Individual Assignment MMD



Fraud Detection Model

Dario Pijman 1658324

[D.Pijman@Student.han.nl](mailto:D.Pijman@Student.han.nl)

## Introduction

I had many doubts about choosing this minor. This was mainly because of my ‘history’ with IT studies. At the start of my HBO career I did ICT in Nijmegen. However, after 6 months of ‘very’ low grades and me not having fun I decided that this wasn’t for me. But, I was willing to challenge myself again since I hate the feeling of not being able to do something. I was looking forward to the minor, but there was one thing that I was afraid of: coding. When choosing a topic for this individual assignment, I was thinking about the possibilities. I could have chosen an easy topic. But, I wanted to challenge myself since I am not fully convinced that I understand coding now. I wanted to see if I could create this project to conclude if I had shaken off these flashbacks from ICT in my first year.

## Fraud detection model

When thinking of an idea for my project I was watching tv. They were talking about online scams and how you can prevent this. I was wondering if I could create a model that can detect fraud in a certain dataset. And I was prepared to give it a try. I looked for a dataset online but I couldn't find one that suited my model. So, I decided to create one by myself.

## CRISP DM Model



Business understanding:   
The goal of my fraud detection model is that it can detect the differences between a fraudulent and a non-fraudulent activity. It doesn’t have to be a perfect score but if it works and it can detect some fraudulent activity, I would be happy. My personal goal would be to code this model by myself without any help of a team around me and really challenge myself to create this.

Sadly, fraud still has an impact on people’s lives daily. If there would be a model that can detect fraud, this could be hugely beneficial for people but also for companies and government instances such as the Dutch ‘belastingdienst’ for example. I understand that this model is extremely difficult to make and because my model is based on a ‘fake’ dataset the outcomes will be different from a real dataset and a real life situation.

Data understanding:

For this project I created the dataset by myself. This was mainly because I couldn’t find the ‘right’ dataset and in this way I could add everything that I needed the dataset to have. First I added a timestamp, so that I and the model could exactly see at what date the transaction was performed. I also added the amount of the transaction. Maybe the model could detect that higher transactions are more likely to be fraud for example. Then I wanted to know the exact hour of the transaction. And lastly I added the fraud label whether each transaction is legitimate (0) or fraudulent (1). Later I also added location, because I found it interesting to see if this could have an impact as well. The CSV file has 711 rows which represent a months’ worth of transactions. This could be a real dataset for a smaller company for example.

Data preparation:  
I generated the synthetic dataset, so there are no missing values or outliers explicitly handled. The 'location' column was label-encoded using LabelEncoder(), converting categorical values ('A', 'B', 'C') into numeric values. I created New features from the timestamp, including 'day\_of\_week', 'day\_of\_month', 'week\_of\_year', 'hour', and 'minute'. These features capture temporal aspects of the transactions. And the dataset was split into training and testing sets using train\_test\_split from sklearn.model\_selection. By splitting the dataset you can train your model on one part and evaluate its performance on another part. Training a model on the entire dataset could also lead to overfitting.

Modeling:  
For modelling I created two models, namely the forest isolation model and the XGBoost model. The Isolation Forest is an ensemble machine learning algorithm used for anomaly detection. The main idea behind the Isolation Forest is to isolate anomalies by randomly partitioning the data. It works by creating isolation trees, which are essentially binary trees. Each tree is built by recursively selecting a random feature and a random split point for that feature until the data points are isolated (placed in leaf nodes). Anomalies, being less frequent, are expected to be isolated closer to the root of the tree in fewer splits compared to normal instances. Normal instances require more splits to be isolated than anomalies. The anomaly score for a data point is calculated based on the average path length across all trees in the forest. The average path length is then normalized to obtain the anomaly score. A threshold is set to distinguish between normal and anomalous instances. Instances with anomaly scores above the threshold are considered anomalies.

