Credit Card Fraud Detection Model – Chervin Daniel

Credit Card Fraud Detection Project Overview

Project Overview: Understanding the Basics and Goals

The goal of this project is to build a robust machine learning model capable of detecting fraudulent credit card transactions. Credit card fraud is a growing concern in today's increasingly digital financial landscape, leading to significant financial losses for both consumers and financial institutions.

This project focuses on analyzing a real-world dataset of credit card transactions in the US to uncover patterns indicative of fraudulent activity. Through:

- 📊 Exploratory Data Analysis
- 🛠️ Feature Engineering
- 🤖 Machine Learning Modeling

● What are we trying to find out?

We aim to identify whether a transaction is fraudulent based on transaction metadata and user behavior.

● What do we already know?

We know that fraud is rare compared to legitimate transactions, approx 0.5% of credit card transactions are fraudulent making this a highly imbalanced classification challenge. Fraudulent behavior often exhibits patterns like abnormal transaction times, high-value amounts, geographic anomalies and combinations between them e.g. a senior customer is less likely to perform transactions at certain amounts and times of the day.

● What are we aiming to achieve?

Our goal is to accurately detect fraudulent transactions while minimizing false positives to avoid disrupting genuine users.

● What factors affect our results?

Class imbalance, feature quality, model selection, model parameters, and evaluation metrics all significantly influence the final model's effectiveness.

● Is there something new we can use?

Yes. We explore advanced ensemble models like XGBoost, feature interaction engineering, and resampling strategies (SMOTE, SMOTETomek) to counter imbalance for the end goal os enhancing prediction accuracy and generalization.

Credit Card Fraud Detection Model – Chervin Daniel

Project Stages Summary

Data Preparation Overview

The notebook, titled "01_ds18_ml-proj_data-prep.ipynb," details the data preparation phase for building a credit card fraud detection system using a large dataset of US credit card transactions, aiming to the data for machine learning modeling through exploratory analysis and feature engineering.

Data Loading and Initial Setup

The full dataset contains 34,636,378 rows, loaded efficiently using Dask to handle its size.

Sample Validation and Analysis

A 300,000-row sample is created, with a fraud rate of approximately 0.496%. Monthly fraud percentages are analyzed for both the full dataset and the sample, showing similar trends, especially for 2020. A Chi-Square test with a p-value of 0.655 confirms the sample's distribution aligns with the full dataset

Feature Engineering and Cleaning

The process involves handling missing values and converting categorical variables, such as gender to binary (1 for male, 0 for female), and grouping job titles, states into regions, and profiles into urban/rural categories. Engineered features include temporal (e.g., month, hour), geographical (e.g., distance between locations), demographic (e.g., age groups), and transactional (e.g., time since last transaction) elements, enhancing the dataset for fraud detection.

Conclusion

The prepared dataset, confirmed representative, is saved as a pickle file for future model training, ensuring it captures essential fraud patterns while maintaining data integrity.

Detailed Survey Note: Comprehensive Analysis of Data Preparation for Credit Card Fraud Detection

This note provides an in-depth examination of the data preparation process outlined in the Jupyter notebook **"01_ds18_ml-proj_data-prep.ipynb,"** focusing on its role in a credit card fraud detection project using a real-world dataset of US credit card transactions. The objective is to build a robust fraud detection system through exploratory data analysis, feature engineering, and machine learning modeling, ensuring the dataset is representative and enriched for accurate analysis.

Introduction and Context

The notebook serves as a critical step in preparing data for detecting fraudulent activities in credit card transactions. Given the scale and complexity of the dataset, the process involves efficient data handling, validation of a subset for analysis, and the creation of meaningful features to capture fraud patterns. This preparation is essential for subsequent machine learning phases, ensuring the data reflects real-world scenarios accurately.

Data Loading and Initial Setup

The dataset, comprising **34,636,378** rows, is loaded using Dask, a library designed for handling large datasets efficiently. A key initial step is combining the **trans_date** and **trans_time** columns into a single **Datetime** column, facilitating temporal analysis. This setup is crucial for later steps, such as monthly fraud analysis and time-based feature engineering.

Monthly Fraud Analysis

To ensure the dataset's usability, a function is defined to calculate monthly fraud percentages for both the full dataset and a 300,000-row sample. This analysis compares distributions, aiming to validate the sample's representativeness. For instance, the full dataset's fraud rate is approximately 0.548%, calculated from 189,915 fraudulent transactions out of an estimated total, while the sample's fraud rate is about 0.496%, based on 1,489

fraudulent transactions out of 300,000 rows. This step is vital for ensuring the sample can reliably represent the larger dataset for further analysis.

Sample Validation

Validation is conducted both visually and statistically to confirm the 300,000-row sample, particularly for 2020, mirrors the full dataset. Bar plots of monthly fraud percentages show similar trends, with higher rates in early months (e.g., January 2020 at 0.817% for full, 0.636% for sample) and lower rates towards year-end (e.g., December 2020 at 0.239% for full, 0.218% for sample). A **Chi-Square test**, yielding a p-value of **0.655**, statistically confirms the sample's distribution is consistent with the full dataset, providing confidence in its representativeness for modeling.

Focused 2020 Sample

The notebook extracts and analyzes the 2020 subset, ensuring its sample aligns with the full dataset. Similar visual and statistical methods, including plotting and testing, confirm consistency, reinforcing the reliability of the subset for focused analysis on recent data, which may capture evolving fraud patterns.

Data Cleaning and Feature Engineering

Data cleaning begins with checking and handling missing values, ensuring dataset completeness. Categorical variables are transformed for analysis:

- Gender is converted to binary (1 for male, 0 for female).
- Job titles and states are narrowed into regions (e.g., Northeast, Midwest, South, West).
- Profiles are categorized into urban or rural based on keywords.

