FACULDADE DE TECNOLOGIA DE BAURU

TECNOLOGIA EM BANCO DE DADOS

DESENVOLVIMENTO DE UM MODELO DE MACHINE LEARNING DE DETECÇÃO DE FRAUDE EM PYTHON

Equipe:
Guilherme Dias Coelho

Bauru/SP
2025

Desenvolvimento de um modelo de machine learning de detecção de fraude em Python

Equipe:
Guilherme Dias Coelho

Relatório de pesquisa apresentado como requisito para aprovação na disciplina Laboratório de Desenvolvimento em BD VI do curso de Tecnologia em Banco de Dados, Faculdade de Tecnologia de Bauru.

Profa. Dra. Patricia Bellin Ribeiro

Bauru/SP
2025.

SUMÁRIO

Pág.

RESUMO

O uso de dados cresce em um alto ritmo, impulsionando, também, o aumento de transações financeiras e, consequentemente, o aumento de fraudes. Assim, o artigo aborda a explicação e recriação de um projeto de ciências de dados encontrado no site “170 projetos de data science e machine learning com python resolvidos e explicados.”. Sendo, o foco do artigo desenvolver o projeto 61, que trata da criação de um modelo de *machine learning* para detectar e classificar transações financeiras em uma base de dados, onde os valores foram classificados em transações legítimas e fraudulentas. Ao longo do conteúdo, é possível ver a replicação e funcionamento do algoritmo e seu modo de classificação, buscando padrões e detectando diferenças em todo padrão da base de dados. Assim, os resultados foram favoráveis, como encontrados no próprio projeto proposto, com uma análise e classificação 100% eficaz. Além disso, esse tipo de algoritmo e análise pode se e, em alguns casos, já é usado em diversas outras áreas, podendo expandir para algoritmos e sistemas mais complexos para auxiliar e evitar futuras fraudes e outros crimes.

Palavras-chave: Machine Learning; Fraudes; Python; Ciência de Dados; Classificação.

ABSTRACT

The use of data is growing at a rapid pace. As a result, financial transactions are increasing as well, and consequently, so are fraud cases. Therefore, this article discusses and recreates a data science project found on the website “170 data science and machine learning projects with Python solved and explained.” The focus of the article is the development of project 61, which involves creating a machine learning model to detect and classify financial transactions in a dataset, where the values were labeled as legitimate or fraudulent transactions. Throughout the content, it is possible to observe the replication and functioning of the algorithm and its classification process, identifying patterns and detecting deviations across the entire dataset. The results were favorable, as also found in the original proposed project, achieving 100% effective analysis and classification. Furthermore, this type of algorithm and analysis can be—and in some cases already is—used in various fields, with the potential to expand into more complex algorithms and assist in preventing fraud and other crimes.

1. INTRODUÇÃO

Com a crescente no uso de dados e das ferramentas de IA, a análise dos dados precisa se tornar mais rápida e precisa. Para isso, a ciência de dados, de acordo com Grus (2016), é uma junção de diversas habilidades focadas em organizar conjuntos de dados de forma lógica, se utilizando de estatística ou modelos de IA para encontrar padrões, sendo extremamente importante para entender, manipular e explorar dados. Para essa finalidade, o Python é uma linguagem muito conhecida, tanto para análise de dados quanto para criação de modelos de *machine learning*, ou aprendizado de máquina, que para Géron (2019), é permitir que um computador tenha a capacidade de aprender utilizando dados. Sendo assim, esses algoritmos permitem explorar, automatizar,

classificar e encontrar padrões em grandes quantidades de dados, o que levaria muito tempo sendo feito manualmente, de forma eficiente.

Por causa disso, o desenvolvimento do artigo se foca em desenvolver e explicar um projeto, pego no site, “170 projetos de data science e machine learning com python resolvidos e explicados.”, focado em desenvolver um modelo de *machine learning* . Com isso, o projeto escolhido foi “Fraud Detection with Machine Learning”, que como diz o nome, será criado um algoritmo de *Machine Learning* utilizando Python para analisar e, com base nos dados, encontrar padrões e detectar possíveis fraudes que ocorreram ou ocorrem no sistema, classificando essas transações em legítimas ou fraudes.

Assim, essa classificação é essencial para entender e aprimorar a segurança da rede que é o pilar de toda organização, evitando fraudes e roubos e permitindo aos usuários terem mais segurança ao acessar e trabalhar com um sistema.

2. OBJETIVOS

Os objetivos do artigo são: analisar a estrutura do projeto, focando em analisar o código criado, explicando ele e seus resultados; desenvolvimento do projeto, criando ele e reproduzindo em um ambiente de testes para conseguir resultados iguais ou diferentes; e comentar sobre sua viabilidade em diversos tipos de sistemas diferentes.

Assim, os objetivos centrais tem como foco, uma análise sobre o modelo de *machine learning* voltado para classificar, com o uso de uma base de dados disponibilizada no próprio projeto, as transações. Com isso, os objetivos, de forma mais expandidas, foram determinados para ter um entendimento completo sobre o algoritmo e como ele ajuda na compreensão da segurança e como ele auxilia para encontrar e classificar suas informações.

