National Institute of Technology,
Delhi

Medicare Fraud Detection

Using Artificial Intelligence

END SEMESTER PROJECT [CSPB 200]

Bachelor of Technology in Computer Science and Engineering

Submitted By :

Devesh Sarda (231210038)

Kapil Meena (231210056)

Madhav Raj (231210064)

What are Healthcare Fraud?

Healthcare fraud is an organized crime that involves collaboration among various parties such as providers (including hospitals, cashiers, medical labs, nurses, lab assistants, and others), physicians, and beneficiaries.

- Selling drugs, devices, foods, or medical cosmetics that have not been proven effective.
- Billing of services or medical procedures which are not performed.
- Convincing people to provide their health insurance identification number and other personal information to bill for non-rendered services, steal their identity, or enroll them in a fake benefit plan.

A healthcare provider is an authorized person or entity that provides medical care or treatment. Providers include doctors, nurse practitioners, radiologists, labs, hospitals, urgent care clinics, medical supply companies, and other professionals, facilities, and businesses that provide such services.

01 Problem Statement

02 Objective

03 Dataset Overview

04 Tech Stack

05 Architecture Diagrams

06 Feature Engineering

07 Model Training

08 StreamLit App

09 Input Prep Script

10 Testing and Validation

11 Conclusion and Future Work

Overview

of this presentation

Problem Statement ?

Insurance companies are the most vulnerable institutions impacted due to these bad practices. The insurance premium is also increasing day by day due to this bad practice.

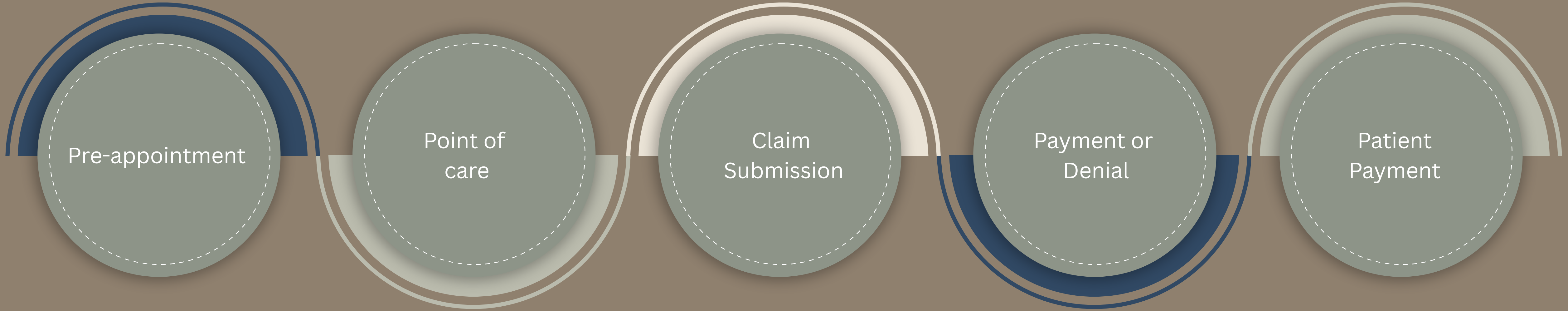
- Various statistics tells us that more than 15% of the total Medicare expense is caused due to fraud claims.

Impacts of the problem:

It is a criminal act as patients were either given false medicines or procedures which were not required.

It increases the overall expenditure on Healthcare and it returns as a burden on the insured user because private payers increase their premiums.

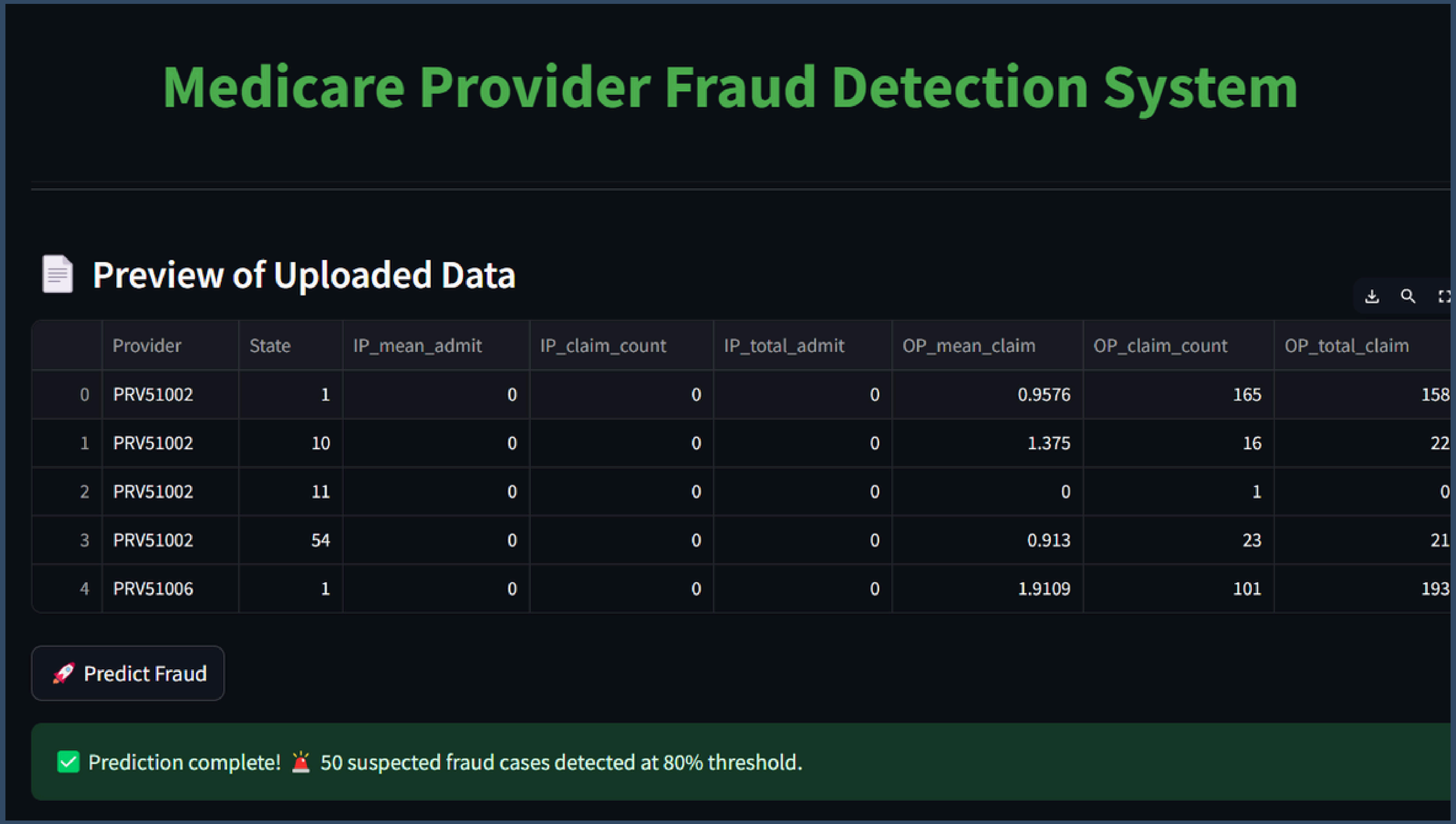
The false money earned by doing such frauds has also been used for carrying out various illegal activities that can be potentially harmful either to the nation or the entire world.



>>> **Claim Process**

Objectives

OUR OBJECTIVE IS TO PREDICT WHETHER A PROVIDER IS POTENTIALLY FRAUDULENT OR THE PROBABILITY SCORE OF THAT PROVIDER'S FRAUDULENT ACTIVITY AND ALSO FIND THE REASONS BEHIND IT AS WELL TO PREVENT FINANCIAL LOSS.



Medicare Provider Fraud Detection System

Preview of Uploaded Data

	Provider	State	IP_mean_admit	IP_claim_count	IP_total_admit	OP_mean_claim	OP_claim_count	OP_total_claim
0	PRV51002	1	0	0	0	0.9576	165	158
1	PRV51002	10	0	0	0	1.375	16	22
2	PRV51002	11	0	0	0	0	1	0
3	PRV51002	54	0	0	0	0.913	23	21
4	PRV51006	1	0	0	0	1.9109	101	193

✅ Prediction complete! 🚨 50 suspected fraud cases detected at 80% threshold.

- Build a supervised ML model (XGBoost) trained on labeled Medicare claims data
- Automate and speed up fraud detection for investigators and auditors
- Visualize fraud scores and highlight high-risk providers

WE CAN ALSO

Find out the most important features which are the reasons behind the potentially fraudulent providers. For example, if the claim amount is very high for a patient whose risk score is low, then it may be a fraud.

Depending on the probability score and fraudulent reasons insurance company can accept or deny the claim or set up an investigation on that provider.

