



DETECÇÃO DE BUGS EM MODELOS DE MACHINE LEARNING

Giovanni Cardoso Pertence dos Santos (giovannicps@al.insper.edu.br)

João Pedro Gianfaldoni de Andrade (joaopga1@al.insper.edu.br)

William Augusto Reis da Silva (williamars@al.insper.edu.br)

Trabalho de Conclusão de Curso

**Relatório
Versão Intermediária
do Projeto Final de Engenharia**

**São Paulo-SP
SETEMBRO 2022**

**Giovanni Cardoso Pertence dos Santos
João Pedro Gianfaldoni dos Santos
William Augusto Reis da Silva**

DETECÇÃO DE BUGS EM MODELOS DE MACHINE LEARNING

Relatório Intermediário do Projeto Final de Engenharia

Relatório apresentado ao curso de Engenharia, como requisito para o Trabalho de Conclusão de Curso.

Professor Orientador: Prof. Dr. Fabrício Jailson Barth

Mentor na Empresa: Thiago Cardoso

Coordenador TCC/PFE: Prof. Dr. Luciano Pereira Soares

**São Paulo - SP
SETEMBRO 2022**

**Giovanni Cardoso Pertence dos Santos
João Pedro Gianfaldoni dos Santos**

William Augusto Reis da Silva

DETECÇÃO DE BUGS EM MODELOS DE MACHINE LEARNING

Projeto Final de Engenharia apresentado ao programa Graduação em Engenharia (Computação/Mecânica/Mecatrônica) como requisito parcial para a obtenção do título de Bacharel em Engenharia.

Orientador: Prof. Dr. Fabrício Jailson Barth

Banca Examinadora

Prof. Dr. Fabrício Jailson Barth

Inspere

Prof. Dr. Fábio José Ayres

Inspere

Prof. Dr. Tiago Fernandes Tavares

Inspere

Sumário

RESUMO.....	4
ABSTRACT.....	5
1. INTRODUÇÃO.....	7
1.1 ESCOPO DO PROJETO	8
1.2 RECURSOS.....	9
1.3 CRONOGRAMA.....	9
1.4 MAPEAMENTO DOS STAKEHOLDERS.....	10
1.5 QUESTÕES ÉTICAS E PROFISSIONAIS	11
1.6 REVISÃO DO ESTADO DA ARTE	11
2. METODOLOGIA	12
2.1 COLETA DE DADOS SOBRE O PROJETO	15
2.2 ANÁLISE DOS DADOS COLETADOS.....	16
2.2.1 <i>Testes de Sanidade.....</i>	<i>16</i>
2.2.2 <i>Análise de Monotonicidade.....</i>	<i>17</i>
2.2.3 <i>Análise de Pontos Críticos.....</i>	<i>17</i>
2.2.4 <i>Calibração do Modelo.....</i>	<i>17</i>
3. RESULTADOS.....	18
3.1 TESTES DE SANIDADE.....	18
3.2 ANÁLISE DE MONOTONICIDADE.....	19
3.3 ANÁLISE DE PONTOS CRÍTICOS	21
3.4 CALIBRAÇÃO DO MODELO	23
4. PRÓXIMAS ETAPAS	25
REFERÊNCIAS.....	26

RESUMO

Modelos de Machine Learning em produção muitas vezes apresentam comportamentos inesperados, que não foram detectados durante as etapas de treino, teste e validação, ao serem expostos a dados do mundo real. Esses comportamentos inesperados, ou “*bugs*” do modelo, ocorrem por diferentes motivos, como o não cumprimento de regras de negócio, dados de treino rotulados incorretamente e generalizações errôneas do próprio modelo.

Este projeto tem como objetivo desenvolver uma biblioteca na linguagem de programação Python que seja capaz de realizar diagnósticos e produzir relatórios que identifiquem esses *bugs* e comportamentos inesperados, permitindo assim que o cliente IFOOD os corrija.

Essa biblioteca poderá ser utilizada em modelos de classificação binária, já em produção, na forma de “caixa-preta”, que foram treinados com dados tabulares.

Palavras-chave: Machine Learning; Identificação de *bugs*; Modelos em produção.

ABSTRACT

Machine Learning models in production phases sometimes present unexpected behavior, which have not been detected during training, testing or validation phases, when faced with data from the real world. Those unexpected behaviors, or model bugs, have different explanations, such as not following business rules, wrong generalizations, or mislabeled training data.

The goal of this project is to develop a Python library that produces diagnosis and reports that identifies those bugs and allows the client IFOOD to correct them.

This library will be compatible with “black boxes” binary classification Machine Learning models that have been trained with structured/tabular data, that are already in production

Keywords: Machine Learning; bug identification; models in production

1. Introdução

A empresa Ifood.com Agência de Restaurantes Online S.A, conhecida como IFOOD, é uma empresa brasileira que aproxima clientes, restaurantes e entregadores, oferecendo serviços de delivery de restaurantes e mercados, vale-refeição, cartão presente, entre outros. Fundada em 2011, a empresa cresceu exponencialmente no país, possuindo atualmente 80% de participação no mercado de entrega de alimentos.

Nessas diversas áreas de atuação, o IFOOD utiliza modelos de Machine Learning para as mais diversas tarefas, como em prevenção de fraudes de pagamentos, recomendações de restaurantes ou categorias e monitoramento de textos e catálogos. Esses modelos, essenciais para o bom funcionamento da empresa, não são perfeitos, e muitas vezes apresentam comportamentos inesperados, conhecidos popularmente como “*bugs*”, quando já estão em produção, apesar desses comportamentos não terem sido detectados durante as etapas de treinamento, teste e validação.

Um exemplo de um *bug* ou comportamento inesperado seria um modelo de prevenção de fraude que deveria identificar, por exemplo, que quanto maior o número de cartões registrados em uma conta, maior a probabilidade de uma transação realizada por essa conta ser uma fraude, ou seja, essa *feature* deveria ter uma relação monotônica com a probabilidade da predição do modelo. Nas etapas de treinamento e teste, esse comportamento ocorreu como previsto, porém ao ser colocado em produção percebeu-se que para determinadas entradas, ao aumentar o número de cartões registrados em determinados momentos o modelo invertia sua predição, o que significa que essa monotonicidade não estava sendo cumprida.

Esse tipo de comportamento é comum em situações adversariais, ou seja, situações em que existe um agente malicioso tentando burlar o sistema, porém também pode ocorrer em situações não maliciosas, como em casos de clientes que estão tentando fazer uma compra legítima, porém, por causa desses *bugs*, o modelo identifica a transação como uma fraude e a cancela, prejudicando a empresa que deixou de realizar uma venda legítima, e frustrando o cliente que se vê em uma situação injusta.

Modelos de Aprendizado de Máquina mais complexos, como os utilizados pelo cliente IFOOD, apesar de apresentarem excelentes métricas de avaliação como *accuracy*, *precision*, *recall*, *f1 score*, *AUC*, etc., nem sempre são capazes de criar generalizações corretas para casos do mundo real nunca antes vistos, e nem sempre são capazes de obedecer a todas as regras de negócio determinadas pela empresa. Essas falhas, somadas a outros fatores circunstanciais,

permitem a presença de *bugs* e comportamentos inesperados que só são detectados quando o modelo já está em produção.

