The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Detecting Financial Anomalies with Machine Learning: Insights from Lloyds Bank Data

by

Aman Syed, S2496727

Dissertation Presented for the Degree of
MSc in Statistics with Data Science

July 2024

Supervised by
Dr Tim Cannings, Dr Cecilia Balocchi and Johnny Lee
School of Mathematics, University of Edinburgh
Callum Hodgkinson, Dimitros Ntakoulas and George Deskas, Lloyd's Bank

Executive Summary

The increasing complexity and volume of financial transactions necessitate advanced methods for anomaly detection to ensure the integrity and security of financial systems. This consultancy project aimed to develop an unsupervised anomaly detection system for Lloyds Bank to identify risk events and behavioral anomalies in transaction data. Advanced machine learning techniques, including Isolation Forests, Local Outlier Factor (LOF), DBSCAN, One-Class SVM and Autoencoders, were employed, with a primary focus on model 1 for risk event detection due to its critical significance. The Isolation Forest algorithm emerged as the most effective method, achieving a balance between precision and recall despite some precision challenges. This model demonstrated substantial capability in anomaly detection, offering a robust framework for real-time risk management. Recommendations include integrating ensemble methods and exploring advanced interpretability techniques such as SHAP and LIME to enhance understanding and decision-making. Future work should concentrate on refining model accuracy and expanding the analysis to include model 2 for a comprehensive risk assessment strategy. These implementations will significantly enhance the bank's ability to preemptively manage potential risk events, ensuring the security of financial operations and maintaining customer trust.

Acknowledgments

I would like to express my deepest gratitude to Dr. Tim Cannings, Dr. Cecilia Balocchi, and Johnny Lee from the School of Mathematics at the University of Edinburgh. Their unwavering support, guidance, and invaluable advice have been instrumental throughout this project. My sincere thanks also go to Callum Hodgkinson, Dimitros Ntakoulas, and George Deskas from Lloyds Bank for their collaboration, insights, and provision of essential data, which significantly contributed to the success of this project. Your collective expertise and support have been fundamental in the completion of this dissertation.

I would also like to extend my appreciation to my friends and family for their constant encouragement and understanding during this journey. Their support has been a cornerstone in achieving this milestone.

University of Edinburgh – Own Work Declaration

This sheet must be filled in, signed and dated - your work will not be marked unless this is done.

Name: Aman Syed
Matriculation Number: S2496727
Title of work: Detecting Financial Anomalies with Machine Learning: Insights from Lloyds Bank Data

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Not sought or used the help of any external professional academic agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course handbook

I understand that any false claim for this work will be penalised in accordance with the University regulations (`https://teaching.maths.ed.ac.uk/main/msc-students/msc-programmes/statistics/data-science/assessment/academic-misconduct`).

Signature

Date
28/07/2024

Contents

1 Introduction **1**
- 1.1 Background and Motivation . . . 1
- 1.2 Objectives of the Project . . . 1

2 Literature Review **1**
- 2.1 Review of Relevant Literature . . . 1
- 2.2 Theoretical Background . . . 2
- 2.3 Existing Methods and Models . . . 2

3 Exploratory Data Analysis **2**
- 3.1 Preprocessing Step . . . 3
- 3.2 Data Visualization . . . 3
- 3.3 Data Initial Findings . . . 6

4 Methodology **6**
- 4.1 Overview of the Methodological Approach . . . 6
- 4.2 Description of Models and Algorithms Used . . . 7
- 4.3 Feature Engineering and Selection . . . 8

5 Implementation and Experimentation **9**
- 5.1 Hyperparameter Tuning . . . 10

6 Results and Discussion **10**
- 6.1 Results of Each Model . . . 10
- 6.2 Comparison of Models . . . 12
- 6.3 Interpretation of Results . . . 13
- 6.4 Limitations of the Analysis . . . 13

7 Conclusion **13**

8 Recommendations **14**
- 8.1 Recommendations for Industrial Partner . . . 14
- 8.2 Suggestions for Future Work . . . 14

Appendices **17**

A Data Characteristics **17**

B Data Cleaning **17**

C Data Preprocessing **17**

D Model Training **18**

E Model Evaluation Metrics **18**

F Code **18**
- F.1 Data Cleaning and Preprocessing . . . 18
- F.2 Data Visualization . . . 20
- F.3 Model 1: At Risk Event . . . 21
- F.4 Model 2: At Risk Behaviour Window . . . 23

List of Tables

1 New Features Summary 9
2 Hyperparameters Summary 10
3 Performance Metrics Comparison 12

List of Figures

1 Transaction Amount Distribution 3
2 Transactions by Hour of the Day 4
3 Spend by Department 4
4 Risk Events by Time of Day 4
5 Hourly Risk Events Analysis – Line Plot 5
6 Hourly Risk Events Analysis – Bar Plot 5
7 Departmental Risk Events 5
8 Flowchart of Methodological Approach 6
9 Working of Isolation Forest 7

1 Introduction

1.1 Background and Motivation

Financial institutions, such as Lloyds Bank, face numerous fraud threats, unauthorized transactions, and other anomalies that could lead to immense financial risks. Traditionally, manual control and monitoring have been used for risk detection, but this approach is costly in terms of labor and is also open to human error. Automated and reliable risk identification systems are, therefore, more crucial now than ever.

Machine learning and data science have some exceptional tools for solving the problem of anomaly detection with financial transactions. Anomaly detection is of utmost importance to uncover atypical patterns that would give hints about fraudulent activities. The project harnesses advanced transitioning from manual to automatic risk detection techniques, augmenting operational efficiency and accuracy.

Automated systems offer immense advantages: they can process large amounts of data in small periods, provide real-time insights, and allow financial institutions to free up human resources for more strategic work. The critical scope of this project is to build models capable of detecting risk events and risky behavior windows within transactional data.

1.2 Objectives of the Project

The primary aim of the consultancy project is to develop an unsupervised anomaly detection system tailored for Lloyds Bank transactional data. Emphasis will be put on the application and comparative evaluation of several machine learning techniques for optimal identification of risk events and behavioral anomalies. Specifically, the project aims to:

1. Analyze the structure and characteristics of the transactional data.
2. Develop and implement various models of unsupervised learning: Isolation Forest, LOF, DBSCAN, One-Class SVM, and Autoencoders.
3. Compare the performances of these models against each other and decide which is the most effective approach.
4. Derive actionable insights and recommendations based on which model performs better and have it integrated into the Lloyds Bank Risk Management framework.

The project will strengthen the performance of Lloyds Bank in safeguarding its financial operations and maintaining customers' trust through proactive management of possible risks in the dynamic economic environment with automatic detection of risks in real-time.

2 Literature Review

2.1 Review of Relevant Literature

Anomaly detection, especially within financial transaction data, has been extensively researched due to its significance in identifying fraudulent activities and, hence, mitigating risks. Much effort from researchers has been made in developing complex models that will work in anomaly detection. Chandola et al. (2009) review techniques that apply in detecting anomalies, with much emphasis on their applicability in different domains, especially in fraud detection in financial transactions. For instance, Zhang et al. (2019) highlighted the machine learning developments in models designed for anomaly detection—for example, both supervised and unsupervised methods, including their relative merits and challenges.

Recent studies, for instance, Akoglu et al. (2015), have used methods of deep learning and ensemble techniques in anomaly detection, demonstrating their ability to improve detection rates. The rise of explainable AI has also propelled the development of models that can not only detect anomalies but also explain their nature and origin, as discussed by Ribeiro et al. (2016).

