Redes Bayesianas aplicadas a predição de *rating* de crédito privado.

Lucas Hattori Costa

5 de julho de 2021

Resumo

Predição de *rating* de crédito privado tem atraído um interesse considerável de pesquisas desde a crise de 2008. Diversas técnicas distintas têm sido aplicadas para este problema, desde abordagens clássicas estatísticas até métodos avançados de inteligência artificial. O presente trabalho se dedica a explorar o uso de redes bayesianas aplicadas a tal contexto, com base em dados históricos de 593 empresas listadas nas bolsas americanas. Primeiramente, são expostos conceitos cruciais para definição do problema, bem como tratados alguns das principais referências bibliográficas na temática. Em seguida, desenvolveu-se um modelo baseado em redes bayesianas que fosse capaz de superar abordagens clássicas de solução. Tal modelo foi capaz de superar o modelo de comparação no quesito de erro médio absoluto (MAE), indicando que pode ser bastante vantajoso em situações onde a incerteza associada ao estado de transição é tão importante quanto o estado predito.

Palavras-chave: risco de crédito, matrizes de transição, redes bayesianas, rating corporativo

1 Introdução

A análise de risco financeiro engloba diferentes tipos de risco, provenientes de diferentes fontes de incerteza. Um deles, o risco de crédito, diz respeito à confiabilidade das empresas honrarem seus pagamentos. Diversas instituições necessitam de uma estimativa precisa a respeito desse risco de crédito, como investidores institucionais, emissores de crédito, órgãos governamentais e até outras empresas.

> "... credit ratings are forward-looking opinions on the relative ability of an entity or obligation to meet financial commitments." Fitch

Dessa forma, é natural que tenham sido desenvolvidos mecanismos capazes de estimar tais riscos para suprir as necessidades do mercado. Para isso, agências de *rating* de crédito foram criadas, datando desde o início do século XX. O objetivo de tais agências é atribuir para cada emissor privado de dívida, ou seja, para cada empresa que esteja envolvida como mutuária no mercado de capitais, uma nota que representa sua capacidade de cumprir seus compromissos financeiros, de forma que outros investidores (institucionais ou não) sejam capazes de avaliar o risco associado a tais empresas de forma mais assertiva.

As três principais agências que realizam tais análises são a Fitch [1], Standard & Poor's [2] e a Moody's [3]. Essas agências tiveram papel central na crise de 2008, de forma que as regulações envolvendo suas análises foram pauta de grandes reformas, por exemplo, o *Dodd-Frank Act*. Seus serviços são altamente buscados no mercado, e exigem uma dedicação de tempo e capital humano, bem como um conhecimento altamente especializado. Por isso, as informações providas por elas nem sempre refletem a situação atual da empresa avaliada, simplesmente por questões temporais.

A avaliação de cada empresa é feita individualmente por cada agência, sendo que, a cada avaliação, existe uma possibilidade de que a empresa avaliada receba um rating melhor ou pior. A incerteza associada ao rating futuro de uma empresa

[1] <https://www.fitchratings.com/>
[2] <https://www.spglobal.com/ratings/pt/>
[3] <https://www.moodys.com/>

pode ser negativa para um investidor institucional que quer efetuar um investimento antes da próxima avaliação de agências. Assim, para esse investidor, é interessante que ele possa ter acesso a um modelo que estime as probabilidades de uma determinada empresa receber cada um dos ratings possíveis, para que este possa fazer o melhor gerenciamento de risco possível de sua carteira. Como exemplo, se um investidor possui um modelo que atribui uma determinada probabilidade da empresa ser avaliada em um rating pior que o atual, ele pode estimar a desvalorização que os produtos de crédito emitidos por tal empresa na data atual terão quando ela for novamente avaliada. Isso se dá pois empresas com rating menor, em sua maioria, pagam taxas de juros (spreads) mais elevados; se uma empresa recebe um *downgrade*, ela é pressionada a elevar seus *spreads*, desvalorizando os títulos emitidos anteriormente com taxas mais baixas.

1.1 Objetivos

Dado, portanto, o contexto de estimar o risco de crédito de uma empresa, tomou-se como situação hipotética, um fundo de investimento em produtos de renda fixa privada que quer avaliar os riscos de produtos a serem potencialmente comprados. Nessa análise, é importante que o investidor obtenha uma estimativa o mais precisa possível a respeito de qual será o rating daquela empresa no futuro, aqui tomado como horizonte de 1 ano. Essa necessidade se dá pois, em uma possível piora de rating, o título comprado hoje se desvalorizará ao longo desse ano, uma vez que seu emissor passará a ser considerado mais arriscado para o restante do mercado. Obviamente isso é indesejado para o fundo de investimento, tanto porque aumenta sua exposição ao risco de crédito, quanto porque desvaloriza sua cota de fundo, afastando potenciais novos clientes. Além disso, o fundo de investimento deseja obter uma probabilidade associada a cada possível estado, de forma a poder estimar uma perda esperada decorrente de cada um desses estados para melhorar balancear sua carteira.

Assim, o objetivo do sistema a ser desenvolvido é ser capaz de estimar qual a probabilidade de uma determinada empresa piorar ou melhorar de rating dentro de um ano, dada sua situação atual.

2 Revisão da literatura

A fim de desenvolver uma base de conhecimento a respeito de ratings de crédito e algoritmos de previsão, obteve-se um arcabouço bibliográfico que servirá de guia para as demais etapas do estudo.

