Fluxo de caixa descontado

Projeto: Abordagens de aprendizado de máquina aplicado ao reconhecimento de padrões não-lineares: Classificação do desempenho de pneus em processos de manufatura inteligente.

Professores: Ricardo Kalid; Ricardo Uchoa

Aluno: Rodrigo Marcel Araujo Oliveira

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from statistics import *
import warnings
import matplotlib.pyplot as plt
import numpy_financial as npf
```

https://numpy.org/numpy-financial/latest/ (https://numpy.org/numpy-financial/latest/)

Abordagem probabilística da avaliação do fluxo de caixa

Leitura dos dados

In [ ]:

```
pd.set_option('display.max_columns', None)
df_vpl = pd.read_excel('FCD.xlsx',sheet_name=9)
df_vpl.head()
```

Out[ ]:

	Unnamed: 0	Tabela FCD: fluxo de caixa descontado	Unnamed: 2	10	ano(s) que deve ser menor ou igual a 4	Unnamed: 5	Unnamed: 6	Un
0	NaN	Siglas e equações:	RV	RB = RV	IPF	IPE	IPM	IF I
1	NaN	Período	Receita com vendas	Receita bruta ou receita total	Impostos proporcionais federais em relação a r...	Impostos proporcionais estaduais em relação a ...	Impostos proporcionais municipais em relação a...	pro (IS IPI
2	NaN	0	0	0	0	0	0	
3	NaN	1	2400000	2400000	192000	60000	0	
4	NaN	2	4800000	4800000	384000	120000	0	

Tratamento dos dados e seleção das variáveis

In [ ]:

```
pd.set_option('display.max_columns', None)
c = df_vpl[1:2]
c = pd.DataFrame(c)
c = c.transpose().reset_index()
c = c[1].to_list()

df_vpl2 = df_vpl.iloc[2:13]
df_vpl2.columns = c
df_vpl2 = df_vpl2.reset_index()
df_vpl2 = df_vpl2.iloc[:, 2:40]
df_vpl2 = df_vpl2.fillna(0)
df_vpl2
```

Out[ ]:

	Período	Receita com vendas	Receita bruta ou receita total	Impostos proporcionais federais em relação a receita bruta	Impostos proporcionais estaduais em relação a receita bruta	Impostos proporcionais municipais em relação a receita bruta	Impostos proporcionais (ISS, ICMS, IPI, PIS, CONFINS etc.) total
0	0	0	0	0	0	0	0
1	1	2400000	2400000	192000	60000	0	252000
2	2	4800000	4800000	384000	120000	0	504000
3	3	7200000	7200000	576000	180000	0	756000
4	4	9600000	9600000	768000	240000	0	1008000
5	5	12000000	12000000	960000	300000	0	1260000
6	6	14400000	14400000	1152000	360000	0	1512000
7	7	16800000	16800000	1344000	420000	0	1764000
8	8	19200000	19200000	1536000	480000	0	2016000
9	9	21600000	21600000	1728000	540000	0	2268000
10	10	24000000	24000000	1920000	600000	0	2520000

Premissas do fluxo de caixa

In [ ]:

```
Inflacao = 0.0
Quantidade_de_períodos_cobertos_pelo_capital_de_giro = 1.0
Impostos_proporcionais_federais = 8.0
Impostos_proporcionais_estaduais =      2.5
Impostos_proporcionais_municipais = 0.0
Investimento_maximo_em_PDI = 10.0
Investimento_maximo_em_marketing =5.0
Outras_despesas_por_exemplo_causas_trabalhistas_multas = 5.0
Taxa_de_imposto_de_renda_sobre_o_LAIR = 10.0
Contribuicao_social_sobre_o_LAIR = 7.5
Outras_taxas_ou_impostos_ou_tributos_sobre_o_LAIR =     5.0
Tempo_de_retorno_minimo = 3.0
Emprestimo_para_capital_de_giro =200000.00
Emprestimo_para_investimento_em_bens_de_capital = 100000.00
Quantidade_de_periodos_para_quitar_o_emprestimo = 5
Periodo_de_carencia_para_pagamento_do_juros_do_emprestimo = 2
Periodo_de_carencia_para_pagamento_do_principal_do_emprestimo = 2
Taxa_de_juros_inclue_a_inflação = 6.0
```

Calculos do fluxo de caixa

In [ ]:

