Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

TAYLON LUAN CONGIO MARTINS

TRABALHO DE CONCLUSÃO DE CURSO

ANÁLISE DE RISCO DE CRÉDITO BASEADA EM APRENDIZADO DE MÁQUINA.

ORIENTADOR:
ROMIS RIBEIRO DE FAISSOL ATTUX

CAMPINAS

2024

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

TAYLON LUAN CONGIO MARTINS

ANÁLISE DE RISCO DE CRÉDITO BASEADA EM APRENDIZADO DE MÁQUINA.

Trabalho apresentado ao curso de Engenharia Elétrica apresentado à UNICAMP, como parte dos requisitos para obtenção do Bacharel em Engenharia Elétrica

Orientador(a): Romis Ribeiro de Faissol Attux

CAMPINAS

2024

aos meus queridos pais, professores e irmão pela confiança.

Agradecimentos

Ao Professor Romis, orientador e amigo, por sua paciência e incentivo nos momentos difíceis e pelas oportunidades que me proporcionou.

Aos professores Levy Boccato, Fernando Von Zuben e Ting, que mesmo não sendo orientadores deste trabalho ofereceram valiosas contribuições no ensino de aprendizado de máquina e ciência de dados que inspiraram este trabalho.

SUMÁRIO

RESUMO.....6
1. INTRODUÇÃO.....7
2. OBJETIVOS
3. METODOLOGIA.....7
    a. Dados Utilizados.....13
    b. Análise dos dados.....13
4. DESENVOLVIMENTO.....19
5. RESULTADOS.....32
6. CONCLUSÃO.....57

REFERÊNCIAS BIBLIOGRÁFICAS.....58

APÊNDICE.....60

ANEXO.....68

RESUMO

O objetivo deste trabalho é desenvolver um classificador binário para análise de crédito sob duas instância deste problema: na base de dados há presença de labels (histórico) e na base de dados não há presença de labels (sem histórico). Esse segundo problema surge da necessidade de se lançar um produto de crédito onde não existe histórico de bons e maus pagadores. Portanto, começaremos desenvolvendo um modelo de aprendizado de máquina supervisionado que se utilizam dos labels. Prosseguindo, a abordagem para resolver o mesmo problema sem labels se baseará em recriar os labels com uso de pseudo-labels gerados por aprendizado não-supervisionado e então treinar o mesmo modelo da primeira parte com os pseudo-labels para observar o impacto nas métricas de desempenho. Por fim, numa última parte, a reconstrução de labels usando pseudo labels será analisada usando uma composição de aprendizado supervisionado e não-supervisionado. Os dados utilizados são do clássico dataset German Credit Data que está disponível em repositório aberto da University of California Irvine.

Palavras-chave: aprendizado de máquina, análise de crédito, aprendizado supervisionado, aprendizado não-supervisionado. aprendizado semi-supervisionado.

1. INTRODUÇÃO

A análise de crédito é um dos problemas clássicos e permanentes na área de finanças, sendo necessário a constante evolução de modelos aplicados a este problema que classifiquem clientes como potenciais pagadores ou não de um empréstimo oferecido por uma instituição financeira. Neste trabalho, este problema será abordado com a ferramenta tecnologia em maior evidência nos últimos anos: o aprendizado automático de máquina. Tendo a área chegado até ao prêmio Nobel de Física em 2024. A abordagem com aprendizado de máquina recorrerá a princípio ao uso de aprendizado supervisionado, utilizando vários algoritmos para benchmark, este é o problema clássico de classificação de crédito, onde um modelo é criado em cima de true-labels. Adiante, o mesmo problema será abordado sem o uso de true-labels, simulando a ausência de histórico, ou seja, sem a possibilidade, de partida, de usar o aprendizado supervisionado.

2. OBJETIVOS

O objetivo deste trabalho é desenvolver um modelo de aprendizado de máquina para o problema de análise de crédito para duas instâncias desse problema: a base de dados possui labels (histórico) e a base de dados não possui labels (sem histórico). Na primeira parte, será desenvolvido um classificador binário que desempenhe bem. Na segunda parte, será desenvolvida uma metodologia que aproxime o desempenho obtido no problema sem labels do problema com labels. Por fim, uma comparação entre as soluções encontradas será feita, avaliando dificuldades e possíveis melhorias a serem implementadas.

3. METODOLOGIA

Todo este trabalho foi programado com Python no ambiente Google Collaboratory. Os principais pacotes utilizados incluem: scikitlearn, pandas, numpy, matplotlib e keras/tensorflow.

A metodologia seguiu um processo iterativo de ajustes de hiperparâmetros e avaliação de métricas de desempenho usando dois conjuntos de dados: treinamento e validação.

Para o caso de geração de pseudo-labels, os true-labels foram utilizados para a avaliação dos hiperparâmetros dos modelo não-supervisionados e da configuração do comitê de máquinas, o que é um processo incomum em treinamentos não-supervisionados, visto que, geralmente não há a existência de true-labels quando se aplica métodos não-supervisionados, daí a peculiaridade deste trabalho. Em resumo, os true-labels são usados como guias de uma configuração de algoritmos não-supervisionados de forma a alcançar um configuração de hiperparâmetros que melhore a rotulação de pseudo-labels.

Criar um modelo supervisionado que é treinado com pseudo-labels gerados por modelos não-supervisionados é importante porque mesmo que os pseudo-labels gerados são perfeitos (todos os correspondentes pseudo-labels correspondem ao seu true-labels), para novos dados (nunca vistos pelo modelo) o padrão subjacente dos dados gerado pode se alterar, portanto, mudando a saída do modelo não-supervisionado (o modelo pode encontrar uma nova

configuração para os dados brutos). Já com um modelo supervisionado treinado, a previsão de labels para novos dados não altera o padrão subjacente dos dados de treinamento.

O ‘split’ na base de dados para separar dados de treinamento e validação produziu um conjunto de treinamento de 800 amostras e um conjunto de validação de 200 amostras (141 classe 1, 59 classe 2). Os mesmos dados (mesmo índice) compõem esses conjuntos para todos experimentos neste trabalho, o que diminui sua estocasticidade. O fluxograma para cada metodologia aplicada são representadas nos fluxogramas de projetos a seguir.

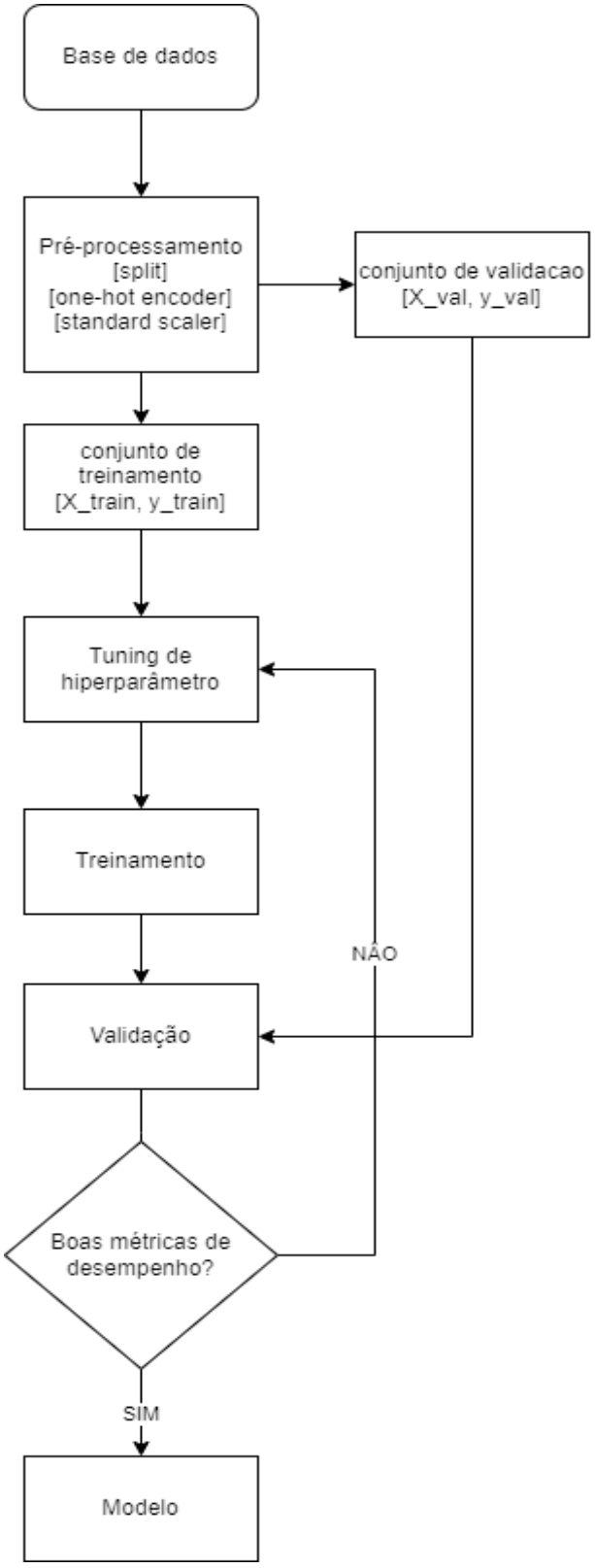

Figura 1 - Fluxograma do treinamento supervisionado.

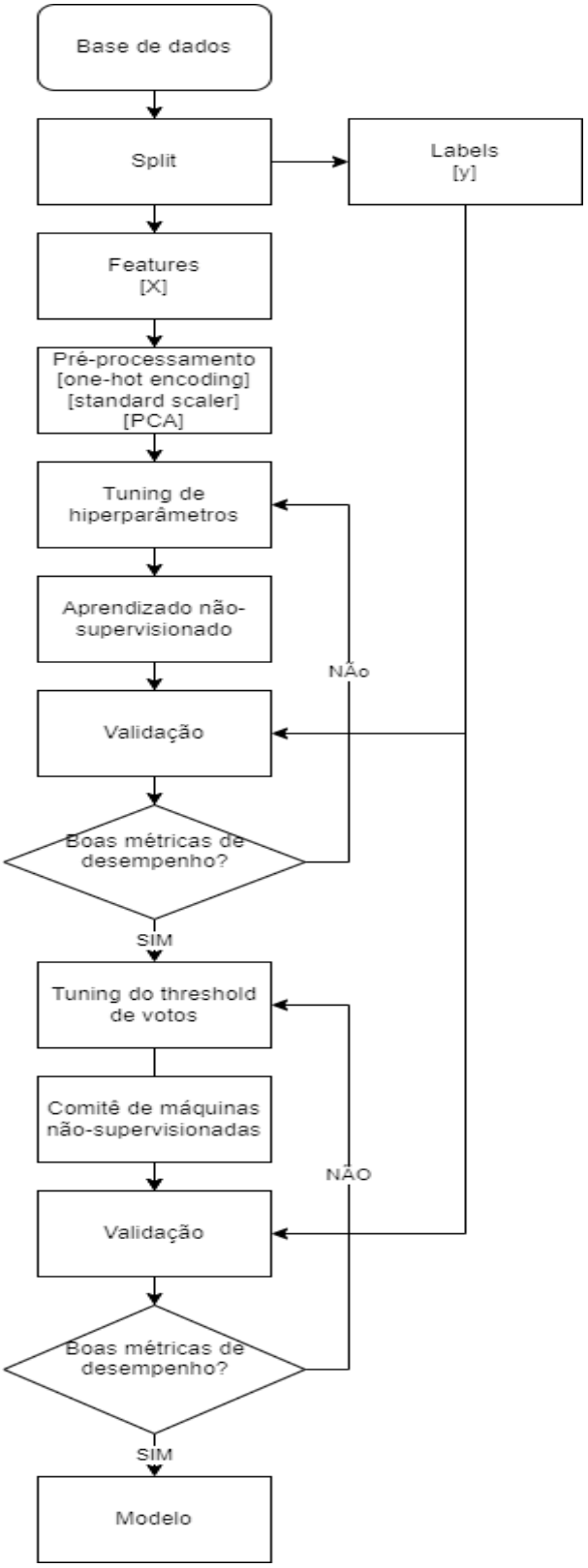

Figura 2 - Fluxograma do treinamento não-supervisionado.

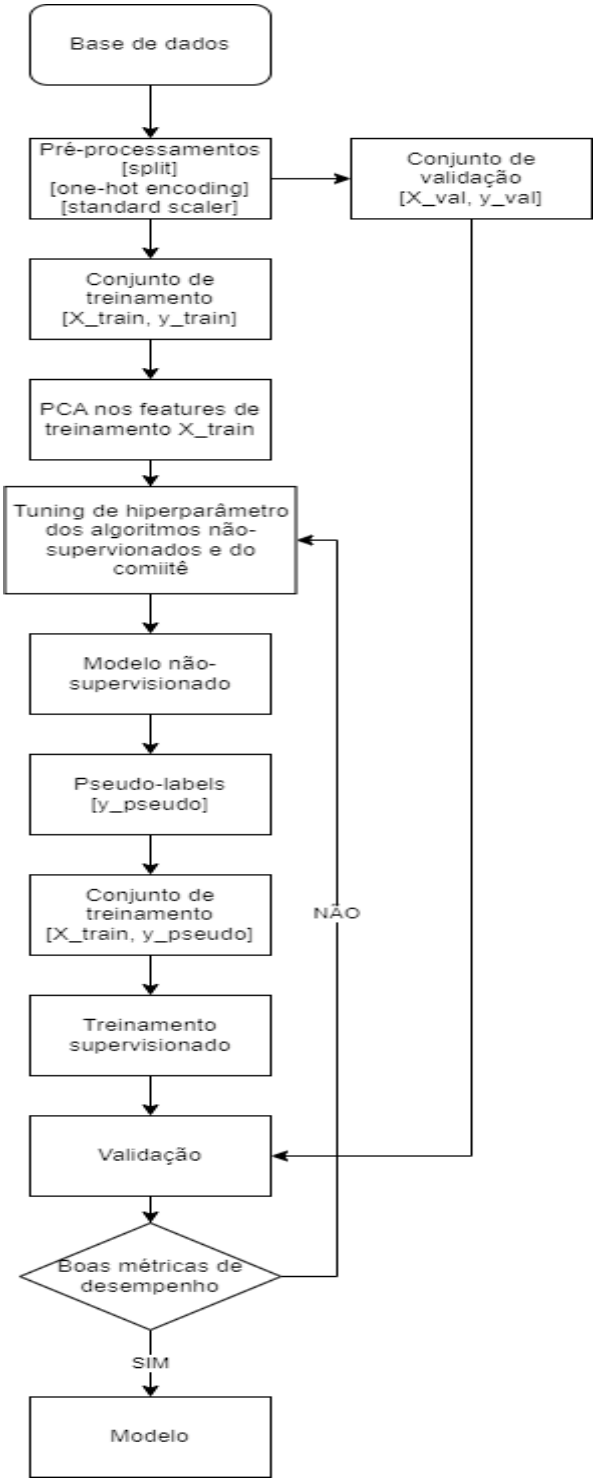

Figura 3 - Fluxograma da combinação de treinamento supervisionado e não-supervisionado.

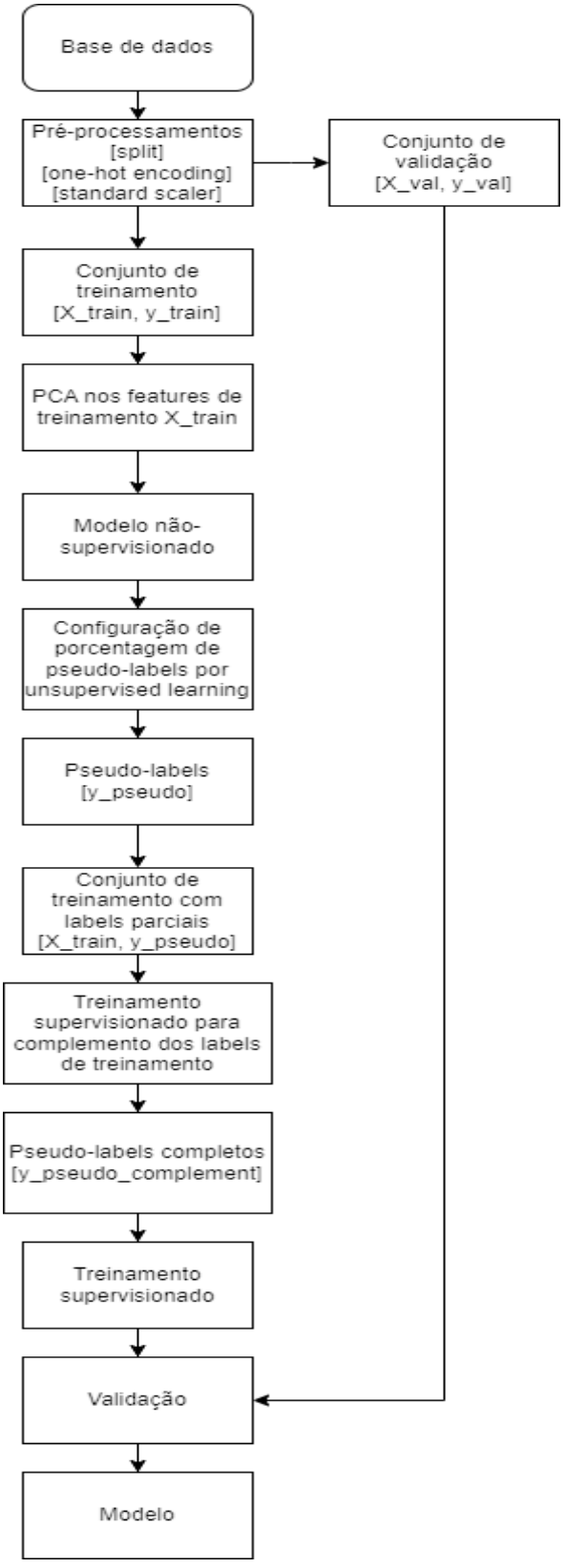

Figura 4 - Fluxograma do treinamento semi-supervisionado.

3.1. Dados Utilizados

A base de dados utilizada é intitulada “German Credit data” e possui como fonte o professor Dr. Hans Hofmann da Universidade de Hamburgo. Essa base de dados se encontra disponível online no site da Universidade da Califórnia, Irvine [1]. Existem duas versões da base de dados, uma com um mistura de dados numéricos com dados categóricos e outra com dados puramente numéricos. A primeira versão foi a utilizada no trabalho.

A base de dados possui 1000 amostras com 20 atributos e 1 coluna de labels (targets), portanto, dimensão 21. A base de dados é desbalanceada, com 700 labels de classe “1” (bons pagadores) e 300 labels de classe “2” (maus pagadores).

Os atributos (features) estão distribuídos em 7 numéricos e 13 categóricos (no apêndice A é apresentado como cada atributo está organizado seguido de uma breve descrição de cada atributo).

3.2. Análise dos dados

A análise de distribuição dos dados originais foi separada em dados numéricos e dados categóricos.

Observa-se na figura 5, por exemplo, que a maior parte dos clientes possuem a idade entre 25 e 40 anos, que o valor médio disponível em conta-corrente é de aproximadamente 2500 DM e que a maioria dos clientes mora na mesma residência há 4 anos. Conforme imagem abaixo.

