Importar Bibliotecas

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import statsmodels.api as sm
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
from scipy.stats import chi2, chisquare
from statsmodels.graphics.regressionplots import abline_plot
import scipy.stats as stats
from sklearn.metrics import roc_curve, auc
import statsmodels.formula.api as smf
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from scipy.stats import ks_2samp
```

Impoirtar base de dados

In [2]:
```
df = pd.read_csv('loan_approval_dataset.csv')
df.head()
```

Out[2]:

	Applicant_ID	Age	Income	Credit_Score	Loan_Amount	Loan_Term	Interest_Rate	E
0	1	56	21920	639	452748	72	4.53	
1	2	69	126121	655	257134	60	5.38	
2	3	46	96872	467	226437	72	3.46	
3	4	32	101132	751	310480	12	14.00	
4	5	60	22093	404	13070	12	9.13	

Renomear colunas

In [3]:
```
df = df.rename(columns={'Applicant_ID': 'ID_Requerente'})
df = df.rename(columns={'Age': 'Idade'})
df = df.rename(columns={'Income': 'Renda'})
df = df.rename(columns={'Credit_Score': 'Score_de_Credito'})
df = df.rename(columns={'Loan_Amount': 'Valor_Emprestimo'})
df = df.rename(columns={'Loan_Term': 'Prazo_Emprestimo'})
df = df.rename(columns={'Interest_Rate': 'Taxa_Juros'})
df = df.rename(columns={'Employment_Status': 'Status_Emprego'})
df = df.rename(columns={'Debt_to_Income_Ratio': 'Perc_Divida_Renda'})
df = df.rename(columns={'Marital_Status': 'Estado_civil'})
df = df.rename(columns={'Number_of_Dependents': 'Numero_dependentes'})
df = df.rename(columns={'Property_Ownership': 'Propriedade'})
df = df.rename(columns={'Loan_Purpose': 'Finalidade_Emprestimo'})
df = df.rename(columns={'Previous_Defaults': 'Target_Aprovado'})
```

```
df
```

Out[3]:

	ID_Requerente	Idade	Renda	Score_de_Credito	Valor_Emprestimo	Prazo_Empres
0	1	56	21920	639	452748	
1	2	69	126121	655	257134	
2	3	46	96872	467	226437	
3	4	32	101132	751	310480	
4	5	60	22093	404	13070	
...	...	...	...	...	...	
4995	4996	24	169594	755	299944	
4996	4997	66	162728	829	15886	
4997	4998	26	166965	468	477830	
4998	4999	53	36493	442	205981	
4999	5000	36	154704	336	183308	

5000 rows × 14 columns

Sumario de informações

Variável	Descrição da variável
ID_Requerente	Código de indentificação.
Idade	Idade.
Renda	Renda anual.
Score_de_Credito	Potuação de Score de crédito.
Valor_Emprestimo	Valor do emprestimo.
Prazo_Emprestimo	Tempo de duração do emprestimo.
Taxa_Juros	Taxa de juros do emprestimo.
Status_Emprego	Situação atual do emprego.
Perc_Divida_Renda	Divisão do valor da emprestimo / Renda.
Estado_civil	Estado cívil.
Numero_dependentes	Número de pessoas depedentes na casa.
Propriedade	Situação atual da propriedade.
Finalidade_Emprestimo	Finalidade do emprestimo.
Target_Aprovado	Flag de aprovados ou não.

Checar informações das colunas

In [4]:
```
df.info();
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   ID_Requerente          5000 non-null   int64  
 1   Idade                  5000 non-null   int64  
 2   Renda                  5000 non-null   int64  
 3   Score_de_Credito       5000 non-null   int64  
 4   Valor_Emprestimo       5000 non-null   int64  
 5   Prazo_Emprestimo       5000 non-null   int64  
 6   Taxa_Juros             5000 non-null   float64
 7   Status_Emprego         5000 non-null   object 
 8   Perc_Divida_Renda      5000 non-null   float64
 9   Estado_civil           5000 non-null   object 
 10  Numero_dependentes     5000 non-null   int64  
 11  Propriedade            5000 non-null   object 
 12  Finalidade_Emprestimo  5000 non-null   object 
 13  Target_Aprovado        5000 non-null   int64  
dtypes: float64(2), int64(8), object(4)
memory usage: 547.0+ KB
```

Checar valores nulos

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
ID_Requerente           0
Idade                   0
Renda                   0
Score_de_Credito        0
Valor_Emprestimo        0
Prazo_Emprestimo        0
Taxa_Juros              0
Status_Emprego          0
Perc_Divida_Renda       0
Estado_civil            0
Numero_dependentes      0
Propriedade             0
Finalidade_Emprestimo   0
Target_Aprovado         0
dtype: int64
```

Analise de variáveis Quantitativas

Analise distribuição da Target

In [12]:
```
df.describe()
```

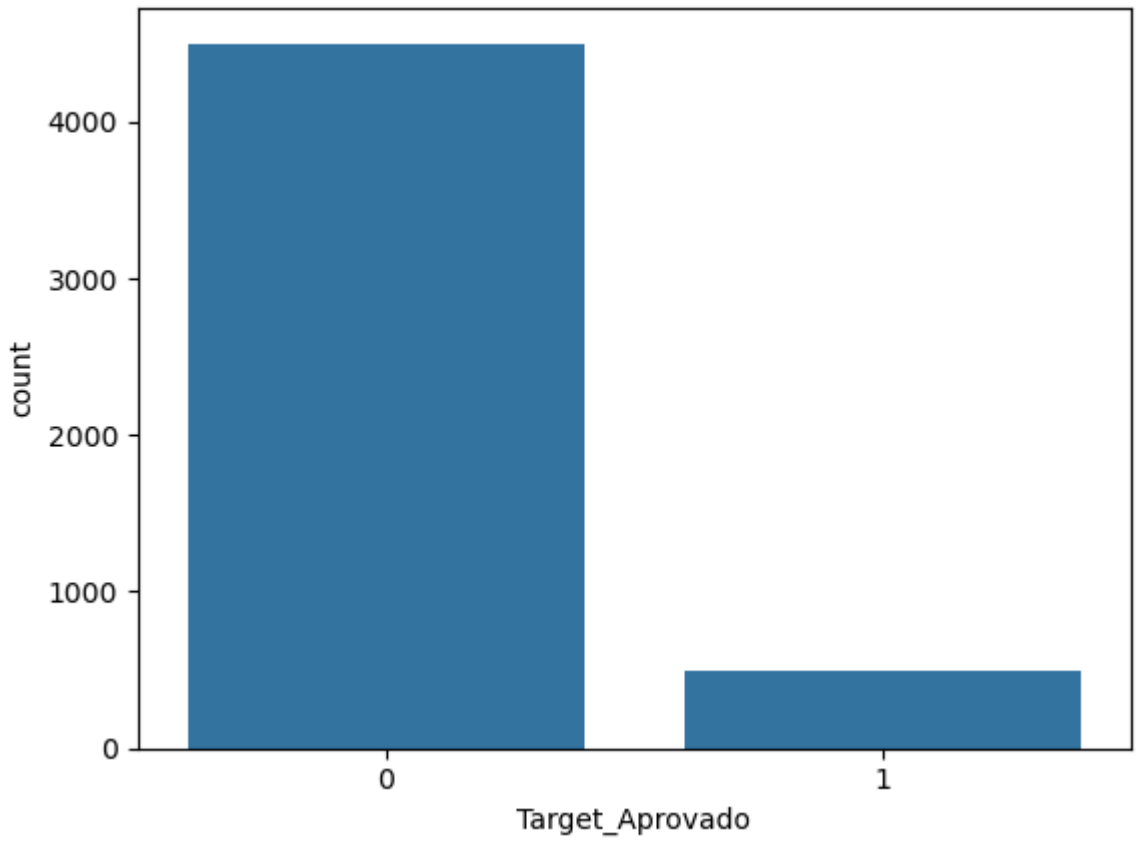
Out[12]:

	ID_Requerente	Idade	Renda	Score_de_Credito	Valor_Emprestimo
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000
mean	2500.500000	43.584600	110220.247000	574.183200	252118.47900
std	1443.520003	14.919094	52075.384023	160.437715	142580.00452
min	1.000000	18.000000	20028.000000	300.000000	5050.00000
25%	1250.750000	31.000000	64751.000000	434.750000	129940.25000
50%	2500.500000	43.000000	110180.500000	573.000000	250846.50000
75%	3750.250000	56.000000	155749.750000	715.000000	378021.75000
max	5000.000000	69.000000	199992.000000	849.000000	499651.00000

In [6]:
```
df["Target_Aprovado"].value_counts(normalize=True) * 100
```

Out[6]:
```
Target_Aprovado
0    90.06
1     9.94
Name: proportion, dtype: float64
```

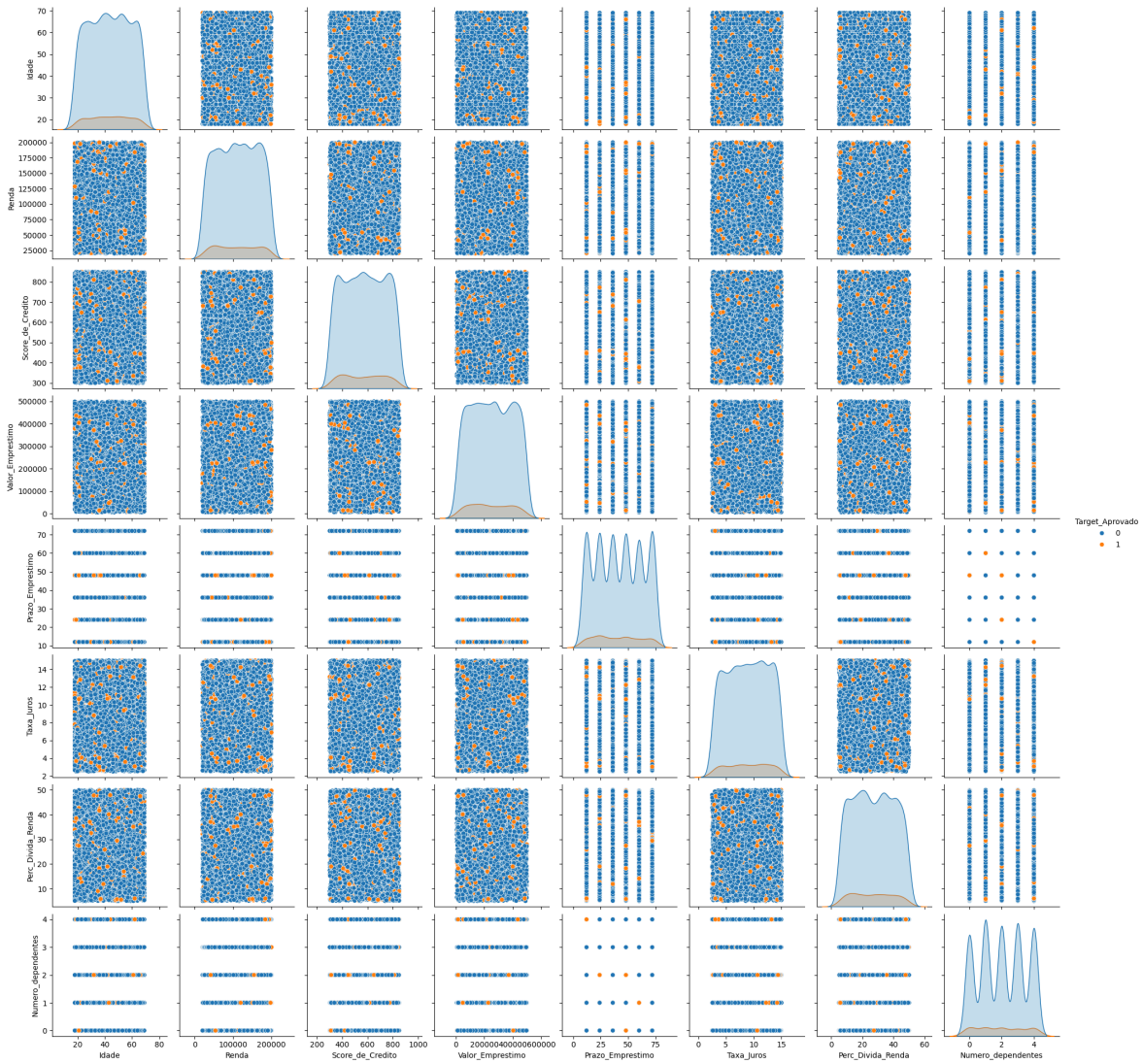
In [7]:
```
sns.countplot(data=df, x='Target_Aprovado');
```



