1. **Modelagem de dados de finanças para a IA**

Bom como visto anteriormente os dados de finanças são mais complexos, e apesar das informações estarem dispostas para o usuário de uma forma mais acessível pelos dados abertos da CVM. É necessário fazer um tratamento inicial antes de começar usar os dados. O tratamento, vai desde um simples filtro nas bases de cada arquivo à separação de indicadores. Vamos mostrar o passo a passo de como fizemos para tratar os dados para serem recebidos pelos modelos iniciais de Clusterização e Regressão Polimonial.

Posteriormente todos esses tratamentos serão realizados no banco de dados, ou na ingestão de dados. Mas primeiramente é importante darmos uma olhada em como são esses dados em sua forma bruta, segue:

Tabela

Descrição gerada automaticamente

DRE -2022 exemplo

Tabela

Descrição gerada automaticamente

BPP -2022

Tabela

Descrição gerada automaticamente

BPA -2022

Essas 3 bases juntas têm aproximadamente 1 milhão de linhas em relação a dados são aproximadamente 14 milhões de dados por ano, como estamos fazendo o tratamento com o histórico das informações (11 anos), estamos falando de aproximadamente de 157 milhões de informações que são processadas.

O motivo pelo qual estamos utilizando apenas 3 tipos de movimentos contábeis (BPP, BPA e DRE) é que esses movimentos têm alguns lançamentos contábeis/indicadores que consideramos interessantes para utilizar pelos modelos de ML, outro detalhe importante é que estamos utilizando apenas os indicadores individuais e não os consolidados. E estamos utilizando um loop para empilhar todo o histórico em uma variável.

1.1 Modelagem do DRE

Basicamente a ideia de modelagem do DRE, consiste em tirar algumas colunas que não serão utilizadas na aplicação do modelo, renomear colunas e filtrar as datas de exercícios respectivos daquele arquivo, exemplo: No DRE de 2022, tem informações trimestrais de 2022 e YTD (Year to Date), ou seja vai ter informações de 2021. O que fazemos é pegar somente as informações dos trimestres do ano daquele arquivo. Olha como ficou o código:

import pandas as pd

import numpy as np

DRE=pd.DataFrame()

for x in range(11,23):

    df=pd.read\_csv(fr"Dados\_Brutos\itr\_cia\_aberta\_DRE\_ind\_20{x}.csv",delimiter=';',encoding="latin-1",decimal='.')

    df=df[['DENOM\_CIA','CNPJ\_CIA','CD\_CONTA','DT\_INI\_EXERC','DT\_REFER','DT\_FIM\_EXERC','ORDEM\_EXERC','DS\_CONTA','VL\_CONTA','ST\_CONTA\_FIXA']]

    df['year1'] = pd.DatetimeIndex(df['DT\_REFER']).year

    df['month1'] = pd.DatetimeIndex(df['DT\_REFER']).month

    ano=int(f"20{x}")

    df=df[(df["year1"]==ano)].reset\_index(drop=True)

    df=df[(df["month1"]==3)|(df["month1"]==6)|(df["month1"]==9)|(df["month1"]==12)].reset\_index(drop=True)

    df['year1'] = pd.DatetimeIndex(df['DT\_FIM\_EXERC']).year

    df['month1'] = pd.DatetimeIndex(df['DT\_FIM\_EXERC']).month

    df['year2'] = pd.DatetimeIndex(df['DT\_INI\_EXERC']).year

    df['month2'] = pd.DatetimeIndex(df['DT\_INI\_EXERC']).month

    df=df[(df["month1"]==3)&(df["month2"]==1)|(df["month1"]==6)&(df["month2"]==4)|(df["month1"]==9)&(df["month2"]==7)|(df["month1"]==12)&(df["month2"]==10)].reset\_index(drop=True)

    df=df[(df["ST\_CONTA\_FIXA"]=="S")&(df["ORDEM\_EXERC"]=="ÚLTIMO")].reset\_index(drop=True)

    df["Nome Empresa"]=df['DENOM\_CIA']

    df=df[['Nome Empresa','CNPJ\_CIA','DT\_REFER','DS\_CONTA','CD\_CONTA','VL\_CONTA']]

    DRE=pd.concat([df,DRE]).reset\_index(drop=True)

1.1 Modelagem do BPA e BPP

Eu faço as mesmas mudanças e aplicação de filtros nos dados de BPA e BPP, como ambos os arquivos têm o mesmo padrão de dados, por serem informações opostas, o código é o mesmo. Vou mostrar o código do BPA, a única diferença é na hora de ler o arquivo que seria “itr\_cia\_aberta\_BPP\_ind”, segue o código:

import pandas as pd

import numpy as np

BPA=pd.DataFrame()

for x in range(11,23):

    df=pd.read\_csv(fr"Dados\_Brutos\itr\_cia\_aberta\_BPA\_ind\_20{x}.csv",delimiter=';',encoding="latin-1",decimal='.')

    df=df[['DENOM\_CIA','CNPJ\_CIA','CD\_CONTA','DT\_REFER','DT\_FIM\_EXERC','ORDEM\_EXERC','DS\_CONTA','VL\_CONTA','ST\_CONTA\_FIXA']]

    ano=int(f"20{x}")

    df['year1'] = pd.DatetimeIndex(df['DT\_FIM\_EXERC']).year

    df['month1'] = pd.DatetimeIndex(df['DT\_FIM\_EXERC']).month

    df=df[(df["year1"]==ano)].reset\_index(drop=True)

    df=df[(df["month1"]==3)|(df["month1"]==6)|(df["month1"]==9)|(df["month1"]==12)].reset\_index(drop=True)

    df=df[(df["ST\_CONTA\_FIXA"]=="S")&(df["ORDEM\_EXERC"]=="ÚLTIMO")].reset\_index(drop=True)

    df["Nome Empresa"]=df['DENOM\_CIA']

    df=df[['Nome Empresa','CNPJ\_CIA','DT\_REFER','DS\_CONTA','CD\_CONTA','VL\_CONTA']]