Com isso, o primeiro objetivo, analisar a estrutura do projeto, serve para explicar as tecnologias usadas e como é feito a classificação dos dados, se focando em explicar o processo de desenvolvimento e como é usado. No segundo objetivo, será desenvolvido o algoritmo em um ambiente de teste e, em seguida, executado para ver seu funcionamento e medir a eficiência do programa. No último objetivo, será analisado sua eficiência na análise de informações fraudulentas e debatido seu uso.

3. MATERIAIS E MÉTODOS

Para iniciar o projeto, será utilizado o Google Colab, para criar e rodar o algoritmo de Machine Learning. Com isso, primeiro passo é baixar o arquivo CSV, chamado payment_fraud, se o arquivo vir em formato .txt é necessário apenas, com o auxílio do Excel, alterar ele para .csv. O arquivo contém diversos dados, incluindo o métodos de pagamento de pagamento, idade da conta em dias, número de itens, entre outros.

Depois, entra a codificação do algoritmo. Para ele, será usado o Python e as bibliotecas Pandas, que de acordo com sua documentação, é um software livre, open source, de alta performance e que permite manipular dados e estruturas de dados de maneira fácil, e o Scikit-Learning, que de acordo com sua documentação, também é um software open source que permite criar modelos de *machine learning* supervisionados ou não supervisionados, além de permitir várias outras utilidades, como processamento de dados, criar modelos de seleção, entre outros. O Google Colab é amplamente usado

para testar modelos voltados para machine learning, pois nele não é necessário instalar essas e algumas outras bibliotecas, o Pandas e o Scikit-Learning já estão nele.

Com isso pronto, a próxima parte é, dentro de um notebook no Google Colab, importar as bibliotecas:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matrix
```

O primeiro *import* é o Pandas, onde será dado um aliás para ele o chamando de “pd”. Após isso, será chamado algumas funções do Scikit-Learning, como a train_test_split, usado para dividir a base de dados em conjunto de teste e conjunto de treino, o LogisticRegresion, usado para classificar de maneira logística, para predição de valores, e algumas métricas, com o acurracy_score, para a precisão do modelo, e o confusion_matrix, para avaliar a precisão da classificação.

Após isso, é necessário importar o arquivo para o Google Colab e transformar ele em um *DataFrame* para ser usado no modelo. Assim, será usado esses código:

```
from google.colab import files

uploaded = files.upload()

df = pd.read_csv('payment_fraud.csv')

df.head()
```

Com isso, na biblioteca google.colab é importado o módulo files, que permite fazer *upload* de um arquivo para dentro do próprio Google Colab. Após, será criado o DataFrame, chamado df, e passado o read_csv, uma função do Pandas para ler e transformar um arquivo CSV, e, por fim, é mostrado os primeiros cinco itens do DataFrame.

Entrando em um possível problema com o DataFrame, pode-se usar o seguinte comando:

- df = pd.get_dummies(df, columns=[‘paymentMethod’], drop_first=True)

Esse comando, corrige um possível erro na leitura da coluna ‘paymentMethod’, que pode não permitir a execução das próximas linhas de código.

Agora, é a parte da construção do modelo, para isso, primeiro é dividido o conjunto de dados em conjuntos de treinos e conjuntos de teste, com esse código:

```
X_train, X_test, y_train, y_test = train_test_split(

    df.drop('label', axis=1), df['label'],

    test_size=0.33, random_state=17)
```

Para essa parte, o conjunto é divido em quatro partes, sendo dois conjuntos para treino e dois para teste. Com isso feito, é usado um algoritmo de regressão logística para realizar uma classificação binária das transações.

```
clf = LogisticRegression().fit(X_train, y_train)
```

Com o algoritmo aplicado, agora é hora de verificar a taxa de acertos do modelo. Para isso, será usado o accuracy_score para mostrar o valor.

```
y_pred = clf.predict(X_test)

from sklearn.metrics import accuracy_score

print(accuracy_score(y_pred, y_test))
```

A saída desse código pode ser um número de 0.0 a 1.0, ou seja, esse valor reprenta 0% a 100% de precisão na taxa de acerto.

Para o final, é usado o algoritmo confusion_matrix para ver nos conjuntos as transações que foram confirmadas com legítimas e as que foram fraudulentas. Para isso, é necessário apenas uma última linha de código.

