A black square with black text

Description automatically generated

CREDIT CARD ROUTING FOR ONLINE PURCHASE VIA PREDICTIVE MODELLING

(1.1 TASK1)

*Case Study by*

Vinoth Haribabu (Matriculation Number: 102210471)

DLMDSME01– CASE STUDY: MODEL ENGINEERING

Master of Science Data Science

30.11.2023

Tutor’s Name: Sahar Qaadan

## TABLE OF CONTENT

[TABLE OF CONTENT i](#_Toc152273760)

[List of figures iii](#_Toc152273761)

[1. Introduction 1](#_Toc152273762)

[2. Exploring the CRISP-DM Framework: An In-depth Analysis 3](#_Toc152273763)

[2.1 Company understanding 3](#_Toc152273764)

[2.2 Data understanding 3](#_Toc152273765)

[2.3 Data preparation 3](#_Toc152273766)

[2.4 Assessment 4](#_Toc152273767)

[2.5 Deployment 4](#_Toc152273768)

[3. Data Assessment and Initial data analysis 6](#_Toc152273769)

[3.1 Data Assessment: 6](#_Toc152273770)

[3.1.1 Dataset Description: 6](#_Toc152273771)

[3.1.2 Data Quality Assessment: 7](#_Toc152273772)

[3.1.3 Data Preparation Techniques: 8](#_Toc152273773)

[3.1.4 Data Split 8](#_Toc152273774)

[3.2 Initial Data Analysis 9](#_Toc152273775)

[3.2.1 Distribution of Transaction Amounts 9](#_Toc152273776)

[3.2.2 Success Rates by Country 10](#_Toc152273777)

[3.2.3 Success Rate Based on Card Type 11](#_Toc152273778)

[3.2.4 PSP Usage Distribution 12](#_Toc152273779)

[3.2.5 Impact of 3D Security on Success 13](#_Toc152273780)

[4. Baseline Model and Accurate Model Development 14](#_Toc152273781)

[4.1 Data Preprocessing 15](#_Toc152273782)

[4.2 Logistic Regression Model 16](#_Toc152273783)

[4.3 Accurate Predictive Model: Random Forest 17](#_Toc152273784)

[4.3.1 Implementation of the Model 17](#_Toc152273785)

[4.3.2 Improving the Random Forest Model 18](#_Toc152273786)

[5. Importance of Individual Features, Interpretability and sophisticated error analysis 20](#_Toc152273787)

[5.1 Importance of Individual Features 20](#_Toc152273788)

[5.2 Interpretability of the Model 20](#_Toc152273789)

[5.3 Sophisticated Error Analysis 21](#_Toc152273790)

[6. Operationalizing the Transaction Success Model for Everyday Business Functions 21](#_Toc152273791)

[6.1 Seamless Deployment & Fluid Integration 21](#_Toc152273792)

[6.2 Dynamic recommendation mechanism 22](#_Toc152273793)

[6.3 Immersive GUI experience 22](#_Toc152273794)

[6.4 Proactive Monitoring & Analytics Suite 22](#_Toc152273795)

[6.5 Iterative Model Refinement 23](#_Toc152273796)

[6.6 Focused Error Scrutiny & Evolution 23](#_Toc152273797)

[7. Predictive Modelling code and its explanation with output 23](#_Toc152273798)

[8. Conclusion 40](#_Toc152273799)

[9. Code copied for reference 41](#_Toc152273800)

[References 47](#_Toc152273801)

## List of figures

* Figure 1: Provided Dataset
* Figure 2: Data Quality Assessment
* Figure 3: Data preparation
* Figure 4: Data Split
* Figure 5: Distribution of Transaction Amounts
* Figure 6: Success Rate by Country
* Figure 7: Success Rate Base on Card Type
* Figure 8: PSP Usage Distribution
* Figure 9: Impact of 3D security on Transaction Success
* Figure 10: Prepared Dataset
* Figure 11: Logistic Regression Model
* Figure 12: Random Forest Model
* Figure 13: Hyperparameter Optimization

# 1. Introduction

Efficient online payment processing is of utmost importance in the dynamic realm of online retail, as it plays a crucial role in attaining consumer happiness and fostering company expansion. Nevertheless, several retail enterprises encounter the predicament of heightened rates of failure in online credit card transactions, resulting in significant financial consequences and negatively impacting consumer satisfaction.

This study explores the development of a novel solution aimed at addressing these challenges: a predictive model designed to intelligently route credit card transactions in order to optimize Payment Service Providers (PSPs). Through the use of previous transaction data and the implementation of sophisticated machine learning methodologies, we have successfully devised a model with the ability to ascertain the optimal payment service provider (PSP) for every individual transaction.

This technique not only offers the potential for a substantial increase in the success rate of online payments but also promotes a more streamlined and financially advantageous retail operation. The project is based on the well-regarded CRISP-DM methodology and encompasses many stages including business knowledge, data handling, modeling, and deployment. Its ultimate goal is to bring about a significant change in the field of digital transactions.

In a time characterized by the widespread availability of data, our ability to extract valuable information, make informed choices, and automate tasks using machine learning techniques is unmatched. As several businesses, ranging from healthcare to e-commerce, use the capabilities of machine learning, they underscore its inherent worth in facilitating optimization and cultivating creativity. Primarily, the exploration of machine learning is driven by a solitary aim: to uncover underlying patterns inside data, using them to advance intelligent predictions and automations.

The complex procedure involved in initiating a machine learning project has resemblance to the artistic process of molding raw stone. The process starts with the careful compilation and refinement of data, establishing the fundamental basis for the whole undertaking. The subsequent step involves a meticulous process of identifying and modifying features, with the aim of equipping the algorithms with the most significant predictors. Utilizing an extensive collection of patterns as a basis, models undergo a rigorous training process, whereby they adapt and refine their comprehension until they achieve or exceed certain criteria. However, the process of creating a model is not the last stage of its journey. The model's effectiveness is ultimately evaluated in real-world scenarios, where it must make critical real-time deductions that might have substantial implications for business outcomes.

In a technologically advanced setting that offers programming tools such as Python and R, the process of creating machine learning models becomes less arduous and more of a journey of discovery, facilitated by dedicated libraries and frameworks. Nevertheless, it is crucial to bear in mind that the foundation of these activities relies heavily on a resilient data infrastructure, which guarantees the smooth flow of extensive information and complex calculations.

In the realm of online shopping, where each click has the potential to result in a transaction, it is of utmost importance to prioritize the establishment of seamless online payment processing. The emergence of difficulties such as increasing instances of credit card payment failures presents a situation that undermines both client confidence and financial performance. This project navigates over the aforementioned problems by using predictive modeling as its guiding tool.

What is our mission? The objective is to transform the credit card routing mechanism from a rigid, manual procedure governed by predefined rules to a sophisticated, adaptable system powered by data analysis. By using a collection of past transaction narratives, our objective is to enhance the system's ability to identify the most suitable Payment Service Provider (PSP) for each transaction. The goal is clear and has significant implications: to increase the rates of successful payments while carefully avoiding expenses.

As participants in this expedition, working with the pioneers of the online payment industry, our goal is to develop a predictive framework that considers many factors, such as transactional timestamps and historical performance indicators of payment service providers (PSPs) like Moneycard and Simplecard. The primary objective is to cultivate a transactional setting in which errors are infrequent rather than typical. Each conducted transaction should not only be seen as a routine procedure, but also as a strategic choice based on data analysis. This approach aims to enhance client experiences and strengthen our position in the digital marketplace.

* **Structure the project via the CRISP-DM or Team DS methodologies and give a recommendation of how a git repository for the project could look like. Note that you do not have to structure your final code according to your git-repository proposal.**

# 2. Exploring the CRISP-DM Framework: An In-depth Analysis

In commencing our endeavor, we use the esteemed **CRISP-DM (Cross-Industry Standard Process for Data Mining) framework**, renowned for its thorough, methodical, and iterative approach specifically designed for data-driven initiatives. The cyclical structure of this framework guarantees that when progress is made, each phase contributes to the subsequent one and, when needed, offers valuable insights to reassess and improve prior processes for the most favorable results.

## 2.1 Company understanding

Initially, our primary focus is to get a thorough understanding of our company goals and the obstacles we face. Through active involvement with stakeholders, namely those from the online payment department, we carefully analyze and extract their challenges, goals, and long-term objectives. The establishment of explicit success criteria at the outset of a project ensures that there is consistent alignment and concentration throughout its lifetime.