Este projeto tem como objetivo desenvolver uma biblioteca na linguagem de programação Python que seja capaz de realizar diagnósticos e produzir relatórios que identifiquem esses *bugs* e comportamentos inesperados, permitindo assim que o cliente IFOOD os corrija.

Esses diagnósticos serão produzidos por meio de testes e funções que analisarão métricas como monotonicidade de *features*, valores críticos de *features* e seus impactos na classificação do modelo, comportamento em situações não vistas durante etapas de treinamento e teste, calibração do modelo, teste de sanidade, dentre outros, além da produção de gráficos e outras análises visuais.

O projeto se demonstra relevante para a empresa cliente tendo em vista que possibilitará a identificação de *bugs* e comportamentos inesperados de seus modelos em produção, permitindo que a empresa corrija essas anomalias, evitando assim casos prejudiciais como fraudes ou negação de serviços para clientes legítimos

Por fim, por se tratar de uma biblioteca construída com linguagem de programação, e frameworks *open source*, não haverá problemas regulatórios ou financeiros, podendo a empresa utilizá-la da forma que julgar mais vantajosa.

1.1 Escopo do projeto

O projeto consiste no desenvolvimento de uma biblioteca, na linguagem de programação Python, capaz de produzir diagnósticos e relatórios que permitam a identificação de *bugs* em modelos de Machine Learning de classificação binária, na forma de “caixa-preta”, já em produção.

1.2 Recursos

Por se tratar de uma biblioteca construída em Python, os únicos recursos necessários são computadores capazes de instalar e rodar as dependências necessárias para o funcionamento da biblioteca, como Python e internamente Numpy¹, Scikit-Learn², Pandas³, entre outras.

1.3 Cronograma

O cronograma do projeto, visando à melhor organização e entendimento do que deve ser feito até o final, foi dividido em etapas, as quais têm determinadas em qual mês é realizada. Posteriormente, com o objetivo de se ter uma noção um pouco mais profunda, também se tem uma análise quinzenal das atividades a serem feitas.

As etapas citadas anteriormente são:

- Etapa 1: Compreensão do problema e busca de literatura relevante ao tema
- Etapa 2: Escolha do *dataset* e treinamento de modelos base
- Etapa 3: Construção do algoritmo inicial capaz de identificar violações de regras de negócio tanto nos dados quanto nos modelos
- Etapa 4: Início da construção da ferramenta que será utilizada pela empresa
- Etapa 5: Inclusão de análises SHAP e ataques adversariais
- Etapa 6: Finalização da ferramenta, aprimoramento e generalização para modelos e datasets

Tabela 1 - Mapeamento mensal das etapas do projeto

	<i>Agosto</i>	<i>Setembro</i>	<i>Outubro</i>	<i>Novembro</i>	<i>Dezembro</i>
<i>Etapa 1</i>					
<i>Etapa 2</i>					
<i>Etapa 3</i>					
<i>Etapa 4</i>					
<i>Etapa 5</i>					
<i>Etapa 6</i>					

¹ Numpy: The fundamental package for scientific computing with Python. Disponível em: <https://numpy.org/>

² Scikit-learn: Machine Learning in Python. Disponível em: <https://scikit-learn.org/stable/>

³ Pandas: Python Data Analysis Library. Disponível em: <https://pandas.pydata.org/>

Tabela 2 - Análise quinzenal de atividades do projeto

Quinzena	Início	Atividades
Primeira	8/ago	Entendimento do problema; Busca por literatura para compreensão da área;
Segunda	22/ago	Produção do Relatório Preliminar; Escolha do <i>dataset</i> ; Treinamento de modelos base;
Terceira	5/set	Construção dos algoritmos utilizados na biblioteca; Início da construção da biblioteca;
Quarta	19/set	Implementar feedbacks do cliente; Produção do Relatório Intermediário;
Quinta	3/out	Bancas Intermediárias e Apresentação para Empresa;
Sexta	17/out	Transformar a ferramenta em uma biblioteca Python completa.
Sétima	31/out	Finalização da biblioteca; Validação com cliente e melhorias finais;
Oitava	14/nov	Produção do Relatório Final;
Nona	28/nov	Bancas Finais;
Décima	12/dez	Apresentação para empresa;

1.4 Mapeamento dos stakeholders

Os stakeholders são as pessoas envolvidas e impactadas com o projeto de certa forma e, por conta disso, o grupo inclui pessoas do Insper, da empresa IFOOD e o próprio grupo desenvolvedor do projeto. O mapeamento encontra-se na Tabela 3.

Tabela 3 - Mapeamento dos stakeholders do projeto

Stakeholder	Posição	Papel no Projeto
Thiago Cardoso	Diretor de Data Science do iFood	Instrução, acompanhamento e avaliação

Guilherme Righetto	Cientista de Dados do iFood	Instrução, acompanhamento e avaliação
Prof. Fabrício Barth	Professor do Insper	Supervisão, acompanhamento e orientação
Equipe desenvolvedora do projeto	Estudantes do Insper	Desenvolvimento da biblioteca

1.5 Questões Éticas e Profissionais

A criação e o uso de modelos de Machine Learning envolvem diversas questões éticas e profissionais. A coleta e o uso de dados para o treinamento de modelos é um exemplo de questão ética – e até legal – que o grupo deve se atentar, uma vez que a manipulação desses dados deve respeitar o consentimento e a privacidade dos usuários que os disponibilizaram.

Nesse projeto, porém, a principal questão ética e profissional que se deve atentar é em relação ao comportamento do modelo, em especial a possíveis vieses, tendências e até mesmos preconceitos que esses modelos podem demonstrar.

Como o projeto trata da identificação e posterior correção de bugs desses modelos, deve se atentar para que a forma como essa correção é feita não induza esses vieses, tendências ou preconceitos na tomada de decisões do modelo.

Na prática, essa é uma possibilidade muito rara de acontecer, uma vez que o treinamento em si, que é onde esse tipo de violação ética mais ocorre, não faz parte do escopo do projeto, porém é importante identificar que essa questão ética existe e deve ser levada em consideração na tomada de decisões.

1.6 Revisão do Estado da Arte

Em relação ao Estado da Arte deste projeto, foram utilizados diversos artigos científicos como ponto de partida, entre eles estão PermutedAttack [1], por Sayedmasoud Hashemi et al, e Proposals for model vulnerability and security [2], por Patrick Hall et al. Estes tratavam de assuntos semelhantes ao proposto para o projeto, nos quesitos de ataques adversariais e vulnerabilidades em modelos de Machine Learning.

As ferramentas SHAP [3] e LIME [4] são ferramentas de análise de features e explicabilidade de modelos eficientes e reconhecidas na área de Machine Learning, e serviram como inspiração para a criação de algumas ferramentas do projeto, porém não foram utilizadas diretamente por fugirem do escopo definido.