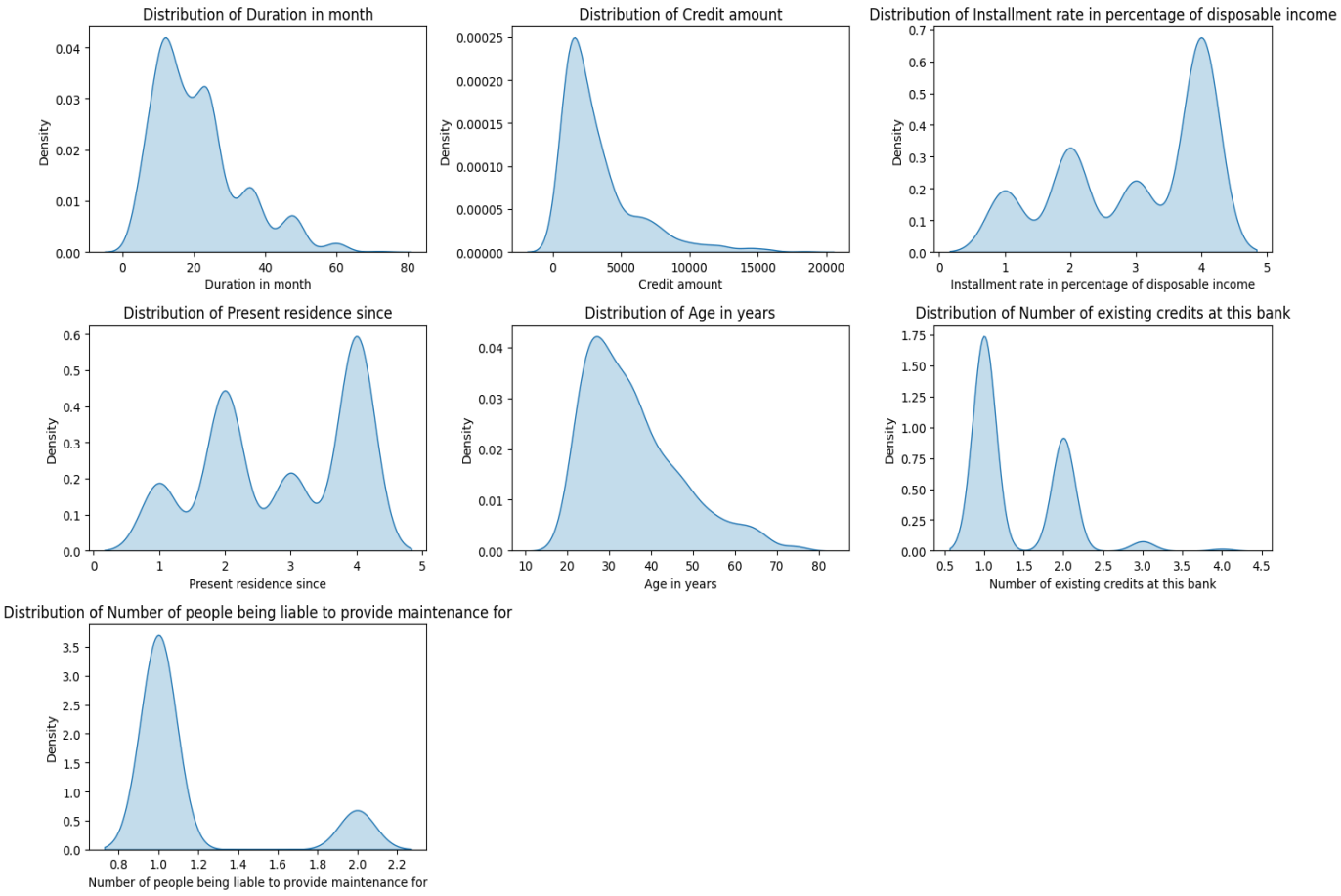

Figura 5 - Distribuição dos dados numéricos.

Quando analisamos a distribuição dos dados sob a ótica dos gráficos de caixa, observa-se que para os atributos que indicam a porcentagem de renda que o cliente possui para pagar o empréstimo e os anos de residência na atual residência não possuem outliers (figuras 8,9, respectivamente). É interessante notar que para o autor, a porcentagem da renda que o cliente possui para pagar o empréstimo após deduzido taxas e outras obrigações é representada em 4 valores numéricos: 1, 2, 3 ou 4; ainda sim, o autor descreve esse atributo como numérico e não categórico. Provavelmente esses quatro números representam a faixa de valores em porcentagem, sendo os clientes com valor 4 os que possuem mais dinheiro após dedução de impostos e obrigações para pagar o empréstimo.

Para os outros atributos há presença de outliers, sendo significativo no valor de empréstimo solicitado (Figura 7), onde se nota um grande intervalo de valor, indo de valores inferiores a 2500 DM até superiores a 17500 DM.

É importante notar que para o atributo que indica o número de pessoas dependentes financeiramente do cliente (figura 12), há dois tipos de clientes: que possuem 1 pessoa dependente ou que possuem 2 pessoas dependentes. Os que possuem 2 pessoas dependentes são identificados como outliers, pois são em quantidade menor de forma significativa.

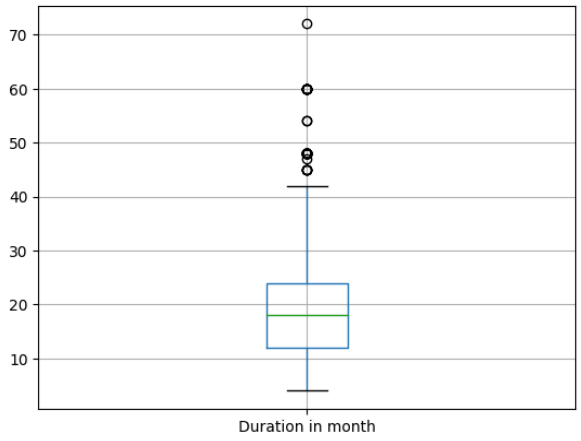

Figura 6 - Análise de outliers do tempo de pagamento.

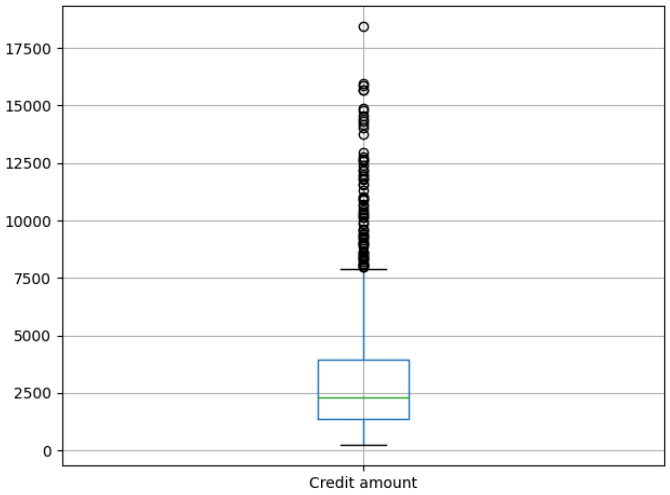

Figura 7 - Análise de outliers de quanto dinheiro está sendo emprestado.

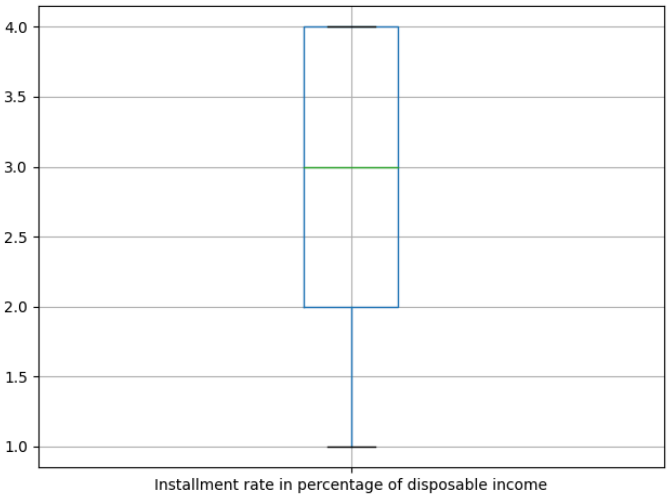

Figura 8 - Análise do quanto de dinheiro o cliente possui para pagar.

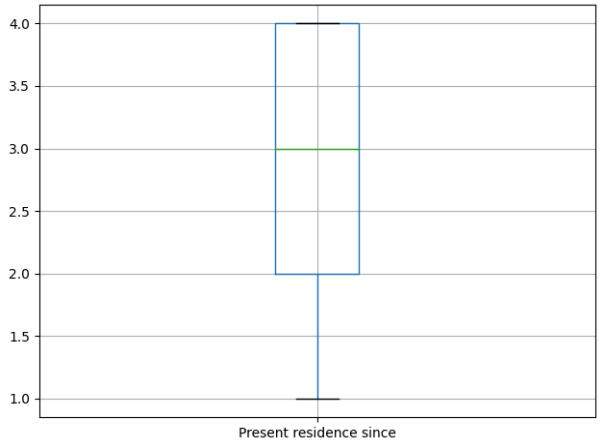

Figura 9 - Análise de outliers de quanto tempo está na residência atual.

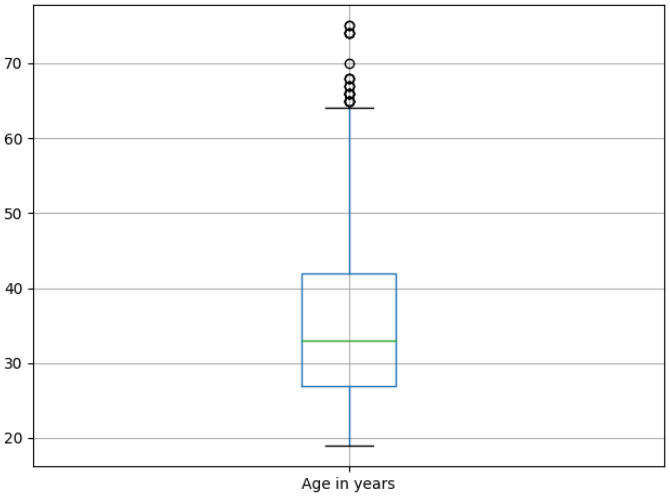

Figura 10 - Análise de outliers da idade do cliente.

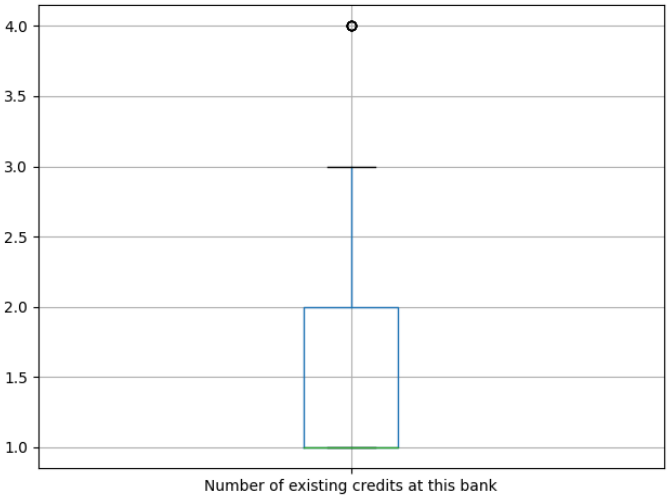

Figura 11 - Análise de outliers da quantidade de modalidades de créditos que o cliente possui.

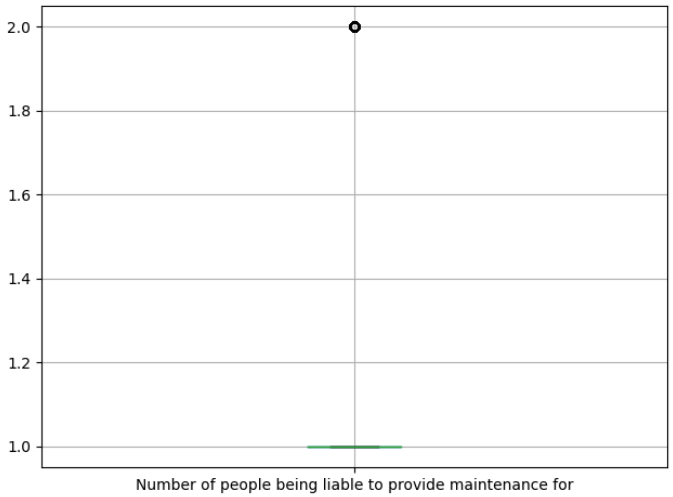

Figura 12 - Análise de outliers de pessoas financeiramente dependentes do cliente.

Agora, partindo para a análise de dados categóricos na figura 13, observa-se que o maior motivo para solicitação de empréstimo é para a compra de rádio ou TV, a maioria está no mesmo emprego de 1 a 4 anos, mais de 80% não possuem um terceiro como garantidor do empréstimo, mais de 70% possuem casa própria e quase todos clientes são trabalhadores estrangeiros.

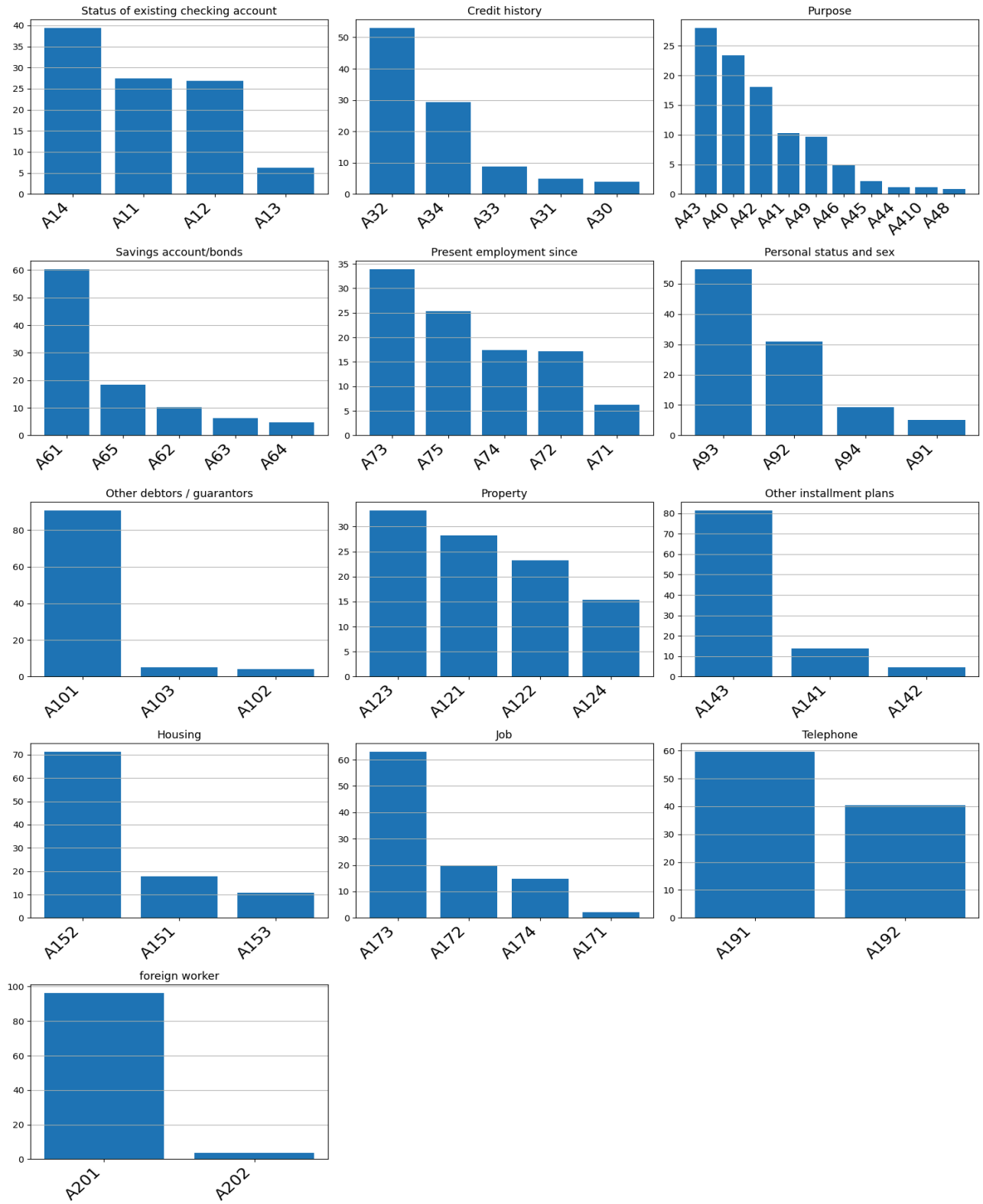

Figura 13 - Proporção para dados categóricos.

Partindo para a análise dos labels, observamos o desequilíbrio na base de dados, 70% dos clientes são bons pagadores (pagaram o empréstimo) e 30% são maus pagadores (não pagaram o empréstimo).

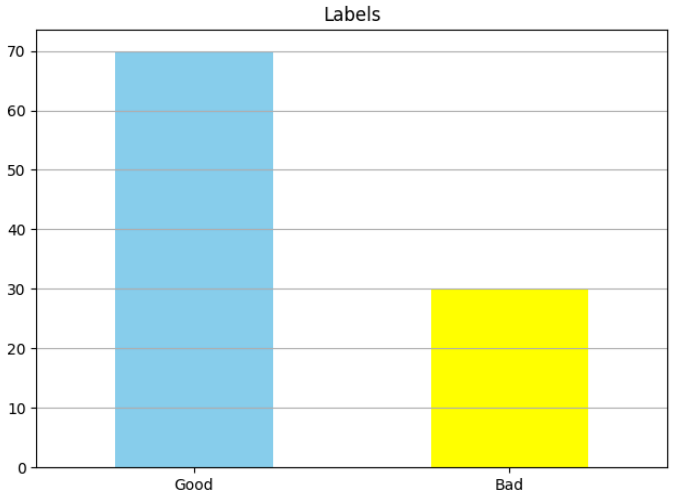

Figura 14 - Proporção das duas classes.

Diante desta informação, o autor da base dados sugere usar uma matriz de custo para tratar esse problema de base de dados desequilibrada, a seguinte matriz de custo é sugerida:

	Predicted: Good (1)	Predicted: Bad (2)
Actual: Good (1)	0	1
Actual: Bad (2)	5	0

Tabela 1 - Matriz de custo.

Essa matriz diz que o custo ao prever como classe 1 um dado que na verdade pertence à classe 2 é 5 vezes mais custoso do que prever classe 2 quando na verdade é classe 1. Esta informação é inserida durante o treinamento para que a predição seja 5 vezes mais penalizado ao classificar um dado como False Positive. Com isto, o modelo a ser desenvolvido trata o desbalanceamento de classe.

No repositório do trabalho, encontra-se o arquivo '*data_analysis*' com a programação completa da análise de dados feita acima.

4. DESENVOLVIMENTO

Na primeira parte do trabalho, um modelo base, resolvido da forma clássica com treinamento supervisionado é programado, as métricas obtidas serão utilizadas na segunda parte do trabalho como comparação de modelos desenvolvidos depois, onde não há presença de true-labels.

