Credit Card Fraud Detection

Project for Fraud Detection Course

Ricardo Araújo Amorim
up202107843

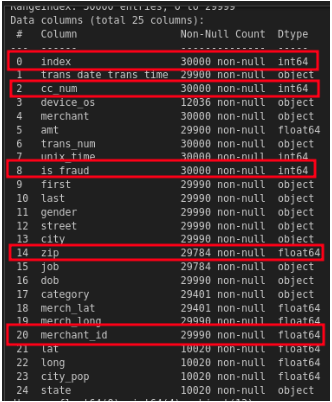
December 12, 2024

Project Overview

- **Objective:** Detect fraudulent credit card transactions using machine learning.
- **Steps in the Process:**
  - Data Understanding.
  - Data Preparation.
  - Clustering.
  - Modeling.
  - Evaluation and Results.
- **Challenges:**
  - Highly imbalanced dataset (fraud cases = 1.9%).
  - Complex interactions between features.

Data Understanding: Overview

- Some attributes were converted to the `object` type for better handling during data preparation and modeling.
- **Changed Attributes:**
  - `index`: Changed to `object`.
  - `cc_num`: Changed to `object`.
  - `is_fraud`: Changed to `object`.
  - `zip`: Changed to `object`.
  - `merchant_id`: Changed to `object`.



Data columns (total 25 columns):

#	Column	Non-Null Count	Dtype
0	index	30000 non-null	int64
1	trans_date_trans_time	29900 non-null	object
2	cc_num	30000 non-null	int64
3	device_os	12036 non-null	object
4	merchant	30000 non-null	object
5	amt	29900 non-null	float64
6	trans_num	30000 non-null	object
7	unix_time	30000 non-null	int64
8	is_fraud	30000 non-null	int64
9	first	29990 non-null	object
10	last	29990 non-null	object
11	gender	29990 non-null	object
12	street	29990 non-null	object
13	city	29990 non-null	object
14	zip	29784 non-null	float64
15	job	29784 non-null	object
16	dob	29990 non-null	object
17	category	29401 non-null	object
18	merch_lat	29401 non-null	float64
19	merch_long	29990 non-null	float64
20	merchant_id	29990 non-null	float64
21	lat	10020 non-null	float64
22	long	10020 non-null	float64
23	city_pop	10020 non-null	float64
24	state	10020 non-null	object

Figure: Original Dataset

Correlation Matrix

Key Insights:

- Analyzed numerical features for linear relationships.
- Most features show weak or no correlation.
- Significant correlations observed:
  - `unix_time` and `index`.
  - `city_pop` with `lat` and `long`.

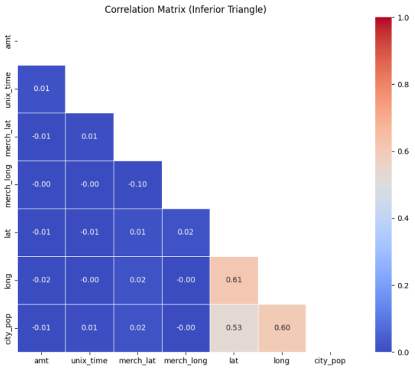


Figure: Correlation Matrix

Chi-Square Test Results

Key Insights:

- Tested independence among categorical variables.
- Significant dependencies found:
  - `gender` and `dob`.
  - `job` and `merchant`.
- Highlighted relationships guided feature engineering.
- Created new interaction variables (e.g., `job_age_group`).