*** Disclaimer: few features were engineered to assist later in EDA*

Feature engineering enriches the dataset with the following features, each designed to capture relevant fraud indicators:

Feature	Description
Datetime	Combines transaction date and time for temporal analysis.
Month	Extracted from Datetime, used for monthly fraud pattern analysis.
Region	Groups states into Northeast, Midwest, South, West for geographic analysis.
Trans_day_of_week	Indicates day of week (0-6), useful for weekday vs. weekend fraud patterns.
Is_weekend	Binary (0 or 1), marks weekend transactions for potential fraud correlation.
Trans_hour	Hour of day (0-23), aids in time-based fraud detection.
Trans_time_segment	Groups hours into segments (e.g., late_night_0-6, morning_6-12) for analysis.
Age	Calculated as transaction year minus birth year, reflects cardholder age.
Age_group	Categorizes age into groups (child, teenager, etc.) for demographic analysis.
Cc_type	Identifies credit card type (e.g., Visa, MasterCard) from card number.
Area_cat	Binary, categorizes profile as urban or rural for geographic insights.
Distance	Measures distance (km) between cardholder and merchant locations, flags unusual transactions.
Time_since_last_trans	Calculates hours since last transaction for the same account, detects frequency anomalies.

These features enhance the dataset's ability to identify fraudulent patterns, covering temporal, geographical, demographic, and transactional dimensions.

Conclusion and Next Steps

The validation process confirms the 300,000-row sample is representative, setting the stage for further analysis. The prepared dataset, with all engineered features, is saved as a pickle file, ensuring it is ready for future model training. This step is crucial, as it maintains data integrity and captures essential patterns for detecting fraudulent activities, aligning with the project's goal of building a robust fraud detection system.

Unexpected Detail: Extensive Feature Engineering

An unexpected aspect is the breadth of feature engineering, with 13 distinct features created, ranging from temporal segments to geographical distances. This level of detail, such as categorizing time into segments like "late_night_0-6," goes beyond basic cleaning, offering nuanced insights for fraud detection that may not be immediately apparent.

Credit Card Fraud Detection Model – Chervin Daniel

EDA – Exploratory Data Analysis

Introduction

The attached notebook, titled **"02_ds18_ml-proj_eda.ipynb,"** is an exploratory data analysis (EDA) for a credit card transaction dataset aimed at detecting fraud.

Dataset and Preprocessing

The dataset contains 299,996 transactions with 41 features, including transaction ID, SSN, credit card number, transaction amount, location data (latitude, longitude), date/time, and the target variable is_fraud. Notably, is_fraud is highly imbalanced, with only 0.5% of transactions being fraudulent. Preprocessing steps included dropping irrelevant features like ID and SSN, and creating new features such as transaction day of week, age group, and distance between locations. Log transformations were applied to skewed numerical features like transaction amount to normalize distributions.

Analysis and Findings

The EDA explored both numerical and categorical features. Numerical analysis showed transaction amount is right-skewed, with fraudulent transactions often involving larger amounts. Categorical analysis highlighted that certain transaction categories and time segments are more prone to fraud. Statistical tests like Chi-Square and T-tests identified significant predictors. Feature importance analysis using Random Forest ranked transaction category, log-transformed amount, and transaction hour as top predictors.

Recommendations

For next steps, consider addressing the class imbalance using techniques like SMOTE, and proceed with modeling using the identified important features. Further feature engineering, such as exploring interaction terms, could enhance model performance.

Dataset Description

The dataset, loaded from a pickle file (**sample_300k_2020_prepared.pkl**), contains 299,996 entries with 41 columns. Key features include:

- Transaction details: ID, SSN, credit card number, transaction amount (amt), and category.
- Location data: Latitude, longitude, and derived distance between locations.
- Temporal data: Date/time, with derived features like transaction hour and day of week.
- Target variable: is_fraud, indicating whether a transaction is fraudulent.

A significant finding is the imbalance in the target variable, with only 1,574 fraudulent transactions out of 299,996, representing approximately 0.5%. This imbalance poses a challenge for modeling and was noted throughout the analysis.

Data Preprocessing and Cleaning

Initial preprocessing steps included:

- **Feature Dropping:** Irrelevant features such as 'ssn', 'id', 'first', 'last', and 'street' were removed, reducing the dataset to 34 columns for analysis.
- **Feature Engineering:** New features were created to enhance predictive power:
    - Temporal features: trans_day_of_week, is_weekend, trans_hour, trans_time_segment.
    - Demographic features: age, age_group derived from customer dates of birth.
    - Categorization: cc_type, area_cat for credit card type and area categorization.
    - Spatial features: distance calculated using geopy, representing the distance between transaction locations.

    - Temporal gaps: time_since_last_trans, tracking time since the last transaction by the same customer.
- **Handling Skewness:** Numerical features like amt, time_since_last_trans, and city_pop were found to be right-skewed. Log transformations (log1p) were applied to create log_amt, log_time_since_last_trans, and log_city_pop to normalize distributions, improving suitability for modeling.

Data quality was assessed, with no missing values reported, and basic statistics were exported to Excel files for further review, including data types, maximum/minimum values, and unique value counts.

Exploratory Data Analysis (EDA)

The EDA was divided into numerical and categorical feature analyses, supported by visualizations and statistical tests.

Numerical Features Analysis

- **Distribution Analysis:** Histograms and box plots were generated for numerical features like amt, distance, and time_since_last_trans. The transaction amount (amt) showed high right-skewness, with fraudulent transactions often involving larger amounts, suggesting potential outliers.
- **Statistical Tests:** T-tests were conducted to compare means between fraud and non-fraud transactions for features like amt, trans_hour, and age. Significant differences ($p < 0.05$) were found, indicating these features are predictive of fraud.
- **Correlation Analysis:** A correlation matrix was generated, revealing:
    - Positive correlations with is_fraud: amt (0.160), log_amt (0.118), and age (0.020).
    - Negative correlations: trans_hour (-0.054), is_weekend (-0.012), suggesting fraud is less common on weekends or at certain hours.
    - Weak correlations for city_pop and distance, indicating limited predictive power.