Hadad et al. (2007) sintetiza o conceito de matrizes de transição de rating, isto é, uma representação matricial que relaciona a probabilidade de uma determinada empresa previamente avaliada em um rating X ser avaliada em um rating Y. Os autores ainda expõe a técnica que veio a ser utilizada como *benchmark* ao longo do estudo, que é a matriz de decisão baseada puramente na abordagem frequentista. A Figura 1 ilustra um exemplo de tal matriz, retirado desta fonte. Além disso, os autores desenvolveram uma segunda abordagem com método contínuo de homogeneidade temporal, baseado em estudos prévios de Lando e Skødeberg (2002), obtendo melhores resultados de predição.

Wang e Ku (2021) desenvolve um algoritmo baseado em redes neurais artificiais, mais especificamente, um PANN (parallel artificial neural network), obtendo resultados positivos em três conjuntos de dados reais obtidos. Apesar de tratar de um algoritmo bem distinto do escolhido para o presente estudo, os autores apresentam uma exploração de variáveis e métricas associadas ao problema de estimação de rating de crédito que se mostra valioso para o prosseguimento do estudo. De forma semelhante, Lee (2007) desenvolve um algoritmo baseado em SVM (Support Vector Machines) que, novamente, se distanciam do contexto de redes bayesianas, mas que apresenta uma referência importante no tratamento das variáveis financeiras analisadas posteriormente no estudo.

Sintetizando e aprofundando essa análise a respeito de variáveis de entrada a serem utilizadas, Hajek e Michalak (2013) se mostra como uma das principais referências do presente estudo, ao expor de forma detalhada análises a respeito de diferentes métricas utilizadas como variáveis de entrada.

Por fim, Tavana et al. (2018) representa a principal referência a ser seguida, desenvolvendo um modelo de rede bayesiana para estimar risco de liquidez em instituições bancárias, que possui alta correlação com o rating de crédito associado às mesmas. Os autores comparam tal método com outro baseado em redes neurais, concluindo que, para o problema estudado, ambos algoritmos foram capazes de identificar os fatores críticos de risco envolvidos. Tavana et al. (2018) se mostra de grande valia para o presente estudo pois detalha especificamente a arquitetura de rede bayesiana implementada, bem como discute a respeito de

Figura 1 – Matriz de transição de ratings.

Corporate rating transition matrix based on the cohort method

%, 2001–05

	Number of companies at period end	AAA	AA	A	BBB	BB	B	CCC	D	NR
AAA	1	100	0	0	0	0	0	0	0	0
AA	2	0	50	50	0	0	0	0	0	0
A	21	0	14.29	61.90	4.76	0	0	4.76	14.29	0
BBB	9	0	0	44.44	44.44	0	11.11	0	0	0
BB	3	0	0	0	66.67	0	0	0	0	33.33
B	3	0	33.33	33.33	0	0	33.33	0	0	0
CCC	1	0	0	100	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0
NR	0	0	0	0	0	0	0	0	0	0
Total	**40**									

Fonte: (HADAD et al., 2007)

quais métricas financeiras avaliar.

3 Desenvolvimento

3.1 Metodologia

A metodologia abordada se baseou em primeiramente obter os dados históricos para estudo. Dado isso, foi necessário uma análise exploratória para se obter as características do conjunto de dados obtido, bem como o tratamento do mesmo para prosseguir para a modelagem. Na etapa de modelagem, primeiramente se estabeleceu um modelo base, que serviu como comparação para outros modelos desenvolvidos, que se deu pela aplicação da abordagem frequentista, tal qual exposta em Hadad et al. (2007). Em seguida, foi feita a modelagem da rede bayesiana, principal foco do estudo, e de outros modelos de aprendizagem de máquina para efeito de comparação. Por fim, computou-se métricas de avaliação com as quais se fez possível a comparação entre os modelos desenvolvidos.

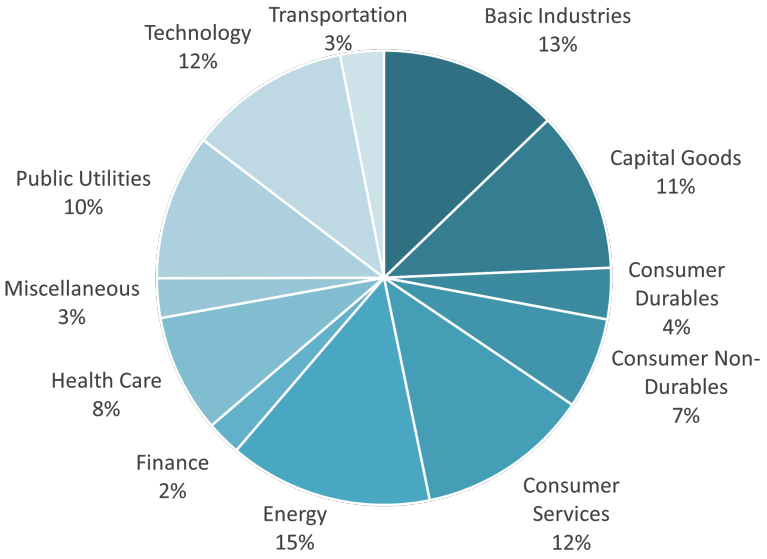
3.2 Base de dados utilizada

3.2.1 Análise exploratória

A base de dados utilizada é de domínio aberto e está disponível no Kaggle, sob o nome de *Corporate Credit Rating*.

