**📑 Fraud Detection Project – Progress Report**

**✅ Work Completed**

**1. Data Ingestion & Transformation**

* Implemented ingestion from both **MongoDB** and **CSV** sources.
* Designed DataTransformation module to handle:
  + Missing values, scaling, encoding.
  + Splitting into train/test sets.
* Stored processed data as artifacts for downstream steps.

**2. Model Training (model\_trainer.py)**

* Implemented multiple classifiers (Random Forest, Gradient Boosting, Decision Tree, XGBoost).
* Handled **class imbalance** using **SMOTE** and **SMOTE+Tomek** resampling techniques.
* Evaluation metrics:
  + Primary → **Recall & F2 Score** (to capture fraud cases).
  + Secondary → Precision, F1, ROC-AUC.
* Logged experiments with MLflow for comparison.
* Best performer so far: **Random Forest with SMOTE+Tomek**.

**3. Prediction Pipeline (prediction\_pipeline.py)**

* Designed pipeline to load trained model & preprocessor from artifacts.
* Can take **sample CSV input** → transform → predict → save results.
* Outputs both:
  + predictions (0 = Not Fraud, 1 = Fraud).
  + probabilities (Fraud/Not Fraud likelihoods).
* artifacts.py supports flexible storage (works with both single & multi-class probability outputs).

**4. Flask Integration (app.py)**

* Built Flask app to:
  + Load model & preprocessor at startup.
  + Accept JSON input from user (via /predict endpoint).
  + Return fraud likelihood as JSON response.
* Running successfully on port 8080.

**5. Bootstrap UI Plan**

* Input fields designed for **fraud-related features** (e.g., transaction amount, time, category, etc.).
* Currently: Flask app expects **sample input from artifacts**.
* Next Step: Replace with **form-based input** in index.html.

**📌 Next Tasks**

1. **Frontend Integration (Flask + Bootstrap)**
   * Create index.html form with input fields for all fraud features.
   * Capture user input → send to /predict endpoint.
   * Display output probabilities clearly:
     + ✅ "Fraud Probability"
     + ✅ "Not Fraud Probability"
2. **Data Logging of User Inputs**
   * Store submitted form data in **CSV file** for tracking & retraining.
   * Append predictions along with input features.
3. **Model Prediction Module Finalization (model\_prediction.py)**
   * Not fully completed yet.
   * Will handle both **batch predictions** (CSV input) and **real-time predictions** (API/UI).
4. **Testing & Validation**
   * Unit tests for prediction\_pipeline.py.
   * Manual tests for Flask UI (submit form, verify results).
   * Edge cases: missing inputs, invalid values.

**🎯 Approach Going Forward**

* **Frontend (UI)**:
  + Build a Bootstrap form with fraud-related input fields.
  + Use AJAX / form submit → send request to Flask backend.
* **Backend (API)**:
  + Use prediction\_pipeline.py for preprocessing + model inference.
  + Return prediction JSON with both probabilities.
* **Output to User**:
  + Show **"Fraud Probability: xx"** and **"Not Fraud Probability: yy"** in a user-friendly card layout.
  + Optionally highlight results if fraud probability > threshold.

✅ In summary:

* **Data ingestion, transformation, training, evaluation, and prediction pipeline are working.**
* **Flask backend is ready**, but **UI form + user input storage are pending**.
* **Next milestone**: Fully integrate UI (Bootstrap form) → backend pipeline → CSV logging → user-friendly fraud prediction output.

**📌 Input Fields & Expected Outputs**

| **Field Name** | **Type** | **Description** | **Example Input** |
| --- | --- | --- | --- |
| TransactionAmount | Number | Total transaction amount | 1500.50 |
| AnomalyScore | Number | Model-generated anomaly score | 0.87 |
| Amount | Number | Normalized or derived amount field | 1200.00 |
| Age | Number | Customer age | 35 |
| AccountBalance | Number | Current balance of account | 52000.75 |
| SuspiciousFlag | Dropdown | Indicates suspicious activity (0 = No, 1 = Yes) | 1 |
| Hour | Number | Hour of transaction (0–23) | 14 |
| Gap | Number | Days since last login | 5 |
| Category | Dropdown | Transaction category | Online |

**📌 Expected Output**

When the user clicks **Predict Fraud**:

* **Prediction**:
  + "Fraud" if fraud probability > threshold.
  + "Not Fraud" otherwise.
* **Probability** (displayed clearly in results section):
  + **Fraud Probability**: e.g., 72.3%
  + **Not Fraud Probability**: e.g., 27.7%
* **Alerts & Validation**:
  + If any field is empty → alert user "Field <name> cannot be empty".
  + Prevents submission until all fields are filled.