Categorical Features Analysis

- **Chi-Square Tests:** Performed to evaluate the association between categorical features and is_fraud. Features with p-values < 0.05, such as category, trans_time_segment, and age_group, were statistically significant, indicating strong relationships with fraud.
- **Visualizations:** Bar plots were created to visualize fraud rates across categories, highlighting:
    - Certain transaction categories (e.g., entertainment, grocery_pos) had higher fraud rates.
    - Specific time segments (e.g., late night) showed increased fraud incidence.
    - Age groups, particularly younger and older demographics, were more associated with fraud.

Automated Visualization

The AutoViz library was used to generate automated visualizations, providing insights into data distribution, correlations, and anomalies. It identified outliers in city_pop and amt, and high cardinality in job, suggesting potential grouping or binning for modeling.

Feature Importance and Modeling Insights

A Random Forest model was employed to assess feature importance, ranking the following as top predictors, category (0.238646) , log_amt (0.127613) ,amt (0.122935) ,unix_time (0.119496) ,trans_hour (0.089957) .

This analysis underscores the importance of transaction category, amount (both raw and log-transformed), and timing in predicting fraud, providing a foundation for feature selection in subsequent modeling stages.

Key Findings and Insights

- **Class Imbalance:** The dataset's imbalance (0.5% fraud rate) necessitates techniques like SMOTE for effective modeling.

- **Significant Predictors:** Transaction category, amount, and time-related features (hour, segment) are critical for fraud detection.
- **Skewness and Transformations:** Log transformations improved the distribution of skewed features, enhancing model suitability.
- **Visualization Challenges:** Some visualizations, particularly time series plots, faced rendering issues, but overall, they provided valuable insights into fraud patterns.

Recommendations for Next Steps

Based on the analysis, the following steps are recommended:

1. **Address Class Imbalance:** Implement SMOTE or similar techniques to balance the dataset for modeling.
2. **Feature Selection and Engineering:** Focus on significant features (category, log_amt, trans_hour) and explore interaction terms, such as combining time and category, to improve predictive power.
3. **Modeling Approach:** Proceed with tree-based models (e.g., Random Forest, Gradient Boosting) that handle categorical variables well, and consider encoding techniques for high-cardinality features.
4. **Data Saving and Documentation:** The processed dataset was saved as fname_eda.pkl, ensuring availability for further analysis. Document findings for reproducibility and stakeholder communication.

Credit Card Fraud Detection Model – Chervin Daniel

Data Cleansing

The notebook **"03_ds18_ml-proj_data-cleanse.ipynb,"** identifies outliers in numerical features using boxplots and decides to retain them due to their importance in detecting fraud. Log transformations are applied to manage skewness in features like transaction amount and city population, ensuring the data is normalized for modeling.

Dataset Description

The dataset, sourced from a pickle file named **'sample_300k_2020_eda.pkl,'** comprises 299,996 transactions, each described by 37 features. These features encompass a wide range of data, including:

- **Identification and Transaction Details:** Credit card number (cc_num), account number (acct_num), transaction number (trans_num), transaction date and time (trans_date, trans_time, unix_time), category (category), amount (amt), and merchant details (merchant, merch_lat, merch_long).
- **Demographic and Location Data:** Gender (gender), city, state, zip code, latitude and longitude (lat, long), city population (city_pop), job (job), date of birth (dob), age (age), and age group (age_group).
- **Derived and Categorical Features:** Profile (profile), credit card type (cc_type), area category (area_cat), job category (job_cat), and indicators like whether the transaction occurred on a weekend (is_weekend), day of the week (trans_day_of_week), hour (trans_hour), and time segment (trans_time_segment).
- **Fraud Indicator:** A binary label (is_fraud) indicating whether the transaction is fraudulent, with a mean of 0.005, suggesting only 0.5% of transactions are fraudulent.

Initial exploration revealed no missing values across all columns, a critical finding that eliminates the need for imputation and simplifies data preparation. The dataset's structure includes a mix of data types: integers, floats, objects (strings), and datetime, providing a comprehensive view of transaction behaviors.

Data Cleansing Process: Detailed Steps

The cleansing process involved several key steps to ensure the dataset is suitable for fraud detection modeling:

1. **Missing Data Analysis:**
    - The notebook employed summary statistics and a heatmap to confirm the absence of missing values. This step is crucial as it ensures data completeness, avoiding potential biases in subsequent analyses.
2. **Outlier Detection and Retention:**
    - Outliers were identified in several numerical features using boxplots, providing a visual representation of data distribution. The features with notable outliers include:
        - amt: Transaction amount, with a range from 1 to 24,006.93, showing significant variability.
        - city_pop: City population, ranging from 17,715 to 2,906,700, indicating potential extreme values.
        - distance: Distance related to the transaction, ranging from 0.06 to 148.46, with outliers possibly reflecting unusual transaction locations.
        - time_since_last_trans: Time since the last transaction, ranging from approximately 77.87 to 7,923.05, suggesting irregular transaction patterns.
    - Despite these outliers, the decision was made to retain them, as they are potentially indicative of fraudulent active=ities, which often involve unusual or extreme transaction behaviors.

3. **Handling Skewness through Log Transformations:**
    - To address skewness in the distribution of numerical features with outliers, log transformations were applied to the following:
        - amt → log_amt: Logarithm of transaction amount to normalize extreme values.
        - city_pop → log_city_pop: Logarithm of city population to manage large variations.
        - time_since_last_trans → log_time_since_last_trans: Logarithm of time since the last transaction to reduce skewness.
4. **Saving the Cleaned Dataset:**
    - Saving the cleaned and transformed dataset, incorporating the log-transformed features, for use in subsequent stages such as feature engineering, model training, and evaluation. This ensures the data is ready for advanced analytics and machine learning modeling.

Implications and Observations

The **rarity of fraud cases** (0.5% of transactions) is a significant observation, aligning with typical fraud detection datasets where positive cases are scarce. This imbalance necessitates careful model selection and potentially the use of techniques like oversampling or class weighting during training to address the class imbalance.

The **retention of outliers is a strategic choice**, reflecting the understanding that fraudulent transactions often deviate significantly from normal patterns. The application of log transformations to key numerical features enhances model performance by normalizing data, which is particularly beneficial for algorithms sensitive to scale and distribution, such as logistic regression or neural networks.