```
def impostos_periodo(lair, taxa, periodo):
    if periodo>0:
        return lair*taxa
    else:
        return 0
```

In [ ]:

```
df_vpl2['Receita bruta ou receita total'] = df_vpl2['Receita com vendas']
df_vpl2['Impostos proporcionais federais em relação a receita bruta'] = df_vpl2['Receit
a com vendas']*Impostos_proporcionais_federais/100
df_vpl2['Impostos proporcionais estaduais em relação a receita bruta'] = df_vpl2['Recei
ta com vendas']*Impostos_proporcionais_estaduais/100
df_vpl2['Impostos proporcionais municipais em relação a receita bruta'] = df_vpl2['Rece
ita com vendas']* Impostos_proporcionais_municipais/100
df_vpl2['Impostos proporcionais (ISS, ICMS, IPI, PIS, CONFINS etc.) total'] = df_vpl2[
'Impostos proporcionais federais em relação a receita bruta']  + df_vpl2['Impostos prop
orcionais estaduais em relação a receita bruta'] + df_vpl2['Impostos proporcionais muni
cipais em relação a receita bruta']
df_vpl2['Receita líquida'] = df_vpl2['Receita bruta ou receita total'] -  df_vpl2['Impo
stos proporcionais (ISS, ICMS, IPI, PIS, CONFINS etc.) total']
df_vpl2['Investimento em PD&I'] = df_vpl2['Receita líquida']*0.0035
df_vpl2['Investimento em marketing'] =df_vpl2['Receita líquida']*0.003
df_vpl2['Investimento em bens exauríveis'] =df_vpl2['Receita líquida']*0.001
df_vpl2['Investimento total'] = df_vpl2['Investimento em PD&I']  + df_vpl2['Investiment
o em marketing'] + df_vpl2['Investimento em bens exauríveis'] + df_vpl2['Investimento e
m bens intangíveis'] + df_vpl2['Investimento em bens de capital']
df_vpl2['Custos e investimentos'] = df_vpl2['Investimento total'] + df_vpl2['Custos ope
racionais']
df_vpl2['Lucro operacional ou LAJIRDA ou EBITDA'] = df_vpl2['Receita líquida'] - df_vpl
2['Custos e investimentos']
df_vpl2['Lucro operacional sem investimentos'] = df_vpl2['Lucro operacional ou LAJIRDA
 ou EBITDA'] + df_vpl2['Investimento total']

df_vpl2['Lucro antes do juros, imposto de renda e das contribuições sociais (LAJIR)'] =
df_vpl2['Lucro operacional ou LAJIRDA ou EBITDA'] - df_vpl2['Somatório da amortização
 (principal) de empréstimos e outros dedutíveis do IR'] - df_vpl2['Depreciações (somató
rio)']

df_vpl2['LAJIR sem investimentos'] = df_vpl2['Lucro antes do juros, imposto de renda e
 das contribuições sociais (LAJIR)'] + df_vpl2['Investimento total']
df_vpl2['Lucro antes do imposto de renda e das contribuições sociais (LAIR)'] = df_vpl2
['Lucro antes do juros, imposto de renda e das contribuições sociais (LAJIR)']+df_vpl2[
'Receitas Não Operacionais']-df_vpl2['Juros de empréstimos pagos a credores']
df_vpl2['LAIR sem investimentos'] = df_vpl2['LAJIR sem investimentos']-df_vpl2['Juros d
e empréstimos pagos a credores']

