Etapa 1

Importação de bibliotecas

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from math import sqrt, ceil
```

Importação de Dados

In [2]:
```
df_raw = pd.read_csv('https://raw.githubusercontent.com/gabrielramos731/Data-science-pro
```

In [3]:
```
df = df_raw.copy()
df = df.drop(['Unnamed: 0', 'id'], axis=1)
```

In [4]:
```
df['Type of Travel'] = df['Type of Travel'].str.replace('Business travel','Work travel')
df.head()
```

Out[4]:

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	entert
0	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	...	
1	Male	disloyal Customer	25	Work travel	Business	235	3	2	3	3	...	
2	Female	Loyal Customer	26	Work travel	Business	1142	2	2	2	2	...	
3	Female	Loyal Customer	25	Work travel	Business	562	2	5	5	5	...	
4	Male	Loyal Customer	61	Work travel	Business	214	3	3	3	3	...	

5 rows × 23 columns

Documentação dos Dados

Gender: Gender of the passengers (Female, Male)

Customer Type: The customer type (Loyal customer, disloyal customer)

Age: The actual age of the passengers

Type of Travel: Purpose of the flight of the passengers (Personal Travel, Business Travel)

Class: Travel class in the plane of the passengers (Business, Eco, Eco Plus)

Flight distance: The flight distance of this journey

Inflight wifi service: Satisfaction level of the inflight wifi service (0:Not Applicable;1-5)

Departure/Arrival time convenient: Satisfaction level of Departure/Arrival time convenient

Ease of Online booking: Satisfaction level of online booking

Gate location: Satisfaction level of Gate location

Food and drink: Satisfaction level of Food and drink

Online boarding: Satisfaction level of online boarding

Seat comfort: Satisfaction level of Seat comfort

Inflight entertainment: Satisfaction level of inflight entertainment

On-board service: Satisfaction level of On-board service

Leg room service: Satisfaction level of Leg room service

Baggage handling: Satisfaction level of baggage handling

Check-in service: Satisfaction level of Check-in service

Inflight service: Satisfaction level of inflight service

Cleanliness: Satisfaction level of Cleanliness

Departure Delay in Minutes: Minutes delayed when departure

Arrival Delay in Minutes: Minutes delayed when Arrival

Satisfaction: Airline satisfaction level(Satisfaction, neutral or dissatisfaction)

Análise de Dados Univariável

In [5]:
```
df.dtypes
```

Out[5]:
```
Gender                                object
Customer Type                         object
Age                                    int64
Type of Travel                        object
Class                                 object
Flight Distance                        int64
Inflight wifi service                  int64
Departure/Arrival time convenient      int64
Ease of Online booking                 int64
Gate location                          int64
Food and drink                         int64
Online boarding                        int64
Seat comfort                           int64
Inflight entertainment                 int64
On-board service                       int64
Leg room service                       int64
Baggage handling                       int64
Checkin service                        int64
Inflight service                       int64
Cleanliness                            int64
Departure Delay in Minutes             int64
```

```
Arrival Delay in Minutes                 float64
satisfaction                              object
dtype: object
```

In [6]:
```
var_cat = []
var_num = []
for i,j in zip(df.columns, df.dtypes):
  if j == 'object':
    var_cat.append(i)
  else:
    var_num.append(i)
```

Valores Duplicados

In [7]:
```
df.duplicated().sum()
```

Out[7]:
```
0
```

Valores Nulos

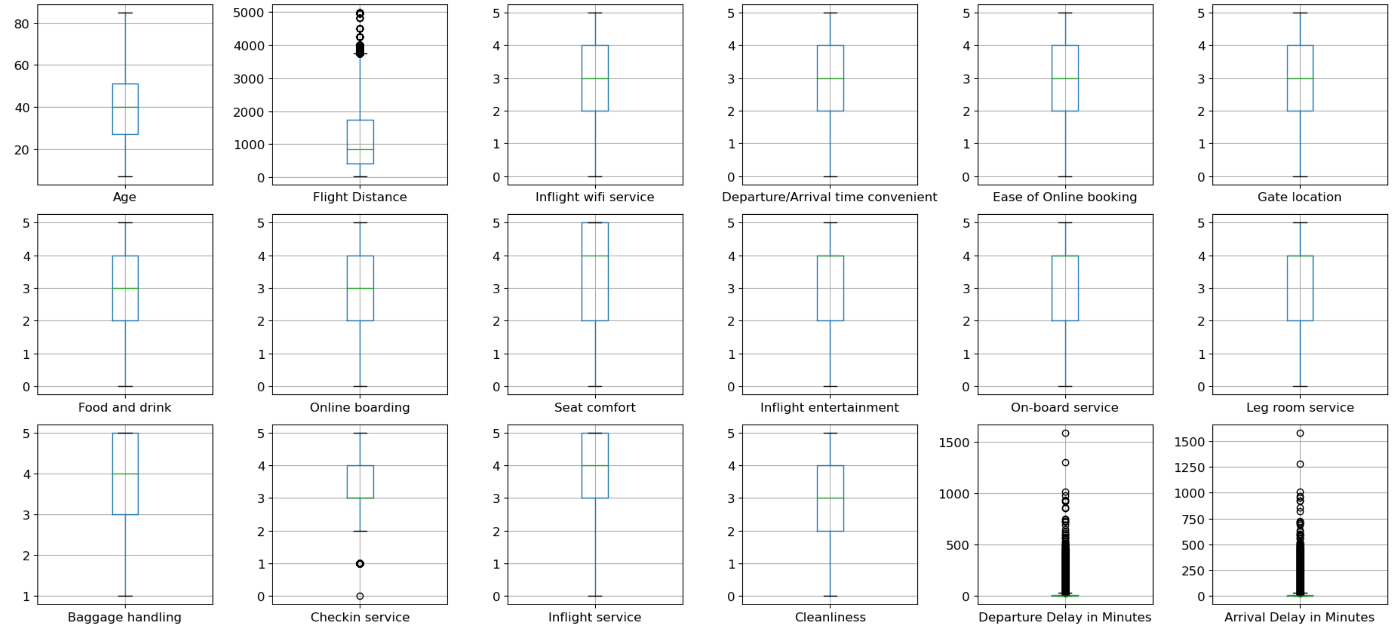
In [8]:
```
df.isna().sum()
```

Out[8]:
```
Gender                                 0
Customer Type                          0
Age                                    0
Type of Travel                         0
Class                                  0
Flight Distance                        0
Inflight wifi service                  0
Departure/Arrival time convenient      0
Ease of Online booking                 0
Gate location                          0
Food and drink                         0
Online boarding                        0
Seat comfort                           0
Inflight entertainment                 0
On-board service                       0
Leg room service                       0
Baggage handling                       0
Checkin service                        0
Inflight service                       0
Cleanliness                            0
Departure Delay in Minutes             0
Arrival Delay in Minutes             310
satisfaction                           0
dtype: int64
```

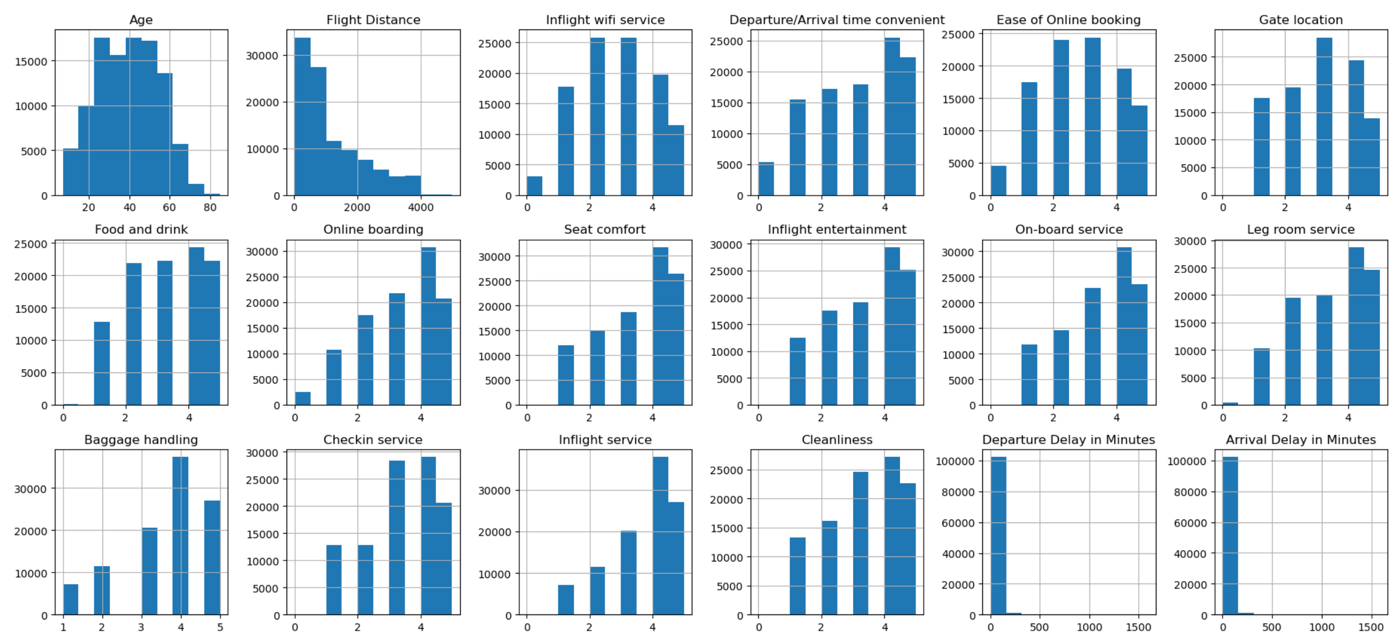
In [9]:
```
df = df.dropna()
```

Box-plots

In [10]:
```
fig, axes = plt.subplots(3, 6)

for i, el in enumerate(var_num):
  a = df.boxplot(el, ax=axes.flatten()[i], fontsize='large')

fig.set_size_inches(18.5, 8.5)
plt.tight_layout()
plt.show()
```

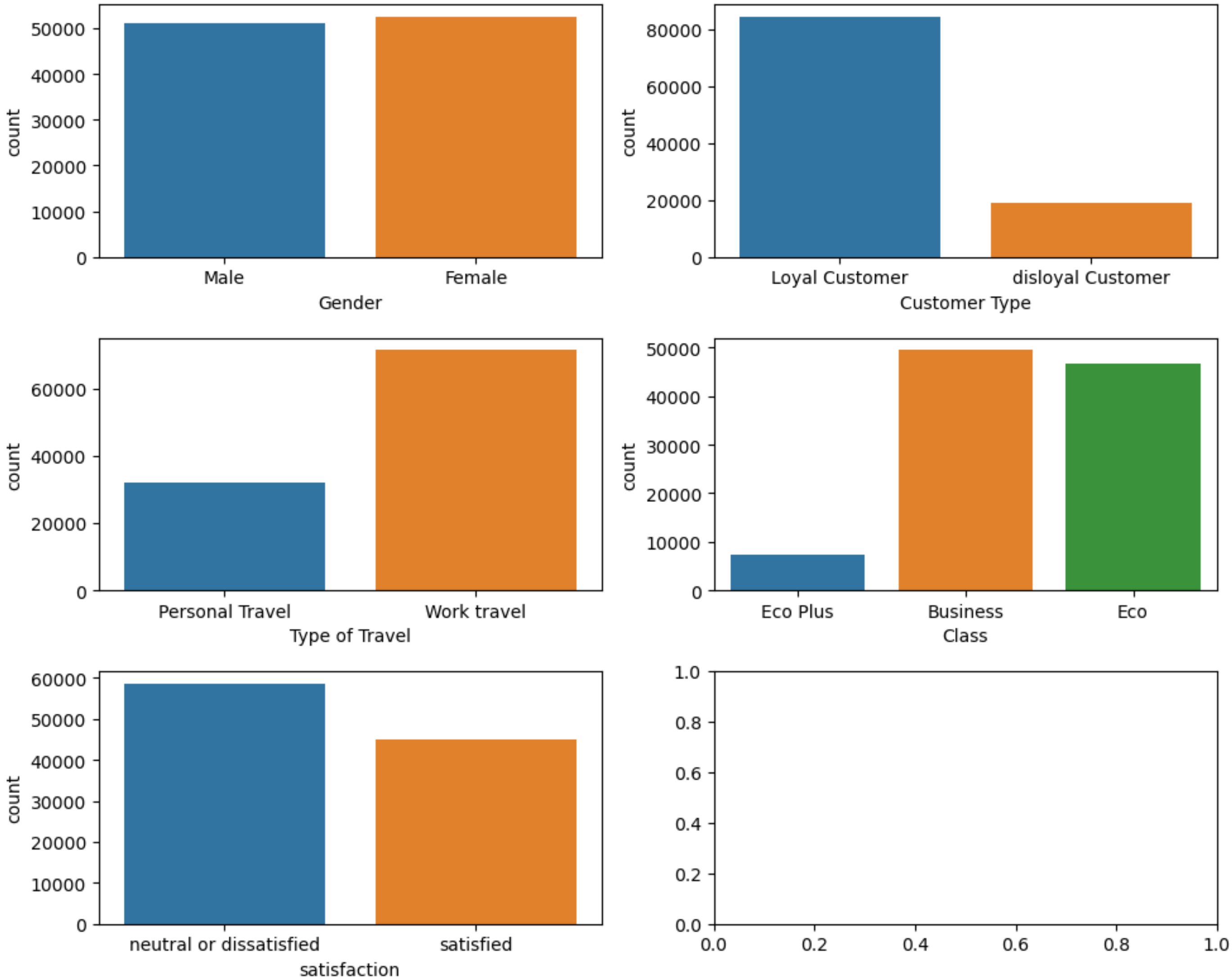


In [11]:
```python
fig, axes = plt.subplots(3, 6)

for i, el in enumerate(var_num):
  a = df.hist(el, ax=axes.flatten()[i])

fig.set_size_inches(18.5, 8.5)
plt.tight_layout()
plt.show()
```



Gráficos de Contagem

In [12]:
```python
fig, axes = plt.subplots(3,2, figsize=(10, 8))

for i, el in enumerate(var_cat):
  sns.countplot(x=df[el], ax=axes.flatten()[i])

  plt.tight_layout()
```



Hipótesis

1. O quão a classe influencia na satisfação do cliente

2. Qual tipo de viagem impacta mais na satisfação

3. O atraso é a variável de maior impacto na satisfação (primeiro modelo)?

Análise de dados bivariável

Influência da classe na satisfação

In [13]:
```
df_grp = df.groupby('Class')['satisfaction'].value_counts().to_frame()

ls = []
for x in df['Class'].value_counts().values:
  ls.append(x)
  ls.append(x)

df_grp['proporcao'] = df_grp['satisfaction'] / ls * 100
df_grp.sort_index(ascending=False, inplace=True)
df_grp
```

Out[13]:

	satisfaction	proporcao

Class	satisfaction		
Eco Plus	**satisfied**	1836	24.584896
	neutral or dissatisfied	5632	75.415104
Eco	**satisfied**	8671	18.610092
	neutral or dissatisfied	37922	81.389908
Business	**satisfied**	34390	69.428462
	neutral or dissatisfied	15143	30.571538

Tipo de viagem e impacto na satisfação

In [14]:
```
df_grp = df.groupby('Type of Travel')['satisfaction'].value_counts().to_frame()

ls = []
for x in df.groupby('Type of Travel')['Type of Travel'].value_counts().values:
  ls.append(x)
  ls.append(x)

df_grp['proporcao'] = df_grp['satisfaction'] / ls *100
df_grp.sort_index(ascending=False, inplace=True)
df_grp
```

Out[14]:

Type of Travel	satisfaction	satisfaction	proporcao
Work travel	**satisfied**	41634	58.257888
	neutral or dissatisfied	29831	41.742112
Personal Travel	**satisfied**	3263	10.155934
	neutral or dissatisfied	28866	89.844066

A justificativa para diferença tão grande na satisfação pelo tipo de viagem.

In [15]:
```
df.groupby('Type of Travel')[['Flight Distance',
                               'Leg room service',
                               'Departure Delay in Minutes',
                               'Departure/Arrival time convenient']].mean()
```

Out[15]:

Type of Travel	Flight Distance	Leg room service	Departure Delay in Minutes	Departure/Arrival time convenient
Personal Travel	791.240375	3.079336	14.404214	3.651125
Work travel	1368.294872	3.473714	14.902470	2.794361

In [16]:
```
df.groupby('Class')[['Flight Distance',
                      'Leg room service',
                      'Departure Delay in Minutes',
                      'Departure/Arrival time convenient']].mean()
```

Out[16]:

	Flight Distance	Leg room service	Departure Delay in Minutes	Departure/Arrival time convenient

Class				
Business	1676.078493	3.644661	14.335554	2.905820
Eco	742.843281	3.086129	15.093147	3.199043
Eco Plus	746.446438	3.061328	15.329405	3.216256

Impacto de classe e viagem na satisfação

In [17]:
```
df_grp2 = df.groupby(['Type of Travel','Class'])['satisfaction'].value_counts().to_frame
```

In [18]:
```
ls_travels = ['Personal Travel','Work travel']
ls_class = ['Business','Eco', 'Eco Plus']
ls = []

for travel in ls_travels:
    for i_class in ls_class:
        ls.append(df[(df['Type of Travel'] == travel) & (df['Class'] == i_class)].shape[
        ls.append(df[(df['Type of Travel'] == travel) & (df['Class'] == i_class)].shape[

df_grp2['proporcao'] = df_grp2['satisfaction']/ls *100
df_grp2.sort_index(ascending=False, inplace=True)
df_grp2
```

Out[18]:

			satisfaction	proporcao
Type of Travel	**Class**	**satisfaction**		
Work travel	**Eco Plus**	**satisfied**	1525	39.314256
		neutral or dissatisfied	2354	60.685744
	Eco	**satisfied**	5983	29.615880
		neutral or dissatisfied	14219	70.384120
	Business	**satisfied**	34126	72.020091
		neutral or dissatisfied	13258	27.979909
Personal Travel	**Eco Plus**	**satisfied**	311	8.665366
		neutral or dissatisfied	3278	91.334634
	Eco	**satisfied**	2688	10.185290
		neutral or dissatisfied	23703	89.814710
	Business	**satisfied**	264	12.284784
		neutral or dissatisfied	1885	87.715216

É possível ver que em uma mesma classe, viagens a trabalho possuem um índice de satisfação superior comparado a viagens pessoais.

Etapa 2

Tratamento das variáveis categóricas

In [19]:
```
!pip install category_encoders
```

```
Requirement already satisfied: category_encoders in c:\users\ramos\anaconda3\lib\site-packages (2.6.1)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\ramos\anaconda3\lib\site-packages (from category_encoders) (1.0.2)
Requirement already satisfied: pandas>=1.0.5 in c:\users\ramos\anaconda3\lib\site-packages (from category_encoders) (1.4.4)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\ramos\anaconda3\lib\site-packages (from category_encoders) (0.13.2)
Requirement already satisfied: numpy>=1.14.0 in c:\users\ramos\anaconda3\lib\site-packages (from category_encoders) (1.21.5)
Requirement already satisfied: patsy>=0.5.1 in c:\users\ramos\anaconda3\lib\site-packages (from category_encoders) (0.5.2)
Requirement already satisfied: scipy>=1.0.0 in c:\users\ramos\anaconda3\lib\site-packages (from category_encoders) (1.9.1)
Requirement already satisfied: pytz>=2020.1 in c:\users\ramos\anaconda3\lib\site-packages (from pandas>=1.0.5->category_encoders) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\ramos\anaconda3\lib\site-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: six in c:\users\ramos\anaconda3\lib\site-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=0.11 in c:\users\ramos\anaconda3\lib\site-packages (from scikit-learn>=0.20.0->category_encoders) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ramos\anaconda3\lib\site-packages (from scikit-learn>=0.20.0->category_encoders) (2.2.0)
Requirement already satisfied: packaging>=21.3 in c:\users\ramos\anaconda3\lib\site-packages (from statsmodels>=0.9.0->category_encoders) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\ramos\anaconda3\lib\site-packages (from packaging>=21.3->statsmodels>=0.9.0->category_encoders) (3.0.9)
```

In [20]:

```python
from category_encoders.one_hot import OneHotEncoder, OrdinalEncoder
from sklearn import preprocessing
```

Map para Class

In [21]:

```python
df_teste = df.copy()
df_teste['Class'] = df_teste['Class'].map({'Business':3, 'Eco Plus':2, 'Eco':1})
df_teste.head(10)
```

Out[21]:

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	In entertair
0	Male	Loyal Customer	13	Personal Travel	2	460	3	4	3	1	...	
1	Male	disloyal Customer	25	Work travel	3	235	3	2	3	3	...	
2	Female	Loyal Customer	26	Work travel	3	1142	2	2	2	2	...	
3	Female	Loyal Customer	25	Work travel	3	562	2	5	5	5	...	
4	Male	Loyal Customer	61	Work travel	3	214	3	3	3	3	...	
5	Female	Loyal Customer	26	Personal Travel	1	1180	3	4	2	1	...	
6	Male	Loyal Customer	47	Personal Travel	1	1276	2	4	2	3	...	
7	Female	Loyal Customer	52	Work travel	3	2035	4	3	4	4	...	

8	Female	Loyal Customer	41	Work travel	3	853	1	2	2	2	...
9	Male	disloyal Customer	20	Work travel	1	1061	3	3	3	4	...

10 rows × 23 columns

Rótulos

1: Eco

2: Eco Plus

3: Business

Label Encoder

In [22]:
```
df_lb_enc = df_teste.copy()
colunas = ['Gender', 'Customer Type', 'Type of Travel', 'satisfaction']
le = preprocessing.LabelEncoder()
for i in colunas:
  df_lb_enc[i] = le.fit_transform(df_teste[i])
df_encod = df_lb_enc.copy()
df_encod.head()
```

Out[22]:

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Infli entertainm
0	1	0	13	0	2	460	3	4	3	1	...	
1	1	1	25	1	3	235	3	2	3	3	...	
2	0	0	26	1	3	1142	2	2	2	2	...	
3	0	0	25	1	3	562	2	5	5	5	...	
4	1	0	61	1	3	214	3	3	3	3	...	

5 rows × 23 columns

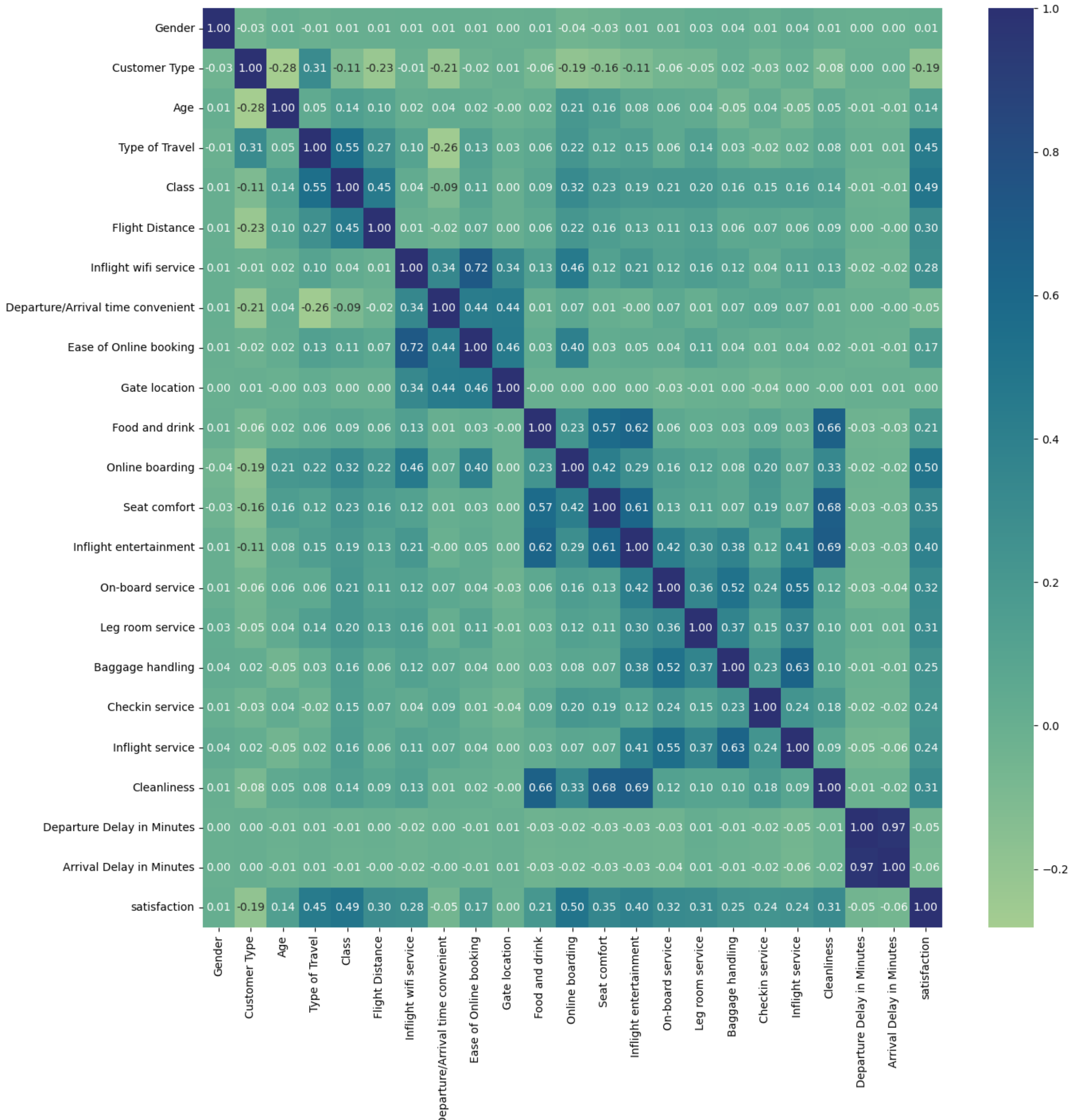
In [23]:
```
df_encod.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103594 entries, 0 to 103903
Data columns (total 23 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   Gender                             103594 non-null  int32
 1   Customer Type                      103594 non-null  int32
 2   Age                                103594 non-null  int64
 3   Type of Travel                     103594 non-null  int32
 4   Class                              103594 non-null  int64
 5   Flight Distance                    103594 non-null  int64
 6   Inflight wifi service              103594 non-null  int64
 7   Departure/Arrival time convenient  103594 non-null  int64
 8   Ease of Online booking             103594 non-null  int64
 9   Gate location                      103594 non-null  int64
 10  Food and drink                     103594 non-null  int64
 11  Online boarding                    103594 non-null  int64
```

```
 12  Seat comfort                       103594 non-null  int64
 13  Inflight entertainment             103594 non-null  int64
 14  On-board service                   103594 non-null  int64
 15  Leg room service                   103594 non-null  int64
 16  Baggage handling                   103594 non-null  int64
 17  Checkin service                    103594 non-null  int64
 18  Inflight service                   103594 non-null  int64
 19  Cleanliness                        103594 non-null  int64
 20  Departure Delay in Minutes         103594 non-null  int64
 21  Arrival Delay in Minutes           103594 non-null  float64
 22  satisfaction                       103594 non-null  int32
dtypes: float64(1), int32(4), int64(18)
memory usage: 17.4 MB
```

Correlações

In [24]:
```
plt.figure(figsize=(15,15));
sns.heatmap(df_encod.corr(), annot=True, fmt='.2f', cmap="crest");
```



Modelo de Classificação

In [25]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold, cross_val_score, train_test_split
from sklearn.metrics import confusion_matrix
```

Árvore de Decisão

In [26]:
```
from sklearn.tree import plot_tree
from sklearn.metrics import accuracy_score
```

In [56]:
```
def treinamento(meu_modelo, minha_max_depth=3, meu_classifier='DecisionTree', k_knn=5, m
```

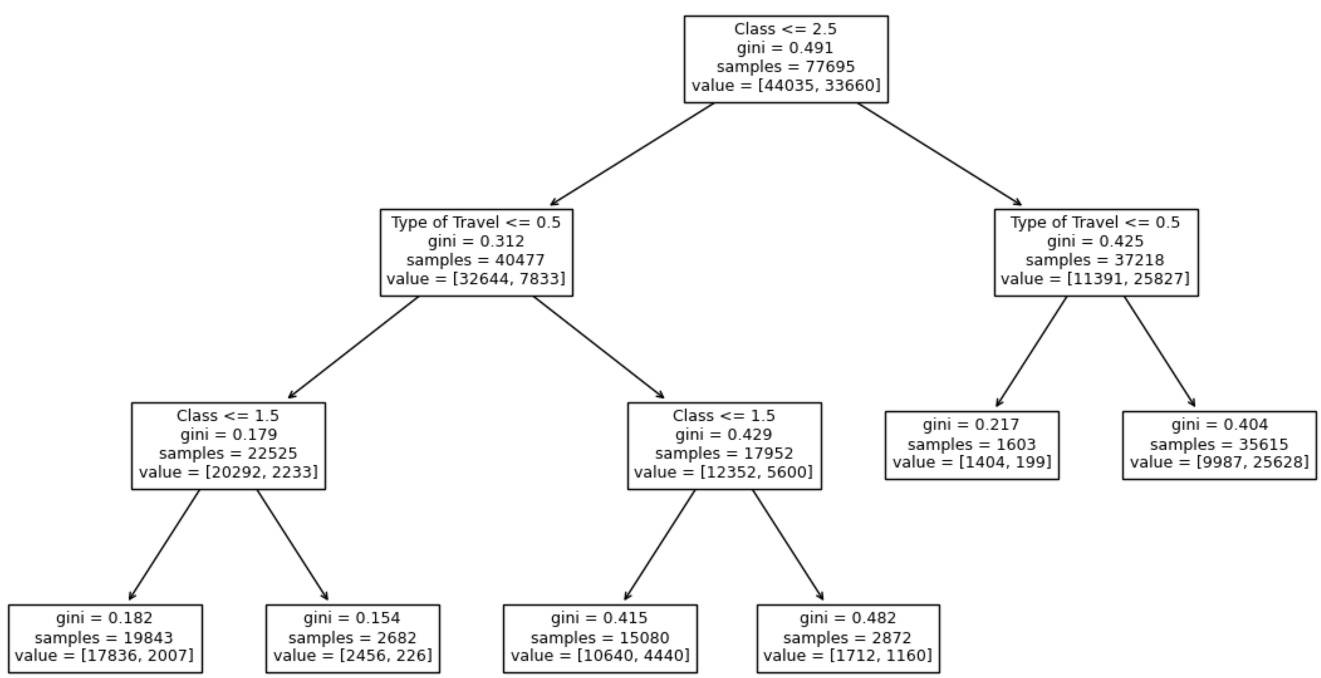
```
    if meu_X == None:
        meu_X = df_encod[meu_modelo]
    if meu_y == None:
        meu_y = df_encod['satisfaction']

    X_train, X_test, y_train, y_test = train_test_split(meu_X,
                                                        meu_y,
                                                        test_size=0.25,
                                                        random_state=42)
    if meu_classifier == 'DecisionTree':
        classifier = DecisionTreeClassifier(random_state=42, max_depth=minha_max_depth)
    # elif meu_classifier == 'KNN':
    else:
        classifier = KNeighborsClassifier(n_neighbors=k_knn)

    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)

    acuracia = classifier.score(X_test, y_test)

    return classifier, acuracia, y_pred, y_test
```

In [57]:
```
modelo = ['Type of Travel', 'Class']

classifier, acuracia, y_pred, y_test = treinamento(modelo)
print(acuracia)
```

```
0.7679447082898954
```

Plot Árvore de Decisão

In [58]:
```
plt.figure(figsize=(14,8), dpi=100)
plot_tree(classifier, feature_names=classifier.feature_names_in_, fontsize=9);
```



Validação Cruzada

In [59]:
```
import warnings
warnings.filterwarnings("ignore")
```

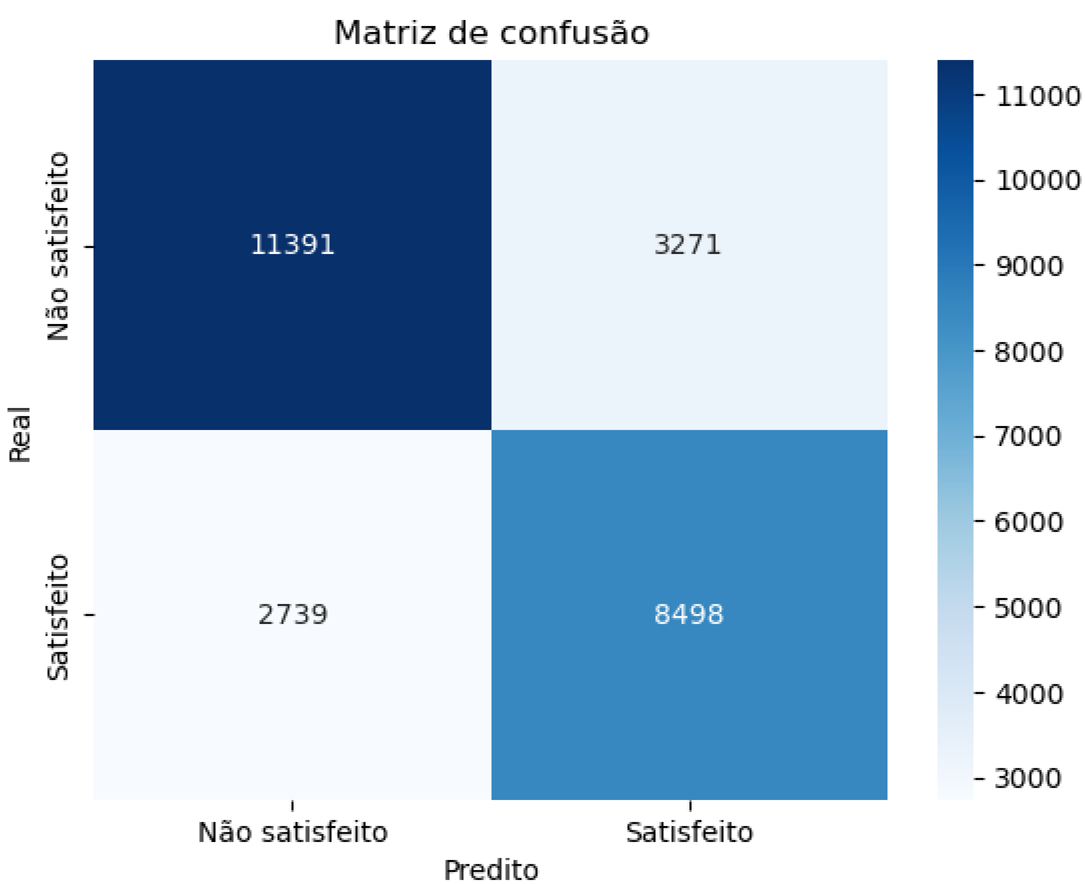
```
from sklearn.model_selection import StratifiedKFold
```

In [60]:
```
modelo = ['Type of Travel', 'Class']
X = df_encod[modelo]
y = df_encod['satisfaction']

cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

values = cross_val_score(classifier, X, y, cv=cv, scoring='accuracy')
print(f'media: {values.mean()}\nmin: {min(values)}\nmax: {max(values)}')
```

```
media: 0.768046470511718
min: 0.76496138996139
max: 0.7758470894874022
```

Matriz de Confusão

In [61]:
```
conf_mat_classifier = confusion_matrix(y_test, y_pred)
a = sns.heatmap(conf_mat_classifier, annot=True,
                cmap='Blues', fmt='.0f',
                xticklabels=['Não satisfeito', 'Satisfeito'],
                yticklabels=['Não satisfeito', 'Satisfeito'])
plt.xlabel('Predito')
plt.ylabel('Real')
plt.title('Matriz de confusão');
```