In [8]:
```
df_quant = df[['Idade', 'Renda', 'Score_de_Credito', 'Valor_Emprestimo', 'Prazo_
```

Analise de correlação e distribuição das variáveis quantitativas

In [9]:
```
sns.pairplot(data=df_quant, hue='Target_Aprovado');
```



Analise gráfica da distribuição dos dados quantitativos

In [10]:
```
def plot_violin_all(df):

    for column in df.columns:
        if pd.api.types.is_numeric_dtype(df[column]):
            print(f"Plotando gráfico para a variável: {column}")
            fig = px.violin(df, y=column, box=True, points='all', title=f"Violin
            fig.show()
        else:
            print(f"Variável '{column}' não é numérica. Pulando...")
```

In [11]:
```
plot_violin_all(df_quant)
```

```
Plotando gráfico para a variável: Idade
```

```
Plotando gráfico para a variável: Renda
```

```
Plotando gráfico para a variável: Score_de_Credito
```

```
Plotando gráfico para a variável: Valor_Emprestimo
```

```
Plotando gráfico para a variável: Prazo_Emprestimo
```

```
Plotando gráfico para a variável: Taxa_Juros
```

```
Plotando gráfico para a variável: Perc_Divida_Renda
```

```
Plotando gráfico para a variável: Numero_dependentes
```

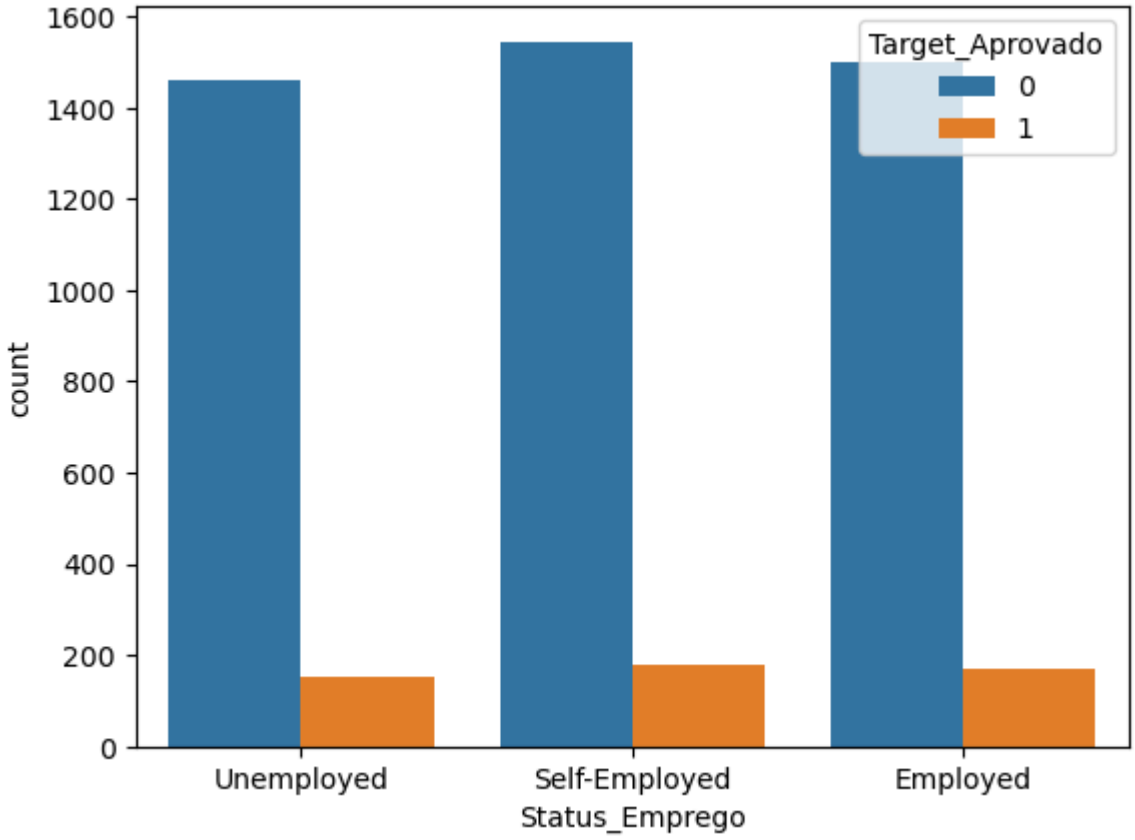
```
Plotando gráfico para a variável: Target_Aprovado
```

- Após analise dos gráficos de violino e distribuição dos pontos, podemos notar que os dados estão muito bem distribuidos e homogêneos

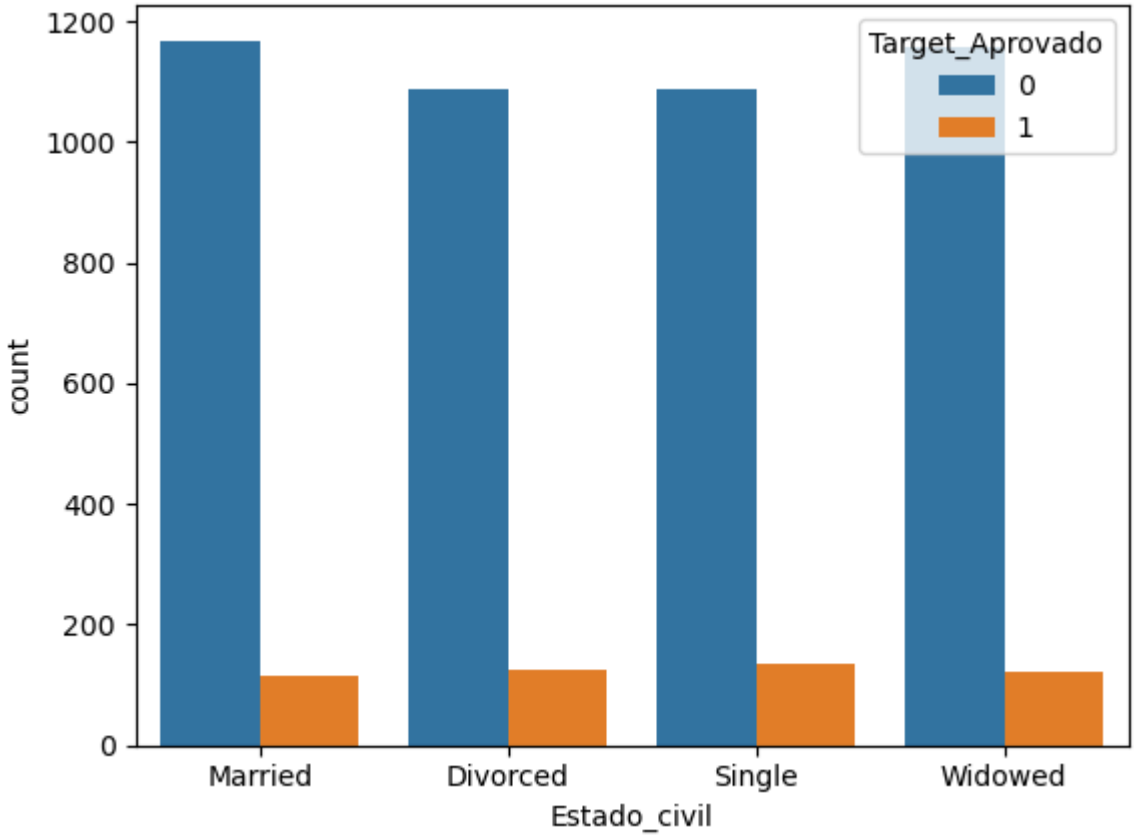
Analise de variáveis qualitativas

In [36]:
```
df_quali = df[['Status_Emprego','Estado_civil', 'Propriedade', 'Finalidade_Empre
```

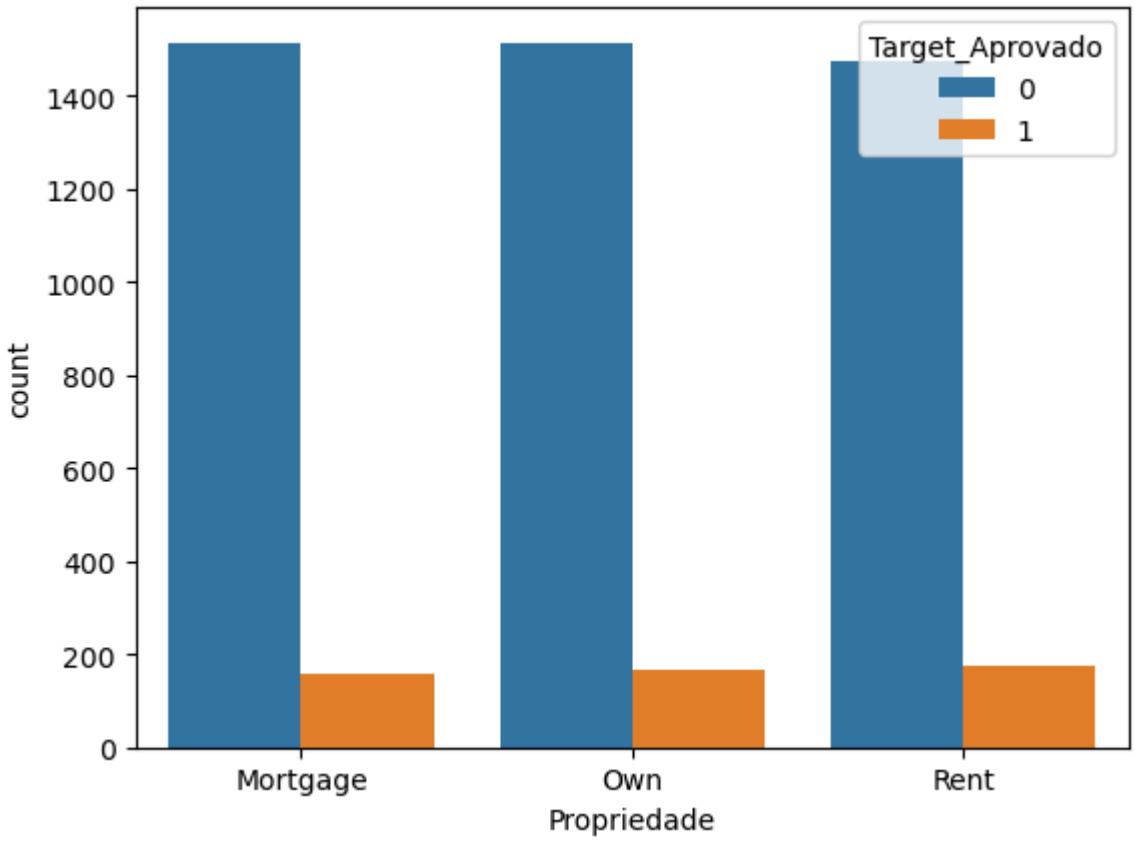
In [38]:
```
sns.countplot(data=df_quali, x='Status_Emprego', hue='Target_Aprovado');
```