A peculiaridade do trabalho se dá no seguinte: os true-labels são usados como guias de configuração dos algoritmos que geram os pseudo-labels. Para isso, os true-labels são usados como um conjunto de validação dos pseudo-labels gerados, o que permite analisar a geração de pseudo-labels sob a mesma ótica de um classificador tradicional, observando o valor de métricas como F1 score, F2, score, recall e etc. Quanto mais próximos de 1 em cada métrica, melhor é a qualidade da geração de pseudo-labels (no apêndice B se encontra uma explicação mais detalhada de cada métrica relevante neste trabalho).

Nota-se: o algoritmo usado (a ser escolhido com base em métricas) para treinamento supervisionado é o mesmo para todos os casos neste trabalho, com objetivo de criar uma condição que permita comparação entre as partes do trabalho.

A utilização do parâmetro random_state [2] garante que o split de dados seja determinístico para todos os experimentos neste trabalho e permite uma melhor comparação de resultados.

Treinamento supervisionado

Em busca de um modelo base de comparação, foi realizado o treinamento supervisionado. Os seguintes 5 algoritmos foram candidatos à solução do problema: regressão logística, floresta aleatória, máquinas de vetores-suporte, xgboost e rede neural. Esses algoritmos são os cujas métricas de desempenho são exibidas na página do repositório da base de dados [1] em outros trabalhos desenvolvidos usando essa base de dados.

A preparação dos dados para ingestão do modelo se deu na seguinte sequência a fim de evitar contaminação entre dados de treinamento e validação:

i) ‘Split’ dos dados entre treinamento e validação.

ii) Escalamento dos dados numéricos usando ‘standard scaler’[3].

iii) Codificação dos dados categóricos usando one-hot encoding.

Há de se observar que se utilizou apenas dois conjuntos (treinamento e validação) para simplificação de posteriores simulações no trabalho, um terceiro conjunto de teste é sugerido, ficando o conjunto de validação apenas para ‘tuning’ de hiperparâmetros.

O escalamento usado obedece a seguinte transformação de dados:

$$z = (x - \mu) / \sigma$$

z: novo valor do feature.

x: valor original do feature.

μ : a média do feature.

σ: o desvio padrão do feature.

Desta forma

A codificação one-hot mapeia um feature categórico para um vetor de dimensão igual ao número de valores que a categoria do dado pode assumir, onde um elemento do vetor correspondente a esse valor é atribuído 1 caso o dado pertença a ela e 0 caso não pertença, com

isso, gera-se um vetor esparso que apresenta um único elemento com valor 1, sendo o resto dos elementos 0 [4]. Essa transformação implica em aumento da dimensão dos dados de entrada para 48.

Nas figuras abaixo, observamos que a forma da distribuição dos dados não se altera ao aplicar o escalamento, conforme esperado.

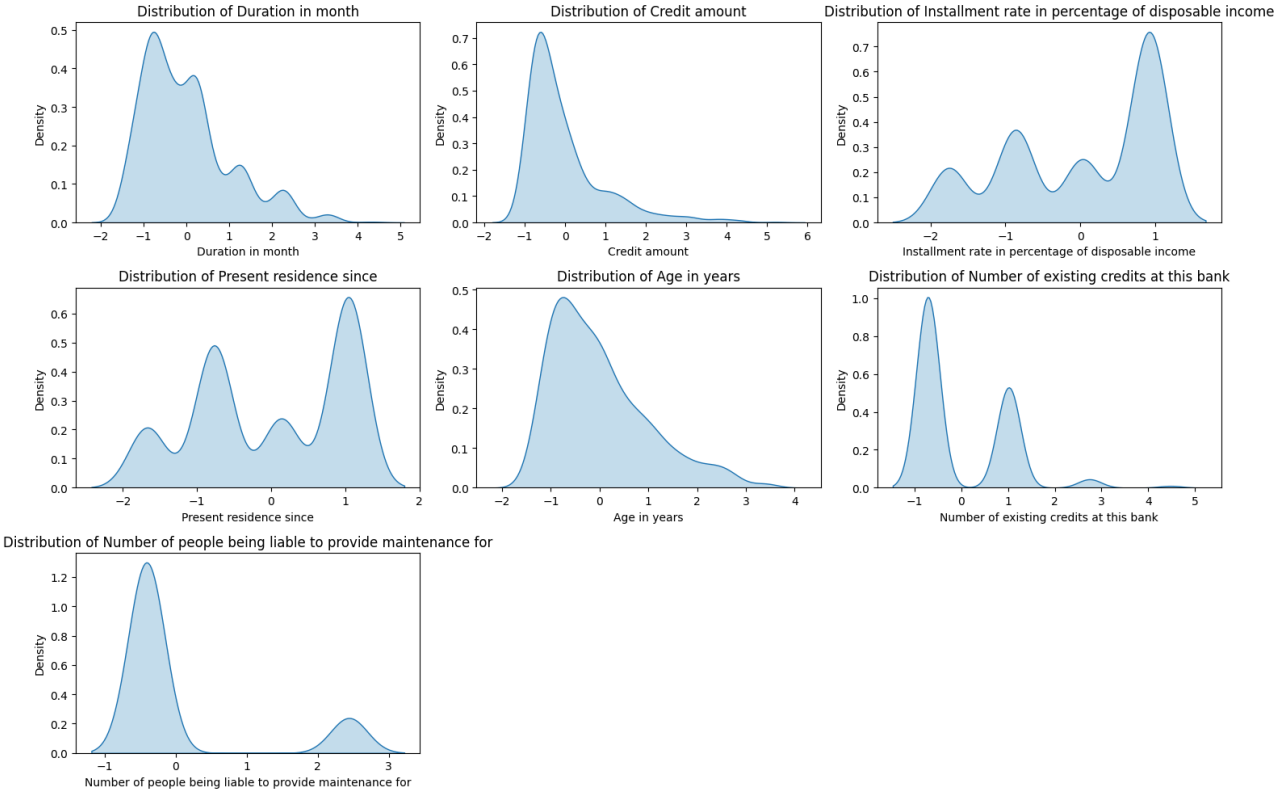

Figura 15 - Distribuição de dados numéricos após escalamento padrão.

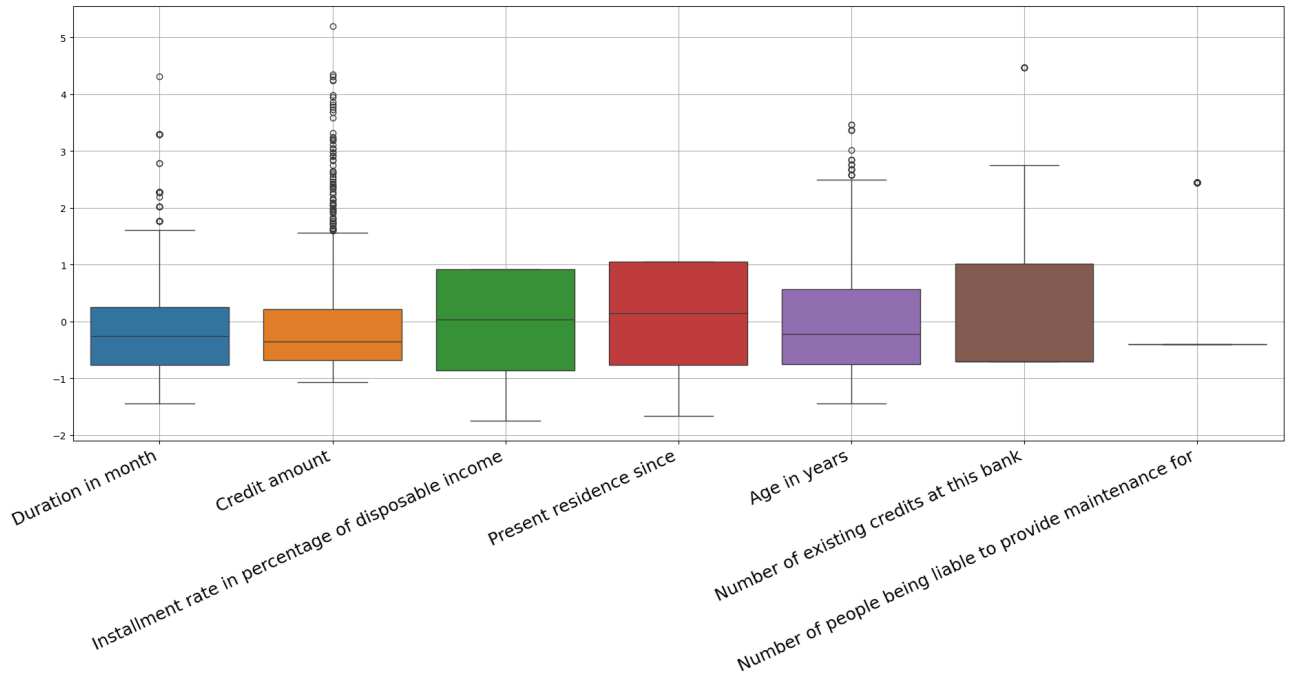

Figura 16 - A grande maioria dos valores após escalamento está concentrado entre -1 e 1.

Para todos os algoritmos se testou dois métodos de busca de melhor hiperparâmetros: um com a função GridSearchCV [5] da biblioteca do scikitlearn e outro com o uso da biblioteca open source Optuna [6].

A bilbioteca Optuna possui maior eficiência para explorar o espaço de hiperparâmetros, mostrando maior rapidez para a convergência da busca de melhores hiperparâmetros, porém, apresentou maior estocasticidade nos resultados, e exigindo maior refinamento em seus argumentos e parâmetros para estabilização dos resultados, o que dificultou estabelecer um modelo que servirá de comparação, por isso, optou-se pelo uso da função GridSearchCv para buscar os melhores hiperparâmetros (é feito uma busca exaustiva combinando uma grade de hiperparâmetros)

Para alguns algoritmos se utilizou o parâmetro 'scoring' da função GridSearchCv customizado a fim de maximizar as métricas para o F2 score da classe 2, visto que há maior preocupação em ter um alto recall sem perder precisão para a classe 2 de maus pagadores.

```
def f2_class_2(y_true, y_pred):

    return fbeta_score(y_true, y_pred, beta=2, pos_label=2,
average='binary')


f2_scorer_class_2 = make_scorer(f2_class_2)

 GridSearchCV(estimator=model, param_grid=param_grid, cv=5,
scoring=f2_scorer_class_2 , verbose=1, n_jobs=-1)
```

A seguir, temos os algoritmos candidatos a resolverem o problema supervisionado seguido da configuração de hiperparâmetros testadas. No repositório do trabalho, encontra-se a programação detalhada de cada notebook.

i) Regressão logística (arquivo '*logistic_regression*' no repositório):

```
param_grid = {
    'C': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5],
    'penalty': ['l1', 'l2', 'elasticnet'],
    'solver': ['liblinear', 'saga', 'lbfgs'],
    'max_iter': [500, 1000, 2000],
    'l1_ratio': [0.25, 0.5, 0.75],
    'tol': [1e-4, 1e-3],
    'warm_start': [True, False],
    'fit_intercept': [True, False]
}
scoring=f2_scorer_class_2
```

ii) Floresta Aleatória (arquivo '*random_forest*' no repositório):

```
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [5, 10, 15],
    'min_samples_split': [5, 10, 15],
    'min_samples_leaf': [2, 4, 6],
    'max_features': ['sqrt', 'log2', 'auto']
}
scoring=f2_scorer_class_2
```

iii) Máquinas de vetores suporte (arquivo '*support_vector_machine*' no repositório):

```
param_grid = {
     'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
    'gamma': ['scale', 'auto', 0.1],
    'degree': [2, 3, 4],
    'coef0': [-1, 0, 1],
```

```
    'shrinking': [True],
    'tol': [1e-3],
}
scoring=f2_scorer_class_2
```

iv) Xgboost (arquivo '*xgboost*' no repositório):

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.3],
    'max_depth': [3, 6, 9],
    'subsample': [0.7, 0.8, 1],
    'scale_pos_weight': [scale_pos_weight],
    'colsample_bytree': [0.7, 0.8, 1],
    'gamma': [0, 0.1, 0.3]
}
scoring=f2_scorer_class_2
```

v) Multilayer perceptron (arquivo '*neural_network*' no repositório):

```
param_grid = {
    'model__optimizer': ['adam', 'rmsprop'],
    'model__learn_rate': [0.001, 0.01],
    'model__dense_units_1': [32],
    'model__dense_units_2': [16],
    'model__dense_units_3': [8],
    'model__dropout_rate': [0.3]
}
scoring=f2_scorer_class_2
```

Para a rede neural (multilayer perceptron) as 3 camadas intermediárias possuem função de ativação Relu e a camada de saída possui função de ativação sigmoide.

Estudo de hiperparâmetros em treinamento não-supervisionado para geração de pseudo-labels

Antes de proceder com a geração de pseudo-labels usando treinamento não-supervisionado, foi feito um estudo de como as configurações de alguns algoritmos e do comitê de máquinas influenciam a geração de pseudo-labels.

Para isso, toda base de dados com 1000 amostras foi utilizada a fim de garantir uma maior variabilidade e representatividade dos dados.

Desta vez, foi-se aplicado a técnica de análise de componentes principais nos dados além da codificação one-hot e do escalamento padrão. A redução de dimensionalidade visa facilitar o trabalho dos algoritmos não-supervisionados ao processarem os dados brutos. Com intuito de preservar 75% da variância dos dados originais, 12 componentes principais foram obtidas.

Os algoritmos utilizados no estudo são os mesmos que foram utilizados posteriormente para a geração de pseudo-labels, bem como, o comitê de máquinas possui a mesma configuração de votos simples.

O processo iterativo de configuração dos algoritmos se deu da seguinte forma: ajusta-se o hiperparâmetro, executa o algoritmo, os pseudo-labels gerados são comparados com os true-labels através de métricas usadas em classificação binária, como se estivéssemos comparando labels gerados num treinamento com os labels de validação. Assim, observa-se o efeito do hiperparâmetros nas métricas de avaliação.

Como os dados são multivariados, foi utilizado os seguintes 8 algoritmos de aprendizado não-supervisionado: k-means, isolation forest, agglomerative clustering, dbscan, maps auto-organizável, local outlier factor, one class svm, distância de mahalanobis [7].

No repositório, encontra-se o arquivo '*unsupervised_pseudo_labels*' com a programação desta parte do trabalho.

i) K-means.

No caso do k-means não houve ajuste de hiperparâmetros de modo a maximizar alguma métrica. Apenas foi ajustado o número de centroide (k=2) e o método de inicialização (k-means++). O objetivo é que o algoritmo separe as duas classes em 2 clusters.

ii) Isolation forest.

A contaminação foi variada de 0.1 a 0.5 em intervalos de 0.1. A classe 2 é considerada anomalia.

iii) Agglomerative clustering

Variou-se os hiperparâmetros metric e linkage. Estressando todas as combinações possíveis.

iv) Dbscan.

Ajustou-se o hiperparâmetro eps (épsilon) que define o raio de vizinho em torno de um dado..

v) Mapas auto-organizáveis.

Ajustou-se o hiperparâmetro que determina o limiar de distância entre neurônios que serão tidos como anomalias.

vi) Local outlier factor.

Ajustou-se a contaminação. Anomalias são tratadas como pertencentes à classe 2.

vii) One class svm

Ajustou-se o nu, que controla o número de outliers que o algoritmo permite identificar.

viii) Distância de Mahalanobis.

Ajustou-se um limiar, onde outliers acima desse limiar são considerados anomalias (pertencente à classe 2)

Em geral, para a solução de criação de pseudo-labels, a classe 2 é tida como anomalia, outliers ou pertencente a um outro cluster que não seja o cluster da classe 1.

Treinamento supervisionado com uso de pseudo-labels gerados por treinamento não-supervisionado

Neste experimento, há split de dados em treinamento e validação, os labels do conjunto de validação serão usados para validar as métricas de treinamento não-supervisionado e supervisionado, portanto, 800 amostras serão usadas para treinamento supervisionado e não-supervisionado, que são os mesmos dados de todos os splits feitos na base de dados, o que reduz a estocasticidade do trabalho.

No conjunto de treinamento é realizada a análise de componentes principais a fim de diminuir a dimensão dos dados e potencializar os resultados com os algoritmos não-supervisionados.

A fim de manter 75% da variância dos dados originais, as primeiras 12 componentes principais são utilizadas. Com isso, mantém-se um equilíbrio entre redução de dimensão e informação retida.

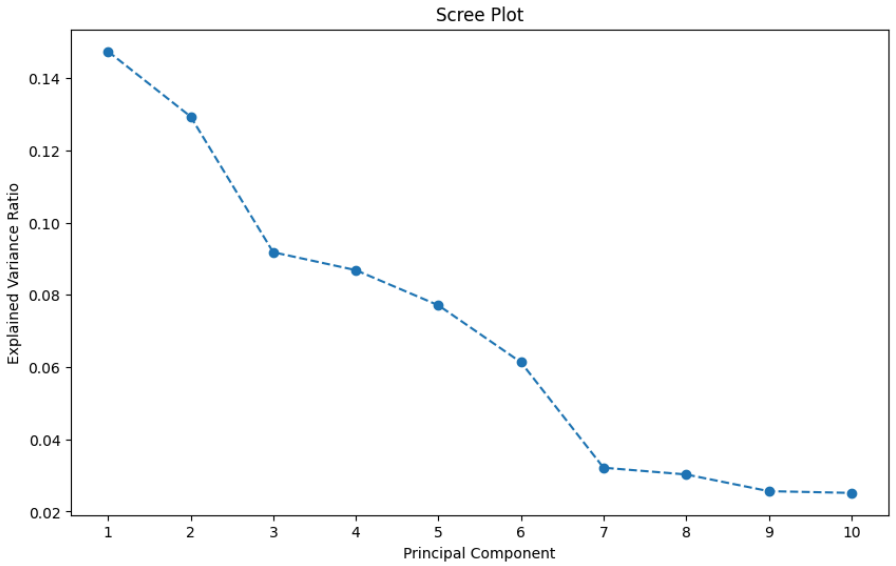

Figura 17 - Explicação de variância para cada componente principal.

Ao observarmos o 'scree plot' na figura acima, é possível notar como a variância dos dados fica bem distribuída entre várias componentes principais, ou seja, é preciso perder muita informação para reduzir o número de componentes principais.

Para visualizarmos a distribuição dos dados num espaço de dimensão ainda mais reduzida, foi feita a análise de componentes principais de forma a manter apenas 3 componentes, seguindo da aplicação do k-means para separação de classes, conforme figura abaixo.

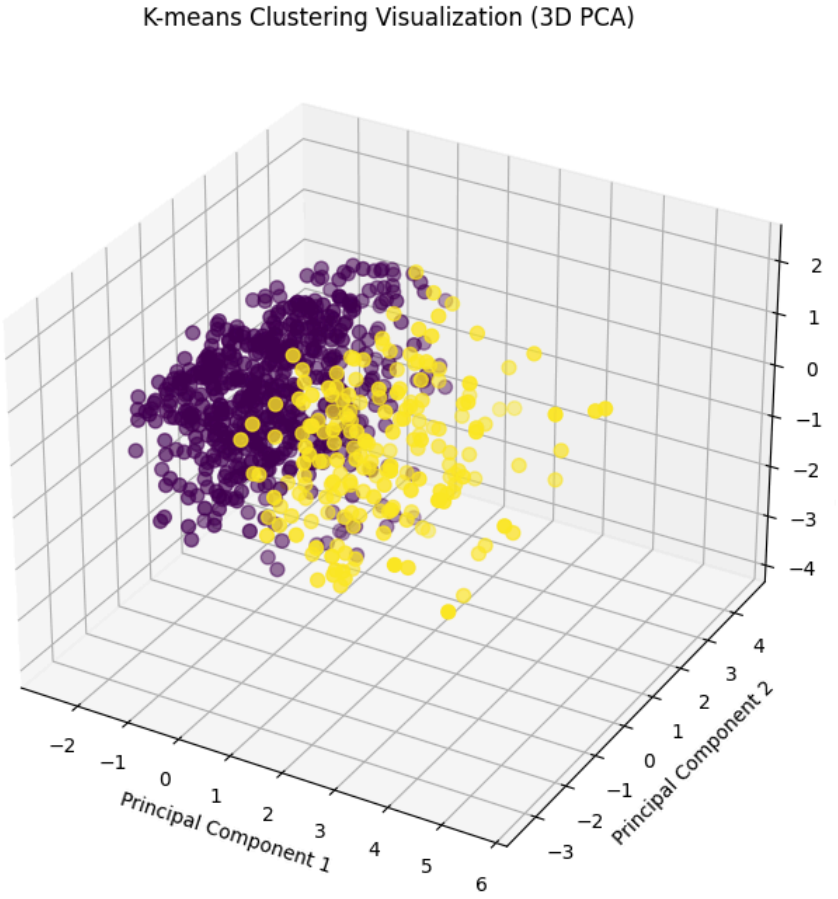

Figura 18 - Visualização da separação dos dados em espaço de dimensão 3.

O ajuste de hiperparâmetros dos algoritmos não-supervisionadas se deu de modo a maximizar o F2 score da classe 2. O processo iterativo seguiu a metodologia: ajusta-se um hiperparâmetro, treina, válida com os true-labels, se o melhor f2 score não foi obtido, reinicia-se o processo.

Os seguintes valores de hiperparâmetros foram obtidos para os 8 algoritmos:

i) K-means.

