model.pkl`).

- **Logs Folder**

`Logs/bank_longterm_investment.logs`
Purpose: This file holds the logs generated during pipeline execution.
Contents:
Detailed runtime logs (INFO, WARNING, ERROR) which help with debugging, performance tracking, and audit trails.

- **Notebook Folder**

`Notebook/bank_longterm_investment.ipynb`
Purpose: A Jupyter Notebook designed for exploratory data analysis (EDA) and prototyping.
Contents:
Interactive cells for data visualization

Source Code (src) Folder (src/)

This folder contains all the Python code modules, divided into several subfolders for modularity and maintainability.

Components

- **`src/components/data_ingestion.py`**

Purpose: Handles data ingestion tasks such as reading ARFF files and converting them to CSV format.
Key Functionality: Loads raw data from the source and saves a standardized CSV file in the `artifacts` folder.

- **`src/components/data_transformation.py`**

Purpose: Responsible for data cleaning, feature engineering, and preparing data for modeling.
Key Functionality: Performs operations like handling missing values, encoding categorical features, and saving the transformed data.

`src/components/model_training.py`
Purpose: Trains multiple candidate models, logs metrics to MLflow, and selects the best performing model.
Key Functionality: Uses scikit-learn models (e.g., Logistic Regression, Random Forest, SVM, Decision Tree), evaluates them using metrics such as accuracy, F1, and ROC-AUC, and saves the best model as `model.pkl`.

Pipeline
`src/pipeline/pipeline.py`
Purpose: Orchestrates the entire workflow by sequentially calling data ingestion, data transformation, and model training components.
Key Functionality: Acts as the main entry point to run the complete ML pipeline end-to-end.

Explain
`src/explain/shap-explain.py`
Purpose: Provides utilities for model explainability using SHAP (SHapley Additive exPlanations).
Key Functionality: Generates visual explanations for model predictions to help stakeholders understand which features drive the model's decisions.

Utils
`src/utils/config.py`
Purpose: Contains configuration dataclasses that define file paths, MLflow credentials, and other settings used throughout the project.
Key Classes:

`DataIngestionConfig`: Configurations for data ingestion.

`DataTransformationConfig`: Configurations for data transformation.

`ModelTrainingConfig`: Paths for train/test data and model saving, plus MLflow credentials.

logging.py (in src/):
A module that sets up the project's logging configuration. It defines how log messages are formatted and where they are stored (e.g., writing to a file or stdout).

exception.py (in src/):
Contains a custom exception class (`CustomException`) that wraps lower-level exceptions with additional context. This enhances error tracking and debugging.

`app.py`
Purpose: A Streamlit application for deploying the trained model in a user-friendly web interface.
Key Functionality: Accepts user input, applies the model to predict if a customer will subscribe to a deposit, and displays results interactively.

`requirements.txt`
Purpose: Lists all Python package dependencies required to run the project.
Key Dependencies: Packages such as `pandas`, `numpy`, `scikit-learn`, `mlflow`, `streamlit`, `joblib`, and others.

4. CI/CD and Deployment

CI/CD Pipeline: The project is set up for CI/CD using GitHub Actions.
The workflow file (`.github/workflows/ci.yml`) runs linting, unit tests, and the complete pipeline to ensure code quality and reproducibility.
MLflow & Deployment:
MLflow is integrated to track model training runs, log metrics, and register the best model.
The Streamlit app (`app.py`) serves as the front-end for real-time predictions.

5. Model Training

During the model training phase, four candidate models were trained and evaluated using a common test dataset. The models considered were:

- **Logistic Regression**
- **Random Forest**
- **SVM (RBF)**
- **Decision Tree**

Evaluation Metrics:
Each model was evaluated using the following metrics:

- **Accuracy**
- **F1-Score**
- **ROC-AUC**
- **Confusion Matrix**
- **Classification Report**

Results:
Logistic Regression:
Accuracy: 82.98%
F1-Score: 81.61%
ROC-AUC: 0.9064
Classification Report:
For class 0: Precision: 0.82, Recall: 0.86, F1-Score: 0.84

For class 1: Precision: 0.84, Recall: 0.80, F1-Score: 0.82

Random Forest:
Accuracy: 86.03%
F1-Score: 85.78%
ROC-AUC: 0.9195
Classification Report:

For class 0: Precision: 0.89, Recall: 0.83, F1-Score: 0.86
For class 1: Precision: 0.83, Recall: 0.89, F1-Score: 0.86

SVM (RBF):
Accuracy: 84.19%
F1-Score: 83.04%
Classification Report:

For class 0: Precision: 0.84, Recall: 0.86, F1-Score: 0.85
For class 1: Precision: 0.84, Recall: 0.82, F1-Score: 0.83

Decision Tree:
Accuracy: 79.44%
F1-Score: 78.05%
ROC-AUC: 0.7933
Classification Report:

For class 0: Precision: 0.80, Recall: 0.82, F1-Score: 0.81
For class 1: Precision: 0.79, Recall: 0.77, F1-Score: 0.78

Best Model Selection:

Based on the ROC-AUC metric, the **Random Forest** model achieved the highest score of **0.9195**. This indicates that the Random Forest model excellently distinguished bank clients who are likely to make long term investment.
Additionally, the Random Forest model demonstrated high overall accuracy (86.03%) and balanced precision/recall values across both classes.

Conclusion:

The Random Forest classifier is selected as the final model for deployment due to its superior ROC-AUC score of 0.9195, along with robust accuracy and balanced performance on both classes. This model has been logged to MLflow for experiment tracking and has been saved as `artifacts/model.pkl` for further deployment and integration into the Streamlit application.