Projeto de categorização de criticas

O objetivo é treinar um modelo para detectar automaticamente resenhas negativas. Usarei um conjunto de dados de resenhas de filmes do IMDB com rotulagem de polaridade para criar um modelo para classificar resenhas como positivas e negativas.

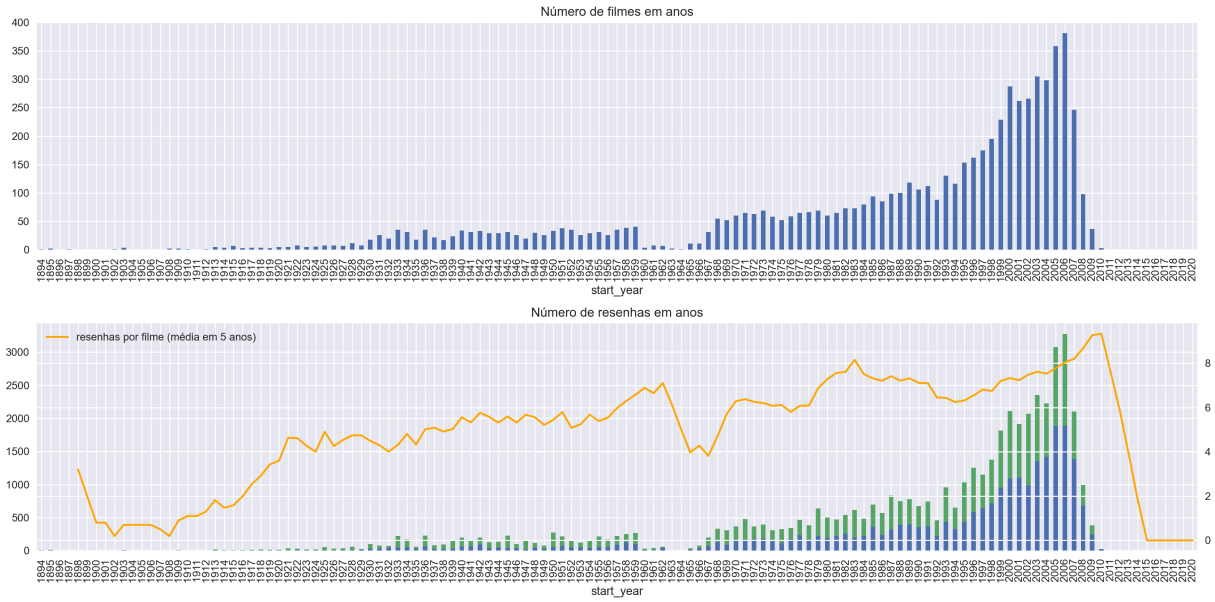
Diversas tecnicas e modelos diferentes foram usados para demonstrar como fazer modelos de classficação de texto.

AED

Mostrando o numero de resenhas por ano.

Azul são as negativas e verde as positivas no segundo grafico.

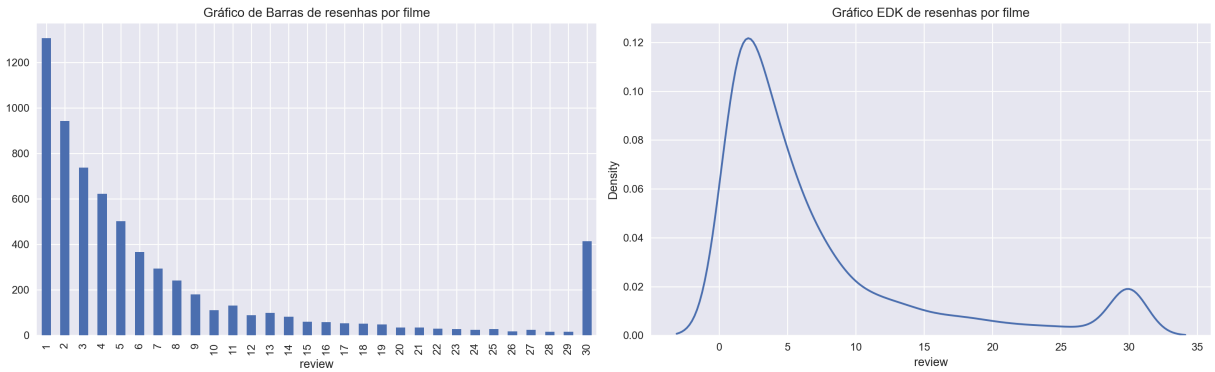
In [29]:
```
fig, axs = plt.subplots(2, 1, figsize=(16, 8))

ax = axs[0]

dft1 = df_reviews[['tconst', 'start_year']].drop_duplicates() \
    ['start_year'].value_counts().sort_index()
dft1 = dft1.reindex(index=np.arange(dft1.index.min(), max(dft1.index.max(),
dft1.plot(kind='bar', ax=ax)
ax.set_title('Número de filmes em anos')

ax = axs[1]

dft2 = df_reviews.groupby(['start_year', 'pos'])['pos'].count().unstack()
dft2 = dft2.reindex(index=np.arange(dft2.index.min(), max(dft2.index.max(),
dft2.plot(kind='bar', stacked=True, label='#reviews (neg, pos)', ax=ax)

dft2 = df_reviews['start_year'].value_counts().sort_index()
dft2 = dft2.reindex(index=np.arange(dft2.index.min(), max(dft2.index.max(),
dft3 = (dft2/dft1).fillna(0)
axt = ax.twinx()
dft3.reset_index(drop=True).rolling(5).mean().plot(color='orange', label='re

lines, labels = axt.get_legend_handles_labels()
ax.legend(lines, labels, loc='upper left')

ax.set_title('Número de resenhas em anos')

fig.tight_layout()
```