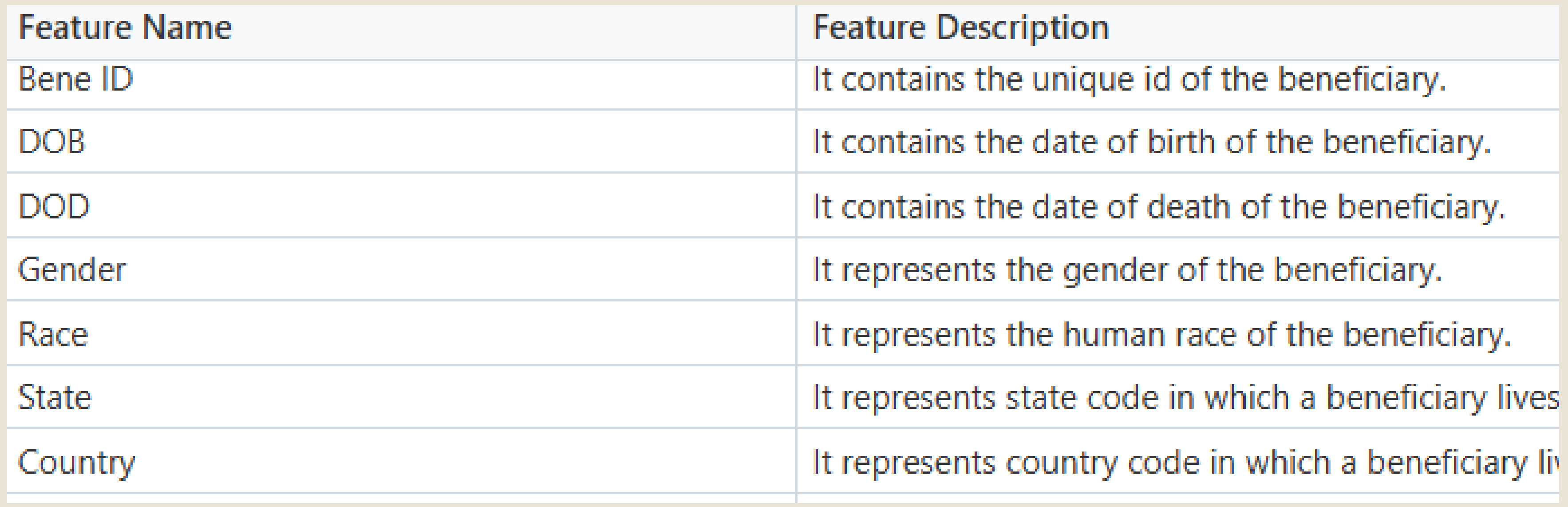
IT is not only about financial loss being a significant concern but also about protecting the healthcare system so that it can provide quality and safe care to legitimate patients.

DATASET

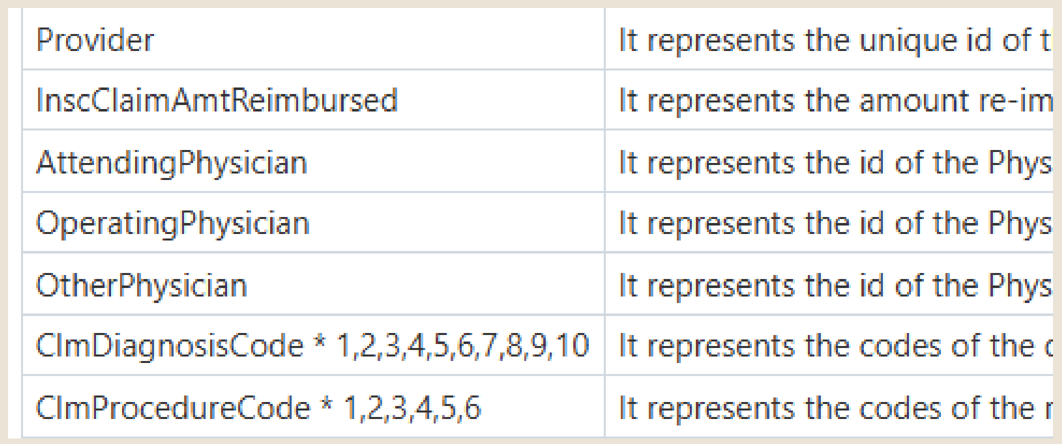
Provided by- kaggle

How do these different levels of information get joined?

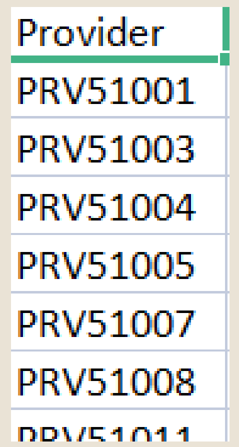
Inpatient and outpatient data are joined based on common columns, then the resultant matrix is joined with beneficiaries' details based on Bene ID. Finally, it gets the provider's potentially Fruadulent information using Provider ID.



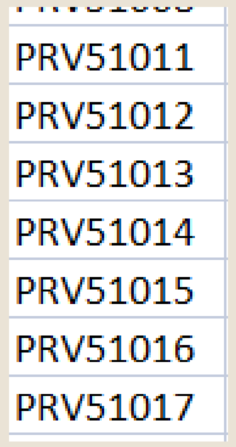
Feature Name	Feature Description
Bene ID	It contains the unique id of the beneficiary.
DOB	It contains the date of birth of the beneficiary.
DOD	It contains the date of death of the beneficiary.
Gender	It represents the gender of the beneficiary.
Race	It represents the human race of the beneficiary.
State	It represents state code in which a beneficiary lives
Country	It represents country code in which a beneficiary li



Provider	It represents the unique id of t
InscClaimAmtReimbursed	It represents the amount re-im
AttendingPhysician	It represents the id of the Phys
OperatingPhysician	It represents the id of the Phys
OtherPhysician	It represents the id of the Phys
ClmDiagnosisCode * 1,2,3,4,5,6,7,8,9,10	It represents the codes of the c
ClmProcedureCode * 1,2,3,4,5,6	It represents the codes of the r



Provider
PRV51001
PRV51003
PRV51004
PRV51005
PRV51007
PRV51008
PRV51011



PRV51011
PRV51012
PRV51013
PRV51014
PRV51015
PRV51016
PRV51017

OUR DASTASET HAS FOUR LEVELS:

Beneficiaries Information

This level of data contains beneficiaries KYC details like: DOB, DOD, Gender, Race, health conditions [Chronic disease if any], State, Country they belong to, etc.

In-Patient Details

This level of data consists of the claim details of the patients who were admitted to the hospital. Thus, it comprises the 3 extra columns: Admission date, Discharge date & Diagnosis group code.

Out-Patient Details

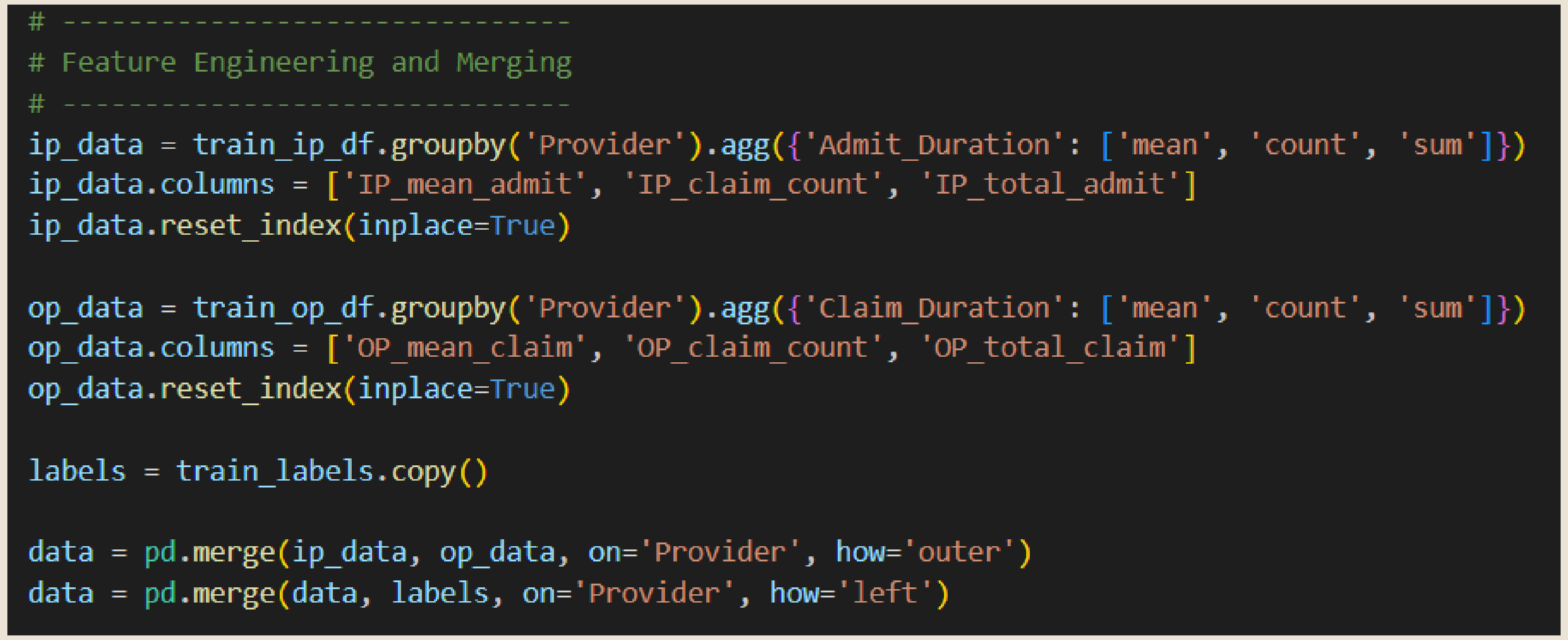
This level of data consists of the claim details for the patients who were not admitted & who only visited the hospital.

Potentially fraudulent & non-fraudulent Providers IDs

It consists of the Medicare provider ids and their corresponding label of being fraudulent or not.