Ela é composta por 2029 observações de rating de 593 empresas listadas na NASDAQ ou na NYSE entre 2005 e 2016, divididas em 12 setores, cuja distribuição está representada graficamente na Figura 2. As avaliações são realizadas entre 5 agências de rating, as 3 citadas anteriormente e outras 2, a Egan-Jones Ratings Company [4] e a DBRS [5], sendo que o rating é dado na escala global em vez de ser dado na escala específica de cada agência.

Figura 2 – Distribuição de setores na base de dados



Em uma análise exploratória inicial, observa-se uma distribuição desbalanceada de atribuições de rating, como exposto na Figura 3. Isso é esperado, uma vez que a maior parte dos emissores de fato devem estar classificados em ratings intermediários, já que a escala é comparativa entre eles.

3.2.2 Indicadores financeiros

Quanto as variáveis numéricas atribuídas a cada observação, tem-se uma lista de 25 indicadores financeiros que representam os valores mais recentes

[4] https://www.egan-jones.com/
[5] https://www.dbrsmorningstar.com/

Figura 3 – Distribuição de rating na base de dados

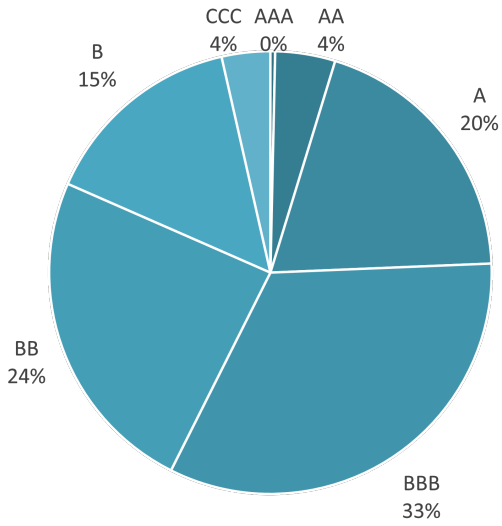
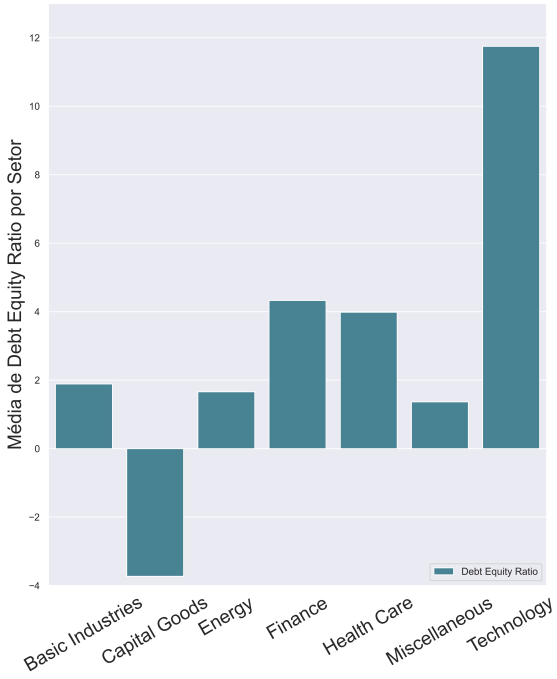


Figura 4 – Média de Debt Equity Ratio por Setor



disponíveis para o público na data da avaliação. Tais indicadores podem ser agrupados em 5 grupos de indicadores: liquidez, rentabilidade, endividamento, performance operacional e fluxo de caixa. A atribuição desses indicadores para cada um desses grupos é comumente utilizada em várias aplicações no mercado financeiro, como indica Hajek e Michalak (2013).

Como o modelo a ser utilizado será baseado em uma rede bayesiana, se fez necessário selecionar uma variável de cada grupo, de forma a reduzir o número de nós na rede, simplificando seu treinamento e criação. Para isso, com o apoio teórico principal dado por Hajek e Michalak (2013), selecionou-se as seguintes variáveis: current ratio, gross profit margin, debt equity ratio, asset turnover, operating cash flow per share.

Ainda a respeito dos indicadores financeiros, é consenso dentro do mercado financeiro que esses indicadores não podem ser utilizados como comparação entre empresas de diferentes setores. Como exemplo, uma instituição finaceira comumente possui um nível de alavancagem muito maior que uma empresa do setor de consumo, uma vez que a própria atividade (captação e redistribuição de recursos) da primeira implica em endividamento. Para validar essa hipótese, a Figura 4 indica a média de um desses indicadores para alguns desses setores, para ilustrar que existe uma diferença muito grande entre setores.

Constatado isso, foi feito um pré-tratamento na base de treino, onde foi realizada uma categorização dessas variáveis numéricas, de acordo com os setores. Para cada setor e para cada indicador, foi computado o percentil de 30% e 70%, classificando cada observação como 0 se estivesse abaixo do percentil 30%, 1 se estivesse entre 30% e 70%, e 2 caso contrário. Assim, para cada observação na base de dados, as variáveis numéricas de interesse foram transformadas em variáveis categóricas ordenadas. Isso foi feito para reduzir a dimensionalidade das matrizes de probabilidade conjuntas da rede bayesiana.