    BPA=pd.concat([df,BPA]).reset\_index(drop=True)

1.2 Modelagem final

Após realizar a etapa de pegar somente as informações necessárias e empilhar um ano no outro. Precisamos juntar tudo isso (como agora as informações têm o mesmo padrão não teria conflitos), precisamos separar somente os indicadores que escolhemos a dedo, acrescento algumas informações adicionais e faço uma pivotagem para colocar os indicadores como coluna, pois dessa forma fica mais fácil para o ingerir os dados no modelo de ML. Um detalhe importante é que como existe vários segmentos de negócio na bolsa, que podem prestar serviços ou vender produtos, o conceito de receita pode mudar e será apresentado nos balanços de formas diferentes, e para isso temos que tratar esse indicador relacionando tudo que é receita em um indicador de receita. A seguir fazemos alguns tratamentos finais:.

#empilhar o DRE. BPA e BPP

Completao=pd.concat([BPP,BPA,DRE]).reset\_index(drop=True)

#Fazer algumas mudanças de nomenclatura do nome dos indicadoress que apresentaram divergências

Completao['DS\_CONTA']=Completao['DS\_CONTA'].str.replace('Reservas de Lucros', 'Reservas de Lucro', regex=True)

Completao['DS\_CONTA']=Completao['DS\_CONTA'].str.replace('Obrigações sociais e trabalhistas', 'Obrigações Sociais e Trabalhistas', regex=True)

Completao['DS\_CONTA']=Completao['DS\_CONTA'].str.replace('Outras Obrigações Sociais e Trabalhistas', 'Obrigações Sociais e Trabalhistas', regex=True)

#Escolher somente os indicadores que escolhermos.

Completao=Completao[(Completao['DS\_CONTA']=='Outras Despesas/Receitas Operacionais')|

                    (Completao['DS\_CONTA']=='Despesas/Receitas Operacionais')|

                    (Completao['DS\_CONTA']=='Receitas de Prestação de Serviços')|

                    (Completao['DS\_CONTA']=='Receita de Venda de Bens e/ou Serviços')|

                    (Completao['DS\_CONTA']=='Lucro/Prejuízo do Período')|

                    (Completao['DS\_CONTA']=='Ativo Total')|

                    (Completao['DS\_CONTA']=='Ativo Circulante')|

                    (Completao['DS\_CONTA']=='Contas a Receber')|

                    (Completao['DS\_CONTA']=='Passivo Total')|

                    (Completao['DS\_CONTA']=='Passivo Circulante')|

                    (Completao['DS\_CONTA']=='Obrigações Sociais e Trabalhistas')|

                    (Completao['DS\_CONTA']=='Obrigações Fiscais')|

                    (Completao['DS\_CONTA']=='Reservas de Lucro')|

                    (Completao['DS\_CONTA']=='Patrimônio Líquido')].reset\_index(drop=True)

Completao2=Completao

#Coloco algumas informações adicionais, como o tick de cada empresa da bolsa

cad=pd.read\_csv(fr"Depara\cad.csv",delimiter=';',encoding="latin-1",decimal='.')

cad=cad[['DENOM\_SOCIAL','SIT','SETOR\_ATIV','TP\_MERC','CNPJ\_CIA']]

tick=pd.read\_excel(fr"Depara\tick.xlsx")

DEpara = pd.merge( tick, cad, left\_on='CNPJ', right\_on='CNPJ\_CIA')

DEpara=DEpara[(DEpara['SIT']=='ATIVO') &(DEpara['TP\_MERC']=='BOLSA')].reset\_index(drop=True)

DEpara=DEpara.drop\_duplicates(subset="Ticker")

DEpara = DEpara.drop(columns=['CNPJ'])

Completao2 = pd.merge( Completao2, DEpara, left\_on='CNPJ\_CIA', right\_on='CNPJ\_CIA')

Completao2 = Completao2.drop(columns=['CNPJ\_CIA','Razão Social','DENOM\_SOCIAL','SIT','TP\_MERC'])

Completao2['Ticker']=Completao2['Ticker'].apply(str).str[0:4]

Completao2=Completao2.drop\_duplicates(subset=['Nome Empresa','Ticker','SETOR\_ATIV','DT\_REFER','DS\_CONTA','VL\_CONTA'])

#Faço uma pivotagem com o dataset e transformo indicadores em colunas

table = pd.pivot\_table(Completao2, values='VL\_CONTA', index=['Nome Empresa','Ticker','SETOR\_ATIV', 'DT\_REFER'],columns=['DS\_CONTA'], aggfunc=np.sum, fill\_value=0)

table=table.reset\_index()

#Faço um tratamento com o indicador receita.

EMPRESAS=table['Ticker'].values

EMPRESAS= list(dict.fromkeys(EMPRESAS))

table2=pd.DataFrame()

for x in EMPRESAS:

    row=table[table["Ticker"]==x].reset\_index(drop=True)

    row["Receita"]=np.where(row['Receita de Venda de Bens e/ou Serviços']==0,row['Despesas/Receitas Operacionais'],row['Receita de Venda de Bens e/ou Serviços'])

    row["Receita"]=np.where(row['Receita']==0,row['Outras Despesas/Receitas Operacionais'],row['Receita'])

    row["Receita"]=np.where(row['Receita']==0,row['Receitas de Prestação de Serviços'],row['Receita'])

    if x=='ITUB':

        print(row[row["Ticker"]=='ITUB'])

    table2=pd.concat([table2,row]).reset\_index(drop=True)

#Faço um drop com as colunas que não serão mais úteis.

table2 = table2.drop(columns=['Receita de Venda de Bens e/ou Serviços', 'Despesas/Receitas Operacionais','Despesas/Receitas Operacionais','Receitas de Prestação de Serviços','Outras Despesas/Receitas Operacionais'])