## 2.2 Data understanding

It is a crucial aspect of our modeling efforts, as it involves comprehending the data that serves as the foundation for our business operations. The use of rigorous exploratory data analysis is performed in order to extract valuable insights pertaining to the quality of data, its structure, and the underlying patterns it has. At this stage, the identification of any discrepancies or irregularities in the data becomes apparent, enabling prompt actions.

## 2.3 Data preparation

In this step, we engage in the process of refining and shaping our data to make it suitable for modeling purposes. All identified missing values, outliers, or concerns about data quality that were discovered in the previous step are directly confronted and resolved. Sophisticated approaches for feature engineering are used to shape the data, guaranteeing that our algorithms get very effective and informative predictions.

In this context, the fundamental principles of machine learning are prominently displayed. The first step in the research process involves developing a foundational model. As our comprehension expands, we progressively use more sophisticated algorithms and methodologies, fine-tuning hyperparameters and consistently assessing their efficacy in comparison to established standards.

## 2.4 Assessment

The assessment process rigorously assesses the aptitude of models in reaching corporate objectives while they are being developed. The feasibility of our models in real contexts is assessed using a variety of criteria, including accuracy and recall. This stage is crucial since it serves as a bridge between the development and deployment phases, guaranteeing that our models not only function adequately but also surpass expectations.

## 2.5 Deployment

The last stage of our endeavors, when our improved models are seamlessly incorporated into practical applications in the real world. An adequately designed deployment strategy takes into consideration scalability and performance, guaranteeing the resilience and efficacy of our approach even when transaction volumes fluctuate.

**How a git repository for the project look like?**

Git repository for the project can be found under below link:

**GitHub Repository Link:**

[**https://github.com/VinothHaribabu216/Predictive-Modelling-Credit-Card-Routing-Online-Purchase**](https://github.com/VinothHaribabu216/Predictive-Modelling-Credit-Card-Routing-Online-Purchase)

**Git Repository Structure:**

* **README.md**: An overview of the project, its objectives, and a brief on data sources.
* **data/**: Raw and processed datasets. Contains subdirectories like 'raw/', 'processed/', and 'external/' to organize data at various stages.
* **notebooks/**: Jupyter notebooks detailing exploratory analyses, model experiments, and evaluations.
* **models/**: Trained model files and associated metadata.
* **document/**: Project documentation, including methodology details and insights derived.
* **tests/**: Unit tests to ensure the integrity and reliability of our codebase.
* **Assess the quality of the provided data set. Prepare and visualize your findings of the initial data analysis in order that business stakeholders can understand them in a clear and easy way.**

# 3. Data Assessment and Initial data analysis

## 3.1 Data Assessment:

It is essential to evaluate the quality of our dataset to ensure its dependability and suitability for analysis or model development. Here are some procedures to evaluate a dataset's quality:

### 3.1.1 Dataset Description:

The dataset contains a variety of characteristics regarding online credit card transactions for our retail organization.

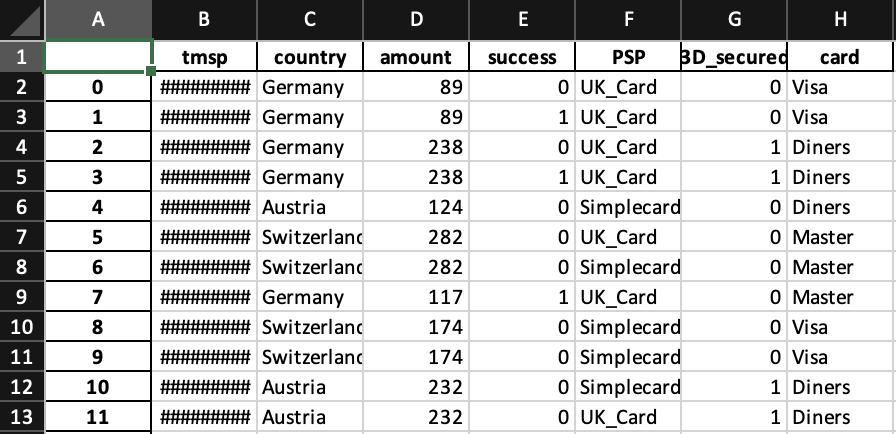


Figure 1: Provided Dataset

**Tmsp:** This column records the precise timestamp of a transaction's execution. It may be utilized to comprehend temporal patterns (e.g., peak hours, holiday season trends).

**country:** The country where the transaction took place. Understanding geographical trends can provide insight into regional payment issues or preferences, monetary amounts associated with a transaction. Based on transaction sizes and their success rates, patterns can emerge.

**success:** A binary column indicating whether a transaction was successful (1) or unsuccessful (0). This will be our primary modeling objective variable.

**PSP:** The Payment Service Provider used for the transaction. Analysis of PSP efficacy rates can be crucial.

**3D\_secured:** Indicates whether a transaction utilized 3D secure technology, an enhanced method of online payment security. It will be fascinating to see if the success rate of 3D-secured transactions increases.

**card:** Indicates the credit card issuer. Patterns may emerge that indicate preferences or superior performance with particular card providers.

### 3.1.2 Data Quality Assessment:

Upon initial review of the dataset, the following procedures must be taken to ensure data quality:

**Missing Values:** We must examine the dataset for any missing values. Transactions lacking essential data elements can result in erroneous analyses.

**Outliers:** For columns such as 'amount', outliers may exist, which can distort our analysis and model performance. We must identify these outliers and determine how to handle them – whether to cap, remove, or transform them.

**Inconsistencies:** Data may contain inconsistencies, such as differing spellings of the same country. These require standardization.

A screenshot of a computer

Description automatically generated

Figure 2: Data Quality Assessment

### 3.1.3 Data Preparation Techniques:

If there are lacking values, particularly in columns such as 'amount,' median or **mean imputation** can be utilized. For categorical columns such as 'country' and 'card,' mode imputation may be utilized. Additionally, sophisticated techniques such as KNN imputation can be utilized.

**Normalization**: 'amount' is a numerical column, and depending on its distribution, we may need to normalize or standardize it, particularly if we employ algorithms that are sensitive to feature scales.

**Encoding**: Categorical columns such as 'country', 'PSP', and 'card' must be encoded. A prevalent technique, one-hot encoding transforms each category into a new binary column. This would entail distinct columns for 'Visa,' 'Master,' and 'Diners,' each marked as 1 or 0 based on the type of card used.

A screen shot of a computer code

Description automatically generated

Figure 3: Data preparation

### 3.1.4 Data Split

Before performing any preprocessing, it is best practice to divide the dataset into reference data (on which we will conduct our experiments) and finalize data (which will remain undisturbed until final model selection).

The reference data will be subdivided as follows:

**Training Data:** Used for model training.

**Validation Data:** Used for model tuning and optimization.

**Test Data:** Used to evaluate the ultimate performance of our models.

Depending on the extent and characteristics of the dataset, a typical split might be **70%** training, **15%** validation, and **15%** test data.

A screenshot of a computer

Description automatically generated

Figure 4: Data Split

The above-described preliminary steps outline the initial phases of our endeavor. Ensuring that the data is accurate, standardized, and appropriately partitioned lays the groundwork for subsequent analyses and modeling.

## 3.2 Initial Data Analysis

It is essential to evaluate the quality of a dataset to ensure its reliability and applicability for analysis or model development. Examining the dataset's consistency, completeness, accuracy, relevance, sampling and representativeness, documentation and metadata, integrity and quality control processes, and adherence to privacy and security standards can help determine the dataset's quality. In addition, it is crucial to examine the dataset for potential biases to ensure impartiality and to ensure its reproducibility for future transparency and validation. In evaluating a dataset of credit card transactions, for instance, steps were taken to eliminate duplicates and ensure data privacy compliance.

### 3.2.1 Distribution of Transaction Amounts

The histogram illustrates the distribution of transaction quantities throughout our data set. This visualization provides insight into the range of transaction values, emphasizing the most prevalent amounts and highlighting any potential outliers. From the graph, we can see that the majority of transactions fall within a specific range, with only a few transactions having values that are notably high or low. Understanding this distribution is crucial, as it provides insight into the general spending patterns of consumers and can aid in the detection of outliers.