Teste da Hipótese - Variáveis Sozinhas

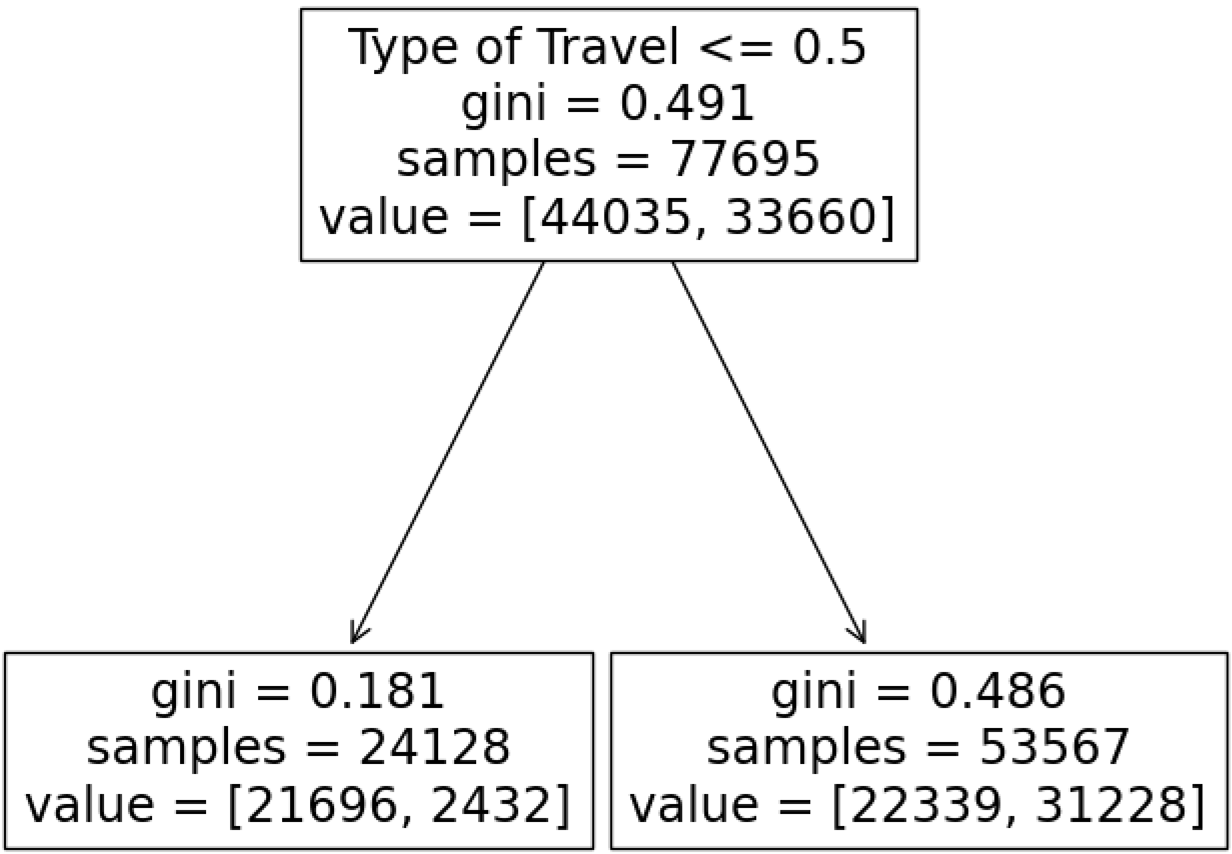
Apenas `Type of Travel`

In [62]:
```
modelo = ['Type of Travel']

classifier, _, y_pred, y_test = treinamento(modelo, 30)
```

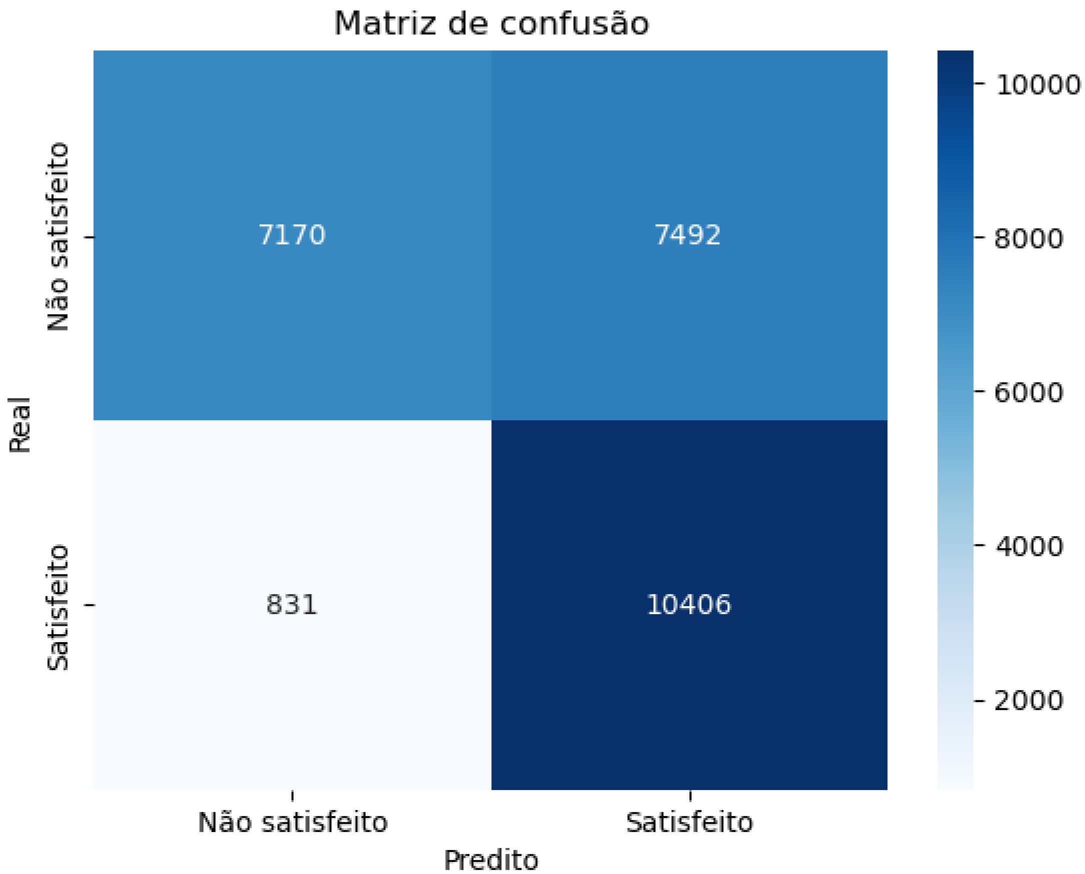
```python
X = df_encod[modelo]
y = df_encod['satisfaction']

cv = StratifiedKFold(n_splits = 10, shuffle=True, random_state=42)

values = cross_val_score(classifier, X, y, cv=cv, scoring='accuracy')
print(f'media: {values.mean()}\nmin: {min(values)}\nmax: {max(values)}')
```

```
media: 0.6805413437515957
min: 0.6744859542426875
max: 0.685490877497828
```

In [63]:
```python
plt.figure(figsize=(8,8), dpi=100)
plot_tree(classifier, feature_names=classifier.feature_names_in_);
```



In [64]:
```python
conf_mat_classifier = confusion_matrix(y_test, y_pred)
a = sns.heatmap(conf_mat_classifier, annot=True,
                cmap='Blues', fmt='.0f',
                xticklabels=['Não satisfeito', 'Satisfeito'],
                yticklabels=['Não satisfeito', 'Satisfeito'])
plt.xlabel('Predito')
plt.ylabel('Real')
plt.title('Matriz de confusão');
```

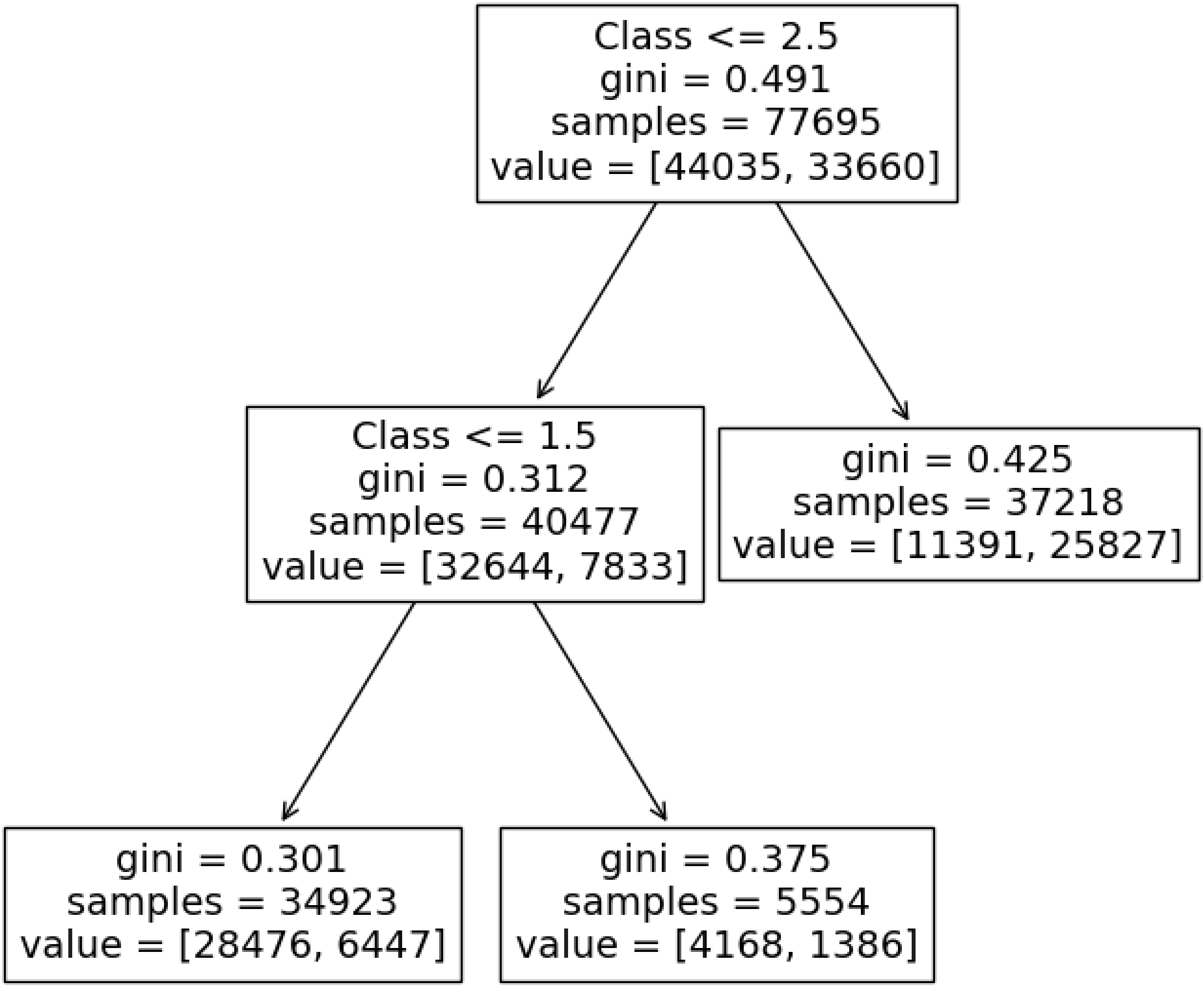


Apenas `Class`

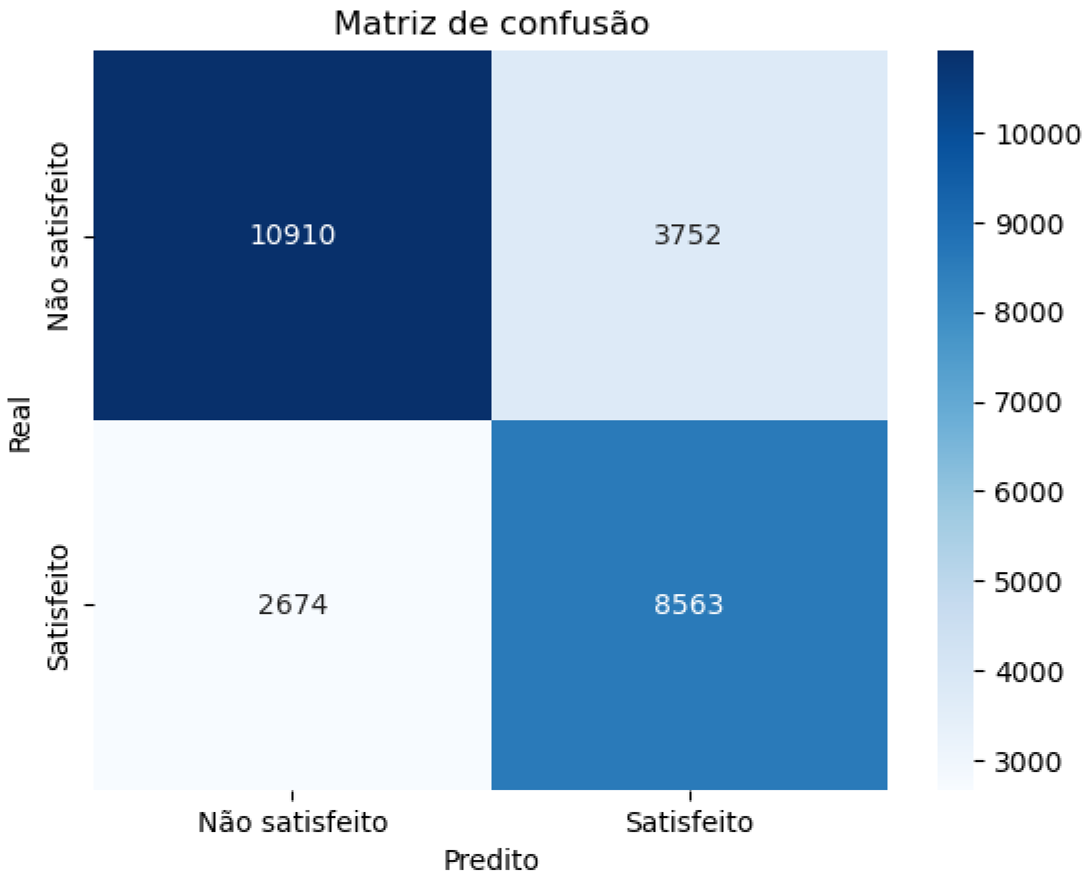
In [65]:
```
modelo = ['Class']

classifier, _, y_pred, y_test = treinamento(modelo)

X = df_encod[modelo]
y = df_encod['satisfaction']

cv = StratifiedKFold(n_splits = 10, shuffle=True, random_state=42)

values = cross_val_score(classifier, X, y, cv=cv, scoring='accuracy')
print(f'media: {values.mean()}\nmin: {min(values)}\nmax: {max(values)}')
```

```
media: 0.7523988298836256
min: 0.7495897287382952
max: 0.7621392026257361
```

In [66]:
```
plt.figure(figsize=(8,8), dpi=100)
plot_tree(classifier, feature_names=classifier.feature_names_in_);
```



In [67]:
```
conf_mat_classifier = confusion_matrix(y_test, y_pred)
a = sns.heatmap(conf_mat_classifier, annot=True,
                cmap='Blues', fmt='.0f',
                xticklabels=['Não satisfeito', 'Satisfeito'],
                yticklabels=['Não satisfeito', 'Satisfeito'])
plt.xlabel('Predito')
plt.ylabel('Real')
plt.title('Matriz de confusão');
```



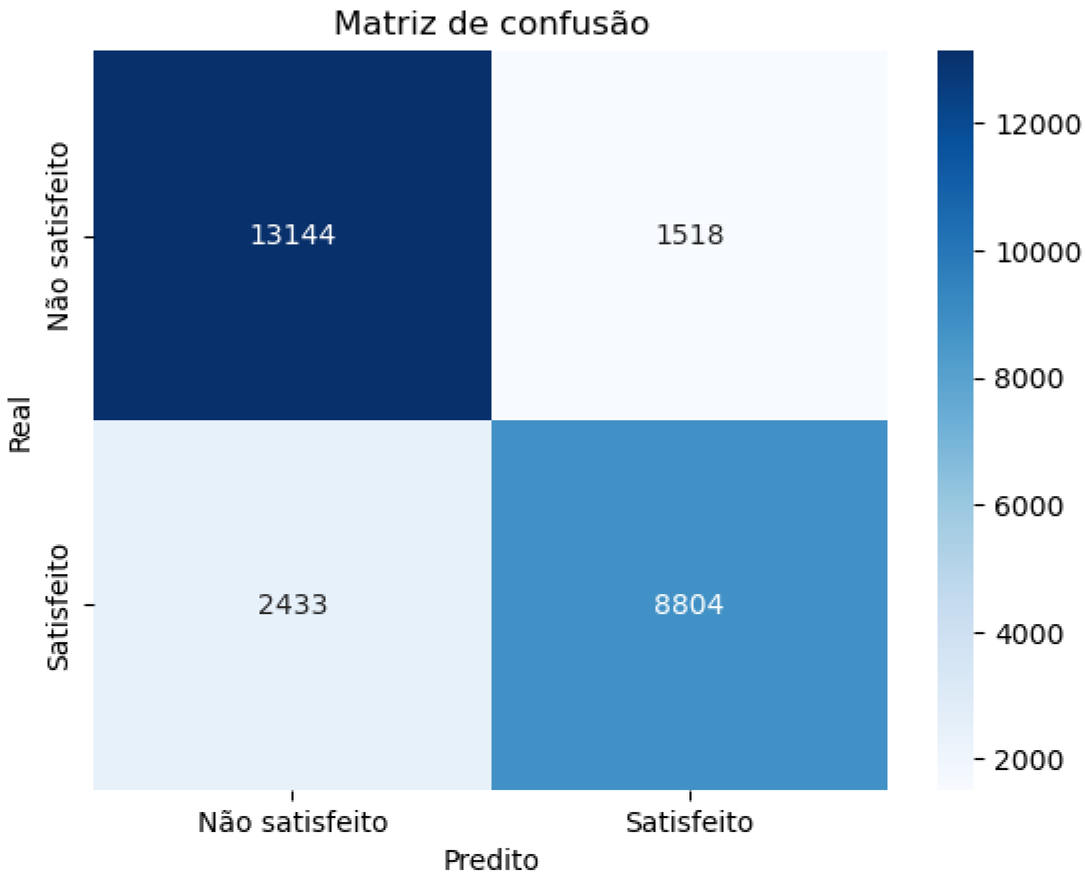
Melhorias da Etapa 2

Sugestão Caio e Talita

In [68]:
```
# Sugestão: Utilizar mais uma variável para a classificação: variável online boarding.

modelo = ['Type of Travel', 'Class', 'Online boarding']

classifier, _, y_pred, y_test = treinamento(modelo)

X = df_encod[modelo]
y = df_encod['satisfaction']

cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

values = cross_val_score(classifier, X, y, cv=cv, scoring='accuracy')
print(f'media: {values.mean()}\nmin: {min(values)}\nmax: {max(values)}')
```

```
media: 0.8464873036745322
min: 0.8422779922779923
max: 0.8526884834443479
```

In [69]:
```
conf_mat_classifier = confusion_matrix(y_test, y_pred)
a = sns.heatmap(conf_mat_classifier, annot=True,
                cmap='Blues', fmt='.0f',
                xticklabels=['Não satisfeito', 'Satisfeito'],
                yticklabels=['Não satisfeito', 'Satisfeito'])
plt.xlabel('Predito')
plt.ylabel('Real')
plt.title('Matriz de confusão');
```

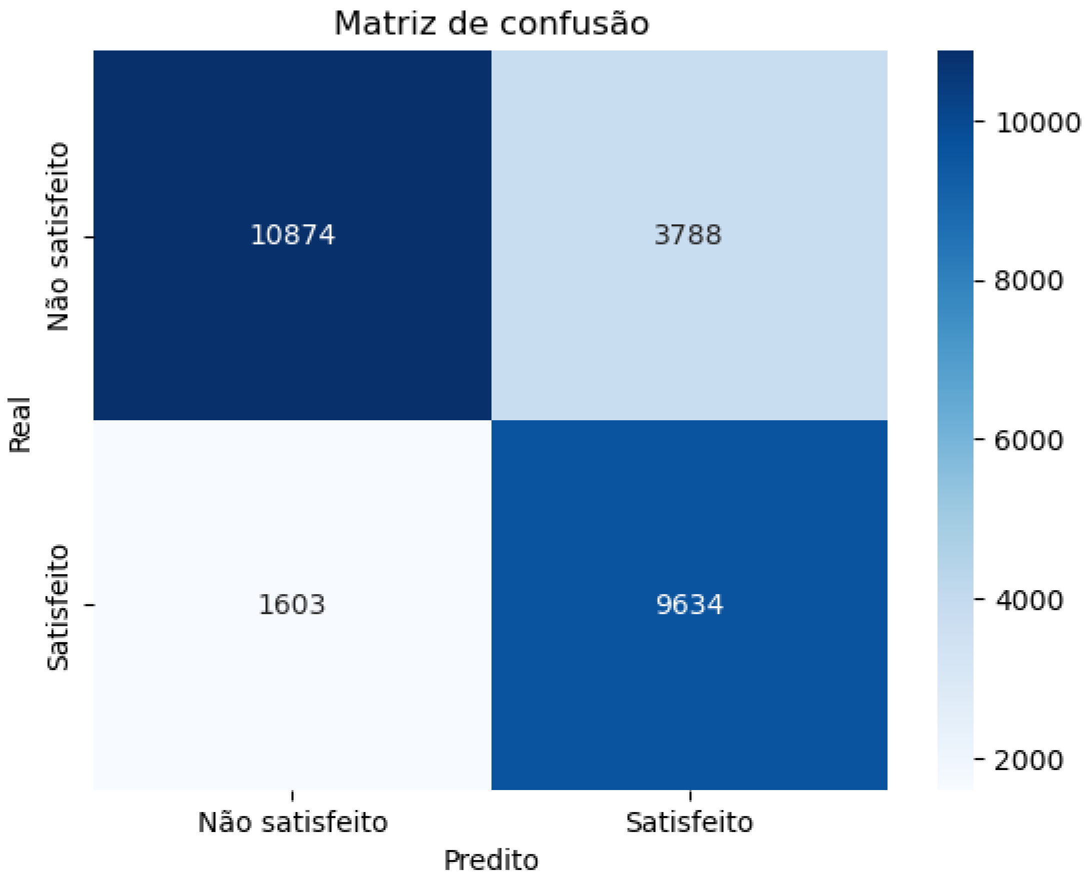


Apenas Online Boarding

In [70]:
```
modelo = ['Online boarding']

classifier, _, y_pred, y_test = treinamento(modelo)

X = df_encod[modelo]
y = df_encod['satisfaction']

cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

values = cross_val_score(classifier, X, y, cv=cv, scoring='accuracy')
print(f'media: {values.mean()}\nmin: {min(values)}\nmax: {max(values)}')
```

```
media: 0.7903354039778887
min: 0.7833011583011583
max: 0.7937059561733758
```

In [71]:
```
conf_mat_classifier = confusion_matrix(y_test, y_pred)
a = sns.heatmap(conf_mat_classifier, annot=True,
                cmap='Blues', fmt='.0f',
                xticklabels=['Não satisfeito', 'Satisfeito'],
                yticklabels=['Não satisfeito', 'Satisfeito'])
plt.xlabel('Predito')
plt.ylabel('Real')
plt.title('Matriz de confusão');
```



Sugestão Paulo Eduardo e Pedro Henrique

In [72]:
```
# Sugestão:
# Utilizar o modelo KNN com as características escolhidas para a hipótese: Type of Trave

modelo = ['Type of Travel', 'Class', 'Online boarding']

X = df_encod[modelo]
y = df_encod['satisfaction']

qtd_dados = df_encod.shape[0]

qtd_primos = ceil(sqrt(qtd_dados))

# Para otimizar o tempo de cálculo
qtd_primos = min(qtd_primos, 70)

primos = []
for i in range(2, qtd_primos):
  ctr_add = True
  raiz_i = ceil(sqrt(i))

  for j in primos:
    if j > raiz_i:
      break

    if i % j == 0:
      ctr_add = False
      break

  if ctr_add == True:
    primos.append(i)

resultados = []
```

```python
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
for n_vizinhos in primos:
  knn_classifier = KNeighborsClassifier(n_neighbors=n_vizinhos)

  values = cross_val_score(knn_classifier, X, y, cv=cv, scoring='accuracy')

  resultados.append(values.mean())

  print(f"{n_vizinhos} vizinhos: ", values.mean())

plt.plot(primos, resultados);
```

```
2 vizinhos:  0.8328475760730323
```

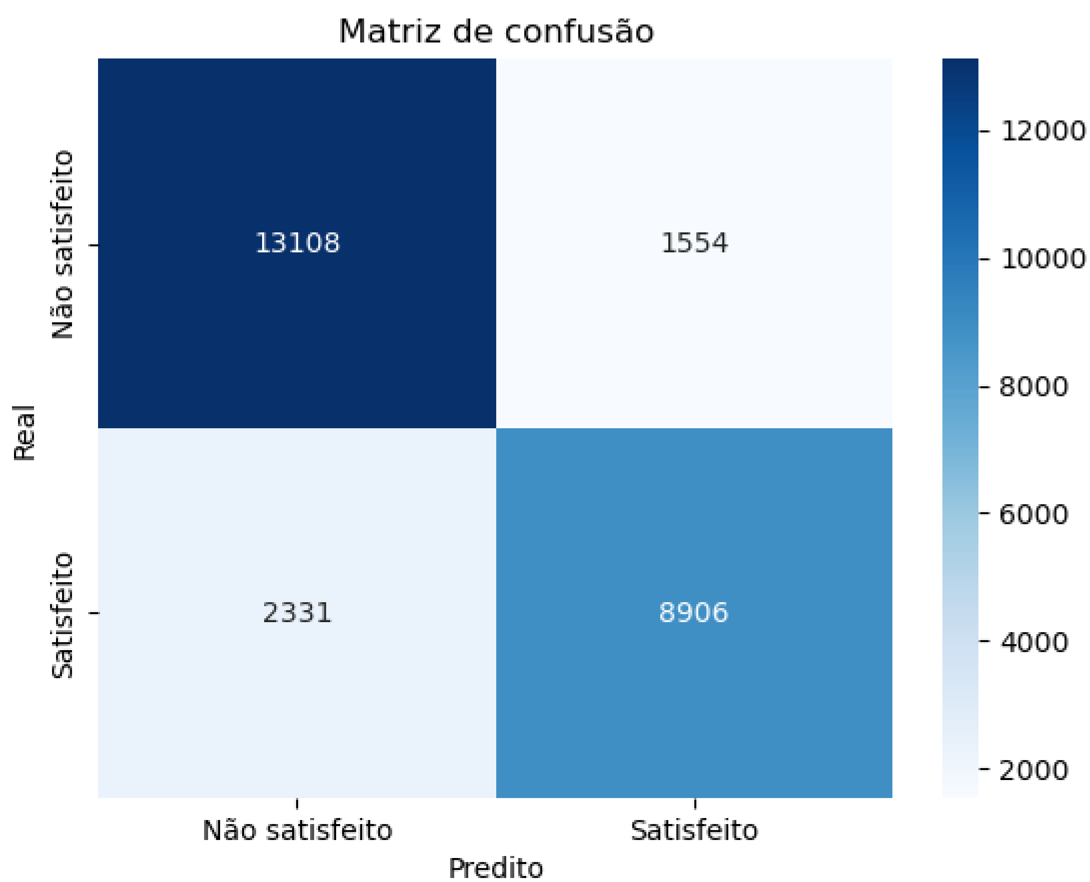
```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12700\2236793127.py in <module>
     36   knn_classifier = KNeighborsClassifier(n_neighbors=n_vizinhos)
     37 
---> 38   values = cross_val_score(knn_classifier, X, y, cv=cv, scoring='accuracy')
     39 
     40   resultados.append(values.mean())

~\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in cross_val_score(estimator, X, y, groups, scoring, cv, n_jobs, verbose, fit_params, pre_dispatch, error_score)
    507     scorer = check_scoring(estimator, scoring=scoring)
    508 
--> 509     cv_results = cross_validate(
    510         estimator=estimator,
    511         X=X,

~\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in cross_validate(estimator, X, y, groups, scoring, cv, n_jobs, verbose, fit_params, pre_dispatch, return_train_score, return_estimator, error_score)
    265     # independent, and that it is pickle-able.
    266     parallel = Parallel(n_jobs=n_jobs, verbose=verbose, pre_dispatch=pre_dispatch)
--> 267     results = parallel(
    268         delayed(_fit_and_score)(
    269             clone(estimator),

~\anaconda3\lib\site-packages\joblib\parallel.py in __call__(self, iterable)
   1044                 self._iterating = self._original_iterator is not None
   1045 
-> 1046             while self.dispatch_one_batch(iterator):
   1047                 pass
   1048 

~\anaconda3\lib\site-packages\joblib\parallel.py in dispatch_one_batch(self, iterator)
    859                 return False
    860             else:
--> 861                 self._dispatch(tasks)
    862                 return True
    863 

~\anaconda3\lib\site-packages\joblib\parallel.py in _dispatch(self, batch)
    777         with self._lock:
    778             job_idx = len(self._jobs)
--> 779             job = self._backend.apply_async(batch, callback=cb)
    780             # A job can complete so quickly than its callback is
    781             # called before we get here, causing self._jobs to

~\anaconda3\lib\site-packages\joblib\_parallel_backends.py in apply_async(self, func, callback)
    206     def apply_async(self, func, callback=None):
```

```
    207         """Schedule a func to be run"""
--> 208         result = ImmediateResult(func)
    209         if callback:
    210             callback(result)

~\anaconda3\lib\site-packages\joblib\_parallel_backends.py in __init__(self, batch)
    570         # Don't delay the application, to avoid keeping the input
    571         # arguments in memory
--> 572         self.results = batch()
    573 
    574     def get(self):

~\anaconda3\lib\site-packages\joblib\parallel.py in __call__(self)
    260         # change the default number of processes to -1
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 262             return [func(*args, **kwargs)
    263                     for func, args, kwargs in self.items]
    264 

~\anaconda3\lib\site-packages\joblib\parallel.py in <listcomp>(.0)
    260         # change the default number of processes to -1
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 262             return [func(*args, **kwargs)
    263                     for func, args, kwargs in self.items]
    264 

~\anaconda3\lib\site-packages\sklearn\utils\fixes.py in __call__(self, *args, **kwargs)
    214     def __call__(self, *args, **kwargs):
    215         with config_context(**self.config):
--> 216             return self.function(*args, **kwargs)
    217 
    218 

~\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in _fit_and_score(e
stimator, X, y, scorer, train, test, verbose, parameters, fit_params, return_train_scor
e, return_parameters, return_n_test_samples, return_times, return_estimator, split_progr
ess, candidate_progress, error_score)
    700 
    701         fit_time = time.time() - start_time
--> 702         test_scores = _score(estimator, X_test, y_test, scorer, error_score)
    703         score_time = time.time() - start_time - fit_time
    704         if return_train_score:

~\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in _score(estimato
r, X_test, y_test, scorer, error_score)
    759             scores = scorer(estimator, X_test)
    760         else:
--> 761             scores = scorer(estimator, X_test, y_test)
    762     except Exception:
    763         if error_score == "raise":

~\anaconda3\lib\site-packages\sklearn\metrics\_scorer.py in __call__(self, estimator, *a
rgs, **kwargs)
    101         for name, scorer in self._scorers.items():
    102             if isinstance(scorer, _BaseScorer):
--> 103                 score = scorer._score(cached_call, estimator, *args, **kwargs)
    104             else:
    105                 score = scorer(estimator, *args, **kwargs)

~\anaconda3\lib\site-packages\sklearn\metrics\_scorer.py in _score(self, method_caller,
 estimator, X, y_true, sample_weight)
    256         """
    257 
--> 258         y_pred = method_caller(estimator, "predict", X)
    259         if sample_weight is not None:
    260             return self._sign * self._score_func(
```

```
~\anaconda3\lib\site-packages\sklearn\metrics\_scorer.py in _cached_call(cache, estimato
r, method, *args, **kwargs)
     66     """Call estimator with method and args and kwargs."""
     67     if cache is None:
---> 68         return getattr(estimator, method)(*args, **kwargs)
     69 
     70     try:

~\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py in predict(self, X)
    212             Class labels for each data sample.
    213         """
--> 214         neigh_dist, neigh_ind = self.kneighbors(X)
    215         classes_ = self.classes_
    216         _y = self._y

~\anaconda3\lib\site-packages\sklearn\neighbors\_base.py in kneighbors(self, X, n_neighb
ors, return_distance)
    774             else:
    775                 parallel_kwargs = {"prefer": "threads"}
--> 776             chunked_results = Parallel(n_jobs, **parallel_kwargs)(
    777                 delayed(_tree_query_parallel_helper)(
    778                     self._tree, X[s], n_neighbors, return_distance

~\anaconda3\lib\site-packages\joblib\parallel.py in __call__(self, iterable)
   1041             # remaining jobs.
   1042             self._iterating = False
-> 1043             if self.dispatch_one_batch(iterator):
   1044                 self._iterating = self._original_iterator is not None
   1045 

~\anaconda3\lib\site-packages\joblib\parallel.py in dispatch_one_batch(self, iterator)
    859                 return False
    860             else:
--> 861                 self._dispatch(tasks)
    862                 return True
    863 

~\anaconda3\lib\site-packages\joblib\parallel.py in _dispatch(self, batch)
    777         with self._lock:
    778             job_idx = len(self._jobs)
--> 779             job = self._backend.apply_async(batch, callback=cb)
    780             # A job can complete so quickly than its callback is
    781             # called before we get here, causing self._jobs to

~\anaconda3\lib\site-packages\joblib\_parallel_backends.py in apply_async(self, func, ca
llback)
    206     def apply_async(self, func, callback=None):
    207         """Schedule a func to be run"""
--> 208         result = ImmediateResult(func)
    209         if callback:
    210             callback(result)

~\anaconda3\lib\site-packages\joblib\_parallel_backends.py in __init__(self, batch)
    570         # Don't delay the application, to avoid keeping the input
    571         # arguments in memory
--> 572         self.results = batch()
    573 
    574     def get(self):

~\anaconda3\lib\site-packages\joblib\parallel.py in __call__(self)
    260         # change the default number of processes to -1
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 262             return [func(*args, **kwargs)
    263                     for func, args, kwargs in self.items]
    264 
```

```
~\anaconda3\lib\site-packages\joblib\parallel.py in <listcomp>(.0)
    260         # change the default number of processes to -1
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 262             return [func(*args, **kwargs)
    263                     for func, args, kwargs in self.items]
    264 

~\anaconda3\lib\site-packages\sklearn\utils\fixes.py in __call__(self, *args, **kwargs)
    214     def __call__(self, *args, **kwargs):
    215         with config_context(**self.config):
--> 216             return self.function(*args, **kwargs)
    217 
    218 

~\anaconda3\lib\site-packages\sklearn\neighbors\_base.py in _tree_query_parallel_helper
(tree, *args, **kwargs)
    598     under PyPy.
    599     """
--> 600     return tree.query(*args, **kwargs)
    601 
    602 

KeyboardInterrupt: 
```

In [73]:
```
classifier, _, y_pred, y_test = treinamento(meu_modelo=modelo, meu_classifier='KNN', k_k

conf_mat_classifier = confusion_matrix(y_test, y_pred)
a = sns.heatmap(conf_mat_classifier, annot=True,
                cmap='Blues', fmt='.0f',
                xticklabels=['Não satisfeito', 'Satisfeito'],
                yticklabels=['Não satisfeito', 'Satisfeito'])
plt.xlabel('Predito')
plt.ylabel('Real')
plt.title('Matriz de confusão');
```



Etapa 3

O melhor modelo até o momento foi KNN, com ênfase nas escolhas com pelo menos 29 vizinhos.