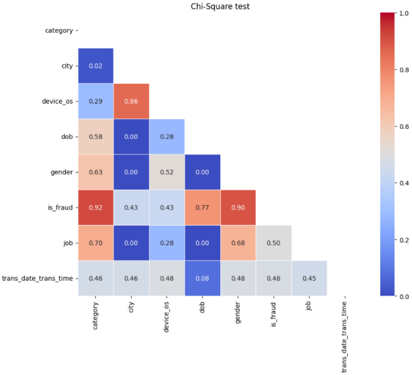


Figure: Chi-Square Test Results

Data Visualization

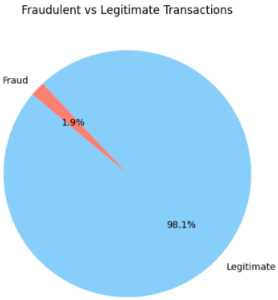


Figure: Fraudulent vs Legitimate Transactions

Figure: Transaction Amount Distribution

Data Visualization

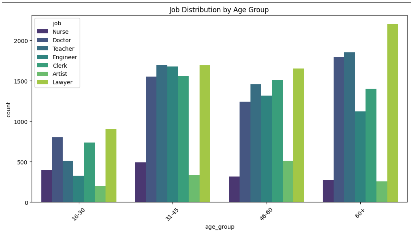


Figure: Job Distribution by Age Group

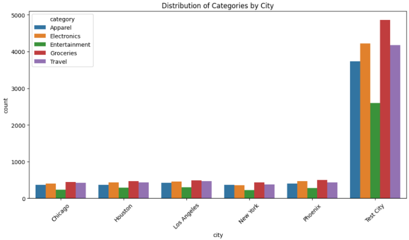


Figure: Distribution of Categories by City

Data Visualization

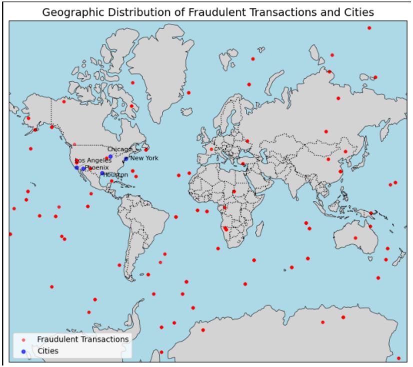


Figure: Geographic Distribution of Fraudulent Transactions

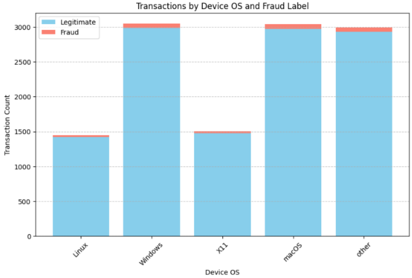


Figure: Transactions by Device OS and Fraud Label

Data Preparation

- **Dataset Split:**
  - Divided into 80% training and 20% testing datasets.
  - Used **stratification** to ensure the same proportion of fraud and non-fraud transactions in both sets.
  - Maintains the balance of the target variable (`is_fraud`) for better model performance and evaluation.
- **Handling Duplicates:**
  - Transaction number (`trans_num`) must be unique to maintain data integrity.
  - Identified duplicate transaction numbers and retained only the record with fewer missing values (NAs).

Feature Engineering

- **Age and Age Group Creation:**
  - Calculated age by subtracting the date of birth (`dob`) from the transaction date (`unix_time`).
  - Grouped age into categorical `age_group`:
    - Bins: *16–30*, *31–45*, *46–60*, *60+*.
- **Interaction Features:**
  - Combined `age_group` with `job` to create interaction variables (`job_age_group`).
- **Cyclical Temporal Features:**
  - Derived `hour_sin`, `hour_cos` for the hour of the day.
    Benefits:
    - Encodes the proximity of hours (e.g., 23:00 and 00:00 are close).
    - Improves model performance by providing meaningful temporal patterns.

Dropped Attributes

- During data preparation, several attributes were removed to improve model efficiency and focus on relevant features.
- **Reasons for Removal:**
  - **Redundancy:**
    - `index`, `trans_num`: Added no predictive value, purely identifiers.
    - `trans_date_trans_time`, `transaction_date`: Replaced by derived features like `hour_sin`, `hour_cos`
  - **Privacy Concerns:**
    - `first`, `last`, `street`, `state`, `dob`, `cc_num`: Contained personal or sensitive information.

Dropped Attributes

- **Low Predictive Value:**
  - `lat`, `long`, `merch_lat`, `merch_long`: Geographic data did not significantly correlate with fraud.
  - `zip`, `merchant`, `merchant_id`: Low variance or redundancy with other features.
- **Replaced by Derived Features:**
  - `age`: Replaced by `age_group`.