```
print(confusion_matrix(y_test, y_pred))
```

4. RESULTADOS E DISCUSSÃO

Como resultado, após a compilação da primeira parte do código, temos como resultado uma tabela com os 5 primeiros itens do DataFrame, mostrando todas as informações referentes ao começo da tabela. Na figura 1, abaixo, não foi usado o upload, pois sua necessidade é apenas de um único uso, ou se optar pode transferir o arquivo manualmente para o Google Colab.

Figura 1 – Primeira parte do código

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

df = pd.read_csv('/content/sample_data/payment_fraud.csv')
df.head()
```

	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays	label
0	29	1	4.745402	paypal	28.204861	0
1	725	1	4.742303	storecredit	0.000000	0
2	845	1	4.921318	creditcard	0.000000	0
3	503	1	4.886641	creditcard	0.000000	0
4	2000	1	5.040929	creditcard	0.000000	0

Fonte: Autor

Para a próxima parte, será adicionado a esse código, toda a parte de teste, treino e apresentação da métrica accuracy_score. Com isso, será visto e classificado a taxa de acertos do modelo. Assim, sendo representado na figura 2, onde tem todo o código anterior mais o novo código adicionado e a taxa de acertos.

Figura 2 – Adição de código e taxa de acertos

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

df = pd.read_csv('/content/sample_data/payment_fraud.csv')

# Apply one-hot encoding to the 'paymentMethod' column
df = pd.get_dummies(df, columns=['paymentMethod'], drop_first=True)

# Split dataset up into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    df.drop('label', axis=1), df['label'],
    test_size=0.33, random_state=17)


clf = LogisticRegression(max_iter=1000).fit(X_train, y_train)

# Make predictions on test set
y_pred = clf.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(y_pred, y_test))
```

```
1.0
```

Fonte: Autor

Com isso, é possível visualizar que a taxa de acertos do modelo foi de 1.0, ou seja, o programa, assim como no projeto apresentado, teve uma média de 100% de acertos na hora de classificar se uma transação é legítima ou fraudulenta.

Assim, teremos a adição da última parte do código que apresenta, em uma matriz, as transações que foram identificadas como legítimas e as que foram identificadas como fraudulentas. Nesse sentido, toda essa informação está na figura 3, que vai conter o código completo do modelo de *machine learning* para detecção de fraudes.

Figura 3 – Código completo

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

df = pd.read_csv('/content/sample_data/payment_fraud.csv')

# Apply one-hot encoding to the 'paymentMethod' column
df = pd.get_dummies(df, columns=['paymentMethod'], drop_first=True)

# Split dataset up into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    df.drop('label', axis=1), df['label'],
    test_size=0.33, random_state=17)


clf = LogisticRegression(max_iter=1000).fit(X_train, y_train)

# Make predictions on test set
y_pred = clf.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(y_pred, y_test))

# Compare test set predictions with ground truth labels
print(confusion_matrix(y_test, y_pred))
```

```
1.0
[[12753     0]
 [    0   190]]
```

Fonte: autor

Dessa maneira, é possível ver na primeira linha, dos resultados, a taxa de acertos e nas próximas a matriz que apresenta a classificação das transações. Assim, o modelo classificou 12753 transações como legítimas, como teve 100% de acerto não obteve nenhum erro, e classificou 190 transações como fraudulentas, também sem nenhum erro.

5. CONCLUSÕES

Em resumo, o modelo de *machine learning* criado e proposto no projeto identifica anomalias nas transações de maneira automatizada, isso facilita e mostra o poder crescente desses modelos. Como no modelo criado, ele obteve um acerto de 100% na hora de classificar as informações sobre as transações, porém ainda é preciso testar em

outras bases de dados, com diferentes dados e/ou mais informações. Com isso, é possível ver sua utilização em um modelo aprimorado para identificação de transações fraudulentas em tempo real, como é utilizado em diversos lugares, e proteger os usuários, tanto pessoas físicas como jurídicas, de fraudes envolvendo suas contas. Além disso, os modelos de *machine learning* podem também conversar com diversos tipos de programas, o que pode ajudar muito mais na identificação de anomalias no sistema.

Em suma, esse projeto mostra a grande e facilitada utilização da linguagem Python para criar modelos de *machine learning* e, integrado com suas diversas bibliotecas, permitir eles a trabalhar e processar uma alta gama de valores. Assim, como o projeto visa reconhecer fraudes, novas metodologias podem ser criadas para evitá-las.

6. REFERÊNCIAS

E-SETORIAL. **170 Projetos de Data Science e Machine Learning com Python, resolvidos e explicados**. Disponível em: https://www.e-setorial.com.br/blog/229-170-projetos-de-data-science-e-machine-learning-com-python-resolvidos-e-explicados. Acesso em: 18 set. de 2025.

GÉRON, A. **Mãos à obra: Aprendizado de Máquina com Scikit-Learning & Tensorflow.** Tradução de Rafael Contatori. Rio de Janeiro: Alta Books, 2019.

GRUS, J. **Data Science do Zero**. Tradução de Wellington Nascimento. Rio de Janeiro: Alta Books, 2016.

KHARWAL, A. **Fraud Detection with Machine learning**. Disponível em: https://amanxai.com/2020/08/04/fraud-detection-with-machine-learning/. Acesso em: 18 de set. 2025.

PANDAS. **Pandas Documentation**. Disponível em: https://pandas.pydata.org/docs/. Acesso em: 20 set. 2025.

SCIKIT-LEARNING. **Getting Start.** Disponível em: https://scikit-learn.org/stable/getting_started.html. Acesso em: 20 set. 2025.