```
k = 2

kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42,
n_init=10)
```

Produzindo o gráfico que indica a melhor configuração da quantidade de cluster para o k-means, nota-se que o valor de 2, como utilizado, fica fora da região do 'elbow', indicando que essa configuração não é a ideal.

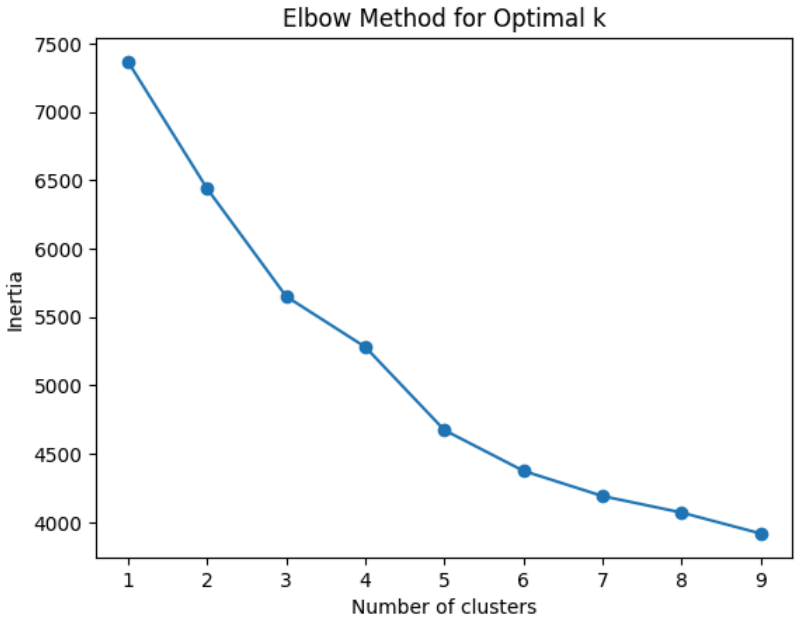

Figura 19 - O gráfico do método 'elbow' indica um valor de 3 para o número de clusters.

ii) Isolation forest.

A contaminação foi variada de 0.1 a 0.5 em intervalos de 0.1. A classe 2 é considerada anomalia.

```
IsolationForest(contamination=0.5, random_state=42)
```

iii) Agglomerative clustering

Variou-se os hiperparâmetros metric e linkage. Estressando todas as combinações possíveis.

```
AgglomerativeClustering(n_clusters=2,
                        metric='cosine',
                        linkage='average',
                         distance_threshold=None)
```

iv) Dbscan.

Ajustou-se o hiperparâmetro eps (épsilon) que define o raio de vizinho em torno de um dado..

```
DBSCAN(eps=2.0, min_samples=24)
```

Na figura abaixo temos uma representação num espaço de dimensão 2 da separação das duas classes.

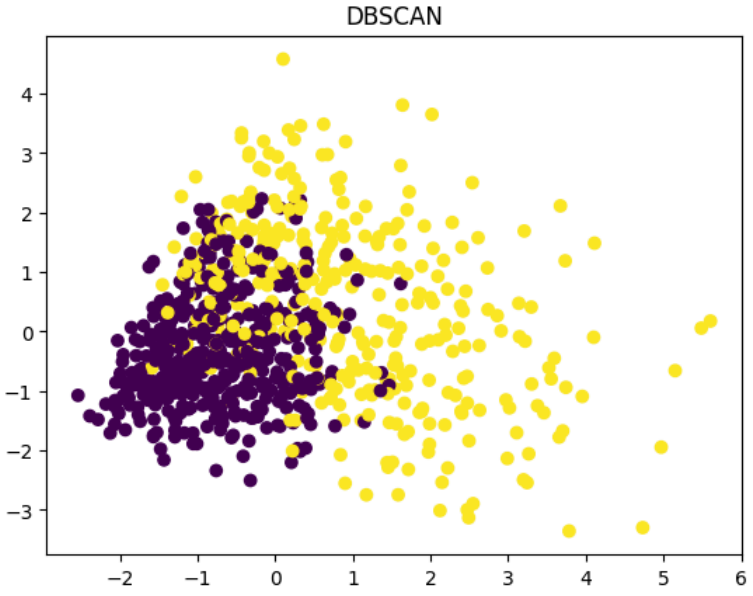

Figura 20 - Separação de classes com épsilon = 2.

v) Mapas auto-organizáveis.

Ajustou-se o hiperparâmetro que determina o limiar de distância entre neurônios que serão tidos como anomalias.

```
threshold = 0.5
```

O mapa de distância abaixo marca neurônios mais distantes na configuração do mapa auto-organizável. Neurônios com distância acima do limiar (threshold) são considerados anomalias (pertencentes à classe 2).

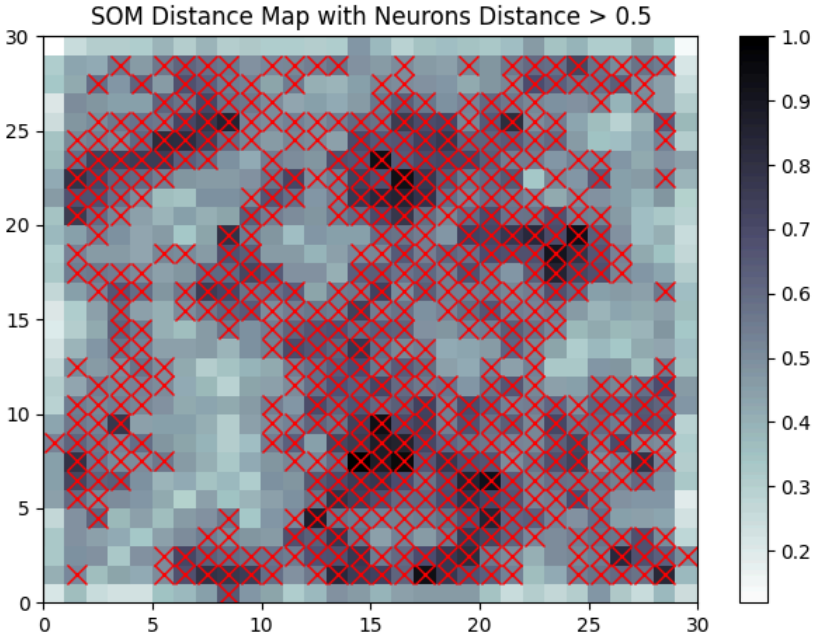

Figura 21 - Mapa de distância com neurônios distantes acima do limiar marcados.

vi) Local outlier factor.

Ajustou-se a contaminação. Anomalias são tratadas como pertencentes à classe 2.

```
LocalOutlierFactor(n_neighbors=20, contamination=0.5)
```

vii) One class svm

Ajustou-se o nu, que controla o número de outliers que o algoritmo permite identificar.

```
OneClassSVM(nu=0.4, kernel="rbf", gamma=0.1)
```

viii) Distância de Mahalanobis.

Ajustou-se um limiar, onde outliers acima desse limiar são considerados anomalias (pertencente à classe 2)

```
threshold = np.percentile(m_dist, 50)
```

A configuração de hiperparâmetros acima visa maximizar o F2-score da classe 2.

Um comitê de máquinas foi utilizado para reduzir a variância dos dados garantindo maior estabilidade dos resultados e diminuição de overfitting, sendo cada um dos 8 algoritmos um membro do comitê.

Esse comitê teve configuração de voto simples, onde o valor 1 significa um voto desse algoritmo para o dado sendo pertencente à classe 2 e o valor 0 para a classe 1, então, procedeu-se para a soma dos votos para cada dado e estabeleceu-se um limiar de votos para que um dado seja dito como pertencente à classe 2. Variou-se esse limiar entre 1 e 8. a coluna 'y_pseudo' representa os pseudo-labels gerados pelo comitê.

	k-means	Agglomerative	Isolation Forest	DBSCAN	SOM_cluster	LOF	OneClassSVM	Mahalanobis	row_sum	y_pseudo
29	1	1	1	1	0	1	1	1	7	2
535	0	0	0	0	0	0	0	0	0	1
695	0	0	1	1	1	1	0	1	5	2
557	1	0	1	1	0	1	0	0	4	2
836	0	0	0	0	1	0	0	0	1	1
596	0	0	0	0	1	1	0	0	2	1
165	0	0	1	0	0	1	0	1	3	1
918	0	0	1	0	0	1	0	1	3	1
495	0	0	0	0	0	0	0	0	0	1
824	0	1	0	1	1	1	0	0	4	2
65	1	1	1	1	1	1	1	1	8	2

Figura 22 - Trecho do dataframe do comitê de máquinas.

Após a geração de pseudo-labels, procedeu-se para o treinamento supervisionado com os pseudo-labels sendo os labels de treinamento. Os labels de validação (true-labels) preservados no 'split' foram usados como validação nesse treinamento a fim de se verificar se há ganhos de métricas a como essa metodologia afeta o desempenho do classificador.

No repositório do trabalho, encontra-se o arquivo '*pseudo_labels*' referente ao treinamento feito acima.

Treinamento semi-supervisionado com uso de pseudo-labels gerados por treinamento não-supervisionado

Para analisar se há ganho de métricas ao usar pseudo-labels gerado pelo classificador, foi testada uma composição de pseudo-labels gerado pela metodologia anterior (não-supervisionado) com pseudo-labels gerado por treinamento supervisionado. Um método que define um treinamento semi-supervisionado, porém, que parte de pseudo-labels em vez de true-labels como tradicionalmente é feito em treinamento semi-supervisionado.

Os seguintes trechos de código são incluídos para fazer essa modificar a composição de pseudo-labels (no repositório do trabalho, o arquivo desta parte é nomeado '*semi_supervised*'):

```
C = 0.9

manual_pseudo = int(len(labels)*C)

y_pseudo = [3]*len(labels)

—-----------------------------------------

X_manual_train = X2.iloc[:manual_pseudo]

y_manual_train = y_pseudo[:manual_pseudo]

X_class = X2.iloc[manual_pseudo:]

—-----------------------------------------

y_pseudo_complement = best_rf_model.predict(X_class)