#E por fim gero um arquivo com o tratamento

table2.to\_csv(rf"Cuspidão\Completao.csv",decimal=',',sep=';', encoding='latin-1',index=False)

Tabela

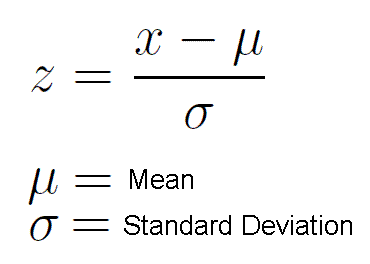
Descrição gerada automaticamente Por fim criamos uma “Base Única”, com todos os indicadores e tratamentos necessários para que possamos aplicar isso no modelo de machine learning. E por fim olha como ficou o resultado de todo esse tratamento:

Talvez pareça irrelevante essa modelagem, mas foi algo extremamente necessário para organizar os dados e como ponto positivo na visão de usuário, democratizamos a leitura das informações visto que nas bases originais tem muitas informações que podem causar confusão na primeira leitura de dados.

Por mais que o tratamento tenha sido simples, o entendimento dos dados para modelagem se deve as análises que fizemos com os documentos do RI de algumas empresas que separamos, a fim de entender qual indicador fazia mais sentido e como faríamos a extração deles. E como estamos falando de um assunto nichado que poucas pessoas de fato entende e é um pouco complexo de entender por ser contabilidade de várias empresas, podemos dizer que apesar da modelagem ser simples, a análise e conceitos aplicados tem muito valor.

Um ponto importante é que só separamos 11 indicadores que consideramos os mais importantes por enquanto. Se fossemos utilizar todos os indicadores presentes, teríamos o trabalho de relacionar indicadores de empresas diferentes, igual fizemos com o indicador de receita, e isso levaria bastante tempo. Futuramente podemos acrescentar todos ou mais alguns indicadores para complementar a análise da IA.

1. **Entendendo os Modelos utilizados.**

 Antes de mostrarmos a aplicação dos modelos, vamos falar um pouco sobre a parte conceitual e o porquê escolhermos esses modelos para escalonar, trazer visões futuras e clusterizar as empresas da bolsa. Bom começando pelo Standard Scaler, a ideia dele é basicamente padronizar os dados para que não tenha discrepância entre os valores. A fórmula dele é bem simples:

<https://towardsai.net/p/l/which-feature-scaling-technique-to-use-standardization-vs-normalization>

Testamos desde o Binarizer até o MinMax Scalling, e o Standard Scaller foi o que trouxe melhores resultados junto com a aplicação do PCA que reduz a dimensionalidade de dados. O PCA além de reduzir a dimensão dos dados, ele cria variáveis independentes, ou seja, os dados não são correlatos igual ao Min Max Scalling. A redução de dados ocorre porque os componentes principais do PCA representam o todo, sendo possível descartar os componentes que são redundantes. A ideia de utilizar o standard scaller é normalizar os inputs antes de colocar no PCA, que analisa a matriz de covariância das informações, calcula as direções de variação dos componentes principais escolhendo a que tiver maior variação, ordena os componentes e por fim projeta os dados. Segue exemplo dos componentes principais:

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

Fonte: http://primo.ai/index.php?title=Principal\_Component\_Analysis\_%28PCA%29

Uma das vantagens que acabamos não explorando com o PCA nessa primeira fase é a redução de componentes, onde poderíamos passar todos os indicadores financeiros como variáveis dependentes e ter um output dos componentes principais de todos os indicadores passados, isso faria nos reduzir tempo de processamento e teríamos uma clusterização definitiva com o kmeans. Visto que nessa primeira fase da APS clusterizamos os indicadores e depois fizemos uma média que chamamos de nota, que seria a avaliação da empresa baseada nas clusterizações dos indicadores. Ter uma visão consolidada e reduzida de todos os indicadores seria muito interessante, porque de fato o cluster seria baseado no desempenho da empresa então a recomendação/cluster seria mais fundamentada sem penalizar uma empresa boa pelo seu indicador ruim.

2.1 Kmeans conceitual

Após as normalizações dos dados, podemos aplicar o kmeans para clusterizar os dados de finanças. A gente utilizou esse modelo de clusterização no nosso projeto porque o nosso objetivo era classificar as empresas baseando-se em indicadores financeiros, e o kmeans tem uma praticidade muito grande quando tratamos de aplicação e entendimento do conceito matemático. O kmeans é um algoritmo não supervisionado, isso significa que ele clusteriza os dados sem precisar de uma classificação inicial, e isso é muito bom para o nosso caso, porém tem uma curva de aprendizado indefinida que é chamada de “elbow”, que significa cotovelo em inglês, e é uma forma da gente mensurar quantos clusters são ideais para os nossos dados.

O Kmeans clusteriza através de centroides que são gerados de forma aleatória baseando-se nos dados inputs e na quantidade de clusters que definimos inicialmente, como os centroides são gerados de forma aleatória eles são atualizados conforme as distancias euclidianas, isso faz com que as clusterizações cheguem em um consenso através de loops, ou seja o algoritmo vai se alto corrigindo, até chegar no cluster final. Segue um exemplo de clusterização:

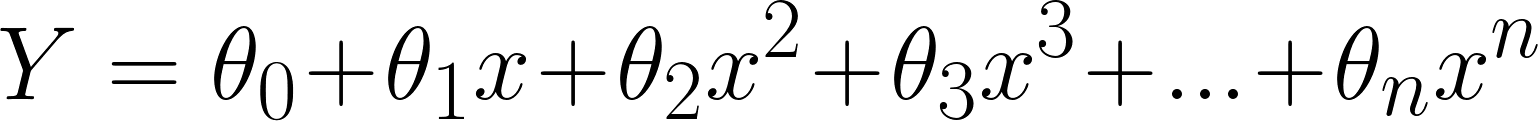


Fonte: <https://www.geeksforgeeks.org/ml-k-means-algorithm/>

No exemplo acima os pontinhos pretos são os centroides aleatórios que são calibrados ao longo do processo de clusterização. E cada conjunto colorido representa um cluster, poderíamos dizer por exemplo que o cluster azul é bom, o cluster amarelo é ruim e o verde é médio. A definição do que representa cada cluster vai depender da leitura dos dados que é feita pelo cientista de dados. O Problema é que quando tratamos de dimensões desproporcionais de dados e conjuntos diferentes, os clusters pode ter significados diferentes isso dificulta o processo de automatização de clusterização. Com os dados de finanças aconteceu isso, onde por exemplo o kmeans clusterizou empresas boas do setor X como 1 e do setor Y como 0. O que nos ajudou a arrumar esse problema, foi usar o PCA. Agora podemos dizer que o cluster 1 é bom em qualquer setor que estivermos analisando.