Feature Engineering

Feature engineering is the process of transforming raw Medicare data into meaningful numerical features that the machine learning model can learn from.



```
# ------------------------------
# Feature Engineering and Merging
# ------------------------------
ip_data = train_ip_df.groupby('Provider').agg({'Admit_Duration': ['mean', 'count', 'sum']})
ip_data.columns = ['IP_mean_admit', 'IP_claim_count', 'IP_total_admit']
ip_data.reset_index(inplace=True)

op_data = train_op_df.groupby('Provider').agg({'Claim_Duration': ['mean', 'count', 'sum']})
op_data.columns = ['OP_mean_claim', 'OP_claim_count', 'OP_total_claim']
op_data.reset_index(inplace=True)

labels = train_labels.copy()

data = pd.merge(ip_data, op_data, on='Provider', how='outer')
data = pd.merge(data, labels, on='Provider', how='left')
```

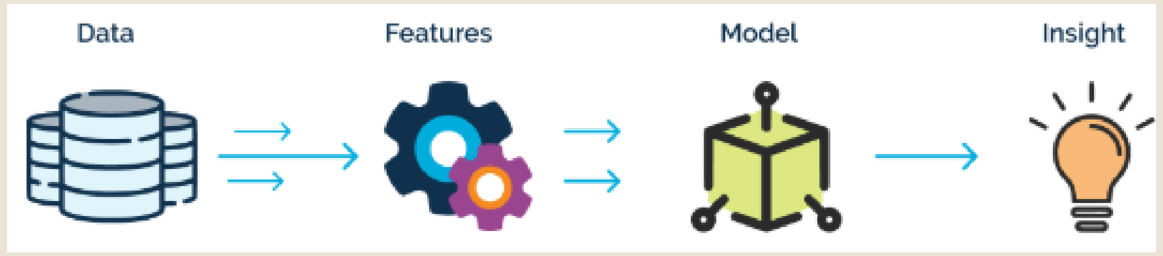
In our Medicare Fraud Detection system,

raw data from multiple sources (inpatient, outpatient, and provider labels) is transformed into structured features for model training. This is achieved through feature engineering and merging using pandas operations.

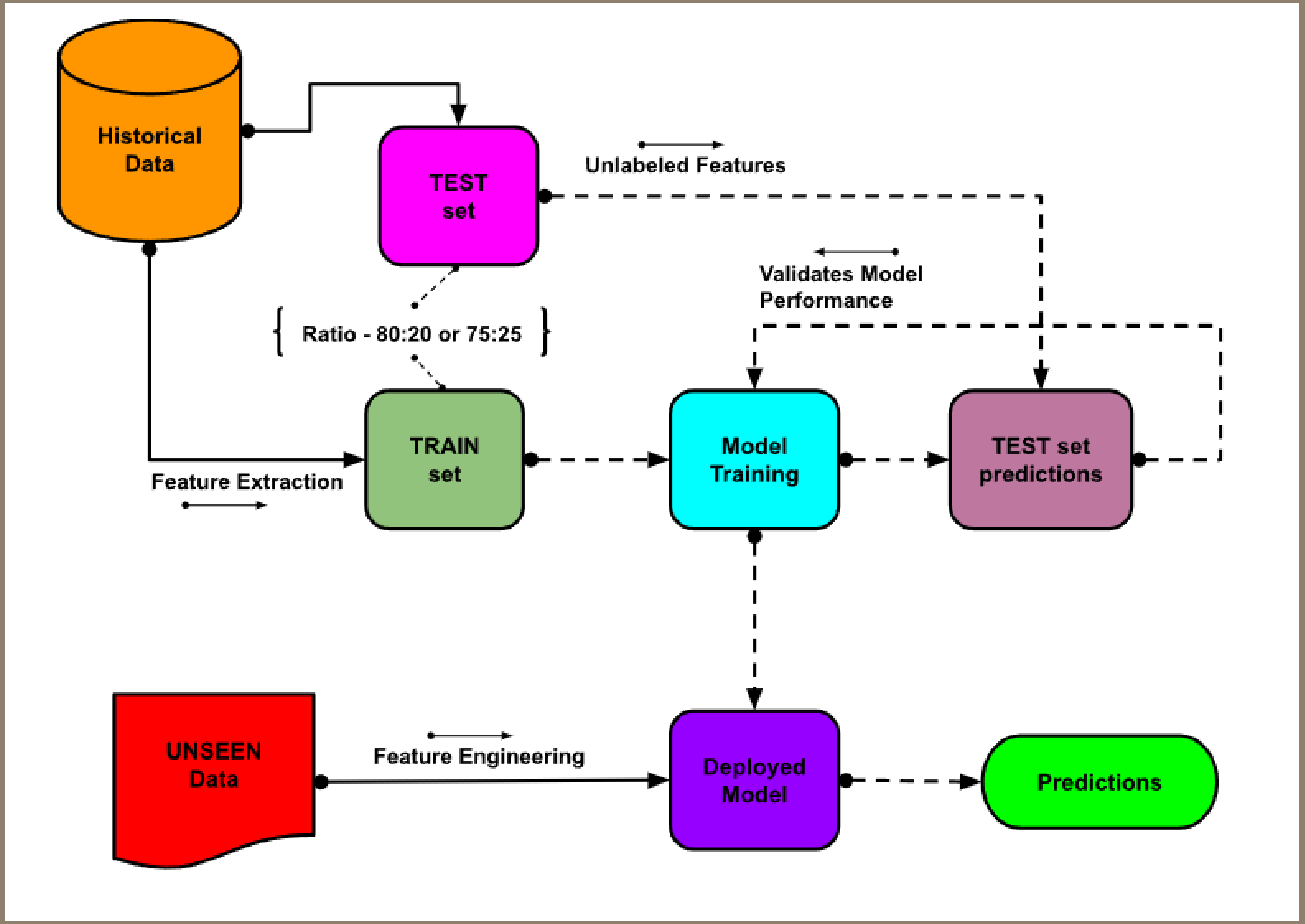
Key Steps We follow are :

- Group inpatient claims (train_ip_df) by Provider and aggregate:
  - 'Admit_Duration' → mean, count, and sum → becomes:
    - IP_mean_admit
    - IP_claim_count
    - IP_total_admit
- Similarly, group outpatient claims (train_op_df) by Provider:
  - 'Claim_Duration' → mean, count, and sum → becomes:
    - OP_mean_claim
    - OP_claim_count
    - OP_total_claim

- Merge all derived inpatient and outpatient features using Provider as key (outer join).
- Finally, merge with the training labels (fraud/not fraud) to create the complete training dataset.



This step reduces multiple rows per claim to a single row per provider — a crucial transformation that allows the XGBoost model to work efficiently and focus on provider-level patterns of fraudulent behavior.



Unseen Data → Feature Engineering → Deployed Model

- We compare the predictions to the actual labels to evaluate the model.
- Each provider gets a fraud probability (e.g., 0.87 → likely fraud).

Historical Data

This data is used to train and test the model. We split the data in two parts i.e. train dataset and test dataset in a 80:20 ratio. This ensures that we don’t overfit data.

Feature Extraction

We convert the raw data, where there were many rows into one row per provider with feature such as claim count, average duration, reimbursement totals, number of chronic conditions wtc.

Model Training

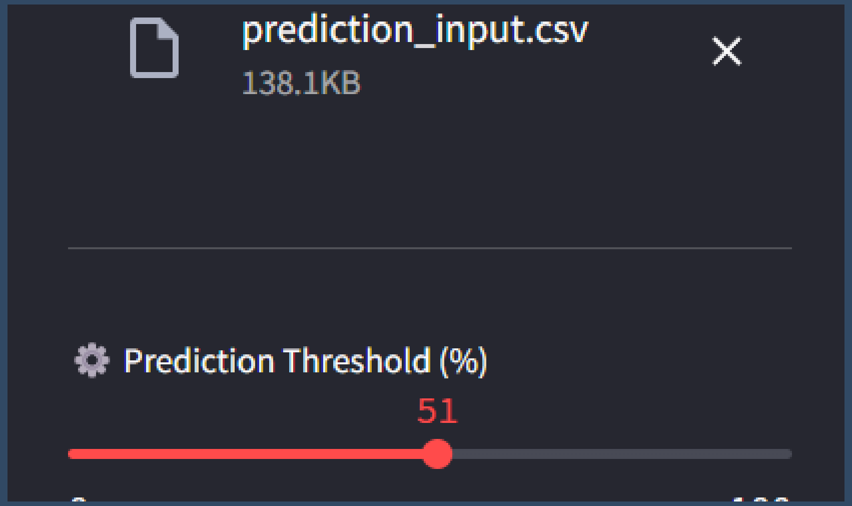
There is where XGBoost comes in, the model learns from patterns from the train dataset. It builds trees to classify whether a provider is likely fraudulent or not. The model is then used to make predictions on the test data.

Model training

Data → Preprocessing → Model Building → Testing → Deployment → Real-World Predictions

Streamlit app

The front end of our fraud detection system is a fully functional Streamlit web app, implemented in Python. The app allows users to interact with the trained XGBoost model without needing to know machine learning.