Portanto, este será o modelo testado na etapa 3.

Todas as variáveis

In [74]:

```
# Obtém todas as colunas que não são `satisfaction`
modelo = list()
modelo.extend( [coluna for coluna in df_encod.columns if coluna != 'satisfaction'] )

X = df_encod[modelo]
y = df_encod['satisfaction']

qtd_dados = df_encod.shape[0]
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

qtd_primos = ceil(sqrt(qtd_dados))

# Pra otimizar o tempo de cálculo
qtd_primos = min(qtd_primos, 70)

primos = []
for i in range(2, qtd_primos):
  ctr_add = True
  raiz_i = ceil(sqrt(i))

  for j in primos:
    if j > raiz_i:
      break

    if i % j == 0:
      ctr_add = False
      break

  if ctr_add == True:
    primos.append(i)

resultados = []

for n_vizinhos in primos:
  knn_classifier = KNeighborsClassifier(n_neighbors=n_vizinhos)

  values = cross_val_score(knn_classifier, X, y, cv=cv, scoring='accuracy')

  resultados.append(values.mean())

  print(f"{n_vizinhos} vizinhos: ", values.mean())

plt.plot(primos, resultados);
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12700\2173217495.py in <module>
     35   knn_classifier = KNeighborsClassifier(n_neighbors=n_vizinhos)
     36 
---> 37   values = cross_val_score(knn_classifier, X, y, cv=cv, scoring='accuracy')
     38 
     39   resultados.append(values.mean())

~\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in cross_val_score
```

```
(estimator, X, y, groups, scoring, cv, n_jobs, verbose, fit_params, pre_dispatch, error_
score)
    507     scorer = check_scoring(estimator, scoring=scoring)
    508 
--> 509     cv_results = cross_validate(
    510         estimator=estimator,
    511         X=X,

~\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in cross_validate(e
stimator, X, y, groups, scoring, cv, n_jobs, verbose, fit_params, pre_dispatch, return_t
rain_score, return_estimator, error_score)
    265     # independent, and that it is pickle-able.
    266     parallel = Parallel(n_jobs=n_jobs, verbose=verbose, pre_dispatch=pre_dispatc
h)
--> 267     results = parallel(
    268         delayed(_fit_and_score)(
    269             clone(estimator),

~\anaconda3\lib\site-packages\joblib\parallel.py in __call__(self, iterable)
   1044                 self._iterating = self._original_iterator is not None
   1045 
-> 1046             while self.dispatch_one_batch(iterator):
   1047                 pass
   1048 

~\anaconda3\lib\site-packages\joblib\parallel.py in dispatch_one_batch(self, iterator)
    859                 return False
    860             else:
--> 861                 self._dispatch(tasks)
    862                 return True
    863 

~\anaconda3\lib\site-packages\joblib\parallel.py in _dispatch(self, batch)
    777         with self._lock:
    778             job_idx = len(self._jobs)
--> 779             job = self._backend.apply_async(batch, callback=cb)
    780             # A job can complete so quickly than its callback is
    781             # called before we get here, causing self._jobs to

~\anaconda3\lib\site-packages\joblib\_parallel_backends.py in apply_async(self, func, ca
llback)
    206     def apply_async(self, func, callback=None):
    207         """Schedule a func to be run"""
--> 208         result = ImmediateResult(func)
    209         if callback:
    210             callback(result)

~\anaconda3\lib\site-packages\joblib\_parallel_backends.py in __init__(self, batch)
    570         # Don't delay the application, to avoid keeping the input
    571         # arguments in memory
--> 572         self.results = batch()
    573 
    574     def get(self):

~\anaconda3\lib\site-packages\joblib\parallel.py in __call__(self)
    260         # change the default number of processes to -1
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 262             return [func(*args, **kwargs)
    263                     for func, args, kwargs in self.items]
    264 

~\anaconda3\lib\site-packages\joblib\parallel.py in <listcomp>(.0)
    260         # change the default number of processes to -1
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 262             return [func(*args, **kwargs)
    263                     for func, args, kwargs in self.items]
```

```
    264

~\anaconda3\lib\site-packages\sklearn\utils\fixes.py in __call__(self, *args, **kwargs)
    214     def __call__(self, *args, **kwargs):
    215         with config_context(**self.config):
--> 216             return self.function(*args, **kwargs)
    217
    218

~\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in _fit_and_score(e
stimator, X, y, scorer, train, test, verbose, parameters, fit_params, return_train_scor
e, return_parameters, return_n_test_samples, return_times, return_estimator, split_progr
ess, candidate_progress, error_score)
    700
    701         fit_time = time.time() - start_time
--> 702         test_scores = _score(estimator, X_test, y_test, scorer, error_score)
    703         score_time = time.time() - start_time - fit_time
    704         if return_train_score:

~\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in _score(estimato
r, X_test, y_test, scorer, error_score)
    759             scores = scorer(estimator, X_test)
    760         else:
--> 761             scores = scorer(estimator, X_test, y_test)
    762     except Exception:
    763         if error_score == "raise":

~\anaconda3\lib\site-packages\sklearn\metrics\_scorer.py in __call__(self, estimator, *a
rgs, **kwargs)
    101         for name, scorer in self._scorers.items():
    102             if isinstance(scorer, _BaseScorer):
--> 103                 score = scorer._score(cached_call, estimator, *args, **kwargs)
    104             else:
    105                 score = scorer(estimator, *args, **kwargs)

~\anaconda3\lib\site-packages\sklearn\metrics\_scorer.py in _score(self, method_caller,
 estimator, X, y_true, sample_weight)
    256         """
    257
--> 258         y_pred = method_caller(estimator, "predict", X)
    259         if sample_weight is not None:
    260             return self._sign * self._score_func(

~\anaconda3\lib\site-packages\sklearn\metrics\_scorer.py in _cached_call(cache, estimato
r, method, *args, **kwargs)
     66     """Call estimator with method and args and kwargs."""
     67     if cache is None:
---> 68         return getattr(estimator, method)(*args, **kwargs)
     69
     70     try:

~\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py in predict(self, X)
    212             Class labels for each data sample.
    213         """
--> 214         neigh_dist, neigh_ind = self.kneighbors(X)
    215         classes_ = self.classes_
    216         _y = self._y

~\anaconda3\lib\site-packages\sklearn\neighbors\_base.py in kneighbors(self, X, n_neighb
ors, return_distance)
    750                 kwds = self.effective_metric_params_
    751
--> 752             chunked_results = list(
    753                 pairwise_distances_chunked(
    754                     X,
```

```
~\anaconda3\lib\site-packages\sklearn\metrics\pairwise.py in pairwise_distances_chunked
(X, Y, reduce_func, metric, n_jobs, working_memory, **kwds)
   1724         if reduce_func is not None:
   1725             chunk_size = D_chunk.shape[0]
-> 1726             D_chunk = reduce_func(D_chunk, sl.start)
   1727             _check_chunk_size(D_chunk, chunk_size)
   1728         yield D_chunk

~\anaconda3\lib\site-packages\sklearn\neighbors\_base.py in _kneighbors_reduce_func(sel
f, dist, start, n_neighbors, return_distance)
    632         """
    633         sample_range = np.arange(dist.shape[0])[:, None]
--> 634         neigh_ind = np.argpartition(dist, n_neighbors - 1, axis=1)
    635         neigh_ind = neigh_ind[:, :n_neighbors]
    636         # argpartition doesn't guarantee sorted order, so we sort again

<__array_function__ internals> in argpartition(*args, **kwargs)

~\anaconda3\lib\site-packages\numpy\core\fromnumeric.py in argpartition(a, kth, axis, ki
nd, order)
    837
    838     """
--> 839     return _wrapfunc(a, 'argpartition', kth, axis=axis, kind=kind, order=order)
    840
    841

~\anaconda3\lib\site-packages\numpy\core\fromnumeric.py in _wrapfunc(obj, method, *args,
 **kwds)
     55
     56     try:
---> 57         return bound(*args, **kwds)
     58     except TypeError:
     59         # A TypeError occurs if the object does have such a method in its

KeyboardInterrupt:
```

Por correlação

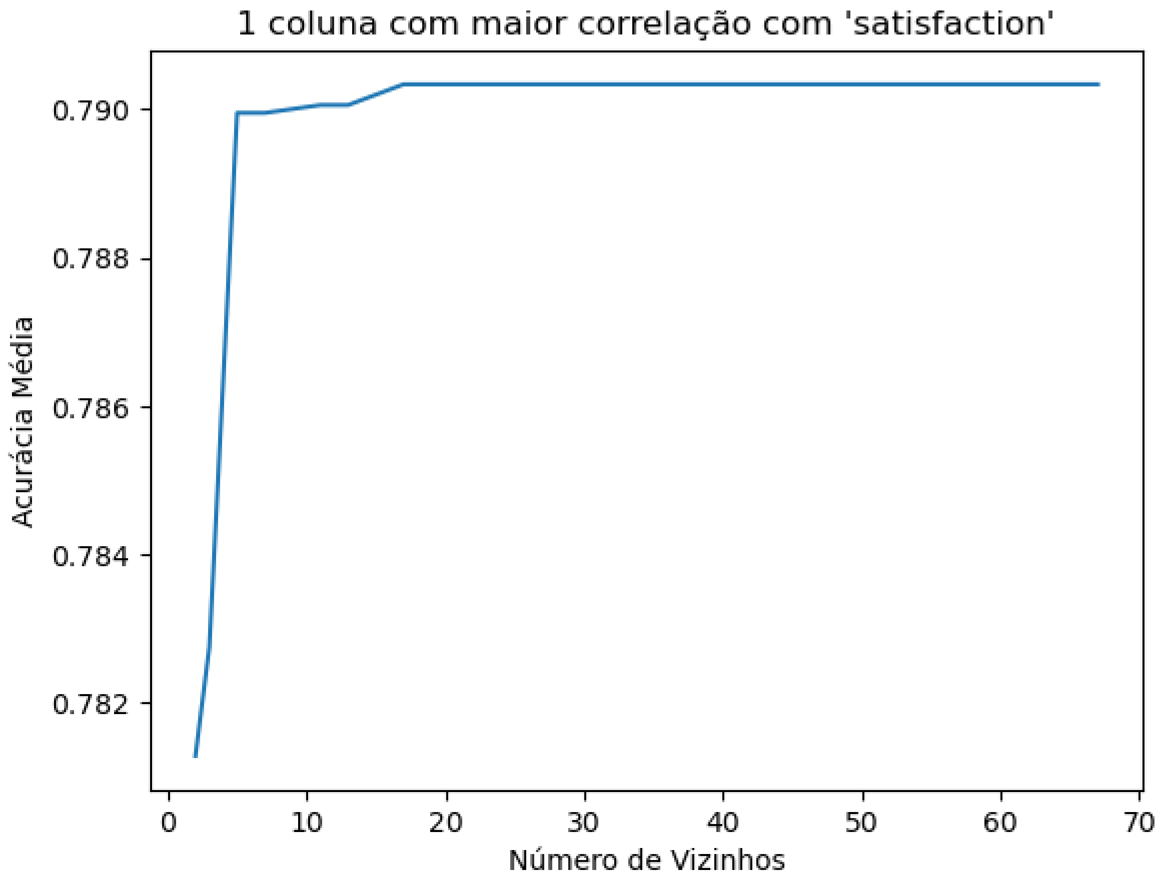
In [78]:

```
# Gera os modelos que incluem as variáveis com maior correlação com 'satisfaction'

cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Obtém a lista de colunas com sua respectiva correlação com 'satisfaction'
correlacoes = df_encod.corr()['satisfaction'].drop('satisfaction')

colunas_modelo = list()
[colunas_modelo.append([x, y]) for x, y in zip(correlacoes.values, correlacoes.index)]

colunas_modelo.sort(key=lambda x : abs(x[0]), reverse=True)

modelo = list()

y = df_encod['satisfaction']

resultados_por_correlacao = list()

for coluna in colunas_modelo[:-1]:
  modelo.append(coluna[1])

  X = df_encod[modelo]

  resultados_modelo = []

  for n_vizinhos in primos:
```

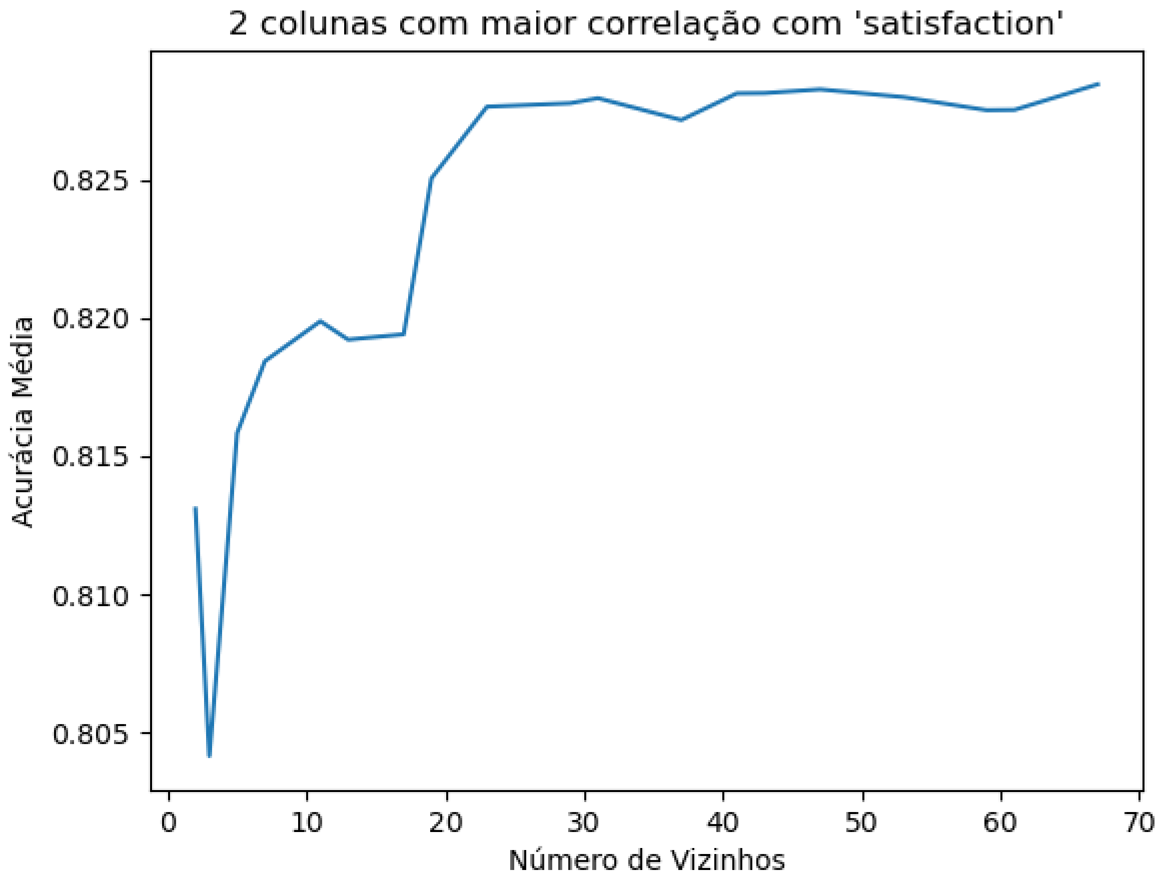
```python
    knn_classifier = KNeighborsClassifier(n_neighbors=n_vizinhos)

    values = cross_val_score(knn_classifier, X, y, cv=cv, scoring='accuracy')

    resultados_modelo.append(values.mean())

  resultados_por_correlacao.append(resultados_modelo.copy())
```

In [79]:
```python
modelo = list()

for coluna in colunas_modelo[:-1]:
  modelo.append(coluna[1])

  resultados_formatados = list()
  for i_primo in range(len(primos)):
    resultados_formatados.append([resultados_por_correlacao[len(modelo)-1], primos[i_pri
  resultados_formatados.sort(key=lambda x : x[0], reverse=True)

  print(f'Modelo [{len(modelo)}]: {modelo}')

  for i in range(3):
    print(f'{resultados_formatados[i][1]} vizinhos: {resultados_formatados[i][0]}')

  plt.plot(primos, resultados_por_correlacao[len(modelo)-1]);
  plt.title(f"{len(modelo)} coluna{'s' if len(modelo) > 1 else ''} com maior correlação
  plt.xlabel("Número de Vizinhos")
  plt.ylabel("Acurácia Média")
  plt.show()
  print('\n')
```

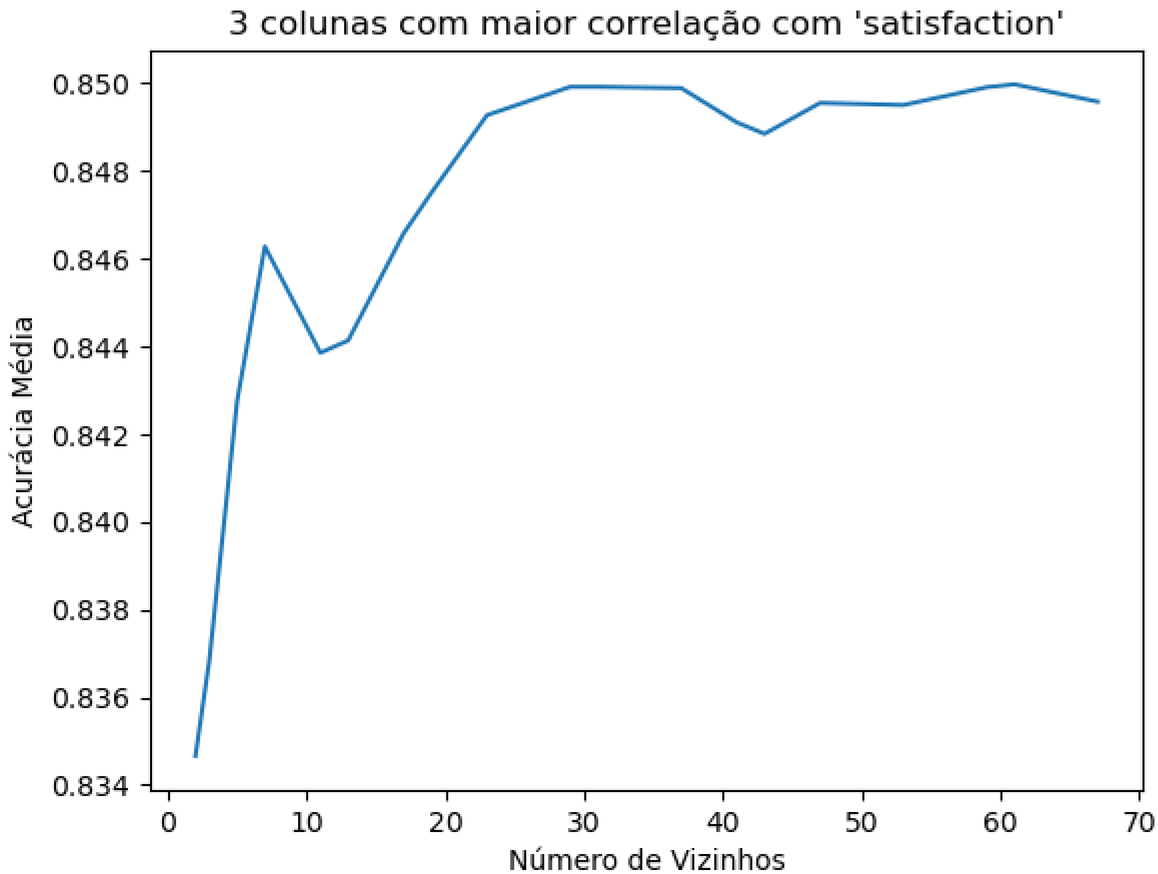
```
Modelo [1]: ['Online boarding']
2 vizinhos: [0.7812806091433372, 0.7827478688816656, 0.7899492663198135, 0.7899492663198
135, 0.7900554541757843, 0.7900554541757843, 0.7903354039778887, 0.7903354039778887, 0.7
903354039778887, 0.7903354039778887, 0.7903354039778887, 0.7903354039778887, 0.790335403
9778887, 0.7903354039778887, 0.7903354039778887, 0.7903354039778887, 0.7903354039778887,
0.7903354039778887, 0.7903354039778887]
3 vizinhos: [0.7812806091433372, 0.7827478688816656, 0.7899492663198135, 0.7899492663198
135, 0.7900554541757843, 0.7900554541757843, 0.7903354039778887, 0.7903354039778887, 0.7
903354039778887, 0.7903354039778887, 0.7903354039778887, 0.7903354039778887, 0.790335403
9778887, 0.7903354039778887, 0.7903354039778887, 0.7903354039778887, 0.7903354039778887,
0.7903354039778887, 0.7903354039778887]
5 vizinhos: [0.7812806091433372, 0.7827478688816656, 0.7899492663198135, 0.7899492663198
135, 0.7900554541757843, 0.7900554541757843, 0.7903354039778887, 0.7903354039778887, 0.7
903354039778887, 0.7903354039778887, 0.7903354039778887, 0.7903354039778887, 0.790335403
9778887, 0.7903354039778887, 0.7903354039778887, 0.7903354039778887, 0.7903354039778887,
0.7903354039778887, 0.7903354039778887]
```



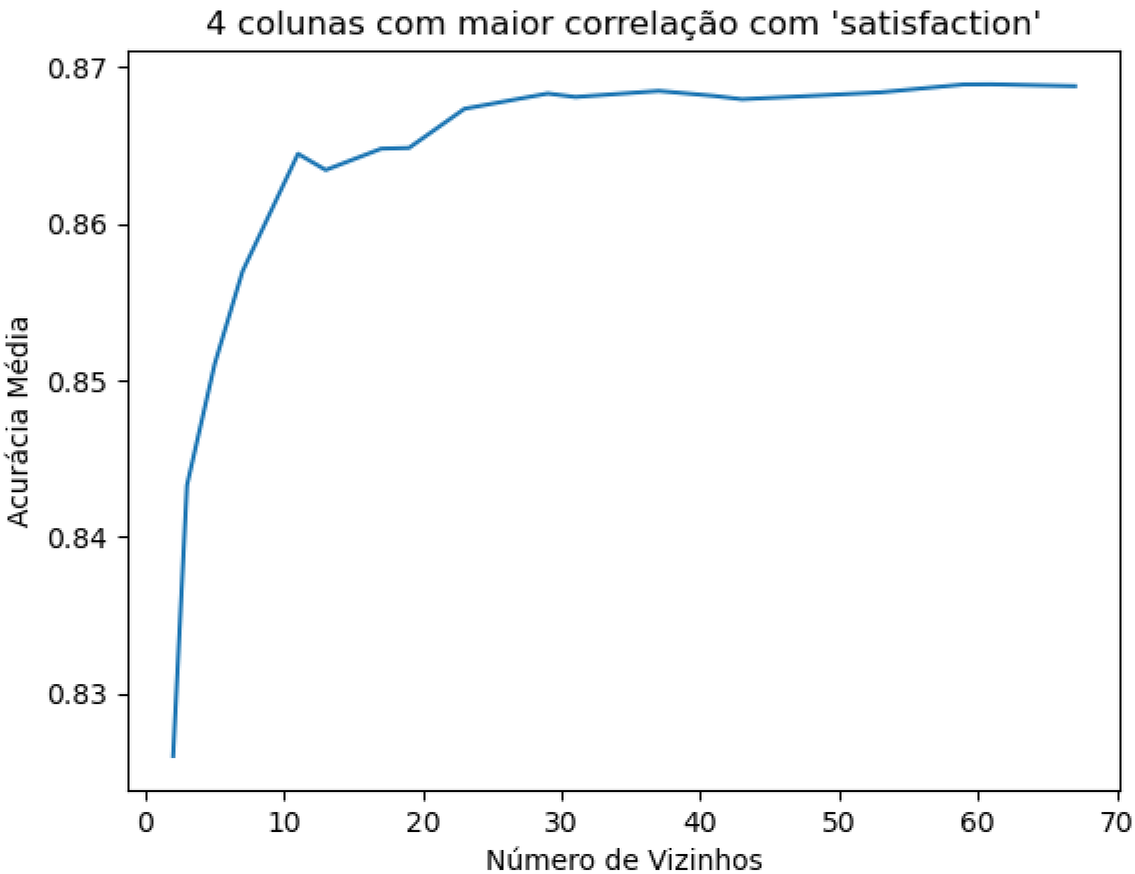

```
Modelo [2]: ['Online boarding', 'Class']
2 vizinhos: [0.813078255119958, 0.8041296882087499, 0.815819530589296, 0.818425616879135
5, 0.8198738045480007, 0.8192077040426302, 0.8194007579628778, 0.8250479196460951, 0.827
6445518995474, 0.8277700512974189, 0.8279534666850046, 0.8271618844859505, 0.82812719415
45617, 0.8281368475960136, 0.8282719901855436, 0.827992090700605, 0.8275190935008485, 0.
8275287180565201, 0.8284554055731294]
3 vizinhos: [0.813078255119958, 0.8041296882087499, 0.815819530589296, 0.818425616879135
5, 0.8198738045480007, 0.8192077040426302, 0.8194007579628778, 0.8250479196460951, 0.827
6445518995474, 0.8277700512974189, 0.8279534666850046, 0.8271618844859505, 0.82812719415
45617, 0.8281368475960136, 0.8282719901855436, 0.827992090700605, 0.8275190935008485, 0.
8275287180565201, 0.8284554055731294]
5 vizinhos: [0.813078255119958, 0.8041296882087499, 0.815819530589296, 0.818425616879135
5, 0.8198738045480007, 0.8192077040426302, 0.8194007579628778, 0.8250479196460951, 0.827
6445518995474, 0.8277700512974189, 0.8279534666850046, 0.8271618844859505, 0.82812719415
45617, 0.8281368475960136, 0.8282719901855436, 0.827992090700605, 0.8275190935008485, 0.
8275287180565201, 0.8284554055731294]
```



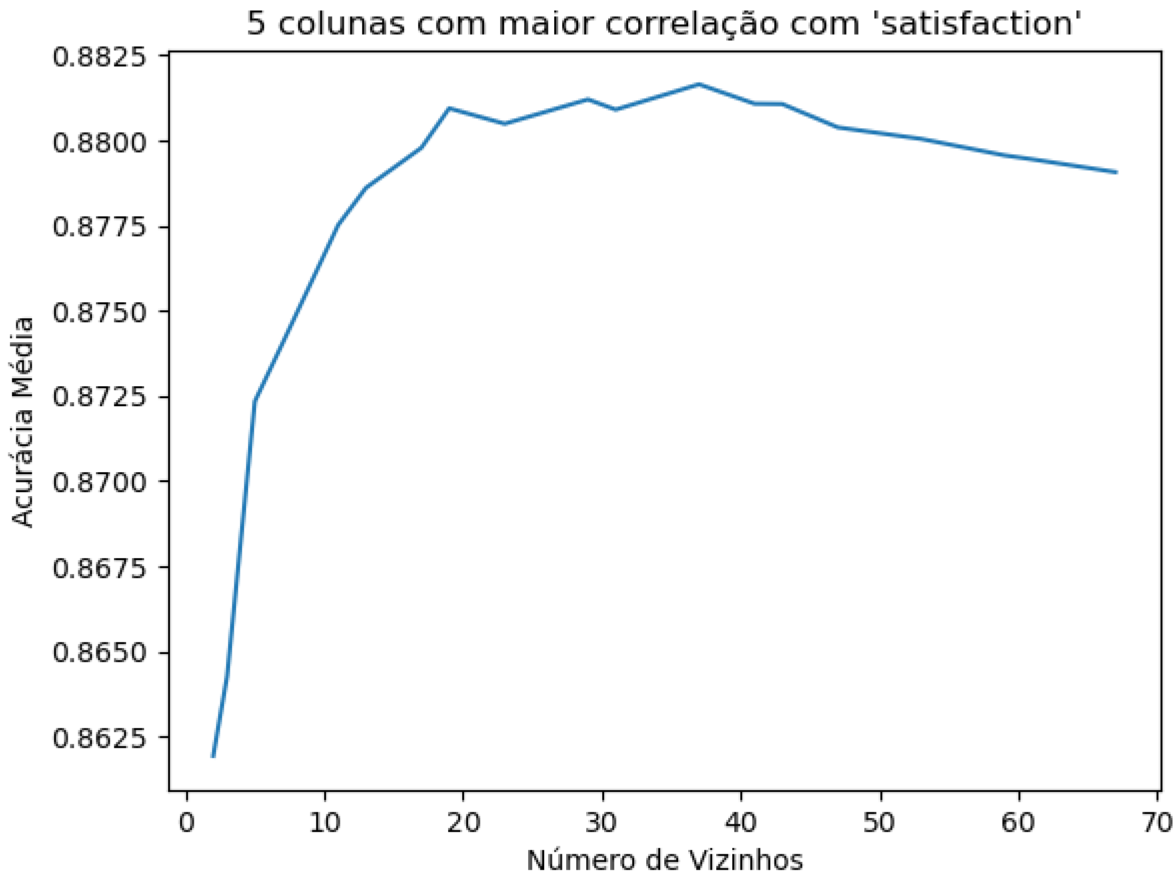

```
Modelo [3]: ['Online boarding', 'Class', 'Type of Travel']
2 vizinhos: [0.834662465928756, 0.8368439144742359, 0.8427514609682291, 0.84627492703079
13, 0.8438520707004633, 0.8441320167753703, 0.8465742135333795, 0.8475009439127597, 0.84
92674389047108, 0.8499141961870025, 0.8499141961870025, 0.8498852358626469, 0.8491033071
050447, 0.8488426641858441, 0.8495473654118312, 0.8494990982045717, 0.8499045427455506,
0.8499721112449174, 0.8495763201453904]
3 vizinhos: [0.834662465928756, 0.8368439144742359, 0.8427514609682291, 0.84627492703079
13, 0.8438520707004633, 0.8441320167753703, 0.8465742135333795, 0.8475009439127597, 0.84
92674389047108, 0.8499141961870025, 0.8499141961870025, 0.8498852358626469, 0.8491033071
050447, 0.8488426641858441, 0.8495473654118312, 0.8494990982045717, 0.8499045427455506,
0.8499721112449174, 0.8495763201453904]
5 vizinhos: [0.834662465928756, 0.8368439144742359, 0.8427514609682291, 0.84627492703079
13, 0.8438520707004633, 0.8441320167753703, 0.8465742135333795, 0.8475009439127597, 0.84
92674389047108, 0.8499141961870025, 0.8499141961870025, 0.8498852358626469, 0.8491033071
050447, 0.8488426641858441, 0.8495473654118312, 0.8494990982045717, 0.8499045427455506,
0.8499721112449174, 0.8495763201453904]
```



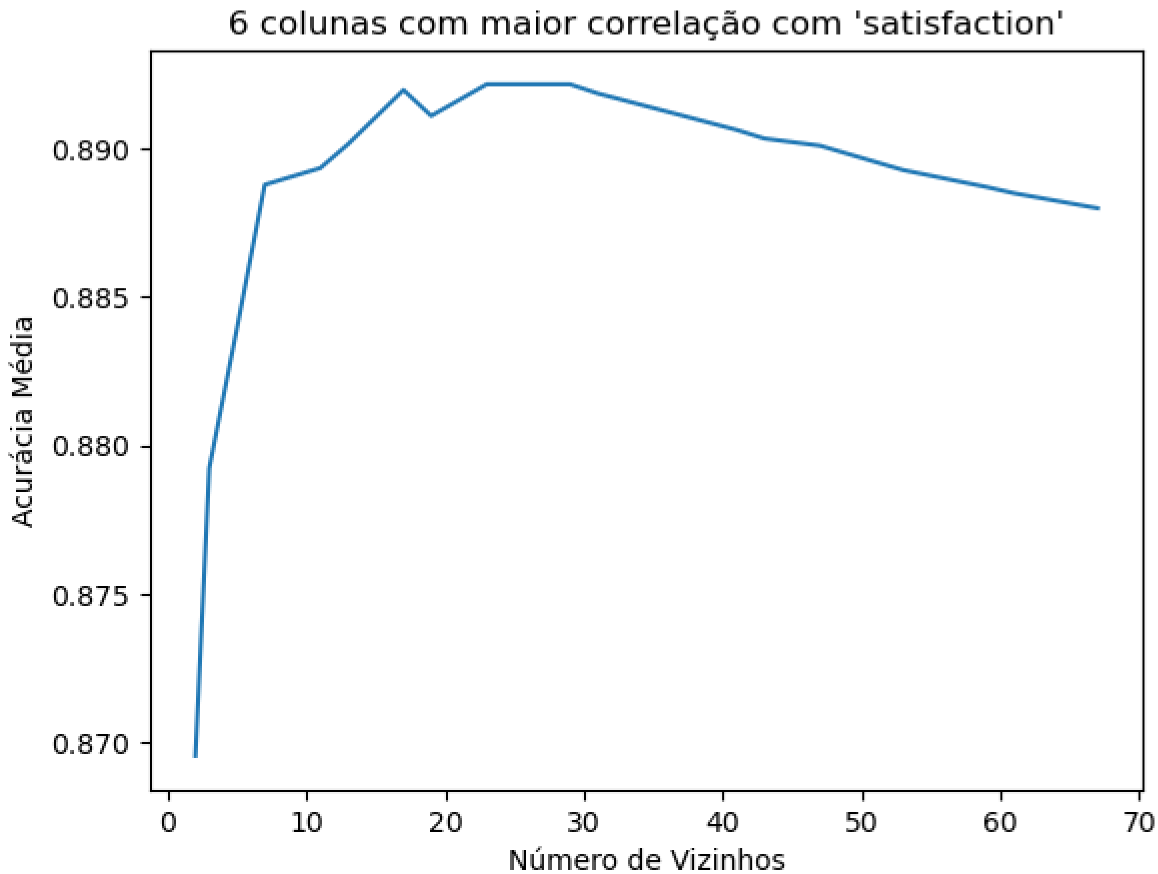

```
Modelo [4]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment']
2 vizinhos: [0.8259937276857346, 0.8432822390467918, 0.851033792263158, 0.85693188565256
34, 0.8644419127455617, 0.8634090802357527, 0.864770098073747, 0.8648184034847806, 0.867
3185628224724, 0.8682935343187298, 0.8680715079607347, 0.8684576484142079, 0.86813910814
12801, 0.8679267464063294, 0.8681004906482752, 0.8683707441461568, 0.8688727268288521,
0.8688823746795077, 0.8687665333820851]
3 vizinhos: [0.8259937276857346, 0.8432822390467918, 0.851033792263158, 0.85693188565256
34, 0.8644419127455617, 0.8634090802357527, 0.864770098073747, 0.8648184034847806, 0.867
3185628224724, 0.8682935343187298, 0.8680715079607347, 0.8684576484142079, 0.86813910814
12801, 0.8679267464063294, 0.8681004906482752, 0.8683707441461568, 0.8688727268288521,
0.8688823746795077, 0.8687665333820851]
5 vizinhos: [0.8259937276857346, 0.8432822390467918, 0.851033792263158, 0.85693188565256
34, 0.8644419127455617, 0.8634090802357527, 0.864770098073747, 0.8648184034847806, 0.867
3185628224724, 0.8682935343187298, 0.8680715079607347, 0.8684576484142079, 0.86813910814
12801, 0.8679267464063294, 0.8681004906482752, 0.8683707441461568, 0.8688727268288521,
0.8688823746795077, 0.8687665333820851]
```