Vamos verificar a distribuição do número de resenhas por filme com a contagem exata e o EDK (Estimativa de densidade kernel)

EDK calcula a densidade de resenhas por filme para aquele pedaço do grafico, ajudando a interpretar dados que não seguem uma distribuição normal, binomial, etc. Basicamente dexa os grafico bonitinho, arrendondado e facil de tirar conclusões.

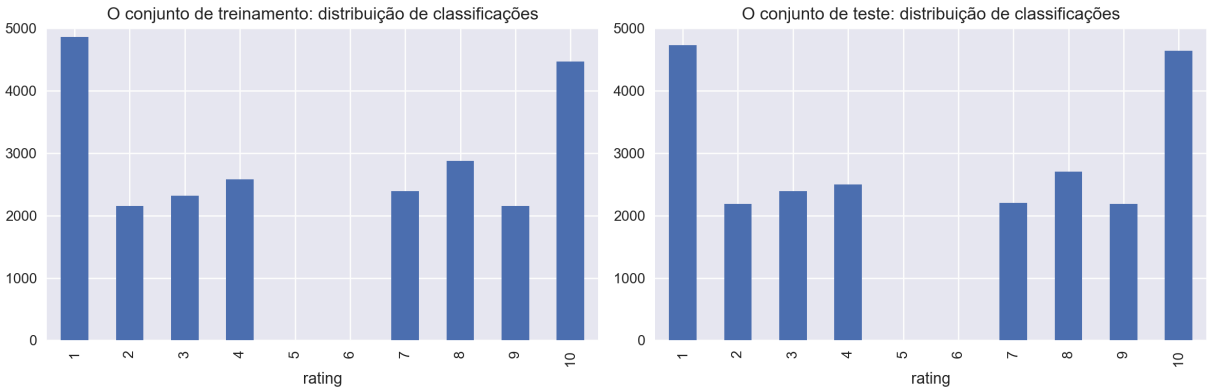
In [30]:
```
fig, axs = plt.subplots(1, 2, figsize=(16, 5))

ax = axs[0]
dft = df_reviews.groupby('tconst')['review'].count() \
    .value_counts() \
    .sort_index()
dft.plot.bar(ax=ax)
ax.set_title('Gráfico de Barras de resenhas por filme')

ax = axs[1]
dft = df_reviews.groupby('tconst')['review'].count()
sns.kdeplot(dft, ax=ax)
ax.set_title('Gráfico EDK de resenhas por filme')

fig.tight_layout()
```



Alguns filmes tem muito mais resenhas que outros, provavelmente por serem mais populares, mas alem de desequilibrar a classe, isso pode levar a uma limitação do vocabulario do modelo

A distribuição de classificações

In [32]:
```
fig, axs = plt.subplots(1, 2, figsize=(12, 4))

ax = axs[0]
dft = df_reviews.query('ds_part == "train"')['rating'].value_counts().sort_i
dft = dft.reindex(index=np.arange(min(dft.index.min(), 1), max(dft.index.max
dft.plot.bar(ax=ax)
ax.set_ylim([0, 5000])
ax.set_title('O conjunto de treinamento: distribuição de classificações')

ax = axs[1]
dft = df_reviews.query('ds_part == "test"')['rating'].value_counts().sort_in
dft = dft.reindex(index=np.arange(min(dft.index.min(), 1), max(dft.index.max
dft.plot.bar(ax=ax)
ax.set_ylim([0, 5000])
ax.set_title('O conjunto de teste: distribuição de classificações')

fig.tight_layout()
```