Conclusion and Future Directions

In conclusion, the notebook prepares the dataset for fraud detection by ensuring data completeness, managing outliers through retention, and applying log transformations to handle skewness. This foundational work is essential for building accurate and efficient machine learning models, particularly given the rarity of fraud cases and the importance of outliers in identifying suspicious activities.

Table: Summary of Numerical Features with Outliers and Transformations

Feature	Range/Description	Outliers Present	Log-Transformed
amt	1 to 24,006.93	Yes	Yes (log_amt)
city_pop	17,715 to 2,906,700	Yes	Yes (log_city_pop)
distance	0.06 to 148.46	Yes	No
time_since_last_trans	~77.87 to 7,923.05	Yes	Yes (log_time_since_last_trans)

This table summarizes the handling of numerical features, highlighting the decision-making process regarding outliers and transformations.

Credit Card Fraud Detection Model – Chervin Daniel

Feature Engineering

Introduction

This summary covers the notebook **"04_ds18_ml-proj_feat-enginering.ipynb,"** which is part of a machine learning project aimed at detecting fraudulent credit card transactions. The notebook's main goal is to prepare the data through feature engineering, making it ready for modeling.

Data Preparation and Exploration

The dataset includes transaction details like credit card numbers, gender, location coordinates, and transaction amounts, with a fraud indicator. A sample of 15,000 rows is taken from the full dataset of 299,996 entries and 37 columns for analysis, ensuring efficiency and manageability.

Feature Engineering and Visualization

The notebook creates new features to enhance fraud detection:

- **Time-based features** include the day of the week, whether it's a weekend, and the transaction hour, helping identify temporal patterns.
- **Demographic features** like age groups are derived from birth dates, exploring if certain groups are more prone to fraud.
- **Categorical features** such as credit card type and area categories based on population density add context.
- **GeoSpatial features** calculate the distance between user and merchant locations, crucial for spotting unusual transactions.
- **Interaction features** combine these elements, though high-cardinality ones are dropped to prevent overfitting.
- Use of Folium for an interactive map, visualizing user and merchant locations with colored markers and lines, making it easier to spot long-distance transactions that might indicate fraud.

Feature Pruning

To optimize for modeling, irrelevant columns like identifiers, raw dates, and high-cardinality categories are removed, leaving a streamlined dataset with 20 engineered features.

Detailed Analysis of the Feature Engineering for Fraud Detection

Data Preparation and Initial Exploration

The dataset, initially loaded from a pickle file named **sample_300k_2020_cleansed.pkl**, encompasses a wide array of transaction details. Key features include credit card number (cc_num), gender, location coordinates (latitude lat, longitude long, merchant latitude merch_lat, merchant longitude merch_long), transaction amount (amt), and a binary fraud indicator (is_fraud), among others. The dataset comprises 299,996 entries and 37 columns, with data types ranging from integers and floats to objects and datetime formats.

Feature Engineering Process

Feature engineering is the cornerstone of the notebook, involving the creation of new variables from existing data to capture meaningful patterns relevant to fraud detection. The process is systematically organized into several categories:

Time-based Features

- **trans_day_of_week**: Derived from the transaction date, this feature captures the day of the week, enabling analysis of temporal patterns that might correlate with fraud, such as higher fraud rates on weekends.

- **is_weekend**: A boolean feature indicating whether the transaction occurred on a weekend, which can be significant given different user behaviors during these periods.
- **trans_hour**: Extracted from the transaction time, this feature identifies the hour of the day, helping to flag transactions at unusual hours that might suggest fraudulent activity.

These time-based features are vital for understanding the temporal dynamics of transactions, which can reveal patterns such as fraud being more prevalent during off-hours or specific days.

Demographic Features

- **age_group**: Calculated from the date of birth (dob), this feature categorizes users into age groups. This is important for analyzing whether certain age demographics are more susceptible to fraud, potentially due to differences in digital literacy or transaction habits.

Categorical Features

- **cc_type**: Categorizes the type of credit card, which might influence fraud rates due to varying security features or usage patterns across different card types.
- **area_cat**: Derived from city population data, this feature categorizes areas based on population density, providing context that might correlate with transaction behaviors, such as higher fraud in densely populated urban areas.

Geospatial Features

- **distance**: Calculated using the latitude and longitude of both user and merchant locations, this feature measures the geographic distance between the two. It is particularly significant for fraud detection, as transactions involving large distances (e.g., a card used far from the user's typical location) can indicate card-not-present fraud or stolen credentials. This feature is visualized later in the notebook, enhancing its interpretability.

Interaction Features

To capture complex relationships, the notebook creates interaction features, such as:

- Combinations of temporal features (e.g., trans_hour × is_weekend, category × trans_day_of_week).
- Transactional interactions (e.g., amt × distance, log_amt × log_city_pop).
- GeoSpatial interactions (e.g., distance × is_weekend, area_cat × distance).
- Demographic interactions (e.g., age_group × category, job_cat × trans_hour).

However, to prevent overfitting, high-cardinality interaction features—those with many unique values—are identified and dropped, ensuring the model remains generalizable.

Additional Transformations

The notebook also includes log-transformations for skewed features:

- **log_amt**: Log-transformed transaction amount to handle skewness and improve model performance.
- **log_city_pop**: Log-transformed city population for similar reasons.
- **log_time_since_last_trans**: Log-transformed time since the last transaction, aiding in handling skewed temporal data.