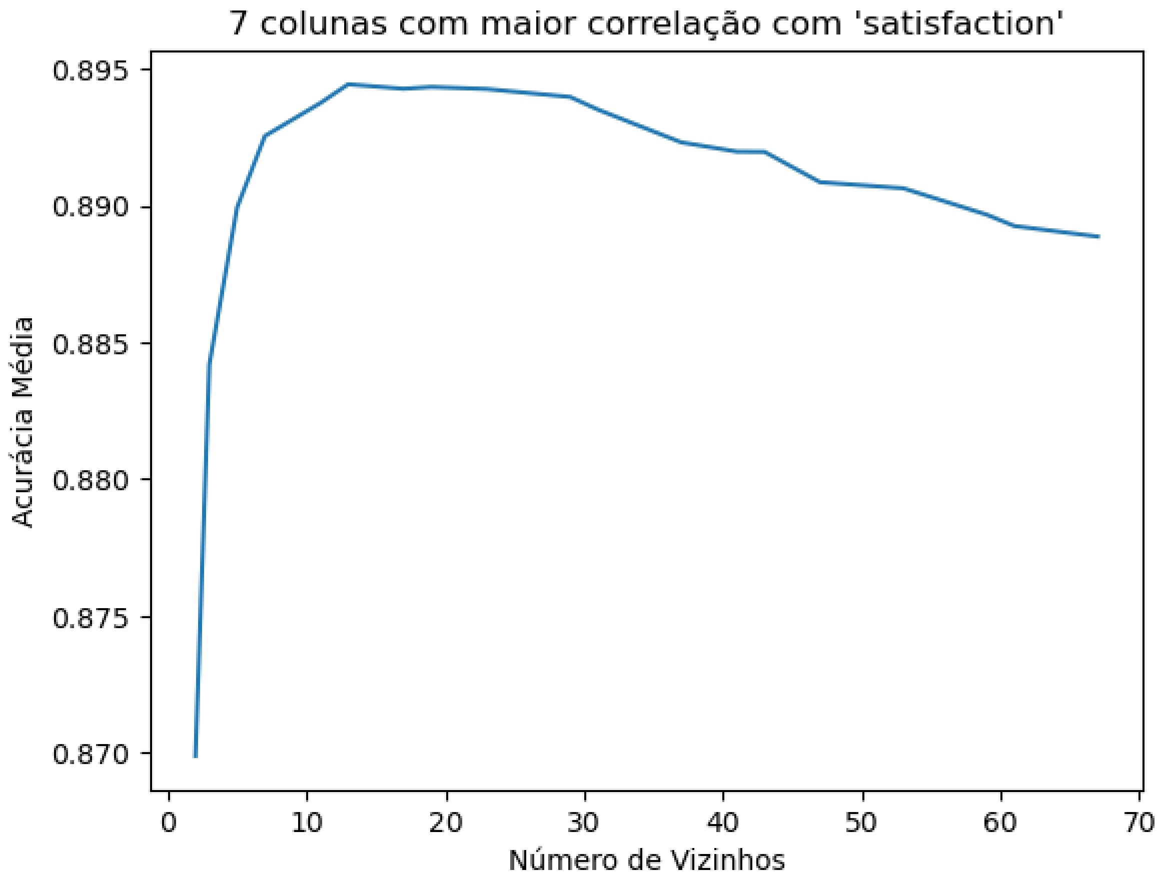

```
Modelo [5]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'Se
at comfort']
2 vizinhos: [0.8619322704857023, 0.8642684405890314, 0.8723383141736749, 0.8740468307453
538, 0.8775122941608606, 0.8786030407967855, 0.8797807149957453, 0.8809390441080277, 0.8
80485384540554, 0.8811997159130087, 0.8809004601597998, 0.881643743470416, 0.88107420533
35451, 0.8810645444376982, 0.8803791584808092, 0.8800412982797866, 0.8795586606837691,
0.8794428165909485, 0.8790663705780994]
3 vizinhos: [0.8619322704857023, 0.8642684405890314, 0.8723383141736749, 0.8740468307453
538, 0.8775122941608606, 0.8786030407967855, 0.8797807149957453, 0.8809390441080277, 0.8
80485384540554, 0.8811997159130087, 0.8809004601597998, 0.881643743470416, 0.88107420533
35451, 0.8810645444376982, 0.8803791584808092, 0.8800412982797866, 0.8795586606837691,
0.8794428165909485, 0.8790663705780994]
5 vizinhos: [0.8619322704857023, 0.8642684405890314, 0.8723383141736749, 0.8740468307453
538, 0.8775122941608606, 0.8786030407967855, 0.8797807149957453, 0.8809390441080277, 0.8
80485384540554, 0.8811997159130087, 0.8809004601597998, 0.881643743470416, 0.88107420533
35451, 0.8810645444376982, 0.8803791584808092, 0.8800412982797866, 0.8795586606837691,
0.8794428165909485, 0.8790663705780994]
```



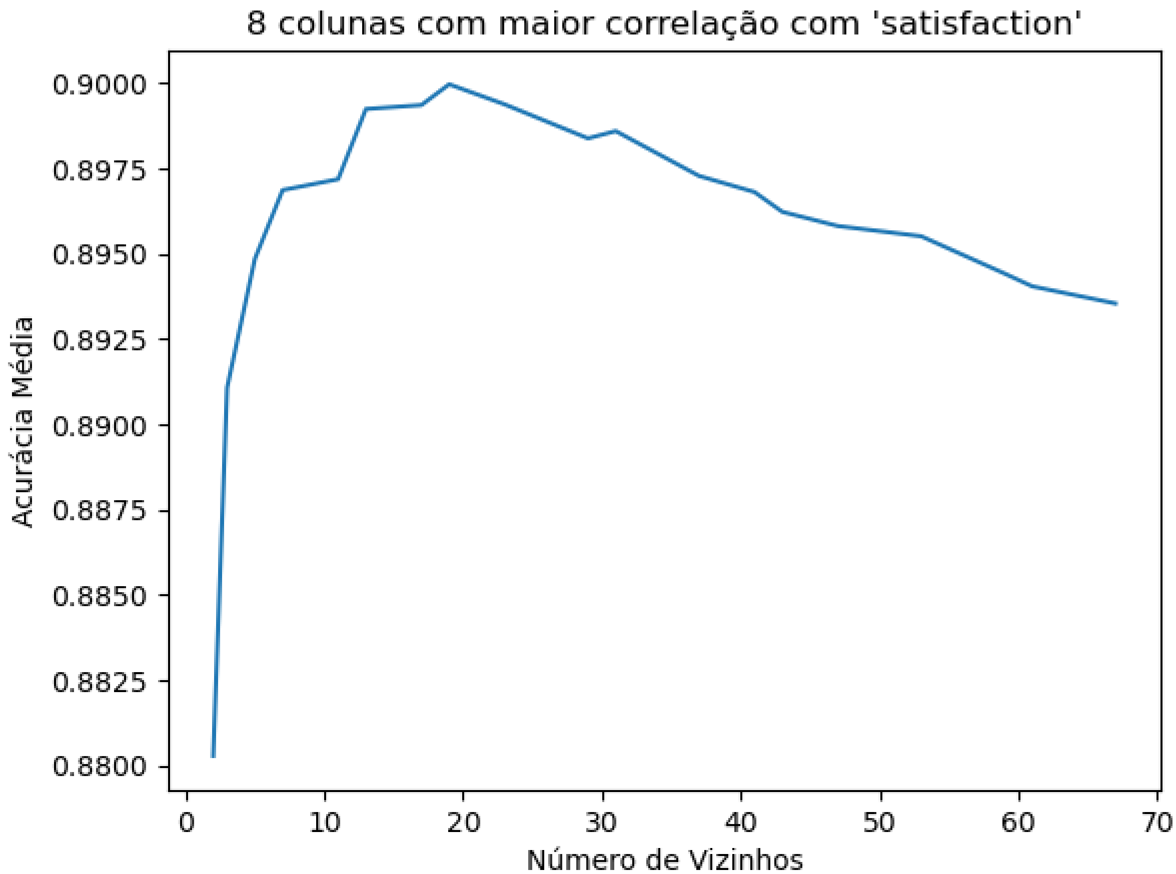

```
Modelo [6]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'Se
at comfort', 'On-board service']
2 vizinhos: [0.8695677037966352, 0.8792402079999821, 0.8839508749782425, 0.8887773925719
189, 0.8893372800627362, 0.8901384952036558, 0.8919628996627258, 0.8910941234768341, 0.8
921559722189609, 0.8921559619691679, 0.8918470443883129, 0.8911133949513619, 0.890621102
9820934, 0.8903315202381232, 0.8900901646340396, 0.889269642610216, 0.8887001072687433,
0.8884877501927892, 0.8879857805552854]
3 vizinhos: [0.8695677037966352, 0.8792402079999821, 0.8839508749782425, 0.8887773925719
189, 0.8893372800627362, 0.8901384952036558, 0.8919628996627258, 0.8910941234768341, 0.8
921559722189609, 0.8921559619691679, 0.8918470443883129, 0.8911133949513619, 0.890621102
9820934, 0.8903315202381232, 0.8900901646340396, 0.889269642610216, 0.8887001072687433,
0.8884877501927892, 0.8879857805552854]
5 vizinhos: [0.8695677037966352, 0.8792402079999821, 0.8839508749782425, 0.8887773925719
189, 0.8893372800627362, 0.8901384952036558, 0.8919628996627258, 0.8910941234768341, 0.8
921559722189609, 0.8921559619691679, 0.8918470443883129, 0.8911133949513619, 0.890621102
9820934, 0.8903315202381232, 0.8900901646340396, 0.889269642610216, 0.8887001072687433,
0.8884877501927892, 0.8879857805552854]
```



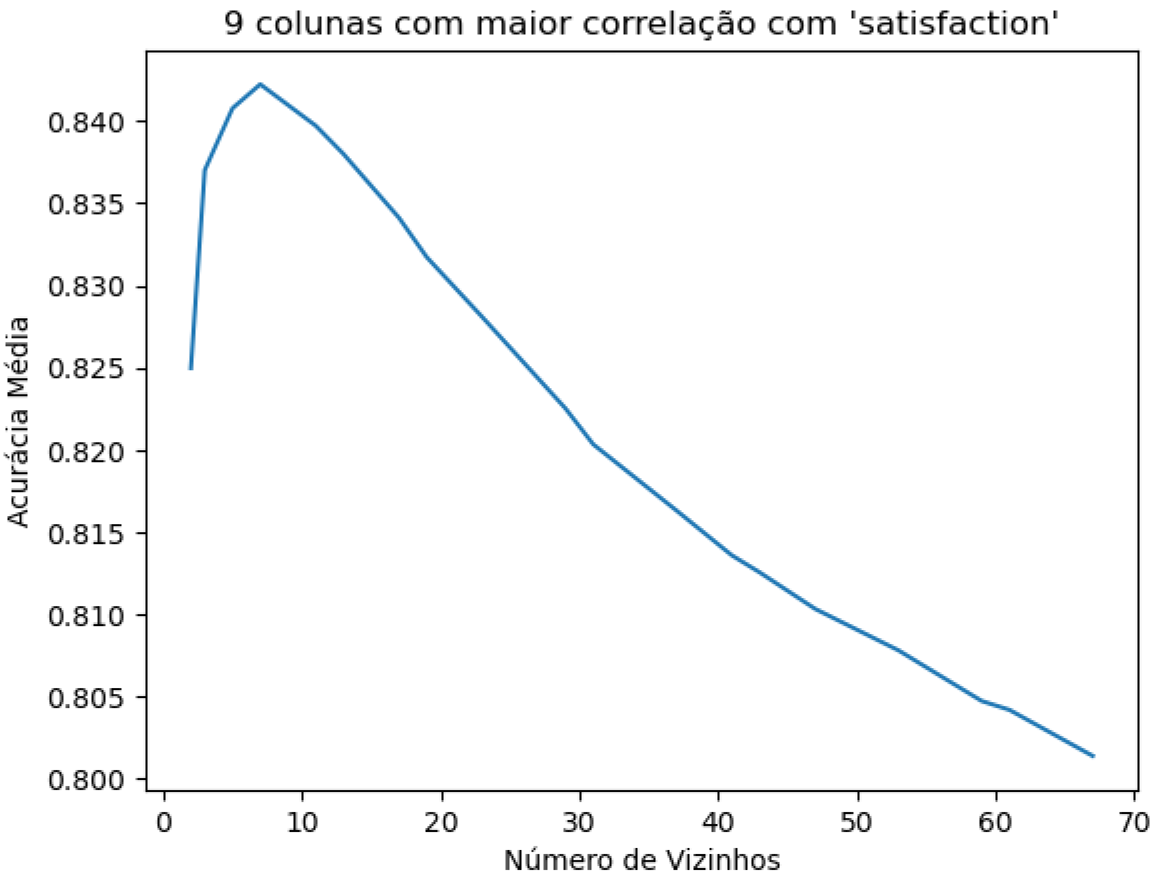

```
Modelo [7]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'Se
at comfort', 'On-board service', 'Leg room service']
2 vizinhos: [0.8698767024440353, 0.8841921933103514, 0.8899357682741698, 0.8925517269783
125, 0.8937583875920104, 0.8944437707535012, 0.8942796818166062, 0.894347224225591, 0.89
42700190571606, 0.8939804419039866, 0.8935170813732934, 0.8923201021550282, 0.8919822344
996107, 0.8919725857171557, 0.8908624697677693, 0.8906404238419878, 0.8896751402637589,
0.8892600646445128, 0.8888739372362309]
3 vizinhos: [0.8698767024440353, 0.8841921933103514, 0.8899357682741698, 0.8925517269783
125, 0.8937583875920104, 0.8944437707535012, 0.8942796818166062, 0.894347224225591, 0.89
42700190571606, 0.8939804419039866, 0.8935170813732934, 0.8923201021550282, 0.8919822344
996107, 0.8919725857171557, 0.8908624697677693, 0.8906404238419878, 0.8896751402637589,
0.8892600646445128, 0.8888739372362309]
5 vizinhos: [0.8698767024440353, 0.8841921933103514, 0.8899357682741698, 0.8925517269783
125, 0.8937583875920104, 0.8944437707535012, 0.8942796818166062, 0.894347224225591, 0.89
42700190571606, 0.8939804419039866, 0.8935170813732934, 0.8923201021550282, 0.8919822344
996107, 0.8919725857171557, 0.8908624697677693, 0.8906404238419878, 0.8896751402637589,
0.8892600646445128, 0.8888739372362309]
```



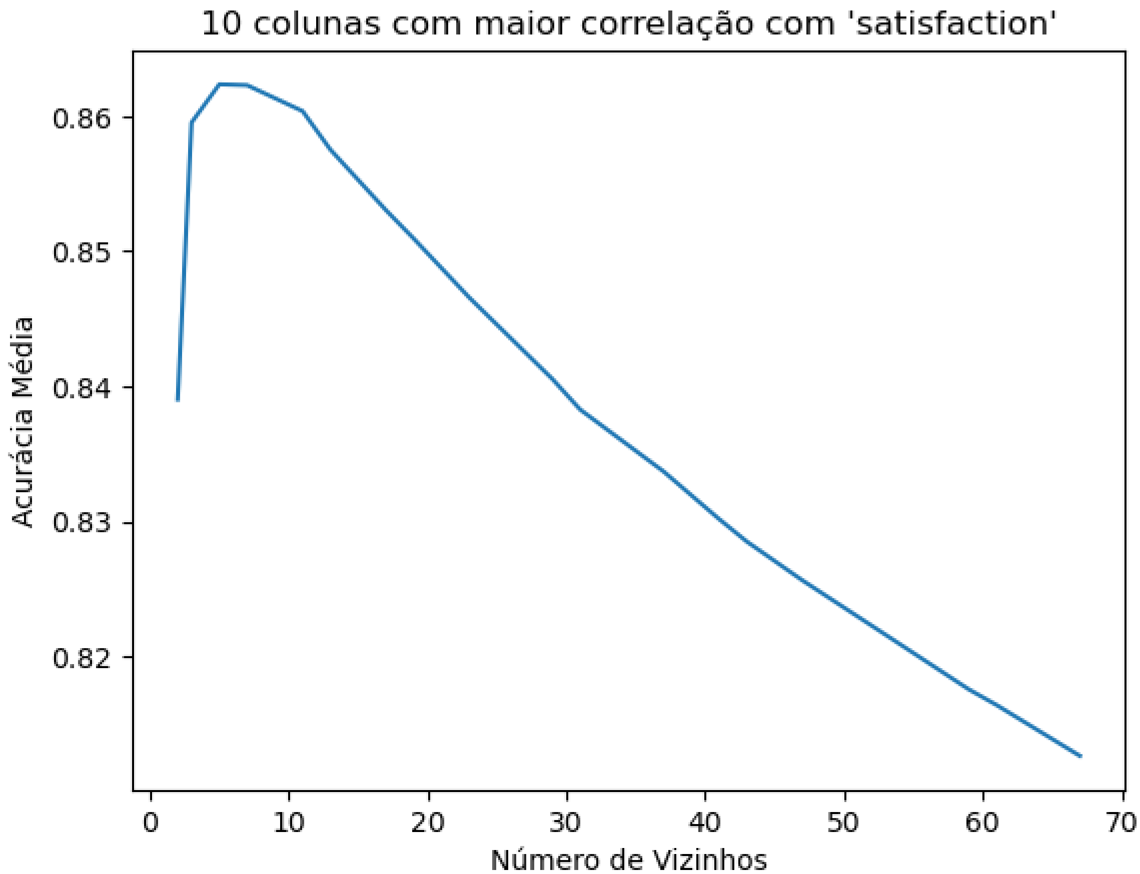

```
Modelo [8]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'Se
at comfort', 'On-board service', 'Leg room service', 'Cleanliness']
2 vizinhos: [0.8802827507910045, 0.8910651678114754, 0.8948394854454802, 0.8968569773695
751, 0.8971755884592548, 0.8992316997399534, 0.8993475224013885, 0.8999556808266623, 0.8
993668143755025, 0.8983725369281409, 0.8985849163672797, 0.8972720921243946, 0.896799094
9246378, 0.8962199061417133, 0.8958048267952698, 0.8955055775646565, 0.8944147768843687,
0.8940383094401338, 0.893536367756611]
3 vizinhos: [0.8802827507910045, 0.8910651678114754, 0.8948394854454802, 0.8968569773695
751, 0.8971755884592548, 0.8992316997399534, 0.8993475224013885, 0.8999556808266623, 0.8
993668143755025, 0.8983725369281409, 0.8985849163672797, 0.8972720921243946, 0.896799094
9246378, 0.8962199061417133, 0.8958048267952698, 0.8955055775646565, 0.8944147768843687,
0.8940383094401338, 0.893536367756611]
5 vizinhos: [0.8802827507910045, 0.8910651678114754, 0.8948394854454802, 0.8968569773695
751, 0.8971755884592548, 0.8992316997399534, 0.8993475224013885, 0.8999556808266623, 0.8
993668143755025, 0.8983725369281409, 0.8985849163672797, 0.8972720921243946, 0.896799094
9246378, 0.8962199061417133, 0.8958048267952698, 0.8955055775646565, 0.8944147768843687,
0.8940383094401338, 0.893536367756611]
```



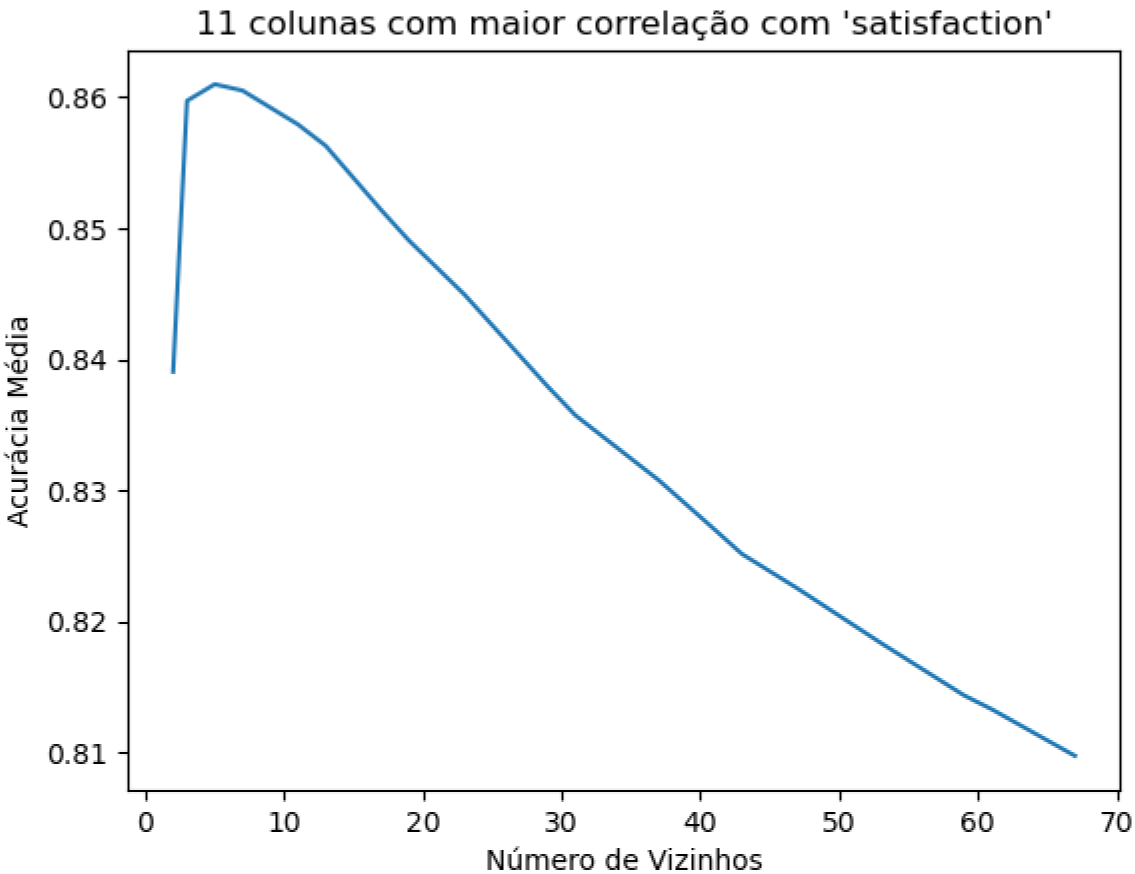

```
Modelo [9]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'Se
at comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance']
2 vizinhos: [0.8249802654211864, 0.8370176074672164, 0.8407822912275563, 0.8422399217512
163, 0.839720484416401, 0.8380022659497028, 0.8341217306421477, 0.8317277591604265, 0.82
80885403213812, 0.8225284077673305, 0.8203468017477574, 0.8163214918406057, 0.8135896694
758553, 0.8125567875806798, 0.810336549159312, 0.8078171174152929, 0.8047377655674788,
0.8041972017319541, 0.8013977987544451]
3 vizinhos: [0.8249802654211864, 0.8370176074672164, 0.8407822912275563, 0.8422399217512
163, 0.839720484416401, 0.8380022659497028, 0.8341217306421477, 0.8317277591604265, 0.82
80885403213812, 0.8225284077673305, 0.8203468017477574, 0.8163214918406057, 0.8135896694
758553, 0.8125567875806798, 0.810336549159312, 0.8078171174152929, 0.8047377655674788,
0.8041972017319541, 0.8013977987544451]
5 vizinhos: [0.8249802654211864, 0.8370176074672164, 0.8407822912275563, 0.8422399217512
163, 0.839720484416401, 0.8380022659497028, 0.8341217306421477, 0.8317277591604265, 0.82
80885403213812, 0.8225284077673305, 0.8203468017477574, 0.8163214918406057, 0.8135896694
758553, 0.8125567875806798, 0.810336549159312, 0.8078171174152929, 0.8047377655674788,
0.8041972017319541, 0.8013977987544451]
```



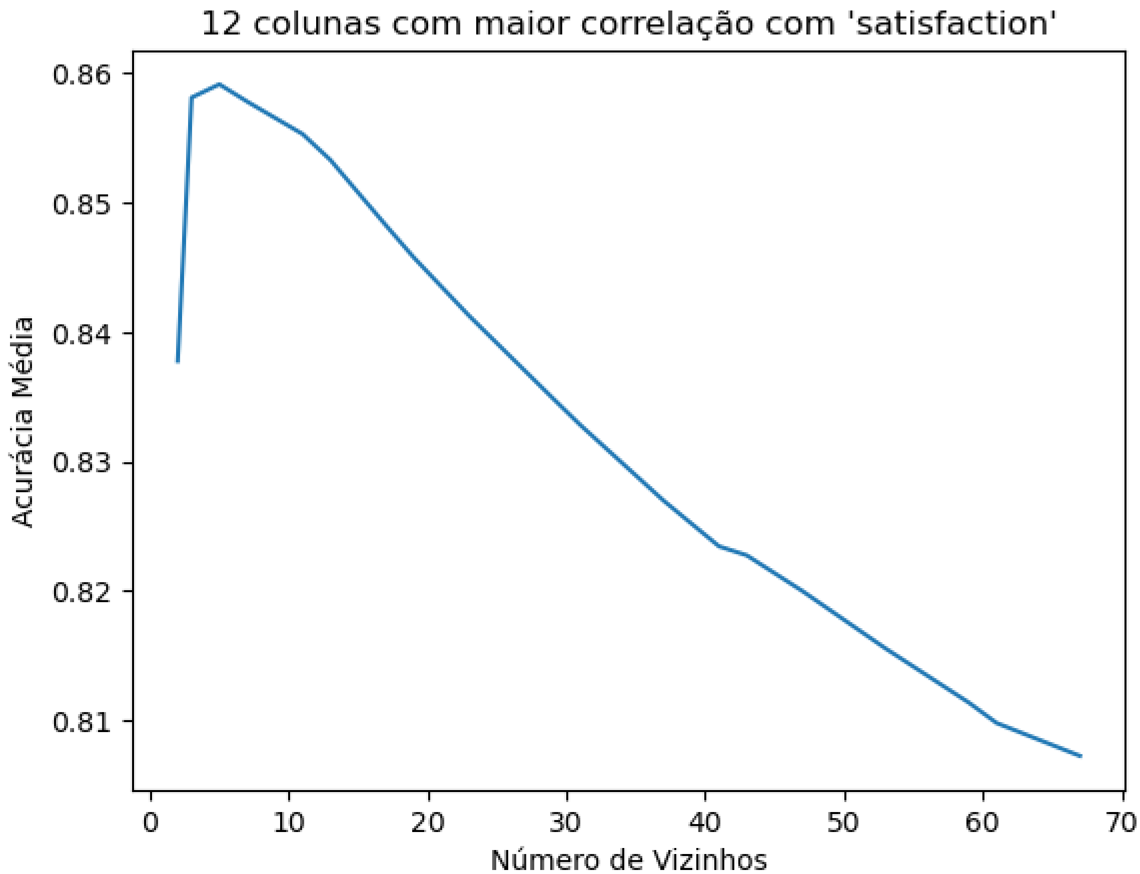

```
Modelo [10]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service']
2 vizinhos: [0.8390351450494805, 0.8595382412324202, 0.8623665812393007, 0.8622990388303
16, 0.8603780971613292, 0.8575304605213381, 0.8530514220935593, 0.8509470184470184, 0.84
65838399526497, 0.8405410027130271, 0.8382821645028423, 0.8337066289325195, 0.8301832430
047025, 0.8285132619276843, 0.8256463174729898, 0.821611317784211, 0.8175667047213528,
0.8164083429960929, 0.8126436350089694]
3 vizinhos: [0.8390351450494805, 0.8595382412324202, 0.8623665812393007, 0.8622990388303
16, 0.8603780971613292, 0.8575304605213381, 0.8530514220935593, 0.8509470184470184, 0.84
65838399526497, 0.8405410027130271, 0.8382821645028423, 0.8337066289325195, 0.8301832430
047025, 0.8285132619276843, 0.8256463174729898, 0.821611317784211, 0.8175667047213528,
0.8164083429960929, 0.8126436350089694]
5 vizinhos: [0.8390351450494805, 0.8595382412324202, 0.8623665812393007, 0.8622990388303
16, 0.8603780971613292, 0.8575304605213381, 0.8530514220935593, 0.8509470184470184, 0.84
65838399526497, 0.8405410027130271, 0.8382821645028423, 0.8337066289325195, 0.8301832430
047025, 0.8285132619276843, 0.8256463174729898, 0.821611317784211, 0.8175667047213528,
0.8164083429960929, 0.8126436350089694]
```



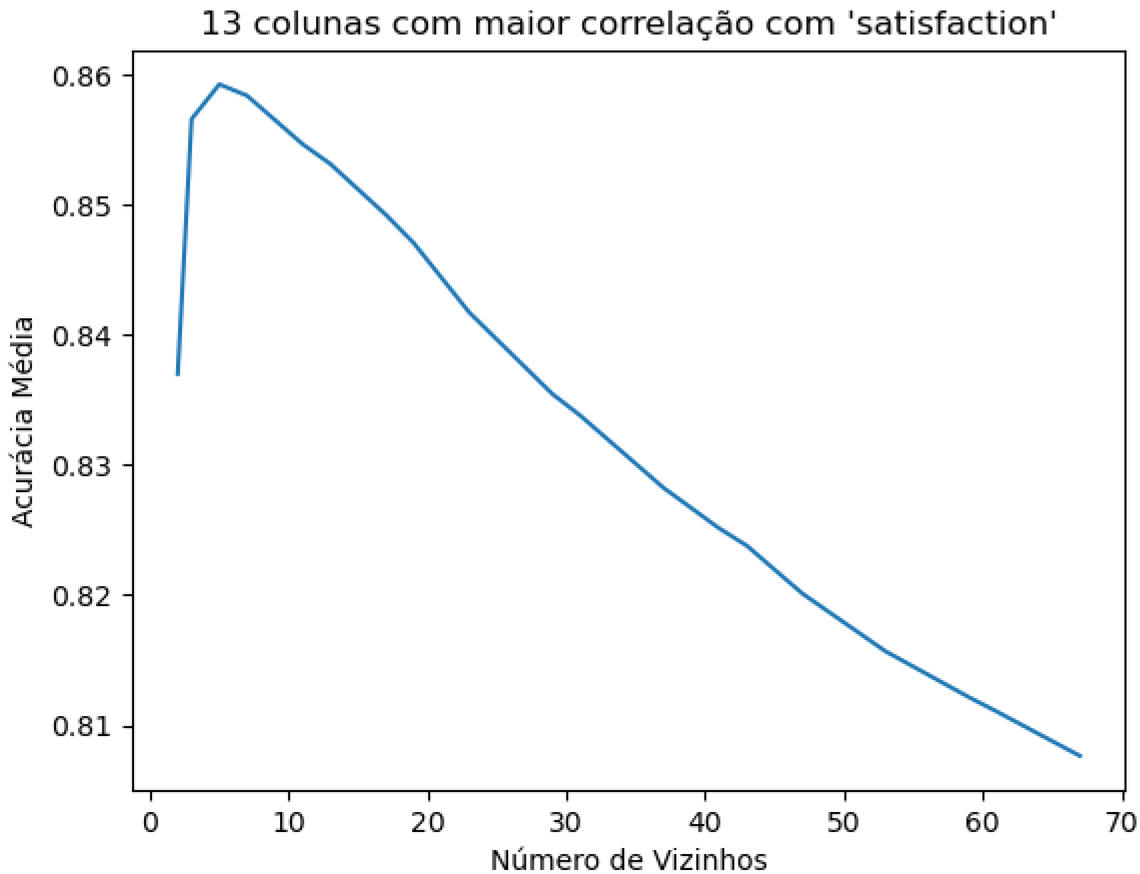

```
Modelo [11]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling']
2 vizinhos: [0.8390351143001012, 0.859711992928761, 0.8609765406463931, 0.86049392075456
38, 0.8579069158521808, 0.8562755382911769, 0.8513717465759167, 0.8490453585023525, 0.84
49235067262869, 0.8379056998540058, 0.8357047962695227, 0.8307624327194267, 0.8270267130
106401, 0.8251346878714385, 0.82250908131664, 0.8183582505802315, 0.8143619047246327, 0.
8133000774138915, 0.8097477488659071]
3 vizinhos: [0.8390351143001012, 0.859711992928761, 0.8609765406463931, 0.86049392075456
38, 0.8579069158521808, 0.8562755382911769, 0.8513717465759167, 0.8490453585023525, 0.84
49235067262869, 0.8379056998540058, 0.8357047962695227, 0.8307624327194267, 0.8270267130
106401, 0.8251346878714385, 0.82250908131664, 0.8183582505802315, 0.8143619047246327, 0.
8133000774138915, 0.8097477488659071]
5 vizinhos: [0.8390351143001012, 0.859711992928761, 0.8609765406463931, 0.86049392075456
38, 0.8579069158521808, 0.8562755382911769, 0.8513717465759167, 0.8490453585023525, 0.84
49235067262869, 0.8379056998540058, 0.8357047962695227, 0.8307624327194267, 0.8270267130
106401, 0.8251346878714385, 0.82250908131664, 0.8183582505802315, 0.8143619047246327, 0.
8133000774138915, 0.8097477488659071]
```



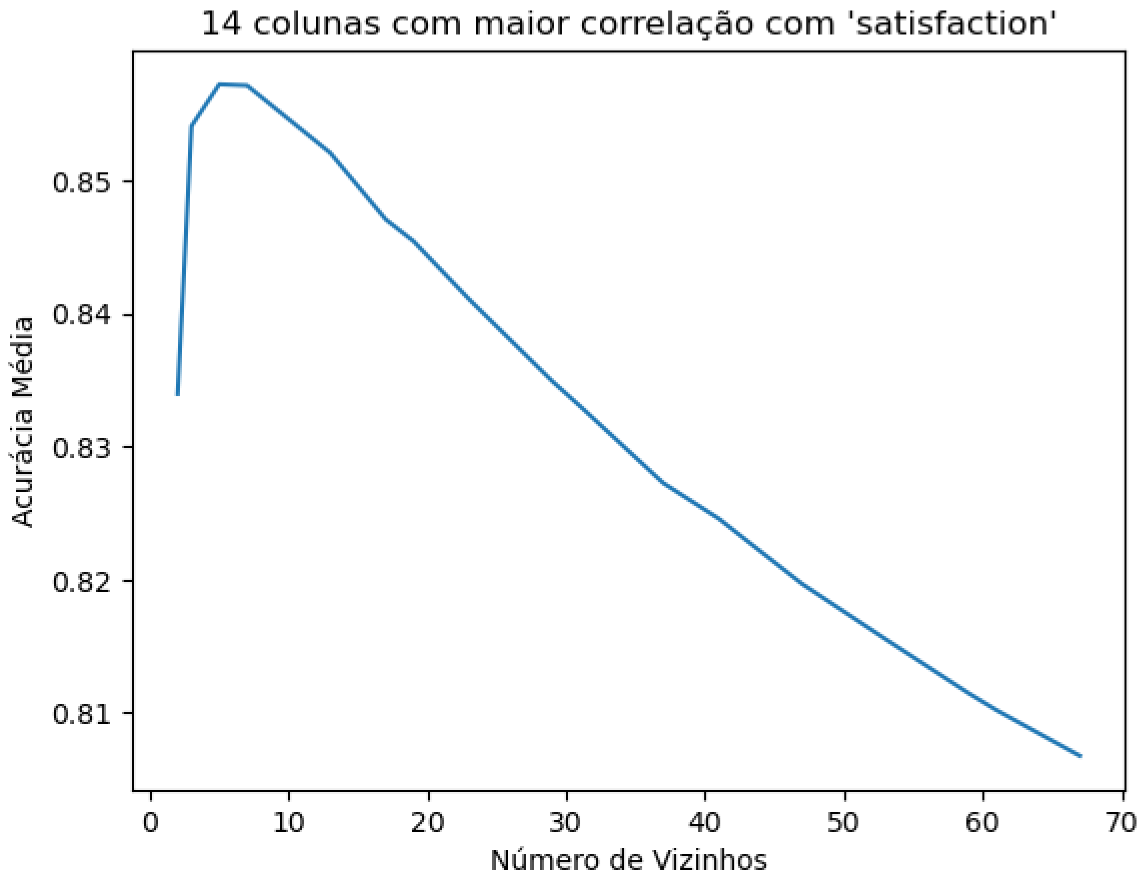

```
Modelo [12]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling', 'Inflight service']
2 vizinhos: [0.8377609550719889, 0.8580999856130178, 0.8591231991579515, 0.8577911155539
306, 0.8552813009111879, 0.853283086052417, 0.8482345579413346, 0.8457633421556098, 0.84
12457039390141, 0.8349132532060421, 0.8328282067595708, 0.8269881011084312, 0.8234550850
341467, 0.8227600540220001, 0.8200089480693302, 0.815568528066356, 0.8113887481871844,
0.8098152838204967, 0.8072765330801822]
3 vizinhos: [0.8377609550719889, 0.8580999856130178, 0.8591231991579515, 0.8577911155539
306, 0.8552813009111879, 0.853283086052417, 0.8482345579413346, 0.8457633421556098, 0.84
12457039390141, 0.8349132532060421, 0.8328282067595708, 0.8269881011084312, 0.8234550850
341467, 0.8227600540220001, 0.8200089480693302, 0.815568528066356, 0.8113887481871844,
0.8098152838204967, 0.8072765330801822]
5 vizinhos: [0.8377609550719889, 0.8580999856130178, 0.8591231991579515, 0.8577911155539
306, 0.8552813009111879, 0.853283086052417, 0.8482345579413346, 0.8457633421556098, 0.84
12457039390141, 0.8349132532060421, 0.8328282067595708, 0.8269881011084312, 0.8234550850
341467, 0.8227600540220001, 0.8200089480693302, 0.815568528066356, 0.8113887481871844,
0.8098152838204967, 0.8072765330801822]
```