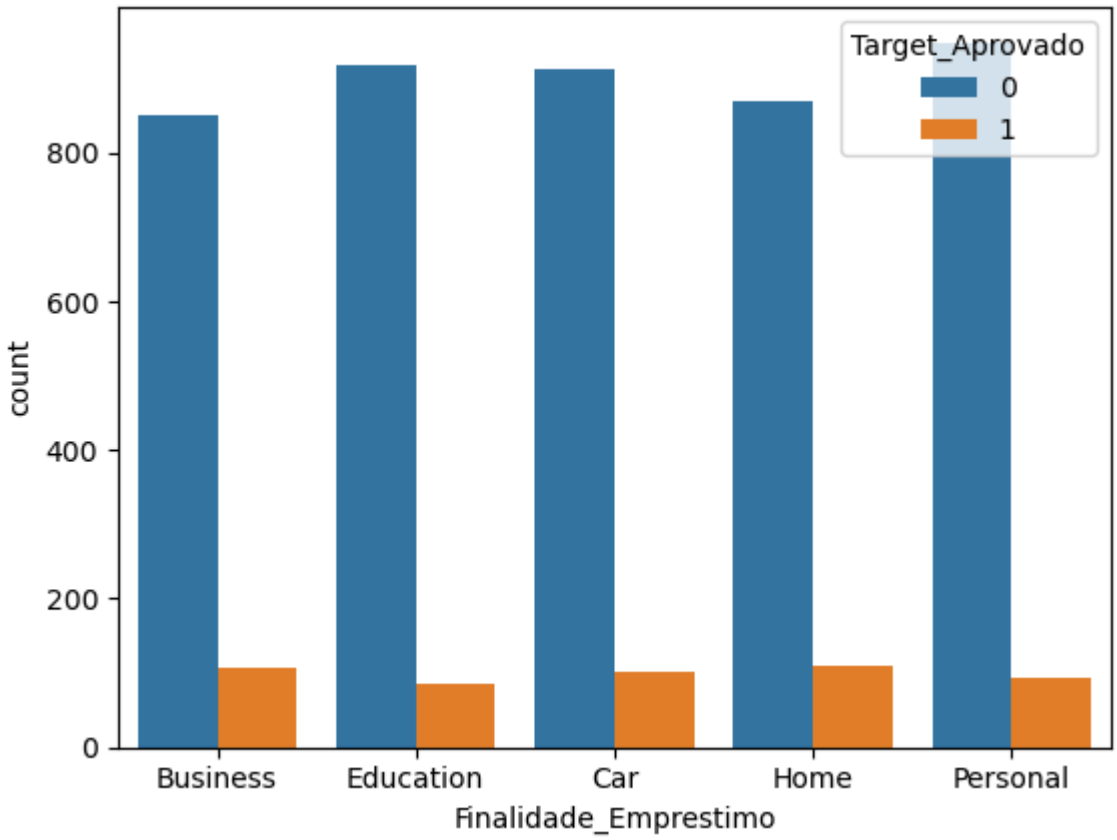
In [39]:
```
sns.countplot(data=df_quali, x='Estado_civil', hue='Target_Aprovado');
```



In [40]:
```
sns.countplot(data=df_quali, x='Propriedade', hue='Target_Aprovado');
```

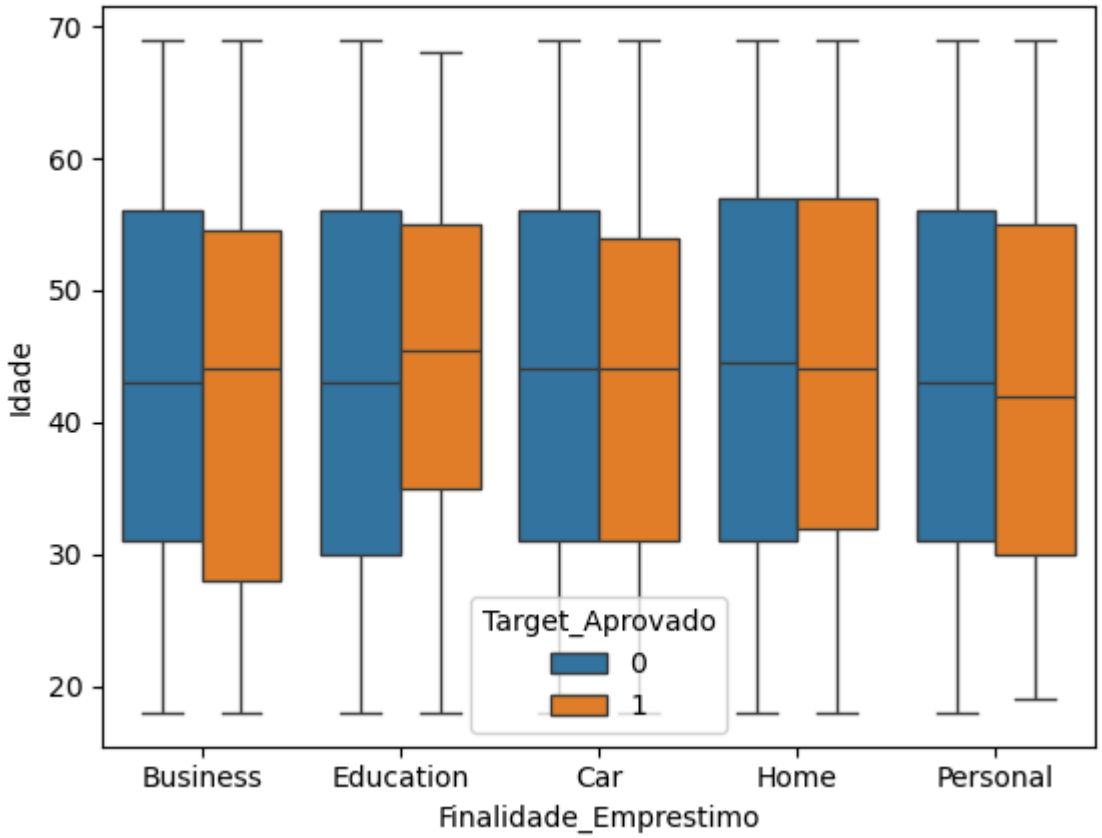


In [41]:
```
sns.countplot(data=df_quali, x='Finalidade_Emprestimo', hue='Target_Aprovado');
```

