Структура проекта

- `data_preprocessing.py` — скрипт для очистки и предобработки данных.
- `feature_engineering.py` — скрипт для создания дополнительных признаков (feature engineering) и анализа корреляций.
- `model_training.py` — скрипт для обучения нескольких моделей и выбора лучшей на основе метрики.
- `model_evaluation.py` — скрипт для оценки качества модели на тестовых данных.
- `model_inference.py` — API на Flask, позволяющий интегрировать модель с внешними системами.

Скрипты и этапы выполнения задания

1. `data_preprocessing.py`

Описание

Этот скрипт загружает данные, выполняет предобработку, очищает их от пропусков, дубликатов и аномальных значений. Признаки кодируются с помощью one-hot encoding, а числовые данные масштабируются. После подготовки данные делятся на тренировочную и тестовую выборки.

Методы и библиотеки

- **Pandas** — для работы с данными (чтение, запись и манипуляции с DataFrame).
- **scikit-learn (StandardScaler)** — для масштабирования числовых признаков и разделения на тренировочные и тестовые наборы данных.

Скрипт сохранит подготовленные данные в `data/X_train.csv`, `data/X_test.csv`, `data/y_train.csv`, `data/y_test.csv`.

2. `feature_engineering.py`

Описание

В этом скрипте создаются дополнительные признаки для улучшения качества модели. Также строится матрица корреляции, которая включает целевую переменную для лучшего понимания взаимосвязей между признаками.

Обоснование и интерпретация новых признаков

1. **`balancePerProduct`**:
    - Вычисляется как отношение баланса к количеству продуктов у клиента. Это может быть полезно, так как клиенты с низким балансом на каждый продукт могут быть менее заинтересованы в услугах банка, что может указывать на повышенный риск оттока.
2. **`tenureToAge`**:
    - Показывает отношение стажа клиента в банке к его возрасту. Это значение может быть полезно, так как более молодые клиенты с высоким стажем

(относительно их возраста) могут иметь низкую вероятность оттока, будучи более лояльными.

3. `creditAge`:

    - Условный показатель кредитного возраста (кредитный рейтинг, делённый на возраст). Высокий кредитный возраст может указывать на более надёжного клиента, который с меньшей вероятностью уйдёт из банка.

Методы и библиотеки

- **Pandas** — для работы с данными и создания новых признаков.
- **Seaborn, Matplotlib** — для построения матрицы корреляции и визуализации взаимосвязей между признаками.

Скрипт выполнит анализ, отобразит матрицу корреляций и сохранит новые данные в `data/X_train_fe.csv`, `data/X_test_fe.csv`, `data/y_train.csv`, `data/y_test.csv` для последующего использования при обучении модели.

3. `model_training.py`

Описание

Скрипт обучает несколько моделей машинного обучения (RandomForest, LogisticRegression и LightGBM), используя поиск по сетке гиперпараметров (RandomizedSearchCV). Лучшая модель выбирается на основе метрики ROC-AUC и сохраняется для дальнейшего использования.

Обоснование выбора методов

- **RandomForest и LogisticRegression** — классические модели, часто применяемые для задач классификации, обеспечивают интерпретируемость и стабильность.
- **LightGBM** — бустинговая модель, которая эффективно работает с большими объемами данных и имеет встроенные функции для обработки категориальных признаков.

Методы и библиотеки

- **scikit-learn (RandomForest, LogisticRegression, RandomizedSearchCV)** — для обучения и подбора гиперпараметров.
- **lightgbm** — для реализации градиентного бустинга.

Скрипт сохранит наилучшую модель в папке **models** (например, LightGBM_churn_model.pkl)

4. `model_evaluation.py`

Описание

Этот скрипт загружает лучшую модель, сохраненную в models, оценивает её на тестовом наборе данных и выводит ключевые метрики: **accuracy, ROC-AUC**, а также визуализирует матрицу ошибок и ROC-кривую.

Методы и библиотеки

- **scikit-learn (accuracy_score, roc_auc_score, classification_report)** — для вычисления метрик модели.
- **Seaborn, Matplotlib** — для визуализации матрицы ошибок и ROC-кривой.

Скрипт выведет метрики и покажет графики для анализа качества модели.

5. `model_inference.py`

Описание

Этот скрипт реализует REST API на Flask, позволяя интегрировать модель с внешними системами. API принимает JSON-данные клиента, предсказывает вероятность оттока и возвращает её в JSON-ответе.

Методы и библиотеки

- **Flask** — для создания REST API и работы с HTTP-запросами.
- **joblib** — для загрузки сохраненной модели.