2.2 Regressão Polimonial

A regressão polimonial é uma regressão linear onde podemos colocar mais variáveis independentes e podemos ajustar o grau dos coeficientes da regressão, esse modelo de regressão é ideal para fazermos predições não lineares. O que nos retorna as curvas de predições invés de retas. Utilizamos principalmente para fazer predições de indicadores para os próximos 4 trimestres, analisando todo o histórico disponível da empresa. A fórmula é bem simples:



Fonte: <https://morioh.com/p/c507177830bd>

O nosso modelo utiliza o polinômio de grau 3 que apresentou um resultado satisfatório, e utilizamos datas como variáveis independentes. Esse modelo não é o core do nosso projeto, é só uma forma de nos mensurar indicadores futuros. O interessante é que para esse modelo não fizemos escalabilidades dos dados.

1. **Aplicando os modelos**

Antes de iniciar tivemos que fazer uma pré modelagem de dados que é extremamente mais simples do que a primeira modelagem que fizemos. A ideia dessa modelagem é nos separar variáveis globais e traduzir os indicadores para o modelo. Segue como ficou:

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler, normalize

from sklearn.decomposition import PCA

import pandas as pd

import numpy as np

from datetime import date, datetime

from dateutil.relativedelta import relativedelta

from sklearn.preprocessing import PolynomialFeatures

from sklearn.pipeline import make\_pipeline

from sklearn.linear\_model import LinearRegression

import warnings

warnings.filterwarnings("ignore")

#Subimos o arquivo que modelamos.

df=pd.read\_csv(r"Cuspidão\Completao.csv",delimiter=';',encoding="latin-1",decimal=',')

#Cria um variável com o nome de todos os setores do dataset e cria uma flag para se tem somente essa empresa no setor

ListofCompanies=df.drop\_duplicates(subset=['SETOR\_ATIV','Nome Empresa'])

ListofCompanies=ListofCompanies['SETOR\_ATIV'].value\_counts()

ListofCompanies=pd.DataFrame(ListofCompanies)

Listof1Companies=ListofCompanies[ListofCompanies['SETOR\_ATIV']<4]

ListofCompanies=ListofCompanies[ListofCompanies['SETOR\_ATIV']>=4]

ListofCompanies['Flag1Company']=0

Listof1Companies['Flag1Company']=1

ListofCompanies=pd.concat([ListofCompanies,Listof1Companies])

ListofCompanies['SETOR\_ATIV']=ListofCompanies.index

#Cria Variáveis de Cuspir

DF\_Polyline=pd.DataFrame()

DF\_Cuspidao=pd.DataFrame()

DF\_Cuspidao2=pd.DataFrame()

DF\_Cuspidao\_BAU=pd.DataFrame()

DF\_Cuspidao\_RESTO=pd.DataFrame()

df\_produ\_kmeans\_REST\_ALL=pd.DataFrame()

#Criar ano e mes

df['Ano']=df['DT\_REFER'].apply(str).str[0:4]

df['Mes']=df['DT\_REFER'].apply(str).str[5:7]

#Transformar colunas negativas em positivas

df['Passivo Circulante']=df['Passivo Circulante']\*-1

df['Passivo Total']=df['Passivo Total']\*-1

#Coloca a Flag de 1 company no dataset central

df1 = pd.merge( ListofCompanies,df, left\_on='SETOR\_ATIV', right\_on='SETOR\_ATIV')

df\_produ=df1

#Transforma as relações de indicadores para clusterizar em numeros, para o for andar de lado.

df\_produ[14]=df\_produ['Ativo Circulante'].clip(lower=0)

df\_produ[13]=df\_produ['Patrimônio Líquido'].clip(lower=0)

df\_produ[12]=df\_produ['Passivo Circulante'].clip(upper=0)

df\_produ[11]=df\_produ['Patrimônio Líquido'].clip(lower=0)

df\_produ[10]=df\_produ['Passivo Total'].clip(upper=0)

df\_produ[9]=df\_produ['Patrimônio Líquido'].clip(lower=0)

df\_produ[8]=df\_produ['Ativo Total'].clip(lower=0)

df\_produ[7]=df\_produ['Patrimônio Líquido'].clip(lower=0)

df\_produ[6]=df\_produ['Reservas de Lucro'].clip(lower=0)

df\_produ[5]=df\_produ['Receita'].clip(lower=0)

df\_produ[4]=df\_produ['Receita'].clip(lower=0)

df\_produ[3]=df\_produ['Patrimônio Líquido'].clip(lower=0)

df\_produ[2]=df\_produ['Receita'].clip(lower=0)

df\_produ[1]=df\_produ['Lucro/Prejuízo do Período'].clip(lower=0)

Como analisado, não fizemos nada de muito importante nessa primeira etapa. Gostaria de pontuar que tivemos que chamar os indicadores por números, para a gente conseguir andar de lado no for da clusterização.

3.1 Kmeans BAU e Rejeitados

Bom após nos modelar e escalar os dados, podemos prosseguir com a clusterização dos indicadores das empresas. Um ponto importante é que tivemos que segregar as análises, porque temos empresas na bolsa que tem proporções variadas, ou seja, é mesma coisa que comparar um trem bala com uma bicicleta. Então resolvemos segregar por setores, ou seja, clusterizamos as empresas de acordo com as empresas do seu setor, assim tivemos comparações mais justas.