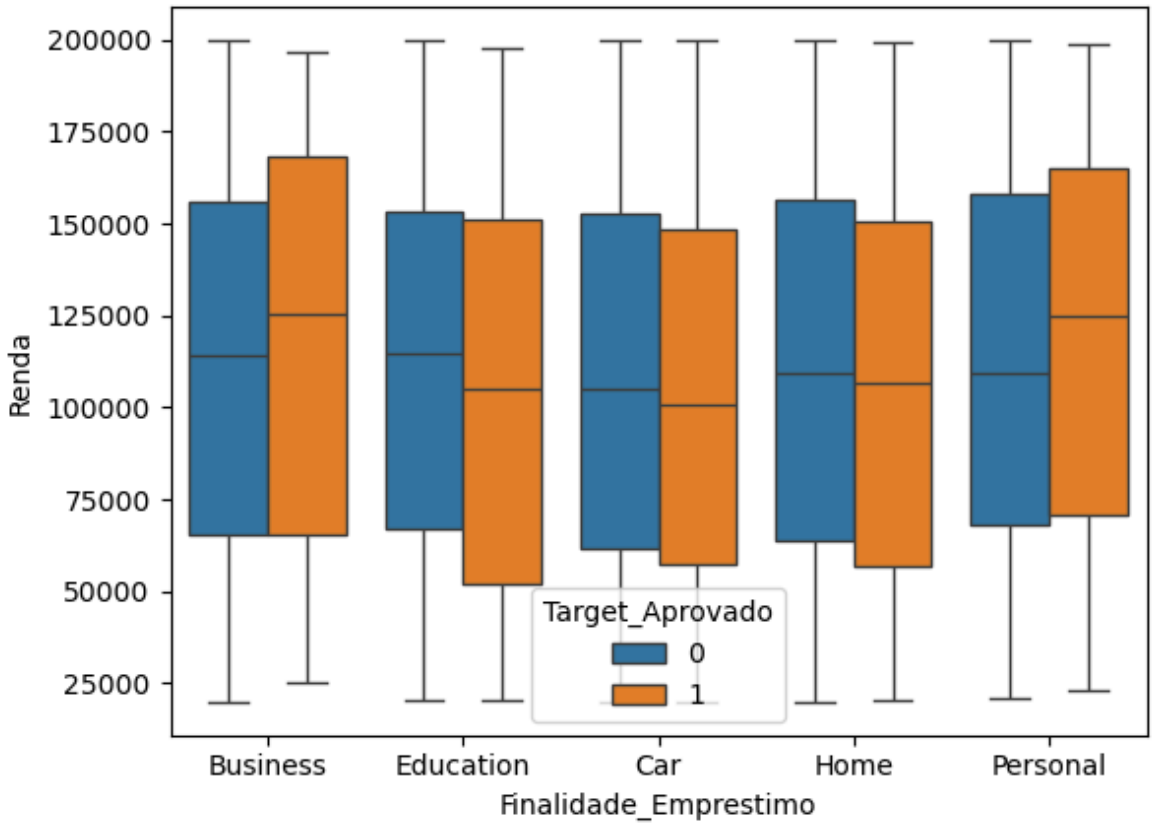


Analise Variáveis Quantitativas e Qualitativas

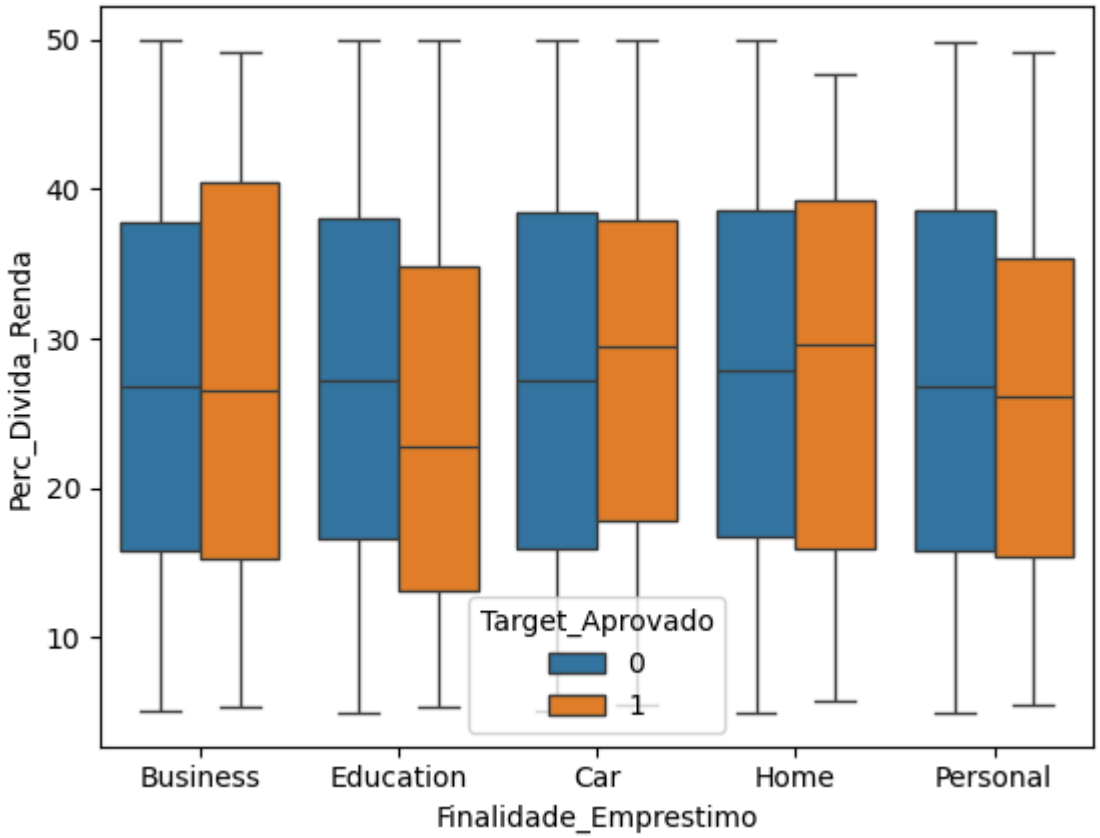
In [47]:
```
sns.boxplot(data=df, x='Finalidade_Emprestimo', y='Idade', hue='Target_Aprovado'
```



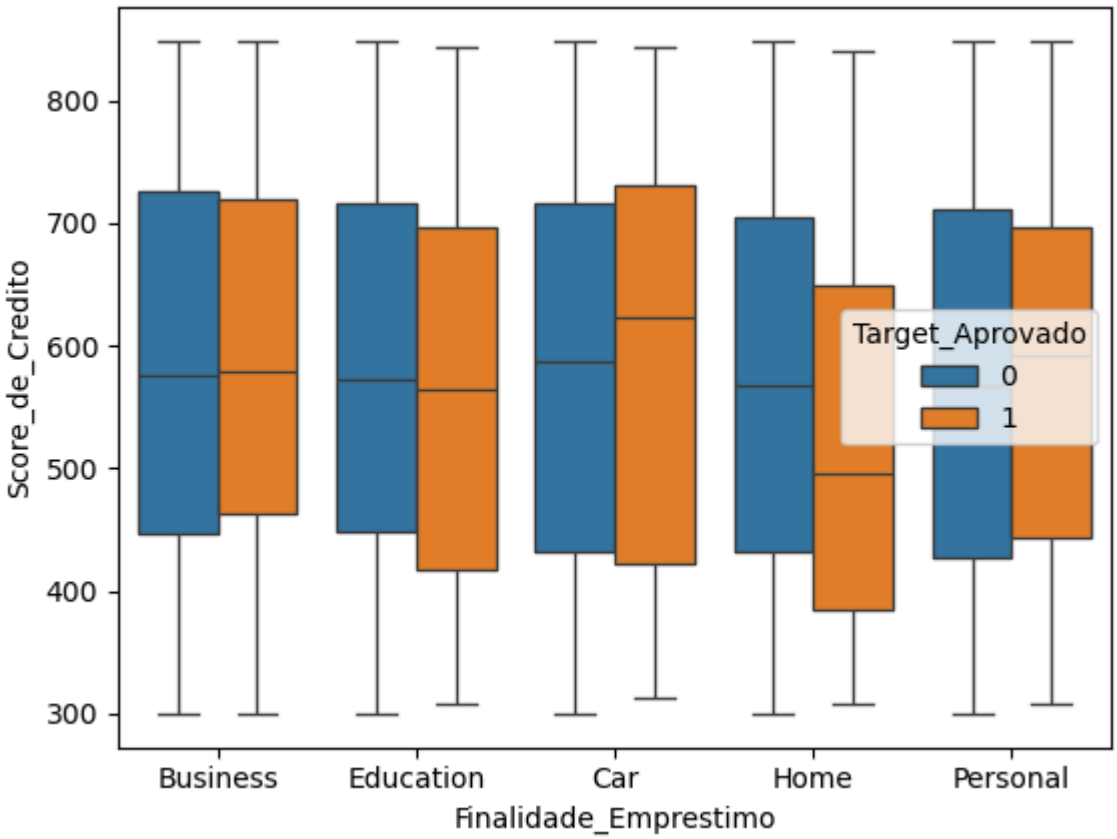
In [46]:
```
sns.boxplot(data=df, x='Finalidade_Emprestimo', y='Renda', hue='Target_Aprovado'
```



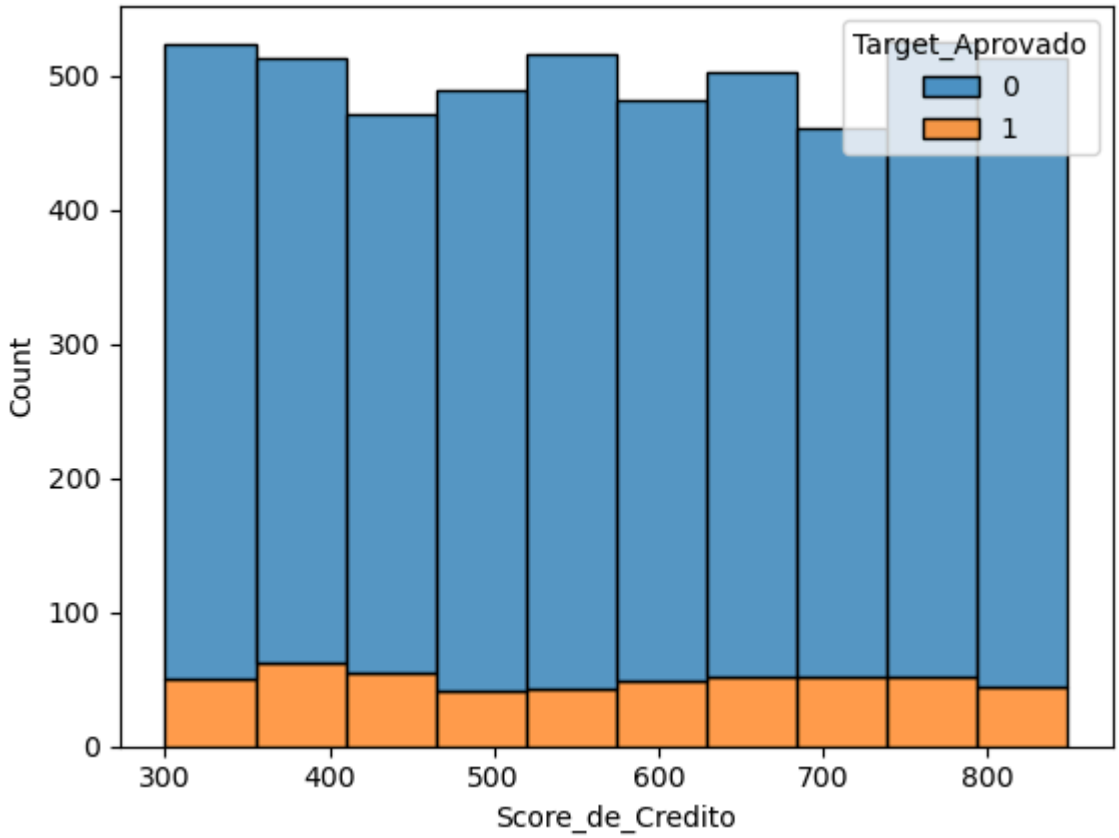
In [50]:
```
sns.boxplot(data=df, x='Finalidade_Emprestimo', y='Perc_Divida_Renda', hue='Targ
```



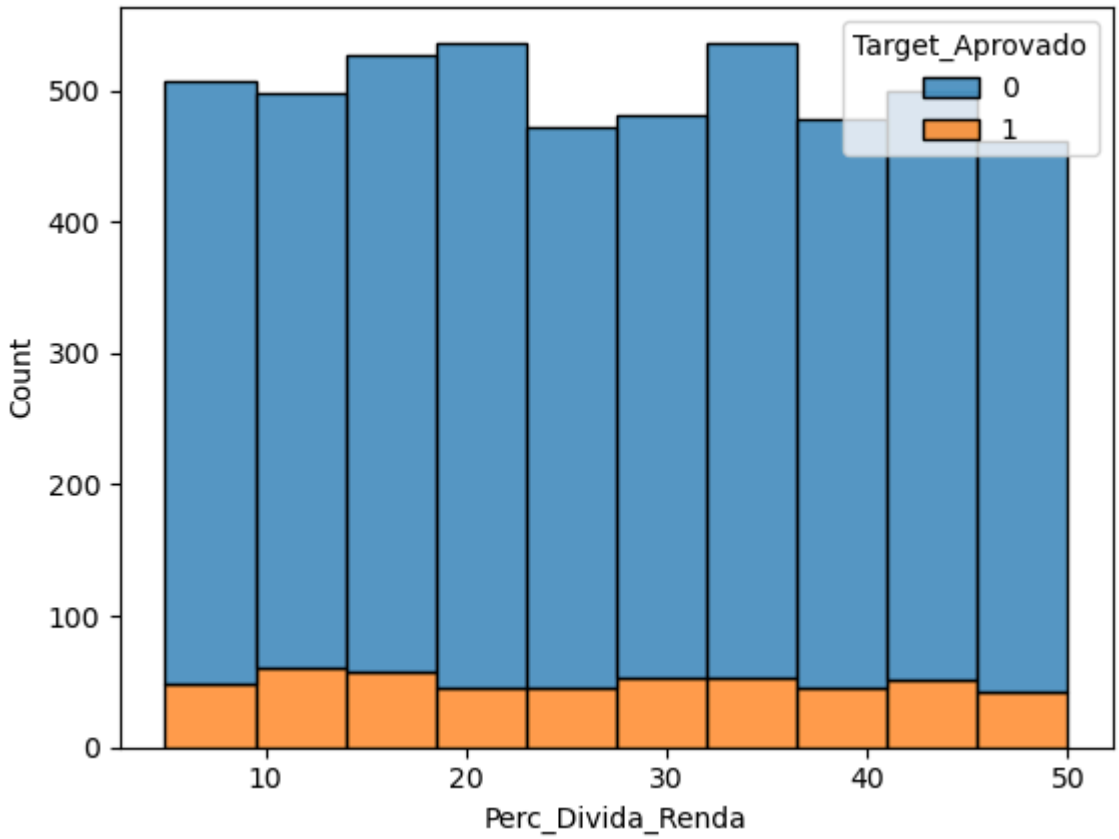
In [48]:
```
sns.boxplot(data=df, x='Finalidade_Emprestimo', y='Score_de_Credito', hue='Targe
```



