Atividade 3: Verificar se é possível melhorar os resultados de aprendizado fazendo transformações em atributos numéricos

- **Nome**: Matheus Freitas Martins
- **Matrícula**: ES111281

1. Ler os dados do arquivo “churn-telecom.csv” e fazer uma análise exploratória inicial, para conhecer os atributos

In [210…
```
# Importando bibliotecas necessárias
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

# Carregar os dados
df = pd.read_csv('churn-telecom.csv')
```

In [211…
```
df.head()
```

Out[211]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	ch
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7	91	11.01	10.0	3	
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4	103	11.45	13.7	3	
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2	5	
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86	6.6	7	
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1	3	

5 rows × 21 columns

In [212…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   3333 non-null   object
 1   account length          3333 non-null   int64
 2   area code               3333 non-null   int64
 3   phone number            3333 non-null   object
 4   international plan      3333 non-null   object
 5   voice mail plan         3333 non-null   object
 6   number vmail messages   3333 non-null   int64
 7   total day minutes       3333 non-null   float64
 8   total day calls         3333 non-null   int64
 9   total day charge        3333 non-null   float64
 10  total eve minutes       3333 non-null   float64
 11  total eve calls         3333 non-null   int64
 12  total eve charge        3333 non-null   float64
 13  total night minutes     3333 non-null   float64
 14  total night calls       3333 non-null   int64
 15  total night charge      3333 non-null   float64
 16  total intl minutes      3333 non-null   float64
 17  total intl calls        3333 non-null   int64
 18  total intl charge       3333 non-null   float64
 19  customer service calls  3333 non-null   int64
 20  churn                   3333 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 524.2+ KB
```

In [213…
```
df.isnull().sum()
```

Out[213]:
```
state                   0
account length          0
area code               0
phone number            0
international plan      0
voice mail plan         0
number vmail messages   0
total day minutes       0
total day calls         0
total day charge        0
total eve minutes       0
total eve calls         0
total eve charge        0
total night minutes     0
total night calls       0
total night charge      0
total intl minutes      0
total intl calls        0
total intl charge       0
customer service calls  0
churn                   0
dtype: int64
```

In [214…
```
df.value_counts()
```

Out[214]:
```
state  account length  area code  phone number  international plan  voice mail plan  number vmail messages  total day minutes  t
otal day calls  total day charge  total eve minutes  total eve calls  total eve charge  total night minutes  total night calls
total night charge  total intl minutes  total intl calls  total intl charge  customer service calls  churn
AK     1               408        373-1028      no                  no               0                      175.2              7
4               29.78             151.7              79               12.89             230.5                109
10.37               5.3               3                1.43              1                      False    1
NV     87              510        331-8484      no                  yes              39                     82.6               1
13              14.04             224.4              63               19.07             163.6                88
7.36                9.5               1                2.57              3                      False    1
OH     52              408        327-9289      no                  yes              31                     142.1              7
7               24.16             193.0              97               16.41             253.4                88
11.40               11.0              4                2.97              1                      False    1
       56              408        349-2654      no                  no               0                      91.1               9
0               15.49             179.3              115              15.24             300.7                89
13.53               11.9              8                3.21              2                      False    1
       61              510        327-5525      yes                 yes              16                     143.5              7
6               24.40             242.6              58               20.62             147.7                95
6.65                11.3              3                3.05              0                      False    1

..
LA     117             408        335-4719      no                  no               0                      184.5              9
7               31.37             351.6              80               29.89             215.8                90
9.71                8.7               4                2.35              1                      False    1
       118             408        405-9496      no                  no               0                      187.4              9
7               31.86             177.8              89               15.11             233.4                97
10.50               12.2              6                3.29              1                      False    1
       121             408        390-8760      no                  no               0                      181.5              1
21              30.86             218.4              98               18.56             161.6                103
7.27                8.5               5                2.30              1                      False    1
       123             415        382-7659      no                  yes              33                     146.6              8
7               24.92             114.8              59               9.76              220.4                99
9.92                2.9               7                0.78              0                      False    1
WY     225             415        374-1213      no                  no               0                      182.7              1
42              31.06             246.5              63               20.95             218.0                103
9.81                8.8               2                2.38              1                      False    1
Length: 3333, dtype: int64
```

In [215…
```
df.describe()
```

Out[215]:

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.980348	100.114311	17.083540	200.872037	100.107711
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.713844	19.922625	4.310668	50.573847	19.568609
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	23.200000	33.000000
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.600000	87.000000	14.160000	167.000000	87.000000
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.400000	100.000000	17.120000	201.200000	100.000000
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.300000	114.000000	20.000000	235.300000	113.000000
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.700000	170.000000	30.910000	395.000000	175.000000

In [216…
```
for column in df.columns:
    unique_values = df[column].unique()
    print(f"Valores únicos na coluna '{column}': {unique_values}")
```