O problema é que quando fazemos isso, temos que fazer uma segunda segregação, pois existe setores com só 1 empresa, e por isso temos 2 códigos de kmeans o BAU (Business as Usual) e os rejeitados, a diferença entre esses dois códigos é que no código dos rejeitados fazemos a clusterização só de dados que tenham 1 empresa no setor, em outras palavras tem um ‘for’ a menos.

O Workflow do BAU é o seguinte, primeiro ele filtra no dataset o setor, filtra o último trimestre, filtra as empresas que tenham informações sobre aquele indicador, depois faz escalabilidade com os dados, treina os dados com o “fit()” e por fim fazemos o “predict()”. Segue como ficou os códigos:

df\_provisorio=df\_produ

df\_provisorio=df\_provisorio[['Nome Empresa',

                             'Ticker',

                             'SETOR\_ATIV',

                             'Flag1Company',

                             'DT\_REFER',

                             'Ano',

                             'Mes',

                             1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

#BAU KMEANS

for y in range(1,13):

    for x in ListofCompanies.index.values:

        if y==1:

            y1=1

            y2=2

        else:

            y1=y+1

            y2=y+2

        #filtra o setor

        df\_produ\_kmeans=df\_provisorio[(df\_provisorio['SETOR\_ATIV']==x)&(df\_provisorio['Flag1Company']==0)].reset\_index(drop=True)

        Tcks\_count=df\_produ\_kmeans.drop\_duplicates(subset=['Ticker','DT\_REFER'])

        Tcks\_count=Tcks\_count['DT\_REFER'].value\_counts()

        Tcks\_count\_mean=Tcks\_count.mean()

        Tcks\_count=pd.DataFrame(Tcks\_count)

        ListOFTRI=Tcks\_count[Tcks\_count['DT\_REFER']>=Tcks\_count\_mean]

        ListOFTRI['DATE']=ListOFTRI.index

        AnoMAX=ListOFTRI['DATE'].apply(str).str[0:4].max()

        MesMAX=ListOFTRI['DATE'].apply(str).str[5:7].max()

        #Filtrar ano e mês (ultimo tri de cada setor)

        df\_produ\_kmeans=df\_produ\_kmeans[(df\_produ\_kmeans["Ano"]==AnoMAX)&(df\_produ\_kmeans["Mes"]==MesMAX)].reset\_index(drop=True)

        #Clusteriza as empresas com valores negativos como 0 (ruim)

        df\_produ\_kmeans\_REST\_NO=df\_produ\_kmeans[(df\_produ\_kmeans[y1]==0)|(df\_produ\_kmeans[y2]==0)].reset\_index(drop=True)

        df\_produ\_kmeans\_REST\_NO[f"{y1}{y2} Cluster"]=0

        #Separa as empresas que são elegíveis a escalabilidade e a clusterização

        df\_produ\_kmeans=df\_produ\_kmeans[(df\_produ\_kmeans[y1]!=0)&(df\_produ\_kmeans[y2]!=0)].reset\_index(drop=True)

        df\_produ\_kmeans=df\_produ\_kmeans.sort\_values(by=[y1,y2]).reset\_index(drop=True)

        #se tiver mais de 1 amostra (empresa)

        if len(df\_produ\_kmeans)>=2:

            #FAÇO A ESCALABILIDADE COM STANDARD

            scaler = StandardScaler()

            pca = PCA(n\_components = 2)

            features= df\_produ\_kmeans[[y1,y2]]

            scaled\_df = scaler.fit\_transform(features)

            normalized\_df = normalize(scaled\_df)

            normalized\_df = pd.DataFrame(normalized\_df)

            #E DEPOIS REDUZO AS DIMENSÕES E CORRELAÇÕES COM O PCA

            X\_principal = pca.fit\_transform(normalized\_df)

            features = pd.DataFrame(X\_principal)

            features.columns = ['P1', 'P2']

            c=3

            if len(features)<3:

                c=len(features)

            #DOU UM FIT NO KMEANS

            kmeans = KMeans(n\_clusters=c, init='random',

                        n\_init=10,

                        max\_iter=300,random\_state=0)

            kmeans.fit(features)

            #DPS EU FAÇO A CLUSTERIZAÇÃO

            d = KMeans(n\_clusters=c, init='random',

                        n\_init=10,

                        max\_iter=300,random\_state=0).fit\_predict(features)

            df\_produ\_kmeans[f"{y1}{y2} Cluster"]=d

            DF\_Cuspidao2=pd.concat([df\_produ\_kmeans,df\_produ\_kmeans\_REST\_NO]).reset\_index(drop=True)

            DF\_Cuspidao\_BAU=pd.concat([DF\_Cuspidao2,DF\_Cuspidao\_BAU]).reset\_index(drop=True)

        else:

            DF\_Cuspidao2=pd.concat([df\_produ\_kmeans,df\_produ\_kmeans\_REST\_NO]).reset\_index(drop=True)

            DF\_Cuspidao\_BAU=pd.concat([DF\_Cuspidao2,DF\_Cuspidao\_BAU]).reset\_index(drop=True)

    df\_provisorio=DF\_Cuspidao\_BAU

    DF\_Cuspidao2=pd.DataFrame()

    DF\_Cuspidao\_BAU=pd.DataFrame()

DF\_Cuspidao\_BAU=df\_provisorio

DF\_Cuspidao=pd.DataFrame()

df\_provisorio=df\_produ

df\_provisorio=df\_provisorio[['Nome Empresa',

                             'Ticker',

                             'SETOR\_ATIV',

                             'Flag1Company',

                             'DT\_REFER',

                             'Ano',

                             'Mes',

                             1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

#REJEITADOS KMEANS

for y in range(1,13):

    if y==1:

        y1=1

        y2=2

    else:

        y1=y+1

        y2=y+2

    #filtra o setor

    df\_produ\_kmeans=df\_provisorio[(df\_provisorio['Flag1Company']==1)].reset\_index(drop=True)