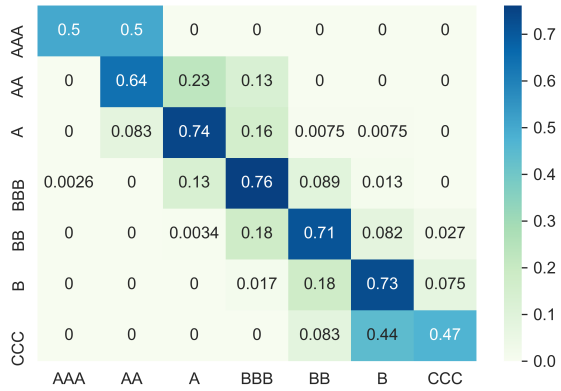
3.2.3 Variável macroeconômica

Baseado em Wang e Ku (2021) e Tavana et al. (2018), uma hipótese aceitável é a de que dados macroeconômicos são evidências válidas para a predição de rating. Com isso, foi escolhido o retorno anual do índice S&P500 como variável explicativa macroeconômica. De fato, existe uma extensa bibliografia, como cita JareÑo e Negrut (2016), que valida essa relação. De forma análoga aos indicadores financeiros, tais retornos também foram categorizados, sendo 1 se o retorno daquele ano fosse maior que a média histórica, e 0 caso contrário. O processo para reunir esse dados, bem como tratá-los está exposto no Apêndice C.

3.3 Matriz de transição

Para se obter um modelo de base de comparação, foi elaborada uma matriz de transição com aborda-

Figura 5 – Matriz de transição frequentista.



gem frequentista, tal qual exposto em Hadad et al. (2007). O código utilizado está exposto no Apêndice A. A abordagem para construir esse modelo é simples. Primeiramente, seleciona-se empresas que tenham sido avaliadas em anos consecutivos. Dado o rating no primeiro ano e o rating no segundo ano, uma contagem é feita de eventuais transições. Por fim, essa contagem é dividida pelo número total de ocorrências, de forma se obter uma probabilidade. A matriz resultante é apresentada na Figura 5.

3.4 Rede Bayesiana

O modelo de rede bayesiana foi desenvolvido com base nos estudos apresentados na Seção 2 e com o auxílio da biblioteca `pgmpy`[6] do Python. Assim como no modelo base, o estado a ser previsto, isto é, o rating futuro, é modelado como uma função do estado anterior, bem como de características fixas da empresa, no caso, o setor.

Assim, a rede desenvolvida, exposta na Figura 6, é divida em três grupos de nós. O primeiro grupo engloba os indicadores financeiros no estado anterior, isto é, os indicadores financeiros mais atuais no momento da previsão do rating. O segundo grupo diz respeito à influência macroeconômica relativa ao setor da empresa. Esses dois grupos compõem as evidências para o terceiro grupo, denominado Micro, que representa uma estimativa dos indicadores financeiros futuros. Cada uma dessas estimativas, por sua vez, configura uma evidência para o real valor do rating futuro, bem como o valor atual do rating.

Apenas para efeito de comparação, criou-se uma matriz de transição utilizando esse modelo. Ressalta-se que essa matriz simplifica o modelo, pois, em uma predição usual do modelo, os indicadores financeiros, bem como o setor da empresa e o sinal do S&P500, seriam utilizados para refinar essa predição. Ainda assim, a Figura 7 representa uma matriz comparável à obtida no modelo anterior, de onde vê-se que o modelo bayesiano prevê uma concentração de probabilidade maior na diagonal principal, indicando que existe uma probabilidade alta da empresa manter seu rating.

Figura 7 – Matriz de transição do modelo bayesiano

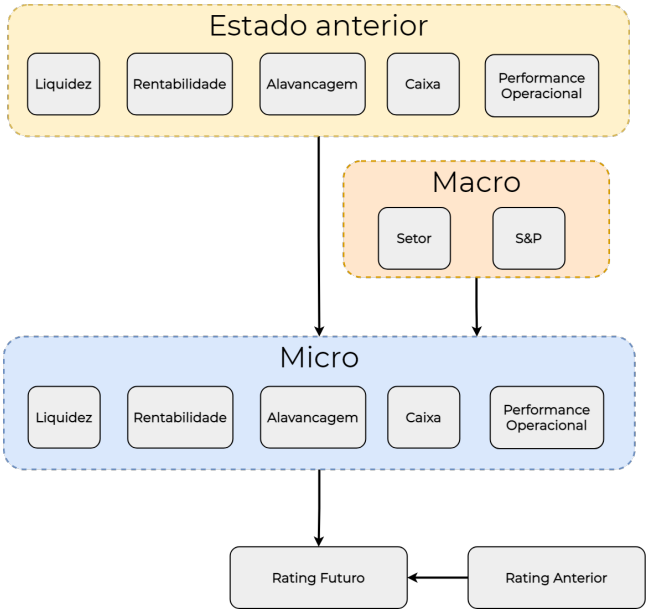
4 Resultados

Para efetuar comparação entre os modelos desenvolvidos, foi selecionado exemplos de teste na base de dados original. Para isso, foram selecionados exemplos de transição, isto é, um conjunto de duas observações de rating em anos seguidos. Para avaliar de forma não enviesada, tomou-se a precaução de balancear as classes de rating, tanto anteriores quanto posteriores, de forma que representassem suficientemente a base total. Esses exemplos foram retirados da base de treino para que não ocorresse enviesamento, também chamado de *data leakage*.