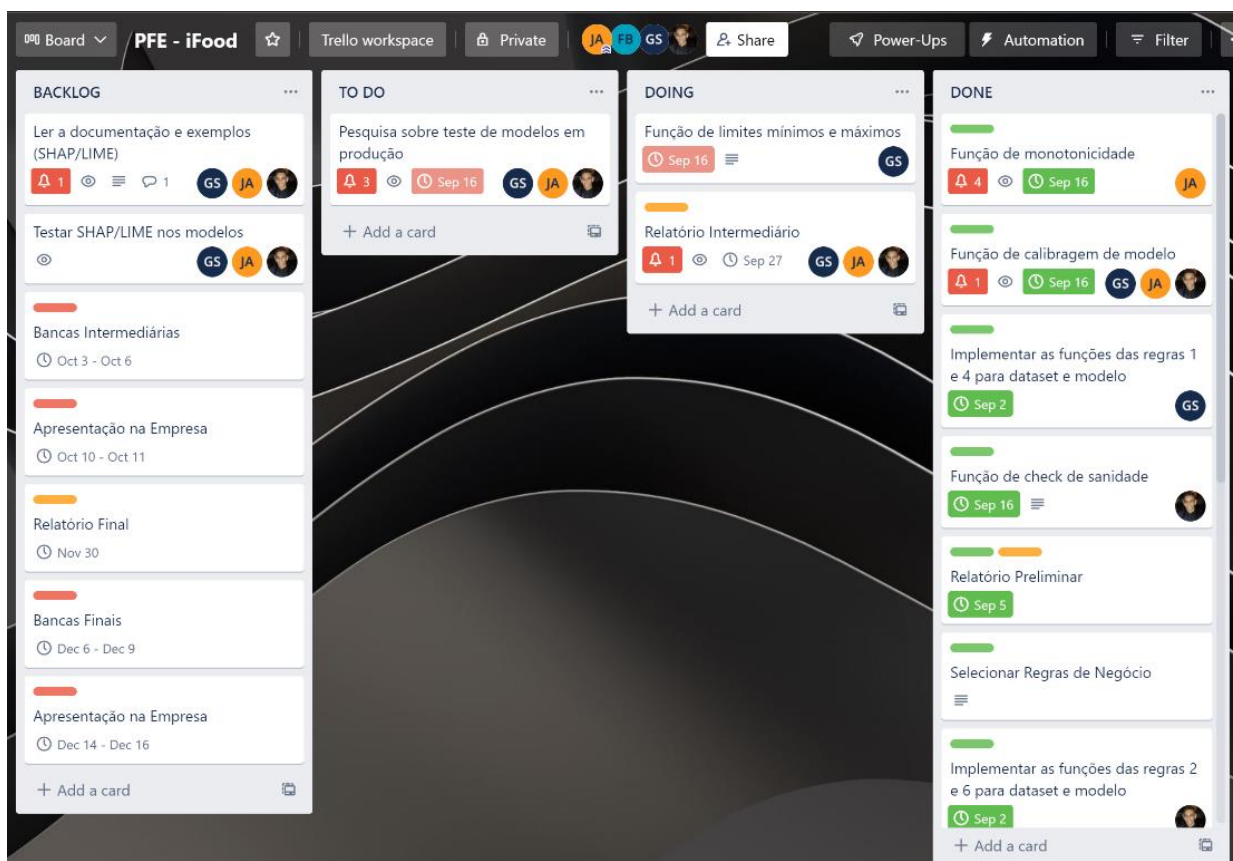
A ferramenta de testes Pytest [5] também serviu de inspiração para a forma como as ferramentas do projeto mostrariam os resultados de suas análises, porém assim como no caso do SHAP e do LIME, não foi utilizada diretamente por fugir do escopo definido.

Esses projetos e artigos considerados estado da arte serviram como inspiração e ideias para o desenvolvimento de uma ferramenta que solucionasse as dores específicas da empresa cliente IFOOD.

2. Metodologia

Seguindo os princípios de Metodologia Ágil, foi inicialmente decidido que a ferramenta de organização das tarefas seguiria o estilo *Kanban*, e a ferramenta utilizada seria o *Trello*, tendo como exemplo de utilização a Figura 1.

Figura 1 – Exemplo da utilização do Trello como ferramenta de Metodologia Ágil



Foi decidido também que o projeto seria armazenado no GitHub, e que as tarefas de cada membro seriam adicionadas ao projeto na forma de *Pull Requests*, necessitando da avaliação e aprovação dos pares do grupo, visando a uma prática de um desenvolvimento profissional.

A forma de divisão das tarefas no tempo foi no formato de Sprints, que eram quinzenais e continham tarefas que duravam diversos períodos diferentes, porém a determinação do quinzenal foi mais presente por conta do intervalo de tempo definido com as reuniões com o cliente IFOOD. Assim, as Sprints foram seguidas da seguinte forma até então:

- Sprint 1: Busca de bibliografia e artigos referentes ao tema em questão, estudo de ferramentas e bibliotecas úteis para o projeto
- Sprint 2: Escolha de um *dataset* e treinamento de diferentes tipos de modelos de Machine Learning de Classificação.
- Sprint 3: Definição de regras de negócio relativas ao *dataset* escolhido e criação de algoritmos de checagem dessas regras.
- Sprint 4: Criação de algoritmos para análise de diversas métricas como monotonicidade de *features*, calibração de modelos, teste de sanidade e intervalos críticos.
- Sprint 5: Aprimoramento dos algoritmos e criação de uma biblioteca em Python que utilize os algoritmos criados.

Os resultados dos trabalhos desenvolvidos em cada Sprint foram discutidos com os membros da empresa cliente IFOOD, que forneciam feedbacks e sugestões para as próximas tarefas, de forma que o grupo sempre realizasse o que fosse mais importante para a empresa.

Esses resultados também foram discutidos com o professor orientador, que analisava tanto o trabalho desenvolvido quanto os feedbacks e sugestões dadas pelo cliente, fornecendo também sua própria visão, que era incorporada na elaboração das próximas Sprints.

A comunicação entre os membros do projeto ocorria de forma presencial, por meio de reuniões nas dependências do Insper, e por meio remoto, utilizando o WhatsApp e Discord como meios de comunicação.

Uma vez que os membros da empresa cliente se encontravam em regime de trabalho remoto, as comunicações entre os membros do projeto e os membros da empresa foram

realizadas também de forma remota, por meio das ferramentas Microsoft Teams e e-mail. A Tabela 4 apresenta informações sobre essas reuniões.