2.2 Theoretical Background

The approach to anomaly detection is mainly based on statistics and identifying data points that deviate radically from the masses. For example, deviation can be measured using Z-scores or distance metrics in multi-dimensional space. The basic idea is that anomalies are rare and therefore differ from standard patterns. Techniques like the Isolation Forests developed by Liu et al. (2008) proceed from the argument that anomalies will be more efficiently isolated in a feature space and thus set apart from regular instances. The algorithms of Isolation Forest build random forests for anomaly isolation by recursively partitioning data points, where very few iterations are needed for partitioning the anomalies since noticeable deviations isolate them.

Another method that has been used to study the effectiveness of anomaly detection techniques is One-Class SVM (Support Vector Machine) described by Schölkopf et al. (2001), by which a decision function for classification of data as similar or different (i.e., outliers) is learned. Thus, this technique is pretty well applied in fraud detection scenarios for high-dimensional datasets.

2.3 Existing Methods and Models

Several models and algorithms have been proposed and implemented for anomaly detection in financial datasets. These include:

Isolation Forest (IF): IF has gained a noticeable reputation due to its efficiency in high-dimensional spaces. For instance, Hariri et al. (2019) found the models using IF particularly appropriate for unsupervised anomaly detection because they do not make assumptions about the underlying data distribution and are easily applied even to large datasets.

Local Outlier Factor (LOF): An anomaly detection method that determines anomalies based on the local density of a point compared to its neighbors in feature space. Breunig et al. (2000) showed that LOF can effectively capture local density deviations and, thus, be good at detecting contextual anomalies.

DBSCAN: Stands for density-based spatial clustering of applications with noise. This technique works well for random-shaped clusters and is noise-robust.

Autoencoders: This is a type of neural network developed for unsupervised learning, used in anomaly detection by training with input data, while anomalies cause high reconstruction errors. An and Cho (2015) noted that since anomalies differ from standard data, they lead to higher reconstruction errors.

One-Class SVM: Normally, this is used to perform anomaly detection by learning a decision function to novel data points. It is highly effective in high-dimensional spaces and forms an essential base for this project.

The combination of these methods is supposed to carry out a good framework for reliability, accuracy, and anomaly detection.

3 Exploratory Data Analysis

The dataset used for this project was sourced from Lloyds Bank, containing time series transaction records for 2,185 individuals across 20 different departments. Each individual has transactions recorded at different time points over 4 months, making this a longitudinal dataset. The data is multivariate, capturing various aspects of each transaction such as the timestamp, transaction amount, department, and whether the transaction was flagged as a risk event.

3.1 Preprocessing Step

1. **Timestamp Conversion**: The timestamp column was converted into total seconds to facilitate time-based calculations.
2. **Date Conversion**: The date column was converted to a datetime format to extract additional time-based features.
3. **Feature Extraction**: Additional features were extracted from the date and timestamp, including:
    - Month, quarter, and day of the week.
    - Whether the transaction occurred on a weekend.
    - Whether the transaction occurred during office hours.
4. **Rolling Statistics**: Generated rolling averages and sums for transaction amounts.
5. **Time Differences**: Calculated the time difference between consecutive transactions for each individual.
6. **Frequency Metrics**: Calculated the transaction frequency in the last 7 days and its deviation from the mean.
7. **Spend Ratios**: Computed the ratio of the spend to the average spend of each individual.
8. **Outlier Detection**: Used Z-scores to identify and cap outliers in the spend data.

The dataset was then cleaned by handling missing values, normalizing continuous variables, and encoding categorical variables.

3.2 Data Visualization

Exploratory Data Analysis (EDA) was performed to develop insights into the distribution of transaction amounts, the frequency of transactions at different times during the day, and the behavior of transactions across departments. The plots created during EDA are as follows:

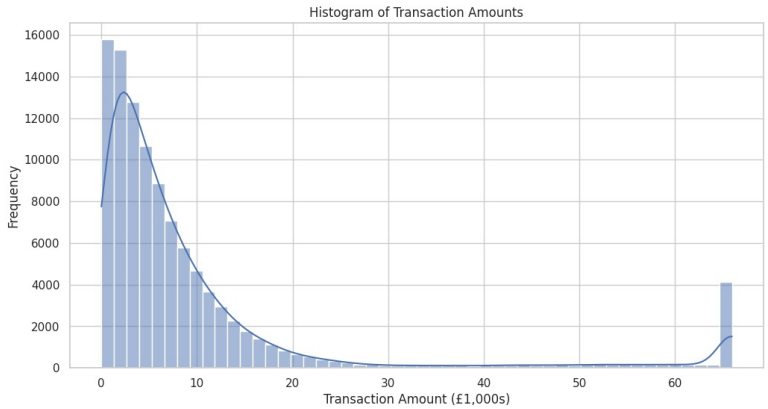

Figure 1: Transaction Amount Distribution

By observing the histogram (1), it is observed that the transaction amount has a right-skewed distribution, where most transactions have relatively small amounts, and there are only a few that are prominent; this could point to the existence of outliers. Most transactions cluster at lower amounts; however, there is a hint of a large number of high-value transactions.

This bar plot below (2) shows that transactions have most dominantly occurred during working hours, from about 8 AM to 6 PM. There are two visible peaks around midday, reflecting regular business activities. Very few transactions are seen throughout the early morning and late night.

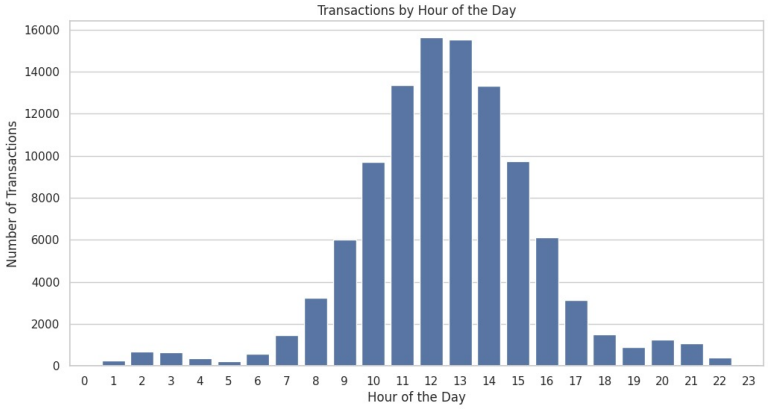

Figure 2: Transactions by Hour of the Day

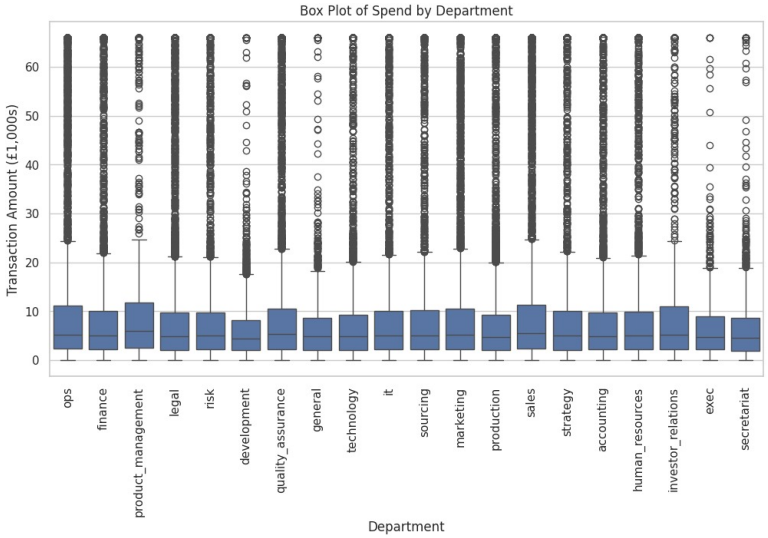

Figure 3: Spend by Department

The box plot (3) describes how different departments have a spread in the transaction amount. Some departments have more spending, which probably is because of the operational nature of those departments. It identifies which departments have higher and more volatile expenditure.