Feito isso, os modelos foram avaliados de acordo com duas métricas. A primeira foi a AUC ROC (*Area Under the Curve Receiver Operating Characteristic*, que pode ser interpretada como a probabilidade do modelo classificar um exemplo corretamente. Definições matemáticas e discussões mais aprofundadas sobre essa métrica podem ser observadas em (CLASSIFICATION. . . ,) e Davis e Goadrich (2006). Por essa medida, os modelos obtiveram resultados semelhantes, em específico, o primeiro modelo obteve um valor de 90.35% enquanto o segundo obteve 92.92%. Isso indica que ambos classificavam os modelos corretamente em pelo menos 9 a cada 10 casos, uma precisão considerável.

[6] <https://pgmpy.org/>

Figura 6 – Modelo de rede bayesiana desenvolvido



Porém, a segunda métrica avaliada apresentou uma discrepância maior. A segunda métrica escolhida foi o Erro Médio Absoluto (MAE), que é computado basicamente obtendo a média das diferenças das probabilidades atribuídas à classe correta e o valor correto, isto é, 1. Com isso, pode-se avaliar a escala das probabilidades atribuídas pelos modelos e não somente a classe prevista, em contraponto ao AUC ROC, que é *scale-invariant*, sendo portanto sensível apenas à qual classe tenha sido prevista e não à probabilidade atribuída a ela. Com essa métrica, o modelo inicial obteve um erro MAE de 0.1438, enquanto o segundo modelo obteve um erro de 0.0968, uma melhoria de quase 5 pontos percentuais.

Os resultados obtidos para cada modelo por métrica estão expostos na Tabela 1.

Tabela 1 – Métricas de comparação

Modelo	**MAE**	**AUC ROC**
Inicial	0.1438	90.35%
Bayesiano	0.0968	92.92%

5 Conclusão

Com os resultados expostos, é natural observar que não houve uma melhoria expressiva na acurácia do modelo ao adotar a rede bayesiana em comparação com o modelo de matriz de transição frequentista. Apesar disso, houve sim uma melhoria considerável quando avalia-se uma métrica sensível à escala da probabilidade atribuída à classe dominante, no caso, o MAE. De fato, os 5 pontos percentuais de melhoria no MAE podem representar uma vantagem competitiva considerável quando se é analisado o risco de crédito de um portfólio, dado que esse risco pode ser estimado como o produto entre a probabilidade de se alterar o rating do emissor e a piora no valor do ativo caso o emissor receba o downgrade. Com isso, conclui-se que o modelo pode ser utilizado para melhores estimativas de tal risco.

Futuras melhorias e possíveis futuros estudos incluem adicionar dados de mercado específicos para cada empresa, realizar melhorias na redução de dimensionalidade aplicada aos indicadores financeiros e, possivelmente, realizar uma versão do modelo para o mercado financeiro brasileiro.

Em contraponto, a bibliografia citada apresenta outras técnicas mais complexas, baseadas em algoritmos de aprendizagem de máquina e até redes neurais profundas que obtêm resultados ainda melhores e mais generalistas, dado que esse modelo está restrito aos dados obtidos. Assim, para aplicações práticas, é esperado que as instituições que necessitem de modelos para predição de rating deem preferência para este tipo em detrimento de modelos baseados em redes bayesianas.

Códigos

Os códigos para desenvolver os dois modelos estão presentes nos Apêndices. Todo o código desenvolvido está disponível no seguinte repositório do Github: <https://github.com/lucas-hattori-costa/PCS5708>.

Referências

CLASSIFICATION: ROC Curve and AUC. <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>. Accesso em: 2021-06-17. Citado na página 5.

DAVIS, J.; GOADRICH, M. The relationship between precision-recall and ROC curves. In: *Proceedings of the 23rd International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2006. (ICML '06), p. 233–240. ISBN 1595933832. Disponível em: <https://doi.org/10.1145/1143844.1143874>. Citado na página 5.

HADAD, M. D. et al. Rating migration matrices: empirical evidence in indonesia. *IFC Bulletin*, n. 31, p. 260–276, 2007. Citado 3 vezes nas páginas 2, 3 e 5.

HAJEK, P.; MICHALAK, K. Feature selection in corporate credit rating prediction. *Knowledge-Based Systems*, v. 51, p. 72–84, 2013. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705113002104>. Citado 2 vezes nas páginas 2 e 4.

JAREÑO, F.; NEGRUT, L. US stock market and macroeconomic factors. *Journal of Applied Business Research*, v. 32, p. 325–340, 01 2016. Citado na página 4.

LANDO, D.; SKØDEBERG, T. M. Analyzing rating transitions and rating drift with continuous observations. *Journal of Banking and Finance*, v. 26, p. 423–444, 2002. Citado na página 2.

LEE, Y.-C. Application of support vector machines to corporate credit rating prediction. *Expert Systems with Applications*, v. 33, n. 1, p. 67–74, 2007. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417406001205>. Citado na página 2.

TAVANA, M. et al. An artificial neural network and bayesian network model for liquidity risk assessment in banking. *Neurocomputing*, v. 275, p. 2525–2554, 2018. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231217317939>. Citado 2 vezes nas páginas 2 e 4.

WANG, M.; KU, H. Utilizing historical data for corporate credit rating assessment. *Expert Systems with Applications*, v. 165, p. 113925, 2021. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417420307156>. Citado 2 vezes nas páginas 2 e 4.