Tabela 4 - Reuniões com Orientador e Empresa Cliente

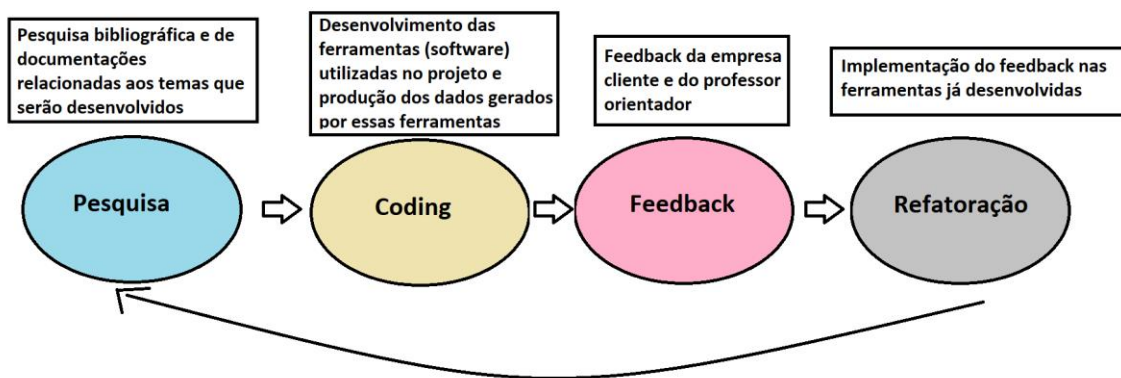
<i>Participantes</i>	<i>Data</i>	<i>Local</i>	<i>Duração</i>	<i>Pauta</i>
<i>Fabício Barth</i>	8/ago.	Remoto	40min	Apresentação do projeto pelo professor e ideias de inicialização.
<i>Thiago Cardoso e Fabício Barth</i>	11/ago.	Remoto	30min	Descrição do problema, proposta e primeiros passos a serem tomados.
<i>Fabício Barth</i>	15/ago.	Inspere	30min	Primeira versão do contexto e objetivos do projeto, soluções de forma macro (SHAP, LIME), agenda e forma de trabalho.
<i>Fabício Barth</i>	22/ago.	Inspere	1h	Levantamento de artigos com temas semelhantes ao problema.
<i>Thiago Cardoso e Guilherme Righetto</i>	24/ago.	Remoto	30min	Aprofundamento do problema, artigos relacionados ao tema, escolha do dataset, treinamento de modelos para teste, ferramentas para exploração de vulnerabilidade.
<i>Fabício Barth</i>	29/ago.	Inspere	1h	Definição de tarefas e atualização do orientador quanto a reunião com o cliente.
<i>Fabício Barth</i>	5/set.	Inspere	1h	Análise do andamento da sprint anterior, feedback e possíveis mudanças.
<i>Thiago Cardoso e Guilherme Righetto</i>	9/set.	Remoto	40min	Especificações de funções mais objetivas para a resolução do problema.
<i>Fabício Barth</i>	12/set.	Inspere	1h	Levantamento de ideias para desenvolver as funções e divisão de tarefas.
<i>Fabício Barth</i>	20/set..	Inspere	1h	Análise das funções definidas na sprint anterior, feedback e possíveis melhorias.

Thiago Cardoso Guilherme Righetto	e	21/set.	Remoto	1h	Refatoração e padronização das funções, ideias para a versão final da ferramenta.
--	---	---------	--------	----	--

2.1 Coleta de dados sobre o projeto

A partir das reuniões iniciais realizadas com a empresa cliente e com o professor orientador, foi definido um fluxo de descoberta e produção de dados, conforme mostra o diagrama abaixo na Figura 2:

Figura 2 – Diagrama do fluxo de trabalho exercido pelo grupo durante o projeto



A primeira etapa de coleta de dados, após as pesquisas bibliográficas referentes ao tema, foi a escolha de datasets que seriam utilizados para o treinamento dos modelos de Machine Learning, e posteriormente também para os testes das ferramentas desenvolvidas.

Foram escolhidos dois datasets de classificação binária: *Synthetic Financial Datasets For Fraud Detection* [6], um *dataset* sintético de transações financeiras, contendo cerca de 6 milhões e 300 mil transações, para classificação de fraudulentas e não fraudulentas, e *Titanic - Machine Learning from Disaster* [7], um *dataset* com informações de 891 passageiros do navio Titanic para classificação de sobrevivência ou não no famoso naufrágio.

Ambos foram coletados do site Kaggle, e a diferença entre os temas foi escolhida propositalmente para testar a capacidade das ferramentas da biblioteca em lidar com datasets e modelos diversos, uma vez que o seu objetivo é analisar modelos na forma de “caixa-preta”.

Em sequência, foram treinados 3 modelos de Machine Learning de classificação com base no dataset *Synthetic Financial Datasets For Fraud Detection* e 4 modelos de Machine Learning de classificação com base no dataset *Titanic - Machine Learning from Disaster*.

Em relação ao *dataset* sintético de fraude, foi utilizada a biblioteca *scikit-learn* para criar os seguintes modelos de classificação: Random Forest Classifier, Logistic Regression Classifier e XGBoost Classifier.

Para o *dataset* do naufrágio do navio Titanic, foram criados também os modelos de classificação Random Forest Classifier, Logistic Regression Classifier e XGBoost Classifier, além do modelo Stochastic Gradient Descent Classifier.

Assim como no caso dos datasets, as ferramentas desenvolvidas para a biblioteca devem ser capazes de lidar com diferentes tipos de modelos de Machine Learning de classificação binária, portanto foram escolhidos modelos que utilizam algoritmos de classificação distintos para efeitos de teste.

Uma vez que a biblioteca lida com esses modelos na forma de “caixa-preta”, as métricas tradicionais de avaliação de performance como *accuracy*, *f1-score* e *AUC*, apesar de terem sido analisadas, independem para a construção e testes das ferramentas da biblioteca.

2.2 Análise dos Dados Coletados

Com os datasets escolhidos e modelos treinados, iniciou-se o processo de desenvolvimento das ferramentas da biblioteca que realizam a análise do modelo em busca de bugs e comportamentos inesperados. Essas ferramentas se dividem em quatro principais frentes, determinadas pela empresa cliente como pontos importantes de se analisar: testes de sanidade, análise de monotonicidade, análise de pontos críticos e calibração do modelo. Cada uma destas análises está detalhada a seguir.

2.2.1 Testes de Sanidade

Os testes de sanidade são um dos testes mais simples que se tem no quesito da programação e servem justamente para serem executados de forma muito rápida e definir se o comportamento do sistema está sendo como o esperado e, caso não esteja, dá um sinal de alerta, demonstrando que não precisa mais fazer outros testes, mas sim consertar a solução.

No caso da problemática do grupo, é necessário que sejam feitos testes de sanidade para confirmar se um modelo de Machine Learning está se comportando adequadamente, isto é, devolvendo a classificação esperada, para casos em que ele nunca pode errar. Assim, é necessário que se tenha o modelo e algumas instâncias do dataset utilizado com casos denominados básicos, onde nossa ferramenta faz a predição e determina se no caso fornecido o modelo passa por estes testes.

É muito relevante para a empresa ter algo assim, pois bugs podem ser identificados desde o início, já que existem casos em que o modelo nunca poderia errar, a exemplo de um

cliente que faz uma compra legítima, sem nenhuma suspeita. Dessa forma, consegue-se evitar problemas futuros caso o modelo apresente problemas desde o início nesses testes.

2.2.2 Análise de Monotonicidade

Algumas regras de negócio dentro da empresa cliente exigem que o comportamento de determinadas features em relação às probabilidades de predição de seus modelos possuam uma relação monotônica, ou muito próxima disso.

Os algoritmos de análise de monotonicidade tem por objetivo identificar a existência ou não dessa relação, e caso não se tenha uma relação monotônica, calcular o erro quadrático médio entre a curva “*feature* vs probabilidade de predição” e a versão monotônica aproximada dessa curva, permitindo a análise se essas regras de negócio estabelecidas estão sendo cumpridas.

2.2.3 Análise de Pontos Críticos

A análise de pontos críticos tem por objetivo identificar valores de determinadas *features* que causam um impacto muito grande na probabilidade de predição do modelo, além da identificação de intervalo de valores onde a classificação de um modelo muda.