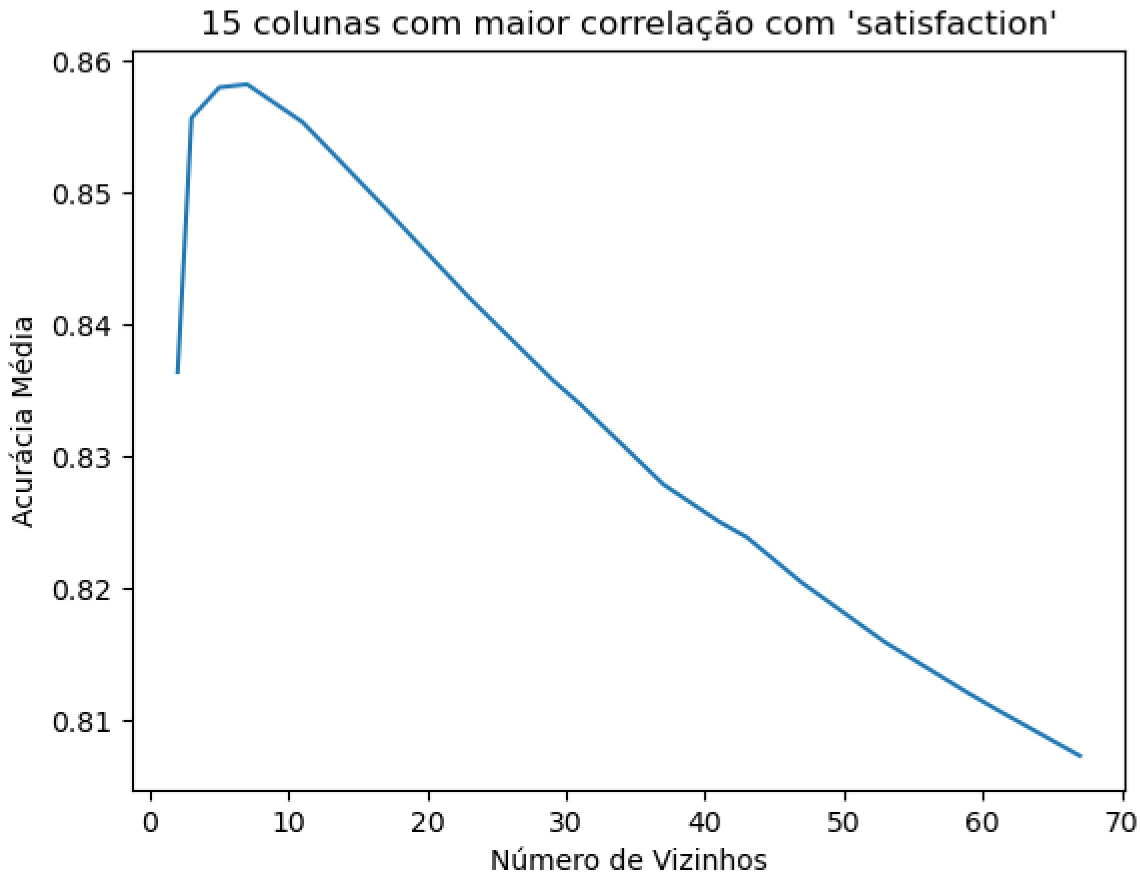

```
Modelo [13]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling', 'Inflight service', 'Checkin service']
2 vizinhos: [0.8370080099337267, 0.8566133947650021, 0.8592872983446398, 0.8584088882850
829, 0.8546828033817606, 0.8531576015633358, 0.8492481059314247, 0.8470858217035454, 0.8
417476540087314, 0.8354635058913947, 0.8338128400834742, 0.8282526730528469, 0.825144372
994069, 0.8238026042674175, 0.8201247604809725, 0.8156747168541262, 0.8121513532894941,
0.8110509010313528, 0.8076337188000959]
3 vizinhos: [0.8370080099337267, 0.8566133947650021, 0.8592872983446398, 0.8584088882850
829, 0.8546828033817606, 0.8531576015633358, 0.8492481059314247, 0.8470858217035454, 0.8
417476540087314, 0.8354635058913947, 0.8338128400834742, 0.8282526730528469, 0.825144372
994069, 0.8238026042674175, 0.8201247604809725, 0.8156747168541262, 0.8121513532894941,
0.8110509010313528, 0.8076337188000959]
5 vizinhos: [0.8370080099337267, 0.8566133947650021, 0.8592872983446398, 0.8584088882850
829, 0.8546828033817606, 0.8531576015633358, 0.8492481059314247, 0.8470858217035454, 0.8
417476540087314, 0.8354635058913947, 0.8338128400834742, 0.8282526730528469, 0.825144372
994069, 0.8238026042674175, 0.8201247604809725, 0.8156747168541262, 0.8121513532894941,
0.8110509010313528, 0.8076337188000959]
```



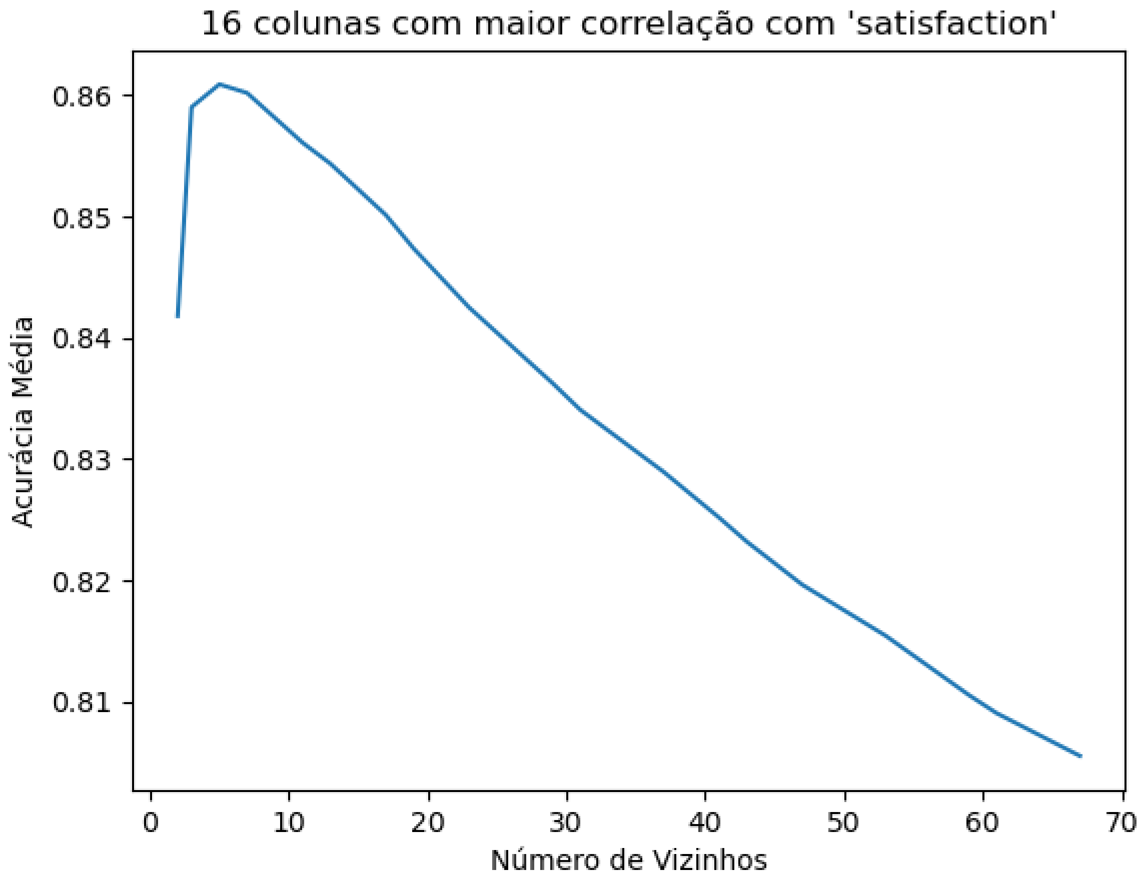

```
Modelo [14]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling', 'Inflight service', 'Checkin service', 'Foo
d and drink']
2 vizinhos: [0.8339672774425164, 0.8541132512678994, 0.857250476242657, 0.85716362601896
92, 0.8537850808485039, 0.8520957584120052, 0.8470858077265548, 0.845454453460535, 0.841
0912758979657, 0.8349229513738636, 0.8330502489581552, 0.827258413309673, 0.824584529297
822, 0.82294350761336, 0.8196807729909381, 0.8155395891733861, 0.8114659962183854, 0.810
2014345237629, 0.8067746165552421]
3 vizinhos: [0.8339672774425164, 0.8541132512678994, 0.857250476242657, 0.85716362601896
92, 0.8537850808485039, 0.8520957584120052, 0.8470858077265548, 0.845454453460535, 0.841
0912758979657, 0.8349229513738636, 0.8330502489581552, 0.827258413309673, 0.824584529297
822, 0.82294350761336, 0.8196807729909381, 0.8155395891733861, 0.8114659962183854, 0.810
2014345237629, 0.8067746165552421]
5 vizinhos: [0.8339672774425164, 0.8541132512678994, 0.857250476242657, 0.85716362601896
92, 0.8537850808485039, 0.8520957584120052, 0.8470858077265548, 0.845454453460535, 0.841
0912758979657, 0.8349229513738636, 0.8330502489581552, 0.827258413309673, 0.824584529297
822, 0.82294350761336, 0.8196807729909381, 0.8155395891733861, 0.8114659962183854, 0.810
2014345237629, 0.8067746165552421]
```



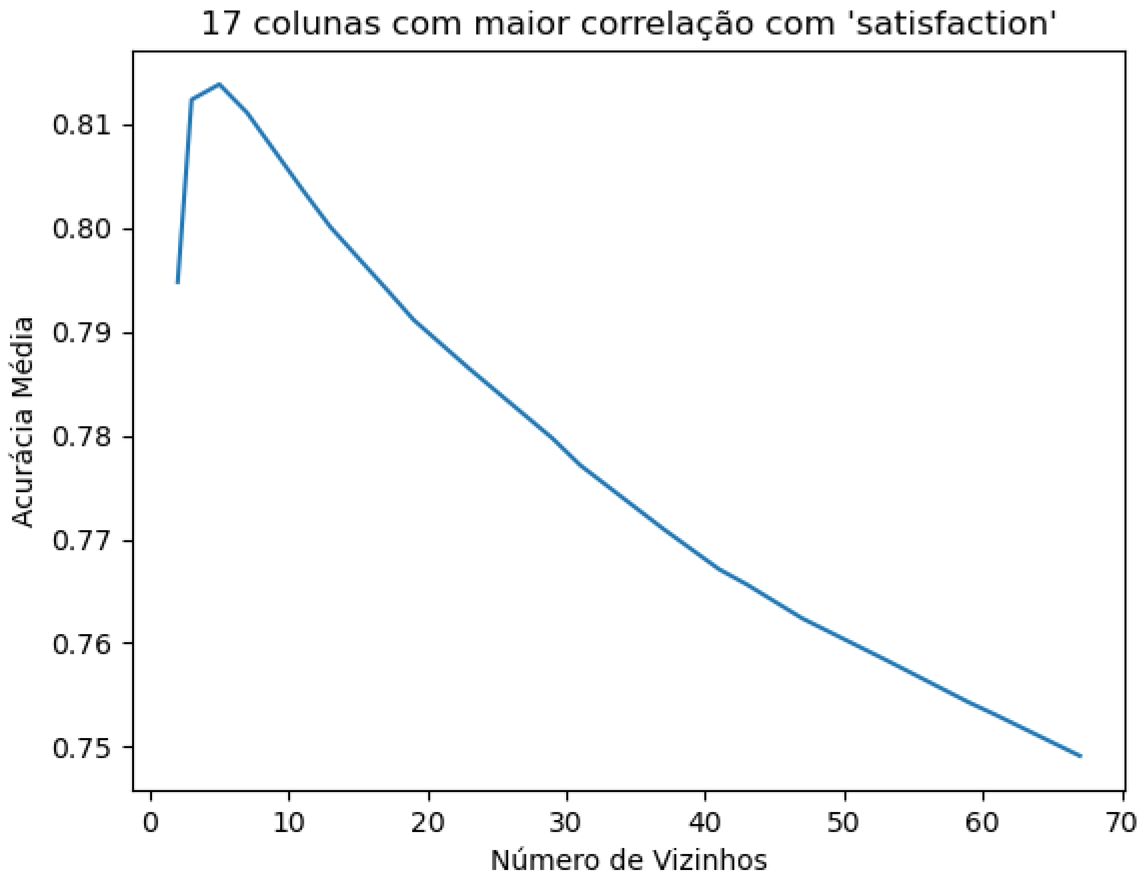

```
Modelo [15]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling', 'Inflight service', 'Checkin service', 'Foo
d and drink', 'Customer Type']
2 vizinhos: [0.8363901812946123, 0.8556577431968397, 0.8579744219209902, 0.8582158082744
531, 0.8553488740695517, 0.853176904719042, 0.8487847463325308, 0.8465259351445278, 0.84
20469311933256, 0.8358206944067067, 0.8339769513835543, 0.8278858497320704, 0.8250671911
206229, 0.8238895001492743, 0.820424066551347, 0.8159064199485572, 0.8120548449653576,
0.8108385476825962, 0.8073248254460245]
3 vizinhos: [0.8363901812946123, 0.8556577431968397, 0.8579744219209902, 0.8582158082744
531, 0.8553488740695517, 0.853176904719042, 0.8487847463325308, 0.8465259351445278, 0.84
20469311933256, 0.8358206944067067, 0.8339769513835543, 0.8278858497320704, 0.8250671911
206229, 0.8238895001492743, 0.820424066551347, 0.8159064199485572, 0.8120548449653576,
0.8108385476825962, 0.8073248254460245]
5 vizinhos: [0.8363901812946123, 0.8556577431968397, 0.8579744219209902, 0.8582158082744
531, 0.8553488740695517, 0.853176904719042, 0.8487847463325308, 0.8465259351445278, 0.84
20469311933256, 0.8358206944067067, 0.8339769513835543, 0.8278858497320704, 0.8250671911
206229, 0.8238895001492743, 0.820424066551347, 0.8159064199485572, 0.8120548449653576,
0.8108385476825962, 0.8073248254460245]
```



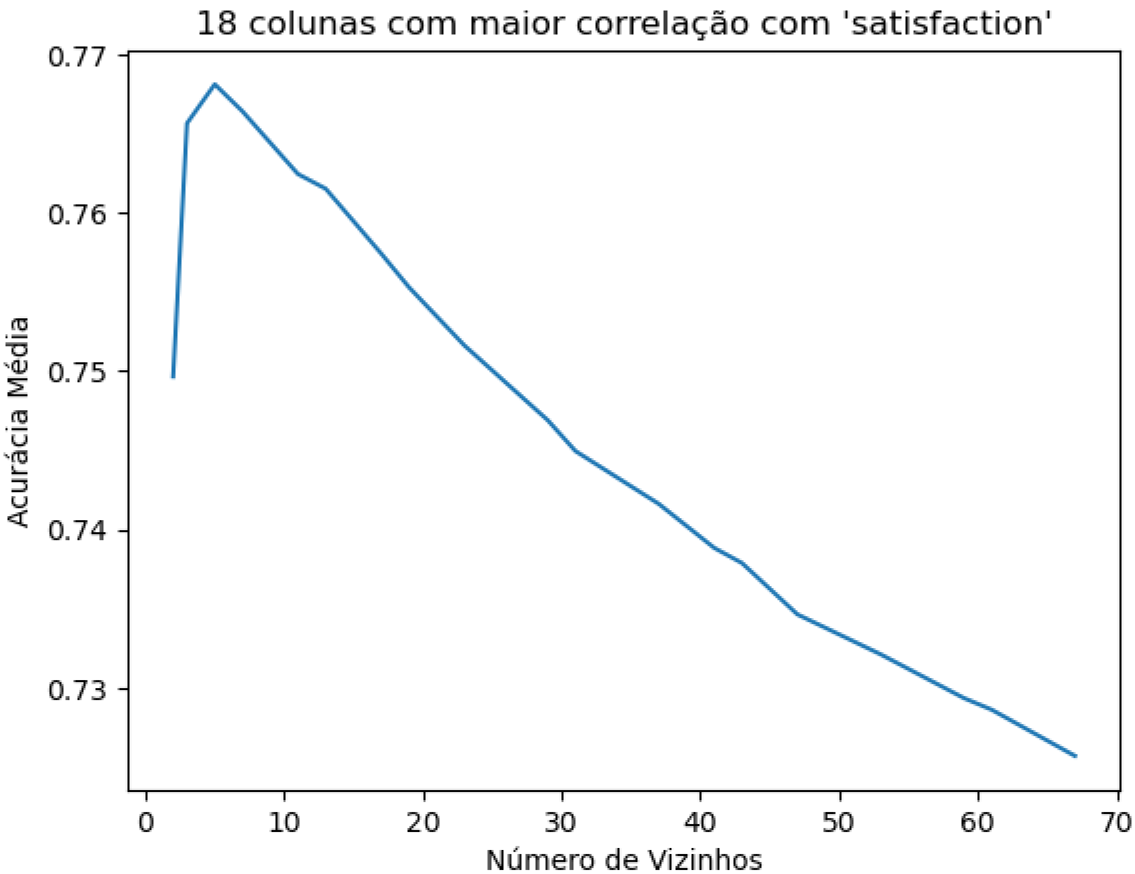

```
Modelo [16]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling', 'Inflight service', 'Checkin service', 'Foo
d and drink', 'Customer Type', 'Ease of Online booking']
2 vizinhos: [0.8417766087422909, 0.8589976755332968, 0.8608799987774793, 0.8601656897682
093, 0.8560632045102071, 0.8543353111706719, 0.8500975687118173, 0.8473368065222975, 0.8
424716453452336, 0.8362454141494107, 0.8340445161557237, 0.8289284074318827, 0.825154064
639295, 0.8231655227897626, 0.819613181196587, 0.8154141102751007, 0.8105199785238881,
0.8090237649837997, 0.8055003874421771]
3 vizinhos: [0.8417766087422909, 0.8589976755332968, 0.8608799987774793, 0.8601656897682
093, 0.8560632045102071, 0.8543353111706719, 0.8500975687118173, 0.8473368065222975, 0.8
424716453452336, 0.8362454141494107, 0.8340445161557237, 0.8289284074318827, 0.825154064
639295, 0.8231655227897626, 0.819613181196587, 0.8154141102751007, 0.8105199785238881,
0.8090237649837997, 0.8055003874421771]
5 vizinhos: [0.8417766087422909, 0.8589976755332968, 0.8608799987774793, 0.8601656897682
093, 0.8560632045102071, 0.8543353111706719, 0.8500975687118173, 0.8473368065222975, 0.8
424716453452336, 0.8362454141494107, 0.8340445161557237, 0.8289284074318827, 0.825154064
639295, 0.8231655227897626, 0.819613181196587, 0.8154141102751007, 0.8105199785238881,
0.8090237649837997, 0.8055003874421771]
```



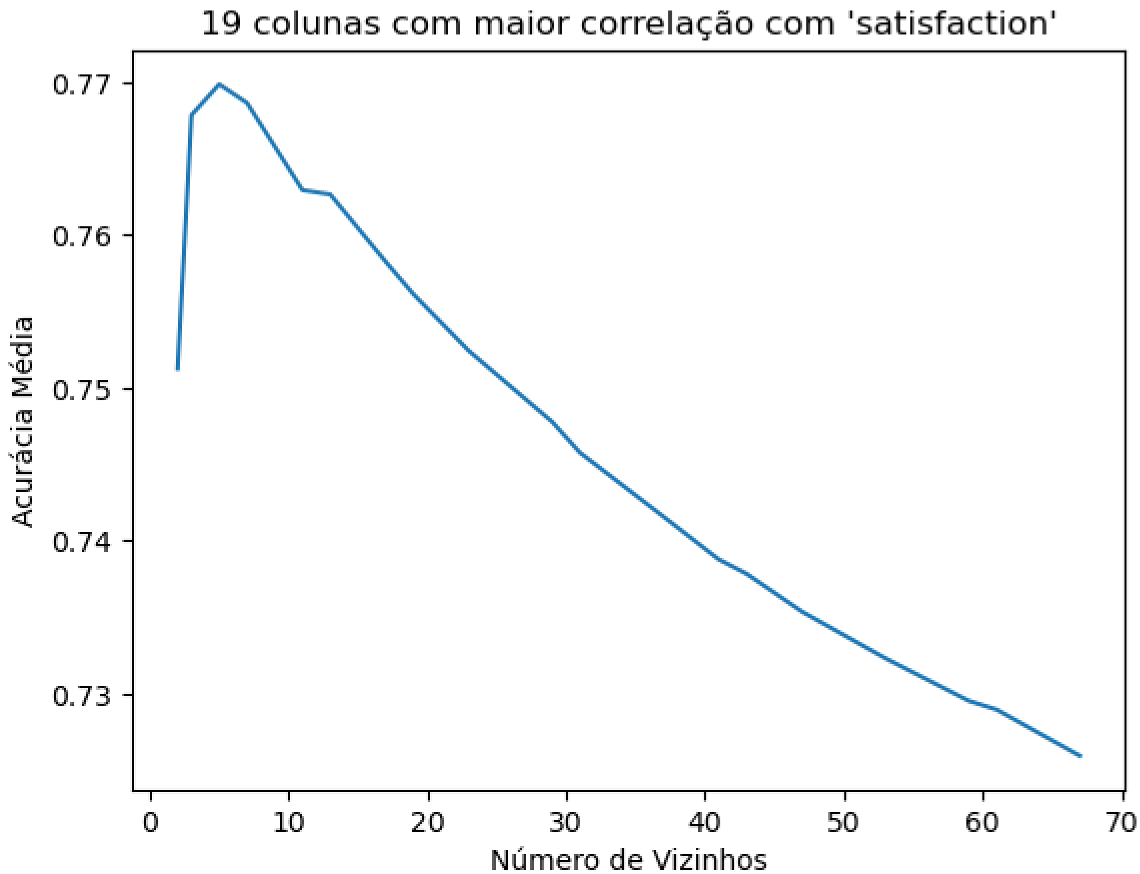

```
Modelo [17]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling', 'Inflight service', 'Checkin service', 'Foo
d and drink', 'Customer Type', 'Ease of Online booking', 'Age']
2 vizinhos: [0.794775785777089, 0.8123444006871461, 0.8138213343665125, 0.81107986508290
59, 0.8036469984319681, 0.8000753676600766, 0.7941483530818891, 0.7911269563593629, 0.78
64355356970475, 0.7797170339633415, 0.777129991788984, 0.7709809965109705, 0.76711013328
08543, 0.7656428819287203, 0.7623704500702763, 0.7583837557925308, 0.7542811894679836,
0.7530359588830484, 0.7491264613875387]
3 vizinhos: [0.794775785777089, 0.8123444006871461, 0.8138213343665125, 0.81107986508290
59, 0.8036469984319681, 0.8000753676600766, 0.7941483530818891, 0.7911269563593629, 0.78
64355356970475, 0.7797170339633415, 0.777129991788984, 0.7709809965109705, 0.76711013328
08543, 0.7656428819287203, 0.7623704500702763, 0.7583837557925308, 0.7542811894679836,
0.7530359588830484, 0.7491264613875387]
5 vizinhos: [0.794775785777089, 0.8123444006871461, 0.8138213343665125, 0.81107986508290
59, 0.8036469984319681, 0.8000753676600766, 0.7941483530818891, 0.7911269563593629, 0.78
64355356970475, 0.7797170339633415, 0.777129991788984, 0.7709809965109705, 0.76711013328
08543, 0.7656428819287203, 0.7623704500702763, 0.7583837557925308, 0.7542811894679836,
0.7530359588830484, 0.7491264613875387]
```



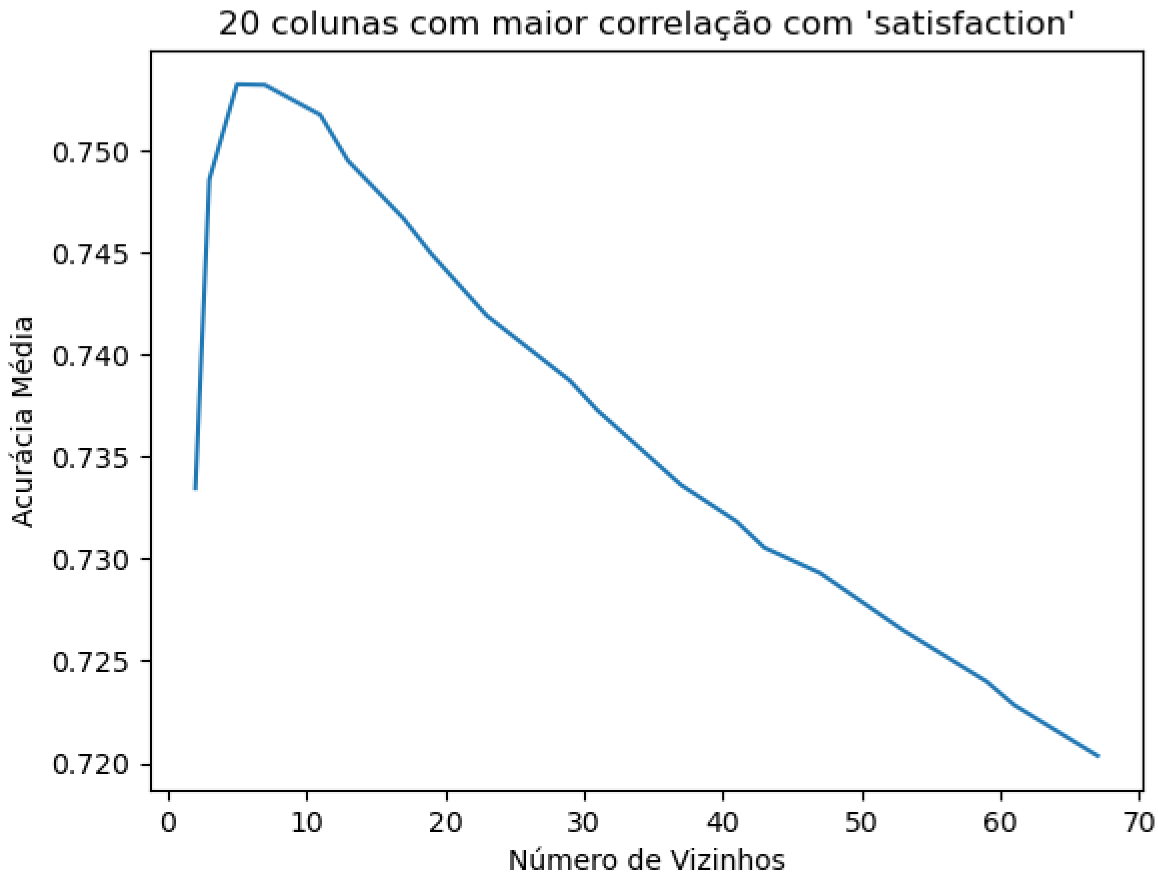

```
Modelo [18]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling', 'Inflight service', 'Checkin service', 'Foo
d and drink', 'Customer Type', 'Ease of Online booking', 'Age', 'Arrival Delay in Minute
s']
2 vizinhos: [0.7496572972376621, 0.7656427831579873, 0.7681043669336459, 0.7663958037719
983, 0.7624379887520635, 0.7615112723496738, 0.7574183808979639, 0.7552947700710516, 0.7
516169113758167, 0.7469062285569671, 0.7449563191092297, 0.7416260206464377, 0.738845919
8928357, 0.737890280438065, 0.7346468592211424, 0.732127365978365, 0.7293665963344503,
0.7286329683288849, 0.7257177538715331]
3 vizinhos: [0.7496572972376621, 0.7656427831579873, 0.7681043669336459, 0.7663958037719
983, 0.7624379887520635, 0.7615112723496738, 0.7574183808979639, 0.7552947700710516, 0.7
516169113758167, 0.7469062285569671, 0.7449563191092297, 0.7416260206464377, 0.738845919
8928357, 0.737890280438065, 0.7346468592211424, 0.732127365978365, 0.7293665963344503,
0.7286329683288849, 0.7257177538715331]
5 vizinhos: [0.7496572972376621, 0.7656427831579873, 0.7681043669336459, 0.7663958037719
983, 0.7624379887520635, 0.7615112723496738, 0.7574183808979639, 0.7552947700710516, 0.7
516169113758167, 0.7469062285569671, 0.7449563191092297, 0.7416260206464377, 0.738845919
8928357, 0.737890280438065, 0.7346468592211424, 0.732127365978365, 0.7293665963344503,
0.7286329683288849, 0.7257177538715331]
```



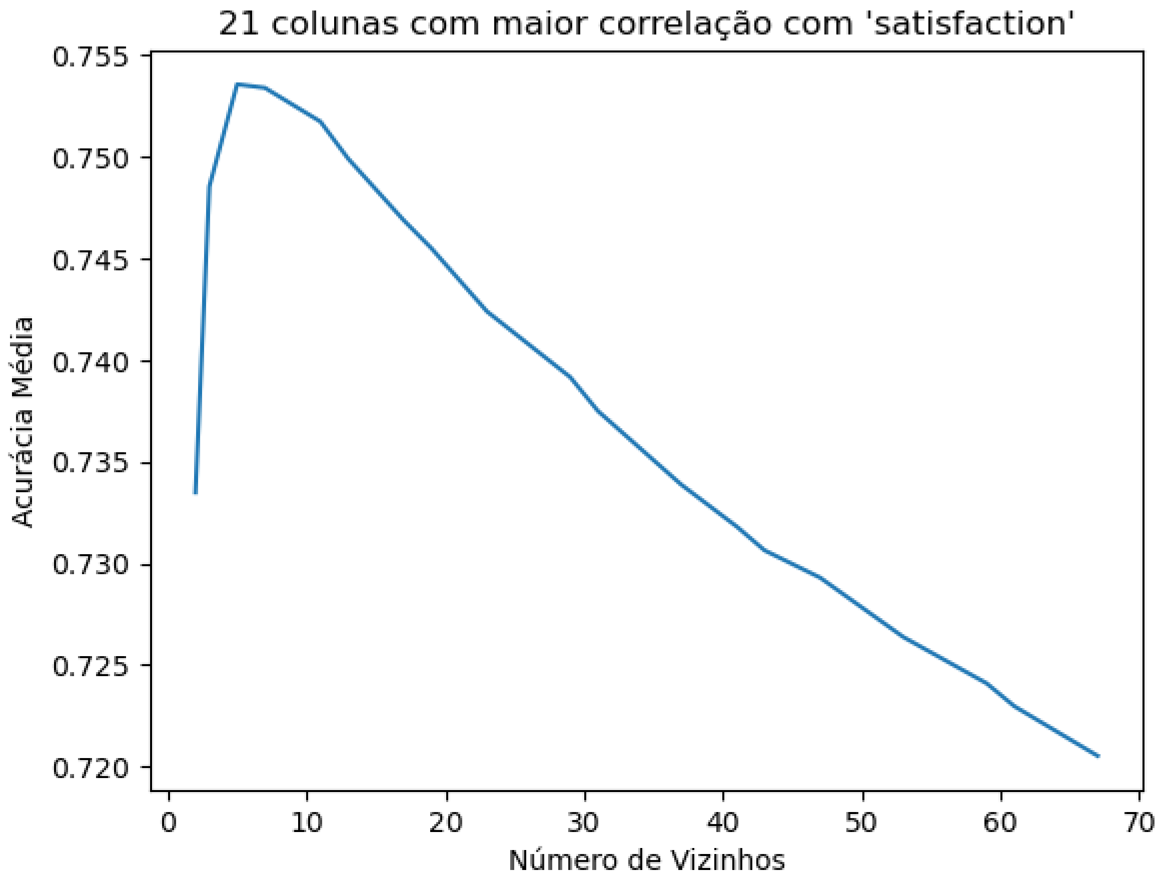

```
Modelo [19]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling', 'Inflight service', 'Checkin service', 'Foo
d and drink', 'Customer Type', 'Ease of Online booking', 'Age', 'Arrival Delay in Minute
s', 'Departure/Arrival time convenient']
2 vizinhos: [0.7512500694190528, 0.7678437258780437, 0.7698612634603078, 0.7686449382235
655, 0.7629399872753478, 0.762660018837256, 0.7582485731356279, 0.7561249241049415, 0.75
24084721434852, 0.747775048537429, 0.7457575351819488, 0.7415873956990378, 0.73877834766
62712, 0.7378419908676208, 0.7353611412082308, 0.7323204189668133, 0.7295403526897879,
0.7289708108257196, 0.7259493805584162]
3 vizinhos: [0.7512500694190528, 0.7678437258780437, 0.7698612634603078, 0.7686449382235
655, 0.7629399872753478, 0.762660018837256, 0.7582485731356279, 0.7561249241049415, 0.75
24084721434852, 0.747775048537429, 0.7457575351819488, 0.7415873956990378, 0.73877834766
62712, 0.7378419908676208, 0.7353611412082308, 0.7323204189668133, 0.7295403526897879,
0.7289708108257196, 0.7259493805584162]
5 vizinhos: [0.7512500694190528, 0.7678437258780437, 0.7698612634603078, 0.7686449382235
655, 0.7629399872753478, 0.762660018837256, 0.7582485731356279, 0.7561249241049415, 0.75
24084721434852, 0.747775048537429, 0.7457575351819488, 0.7415873956990378, 0.73877834766
62712, 0.7378419908676208, 0.7353611412082308, 0.7323204189668133, 0.7295403526897879,
0.7289708108257196, 0.7259493805584162]
```




```
Modelo [20]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling', 'Inflight service', 'Checkin service', 'Foo
d and drink', 'Customer Type', 'Ease of Online booking', 'Age', 'Arrival Delay in Minute
s', 'Departure/Arrival time convenient', 'Departure Delay in Minutes']
2 vizinhos: [0.7334401576082723, 0.7485568291389315, 0.7532289466455409, 0.7531999919119
814, 0.7517327741046247, 0.7494836107672771, 0.7466553098959701, 0.7449273308308929, 0.7
418963225978864, 0.7387108388020638, 0.737243567882143, 0.7336043173619193, 0.7318184884
64883, 0.7305346161601591, 0.7292990334258797, 0.7264900068244985, 0.723989858668399, 0.
7228314727163555, 0.720350629579561]
3 vizinhos: [0.7334401576082723, 0.7485568291389315, 0.7532289466455409, 0.7531999919119
814, 0.7517327741046247, 0.7494836107672771, 0.7466553098959701, 0.7449273308308929, 0.7
418963225978864, 0.7387108388020638, 0.737243567882143, 0.7336043173619193, 0.7318184884
64883, 0.7305346161601591, 0.7292990334258797, 0.7264900068244985, 0.723989858668399, 0.
7228314727163555, 0.720350629579561]
5 vizinhos: [0.7334401576082723, 0.7485568291389315, 0.7532289466455409, 0.7531999919119
814, 0.7517327741046247, 0.7494836107672771, 0.7466553098959701, 0.7449273308308929, 0.7
418963225978864, 0.7387108388020638, 0.737243567882143, 0.7336043173619193, 0.7318184884
64883, 0.7305346161601591, 0.7292990334258797, 0.7264900068244985, 0.723989858668399, 0.
7228314727163555, 0.720350629579561]
```




```
Modelo [21]: ['Online boarding', 'Class', 'Type of Travel', 'Inflight entertainment', 'S
eat comfort', 'On-board service', 'Leg room service', 'Cleanliness', 'Flight Distance',
'Inflight wifi service', 'Baggage handling', 'Inflight service', 'Checkin service', 'Foo
d and drink', 'Customer Type', 'Ease of Online booking', 'Age', 'Arrival Delay in Minute
s', 'Departure/Arrival time convenient', 'Departure Delay in Minutes', 'Gender']
2 vizinhos: [0.7334884192247355, 0.7485471794246773, 0.7535764593562161, 0.7534027253640
633, 0.7517424387276689, 0.7499179932694268, 0.7468869682640317, 0.7455065177502188, 0.7
423982829173967, 0.7391645225963209, 0.7374945461782995, 0.7338746118589732, 0.731808824
773638, 0.73064078910734, 0.7292990278350834, 0.726374162731678, 0.7240960502515671, 0.7
229666181012837, 0.7205243877984973]
3 vizinhos: [0.7334884192247355, 0.7485471794246773, 0.7535764593562161, 0.7534027253640
633, 0.7517424387276689, 0.7499179932694268, 0.7468869682640317, 0.7455065177502188, 0.7
423982829173967, 0.7391645225963209, 0.7374945461782995, 0.7338746118589732, 0.731808824
773638, 0.73064078910734, 0.7292990278350834, 0.726374162731678, 0.7240960502515671, 0.7
229666181012837, 0.7205243877984973]
5 vizinhos: [0.7334884192247355, 0.7485471794246773, 0.7535764593562161, 0.7534027253640
633, 0.7517424387276689, 0.7499179932694268, 0.7468869682640317, 0.7455065177502188, 0.7
423982829173967, 0.7391645225963209, 0.7374945461782995, 0.7338746118589732, 0.731808824
773638, 0.73064078910734, 0.7292990278350834, 0.726374162731678, 0.7240960502515671, 0.7
229666181012837, 0.7205243877984973]
```