```
Valores únicos na coluna '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']
Valores únicos na coluna 'account length': [128 107 137  84  75 118 121 147 117 141  65  74 168  95  62 161  85  93
  76  73  77 130 111 132 174  57  54  20  49 142 172  12  72  36  78 136
 149  98 135  34 160  64  59 119  97  52  60  10  96  87  81  68 125 116
  38  40  43 113 126 150 138 162  90  50  82 144  46  70  55 106  94 155
  80 104  99 120 108 122 157 103  63 112  41 193  61  92 131 163  91 127
 110 140  83 145  56 151 139   6 115 146 185 148  32  25 179  67  19 170
 164  51 208  53 105  66  86  35  88 123  45 100 215  22  33 114  24 101
 143  48  71 167  89 199 166 158 196 209  16  39 173 129  44  79  31 124
  37 159 194 154  21 133 224  58  11 109 102 165  18  30 176  47 190 152
  26  69 186 171  28 153 169  13  27   3  42 189 156 134 243  23   1 205
 200   5   9 178 181 182 217 177 210  29 180   2  17   7 212 232 192 195
 197 225 184 191 201  15 183 202   8 175   4 188 204 221]
Valores únicos na coluna 'area code': [415 408 510]
Valores únicos na coluna 'phone number': ['382-4657' '371-7191' '358-1921' ... '328-8230' '364-6381' '400-4344']
Valores únicos na coluna 'international plan': ['no' 'yes']
Valores únicos na coluna 'voice mail plan': ['yes' 'no']
Valores únicos na coluna 'number vmail messages': [25 26  0 24 37 27 33 39 30 41 28 34 46 29 35 21 32 42 36 22 23 43 31 38
 40 48 18 17 45 16 20 14 19 51 15 11 12 47  8 44 49  4 10 13 50  9]
Valores únicos na coluna 'total day minutes': [265.1 161.6 243.4 ... 321.1 231.1 180.8]
Valores únicos na coluna 'total day calls': [110 123 114  71 113  98  88  79  97  84 137 127  96  70  67 139  66  90
 117  89 112 103  86  76 115  73 109  95 105 121 118  94  80 128  64 106
 102  85  82  77 120 133 135 108  57  83 129  91  92  74  93 101 146  72
  99 104 125  61 100  87 131  65 124 119  52  68 107  47 116 151 126 122
 111 145  78 136 140 148  81  55  69 158 134 130  63  53  75 141 163  59
 132 138  54  58  62 144 143 147  36  40 150  56  51 165  30  48  60  42
   0  45 160 149 152 142 156  35  49 157  44]
Valores únicos na coluna 'total day charge': [45.07 27.47 41.38 ... 54.59 39.29 30.74]
Valores únicos na coluna 'total eve minutes': [197.4 195.5 121.2 ... 153.4 288.8 265.9]
Valores únicos na coluna 'total eve calls': [ 99 103 110  88 122 101 108  94  80 111  83 148  71  75  76  97  90  65
  93 121 102  72 112 100  84 109  63 107 115 119 116  92  85  98 118  74
 117  58  96  66  67  62  77 164 126 142  64 104  79  95  86 105  81 113
 106  59  48  82  87 123 114 140 128  60  78 125  91  46 138 129  89 133
 136  57 135 139  51  70 151 137 134  73 152 168  68 120  69 127 132 143
  61 124  42  54 131  52 149  56  37 130  49 146 147  55  12  50 157 155
  45 144  36 156  53 141  44 153 154 150  43   0 145 159 170]
Valores únicos na coluna 'total eve charge': [16.78 16.62 10.3  ... 13.04 24.55 22.6 ]
Valores únicos na coluna 'total night minutes': [244.7 254.4 162.6 ... 280.9 120.1 279.1]
Valores únicos na coluna 'total night calls': [ 91 103 104  89 121 118  96  90  97 111  94 128 115  99  75 108  74 133
  64  78 105  68 102 148  98 116  71 109 107 135  92  86 127  79  87 129
  57  77  95  54 106  53  67 139  60 100  61  73 113  76 119  88  84  62
 137  72 142 114 126 122  81 123 117  82  80 120 130 134  59 112 132 110
 101 150  69 131  83  93 124 136 125  66 143  58  55  85  56  70  46  42
 152  44 145  50 153  49 175  63 138 154 140 141 146  65  51 151 158 155
 157 147 144 149 166  52  33 156  38  36  48 164]
Valores únicos na coluna 'total night charge': [11.01 11.45  7.32  8.86  8.41  9.18  9.57  9.53  9.71 14.69  9.4   8.82
  6.35  8.65  9.14  7.23  4.02  5.83  7.46  8.68  9.43  8.18  8.53 10.67
 11.28  8.22  4.59  8.17  8.04 11.27 11.08 13.2  12.61  9.61  6.88  5.82
 10.25  4.58  8.47  8.45  5.5  14.02  8.03 11.94  7.34  6.06 10.9   6.44
  3.18 10.66 11.21 12.73 10.28 12.16  6.34  8.15  5.84  8.52  7.5   7.48
  6.21 11.95  7.15  9.63  7.1   6.91  6.69 13.29 11.46  7.76  6.86  8.16
 12.15  7.79  7.99 10.29 10.08 12.53  7.91 10.02  8.61 14.54  8.21  9.09
  4.93 11.39 11.88  5.75  7.83  8.59  7.52 12.38  7.21  5.81  8.1  11.04
 11.19  8.55  8.42  9.76  9.87 10.86  5.36 10.03 11.15  9.51  6.22  2.59
  7.65  6.45  9.    6.4   9.94  5.08 10.23 11.36  6.97 10.16  7.88 11.91
  6.61 11.55 11.76  9.27  9.29 11.12 10.69  8.8  11.85  7.14  8.71 11.42
  4.94  9.02 11.22  4.97  9.15  5.45  7.27 12.91  7.75 13.46  6.32 12.13
 11.97  6.93 11.66  7.42  6.19 11.41 10.33 10.65 11.92  4.77  4.38  7.41
 12.1   7.69  8.78  9.36  9.05 12.7   6.16  6.05 10.85  8.93  3.48 10.4
  5.05 10.71  9.37  6.75  8.12 11.77 11.49 11.06 11.25 11.03 10.82  8.91
  8.57  8.09 10.05 11.7  10.17  8.74  5.51 11.11  3.29 10.13  6.8   8.49
  9.55 11.02  9.91  7.84 10.62  9.97  3.44  7.35  9.79  8.89  8.14  6.94
 10.49 10.57 10.2   6.29  8.79 10.04 12.41 15.97  9.1  11.78 12.75 11.07
 12.56  8.63  8.02 10.42  8.7   9.98  7.62  8.33  6.59 13.12 10.46  6.63
  8.32  9.04  9.28 10.76  9.64 11.44  6.48 10.81 12.66 11.34  8.75 13.05
 11.48 14.04 13.47  5.63  6.6   9.72 11.68  6.41  9.32 12.95 13.37  9.62
  6.03  8.25  8.26 11.96  9.9   9.23  5.58  7.22  6.64 12.29 12.93 11.32
  6.85  8.88  7.03  8.48  3.59  5.86  6.23  7.61  7.66 13.63  7.9  11.82
  7.47  6.08  8.4   5.74 10.94 10.35 10.68  4.34  8.73  5.14  8.24  9.99
 13.93  8.64 11.43  5.79  9.2  10.14 12.11  7.53 12.46  8.46  8.95  9.84
 10.8  11.23 10.15  9.21 14.46  6.67 12.83  9.66  9.59 10.48  8.36  4.84
 10.54  8.39  7.43  9.06  8.94 11.13  8.87  8.5   7.6  10.73  9.56 10.77
  7.73  3.47 11.86  8.11  9.78  9.42  9.65  7.    7.39  9.88  6.56  5.92
  6.95 15.71  8.06  4.86  7.8   8.58 10.06  5.21  6.92  6.15 13.49  9.38
 12.62 12.26  8.19 11.65 11.62 10.83  7.92  7.33 13.01 13.26 12.22 11.58
  5.97 10.99  8.38  9.17  8.08  5.71  3.41 12.63 11.79 12.96  7.64  6.58
 10.84 10.22  6.52  5.55  7.63  5.11  5.89 10.78  3.05 11.89  8.97 10.44
 10.5   9.35  5.66 11.09  9.83  5.44 10.11  6.39 11.93  8.62 12.06  6.02
  8.85  5.25  8.66  6.73 10.21 11.59 13.87  7.77 10.39  5.54  6.62 13.33
  6.24 12.59  6.3   6.79  8.28  9.03  8.07  5.52 12.14 10.59  7.54  7.67
  5.47  8.81  8.51 13.45  8.77  6.43 12.01 12.08  7.07  6.51  6.84  9.48
 13.78 11.54 11.67  8.13 10.79  7.13  4.72  4.64  8.96 13.03  6.07  3.51
  6.83  6.12  9.31  9.58  4.68  5.32  9.26 11.52  9.11 10.55 11.47  9.3
 13.82  8.44  5.77 10.96 11.74  8.9  10.47  7.85 10.92  4.74  9.74 10.43
  9.96 10.18  9.54  7.89 12.36  8.54 10.07  9.46  7.3  11.16  9.16 10.19
  5.99 10.88  5.8   7.19  4.55  8.31  8.01 14.43  8.3  14.3   6.53  8.2
 11.31 13.    6.42  4.24  7.44  7.51 13.1   9.49  6.14  8.76  6.65 10.56
  6.72  8.29 12.09  5.39  2.96  7.59  7.24  4.28  9.7   8.83 13.3  11.37
  9.33  5.01  3.26 11.71  8.43  9.68 15.56  9.8   3.61  6.96 11.61 12.81
```