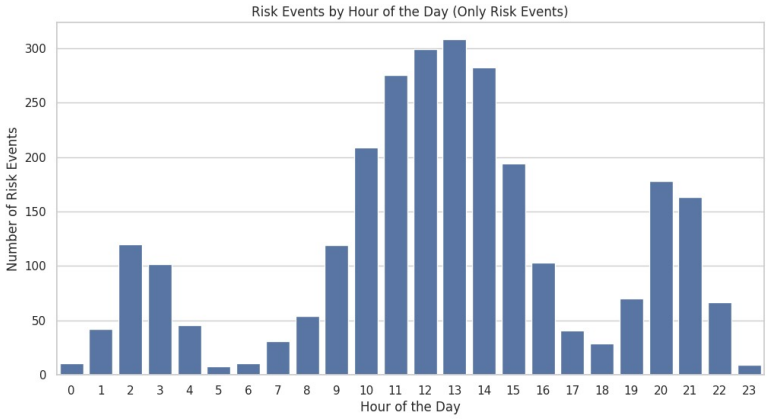

Figure 4: Risk Events by Time of Day

Analysis of risk events (4) depicts the transactions tagged as risk events at different hours of the day. This shows that most risk events happen through business hours and group around midday. Indicatively, this pattern leads one to the assumption that it is an indication of risk events within

high-transaction periods.

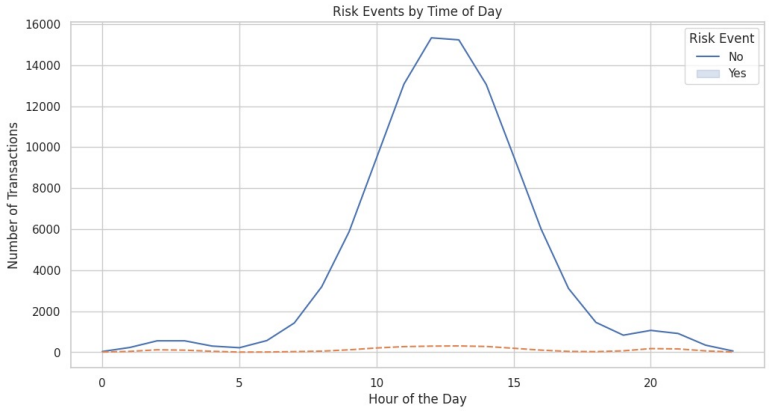

Figure 5: Hourly Risk Events Analysis – Line Plot

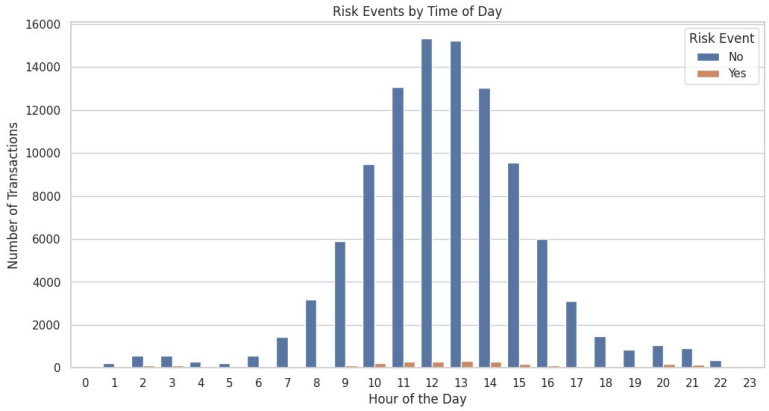

Figure 6: Hourly Risk Events Analysis – Bar Plot

The hourly risk event analysis is shown in 5 and 6. The data supports that risk events are sporadic but seem to have a specific synchronization with the general transaction trend.

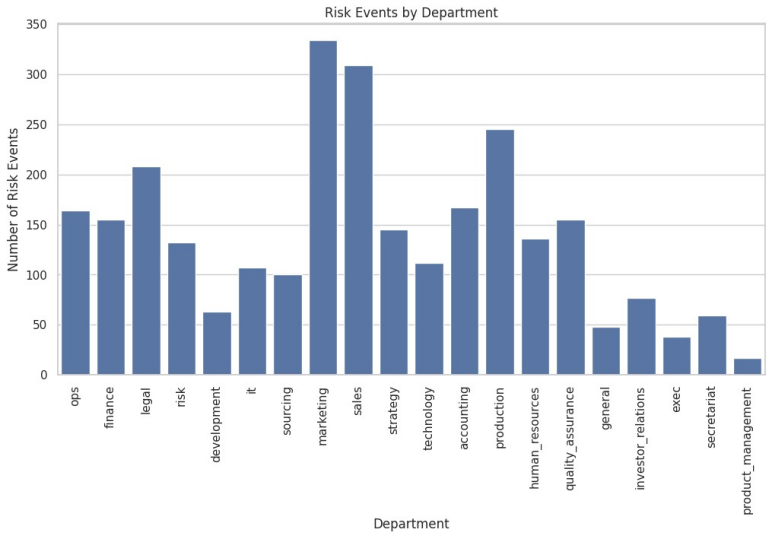

Figure 7: Departmental Risk Events

The bar plot (7) displays the count of risk events by the department and allows us to see where the frequency of risk events is higher. It also reveals that the risk frequency tends to be larger in

departments, particularly in marketing and sales, possibly because their activities are characterized by bigger and more frequent transactions.

3.3 Data Initial Findings

From the exploratory data analysis, several key findings emerged:

1. The distribution of transaction amounts is highly skewed, with most transactions being small but a few large outliers.
2. Transaction activity is concentrated during business hours, peaking around midday.
3. Transactions outside of these hours are less frequent but might be more significant in terms of risk.
4. Different departments exhibit varied spending behaviors, with some departments having consistently higher transaction amounts.
5. Certain departments exhibit higher spending and a greater number of risk events, indicating potential areas for further investigation.
6. Risk events follow a similar temporal pattern to regular transactions but are relatively infrequent.

These initial insights provide a foundation for developing models to detect at-risk events and at-risk behavior windows, guiding the feature engineering and selection process. This detailed initial analysis sets the stage for the subsequent modeling efforts, ensuring that the data is well-understood and appropriately preprocessed for accurate and effective risk detection.