Handling Missing Values

- **Training Data:**
  - **Numeric Variables:**
    - Missing values imputed using the **k-Nearest Neighbors (kNN)** algorithm:
    - Numeric columns standardized before applying `kNN`.
    - Post-imputation, values re-scaled to their original distributions using stored `mean` and `standard deviation`.
  - **Categorical Variables:**
    - Handled automatically by `kNN`, using the **mode of the nearest neighbors**.
  - Final dataset saved as `X_train_without_missing_values.csv`.

Handling Missing Values

- **Testing Data:**
  - **Numeric Variables:**
    - Imputed using the **mean values** computed from the training dataset.
  - **Categorical Variables:**
    - Imputed using the **mode values** from the training dataset.

Ensured consistency by aligning imputed values with the training dataset.

One-Hot Encoding

- **Purpose:**
  - Convert categorical variables into a numerical format suitable for machine learning models.
- **Process:**
  - Identified categorical columns using `select_dtypes`.
  - Applied **one-hot encoding** to transform these columns:
    - Used `drop_first` for binary categories to avoid multicollinearity.
    - Retained all categories for non-binary columns.
  - Ensured the same encoding scheme was applied to both training and testing datasets.
- **Output:**
  - Each categorical column was replaced with multiple binary columns representing the categories.

Normalization and Scaling

- **Purpose:**
  - Normalize numerical variables to ensure all features have comparable scales.
  - Prevent variables with larger ranges from dominating the model.
- **Why MinMaxScaler?**
  - Chosen because the distribution of numerical variables is not uniform.
  - Scales values to a specified range, typically $[0, 1]$, preserving the original shape of the data distribution.

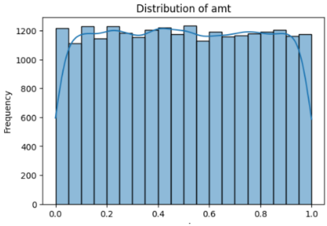


Figure: Distribution of `amt` attribute

Normalization and Scaling

- **Benefits of Scaling:**
  - Improves the performance of distance-based algorithms (e.g., k-Nearest Neighbors, clustering).
  - Ensures gradient-based optimization algorithms converge faster and more reliably.
  - Helps prevent bias in models sensitive to variable magnitude.

Clustering Overview

- **Objective:**
  - Group transactions based on their similarity using unsupervised learning.
- **Features Used for Clustering:**
  - `amt`: Transaction amount.
  - `hour_sin`, `hour_cos`: Temporal features representing the hour of the transaction.
  - `city_pop`: Population size of the city where the transaction occurred.
- **Clustering Techniques Explored:**
  - **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**.
  - **K-Means Clustering**.

DBSCAN Clustering

- **Approach:**
  - Identifies clusters based on density of points.
  - Parameters used:
    - `eps`: 0.5 (maximum distance between points in a cluster).
    - `min_samples`: 2 (minimum points to form a dense region).

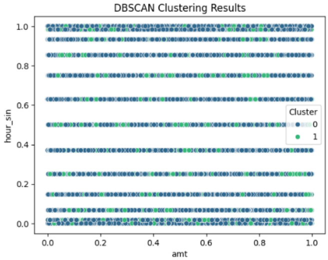


Figure: DBSCAN Clustering Results

K-Means Clustering

- **Approach:**
  - Groups data into `k` clusters by minimizing the within-cluster variance.
  - Tested different values of `k` (2 to 10) to find the best clustering structure.

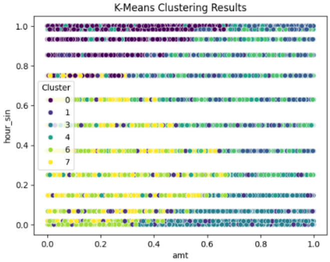


Figure: K-Means Clustering Results (Best `k`)

Handling Class Imbalance

- **Class Imbalance:**
  - The dataset exhibits significant class imbalance:
    - Majority class: `is_fraud = 0`.
    - Minority class: `is_fraud = 1`.
  - Imbalance can lead to biased models with poor recall for fraud detection.
- **Solution:**
  - Combined **SMOTE (Synthetic Minority Oversampling)** and **RandomUnderSampling**.
  - SMOTE increases minority class representation by generating synthetic samples.
  - RandomUnderSampling reduces majority class size, balancing the dataset and preventing computational overhead.