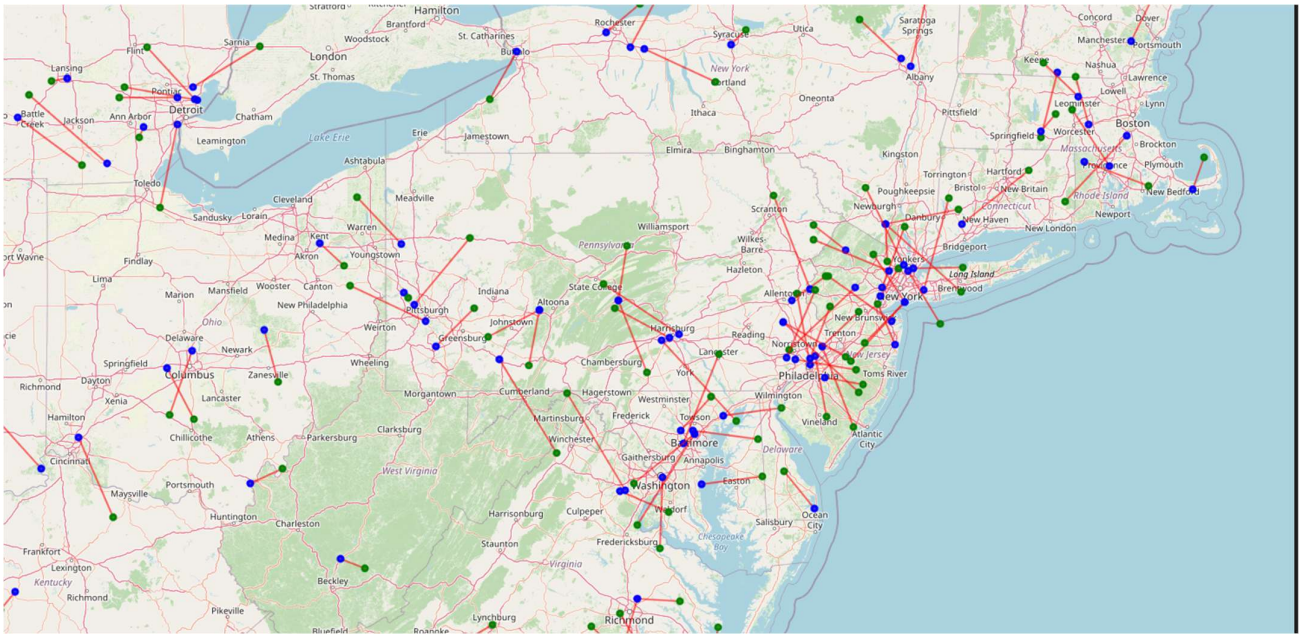
These transformations are crucial for normalizing data distributions, making them more suitable for machine learning algorithms.

Visualization and GeSpatial Analysis

A notable aspect of the notebook is its use of the Folium library for Geospatial visualization. An interactive map (fraud_map) is created, centered on the United States (around 37.76 latitude, -92.50 longitude) with a zoom level.

Credit Card Fraud Detection Model – Chervin Daniel

Fraudulent transactions



This map plots:

- User locations with blue circle markers.
- Merchant locations with green circle markers.
- Transaction paths connecting user and merchant locations with red lines, highlighting the geographic relationship.

This visualization is particularly insightful for fraud detection, as it reveals patterns such as long-distance transactions, which are often indicative of fraud. For instance, a transaction where the user is in California and the merchant is in New York might suggest card-not-present fraud or stolen credentials. The map's interactive nature allows for deeper exploration, such as zooming into specific regions to analyze clusters of fraudulent activities.

Feature Pruning and Final Dataset

To streamline the dataset for modeling, the notebook performs feature pruning, removing columns that are either irrelevant or likely to introduce noise:

- **Identifiers**: Columns like cc_num, acct_num, trans_num, and profile, which are unique identifiers and not predictive of fraud.
- **Raw date/time fields**: trans_date, trans_time, and dob, as their derived features (e.g., trans_hour, age_group) are already included.
- **Redundant raw features**: amt, city_pop, time_since_last_trans, as their transformed versions (e.g., log_amt) are used instead.
- **High-cardinality object columns**: merchant, job, city, which have many unique values and can lead to overfitting.

After pruning, the final dataset, df_feat_eng, contains 20 columns, focusing on engineered features such as gender, unix_time, category, is_fraud, region, trans_day_of_week, is_weekend, trans_hour, age_group, cc_type, area_cat, distance, job_cat, and various interaction features. This pruned dataset is saved as a pickle file (df_feat_eng_pruned.pkl) for use in subsequent modeling stages.

Credit Card Fraud Detection Model – Chervin Daniel

Key Insights and Implications

The notebook underscores the importance of geSpatial and temporal analysis in fraud detection. The distance feature emerges as a critical indicator, with visualizations revealing that fraud often involves transactions spanning large geographic distances. This insight suggests potential model enhancements, such as including features like distance_km or is_local_transaction to capture such anomalies.

Time-based features also play a significant role, with patterns such as transactions at unusual hours or on weekends potentially flagging fraud. The demographic and categorical features add layers of context, enabling the model to account for variations across different user groups and transaction types.

The feature pruning step is crucial for reducing dimensionality, which can improve model training time and accuracy by focusing on the most relevant predictors. This process ensures that the dataset is not only clean but also optimized for machine learning, setting the stage for effective fraud detection.

Conclusion

In summary, the notebook "04_ds18_ml-proj_feat-enginering.ipynb" provides a comprehensive workflow for preparing and analyzing a transaction dataset for fraud detection. It emphasizes geSpatial and temporal feature engineering, supported by visualizations that highlight key patterns. The final pruned dataset, with 20 engineered features, is well-suited for building machine learning models to predict is_fraud, offering a solid foundation for accurate and efficient fraud detection systems.

Derived Features and Their Purposes

The following table lists the derived features and their purposes, providing a clear overview of the feature engineering process:

Derived Feature	Purpose
trans_day_of_week	To capture the day of the week for transaction timing analysis.
is_weekend	To identify if the transaction occurred on a weekend, which might have different fraud patterns.
trans_hour	To capture the hour of the transaction, as certain hours might be more prone to fraud.
age_group	To categorize age into groups, which can help in understanding if certain age groups are more susceptible to fraud.
cc_type	To categorize credit card types, which might have different fraud rates.
area_cat	To categorize areas based on population density, which might correlate with transaction behaviors.
distance	To calculate the distance between user and merchant locations, helping to identify unusual transaction locations that could indicate fraud.
log_amt	Log-transformed amount to handle skewed data and improve model performance.
log_city_pop	Log-transformed city population for similar reasons as above.
log_time_since_last_trans	Log-transformed time since the last transaction to handle skewed data.
category_x_trans_day_of_week	Interaction feature to capture combined effects of different features, enhancing the model's ability to detect complex fraud patterns.