Essa análise é feita a partir de um modelo, uma *feature* e um *range* de valores que se quer analisar. A partir dessas informações o algoritmo calcula como o modelo se comporta em situações nas quais determinada *feature* possui esse range de valores, em especial quando esses valores são maiores que o valor máximo ou menores que o valor mínimo vistos nos conjuntos de treinamento e teste, buscando situações em que pequenas alterações nesses valores causam uma variação brusca na probabilidade de predição.

Quando um modelo é muito complexo é possível que faça generalizações para casos nunca vistos que não correspondam ao comportamento do mundo real, gerando *bugs* ou comportamentos inesperados, o que pode ser explorado por atacantes com o objetivo de manipular o resultado do modelo.

Essa análise permite que a empresa encontre esses pontos críticos e tome as decisões necessárias para que esses *bugs* sejam corrigidos.

2.2.4 Calibração do Modelo

As probabilidades de predição de um modelo de Machine Learning estão calibradas quando a predição de uma classe está correta 100 x p por cento das vezes, sendo p o valor de confiança.

O algoritmo de verificação de calibração faz uma comparação com um modelo perfeitamente calibrado, utilizando a métrica *Brier Score Loss*, para calcular o grau de calibração.

Como os modelos em produção estão sendo constantemente re-treinados e alimentados com novos dados, é importante que a empresa tenha controle da variação da calibração desses modelos entre diferentes versões, uma vez que essa métrica é utilizada para tomada de decisões em diferentes áreas dentro da empresa.

3. Resultados

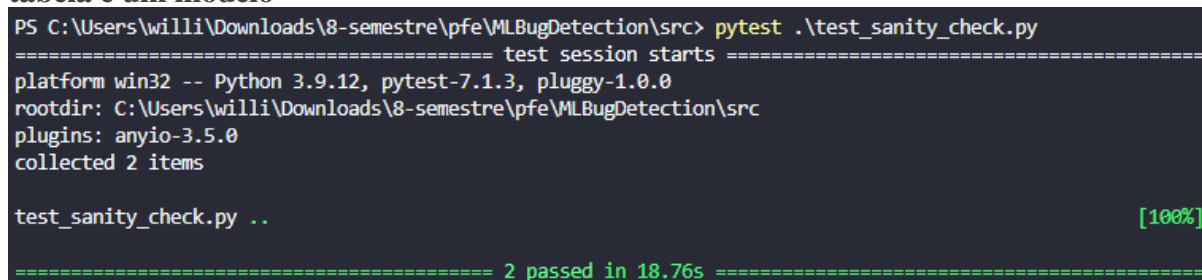
Por meio das ferramentas descritas no tópico anterior, foi possível realizar a análise dos dados coletados e dos modelos treinados, cujos resultados se encontram em sequência para cada função implementada.

3.1 Testes de Sanidade

Como citado anteriormente, o Pytest é uma ferramenta que se assemelha no quesito de testes e, seguindo recomendações por conta disso, o grupo implementou essa função de teste de sanidade do modo tradicional e com a biblioteca de testes do Python. Tendo em vista que não é uma função que possui gráficos, já que ela serve como uma conferência básica para o modelo: ao passar algumas instâncias para o modelo e os resultados esperados, o modelo tem que resultar igual em todas elas. Caso isso não ocorra, é um sinal de que o modelo não está adequado e deve ser revisto.

Na Figura 3, é possível observar o resultado obtido após utilizar o Pytest com algumas instâncias e com os modelos Logistic Regression e Random Forest, demonstrando que esses modelos apresentam o comportamento esperado nesses exemplos básicos.

Figura 3 – Resultado no *prompt* do teste de sanidade para determinadas linhas de uma tabela e um modelo



```

PS C:\Users\willi\Downloads\8-semester\pfe\MLBugDetection\src> pytest .\test_sanity_check.py
===== test session starts =====
platform win32 -- Python 3.9.12, pytest-7.1.3, pluggy-1.0.0
rootdir: C:\Users\willi\Downloads\8-semester\pfe\MLBugDetection\src
plugins: anyio-3.5.0
collected 2 items

test_sanity_check.py .. [100%]

===== 2 passed in 18.76s =====

```

É importante ressaltar que o Pytest não será utilizado em nosso escopo, porém é muito importante para se ter uma comparação de demonstração de resultados e dar uma ideia à empresa cliente se é algo parecido com o que almejam.

Esse teste é muito relevante para a companhia já que o intuito é passar como parâmetro um modelo e exemplos básicos que o modelo nunca pode errar, o que daria uma confiança maior para a continuação de seu deploy e utilização real, facilitando o trabalho e fazendo com que não haja bugs em casos básicos, demonstrando a importância do papel do grupo neste tópico.

3.2 Análise de Monotonicidade

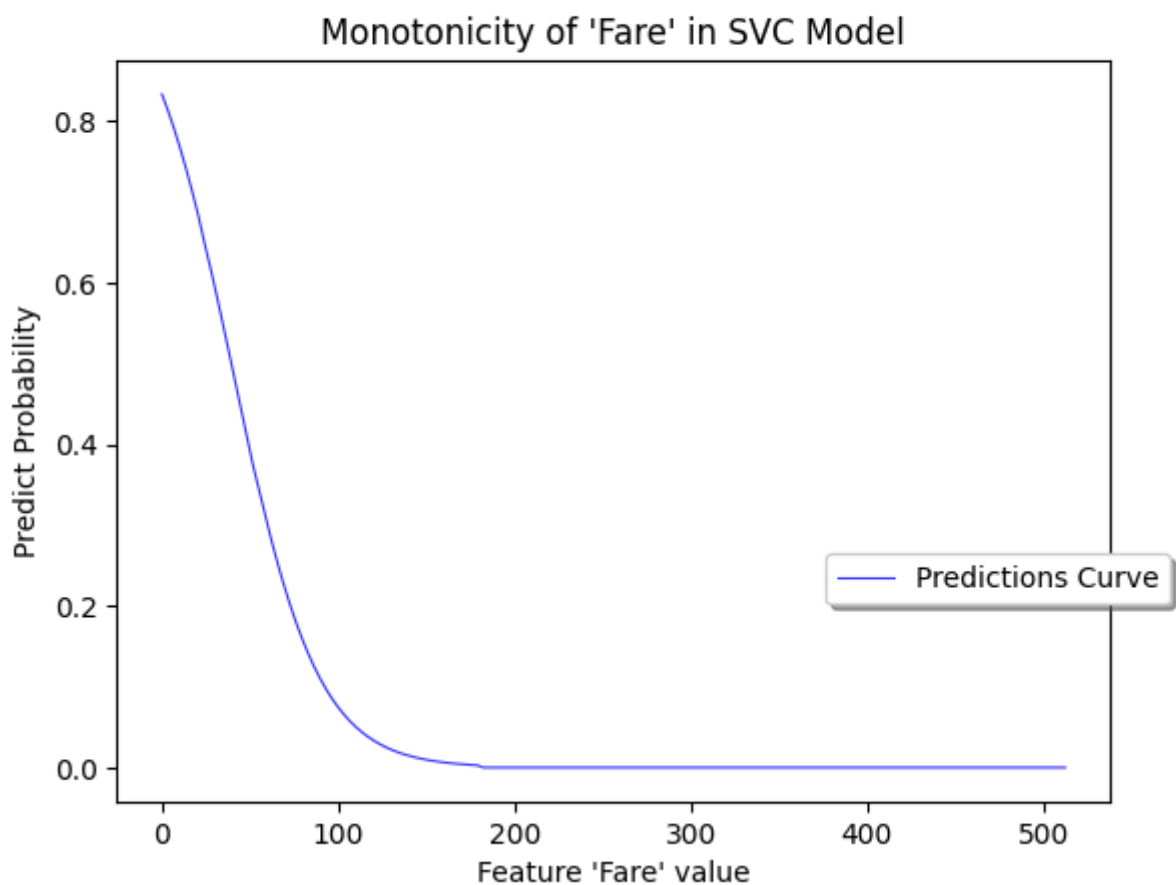
A ferramenta de análise de monotonicidade é capaz de identificar a relação monotônica ou não entre determinada *feature* e a probabilidade de predição do modelo, além de calcular o erro quadrático médio entre a curva de *feature* com sua probabilidade de predição e a versão monotônica aproximada dessa curva.