XGBoost, which means eXtreme Gradient Boosting, is a powerful and widely used machine learning algorithm that belongs to the class of gradient boosting methods. It is an ensemble learning algorithm that builds a strong predictive model by combining the predictions of multiple weak models, typically decision trees. It uses a technique called gradient boosting, where each new tree corrects the errors of the previous ones. The model uses decision trees as its base learners (weak models). These trees are often shallow and are added sequentially to the mix. It optimizes an objective function, which is a combination of a loss function and a regularization term. The objective function guides the model to minimize both the training error and the complexity of the model to prevent overfitting. XGBoost incorporates L1 (Lasso) and L2 (Ridge) regularization terms in the objective function to control the complexity of the individual trees and the overall model. It uses gradient descent optimization techniques to minimize the objective function. The algorithm calculates the gradient of the objective function with respect to the predicted values, and updates the model parameters (tree weights) in the direction that reduces the gradient. The model employs a pruning technique to control the depth of the trees during the construction process. This helps prevent overfitting. It supports built-in cross-validation to help users find the optimal number of boosting rounds (iterations) and tune hyperparameters effectively. It is designed for efficiency and can take advantage of parallel processing. It also supports distributed computing for large datasets.

Using both Isolation Forest and XGBoost provides a diverse set of approaches. Isolation Forest focuses on identifying anomalies in the feature space, while XGBoost builds a powerful classifier. Combining these models can capture different aspects of fraud patterns.

The evaluation and deployment part of the crisp dm model will be covered later.

## Data loading and preprocessing

I loaded the synthetic dataset that I generated, which includes timestamps, transaction amounts, and a fraud label indicating whether each transaction is legitimate (0) or fraudulent (1). I converted the 'timestamp' column to a datetime format and extracted features such as day of the week, hour, and minute. This step was necessary to make the timestamps usable as features for machine learning models. I split the dataset into training and testing sets to train the models on one subset of the data and evaluate their performance on another.

## Isolation Forest Model

I used the Isolation Forest algorithm for anomaly detection. It's an unsupervised learning algorithm that identifies anomalies (fraudulent transactions in this case) by isolating them in the feature space. The predictions from Isolation Forest are typically -1 for anomalies and 1 for normal instances. I adjusted the predictions to 0 for normal instances and 1 for anomalies since those were the numbers already in use. I evaluated the Isolation Forest model using metrics such as precision, recall, F1-score, and accuracy. The model performed well, achieving high precision and recall for detecting fraudulent transactions.

**Isolation Forest Model Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
| **0** | 0.98 | 0.96 | 0.97 | 140 |
| **1** | 0.33 | 0.50 | 0.40 | 6 |
| **Accuracy** | **-** | - | 0.94 | 146 |
| **Macro avg** | 0.66 | 0.73 | 0.68 | 146 |
| **Weighted avg** | 0.95 | 0.94 | 0.94 | 146 |

**Accuracy = 0.9383561643835616**

The model demonstrated high precision and recall for normal transactions, indicating its ability to accurately classify non-fraudulent instances. However, for fraudulent transactions, the model exhibited lower precision and recall, suggesting that it could detect some instances of fraud. The overall accuracy of 0.94 highlights the model's success in distinguishing between normal and anomalous transactions.

## XGBoost Model

I used the XGBoost classifier to directly classify transactions as either legitimate (0) or fraudulent (1). XGBoost is a powerful gradient boosting algorithm often used for classification tasks. I evaluated the XGBoost model using the same metrics. The XGBoost model also performed very well, achieving high precision, recall, F1-score, and accuracy.

**XGBoost Model Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
| **0** | 1.00 | 1.00 | 1.00 | 140 |
| **1** | 1.00 | 1.00 | 1.00 | 6 |
| **Accuracy** | **-** | - | 1.00 | 146 |
| **Macro avg** | 1.00 | 1.00 | 1.00 | 146 |
| **Weighted avg** | 1.00 | 1.00 | 1.00 | 146 |

**Accuracy = 1.0**The XGBoost model achieved perfect precision and recall for both legitimate and fraudulent transactions. This signifies the ability to correctly identify and classify instances, crucial for minimizing false positives and negatives in fraud detection. With an F1-score of 1.00 for both classes, the XGBoost model strikes a balance between precision and recall. This indicates a model capable of maintaining high accuracy while effectively handling imbalanced datasets. The overall accuracy of 1.00 underscores the model's ability to make correct predictions across the entire dataset, reinforcing its reliability in distinguishing between legitimate and fraudulent transactions.

The Isolation Forest model is suitable for anomaly detection when you have a mix of normal and anomalous instances. The XGBoost model is a versatile classifier and can be powerful for fraud detection when we there is labeled data.   
  