    Tcks\_count=df\_produ\_kmeans.drop\_duplicates(subset=['Ticker','DT\_REFER'])

    Tcks\_count=Tcks\_count['DT\_REFER'].value\_counts()

    Tcks\_count\_mean=Tcks\_count.mean()

    Tcks\_count=pd.DataFrame(Tcks\_count)

    ListOFTRI=Tcks\_count[Tcks\_count['DT\_REFER']>=Tcks\_count\_mean]

    ListOFTRI['DATE']=ListOFTRI.index

    AnoMAX=ListOFTRI['DATE'].apply(str).str[0:4].max()

    MesMAX=ListOFTRI['DATE'].apply(str).str[5:7].max()

    #Filtrar ano e mês (ultimo tri de cada setor)

    df\_produ\_kmeans=df\_produ\_kmeans[(df\_produ\_kmeans["Ano"]==AnoMAX)&(df\_produ\_kmeans["Mes"]==MesMAX)].reset\_index(drop=True)

    #Clusteriza as empresas com valores negativos como 0 (ruim)

    df\_produ\_kmeans\_REST\_NO=df\_produ\_kmeans[(df\_produ\_kmeans[y1]==0)|(df\_produ\_kmeans[y2]==0)].reset\_index(drop=True)

    df\_produ\_kmeans\_REST\_NO[f"{y1}{y2} Cluster"]=0

    #Separa as empresas que são elegíveis a escalabilidade e a clusterização

    df\_produ\_kmeans=df\_produ\_kmeans[(df\_produ\_kmeans[y1]!=0)&(df\_produ\_kmeans[y2]!=0)].reset\_index(drop=True)

    df\_produ\_kmeans=df\_produ\_kmeans.sort\_values(by=[y1,y2]).reset\_index(drop=True)

    #se tiver mais de 1 amostra (empresa)

    if len(df\_produ\_kmeans)>=2:

        #FAÇO A ESCALABILIDADE COM STANDARD

        scaler = StandardScaler()

        pca = PCA(n\_components = 2)

        features= df\_produ\_kmeans[[y1,y2]]

        scaled\_df = scaler.fit\_transform(features)

        normalized\_df = normalize(scaled\_df)

        normalized\_df = pd.DataFrame(normalized\_df)

        #E DEPOIS REDUZO AS DIMENSÕES E CORRELAÇÕES COM O PCA

        X\_principal = pca.fit\_transform(normalized\_df)

        features = pd.DataFrame(X\_principal)

        features.columns = ['P1', 'P2']

        c=3

        if len(features)<3:

            c=len(features)

        #DOU UM FIT NO KMEANS

        kmeans = KMeans(n\_clusters=c, init='random',

                    n\_init=10,

                    max\_iter=300,random\_state=0)

        kmeans.fit(features)

        #DPS EU FAÇO A CLUSTERIZAÇÃO

        d = KMeans(n\_clusters=c, init='random',

                    n\_init=10,

                    max\_iter=300,random\_state=0).fit\_predict(features)

        df\_produ\_kmeans[f"{y1}{y2} Cluster"]=d

        DF\_Cuspidao2=pd.concat([df\_produ\_kmeans,df\_produ\_kmeans\_REST\_NO]).reset\_index(drop=True)

        DF\_Cuspidao\_RESTO=pd.concat([DF\_Cuspidao2,DF\_Cuspidao\_RESTO]).reset\_index(drop=True)

    else:

        DF\_Cuspidao2=pd.concat([df\_produ\_kmeans,df\_produ\_kmeans\_REST\_NO]).reset\_index(drop=True)

        DF\_Cuspidao\_RESTO=pd.concat([DF\_Cuspidao2,DF\_Cuspidao\_RESTO]).reset\_index(drop=True)

    df\_provisorio=DF\_Cuspidao\_RESTO

    DF\_Cuspidao2=pd.DataFrame()

    DF\_Cuspidao\_RESTO=pd.DataFrame()

DF\_kmeans=pd.concat([DF\_Cuspidao\_BAU,df\_provisorio]).reset\_index(drop=True)

DF\_kmeans["Dados"]="Realizado"

DF\_kmeans=DF\_kmeans[['Nome Empresa','Ticker','SETOR\_ATIV','DT\_REFER',"Dados",

             1,2,3,4,5,6,7,8,9,10,11,12,13,14,

            '12 Cluster',

            '34 Cluster',

            '56 Cluster',

            '78 Cluster',

            '910 Cluster',

            '1112 Cluster',

            '1314 Cluster',

             ]]

Como pode ser observado na penúltima linha, temos um dataframe “K\_means” com as clusterizações, após isso a vamos para a penúltima etapa que é fazer as predições dos próximos semestres.

3.1 Poli

Como mencionando anteriormente as predições dos indicadores não é o core do nosso projeto, apesar de ser interessante para enxergamos qual seria a expectativa dos indicadores dos próximos trimestres e estipular alguns indicadores relativos, e até mesmo clusterizar as predições para entender se nos próximos momentos há a possibilidade de reversão de recomendação baseada no histórico da empresa. Outro ponto importante é que diferente das clusterizações do kmeans, a regressão polimonial é feita por negócio e não por setor, e se baseia no histórico da empresa e não no último trimestre.