Credit Card Fraud Detection Model – Chervin Daniel

Imbalanced Data Handling

Applied ROS, RUS, SMOTE, and SMOTETomek to improve recall while maintaining precision.

Introduction

This summary covers the notebook **"05_ds18_ml-proj_feat-selection_imbalance.ipynb,"** part of a machine learning project aimed at detecting fraudulent credit card transactions. It focuses on feature selection to identify the most valuable predictors and methods to handle the dataset's class imbalance, where fraudulent transactions are rare.

Feature Selection

The notebook uses five machine learning models—**Lasso**, **Linear SVC**, **Gradient Boosting, Random Forest**, and **Ridge**—to rank features based on their importance. String features like gender and region were encoded numerically using LabelEncoder for analysis. Key findings include:

- Features selected by all five models, such as category, log_amt (log-transformed transaction amount), unix_time, trans_hour, and interaction terms like category_x_trans_day_of_week, are likely the most predictive of fraud.
- Features selected by four models, including cc_type, distance, and log_city_pop, are also strongly important.
- The recommendation is to focus on features with a consensus of four or more models for the final model, enhancing performance and reducing overfitting.

Handling Class Imbalance

With only 0.5% of transactions being fraudulent, the notebook evaluates several techniques to address this imbalance:

- **Random Oversampling (ROS)** duplicates minority class samples.
- **Random Undersampling (RUS)** removes majority class samples.
- **SMOTE** generates synthetic minority class samples.
- **SMOTETomek** combines SMOTE with Tomek Links for cleaning.

A Random Forest Classifier was used to evaluate performance, with metrics including precision (minimizing false positives) and recall (catching actual frauds). The results show:

- Without balancing, the model achieves 99.47% precision and 84.62% recall.
- ROS achieves 100.00% precision and 84.16% recall, maintaining high accuracy.
- RUS has 18.67% precision but 95.93% recall, indicating many false positives.
- SMOTE offers a balanced performance with 92.22% precision and 88.46% recall, while SMOTETomek is similar at 90.93% precision and 88.46% recall.

SMOTE or ROS are recommended for production, with a focus on monitoring both metrics to balance false positives and missed frauds.

Credit Card Fraud Detection Model – Chervin Daniel

Feature selection and imbalance

Introduction

The attached notebook, **"05_ds18_ml-proj_feat-selection_imbalance.ipynb,"** aims to identify the most relevant features for predicting fraud and address the challenge of class imbalance, where fraudulent transactions are rare compared to legitimate ones.

Feature Selection and Methods

The notebook employs several model-based techniques to select important features, including Lasso, Ridge, Linear SVM, Gradient Boosting, and Random Forest. These methods rank features based on their predictive power, with features like category, log_amt, unix_time, and trans_hour consistently selected across models. Features chosen by at least four out of five models are recommended for focus, ensuring robust fraud detection.

Handling Class Imbalance

Given the imbalance, the notebook tests four resampling techniques: Random Oversampling (ROS), Random Undersampling (RUS), SMOTE, and SMOTETomek. These methods balance the dataset to improve model performance. SMOTE and ROS emerged as top performers, with SMOTE preferred for production to avoid overfitting, achieving a good balance of precision and recall.

Model Evaluation and Performance

The Random Forest Classifier is the primary model, evaluated using metrics like accuracy, precision, recall, and F1-score. Without balancing, it shows 99.92% accuracy and 99.47% precision but lower recall (84.62%). After applying balancing techniques, performance varies, with SMOTE improving recall to 88.46% while maintaining high precision (92.22%).

Detailed Analysis and Recommendations

The notebook, "05_ds18_ml-proj_feat-selection_imbalance.ipynb," provides a comprehensive analysis of fraud detection, focusing on feature selection and class imbalance handling. Below is a detailed breakdown, suitable for a professional 1-2 page Word document, with formatting suggestions for clarity and visual appeal.

Introduction and Context

The notebook addresses the critical problem of fraud detection in financial transactions, aiming to enhance model accuracy by selecting relevant features and mitigating class imbalance. The dataset, 'sample_300k_2020', loaded from a pickle file, contains approximately 300,000 transaction records with 20 features, including gender, unix_time, category, is_fraud, region, trans_day_of_week, is_weekend, trans_hour, trans_time_segment, age_group, cc_type, area_cat, distance, job_cat, log_amt, log_time_since_last_trans, log_city_pop, trans_hour_x_is_weekend, category_x_trans_day_of_week, and age_group_x_category. The target variable, is_fraud, indicates whether a transaction is fraudulent (1) or not (0), with a severe imbalance favoring non-fraudulent cases.

Feature Selection Process

To identify the most predictive features, the notebook applies five model-based selectors:

- **Lasso (L1 Regularization)**: Selects features by shrinking less important coefficients to zero.
- **Ridge (L2 Regularization)**: Assesses feature importance, retaining more features than Lasso.
- **Linear SVM (with L1 penalty)**: Selects features based on their contribution to the decision boundary.
- **Gradient Boosting Classifier**: Evaluates feature importance through tree-based scores.
- **Random Forest Classifier**: Ranks features using importance scores from tree ensembles.

Each model determines whether a feature is selected (1) or not (0), and the results are aggregated to assess consensus. Features selected by at least four models are deemed most important, including:

- Universally selected (5/5 models): category, log_amt, unix_time, trans_hour, category_x_trans_day_of_week, trans_hour_x_is_weekend.
- Strong features (4/5 models): cc_type, distance, log_city_pop, log_time_since_last_trans, job_cat, age_group_x_category, trans_time_segment, region.

The final recommended feature set includes 15 features, such as 'unix_time', 'category', 'region', 'trans_day_of_week', 'trans_hour', 'trans_time_segment', 'cc_type', 'distance', 'job_cat', 'log_amt', 'log_time_since_last_trans', 'log_city_pop', 'trans_hour_x_is_weekend', 'category_x_trans_day_of_week', 'age_group_x_category'. These are saved in a processed dataset for further modeling.