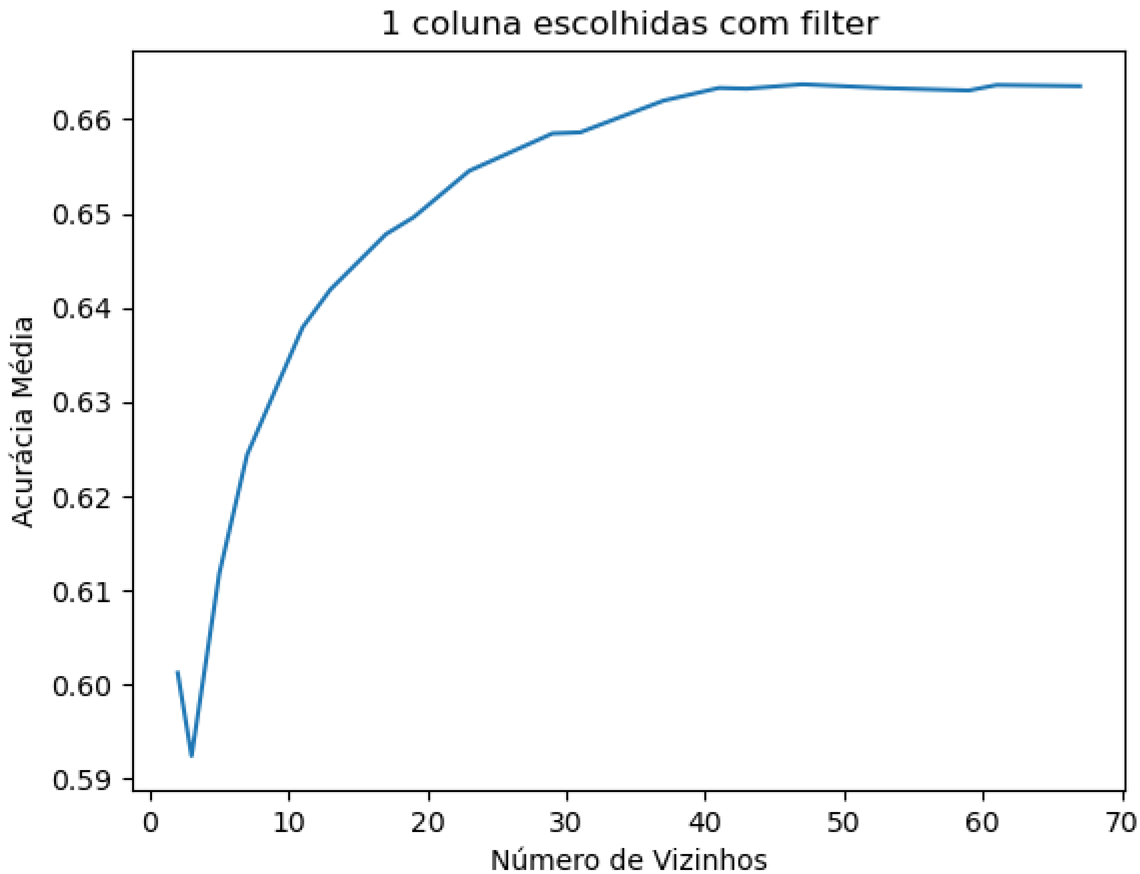


Filtros

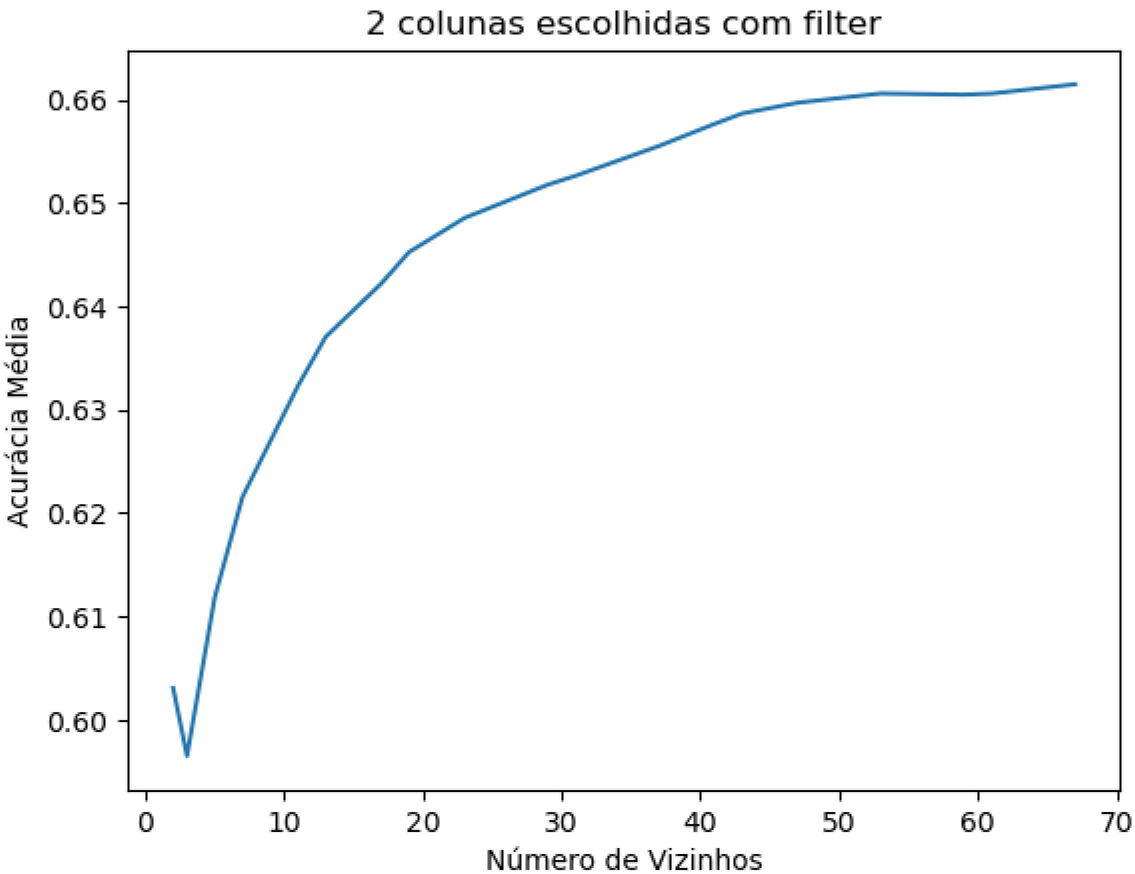
In [80]:
```
from sklearn.feature_selection import SelectKBest, chi2

# Aplicando o Filter

cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Obtém todas as colunas que não são `satisfaction`
todas_variaveis = list()
todas_variaveis.extend( [coluna for coluna in df_encod.columns if coluna != 'satisfactio

y = df_encod['satisfaction']

for n_colunas in range(1, 23):
  # Realiza a seleção de características usando SelectKBest e chi2
  X = df_encod[todas_variaveis]

  seletor = SelectKBest(chi2, k=n_colunas)
  seletor.fit_transform(X, y)

  # Obtém os índices das características selecionadas
  indice_caracteristicas_selecionadas = seletor.get_support(indices=True)

  # Obtém os nomes das características selecionadas
  caracteristicas_selecionadas = X.columns[indice_caracteristicas_selecionadas]
  modelo = list(caracteristicas_selecionadas)

  print(f'Modelo [{len(modelo)}]: {modelo}')

  X = df_encod[modelo]

  resultados = []
```

```python
  # Testa-se 10 `n_vizinhos` distintos por modelo.
  # Como a mudança entre números de vizinhos adjacentes é pequena, pula-se de 3 em 3.
  for n_vizinhos in primos:
    knn_classifier = KNeighborsClassifier(n_neighbors=n_vizinhos)

    values = cross_val_score(knn_classifier, X, y, cv=cv, scoring='accuracy')

    resultados.append(values.mean())

    print(f"{n_vizinhos} vizinhos: ", values.mean())

  plt.plot(primos, resultados);
  plt.title(f"{len(modelo)} coluna{'s' if len(modelo) > 1 else ''} escolhidas com filter
  plt.xlabel("Número de Vizinhos")
  plt.ylabel("Acurácia Média")
  plt.show()
  print('\n')
```

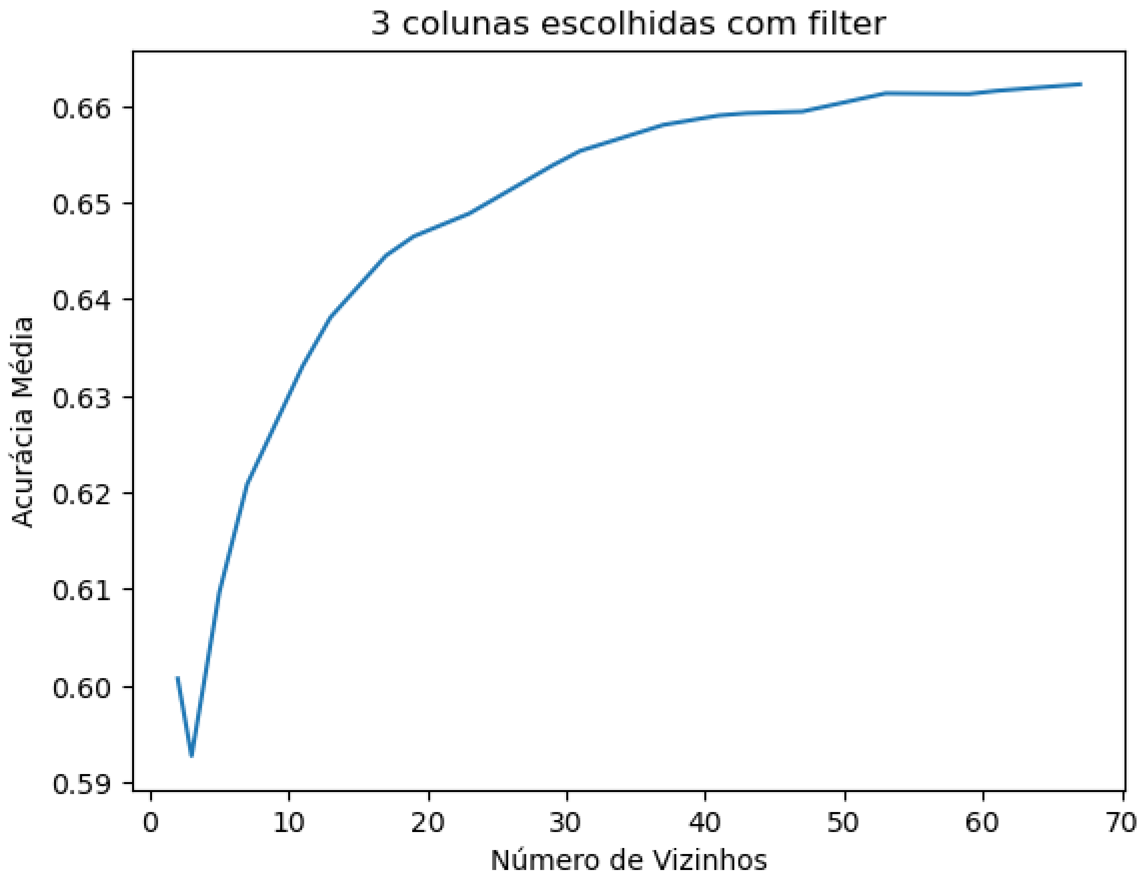
```
Modelo [1]: ['Flight Distance']
2 vizinhos:  0.6011930824333084
3 vizinhos:  0.5923701574852747
5 vizinhos:  0.6117728228414588
7 vizinhos:  0.6243701157406631
11 vizinhos:  0.637903672258581
13 vizinhos:  0.6419193613372588
17 vizinhos:  0.6477980528002248
19 vizinhos:  0.6496032109433499
23 vizinhos:  0.6545262862465295
29 vizinhos:  0.6585033606276004
31 vizinhos:  0.6586191823572363
37 vizinhos:  0.661988120676218
41 vizinhos:  0.663339552162315
43 vizinhos:  0.6632623488574836
47 vizinhos:  0.6637256954111863
53 vizinhos:  0.6633202611200005
59 vizinhos:  0.663069322891217
61 vizinhos:  0.6636485181967371
67 vizinhos:  0.6635327123076906
```



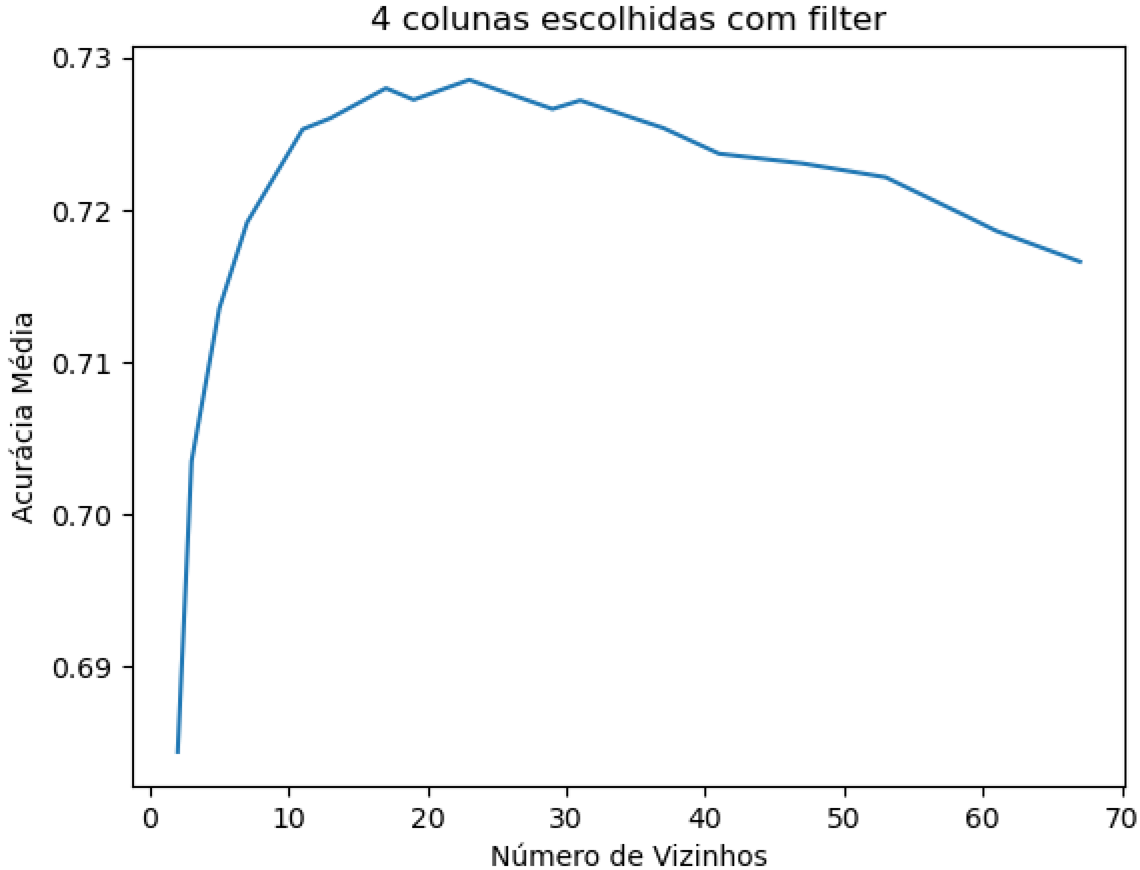

```
Modelo [2]: ['Flight Distance', 'Arrival Delay in Minutes']
2 vizinhos:  0.6031043762516395
3 vizinhos:  0.5965307050254921
5 vizinhos:  0.6118694690719018
7 vizinhos:  0.6215708050112915
11 vizinhos:  0.6323146222429454
13 vizinhos:  0.6370542774995425
17 vizinhos:  0.6421993307071501
19 vizinhos:  0.6452400781071501
23 vizinhos:  0.6485413556786275
29 vizinhos:  0.6517559078875326
31 vizinhos:  0.652615026904775
37 vizinhos:  0.6554916490277045
41 vizinhos:  0.6576249971580118
43 vizinhos:  0.6586386047832615
47 vizinhos:  0.6596811438470864
53 vizinhos:  0.6605788980615219
59 vizinhos:  0.6604630362645133
61 vizinhos:  0.6605692325066782
67 vizinhos:  0.6614669569035339
```



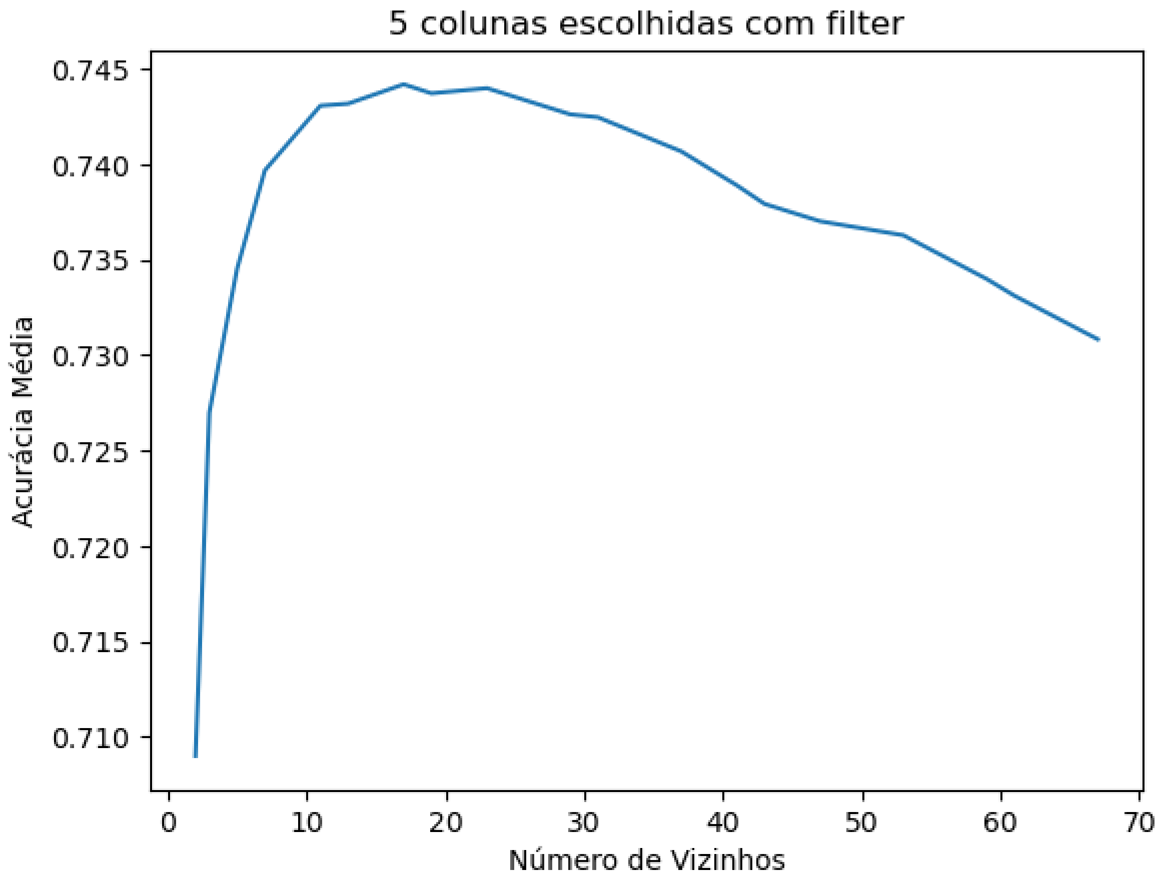

```
Modelo [3]: ['Flight Distance', 'Departure Delay in Minutes', 'Arrival Delay in Minute
s']
2 vizinhos:  0.6007394303202296
3 vizinhos:  0.5927370320550163
5 vizinhos:  0.6096685412606351
7 vizinhos:  0.6208467987659995
11 vizinhos:  0.6330868696051145
13 vizinhos:  0.6381160759245035
17 vizinhos:  0.6445161063384347
19 vizinhos:  0.6465046453925689
23 vizinhos:  0.6488696565499346
29 vizinhos:  0.6538506543654241
31 vizinhos:  0.6553565306649581
37 vizinhos:  0.6580497523090921
41 vizinhos:  0.6590246930559702
43 vizinhos:  0.6592563775144139
47 vizinhos:  0.6594301506421403
53 vizinhos:  0.6613124925223101
59 vizinhos:  0.6612449296137393
61 vizinhos:  0.6615731205327208
67 vizinhos:  0.6622488390711675
```



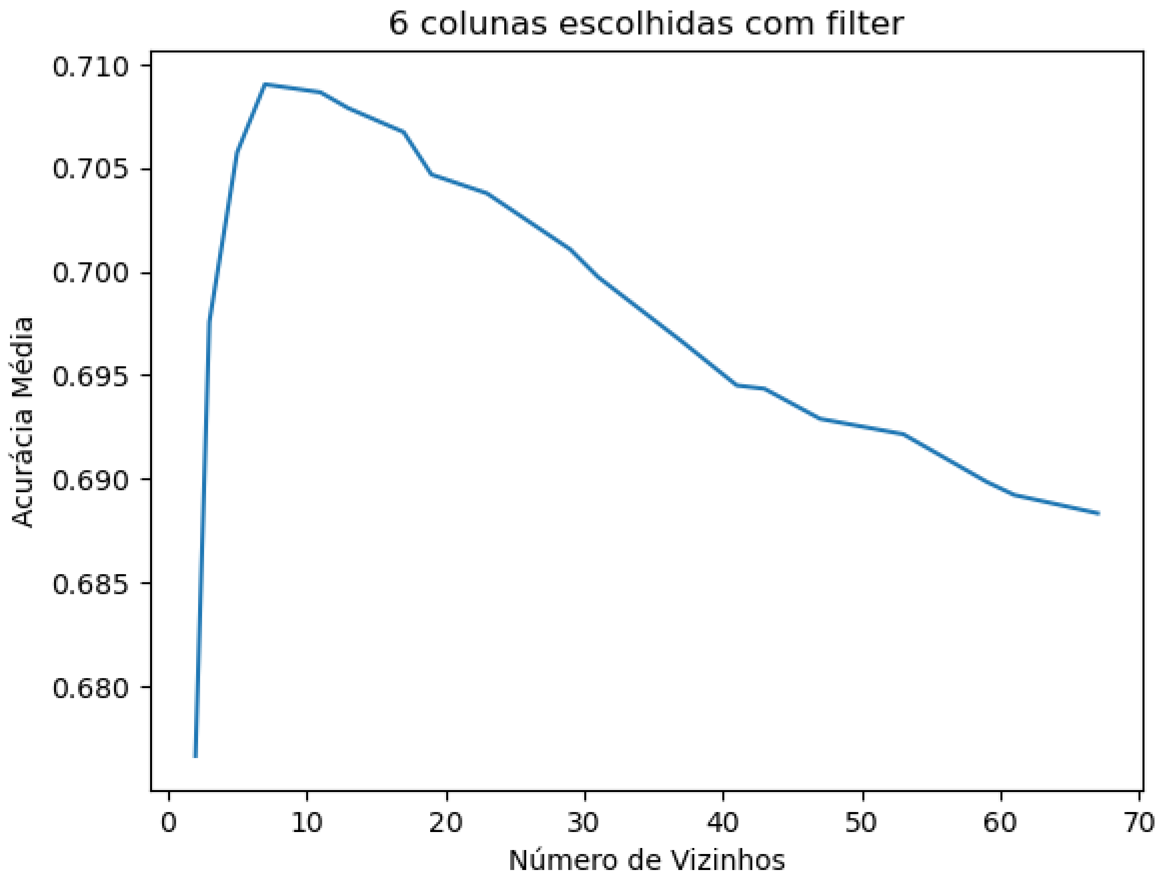

```
Modelo [4]: ['Flight Distance', 'Online boarding', 'Departure Delay in Minutes', 'Arriva
l Delay in Minutes']
2 vizinhos:  0.6844219284445174
3 vizinhos:  0.703496367473344
5 vizinhos:  0.713525918558499
7 vizinhos:  0.7191633000755503
11 vizinhos:  0.7252735651128354
13 vizinhos:  0.7259975573811369
17 vizinhos:  0.7279668221653453
19 vizinhos:  0.7272138919358728
23 vizinhos:  0.7285170636691054
29 vizinhos:  0.7266057791687679
31 vizinhos:  0.7271656573415913
37 vizinhos:  0.7253508765063935
41 vizinhos:  0.7236712298745314
43 vizinhos:  0.7234588746621762
47 vizinhos:  0.7230341633056663
53 vizinhos:  0.7221267975807507
59 vizinhos:  0.7194915291983058
61 vizinhos:  0.7185937945516573
67 vizinhos:  0.7165859849547946
```



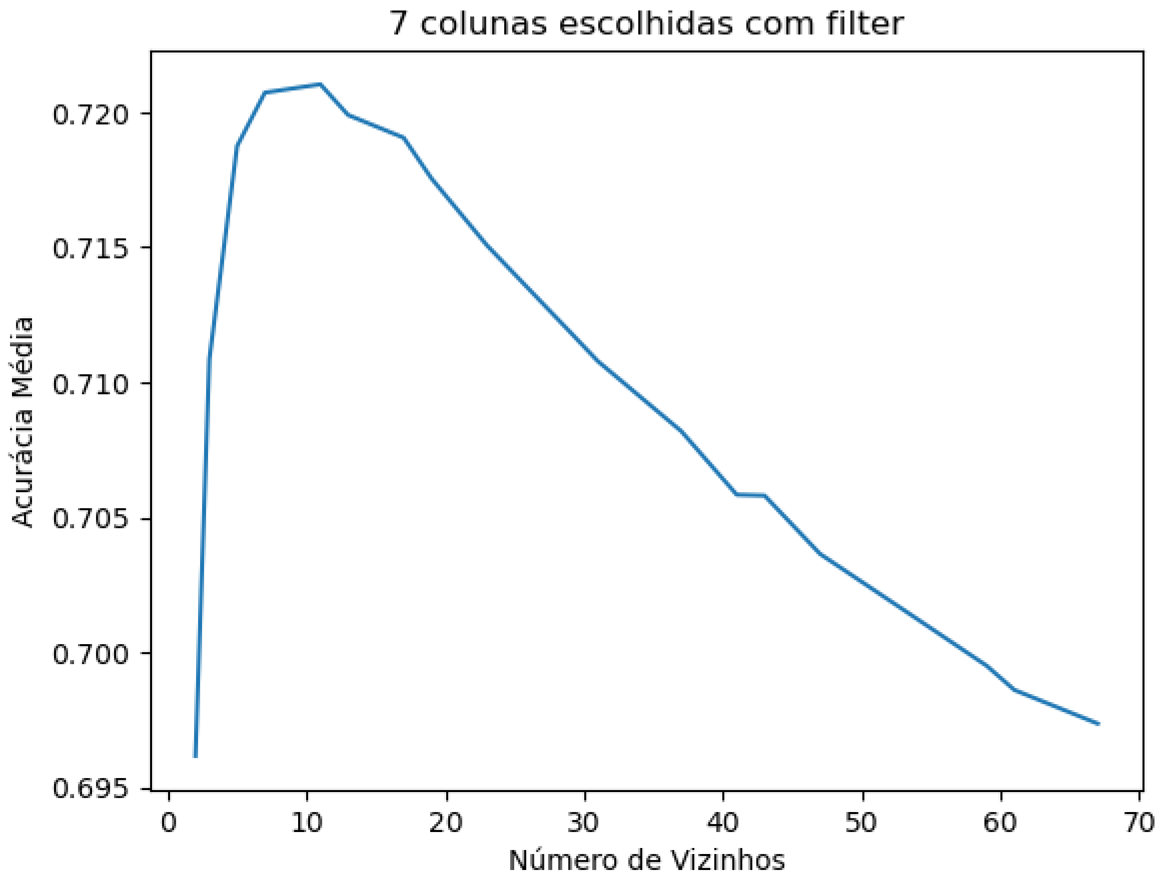

```
Modelo [5]: ['Class', 'Flight Distance', 'Online boarding', 'Departure Delay in Minute
s', 'Arrival Delay in Minutes']
2 vizinhos:  0.7089986734904199
3 vizinhos:  0.7269823053163627
5 vizinhos:  0.7345116784278383
7 vizinhos:  0.7396663813497003
11 vizinhos:  0.7430546256197864
13 vizinhos:  0.7431608097485596
17 vizinhos:  0.74417437171564
19 vizinhos:  0.7437013912882724
23 vizinhos:  0.7439716941715204
29 vizinhos:  0.7426009576661183
31 vizinhos:  0.7424465147162802
37 vizinhos:  0.7406606820920462
41 vizinhos:  0.7388845168862546
43 vizinhos:  0.7379095500489941
47 vizinhos:  0.7370021628926929
53 vizinhos:  0.7362685535231148
59 vizinhos:  0.733980767101966
61 vizinhos:  0.7331023635650047
67 vizinhos:  0.7308242641300852
```



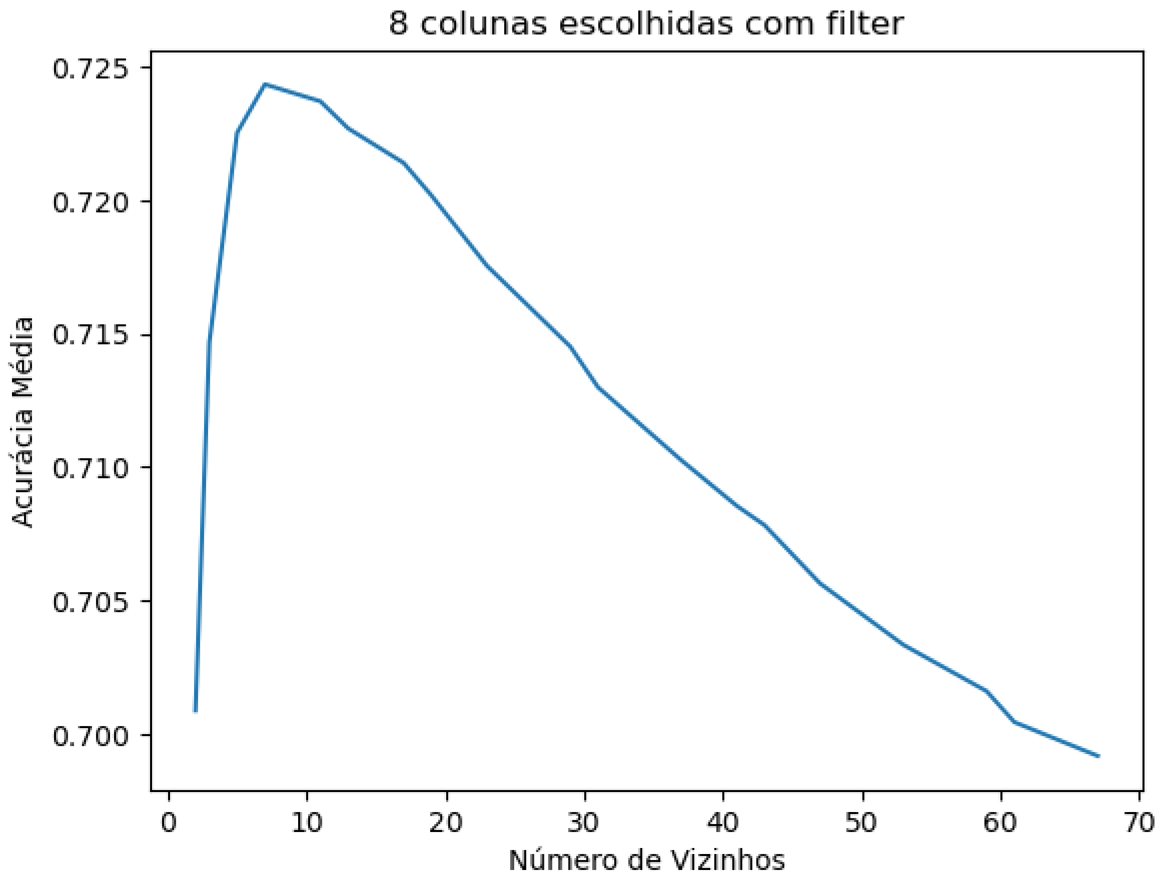

```
Modelo [6]: ['Age', 'Class', 'Flight Distance', 'Online boarding', 'Departure Delay in M
inutes', 'Arrival Delay in Minutes']
2 vizinhos:  0.6766318117795095
3 vizinhos:  0.6975692867374015
5 vizinhos:  0.7057454693119332
7 vizinhos:  0.709037109282548
11 vizinhos:  0.7086509911922596
13 vizinhos:  0.7078884475887083
17 vizinhos:  0.7067301352488147
19 vizinhos:  0.7046836923183578
23 vizinhos:  0.703776293048665
29 vizinhos:  0.7010734282128721
31 vizinhos:  0.6997413185184689
37 vizinhos:  0.696652379386958
41 vizinhos:  0.6945093573156128
43 vizinhos:  0.6943452348339403
47 vizinhos:  0.692897300614503
53 vizinhos:  0.6921540284854795
59 vizinhos:  0.6898565942136751
61 vizinhos:  0.6892194791912429
67 vizinhos:  0.6883410374505075
```