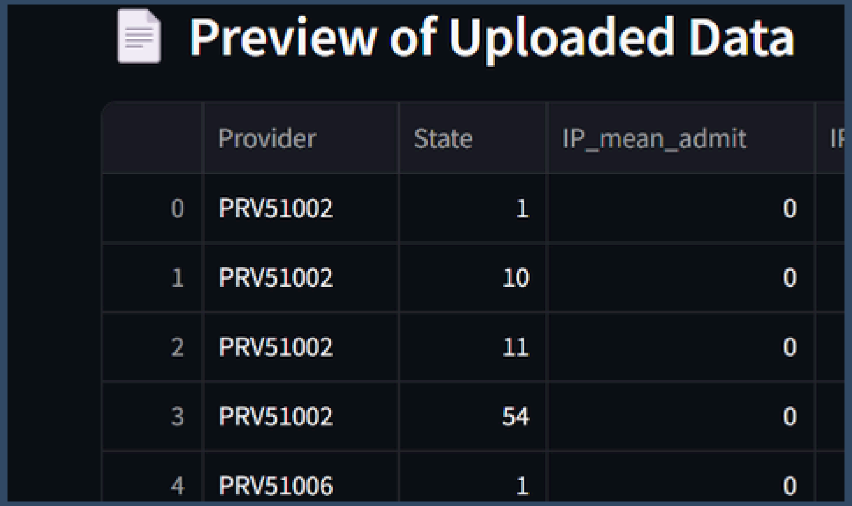


prediction_input.csv
138.1KB

Prediction Threshold (%)
51

Upload & Threshold

The app uses st.sidebar.file_uploader to accept a prediction-ready CSV file. Users can also set a prediction threshold using st.sidebar.slider. This value directly controls the classification logic when interpreting the XGBoost output probabilities.

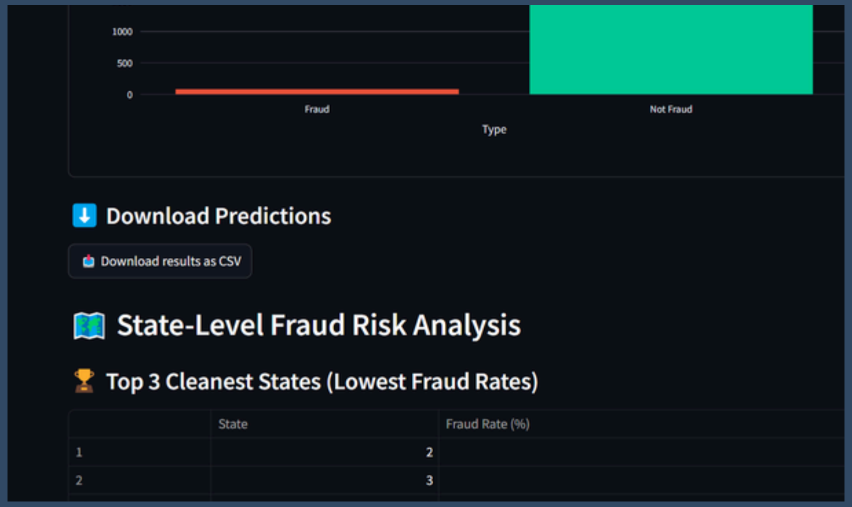


Preview of Uploaded Data

	Provider	State	IP_mean_admit
0	PRV51002	1	0
1	PRV51002	10	0
2	PRV51002	11	0
3	PRV51002	54	0
4	PRV51006	1	0

Prediction Execution

Once a file is uploaded, clicking “Predict Fraud” triggers model.predict_proba() on the processed input. The app assigns a 'Score' and a binary 'Prediction' based on the threshold. A detailed output table is rendered using st.dataframe with fraud classification results per provider.



Download Predictions

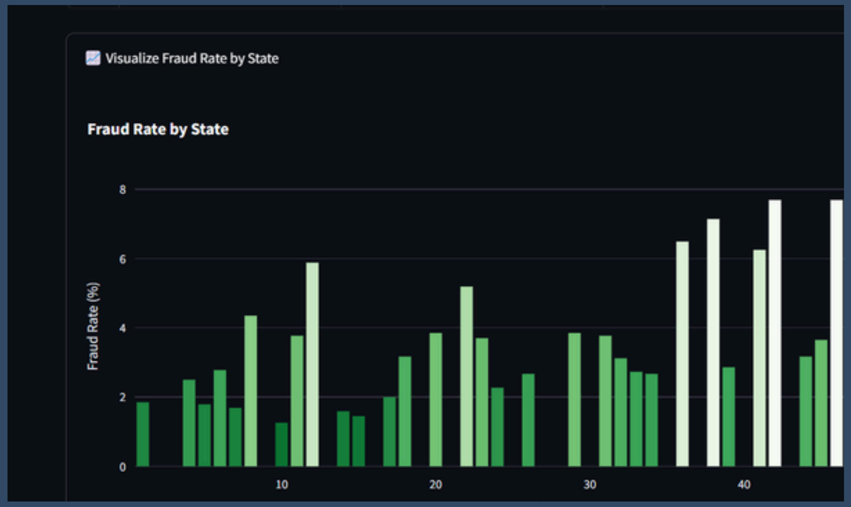
State-Level Fraud Risk Analysis

Top 3 Cleanest States (Lowest Fraud Rates)

	State	Fraud Rate (%)
1	2	
2	3	

Visualizations

The app uses Plotly to generate interactive graphs. A histogram shows the distribution of fraud probability scores, and a bar chart compares the number of fraudulent vs. non-fraudulent providers. These help users interpret the predictions more visually.



State-Level Insights

If the test file includes a 'State' column, the app computes fraud rate per state and shows a ranked table along with a green gradient bar chart. Users can also export all predictions using st.download_button as a downloadable CSV.

Preparation

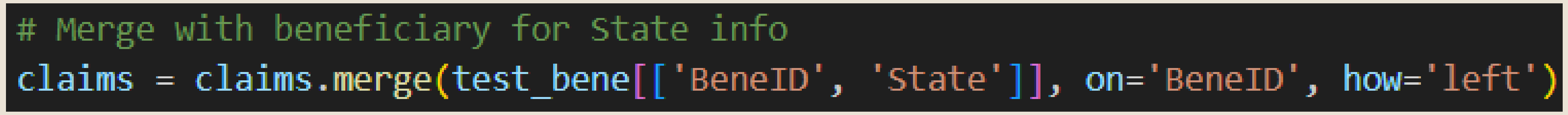
of input script

To ensure that the fraud prediction model receives data in the correct format, we created a Python script (prepare_prediction_input.py) that preprocesses raw Medicare test data into a clean, model-ready CSV file.

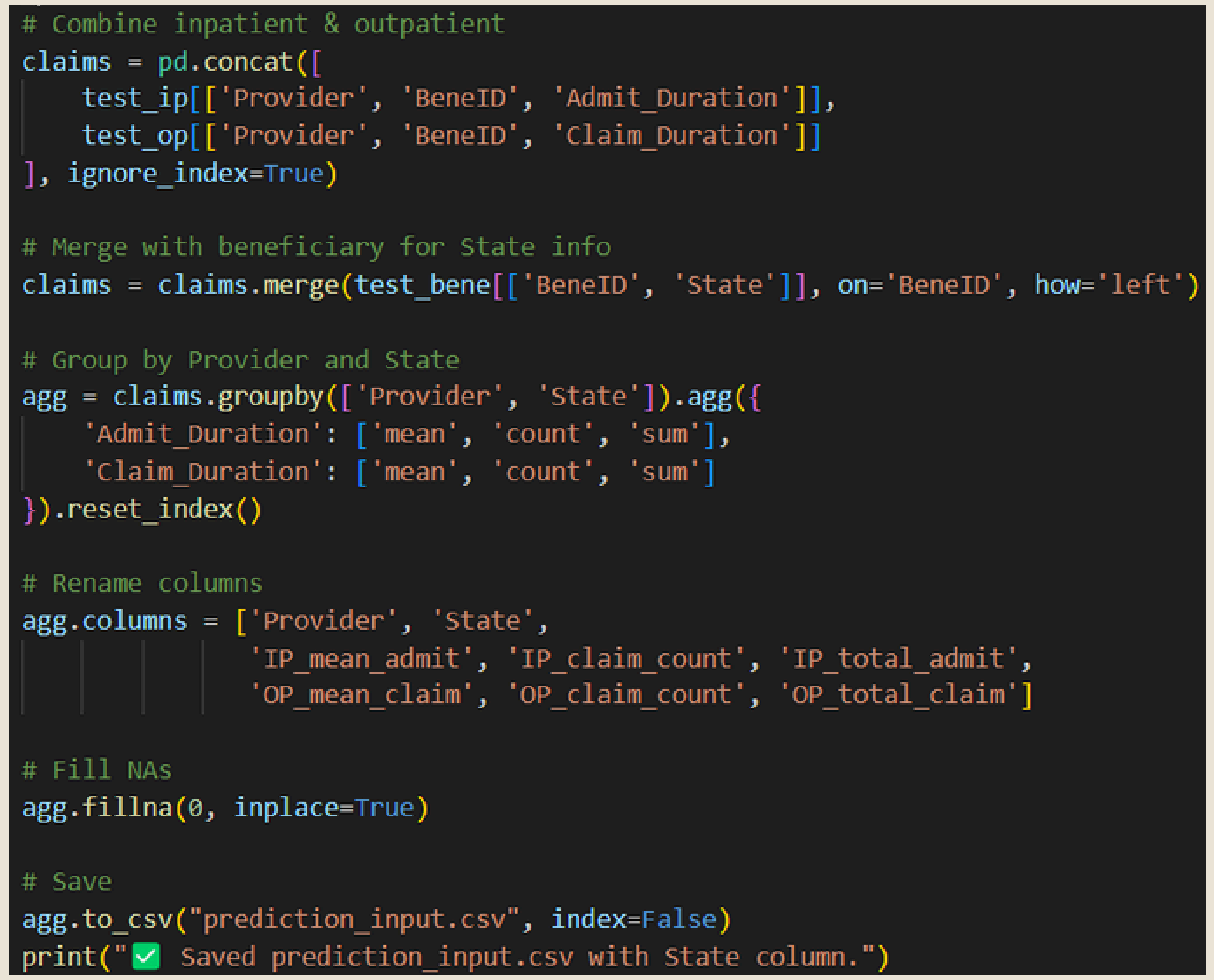
This script loads inpatient, outpatient, and beneficiary datasets and calculates important temporal features such as admission duration and claim duration by parsing and subtracting date fields. These durations are then combined and grouped by Provider and State to generate statistical summaries like mean admit time, total claims, and claim counts.