4 Methodology

4.1 Overview of the Methodological Approach

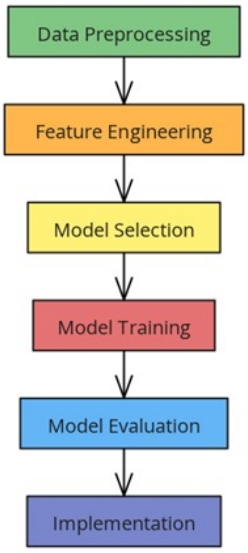

Figure 8: Flowchart of Methodological Approach

The primary objective of this project is to develop an effective unsupervised model capable of detecting risk events and identifying risk behaviour windows in transaction data. The methodological approach adopted in this study comprises several critical steps, including data preprocessing, feature engineering, and the application of various anomaly detection algorithms. The focus is predominantly on Model 1, which targets risk events, while Model 2 addresses the risk behaviour window. The chosen

methods include Isolation Forest, Autoencoders, Support Vector Machines (SVM), Local Outlier Factor (LOF), and DBSCAN Clustering, each serving to enhance the detection of anomalies in different contexts. Both models relied heavily on feature engineering to extract meaningful patterns from the raw transactional data.

A visual representation of the methodological approach can be seen in the figure(9) above. This comprehensive approach ensures that the anomaly detection models are well-prepared to identify and flag risk events effectively, leveraging a variety of engineered features and state-of-the-art machine learning algorithms.

4.2 Description of Models and Algorithms Used

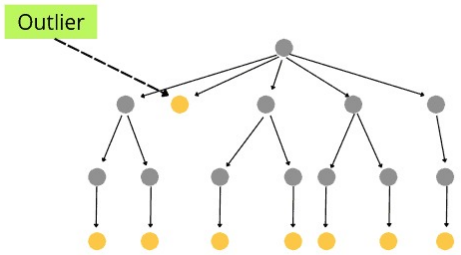

Figure 9: Working of Isolation Forest

Isolation Forest

- **Algorithm Description**: Isolation Forest is an ensemble algorithm specifically designed for anomaly detection. It works by isolating observations through random partitioning of data. The premise is that anomalies are few and different, making them easier to isolate.
- **Implementation**: I implemented the Isolation Forest using the scikit-learn library, with specific parameters tailored to maximize detection accuracy. The contamination parameter, which defines the proportion of anomalies, was adjusted to various levels to understand its impact.
- **Key parameters**:
  - *n_estimators*: Number of trees in the forest.
  - *max_samples*: The number of samples to draw from the dataset to train each base estimator.
  - *contamination*: The proportion of outliers in the dataset.
  - *max_features*: The number of features to draw from the dataset to train each base estimator.

Local Outlier Factor (LOF)

- **Algorithm Description**: LOF measures the local density deviation of a given data point with respect to its neighbors. It identifies anomalies as points that have a significantly lower density than their neighbors.
- **Implementation**: The LOF model was implemented using the scikit-learn library with a contamination level to specify the expected proportion of anomalies.
- **Key parameters**:
  - *n_neighbors*: Number of neighbors to use for k-neighbors queries.
  - *contamination*: The proportion of outliers in the dataset.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- **Algorithm Description**: DBSCAN is a density-based clustering algorithm that treats points laying in low-density regions as outliers. It is applicable and practical for finding anomalies in large datasets that may have densities of irregular shapes.

- **Implementation**: DBSCAN was applied with parameters optimized for the transaction data's characteristics.
- **Key parameters**:
  - *eps*: The maximum distance between two samples for one to be considered as in the neighborhood of the other.
  - *min_samples*: The number of samples in a neighborhood for a point to be considered as a core point.

Autoencoders

- **Algorithm Description**: An autoencoder is a neural network that learns to code data effectively. During anomaly detection, an autoencoder is trained on everyday transactions to learn how to reconstruct the input, while the anomalies cause significant errors in the reconstructed transactions.
- **Implementation**: The autoencoder was built using TensorFlow and Keras, focusing on reducing the Mean Squared Error (MSE) between the input and reconstruction as the primary loss function.
- **Key parameters**:
  - *input_dim*: The number of input features.
  - *encoding_dim*: The dimension of the encoded representation.

One-Class SVM (Support Vector Machine)

- **Algorithm Description**: One-Class SVM is used for novelty detection. It learns a decision function for anomaly detection and classifies new data as similar or different.
- **Implementation**: The One-Class SVM was applied using the radial basis function (RBF) kernel, which is effective in high-dimensional spaces.
- **Key parameters**:
  - *kernel*: Specifies the kernel type to be used in the algorithm.
  - *gamma*: Kernel coefficient.
  - *nu*: An upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors.

4.3 Feature Engineering and Selection

Feature engineering is crucial in preparing data for modeling, as it involves creating new features or modifying existing ones to enhance machine learning model performance. For this project, the original dataset was enriched with various derived features to improve anomaly detection models. Several features were engineered to capture different aspects of transaction behavior, including:

1. **Temporal Features**: Extracting the hour of the day, day of the week, and determining if the transaction occurred during office hours or on weekends.
2. **Rolling Statistics**: Calculating rolling means and sums of spending to capture short-term spending patterns, such as over 3-day periods.
3. **Frequency Features**: Measuring the frequency of transactions over various time windows, like the last 7 days and the last 30 days.
4. **Transaction Ratios and Deviations**: Computing ratios of spend to individual means and deviations from rolling averages to highlight outliers.

5. **Deviation Metrics**: Calculating deviations from average spending to identify unusual spikes or drops.

6. **Cumulative Features**: Tracking cumulative spend per individual and the differences in spend from previous transactions.

7. **Department-wise Statistics**: Including mean spending per department to account for variations across different organizational units.

Feature	Description
hour_of_day	Hour of the transaction
day_of_week	Day of the week the transaction occurred
is_weekend	Indicator if the transaction was on a weekend
is_office_hours	Indicator if the transaction was during office hours
rolling_spend_mean	Rolling mean of spending
rolling_spend_sum	Rolling sum of spending
time_diff	Time difference between consecutive transactions
spend_to_mean_ratio	Ratio of spending to the average spend
spend_deviation	Deviation from the average spending
transactions_last_7_days	Number of transactions in the last 7 days
transactions_freq_deviation	Deviation in transaction frequency from the mean
mean_spend_department	Mean spending in the department
cumulative_spend	Cumulative spending per individual
spend_diff	Difference in spend from the previous transaction

Table 1: New Features Summary

5 Implementation and Experimentation

The implementation of the anomaly detection system for Lloyds Bank involved using various machine learning libraries and frameworks to ensure robust model development and evaluation. Key libraries included scikit-learn for traditional machine learning models, TensorFlow and Keras for deep learning models, and pandas and NumPy for data manipulation and preprocessing. The project followed a systematic approach to handle different implementation phases, from data preprocessing and feature engineering to model training and evaluation.

Libraries Used

- **scikit-learn**: For implementing Isolation Forest, Local Outlier Factor (LOF), One-Class SVM, and DBSCAN.
- **TensorFlow and Keras**: For building and training the Autoencoder model.
- **pandas and NumPy**: For data manipulation, cleaning, and feature engineering.
- **Matplotlib and Seaborn**: For data visualization and exploratory data analysis.

Computing Environment

- **Hardware**: The experiments were conducted on a high-performance workstation with an Intel Core i5 processor, 16 GB RAM, and an Integrated GPU.
- **Software**: The development environment included Python 3.8, Jupyter Notebooks for interactive coding, and Git for version control.

