**The Story of Heuristics in Fraud Detection**

**Background: The Challenge of Fraud Detection**

Fraud detection in the financial ecosystem is akin to finding a needle in a haystack, where the stakes are high and the cost of missed detection is even higher. Financial institutions process millions of transactions daily, each potentially susceptible to fraud. Yet, distinguishing fraudulent transactions from genuine ones is far from straightforward. This complexity arises due to the evolving tactics of fraudsters, the scale of data, and the need for real-time decisions.

In the quest for better fraud detection, machine learning models became a cornerstone. These models relied on historical data, crafted predictors, and customer-provided feedback to assign risk scores to transactions. However, while models evolved, one challenge remained constant: **how to handle ambiguous or incomplete feedback provided by customers.**

גילוי fraud במערכת האקולוגית הפיננסית דומה למציאת מחט בערימת שחת, שבה ההימור גבוה והעלות של איתור החמצה גבוהה אף יותר. מוסדות פיננסיים מעבדים מיליוני עסקאות מדי יום, שכל אחת מהן עשויה להיות חשופה להונאה. עם זאת, ההבחנה בין עסקאות הונאה לבין עסקאות אמיתיות היא רחוקה מלהיות פשוטה. מורכבות זו מתעוררת עקב הטקטיקות המתפתחות של רמאים, היקף הנתונים והצורך בהחלטות בזמן אמת.

בחיפוש אחר זיהוי טוב יותר של הונאה, מודלים של למידת מכונה הפכו לאבן יסוד. מודלים אלה הסתמכו על נתונים היסטוריים, מנבאים מיוצרים ומשוב שסופק על ידי הלקוח כדי להקצות ציוני סיכון לעסקאות. עם זאת, בזמן שהמודלים התפתחו, אתגר אחד נשאר קבוע: כיצד לטפל במשוב מעורפל או לא שלם שסופק על ידי הלקוחות.

**Initial Steps: Basic Heuristics**

Heuristics were first introduced as rules to propagate customer-provided feedback across similar transactions. For example, if a customer definitively marked a transaction as fraudulent (RESOLUTION = 'F'), the system would propagate this marking to other transactions sharing key traits, such as being from the same user or device. This ensured that **feedback enriched the model training process, even when only a subset of transactions was explicitly marked.**

For instance:

1. If a transaction was confirmed fraudulent with high confidence (RESOLUTION\_CONF\_LEVEL = 10), other transactions from the same user on the same day were heuristically marked with reduced confidence levels.
2. For genuine transactions, heuristics identified patterns like consistent device usage over time, signaling legitimate behavior.

These heuristics provided valuable input to models, improving their ability to classify unseen transactions. However, as the complexity of fraud scenarios grew, so did the limitations of basic heuristics.

**The Gap: Limitations of Traditional Approaches**

Despite their success, traditional heuristics had shortcomings:

* **Static Rules:** Heuristics relied on predefined rules, often lacking the flexibility to adapt to nuanced fraud patterns or customer-specific behaviors.
* **Partial Feedback Utilization:** Only a small fraction of cases received definitive feedback, leaving the majority unresolved and untreated.
* **Missed Context:** Traditional heuristics operated at the transaction level, often ignoring broader patterns within clusters or groups of transactions.

These gaps created a pressing need to refine the process. How could we move beyond static rules and leverage advanced methodologies to propagate feedback intelligently? Enter **case clustering and advanced heuristics.**

**The Innovation: Case Clustering with Advanced Heuristics**

To address the gaps, researchers turned to clustering techniques, particularly **K-Means clustering**, to group transactions into clusters based on similarity in features. The primary goal was to identify unresolved cases within a cluster and propagate markings intelligently based on the cluster's overall feedback.

**How it works:**

1. **Feature Selection:** Clustering leveraged features like transaction amount (AMOUNT\_USD) and preliminary score to define similarity. These features were chosen based on exploratory data analysis (EDA) to optimize separation between fraud and genuine clusters.
2. **Cluster Size Optimization:** The number of clusters was adjusted dynamically, typically set to 20% of daily cases. This ensured that clusters were representative yet granular enough for effective propagation.
3. **Marking Propagation:** Feedback from marked cases (e.g., fraud or genuine) within a cluster was used to heuristically mark unresolved cases, weighted by factors such as confidence level and distance from cluster centroids.

This approach added a layer of sophistication:

* Unmarked transactions in a cluster inherited feedback based on similarity.
* Weighting mechanisms ensured that closer cases (in feature space) received higher confidence adjustments.

**Impact: A Data-Enriched Model**

The adoption of clustering-based heuristics transformed the feedback enrichment process:

* **Higher Data Utilization:** By synthesizing markings for unresolved cases, models trained on enriched datasets, leading to improved performance.
* **Dynamic Rules:** Clustering allowed the system to adapt to customer-specific behaviors, transaction trends, and emerging fraud patterns.
* **Improved Predictions:** Simulations showed that models trained with clustered heuristics consistently outperformed those trained on original data, particularly in scenarios with sparse feedback.

**Conclusion: The Journey of Heuristics**

The evolution of heuristics in fraud detection reflects the broader trend of integrating data-driven methodologies into traditional rule-based systems. From static heuristics to clustering-based feedback propagation, each step has been driven by the need to **maximize feedback utilization, enhance model accuracy, and stay ahead of sophisticated fraud schemes.**

As fraud continues to evolve, so too will heuristics. With advancements like unsupervised learning, deep clustering, and adaptive feedback mechanisms on the horizon, the future promises even greater synergy between human expertise and machine intelligence in combating fraud.

**Approach:**  
We introduced a clustering-based heuristic mechanism to address this issue:

1. **Feature Selection and Exploration:**
   * We collaborated with fraud analysts to identify key features, such as **PRELIMINARY\_SCORE** (risk score), **AMOUNT\_USD**, **PAN\_AGE**, and **PAN\_MERCH\_NUM\_OF\_HITS** (frequency of user-merchant interactions).
   * Using EDA (e.g., correlation plots, box plots), we confirmed these features offered sufficient separation between fraudulent and genuine transactions.
2. **Clustering Implementation:**
3. **Heuristic Label Propagation:**
   * For each cluster, we calculated the proportion of marked cases labeled as fraud or genuine.
   * Unmarked cases were assigned a label based on this ratio, with confidence levels derived from their distance to the cluster centroid.
4. **Simulations and Testing:**
   * We created a train set from three months of historical data with 20% undersampling of users with markings.
   * Models trained on heuristic-augmented data were compared against those trained on the original sparse dataset.

**Results:**

* **Model Improvement:** The enriched dataset improved fraud detection metrics, especially recall, enabling the detection of more high-value fraud cases.
* **Operational Efficiency:** The heuristic mechanism reduced the dependency on manual markings, lowering operational costs for clients.
* **Dynamic Adaptability:** The clustering approach dynamically adapted to changing customer behaviors and transaction patterns.

An interesting observation was that even weaker features (e.g., **AMOUNT\_USD**) contributed to improving performance, as identifying high-value fraudulent transactions significantly impacted key metrics.

**Conclusion:**  
This solution automated the feedback propagation process, enriching the dataset and making the fraud detection model more robust. It also demonstrated the importance of integrating domain knowledge with machine learning techniques to address practical challenges in real-world systems.

This version is concise, technical, and structured to suit an interview setting. Let me know if you’d like further refinements!

בזיהוי fraud, משתמשים במודל של supervised learning שמסתמך במידה רבה על feedback שסופק על ידי הלקוח. המשוב מגיע בשתי צורות

1. **Case Management Markings** : אלו טרנזאקציות אשר מסומנות באופן ידני או אוטומטי על ידי לקוח (issuer או בנק) המציינים אם fraudulent או genuine. הfeedback - כולל גם confidence level (למשל, "suspected fraud" או "confirmed fraud" ) והוא מהוה משקולות בתהליך האימון של מודל.

פידבק מהסוג הזה מספק לנו מקרים של false positives ו- true positives

1. **Chargebacks**: אלה מתרחשים כאשר לקוח מערער על עסקה שאושרה בעבר. כלומר אלו false negatives, מכיוון שהם מזהים כ- fraud שהחמיץ המודל.

אחד האתגרים העיקריים הוא שלקוחות מסמנים רק תת-קבוצה קטנה של מקרים עקב אילוצי משאבים, וכתוצאה מכך מתקבל biased data. כדי לשפר את ביצועי המודל, זיהינו את הצורך במנגנון להפצת labeling מטרנזאקציות מסומנות לאלו שלא מסומנות דומות עם confidence level שניתן לעמוד כמותית, כדי להבטיח שתהליך פרופגציה יהיה אוטומטי, מוגדר היטב, ניתן למדידה וניתן לכוונון.

במודל הישן היוריסטיקות היו מוגדרות ידנית בעזרת fraud analysts . אלה היו חוקים שהסתמכו על פאטרנים שנראו בדטא. למשל,

* + if PRELIMINARY\_SCORE > 0.8 and PAN\_AGE < 30:
  + label = "fraud"
  + confidence = 0.9

**הגבלות של שיטה ישנה** :

* + 1. הכללים הם סטטיים, אין גמישות של טיונינג של בהתבסס על פאטרנים משתנים פר לקוח
    2. רק חלק קטן של טרנזאקציות מקבל פידבק מוגדר והרוב נשאר לא מטופל
    3. יוריסטיקות כאלה עובדות רק ברובד של טרנזאקציות ולא טופסות פאטרנים בתוך קלסטרים או קבוצות של טרנזאקציות

כל זה הוביל לצורך של חיפוש אחרי מכניזם יותר מתוחכם והוביל לשיטה של clustering ויוריסטיקות מתקדמות.

**גישה**

1. **Feature Selection and Exploration**

* **בקולבורציה עם fraud analysts נבחרו features כמו**

**PRELIMINARY\_SCORE** (risk score)

, **AMOUNT\_USD**,

**PAN\_AGE**, and

**PAN\_MERCH\_NUM\_OF\_HITS** תדירות אינטראקציות בין משתמש-סוחר (user-merchant)

* נעשה EDA (box plots, קורלציות) כדי לבדוק שאכן קיימת ספרציה טובה בין fraud and genuine טרנזאקציות

1. **אימפלמנטציה של קלסטרינג :**

* **השתמשנו ב K-means בגלל הרובסטיות שלו כנגד outliers ופשטות ב production**
* **מספר קלסטרים מוגדר כ 20% מכמות טרנזאקציות יומית**
* **Features היו מנורמלות בעזרת** standard scaler

1. **פרופגציה של סימני יוריסטיקות :**

* לכל קלסטר מחשבים פרופורציה של טרנזאקציות מסומנות כ fraud או genuine
* טרנזאקציות לא מסומנות מקבלות סימון מתבסס על היחס הזה עם confidence level שמוגדר מתבסס על מרחק מ centroid של קלסטר
* קלסטרים נבנו על בסיס יומי
* Centroid : מוגדר כממוצע של ערכי features בו (למשל, בקלסטר מסויים יצאו 30% genuine ו- 70 fraud % , 70< 50 ולכן כל הטרנזאקציות הלא מסומנות יהיו fraud )
* כל מקרה לא מסומן משוייך לסנטרואיד הקרוב

הערות :

* **איזון בין granularity ו- יעילות:**

פירוט: על ידי הגבלת מספר הclusters ל-20%, כל cluster נשאר מובהק סטטיסטית תוך שהוא מייצג דפוסים משמעותיים בנתונים.

יעילות: מספר קטן יותר ומבוקר של clusters ממזער את המחיר החישובים וזמן

* 

1. **סימולציות וולידציה**

* יצרנו train set משלושה חודשים רציפים של דטא היסטורי עם 20% undersampling של יוזרים עם סימונים
* מודלים שתאמנו על דטא עם תוספת יוריסטיקות הושוו מול אלו שהתאמנו על data set מקורי .

Your code and description showcase a well-thought-out process for clustering transaction data and applying advanced heuristics to enhance fraud detection using K-Means clustering. Here’s a summary and breakdown of key elements:

**Summary of the Process:**

**1. Data Preparation:**

* **Input:** Raw transaction logs from CSV.
* **Filtering:**
  + Transactions are filtered by date (SHORT\_DATE) and user hash range (USER\_HASH).
  + Key cases, such as those marked as fraud or suspicious (RESOLUTION\_SOURCE and POLICY\_ACTION), are selected.
* **Undersampling:** User IDs are filtered based on specific conditions to limit the size of data being processed.

**Undersampling** balances the classes by reducing the number of majority class samples (genuine transactions), allowing the model to focus more on the minority class (fraudulent transactions).

undersampling was applied to select a representative subset of users with labeled (marked) transactions. Here's how it might have been implemented:

**Steps:**

1. **Filter Users with Marked Transactions**  
   Only users who had at least one marked transaction (fraud or genuine) were considered. This ensures that the subset contains relevant examples for training.

marked\_users = full\_dataset[full\_dataset["is\_marked"] == True]["user\_id"].unique()

1. **Select a Subset of Users**  
   A subset of users was randomly sampled to include around 20% of the users who had marked transactions. This step reduces the dataset size while retaining enough variety.

sampled\_users = random.sample(list(marked\_users), int(0.2 \* len(marked\_users)))

1. **Extract Transactions for Sampled Users**  
   All transactions for the sampled users were included in the training set. This ensures that patterns for these users are preserved.

undersampled\_dataset = full\_dataset[full\_dataset["user\_id"].isin(sampled\_users)]

1. **Balance Fraudulent and Genuine Transactions (Optional)**  
   If needed, further undersampling was applied within the subset to balance the ratio of fraudulent and genuine transactions.

fraud\_transactions = undersampled\_dataset[undersampled\_dataset["label"] == "fraud"] genuine\_transactions = undersampled\_dataset[undersampled\_dataset["label"] == "genuine"]

# Balance the classes

genuine\_sample = genuine\_transactions.sample(n=len(fraud\_transactions), random\_state=42) balanced\_dataset = pd.concat([fraud\_transactions, genuine\_sample])

**Parameters Used in Undersampling**

Key parameters were:

* **Train Set Period:** Three months of transaction data were used for training.
* **Sampling Ratio:** 20% of users with marked transactions were undersampled.
* **Class Balance:** Ensured a reasonable balance between fraud and genuine cases in the training data.
* **Random Seed (Optional):** A seed was used to make the sampling process reproducible.

**2. Clustering:**

* **Scaling:** Data is scaled using either StandardScaler or MinMaxScaler, depending on the input parameter.
* **Feature Selection:** Only a subset of features (AMOUNT\_USD, or additional like PAN\_AGE, etc.) is used for clustering.
* **K-Means Algorithm:**
  + Clusters are defined based on the number of transactions and a cluster ratio.
  + Distances from the cluster centroid are calculated to determine weightings.

**3. Marking and Propagation:**

* **Marking Clusters:** Based on RESOLUTION and confidence levels (RESOLUTION\_CONF\_LEVEL), a representative marking for each cluster is calculated.
* **Confidence Adjustment:**
  + Distances from the centroid are used as weights.
  + Markings are propagated to all unresolved cases in the cluster.
* **Rules for Prioritization:** Existing markings have priority over cluster-derived markings.

**4. Output Generation:**

* The modified event logs with updated markings are saved for further analysis or training of fraud detection models.

**Key Innovations in the Approach:**

1. **Cluster-Based Marking:**
   * By clustering similar cases and propagating markings, the process leverages existing feedback (e.g., fraud labels) to enrich data and improve model training.
2. **Dynamic Cluster Sizing:**
   * The number of clusters is dynamically adjusted based on the transaction count and a predefined cluster ratio, ensuring flexibility across datasets.
3. **Confidence Weighting:**
   * Using distance-to-centroid as a weight introduces a nuanced way of propagating confidence levels within clusters.
4. **Heuristics Integration:**
   * The approach integrates client-provided heuristics with data-driven clustering, balancing rule-based and statistical methodologies.

**Heuristics Implementation:**

The heuristics rules ensure logical consistency, such as:

* Marking all same-day transactions from a user as fraud if there is a definitive fraud marking (RESOLUTION = F).
* Removing heuristic-based markings if no definitive markings remain.
* Applying confidence adjustments based on recent transaction activity or device characteristics.

**Improvements for Future Implementation:**

1. **Feature Engineering:**
   * Include additional derived features (e.g., transaction time, location, or device characteristics) to improve cluster separation.
2. **Clustering Algorithm:**
   * Experiment with other clustering methods (e.g., DBSCAN, hierarchical clustering) that might better capture non-linear relationships.
3. **Parallel Processing:**
   * Implement multiprocessing to handle large datasets more efficiently.
4. **Evaluation Metrics:**
   * Include detailed metrics to evaluate the impact of synthesized markings on model performance, such as precision, recall, and AUC on test data.
5. **Parameter Optimization:**
   * Automate the selection of parameters like cluster\_ratio using techniques like grid search or Bayesian optimization.

Would you like a deeper dive into any specific part, such as optimizing the clustering process, enhancing heuristics, or integrating these steps into a broader ML pipeline?

**Heuristics Implementation in Your Code**

Heuristics in your code were used to assign labels and confidence levels to unmarked transactions based on patterns identified in marked cases. Below is an expanded explanation of the heuristics implementation process:

**1. Why Use Heuristics?**

Supervised learning relies on labeled data, but in fraud detection:

* A significant portion of transactions are **unmarked** due to operational constraints.
* Gathering enough labeled data is costly and time-consuming for clients.

To address this, heuristics were implemented to:

* Automatically assign labels (fraud or genuine) to unmarked transactions.
* Estimate a **confidence level** for each heuristic label to reflect the certainty of the assignment.

**2. Steps in Heuristics Implementation**

**Step 1: Exploratory Data Analysis (EDA)**

To develop heuristics, **EDA** was conducted to identify features that separate fraudulent and genuine transactions.  
Examples of **features studied**:

* PRELIMINARY\_SCORE: A risk score computed by the engine.
* AMOUNT\_USD: The transaction amount in USD.
* Age-related features like PAN\_AGE (card age) and DEVICE\_AGE.
* Interaction features like PAN\_MERCH\_NUM\_OF\_HITS (number of hits between card and merchant).

Techniques used:

* **Scatterplots and Correlation Analysis:** To observe separations between fraud (F) and genuine (G) transactions.
  + Example: sns.lmplot() was used to visualize the correlation between AMOUNT\_USD and PRELIMINARY\_SCORE.
* **Histograms and Boxplots:** To analyze distributions of features like age or transaction amounts and identify anomalies.

**Step 2: Define Initial Heuristics**

Heuristics were derived based on patterns observed in marked transactions:

* **Feature Thresholds:** Rules like "High PRELIMINARY\_SCORE and low PAN\_AGE correlate with fraud."
  + Example:
  + if PRELIMINARY\_SCORE > 0.8 and PAN\_AGE < 30:
  + label = "fraud"
  + confidence = 0.9
* **Clusters of Transactions:** Group similar transactions and assign the cluster a label and confidence based on majority voting from marked transactions.

**Step 3: Clustering for Unmarked Transactions**

To assign labels to unmarked transactions, clustering was used:

* **Clustering Algorithm:** K-Means clustering was selected due to:
  + Its simplicity and robustness to outliers.
  + Production constraints.
* **Parameters:**
  + Clusters: Defined as 20% of the daily cases.
  + Distance from Centroid: Used to estimate the confidence level.

Each cluster's transactions were labeled based on:

* The **majority label** of the marked cases in the cluster.
* A **confidence level** derived from the distance to the cluster centroid.

Example:

# Compute confidence as distance from cluster centroid

cluster\_confidence = 1 / (1 + distance\_from\_centroid)

**Step 4: Simulation and Validation**

Heuristics were tested in a simulated environment:

1. Simulate labeled data by applying heuristics to the unmarked dataset.
2. Train the model on this heuristic-labeled dataset.
3. Evaluate performance on a separate test set to compare with the original supervised learning model.

**3. Key Features of Heuristics Implementation**

1. **Dynamic Adaptation:**
   * Heuristics were re-computed daily using fresh transaction data.
   * Clustering parameters were tuned for each client to ensure relevance.
2. **Confidence Scoring:**
   * Each transaction was assigned a confidence level, reflecting the certainty of its heuristic label.
   * Confidence was derived from:
     + Feature thresholds (e.g., high risk scores).
     + Distance from cluster centroid.
3. **Client-Specific Customization:**
   * Features and thresholds were selected in consultation with fraud analysts for each client.
   * Parameters like cluster size and ratio of marked cases per cluster were adjusted to align with client needs.

**4. Example of Heuristics in Code**

Here’s a simplified implementation example based on your description:

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

# Sample dataset

data = {

"PRELIMINARY\_SCORE": [0.85, 0.3, 0.7, 0.95, 0.4],

"PAN\_AGE": [20, 50, 30, 25, 60],

"AMOUNT\_USD": [100, 500, 200, 1000, 300],

"is\_marked": [True, True, False, False, False],

"label": ["fraud", "genuine", None, None, None],

}

df = pd.DataFrame(data)

# Standardize features

scaler = StandardScaler()

features = scaler.fit\_transform(df[["PRELIMINARY\_SCORE", "PAN\_AGE", "AMOUNT\_USD"]])

# K-Means clustering

kmeans = KMeans(n\_clusters=2, random\_state=42)

df["cluster"] = kmeans.fit\_predict(features)

# Assign heuristic labels based on cluster majority

for cluster in df["cluster"].unique():

cluster\_data = df[df["cluster"] == cluster]

marked\_data = cluster\_data[cluster\_data["is\_marked"]]

if not marked\_data.empty:

# Determine majority label and assign to unmarked transactions

majority\_label = marked\_data["label"].mode()[0]

confidence = 1 / (1 + cluster\_data["PRELIMINARY\_SCORE"].std())

df.loc[df["cluster"] == cluster, "heuristic\_label"] = majority\_label

df.loc[df["cluster"] == cluster, "confidence"] = confidence

print(df)

**5. Results and Improvements**

**Results**

* **Efficiency:** Automated labeling reduced reliance on manual feedback from clients.
* **Performance:** Simulations showed comparable or improved model performance compared to using only client-marked data.
* **Fraud Detection:** Heuristics effectively captured high-value fraudulent transactions, significantly boosting performance metrics like recall.

**Improvements:**

* **Enhanced Features:** Additional derived features could further refine heuristics (e.g., time-of-day patterns).
* **Dynamic Clustering:** Explore adaptive clustering algorithms to handle evolving transaction patterns.
* **Confidence Calibration:** Use a probabilistic model to fine-tune confidence levels.

**Summary**

Your heuristics implementation combined feature-based rules and clustering to automate transaction labeling and confidence scoring. It addressed the challenges of limited labeled data while maintaining adaptability and relevance to client-specific needs. This approach demonstrated significant potential to improve model performance and operational efficiency.