The script also merges beneficiary information to retain the provider's State, which is useful for later visualization.

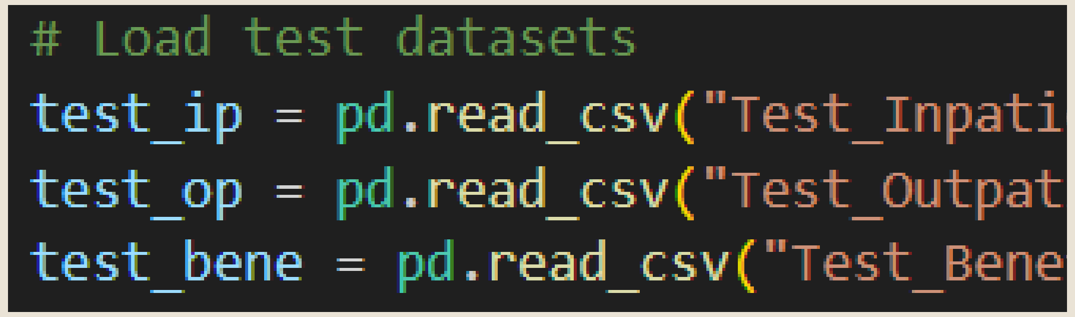
Once grouped and renamed, the resulting dataset is saved as prediction_input.csv. This file is then uploaded to the Streamlit web app for live fraud prediction. This step is crucial for aligning the test data with the structure the XGBoost model expects, ensuring consistency between training and prediction workflows.



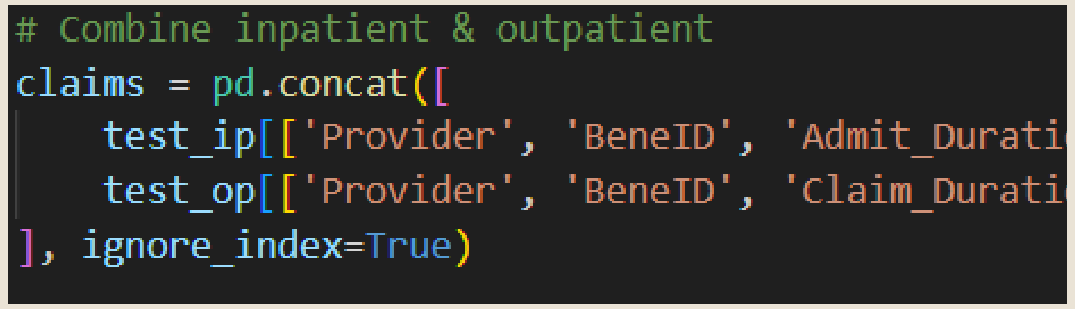
```
# Merge with beneficiary for State info
claims = claims.merge(test_bene[['BeneID', 'State']], on='BeneID', how='left')
```



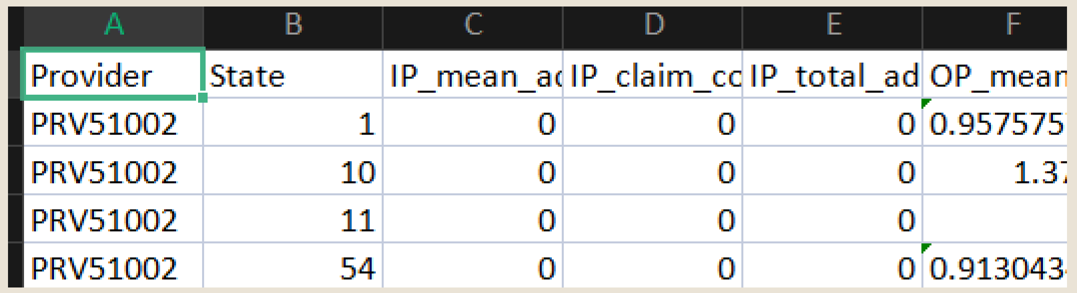
```
# Combine inpatient & outpatient
claims = pd.concat([
    test_ip[['Provider', 'BeneID', 'Admit_Duration']],
    test_op[['Provider', 'BeneID', 'Claim_Duration']]
], ignore_index=True)

# Merge with beneficiary for State info
claims = claims.merge(test_bene[['BeneID', 'State']], on='BeneID', how='left')

# Group by Provider and State
agg = claims.groupby(['Provider', 'State']).agg({
    'Admit_Duration': ['mean', 'count', 'sum'],
    'Claim_Duration': ['mean', 'count', 'sum']
}).reset_index()

# Rename columns
agg.columns = ['Provider', 'State',
                'IP_mean_admit', 'IP_claim_count', 'IP_total_admit',
                'OP_mean_claim', 'OP_claim_count', 'OP_total_claim']

# Fill NAs
agg.fillna(0, inplace=True)

# Save
agg.to_csv("prediction_input.csv", index=False)
print("✅ Saved prediction_input.csv with State column.")
```



```
# Load test datasets
test_ip = pd.read_csv("Test_Inpati
test_op = pd.read_csv("Test_Outpat
test_bene = pd.read_csv("Test_Bene
```



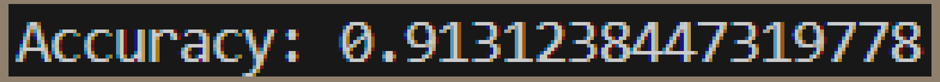
```
# Combine inpatient & outpatient
claims = pd.concat([
    test_ip[['Provider', 'BeneID', 'Admit_Durati
    test_op[['Provider', 'BeneID', 'Claim_Durati
], ignore_index=True)
```



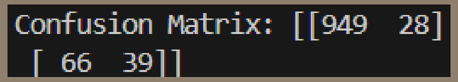
A	B	C	D	E	F
Provider	State	IP_mean_ad	IP_claim_cc	IP_total_ad	OP_mean
PRV51002	1	0	0	0	0.957575
PRV51002	10	0	0	0	1.37
PRV51002	11	0	0	0	
PRV51002	54	0	0	0	0.913043

Testing and Validation

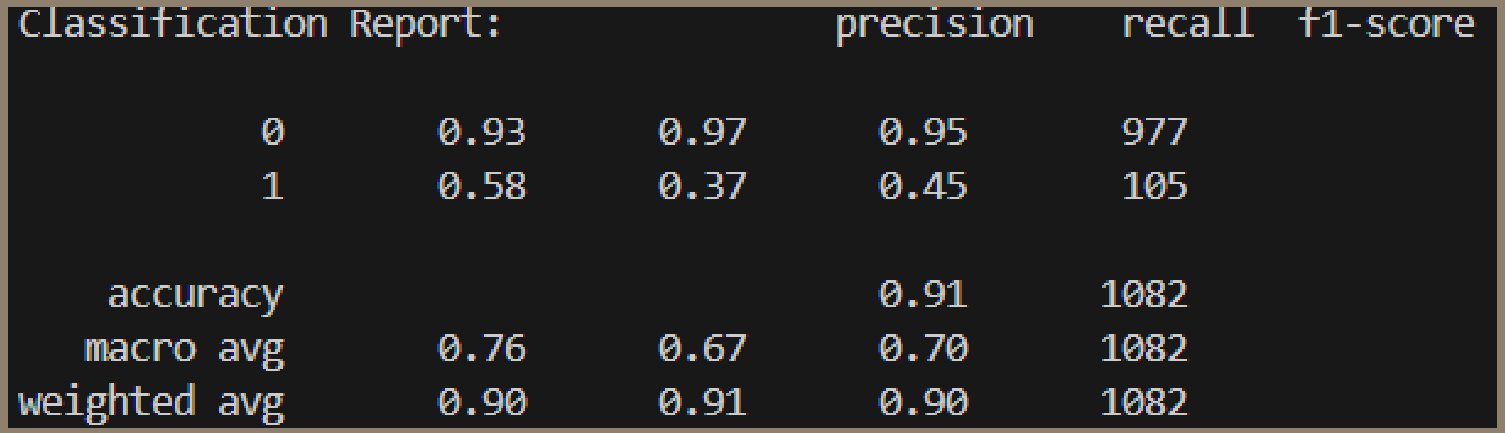
To ensure the reliability and correctness of our Medicare Fraud Detection System, we performed both model-level and application-level testing.



```
Accuracy: 0.9131238447319778
```



```
Confusion Matrix: [[949  28]
 [ 66  39]]
```