y_pseudo[manual_pseudo:] = y_pseudo_complement

—-----------------------------------------

labels['y_pseudo'].iloc[manual_pseudo:] =
y_pseudo[manual_pseudo:]#conjunto de validação completo
```

A constante C define a proporção de pseudo-labels gerados por treinamento não-supervisionado e treinamento supervisionado, por exemplo, para C = 0.9 tem-se 90% dos pseudo-labels gerados por treinamento não-supervisionado e 10% por treinamento supervisionado.

Variou-se C de 0.1 a 0.9 com passos de 0.1.

Após, procedeu-se com treinamento supervisionado usando esses pseudo-labels gerados por treinamento semi-supervisionado para verificar o desempenho do classificador.

5. RESULTADOS

Treinamento supervisionado

Segue abaixo os resultados obtidos detalhados para cada algoritmo, onde são apresentados suas métricas de desempenho obtidas, a matriz de confusão, a curva de aprendizado, o f2 score da classe 2 para o conjunto de treinamento e validação, a curva AUC-ROC e a curva precisão-recall (no apêndice C se encontra uma explicação mais detalhada sobre as curvas auc-roc, precisão-recall e curva de aprendizado).

i) Regressão logística (Logistic Regression).

Algoritmo	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
Logistic Regression	1	0.92	0.46	0.61	0.51	0.62	0.58
	2	0.41	0.90	0.56	0.73		

Tabela 2 - Métricas de desempenho para Regressão logística.

Logistic Regression		Actual values	
		Positive (2)	Negative (1)
Predicted values	Positive (2)	53	6
	Negative (1)	76	65

Tabela 3 - Matriz de confusão para Regressão logística.

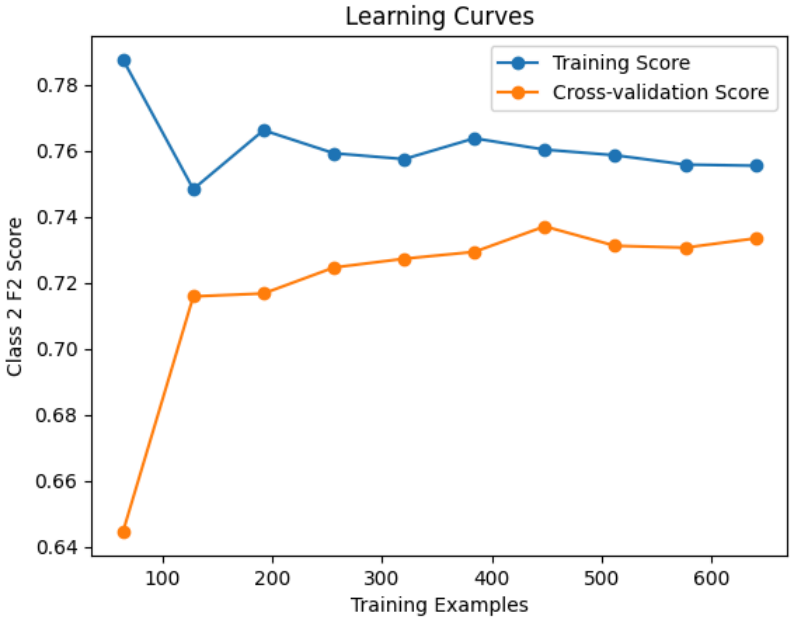

Figura 23 - Tendência de overfitting para Regressão logística.

	F2 score		
Algoritmo	Treinamento	Validação	Diferença
Logistic Regression	0.75	0.73	0.02

Tabela 4 - Análise de overfitting para Regressão logística.

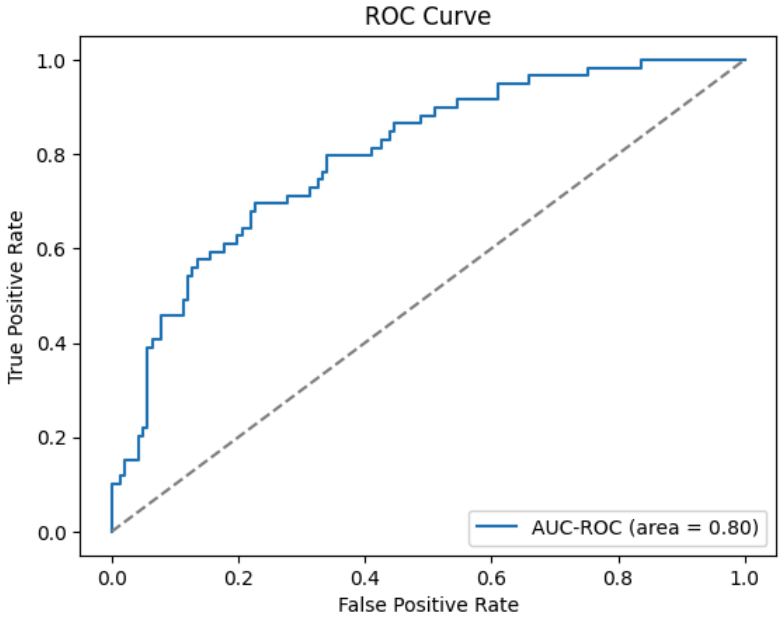

Figura 24 - Área sob a curva para Regressão logística.

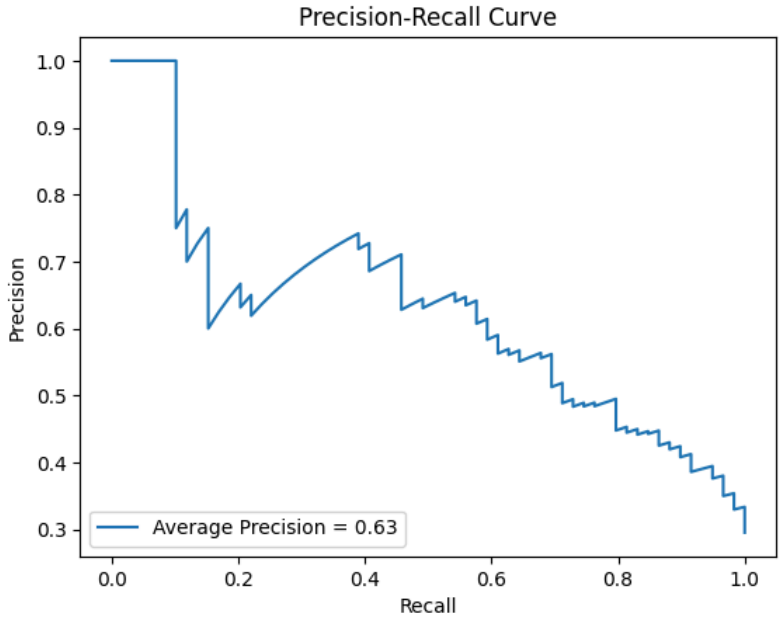

Figura 25 - Compromisso de precisão e recall para classe 2 usando Regressão logística.

ii) Floresta Aleatória (Random Forest).

Algoritmo	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
Random Forest	1	0.94	0.53	0.68	0.58	0.67	0.63

	2	0.45	0.92	0.60	0.76		

Tabela 5 - Métricas de desempenho para Floresta aleatória.

Tabela 6 - Matriz de confusão para Floresta aleatória.

Random Forest		Actual values	
		Positive (2)	Negative (1)
Predicted values	Positive (2)	54	5
	Negative (1)	66	75

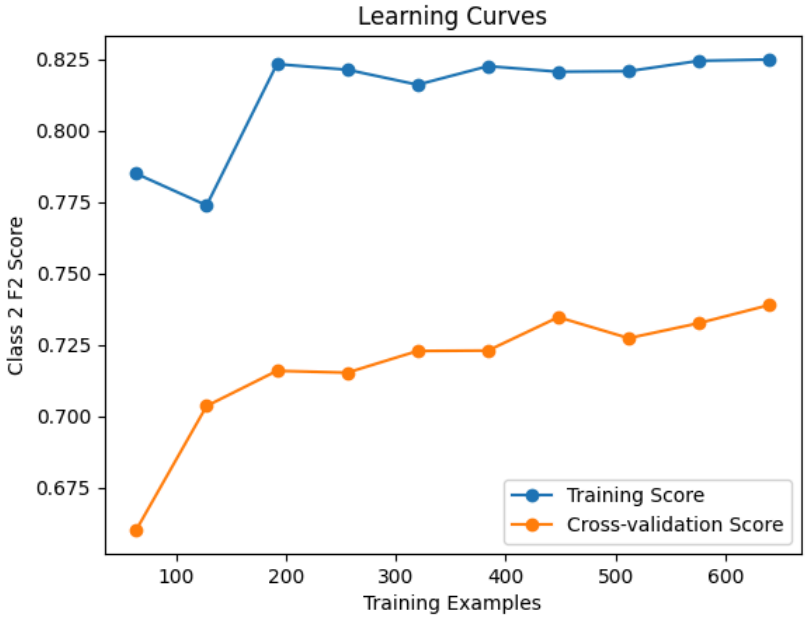

Figura 26 - Tendência de overfitting para Floresta aleatória.

	F2 score		
Algoritmo	Treinamento	Validação	Diferença
Random Forest	0.81	0.76	0.05

Tabela 7 - Análise de overfitting para Floresta aleatória.

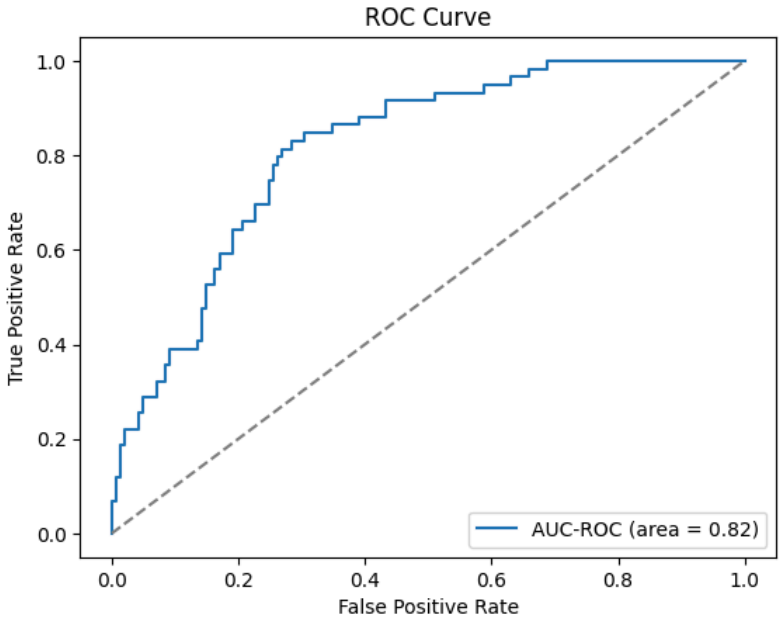

Figura 27 - Área sob a curva para Floresta aleatória.

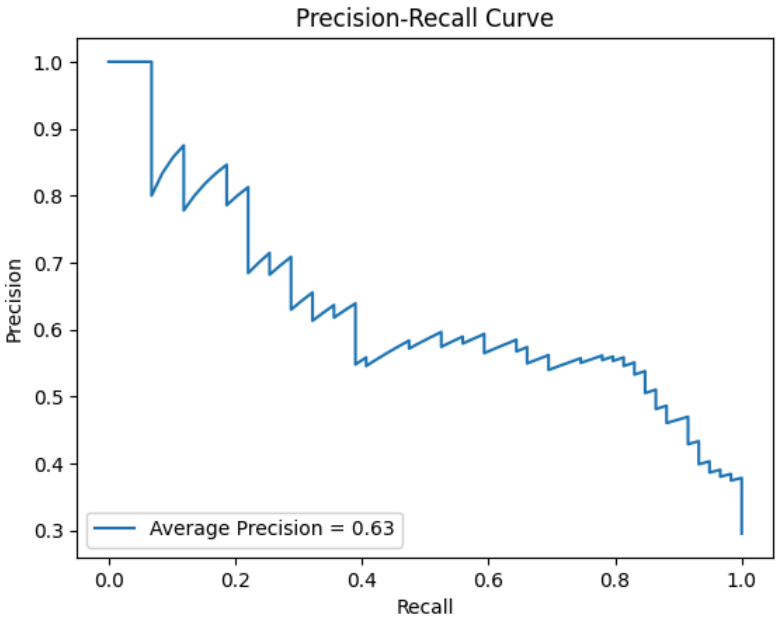

Figura 28 - Compromisso de precisão e recall para a classe 2 usando Floresta aleatória.

iii) Máquinas de vetores-suporte.

Algoritimo	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
Support Vector Machines	1	0.96	0.47	0.63	0.52	0.64	0.59
	2	0.43	0.95	0.59	0.76		

Tabela 8 - Métricas de desempenho para Máquinas de vetores-suporte.

Support Vector Machines		Actual values	
		Positive (2)	Negative (1)
Predicted values	Positive (2)	56	3
	Negative (1)	75	66

Tabela 9 - Matriz de confusão para Máquinas de vetores-suporte.

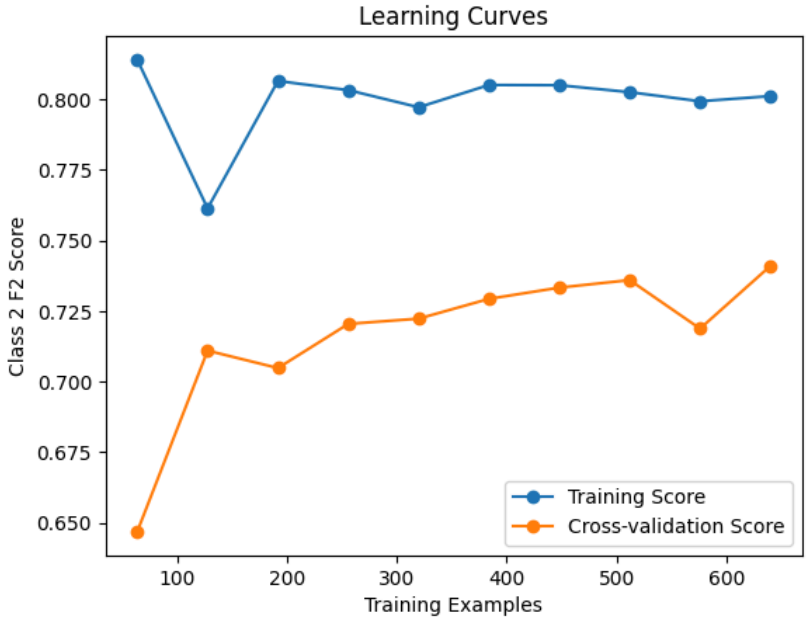

Figura 29 - Tendência de overfitting para Máquinas de vetores-suporte.

	F2 score		
Algoritmo	Treinamento	Validação	Diferença
Support Vector Machines	0.8	0.76	0.04

Tabela 10 - Análise de overfitting para Máquinas de vetores-suporte.

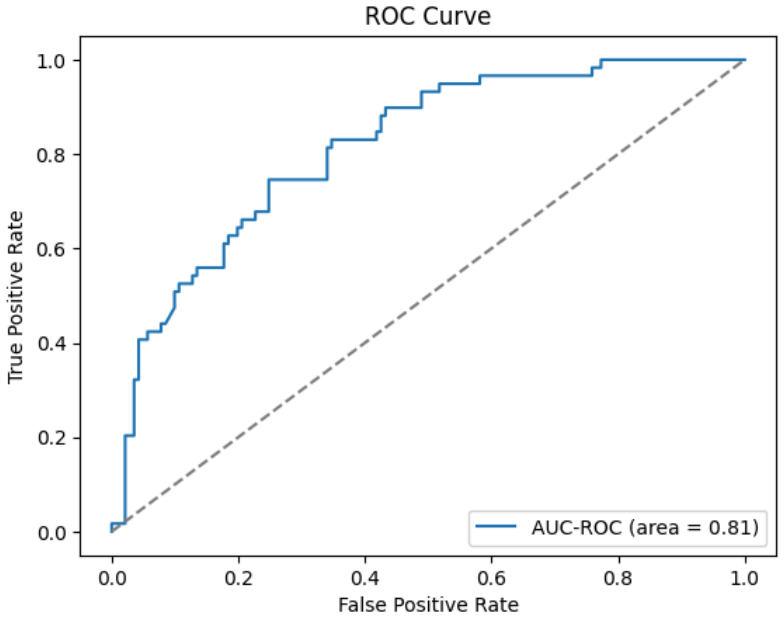

Figura 30 - Área sob a curva para Máquinas de vetores-suporte.

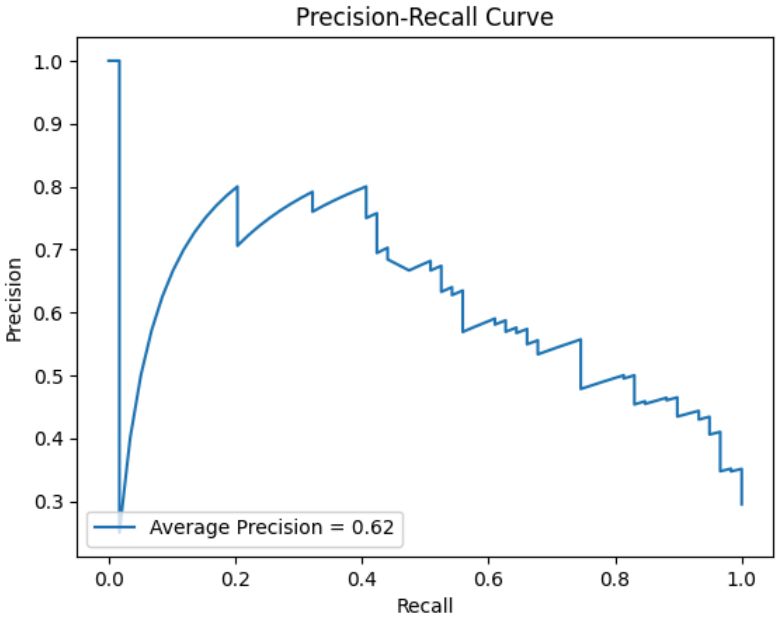

Figura 31 - Compromisso de precisão e recall para classe 2 usando Máquinas de vetores suporte.

iv) Xgboost.

Algoritmo	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
Xgboost	1	0.96	0.33	0.49	0.38	0.56	0.49
	2	0.38	0.97	0.54	0.74		

Tabela 11 - Métricas de desempenho para Xgboost.

Xgboost		Actual values	
		Positive (2)	Negative (1)
Predicted values	Positive (2)	57	2
	Negative (1)	94	47

Tabela 12 - Matriz de confusão para Xgboost.

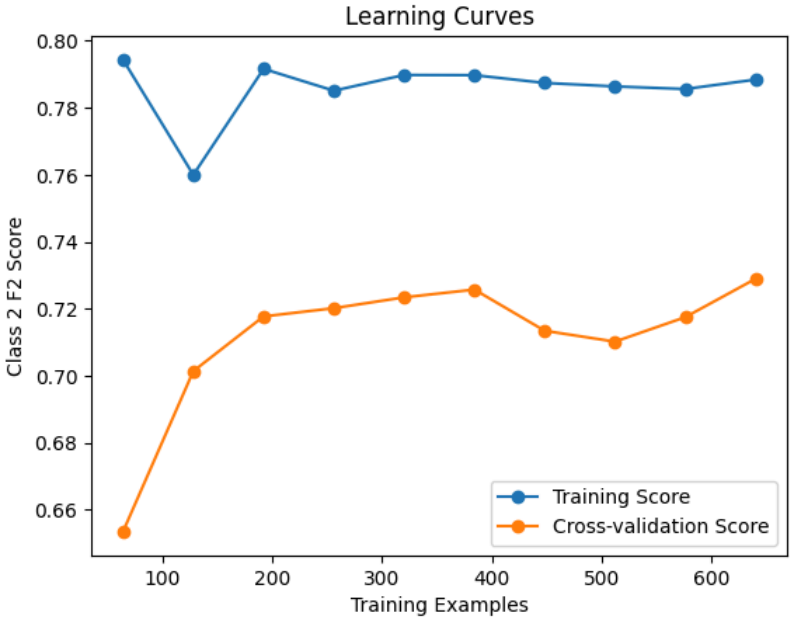

Figura 32 - Tendência de overfitting para Xgboost.

	F2 score		
Algoritmo	Treinamento	Validação	Diferença
Xgboost	0.78	0.74	0.04

Tabela 13 - Análise de overfitting para Xgboost.

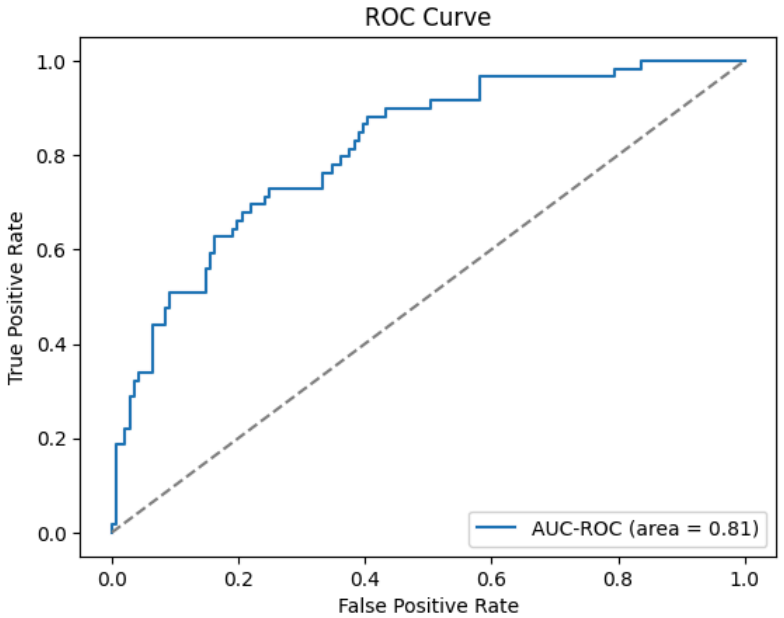

Figura 33 - Área sob a curva para Xgboost.

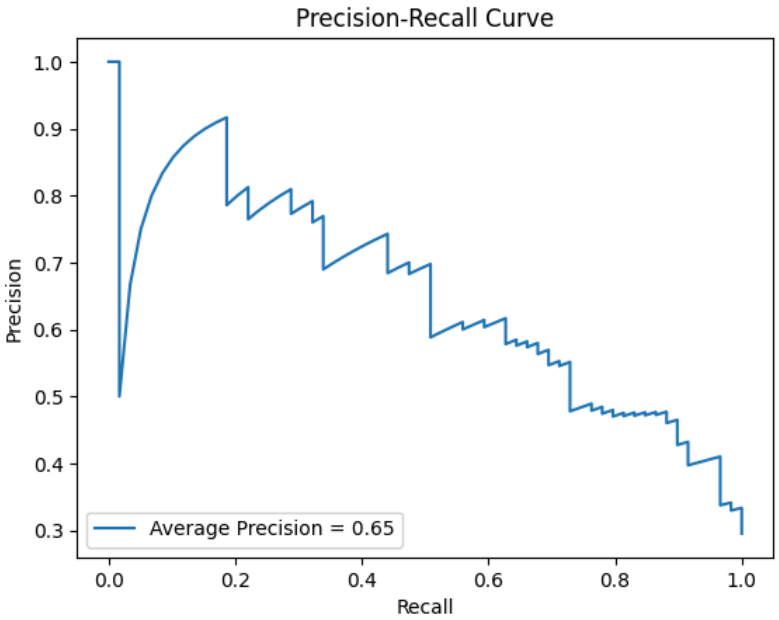

Figura 34 - Compromisso de precisão e recall para classe 2 usando Xgboost.

v) Multilayer Perceptron.

Algoritmo	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
Multilayer Perceptron	1	0.85	0.58	0.69	0.62	0.63	0.63
	2	0.43	0.75	0.54	0.65		

Tabela 14 - Métricas de desempenho para Multilayer Perceptron.

Multilayer Perceptron		Actual values	
		Positive (2)	Negative (1)
Predicted values	Positive (2)	44	15
	Negative (1)	59	82

Tabela 15 - Matriz de confusão para Multilayer Perceptron.

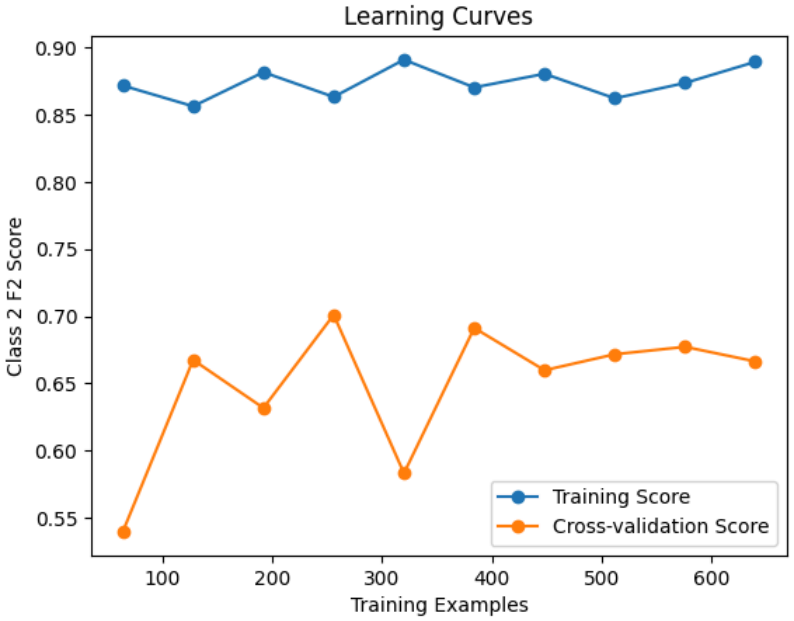

Figura 35 - Tendência de overfitting para Multilayer Perceptron.

	F2 score		
Algoritmo	Treinamento	Validação	Diferença
Multilayer Perceptron	0.85	0.65	0.2

Tabela 16 - Análise de overfitting para Multilayer Perceptron.

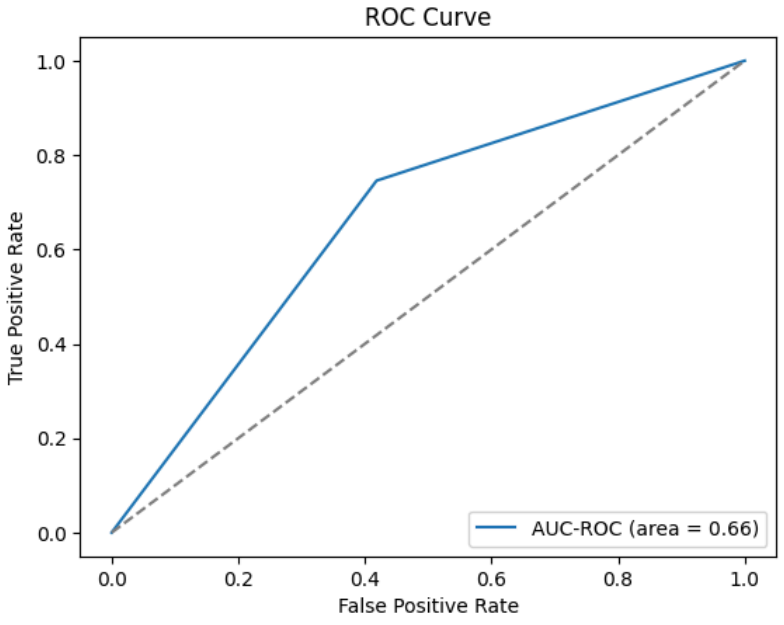

Figura 36 - Área sob a curva para Multilayer Perceptron.

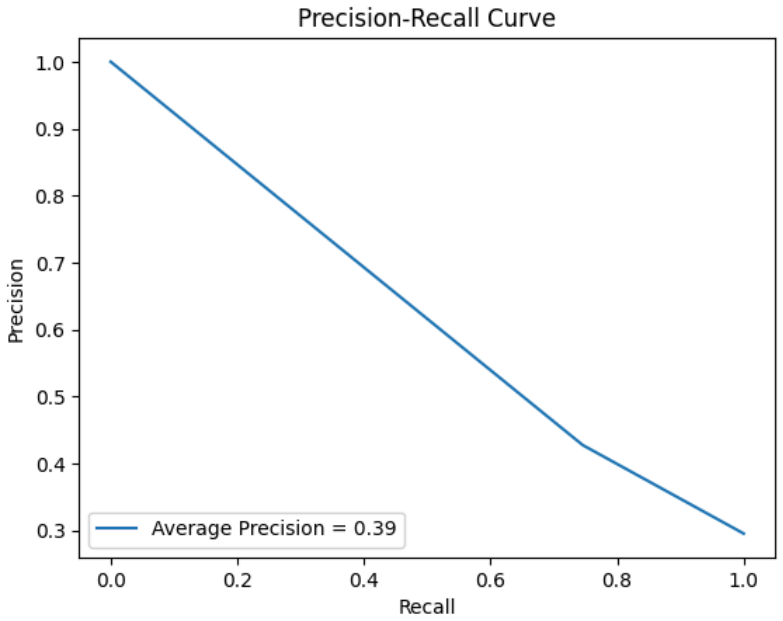

Figura 37 - Compromisso de precisão e recall para classe 2 usando Multilayer Perceptron.

Algoritmo	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
Logistic Regression	1	0.92	0.46	0.61	0.51	0.62	0.58
	2	0.41	0.90	0.56	0.73		
Random Forest	1	0.94	0.53	0.68	0.58	0.67	0.63
	2	0.45	0.92	0.60	0.76		
Support Vector Machines	1	0.96	0.47	0.63	0.52	0.64	0.59
	2	0.43	0.95	0.59	0.76		
Xgboost	1	0.96	0.33	0.49	0.38	0.56	0.49
	2	0.38	0.97	0.54	0.74		
Multilayer Perceptron	1	0.85	0.58	0.69	0.62	0.63	0.63
	2	0.43	0.75	0.54	0.65		

Tabela 17 - Comparativo de métricas de desempenho entre vários algoritmos.

	F2 score (classe 2)		
Algoritmo	Treinamento	Validação	Diferença
Logistic Regression	0.75	0.73	0.02

Random Forest	0.81	0.76	0.05
Support Vector Machines	0.8	0.76	0.04
Xgboost	0.78	0.74	0.04
Multilayer Perceptron	0.85	0.65	0.2

Tabela 18 - Comparativo de métricas de desempenho entre vários algoritmos.

A análise do algoritmo que melhor desempenha é feita de forma analítica, porém seguindo alguns critérios:

i) O recall da classe 2 deve ser o maior possível.

ii) O F2 score ponderado da classe 2 deve ser o maior possível.

iii) Dado as condições acima, o desempenho (precision e recall) na classe 1 não deve degradar muito.

iv) O overfitting está dentro de uma faixa aceitável de aproximadamente 5%.

A análise de overfitting é feita com base na diferença entre o f2 score na classe 2 para o conjunto de treinamento e validação.

Com isso, o algoritmo escolhido é o floresta aleatória. Portanto, este algoritmo será usado em todo trabalho que exija treinamento supervisionado.

É possível observar das tabelas (tabelas 17 e 18) que o floresta aleatória possui ótimo recall para classe 2 (0.92), bom f2 score (0.76) e possui uma quantidade de overfitting de 5%, dentro do aceitável para este trabalho.

Algumas outras observações: o maior recall para a classe 2 é do xgboost (0.97); a maior precisão para a classe 2 é do floresta aleatória (0.45), porém há pouca mudanças entre algoritmos; para todos os algoritmo clássicos de aprendizado de máquina o overfitting permaneceu dentro da faixa aceitável, porém, para a rede neural (multilayer perceptron) há um grande overfitting (0.2), indicando uma necessidade maior de refinamento de hiperparâmetro e tempo de treinamento (limitado pelo escopo do trabalho).

Estudo de hiperparâmetros em treinamento não-supervisionado para geração de pseudo-labels

i)K-means.

Não foi realizado estudo neste algoritmo de clusterização, apenas se configurou o algoritmo a ter 2 centroides e inicialização k-means++.

ii) Isolation Forest.

Contamination	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
0.1	1	0.71	0.91	0.8	0.86	0.51	0.65

	2	0.39	0.13	0.2	0.15		
0.2	1	0.72	0.82	0.77	0.8	0.54	0.64
	2	0.38	0.25	0.3	0.27		
0.3	1	0.72	0.72	0.72	0.72	0.53	0.61
	2	0.35	0.35	0.35	0.35		
0.4	1	0.73	0.62	0.67	0.64	0.53	0.58
	2	0.34	0.45	0.39	0.42		
0.5	1	0.72	0.51	0.6	0.54	0.51	0.52
	2	0.32	0.53	0.4	0.47		

Tabela 19- Influência de hiperparâmetros nas métricas para o algoritmo Isolation Forest.

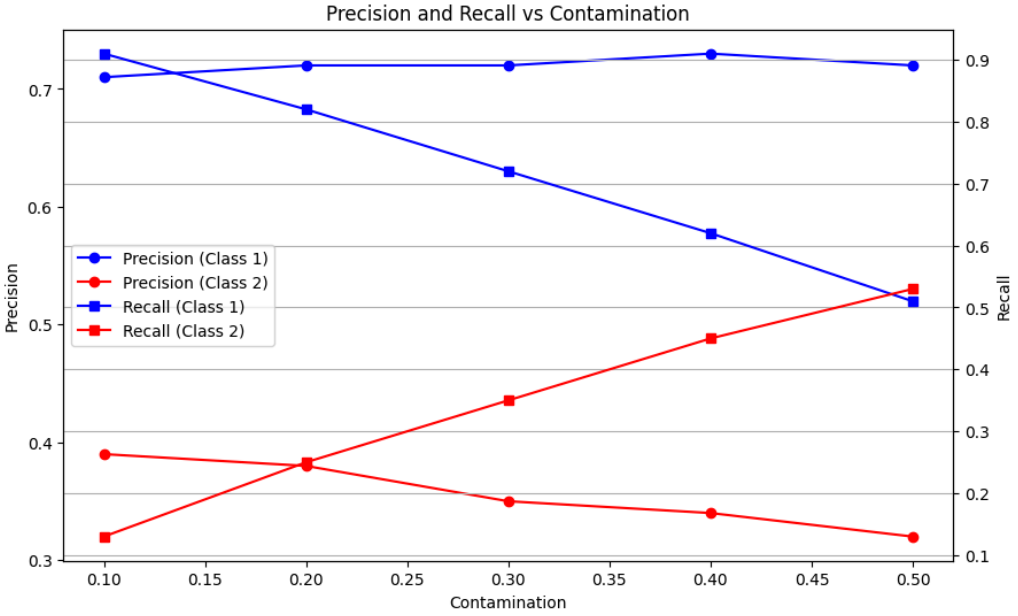

Figura 38 - Influência do hiperparâmetro para métricas de ambas classes usando isolation forest.

Para o isolation forest, observa-se que o ajuste de contaminação (o tanto de anomalias esperadas) tem pouco efeito na precisão de ambas classes, porém, possui influência no recall de ambas classes, apresentando um compromisso entre a classe 1 e classe 2, conforme se aumenta a contaminação o recall da classe 2 aumenta enquanto o da classe 1 diminui.

iii) Agglomerative clustering.

Linkage	Metric	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
ward	euclidean	1	0.71	0.16	0.26	0.18	0.4	0.32
		2	0.3	0.85	0.45	0.62		

average	l1	1	1	0	0	0	0.34	0.21
		2	0.3	1	0.46	0.68		
	l2	1	0.5	0	0	0	0.34	0.21
		2	0.3	1	0.46	0.68		
	manhattan	1	1	0	0	0	0.34	0.21
		2	0.3	1	0.46	0.68		
	cosine	1	0.7	0.16	0.25	0.18	0.4	0.32
		2	0.3	0.85	0.44	0.62		
complete	l1	1	0.79	0.29	0.32	0.23	0.44	0.36
		2	0.32	0.88	0.47	0.65		
	l2	1	0.59	0.03	0.06	0.04	0.35	0.22
		2	0.3	0.95	0.45	0.66		
	manhattan	1	0.79	0.2	0.32	0.23	0.44	0.36
		2	0.32	0.88	0.47	0.65		
	cosine	1	0.7	0.4	0.51	0.43	0.47	0.45
		2	0.3	0.6	0.4	0.5		
single	l1	1	0	0	0	0	0.34	0.2
		2	0.3	1	0.46	0.68		
	l2	1	1	0	0	0	0.34	0.21
		2	0.3	1	0.46	0.68		
	manhattan	1	0	0	0	0	0.34	0.2
		2	0.3	1	0.46	0.68		
	cosine	1	1	0	0	0	0.34	0.21
		2	0.3	1	0.46	0.68		

Tabela 20 - Influência de hiperparâmetros nas métricas para o algoritmo Agglomerative Clustering.