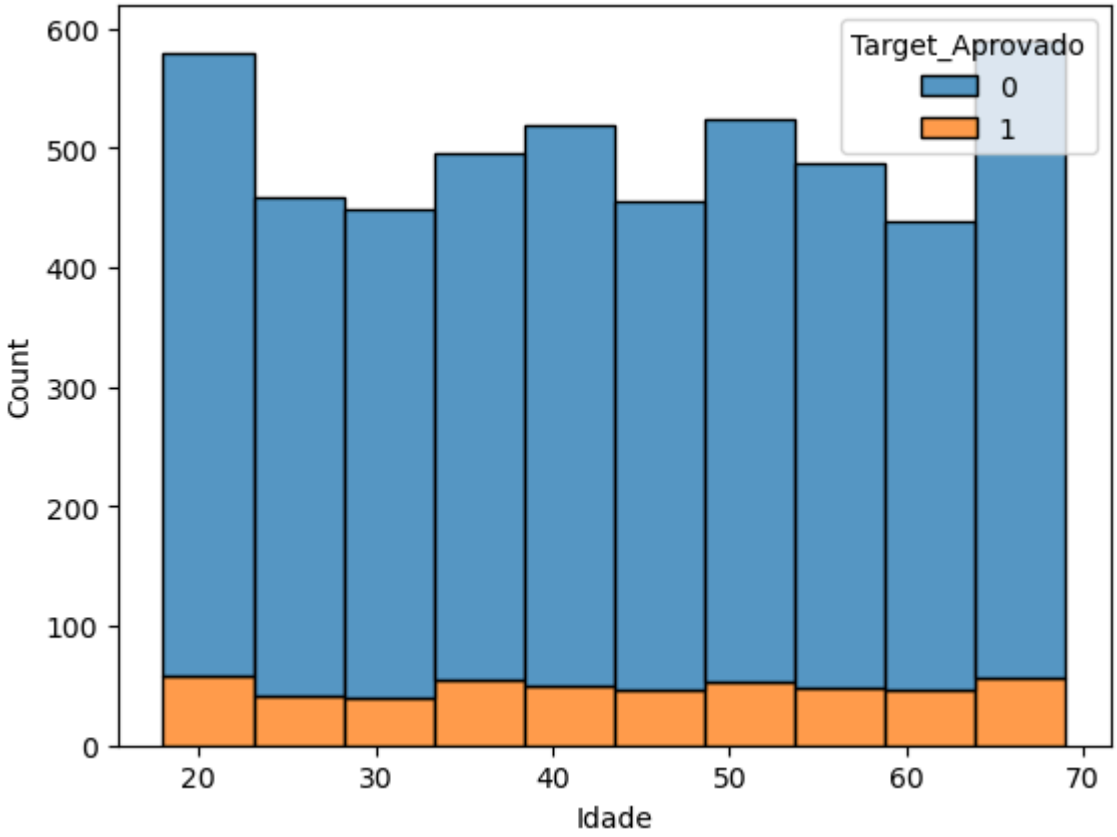
In [132…
```
sns.histplot(data=df, x='Score_de_Credito', bins=10 , hue='Target_Aprovado', mul
```



In [134…
```
sns.histplot(data=df, x='Perc_Divida_Renda',bins=10, hue='Target_Aprovado', mult
```



In [133…
```
sns.histplot(data=df, x='Idade',bins=10, hue='Target_Aprovado', multiple="stack"
```



Analise do resultado dos gráficos

- Podemos notar que cada uma das variáveis estão muito bem distribuidos em cada uma das faixas, quase de maneira homogênea;
- Percebe-se que a finalidade do emprestimo para negócios e pessoal tem uma renda maior para os aprovados;
- A finalidade do emprestimo para carros tem um score de crédito mais alto para os aprovados;
- Podemos perceber também um número maior de aprovados com Score de crédito na faixa 400;
- Clientes com percentual de relação da dívida com a renda 10% e 20%, tem maior volume de aprovador

Criação de variáveis de Dummy

Passamos a variáveis qualitativas para dummy, através do "get_dummies" e excluímo a primeira categoria de cada variável para evitar mulicolinearidade

In [135…

```
df_quali_2 = df_quali.drop(columns=['Target_Aprovado'])
df_dummies = pd.get_dummies(df_quali_2, drop_first=True).astype(int)
df_dummies.head()
```

Out[135...

	Status_Emprego_Self-Employed	Status_Emprego_Unemployed	Estado_civil_Married	Estado_civil_
0	0	1	1	
1	0	1	0	
2	1	0	0	
3	0	1	0	
4	1	0	0	

In [136...

```
df_final = df.drop(columns=df_quali_2)
df_final.columns
```

Out[136...

```
Index(['ID_Requerente', 'Idade', 'Renda', 'Score_de_Credito',
       'Valor_Emprestimo', 'Prazo_Emprestimo', 'Taxa_Juros',
       'Perc_Divida_Renda', 'Numero_dependentes', 'Target_Aprovado'],
      dtype='object')
```

In [137...

```
df_final = pd.concat([df_final, df_dummies], axis=1)
df_final.columns
```

Out[137...

```
Index(['ID_Requerente', 'Idade', 'Renda', 'Score_de_Credito',
       'Valor_Emprestimo', 'Prazo_Emprestimo', 'Taxa_Juros',
       'Perc_Divida_Renda', 'Numero_dependentes', 'Target_Aprovado',
       'Status_Emprego_Self-Employed', 'Status_Emprego_Unemployed',
       'Estado_civil_Married', 'Estado_civil_Single', 'Estado_civil_Widowed',
       'Propriedade_Own', 'Propriedade_Rent', 'Finalidade_Emprestimo_Car',
       'Finalidade_Emprestimo_Education', 'Finalidade_Emprestimo_Home',
       'Finalidade_Emprestimo_Personal'],
      dtype='object')
```

base final para estudo de regressão

In [139...

```
df_final.head()
```

Out[139...

	ID_Requerente	Idade	Renda	Score_de_Credito	Valor_Emprestimo	Prazo_Emprestim
0	1	56	21920	639	452748	7
1	2	69	126121	655	257134	6
2	3	46	96872	467	226437	7
3	4	32	101132	751	310480	1
4	5	60	22093	404	13070	1

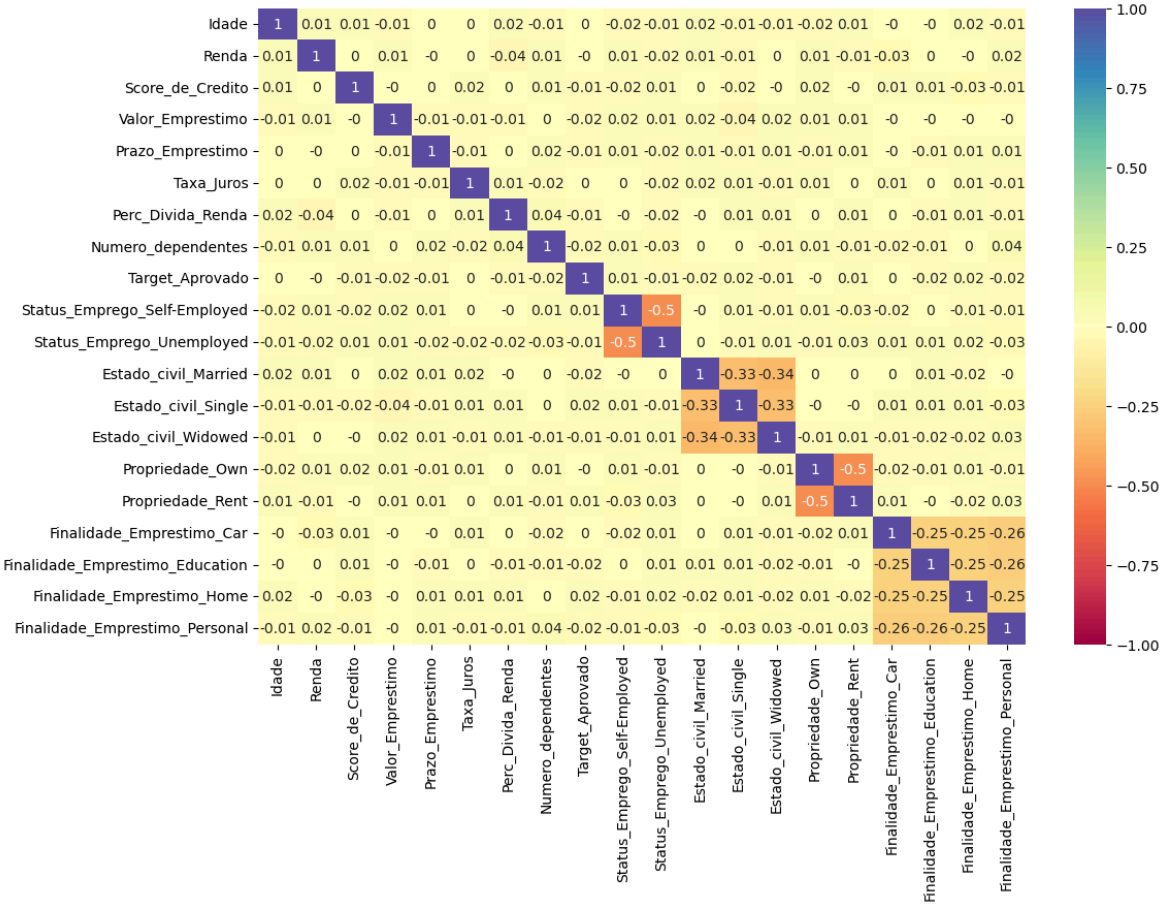
5 rows × 21 columns

In [140...

```
df_final = df_final.drop(columns=['ID_Requerente'])
df_final.columns
```