5.1 Hyperparameter Tuning

Hyperparameter tuning was a critical aspect, aimed at enhancing model performance. Techniques such as Grid Search and Random Search were employed to systematically explore different combinations of hyperparameters. Here, the Isolation Forest's contamination parameter was fine-tuned to balance sensitivity to anomalies with the rate of false positives. The following table summarizes the tuned hyperparameters and their respective values for each model:

Model	Hyperparameters
Isolation Forest	n_estimators=500, max_samples=0.8, contamination=0.025
Local Outlier Factor (LOF)	n_neighbors=20, contamination=0.025
DBSCAN	eps=0.2, min_samples=1000
Autoencoder	encoding_dim=3, epochs=50, batch_size=32
One-Class SVM	kernel='rbf', gamma=0.001, nu=0.1

Table 2: Hyperparameters Summary

This rigorous process of hyperparameter tuning significantly improved the performance and reliability of the models, ensuring accurate detection of anomalies within the transaction data.

Challenges Faced

- **Data Imbalance**: The dataset had a significant imbalance between normal and anomalous transactions, necessitating techniques like oversampling and undersampling.
- **Computational Resources**: Training deep learning models such as Autoencoders required substantial computational power, highlighting the importance of efficient resource allocation.

6 Results and Discussion

6.1 Results of Each Model

The core focus of this project was to detect risk events using unsupervised learning models, particularly for two types of risks: risk events (Model 1) and risk behaviour windows (Model 2). The primary emphasis was placed on Model 1, which used several algorithms to identify anomalies in transaction data.

Model 1: Risk Event Detection
For Model 1, I implemented various anomaly detection algorithms, including Isolation Forest, Local Outlier Factor (LOF), DBSCAN, Autoencoders, and One-Class SVM. The features used included timestamp_seconds, month, quarter, day_of_week, is_weekend, is_office_hours, rolling_spend_mean, rolling_spend_sum, time_diff, is_first_transaction, spend_to_mean_ratio, spend_deviation, and transactions_freq_deviation.
Below are the detailed results of each model:

1. **Isolation Forest**: The Isolation Forest algorithm was employed due to its effectiveness in detecting anomalies in high-dimensional data. Here are the key results:

- Precision: 0.207
- Recall: 0.409
- F1-Score: 0.276
- Accuracy: 94%

The model identified anomalies with a precision of 20.7%, a recall of 40.9%, and an F1-score of 27.6%. This indicates that while the model could identify a reasonable number of true anomalies, it also had

a significant number of false positives.

2. **Local Outlier Factor (LOF)**: LOF was used to identify anomalies based on local density deviations. The results were as follows:

- Precision: 0.133
- Recall: 0.127
- F1-Score: 0.130
- Accuracy: 96%

LOF showed a lower performance compared to Isolation Forest, with a precision of 13.3% and a recall of 12.7%, leading to an F1-score of 13%.

3. **DBSCAN**: DBSCAN, a density-based clustering algorithm, was applied to detect outliers. The results were:

- Precision: 0.026
- Recall: 1.000
- F1-Score: 0.051
- Accuracy: 3%

DBSCAN identified nearly all instances as anomalies, resulting in a high recall but extremely low precision, which made it impractical for effective anomaly detection in this context.

4. **Autoencoders**: Autoencoders, which are neural networks used for unsupervised learning of efficient codings, were trained to detect anomalies. The results were:

- Precision: 0.020
- Recall: 0.008
- F1-Score: 0.011
- Accuracy: 97%

Autoencoders showed very low precision and recall, indicating they were not effective for this specific anomaly detection task in financial transaction data.

5. **One-Class SVM**: One-Class SVM was employed to identify anomalies by finding the maximum margin hyperplane. The results were:

- Precision: 0.176
- Recall: 0.298
- F1-Score: 0.222
- Accuracy: 94%

The One-Class SVM had better performance than LOF and Autoencoders but still showed lower effectiveness compared to Isolation Forest.

Model 2: Risk Behaviour Window Detection

Model 2 focused on detecting behavioural changes over a period. For Model 2, Isolation Forest was used again with the following features: month, quarter, day_of_week, is_weekend, is_office_hours, spend, rolling_spend_mean, rolling_spend_sum, time_diff, transactions_last_7_days, transactions_freq_deviation, spend_to_mean_ratio, spend_deviation, cumulative_spend, moving_avg_spend_7d, moving_avg_spend_14d, trans_freq_7d, trans_freq_14d, trans_freq_30d, spend_vs_dept, sequence_length.

Although less emphasis was placed on this model, here are the key results from the Isolation Forest and DBSCAN implementations:

- **Isolation Forest**:
  - Precision: 0.07
  - Recall: 0.03
  - F1-Score: 0.04
  - Accuracy: 97%
- **DBSCAN**:
  - Precision: 0.03
  - Recall: 1.00
  - F1-Score: 0.05
  - Accuracy: 3%

Both models showed similar trends to their counterparts in Model 1, with DBSCAN identifying nearly all instances as anomalies and Isolation Forest performing moderately better.

6.2 Comparison of Models

The performance comparison of the different models reveals that the Isolation Forest model provided the best balance between precision and recall. The One-Class SVM and LOF models showed similar performance but were less effective than the Isolation Forest. DBSCAN and Autoencoders performed poorly, highlighting the complexities of anomaly detection in this financial dataset.

To compare the models, I summarized the performance metrics in the following table:

Model	Anomalies Detected	F1 Score	Precision	Recall	Accuracy
Isolation Forest	5,461	0.2758	0.2078	0.4096	0.94
One-Class SVM	4,832	0.1299	0.1334	0.1267	0.89
Local Outlier Factor	5,651	0.1299	0.1334	0.1267	0.96
DBSCAN	10,208	0.0513	0.0263	1.0	0.03
Autoencoder	5,978	0.0110	0.0199	0.0076	0.96

Table 3: Performance Metrics Comparison

This table illustrates that the Isolation Forest was the most effective model, achieving a good balance between detecting anomalies (precision) and identifying true positives (recall). The One-Class SVM and LOF models had similar performance but were less effective. DBSCAN and Autoencoders struggled with the dataset, with DBSCAN detecting many false positives and Autoencoders showing very low precision and recall.

6.3 Interpretation of Results

The high precision and recall of the Isolation Forest indicate its robustness in detecting anomalies without producing excessive false positives. Its efficiency can be attributed to its capability to isolate anomalies based on the distinctiveness of data partitions. The lower performance of DBSCAN and Autoencoders suggests that these models might not be well-suited for the high-dimensional and potentially noisy transactional data.

Potential reasons for the varied performance include:

- **Isolation Forest's Strengths**: Effective handling of high-dimensional data and non-reliance on data distribution assumptions.
- **LOF's Contextual Sensitivity**: LOF's performance is influenced by local density variations, which may not always align with global anomaly patterns.
- **DBSCAN's Parameter Sensitivity**: DBSCAN requires careful tuning of parameters like *eps* and *min_samples*, which can significantly impact its clustering and anomaly detection capabilities.
- **Autoencoder's Reconstruction Error**: Autoencoders may not always capture the subtle nuances of normal vs. anomalous patterns, leading to higher reconstruction errors.

6.4 Limitations of the Analysis

Despite the promising results, several limitations need to be addressed:

1. **Imbalanced Data**: The significant imbalance between normal and anomalous transactions posed challenges in model training and evaluation.
2. **Feature Selection**: The selected features, while comprehensive, may not capture all the nuances of transactional behavior. Additional domain-specific features could improve model performance.
3. **Parameter Sensitivity**: Models like DBSCAN and Autoencoders are highly sensitive to parameter settings, requiring extensive tuning for optimal performance.
4. **Threshold Setting**: Determining appropriate thresholds for anomaly detection (e.g., reconstruction error in Autoencoders) can be subjective and requires careful tuning.

These limitations highlight the need for ongoing model refinement and the incorporation of more advanced techniques to improve the accuracy and reliability of anomaly detection in financial transactions.