Bom, o código é bem simples e vai em linha com o que já conversamos anteriormente. O workflow é o seguinte, primeiro renomeamos as colunas como número para conseguir analisar os indicadores com o for, faço uma lista de tickers (código da empresa), filtro a empresa no dataset, extraímos as informações da empresa, crio datas futuras baseado no último trimestre daquela empresa e faço a predição por indicador. Segue o código:

#Separo os indicadores que eu quero prever

df\_produ[11]=df\_produ['Obrigações Fiscais']

df\_produ[10]=df\_produ['Obrigações Sociais e Trabalhistas']

df\_produ[9]=df\_produ['Contas a Receber']

df\_produ[8]=df\_produ['Ativo Circulante']

df\_produ[7]=df\_produ['Passivo Circulante']

df\_produ[6]=df\_produ['Passivo Total']

df\_produ[5]=df\_produ['Ativo Total']

df\_produ[4]=df\_produ['Reservas de Lucro']

df\_produ[3]=df\_produ['Patrimônio Líquido']

df\_produ[2]=df\_produ['Receita']

df\_produ[1]=df\_produ['Lucro/Prejuízo do Período']

df\_provisorio=df\_produ

df\_provisorio=df\_provisorio[['Nome Empresa',

                             'Ticker',

                             'SETOR\_ATIV',

                             'Flag1Company',

                             'DT\_REFER',

                             'Ano',

                             'Mes',

                             1,2,3,4,5,6,7,8,9,10,11]]

#pego a lista de empresas para analisar

LitsofTickers=df\_provisorio.drop\_duplicates(subset='Ticker')

LitsofTickers=LitsofTickers['Ticker'].values

for x in LitsofTickers:

    DF\_Poly=df\_provisorio[df\_provisorio['Ticker']==x].reset\_index(drop=True)

    DF\_Poly.sort\_values(by=['DT\_REFER']).reset\_index(drop=True)

    #transformo os indexes em variáveis e a data em datetime

    DF\_Poly["indexes"]=DF\_Poly.index

    DF\_Poly['DT\_REFER']=pd.to\_datetime(DF\_Poly['DT\_REFER'])

    #pego a ultima linha dessa empresa para pegar as informações principais para criar um novo dataset

    Lastmonth=DF\_Poly.iloc[-1]['DT\_REFER']

    Lastname=DF\_Poly.iloc[-1]['Nome Empresa']

    Lastsector=DF\_Poly.iloc[-1]['SETOR\_ATIV']

    Lastnamesub=DF\_Poly.iloc[-1]['Ticker']

    Lastindex=DF\_Poly.iloc[-1]['indexes']

    #Crio as datas que eu quero ver la na frente,baseado na data do ultimo balanço

    Lastmonth1=Lastmonth+ relativedelta(months=+3)

    Lastmonth2=Lastmonth+ relativedelta(months=+6)

    Lastmonth3=Lastmonth+ relativedelta(months=+9)

    Lastmonth4=Lastmonth+ relativedelta(months=+12)

    #Crio um dataframe novo com essas datas

    Dateto=pd.DataFrame({'Data\_REF': [Lastmonth1,Lastmonth2,Lastmonth3,Lastmonth4]})

    X\_seq=pd.DataFrame({'x': [Lastindex+1,Lastindex+2,Lastindex+3,Lastindex+4]})

    #Configuro o modelo polimonial

    polyreg=make\_pipeline(PolynomialFeatures(3),LinearRegression())

    Pred=Dateto

    Pred["Nome Empresa"]=Lastname

    Pred['Ticker']=Lastnamesub

    Pred['SETOR\_ATIV']=Lastsector

    #Leio o historico de todos os indicadores e faço a previsão

    for y in range(1,12):

        yy=DF\_Poly[['indexes']]

        xx=DF\_Poly[[y]]

        polyreg.fit(yy,xx)

        Pred2=polyreg.predict(X\_seq)

        Pred2=pd.DataFrame(Pred2,columns=[y])

        Pred[y]=Pred2

    DF\_Polyline=pd.concat([Pred,DF\_Polyline]).reset\_index(drop=True)

DF\_Polyline['Ano']=DF\_Polyline['Data\_REF'].apply(str).str[0:4]

DF\_Polyline['Mes']=DF\_Polyline['Data\_REF'].apply(str).str[5:7]

DF\_Polyline['Dados']="Previsão"

3.2 Pós modelagem

Já fizemos dois tipos de modelagem, uma inicial onde limpamos o dataset e organizamos de um jeito legível e com as nossas regras de negócio, um pré-tratamento onde fizemos algumas alterações básicas nos dados para que pudesse ser utilizado nos modelos, e por fim temos que fazer a última modelagem onde deixamos as informações pós aplicação da IA legível para um front end.

A pós modelagem é essencial para conseguir consumir os dados que geramos através da IA. Pense que está o dataframe “DF\_Polyline” e “K\_means” são outputs que estão bagunçados e que precisam de um pequeno tratamento para serem consumidos, e para isso construímos uma “Base Única” onde juntamos tanto informações de clusterização quanto de regressão. Criar uma base única é uma forma de otimizar e unificar datasets que tem muitas informações em comum.

Nessa última etapa fazemos o seguinte, a gente cria a nota da empresa baseado nas clusterizações dos indicadores, criamos algumas colunas como “o melhor indicador” e “Tipo de Análise”, renomeio colunas e gero o arquivo final. Olha como ficou o código:

DF\_kmeans=DF\_kmeans.fillna(0)

#Separo as colunas principais do dataset Kmeans e dou um replace nas clusterizações para criar uma nota

DF\_kmeans\_2=DF\_kmeans.loc[:,('Ticker','DT\_REFER',"Dados",'12 Cluster',

            '34 Cluster',

            '56 Cluster',

            '78 Cluster',

            '910 Cluster',

            '1112 Cluster',

            '1314 Cluster')].replace([0,1,2],[0,10,5])

#Crio a nota da empresa baseada nos clusters (0 é ruim, 1 é bom e 2 é mediano)

DF\_kmeans\_2['Nota da Empresa']=DF\_kmeans\_2['12 Cluster']+DF\_kmeans\_2['34 Cluster']+DF\_kmeans\_2['56 Cluster']+DF\_kmeans\_2['78 Cluster']+DF\_kmeans\_2['910 Cluster']+DF\_kmeans\_2['1112 Cluster']+DF\_kmeans\_2['1314 Cluster']