Why SMOTE + RandomUnderSampling?

- **Comparison with SMOTE-Tomek:**
  - SMOTE-Tomek removes Tomek links (overlapping samples between classes).
  - Analysis showed no significant class overlap, making SMOTE-Tomek less relevant.
- **Advantages of SMOTE + RandomUnderSampling:**
  - Simpler and faster than SMOTE-Tomek.
  - Balances the dataset without unnecessary removal of data points.
- **Oversampling and Undersampling Rates:**
  - Oversampling rates tested: `0.5, 0.7` (fractions of the majority class).
  - Undersampling rates tested: `0.8, 0.9` (fractions of the total dataset for the majority class).

Model Pipeline

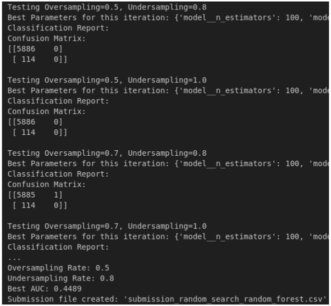
- **Pipeline Steps:**
  1. Apply **SMOTE** for oversampling the minority class.
  2. Apply **RandomUnderSampling** to balance the dataset.
  3. Train the model on the balanced dataset.
- **Evaluation Metrics:**
  - **Precision:** Accuracy of fraud predictions.
  - **Recall:** Ability to detect fraudulent transactions.
  - **F1-Score:** Balance between precision and recall.
  - **AUC-ROC:** Overall performance across classification thresholds.

Hyperparameter Tuning: Random Search

- **Objective:** Improve model performance by finding the best combination of hyperparameters.
- **Why Random Search?**
  - More efficient than Grid Search for large hyperparameter spaces.
  - Allows exploring a wide range of combinations with fewer iterations.
- **Implementation:**
  - Used `RandomizedSearchCV` with 5-fold cross-validation.
  - **What is 5-fold cross-validation?** A technique to evaluate model performance by splitting the data into 5 equally sized subsets, or "folds."
  - Evaluated models based on the **AUC-ROC** score to handle class imbalance effectively.

Models Overview

- **Random Forest:**
  - Ensemble-based model combining multiple decision trees.
  - Effective for imbalanced datasets and interpretable results.



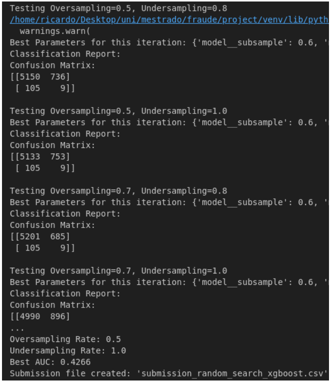
```
Testing Oversampling=0.5, Undersampling=0.8
Best Parameters for this iteration: {'model__n_estimators': 100, 'mode
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__n_estimators': 100, 'mode
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__n_estimators': 100, 'mode
Classification Report:
Confusion Matrix:
[[5885    1]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=1.0
Best Parameters for this iteration: {'model__n_estimators': 100, 'mode
Classification Report:
...
Oversampling Rate: 0.5
Undersampling Rate: 0.8
Best AUC: 0.4489
Submission file created: 'submission_random_search_random_forest.csv'
```

Figure: Random Forest Results Overview

Models Overview

XGBoost:

- Gradient boosting framework optimized for speed and performance.
- Excellent at capturing complex, non-linear patterns in data.



```
Testing Oversampling=0.5, Undersampling=0.8
/home/ricardo/Desktop/uni/mestrado/fraude/project/venv/lib/pyth
  warnings.warn(
Best Parameters for this iteration: {'model__subsample': 0.6, '
Classification Report:
Confusion Matrix:
[[5150  736]
 [ 105    9]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__subsample': 0.6, '
Classification Report:
Confusion Matrix:
[[5133  753]
 [ 105    9]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__subsample': 0.6, '
Classification Report:
Confusion Matrix:
[[5201  685]
 [ 105    9]]

Testing Oversampling=0.7, Undersampling=1.0
Best Parameters for this iteration: {'model__subsample': 0.6, '
Classification Report:
Confusion Matrix:
[[4990  896]
...
Oversampling Rate: 0.5
Undersampling Rate: 1.0
Best AUC: 0.4266
Submission file created: 'submission_random_search_xgboost.csv'
```

Figure: XGBoost Results Overview

Models Overview

Decision Tree (Best AUC-ROC Score):

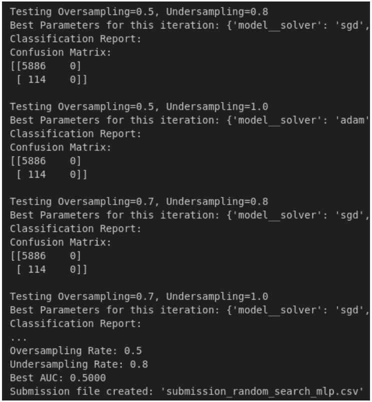
- Simple and interpretable tree-based model.
- Tends to overfit but works well with proper pruning and parameter tuning.

```
Testing Oversampling=0.5, Undersampling=0.8
Best Parameters for this iteration: {'model__min_samples_split': 10,
Classification Report:
Confusion Matrix:
[[5282  604]
 [  97   17]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__min_samples_split': 20,
Classification Report:
Confusion Matrix:
[[3691 2195]
 [  60   54]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__min_samples_split': 20,
Classification Report:
Confusion Matrix:
[[4537 1349]
 [  85   29]]

Testing Oversampling=0.7, Undersampling=1.0
Best Parameters for this iteration: {'model__min_samples_split': 10,
Classification Report:
...
Oversampling Rate: 0.5
Undersampling Rate: 1.0
Best AUC: 0.5333
Submission file created: 'submission_decision_tree.csv'
```

Figure: Decision Tree Results Overview

Models Overview

- **Multi-Layer Perceptron (MLP):**
  - Neural network model with hidden layers.
  - Effective for capturing non-linear relationships in data.



```
Testing Oversampling=0.5, Undersampling=0.8
Best Parameters for this iteration: {'model__solver': 'sgd',
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.5, Undersampling=1.0
Best Parameters for this iteration: {'model__solver': 'adam'
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=0.8
Best Parameters for this iteration: {'model__solver': 'sgd',
Classification Report:
Confusion Matrix:
[[5886    0]
 [ 114    0]]

Testing Oversampling=0.7, Undersampling=1.0
Best Parameters for this iteration: {'model__solver': 'sgd',
Classification Report:
...
Oversampling Rate: 0.5
Undersampling Rate: 0.8
Best AUC: 0.5000
Submission file created: 'submission_random_search_mlp.csv'
```

Figure: MLP Results Overview

Models Overview

Support Vector Machine (SVM):

- Separates data using hyperplanes in high-dimensional space.
- Effective for smaller datasets and well-separated classes.

```
Testing Oversampling=0.5, Undersampling=0.8
Classification Report:
Confusion Matrix:
[[   0 5886]
 [   0  114]]

Testing Oversampling=0.5, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[   0 5886]
 [   0  114]]

Testing Oversampling=0.7, Undersampling=0.8
Classification Report:
Confusion Matrix:
[[   0 5886]
 [   0  114]]

Testing Oversampling=0.7, Undersampling=1.0
Classification Report:
Confusion Matrix:
[[   0 5886]
 [   0  114]]
...
Oversampling Rate: 0.5
Undersampling Rate: 0.8
Best AUC: 0.5000
Submission file created: 'submission_svm.csv'
```

Figure: SVM Results Overview

Conclusion

- The **Decision Tree** performed the best, but with modest results.
- Valuable insights were gained by following systematic steps.
- The approach shows potential for better outcomes with real-world data.

Thank you for your attention!