Para o algoritmo agglomerative clustering, observa-se que algumas configurações como linkage = average e metric = l1 apresentam um desempenho perfeito na precisão da classe 1, porém nulo no recall dessa classe. São configurações não recomendadas para a geração de pseudo-labels.

iv) Dbscan.

EPS	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
1.0	1	0	0	0	0	0.34	0.2
	2	0.3	1	0.46	0.68		
1.5	1	0.65	0.06	0.11	0.07	0.36	0.25

	2	0.3	0.92	0.45	0.65		
2.0	1	0.73	0.62	0.67	0.64	0.54	0.58
	2	0.34	0.46	0.39	0.43		
2.5	1	0.71	0.9	0.8	0.86	0.51	0.65
	2	0.4	0.15	0.22	0.17		
3	1	0.7	0.99	0.82	0.92	0.47	0.65
	2	0.55	0.02	0.04	0.02		

Tabela 21- Influência de hiperparâmetros nas métricas para o algoritmo Dbscan.

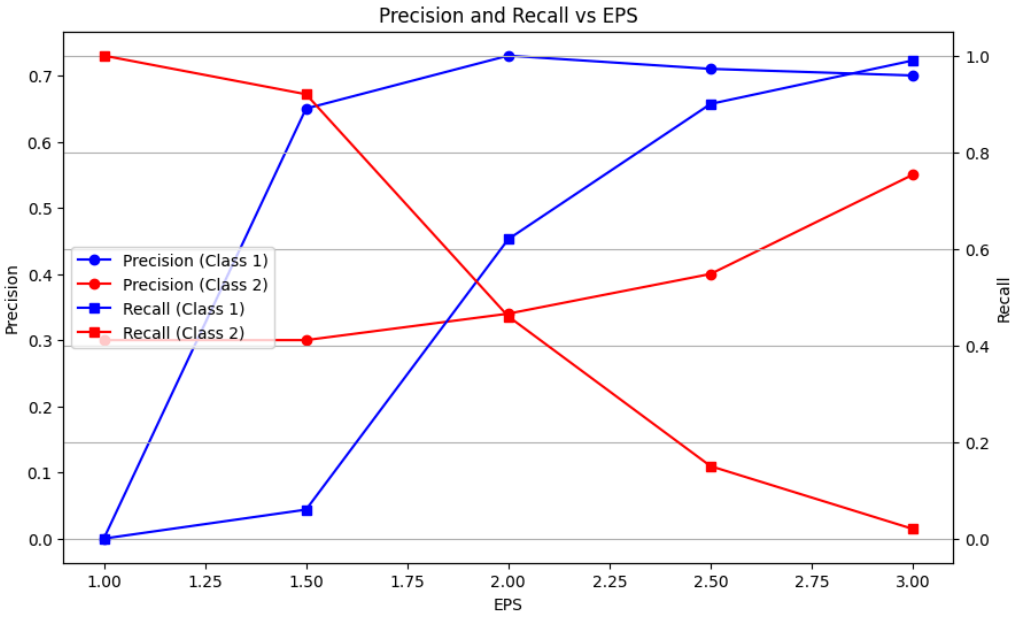

Figura 39 - Influência do hiperparâmetro para métricas de ambas classes usando dbscan.

Para o algoritmo dbscan, ao variar-se o eps, observa-se pouca influência na precisão e grande influência no recall das duas classes, na verdade, para um eps abaixo de o algoritmo separa num único cluster onde todos elementos são da classe 2, para um eps acima de 3 o algoritmo separa num único cluster onde todos os dados são da classe 1.

v) Mapa auto-organizável.

Threshold	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
0.3	1	0.77	0.1	0.18	0.12	0.39	0.28
	2	0.31	0.93	0.46	0.66		
0.4	1	0.61	0.17	0.26	0.19	0.38	0.31

	2	0.28	0.76	0.41	0.56		
0.5	1	0.71	0.54	0.61	0.56	0.51	0.53
	2	0.32	0.5	0.39	0.45		
0.6	1	0.71	0.67	0.69	0.68	0.51	0.58
	2	0.32	0.36	0.34	0.35		
0.7	1	0.72	0.93	0.81	0.88	0.52	0.66
	2	0.45	0.14	0.21	0.16		

Tabela 22 - Influência de hiperparâmetros nas métricas para o algoritmo Mapa auto-organizável.

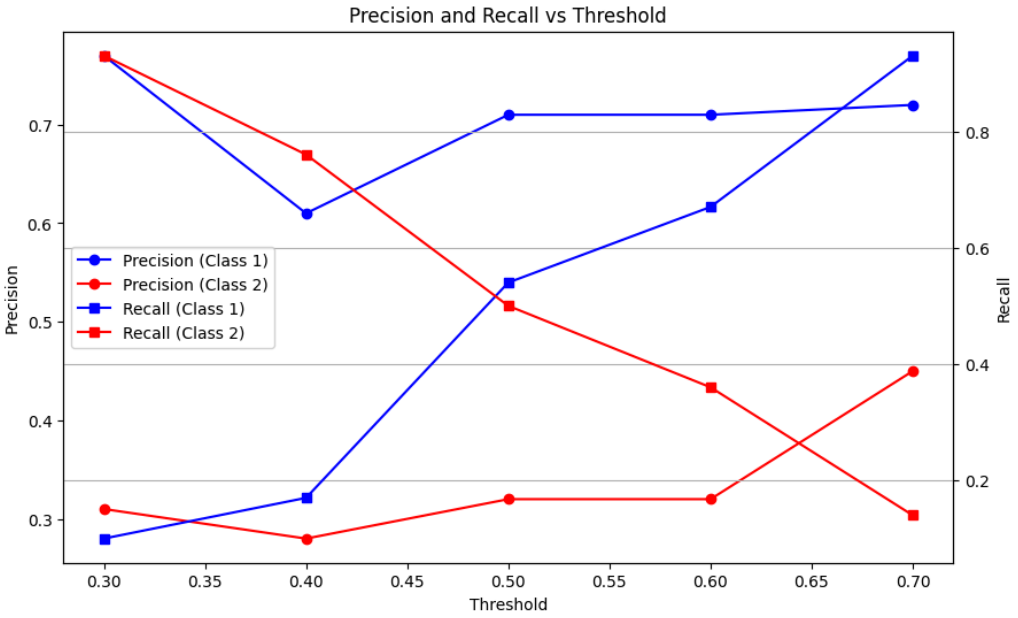

Figura 40 - Influência do hiperparâmetro para métricas de ambas classes usando SOM.

Para o mapa auto-organizável, observa-se mais uma vez o compromisso de recall entre ambas as classes, porém, agora há um maior efeito sobre a precisão da classe 2 conforme se ajusta o limiar da distância entre os neurônios, quanto maior esse limiar, maior a precisão da classe 2.

vi) Local Outlier Factor

Contamination	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
0.1	1	0.71	0.91	0.8	0.86	0.51	0.65
	2	0.39	0.13	0.2	0.15		
0.2	1	0.71	0.81	0.76	0.79	0.52	0.63
	2	0.35	0.23	0.28	0.25		
0.3	1	0.72	0.72	0.72	0.72	0.54	0.61
	2	0.35	0.35	0.35	0.35		

0.4	1	0.74	0.64	0.68	0.65	0.55	0.59
	2	0.36	0.48	0.41	0.45		
0.5	1	0.74	0.53	0.62	0.56	0.53	0.55
	2	0.34	0.57	0.43	0.51		

Tabela 23 - Influência de hiperparâmetros nas métricas para o algoritmo Local Outlier Factor.

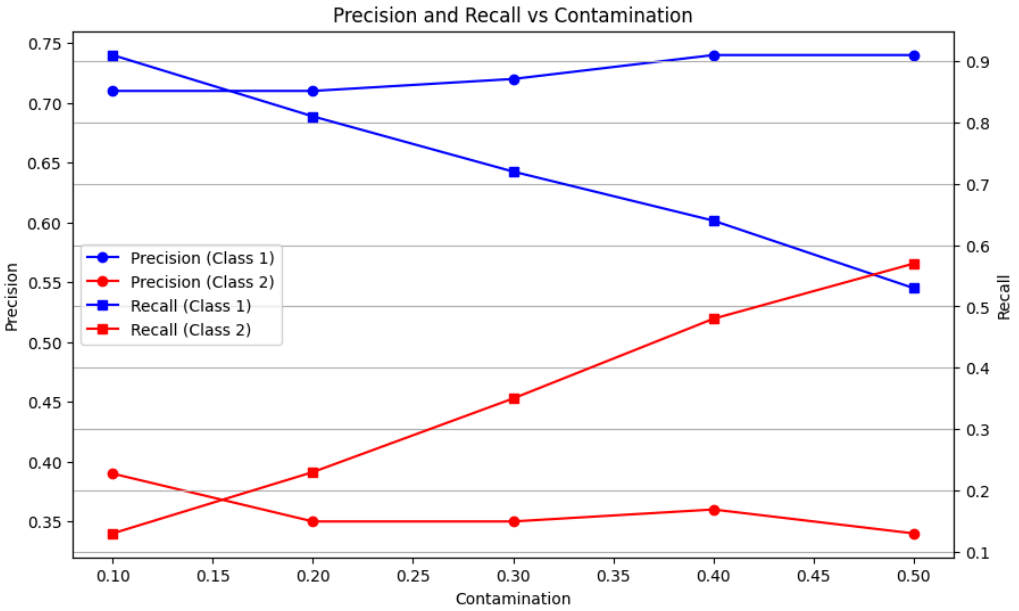

Figura 41 - Influência do hiperparâmetro para métricas de ambas classes usando local outlier factor.

vii) One class svm.

Nu	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
0.1	1	0.71	0.9	0.8	0.86	0.51	0.65
	2	0.39	0.14	0.21	0.16		
0.3	1	0.72	0.72	0.72	0.72	0.53	0.61
	2	0.35	0.34	0.35	0.34		
0.5	1	0.73	0.52	0.61	0.55	0.52	0.53
	2	0.33	0.55	0.41	0.49		
0.7	1	0.7	0.3	0.42	0.34	0.44	0.4
	2	0.3	0.7	0.42	0.55		
0.9	1	0.66	0.1	0.14	0.12	0.37	0.27
	2	0.3	0.89	0.44	0.63		

Tabela 24 - Influência de hiperparâmetros nas métricas para o algoritmo one class svm.

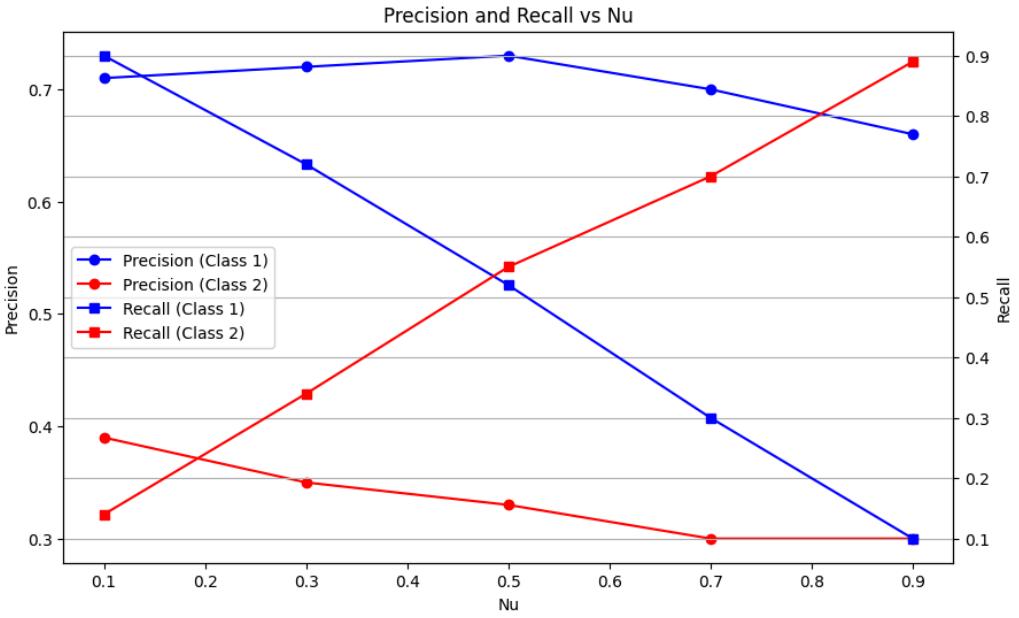

Figura 42 - Influência do hiperparâmetro para métricas de ambas classes usando one class svm.

viii) Distância de Mahalanobis.

Percentile	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
10	1	0.64	0.09	0.16	0.11	0.37	0.27
	2	0.29	0.88	0.44	0.63		
30	1	0.69	0.3	0.42	0.34	0.44	0.4
	2	0.3	0.69	0.42	0.55		
50	1	0.73	0.52	0.61	0.55	0.52	0.53
	2	0.33	0.55	0.41	0.49		
70	1	0.71	0.71	0.71	0.71	0.52	0.6
	2	0.33	0.33	0.33	0.33		
90	1	0.72	0.92	0.81	0.87	0.52	0.66
	2	0.44	0.15	0.22	0.17		

Tabela 25- Influência de hiperparâmetros nas métricas para o algoritmo Distância de Mahalanobis.

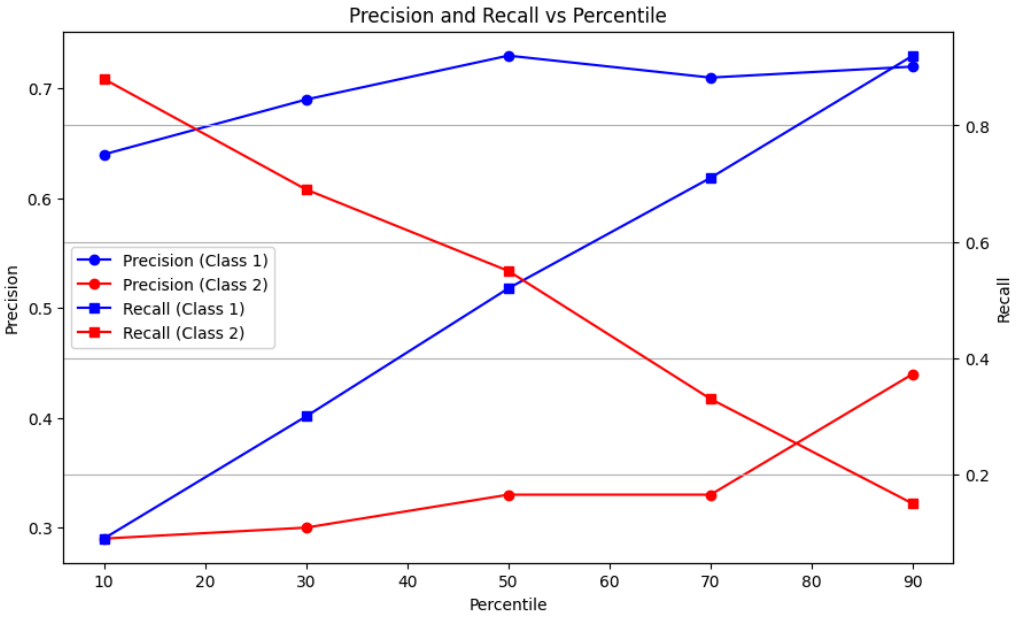

Figura 43 - Influência do hiperparâmetro para métricas de ambas classes usando distância de Mahalanobis.

Para os três algoritmo restantes (Local Outlier factor, one class svm e distância de mahalanobis) há efeito na precisão e recall de ambas classes ao se ajustar hiperparâmetros dos algoritmos, o comportamento observado em outros algoritmos é similar, com a precisão de cada classe variando pouco, porém com grande variação no recall.

Em geral, quanto mais pseudo-labels de uma classe o algoritmo gera (estando correto ou não) maior o recall desta classe, por exemplo, no algoritmo isolation forest vemos que quanto maior a contaminação, maior o recall da classe 2 (dados da classe 2 são tratados como anomalias/contaminação), levando a uma ideia de que se pode gerar labels de uma classe o tanto que se quer ter maior recall nesta classe. Entretanto, essa configuração terá efeito em outro conceito muito importante: veremos a seguir que quando treinamos com pseudo-labels gerados por esses algoritmos não-supervisionados há um grande impacto no overfitting do modelo, o qual está intimamente ligado com a capacidade de gerar pseudo-labels de boa qualidade, isto é, que estão próximos aos labels originais.

Treinamento supervisionado com uso de pseudo-labels gerados por treinamento não-supervisionado

A tabela abaixo (tabela 26) compara a geração de pseudo-labels por treinamento supervisionados com os true-labels, métricas próximas de 1, indicam uma melhor capacidade de gerar pseudo-labels iguais aos true-labels para cada configuração de limiar de votos do comitê de máquinas. Por exemplo, para um limiar (threshold) de 1, todos os dados que tiveram pelo menos um voto do comitê são considerados pertencentes à classe 2, como a classe 1 possui todas métricas nulas, isto indica que todos os dados tiveram pelo menos um voto e só classe 2 é gerada como pseudo-label.

Threshold	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg

1	1	0	0	0	0	0.34	0.2
	2	0.3	1	0.46	0.68		
2	1	0.75	0	0.01	0.01	0.34	0.21
	2	0.3	1	0.46	0.68		
3	1	0.76	0.3	0.43	0.34	0.47	0.42
	2	0.32	0.78	0.45	0.6		
4	1	0.73	0.44	0.55	0.47	0.5	0.49
	2	0.32	0.62	0.42	0.52		
5	1	0.72	0.56	0.63	0.58	0.52	0.54
	2	0.33	0.5	0.39	0.45		
6	1	0.72	0.7	0.71	0.7	0.53	0.6
	2	0.34	0.36	0.35	0.35		
7	1	0.71	0.85	0.77	0.82	0.51	0.64
	2	0.35	0.19	0.25	0.21		
8	1	0.7	0.96	0.81	0.89	0.46	0.63
	2	0.18	0.02	0.04	0.02		

Tabela 26 - Métricas de desempenho para diversos valores de limiares de voto após treinamento não supervisionado para geração de pseudo-labels.

Após o modelo ser treinado com os dados gerados pelo comitê e usando as mesmas condições para a seleção do melhor algoritmo supervisionado, o limiar de valor 4 é escolhido como parâmetro que melhor desempenha, ou seja, para dados que receberam 4 ou mais votos se atribui o pseudo-label de classe 2.

A tabela abaixo apresenta esses resultados.

Threshold	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
1	1	0	0	0	0	0.34	0.2
	2	0.29	1	0.46	0.68		
2	1	0.81	0.18	0.29	0.21	0.43	0.34
	2	0.31	0.9	0.46	0.65		
3	1	0.8	0.25	0.38	0.29	0.46	0.39
	2	0.32	0.85	0.47	0.64		
4	1	0.8	0.3	0.44	0.35	0.49	0.43
	2	0.33	0.81	0.47	0.63		
5	1	0.76	0.5	0.61	0.54	0.54	0.54
	2	0.35	0.63	0.45	0.54		

6	1	0.74	0.65	0.69	0.67	0.55	0.6
	2	0.36	0.46	0.4	0.43		
7	1	0.73	0.89	0.8	0.65	0.54	0.67
	2	0.43	0.2	0.28	0.23		
8	1	0.72	0.94	0.81	0.88	0.52	0.67
	2	0.47	0.14	0.21	0.16		

Tabela 27 - Métricas de desempenho para diversos valores de limiares de voto após treinamento supervisionado com pseudo-labels.

Quando se compara as métricas de desempenho do comitê não-supervisionado com o modelo supervisionado treinado com pseudo-labels gerados pelo comitê, observa-se que: há um ganho de precisão para ambas as classes e há um ganho de recall para a classe 2 (tabela 28).

Threshold	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
1	1	0	0	0	0	0	0
	2	-0.01	0	0	0		
2	1	0.06	0.18	0.28	0.2	0.09	0.13
	2	0.01	-0.1	0	-0.03		
3	1	0.04	-0.05	-0.05	-0.05	-0.01	-0.03
	2	0	0.07	0.02	0.04		
4	1	0.07	-0.14	-0.11	-0.12	-0.01	-0.06
	2	0.01	0.19	0.05	0.11		
5	1	0.04	-0.06	-0.02	-0.04	0.02	0
	2	0.02	0.13	0.06	0.09		
6	1	0.02	-0.05	-0.02	-0.03	0.02	0
	2	0.02	0.1	0.05	0.08		
7	1	0.02	0.04	0.03	-0.17	0.03	0.03
	2	0.08	0.01	0.03	0.02		
8	1	0.02	-0.02	0	-0.01	0.06	0.04
	2	0.29	0.12	0.17	0.14		

Tabela 28 - Ganho de desempenho após treinamento supervisionado com pseudo-labels.

Procedendo a um ponto importante do trabalho: comparar o modelo supervisionado que utilizou true-labels com o modelo supervisionado que utilizou pseudo-labels gerados por treinamento não supervisionado, abaixo, a tabela que compara a diferença entre o primeiro caso e o segundo.

Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
1	-0.14	-0.23	-0.24	-0.23	-0.18	-0.20
2	-0.12	-0.11	-0.13	-0.13		

Tabela 29 - Comparação treinamento supervisionado com true-labels e pseudo-labels.

Como esperado, observa-se que as métricas de desempenho tiveram significativa piora, pois existe o erro associado ao criar pseudo-labels. Esses dados contribuem com resultado quantitativo do que era esperado.

Adiante, quando partimos para uma análise de overfitting na tabela 30, observamos que: há um grande overfitting para todos os limiares, isso é explicado pela propagação de ruídos na geração de dados (geração ruim de pseudo-labels) , o que ficará explícito na última parte do projeto e para todos os limiares, há um grande overfitting que aumenta à medida que o limiar aumenta.

Esse segundo resultado pode ser explicado por alguns fatos: ao se aumentar o limiar, se rotula como classe 2 apenas dados que há grande concordância que são pertencentes a classe 2 pelos algoritmos, isso pode levar o modelo a identificar padrões que são específicos aos dados de treinamento; ao se aumentar o limiar, diminui-se a capacidade do modelo detectar padrões sutis dos dados, o comitê fica menos sensível à variabilidade dos dados; como todos algoritmos foram ajustados de modo a aumentar o recall da classe 2, esse bias está sendo reforçado pelo comitê.

	F2 score		