df_vpl2['Imposto de renda,  % sobre o LAIR'] = df_vpl2.apply(lambda row: impostos_perio
do(row['Lucro antes do imposto de renda e das contribuições sociais (LAIR)'], Taxa_de_i
mposto_de_renda_sobre_o_LAIR/100, row['Período']), axis=1)
df_vpl2['Contribuições sociais, % sobre o LAIR'] = df_vpl2.apply(lambda row: impostos_p
eriodo(row['Lucro antes do imposto de renda e das contribuições sociais (LAIR)'], Contr
ibuicao_social_sobre_o_LAIR/100, row['Período']), axis=1)
df_vpl2['Outras taxas ou impostos ou tributos sobre o LAIR'] = df_vpl2.apply(lambda row
: impostos_periodo(row['Lucro antes do imposto de renda e das contribuições sociais (LA
IR)'], Outras_taxas_ou_impostos_ou_tributos_sobre_o_LAIR/100, row['Período']), axis=1)

df_vpl2['Total de impostos e taxas sobre o LAIR'] = df_vpl2['Imposto de renda,  % sobre
o LAIR'] + df_vpl2['Contribuições sociais, % sobre o LAIR'] + df_vpl2['Outras taxas ou
 impostos ou tributos sobre o LAIR']
df_vpl2['Lucro líquido'] = df_vpl2['Lucro antes do imposto de renda e das contribuições
sociais (LAIR)']-df_vpl2['Total de impostos e taxas sobre o LAIR']
```

In [ ]:

```
pd.set_option('display.max_columns', None)
df_vpl2
```

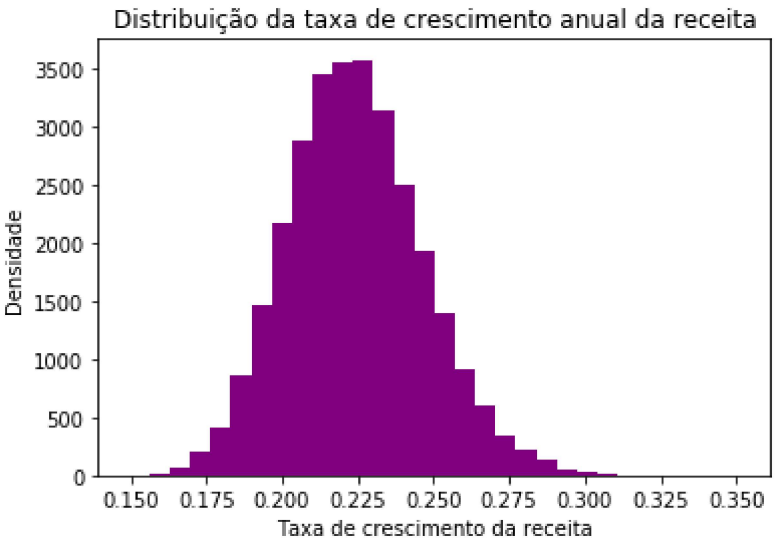
Out[ ]:

	Período	Receita com vendas	Receita bruta ou receita total	Impostos proporcionais federais em relação a receita bruta	Impostos proporcionais estaduais em relação a receita bruta	Impostos proporcionais municipais em relação a receita bruta	Impostos proporcionais (ISS, ICMS, IPI, PIS, CONFINS etc.) total
0	0	0	0	0.0	0.0	0.0	0.0
1	1	2400000	2400000	192000.0	60000.0	0.0	252000.0
2	2	4800000	4800000	384000.0	120000.0	0.0	504000.0
3	3	7200000	7200000	576000.0	180000.0	0.0	756000.0
4	4	9600000	9600000	768000.0	240000.0	0.0	1008000.0
5	5	12000000	12000000	960000.0	300000.0	0.0	1260000.0
6	6	14400000	14400000	1152000.0	360000.0	0.0	1512000.0
7	7	16800000	16800000	1344000.0	420000.0	0.0	1764000.0
8	8	19200000	19200000	1536000.0	480000.0	0.0	2016000.0
9	9	21600000	21600000	1728000.0	540000.0	0.0	2268000.0
10	10	24000000	24000000	1920000.0	600000.0	0.0	2520000.0

Distribuição da taxa de crescimento da receita

In [ ]:

```
n_interacoes = 30000
taxa_cresc = np.random.lognormal(mean = -3/2, sigma=1/10, size=n_interacoes)# np.rando
m.normal(loc=0.1, scale=0.01, size=n_interacoes)
plt.hist(taxa_cresc, bins=30,  color='purple')
plt.xlabel('Taxa de crescimento da receita')
plt.ylabel('Densidade')
plt.title('Distribuição da taxa de crescimento anual da receita')
plt.show()
```



In [ ]:

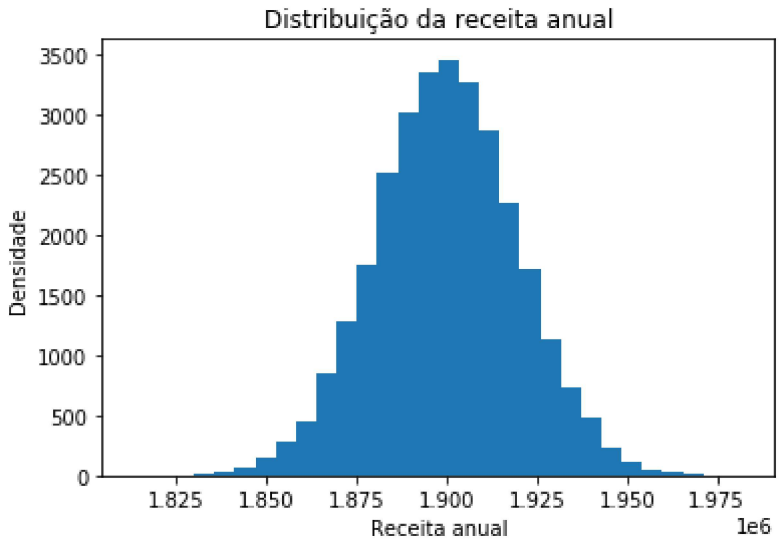
```
print("devio padrão: ", stdev(taxa_cresc))
print("média: ", mean(taxa_cresc))
print("mediana: ", median(taxa_cresc))
```

```
devio padrão:  0.022417271354659093
média:  0.2241928651799217
mediana:  0.2231732625019075
```

Distribuição da receita

In [ ]:

```
shape, scale = 9500, 200  # mean=4, std=2*sqrt(2)
receita_dist = np.random.gamma(shape, scale, n_interacoes)
plt.hist(receita_dist, bins=30)
plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
plt.xlabel('Receita anual')
plt.ylabel('Densidade')
plt.title('Distribuição da receita anual')
plt.show()
```



In [ ]:

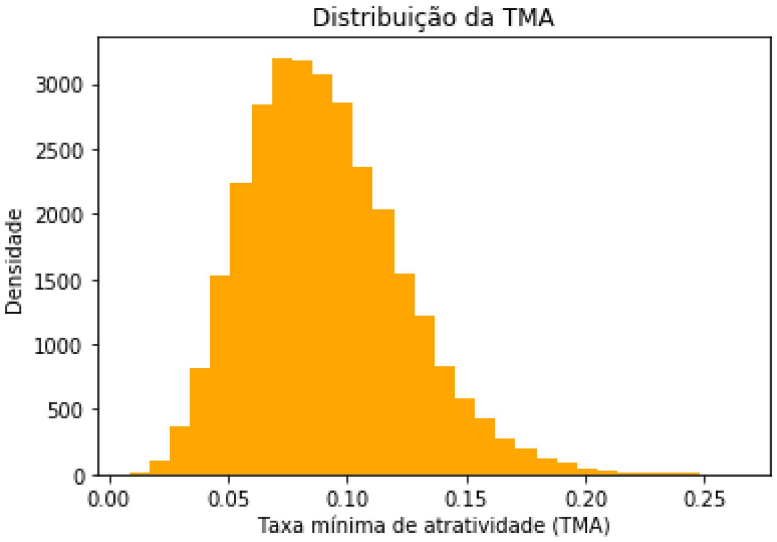
```
print("devio padrão: ", stdev(receita_dist))
print("média: ", mean(receita_dist))
print("mediana: ", median(receita_dist))
```

```
devio padrão:  19577.20960662543
média:  1900021.8502075141
mediana:  1900022.2389936452
```

Distribuição da TMA

In [ ]:

```
tma_dist = np.random.beta(7, 70, size=n_interacoes)
plt.hist(tma_dist, bins=30, color='orange')
plt.xlabel('Taxa mínima de atratividade (TMA)')
plt.ylabel('Densidade')
plt.title('Distribuição da TMA')
plt.show()
```



In [ ]:

```
print("devio padrão: ", stdev(tma_dist))
print("média: ", mean(tma_dist))
print("mediana: ", median(tma_dist))
```

```
devio padrão:  0.032493503663874546
média:  0.09077786842861049
mediana:  0.08741939235439455
```

In [ ]:

```
receita_inicial = mean(receita_dist)
taxa_cresc_receita = mean(taxa_cresc)
tma = mean(tma_dist)
```

Receita

In [ ]:

```
warnings.filterwarnings('ignore')

df_vpl2['Receita com vendas'].iloc[1] = receita_inicial

for periodo in range(2, 11):
    df_vpl2['Receita com vendas'].iloc[periodo] = (df_vpl2['Receita com vendas'].iloc[p
eriodo - 1]) * (1 + taxa_cresc_receita)

df_vpl2
```

Out[ ]:

	Período	Receita com vendas	Receita bruta ou receita total	Impostos proporcionais federais em relação a receita bruta	Impostos proporcionais estaduais em relação a receita bruta	Impostos proporcionais municipais em relação a receita bruta	Impos proporcio (ISS, IC IPI, CONF etc.)
0	0	0.000000e+00	0	0.0	0.0	0.0	
1	1	1.900022e+06	2400000	192000.0	60000.0	0.0	2520
2	2	2.089957e+06	4800000	384000.0	120000.0	0.0	5040
3	3	2.298879e+06	7200000	576000.0	180000.0	0.0	7560
4	4	2.528685e+06	9600000	768000.0	240000.0	0.0	10080
5	5	2.781464e+06	12000000	960000.0	300000.0	0.0	12600
6	6	3.059512e+06	14400000	1152000.0	360000.0	0.0	15120
7	7	3.365355e+06	16800000	1344000.0	420000.0	0.0	17640
8	8	3.701772e+06	19200000	1536000.0	480000.0	0.0	20160
9	9	4.071818e+06	21600000	1728000.0	540000.0	0.0	22680
10	10	4.478856e+06	24000000	1920000.0	600000.0	0.0	25200

Função automatizada do fluxo de caixa

In [ ]:

```
def impostos_periodo(lair, taxa, periodo):
    if periodo>0:
        return lair*taxa
    else:
        return 0


def VPL_FCD(data, n_interacoes):

    # Calculando fluxo de caixa
    dist_vpl= list()
    dist_tir = list()

    taxa_cresc = np.random.lognormal(mean = -3/2, sigma=1/10, size=n_interacoes) #np.ra
ndom.normal(loc=0.1, scale=0.01, size=n_interacoes)

    shape, scale = 9500, 200  # mean=2*shape, std=2*sqrt(scale)
    receita_dist = np.random.gamma(shape, scale, n_interacoes)

    tma_dist = np.random.beta(7, 70, size=n_interacoes)


    receita_inicial = mean(receita_dist)
    taxa_cresc_receita = mean(taxa_cresc)
    tma = mean(tma_dist)


    for i in range(n_interacoes):

        data['Receita com vendas'].iloc[1] = receita_inicial

        for periodo in range(2, 11):
            data['Receita com vendas'].iloc[periodo] = (data['Receita com vendas'].iloc
[periodo - 1]) * (1 + taxa_cresc[i])

        Impostos_proporcionais_federais = 8.0/100
        Impostos_proporcionais_estaduais =      2.5/100
        Impostos_proporcionais_municipais = 0.0/100
        Taxa_de_imposto_de_renda_sobre_o_LAIR = 10.0/100
        Contribuicao_social_sobre_o_LAIR = 7.5/100
        Outras_taxas_ou_impostos_ou_tributos_sobre_o_LAIR =     5.0/100