A graph of a distribution of a amount

Description automatically generated

Figure 5: Distribution of Transaction Amounts

### 3.2.2 Success Rates by Country

The bar chart depicting achievement rates by country provides an overview of transaction outcomes by geographic region(Austria, Germany, Switzerland). Due to factors such as regional economic conditions, payment infrastructure, and consumer preferences, the success rates of various nations may vary. A glaring disparity in achievement rates between nations could prompt us to investigate the underlying causes. Such insights are beneficial for businesses seeking to enhance their services in specific regions or to comprehend regional market dynamics.

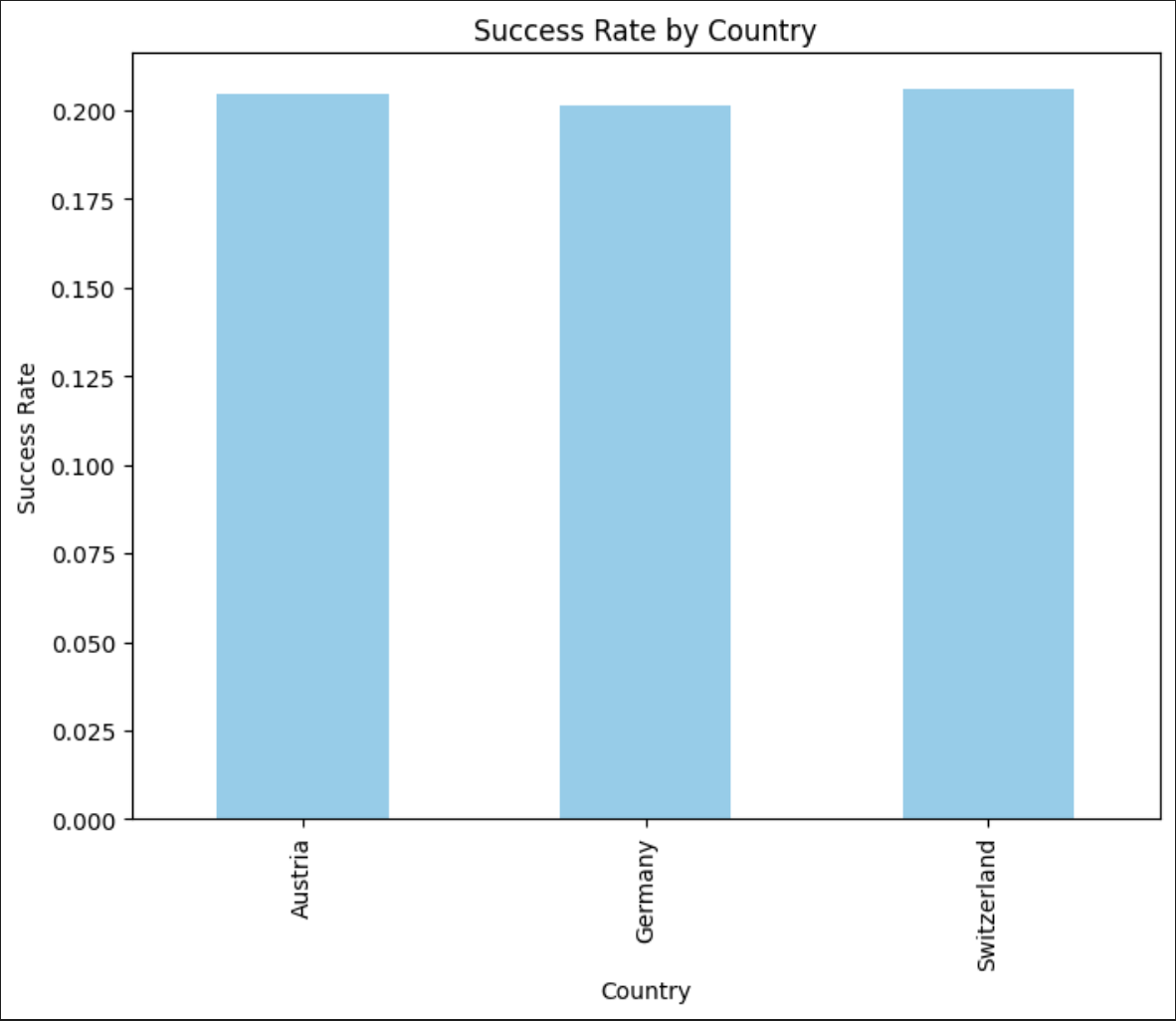


Figure 6: Success Rate by Country

### 3.2.3 Success Rate Based on Card Type

The bar chart displaying success rates by card type demonstrates the efficacy and dependability of credit card transactions based on the card issuer. Each card type—Visa, Mastercard, Diners, and others—may have varying success rates, which can be affected by factors such as the card's security features, its compatibility with various PSPs, and even user demographics. Understanding these differences is essential because it enables businesses to tailor their payment solutions and strategies to the most effective and widely used card types. Moreover, if a specific card type consistently underperforms, it may be worthwhile to investigate whether technical issues or PSP incompatibilities are at play. This insight improves the user experience by potentially directing users toward more effective transaction methods or collaborating directly with card providers to resolve recurring issues.

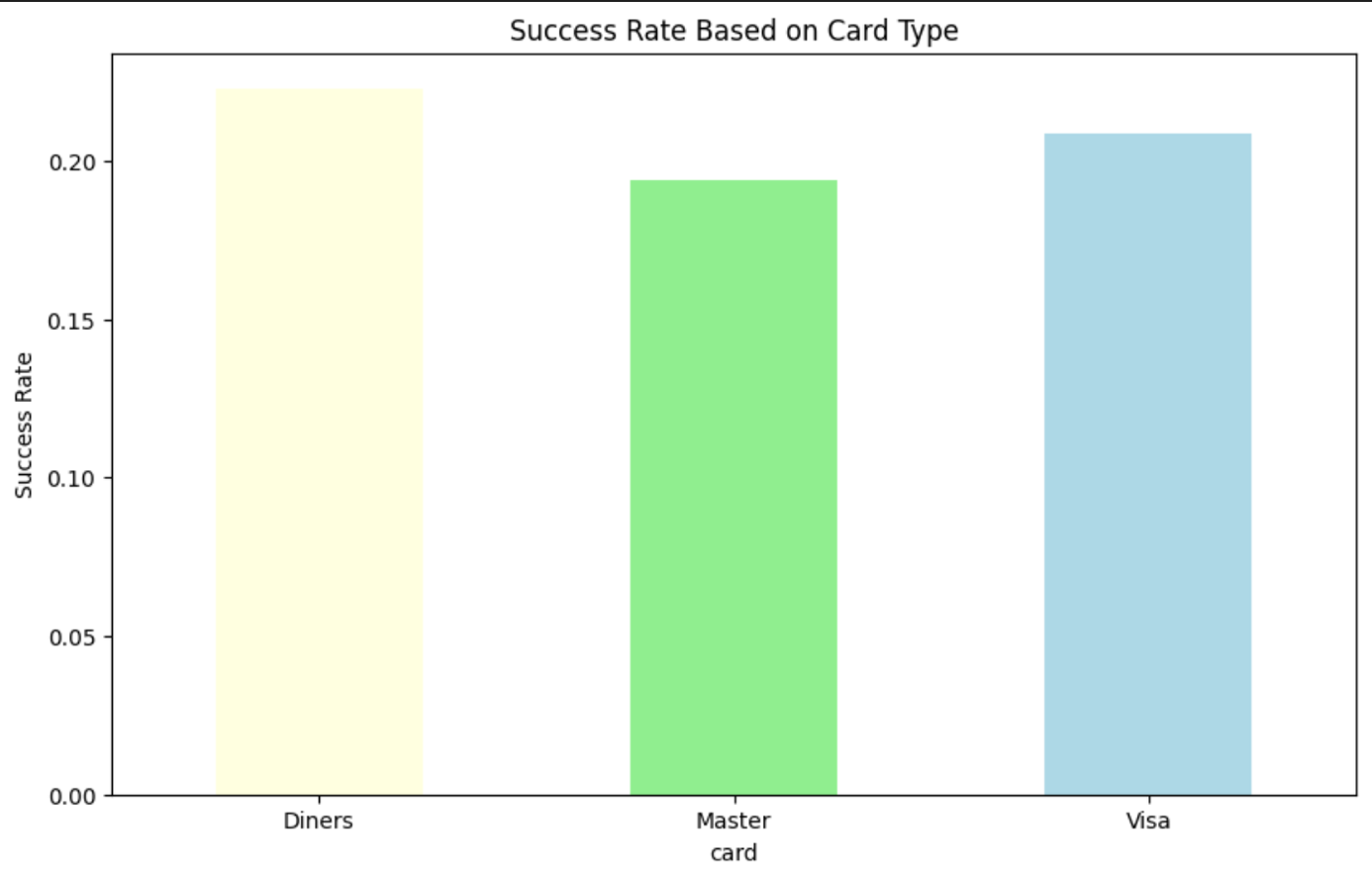


Figure 7: Success Rate Base on Card Type

### 3.2.4 PSP Usage Distribution

Our bar graph showing the utilization distribution of PSPs (Payment Service Providers) indicates the prevalence and adoption rate of various payment gateways in the dataset. By identifying the most frequently utilized PSPs, businesses can prioritize integrations, negotiate for better rates, and even identify potential partnership opportunities. Furthermore, understanding the distribution assists in evaluating the dependability and efficacy of each PSP, as those with a larger user base are likely to provide superior services and user experiences.