Threshold	Treinamento	Validação	Diferença
1	0.97	0.68	0.29
2	0.97	0.65	0.32
3	0.96	0.64	0.32
4	0.94	0.63	0.31
5	0.96	0.54	0.42
6	0.93	0.43	0.5
7	0.94	0.23	0.71
8	0.93	0.16	0.77

Tabela 30 - Análise de overfitting para o comitê de máquinas.

Portanto, essa metodologia se mostra ineficiente para criar um modelo sem overfitting, apesar de indicar em média um ganho de métricas de desempenho quando se compara pseudo-labels, principalmente em precisão de ambas classes e recall da classe 2.

Treinamento semi-supervisionado com uso de pseudo-labels gerados por treinamento não-supervisionado

Para o treinamento semi-supervisionado, partiu-se de poucas amostras de pseudo-labels gerados pelo comitê de máquinas e então o modelo supervisionado foi treinado com esses pseudo-labels a fim de completar a rotulação de pseudo-labels.

Os resultados obtidos estão reunidos nas tabelas abaixo. Não há grande tendência exibida.

Unsupervised pseudo labels [%]	Classe	Precision	Recall	F1 score	F2 score	F2 Macro Avg	F2 Weighted Avg
10	1	0.75	0.23	0.36	0.27	0.44	0.37
	2	0.31	0.81	0.45	0.61		
20	1	1	0.06	0.12	0.08	0.38	0.26
	2	0.31	1	0.47	0.69		
30	1	1	0.05	0.09	0.06	0.37	0.25
	2	0.31	1	0.47	0.69		
40	1	0.86	0.17	0.28	0.2	0.44	0.34
	2	0.32	0.93	0.48	0.67		
50	1	0.85	0.23	0.37	0.27	0.47	0.39
	2	0.33	0.9	0.48	0.67		
60	1	0.8	0.28	0.42	0.33	0.48	0.42
	2	0.33	0.83	0.47	0.63		
70	1	0.82	0.16	0.27	0.19	0.43	0.33
	2	0.31	0.92	0.47	0.66		
80	1	0.8	0.31	0.45	0.36	0.49	0.44
	2	0.33	0.81	0.47	0.63		
90	1	0.82	0.26	0.39	0.3	0.47	0.4
	2	0.33	0.86	0.47	0.65		

Tabela 31 - Métricas de desempenho para diversos valores de composição de pseudo-labels.

Quando se parte para análise de overfitting, observa-se o mesmo comportamento da parte anterior do projeto, há grande overfitting, o que inviabiliza a metodologia aplicada, a explicação deste overfitting é detalhada a seguir, bem como a sugestão de solução deste problema.

	F2 score (classe 2)		
Unsupervised pseudo-labels [%]	Treinamento	Validação	Diferença
10	0.98	0.61	0.37
20	0.97	0.69	0.28

30	0.97	0.69	0.28
40	0.98	0.67	0.31
50	0.97	0.67	0.3
60	0.96	0.63	0.33
70	0.94	0.66	0.28
80	0.97	0.63	0.34
90	0.95	0.65	0.3

Tabela 32 - Análise de overfitting para treinamento semi-supervisionado.

Com a intenção de exibir a causa do problema, foi feito um experimento onde uma parte dos pseudo-labels gerados pelo comitê são substituídos pelos true-labels, a tabela abaixo exibe a variação de proporção, bem como o valor de overfitting para cada proporção.

	F2 score (classe 2)		
True-labels [%]	Treinamento	Validação	Diferença
0	0.94	0.6	0.34
25	0.87	0.65	0.22
50	0.76	0.68	0.08
75	0.73	0.68	0.05
100	0.81	0.76	0.05

Tabela 33 - A propagação de erros ao gerar pseudo-labels amplifica o overfitting.

Observa-se que: conforme a proporção de pseudo-labels aumenta em favor de true-labels, o overfitting diminui drasticamente, reduzindo-se praticamente aos níveis esperado (0.05) já que se utilizou o mesmo algoritmo que o treinamento supervisionado na primeira parte do projeto. Portanto, o overfitting para o comitê de máquinas é explicado por uma geração ruim de pseudo-labels, isto é, há muito erro na geração de pseudo-labels desde os algoritmos individuais até o comitê de máquinas que propaga e potencializa esses erros, pode-se dizer que a metodologia termina por gerar dados muito ruidosos que provocam overfitting num treinamento supervisionado com pseudo-labels gerados por comitê de máquinas de algoritmos não supervisionados.

Este resultado exibido na tabela anterior guia a solução para o problema de overfitting: aumentar a qualidade dos labels gerados de tal forma que pelo menos metade dos pseudo-labels estejam corretamente rotulados (overfitting = 8%). Para isso, há três soluções propostas que melhoraram a qualidade dos pseudo-labels:

i) Aplicação de conhecimento do domínio dos dados: o que quebra a premissa do trabalho, pois, apenas o único conhecimento de domínio utilizado é que a classe 2 de maus pagadores é minoritária.

ii) Refinamento dos pseudo-labels gerados em cada algoritmo não-supervisionado: nem todos agrupamentos/previsões são as mesmas, há algumas que o algoritmo deposita mais confiança, em alguns algoritmos não-supervisionadas é possível acessar o quão o algoritmo confia em uma

previsão e definir um limiar para cada algoritmo onde apenas previsões acima desse limiar são aceitas (refinamento um-a-um).

iii) Melhorar o critério dos votos do comitê: a configuração de voto simples e concordância com a maioria pode implicar em bias e overfitting, um comitê de pesos ponderados pode ser explorado, onde algoritmos que desempenham melhor podem ganhar mais peso em seus votos.

A proposta i inviabiliza a premissa do trabalho, portanto a solução sugerida propõe o uso das propostas ii e iii.

6. CONCLUSÃO

Este trabalho desenvolveu uma metodologia baseada em aprendizado de máquina para a abordagem do mesmo problema de risco de crédito em dois casos: com a presença de labels e sem a presença de labels, de tal modo, que a diferença esperada nesses dois casos foram quantificadas com métricas de desempenho e análise de overfitting.

Criou-se, através de treinamento supervisionado, um modelo base para comparação entre instâncias. Este modelo apresentou boas métricas de desempenho e baixo overfitting, sendo o algoritmo de floresta aleatória o que melhor desempenhou, com excelente valor de recall (0.92) para a classe 2 de maus pagadores, sendo esta uma das métricas prioritárias no mercado de análise de risco de crédito.

Partindo para o que é peculiar a este trabalho, os true-labels da base de dados foram removidos e o problema de análise de crédito é transformado em um problema onde não há histórico de bons e maus pagadores.

Precedendo a metodologia a ser desenvolvida, foi feito um estudo da influência dos hiperparâmetros dos algoritmos não-supervisionados na geração de pseudo labels, para isso, munido dos true-labels, pode-se comparar os pseudo-labels gerados com os true-labels e analisar sob a mesma ótica de um problema de classificação binária (true-labels são uma espécie de conjunto de validação), onde métricas próximas de 1 indicam um melhor desempenho na qualidade de geração desses pseudo-labels. Ao analisarmos os resultados, vimos que dependendo da configuração de hiperparâmetros existe pouca influência na precisão de ambas as classes e grande influência no recall de ambas as classes, num compromisso entre recall da classe 1 e recall da classe 2. Outro resultado é: o limiar de votos do comitê influencia a qualidade das métricas de desempenho do modelo, para esta metodologia de 8 algoritmos, um limiar de 4 (dados com 4 ou mais votos são rotulados classe 2) se mostrou o melhor limiar.

Ainda munido dos true-labels, foi desenvolvido uma metodologia onde algoritmos não supervisionados têm seus hiperparâmetros ajustados com base nos true-labels, isto é, faz-se ajuste de hiperparâmetros e valida os pseudo-labels gerados com os true-labels. Estes ajustes foram feitos para maximizar o recall da classe 2 sem degradar as métricas da classe 1, tal como parecido com o treinamento supervisionado. A seguir, um comitê de máquinas de votos simples definiu o que são pseudo-labels de classe 1 e classe 2. Ao analisar os resultados, observou-se que ao treinar o modelo supervisionado com pseudo-labels que são gerados por treinamento não-supervisionado há um ganho nas métricas de precisão de ambas as classes e recall da classe 2, porém há uma grande presença de overfitting na metodologia, o que inviabiliza sua aplicação de tal forma posta neste trabalho.

A fim de encontrar uma resposta e solução para esse overfitting, na última parte do trabalho é feito um experimento onde parte do conjunto dos pseudo-labels são substituídos por true-labels, e então ao se fazer o treinamento supervisionado vemos que conforme se aumenta a proporção de true-labels no treinamento, o overfitting diminui drasticamente a ponto de ser igual ao treinamento supervisionado feito com o conjunto de true-labels na primeira parte do trabalho, observa-se que para 50% de true-labels o overfitting cai para 8%, ou seja, isto indica que a geração de pseudo-labels não é de boa qualidade, o que implica na geração de overfitting. Logo, para resolver este problema é precisa aumentar a qualidade dos pseudo-labels gerados e para isso é proposto duas soluções que podem ser tratadas em um trabalho posterior: filtrar os pseudo-labels gerados por cada algoritmo não-supervisionado de modo a aceitar apenas agrupamentos ou detecções de alta confiança e modificar a configuração de votos das máquinas de modo a incluir uma ponderação de votos de modo que algoritmos mais confiáveis em suas previsões tenham maior valor no voto.

Portanto, o algoritmo de floresta aleatória foi o melhor para resolver o problema de forma clássica com treinamento supervisionado. A fim de aplicar uma metodologia de pseudo-labels que assume uma única condição de conhecimento do domínio (a classe de maus pagadores é minoritária) é necessário maior filtragem e melhor configuração do comitê de máquinas para diminuir o overfitting do modelo. Para uma empresa que precise lançar um produto no mercado de crédito onde não possua histórico de bons e maus pagadores, recorrer a métodos clássicos de rotulação manual de pseudo-labels utilizando profissionais com conhecimento do domínio dos dados para realizar a engenharia de features pode ser uma alternativa para criar um modelo com aprendizado de máquina que desempenha melhor para dados novos.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. University of California, Irvine, School of Information and Computer Science. Disponível em: http://archive.ics.uci.edu/dataset/144/statlog+german+credit+data. Acesso em: 1 ago. 2024.

[2] Pramoditha, Rukshan. Why do we set a random state in machine learning models? Towards Data Science, 2022. Disponível em: https://towardsdatascience.com/why-do-we-set-a-random-state-in-machine-learning-models-bb2dc68d8431. Acesso em: 20 set. 2024.