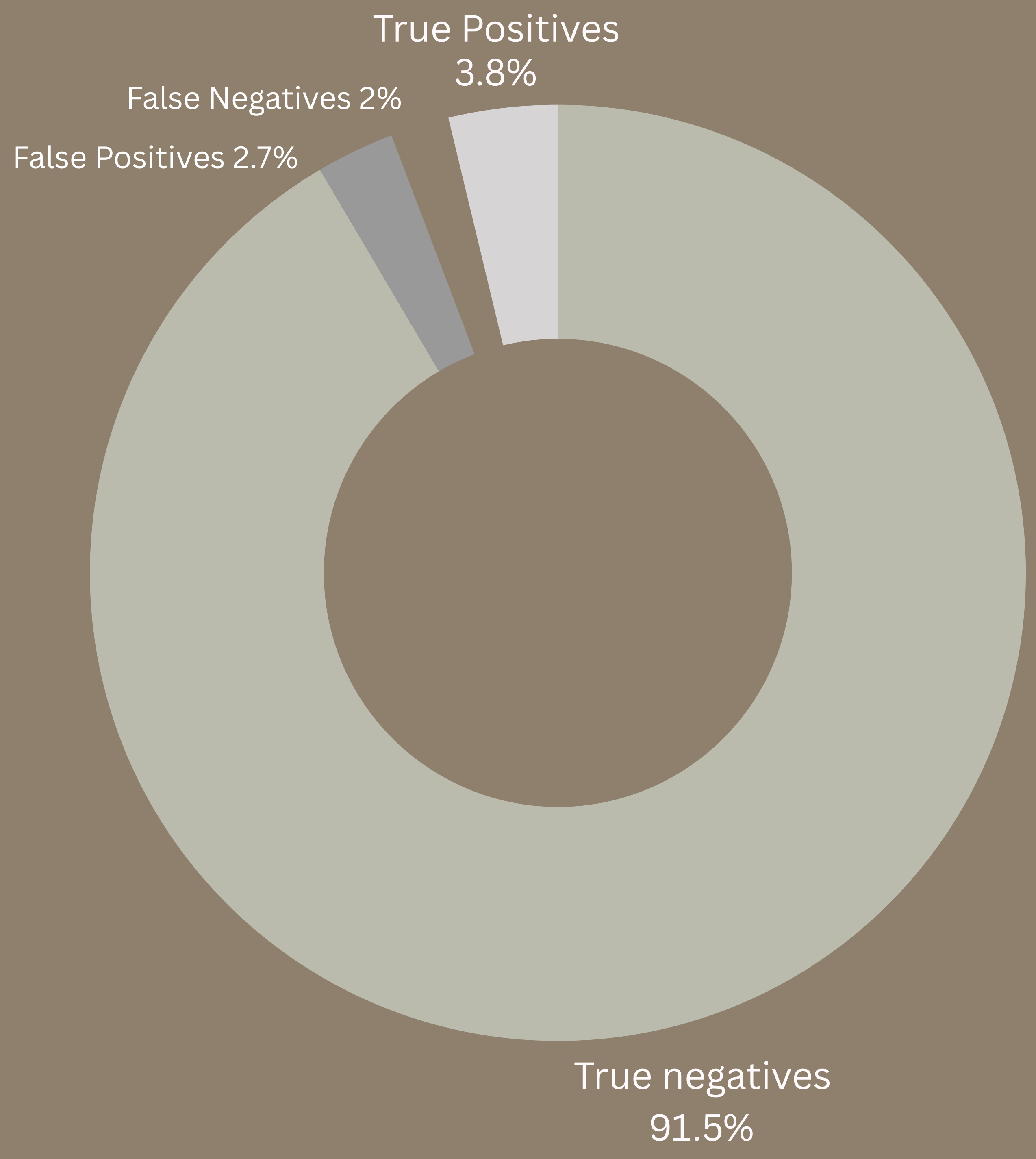
```
Classification Report:              precision    recall  f1-score

           0       0.93      0.97      0.95       977
           1       0.58      0.37      0.45       105

    accuracy                           0.91      1082
   macro avg       0.76      0.67      0.70      1082
weighted avg       0.90      0.91      0.90      1082
```

During model development, we evaluated the XGBoost classifier using a confusion matrix and accuracy score on the test dataset. This allowed us to confirm that the model was correctly distinguishing between fraudulent and non-fraudulent providers.

We also validated that the fraud threshold slider correctly influences the binary classification logic and that all charts and downloadable outputs behave as expected. This end-to-end testing confirmed that both the model and UI operate reliably, producing accurate and user-friendly fraud predictions.

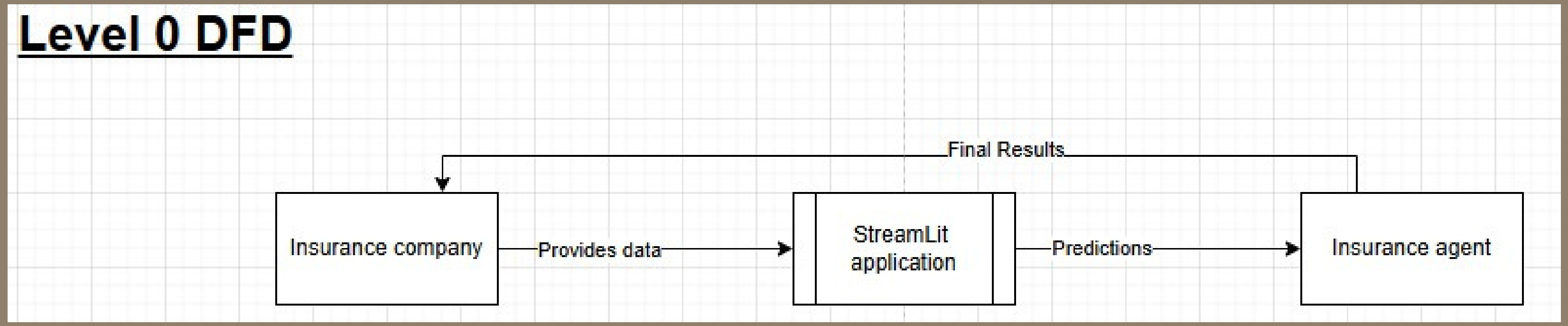
The classification report :

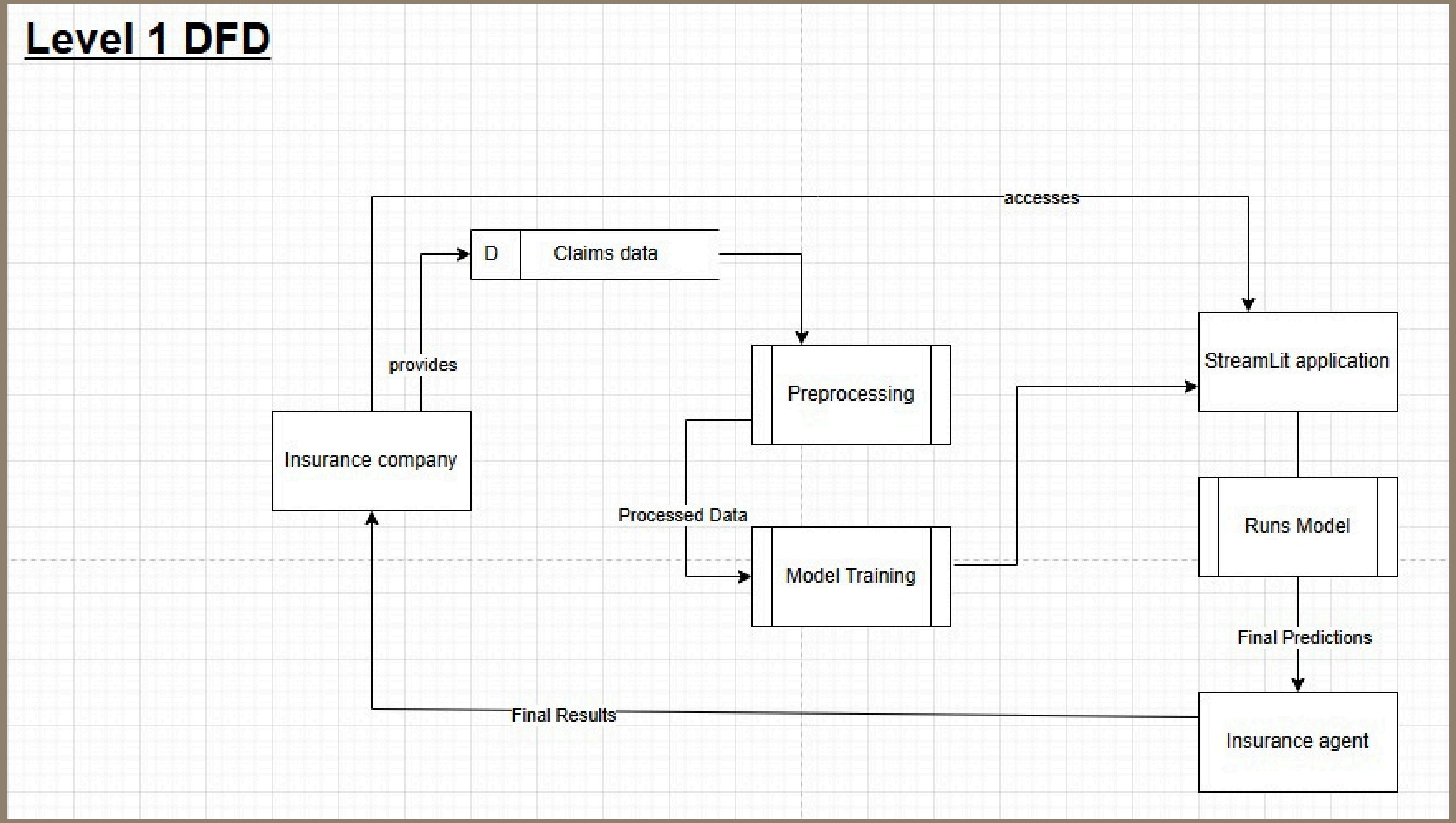
- Class 0 :
    - Precision = 0.93 (Out of all the times your model said “Not Fraud,” 93% were correct)
    - Recall = 0.97 (Out of all the actual Not Fraud cases, it caught 97% of them)
    - F1-score = 0.95 (This is a combined measure of both precision and recall. Higher is better)
    - Support = 977 (There were 977 Not Fraud providers in the test set)
- Class 1 :
    - Precision = 0.58 (Out of all the times your model predicted “Fraud,” only 58% were actually fraud)
    - Recall = 0.37 (Out of all the real fraud cases, it only found 37% of them)
    - F1-score = 0.45 (That’s not great — it means the model is struggling to catch frauds)
    - Support = 105 (There were 105 fraud cases in total in the test set)