Na Figura 4, é possível identificar um exemplo de *feature* (Fare, ou valor da passagem do Titanic) com comportamento monotônico, qual intervalo esse comportamento ocorre e qual o tipo de modelo utilizado (Support Vector Classification).

Figura 4 – Comportamento monotônico da *feature* “fare” do dataset do Titanic com o modelo SVC

Model: SVC

Feature Fare has monotonic behavior between ranges 0.0 and 512.3292



Na Figura 5 e Figura 6 a ferramenta é capaz de identificar casos em que não há um comportamento monotônico, identificando o intervalo que isso ocorre, além de calcular a melhor aproximação monotônica desse conjunto de dados, e calcular o erro quadrático médio entre esses dados e a aproximação.

Figura 5 – Comportamento não monotônico da *feature* “step” no modelo XGBoost e seu MSE reportado

Model: XGBClassifier

Warning: Feature 'step' doesn't have monotonic behavior between ranges 1 and 743

Feature 'step' has a monotonicity MSE score of 0.009391251020133495

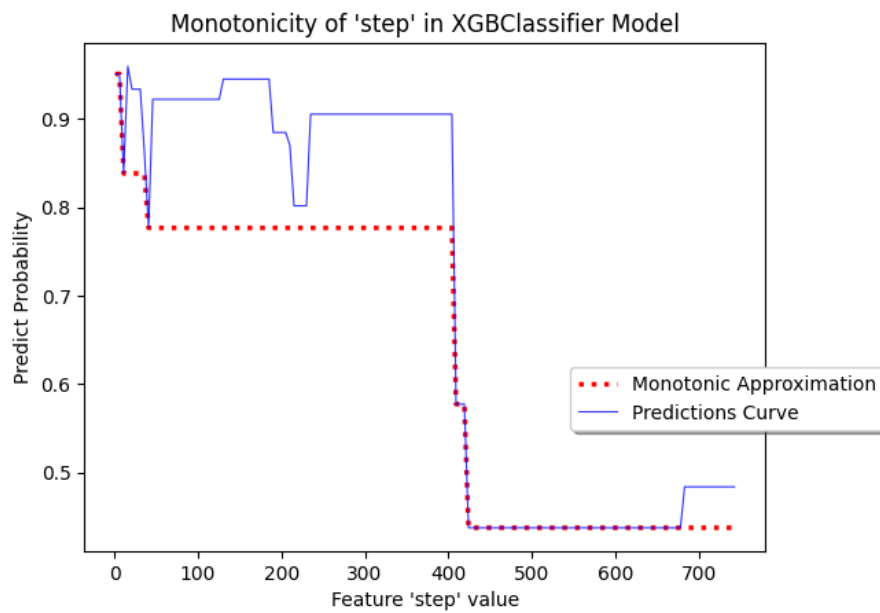
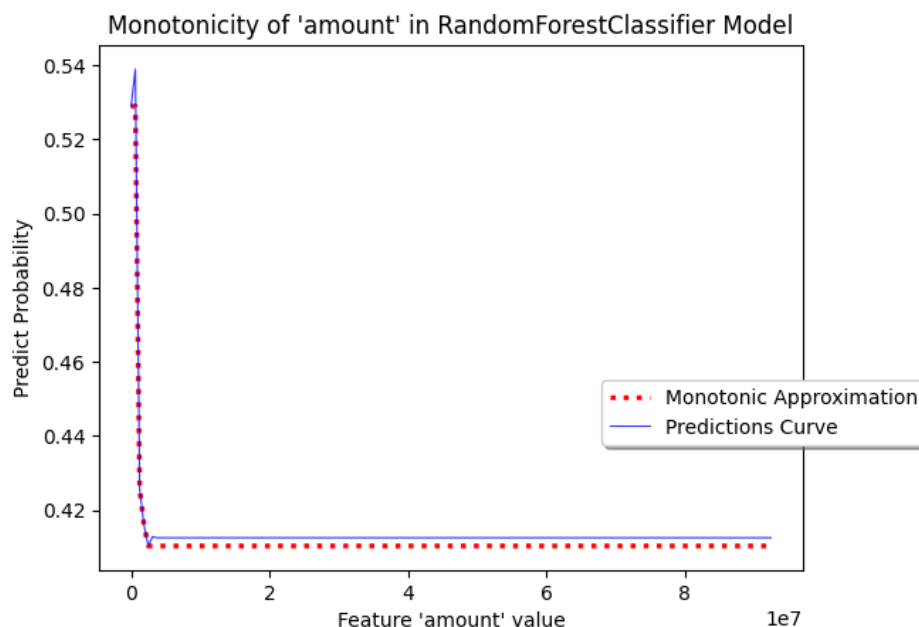


Figura 6 – Comportamento não monotônico da *feature* “amount” no modelo Random Forest e seu MSE reportado

Model: RandomForestClassifier

Warning: Feature 'amount' doesn't have monotonic behavior between ranges 0.0 and 92445516.64