[3] Scikit-Learn. sklearn.preprocessing.StandardScaler — scikit-learn 1.5.2 documentation. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Acesso em: 6 ago. 2024.

[4] Brownlee, Jason. Why One-Hot Encode Data in Machine Learning? Machine Learning Mastery, 2020. Disponível em : https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/. Acesso em: 1 ago. 2024.

[5] Scikit-Learn. sklearn.model_selection.GridSearchCV — scikit-learn 1.5.2 documentation. Disponível em:https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Acesso em: 31 out. 2024.

[6] Aggarwal, Tushar. Master the Power of Optuna: A Step-by-Step Guide. Medium, 2023. Disponível em: https://medium.com/data-and-beyond/master-the-power-of-optuna-a-step-by-step-guide-ed43500e9b95. Acesso em: 5 set. 2024

[7] Google AI. Gemini. Google, 2023. Disponível em: https://gemini.google.com/. Acesso em: 15 ago. 2024.

[8] Attux, Romis; Boccato, Levy. IA048 - Aprendizado de máquina. 2024. Slides apresentados na disciplina Aprendizado de máquina, Universidade Estadual de Campinas, Campinas - SP, 2024.

[9] Von Zuben, Fernando. 2024. IA353 - Redes Neurais. Slides apresentados na disciplina de Redes Neurais, Universidade Estadual de Campinas, Campinas - SP, 2024.

[10] Wu, Shin-Ting. IA376 - Tópicos em engenharia de computação VII. 2024. Notas de aula apresentadas na disciplina de Análise visual em ciência de dados, Universidade Estadual de Campinas, Campinas - SP, 2024.

[11] Thulin, Måns. Modern statistics with R: from wrangling and exploring data to inference and predictive modelling. 2. ed. Boca Raton: Chapman and Hall/CRC, 2024. Disponível em: https://modernstatisticswithr.com/.

[12] Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems. 2. ed. Sebastopol: O'Reilly Media, 2019.

[13] SCIkit-learn. Versão 1.5.2. [S.l.]: Scikit-learn developers, 2024. Disponível em: https://scikit-learn.org/stable/.

[14] Pandas. Versão 2.1.1. [S.l.]: Pandas Development Team, 2024. Disponível em: https://pandas.pydata.org/docs/.

[15] Numpy. Versão 1.26.1. [S.l.]: NumPy Developers, 2024. Disponível em: https://numpy.org/doc/stable/.

[16] Tensorflow. Versão 2.14.0. [S.l.]: Google, 2023. Inclui a API Keras para construção e treinamento de modelos de aprendizado profundo. Disponível em: https://www.tensorflow.org/.

[17] MATPLOTLIB. Versão 3.8.0. [S.l.]: Matplotlib Development Team, 2023. Disponível em: https://matplotlib.org/stable/.

APÊNDICE

Apêndice A: Atributos da base de dados

Attribute 1: (qualitative)

Status of existing checking account

A11 : ... < 0 DM

A12 : 0 <= ... < 200 DM

A13 : ... >= 200 DM /

salary assignments for at least 1 year

A14 : no checking account

Indica o quanto a pessoa possui em sua conta-corrente. A moeda é o DM (Deutsche Mark) que era a moeda na Alemanha antes da introdução do euro.

Attribute 2: (numerical)

Duration in month

Indica o tempo que o cliente demorará para pagar o empréstimo.

Attribute 3: (qualitative)

Credit history

A30 : no credits taken/

all credits paid back duly

A31 : all credits at this bank paid back duly

A32 : existing credits paid back duly till now

A33 : delay in paying off in the past

A34 : critical account/

other credits existing (not at this bank)

Indica o histórico de crédito do cliente

Attribute 4: (qualitative)

Purpose

A40 : car (new)

A41 : car (used)

A42 : furniture/equipment

A43 : radio/television

A44 : domestic appliances

A45 : repairs

A46 : education

A47 : (vacation - does not exist?)

A48 : retraining

A49 : business

A410 : others

Indica para qual propósito o dinheiro está sendo emprestado.

.Attribute 5: (numerical)

Credit amount

Indica o tanto de dinheiro que o cliente está emprestando.

Attribute 6: (qualitative)

Savings account/bonds

A61 : ... < 100 DM

A62 : 100 <= ... < 500 DM

A63 : 500 <= ... < 1000 DM

A64 : .. >= 1000 DM

A65 : unknown/ no savings account

Indica o tanto que o cliente possui de dinheiro em poupança.

Attribute 7: (qualitative)

Present employment since

A71 : unemployed

A72 : ... < 1 year

A73 : 1 <= ... < 4 years

A74 : 4 <= ... < 7 years

A75 : .. >= 7 years

Indica a quanto tempo o cliente está empregado no empregador atual.

Attribute 8: (numerical)

Installment rate in percentage of disposable income

Indica o quanto de dinheiro após deduzido impostos e cobranças obrigatórias que o cliente possui para pagar o empréstimo.

Attribute 9: (qualitative)

Personal status and sex

A91 : male : divorced/separated

A92 : female : divorced/separated/married

A93 : male : single

A94 : male : married/widowed

A95 : female : single

Indica o estado matrimonial do cliente.

Attribute 10: (qualitative)

Other debtors / guarantors

A101 : none

A102 : co-applicant

A103 : guarantor

Indica se o cliente possui outra pessoa apoiando a aplicação do empréstimo, como um co-aplicante ou garantidor.

Attribute 11: (numerical)

Present residence since

Indica o número de anos que o cliente vive na mesma casa.

Attribute 12: (qualitative)

Property

A121 : real estate

A122 : if not A121 : building society savings agreement/
life insurance

A123 : if not A121/A122 : car or other, not in attribute 6

A124 : unknown / no property

Indica o tipo de residência que o cliente possui.

Attribute 13: (numerical)

Age in years

Indica a idade do cliente.

Attribute 14: (qualitative)

Other installment plans

A141 : bank

A142 : stores

A143 : none

Indica se o cliente possui outros compromissos financeiros.

Attribute 15: (qualitative)

Housing

A151 : rent

A152 : own

A153 : for free

Indica se o cliente possui casa própria.

Attribute 16: (numerical)

Number of existing credits at this bank

Indica se o cliente possui outros créditos com o banco no qual solicitou o empréstimo.

Attribute 17: (qualitative)

Job

A171 : unemployed/ unskilled - non-resident

A172 : unskilled - resident

A173 : skilled employee / official

A174 : management/ self-employed/

highly qualified employee/ officer

Indica se o cliente é empregado com alta capacitação e se possui residência no país.

Attribute 18: (numerical)

Number of people being liable to provide maintenance for

Indica o número de pessoas que o cliente é responsável financeiramente.

Attribute 19: (qualitative)

Telephone

A191 : none

A192 : yes, registered under the customers name

Indica se o cliente possui um telefone registrado.

Attribute 20: (qualitative)

foreign worker

A201 : yes

A202 : no

Indica se o cliente é um trabalhador estrangeiro.

Apêndice B: Métricas de desempenho para classificador binário

Classe positiva: classe 2 de maus pagadores (classe minoritária)

Classe negativa: classe 1 de bons pagadores (classe majoritária)

True positive (TP): número de instâncias corretamente previstas como pertencente à classe 2.

True negative (TN): número de instâncias corretamente previstas como pertencente à classe 1.

False positive (FP): número de instância incorretamente previstas como pertencente à classe 2.

False negative (FN): número de instância incorretamente previstas como pertencente à classe 1.

Accuracy: proporção de previsões corretas considerando ambas classes.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: proporção de previsões corretas para uma classe

Para classe positiva (classe 2):

$$Precision\ classe\ 2 = \frac{TP}{TP + FP}$$

Para a classe negativa (classe 1):

$$Precision\ classe\ 1 = \frac{TN}{TN + FN}$$

Recall: proporção de instâncias de uma classe que são previstas corretamente.

Para a classe positiva (classe 2):

$$Recall\ classe\ 2 = \frac{TP}{TP + FN}$$

Para a classe negativa (classe 1):

$$Recall\ classe\ 1 = \frac{TN}{TN + FP}$$

F1 score: média harmônica de precisão e recall (calculada para cada classe).

$$F1\ score = \frac{2(precision)recall}{precision + recall}$$

F2 score: média harmônica de precisão e recall que dá ênfase no recall (calculada para cada classe).

$$F2\ score = \frac{5(precision)recall}{4(precision) + recall}$$

F2 macro average: média de f2 score para ambas classes.

$$F2\ macro\ avg = \frac{f2\ classe\ 1 + f2\ classe\ 2}{2}$$

F2 weighted average: média de f2 ponderada pelo número de instâncias de cada classe, considera o desbalanceamento de classe.

$$F2\ weigth\ avg = \frac{f2\ classe\ 1(nº\ instâncias\ da\ classe) + f2\ classe\ 2(nº\ instâncias\ da\ classe\ 2)}{nº\ instâncias\ da\ classe\ 1 + nº\ instâncias\ da\ classe\ 2}$$

Matriz de confusão:

		Actual values	
		Positive (2)	Negative (1)
Predicted values	Positive (2)	TP	FP
	Negative (1)	FN	TN

Tabela B1 - Matriz de confusão.

Apêndice C: Curvas de desempenho para classificador binário

Roc-Auc (Receiver operating curve - Area under curve): a curva ROC indica a capacidade do classificador distinguir entre classes. A curva representa a taxa de verdadeiros positivos (TPR) x taxa de falsos positivos (FPR) para diferentes limiares de decisão. A área sob a curva (AUC) é a área sob a curva ROC, que possui valor entre 0 e 1, sendo que 1 indica um classificador perfeito e 0.5 um classificador aleatório (curva de 45º pontilhada abaixo). Pode ser gerada para ambas as classes.

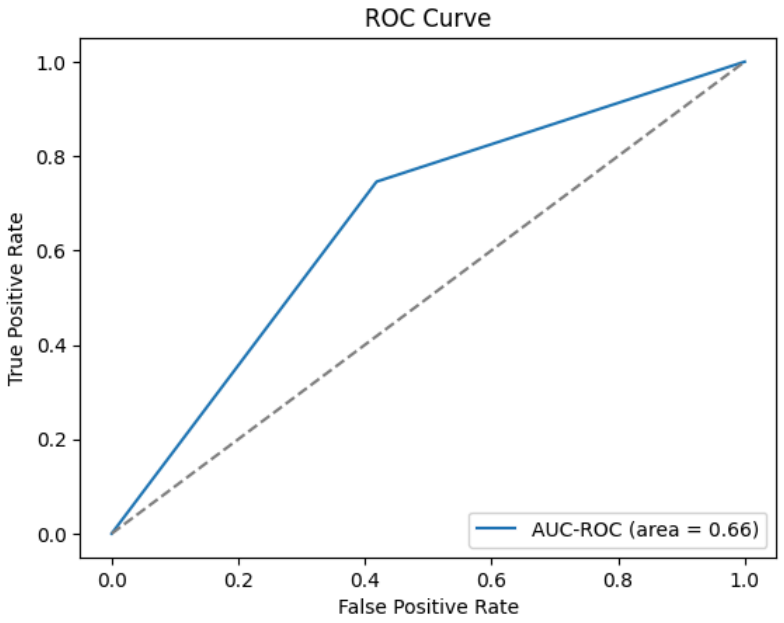

Figura C1 - Exemplo de curva AUC-ROC.

Precision-recall: curva que mostra a relação entre precisão e recall para uma das classes ao se variar o limiar de decisão que define a separação de classes.

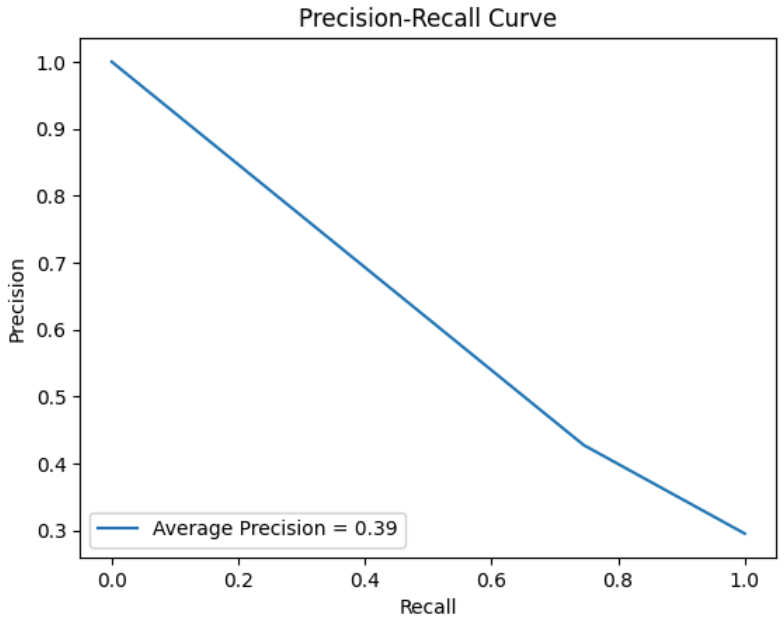

Figura C2 - Exemplo de curva precision-recall.

Learning curve: curva que mostra como a variação da quantidade de dados de treinamento afeta alguma métrica de interesse. Com a curva da métrica sendo gerada para conjunto de treinamento e validação é possível observar uma tendência de overfitting.

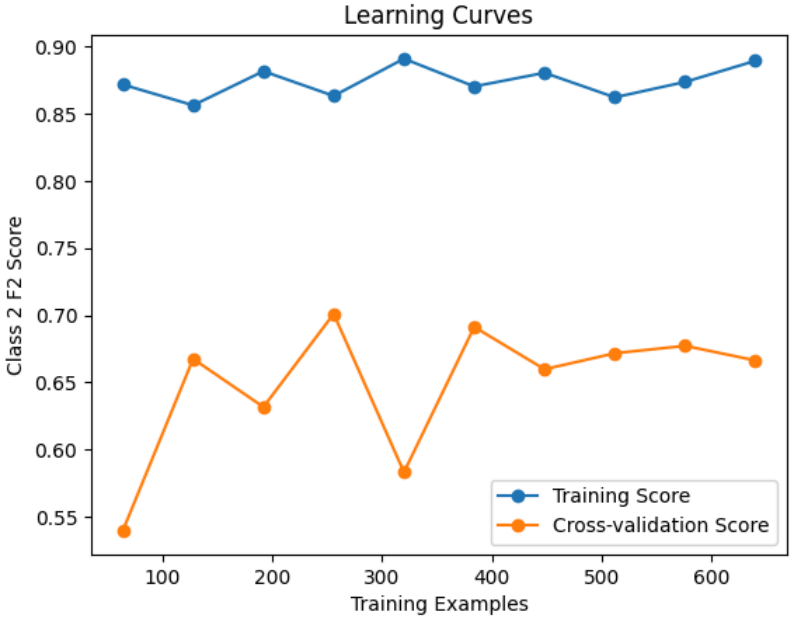

Figura C3 - Exemplo de learning curve (curva de aprendizado).

ANEXO

Link para o repositório contendo todos os notebooks utilizados neste trabalho:

https://github.com/tluancm/Final_year_project-TFC-