```
10.87 13.84  5.03  5.17  2.03 10.34  9.34  7.95 10.09  9.95  7.11  9.22
 6.13 11.05  9.89  9.39 14.06 10.26 13.31 15.43 16.39  6.27 10.64 11.5
12.48  8.27 13.53 10.36 12.24  8.69 10.52  9.07 11.51  9.25  8.72  6.78
 8.6  11.84  5.78  5.85 12.3   5.76 12.07  9.6   8.84 12.39 10.1   9.73
 2.85  6.66  2.45  5.28 11.73 10.75  7.74  6.76  6.    7.58 13.69  7.93
 7.68  9.75  4.96  5.49 11.83  7.18  9.19  7.7   7.25 10.74  4.27 13.8
 9.12  4.75  7.78 11.63  7.55  2.25  9.45  9.86  7.71  4.95  7.4  11.17
11.33  6.82 13.7   1.97 10.89 12.77 10.31  5.23  5.27  9.41  6.09 10.61
 7.29  4.23  7.57  3.67 12.69 14.5   5.95  7.87  5.96  5.94 12.23  4.9
12.33  6.89  9.67 12.68 12.87  3.7   6.04 13.13 15.74 11.87  4.7   4.67
 7.05  5.42  4.09  5.73  9.47  8.05  6.87  3.71 15.86  7.49 11.69  6.46
10.45 12.9   5.41 11.26  1.04  6.49  6.37 12.21  6.77 12.65  7.86  9.44
 4.3   7.38  5.02 10.63  2.86 17.19  8.67  8.37  6.9  10.93 10.38  7.36
10.27 10.95  6.11  4.45 11.9  15.01 12.84  7.45  6.98 11.72  7.56 11.38
10.    4.42  9.81  5.56  6.01 10.12 12.4  16.99  5.68 11.64  3.78  7.82
 9.85 13.74 12.71 10.98 10.01  9.52  7.31  8.35 11.35  9.5  14.03  3.2
 7.72 13.22 10.7   8.99 10.6  13.02  9.77 12.58 12.35 12.2  11.4  13.91
 3.57 14.65 12.28  5.13 10.72 12.86 14.    7.12 12.17  4.71  6.28  8.
 7.01  5.91  5.2  12.   12.02 12.88  7.28  5.4  12.04  5.24 10.3  10.41
13.41 12.72  9.08  7.08 13.5   5.35 12.45  5.3  10.32  5.15 12.67  5.22
 5.57  3.94  4.41 13.27 10.24  4.25 12.89  5.72 12.5  11.29  3.25 11.53
 9.82  7.26  4.1  10.37  4.98  6.74 12.52 14.56  8.34  3.82  3.86 13.97
11.57  6.5  13.58 14.32 13.75 11.14 14.18  9.13  4.46  4.83  9.69 14.13
 7.16  7.98 13.66 14.78 11.2   9.93 11.    5.29  9.92  4.29 11.1  10.51
12.49  4.04 12.94  7.09  6.71  7.94  5.31  5.98  7.2  14.82 13.21 12.32
10.58  4.92  6.2   4.47 11.98  6.18  7.81  4.54  5.37  7.17  5.33 14.1
 5.7  12.18  8.98  5.1  14.67 13.95 16.55 11.18  4.44  4.73  2.55  6.31
 2.43  9.24  7.37 13.42 12.42 11.8  14.45  2.89 13.23 12.6  13.18 12.19
14.81  6.55 11.3  12.27 13.98  8.23 15.49  6.47 13.48 13.59 13.25 17.77
13.9   3.97 11.56 14.08 13.6   6.26  4.61 12.76 15.76  6.38  3.6  12.8
 5.9   7.97  5.   10.97  5.88 12.34 12.03 14.97 15.06 12.85  6.54 11.24
12.64  7.06  5.38 13.14  3.99  3.32  4.51  4.12  3.93  2.4  11.75  4.03
15.85  6.81 14.25 14.09 16.42  6.7  12.74  2.76 12.12  6.99  6.68 11.81
 7.96  5.06 13.16  2.13 13.17  5.12  5.65 12.37 10.53]
Valores únicos na coluna 'total intl minutes': [10.  13.7 12.2  6.6 10.1  6.3  7.5  7.1  8.7 11.2 12.7  9.1 12.3 13.1
 5.4 13.8  8.1 13.  10.6  5.7  9.5  7.7 10.3 15.5 14.7 11.1 14.2 12.6
11.8  8.3 14.5 10.5  9.4 14.6  9.2  3.5  8.5 13.2  7.4  8.8 11.   7.8
 6.8 11.4  9.3  9.7 10.2  8.   5.8 12.1 12.  11.6  8.2  6.2  7.3  6.1
11.7 15.   9.8 12.4  8.6 10.9 13.9  8.9  7.9  5.3  4.4 12.5 11.3  9.
 9.6 13.3 20.   7.2  6.4 14.1 14.3  6.9 11.5 15.8 12.8 16.2  0.  11.9
 9.9  8.4 10.8 13.4 10.7 17.6  4.7  2.7 13.5 12.9 14.4 10.4  6.7 15.4
 4.5  6.5 15.6  5.9 18.9  7.6  5.   7.  14.  18.  16.  14.8  3.7  2.
 4.8 15.3  6.  13.6 17.2 17.5  5.6 18.2  3.6 16.5  4.6  5.1  4.1 16.3
14.9 16.4 16.7  1.3 15.2 15.1 15.9  5.5 16.1  4.  16.9  5.2  4.2 15.7
17.   3.9  3.8  2.2 17.1  4.9 17.9 17.3 18.4 17.8  4.3  2.9  3.1  3.3
 2.6  3.4  1.1 18.3 16.6  2.1  2.4  2.5]
Valores únicos na coluna 'total intl calls': [ 3  5  7  6  4  2  9 19  1 10 15  8 11  0 12 13 18 14 16 20 17]
Valores únicos na coluna 'total intl charge': [2.7  3.7  3.29 1.78 2.73 1.7  2.03 1.92 2.35 3.02 3.43 2.46 3.32 3.54
 1.46 3.73 2.19 3.51 2.86 1.54 2.57 2.08 2.78 4.19 3.97 3.   3.83 3.4
 3.19 2.24 3.92 2.84 2.54 3.94 2.48 0.95 2.3  3.56 2.   2.38 2.97 2.11
 1.84 3.08 2.51 2.62 2.75 2.16 1.57 3.27 3.24 3.13 2.21 1.67 1.97 1.65
 3.16 4.05 2.65 3.35 2.32 2.94 3.75 2.4  2.13 1.43 1.19 3.38 3.05 2.43
 2.59 3.59 5.4  1.94 1.73 3.81 3.86 1.86 3.11 4.27 3.46 4.37 0.   3.21
 2.67 2.27 2.92 3.62 2.89 4.75 1.27 0.73 3.65 3.48 3.89 2.81 1.81 4.16
 1.22 1.76 4.21 1.59 5.1  2.05 1.35 1.89 3.78 4.86 4.32 4.   1.   0.54
 1.3  4.13 1.62 3.67 4.64 4.73 1.51 4.91 0.97 4.46 1.24 1.38 1.11 4.4
 4.02 4.43 4.51 0.35 4.1  4.08 4.29 1.49 4.35 1.08 4.56 1.4  1.13 4.24
 4.59 1.05 1.03 0.59 4.62 1.32 4.83 4.67 4.97 4.81 1.16 0.78 0.84 0.89
 0.7  0.92 0.3  4.94 4.48 0.57 0.65 0.68]
Valores únicos na coluna 'customer service calls': [1 0 2 3 4 5 7 9 6 8]
Valores únicos na coluna 'churn': [False  True]
```

- **state**: Atributo categórico que representa o estado dos EUA em que o cliente reside.
- **account length**: Atributo numérico que indica a duração da conta do cliente em meses.
- **area code**: Atributo numérico que indica o código de área do telefone do cliente.
- **phone number**: Atributo categórico que representa o número de telefone do cliente.
- **international plan**: Atributo categórico que indica se o cliente possui um plano internacional (yes/no).
- **voice mail plan**: Atributo categórico que indica se o cliente possui um plano de correio de voz (yes/no).
- **number vmail messages**: Atributo numérico que representa o número de mensagens de correio de voz que o cliente possui.
- **total day minutes**: Atributo numérico que representa a quantidade total de minutos que o cliente usa durante o dia.
- **total day calls**: Atributo numérico que representa a quantidade total de chamadas feitas pelo cliente durante o dia.
- **total day charge**: Atributo numérico que representa a cobrança total do cliente pelas chamadas feitas durante o dia.
- **total eve minutes**: Atributo numérico que representa a quantidade total de minutos que o cliente usa durante a noite.
- **total eve calls**: Atributo numérico que representa a quantidade total de chamadas feitas pelo cliente durante a noite.
- **total eve charge**: Atributo numérico que representa a cobrança total do cliente pelas chamadas feitas durante a noite.
- **total night minutes**: Atributo numérico que representa a quantidade total de minutos que o cliente usa durante a noite.
- **total night calls**: Atributo numérico que representa a quantidade total de chamadas feitas pelo cliente durante a noite.
- **total night charge**: Atributo numérico que representa a cobrança total do cliente pelas chamadas feitas durante a noite.
- **total intl minutes**: Atributo numérico que representa a quantidade total de minutos que o cliente usa para chamadas internacionais.
- **total intl calls**: Atributo numérico que representa a quantidade total de chamadas internacionais feitas pelo cliente.
- **total intl charge**: Atributo numérico que representa a cobrança total do cliente pelas chamadas internacionais feitas.
- **customer service calls**: Atributo numérico que representa o número de chamadas de atendimento ao cliente feitas pelo cliente.
- **churn**: Atributo categórico que indica se o cliente cancelou o serviço (True/False).

In [217…
```python
# Função para encapsular o processo de treinamento e teste do classificador
def train_and_evaluate(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # Dividindo os dados em conjuntos d
    clf = RandomForestClassifier(random_state=42) # Criando o classificador Random Forest
    clf.fit(X_train, y_train) # Ajustando o classificador aos dados de treinamento.
    y_pred = clf.predict(X_test) # Fazendo previsões no conjunto de teste.
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy

# Função com Scaled para encapsular o processo de treinamento e teste do classificador
def train_and_evaluate_scaled(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    clf = RandomForestClassifier(random_state=42)
    clf.fit(X_train_scaled, y_train)
    y_pred = clf.predict(X_test_scaled)
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy
```

Pré-processamento

In [218…
```python
# Convertendo as colunas 'international plan' e 'voice mail plan' em valores numéricos (1 para 'yes' e 0 para 'no').
df['international plan'] = df['international plan'].map({'yes': 1, 'no': 0})
df['voice mail plan'] = df['voice mail plan'].map({'yes': 1, 'no': 0})

# Convertendo a coluna 'churn' em um tipo de dados inteiro.
df['churn'] = df['churn'].astype(int)
```

2. Aplicar um algoritmo de classificação de sua escolha e calcular a precisão. A variável alvo para ser classificada é “churn”

"Churn" é um termo utilizado em negócios para descrever a taxa de clientes que param de usar um produto ou serviço durante um determinado período de tempo.

Algoritmo de classificação escolhido: Random Forest

Dividindo os dados em conjuntos de treinamento e teste

In [219…
```python
# Selecionando as colunas necessárias para treinar o modelo (X) e a variável de destino (y).
X = df.drop(['state', 'area code', 'phone number', 'churn'], axis=1)
y = df['churn']
```

In [220…
```python
# Calculando a acurácia antes das transformações
accuracy_before = train_and_evaluate_scaled(X, y)
print(f"Acurácia antes das transformações: {accuracy_before}")
```

```
Acurácia antes das transformações: 0.9475262368815592
```

3. Realizar transformações (arredondamento, tornar binário, binning) em atributos numéricos que aparentemente podem se beneficiar dessas transformações

In [221…
```python
# Realizando transformações nos atributos numéricos
# Arredondando os valores das colunas 'total day minutes', 'total eve minutes', 'total night minutes' e 'total intl minutes' para
df['total day minutes'] = df['total day minutes'].apply(lambda x: round(x))
df['total eve minutes'] = df['total eve minutes'].apply(lambda x: round(x))
df['total night minutes'] = df['total night minutes'].apply(lambda x: round(x))
df['total intl minutes'] = df['total intl minutes'].apply(lambda x: round(x))

# convertendo a coluna "number vmail messages" em uma coluna binária, onde os valores serão 1 se houver mensagens de correio de v
df['number vmail messages'] = df['number vmail messages'].apply(lambda x: 1 if x > 0 else 0)
```

4. Aplicar novamente o algoritmo de classificação e comparar os resultados

In [222…
```python
# Dividindo os dados em conjuntos de treinamento e teste novamente após as transformações
X = df.drop(['state', 'area code', 'phone number', 'churn'], axis=1)

# Calculando a acurácia após as transformações
accuracy_after = train_and_evaluate_scaled(X, y)
print(f"Acurácia após transformações: {accuracy_after}")

# Comparando os resultados
print(f"Diferença na acurácia: {accuracy_after - accuracy_before}")
```

```
Acurácia após transformações: 0.9550224887556222
Diferença na acurácia: 0.007496251874062998
```

5. Repetir os passos 3 e 4 para diferentes atributos e formas de transformação

In [223…
```
# Definindo o número de intervalos
num_intervals = 3
labels = ['baixo', 'médio', 'alto']
# Realizando o binning usando o método qcut, dividindo os dados em intervalos de quantis igualmente espaçados.
df['total day minutes'] = pd.qcut(df['total day minutes'], num_intervals, labels=labels)
# Convertendo a coluna 'total day minutes category' para códigos numéricos
df['total day minutes'] = df['total day minutes'].astype('category').cat.codes

# Realizando o binning usando o método qcut diretamente na coluna 'total eve minutes'
df['total eve minutes'] = pd.qcut(df['total eve minutes'], num_intervals, labels=labels)
# Convertendo a coluna 'total eve minutes' para códigos numéricos
df['total eve minutes'] = df['total eve minutes'].astype('category').cat.codes

# Realizando o binning usando o método qcut diretamente na coluna 'total night minutes'
df['total night minutes'] = pd.qcut(df['total night minutes'], num_intervals, labels=labels)
# Convertendo a coluna 'total night minutes' para códigos numéricos
df['total night minutes'] = df['total night minutes'].astype('category').cat.codes

# Descobrindo os percentis para criar os intervalos
labels_intl = ['intervalo1', 'intervalo2', 'intervalo3', 'intervalo4']
p25 = df['total intl minutes'].quantile(0.25)
p50 = df['total intl minutes'].quantile(0.5)
p75 = df['total intl minutes'].quantile(0.75)
# Definindo os intervalos usando os percentis
eve_minutes_bins = [df['total intl minutes'].min(), p25, p50, p75, df['total intl minutes'].max()]
# Realizando o binning usando o método pd.cut() diretamente na coluna 'total eve minutes'
df['total intl minutes'] = pd.cut(df['total intl minutes'], bins=eve_minutes_bins, labels=labels_intl)
# Convertendo a coluna 'total eve minutes' para códigos numéricos
df['total intl minutes'] = df['total intl minutes'].astype('category').cat.codes
```

In [224…
```
# Dividindo os dados em conjuntos de treinamento e teste novamente após as transformações
X = df.drop(['state', 'area code', 'phone number', 'churn'], axis=1)

# Calculando a acurácia após as transformações
new_accuracy_after = train_and_evaluate_scaled(X, y)
print(f"Acurácia após transformações: {accuracy_after}")

# Comparando os resultados
print(f"Diferença na acurácia: {new_accuracy_after - accuracy_after}")
```

```
Acurácia após transformações: 0.9550224887556222
Diferença na acurácia: -0.011994002998500841
```

Fazendo o mesmo processo para o arquivo “heart.csv”.

1. Ler os dados do arquivo “heart.csv” e fazer uma análise exploratória inicial, para conhecer os atributos

In [225…
```
df2 = pd.read_csv("heart.csv")
df2.head()
```

Out[225]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [226…
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [227…
```
df2.describe()
```

Out[227]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000

In [228…
```
df2.value_counts()
```

Out[228]:
```
age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  thal  target
38   1    2   138       175   0    1        173      0      0.0      2      4   2     1         2
59   1    0   110       239   0    0        142      1      1.2      1      1   3     0         1
              126       218   1    1        134      0      2.2      1      1   1     0         1
              140       221   0    1        164      1      0.0      2      0   2     1         1
              170       326   0    0        140      1      3.4      0      0   3     0         1
                                                                                              ..
51   1    2   94        227   0    1        154      1      0.0      2      1   3     1         1
              140       299   0    1        173      1      1.6      2      0   3     0         1
                        298   0    1        122      1      4.2      1      3   3     0         1
                        261   0    0        186      1      0.0      2      0   2     1         1
77   1    0   125       304   0    0        162      1      0.0      2      3   2     0         1
Length: 302, dtype: int64
```

In [229…
```
df2.isnull().sum()
```

Out[229]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [230…
```
for column in df2.columns:
    unique_values = df2[column].unique()
    print(f"Valores únicos na coluna '{column}': {unique_values}")
```

```
Valores únicos na coluna 'age': [63 37 41 56 57 44 52 54 48 49 64 58 50 66 43 69 59 42 61 40 71 51 65 53
 46 45 39 47 62 34 35 29 55 60 67 68 74 76 70 38 77]
Valores únicos na coluna 'sex': [1 0]
Valores únicos na coluna 'cp': [3 2 1 0]
Valores únicos na coluna 'trestbps': [145 130 120 140 172 150 110 135 160 105 125 142 155 104 138 128 108 134
 122 115 118 100 124  94 112 102 152 101 132 148 178 129 180 136 126 106
 156 170 146 117 200 165 174 192 144 123 154 114 164]
Valores únicos na coluna 'chol': [233 250 204 236 354 192 294 263 199 168 239 275 266 211 283 219 340 226
 247 234 243 302 212 175 417 197 198 177 273 213 304 232 269 360 308 245
 208 264 321 325 235 257 216 256 231 141 252 201 222 260 182 303 265 309
 186 203 183 220 209 258 227 261 221 205 240 318 298 564 277 214 248 255
 207 223 288 160 394 315 246 244 270 195 196 254 126 313 262 215 193 271
 268 267 210 295 306 178 242 180 228 149 278 253 342 157 286 229 284 224
 206 167 230 335 276 353 225 330 290 172 305 188 282 185 326 274 164 307
 249 341 407 217 174 281 289 322 299 300 293 184 409 259 200 327 237 218
 319 166 311 169 187 176 241 131]
Valores únicos na coluna 'fbs': [1 0]
Valores únicos na coluna 'restecg': [0 1 2]
Valores únicos na coluna 'thalach': [150 187 172 178 163 148 153 173 162 174 160 139 171 144 158 114 151 161
 179 137 157 123 152 168 140 188 125 170 165 142 180 143 182 156 115 149
 146 175 186 185 159 130 190 132 147 154 202 166 164 184 122 169 138 111
 145 194 131 133 155 167 192 121  96 126 105 181 116 108 129 120 112 128
 109 113  99 177 141 136  97 127 103 124  88 195 106  95 117  71 118 134
  90]
Valores únicos na coluna 'exang': [0 1]
Valores únicos na coluna 'oldpeak': [2.3 3.5 1.4 0.8 0.6 0.4 1.3 0.  0.5 1.6 1.2 0.2 1.8 1.  2.6 1.5 3.  2.4
 0.1 1.9 4.2 1.1 2.  0.7 0.3 0.9 3.6 3.1 3.2 2.5 2.2 2.8 3.4 6.2 4.  5.6
 2.9 2.1 3.8 4.4]
Valores únicos na coluna 'slope': [0 2 1]
Valores únicos na coluna 'ca': [0 2 1 3 4]
Valores únicos na coluna 'thal': [1 2 3 0]
Valores únicos na coluna 'target': [1 0]
```

In [231…
```
# Contando a quantidade de ocorrências de cada valor na coluna 'thal'
thal_counts = df2['thal'].value_counts()
```

```
# Exibindo os resultados
print(thal_counts)
```

```
2    166
3    117
1     18
0      2
Name: thal, dtype: int64
```

In [232…

```
# Contando a quantidade de ocorrências de cada valor na coluna 'ca'
thal_counts = df2['ca'].value_counts()

# Exibindo os resultados
print(thal_counts)
```

```
0    175
1     65
2     38
3     20
4      5
Name: ca, dtype: int64
```

In [233…

```
# Removendo linhas com valores de 'ca' fora do intervalo [0, 3] e valores de 'thal' iguais a 0
df2 = df2.query('0 <= ca <= 3 and thal != 0')

# Verificando o DataFrame após a remoção das linhas
for column in df2.columns:
    unique_values = df2[column].unique()
    print(f"Valores únicos na coluna '{column}': {unique_values}")
```