Out[140…
```
Index(['Idade', 'Renda', 'Score_de_Credito', 'Valor_Emprestimo',
       'Prazo_Emprestimo', 'Taxa_Juros', 'Perc_Divida_Renda',
       'Numero_dependentes', 'Target_Aprovado', 'Status_Emprego_Self-Employed',
       'Status_Emprego_Unemployed', 'Estado_civil_Married',
       'Estado_civil_Single', 'Estado_civil_Widowed', 'Propriedade_Own',
       'Propriedade_Rent', 'Finalidade_Emprestimo_Car',
       'Finalidade_Emprestimo_Education', 'Finalidade_Emprestimo_Home',
       'Finalidade_Emprestimo_Personal'],
      dtype='object')
```

In [143…
```
plt.figure(figsize=(12, 8))
sns.heatmap(data=round(df_final.corr(),2), annot=True, vmax=1, vmin=-1, cmap='Sp
```



Podemos perceber uma correlação muita baixa entre os dados:

- Isso se da por conta da distribuição da base as únicas variáveis que espresam alguma correlação são as variáveis dummy e mesmo assim não estamos vendo uma correlação alta o suficiente para possívelmente excluir uma delas

Construção do Modelo de Binomial da previssão de resultado e analise de significância do modelo

In [175…
```
x = df_final.loc[:, df_final.columns != 'Target_Aprovado']
y = df_final[['Target_Aprovado']]
```

In [176…
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_
print(x_train.shape)
print(x_test.shape)
```

```
print(y_train.shape)
print(y_test.shape)
```

```
(3500, 19)
(1500, 19)
(3500, 1)
(1500, 1)
```

In [192…
```
y_test.value_counts()
```

Out[192…
```
Target_Aprovado
0                1346
1                 154
Name: count, dtype: int64
```

In [177…
```
x_train = sm.add_constant(x_train)
x_test = sm.add_constant(x_test)
```

In [178…
```
modelo = sm.GLM(y_train, x_train, family=sm.families.Binomial()).fit()
print(modelo.summary())
```

Generalized Linear Model Regression Results

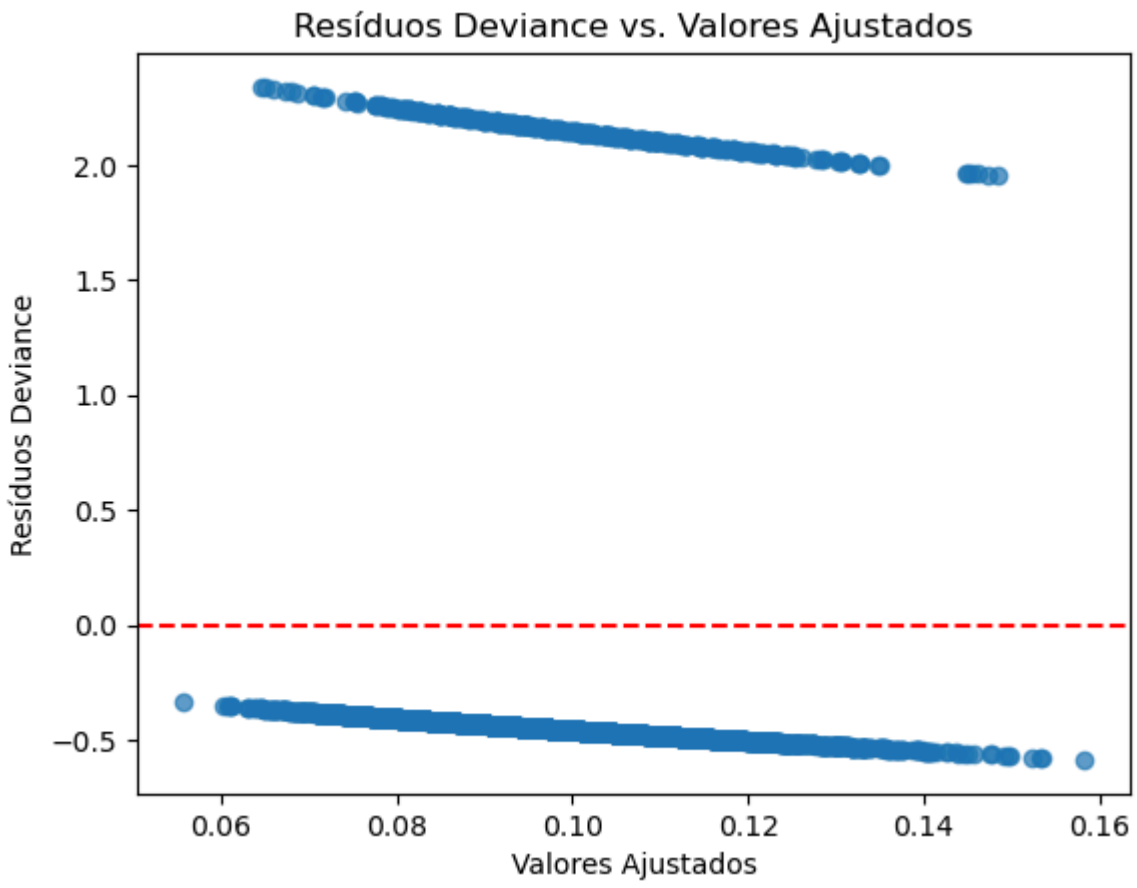
Dep. Variable:	Target_Aprovado	No. Observations:	3500
Model:	GLM	Df Residuals:	3480
Model Family:	Binomial	Df Model:	19
Link Function:	Logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-1117.4
Date:	Sun, 16 Mar 2025	Deviance:	2234.7
Time:	18:50:32	Pearson chi2:	3.50e+03
No. Iterations:	5	Pseudo R-squ. (CS):	0.002830
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.9205	0.433	-4.440	0.000	-2.768	-1.073
Idade	-5.207e-05	0.004	-0.014	0.989	-0.007	0.007
Renda	-1.21e-06	1.09e-06	-1.105	0.269	-3.36e-06	9.36e-07
Score_de_Credito	4.439e-05	0.000	0.124	0.901	-0.001	0.001
Valor_Emprestimo	-3.398e-07	3.99e-07	-0.852	0.394	-1.12e-06	4.42e-07
Prazo_Emprestimo	0.0004	0.003	0.132	0.895	-0.005	0.006
Taxa_Juros	0.0058	0.016	0.372	0.710	-0.025	0.036
Perc_Divida_Renda	-0.0051	0.004	-1.145	0.252	-0.014	0.004
Numero_dependentes	0.0162	0.041	0.393	0.694	-0.064	0.097
Status_Emprego_Self-Employed	0.0562	0.139	0.405	0.686	-0.216	0.328
Status_Emprego_Unemployed	0.0563	0.142	0.395	0.693	-0.223	0.336
Estado_civil_Married	-0.1656	0.159	-1.039	0.299	-0.478	0.147
Estado_civil_Single	-0.0519	0.159	-0.327	0.744	-0.363	0.259
Estado_civil_Widowed	-0.2034	0.163	-1.245	0.213	-0.524	0.117
Propriedade_Own	0.0878	0.142	0.618	0.537	-0.191	0.366
Propriedade_Rent	0.1372	0.140	0.978	0.328	-0.138	0.412
Finalidade_Emprestimo_Car	-0.0793	0.178	-0.446	0.655	-0.427	0.269
Finalidade_Emprestimo_Education	-0.2214	0.184	-1.206	0.228	-0.581	0.138
Finalidade_Emprestimo_Home	0.0587	0.176	0.334	0.739	-0.286	0.404
Finalidade_Emprestimo_Personal	-0.1642	0.178	-0.921	0.357	-0.514	0.185

Analise p-value :

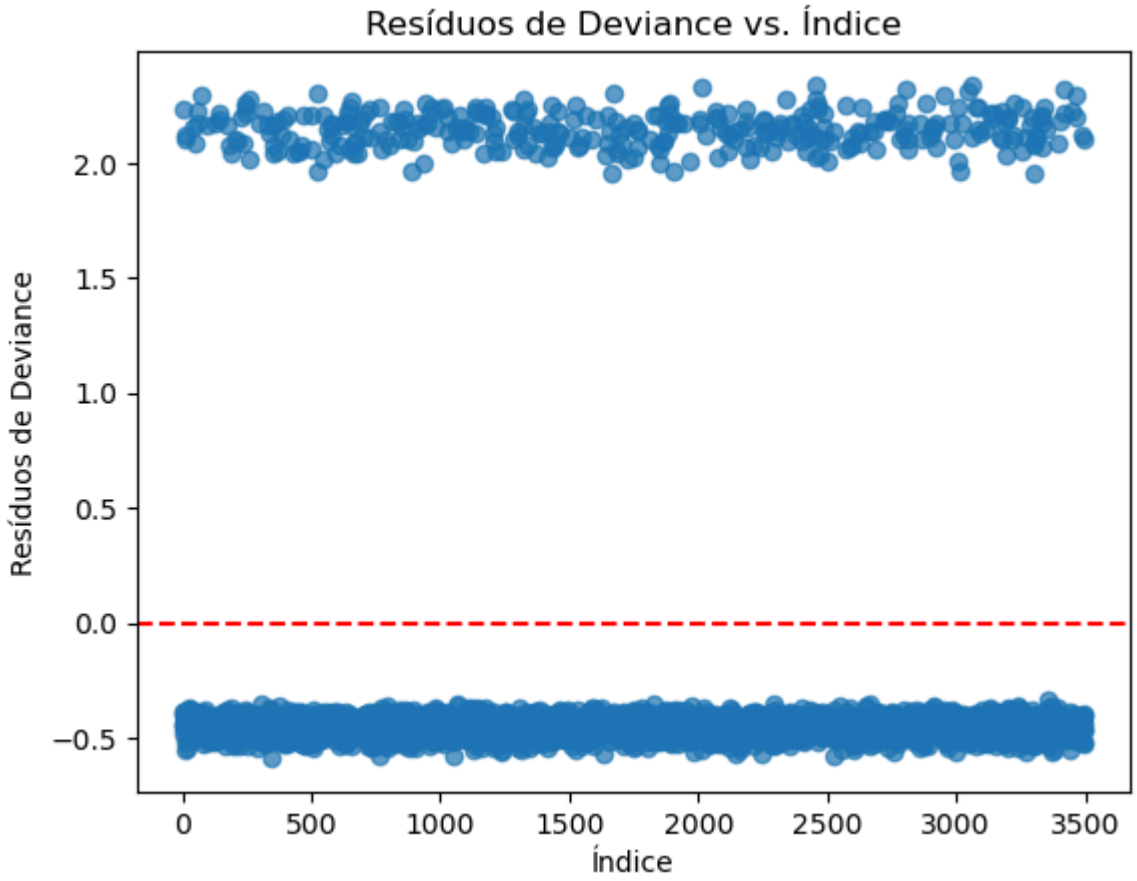
- Analisando o resultado do p-value das variáveis do modelo binomial, podemos perceber que nenhuma das variáveis possuem valores significativos. Uma das teorias seria por conta do desbalanceamento das variáveis alvos já que a variável target = 1 possui menos de 10% de representação, e o modelo pode estar tendo dificualdade para capturar essa característica nos dados.
- Podemos testar mais na frete se utilizando o modelo onde possamos adicionar um pessoa maior a essa variável, ou fazendo um rebalanceamento das variáveis podemos obter resultados mais aceitáveis.

In [179…
```
residuos_deviance = modelo.resid_deviance
residuos_pearson = modelo.resid_pearson
valores_ajustados = modelo.fittedvalues
plt.scatter(valores_ajustados, residuos_deviance, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.title("Resíduos Deviance vs. Valores Ajustados")
plt.xlabel("Valores Ajustados")
plt.ylabel("Resíduos Deviance")
plt.show()
```



In [180…
```
plt.scatter(range(len(residuos_deviance)), residuos_deviance, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Índice')
plt.ylabel('Resíduos de Deviance')
plt.title('Resíduos de Deviance vs. Índice')
```

Out[180… `Text(0.5, 1.0, 'Resíduos de Deviance vs. Índice')`



In [181…
```
deviance_test_statistic = modelo.deviance
deviance_df = modelo.df_resid
deviance_p_value = 1 - stats.chi2.cdf(deviance_test_statistic, deviance_df)
print("Teste de Deviance para os Resíduos:")
print("Estatística de teste:", deviance_test_statistic)
print("Graus de liberdade:", deviance_df)
print("Valor p:", deviance_p_value)
```