APÊNDICE A – MATRIZ DE TRANSIÇÃO

```python
import pandas as pd
from .utils import *

class TransitionMatrix():
    def __init__(self, df=None, val_df=None):
        self.name = 'Transition Matrix generated purely on frequentist approach'
        if df is None or val_df is None:
            self.df, self.val_df = read_data()
        else:
            self.df, self.val_df = df, val_df
        self.df['rating'] = self.df['n_rating']
        self.compute_transition_matrix()

    def make_prediction(self,prevRating):
        return self.matrix.loc[prevRating]

    def compute_val_score(self):
        val = self.val_df
        yPred = []
        yTrue = []
        for _, row in tqdm(val.iterrows()):
            try:
                nextRating = val.loc[(val['year']==row['year']+1) & (val['Symbol']==row['Symbol']), 'n_rating'].iloc[0]
            except IndexError:
                continue
            probNextRating = self.make_prediction(row['Rating']).values
            yPred.append(list(probNextRating))
            yTrue.append(nextRating)
        return np.array(yPred),np.array(yTrue)


    def compute_transition_matrix(self, y_final: int=2017) -> pd.DataFrame:
        '''
        Computa a matrix de transicao dos ratings utilizando os dados disponiveis em input_df para o periodo ate y_final.
        '''
        df = self.df.copy()
        df = df.sort_values(['year'], ascending=True)
        years = np.arange(2010, y_final)
        freq_matrix = {prev_rating:{next_rating:0 for next_rating in range(7)} for prev_rating in range(7)}
```

```
        total_cases_matrix = {prev_rating:0 for prev_rating in range(7)}

        for y in tqdm(years):
            for t in tickers_in_years(y,df):
                prev_rating, next_rating = df.loc[
                    ( (df['Symbol']==t) & ((df['year']==y) | (df['year']==y
+1)) ),
                    'rating'
                ][:2]
                freq_matrix[prev_rating][next_rating] += 1

        freq_matrix = pd.DataFrame(freq_matrix)
        prev_total_cases = freq_matrix.sum()
        for rating in range(7):
            freq_matrix[rating] /= prev_total_cases[rating]
        transition_matrix = freq_matrix.T.fillna(0)
        for rating in range(7):
            if transition_matrix.loc[rating].sum() == 0:
                transition_matrix.loc[rating, rating] = 1.0

        transition_matrix.columns = RATINGS
        transition_matrix.index = RATINGS
        prev_total_cases.index = RATINGS

        self.matrix = transition_matrix
        self.prev_total_cases = prev_total_cases
```