7 Conclusion

In this project, I employed various unsupervised learning models to detect risk events and risk behavior windows within a financial transaction dataset provided by Lloyds Bank. The primary focus was on Model 1, which aimed to identify anomalies indicative of risk events using a comprehensive suite of algorithms, including Isolation Forest, Local Outlier Factor (LOF), DBSCAN, Autoencoders, and One-Class SVM. Model 2, though less emphasized, was designed to identify risk behavior windows using similar methodologies. The results of Model 1 demonstrated that the Isolation Forest algorithm consistently outperformed other models in terms of balanced precision, recall, F1-score, and overall accuracy. Specifically, Isolation Forest achieved an F1-score of 0.276, a precision of 0.207, and a recall of 0.409, with an accuracy rate of 94%. These metrics indicate that while Isolation Forest was effective at detecting anomalies, there remains a considerable number of false positives, which is a common challenge in anomaly detection tasks.

The Local Outlier Factor (LOF) and One-Class SVM algorithms showed moderate performance, with LOF achieving an F1-score of 0.130 and One-Class SVM reaching an F1-score of 0.222. Both models demonstrated lower effectiveness compared to Isolation Forest, with LOF particularly struggling with low precision and recall. The DBSCAN and Autoencoder models, while innovative in approach, were less practical for this dataset. DBSCAN identified nearly all instances as anomalies, resulting in impractically low precision, while Autoencoders displayed very low precision and recall, making them less suitable for effective anomaly detection in this context. Model 2, focused on detecting risk behavior windows, provided additional insights but showed similar performance trends to Model 1. The Isolation Forest and DBSCAN implementations for Model 2 revealed that while these algorithms could identify behavioral changes, their precision and recall metrics were suboptimal for practical applications. Specifically, Isolation Forest achieved an F1-score of 0.04 and DBSCAN demonstrated an F1-score of 0.05.

Overall, the analysis highlighted the effectiveness of Isolation Forest as the most reliable model for detecting anomalies in financial transactions. However, the challenge of balancing precision and recall underscores the need for further refinement and enhancement of these models. The insights gained from this project emphasize the importance of continuous model improvement and the exploration of additional features that could enhance the accuracy and reliability of anomaly detection. In conclusion, this project provided a comprehensive examination of unsupervised learning models for risk event detection in financial transactions. The results underscore the potential of Isolation Forest as a robust tool for anomaly detection while also highlighting the limitations and areas for further research and development. Future work should focus on refining model parameters, exploring new features, and developing strategies for real-world deployment to ensure these models can effectively support risk management efforts in financial institutions.

8 Recommendations

8.1 Recommendations for Industrial Partner

Based on the analysis conducted, several recommendations can be made to enhance the risk detection capabilities of Lloyds Bank. First, integrating the Isolation Forest model into the existing transaction monitoring systems is advisable. This model demonstrated a reasonable balance between precision and recall, making it suitable for identifying anomalous transactions that could indicate potential risks or fraudulent activities. Regular retraining of the model with updated transaction data will ensure its continued accuracy and relevance.

Adopting a multi-model approach will improve the robustness of anomaly detection. Combining the strengths of various models, such as Isolation Forest, LOF, and Autoencoders, into a single framework may be an ensemble to give overall improved performance. It uses the independent, strong points of each model to complement its weak points and thus provide a more detailed detection mechanism.

Incorporating real-time anomaly detection capabilities is another crucial step. Deploying the models in a live environment where they can process incoming transaction data in real-time and flag potential risks immediately is essential. This proactive approach can prevent significant financial losses by enabling timely interventions.

8.2 Suggestions for Future Work

Given the constraints of a four-week project duration, there are several avenues for future work that can build upon the foundations laid by this study. One key area is the enhancement of feature engineering techniques. Developing more sophisticated features that capture complex transactional patterns could improve model performance. For instance, incorporating advanced temporal features and behavioral analytics can provide deeper insights into transaction anomalies.

Exploring advanced deep learning techniques, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, could be beneficial. These models are particularly adept at handling sequential data and can capture intricate temporal dependencies that might be indicative of fraudulent behavior. Additionally, conducting a more extensive hyperparameter tuning for all models can yield better performance. Automated techniques such as grid search and random search can systematically explore a wider range of parameters to identify the optimal settings for each model.

Lastly, expanding the scope of analysis to include more diverse datasets can enhance the generalizability of the findings. Collaborating with other financial institutions to create a more comprehensive dataset could provide a broader perspective on transaction anomalies, leading to more robust models. These recommendations and future directions are designed to bolster the anomaly detection capabilities of Lloyds Bank, ensuring a more secure and fraud-resistant transaction environment.

References

[1] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. In *Special Lecture on IE*, 2015.

[2] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 93–104. ACM, 2000.

[3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):1–58, 2009.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231. AAAI Press, 1996.

[5] S. Hariri, M. Kind, and G. Brunner. Extended isolation forest. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 893–898. IEEE, 2019.

[6] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[7] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. *Future Generation Computer Systems*, 88:463–473, 2018.

[8] L. M. Manevitz and M. Yousef. One-class svms for document classification. *Journal of Machine Learning Research*, 2(Dec):139–154, 2001.

[9] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[10] X. Xu, H.-P. Kriegel, J. Sander, and M. Ester. A systematic review of anomaly detection using machine and deep learning techniques. *Journal of Big Data*, 9(1):1–41, 2022.

[11] Y. Zhang, P. Li, S. Wang, and Y. Liu. *A review of machine learning algorithms for anomaly detection*. Springer, Singapore, 2019.

[12] Y. Zhang, P. Li, S. Wang, and Y. Liu. Explainable ai: Using shapley value to explain complex anomaly detection ml-based systems. *Journal of Big Data*, 7(1):1–41, 2020.

Appendices

A Data Characteristics

The initial dataset consists of the following columns:

Column Name	Description
individual_id	A unique identifier for each individual.
timestamp	The timestamp of the transaction in the format MM.sss.
date	The date of the transaction.
day_of_week	The day of the week the transaction occurred.
hour_of_day	The hour of the day the transaction occurred.
department	The department of the individual.
spend	The amount spent in the transaction (in thousands of pounds).
at_risk_event	A flag indicating whether the event posed a risk.
at_risk_behaviour_window	A flag indicating whether the individual started to become a risk.

B Data Cleaning

Data cleaning involved handling missing values, correcting data types, and ensuring consistency. Specific steps taken include:

1. **Missing Values**: Dataset contain no missing value but later when creating features I identified and handled missing values using imputation techniques where feasible.

2. **Data Types**: Converted date and time columns to appropriate datetime formats for easier manipulation and feature extraction.

3. **Outliers**: Identified outliers in the spend attribute using Z-scores and capped them to reduce their impact on the model.

C Data Preprocessing

To prepare the data for modeling, several preprocessing steps were conducted:

1. **Feature Extraction**: Extracted additional features from the date and time columns, such as hour, minute, month, quarter, day_of_week, is_weekend, and is_office_hours.

2. **Rolling Statistics**: Calculated rolling averages and sums for spend to capture short-term trends.

3. **Time Differences**: Computed the time difference between consecutive transactions for each individual.

4. **Transaction Frequency**: Measured transaction frequency over the last 7 days and compared it to the individual's average transaction frequency.

5. **Categorical Encoding**: Encoded the categorical variable department using one-hot encoding.