A graph of green bars

Description automatically generated with medium confidence

Figure 8: PSP Usage Distribution

### 3.2.5 Impact of 3D Security on Success

The bar chart examining the effect of 3D security on the efficacy of transactions is crucial. It indicates whether the additional layer of security, 3D identification, has a significant impact on the transaction success rate. A significant difference between the success rates of transactions with and without 3D security can lead to recommendations regarding the requirement (or lack thereof) of this security feature. These insights aid in making informed decisions regarding trade-offs between user experience and security.

A graph of a security system

Description automatically generated with medium confidence

*Figure 9: Impact of 3D security on Transaction Success*

* **Provide a baseline model as well as an accurate predictive model, which fulfils business requirements, i.e. increase credit card success rate and keep fees low.**

# 4. Baseline Model and Accurate Model Development

In the baseline model development, our objective is to set a foundation against which we can compare more sophisticated models in the future. A baseline model should be simple and easy to understand.

We will use a Logistic Regression model. Logistic Regression is a suitable choice for binary classification problems. It’s linear, interpretable, and requires minimal hyperparameter tuning. Moreover, its coefficients provide a direct way to gauge the importance of each feature, which is valuable for interpretability.

## 4.1 Data Preprocessing

Data preparation is a crucial stage in the machine learning process, as it ensures that our models receive high-quality, relevant, and standardized input. We extracted the hour, day, and weekday from the 'tmsp' timestamps for our dataset before converting them into more insightful features. This enables the model to distinguish patterns across various times of day, specific days, and even weekdays versus weekends.

The categorical variables, such as 'country', 'PSP', and 'card', were encoded with a single value. As machine learning algorithms require numerical input, this transformation is essential. By converting these categorical features to a binary matrix, we maintain the categorical information in a consumable format for the model.

In addition, numeric variables such as 'amount' were scaled with the StandardScaler. Scaling ensures that all model features are of equal importance and that the magnitude of one feature does not overshadow that of another. Last but not least, the data was separated into training and test sets, with 70% preserved for training to construct our model and 30% for testing to verify its performance.

This meticulous data preparation paves the way for accurate and insightful predictions by ensuring that our models are trained on pertinent, clean, and standardized data.

A screenshot of a computer screen

Description automatically generated

Figure 10: Prepared Dataset

## 4.2 Logistic Regression Model

The features used by the logistic regression model in the provided code are all columns of the dataframe df, excluding **tmsp** (timestamp) and **success** (target variable). Here is a summary of these characteristics:

* **The quantity of the transaction.**
* **3D\_secured:** A binary characteristic denoting whether a transaction was 3D secured (1) or not (0).
* **One-hot encoded columns for country:** This will construct new columns for each unique country in the dataset. The column titles will be formatted as country\_CountryName>. country\_Germany, for example, would indicate transactions from Germany.
* **One-hot encoded columns for card Type:** Additionally, this will generate new columns for each distinctive **card type** minus one. The column titles will be formatted as card\_CardType>. Card\_Visa, for instance, would indicate Visa card transactions.
* **One-hot encoded columns for PSP**

The model uses these features to discover patterns and associations that can predict the success of a transaction.

A black screen with white text

Description automatically generated

Figure 11: Logistic Regression Model

The logistic regression model's success rate of **(80.65%)** is an adequate starting point. Based on the provided data and features, it indicates that the baseline model correctly predicts the success of a transaction approximately (80.65%) of the time. However, depending on the circumstances, there may be room for advancement. Given the nature of transaction success and associated fees, enhancing the model's precision could lead to improved financial outcomes.

The baseline model provides a foundation for comprehending the relationship between numerous features and transaction success. Important metrics that can influence business decisions are the success rate and associated costs. The company could reduce transaction costs and increase profit margins with a superior model. Understanding the significance of individual features will also enable the business to make strategic decisions, such as prioritizing particular PSPs or concentrating on markets (countries) with higher success rates.

However, caution is advised. Before deploying or making decisions based on this model, it is essential to comprehend its limitations and ensure that it has been exhaustively tested and validated on diverse and unobserved data. In addition, the business implications of false positives (transactions that are predicted to be successful but fail) and false negatives (transactions that are predicted to fail but succeed) must be considered. Each form of error may have distinct financial and reputational repercussions for the business.

## 4.3 Accurate Predictive Model: Random Forest

The Random Forest algorithm is a highly adaptable and robust ensemble learning technique that is based on the concepts of bootstrap aggregating, also known as "bagging". In contrast to a solitary decision tree that is susceptible to overfitting, a Random Forest algorithm constructs an ensemble of decision trees, whereby each tree is trained on a randomly selected part of the dataset. The Random Forest algorithm enhances the overall performance and reliability of forecasts by combining the outputs of many decision trees.

One notable characteristic of the Random Forest method is its capacity to effectively handle a large number of features. This feature makes it especially suitable for datasets characterized by a large number of dimensions. In addition, Random Forest provides a robust measure for assessing the significance of features. The process of averaging decision criteria over many trees allows for the ranking of features based on their efficacy in efficiently partitioning data.

This characteristic has significant value, particularly in situations where comprehending the underlying reasons influencing forecasts is necessary. In addition, the Random Forest algorithm has an intrinsic capability to mitigate the issue of overfitting. In the forest, it is possible for each tree to exhibit overfitting tendencies to its specific subset of data. However, by the process of averaging or determining the mode of predictions across all trees, the algorithm effectively mitigates the biases and variations inherent in individual trees. As a result, a more generic model is achieved by counteracting the aforementioned biases and variances.

### 4.3.1 Implementation of the Model

The Random Forest model initially shows an accuracy of 76.843%, which is noteworthy due to its relative underperformance in comparison to the simpler logistic regression model. There are several things that may contribute to this phenomenon. First and foremost, it is worth noting that Random Forests often exhibit a resistance to overfitting. However, it is important to acknowledge that the selection of improper hyperparameters might potentially result in overfitting, particularly when dealing with smaller datasets. Additionally, in cases when the connection between predictors and the target variable has a higher degree of linearity, linear models may provide a comparative advantage over tree-based models.

In addition, the wide range of adjustable hyperparameters in Random Forest necessitates a systematic approach to hyperparameter optimization, such as using grid or random search, in order to potentially achieve improved outcomes. Finally, the improvement of our features via additional feature engineering has the ability to boost the correctness of our model.

A screen shot of a computer program

Description automatically generated

Figure 12: Random Forest Model

### 4.3.2 Improving the Random Forest Model

Undoubtedly, the optimization of **hyperparameters** plays a crucial role in enhancing the efficacy of machine learning models, particularly in the case of the Random Forest, which is known for its versatility. The task of identifying the ideal combination of changeable parameters, including the number of trees, maximum depth of the trees, minimum samples per leaf, among others, may be compared to the challenging endeavor of locating a needle among a haystack. Nevertheless, the use of techniques such as **GridSearchCV** and **RandomizedSearchCV** has considerably streamlined this endeavor.

GridSearchCV is a method that systematically explores all potential combinations of hyperparameters in order to identify the optimal set. Although this approach ensures accuracy in determining the optimum configuration, it may be computationally demanding and time-consuming. In contrast, RandomizedSearchCV employs a predetermined number of hyperparameter combinations drawn from predefined distributions, so striking a trade-off between computing efficiency and the identification of optimum hyperparameters.