Feature 'amount' has a monotonicity MSE score of 5.440966907217464e-06



3.3 Análise de Pontos Críticos

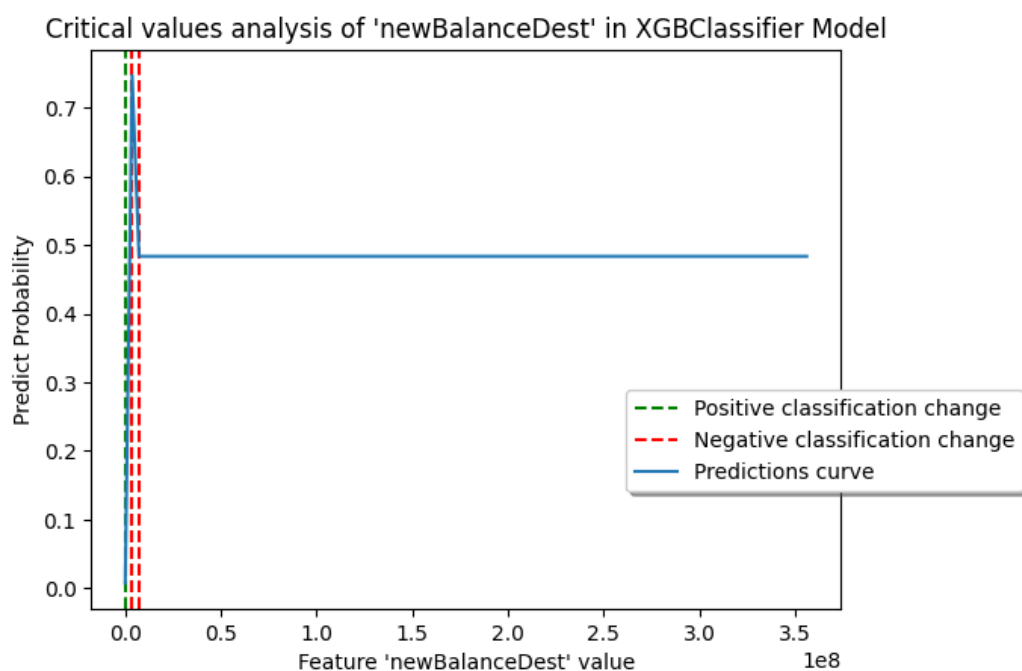
A ferramenta de análise de pontos críticos é capaz de identificar quais valores de quais *features* resultaram na maior variação da probabilidade de predição do modelo, além de mostrar em quais momentos essa variação resultou em uma mudança de classificação.

Além disso, essa ferramenta permite uma análise visual, por meio de gráficos, do comportamento dessas *features*, e a localização desses pontos críticos encontrados, conforme os exemplos abaixo.

A Figura 7 demonstra o impacto da coluna “newBalanceDest”, que representa o valor da conta bancária de destino do dataset sintético de transições bancárias. Pode-se observar que em um intervalo muito pequeno de valores, a probabilidade do modelo se altera bruscamente, tanto em sentido de crescimento quanto em sentido de decrescimento, o que gera mudança na classificação de fraude ou não pelo XGBoost, modelo analisado neste caso específico.

Figura 7 – Resultado da ferramenta na análise da coluna “newBalanceDest” com o modelo XGBoost

```
Highest positives changes identified on feature 'newBalanceDest':
  From values 0.0 to 3597770.494 : Predict proba diff = 0.7393982410430908
  Warning, prediction has changed
Highest negatives changes identified on feature 'newBalanceDest':
  From values 3597770.494 to 7195540.988 : Predict proba diff = -0.2629999816417694
  Warning, prediction has changed
```



A partir disso, com esse gráfico, é muito importante tomar uma ação, seja reavaliando o modelo, ou analisar melhor o comportamento dessa *feature* em específico. Este não é o escopo deste projeto, porém a ferramenta, que está dentro do objetivo, consegue apontar essas mudanças de comportamento repentinas e alertar quem está analisando.

A Figura 8 também demonstra duas mudanças repentinas de probabilidade de predição, como demonstra as impressões. A primeira, foi positiva, porém isso não alterou a determinação de fraude ou não na classificação do modelo e, por conta disso, não há a demonstração no gráfico, porém posteriormente há uma diferença grande negativa, que acaba mudando a classificação final.

Figura 8 – Modelo Random Forest e o impacto da *feature* “amount” com seus valores críticos

Highest positives changes identified on feature 'amount':

From values 0.0 to 933793.097 : Predict proba diff = 0.009985491924094925

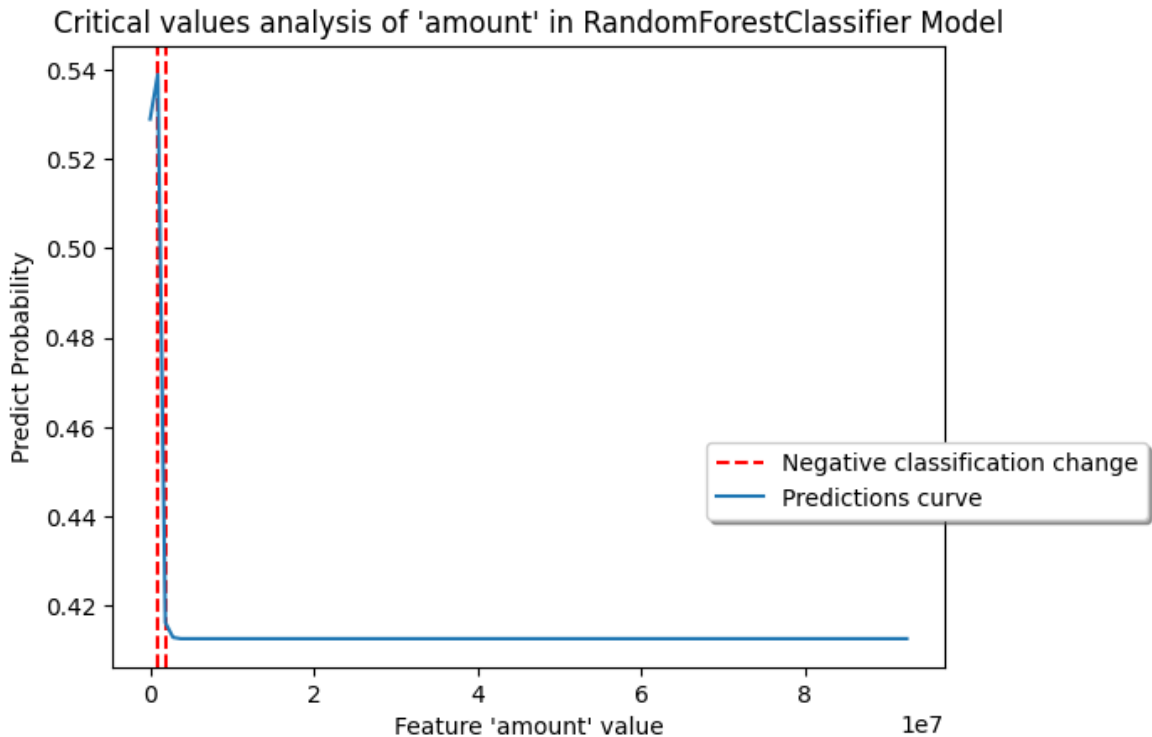
Highest negatives changes identified on feature 'amount':

From values 933793.097 to 1867586.195 : Predict proba diff = -0.12300000000000005

Warning, prediction has changed

From values 1867586.195 to 2801379.292 : Predict proba diff = -0.0030000000000000027