D Model Training

Several unsupervised machine learning models were implemented and tested:

1. **Isolation Forest**: Trained to detect anomalies based on the extracted features.
2. **Local Outlier Factor (LOF)**: Used to identify local outliers by comparing the density of each point with its neighbors.
3. **DBSCAN**: Applied to cluster the data and identify anomalies as points that do not belong to any cluster.
4. **Autoencoders**: Neural network-based approach used to reconstruct the input data and detect anomalies based on reconstruction errors.
5. **One-Class SVM**: Trained to identify the normal data points and flag anomalies.

E Model Evaluation Metrics

The models were evaluated using precision, recall, and F1 score, with a focus on balancing these metrics to achieve robust anomaly detection. The `classification_report` from scikit-learn provided detailed insights into the performance of each model.

F Code

F.1 Data Cleaning and Preprocessing

```
import pandas as pd
from scipy import stats
from sklearn.ensemble import IsolationForest
from sklearn.metrics import f1_score, precision_score, recall_score, classification_report
from sklearn.preprocessing import StandardScaler

# Load the dataset
data = pd.read_csv('Lloyds_data_real.csv')

# Summary statistics
print(data.describe())

# Check for missing values
print(data.isnull().sum())

# Convert 'date' to datetime
data['date'] = pd.to_datetime(data['date'], format='%d/%m/%Y')

# Extract hours and minutes from 'hour_of_day'
data['hour'] = data['hour_of_day'].astype(int)
data['minute'] = ((data['hour_of_day'] - data['hour']) * 60).astype(int)

# Convert 'timestamp' to total seconds
data['timestamp_seconds'] = data['timestamp'].apply(lambda x: int(x.split(':')[0]) * 60 + floa

# Extract additional time-based features from the 'date' and 'timestamp'
data['month'] = data['date'].dt.month
data['quarter'] = data['date'].dt.quarter
```

```
data['day_of_week'] = data['date'].dt.dayofweek
data['is_weekend'] = data['day_of_week'].apply(lambda x: 1 if x >= 5 else 0)
data['is_office_hours'] = data['hour'].apply(lambda x: 1 if 8 <= x < 18 else 0)

# Generate rolling averages, rolling sums, and lagged features for 'spend'
data['rolling_spend_mean'] = data.groupby('individual_id')['spend'].transform(lambda x: x.roll
data['rolling_spend_sum'] = data.groupby('individual_id')['spend'].transform(lambda x: x.rolli

# Calculate the time difference between consecutive transactions for each individual
data['time_diff'] = data.groupby('individual_id')['timestamp_seconds'].diff().fillna(0)

# Check if the transaction is the first transaction of the day
data['is_first_transaction'] = data.groupby(['individual_id', 'date'])['timestamp_seconds'].ra

# Calculate the transaction frequency in the last 7 days
data['transactions_last_7_days'] = data.groupby('individual_id')['date'].transform(lambda x: x

# Calculate the transaction frequency deviation from the mean
data['transactions_freq_deviation'] = data['transactions_last_7_days'] - data.groupby('individ

# Calculate the ratio of the spend to the average spend of that individual
data['spend_to_mean_ratio'] = data['spend'] / data.groupby('individual_id')['spend'].transform

# Calculate the spend deviation from the rolling mean
data['spend_deviation'] = data['spend'] - data['rolling_spend_mean']

# Cumulative spend per individual
data['cumulative_spend'] = data.groupby('individual_id')['spend'].cumsum()

# Difference in spend from previous transaction
data['spend_diff'] = data.groupby('individual_id')['spend'].diff().fillna(0)

# Identify outliers in 'spend'
z_scores = stats.zscore(data['spend'])
outliers = data[(z_scores > 3) | (z_scores < -3)]

# Cap outliers
data['spend'] = data['spend'].apply(lambda x: x if x < outliers['spend'].min() else outliers['

# Display summary statistics to verify outlier treatment
print(data['spend'].describe())

# Mean spend per department
mean_spend_department = data.groupby('department')['spend'].mean().to_dict()
data['mean_spend_department'] = data['department'].map(mean_spend_department)

# Load test dataset
df_test = pd.read_csv("Lloyds_data.csv")

# True labels are in the column 'at_risk_event'
y_true = df_test['at_risk_event']
```

F.2 Data Visualization

```
    import matplotlib.pyplot as plt
import seaborn as sns

# Set the style for the plots
sns.set(style="whitegrid")

# Plot Figure: Histogram of Transaction Amounts
plt.figure(figsize=(12, 6))
sns.histplot(data['spend'], bins=50, kde=True)
plt.title('Histogram of Transaction Amounts')
plt.xlabel('Transaction Amount (£1,000s)')
plt.ylabel('Frequency')
plt.show()

# Plot Figure: Transactions by Hour of the Day
plt.figure(figsize=(12, 6))
sns.countplot(x='hour', data=data)
plt.title('Transactions by Hour of the Day')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Transactions')
plt.show()

# Plot Figure: Box Plot of Spend by Department
plt.figure(figsize=(12, 6))
sns.boxplot(x='department', y='spend', data=data)
plt.title('Box Plot of Spend by Department')
plt.xlabel('Department')
plt.ylabel('Transaction Amount (£1,000s)')
plt.xticks(rotation=90)
plt.show()

# Plot Figure: Risk Events by Time of Day
df_test['hour'] = df_test['hour_of_day'].astype(int)
plt.figure(figsize=(12, 6))
sns.countplot(x='hour', hue='at_risk_event', data=df_test)
plt.title('Risk Events by Time of Day')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Transactions')
plt.legend(title='Risk Event', loc='upper right', labels=['No', 'Yes'])
plt.show()

# Aggregate data by hour and risk event status
hourly_data = df_test.groupby(['hour', 'at_risk_event']).size().unstack().fillna(0)
hourly_data.columns = ['No', 'Yes']

# Plot Figure: Risk Events by Time of Day (Line Plot)
plt.figure(figsize=(12, 6))
sns.lineplot(data=hourly_data)
plt.title('Risk Events by Time of Day')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Transactions')
plt.legend(title='Risk Event', loc='upper right', labels=['No', 'Yes'])
```

```
plt.show()

# Filter the data to include only risk events (at_risk_event == 1)
risk_event_data = df_test[df_test['at_risk_event'] == 1]

# Plot Figure: Risk Events by Hour of the Day (Bar Plot)
plt.figure(figsize=(12, 6))
sns.countplot(x='hour', data=risk_event_data)
plt.title('Risk Events by Hour of the Day (Only Risk Events)')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Risk Events')
plt.show()

# Plotting department-wise risk event count
plt.figure(figsize=(12, 6))
sns.countplot(x='department', data=risk_event_data)
plt.title('Risk Events by Department')
plt.xlabel('Department')
plt.ylabel('Number of Risk Events')
plt.xticks(rotation=90)
plt.show()
```