The Isolation Forest model, made for anomaly detection, performs best when presented with a mix of normal and anomalous instances. In contrast, the XGBoost model, a versatile and robust classifier, works best in scenarios where labeled data is available. The synergy between these models provides a comprehensive approach to fraud detection, addressing varied aspects of the complex and dynamic nature of financial transactions. As we navigate fraud detection, the XGBoost model emerges as a valuable asset, demonstrating not only its proficiency in classification tasks but also its adaptability to the fraudulent activities.

## Location makes fraud?

After creating the isolation forest and XGBoost model I wanted to investigate the correlation between location and fraud more. Even though both models worked very well, I felt the need to do more research and also show my visualization capabilities, that I have learned during the minor.

First I created a code in Spyder that creates a visualization that shows the locations in the dataset and also the amount of times there was no fraud or fraud committed in that certain country. This gives a good overview if there could be a correlation between the two.

Afbeelding met tekst, schermopname, diagram, nummer

Automatisch gegenereerde beschrijving

This is the visualization I created. As you can see the fraud to no fraud ratio is low. Location B has the lowest fraud number with only 5 frauds happening here. The other two are a little higher with location A 11 frauds and location C, 14 frauds. These numbers are very close to each other so my first thought was that there was no correlation between location and fraud. I decided to create a different visual that shows the percentage of fraud in a better way.

Afbeelding met tekst, schermopname, diagram, nummer

Automatisch gegenereerde beschrijving

When you see this visual you immediately get a different react being that location C has the most fraud and thereby there is a correlation between the two. I decided to perform a chi-squared test and calculate the p-value, which resulted in:   
Chi-squared value: 3.7779823825689975  
P-value: 0.151224288292162

Based on the chi-squared test, the p-value is 0.1512 (rounded), which is greater than the commonly used significance level of 0.05. Therefore, we do not have enough evidence to reject the null hypothesis, and we conclude that the distribution of fraud across locations is not statistically significant. This suggests that, according to the chi-squared test, there isn't a significant difference in the distribution of fraud across the different locations in the dataset. From these results I could conclude that my initial thought was right and that there isn’t a significant difference in the distribution of fraud across the locations in the dataset.

## Amount makes fraud?

After concluding that location doesn’t necessarily makes fraud, I wanted to see if the amount could. And the best way to do this in my opinion was to create a boxplot. So that’s exactly what I did.

Afbeelding met tekst, schermopname, diagram, scherm

Automatisch gegenereerde beschrijving

In the chart you can see that there is a big difference between in amount between no fraud (0) and fraud (1). With fraud being more stretched out, meaning a lot of different amounts, and no fraud very pressed up with no amount higher than 1000. You immediately think that there is a correlation between the two but you just never now. I performed a T-test by calculating the T-statistic and the p-value, which are:   
T-statistic: 8.135705334692295  
P-value: 5.654607408785541e-09  
  
The t-test results indicate that the difference in mean amounts between fraud and no fraud transactions is statistically significant. The t-statistic of 8.1357 and a very small p-value (close to zero) suggest that there is a significant difference in the amounts for fraud and no fraud. This implies that amount can be a meaningful predictor for identifying fraudulent transactions in the dataset. Transactions with significantly different amounts might be indicative of potential fraud.

## Naïve bayes model

I believe my research was very good so far and I could’ve stopped here. But I wanted to test and show another model. And I choose to create a naïve bayes model. Naive Bayes is a family of probabilistic classification algorithms based on Bayes' theorem with the "naive" assumption of independence between features. It's a simple yet effective model that is particularly useful for text classification and spam filtering but it can also be used as a fraud detection model. And it performed really well:

**Naïve Bayes Model Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
| **0** | 1.00 | 1.00 | 1.00 | 140 |
| **1** | 1.00 | 1.00 | 1.00 | 6 |
| **Accuracy** | **-** | - | 1.00 | 146 |
| **Macro avg** | 1.00 | 1.00 | 1.00 | 146 |
| **Weighted avg** | 1.00 | 1.00 | 1.00 | 146 |