DF\_kmeans\_2['Nota da Empresa']=DF\_kmeans\_2['Nota da Empresa']/7

DF\_kmeans\_2['Desempenho do Trimestre']=np.where(DF\_kmeans\_2['Nota da Empresa']<5,"Ruim",

                                        (np.where((DF\_kmeans\_2['Nota da Empresa']>=5) & (DF\_kmeans\_2['Nota da Empresa']<7) ,"Médio","Bom")))

#Pego o indicador com a melhor nota de clusterização (indicador denominador)

DF\_kmeans\_2['Melhor Indicador1']=(DF\_kmeans\_2['12 Cluster']+DF\_kmeans\_2['34 Cluster']+DF\_kmeans\_2['56 Cluster'])/3

DF\_kmeans\_2['Melhor Indicador2']=(DF\_kmeans\_2['34 Cluster']+DF\_kmeans\_2['78 Cluster']+DF\_kmeans\_2['910 Cluster']+DF\_kmeans\_2['1112 Cluster']+DF\_kmeans\_2['1314 Cluster'])/5

DF\_kmeans\_2['Melhor Indicador']=np.where(DF\_kmeans\_2['Melhor Indicador1']>DF\_kmeans\_2['Melhor Indicador2'],"Receita",

                                        (np.where(DF\_kmeans\_2['Melhor Indicador1']<DF\_kmeans\_2['Melhor Indicador2'],"Patrimônio Líquido","Nenhum")))

df\_produ=df1

#Pego as colunas antigas sem tratamento de clips

DF\_kmeans\_2 = pd.merge(df\_produ,DF\_kmeans\_2 ,left\_on=['Ticker','DT\_REFER'], right\_on=['Ticker','DT\_REFER'])

DF\_kmeans\_2['Analise']=DF\_kmeans\_2['Flag1Company'].replace([0,1],["Analise por Setor",'Analise com outras empresas'])

DF\_kmeans\_2['Tipo de Analise']='Kmeans'

#Deixo tudo limpinho e arrumado no dataset kmeans

DF\_kmeans\_2=DF\_kmeans\_2[['Nome Empresa','Ticker','SETOR\_ATIV','Dados','Analise','Tipo de Analise','Melhor Indicador','Desempenho do Trimestre','DT\_REFER','Lucro/Prejuízo do Período','Ativo Total','Ativo Circulante','Contas a Receber','Passivo Total','Passivo Circulante','Obrigações Sociais e Trabalhistas','Obrigações Fiscais','Reservas de Lucro','Patrimônio Líquido','Receita']]

#Dou nome aos bois no dataset do polimonial

DF\_Polyline\_2=DF\_Polyline.rename({'Data\_REF': 'DT\_REFER',

                                  1: 'Lucro/Prejuízo do Período',

                                  2: 'Receita',

                                  3: 'Patrimônio Líquido',

                                  4: 'Reservas de Lucro',

                                  5: 'Ativo Total',

                                  6: 'Passivo Total',

                                  7: 'Passivo Circulante',

                                  8: 'Ativo Circulante',

                                  9: 'Contas a Receber',

                                  10: 'Obrigações Sociais e Trabalhistas',

                                  11: 'Obrigações Fiscais',

                                 }, axis=1)

#Seto as mesmas colunas do kmeans

DF\_Polyline\_2['Melhor Indicador']='Não Analisado'

DF\_Polyline\_2['Desempenho do Trimestre']='Não Analisado'

DF\_Polyline\_2['Nota da Empresa']='Não Analisado'

DF\_Polyline\_2['Analise']='Analise de Historico Financeiro'

DF\_Polyline\_2['Tipo de Analise']='Regressão Polimonial'

#Deixo tudo limpinho e arrumado no dataset polimonial

DF\_Polyline\_2=DF\_Polyline\_2[['Nome Empresa','Ticker','SETOR\_ATIV','Dados','Analise','Tipo de Analise','Melhor Indicador','Desempenho do Trimestre','DT\_REFER','Lucro/Prejuízo do Período','Ativo Total','Ativo Circulante','Contas a Receber','Passivo Total','Passivo Circulante','Obrigações Sociais e Trabalhistas','Obrigações Fiscais','Reservas de Lucro','Patrimônio Líquido','Receita']]

#Deipois junto tudo e cuspo

Base\_Unica=pd.concat([DF\_Polyline\_2,DF\_kmeans\_2]).reset\_index(drop=True)

Base\_Unica.to\_csv(rf"Cuspidão\Base\_Unica.csv",decimal=',',sep=';', encoding='latin-1',index=False)

Por fim temos um output no qual podemos consumir os dados que geramos com a IA. E ele ficou desse jeito:

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente com confiança média

Uma imagem contendo Calendário

Descrição gerada automaticamente

1. **Front End MVP**

Quando falamos de ciência de dados, geralmente muitas pessoas ver necessidade de criar gráficos ao longo do código para analisar os dados. No caso de finanças a leitura de gráficos durante o processo se faz desnecessário visto que não há uma granularidade a ser analisada principalmente na fase de clusters. Mas precisamos consumir os dados que geramos até mesmo para entender se faz sentido o nosso trabalho e nosso wireframe que está sendo desenvolvido.

E como construir um front end neste momento faria com que nos perdesse muito tempo, optamos por ler os dados de finanças através do Excel, onde criamos um front end MVP (minimum viable product). E ficou dessa forma:



Temos mais de 288 análises de empresas que fizemos, e a gente consegue visualizar isso no front end do excel.

4.1 Análise de resultados

Eu diria que nosso modelo de clusterização consegue apontar novas oportunidades do mercado, visto que analisamos diversas empresas, muitas que a gente nunca ouviu falar, porém existe alguns gaps que vamos resolver ao longo do ano, como por exemplo analisar o desempenho das empresas desde sua criação. Mas por exemplo em uma análise dos indicadores, as clusterizações fazem sentido em boa parte dos casos.