```
Teste de Deviance para os Resíduos:
Estatística de teste: 2234.7436370163286
Graus de liberdade: 3480
Valor p: 1.0
```

Deviance Residuals :

- Com um Valor p maior que (0,05), concluímos que o modelo ajusta bem os dados.

In [182…
```
residuos_pearson = modelo.resid_pearson

pearson_test_statistic = np.sum(residuos_pearson**2)
pearson_df = len(residuos_pearson) - modelo.df_model - 1
pearson_p_value = 1 - stats.chi2.cdf(pearson_test_statistic, pearson_df)
print("\nTeste de Pearson para os Resíduos:")
print("Estatística de teste:", pearson_test_statistic)
print("Graus de liberdade:", pearson_df)
print("Valor p:", pearson_p_value)
```

```
Teste de Pearson para os Resíduos:
Estatística de teste: 3499.5779822928516
Graus de liberdade: 3480
Valor p: 0.40430411862062765
```

Pearson Chi-Square Test

- Com um Valor p maior que (0,05), concluímos que o modelo ajusta bem os dados.

In [196…
```
y_test_pred = modelo.predict(x_test)
y_test_result = (y_test_pred >= 0.12).astype(int)
y_test_result.value_counts()
```

Out[196…
```
0    1365
1     135
Name: count, dtype: int64
```

In [197…
```
y_train_pred = modelo.fittedvalues
y_train_result = (y_train_pred >= 0.12).astype(int)
y_train_result.value_counts()
```

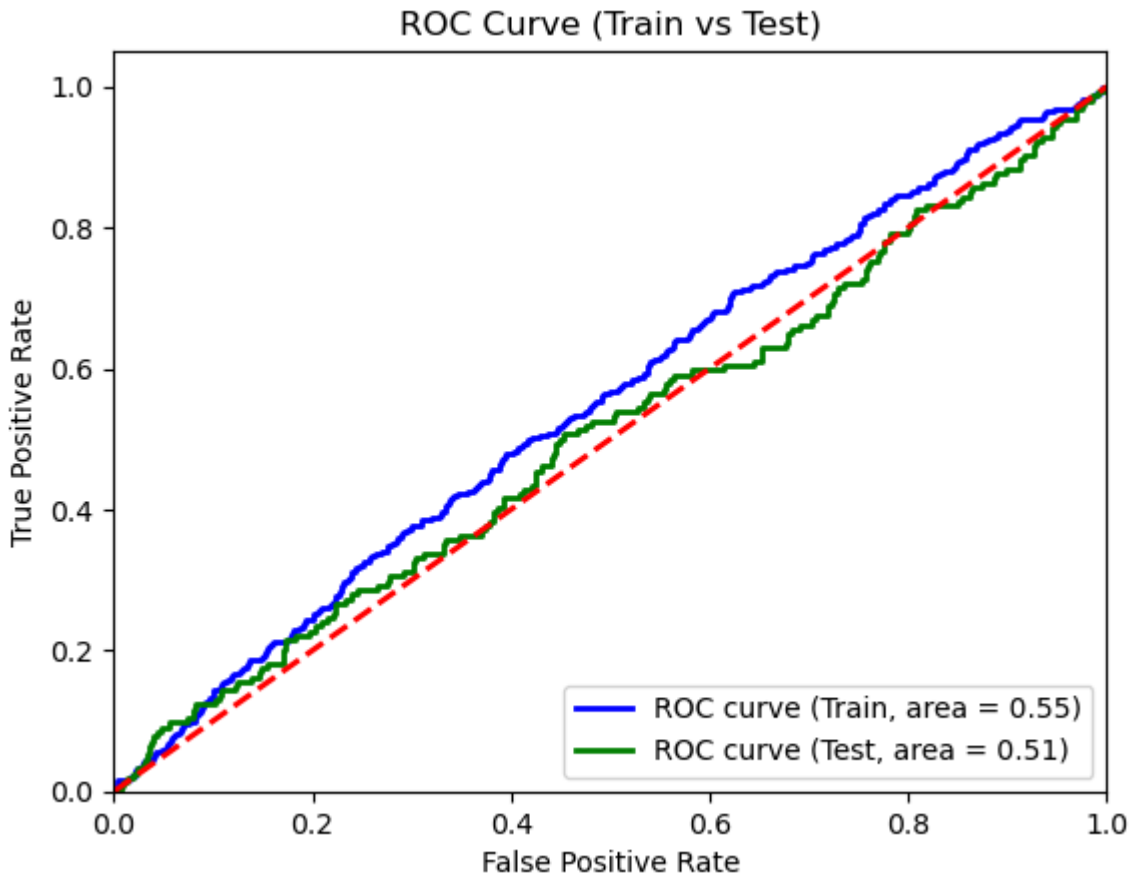
Out[197…
```
0    3163
1     337
Name: count, dtype: int64
```

Curva ROC e AUC

In [206…
```
# Curva ROC e AUC para a base de treino
fpr_train, tpr_train, _ = roc_curve(y_train, modelo.fittedvalues)
roc_auc_train = auc(fpr_train, tpr_train)

# Curva ROC e AUC para a base de teste
fpr_test, tpr_test, _ = roc_curve(y_test, y_test_pred)
roc_auc_test = auc(fpr_test, tpr_test)

plt.figure()
plt.plot(fpr_train, tpr_train, color='blue', lw=2, label='ROC curve (Train, area
plt.plot(fpr_test, tpr_test, color='green', lw=2, label='ROC curve (Test, area =
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve (Train vs Test)')
plt.legend(loc="lower right")
plt.show()
```



Analise Curva ROC

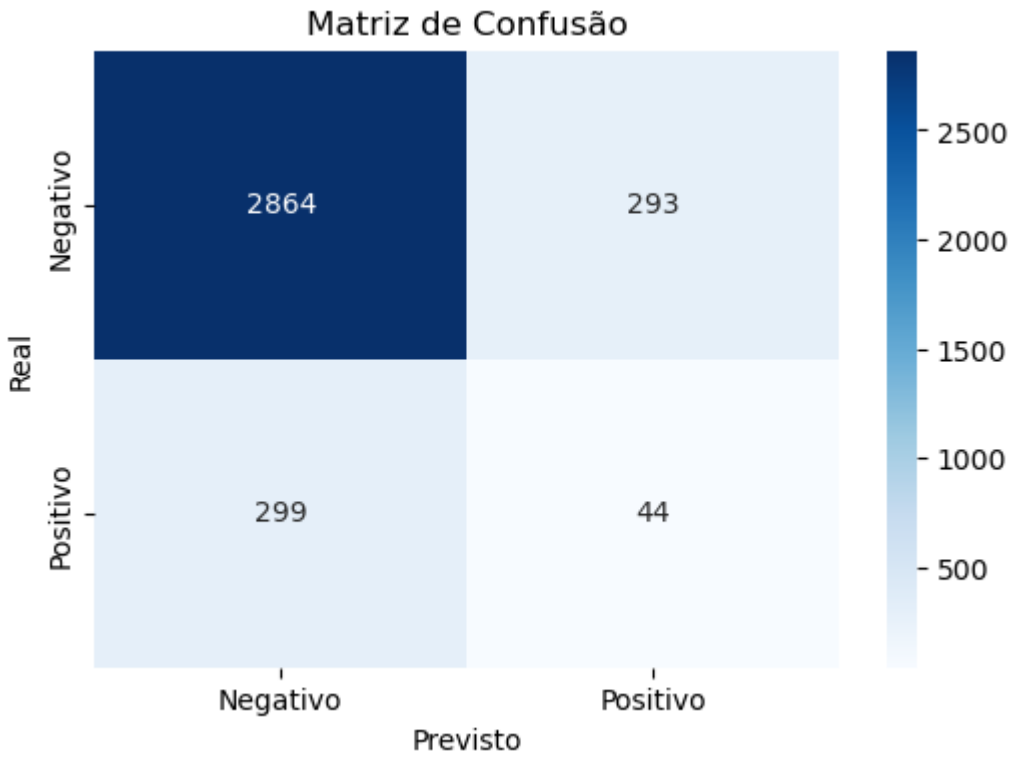
- Podemos ver um resultado, um pocuo abaixo da curva muito por conta da questão dos dados estarem desbalanceados

Matriz de confução

In [199…
```
cm_train = confusion_matrix(y_train, y_train_result)
```

In [200…
```
plt.figure(figsize=(6,4))
sns.heatmap(cm_train, annot=True, fmt="d", cmap="Blues", xticklabels=["Negativo"

plt.xlabel("Previsto")
plt.ylabel("Real")
plt.title("Matriz de Confusão")
plt.show()
```



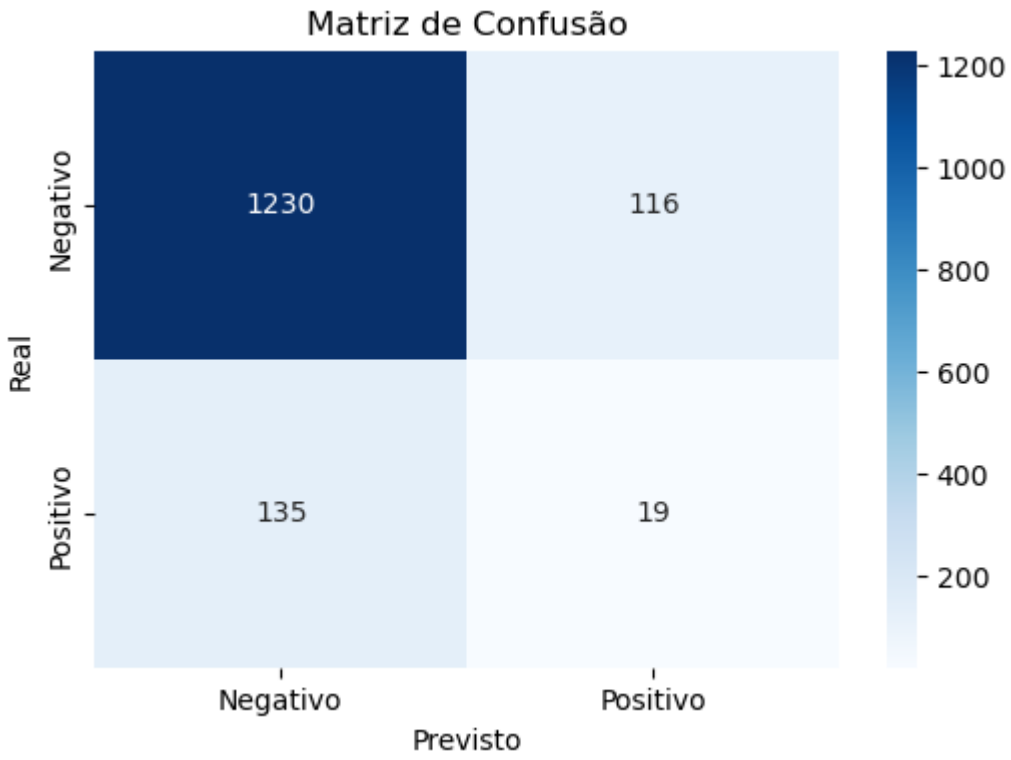
In [201…
```
accuracy = accuracy_score(y_train, y_train_result)
print('Accuracy no Test:', accuracy)
```