Addressing Class Imbalance

The notebook tackles the challenge of class imbalance, where fraudulent transactions are a minority, using four resampling techniques:

- **Random Oversampling (ROS)**: Duplicates minority class samples to balance the dataset, risking overfitting.
- **Random Undersampling (RUS)**: Removes majority class samples, potentially losing valuable information.
- **SMOTE (Synthetic Minority Over-sampling Technique)**: Generates synthetic minority samples by interpolating between existing ones, reducing overfitting risk.
- **SMOTETomek**: Combines SMOTE with Tomek Links to remove borderline majority samples, enhancing class separation.

A Random Forest Classifier is trained on each balanced dataset, evaluated using accuracy, precision, recall, and F1-score. The performance comparison is summarized in the following table:

Technique	Accuracy	Precision	Recall	F1-Score
ROS	0.9992	1.0000	0.8416	0.9140
RUS	0.9793	0.1867	0.9593	0.3126
SMOTE	0.9991	0.9222	0.8846	0.9030
SMOTETomek	0.9990	0.9093	0.8846	0.8968

ROS achieves the highest F1-score (0.9140) and perfect precision (1.0), indicating minimal false positives, but its recall (0.8416) suggests missed fraud cases. RUS has high recall (0.9593) but poor precision (0.1867), leading to many false positives. SMOTE and SMOTETomek offer a balanced approach, with SMOTE slightly outperforming in F1-score (0.9030) and precision (0.9222), making it a robust choice for production to avoid overfitting.

Model Training and Evaluation

The Random Forest Classifier is the primary model for classification, evaluated both with and without balancing. Without balancing, it achieves:

- Accuracy: 99.92%
- Precision: 99.47%
- Recall: 84.62%
- F1-Score: 91.44%

This indicates excellent overall performance but highlights the need for better recall to catch more fraud cases. After applying balancing techniques, the metrics vary, with SMOTE improving recall to 88.46% while maintaining high precision (92.22%). Confusion matrices and distribution plots (e.g., fraud vs. non-fraud) are used to visualize results, ensuring a comprehensive evaluation.

Credit Card Fraud Detection Model – Chervin Daniel

Conclusions and Recommendations

The notebook concludes with several key findings and recommendations:

- **Feature Selection**: Focus on core features like category, log_amt, and trans_hour, selected by multiple models, for robust fraud detection. Experiment with additional features like cc_type and distance for potential enhancements.
- **Class Imbalance Handling**: Recommend using SMOTE or ROS for production, with SMOTE preferred to mitigate overfitting risks. Monitor the trade-off between precision (minimizing false positives) and recall (catching frauds), given their business implications.
- **Model Performance**: The Random Forest Classifier performs well, but further improvements could involve threshold tuning or ensemble methods to balance precision and recall. Export balanced datasets (e.g., using SMOTE) for reproducibility and future analysis.

These recommendations provide a clear path for building an effective fraud detection model, balancing predictive power with practical considerations.

Credit Card Fraud Detection Model – Chervin Daniel

Model Selection and Fine-tuning

Introduction

The notebook, titled "06_ds18_ml-proj_model_sel_fine_tuning.ipynb," explores machine learning model selection and optimization for detecting fraudulent transactions. It aims to balance precision (avoiding false positives) and recall (catching actual frauds), which are crucial for effective fraud detection systems.

Models and Evaluation

The notebook evaluates several models, including Linear Regression, Decision Tree, Random Forest, Adaptive Boosting (AdaBoost), Gradient Boosting Machine (GBM), Support Vector Machine (SVM), and XGBoost Classifier. Performance is assessed using metrics like accuracy, precision, recall, and F1-score, with a focus on how well each model identifies fraud while minimizing errors.

Key Findings and Comparison

- Linear Regression performed adequately but was outperformed by ensemble methods like Random Forest and XGBoost.
- Decision Tree and AdaBoost showed good balance, while GBM had near-perfect precision and recall.
- Random Forest and XGBoost stood out, with optimized versions excelling in recall (Random Forest) and precision (XGBoost).
- An unexpected detail is the use of RandomizedSearchCV for fine-tuning Random Forest, showing practical optimization techniques.

Conclusion

Both Random Forest and XGBoost are highly effective after fine-tuning. Choose Random Forest to minimize missed frauds or XGBoost to reduce false alarms, based on your business priorities. This approach ensures a reliable fraud detection system.

Detailed Analysis and Observations

The analysis of the Jupyter notebook "06_ds18_ml-proj_model_sel_fine_tuning.ipynb" provides a comprehensive exploration of machine learning model selection and fine-tuning for fraud detection, a critical application in financial security. This section expands on the key points, offering a detailed breakdown of the content, methodologies, and conclusions, suitable for a professional audience seeking a thorough understanding.

Overview and Objectives

The notebook begins by outlining its purpose: to train and optimize machine learning models to distinguish between fraudulent and legitimate transactions. The primary goals include training various classifiers, evaluating them using metrics like precision, recall, and F1-score, and applying hyperparameter tuning to enhance performance. This focus on fraud detection highlights the importance of minimizing both false positives (legitimate transactions flagged as fraud) and false negatives (missed fraudulent transactions), which can have significant financial and operational impacts.

The notebook is structured with markdown cells that provide explanatory text, alongside code cells (not extracted here) likely containing the implementation details. The markdown cells serve as a narrative, guiding the reader through the modeling process, evaluation, and final recommendations.

Models Evaluated

A diverse set of machine learning models was assessed, reflecting a robust approach to model selection:

- **Linear Regression**: Included despite being less common for classification tasks, it served as a baseline. The notebook notes its adequate performance but highlights its inferiority to ensemble methods.