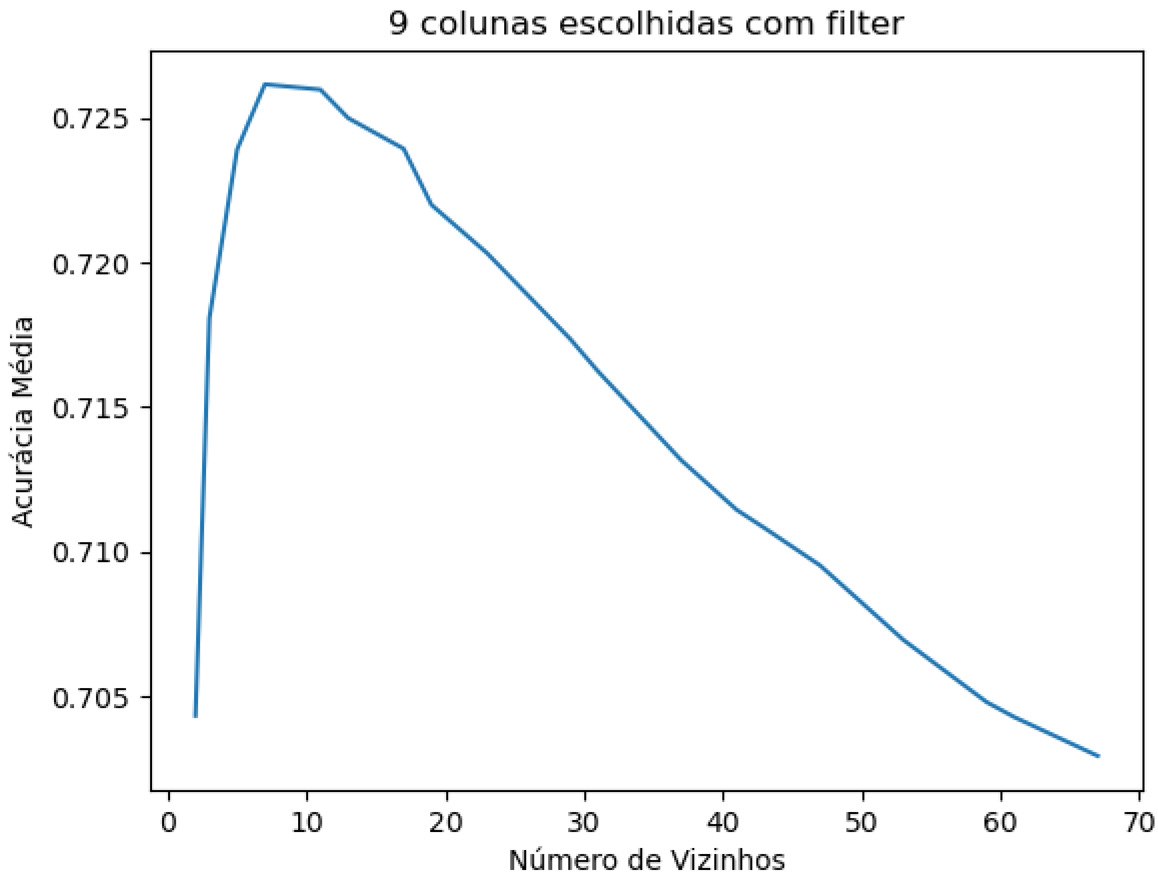

```
Modelo [7]: ['Age', 'Class', 'Flight Distance', 'Online boarding', 'Inflight entertainme
nt', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']
2 vizinhos:  0.6961599728063672
3 vizinhos:  0.710851984229482
5 vizinhos:  0.7187481340717656
7 vizinhos:  0.7207270513656265
11 vizinhos:  0.7210359307427076
13 vizinhos:  0.7198969094451285
17 vizinhos:  0.7190571075605827
19 vizinhos:  0.7175512182158578
23 vizinhos:  0.7150703834652574
29 vizinhos:  0.7118559207090919
31 vizinhos:  0.71077479676524
37 vizinhos:  0.7081878011808508
41 vizinhos:  0.7058323978067679
43 vizinhos:  0.7058034440050079
47 vizinhos:  0.7036411905265078
53 vizinhos:  0.7015754248725578
59 vizinhos:  0.6995000104361531
61 vizinhos:  0.6986022739259056
67 vizinhos:  0.6973570330911774
```



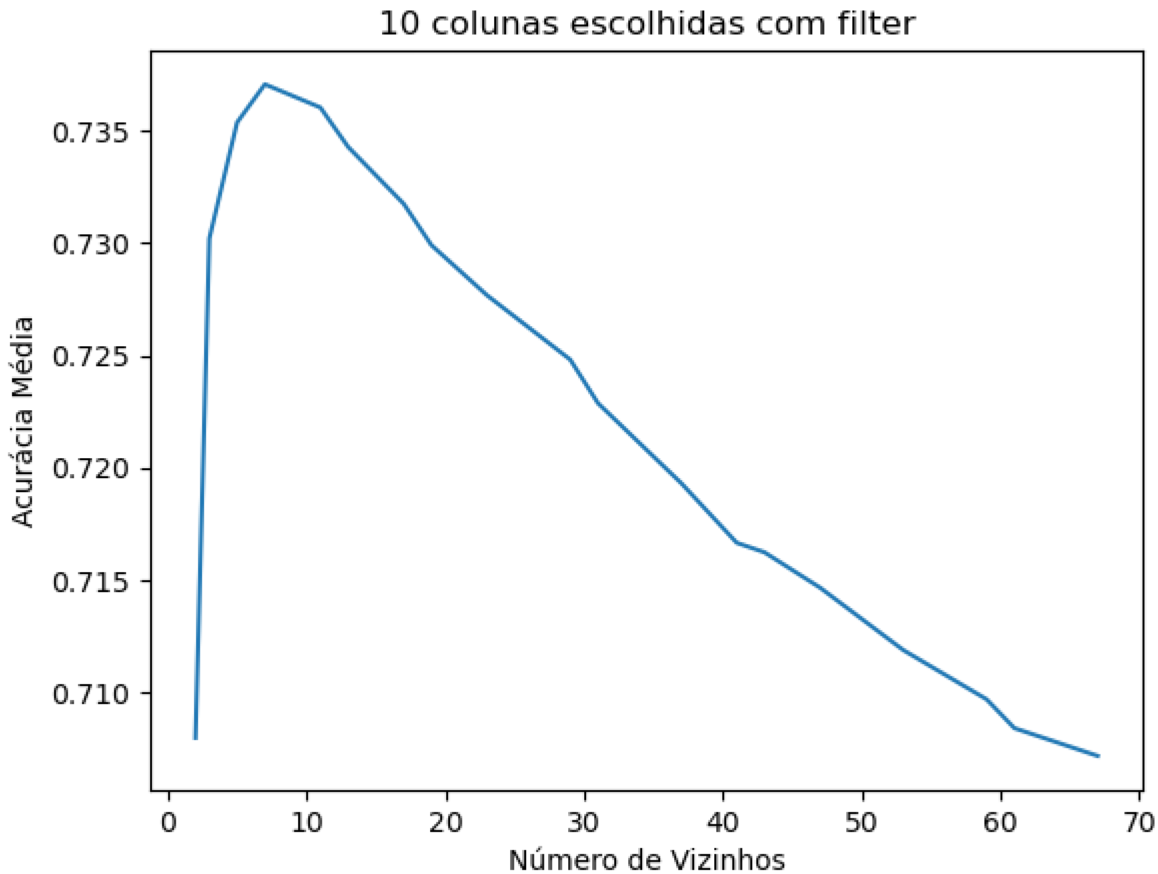

```
Modelo [8]: ['Age', 'Type of Travel', 'Class', 'Flight Distance', 'Online boarding', 'In
flight entertainment', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']
2 vizinhos:  0.7008899904621018
3 vizinhos:  0.7146745867749342
5 vizinhos:  0.722522495500341
7 vizinhos:  0.7243372837899337
11 vizinhos:  0.7237001715628997
13 vizinhos:  0.722686634754402
17 vizinhos:  0.7213931677115866
19 vizinhos:  0.7201768555200353
23 vizinhos:  0.7175415824785938
29 vizinhos:  0.7145201605974846
31 vizinhos:  0.7129950090962254
37 vizinhos:  0.7102632007084657
41 vizinhos:  0.7085545620710694
43 vizinhos:  0.7078305986885483
47 vizinhos:  0.7056490178275582
53 vizinhos:  0.7033515807603558
59 vizinhos:  0.7016140377065658
61 vizinhos:  0.7004556601407166
67 vizinhos:  0.6991911077640879
```



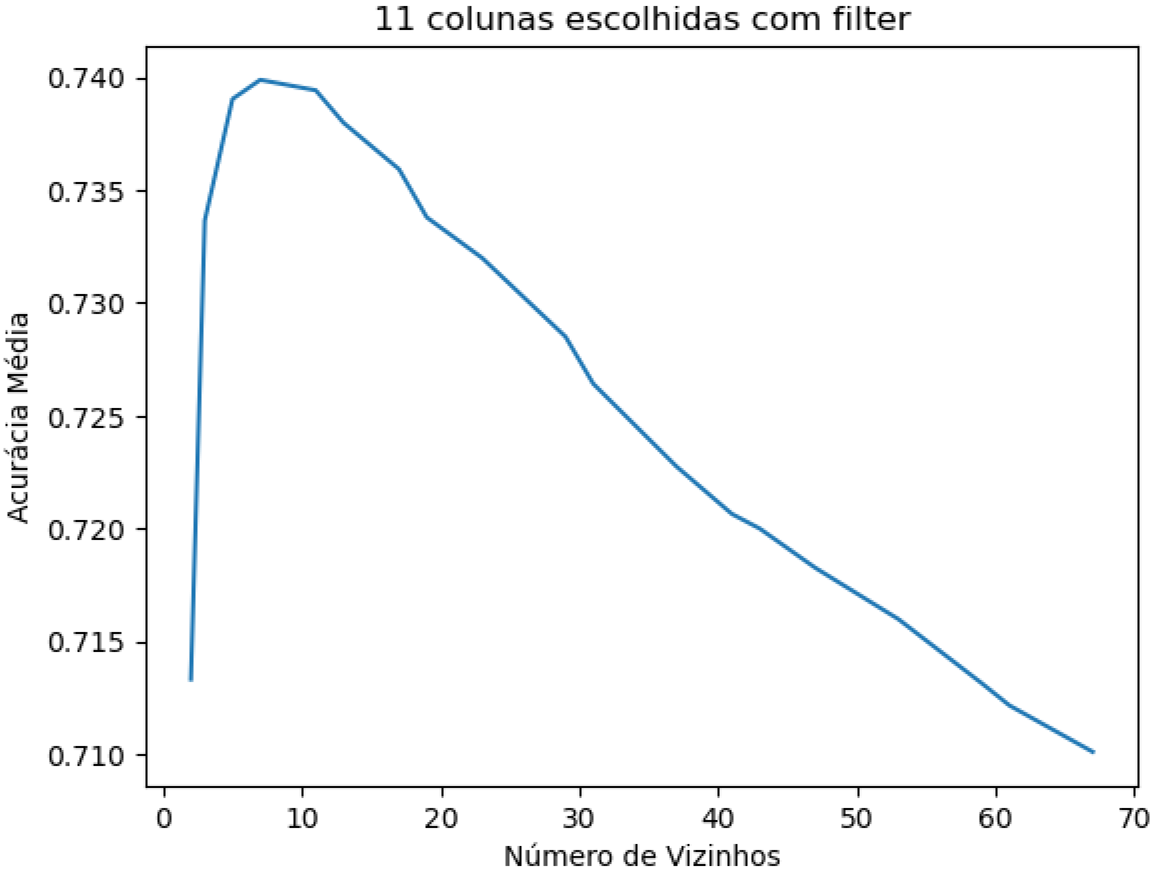

```
Modelo [9]: ['Age', 'Type of Travel', 'Class', 'Flight Distance', 'Online boarding', 'Se
at comfort', 'Inflight entertainment', 'Departure Delay in Minutes', 'Arrival Delay in M
inutes']
2 vizinhos:  0.7043071848067505
3 vizinhos:  0.7180628189316287
5 vizinhos:  0.723902926446367
7 vizinhos:  0.7261617273845771
11 vizinhos:  0.7259783073379945
13 vizinhos:  0.7249937028998714
17 vizinhos:  0.723931933360691
19 vizinhos:  0.72199164101423
23 vizinhos:  0.7203216860275939
29 vizinhos:  0.7173485276265468
31 vizinhos:  0.7162287573039092
37 vizinhos:  0.71314945763686
41 vizinhos:  0.7114311953755916
43 vizinhos:  0.7108037487034011
47 vizinhos:  0.7095102565020028
53 vizinhos:  0.7069328733598934
59 vizinhos:  0.7047801969152968
61 vizinhos:  0.7042589688484562
67 vizinhos:  0.7029171917356105
```



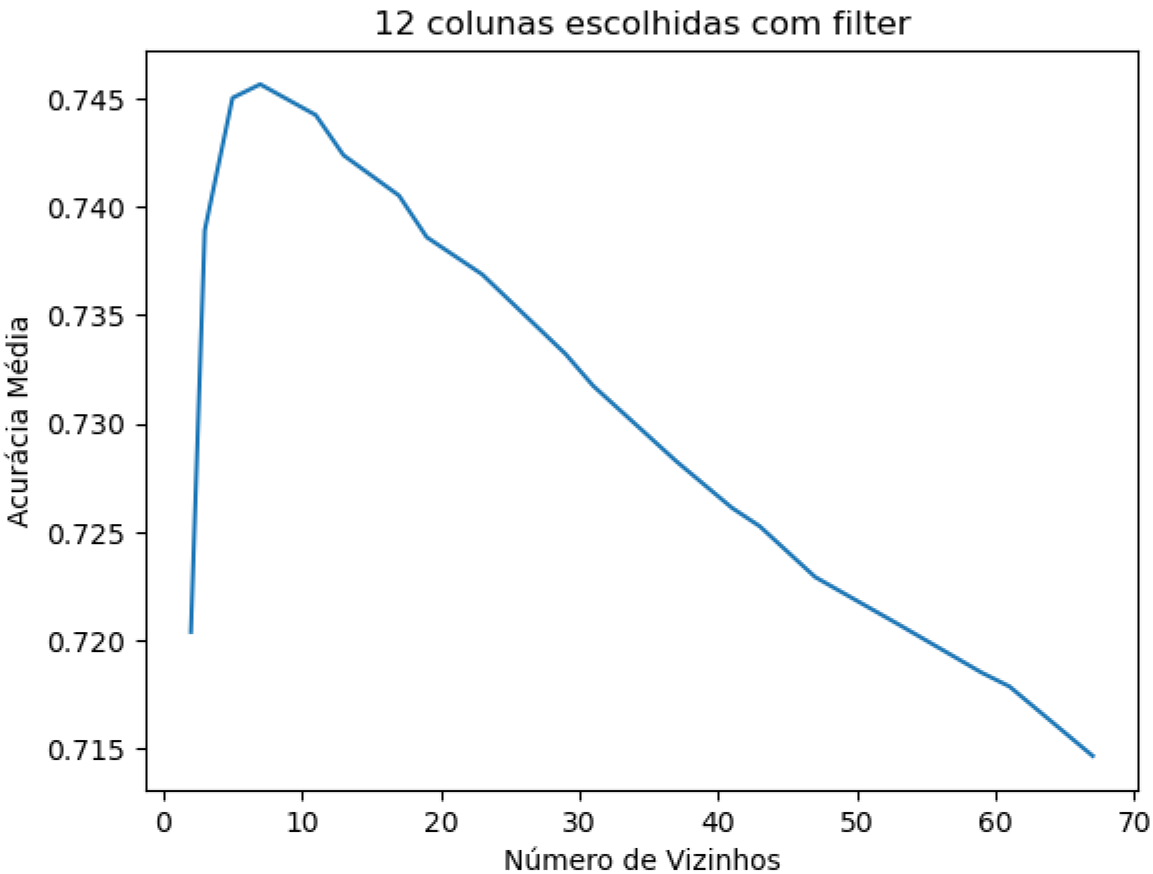

```
Modelo [10]: ['Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inflight wifi servic
e', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'Departure Delay in Min
utes', 'Arrival Delay in Minutes']
2 vizinhos:  0.7079656480981417
3 vizinhos:  0.7302063087662567
5 vizinhos:  0.7353706828337584
7 vizinhos:  0.7370697258012637
11 vizinhos:  0.7360368345880943
13 vizinhos:  0.734279979992404
17 vizinhos:  0.7317509264881116
19 vizinhos:  0.7299071424657871
23 vizinhos:  0.7276772804205472
29 vizinhos:  0.7248103229206617
31 vizinhos:  0.7228700603917806
37 vizinhos:  0.7193081100835228
41 vizinhos:  0.7166535068641932
43 vizinhos:  0.7162287740762979
47 vizinhos:  0.7146553339363939
53 vizinhos:  0.711875233182792
59 vizinhos:  0.7096839746535663
61 vizinhos:  0.708400144279814
67 vizinhos:  0.7071645280007574
```



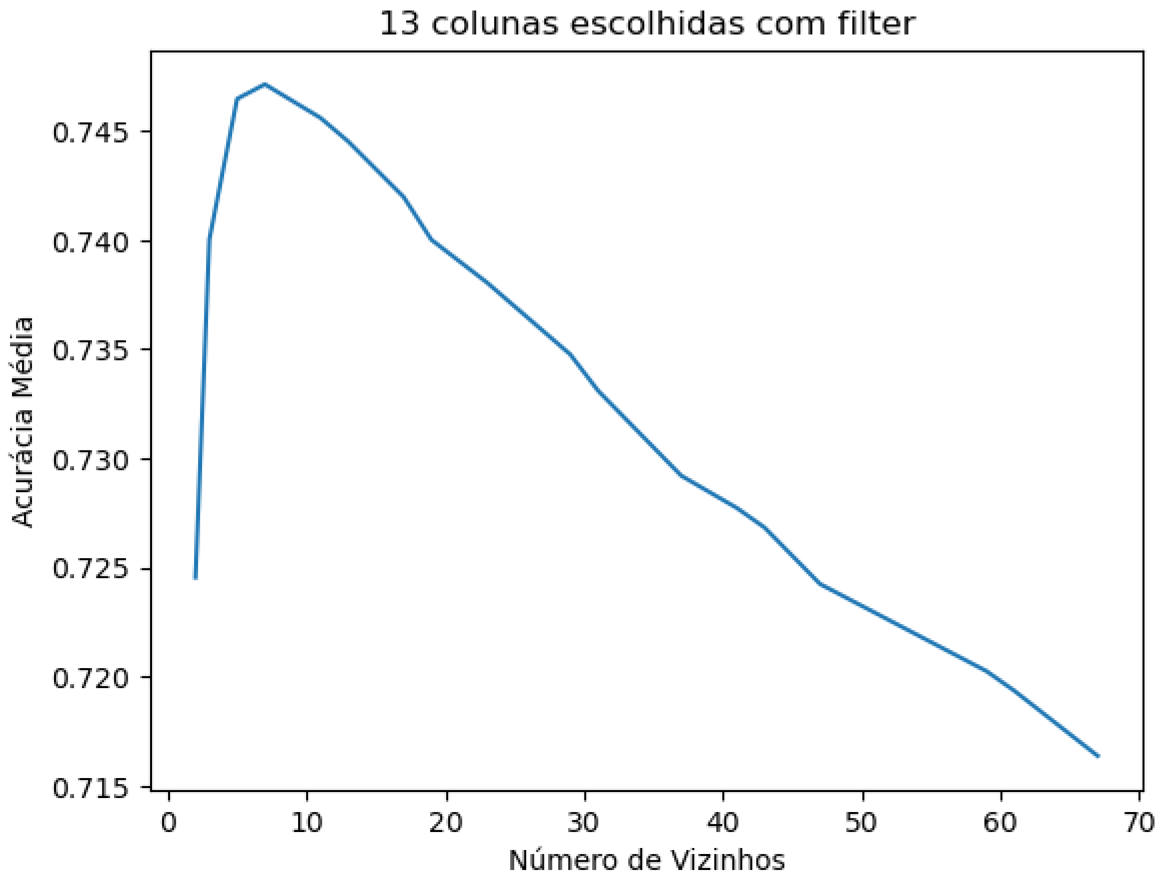

```
Modelo [11]: ['Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inflight wifi servic
e', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-board service', 'De
parture Delay in Minutes', 'Arrival Delay in Minutes']
2 vizinhos:  0.7133231142896651
3 vizinhos:  0.733652438276678
5 vizinhos:  0.7390388843603439
7 vizinhos:  0.7398884132984915
11 vizinhos:  0.7394250704719862
13 vizinhos:  0.7379867887622014
17 vizinhos:  0.7359307017082864
19 vizinhos:  0.7337973423963868
23 vizinhos:  0.7319922364340262
29 vizinhos:  0.7285074912941985
31 vizinhos:  0.7264224467113258
37 vizinhos:  0.7227542861839127
41 vizinhos:  0.7206498788101743
43 vizinhos:  0.7200031308458763
47 vizinhos:  0.7182655738150959
53 vizinhos:  0.715987463198584
59 vizinhos:  0.7131494362054744
61 vizinhos:  0.7121648345627494
67 vizinhos:  0.7101087279410476
```



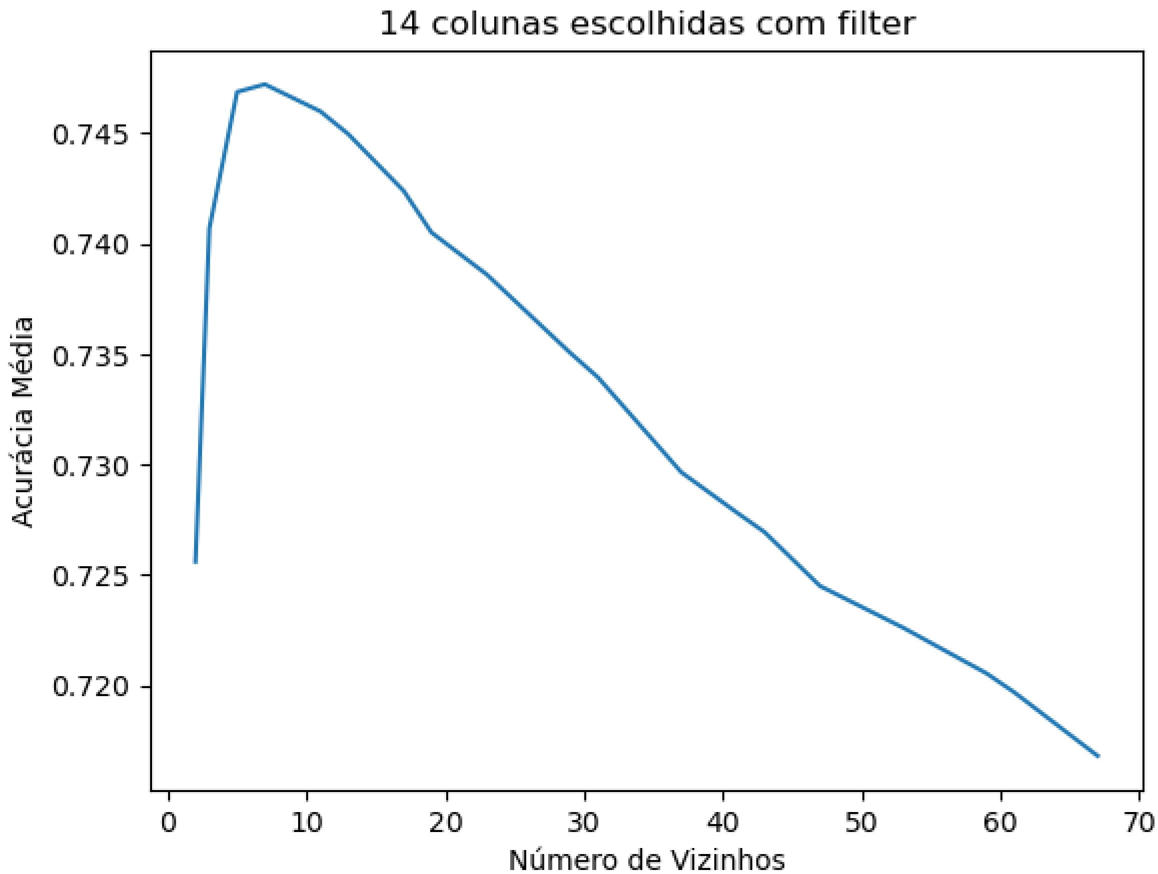

```
Modelo [12]: ['Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inflight wifi servic
e', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-board service', 'Le
g room service', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']
2 vizinhos:  0.720389232163776
3 vizinhos:  0.7389231092206766
5 vizinhos:  0.7450334320295224
7 vizinhos:  0.7456609280870793
11 vizinhos:  0.7442419383514084
13 vizinhos:  0.7423886005901645
17 vizinhos:  0.7405255683882965
19 vizinhos:  0.7386046155377172
23 vizinhos:  0.7368863635262419
29 vizinhos:  0.7332181778402457
31 vizinhos:  0.7317412944780453
37 vizinhos:  0.7282564971574529
41 vizinhos:  0.7261135226078754
43 vizinhos:  0.7252640579638842
47 vizinhos:  0.7229183508940242
53 vizinhos:  0.7207367709648336
59 vizinhos:  0.7185069331463771
61 vizinhos:  0.7178698004197569
67 vizinhos:  0.7146649911050432
```



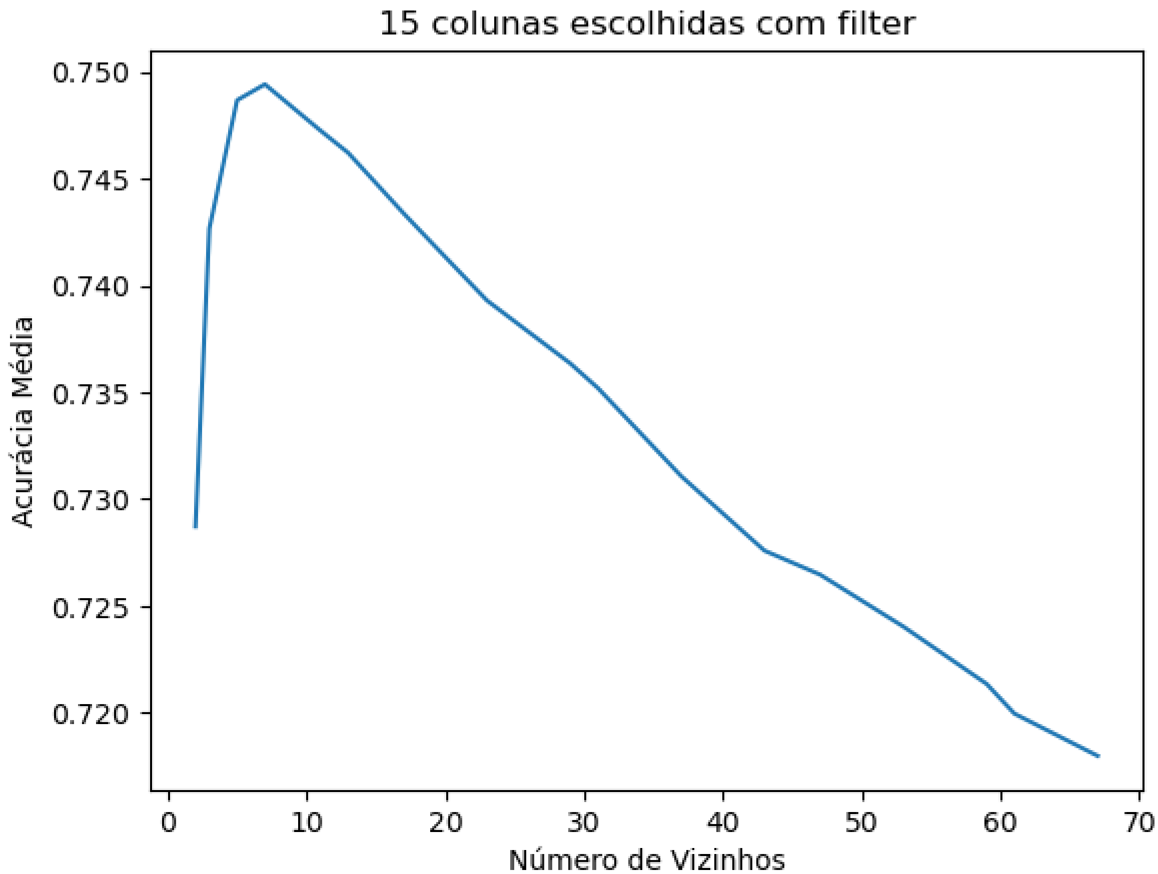

```
Modelo [13]: ['Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inflight wifi servic
e', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-board service', 'Le
g room service', 'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minute
s']
2 vizinhos:  0.7245400833997706
3 vizinhos:  0.740023567069614
5 vizinhos:  0.7464717603292755
7 vizinhos:  0.7471378291534676
11 vizinhos:  0.7456030400513459
13 vizinhos:  0.7445219123802965
17 vizinhos:  0.7419735557202978
19 vizinhos:  0.7400139658089268
23 vizinhos:  0.7380543926699443
29 vizinhos:  0.7347626846779758
31 vizinhos:  0.7331023672922022
37 vizinhos:  0.7292025008749597
41 vizinhos:  0.7277255914223768
43 vizinhos:  0.7268278511849319
47 vizinhos:  0.7242408295101604
53 vizinhos:  0.7222426379463738
59 vizinhos:  0.720244461291377
61 vizinhos:  0.7193466968271486
67 vizinhos:  0.7163639287792198
```



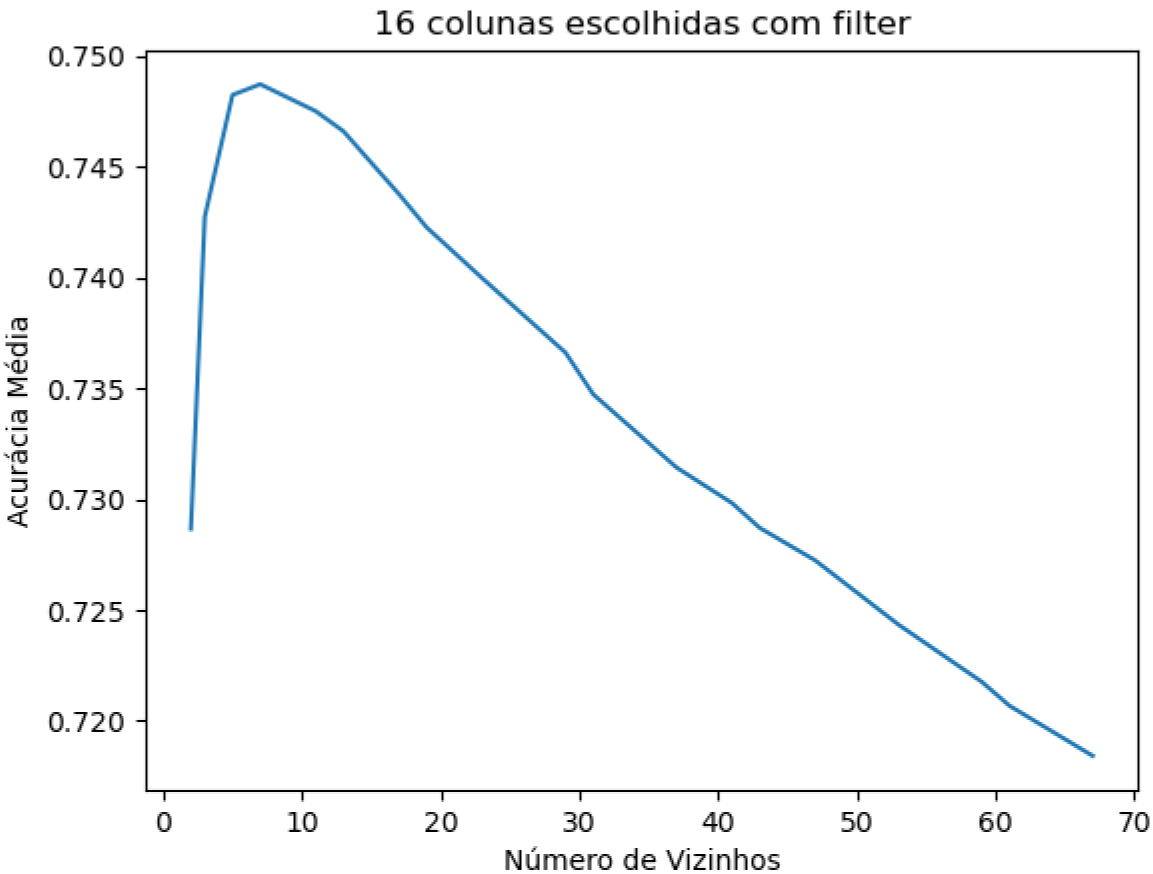

```
Modelo [14]: ['Customer Type', 'Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inf
light wifi service', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-bo
ard service', 'Leg room service', 'Cleanliness', 'Departure Delay in Minutes', 'Arrival
Delay in Minutes']
2 vizinhos:  0.7255922684506524
3 vizinhos:  0.7406896293712106
5 vizinhos:  0.74686754024721
7 vizinhos:  0.7472150510942865
11 vizinhos:  0.7459891665278284
13 vizinhos:  0.744965948323898
17 vizinhos:  0.7423886481119323
19 vizinhos:  0.7405062689597877
23 vizinhos:  0.7385949593008673
29 vizinhos:  0.7350522646265479
31 vizinhos:  0.733942174767544
37 vizinhos:  0.7296465433411568
41 vizinhos:  0.7278221286322937
43 vizinhos:  0.7269340511542944
47 vizinhos:  0.724491800351922
53 vizinhos:  0.7225998022349021
59 vizinhos:  0.7205340384445511
61 vizinhos:  0.7196749082457161
67 vizinhos:  0.7167983159403664
```