```
Accuracy no Test: 0.8308571428571428
```

In [203…
```
cm_test = confusion_matrix(y_test, y_test_result)
```

In [204…
```
plt.figure(figsize=(6,4))
sns.heatmap(cm_test, annot=True, fmt="d", cmap="Blues", xticklabels=["Negativo",

plt.xlabel("Previsto")
plt.ylabel("Real")
plt.title("Matriz de Confusão")
plt.show()
```



In [205…
```
accuracy = accuracy_score(y_test, y_test_result)
print('Accuracy no Test:', accuracy)
```

```
Accuracy no Test: 0.8326666666666667
```

Analisando a matriz de confução

- Podemos notar que foi necessário uma nota de corte muita baixa, muito por conta da dificualdade de capitar informações da target

In [207…
```
def hosmer_lemeshow_test(y_true, y_pred, n_groups=10):

    df = pd.DataFrame({'y_true': y_true, 'y_pred': y_pred})
    df = df.sort_values(by='y_pred').reset_index(drop=True)

    df['group'] = pd.qcut(df['y_pred'], q=n_groups, duplicates='drop')
    n_groups = df['group'].nunique()

    # Calcular estatísticas por grupo
    observed = df.groupby('group')['y_true'].sum()
    expected = df.groupby('group')['y_pred'].sum()
    total = df.groupby('group').size()

    chi2_stat = ((observed - expected) ** 2 / (expected * (1 - expected / total)
    p_value = 1 - chi2.cdf(chi2_stat, df=n_groups - 2)

    return chi2_stat, p_value
```

In [208…
```
x_train = x_train.to_numpy()
x_test = x_test.to_numpy()
y_train = y_train.to_numpy()
y_test = y_test.to_numpy()

x_train = np.array(x_train)
x_test = np.array(x_test)
```

```
y_train = np.array(y_train).ravel()
y_test = np.array(y_test).ravel()
```

In [209…
```
# Realizar0
chi2_stat, p_value = hosmer_lemeshow_test(y_test, y_test_pred)
print(f"Estatística do Teste de Hosmer-Lemeshow: {chi2_stat:.4f}")
print(f"P-valor: {p_value:.4f}")
```

```
Estatística do Teste de Hosmer-Lemeshow: 12.0096
P-valor: 0.1508
```

```
C:\Users\gabri\AppData\Local\Temp\ipykernel_10156\2137551005.py:10: FutureWarnin
g:

The default of observed=False is deprecated and will be changed to True in a futu
re version of pandas. Pass observed=False to retain current behavior or observed=
True to adopt the future default and silence this warning.

C:\Users\gabri\AppData\Local\Temp\ipykernel_10156\2137551005.py:11: FutureWarnin
g:

The default of observed=False is deprecated and will be changed to True in a futu
re version of pandas. Pass observed=False to retain current behavior or observed=
True to adopt the future default and silence this warning.

C:\Users\gabri\AppData\Local\Temp\ipykernel_10156\2137551005.py:12: FutureWarnin
g:

The default of observed=False is deprecated and will be changed to True in a futu
re version of pandas. Pass observed=False to retain current behavior or observed=
True to adopt the future default and silence this warning.
```

Teste hosmer_lemeshow

- p-value > 0.05 → O modelo se ajusta bem aos dados (não há evidências contra a qualidade do ajuste).

In [210…
```
# Calculando o KS2 train
ks_stat, p_value = ks_2samp(y_train_pred[y_train == 1], y_train_pred[y_train ==

print(f"KS2: {ks_stat:.4f} (ou {ks_stat*100:.2f}%)")
```

```
KS2: 0.0837 (ou 8.37%)
```

In [211…
```
# Calculando o KS2 test
ks_stat, p_value = ks_2samp(y_test_pred[y_test == 1], y_test_pred[y_test == 0])

print(f"KS2: {ks_stat:.4f} (ou {ks_stat*100:.2f}%)")
```

```
KS2: 0.0533 (ou 5.33%)
```

teste kolmogorov smirnov

- Podemos notar que o modelo tem bastante dificuldade de dividir as variáveis 1 e 0

In [166…
```
coeficientes = modelo.params
odds_ratio = np.exp(coeficientes)
print("\nOdds Ratio:")
print(odds_ratio)
```

```
Odds Ratio:
const                                0.146528
Idade                                0.999948
Renda                                0.999999
Score_de_Credito                     1.000044
Valor_Emprestimo                     1.000000
Prazo_Emprestimo                     1.000366
Taxa_Juros                           1.005833
Perc_Divida_Renda                    0.994923
Numero_dependentes                   1.016294
Status_Emprego_Self-Employed         1.057805
Status_Emprego_Unemployed            1.057903
Estado_civil_Married                 0.847385
Estado_civil_Single                  0.949427
Estado_civil_Widowed                 0.815962
Propriedade_Own                      1.091728
Propriedade_Rent                     1.147108
Finalidade_Emprestimo_Car            0.923776
Finalidade_Emprestimo_Education      0.801405
Finalidade_Emprestimo_Home           1.060489
Finalidade_Emprestimo_Personal       0.848605
dtype: float64
```

In [167…
```
def calcular_impacto_percentual(modelo):

    coeficientes = modelo.params

    odds_ratio = np.exp(coeficientes)

    impacto_percentual = (odds_ratio - 1) * 100

    df_impacto = pd.DataFrame({
        "Coeficiente": coeficientes,
        "Odds Ratio": odds_ratio,
        "Impacto %": impacto_percentual
    })

    return df_impacto

df_impacto = calcular_impacto_percentual(modelo)
print(df_impacto)
```

	Coeficiente	Odds Ratio	Impacto %
const	-1.920536e+00	0.146528	-85.347167
Idade	-5.207073e-05	0.999948	-0.005207
Renda	-1.209858e-06	0.999999	-0.000121
Score_de_Credito	4.438688e-05	1.000044	0.004439
Valor_Emprestimo	-3.397803e-07	1.000000	-0.000034
Prazo_Emprestimo	3.655420e-04	1.000366	0.036561
Taxa_Juros	5.816482e-03	1.005833	0.583343
Perc_Divida_Renda	-5.089977e-03	0.994923	-0.507704
Numero_dependentes	1.616250e-02	1.016294	1.629382
Status_Emprego_Self-Employed	5.619610e-02	1.057805	5.780510
Status_Emprego_Unemployed	5.628878e-02	1.057903	5.790314
Estado_civil_Married	-1.656000e-01	0.847385	-15.261489
Estado_civil_Single	-5.189660e-02	0.949427	-5.057297
Estado_civil_Widowed	-2.033880e-01	0.815962	-18.403839
Propriedade_Own	8.776183e-02	1.091728	9.172807
Propriedade_Rent	1.372439e-01	1.147108	14.710784
Finalidade_Emprestimo_Car	-7.928578e-02	0.923776	-7.622411
Finalidade_Emprestimo_Education	-2.213884e-01	0.801405	-19.859462
Finalidade_Emprestimo_Home	5.873016e-02	1.060489	6.048904
Finalidade_Emprestimo_Personal	-1.641616e-01	0.848605	-15.139516

Analise Odds Ratio

- Na variável Idade podemos ver que para cada unidade de aumento, as chances do evento ocorrer reduzem em aproximadamente -0.005%.
- Na variável Renda podemos ver que para cada unidade de aumento, as chances do evento ocorrer reduzem em aproximadamente -0.0001%.
- Na variável Score_de_Credito podemos ver que para cada unidade de aumento, as chances do evento ocorrer aumentam em aproximadamente 0.004%.
- Na variável Valor_Emprestimo podemos ver que para cada unidade de aumento, as chances do evento ocorrer reduzem em aproximadamente -0.00003%.
- Na variável Prazo_Emprestimo podemos ver que para cada unidade de aumento, as chances do evento ocorrer aumentam em aproximadamente 0.036%.
- Na variável Taxa_Juros podemos ver que para cada unidade de aumento, as chances do evento ocorrer aumentam em aproximadamente 0.58%.
- Na variável Perc_Divida_Renda podemos ver que para cada unidade de aumento, as chances do evento ocorrer reduzem em aproximadamente -0.50%.
- Na variável Numero_dependentes podemos ver que para cada unidade de aumento, as chances do evento ocorrer aumentam em aproximadamente 1.62%.
- Indivíduos com Status_Emprego_Self-Employed, as chances do evento ocorrer aumentam em aproximadamente 5.78%.
- Indivíduos com Status_Emprego_Unemployed, as chances do evento ocorrer aumentam em aproximadamente 5.79%.
- Indivíduos com Estado_civil_Married , as chances do evento ocorrer reduzem em aproximadamente -15.26%.
- Indivíduos com Estado_civil_Single, as chances do evento ocorrer reduzem em aproximadamente -5.05%.
- Indivíduos com Estado_civil_Widowed, as chances do evento ocorrer reduzem em aproximadamente -18.40%.

- Indivíduos com Propriedade_Own, as chances do evento ocorrer aumentam em aproximadamente 9.17%.
- Indivíduos com Propriedade_Rent, as chances do evento ocorrer aumentam em aproximadamente 14.71%.
- Indivíduos com Finalidade_Emprestimo_Car, as chances do evento ocorrer reduzem em aproximadamente -7.62%.
- Indivíduos com Finalidade_Emprestimo_Education, as chances do evento ocorrer reduzem em aproximadamente -19.85%.
- Indivíduos com Finalidade_Emprestimo_Home, as chances do evento ocorrer aumentam em aproximadamente 6.04%.
- Indivíduos com Finalidade_Emprestimo_Personal, as chances do evento ocorrer reduzem em aproximadamente -15.13%.