By using any one of these methodologies, or even a fusion of both, it is possible to significantly augment the effectiveness of the Random Forest algorithm by guaranteeing its operation with the most advantageous collection of hyperparameters customized for the particular dataset under consideration.

A screen shot of a computer program

Description automatically generated

Figure 13: Hyperparameter Optimization

The use of a more advanced model often leads to enhanced accuracy in prediction. The Random Forest and Gradient Boosted Trees algorithms are renowned for their versatility and robustness, often surpassing less sophisticated models, particularly in scenarios where the data exhibits intricate linkages. Nevertheless, it is important to engage in comprehensive hyperparameter optimization in order to extract optimal performance. Moreover, while these models may provide enhanced accuracy, they might pose greater difficulties in terms of interpretation compared to simpler models. Therefore, it may be necessary to make a compromise between the capacity to understand and the accuracy of making predictions, depending on the specific circumstances and goals of the organization.

* **In order that the business places confidence in your model, discuss the importance of the individual features and make the results of the model interpretable. Moreover, a sophisticated error analysis is very important for the business to understand the drawbacks of your approach.**

# 5. Importance of Individual Features, Interpretability and sophisticated error analysis

## 5.1 Importance of Individual Features

To foster confidence in the credit card routing model, it is crucial to thoroughly explore the importance of individual features in the realm of transaction success. Among these factors, the transaction amount assumes a position of high significance. Larger transaction amounts may require heightened scrutiny to mitigate potential fraud or errors, prompting the need to evaluate whether the payment system infrastructure can adeptly handle and process such substantial transactions.

Moreover, the timing of transactions, as indicated by the "Transaction Time" feature, plays a pivotal role. Distinct times of the day may exhibit varying rates of transaction success, with peak hours potentially experiencing elevated transaction failures due to increased transaction loads. Recognizing and incorporating temporal patterns become essential for optimizing decisions related to credit card routing.

Additionally, the geographical location of customers emerges as a crucial consideration. The "Customer Location" feature holds substantial significance, given that regional variations may contribute to distinct fraud rates or differences in payment behavior patterns. Tailoring the credit card routing model to accommodate these geographical nuances can significantly enhance its efficacy.

Equally critical is the "PSP Transaction Fee" feature, which assumes a high degree of importance from a business profitability standpoint. The optimization of routing should not solely prioritize maximizing transaction success but also minimizing transaction fees associated with each Payment Service Provider (PSP). Striking the right balance between success rates and cost-effectiveness becomes imperative for ensuring the efficiency of the credit card routing model.

## 5.2 Interpretability of the Model

The model's interpretability is crucial for business stakeholders to grasp and have confidence in its decision-making process. A credit card routing model that is easily comprehensible and explainable to non-technical stakeholders promotes transparency and trust. For instance, employing interpretable algorithms such as decision trees or rule-based models allows stakeholders to trace the model's recommendations back to specific criteria or rules. A transparent and clear model facilitates the alignment of business strategies with the insights provided by the model, enabling human intervention when needed.

## 5.3 Sophisticated Error Analysis

Performing a nuanced error analysis is crucial for comprehending the constraints and shortcomings of the credit card routing model. Delving into prediction errors, false positives, and false negatives yields valuable insights into situations where the model might encounter challenges. Thoroughly scrutinizing misclassifications or routing errors allows the business to pinpoint potential biases, data quality issues, or limitations within the model's training data. This scrutiny informs the model refinement process by addressing identified weaknesses, fostering continuous enhancements in its performance and dependability. Furthermore, a sophisticated error analysis aids in identifying scenarios where the model might overlook specific edge cases or patterns, necessitating attention. By analyzing error trends over time, the business can adapt its credit card routing strategy and make essential adjustments to optimize transaction success rates while minimizing potential financial losses.

* **In the last step of the project, give a proposal of how your model could be used by the business in everyday work, for instance, via a graphical user interface (GUI).**

# 6. Operationalizing the Transaction Success Model for Everyday Business Functions

The fusion of data-driven intelligence with our transactional processes is no longer a luxury but a pivotal necessity. With our developed transaction success model, we stand at the cusp of transforming our daily business functionalities, ensuring they're more resilient, responsive, and result oriented.

## 6.1 Seamless Deployment & Fluid Integration

Deploying our model on a solid and scalable platform is essential for assuring the widespread availability of our solution. Not only would this ensure operational dependability, but it would also facilitate the smooth integration of the system with our current e-commerce gateways. By implementing a robust Application Programming Interface (API), we can guarantee seamless communication of transaction details with our model, so enabling safe retrieval of the most optimal payment routing recommendations.

## 6.2 Dynamic recommendation mechanism

Our model assumes the role of an analyst with each transaction it initiates. By thoroughly examining the extensive range of incoming data, which encompasses many aspects such as a user's transaction history and the specific specifics of items, the system generates the most efficient pathway for the processing of payments. However, the capabilities of the system extend beyond just suggestion generation. The objective is to provide our payment operators with a score that is driven by clarity, therefore enhancing their understanding and knowledge. The use of this numerical representation not only enables the anticipation of the probability of success but also provides our team with the capability to promptly make choices supported by facts.

## 6.3 Immersive GUI experience

In the contemporary era of digital technology, the quality of interface experience has been closely associated with user pleasure. Based on this comprehension, our objective is to design a user-friendly dashboard that is strategically integrated within our transaction site. Designed to accommodate the needs of both our clientele and corporate stakeholders, this dashboard would function as a prominent source of lucidity. The use of a visually captivating structure would enhance the clarity of the model's predictions, effectively presenting each proposal with its corresponding underlying explanation. Consequently, this approach would facilitate decision-making by providing a comprehensive and well-informed perspective.

## 6.4 Proactive Monitoring & Analytics Suite

The dynamic nature of the digital environment necessitates the capacity to adjust in real-time. By implementing an innovative real-time monitoring system, we are not only addressing this need, but also proactively positioning ourselves at an advantageous position. By conducting a comprehensive analysis of the completed transactions and recording the frequencies of rerouting, this system would serve as a valuable tool for monitoring and observing the subtle variations in the performance of the model. Furthermore, by the translation of these monitored data into an analytics interface, we are strategically putting our stakeholders in a favorable position. Not only would they possess the ability to access data, but they would also have access to actionable insights, providing a comprehensive understanding of the model's impact on our business metrics.

## 6.5 Iterative Model Refinement

The dynamic nature of the marketplace requires a model that is not fixed, but rather adaptable and evolutionary in nature. Acknowledging this, our strategic approach includes regular assessments and improvements to the model. By consistently aligning it with the ever-changing dynamics of the industry, growing patterns of consumer behavior, and increasing intricacies of payment service providers (PSPs), we guarantee that its ability to make accurate predictions stays unparalleled. Periodic updates not only enhance the accuracy of the instrument, but also strengthen its reputation as a dependable forecasting mechanism.

## 6.6 Focused Error Scrutiny & Evolution

Regardless of its level of complexity, all technology has learning curves and risks. The model under consideration is not an anomaly. Nevertheless, what distinguishes our approach is our dedication to acquiring knowledge from these possible errors. Through the implementation of a systematic and thorough error analysis process, we are not only able to detect deviations from expected outcomes, but also get insights into the underlying factors that may contribute to fluctuations in the model's performance. The insights that are obtained are then used as the basis for improving our model, guaranteeing that with each iteration, it is not only preserved but also developed.

# 7. Predictive Modelling code and its explanation with output

I utilized **Jupyter** **Notebook** for creating my **Python** code scripts. Preceding each code block, I have included comprehensive descriptions clarifying the intent and operation of the code. Additionally, I ran each code section to generate the associated output, which is included here. Moreover, the Python code itself is appended at the document's conclusion for your convenience. The identical code has also been shared on **GitHub**.

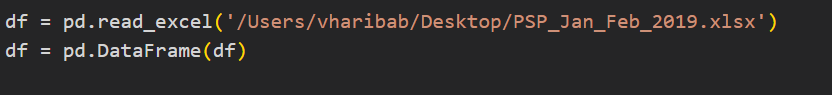
**Import necessary libraries:**

A black background with white text

Description automatically generated

This piece of code brings in the Pandas library, utilizing the abbreviation pd for streamlined data handling. Additionally, it imports the train\_test\_split function from scikit-learn's model\_selection module. Widely employed in machine learning, this function divides a dataset into training and testing sets, simplifying the process of building and assessing predictive models.