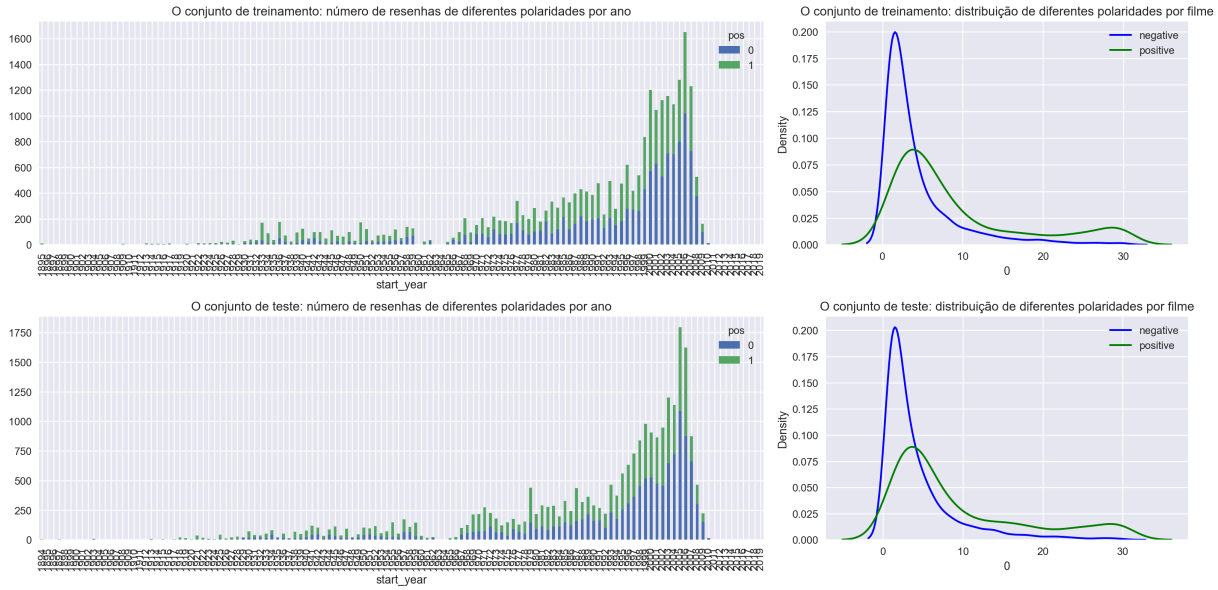


Distribuição de resenhas negativas e positivas ao longo dos anos para duas partes do conjunto de dados

In [33]:
```
fig, axs = plt.subplots(2, 2, figsize=(16, 8), gridspec_kw=dict(width_ratios

ax = axs[0][0]

dft = df_reviews.query('ds_part == "train"').groupby(['start_year', 'pos'])[
dft.index = dft.index.astype('int')
dft = dft.reindex(index=np.arange(dft.index.min(), max(dft.index.max(), 2020
dft.plot(kind='bar', stacked=True, ax=ax)
ax.set_title('O conjunto de treinamento: número de resenhas de diferentes po

ax = axs[0][1]

dft = df_reviews.query('ds_part == "train"').groupby(['tconst', 'pos'])['pos
sns.kdeplot(dft[0], color='blue', label='negative', kernel='epa', ax=ax)
```

```
sns.kdeplot(dft[1], color='green', label='positive', kernel='epa', ax=ax)
ax.legend()
ax.set_title('O conjunto de treinamento: distribuição de diferentes polarida

ax = axs[1][0]

dft = df_reviews.query('ds_part == "test"').groupby(['start_year', 'pos'])['
dft.index = dft.index.astype('int')
dft = dft.reindex(index=np.arange(dft.index.min(), max(dft.index.max(), 2020
dft.plot(kind='bar', stacked=True, ax=ax)
ax.set_title('O conjunto de teste: número de resenhas de diferentes polarida

ax = axs[1][1]

dft = df_reviews.query('ds_part == "test"').groupby(['tconst', 'pos'])['pos'
sns.kdeplot(dft[0], color='blue', label='negative', kernel='epa', ax=ax)
sns.kdeplot(dft[1], color='green', label='positive', kernel='epa', ax=ax)
ax.legend()
ax.set_title('O conjunto de teste: distribuição de diferentes polaridades po

fig.tight_layout()
```

```
/var/folders/6m/8zrf46xx10z087pclwk1njth0000gn/T/ipykernel_28377/364716119.p
y:14: UserWarning:

Support for alternate kernels has been removed; using Gaussian kernel.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(dft[0], color='blue', label='negative', kernel='epa', ax=ax)
/var/folders/6m/8zrf46xx10z087pclwk1njth0000gn/T/ipykernel_28377/364716119.p
y:15: UserWarning:

Support for alternate kernels has been removed; using Gaussian kernel.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(dft[1], color='green', label='positive', kernel='epa', ax=ax)
/var/folders/6m/8zrf46xx10z087pclwk1njth0000gn/T/ipykernel_28377/364716119.p
y:30: UserWarning:

Support for alternate kernels has been removed; using Gaussian kernel.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(dft[0], color='blue', label='negative', kernel='epa', ax=ax)
/var/folders/6m/8zrf46xx10z087pclwk1njth0000gn/T/ipykernel_28377/364716119.p
y:31: UserWarning:

Support for alternate kernels has been removed; using Gaussian kernel.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(dft[1], color='green', label='positive', kernel='epa', ax=ax)
```