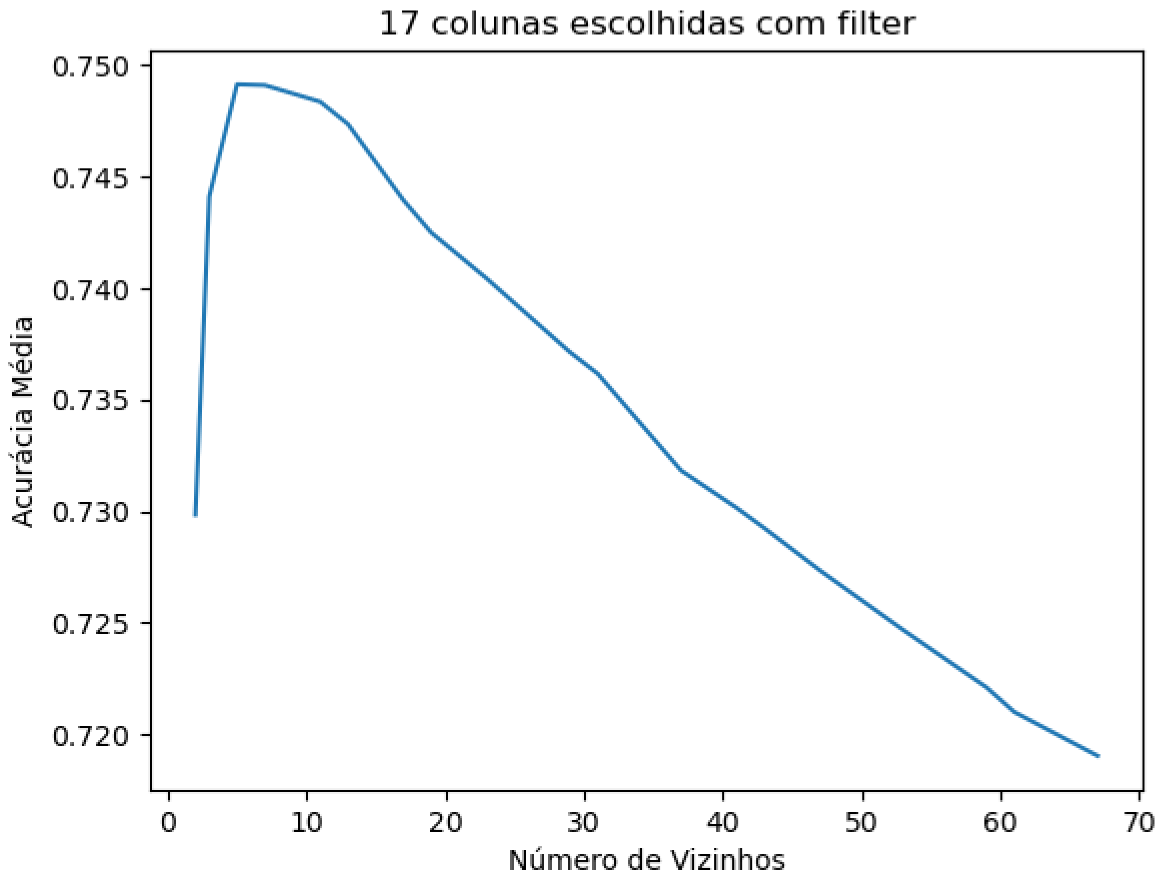

```
Modelo [15]: ['Customer Type', 'Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inf
light wifi service', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-bo
ard service', 'Leg room service', 'Checkin service', 'Cleanliness', 'Departure Delay in
Minutes', 'Arrival Delay in Minutes']
2 vizinhos:  0.7287198250751683
3 vizinhos:  0.7426684273947524
5 vizinhos:  0.7486726443459718
7 vizinhos:  0.7494160003369387
11 vizinhos:  0.7472537291542505
13 vizinhos:  0.7462112254988015
17 vizinhos:  0.7433732441638611
19 vizinhos:  0.74201213314593
23 vizinhos:  0.7392996158004846
29 vizinhos:  0.7363554540639684
31 vizinhos:  0.7351970895433102
37 vizinhos:  0.731075197699872
41 vizinhos:  0.7287488198761005
43 vizinhos:  0.7275904609462385
47 vizinhos:  0.7264706822374069
53 vizinhos:  0.7240381202848623
59 vizinhos:  0.721354579104362
61 vizinhos:  0.7199645282616611
67 vizinhos:  0.7179856501033737
```



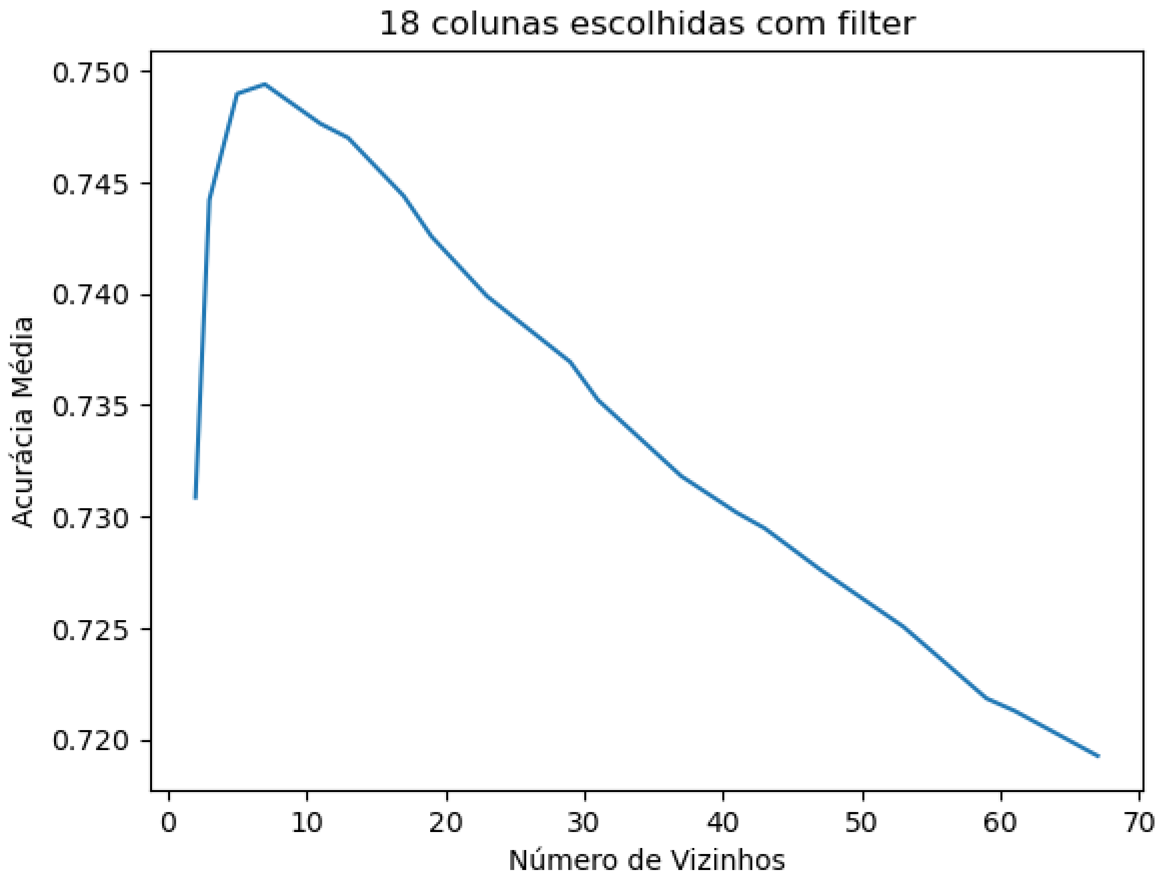

```
Modelo [16]: ['Customer Type', 'Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inf
light wifi service', 'Food and drink', 'Online boarding', 'Seat comfort', 'Inflight ente
rtainment', 'On-board service', 'Leg room service', 'Checkin service', 'Cleanliness', 'D
eparture Delay in Minutes', 'Arrival Delay in Minutes']
2 vizinhos:  0.7286522714845912
3 vizinhos:  0.7427360555292788
5 vizinhos:  0.7482382879342044
7 vizinhos:  0.748720984233582
11 vizinhos:  0.7475047298135917
13 vizinhos:  0.7465973463844879
17 vizinhos:  0.7437593594587513
19 vizinhos:  0.7422534775684212
23 vizinhos:  0.7399657069878616
29 vizinhos:  0.7366064500643128
31 vizinhos:  0.7347240820937606
37 vizinhos:  0.7314227402281268
41 vizinhos:  0.7298202950374975
43 vizinhos:  0.7287005386918505
47 vizinhos:  0.7272332761581242
53 vizinhos:  0.7243373527430869
59 vizinhos:  0.7217599798507705
61 vizinhos:  0.7206788382027305
67 vizinhos:  0.7184200391281189
```



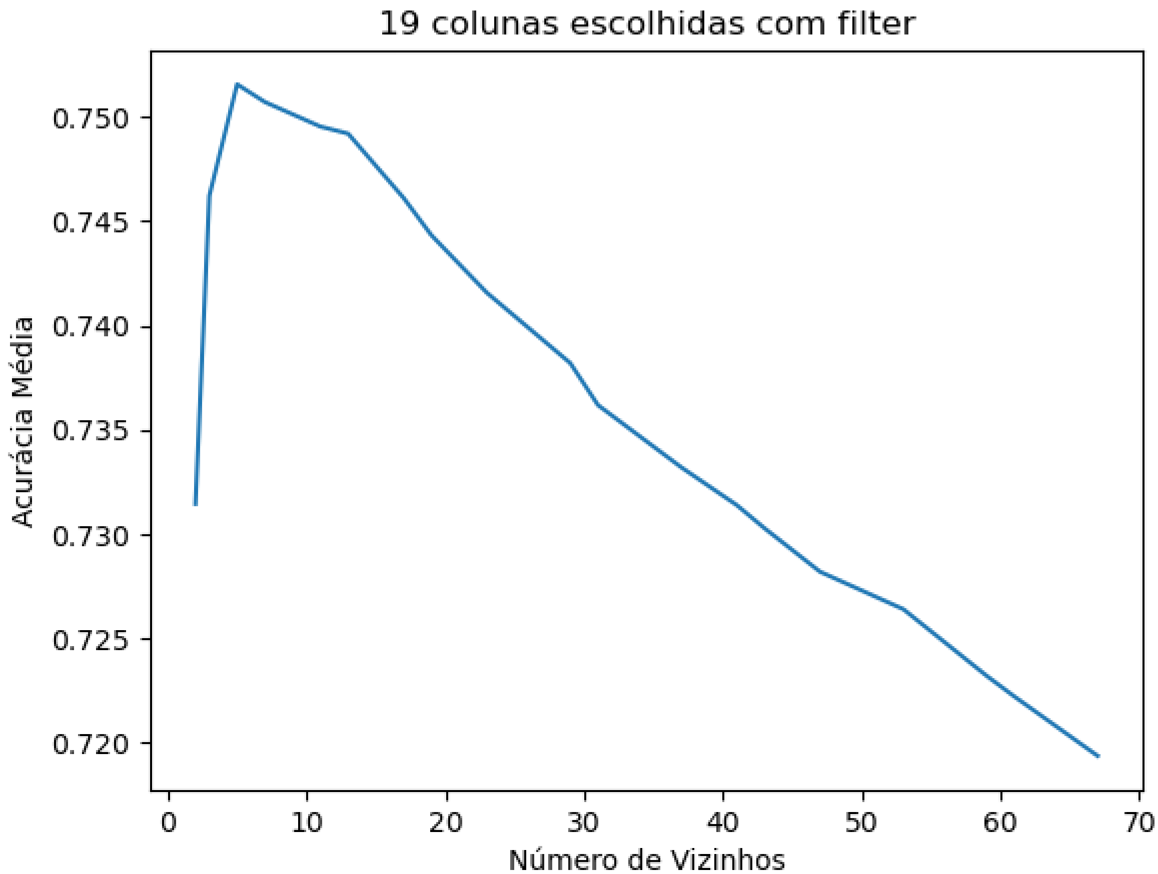

```
Modelo [17]: ['Customer Type', 'Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inf
light wifi service', 'Food and drink', 'Online boarding', 'Seat comfort', 'Inflight ente
rtainment', 'On-board service', 'Leg room service', 'Baggage handling', 'Checkin servic
e', 'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']
2 vizinhos:  0.7298395804890158
3 vizinhos:  0.744087472106586
5 vizinhos:  0.7491167529699241
7 vizinhos:  0.7490781187045304
11 vizinhos:  0.7483348633478956
13 vizinhos:  0.7473309753218528
17 vizinhos:  0.7439234735542294
19 vizinhos:  0.7424754815632313
23 vizinhos:  0.740419375873329
29 vizinhos:  0.7371180573026793
31 vizinhos:  0.7361527541566637
37 vizinhos:  0.7318184754196917
41 vizinhos:  0.7301485073878645
43 vizinhos:  0.7292411202315633
47 vizinhos:  0.7273491314325372
53 vizinhos:  0.7246848859533481
59 vizinhos:  0.7221171329577064
61 vizinhos:  0.7210166862903613
67 vizinhos:  0.7190571066287834
```



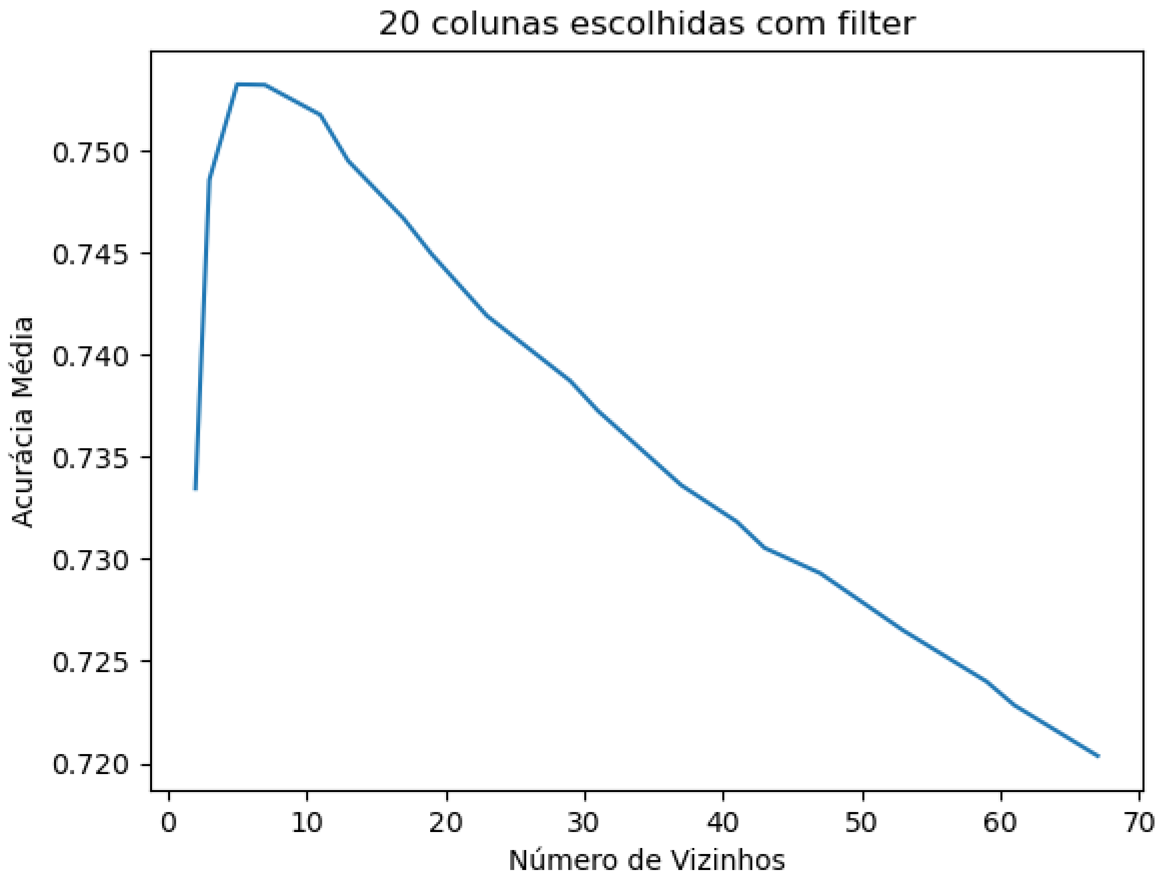

```
Modelo [18]: ['Customer Type', 'Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inf
light wifi service', 'Food and drink', 'Online boarding', 'Seat comfort', 'Inflight ente
rtainment', 'On-board service', 'Leg room service', 'Baggage handling', 'Checkin servic
e', 'Inflight service', 'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in M
inutes']
2 vizinhos:  0.7308531256837079
3 vizinhos:  0.7442129780270528
5 vizinhos:  0.7489719625297384
7 vizinhos:  0.7493966869314395
11 vizinhos:  0.7476301844850932
13 vizinhos:  0.7469931263024225
17 vizinhos:  0.7443868256987284
19 vizinhos:  0.7425720159777501
23 vizinhos:  0.7398884412524727
29 vizinhos:  0.7369346344606986
31 vizinhos:  0.7352260312316784
37 vizinhos:  0.7318088480686222
41 vizinhos:  0.7301678217251631
43 vizinhos:  0.7294728046900072
47 vizinhos:  0.7276194100890017
53 vizinhos:  0.725051702751529
59 vizinhos:  0.7218275641907267
61 vizinhos:  0.7212773413229538
67 vizinhos:  0.7192501614808305
```



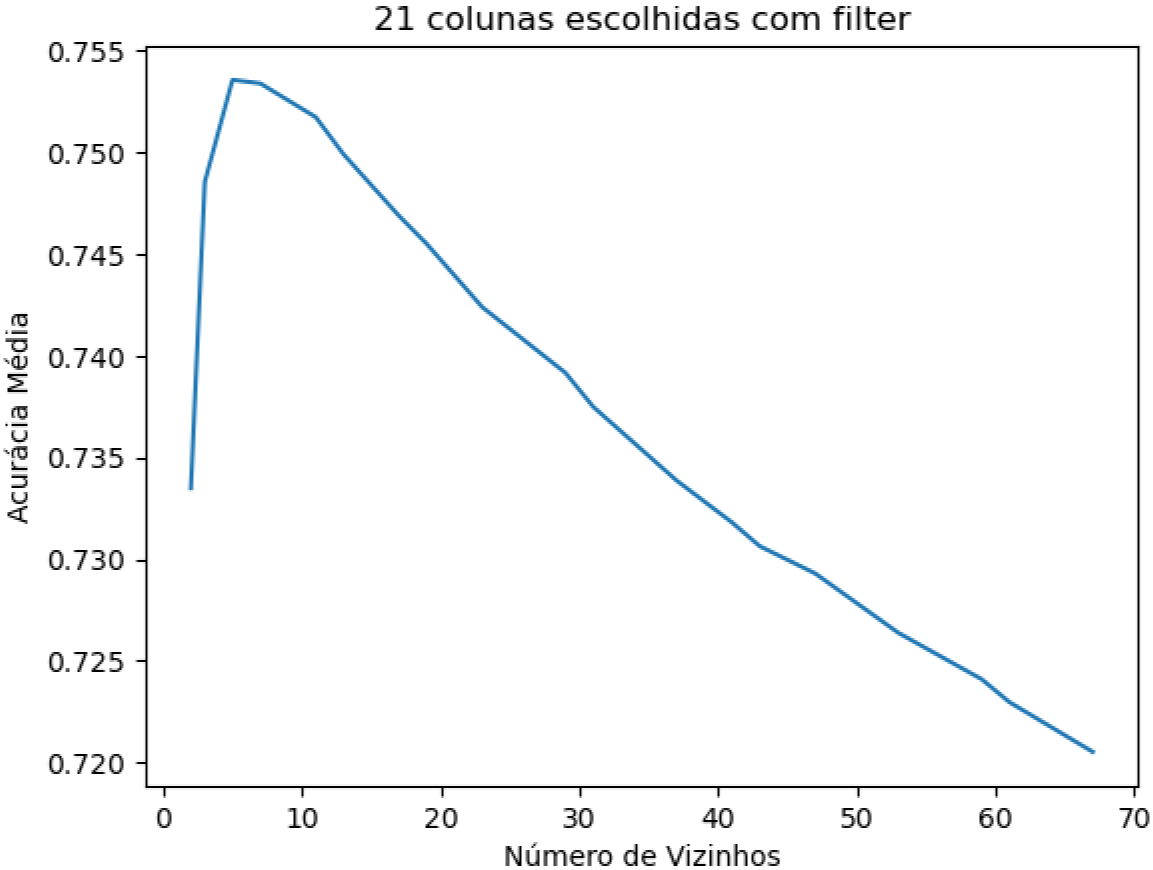

```
Modelo [19]: ['Customer Type', 'Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inf
light wifi service', 'Ease of Online booking', 'Food and drink', 'Online boarding', 'Sea
t comfort', 'Inflight entertainment', 'On-board service', 'Leg room service', 'Baggage h
andling', 'Checkin service', 'Inflight service', 'Cleanliness', 'Departure Delay in Minu
tes', 'Arrival Delay in Minutes']
2 vizinhos:  0.7314419679080844
3 vizinhos:  0.7462207755105236
5 vizinhos:  0.751568618078175
7 vizinhos:  0.7507191459797888
11 vizinhos:  0.749541495075813
13 vizinhos:  0.7492036618969721
17 vizinhos:  0.7461147013340758
19 vizinhos:  0.744328828642469
23 vizinhos:  0.7415681055885226
29 vizinhos:  0.7381992194503055
31 vizinhos:  0.7361816883906371
37 vizinhos:  0.7331988849343323
41 vizinhos:  0.7313841022355357
43 vizinhos:  0.7302739983995414
47 vizinhos:  0.7281889351806814
53 vizinhos:  0.7263934593647886
59 vizinhos:  0.7231983006961287
61 vizinhos:  0.722204036293958
67 vizinhos:  0.7193563595865942
```



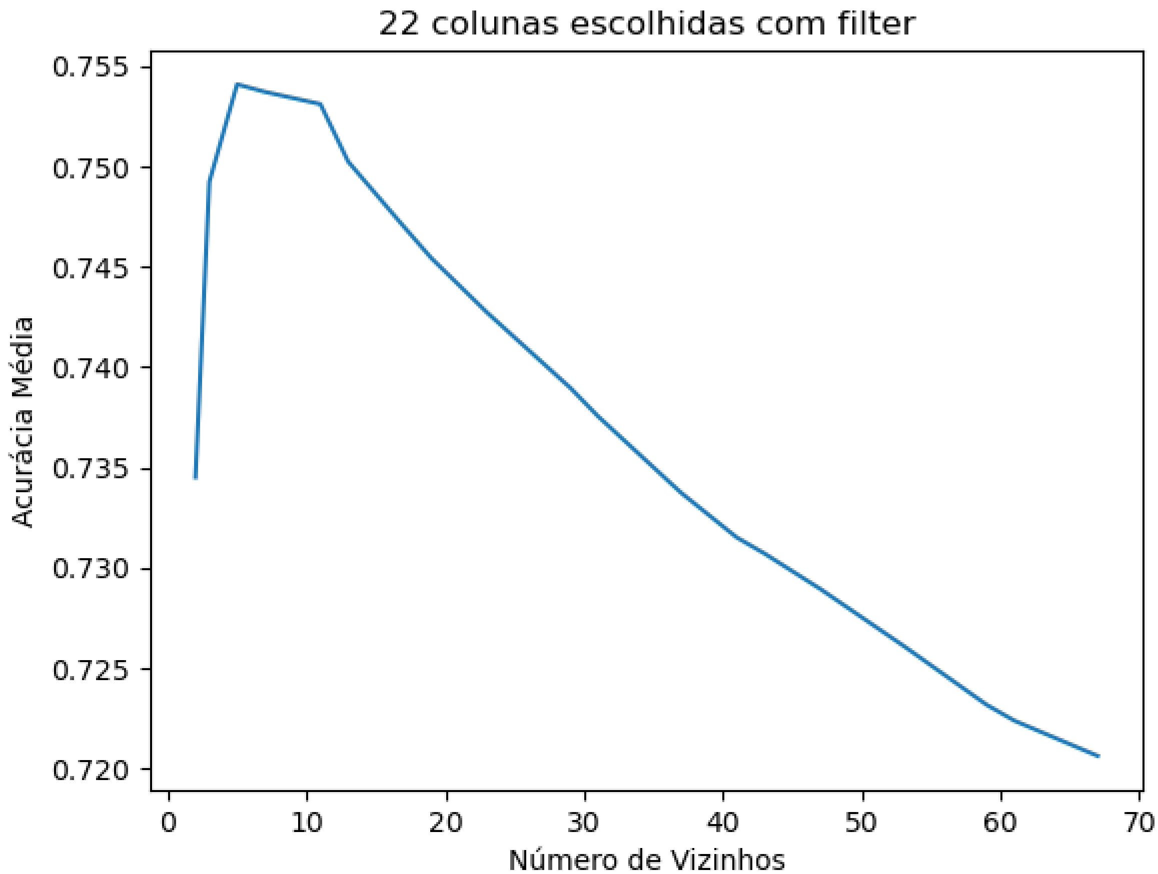

```
Modelo [20]: ['Customer Type', 'Age', 'Type of Travel', 'Class', 'Flight Distance', 'Inf
light wifi service', 'Departure/Arrival time convenient', 'Ease of Online booking', 'Foo
d and drink', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-board ser
vice', 'Leg room service', 'Baggage handling', 'Checkin service', 'Inflight service', 'C
leanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']
2 vizinhos:  0.7334401576082723
3 vizinhos:  0.7485568291389315
5 vizinhos:  0.7532289466455409
7 vizinhos:  0.7531999919119814
11 vizinhos:  0.7517327741046247
13 vizinhos:  0.7494836107672771
17 vizinhos:  0.7466553098959701
19 vizinhos:  0.7449273308308929
23 vizinhos:  0.7418963225978864
29 vizinhos:  0.7387108388020638
31 vizinhos:  0.737243567882143
37 vizinhos:  0.7336043173619193
41 vizinhos:  0.731818488464883
43 vizinhos:  0.7305346161601591
47 vizinhos:  0.7292990334258797
53 vizinhos:  0.7264900068244985
59 vizinhos:  0.723989858668399
61 vizinhos:  0.7228314727163555
67 vizinhos:  0.720350629579561
```




```
Modelo [21]: ['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class', 'Flight Dista
nce', 'Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online book
ing', 'Food and drink', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On
-board service', 'Leg room service', 'Baggage handling', 'Checkin service', 'Inflight se
rvice', 'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']
2 vizinhos:  0.7334884192247355
3 vizinhos:  0.7485471794246773
5 vizinhos:  0.7535764593562161
7 vizinhos:  0.7534027253640633
11 vizinhos:  0.7517424387276689
13 vizinhos:  0.7499179932694268
17 vizinhos:  0.7468869682640317
19 vizinhos:  0.7455065177502188
23 vizinhos:  0.7423982829173967
29 vizinhos:  0.7391645225963209
31 vizinhos:  0.7374945461782995
37 vizinhos:  0.7338746118589732
41 vizinhos:  0.731808824773638
43 vizinhos:  0.73064078910734
47 vizinhos:  0.7292990278350834
53 vizinhos:  0.726374162731678
59 vizinhos:  0.7240960502515671
61 vizinhos:  0.7229666181012837
67 vizinhos:  0.7205243877984973
```




```
Modelo [22]: ['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class', 'Flight Dista
nce', 'Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online book
ing', 'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort', 'Inflight en
tertainment', 'On-board service', 'Leg room service', 'Baggage handling', 'Checkin servi
ce', 'Inflight service', 'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in
Minutes']
2 vizinhos:  0.7345020156683927
3 vizinhos:  0.7492229091447162
5 vizinhos:  0.7540880684581814
7 vizinhos:  0.7537116261725297
11 vizinhos:  0.7531034947694374
13 vizinhos:  0.7502365372695521
17 vizinhos:  0.7470028067660561
19 vizinhos:  0.74541965448134
23 vizinhos:  0.7427168315765188
29 vizinhos:  0.7389618226890164
31 vizinhos:  0.7375428171127563
37 vizinhos:  0.7337201512049469
41 vizinhos:  0.7314999537827513
43 vizinhos:  0.7306987516870228
47 vizinhos:  0.7289418654101538
53 vizinhos:  0.726103896188605
59 vizinhos:  0.7231597055663086
61 vizinhos:  0.7223681410714425
67 vizinhos:  0.7206112706351628
```



Wrapper

In [81]:
```
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.model_selection import KFold, cross_val_score, StratifiedKFold
```

In [82]:
```
# Fazer a seleção de características com train_split_test e o kfold depois

X_train, X_test, y_train, y_test = train_test_split(df_encod.iloc[:, :-1], df_encod.iloc
                                                    test_size=0.2, random_state=42)
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

for quant_var in range(18, 22):
    knn_classifier = KNeighborsClassifier(n_neighbors = 5)

    # Cria um objeto Sequential Feature Selector para selecionar as duas melhores caract
    sfs = SequentialFeatureSelector(knn_classifier, n_features_to_select=quant_var)

    # Aplica o Sequential Feature Selector ao conjunto de dados de treino
    X_train_sfs = sfs.fit_transform(X_train, y_train)

    # Obtém o índice das características selecionadas após o último ajuste
    selected_features_idx = sfs.get_support(indices=True)

    # Obtém os nomes das características selecionadas
    selected_features_names = df_encod.columns[selected_features_idx]

    # Imprime as características selecionadas
    print("Características selecionadas: ", selected_features_names)

    values = cross_val_score(knn_classifier,
                             df_encod[selected_features_names],
                             df_encod['satisfaction'],
```

```
                        cv=cv, scoring='accuracy')
    print("Acurácia média: ", values.mean())
```

```
Características selecionadas:  Index(['Gender', 'Customer Type', 'Type of Travel', 'Clas
s',
       'Inflight wifi service', 'Departure/Arrival time convenient',
       'Ease of Online booking', 'Gate location', 'Food and drink',
       'Online boarding', 'Seat comfort', 'Inflight entertainment',
       'On-board service', 'Leg room service', 'Baggage handling',
       'Checkin service', 'Inflight service', 'Cleanliness'],
      dtype='object')
Acurácia média:  0.9315984188855605
Características selecionadas:  Index(['Gender', 'Customer Type', 'Age', 'Type of Trave
l', 'Class',
       'Inflight wifi service', 'Departure/Arrival time convenient',
       'Ease of Online booking', 'Gate location', 'Food and drink',
       'Online boarding', 'Seat comfort', 'Inflight entertainment',
       'On-board service', 'Leg room service', 'Baggage handling',
       'Checkin service', 'Inflight service', 'Cleanliness'],
      dtype='object')
Acurácia média:  0.9161342038948469
Características selecionadas:  Index(['Gender', 'Customer Type', 'Age', 'Type of Trave
l', 'Class',
       'Inflight wifi service', 'Departure/Arrival time convenient',
       'Ease of Online booking', 'Gate location', 'Food and drink',
       'Online boarding', 'Seat comfort', 'Inflight entertainment',
       'On-board service', 'Leg room service', 'Baggage handling',
       'Checkin service', 'Inflight service', 'Cleanliness',
       'Arrival Delay in Minutes'],
      dtype='object')
Acurácia média:  0.895766219552058
Características selecionadas:  Index(['Gender', 'Customer Type', 'Age', 'Type of Trave
l', 'Class',
       'Inflight wifi service', 'Departure/Arrival time convenient',
       'Ease of Online booking', 'Gate location', 'Food and drink',
       'Online boarding', 'Seat comfort', 'Inflight entertainment',
       'On-board service', 'Leg room service', 'Baggage handling',
       'Checkin service', 'Inflight service', 'Cleanliness',
       'Departure Delay in Minutes', 'Arrival Delay in Minutes'],
      dtype='object')
Acurácia média:  0.8763345836217253
```

1 Característica selecionada: ['Online boarding']

Acurácia média: 0.7752471085333814

2 Características selecionadas: ['Type of Travel', 'Online boarding']

Acurácia média: 0.8487943466614188

3 Características selecionadas: ['Type of Travel', 'Inflight wifi service', 'Online boarding']

Acurácia média: 0.8818370238179101

4 Características selecionadas: ['Type of Travel', 'Inflight wifi service', 'Gate location', 'Online boarding']

Acurácia média: 0.915921765752348

5 Características selecionadas: ['Type of Travel', 'Inflight wifi service', 'Gate location', 'Online boarding', 'Baggage handling']

Acurácia média: 0.9209221543126841

6 Características selecionadas: ['Customer Type', 'Type of Travel', 'Inflight wifi service', 'Gate location', 'Online boarding', 'Baggage handling']

Acurácia média: 0.931994188553702

7 Características selecionadas: ['Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Gate location', 'Online boarding', 'Baggage handling']

Acurácia média: 0.9394076924137739

8 Características selecionadas: ['Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Gate location', 'Online boarding', 'Baggage handling', 'Inflight service']

Acurácia média: 0.9402668692025772

9 Características selecionadas: ['Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Gate location', 'Online boarding', 'Inflight entertainment', 'Baggage handling', 'Inflight service']

Acurácia média: 0.940508171694097

10 Características selecionadas: ['Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Gate location', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'Baggage handling', 'Inflight service']

Acurácia média: 0.9414734319773416

11 Características selecionadas: ['Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Ease of Online booking', 'Gate location', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'Baggage handling', 'Inflight service']

Acurácia média: 0.9421974354272356

12 Características selecionadas: ['Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Ease of Online booking', 'Gate location', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'Leg room service', 'Baggage handling', 'Inflight service']

Acurácia média: 0.9401799528211345

13 Características selecionadas: ['Gender', 'Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Ease of Online booking', 'Gate location', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'Leg room service', 'Baggage handling', 'Inflight service']

Acurácia média: 0.9397745092119549

14 Características selecionadas: ['Gender', 'Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Ease of Online booking', 'Gate location', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'Leg room service', 'Baggage handling', 'Inflight service', 'Cleanliness']

Acurácia média: 0.938162485123823

15 Características selecionadas: ['Gender', 'Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online booking', 'Gate location', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'Leg room service', 'Baggage handling', 'Inflight service', 'Cleanliness']

Acurácia média: 0.9374288804132418

16 Características selecionadas: ['Gender', 'Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online booking', 'Gate location', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'Leg room service', 'Baggage handling', 'Checkin service', 'Inflight service', 'Cleanliness']

Acurácia média: 0.935382396483613

17 Características selecionadas: ['Gender', 'Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online booking', 'Gate location', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-board service', 'Leg room service', 'Baggage handling', 'Checkin service', 'Inflight service', 'Cleanliness']

Acurácia média: 0.932930531375362

18 Características selecionadas: ['Gender', 'Customer Type', 'Type of Travel', 'Class', 'Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online booking', 'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-board service', 'Leg room service', 'Baggage handling', 'Checkin service', 'Inflight service', 'Cleanliness']

Acurácia média: 0.9315984188855605

19 Características selecionadas: ['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class', 'Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online booking', 'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-board service', 'Leg room service', 'Baggage handling', 'Checkin service', 'Inflight service', 'Cleanliness']

Acurácia média: 0.9161342038948469

20 Características selecionadas: ['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class', 'Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online booking', 'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-board service', 'Leg room service', 'Baggage handling', 'Checkin service', 'Inflight service', 'Cleanliness', 'Arrival Delay in Minutes']

Acurácia média: 0.895766219552058

21Características selecionadas: Index(['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class', 'Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online booking', 'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort', 'Inflight entertainment', 'On-board service', 'Leg room service', 'Baggage handling', 'Checkin service', 'Inflight service', 'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes'], dtype='object')

Acurácia média: 0.8763345836217253

Etapa 4

In [ ]:

```
# from sklearn.linear_model import LogisticRegression
# from sklearn.ensemble import RandomForestClassifier
# from sklearn.tree import DecisionTreeClassifier
# from sklearn.metrics import accuracy_score

# SEED = 42
# np.random.seed(SEED)
```

```
# cv = StratifiedKFold(n_splits = 10, shuffle = True)
# knn = KNeighborsClassifier(n_neighbors=3)
# model = DecisionTreeClassifier(max_depth=3)
# model_log = LogisticRegression(solver='liblinear')
# model_rand = RandomForestClassifier(n_estimators=100)

# models = [knn, model, model_log, model_rand]
# name = ['KNN','Árvore de Decisão',
#         'Regressão Logística', 'Random Forest']
# count = 0
# for item in models:
#     np.random.seed(SEED)
#     results = cross_val_score(item, X, y, cv = cv, scoring = 'accuracy')
#     mean = results.mean()
#     dv = results.std()

#     print('Acurácia média - Modelo {}: {:.2f}%'.format(name[count], mean*100))
#     print('Intervalo de acurácia - Modelo {}: [{:.2f}% ~ {:.2f}%]\n'.format(name[count
#     count += 1
```