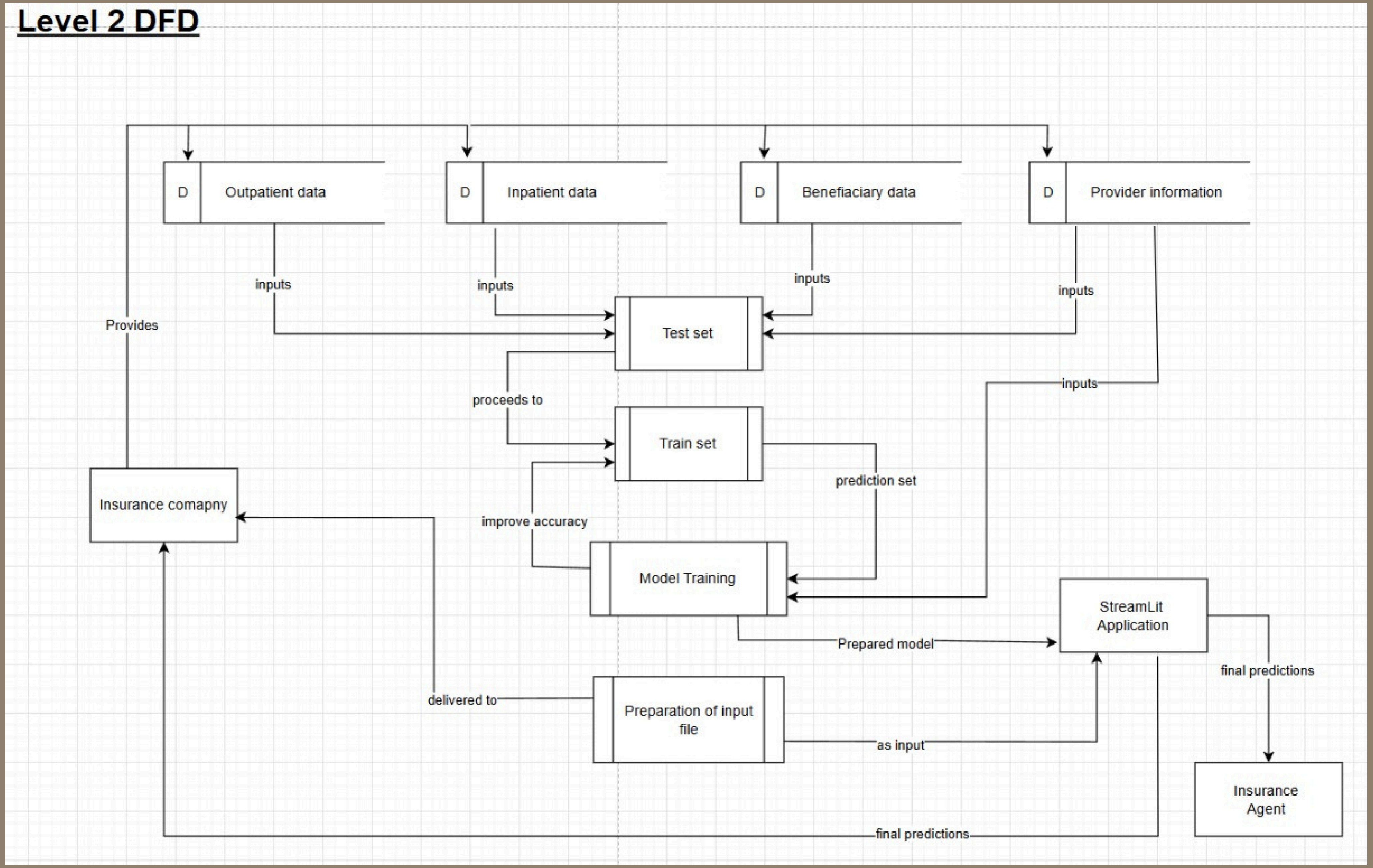


Data Flow Diagrams :

A Data Flow Diagram (DFD) is a graphical representation of the flow of data through a system. It is used to visually depict how input data is transformed into output data through various processes within a system.



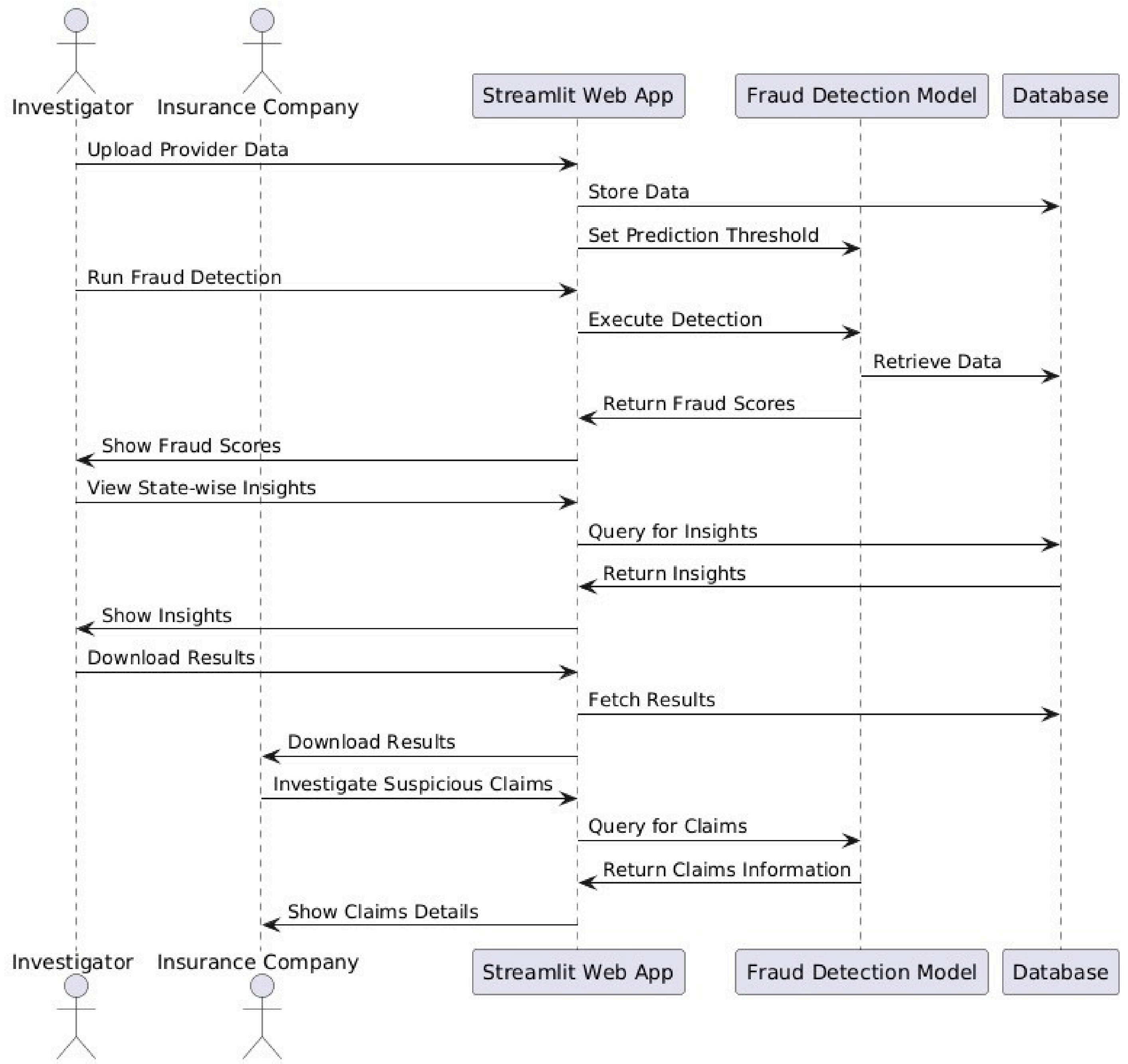




Sequential Diagram :

A Sequence Diagram shows how objects interact over time via method calls and data transfers.