As classes estão um pouco desequilibradas, com a quantidade de 1 ou positivos sendo o dobro de negativos ou 0.

Isso pode parecer um desequilibrio grande, mas em dados reais esse desequilibrio entre classes tende a ser muito maior.

Procedimento de Avaliação

Accuracy nos da a porcentagem de acertos em relação ao total de tentativas, não é uma metrica muito boa pois não avalia avaliações falsas e verdadeiras, só se errou ou não.

Precision é a porcentagem de acertos positivos verdadeiros pelo total de tudo classificado como positivo, falso ou verdadeiro, não é muito util quando temos muito mais classificações negativas que positivas, da enfase em falsos positivos.

Recall é a porcentagem de acertos positivos verdadeiros pelo total de positivos verdadeiros mais os falsos negativos, é bom para classificar eventos que ja ocorreram, como fraude e atos de violencia, pois da enfase em falsos negativos.

F1 é calculado usando falsos e verdadeiros positivos e negativos, pode ser considerada uma matrica que equilibra precision e recall.É uma boa metrica principal para a maioria dos modelos.

ROC é um grafico de curva baseado na probabilidade de um modelo acertar a classifação dado um certo nivel de treinamento.Desde que o dataset seja razoavelmente balanceado, quanto maior a area abaixo da curva(AUC), melhor o modelo.

PRC, ou curva de precisão recall é usada de maneira semelhante ao ROC, porem para dados desbalanceados.

Normalização

Todos os modelos abaixo aceitam textos em letras minúsculas e sem dígitos, sinais de pontuação, etc.

Então colocamos todo o texto em minusculo e separamos palavra por palavra(tokens), aplicando nessas palavras o lematizador, que vai padronizar a conjugação, alem de tirar emojis, sinais de pontuação e qualquer coisa que possa interferir no treinamento.

Precisamos que as palavras estejam padronizadas, pois na hora que convertermos o texto em um vetor, se uma mesma palavra começar com minuscula ou maiuscula essas duas versões serão reconhecidas como palavras diferentes.

Temos uma versão de lematizador em portugues, o enelvo.

Treinar / Testar Conjunto

Felizmente, todo o conjunto de dados já está dividido em partes de treinamento/teste. A opção correspondente é 'ds_part'.