APÊNDICE B – REDE BAYESIANA

```
'''
Architecture that relies only on microeconomic data, i.e., the ratios.
'''
import pandas as pd
import numpy as np
from tqdm import tqdm
from datetime import datetime
import itertools
import pickle

from pgmpy.factors.discrete import TabularCPD
from pgmpy.models import BayesianModel
from pgmpy.inference import BeliefPropagation

from .utils import *

class MicroRatio():
    def __init__(self, name):
        self.name = name

    def _compute_n_line(self, micro: tuple, nextRatio: int = 0) -> int:
        n_line= micro[0]*12*3*2+\
                micro[1]*3*2+\
                micro[2]*3+\
                nextRatio
        return int(n_line)

    def compute_cpd_matrix(self,df):
        freqs = np.zeros((12*3*3*2,))
        combinations = itertools.product(np.arange(3),np.arange(12),np.
    arange(2))
        normalizers = {(r,s,sp):0 for (r,s,sp) in combinations}
        probs = np.zeros((12*3*3*2,))
        df = df.sort_values(['year'])


        for y in (df['year'].unique()):
            for t in tickers_in_years(y,df):
                df1 = df.loc[((df['year']==y) | (df['year']==y+1)) & (df['
    Symbol']==t)]
                prevs = df1.iloc[0]
                ratioIdx = np.where(df1.columns==self.name)[0][0]
                nextRating = df1.iloc[1,ratioIdx]
```

```
                aux_var = (prevs[self.name], prevs['sector'], prevs['sp_500
    '])
                n_line = self._compute_n_line(aux_var, nextRating)
                freqs[n_line] += 1
                normalizers[tuple(int(x) for x in aux_var)] += 1


        for k in normalizers.keys():
            if normalizers[k] == 0:
                # Se nenhum caso foi observado, a probabilidade padrao
    associada
                # é de 1.0 para o caso de manter o rating.
                n_line= self._compute_n_line(k, k[0])
                probs[n_line] = 1
            else:
                for i in range(3):
                    n_line= self._compute_n_line(k, i)
                    probs[n_line] = freqs[n_line] / normalizers[k]
        cpd_matrix = []
        for i in range(3):
            cpd_row = []
            for tup in normalizers.keys():
                cpd_row.append(probs[self._compute_n_line(tup,i)])
            cpd_matrix.append(cpd_row)
        return cpd_matrix

class NextRating():
    def __init__(self):
        pass

    def _compute_n_line(self, micro: tuple, nextRating: int = 0) -> int:
        n_line= micro[0]*7*(3**5)+\
                micro[1]*7*(3**4)+\
                micro[2]*7*(3**3)+\
                micro[3]*7*(3**2)+\
                micro[4]*7*(3**1)+\
                micro[5]*7*(3**0)+\
                nextRating
        return int(n_line)

    def compute_cpd_matrix(self,df):
        ## Criar a matrix de probabilidade de NEXT RATING
        freqs = np.zeros(((7**2)*(3**5),))
        combinations = itertools.product(np.arange(7),np.arange(3),np.
    arange(3),np.arange(3),np.arange(3),np.arange(3))
        normalizers = {tup:0 for tup in combinations}
        probs = np.zeros(((7**2)*(3**5),))
```

```
86          df = df.sort_values(['year'])
87
88
89          for y in (df['year'].unique()):
90              for t in tickers_in_years(y,df):
91                  df1 = df.loc[((df['year']==y) | (df['year']==y+1)) & (df['
    Symbol']==t)]
92                  prevs = df1.iloc[0]
93                  nextRating = df1.iloc[1,2]
94                  aux_var = (prevs['rating'], prevs['liquidity'], prevs['
    profitability'], prevs['debt'], prevs['cash'],prevs['asset'])
95                  n_line = self._compute_n_line(aux_var, nextRating)
96                  freqs[n_line] += 1
97                  normalizers[tuple(int(x) for x in aux_var)] += 1
98
99
100         for k in normalizers.keys():
101             if normalizers[k] == 0:
102                 # Se nenhum caso foi observado, a probabilidade padrao
    associada
103                 # é de 1.0 para o caso de manter o rating.
104                 n_line= self._compute_n_line(k, k[0])
105                 probs[n_line] = 1
106             else:
107                 for i in range(7):
108                     n_line= self._compute_n_line(k, i)
109                     probs[n_line] = freqs[n_line] / normalizers[k]
110         cpd_matrix = []
111         for i in range(7):
112             cpd_row = []
113             for tup in normalizers.keys():
114                 cpd_row.append(probs[self._compute_n_line(tup,i)])
115             cpd_matrix.append(cpd_row)
116         return cpd_matrix
117
118 class MicroNet():
119     def __init__(self):
120         self.name = 'Bayesian Network that relies only on microeconomic
    data to predict credit rating'
121         self.chosen_vars = ['currentRatio', 'grossProfitMargin', '
    debtEquityRatio', 'assetTurnover', 'operatingCashFlowPerShare']
122         self.df, self.val_df = read_data()
123
124     def save_model(self, path):
125         if not hasattr(self, 'model'):
126             print('Buildando com df padrao')
127             self.build()
```

```
128         with open(path, 'wb') as file:
129             pickle.dump(self.model, file)
130 
131     @staticmethod
132     def build_from_pickle(path):
133         with open(path,'rb') as file:
134             model = pickle.load(file)
135         return model
136 
137     def build(self, df=None):
138         if df is None:
139             df = self.df.copy()
140         df = enconding_ratios(df,self.chosen_vars)
141         df = df[['Symbol','n_sector','n_rating','year']+[f"d_{x}" for x in
    self.chosen_vars]+['sp500']]
142         df.columns = ['Symbol','sector','rating','year','liquidity','
    profitability','debt','asset','cash','sp_500']
143 
144         ## Indenpendent Variables
145         prevLiqu = TabularCPD(
146             variable='previousLiquidity',variable_card=3,
147             values=np.array([df.groupby(['liquidity']).count().iloc[:,0] /
    df.shape[0]]).T
148         )
149         prevProf = TabularCPD(
150             variable='previousProfitability',variable_card=3,
151             values=np.array([df.groupby(['profitability']).count().iloc
    [:,0] / df.shape[0]]).T
152         )
153         prevDebt = TabularCPD(
154             variable='previousDebt',variable_card=3,
155             values=np.array([df.groupby(['debt']).count().iloc[:,0] / df.
    shape[0]]).T
156         )
157         prevCash = TabularCPD(
158             variable='previousCash',variable_card=3,
159             values=np.array([df.groupby(['cash']).count().iloc[:,0] / df.
    shape[0]]).T
160         )
161         prevAsset = TabularCPD(
162             variable='previousAssetTurnover',variable_card=3,
163             values=np.array([df.groupby(['asset']).count().iloc[:,0] / df.
    shape[0]]).T
164         )
165         sector = TabularCPD(
166             variable='sector',variable_card=12,