From values 2801379.292 to 3735172.389 : Predict proba diff = 0.0



3.4 Calibração do Modelo

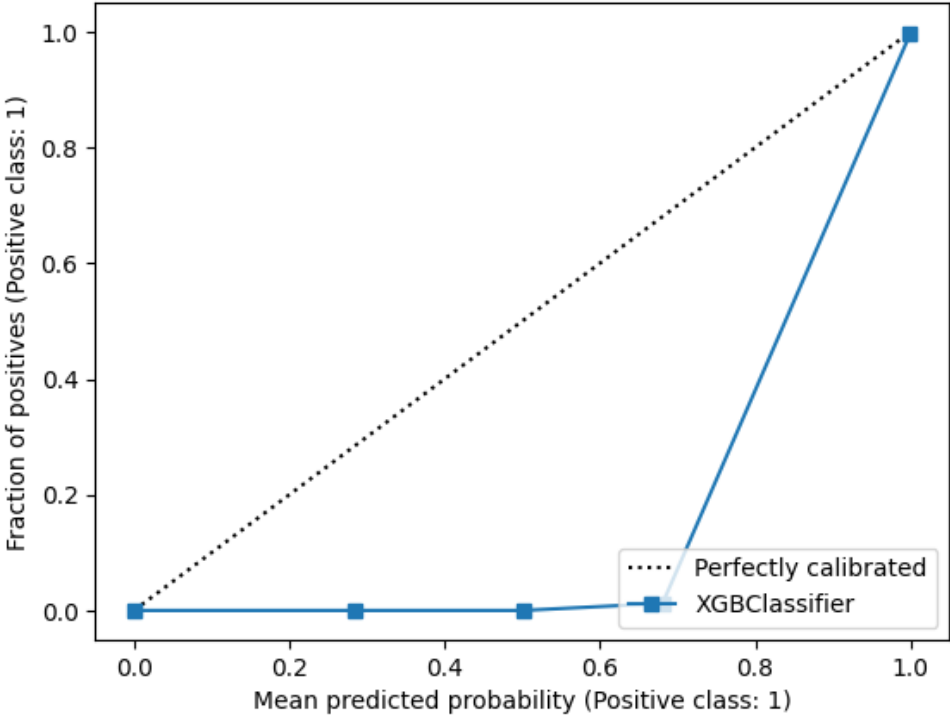
O algoritmo de calibração do modelo é capaz de calcular o *Brier Score Loss*, que é uma métrica calculada por meio do erro quadrático médio entre a probabilidade predita e o resultado real, além de gerar uma análise visual, por meio de gráficos, da calibração do modelo comparada a um modelo perfeitamente calibrado.

Uma vez que os modelos em produção da empresa cliente estão em constante aprimoramento com base em novos dados gerados, é essencial ter um controle das métricas de calibração ao longo do tempo, uma vez que essas métricas são utilizadas na tomada de decisões em diversas áreas dentro da empresa.

As Figura 9 e Figura 10 são exemplos feitos com os modelos XGBoost e Logistic Regression treinados com o dataset *Synthetic Financial Datasets For Fraud Detection*.

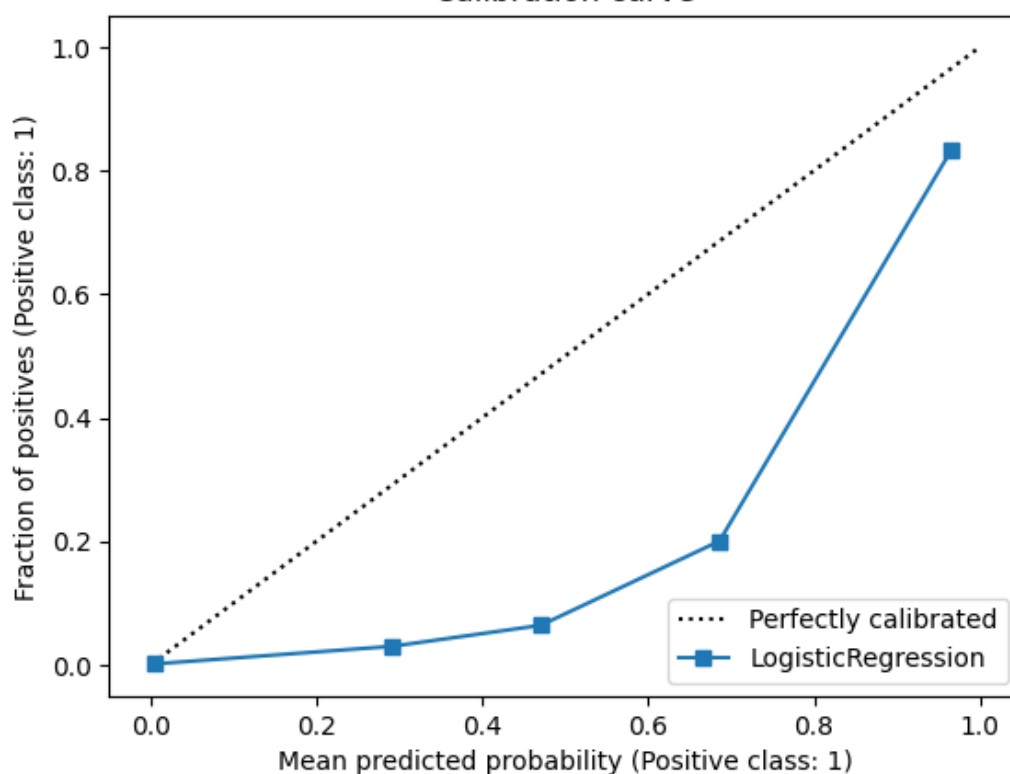
Figura 9 – Curva de calibração do modelo XGBoost construído pelo grupo

Calibration curve



Brier Score Loss: 3.798964224303799e-05 for model XGBClassifier

Figura 10 – Curva de calibração do modelo Logistic Regression construído pelo grupo
Calibration curve



Brier Score Loss: 0.003402447847732624 for model LogisticRegression

4. Próximas etapas

Conforme estabelecido no cronograma, foram estabelecidas quatro frentes de análise de modelos de Machine Learning, e desenvolvidas ferramentas capazes de realizar essas análises propostas em datasets e modelos de testes.

As próximas etapas do projeto consistem em transformar essas ferramentas em uma biblioteca Python, para que a empresa cliente possa baixá-la e testá-la em seus próprios modelos e aprimorar os algoritmos com base no feedback recebido.

A partir dessa análise inicial, outros pontos de interesse serão levantados pela empresa cliente, e a partir deles novas ferramentas e algoritmos serão incorporados à biblioteca, permitindo uma análise cada vez mais completa da presença de bugs em seus modelos de Machine Learning.

Referências

- [1] HASHEMI, M., FATHI, A. PermuteAttack: Conterfactual Explanation of Machine Learning Credit Scorecards. **ArXiv, ago. 2020.** Disponível em: <https://arxiv.org/abs/2008.10138>. Acesso em: 20 ago. 2022.
- [2] HALL, P., Proposals for model vulnerability and security. **O'Reilly, mar. 2019.** Disponível em: <https://www.oreilly.com/content/proposals-for-model-vulnerability-and-security/#F3>. Acesso em: 10 ago. 2022.
- [3] SHAP (SHapley Additive exPlanations). Disponível em: <https://shap.readthedocs.io/en/latest/index.html>
- [4] RIBEIRO, Marco Tulio; SINGH, Sameer; GUESTRIN, Carlos. " Why should i trust you?" Explaining the predictions of any classifier. Disponível em: <https://arxiv.org/abs/1602.04938>
- [5] Pytest: Python testing tool that helps you write better programs. Disponível em: <https://docs.pytest.org/en/7.1.x/>.
- [6] Synthetic Financial Datasets For Fraud Detection. Disponível em: <https://www.kaggle.com/datasets/ealaxi/paysim1>
- [7] Titanic - Machine Learning from Disaster. Disponível em: <https://www.kaggle.com/competitions/titanic/data>