Caso não estivesse usariamos o train_test_spli abaixo, do scikit

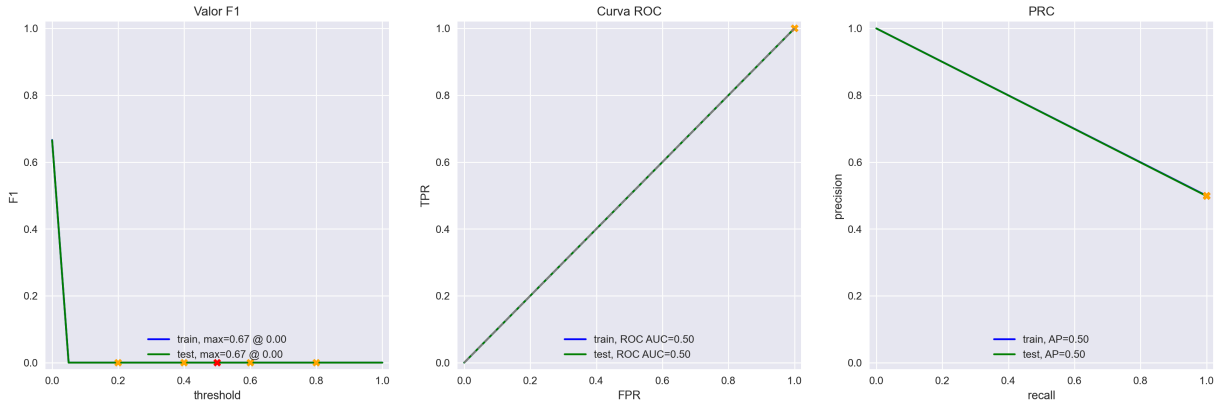
Trabalhando com modelos

Modelo 0 - Constante

Modelo para simular como seria adivinhar as classes escolhendo a que acontece com mais frequencia sempre, serve como benchmark para a perfomance de modelos reais.

In [42]:
```
evaluate_model(model, train_features_1, train_target, test_features_1, test_
```

```
          train  test
Acurácia    NaN   NaN
F1          NaN   NaN
APS         0.5   0.5
ROC AUC     0.5   0.5
```



Modelo 1 - NLTK, TF-IDF e Regressão Logistica

TF-IDF é um metodo de vetorização de palavras, onde cada palavra ganha importancia baseado em seu contexto e quantas vezes aparece no texto.

Ele é calculado em duas partes:

O TF é o numero de occorencias de uma palavra em texto especifico, divido pelo numero total de palavras no texto. Ele é calculado por texto ou frase, cada linha do csv.

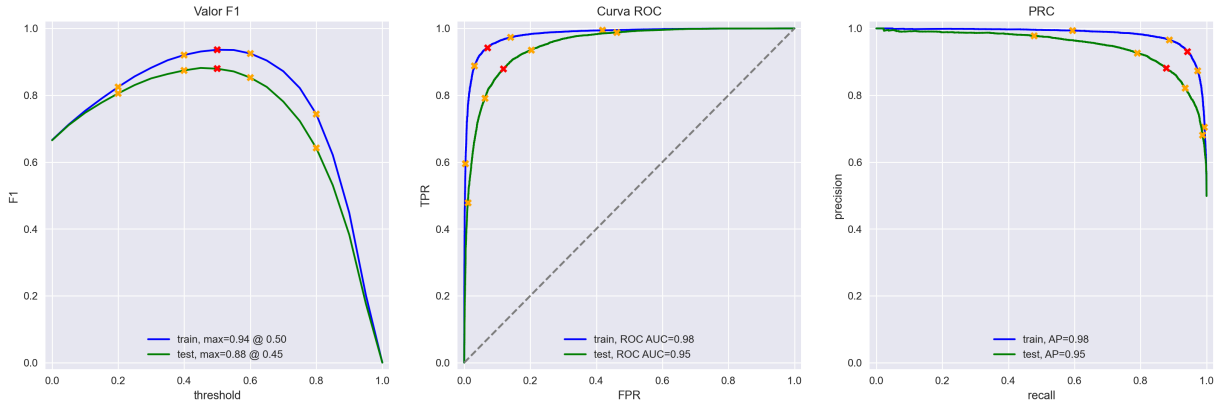
O IDF é multiplicado ao TF,seu papel é diminuir a importancia de palavras comuns, que aparecem muitas vezes nos dados. Ele é baseado em um calculo usando todo os textos contidos nos dados, sendo o log de base 10 do numero de textos que a palavra apareceu dividido pelo total de textos. Caso voce separe os dados em teste e treinamento voce so treina a parte do idf com os dados de treinamento, para os dados sempre ficarem no mesmo formato final, algo necessario para o uso de modelos de IA, alem de ser mais justo pois o metodo de teste dos modelos tem que ser consistente.

Com a exceção de modelos com sua propria vetorização, como o BERT, usamos o TF-IDF. Podemos considerar cada palavra unica como uma caracteristica que o modelo avalia.

Estarei usando como primeiro exemplo a regressão logistica, que ja vimos como funciona na aula, para mostrar um dos modelos que aprendemos em ação e compara-lo a outros modelos.

In [45]:
```
evaluate_model(model_1, train_features_1, train_target, test_features_1, tes
```

```
           train  test
Acurácia     NaN   NaN
F1           NaN   NaN
APS         0.98  0.95
ROC AUC     0.98  0.95
```



Modelo 2 - NLTK, TF-IDF e Arvore de classificação

A arvore de classificação é um otimo modelo para quando as classes estão desbalanceadas, temos muito mais reviews positivos que negativas ou vice versa, alem de ter alta explicabilidade, diferente de modelos usando redes neurais, que muitas vezes funcionam como caixas pretas.

Elas funcionam em camadas, e o numero de camadas é a depth(Profundida), e o maximo de camadas possiveis pode ser limitado no inicio para evitar overfit.

Cada camada e composta de modulos de decisão, que são um if para uma certa caracteristica, então temos um if na primeira camada, até dois na segunda, até quatro na terceira, etc, com um numero maximo de if igual 2^max_depth-1.

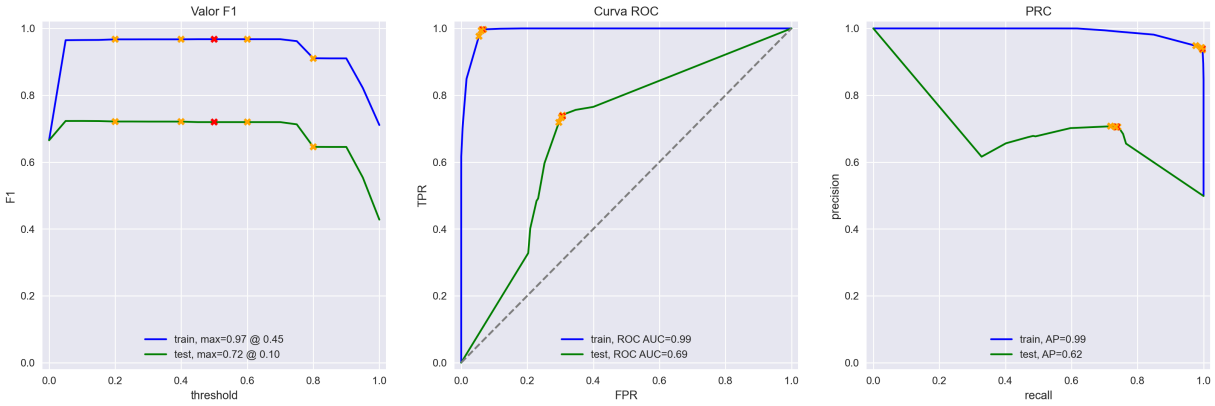
Esses ifs são feitos determinando qual caracteristica é mais importante para classificação e qual é o melhor ponto para dividir os dados baseado nessa caracteristica, cada camada subsequente pegando caracteristicas cada vez mais importantes.

Com camadas o suficiente voce pode avaliar todas as caracteristicas dos dados e alcançar um precisão de 100% nos dados de treino, mas isso provavelmente causaria um overfit, e os dados de teste teriam uma pessima precisão.

As nossas classes não estão muito desbalançeadas, então a arvore não deve ter uma precisão muito maior que a regressão logisticas

In [48]:
```
evaluate_model(model_2, train_features_1, train_target, test_features_1, tes
```

```
           train  test
Acurácia     NaN   NaN
F1           NaN   NaN
APS         0.99  0.62
ROC AUC     0.99  0.69
```



Aqui claramente tivemos problemas de overfit, por isso a grande diferença entre o treino e teste.

Esse overfit é inerente da estrutura da arvore, pois não temos uma profundidade com melhor resultados de teste.

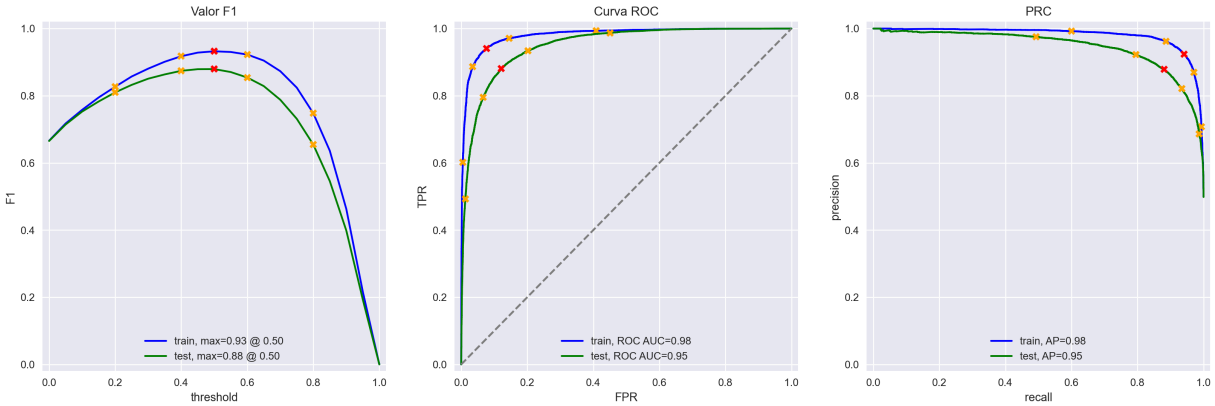
Modelo 3 - SpaCy, TF-IDF e Regressão Logistica

O spacy é um conjunto de stop words diferentes.

Com ele a vetorização do texto vai ser diferente, então os resultados podem ser melhores, piores ou o mesmo.

In [52]:
```
evaluate_model(model_2, train_features_2, train_target, test_features_2, tes
```

```
           train  test
Acurácia     NaN   NaN
F1           NaN   NaN
APS         0.98  0.95
ROC AUC     0.98  0.95
```



Tivemos praticamente a mesma perfomance, seja usando spacy como nosso lematizador ou NLTK, porem spacy demora bem mais tempo, por isso consideramos sua perfomance pior nesse caso

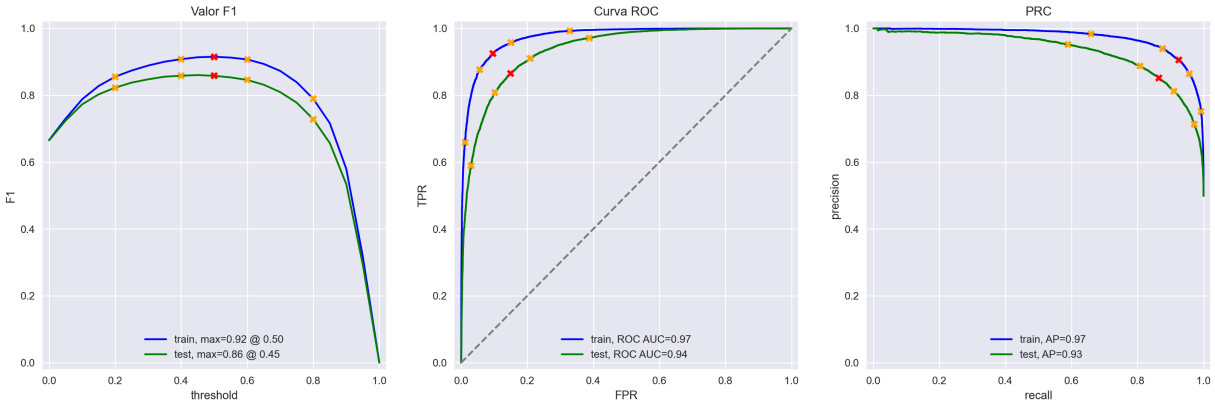
Modelo 4 - SpaCy, TF-IDF e LGBMClassifier

O LGBMClassifier é um classificador da microsoft que tem boa precisão em comparação a outros classficadores que não usam redes neurais, alem de perfomance excelente com o multi-threading automatico e sua capacidade de converter colunas categoricas.

Eu recomendo sempre testar ele para tarefas de classificação.

In [55]:
```
evaluate_model(model_3, train_features_2, train_target, test_features_2, tes
```

	train	test
Acurácia	NaN	NaN
F1	NaN	NaN
APS	0.97	0.93
ROC AUC	0.97	0.94



Minhas Resenhas

In [56]:
```
my_reviews = pd.DataFrame([
    'Bem, eu fiquei entediado quando os macacos começaram a gritar no obelis
    'Eu fiquei realmente fascinado com o filme',
    'Que tentativa podre de comédia. Nem uma única piada cai, todo mundo age
    'Lançar na Netflix foi uma jogada corajosa e eu realmente aprecio ser ca
    'The actors were old and not interested in their roles, only being there
    'i didnt expect the new version to be so good! The writers really cared
    'the film has its goods and bads, but it is decent. I could see myself s
], columns=['review'])

my_reviews['review_norm'] = my_reviews['review']

my_reviews
```

Out[56]:

	review	review_norm
0	Bem, eu fiquei entediado quando os macacos com...	Bem, eu fiquei entediado quando os macacos com...
1	Eu fiquei realmente fascinado com o filme	Eu fiquei realmente fascinado com o filme
2	Que tentativa podre de comédia. Nem uma única ...	Que tentativa podre de comédia. Nem uma única ...
3	Lançar na Netflix foi uma jogada corajosa e eu...	Lançar na Netflix foi uma jogada corajosa e eu...
4	The actors were old and not interested in thei...	The actors were old and not interested in thei...
5	i didnt expect the new version to be so good! ...	i didnt expect the new version to be so good! ...
6	the film has its goods and bads, but it is dec...	the film has its goods and bads, but it is dec...

Modelo 2

In [57]:
```
texts = my_reviews['review_norm']

my_reviews_pred_prob = model_2.predict_proba(vect.transform(texts))[:, 1]

for i, review in enumerate(texts.str.slice(0, 100)):
    print(f'{my_reviews_pred_prob[i]:.2f}:  {review}')
```

```
0.48:  Bem, eu fiquei entediado quando os macacos começaram a gritar no obel
isko gigante e dormi no meio do
0.59:  Eu fiquei realmente fascinado com o filme
0.57:  Que tentativa podre de comédia. Nem uma única piada cai, todo mundo a
ge de forma irritante e barulhe
0.48:  Lançar na Netflix foi uma jogada corajosa e eu realmente aprecio ser
capaz de assistir episódio após
0.35:  The actors were old and not interested in their roles, only being the
re to get some money.
0.58:  i didnt expect the new version to be so good! The writers really care
d for the original
0.33:  the film has its goods and bads, but it is decent. I could see myself
seeing it again
```

Modelo 3

In [58]:
```
texts = my_reviews['review_norm']

my_reviews_pred_prob = model_3.predict_proba(vect.transform(texts.apply(lamb

for i, review in enumerate(texts.str.slice(0, 100)):
    print(f'{my_reviews_pred_prob[i]:.2f}:  {review}')
```

```
0.64:  Bem, eu fiquei entediado quando os macacos começaram a gritar no obel
isko gigante e dormi no meio do
0.67:  Eu fiquei realmente fascinado com o filme
0.66:  Que tentativa podre de comédia. Nem uma única piada cai, todo mundo a
ge de forma irritante e barulhe
0.64:  Lançar na Netflix foi uma jogada corajosa e eu realmente aprecio ser
capaz de assistir episódio após
0.50:  The actors were old and not interested in their roles, only being the
re to get some money.
0.76:  i didnt expect the new version to be so good! The writers really care
d for the original
0.23:  the film has its goods and bads, but it is decent. I could see myself
seeing it again
```

Parece que o modelo não consegue classficar avalições em portugues por não estar presente em seu vocabulario, como esperado.

Alem disso, nas avaliações em ingles, foi capaz de prever a positiva de maneira correta em ambos os casos, mas a negativa em só um deles, com o LGBMClassifier o caracterizando como 0.50, exatamente no ponto de virada entre positivo e negativo.

Ademais, a avaliação que diz que o filme é mais ou menos em ingles é reconhecida de maneira excessivamente negativa por ambos os modelos, talvez mostrando uma falta de compreensão da organização de frases e nuancia da linguaguem em relação a avaliações que não são extremas, seria interessante analizar o quão polarizadas ou radicais são as avaliações no banco de dados e ver se é um fator que influencia a falta de nuancia do modelo.

Parece que o modelo LightLGBM tem a tendencia a apresentar resultados mais positivos comparado a regressão logistica

Conclusões

O motivo da diferença dos resultados nas resenhas para o banco de dados provavelmente vem do fato de estarem em linguagems diferentes, assim o "dicionario" em que o modelo foi treinado não inclui as palavras em portugues

Alguns filmes tem muito mais resenhas que outros, provavelmente por serem mais populares, mas alem de desequilibrar a classe, isso pode levar a uma limitação do vocabulario do modelo

E alguns anos tambem tem muitos mais filmes que outros, com uma faixa de 1999 a 2009, 10 anos, como pico

As classes estão desequilibradas, com a quantidade de 1 ou positivos sendo o dobro de negativos ou 0, talvez seja interessante aplicar tecnicas para equilibrar mais as classes como upsampling ou dar pesos em modelos como no BERT, caso ele fosse usado.

O melhor modelo foi a regressão logistica sem o processamento de texto do spacy, tendo a melhor precisão de todos os modelos e a melhor velocidade, com 3 minutos para processar o texto e baixo tempo de execução, comparado aos 20 minutos de processamento do spacy, e menor tempo de execução em relação ao LGBM.

O pior modelo foi o classificador LGBM,mesmo sendo uma otima biblioteca para outros tipos de classificação, que envolvem dados categoricos por exemplo.