        data['Receita bruta ou receita total'] = data['Receita com vendas']
        data['Impostos proporcionais federais em relação a receita bruta'] = data['Rece
ita com vendas']*Impostos_proporcionais_federais
        data['Impostos proporcionais estaduais em relação a receita bruta'] = data['Rec
eita com vendas']*Impostos_proporcionais_estaduais
        data['Impostos proporcionais municipais em relação a receita bruta'] = data['Re
ceita com vendas']* Impostos_proporcionais_municipais
        data['Impostos proporcionais (ISS, ICMS, IPI, PIS, CONFINS etc.) total'] = data
['Impostos proporcionais federais em relação a receita bruta']  + data['Impostos propor
cionais estaduais em relação a receita bruta'] + data['Impostos proporcionais municipai
s em relação a receita bruta']
        data['Receita líquida'] = data['Receita bruta ou receita total'] -  data['Impos
tos proporcionais (ISS, ICMS, IPI, PIS, CONFINS etc.) total']
        data['Investimento em PD&I'] = data['Receita líquida']*0.0035
        data['Investimento em marketing'] =data['Receita líquida']*0.003
        data['Investimento em bens exauríveis'] =data['Receita líquida']*0.001
        data['Investimento total'] = data['Investimento em PD&I']  + data['Investimento
```

```
em marketing'] + data['Investimento em bens exauríveis'] + data['Investimento em bens i
ntangíveis'] + data['Investimento em bens de capital']
        data['Custos e investimentos'] = data['Investimento total'] + data['Custos oper
acionais']
        data['Lucro operacional ou LAJIRDA ou EBITDA'] = data['Receita líquida'] - data
['Custos e investimentos']
        data['Lucro operacional sem investimentos'] = data['Lucro operacional ou LAJIRD
A ou EBITDA'] + data['Investimento total']

        data['Lucro antes do juros, imposto de renda e das contribuições sociais (LAJI
R)'] = data['Lucro operacional ou LAJIRDA ou EBITDA'] - data['Somatório da amortização
 (principal) de empréstimos e outros dedutíveis do IR'] - data['Depreciações (somatóri
o)']

        data['LAJIR sem investimentos'] = data['Lucro antes do juros, imposto de renda
 e das contribuições sociais (LAJIR)'] + data['Investimento total']
        data['Lucro antes do imposto de renda e das contribuições sociais (LAIR)'] = da
ta['Lucro antes do juros, imposto de renda e das contribuições sociais (LAJIR)']+data[
'Receitas Não Operacionais']-data['Juros de empréstimos pagos a credores']
        data['LAIR sem investimentos'] = data['LAJIR sem investimentos']-data['Juros de
empréstimos pagos a credores']

        data['Imposto de renda,  % sobre o LAIR'] = data.apply(lambda row: impostos_per
iodo(row['Lucro antes do imposto de renda e das contribuições sociais (LAIR)'], Taxa_de
_imposto_de_renda_sobre_o_LAIR, row['Período']), axis=1)
        data['Contribuições sociais, % sobre o LAIR'] = data.apply(lambda row: impostos
_periodo(row['Lucro antes do imposto de renda e das contribuições sociais (LAIR)'], Con
tribuicao_social_sobre_o_LAIR, row['Período']), axis=1)
        data['Outras taxas ou impostos ou tributos sobre o LAIR'] = data.apply(lambda r
ow: impostos_periodo(row['Lucro antes do imposto de renda e das contribuições sociais
 (LAIR)'], Outras_taxas_ou_impostos_ou_tributos_sobre_o_LAIR, row['Período']), axis=1)

        data['Total de impostos e taxas sobre o LAIR'] = data['Imposto de renda,  % sob
re o LAIR'] + data['Contribuições sociais, % sobre o LAIR'] + data['Outras taxas ou imp
ostos ou tributos sobre o LAIR']
        data['Lucro líquido'] = data['Lucro antes do imposto de renda e das contribuiçõ
es sociais (LAIR)']-data['Total de impostos e taxas sobre o LAIR']

        vpl = npf.npv(float(tma), data['Lucro líquido'])
        flx= np.array(list(data['Lucro líquido']))
        tir = npf.irr(flx)

        dist_vpl.append(vpl)
        dist_tir.append(tir)

    return dist_vpl, dist_tir
```

In [ ]:

```
warnings.filterwarnings('ignore')

vpl_dist, tir_dist = VPL_FCD(df_vpl2, n_interacoes = 30000)
```

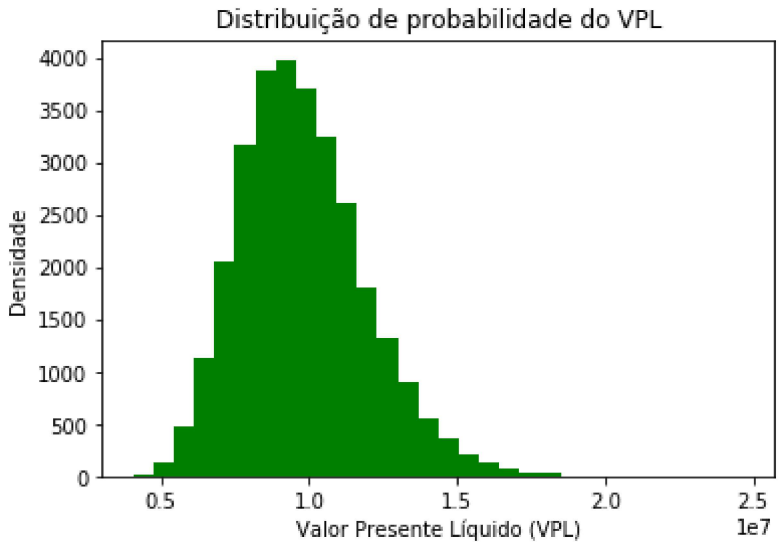
In [ ]:

```
warnings.filterwarnings('ignore')

print("devio padrão: ", stdev(vpl_dist))
print("média: ", mean(vpl_dist))
print("mediana: ", median(vpl_dist))

plt.hist(vpl_dist, bins=30, color='green')
plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
plt.xlabel('Valor Presente Líquido (VPL)')
plt.ylabel('Densidade')
plt.title('Distribuição de probabilidade do VPL')
plt.show()
```

```
devio padrão:  2168423.3948924905
média:  9796681.927745767
mediana:  9567978.8803814
```



In [ ]:

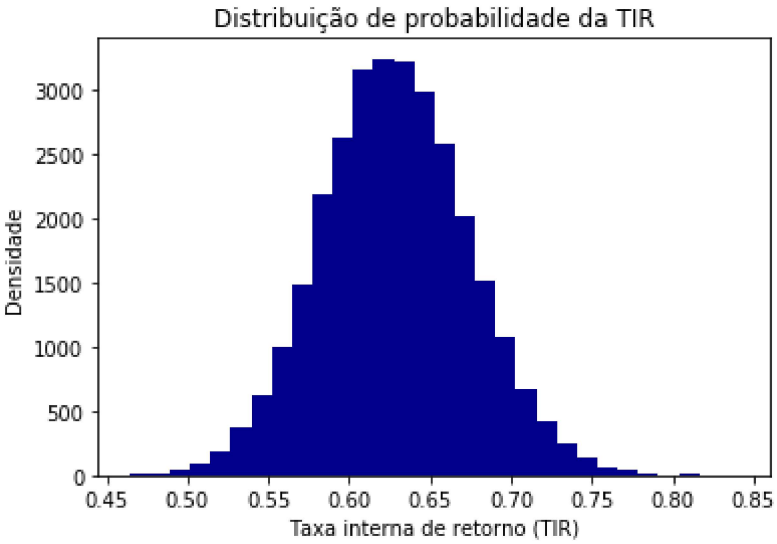
```
warnings.filterwarnings('ignore')

tir_dist_f = [j for j in tir_dist if str(j)!= 'nan']

print("devio padrão: ", stdev(tir_dist_f))
print("média: ", mean(tir_dist_f))
print("mediana: ", median(tir_dist_f))

plt.hist(tir_dist_f, bins=30, color='darkblue')
plt.xlabel('Taxa interna de retorno (TIR)')
plt.ylabel('Densidade')
plt.title('Distribuição de probabilidade da TIR')
plt.show()
```

```
devio padrão:  0.04571020237834249
média:  0.6282429930561269
mediana:  0.6273406349251412
```



In [ ]: