Projeto_04_v1

September 12, 2019

1 Formação Cientista de Dados - DSA

1.0.1 Big Data Real-Time Analytics com Python e Spark

1.1 Projeto com Feedback 4 - Prevendo Customer Churn em Operadoras de Telecom

1.1.1 Leonardo Molero

2 Análise Exploratória

```
In [1]: # Importação pacotes iniciais
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
```

```
In [2]: # Faz ajustes para não exibir warnings
        warnings.filterwarnings("ignore")

        # Parametriza impressão dos gráficos dentro do notebook
        %matplotlib inline
```

```
In [3]: # Carrega o dados de treino colocando a primeira coluna como index
        df = pd.read_csv('dados/projeto4_telecom_treino.csv',index_col=[0])
```

```
In [4]: # Checa o tamanho do dataframe
        print(df.shape)
```

```
(3333, 20)
```

```
In [5]: # Visualiza os dados treino
        df.head(10)
```

```
Out[5]:    state  account_length      area_code international_plan voice_mail_plan  \
        1     KS             128  area_code_415                 no             yes
        2     OH             107  area_code_415                 no             yes
        3     NJ             137  area_code_415                 no              no
```

```
4     OH            84  area_code_408                yes             no
5     OK            75  area_code_415                yes             no
6     AL           118  area_code_510                yes             no
7     MA           121  area_code_510                 no            yes
8     MO           147  area_code_415                yes             no
9     LA           117  area_code_408                 no             no
10    WV           141  area_code_415                yes            yes

    number_vmail_messages  total_day_minutes  total_day_calls  \
1                      25              265.1              110
2                      26              161.6              123
3                       0              243.4              114
4                       0              299.4               71
5                       0              166.7              113
6                       0              223.4               98
7                      24              218.2               88
8                       0              157.0               79
9                       0              184.5               97
10                     37              258.6               84

    total_day_charge  total_eve_minutes  total_eve_calls  total_eve_charge  \
1              45.07              197.4               99             16.78
2              27.47              195.5              103             16.62
3              41.38              121.2              110             10.30
4              50.90               61.9               88              5.26
5              28.34              148.3              122             12.61
6              37.98              220.6              101             18.75
7              37.09              348.5              108             29.62
8              26.69              103.1               94              8.76
9              31.37              351.6               80             29.89
10             43.96              222.0              111             18.87

    total_night_minutes  total_night_calls  total_night_charge  \
1                 244.7                 91               11.01
2                 254.4                103               11.45
3                 162.6                104                7.32
4                 196.9                 89                8.86
5                 186.9                121                8.41
6                 203.9                118                9.18
7                 212.6                118                9.57
8                 211.8                 96                9.53
9                 215.8                 90                9.71
10                326.4                 97               14.69

    total_intl_minutes  total_intl_calls  total_intl_charge  \
1                 10.0                 3               2.70
2                 13.7                 3               3.70
3                 12.2                 5               3.29
```

```
4                  6.6                 7              1.78
5                 10.1                 3              2.73
6                  6.3                 6              1.70
7                  7.5                 7              2.03
8                  7.1                 6              1.92
9                  8.7                 4              2.35
10                11.2                 5              3.02

    number_customer_service_calls churn
1                               1    no
2                               1    no
3                               0    no
4                               2    no
5                               3    no
6                               0    no
7                               3    no
8                               0    no
9                               1    no
10                              0    no
```

In [6]:
```
# Verifica os tipos das colunas
df.dtypes
```

Out[6]:
```
state                             object
account_length                     int64
area_code                         object
international_plan                object
voice_mail_plan                   object
number_vmail_messages              int64
total_day_minutes                float64
total_day_calls                    int64
total_day_charge                 float64
total_eve_minutes                float64
total_eve_calls                    int64
total_eve_charge                 float64
total_night_minutes              float64
total_night_calls                  int64
total_night_charge               float64
total_intl_minutes               float64
total_intl_calls                   int64
total_intl_charge                float64
number_customer_service_calls      int64
churn                             object
dtype: object
```

In [7]:
```
# Verifica estatísticas das colunas numéricas
df.describe()
```

Out[7]:
```
       account_length  number_vmail_messages  total_day_minutes  \
count     3333.000000            3333.000000        3333.000000
```

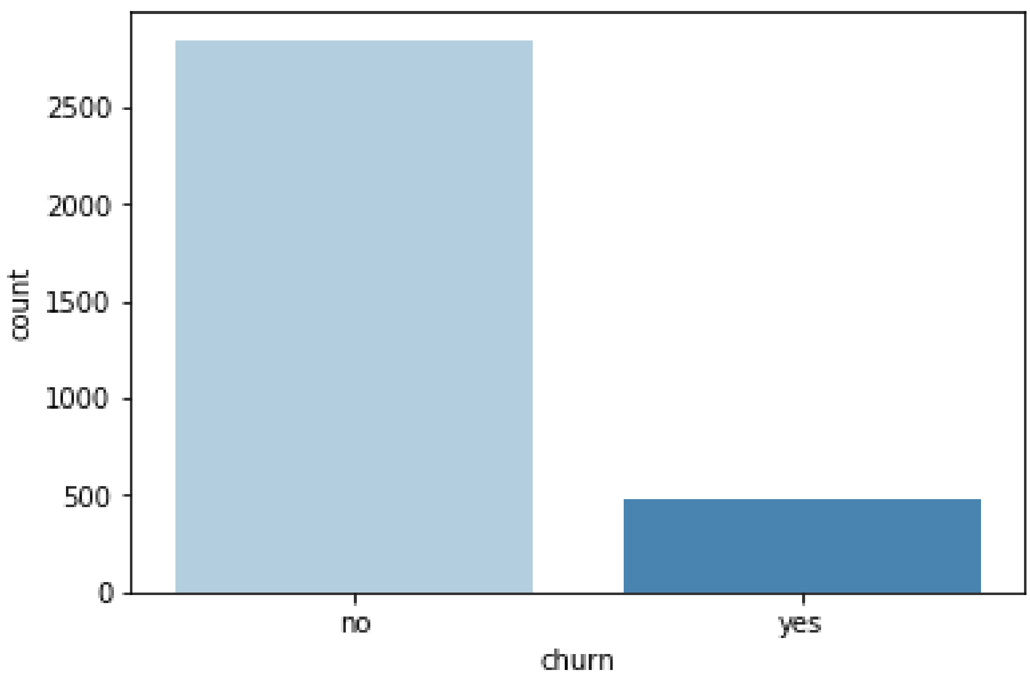
```
mean      101.064806              8.099010          179.775098
std        39.822106             13.688365           54.467389
min         1.000000              0.000000            0.000000
25%        74.000000              0.000000          143.700000
50%       101.000000              0.000000          179.400000
75%       127.000000             20.000000          216.400000
max       243.000000             51.000000          350.800000

       total_day_calls  total_day_charge  total_eve_minutes  total_eve_calls  \
count      3333.000000       3333.000000        3333.000000      3333.000000
mean        100.435644         30.562307         200.980348       100.114311
std          20.069084          9.259435          50.713844        19.922625
min           0.000000          0.000000           0.000000         0.000000
25%          87.000000         24.430000         166.600000        87.000000
50%         101.000000         30.500000         201.400000       100.000000
75%         114.000000         36.790000         235.300000       114.000000
max         165.000000         59.640000         363.700000       170.000000

       total_eve_charge  total_night_minutes  total_night_calls  \
count       3333.000000          3333.000000        3333.000000
mean          17.083540           200.872037         100.107711
std            4.310668            50.573847          19.568609
min            0.000000            23.200000          33.000000
25%           14.160000           167.000000          87.000000
50%           17.120000           201.200000         100.000000
75%           20.000000           235.300000         113.000000
max           30.910000           395.000000         175.000000

       total_night_charge  total_intl_minutes  total_intl_calls  \
count         3333.000000         3333.000000       3333.000000
mean             9.039325           10.237294          4.479448
std              2.275873            2.791840          2.461214
min              1.040000            0.000000          0.000000
25%              7.520000            8.500000          3.000000
50%              9.050000           10.300000          4.000000
75%             10.590000           12.100000          6.000000
max             17.770000           20.000000         20.000000

       total_intl_charge  number_customer_service_calls
count        3333.000000                    3333.000000
mean            2.764581                       1.562856
std             0.753773                       1.315491
min             0.000000                       0.000000
25%             2.300000                       1.000000
50%             2.780000                       1.000000
75%             3.270000                       2.000000
max             5.400000                       9.000000
```

In [8]:
```
# Distribuição da variável alvo
df.groupby('churn').size()
```

Out[8]:
```
churn
no     2850
yes     483
dtype: int64
```

In [9]:
```
# Plota a distribuição da variável alvo
sns.countplot(x='churn',data=df,palette="Blues")
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x14a34749278>
```



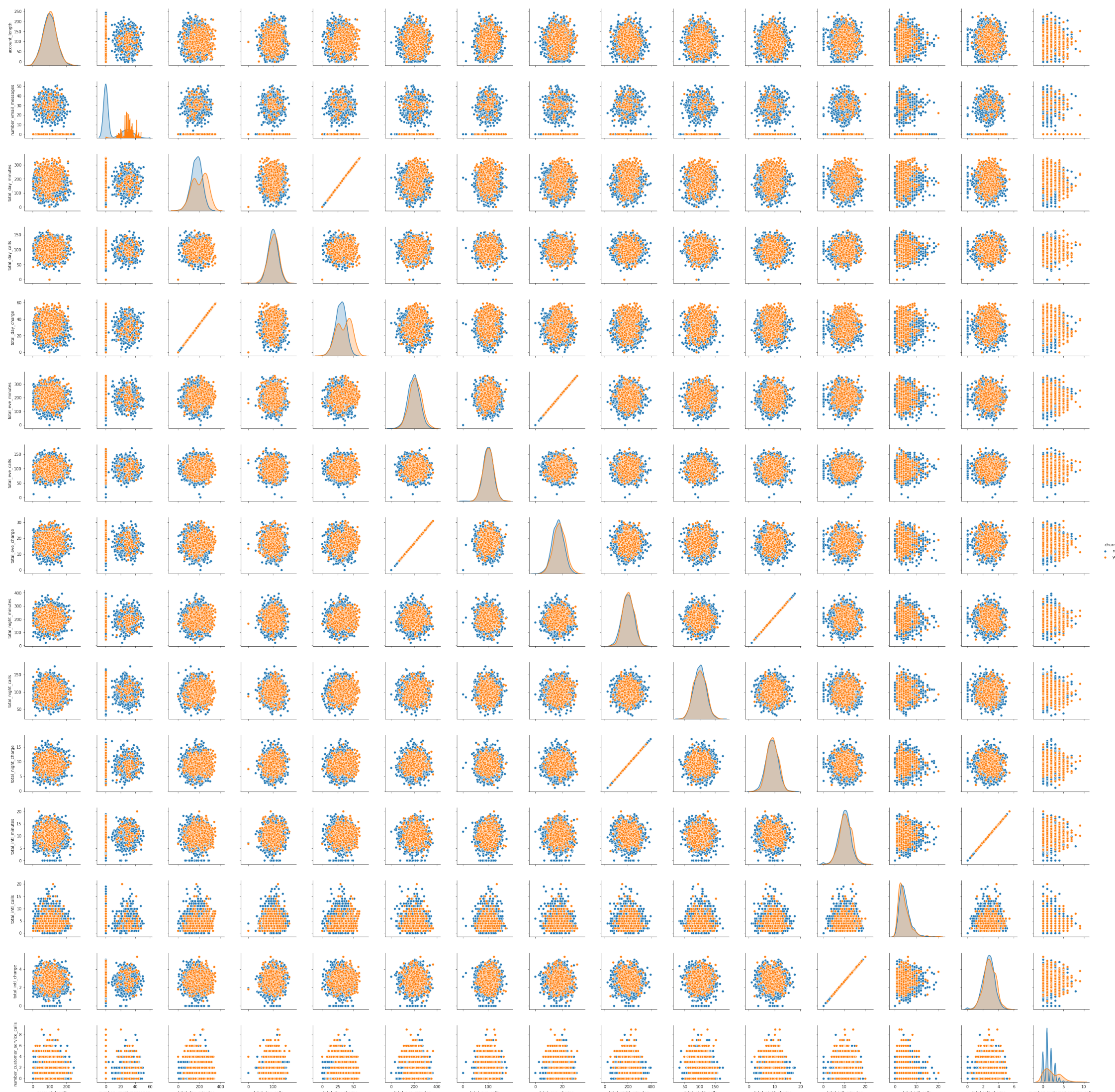
In [10]:
```
# Verifica variáveis tipo texto
print(df.groupby('area_code').size())
print(df.groupby('international_plan').size())
print(df.groupby('voice_mail_plan').size())
```

```
area_code
area_code_408     838
area_code_415    1655
area_code_510     840
dtype: int64
international_plan
no     3010
```

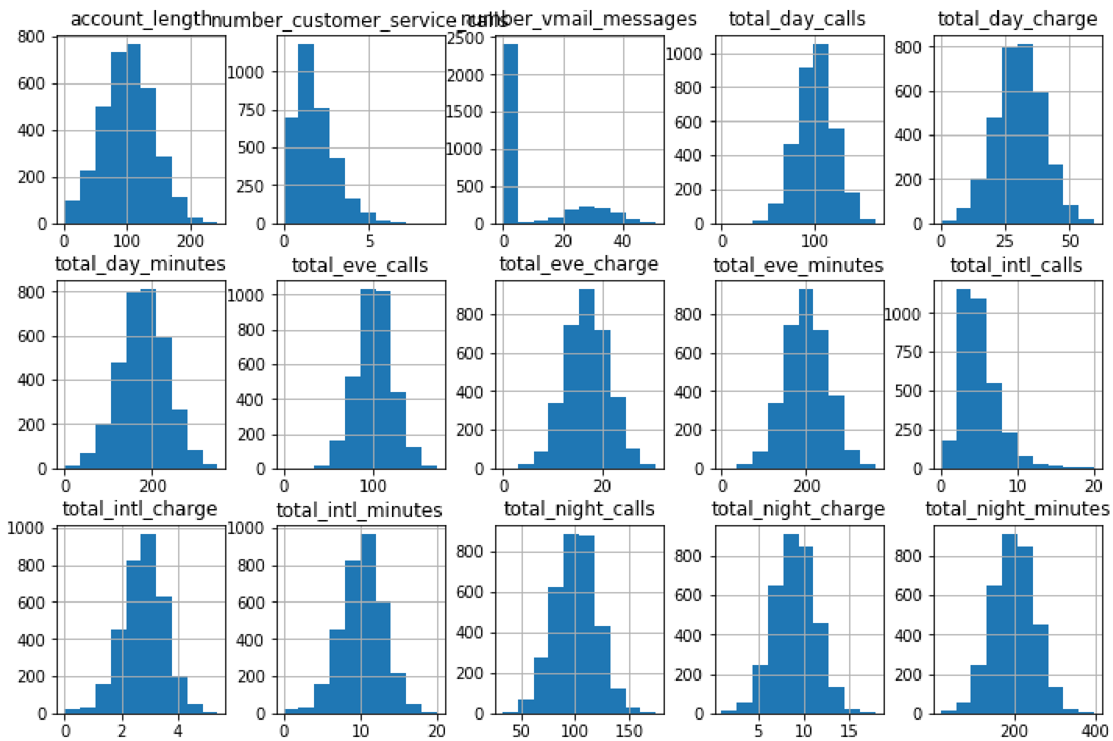
```
yes     323
dtype: int64
voice_mail_plan
no     2411
yes     922
dtype: int64
```

In [11]:
```python
# Plota um gráfico de relação de todas as variáveis do dataset
sns.pairplot(df, hue='churn')
```

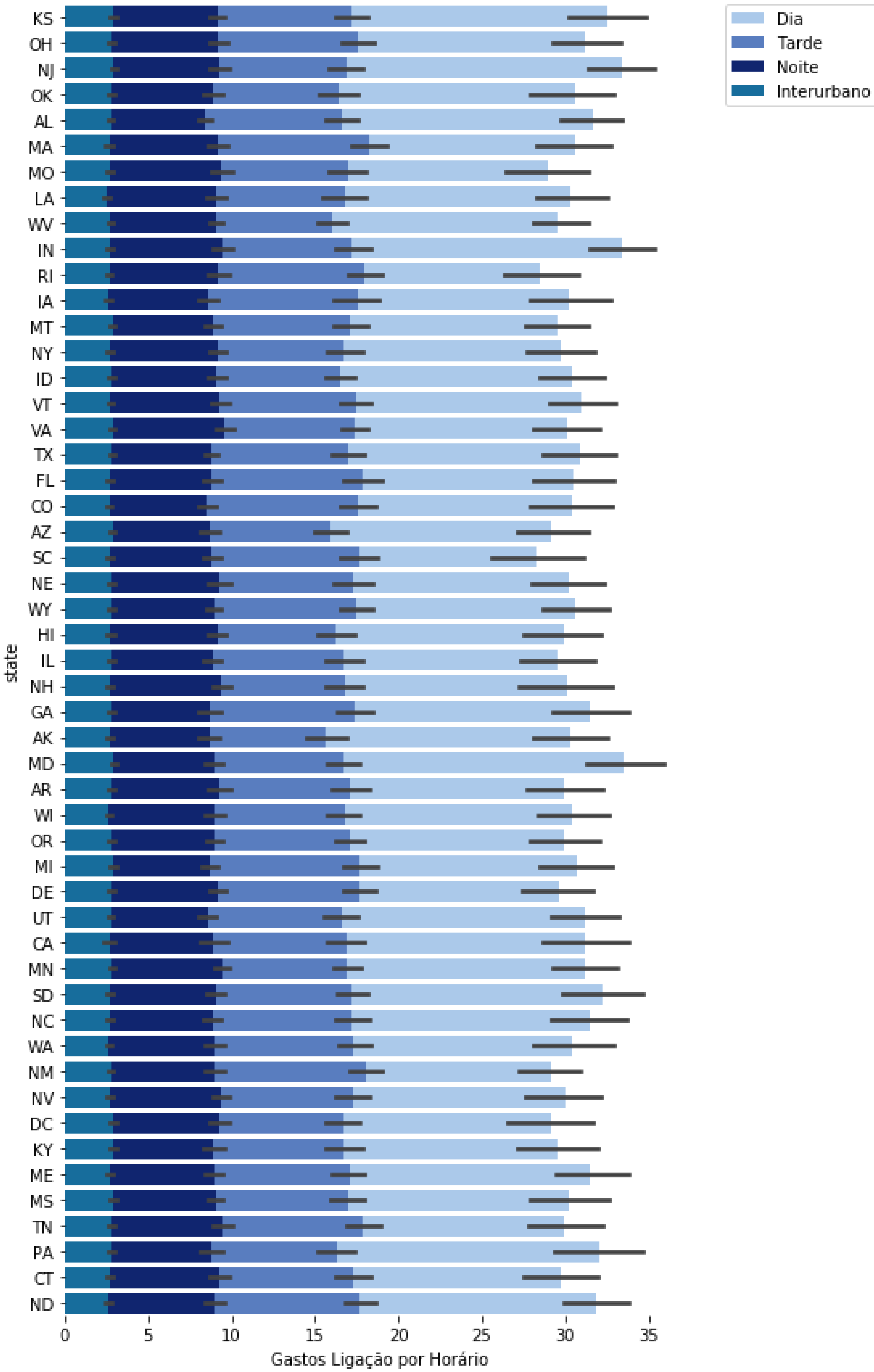
Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x14a35abfc18>
```



In [12]:
```python
# Verifica distribuição das variáveis numéricas
df.hist(figsize=(12, 8),layout=(3,5))
plt.show()
```



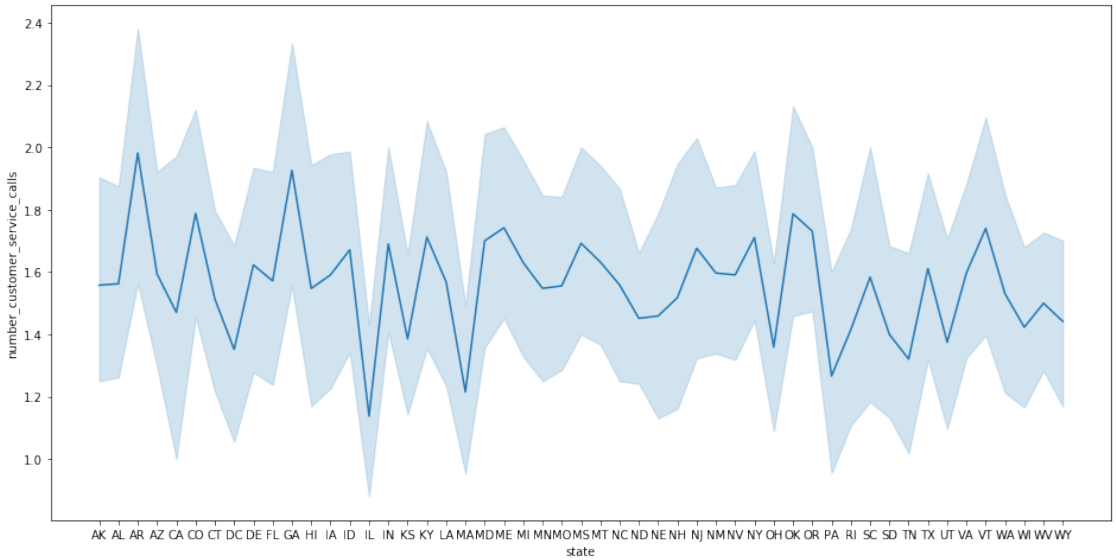

```
In [13]: # Verifica os gastos de ligação / horário por estado

         f, ax = plt.subplots(figsize=(7, 15))

         sns.set_color_codes("pastel")
         sns.barplot(x="total_day_charge", y="state", data=df,
                     label="Dia", color="b")


         sns.set_color_codes("muted")
         sns.barplot(x="total_eve_charge", y="state", data=df,
                     label="Tarde", color="b")

         sns.set_color_codes("dark")
         sns.barplot(x="total_night_charge", y="state", data=df,
                     label="Noite", color="b")

         sns.set_color_codes("colorblind")
         sns.barplot(x="total_intl_charge", y="state", data=df,
                     label="Interurbano", color="b")

         plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
         ax.set(xlabel="Gastos Ligação por Horário")
         sns.despine(left=True, bottom=True)
```



Dia
Tarde
Noite
Interurbano
state
KS
OH
NJ
OK
AL
MA
MO
LA
WV
IN
RI
IA
MT
NY
ID
VT
VA
TX
FL
CO
AZ
SC
NE
WY
HI
IL
NH
GA
AK
MD
AR
WI
OR
MI
DE
UT
CA
MN
SD
NC
WA
NM
NV
DC
KY
ME
MS
TN
PA
CT
ND
0
5
10
15
20
25
30
35
Gastos Ligação por Horário

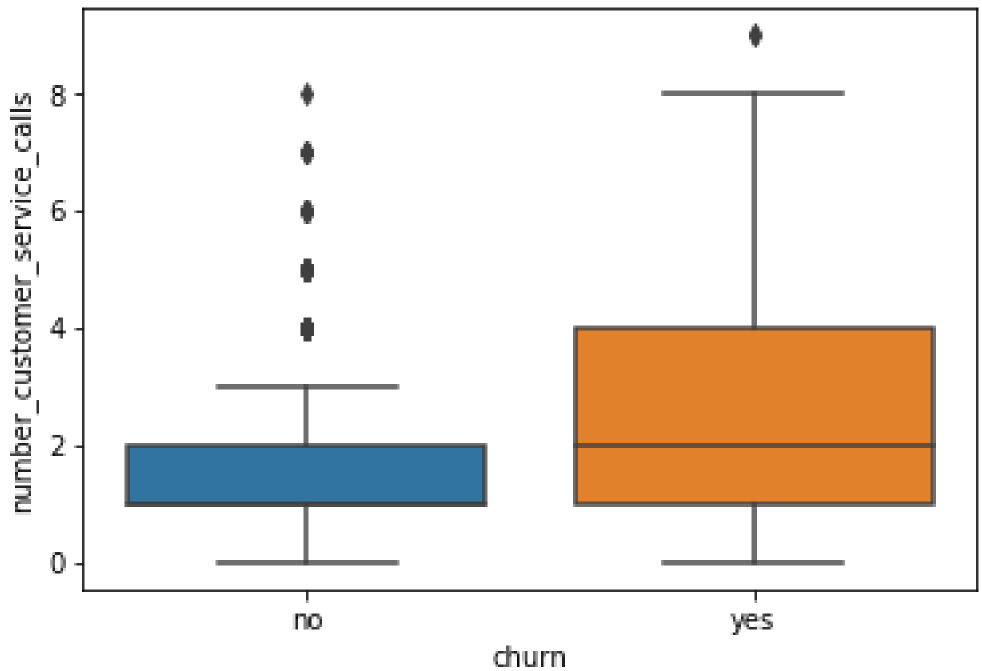
```
In [14]: # # Ligações para atendimento ao consumidor por estado
         f, ax = plt.subplots(figsize=(16, 8))
         sns.lineplot(x='state',y='number_customer_service_calls',data=df)
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x14a44485e48>
```



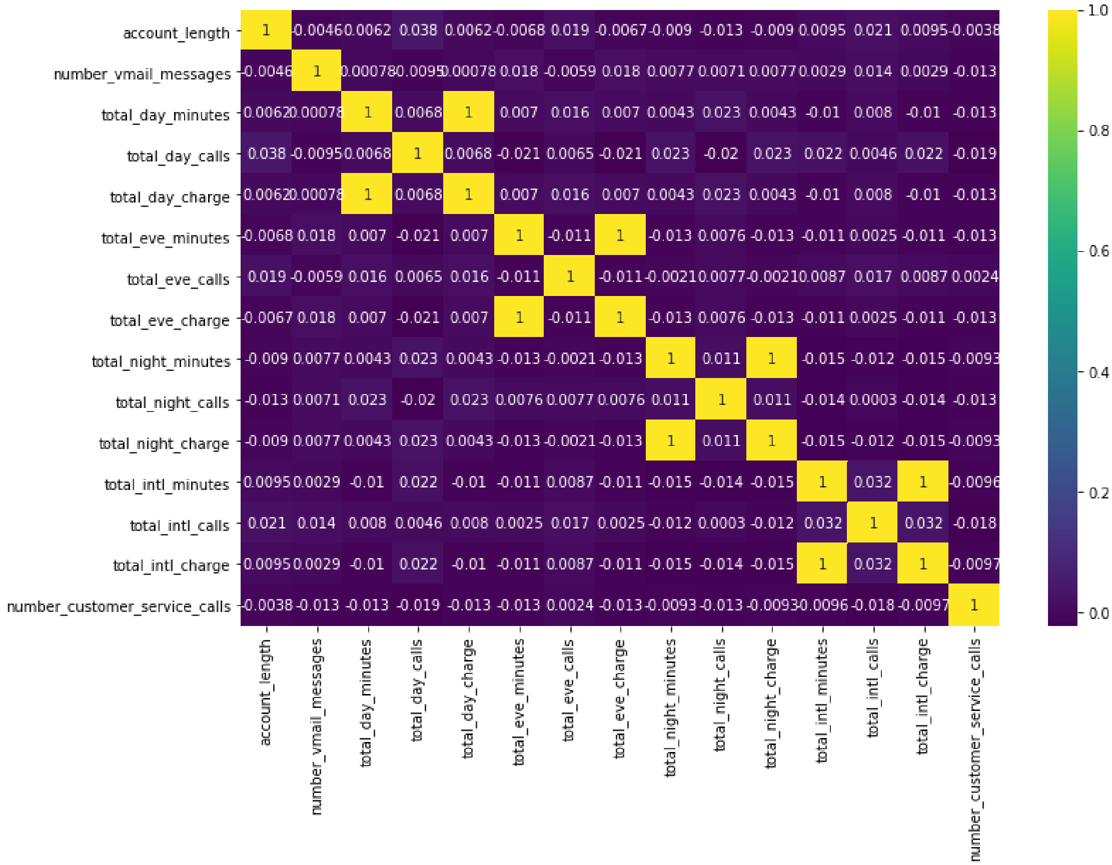