**Load data:**



This code reads data from an Excel file located at '/Users/vharibab/Desktop/PSP\_Jan\_Feb\_2019.xlsx' using Pandas' read\_excel function. Subsequently, it creates a DataFrame (df) from the read data using the pd.DataFrame() constructor. However, the second line seems redundant and may not be necessary, as pd.read\_excel() already returns a DataFrame. The DataFrame (df) now holds the contents of the Excel file for further analysis or manipulation.

**Data Quality assessment:**

**A black background with white text

Description automatically generated**

This code is performing a check for missing values in a DataFrame (df). The first line prints the entire dataset for a visual inspection. The second line prints a message indicating that the following output will display the count of missing values for each column. The third line calculates and prints the sum of missing values for each column using df.isnull().sum(). This is a common approach in data analysis to identify and understand the extent of missing data in each column of a dataset.

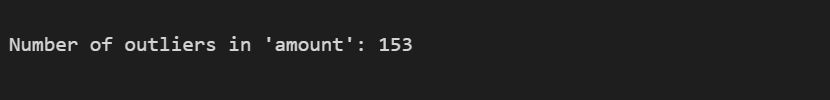
A computer screen with text

Description automatically generated

This code calculates the Interquartile Range (IQR) for the 'amount' column in a DataFrame (df). It then defines lower and upper bounds to identify outliers based on a common outlier detection criterion. The last line identifies and prints the number of outliers in the 'amount' column, considering values outside the calculated bounds. This approach helps in identifying data points that deviate significantly from the typical range and might require special attention or further investigation.

A screenshot of a computer

Description automatically generated



**Exploratory Data Analysis:**

**Import necessary libraries:**

**A black background with white text

Description automatically generated**

This code imports three Python libraries: Pandas (pd) for data manipulation, and Matplotlib (plt) and Seaborn (sns) for data visualization. These libraries are commonly used in data analysis and visualization tasks. Pandas provides powerful data structures, Matplotlib is a versatile plotting library, and Seaborn is built on top of Matplotlib, offering additional visualization capabilities and a high-level interface for creating attractive and informative statistical graphics. The use of these libraries is a standard practice in the data science and analysis community.

**Histogram of transaction amounts:**

**A screen shot of a computer

Description automatically generated**

This code creates a histogram using Matplotlib (plt) and Seaborn (sns). The plt.figure(figsize=(8,6)) line sets the size of the figure. The sns.histplot(df['amount'], kde=True, bins=30) line generates a histogram of the 'amount' column in the DataFrame (df), including a Kernel Density Estimate (KDE) and defining 30 bins.

The subsequent lines set the title, xlabel, and ylabel for the plot. Finally, plt.show() displays the histogram. This visualization provides insights into the distribution of transaction amounts, showing the frequency of different amounts and the estimated probability density.

A graph of a distribution of transaction amounts

Description automatically generated

**Success rate by country:**

**A screen shot of a computer code

Description automatically generated**

The above code starts by computing the mean success rate for each distinct country in the 'df' DataFrame. Using the 'groupby' function, the data is grouped by country, and then the 'success' column is chosen. Subsequently, the 'mean()' function is applied to determine the average success rate for each group. Following this, a new matplotlib figure is generated with dimensions of 8 inches by 6 inches. The resulting success rates are depicted through a bar plot, with each country represented by a bar and the bars colored sky blue for enhanced clarity. Further customization involves adding a title, "Success Rate by Country," to the plot, along with labeling the x-axis as "Country" and the y-axis as "Success Rate." Finally, the 'show()' function is utilized to present the finalized bar plot. In summary, this code effectively analyzes and visualizes the success rates of various countries, offering a lucid representation of the data.

A graph with blue bars

Description automatically generated

**PSP Usage distribution:**

**A screen shot of a computer code

Description automatically generated**

This code analyzes and depicts the distribution of Payment Service Provider (PSP) usage within a DataFrame denoted as 'df'. In the initial step, the code utilizes the 'value\_counts()' method to compute the frequency of each unique value in the 'PSP' column, generating a Pandas Series named 'psp\_count.' This series serves as a representation of the occurrence frequency of each PSP in the dataset.

Moving on to the subsequent section, a fresh matplotlib figure is instantiated with dimensions set at 8 inches by 6 inches, achieved through the 'plt.figure' function. Following this, the 'psp\_count' Series is employed to construct a bar plot using the 'plot' method, specifying 'kind='bar'' to denote a bar plot. The bars are specifically colored light green to enhance visibility, creating a visual illustration of the distribution of PSP usage.

The final segment introduces customizations to the plot to enhance interpretability. The 'plt.title' function assigns a title to the plot, denoted as "PSP Usage Distribution." The x-axis is labeled as "PSP," representing the distinct payment service providers, while the y-axis is labeled as "Number of Transactions," indicating the frequency of each PSP. Ultimately, the 'plt.show()' function is invoked to exhibit the finalized bar plot, delivering a lucid visual summary of the PSP usage distribution in the dataset. To sum up, this code proficiently computes and visually represents the distribution of transactions across various payment service providers, providing valuable insights into their respective frequencies within the dataset.

A graph of a number of green bars

Description automatically generated with medium confidence

**Correlation matrix to identify potential relationships:**

**A screen shot of a computer

Description automatically generated**

This code creates and visualizes a correlation matrix for the numeric columns within the 'df' DataFrame. Initially, the 'select\_dtypes' method is utilized in the first line to filter and extract solely the numeric columns, specifically those of type 'float64' and 'int64'. This operation results in the creation of a new DataFrame named 'numeric\_cols' containing exclusively numeric data.

Subsequently, the 'corr()' method is applied to the 'numeric\_cols' DataFrame, computing correlation coefficients for all pairs of numeric columns. The resulting correlation matrix is then stored in the variable 'correlation\_matrix'.

In the following steps, a new matplotlib figure is established with dimensions set to 6 inches by 4 inches using 'plt.figure'. The 'heatmap' function from the seaborn library is employed to visually represent the correlation matrix. This heatmap is annotated with correlation values, and the 'coolwarm' color map is applied to differentiate between various levels of correlation. Additionally, grid lines with a linewidth of 0.5 are incorporated to enhance the visual clarity of the plot.

Further customization involves the addition of a title to the plot using 'plt.title', specifying it as "Correlation Matrix." Finally, 'plt.show()' is invoked to showcase the completed heatmap, providing a visual understanding of the correlation relationships existing among the numeric columns in the dataset. In summary, this code furnishes a comprehensive summary of the correlations between numeric variables, aiding in the recognition of patterns and relationships within the data.

A screenshot of a graph

Description automatically generated

**Relationship between variables: ‘amount’ and ‘success’**

**A computer screen shot of a black background

Description automatically generated**

This code generates a statistical visualization to explore the connection between the success of transactions and their corresponding amounts in a DataFrame labeled 'df.' The initial line initializes a new matplotlib figure with dimensions 8 inches by 6 inches, determining the plot's size. Subsequently, the seaborn library's boxplot function is utilized to create a boxplot. The 'x' parameter designates the variable for the x-axis ('success'), while the 'y' parameter specifies the variable for the y-axis ('amount'). The inclusion of 'data=df' ensures that the data is sourced from the given DataFrame. The resulting boxplot visually illustrates the distribution of transaction amounts for both successful and failed transactions, providing insights into potential patterns or distinctions.

The subsequent lines introduce further details to enhance the clarity of the plot. The title function assigns a descriptive title to the plot, characterizing it as the "Relationship between Transaction Amount and Success." Additionally, the x-axis is labeled using xlabel to clarify that '1' represents a successful transaction, whereas '0' represents a failed transaction. The y-axis is labeled as "Amount," indicating the transaction amounts. Finally, the show function is executed to present the finalized boxplot, delivering a clear visual depiction of how transaction amounts are distributed concerning transaction success or failure. In summary, this code facilitates a meaningful exploration of the interplay between success and transaction amounts through a meticulously customized boxplot visualization.

A diagram of a diagram of a success

Description automatically generated with medium confidence

**3D secure transactions:**

**A screen shot of a computer

Description automatically generated**

This code constructs a bar plot examining the influence of 3D security on transaction success within a DataFrame labeled 'df'. The 'barplot' function is employed, specifying the 'x' parameter for the variable on the x-axis ('3D\_secured') and the 'y' parameter for the variable on the y-axis ('success'). The 'palette='pastel'' argument selects a pastel color palette for the bars. The resulting bar plot effectively visualizes the success rates of transactions, distinguishing between those with and without 3D security. Further customization is introduced using the 'title' function, designating the plot as the "Impact of 3D Security on Transaction Success." The y-axis is labeled as "Success Rate," and the x-axis is marked as "3D Secured (0: No, 1: Yes)," ensuring clarity in data representation. Finally, the 'show' function is invoked to present the completed bar plot, offering a distinct visual understanding of how the presence of 3D security aligns with transaction success rates.