```
Valores únicos na coluna 'age': [63 37 41 56 57 44 52 54 48 49 64 58 50 66 43 69 59 42 61 40 71 51 65 53
 46 45 39 47 62 34 35 29 55 60 67 68 74 76 70 77 38]
Valores únicos na coluna 'sex': [1 0]
Valores únicos na coluna 'cp': [3 2 1 0]
Valores únicos na coluna 'trestbps': [145 130 120 140 172 150 110 135 160 105 125 142 155 104 138 108 134 122
 115 118 128 100 124  94 112 102 152 101 132 148 178 129 180 136 126 106
 156 170 146 117 200 165 174 192 144 123 154 114 164]
Valores únicos na coluna 'chol': [233 250 204 236 354 192 294 263 199 168 239 275 266 211 283 219 340 226
 247 234 243 302 212 175 417 197 198 177 273 213 304 232 269 360 308 245
 208 264 321 325 235 257 256 231 141 252 201 222 260 182 303 265 309 186
 203 183 220 209 258 227 261 221 205 240 318 298 564 277 214 248 255 207
 288 160 394 315 246 244 270 195 196 254 126 313 262 215 193 271 268 267
 210 295 306 178 223 242 180 228 149 278 253 342 157 286 229 284 224 206
 167 230 335 276 353 225 330 290 172 305 216 188 282 185 326 274 164 307
 249 341 407 217 174 281 289 322 299 300 293 184 409 259 200 327 237 218
 319 166 311 169 187 176 241 131]
Valores únicos na coluna 'fbs': [1 0]
Valores únicos na coluna 'restecg': [0 1 2]
Valores únicos na coluna 'thalach': [150 187 172 178 163 148 153 173 162 174 160 139 171 144 158 114 151 161
 179 137 157 123 152 168 140 188 125 170 165 142 180 143 182 156 149 146
 175 186 185 159 130 190 132 147 154 202 166 164 184 122 138 111 145 194
 115 131 133 155 167 192 169 121  96 126 105 181 116 108 129 120 112 128
 109 113  99 177 141 136  97 127 103 124  88 195 106  95 117  71 118 134
  90]
Valores únicos na coluna 'exang': [0 1]
Valores únicos na coluna 'oldpeak': [2.3 3.5 1.4 0.8 0.6 0.4 1.3 0.  0.5 1.6 1.2 0.2 1.8 1.  2.6 1.5 3.  2.4
 0.1 1.9 4.2 1.1 2.  0.7 0.3 0.9 3.6 3.1 3.2 2.5 2.2 2.8 3.4 6.2 4.  5.6
 2.9 2.1 3.8 4.4]
Valores únicos na coluna 'slope': [0 2 1]
Valores únicos na coluna 'ca': [0 2 1 3]
Valores únicos na coluna 'thal': [1 2 3]
Valores únicos na coluna 'target': [1 0]
```

Dados relacionados a um estudo sobre doenças cardíacas. Cada amostra possui informações sobre um indivíduo e suas características médicas.

- **age**: Atributo numérico que representa a idade do paciente em anos.
- **sex**: Atributo categórico binário que indica o sexo do paciente (0 ou 1).
- **cp**: Atributo categórico que indica o tipo de dor no peito (valores de 0 a 3).
- **trestbps**: Atributo numérico que indica a pressão arterial em repouso do paciente em mm Hg.
- **chol**: Atributo numérico que representa o nível de colesterol sérico do paciente em mg/dl.
- **fbs**: Atributo categórico binário que indica se o paciente tem açúcar no sangue em jejum maior que 120 mg/dl (1) ou não (0).
- **restecg**: Atributo categórico que indica os resultados eletrocardiográficos em repouso (valores de 0 a 2).
- **thalach**: Atributo numérico que representa a frequência cardíaca máxima alcançada pelo paciente em bpm.
- **exang**: Atributo categórico binário que indica se o paciente teve angina induzida por exercício (1) ou não (0).
- **oldpeak**: Atributo numérico que indica a depressão do segmento ST induzida pelo exercício em relação ao repouso.
- **slope**: Atributo categórico que indica a inclinação do segmento ST no pico do exercício (valores de 0 a 2).
- **ca**: Atributo numérico que indica o número de vasos sanguíneos principais coloridos por fluoroscopia (valores de 0 a 4).
- **thal**: Atributo categórico que indica a presença de um defeito cardíaco reversível - talassemia (valores de 0 a 3).
- **target**: Atributo categórico binário que indica se o paciente foi diagnosticado com doença cardíaca (1) ou não (0).

2. Aplicar um algoritmo de classificação de sua escolha e calcular a precisão. A variável alvo para ser classificada é “target”

A variável "target" foi escolhida como variável alvo porque ela representa a classificação que queremos prever com base nos atributos disponíveis no conjunto de dados. Neste caso específico, "target" indica a presença (1) ou ausência (0) de uma doença cardíaca em um paciente.

Algoritmo de classificação escolhido: Random Forest

In [234…
```
# Separando atributos e rótulo
X = df2.drop('target', axis=1)
y = df2['target']  # variável alvo
```

In [235…
```
accuracy_before = train_and_evaluate_scaled(X, y)
print(f"Acurácia antes das transformações: {accuracy_before:.4f}")
```

```
Acurácia antes das transformações: 0.8500
```

3. Realizar transformações (arredondamento, tornar binário, binning) em atributos numéricos que aparentemente podem se beneficiar dessas transformações

In [236…
```
# 1 indica um oldpeak maior ou igual a 1 e 0 indica um oldpeak menor que 1.
df2['oldpeak'] = (df2['oldpeak'] >= 1).astype(int)

# Dividindo a coluna 'age' (idade) em intervalos (bins) baseados em quantis.
df2['age'] = pd.qcut(df2['age'], 4, labels=False)

X = df2.drop('target', axis=1) # Atributos transformados
```

4. Aplicar novamente o algoritmo de classificação e comparar os resultados

In [237…
```
# Realizando a divisão dos dados novamente para avaliar o impacto das transformações nos atributos numéricos no desempenho do mod
# Aplicando o Random Forest novamente
# Calculando a acurácia após as transformações

accuracy_after = train_and_evaluate_scaled(X, y)
print(f"Acurácia após transformação: {accuracy_after:.4f}")

# Comparando os resultados
print(f"Diferença na acurácia: {accuracy_after - accuracy_before}")
```

```
Acurácia após transformação: 0.8500
Diferença na acurácia: 0.0
```

5. Repetir os passos 3 e 4 para diferentes atributos e formas de transformação

In [238…
```
# Normalizando a coluna 'thalach' (frequência cardíaca máxima alcançada) usando a técnica Min-Max para que seus valores estejam n
min_thalach = df2['thalach'].min()
max_thalach = df2['thalach'].max()
df2['thalach'] = (df2['thalach'] - min_thalach) / (max_thalach - min_thalach)

# Transformação logarítmica na coluna 'chol' para reduzir o impacto de valores extremos de colesterol
df2['chol'] = np.log(df2['chol'])

# Transformando a coluna 'trestbps' para um valor binário, onde 1 indica uma pressão arterial igual ou maior que 140, e 0 indica
df2['trestbps'] = (df2['trestbps'] >= 140).astype(int)

X = df2.drop('target', axis=1)
```

In [239…
```
# Calculando a acurácia após as transformações
new_accuracy_after = train_and_evaluate_scaled(X, y)
print(f"Acurácia após transformação: {new_accuracy_after:.4f}")

# Comparando os resultados
print(f"Diferença na acurácia: {new_accuracy_after - accuracy_after}")
```

```
Acurácia após transformação: 0.8833
Diferença na acurácia: 0.033333333333333326
```