F.3 Model 1: At Risk Event

```
    # Function to evaluate model performance
def evaluate_model(model_name, y_true, y_pred):
    """
    Evaluate the model performance and print the classification report.

    Parameters:
    model_name (str): The name of the model.
    y_true (pd.Series): The true labels.
    y_pred (pd.Series): The predicted labels.
    """
    print(f"{model_name} Model Performance:")
    print(f"F1 Score: {f1_score(y_true, y_pred)}")
    print(f"Precision: {precision_score(y_true, y_pred)}")
    print(f"Recall: {recall_score(y_true, y_pred)}")
    print(f"{model_name}:\n", classification_report(y_true, y_pred))

    import numpy
from sklearn.ensemble import IsolationForest

# Encode categorical variable 'department' using one-hot encoding
data_encoded = pd.get_dummies(data, columns=['department'], drop_first=True)
# Select relevant features for the model
features = ['day_of_week', 'is_weekend', 'is_office_hours',
            'rolling_spend_mean', 'rolling_spend_sum', 'time_diff', 'spend_to_mean_ratio',
            'spend_deviation', 'transactions_last_7_days', 'transactions_freq_deviation']
features.extend([col for col in data.columns if col.startswith('department_')])

# Isolation Forest
iso_forest = IsolationForest(max_samples=0.8, random_state=30, n_estimators=500, n_jobs=-1, ma
```

```
iso_forest.fit(data_encoded[features])

# Detect anomalies
data_encoded['anomaly_score'] = iso_forest.decision_function(data_encoded[features])
data_encoded['at_risk_event_if'] = iso_forest.predict(data_encoded[features])
data_encoded['at_risk_event_if'] = data_encoded['at_risk_event_if'].map({1: 0, -1: 1})

# Evaluate Isolation Forest
evaluate_model("Isolation Forest", y_true, data_encoded['at_risk_event_if'])
# Local Outlier Factor (LOF)
from sklearn.neighbors import LocalOutlierFactor

# Initialize and fit the Local Outlier Factor model
lof = LocalOutlierFactor(n_neighbors=20, contamination=0.025)
data_encoded['anomaly_score_lof'] = lof.fit_predict(data_encoded[features])

# Convert scores to binary labels
data_encoded['at_risk_event_lof'] = data_encoded['anomaly_score_lof'].apply(lambda x: 1 if x =
# Evaluate Local Outlier Factor
evaluate_model("Local Outlier Factor", y_true, data_encoded['at_risk_event_lof'])

# DBSCAN
from sklearn.cluster import DBSCAN

# Initialize and fit the DBSCAN model
dbscan = DBSCAN(eps=0.2, min_samples=1000)
data_encoded['anomaly_score_dbscan'] = dbscan.fit_predict(data_encoded[features])

# Convert scores to binary labels (DBSCAN labels core points with cluster ids and noise points
data_encoded['at_risk_event_dbscan'] = data_encoded['anomaly_score_dbscan'].apply(lambda x: 1
# Evaluate DBSCAN
evaluate_model("DBSCAN", y_true, data_encoded['at_risk_event_dbscan'])

# Autoencoders
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
# Define the Autoencoder model
input_dim = data_encoded[features].shape[1]
encoding_dim = 3

input_layer = Input(shape=(input_dim,))
encoder = Dense(encoding_dim, activation="tanh")(input_layer)
decoder = Dense(input_dim, activation="linear")(encoder)

autoencoder = Model(inputs=input_layer, outputs=decoder)
autoencoder.compile(optimizer='adam', loss='mse')

# Train the model
autoencoder.fit(data_encoded[features], data_encoded[features], epochs=50, batch_size=32, shuf

# Get reconstruction error
```

```
reconstructions = autoencoder.predict(data_encoded[features])
mse = np.mean(np.power(data_encoded[features] - reconstructions, 2), axis=1)
data_encoded['anomaly_score_ae'] = mse

# Set a threshold for anomaly detection (e.g., top 1% as anomalies)
threshold = np.percentile(mse, 99)
data_encoded['at_risk_event_ae'] = data_encoded['anomaly_score_ae'].apply(lambda x: 1 if x > t
# Evaluate Autoencoder
evaluate_model("Autoencoder", y_true, data_encoded['at_risk_event_ae'])

# One-Class SVM
from sklearn.svm import OneClassSVM

# Initialize and fit the One-Class SVM model
oc_svm = OneClassSVM(kernel='rbf', gamma=0.001, nu=0.1)
data_encoded['anomaly_score_svm'] = oc_svm.fit_predict(data_encoded[features])

# Convert scores to binary labels
data_encoded['at_risk_event_svm'] = data_encoded['anomaly_score_svm'].apply(lambda x: 1 if x =
# Evaluate One-Class SVM
evaluate_model("One-Class SVM", y_true, data_encoded['at_risk_event_svm'])
```

F.4 Model 2: At Risk Behaviour Window

```
    # More features for detecting behaviour window
# Moving average spend
data['moving_avg_spend_7d'] = data.groupby('individual_id')['spend'].transform(lambda x: x.rol
data['moving_avg_spend_14d'] = data.groupby('individual_id')['spend'].transform(lambda x: x.ro

# Transaction frequency
data['trans_freq_7d'] = data.groupby('individual_id')['date'].transform(lambda x: x.rolling(wi
data['trans_freq_14d'] = data.groupby('individual_id')['date'].transform(lambda x: x.rolling(w
data['trans_freq_30d'] = data.groupby('individual_id')['date'].transform(lambda x: x.rolling(w

# Spend deviation
data['spend_deviation'] = data['spend'] - data['moving_avg_spend_7d']

import numpy
from sklearn.ensemble import IsolationForest

# Encode categorical variable 'department' using one-hot encoding
data_encoded = pd.get_dummies(data, columns=['department'], drop_first=True)
# Select relevant features for the model
features = ['day_of_week', 'is_weekend', 'is_office_hours',
            'rolling_spend_mean', 'rolling_spend_sum', 'time_diff', 'spend_to_mean_ratio',
            'spend_deviation', 'transactions_last_7_days', 'transactions_freq_deviation',
            'moving_avg_spend_7d', 'moving_avg_spend_14d', 'trans_freq_7d', 'trans_freq_14d',
            'trans_freq_30d', 'spend_deviation']
features.extend([col for col in data.columns if col.startswith('department_')])

X = data_encoded[features]
X.fillna(0, inplace=True)
```

```
# Standardize the features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Convert numbers to True and NaN to False in a specified column
data_encoded['at_risk_behaviour_window'] = df_test['at_risk_behaviour_window'].apply(lambda x:

from sklearn.ensemble import IsolationForest

# Initialize and train Isolation Forest
iso_forest = IsolationForest(random_state=42)
iso_forest.fit(X_scaled)

# Predict anomalies
data_encoded['anomaly_window_score_iso'] = iso_forest.decision_function(X_scaled)
data_encoded['anomaly_window_iso'] = iso_forest.predict(X_scaled)

# Map -1 to 1 (anomaly) and 1 to 0 (normal)
data_encoded['anomaly_window_iso'] = data_encoded['anomaly_window_iso'].map({1: 0, -1: 1})

# Verify predictions
print(data_encoded[['anomaly_window_score_iso', 'anomaly_window_iso']].head())
print(data_encoded['anomaly_window_iso'].value_counts())
evaluate_model("Isolation Forest", y_true, data_encoded['anomaly_window_iso'])

from sklearn.cluster import DBSCAN

# Initialize and fit DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=100)
data_encoded['anomaly_window_dbscan_labels'] = dbscan.fit_predict(X_scaled)

# Identify anomalies (points labeled as -1 by DBSCAN)
data_encoded['anomaly_window_dbscan'] = (data_encoded['anomaly_window_dbscan_labels'] == -1).a

# Verify the clustering and anomaly detection
print(data_encoded[['anomaly_window_dbscan_labels', 'anomaly_window_dbscan']].head())
print(data_encoded['anomaly_window_dbscan'].value_counts())
evaluate_model("DBSCAN", y_true, data_encoded['anomaly_window_dbscan'])
```