```
In [15]: # Ligações para atendimento ao consumidor X rotatividade
         sns.boxplot(y='number_customer_service_calls',x='churn',data=df)
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x14a444c6e48>
```




```
In [16]: # Verifica a correlação das variáveis numéricas
         fig,ax = plt.subplots(figsize=(12,8))
         sns.heatmap(df.corr(),annot=True,cmap='viridis')
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x14a431d42b0>
```




```
In [17]: # Procura por valores nulos
         df.isnull().values.any()
```

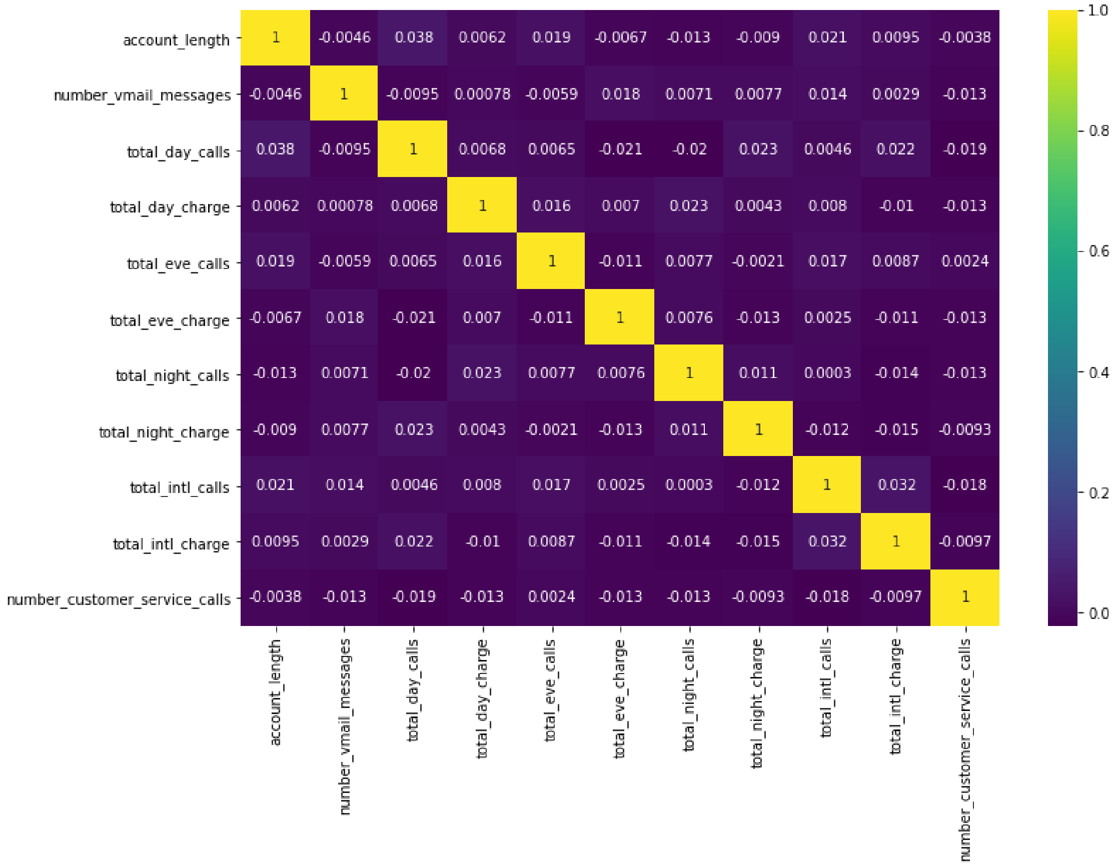
```
Out[17]: False
```

3 Tratamento dos Dados

```
In [18]: # Retira as varíaveis com alta correlação
         df = df.drop(['total_day_minutes','total_eve_minutes','total_night_minutes','total_in
```