A graph of a security system

Description automatically generated with medium confidence

**Success rate based on Card Type:**

**A screen shot of a computer code

Description automatically generated**

This code examines and illustrates success rates based on distinct card types within the DataFrame 'df'. Initially, the code utilizes the 'groupby' function to group the data by the 'card' column and computes the mean success rate for each card type. The outcome is encapsulated in a Pandas Series named 'card\_success\_rates,' representing the average success rate for each unique card type.

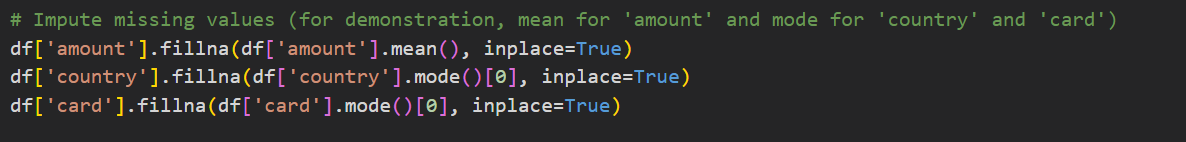
Then a new matplotlib figure is generated with dimensions 10 inches by 6 inches using 'plt.figure'. The 'plot' method is then applied to the 'card\_success\_rates' Series, specifying 'kind='bar'' to craft a bar plot. To enhance visibility and distinction, the bars are colored using a palette of light yellow, light green, and light blue.

Additional customization is introduced to the plot through the 'title' function, designating it as "Success Rate Based on Card Type." The y-axis is labeled as "Success Rate." Moreover, 'plt.xticks(rotation=0)' ensures that the x-axis labels (card types) remain unrotated for improved legibility. Finally, 'plt.show()' is invoked to showcase the finalized bar plot, offering a visual depiction of success rates across various card types in the dataset.

A graph of different colored bars

Description automatically generated

**Data Preparation:**

****

For the 'amount' column, missing values are replaced with the mean of the existing values in that column. The fillna method is used, and the mean value is calculated using df['amount'].mean(). The inplace=True argument ensures that the changes are applied directly to the original DataFrame.

For the 'country' column, missing values are filled with the mode (most frequent value) of the existing values in that column. The mode()[0] extracts the first mode (in case there are multiple modes). Similar to the 'amount' column, the changes are applied in-place.

Similarly, for the 'card' column, missing values are imputed using the mode of the existing values. The mode()[0] ensures the first mode is selected if there are multiple modes. The changes are applied in-place.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

This code conducts a sequence of preprocessing operations on the DataFrame 'df' to ready it for machine learning tasks. Initially, it discards the 'Unnamed: 0' column, presuming it to be an unnecessary identifier. Subsequently, temporal features are derived from the 'tmsp' column, which is converted to datetime format, leading to the creation of new columns for the hour, day, and weekday. Following this, the original 'tmsp' column is removed. In the subsequent stage of preprocessing, categorical columns ('country', 'PSP', 'card') undergo one-hot encoding using the OneHotEncoder. The resultant encoded features are then merged back into the original DataFrame, with the original categorical columns subsequently eliminated.

Continuing the preprocessing pipeline, the numerical 'amount' column undergoes scaling using the StandardScaler to standardize its values. Finally, the dataset undergoes a division into training and testing sets (X\_train, X\_test, y\_train, y\_test) through the train\_test\_split function. This involves allocating 30% of the data for testing, and a specific random state is designated for reproducibility. The resultant DataFrame is printed, providing a visual representation of the impact of these preprocessing steps, encompassing one-hot encoding, feature extraction, and scaling, in preparing the data for machine learning endeavors.

A screenshot of a computer

Description automatically generated

**Baseline model development:**

**Logistic Regression:**

**A screen shot of a computer program

Description automatically generated**

This code utilizes the scikit-learn library to train and assess a logistic regression model on a dataset divided into training and testing sets. It begins by importing necessary libraries, such as pandas for data manipulation and scikit-learn for machine learning functionality. The assumption is made that the dataset has undergone preprocessing, featuring 'X\_train' for training features and 'y\_train' for corresponding labels, as well as analogous sets ('X\_test' and 'y\_test') for evaluation purposes.

The script proceeds to instantiate a logistic regression model with a maximum iteration limit of 1000, training it through the 'fit' method on the provided training data. Subsequent predictions are generated on the test set using the 'predict' method, and the model's accuracy is computed using the 'accuracy\_score' function from scikit-learn. The resulting success rate is then printed as a percentage.

Beyond model evaluation, the script calculates total fees by applying a custom 'calculate\_fees' function to each row in the DataFrame 'df' using the 'apply' method with 'axis=1'. The accumulated fees are summed and presented, offering insights into the cumulative financial impact derived from the calculated fees for each individual data point.

A black rectangle with white text

Description automatically generated

**Random Forest Model:**

**A computer screen with colorful text

Description automatically generated**

This code leverages the scikit-learn library to deploy a Random Forest Classifier for a machine learning objective. Initially, the code imports essential modules, including RandomForestClassifier for constructing the model and accuracy\_score for evaluating its effectiveness. The RandomForestClassifier is configured with specific parameters, specifying 100 decision trees (n\_estimators=100), no maximum depth constraint (max\_depth=None), and a fixed random state (random\_state=0) for reproducibility.

Following initialization, the model is trained on the provided training data using the 'fit' method, employing 'X\_train' as the feature set and 'y\_train' as the corresponding labels. Subsequent predictions are generated on the test set via the 'predict' method, and the model's success rate is computed by comparing the predicted labels ('y\_pred\_rf') with the actual labels ('y\_test'). The resulting success rate, denoting the accuracy of the Random Forest Classifier on the test data, is then printed as a percentage with three decimal places to ensure precision. This code offers a succinct and efficient implementation of a Random Forest model tailored for classification tasks, with the success rate serving as a metric to assess its predictive performance.

We obtained a Success rate of 76.843% and hence we would like to improve the model further.

A black rectangle with white text

Description automatically generated

**Improving Random Forest Model:**

**A screen shot of a computer program

Description automatically generated**

This code utilizes the scikit-learn library to conduct hyperparameter tuning for a Random Forest Classifier employing RandomizedSearchCV. In the initial section, a dictionary named param\_distributions is crafted to encompass diverse hyperparameter values, encompassing aspects like the number of trees in the forest (n\_estimators), maximum depth of the trees (max\_depth), minimum samples necessary to split an internal node (min\_samples\_split), and the minimum number of samples required to form a leaf node (min\_samples\_leaf).

Subsequently, a RandomizedSearchCV object, denoted as random\_search, is instantiated, incorporating the Random Forest Classifier (rf), the predefined hyperparameter distributions, and specific configurations, such as the number of iterations (n\_iter=10), cross-validation folds (cv=5), and the chosen scoring metric (scoring='accuracy'). The fit method is then applied to execute a randomized search across the hyperparameter space using the provided training data (X\_train and y\_train).

The code retrieves and displays the optimal hyperparameters identified during the search, stored within the best\_params variable. Subsequently, the Random Forest model is assessed using these fine-tuned parameters on the test set (X\_test). The model's predictions (y\_pred\_best\_rf) are juxtaposed with the actual labels (y\_test), and the accuracy of the optimized Random Forest model is computed through the accuracy\_score function. The resultant accuracy is presented as a percentage, offering a transparent representation of the performance enhancement achieved through the strategic exploration of hyperparameter combinations. Collectively, this code illustrates a streamlined methodology for augmenting the predictive accuracy of the Random Forest model via the investigation of diverse hyperparameter configurations.

# 8. Conclusion

As we conclude this phase of using data-driven insights to enhance our transactional achievements, it is crucial to contemplate the voyage we undertook and the expanded perspectives we have gained. The primary objective of our undertaking was not just focused on the integration of a model into our business pipeline. Instead, our aim was to establish a culture that prioritizes data-conscious decision-making, so transforming each transaction from a routine procedure into a well-informed strategic action.

