Week 8&9 Challenge Documentation

Project Overview

This project focuses on developing a **fraud detection system** using machine learning, deploying the model as a Flask API, and building an interactive dashboard for fraud insights. The implementation is divided into five tasks:

1. **Task 1: Data Collection and Preprocessing**
2. **Task 2: Feature Engineering and Model Training**
3. **Task 3: Model Evaluation and Optimization**
4. **Task 4: Model Deployment and API Development**
5. **Task 5: Building a Fraud Detection Dashboard**

My Contributions

Task 1: Data Collection and Preprocessing

1. **Collected and Preprocessed Fraud Detection Data**
   - Imported and analyzed raw transaction data.
   - Handled missing values, outliers, and inconsistent records.
   - Converted categorical data into numerical format (One-Hot Encoding).
   - Normalized numerical features for better model performance.
2. **Saved Cleaned Data for Further Processing**
   - Stored the cleaned dataset in a structured format (CSV/Parquet).

Task 2: Feature Engineering and Model Training

1. **Engineered Features to Enhance Model Performance**
   - Created new features such as transaction frequency and location-based patterns.
   - Applied feature selection techniques to reduce dimensionality.

2. **Split Data into Training and Testing Sets**
   - Used `train_test_split()` from `sklearn` to create an 80/20 split.
3. **Trained a Machine Learning Model for Fraud Detection**
   - Implemented multiple models (`Random Forest`, `Logistic Regression`, `XGBoost`).
   - Tuned hyperparameters to optimize performance.
4. **Saved the Best Performing Model for Deployment**
   - Used `joblib` to store the trained model.

Task 3: Model Evaluation and Optimization

1. **Evaluated Model Performance**
   - Used accuracy, precision, recall, and F1-score to measure effectiveness.
   - Visualized confusion matrices and ROC curves.
2. **Optimized the Model for Better Predictions**
   - Applied hyperparameter tuning (`GridSearchCV`, `RandomizedSearchCV`).
   - Performed cross-validation to ensure generalization.
3. **Finalized the Model for Deployment**
   - Selected the best model based on evaluation metrics.
   - Stored the optimized model for serving predictions.

Task 4: Model Deployment and API Development

1. **Developed a Flask API to Serve Fraud Predictions (`serve_model.py`)**
   - Created RESTful endpoints for fraud detection.
   - Loaded the trained model using `joblib` for inference.
2. **Implemented Logging for Continuous Monitoring**
   - Configured `Flask-Logging` to track incoming requests and errors.
3. **Dockerized the API for Deployment**

    - Created a `Dockerfile` to containerize the application.
    - Installed dependencies via `requirements.txt`.
    - Built and deployed the container using Docker.
4. **Tested the API Locally and in a Docker Container**

    - Used `Postman` and `cURL` for API testing.
    - Verified prediction outputs and response times.

Task 5: Building a Fraud Detection Dashboard

1. **Developed a Flask API Endpoint to Serve Fraud Data**

    - Created `/fraud-summary` endpoint to return fraud statistics from a CSV dataset.
    - Processed data using `pandas` to generate insights.
2. **Built an Interactive Dashboard Using Dash (`dashboard.py`)**

    - Designed the layout using `Dash` and `HTML` components.
    - Implemented visualizations using `Plotly`:
        - **Fraud trends over time (line chart)**
        - **Fraud cases by location (bar chart)**
        - **Fraud cases by device/browser (comparison charts)**
3. **Dockerized the Dashboard for Deployment**

    - Created a `Dockerfile-dashboard` to containerize the Dash application.
    - Built and ran the containerized dashboard using Docker.

Project Structure

```
fraud-detection/
|── data/                  # Raw and processed datasets
|── models/                  # Saved ML models
|── api/                  # Flask API files
│   ├── serve_model.py       # Flask API for fraud detection
│   ├── requirements.txt     # API dependencies
│   ├── Dockerfile          # Docker configuration for API
|── dashboard/             # Fraud detection dashboard
│   ├── dashboard.py         # Dash visualization
│   ├── Dockerfile-dashboard  # Docker config for dashboard
```

```
│── notebooks/           # Jupyter notebooks for experimentation
│── logs/              # Log files for monitoring API requests
│── README.md            # Project documentation
```

Conclusion

This project successfully developed a **fraud detection system** from data preprocessing to deployment, allowing users to detect fraudulent transactions and visualize fraud trends. The integration of Flask and Dash provides an interactive and scalable solution.