167             values=np.array([df.groupby(['sector']).count().iloc[:,0] / df.
```

```
shape[0]]).T
        )
        rating = TabularCPD(
            variable='rating',variable_card=7,
            values=np.array([df.groupby(['rating']).count().iloc[:,0] / df.
shape[0]]).T
        )
        sp500 = TabularCPD(
            variable='sp_500',variable_card=2,
            values=np.array([df.groupby(['sp_500']).count().iloc[:,0] / df.
shape[0]]).T
        )

        ## DEPENDENT VARIABLES
        liqu = TabularCPD(
            variable='liquidity',variable_card=3,
            evidence=['previousLiquidity','sector','sp_500'],evidence_card
=[3,12,2],
            values=MicroRatio('liquidity').compute_cpd_matrix(df)
        )
        prof = TabularCPD(
            variable='profitability',variable_card=3,
            evidence=['previousProfitability','sector','sp_500'],
evidence_card=[3,12,2],
            values=MicroRatio('profitability').compute_cpd_matrix(df)
        )
        debt = TabularCPD(
            variable='debt',variable_card=3,
            evidence=['previousDebt','sector','sp_500'],evidence_card
=[3,12,2],
            values=MicroRatio('debt').compute_cpd_matrix(df)
        )
        cash = TabularCPD(
            variable='cash',variable_card=3,
            evidence=['previousCash','sector','sp_500'],evidence_card
=[3,12,2],
            values=MicroRatio('cash').compute_cpd_matrix(df)
        )
        asset = TabularCPD(
            variable='asset',variable_card=3,
            evidence=['previousAssetTurnover','sector','sp_500'],
evidence_card=[3,12,2],
            values=MicroRatio('asset').compute_cpd_matrix(df)
        )

        ## FINAL VARIABLE
        nextRatingCPD = TabularCPD(
```

```
207             variable='nextRating',variable_card=7,
208             evidence=['rating','liquidity','profitability','debt','cash','
    asset'],evidence_card=[7,3,3,3,3,3],
209             values=NextRating().compute_cpd_matrix(df)
210         )
211 
212         model = BayesianModel([
213             ('sector','liquidity'),
214             ('sector','profitability'),
215             ('sector','debt'),
216             ('sector','cash'),
217             ('sector','asset'),
218             ('sp_500','liquidity'),
219             ('sp_500','profitability'),
220             ('sp_500','debt'),
221             ('sp_500','cash'),
222             ('sp_500','asset'),
223             ('previousLiquidity','liquidity'),
224             ('previousProfitability','profitability'),
225             ('previousDebt','debt'),
226             ('previousCash','cash'),
227             ('previousAssetTurnover','asset'),
228             ('liquidity','nextRating'),
229             ('profitability','nextRating'),
230             ('debt','nextRating'),
231             ('cash','nextRating'),
232             ('asset','nextRating'),
233             ('rating','nextRating')
234         ])
235 
236         # Associating the parameters with the model structure.
237         model.add_cpds(
238             prevLiqu, prevProf, prevDebt, prevCash, prevAsset,
239             rating, sector, sp500,
240             liqu, prof, debt, cash, asset,
241             nextRatingCPD
242         )
243         assert model.check_model()
244 
245         self.model = model
246 
247     def create_bayesian_matrix(self):
248         if not hasattr(self, 'model'):
249             raise ValueError('You need to build the model first! Use the
    method .build()')
250 
251         matrix = pd.DataFrame(columns=RATINGS)
```

```
252         for rat in range(7):
253             q = BeliefPropagation(self.model).query(variables=['nextRating'], evidence={'rating':rat})
254             matrix.loc[RATINGS[rat]] = q.values
255 
256         return matrix
257 
258     def make_prediction(self, evidence: dict):
259         q = BeliefPropagation(self.model).query(variables=['nextRating'], evidence=evidence)
260         return q.values
261 
262     def compute_val_score(self):
263         if not hasattr(self, 'model'):
264             raise ValueError('You need to build the model first! Use the method .build()')
265         yPred = []
266         yTrue = []
267         val = self.val_df.copy()
268 
269         for _, row in tqdm(val.iterrows()):
270             try:
271                 nextRating = val.loc[(val['year']==row['year']+1) & (val['Symbol']==row['Symbol']), 'n_rating'].iloc[0]
272             except IndexError:
273                 continue
274             probNextRating = self.make_prediction(evidence={
275                 'rating':row['n_rating'],
276                 'sector':row['n_sector'],
277                 'sp_500':row['sp500'],
278                 'previousLiquidity':row['d_currentRatio'],
279                 'previousProfitability':row['d_grossProfitMargin'],
280                 'previousDebt':row['d_debtEquityRatio'],
281                 'previousCash':row['d_assetTurnover'],
282                 'previousAssetTurnover':row['d_operatingCashFlowPerShare']
283             })
284             yPred.append(list(probNextRating))
285             yTrue.append(nextRating)
286 
287         return np.array(yPred),np.array(yTrue)
288 
289     def LOOCV(self):
290         total_error = []
291         print(f'Comeca LOOCV - {datetime.now()}')
292         for i in tqdm(range(1230)):
293             df, ex = select_val_example(df=self.df,chosen_vars=self.chosen_vars,idx=i)
```

```
294             predIdx, trueIdx = ex.index
295             NextRating = ex.loc[trueIdx,'n_rating']
296             self.build(df)
297             probNextRating = self.make_prediction(evidence={
298                 'rating':ex.loc[predIdx,'n_rating'],
299                 'previousLiquidity':ex.loc[predIdx,'d_currentRatio'],
300                 'previousProfitability':ex.loc[predIdx,'d_grossProfitMargin
    '],
301                 'previousDebt':ex.loc[predIdx,'d_debtEquityRatio'],
302                 'previousCash':ex.loc[predIdx,'d_assetTurnover'],
303                 'previousAssetTurnover':ex.loc[predIdx,'
    d_operatingCashFlowPerShare']
304             }).loc[RATINGS[NextRating],0]
305 
306             total_error.append(1-probNextRating)
307 
308         return total_error
```

APÊNDICE C – DADOS MACROECONÔMICOS

```
import yfinance as yf
import numpy as np

original_df = yf.download(tickers="^GSPC", interval="3mo")

df = original_df.copy()
df = df.iloc[:-1]
df = df[['Adj Close']].reset_index()
df['month'] = df['Date'].map(lambda x: x.month)
df['year'] = df['Date'].map(lambda x: x.year)
df = df.loc[df['month']==6]
df['Annual Returns'] = np.log(df['Adj Close']) - np.log(df['Adj Close'].
    shift(1))

enconding_sp500 = {}
for y in np.arange(2009, 2017+1):
    if df.loc[df.year==y, 'Annual Returns'].iloc[0] >= df.loc[df.year<y, '
    Annual Returns'].quantile(0.50):
        enconding_sp500[y] = 1
    else:
        enconding_sp500[y] = 0
```