The dataset contains detailed information about healthcare providers, claims, and associated metrics. It includes a column named is\_fraud, which indicates whether a claim is fraudulent (1) or not (0). This makes it suitable for a supervised learning classification problem.

### **Steps to Devise a Supervised Learning Algorithm**

1. **Understand the Data**
   * Identify the features that might influence the classification (e.g., Procedure\_Code, Provider\_Type, Valid\_Charge\_Ratio).
   * Check for missing or inconsistent data.
2. **Data Preprocessing**
   * Handle missing values.
   * Encode categorical variables (e.g., Provider\_Type, Procedure\_Code).
   * Normalize or scale numerical features to improve model performance.
3. **Feature Selection**
   * Perform feature importance analysis to identify which features contribute most to the classification.
4. **Model Selection**
   * Choose a supervised learning algorithm suitable for binary classification:
     + Logistic Regression
     + Decision Trees or Random Forest
     + Gradient Boosting Machines (e.g., XGBoost, LightGBM)
     + Neural Networks (if the dataset is large and complex)
5. **Model Evaluation**
   * Split the data into training and testing sets.
   * Use evaluation metrics such as accuracy, precision, recall, F1-score, and AUC-ROC.
6. **Explain Fraudulent Claims**
   * Use SHAP (SHapley Additive exPlanations) or feature importance plots to explain why the model flagged a claim as fraudulent.
7. **Deployment on Google Cloud Platform (GCP)**
   * Use GCP's AI tools like AutoML Tables, BigQuery ML, or Vertex AI for training and deploying the model.
   * Store the data in BigQuery for efficient querying and integration with the model.

Would you like me to proceed with exploratory data analysis (EDA) and preprocessing to prepare this dataset for modeling? ​

Code:  
from google.cloud import bigquery

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, roc\_auc\_score

from imblearn.over\_sampling import SMOTE

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

# Initialize BigQuery client

client = bigquery.Client()

# Query to fetch data

query = f"""

SELECT

Provider\_ID,

Provider\_Last\_Name,

Provider\_First\_Name,

Provider\_Type,

Provider\_Credentials,

Procedure\_Code,

Valid\_Charge\_Ratio,

Medicare\_Participation,

is\_fraud

FROM `eecse6893-436016.eecse6893\_fraud\_detect.eecse6893\_fraud\_detect\_table`

"""

fraud\_data = client.query(query).to\_dataframe()

if not fraud\_data.empty:

# Handle missing values

fraud\_data\_cleaned = fraud\_data.fillna({

"Provider\_ID": 0,

"Provider\_Last\_Name": "Unknown",

"Provider\_First\_Name": "Unknown",

"Provider\_Type": "Unknown",

"Provider\_Credentials": "Unknown",

"Procedure\_Code": "Unknown",

"Valid\_Charge\_Ratio": 0,

"Medicare\_Participation": "N/A"

})

# Select relevant columns

selected\_columns = [

"Provider\_ID", "Provider\_Last\_Name", "Provider\_First\_Name",

"Provider\_Type", "Provider\_Credentials", "Procedure\_Code",

"Valid\_Charge\_Ratio", "Medicare\_Participation", "is\_fraud"

]

filtered\_data = fraud\_data\_cleaned[selected\_columns]

# Separate features (X) and target (y)

X = filtered\_data.drop(columns=["is\_fraud"])

y = filtered\_data["is\_fraud"]

# Define categorical columns

categorical\_columns = [

"Provider\_Last\_Name", "Provider\_First\_Name", "Provider\_Type",

"Provider\_Credentials", "Procedure\_Code", "Medicare\_Participation"

]

# Preprocessing

preprocessor = ColumnTransformer(

transformers=[

("cat", OneHotEncoder(handle\_unknown="ignore"), categorical\_columns)

],

remainder="passthrough"

)

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42, stratify=y

)

# Apply SMOTE for class balancing

smote = SMOTE(random\_state=42)

X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(

preprocessor.fit\_transform(X\_train), y\_train

)

# Train Random Forest

model = RandomForestClassifier(random\_state=42, class\_weight="balanced")

model.fit(X\_train\_resampled, y\_train\_resampled)

# Predictions

y\_pred = model.predict(preprocessor.transform(X\_test))

# Evaluation

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

print(f"AUC-ROC: {roc\_auc\_score(y\_test, y\_pred):.2f}")

# Save results

encoder = preprocessor.named\_transformers\_["cat"]

categorical\_data = encoder.inverse\_transform(X\_test[:, :len(encoder.categories\_)])

numeric\_data = X\_test[:, len(encoder.categories\_):]

results = pd.DataFrame(numeric\_data, columns=X.columns[len(encoder.categories\_):])

results[categorical\_columns] = categorical\_data

results["Actual"] = y\_test.values

results["Predicted"] = y\_pred

results.to\_csv("fraud\_detection\_results.csv", index=False)

print("Results saved to 'fraud\_detection\_results.csv'.")

else:

print("No data available for training.")