```
In [19]: # Verifica novamente a correlação das variáveis
         fig,ax = plt.subplots(figsize=(12,8))
         sns.heatmap(df.corr(),annot=True,cmap='viridis')
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x14a44b61470>
```



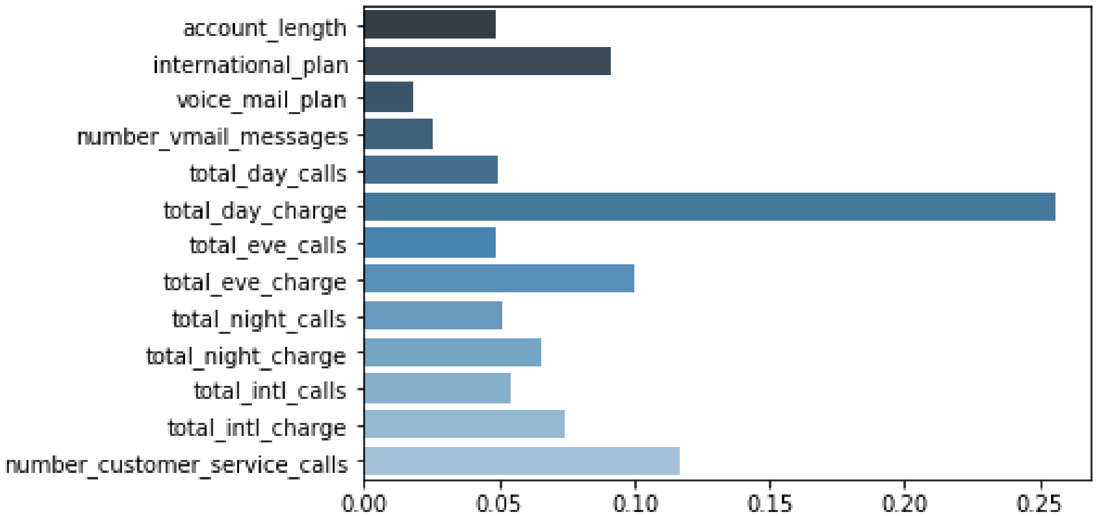
In [20]:
```
# Converte a variável alvo para númerica (0 e 1)

# Cria função para substituir varíaveis "Sim" e "Não" por 1 e 0
def binar(x):
    if x == 'yes':
        return 1
    else:
        return 0

# Aplica função na coluna alvo
df['churn'] = df['churn'].apply(binar)
```

In [21]:
```
# Converte demais variáveis de sim e não para numérica (1 e 0)
df['international_plan'] = df['international_plan'].apply(binar)
df['voice_mail_plan'] = df['voice_mail_plan'].apply(binar)
```

In [22]:
```
# Verifica a importância das variáveis com o RandomFlorest

from sklearn.ensemble import RandomForestClassifier
```

```
rfc = RandomForestClassifier()

var_n = df.drop(['churn','state','area_code'],axis=1)
target = df['churn']

rfc.fit(var_n,target)

sns.barplot(x=rfc.feature_importances_, y=var_n.columns,palette="Blues_d")
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x14a357a71d0>



4 Construção do Modelo

In [23]:
```
# Cria os datasets de variáveis / alvo
# Serão descartadas as varíaveis não numéricas "state" e "area_code"
# Serão descartadas as varíaveis numéricas "voice_mail_plan" e "number_vmail_messages

X = df.drop(['churn','state','area_code','voice_mail_plan','number_vmail_messages'],a
y = df['churn']
```

In [24]:
```
# Treina Modelo Regressão Logística

# Importação dos módulos
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

# Definindo os valores para o número de folds
num_folds = 20
```

```
# Separando os dados em folds
kfold = KFold(num_folds, True)

# Criando o modelo
modelo = LogisticRegression()

# Cross Validation
resultado = cross_val_score(modelo, X, y, cv = kfold)

# Print do resultado
print("Acurácia Modelo 1: %.3f%%" % (resultado.mean() * 100))
```

```
Acurácia Modelo 1: 85.658%
```

In [25]:
```
# Treina o modelo
modelo.fit(X,y)
```

Out[25]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

5 Testa o modelo

In [26]:
```
# Importa dados de teste

teste = pd.read_csv('dados/projeto4_telecom_teste.csv',index_col=[0])
```

In [27]:
```
# Prepara dados de teste

teste = teste.drop(['total_day_minutes','total_eve_minutes','total_night_minutes','to

teste['churn'] = teste['churn'].apply(binar)
teste['international_plan'] = teste['international_plan'].apply(binar)
```

In [28]:
```
# Realiza as previsões

var = teste.drop(['churn','state','area_code','voice_mail_plan','number_vmail_message
alvo = teste['churn']

teste_prev = modelo.predict(var)
```

In [29]:
```
# Verifica a performance do modelo nos dados de teste

from sklearn.metrics import classification_report
```

```python
from sklearn.metrics import confusion_matrix

report = classification_report(alvo, teste_prev)
matrix = confusion_matrix(alvo, teste_prev)

print(report)
print('\n')
print(matrix)
```

```
              precision    recall  f1-score   support

           0       0.88      0.98      0.93      1443
           1       0.59      0.17      0.27       224

    accuracy                           0.87      1667
   macro avg       0.74      0.58      0.60      1667
weighted avg       0.85      0.87      0.84      1667



[[1416   27]
 [ 185   39]]
```