Commencing with a thorough analysis of the data, we conducted an in-depth examination of our transactional patterns, aiming to comprehend not just the surface-level trends but also the underlying complexities of human behavior. The establishment of this foundation laid the groundwork for our fundamental model, serving as evidence of the capacity of data to forecast results. Although our early findings were praiseworthy, our pursuit of greatness compelled us to question the established standard and go into the extensive realms of predictive modeling. Throughout several rounds, ranging from logistic regression to random forests, we have seen the profound impact of fine-tuning, demonstrating that achieving perfection is truly a gradual and iterative endeavor.

Nevertheless, the evaluation of a model's effectiveness should not just rely on its predictive accuracy, but also on its concrete influence on the business environment. Upon careful consideration of its daily operational integration, it became apparent that a data-driven model had the ability to serve as a guiding force, leading to not just transactional success but also improved operational efficiency, cost reduction, and an enhanced experience for stakeholders. Our suggested integration approach encompasses smooth API connections and intuitive GUI interfaces, which exemplify our dedication to transforming technology into a facilitator that is accessible and comprehensible to all individuals.

However, it is essential for any technology intervention, regardless of its level of complexity, to be firmly grounded in ongoing learning and development. Our methodology, which prioritizes regular assessments, meticulous mistake analysis, and stakeholder education, underscores the principle of continuous development.

# 9. Code copied for reference

#Import necessary libraries

import pandas as pd

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

#Load data

df = pd.read\_excel('/Users/vharibab/Desktop/PSP\_Jan\_Feb\_2019.xlsx')

df = pd.DataFrame(df)

#Perform Data Quality Assessment

# Check for missing values|

print('this is our dataset : \n', df)

print("Missing values for each column:")

print(df.isnull().sum())

Q1 = df['amount'].quantile(0.25)

Q3 = df['amount'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

outliers = df[(df['amount'] < lower\_bound) | (df['amount'] > upper\_bound)]

print("\nNumber of outliers in 'amount':", len(outliers))

#Perform Exploratory Data Analysis

# Import necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

#Histogram of transaction amounts

plt.figure(figsize=(8,6))

sns.histplot(df['amount'], kde=True, bins=30)

plt.title("Distribution of Transaction Amounts")

plt.xlabel("Amount")

plt.ylabel("Frequency")

plt.show()

#Success rate by country

success\_rate = df.groupby('country')['success'].mean()

plt.figure(figsize=(8,6))

success\_rate.plot(kind='bar', color='skyblue')

plt.title("Success Rate by Country")

plt.xlabel("Country")

plt.ylabel("Success Rate")

plt.show()

#PSP Usage Distribution

psp\_count = df['PSP'].value\_counts()

plt.figure(figsize=(8,6))

psp\_count.plot(kind='bar', color='lightgreen')

plt.title("PSP Usage Distribution")

plt.xlabel("PSP")

plt.ylabel("Number of Transactions")

plt.show()

#correlation matrix to identify potential relationships

numeric\_cols = df.select\_dtypes(include=['float64', 'int64']) # Selecting only numeric columns

correlation\_matrix = numeric\_cols.corr()

plt.figure(figsize=(6,4))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title("Correlation Matrix")

plt.show()

#Relationships between variables: ‘amount’ and ‘success’

plt.figure(figsize=(8,6))

sns.boxplot(x='success', y='amount', data=df)

plt.title("Relationship between Transaction Amount and Success")

plt.xlabel("Success (1 = successful transaction, 0 = failed transaction)")

plt.ylabel("Amount")

plt.show()

#3D Secure Transactions

sns.barplot(x=df['3D\_secured'], y=df['success'], palette='pastel')

plt.title('Impact of 3D Security on Transaction Success')

plt.ylabel('Success Rate')

plt.xlabel('3D Secured (0: No, 1: Yes)')

plt.show()

#Success Rate based on card type

card\_success\_rates = df.groupby('card')['success'].mean()

plt.figure(figsize=(10,6))

card\_success\_rates.plot(kind='bar', color=['Lightyellow', 'lightgreen', 'lightblue'])

plt.title('Success Rate Based on Card Type')

plt.ylabel('Success Rate')

plt.xticks(rotation=0)

plt.show()

#Data preparation

# Impute missing values (for demonstration, mean for 'amount' and mode for 'country' and 'card')

df['amount'].fillna(df['amount'].mean(), inplace=True)

df['country'].fillna(df['country'].mode()[0], inplace=True)

df['card'].fillna(df['card'].mode()[0], inplace=True)

import pandas as pd

from sklearn.preprocessing import OneHotEncoder, StandardScaler

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

df = df.drop(columns=['Unnamed: 0'])

# Extracting features from 'tmsp' column

df['tmsp'] = pd.to\_datetime(df['tmsp'])

df['hour'] = df['tmsp'].dt.hour

df['day'] = df['tmsp'].dt.day

df['weekday'] = df['tmsp'].dt.weekday

# Drop original 'tmsp' column

df = df.drop(columns=['tmsp'])

# One-hot encoding categorical columns

encoder = OneHotEncoder()

encoded\_features = encoder.fit\_transform(df[['country', 'PSP', 'card']])

encoded\_df = pd.DataFrame(encoded\_features.toarray(), columns=encoder.get\_feature\_names\_out(['country', 'PSP', 'card']))

# Merging one-hot encoded columns with original df

df = df.join(encoded\_df)

# Drop original categorical columns

df = df.drop(columns=['country', 'PSP', 'card'])

# Scaling the 'amount' column

scaler = StandardScaler()

df['amount'] = scaler.fit\_transform(df[['amount']])

# Train/test split

X = df.drop(columns=['success'])

y = df['success']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

print (df)

#Baseline model development

#Logistic Regression

import pandas as pd

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# Train a logistic regression model

model = LogisticRegression(max\_iter=1000)

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

# Predict on test set

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

# Calculate success rate

success\_rate = accuracy\_score(y\_test, y\_pred)

print(f"Success Rate: {success\_rate \* 100:.2f}%")

total\_fees = df.apply(calculate\_fees, axis=1).sum()

print(f"Total Fees: ${total\_fees:.2f}")

#Random Forest Model

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Initialize the Random Forest Classifier

rf = RandomForestClassifier(n\_estimators=100, max\_depth=None, random\_state=0)

# Fit the model on the training data

rf.fit(X\_train, y\_train)

# Predict on the test set

y\_pred\_rf = rf.predict(X\_test)

# Calculate success rate

success\_rate = accuracy\_score(y\_test, y\_pred\_rf)

print(f"Success Rate: {success\_rate \* 100:.3f}%")

#Improving Random Forest model

from sklearn.model\_selection import RandomizedSearchCV

param\_distributions = {

'n\_estimators': [50, 100, 200],

'max\_depth': [None, 10, 20, 30],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

random\_search = RandomizedSearchCV(rf, param\_distributions, n\_iter=10, cv=5, scoring='accuracy')

random\_search.fit(X\_train, y\_train)

best\_params = random\_search.best\_params\_

print("Best parameters found: ", best\_params)

# Evaluate the model with best parameters

best\_rf = random\_search.best\_estimator\_

y\_pred\_best\_rf = best\_rf.predict(X\_test)

accuracy\_best\_rf = accuracy\_score(y\_test, y\_pred\_best\_rf) \* 100

print(f"Optimized Random Forest Accuracy: {accuracy\_best\_rf:.2f}%")

# References

Hotz, N. (2023, January 19). ***What is CRISP DM****? - Data Science Process Alliance*. Data Science Process Alliance. https://www.datascience-pm.com/crisp-dm-2/

Jung, K., Kashyap, S., Avati, A., Harman, S., Shaw, H., Li, R., Smith, M., Shum, K., Javitz, J., Vetteth, Y., Seto, T., Bagley, S. C., & Shah, N. H. (2020, December 22). ***A framework for making predictive models useful in practice*.** Journal of the American Medical Informatics Association; Oxford University Press. <https://doi.org/10.1093/jamia/ocaa318>

Chaffey, D., & Ellis-Chadwick, F. (2019, February 5). ***Digital Marketing***. Pearson UK.

***sklearn.ensemble.RandomForestClassifier*. (n.d**.). Scikit-learn. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

Kane, G. C., & Euchner, J. (2021, October 22). **Leading Digital Transformati**on. *Research-Technology Management*, *64*(6), 11–16. https://doi.org/10.1080/08956308.2021.1974764