- **Decision Tree**: Showed good performance, particularly in recall, making it suitable for catching most fraud cases.
- **Random Forest**: An ensemble method that achieved high accuracy and F1-score, with the optimized version excelling in recall, minimizing false negatives.
- **Adaptive Boosting (AdaBoost)**: Provided a strong balance between precision and recall, making it a viable candidate for production.
- **Gradient Boosting Machine (GBM)**: Demonstrated near-perfect precision and recall, indicating excellent fraud detection capability.
- **Support Vector Machine (SVM)**: Mentioned but with limited evaluation details provided in the extracted markdown, suggesting it may have been less emphasized.
- **XGBoost Classifier**: Delivered the best overall performance, with near-perfect metrics across accuracy, precision, recall, and F1-score, positioning it as a top contender.

This variety ensures a comprehensive comparison, covering both simpler models (like Linear Regression) and advanced ensemble methods (like Random Forest and XGBoost), which are known for their robustness in classification tasks.

Evaluation Metrics and Analysis

The evaluation relied on standard classification metrics, each serving a specific purpose in fraud detection:

- **Accuracy**: Measures overall correctness, useful for understanding general performance but less critical in imbalanced datasets like fraud detection.
- **Precision**: Critical for minimizing false positives, ensuring that transactions flagged as fraud are indeed fraudulent, preserving customer trust.
- **Recall**: Essential for catching actual frauds, minimizing false negatives, which can lead to financial losses.
- **F1-Score**: Balances precision and recall, providing a single metric for overall model reliability.

The notebook includes detailed evaluations for each model, with tables summarizing metrics. For example, the XGBoost Classifier achieved an accuracy of 99.97%, precision of 99.98%, recall of 99.97%, and F1-score of 99.97%, indicating near-perfect performance. Similarly, Random Forest showed comparable metrics, with slight variations in recall and precision post-optimization.

Hyperparameter Fine-Tuning

A significant aspect of the notebook is the focus on hyperparameter fine-tuning, which is crucial for maximizing model performance. The notebook discusses several techniques:

- **Grid Search**: Exhaustively searches through predefined hyperparameter combinations, ensuring thorough exploration but potentially computationally expensive.
- **Randomized Search**: Randomly samples combinations, offering a faster alternative for large search spaces, as seen with RandomizedSearchCV applied to Random Forest.
- **Bayesian Optimization / Optuna**: Mentioned as an advanced option, suggesting a probability-based approach for smarter parameter selection.

The application of RandomizedSearchCV to Random Forest is particularly notable, with comparisons between default and optimized versions using confusion matrices. For instance, the optimized Random Forest reduced false positives from 18 to 12 and false negatives from 10 to 7, improving both precision and recall. This practical example underscores the importance of tuning for real-world deployment.

Model Comparison and Confusion Matrix Analysis

The notebook provides a detailed comparison, especially between optimized Random Forest and XGBoost, using confusion matrices to highlight performance differences:

Credit Card Fraud Detection Model – Chervin Daniel

Model	True Negatives (TN)	False Positives (FP)	False Negatives (FN)	True Positives (TP)	Notes
Random Forest	41,601	12	7	41,926	Lower false negatives (better recall)
XGBoost	41,610	3	12	41,921	Lower false positives (better precision)

This comparison reveals:

- **Random Forest**: Excels in recall, with fewer missed frauds (FN = 7), making it ideal when catching all frauds is critical.
- **XGBoost**: Excels in precision, with fewer false alarms (FP = 3), suitable when avoiding disruption to legitimate users is a priority.

The notebook also includes a broader model comparison table, summarizing metrics across all evaluated models:

Model	Accuracy	Precision	Recall	F1-Score
Linear Regression	0.8801	0.8969	0.8601	0.8781
Decision Tree	0.9972	0.9964	0.9979	0.9972
Random Forest	0.9997	0.9996	0.9998	0.9997
Adaptive Boosting (ADABoost)	0.9672	0.9856	0.9485	0.9667
Gradient Boosting Machine (GBM)	0.9868	0.9948	0.9788	0.9868
XGBoost Classifier	0.9997	0.9998	0.9997	0.9997

This table reinforces XGBoost and Random Forest as top performers, with XGBoost slightly edging out due to its precision, while Random Forest offers a slight edge in recall post-optimization.

Conclusions and Recommendations

Recommendations based on the analysis:

- **XGBoost** is recommended as the final model for deployment, given its exceptional precision and recall, making it ideal for high-stakes fraud detection where accuracy is paramount.
- **Random Forest** serves as a reliable fallback, especially when interpretability is desired or when minimizing false negatives is critical.
- The choice between them depends on business tolerance for missed frauds versus false flags, highlighting the need for alignment with organizational priorities.

This conclusion is supported by the detailed evaluations and comparisons, ensuring that the recommendations are data-driven and practical for real-world implementation.

Credit Card Fraud Detection Model – Chervin Daniel

Project Summary: Deployment and Beneficiaries of Machine Learning

Who will use and benefit from Machine Learning?
Fraud analysts, banks, credit card companies, and end users benefit directly. It enhances fraud prevention efforts, reduces financial losses, and improves trust in digital transactions.

🎯 Prioritizing In-Person Transactions (Minimize Interruptions)

Goal:

Avoid **false positives** (flagging legit transactions as fraud), since rejecting a valid card in-person is annoying or embarrassing for the user.

Metric to Prioritize:

- ✅ **Precision** (i.e., when we predict fraud, it's truly fraud)
- Minimize **False Positives**

Strategy:

- Choose the model with **higher precision**, even if recall drops slightly.
- Look at the **Precision-Recall trade-off** and tune the decision threshold accordingly.
- Possibly add a **human-in-the-loop** fallback for borderline cases.

🌐 Prioritizing Online Transactions (Minimize Fraud Risk)

Goal:

Catch as much fraud as possible, even if it means occasionally blocking a legit transaction (which can be retried or verified more easily online).

Metric to Prioritize:

- ✅ **Recall** (i.e., catch as many actual frauds as possible)
- Minimize **False Negatives**

Strategy:

- Choose the model with **higher recall**, accepting slightly lower precision.
- Make use of **two-factor authentication** or **user verification** for suspicious transactions.

⚖️ Balanced Scenario: Accuracy and F1 Score

If both types of transactions are important, or you want an overall balance:

- Consider **F1 score** (harmonic mean of precision and recall).
- Use **accuracy** only if the dataset is balanced (which fraud datasets often are **not**).