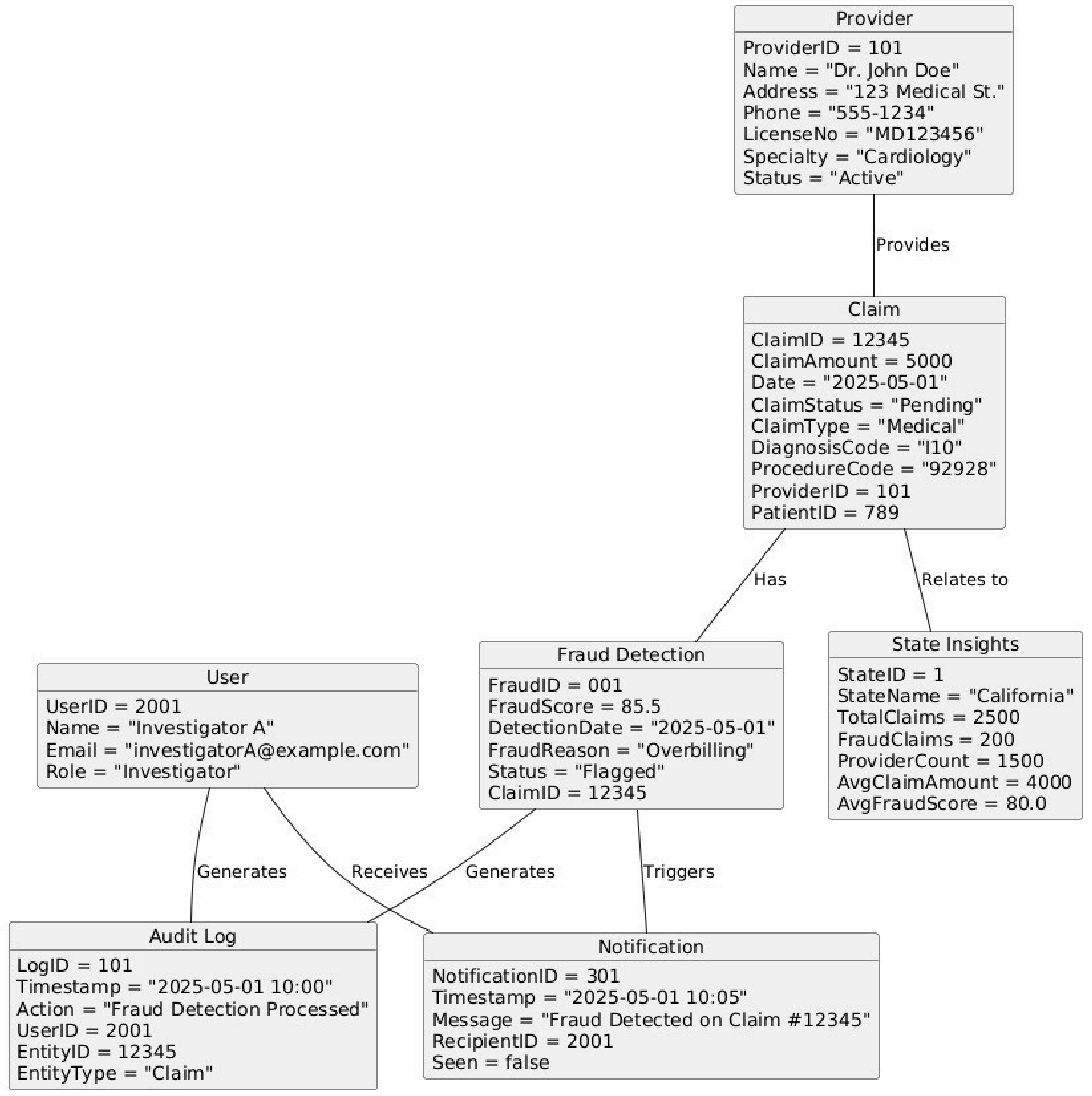
To model the dynamic behavior — e.g., how a user uploads a dataset, which is preprocessed, fed into the ML model, and then predictions are displayed.



Object Diagram :

An Object Diagram shows a snapshot of instances (objects) of classes at a particular time. It represents the system's static view.

To show how objects like Dataset, Model, Prediction, and User interact at a specific moment — for example, right after a fraud prediction is generated.



Provider
ProviderID = 101
Name = "Dr. John Doe"
Address = "123 Medical St."
Phone = "555-1234"
LicenseNo = "MD123456"
Specialty = "Cardiology"
Status = "Active"

Claim
ClaimID = 12345
ClaimAmount = 5000
Date = "2025-05-01"
ClaimStatus = "Pending"
ClaimType = "Medical"
DiagnosisCode = "I10"
ProcedureCode = "92928"
ProviderID = 101
PatientID = 789

User
UserID = 2001
Name = "Investigator A"
Email = "investigatorA@example.com"
Role = "Investigator"

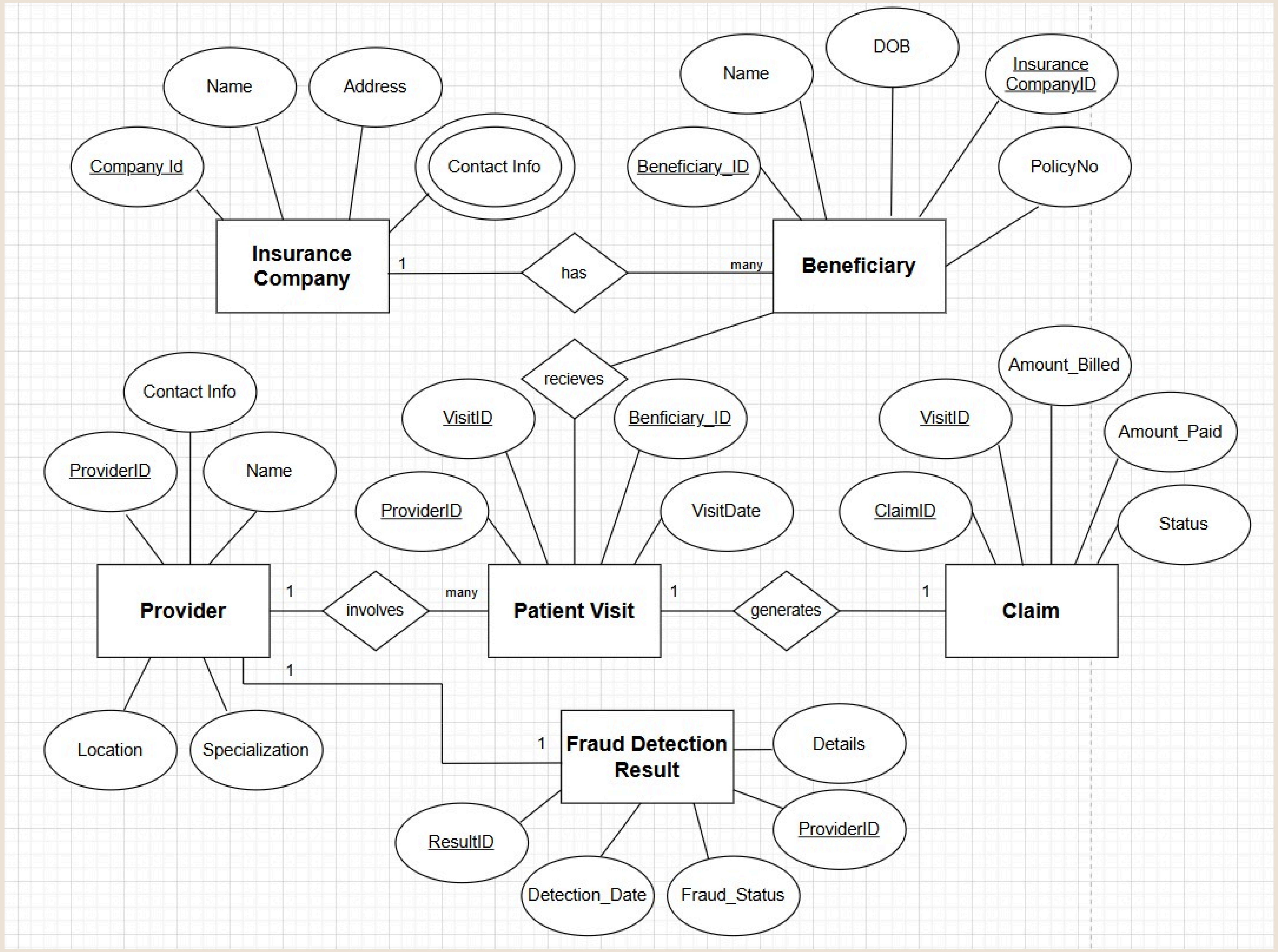
Fraud Detection
FraudID = 001
FraudScore = 85.5
DetectionDate = "2025-05-01"
FraudReason = "Overbilling"
Status = "Flagged"
ClaimID = 12345

State Insights
StateID = 1
StateName = "California"
TotalClaims = 2500
FraudClaims = 200
ProviderCount = 1500
AvgClaimAmount = 4000
AvgFraudScore = 80.0

Audit Log
LogID = 101
Timestamp = "2025-05-01 10:00"
Action = "Fraud Detection Processed"
UserID = 2001
EntityID = 12345
EntityType = "Claim"

Notification
NotificationID = 301
Timestamp = "2025-05-01 10:05"
Message = "Fraud Detected on Claim #12345"
RecipientID = 2001
Seen = false

Entity Relationship (ER) Diagram :



- An Entity-Relationship (ER) Diagram is a graphical representation of entities and their relationships in a database system. It is used during the database design phase to structure and understand the system's data requirements.
- Our system processes claims and patient visits to detect potential fraud.
- The machine learning model takes into account claim details, provider behaviour, and visit patterns to determine fraud.
- Detected results are linked to providers and stored with statuses like fraudulent or legitimate.

Significance of an ER Diagram:

- Clarifies system structure: Defines how data is organized and interrelated.
- Improves communication: Helps developers and stakeholders understand the data model.
- Foundation for DB design: Used to generate relational schemas for SQL databases.

National Institute of Technology,
Delhi

Thank you!

Contact Details : 231210064@nitdelhi.ac.in (Madhav Raj)
231210056@nitdelhi.ac.in (Kapil Meena)
231210038@nitdelhi.ac.in (Devesh Sarda)