**Accuracy = 1.0**

The Naive Bayes model showed great performance, showcasing precision, recall, and F1-score of 1.00 for both legitimate and fraudulent transactions. This performance translated to an overall accuracy of 1.00, highlighting the model's capability to accurately classify instances across the dataset. The exceptional precision and recall values signify the Naive Bayes model's proficiency in distinguishing between legitimate and fraudulent transactions. The model's accuracy of 1.00 underscores its ability to make precise predictions, minimizing the likelihood of false positives and false negatives. While the Naive Bayes model exhibited outstanding results, it's essential to recognize that its performance can vary based on the dataset characteristics. The simplicity of the model makes it computationally efficient, yet its "naive" assumption may not always align with complex relationships within the data. Do not that for both the naïve bayes model and the XGBoost model the accuracy = 100. Usually, when dealing with large datasets it is impossible to get an accuracy rate of 100. And most of the times this indicates that something is wrong. However since my dataset is not that big (only 711 lines) this shouldn’t be an easy and thereby performs exceptionally well.

## Evaluation

The fraud detection model has been successfully trained and evaluated using various machine learning algorithms:

- The Isolation Forest model demonstrated high precision and recall in detecting anomalies.

- The XGBoost model achieved accurate fraud classification with a high accuracy score.

- The correlation between location and fraud has been visualized using heatmaps and bar plots.

- Statistical tests, such as chi-squared and t-tests, were conducted to assess the significance of certain features.

- K-Means clustering was applied to explore patterns and relationships within the dataset.

- The Naive Bayes model also showed promising results in fraud classification.

## Deployment of Models in Fraud Detection

The first step in the deployment phase involves saving the trained models for Isolation Forest, XGBoost, and Naive Bayes. These models need to be stored in a format that allows easy retrieval and integration into the deployment environment.   
  
If the deployment scenario involves serving predictions via an API, creating an API endpoint becomes crucial. An API allows external systems or applications to interact with the models. Using a web framework like Flask or FastAPI, developers can define an endpoint that receives input data, processes it through the models, and returns predictions. This API acts as a bridge between the machine learning models and the systems or applications that need fraud detection services.

The API endpoint should be designed to handle incoming requests containing input data. For fraud detection, this input data may include features such as transaction amount, timestamp, and location. Input validation is essential to ensure that the incoming data meets the expected format and contains the necessary features for model predictions. The API should handle errors and provide informative responses.

Before making predictions, the input data must undergo the same preprocessing steps applied during model training. For example, in the provided Isolation Forest and XGBoost models, the data was standardized using StandardScaler. The deployment code should include these preprocessing steps to ensure that incoming data is scaled appropriately before being fed into the models.

Collaboration with relevant stakeholders is crucial for integrating the deployed models into existing fraud detection workflows. Understanding the data sources, communication protocols, and integration requirements is key. The API endpoints can be integrated into larger systems, allowing seamless incorporation of fraud detection predictions into business processes.

Continuous monitoring is essential for ensuring the ongoing effectiveness of the deployed models. Metrics such as prediction accuracy, false positives, and false negatives should be tracked over time. Anomaly detection mechanisms can be implemented to identify potential issues or drift in the input data distribution. Strategies for model updates and retraining should be defined, taking into account evolving patterns in fraudulent activities and shifts in the underlying data.

Deploying models for fraud detection demands a heightened focus on security. Encryption protocols should be in place to secure data during transmission. Access controls and authentication mechanisms must be implemented to restrict unauthorized access to the API and protect sensitive information. Regular security audits and assessments should be conducted to identify and address potential vulnerabilities.

Maintaining version control for the deployed models is important for tracking changes and ensuring reproducibility. Detailed documentation should be provided, covering aspects such as model versions, preprocessing steps, and API specifications. This documentation serves as a valuable resource for developers, data scientists, and other stakeholders involved in the deployment and maintenance processes.

As the deployment environment scales to handle increased traffic and data volume, considerations for scalability and performance optimization become more important. This may involve deploying the API on cloud infrastructure, optimizing code for efficient execution, and implementing caching mechanisms to enhance response times.

User feedback plays an important role in the continuous improvement of the deployed models. Establishing communication channels for receiving feedback from end-users and stakeholders helps in identifying potential shortcomings or areas for enhancement. This feedback loop can inform future model updates and contribute to the refinement of fraud detection strategies.

**Thanks for reading my portfolio! I hope it was interesting!**

**Link to github:** [**https://github.com/DarioPijman/datasciencefraudmodel/tree/main**](https://github.com/DarioPijman/datasciencefraudmodel/tree/main)