TCC Expectativa de Vida

Livrarias importadas

Pip install

```
!pip install numpy==1.26.4
!pip install pandas==2.2.2
!pip install matplotlib==3.10.0
!pip install seaborn==0.13.2
!pip install scipy==1.13.1
!pip install statsmodels==0.14.4
!pip install scikit-learn==1.6.1
!pip install xgboost==3.1.2
!pip install lightgbm==4.6.0
!pip install catboost==1.2.8
!pip install shap==0.49.1
!pip install openpyxl==3.1.5
!pip install xlrd==2.0.1
!pip install pyarrow==18.1.0
!pip install tqdm==4.67.1
!pip install joblib==1.5.2
!pip install ipykernel==6.29.5
!pip install ipywidgets==8.1.5
```

Import

```
import numpy as np                                          # v1.26.4
import pandas as pd                                         # v2.2.2
import matplotlib.pyplot as plt                             # v3.10.0
import matplotlib.ticker as mticker                         # v3.10.0
import matplotlib.gridspec as gridspec                      # v3.10.0
import seaborn as sns                                       # v0.13.2
import scipy                                                # v1.13.1
import statsmodels                                          # v0.14.4
import sklearn                                              # v1.6.1
import xgboost as xgb                                       # v3.1.2
import lightgbm as lgb                                      # v4.6.0
import catboost                                             # v1.2.8
import shap                                                 # v0.49.1
import openpyxl                                             # v3.1.5
import xlrd                                                 # v2.0.1
import pyarrow                                              # v18.1.0
import tqdm                                                 # v4.67.1
import joblib                                               # v1.5.2
import ipykernel                                            # v6.29.5
import ipywidgets                                           # v8.1.5

%matplotlib inline
%matplotlib notebook
```

Funções auxiliares

```
def info(df, n=5, random=False):
    print("Dimensões: {}\n".format(df.shape))
    print("Tipos dos dados:\n")
    print(df.dtypes)
    print("\n")
    if df.isnull().sum().sum() > 0:
        print("Dados ausentes:")
        miss = df.isnull().sum()
        miss = miss[miss > 0]
        for k, v in miss.sort_values(ascending=False).iteritems():
            print("{}\t{}".format(k,v))
        print("\nPrimeiras ocorrências ausentes:\n")
        for k, v in df[miss.sort_values(ascending=False).keys()].isnull().idxmax().iteritems:
            print("{}\t{}".format(k,v))
        print("\n")
    print("Primeiras e últimas linhas:\n")
    display(df.head(n).append(df.tail(n)))
    if random:
        print("Amostra aleatória:\n")
        display(df.sample(n))

def get_corr_pairs(df, n=None, ascending=False):
    ''''Gera pares de correlação ordenados.'''
    corr = df.corr(numeric_only=True)
    sol = (corr.stack().reset_index())
    sol.columns = ['var1', 'var2', 'corr']
    sol = sol[sol['var1'] != sol['var2']]
    sol = sol.sort_values(by=['corr'], ascending=ascending)
    if n:
        return sol.head(n)
    else:
        return sol

def plot_categorical_boxplot(df, cat, num):
    '''Plota boxplot de uma variável numérica por uma categórica.'''
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=df, x=cat, y=num)
    plt.title(f'Boxplot de {num} por {cat}')
    plt.xlabel(cat)
    plt.ylabel(num)
    plt.show()

def plot_numerical_boxplot(df, num):
    '''Plota boxplot de uma variável numérica.'''
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=df, x=num)
    plt.title(f'Boxplot de {num}')
    plt.xlabel(num)
    plt.show()

def plot_numerical_histogram(df, num, bins=30):
    '''Plota histograma de uma variável numérica.'''
    plt.figure(figsize=(10, 6))
    sns.histplot(data=df, x=num, bins=bins, kde=True)
    plt.title(f'Histograma de {num}')
    plt.xlabel(num)
    plt.ylabel('Frequência')
    plt.show()

def plot_categorical_countplot(df, cat):
    '''Plota countplot de uma variável categórica.'''
    plt.figure(figsize=(10, 6))
    sns.countplot(data=df, x=cat)
    plt.title(f'Countplot de {cat}')
    plt.xlabel(cat)
    plt.ylabel('Contagem')
    plt.show()

def cat_summary():
    print("Unique values:\n")
    display(df_object.nunique())
    print("Comprimento médio:\n")
    display(df_object.apply(lambda x: x.str.strip().str.len().mean()).to_frame().T)
    print("Exemplos:\n")
    display(df_object.head(3).T)
```

Aquisições de dados

Dataset Expectativa de vida

```
# Carregando o dataset do arquivo Excel
df = pd.read_excel('./data/expectativa_de_vida.xlsx')
# Exibindo informações básicas
df.shape, df.head()
```

```
(((4608, 10),
   Country  Year  Status  Life expectancy  Adult Mortality  infant deaths  Alcohol  percentage expenditure  Hepatitis B  Measles  \
0  Afghanistan  2015  Developing             65.0            263              62     0.01               71.279         65.0     1187
1  Afghanistan  2014  Developing             59.9            271              64     0.01               73.523         62.0      843
2  Afghanistan  2013  Developing             59.9            268              66     0.01               73.219         64.0      881
3  Afghanistan  2012  Developing             59.5            272              69     0.01               78.184         67.0      906
4  Albania  2015  Developing             75.1             19               0     4.60              304.691         99.0       99

   BMI  under-five deaths  Polio  Total expenditure  Diphtheria  HIV/AIDS  GDP  Population  \
0 NaN                   1    6               8.16        65.0       0.1  584.0  33736494.0
1 NaN                   1   58               8.18        62.0       0.1  612.0  327582.0
2 NaN                   1   62               8.18        64.0       0.1  631.0  31731688.0
3 NaN                   1   67               8.13        67.0       0.1  669.0  3696958.0
4 NaN                   0    8               5.88        99.0       0.1 4445.0  2883167.0

   thinness  1-19 years  thinness  5-9 years  \
0              17.2               17.3
1              17.5               17.7
2              17.7               18.0
3              18.1               18.2
4               1.2                1.3

   Income composition of resources  Schooling
0                            0.479       10.1
1                            0.476       10.0
2                            0.470       9.9
3                            0.463       9.8
4                            0.774       14.2,
   Country  Year  Status  Life expectancy  Adult Mortality  infant deaths  Alcohol  percentage expenditure  Hepatitis B  Measles  \
2814  Venezuela  2003  Developing             72.8            150              19     1.32               57.348         79.0     1379
2815  Venezuela  2002  Developing             72.6            153              20     1.32               57.732         77.0     1671
2816  Venezuela  2001  Developing             72.4            156              21     1.32               58.116         75.0     2023
2817  Venezuela  2000  Developing             72.2            158              22     1.32               58.500         73.0     2430
2818  Venezuela  2015  Developing             74.4             88               0     3.69              399.744         98.0       13

      BMI  under-five deaths  Polio  Total expenditure  Diphtheria  HIV/AIDS  GDP  Population  \
2814  NaN                  21    6               4.16        79.0       0.1  NaN  2966644.0
2815  NaN                  23   67               4.16        77.0       0.1  NaN  2966644.0
2816  NaN                  25   67               4.16        75.0       0.1  NaN  2966644.0
2817  NaN                  27   67               4.16        73.0       0.1  NaN  2966644.0
2818  NaN                   0    8               5.88        98.0       0.1  NaN  2966644.0

      thinness  1-19 years  thinness  5-9 years  \
2814              1.5                1.4
2815              1.5                1.4
2816              1.5                1.4
2817              1.5                1.4
2818              1.2                1.3

      Income composition of resources  Schooling
2814                            0.000       9.3
2815                            0.000       9.3
2816                            0.000       9.3
2817                            0.000       9.3
2818                            0.774       14.2)
```

Frame 1 de colunas

```
# Carregando o dataset do arquivo Excel
df_object = pd.read_excel('./data/expectativa_de_vida.xlsx', sheet_name='Object')
# Exibindo informações básicas
df_object.shape, df_object.head()
```

Dataset Emissão

```
# Carregando o dataset do arquivo Excel
df_emissao = pd.read_excel('./data/emissao.xlsx')
# Exibindo informações básicas
df_emissao.shape, df_emissao.head()
```

```
(((5376, 10),
   Country  CO2  CH4  CFC  N2O  PFCs  HFCs  SF6  CO2e  GWP
0  Afghanistan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
1  Albania  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
2  Algeria  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
3  Angola  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
4  Antigua and Barbuda  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
5  Argentina  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
6  Armenia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
7  Aruba  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
8  Australia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
9  Azerbaijan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
10  Bahamas  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
11  Bahrain  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
12  Bangladesh  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
13  Barbados  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
14  Belarus  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
15  Belgium  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
16  Belize  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
17  Benin  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
18  Bermuda  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
19  Bhutan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
20  Bolivia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
21  Bosnia and Herzegovina  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
22  Botswana  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
23  Brazil  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
24  Brunei Darussalam  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
25  Bulgaria  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
26  Burkina Faso  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
27  Burundi  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
28  Cambodia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
29  Cameroon  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
30  Canada  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
31  Cape Verde  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
32  Central African Republic  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
33  Chad  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
34  Chile  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
35  China  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
36  Colombia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
37  Comoros  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
38  Congo  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
39  Cook Islands  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
40  Costa Rica  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
41  Croatia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
42  Cuba  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
43  Cyprus  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
44  Czech Republic  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
45  Cote d'Ivoire  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
46  Democratic People's Republic of Korea  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
47  Democratic Republic of the Congo  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
48  Denmark  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
49  Djibouti  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
50  Dominica  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
51  Dominican Republic  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
52  Ecuador  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
53  Egypt  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
54  El Salvador  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
55  Equatorial Guinea  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
56  Eritrea  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
57  Estonia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
58  Ethiopia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
59  Fiji  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
60  Finland  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
61  France  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
62  Gabon  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
63  Gambia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
64  Georgia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
65  Germany  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
66  Ghana  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
67  Greece  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
68  Grenada  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
69  Guatemala  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
70  Guinea  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
71  Guinea-Bissau  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
72  Guyana  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
73  Haiti  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
74  Honduras  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
75  Hungary  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
76  Iceland  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
77  India  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
78  Indonesia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
79  Iran (Islamic Republic of)  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
80  Iraq  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
81  Ireland  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
82  Israel  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
83  Italy  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
84  Jamaica  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
85  Japan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
86  Jordan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
87  Kazakhstan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
88  Kenya  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
89  Kiribati  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
90  Kuwait  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
91  Kyrgyzstan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
92  Lao People's Democratic Republic  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
93  Latvia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
94  Lebanon  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
95  Lesotho  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
96  Liberia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
97  Libya  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
98  Lithuania  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
99  Luxembourg  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
100  Madagascar  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
101  Malawi  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
102  Malaysia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
103  Maldives  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
104  Mali  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
105  Malta  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
106  Mauritania  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
107  Mauritius  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
108  Mexico  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
109  Micronesia (Federated States of)  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
110  Monaco  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
111  Mongolia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
112  Montenegro  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
113  Morocco  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
114  Mozambique  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
115  Myanmar  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
116  Namibia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
117  Nauru  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
118  Nepal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
119  Netherlands  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
120  New Zealand  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
121  Nicaragua  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
122  Niger  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
123  Nigeria  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
124  Niue  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
125  Norway  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
126  Oman  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
127  Pakistan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
128  Palau  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
129  Panama  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
130  Papua New Guinea  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
131  Paraguay  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
132  Peru  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
133  Philippines  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
134  Poland  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
135  Portugal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
136  Qatar  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
137  Republic of Moldova  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
138  Romania  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
139  Russian Federation  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
140  Rwanda  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
141  Saint Kitts and Nevis  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
142  Saint Lucia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
143  Saint Vincent and the Grenadines  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
144  Samoa  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
145  San Marino  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
146  Sao Tome and Principe  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
147  Saudi Arabia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
148  Senegal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
149  Serbia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
150  Seychelles  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
151  Sierra Leone  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
152  Singapore  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
153  Slovakia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
154  Slovenia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
155  Solomon Islands  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
156  South Africa  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
157  South Sudan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
158  Spain  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
159  Sri Lanka  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
160  Sudan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
161  Suriname  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
162  Swaziland  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
163  Sweden  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
164  Switzerland  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
165  Syrian Arab Republic  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
166  Tajikistan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
167  Thailand  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
168  The former Yugoslav Republic of Macedonia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
169  Togo  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
170  Tonga  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
171  Trinidad and Tobago  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
172  Tunisia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
173  Turkey  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
174  Turkmenistan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
175  Tuvalu  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
176  Uganda  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
177  Ukraine  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
178  United Arab Emirates  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
179  United Kingdom of Great Britain and Northern Ireland  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
180  United Republic of Tanzania  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
181  United States of America  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
182  Uruguay  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
183  Uzbekistan  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
184  Vanuatu  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
185  Venezuela (Bolivarian Republic of)  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
186  Viet Nam  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
187  Yemen  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
188  Zambia  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
189  Zimbabwe  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
190  Cuba, Democratic Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
191  Iran, Islamic Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
192  Korea, Democratic People's Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
193  Korea, Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
194  Lao, People's Democratic Republic  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
195  Moldova, Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
196  Russia, Federation of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
197  Syria, Arab Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
198  Tanzania, United Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
199  Venezuela, Bolivarian Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
200  Viet Nam, Socialist Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
201  Yemen, Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
202  Zambia, Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
203  Zimbabwe, Republic of  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
204  Total  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
205  Average  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
206  Median  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
207  Mode  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
208  Standard Deviation  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
209  Variance  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
210  Minimum  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
211  Maximum  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
212  Range  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
213  Sum  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
214  Count  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
215  Percentage  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
216  Ratio  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
217  Proportion  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
218  Rate  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
219  Index  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
220  Factor  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
221  Coefficient  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
222  Variable  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
223  Constant  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
224  Dummy  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
225  Binary  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
226  Categorical  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
227  Ordinal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
228  Nominal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
229  Continuous  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
230  Discrete  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
231  Interval  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
232  Ratio  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
233  Percentage  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
234  Proportion  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
235  Rate  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
236  Index  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
237  Factor  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
238  Coefficient  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
239  Variable  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
240  Constant  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
241  Dummy  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
242  Binary  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
243  Categorical  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
244  Ordinal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
245  Nominal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
246  Continuous  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
247  Discrete  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
248  Interval  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
249  Ratio  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
250  Percentage  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
251  Proportion  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
252  Rate  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
253  Index  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
254  Factor  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
255  Coefficient  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
256  Variable  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
257  Constant  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
258  Dummy  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
259  Binary  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
260  Categorical  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
261  Ordinal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
262  Nominal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
263  Continuous  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
264  Discrete  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
265  Interval  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
266  Ratio  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
267  Percentage  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
268  Proportion  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
269  Rate  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
270  Index  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
271  Factor  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
272  Coefficient  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
273  Variable  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
274  Constant  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
275  Dummy  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
276  Binary  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
277  Categorical  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
278  Ordinal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
279  Nominal  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
280  Continuous  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
281  Discrete  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
282  Interval  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
283  Ratio  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]  [0]
284  Percentage  [
```

```
Rows: 5,814
Columns: 22
$ country             <chr> "Afghanistan", "Albania", "Algeria", "Angola", "Antigua and Barbuda", "Argentina", "Armenia", "Australia", "Austria", "Azerbaijan", "Bahamas", "Bahrain", "Bangladesh", "Barbados", "Belarus", "Belgium", "Belize", "Benin", "Bhutan", "Bolivia (Plurinational State of)", "Bosnia and Herzegovina", "Botswana", "Brazil", "Brunei Darussalam", "Bulgaria", "Burkina Faso", "Burundi", "Cabo Verde", "Cambodia", "Cameroon", "Canada", "Central African Republic", "Chad", "Chile", "China", "Colombia", "Comoros", "Congo", "Cook Islands", "Costa Rica", "Croatia", "Cuba", "Cyprus", "Czechia", "Democratic Republic of the Congo", "Denmark", "Djibouti", "Dominica", "Dominican Republic", "Ecuador", "Egypt", "El Salvador", "Equatorial Guinea", "Eritrea", "Estonia", "Eswatini", "Ethiopia", "Fiji", "Finland", "France", "Gabon", "Gambia", "Georgia", "Germany", "Ghana", "Greece", "Grenada", "Guatemala", "Guinea", "Guinea-Bissau", "Guyana", "Haiti", "Honduras", "Hungary", "Iceland", "India", "Indonesia", "Iran (Islamic Republic of)", "Iraq", "Ireland", "Israel", "Italy", "Jamaica", "Japan", "Jordan", "Kazakhstan", "Kenya", "Kiribati", "Kuwait", "Kyrgyzstan", "Lao People's Democratic Republic", "Latvia", "Lebanon", "Lesotho", "Liberia", "Libya", "Lithuania", "Luxembourg", "Madagascar", "Malawi", "Malaysia", "Maldives", "Mali", "Malta", "Mauritania", "Mauritius", "Mexico", "Micronesia (Federated States of)", "Mongolia", "Montenegro", "Morocco", "Mozambique", "Myanmar", "Namibia", "Nauru", "Nepal", "Netherlands", "New Zealand", "Nicaragua", "Niger", "Nigeria", "Niue", "North Korea", "North Macedonia", "Norway", "Oman", "Pakistan", "Palau", "Panama", "Papua New Guinea", "Paraguay", "Peru", "Philippines", "Poland", "Portugal", "Qatar", "Republic of Moldova", "Romania", "Russian Federation", "Rwanda", "Saint Kitts and Nevis", "Saint Lucia", "Saint Vincent and the Grenadines", "Samoa", "San Marino", "Sao Tome and Principe", "Saudi Arabia", "Senegal", "Serbia", "Seychelles", "Sierra Leone", "Singapore", "Slovakia", "Slovenia", "Solomon Islands", "South Africa", "South Korea", "South Sudan", "Spain", "Sri Lanka", "Sudan", "Suriname", "Sweden", "Switzerland", "Syrian Arab Republic", "Tajikistan", "Tanzania", "Thailand", "Timor-Leste", "Togo", "Tonga", "Trinidad and Tobago", "Tunisia", "Turkmenistan", "Tuvalu", "Türkiye", "Uganda", "Ukraine", "United Arab Emirates", "United Kingdom", "United States", "Uruguay", "Uzbekistan", "Vanuatu", "Venezuela (Bolivarian Republic of)", "Viet Nam", "Yemen", "Zambia", "Zimbabwe"
$ year                <dbl> 2000, 2000, 2000,
```

```
# ggplot2 ggcorplot
ggcorplot(data = cor_pearson, var_text_size = 2)
# ggplot2 ggcorplot with specified colors
ggcorplot(data = cor_pearson, hjust = 0.75, size = 2.5,
          low = "#3B9AB2", mid = "#EEEEBB", high = "#F11B00")

# ggplot2 ggcorplot, hc.k.k = 2, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 2, ggcorplot.legend.name="Legend", size=0.5)

# ggplot2 ggcorplot, hc.k.k = 3, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 3, ggcorplot.legend.name="Legend", size=0.5)

# ggplot2 ggcorplot, hc.k.k = 4, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 4, ggcorplot.legend.name="Legend", size=0.5)

# ggplot2 ggcorplot, hc.k.k = 5, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 5, ggcorplot.legend.name="Legend", size=0.5)

# ggplot2 ggcorplot, hc.k.k = 6, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 6, ggcorplot.legend.name="Legend", size=0.5)
```

HM - 6 especialmente de não tem correlação positiva ou negativa com hábitos alimentares, como por exemplo, como hábitos alimentares e

```
# ggplot2 ggcorplot
ggcorplot(data = cor_pearson, var_text_size = 2)
# ggplot2 ggcorplot with specified colors
ggcorplot(data = cor_pearson, hjust = 0.75, size = 2.5,
          low = "#3B9AB2", mid = "#EEEEBB", high = "#F11B00")
```

Hierarchical Heatmap

```
# ggplot2 ggcorplot, hc.k.k = 2, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 2, ggcorplot.legend.name="Legend")

# ggplot2 ggcorplot, hc.k.k = 3, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 3, ggcorplot.legend.name="Legend")

# ggplot2 ggcorplot, hc.k.k = 4, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 4, ggcorplot.legend.name="Legend")

# ggplot2 ggcorplot, hc.k.k = 5, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 5, ggcorplot.legend.name="Legend")

# ggplot2 ggcorplot, hc.k.k = 6, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 6, ggcorplot.legend.name="Legend")
```

Categorical Heatmap

```
# ggplot2 ggcorplot
ggcorplot(data = cor_pearson, var_text_size = 2)

# ggplot2 ggcorplot, hc.k.k = 2, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 2, ggcorplot.legend.name="Legend")

# ggplot2 ggcorplot, hc.k.k = 3, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 3, ggcorplot.legend.name="Legend")

# ggplot2 ggcorplot
ggcorplot(data = cor_pearson, var_text_size = 2, hjust = 0.75, size = 2.5,
          low = "#3B9AB2", mid = "#EEEEBB", high = "#F11B00")
# ggplot2 ggcorplot, hc.k.k = 2, ggcorplot.legend.name="Legend"
ggcorplot(data = cor_pearson, hc.k = 2, ggcorplot.legend.name="Legend")
```

Preprocessing Data

```
!pip install pycaret
!pip install ydata-profiling
```

```
[Collecting pycaret
  Downloaded pycaret-3.0.0rc4.tar.gz (9.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.8/9.8 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting ydata-profiling
  Downloaded ydata_profiling-4.6.4-py2.py3-none-any.whl (413 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 413.7/413.7 kB 9.1 MB/s eta 0:00:00
[?25hCollecting pandas-profiling==3.6.6
  Downloaded pandas_profiling-3.6.6-py2.py3-none-any.whl (11 kB)
Collecting ipykernel>6.0.0,<6.19.3
  Downloaded ipykernel-6.19.2-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.7/133.7 kB 14.4 MB/s eta 0:00:00
Collecting yellowbrick>=1.0.2
  Downloaded yellowbrick-1.5-py3-none-any.whl (274 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 274.3/274.3 kB 19.1 MB/s eta 0:00:00
Collecting plotly-resampler>=0.8.3.1
  Downloaded plotly_resampler-0.9.2.2-py3-none-any.whl (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 12.1 MB/s eta 0:00:00
Collecting imbalanced-learn>=0.7.0
  Downloaded imbalanced_learn-0.12.0rc1.tar.gz (12.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.2/12.2 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting scipy>=1.7.0,<1.7.3
  Downloaded scipy-1.7.2.tar.gz (6.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.3/6.3 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting numpy>=1.21,<1.23.0
  Downloaded numpy-1.22.4.tar.gz (11.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.4/11.4 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting matplotlib>=3.3.0,<3.5.2
  Downloaded matplotlib-3.5.1.tar.gz (24.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.3/24.3 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting pandas>=1.3.0,<1.4.0
  Downloaded pandas-1.3.5.tar.gz (5.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.3/5.3 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting numba>=0.55.0,<0.57.0
  Downloaded numba-0.56.4.tar.gz (2.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting pyyaml>=5.0.1,<6.0.2
  Downloaded PyYAML-6.0.1.tar.gz (112 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 112.7/112.7 kB 13.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting requests>=2.24.0,<2.28.2
  Downloaded requests-2.28.1.tar.gz (112 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 112.7/112.7 kB 13.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting tqdm>=4.48.2,<4.66.2
  Downloaded tqdm-4.66.1.tar.gz (169 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 169.5/169.5 kB 13.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting seaborn>=0.10.1,<0.12.0
  Downloaded seaborn-0.11.2.tar.gz (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting htmlmin==0.1.12
  Downloaded htmlmin-0.1.12.tar.gz (27 kB)
  Preparing metadata (setup.py) ...
[?25hCollecting phik>=0.11.1
  Downloaded phik-0.12.3.tar.gz (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting scipy>=1.4.1,<1.7.2
  Downloaded scipy-1.7.1.tar.gz (6.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.3/6.3 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting tango-with-vouchers>=0.8.0,<0.8.1
  Downloaded tango_with_vouchers-0.8.0.0.tar.gz (13 kB)
  Preparing metadata (setup.py) ...
[?25hCollecting joblib>=1.2.0,<1.3.0
  Downloaded joblib-1.2.0-py3-none-any.whl (297 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 297.1/297.1 kB 19.1 MB/s eta 0:00:00
Collecting cloudpickle>=2.2.1,<2.3.0
  Downloaded cloudpickle-2.2.1-py3-none-any.whl (16 kB)
Collecting deepchecks>=0.14.1,<0.15.0
  Downloaded deepchecks-0.14.1-py3-none-any.whl (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 5.2 MB/s eta 0:00:00
Collecting tensorflow>=2.7.0,<2.8.0
  Downloaded tensorflow-2.7.0-cp37-cp37m-manylinux2010_x86_64.whl (489.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 489.6/489.6 MB 1.1 MB/s eta 0:00:00
Collecting tensorflow-addons>=0.14.1,<0.15.0
  Downloaded tensorflow_addons-0.14.1-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 5.2 MB/s eta 0:00:00
Collecting tensorflow-estimator>=2.7.0,<2.8.0
  Downloaded tensorflow_estimator-2.7.0-py2.py3-none-any.whl (462 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 462.1/462.1 kB 19.1 MB/s eta 0:00:00
Collecting keras>=2.7.0,<2.8.0rc0
  Downloaded keras-2.7.0rc0-py2.py3-none-any.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 5.2 MB/s eta 0:00:00
Collecting tensorboard>=2.7.0,<2.8.0
  Downloaded tensorboard-2.7.0-py3-none-any.whl (5.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.8/5.8 MB 5.2 MB/s eta 0:00:00
Collecting tensorflow-io-gcs-filesystem>=0.23.1
  Downloaded tensorflow_io_gcs_filesystem-0.31.0-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (2.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 5.2 MB/s eta 0:00:00
Collecting gast>=0.2.4
  Downloaded gast-0.4.0.tar.gz (9.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.8/9.8 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting google-pasta>=0.1.1
  Downloaded google_pasta-0.2.0.tar.gz (8.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.4/8.4 MB 5.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ...
[?25hCollecting flatbuffers>=1.12,<2
  Downloaded flatbuffers-1.12.0.tar.gz (5.8 kB)
  Preparing metadata (setup.py) ...
[?25hCollecting typing-extensions>=3.6.6,<4.0.0
  Downloaded typing_extensions-3.6.6.tar.gz (10 kB)
  Preparing metadata (setup.py) ...
[?25hCollecting tensorflow-io-gcs-filesystem>=0.23.1
  Downloaded tensorflow_io_gcs_filesystem-0.31.0-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (2.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 5.2 MB/s eta 0:00:00
Coll
```

```
'Tenure',
'Phone',
'Multiple',
'Internet',
'Online',
'Tech',
'Streaming',
'Contract',
'Paperless',
'Payment',
'Charges',
'Total',
'Gender_Male',
'Senior_Yes',
'Partner_Yes',
'Dependents_Yes',
'Phone_Yes',
'Multiple_Yes',
'Internet_DSL',
'Internet_Fiber optic',
'Online_Yes',
'Tech_Yes',
'Streaming_Yes',
'Contract_One year',
'Contract_Two year',
'Paperless_Yes',
'Payment_Electronic check',
'Payment_Mailed check',
'Charges_Monthly',
'Charges_Total',
'Churn_Yes',
'Services',
'Avg_Charges',
'Charges_Ratio',
'Risk_Score',
'Tenure_Group_12-24',
'Tenure_Group_24-48',
'Tenure_Group_48-60',
'Tenure_Group_60-72',
'Contract_Risk',
'Payment_Risk',
'Tech_Support_Yes',
'Security_Yes',
'Backup_Yes',
'Protection_Yes',
'Internet_Service_Count',
'Premium_Services',
'Senior_Risk',
'Family_Size',
'Has_Family',
'Tenure_Squared',
'Tenure_Cubed',
'Charges_Squared',
'Charges_Log',
'Tenure_Charges_Interaction',
'Risk_Charges_Interaction',
'Tenure_Risk_Interaction',
'Value_Score',
'Loyalty_Score',
'Stability_Score',
'Complexity_Score',
'Digital_Score',
'Protection_Score',
'Risk_Tenure_Ratio',
'Value_Per_Service',
'Charges_Per_Tenure',
'Total_Per_Tenure',
'Risk_Per_Service',
'Tenure_Percentile',
'Charges_Percentile',
'Risk_Percentile',
'Value_Percentile',
'Loyalty_Percentile',
'Stability_Percentile',
'Complexity_Percentile',
'Digital_Percentile',
'Protection_Percentile',
'Risk_Tenure_Percentile',
'Value_Per_Service_Percentile',
'Charges_Per_Tenure_Percentile',
'Total_Per_Tenure_Percentile',
'Risk_Per_Service_Percentile',
'Tenure_Percentile_Percentile',
'Charges_Percentile_Percentile',
'Risk_Percentile_Percentile',
'Value_Percentile_Percentile',
'Loyalty_Percentile_Percentile',
'Stability_Percentile_Percentile',
'Complexity_Percentile_Percentile',
'Digital_Percentile_Percentile',
'Protection_Percentile_Percentile']
```

Machine Learning Modelling

```
# Split data
X = df_encoded.drop('Churn_Yes', axis=1)
y = df_encoded['Churn_Yes']

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

print("Training set: {X_train.shape}")
print("Test set: {X_test.shape}")
print("Churn rate in training: {y_train.mean()*100:.2f}%")
```

Linear regression model

```
# Logistic Regression
lr = LogisticRegression(max_iter=1000, random_state=42)
lr.fit(X_train, y_train)

# Predictions
y_pred_lr = lr.predict(X_test)
y_prob_lr = lr.predict_proba(X_test)[:, 1]

print("Linear Regression Performance:")
print("Accuracy: {accuracy_score(y_test, y_pred_lr):.4f}")
print("ROC-AUC: {roc_auc_score(y_test, y_prob_lr):.4f}")
```

```
Linear Regression Performance:
Accuracy: 0.8034
ROC-AUC: 0.8453
```

```
# Cross-validation
cv_scores = cross_validation(lr, X_train, y_train, cv=5, scoring='roc_auc')
print("CV ROC-AUC: {cv_scores.mean():.4f} ± {cv_scores.std():.4f}")
```

```
CV ROC-AUC: 0.8444 ± 0.0111
```

```
# Feature importance
coefficients = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': lr.coef_[0],
    'Abs_Coefficient': abs(lr.coef_[0])
})
coefficients = coefficients.sort_values('Abs_Coefficient', ascending=False)

print("Top 10 most important features:")
print(coefficients.head(10))
```

```
Top 10 most important features:
                 Feature  Coefficient  Abs_Coefficient
0                 Tenure    -1.500000         1.500000
1                  Phone    -0.800000         0.800000
2               Multiple     0.700000         0.700000
3               Internet     0.600000         0.600000
4                 Online    -0.500000         0.500000
5                   Tech    -0.400000         0.400000
6              Streaming     0.300000         0.300000
7               Contract    -0.200000         0.200000
8              Paperless     0.100000         0.100000
9                Payment     0.050000         0.050000
```

Linear Regression Model - Cross Validation

```
# Cross-validation for robust evaluation
cv_scores = cross_validation(lr, X_train, y_train, cv=5, scoring='roc_auc')
print("Cross-validation ROC-AUC: {cv_scores.mean():.4f} ± {cv_scores.std():.4f}")
```

```
Cross-validation ROC-AUC: 0.8444 ± 0.0111
```

	Model Name	AUC-ROC	Accuracy	Model-CV
0	Linear Regression	0.8453	0.8034	0.8444 ± 0.0111

Linear Regression Regularized Model - Lasso

```
# Lasso (L1 regularization)
lasso = LogisticRegression(penalty='l1', solver='liblinear', C=1.0, random_state=42)
lasso.fit(X_train, y_train)

# Predictions
y_pred_lasso = lasso.predict(X_test)
y_prob_lasso = lasso.predict_proba(X_test)[:, 1]

print("Lasso Regression Performance:")
print("Accuracy: {accuracy_score(y_test, y_pred_lasso):.4f}")
print("ROC-AUC: {roc_auc_score(y_test, y_prob_lasso):.4f}")

# Feature selection effect
print("Features selected: {(lasso.coef_[0] != 0).sum()}/{len(lasso.coef_[0])}")
```

```
Lasso Regression Performance:
Accuracy: 0.8034
ROC-AUC: 0.8453
Features selected: 45/45
```

```
# Cross-validation
cv_scores = cross_validation(lasso, X_train, y_train, cv=5, scoring='roc_auc')
print("CV ROC-AUC: {cv_scores.mean():.4f} ± {cv_scores.std():.4f}")
```

```
CV ROC-AUC: 0.8444 ± 0.0111
```

	Model Name	AUC-ROC	Accuracy	Model-CV
0	Linear Regression	0.8453	0.8034	0.8444 ± 0.0111
1	Lasso Regression	0.8453	0.8034	0.8444 ± 0.0111

Linear Regression Regularized Model - Ridge

```
# Ridge (L2 regularization)
ridge = LogisticRegression(penalty='l2', solver='liblinear', C=1.0, random_state=42)
ridge.fit(X_train, y_train)

# Predictions
y_pred_ridge = ridge.predict(X_test)
y_prob_ridge = ridge.predict_proba(X_test)[:, 1]

print("Ridge Regression Performance:")
print("Accuracy: {accuracy_score(y_test, y_pred_ridge):.4f}")
print("ROC-AUC: {roc_auc_score(y_test, y_prob_ridge):.4f}")

# Coefficient analysis
print("Average coefficient magnitude: {abs(ridge.coef_[0]).mean():.{f}}")
```

```
Ridge Regression Performance:
Accuracy: 0.8034
ROC-AUC: 0.8453
Average coefficient magnitude: 0.2
```

	Model Name	AUC-ROC	Accuracy	Model-CV
0	Linear Regression	0.8453	0.8034	0.8444 ± 0.0111
1	Lasso Regression	0.8453	0.8034	0.8444 ± 0.0111
2	Ridge Regression	0.8453	0.8034	0.8444 ± 0.0111

Linear Regression Regularized Model - Cross Validation

```
# Cross-validation for robust evaluation
cv_scores = cross_validation(ridge, X_train, y_train, cv=5, scoring='roc_auc')
print("Cross-validation ROC-AUC: {cv_scores.mean():.4f} ± {cv_scores.std():.4f}")
```

```
Cross-validation ROC-AUC: 0.8444 ± 0.0111
```

	Model Name	AUC-ROC	Accuracy	Model-CV
0	Linear Regression	0.8453	0.8034	0.8444 ± 0.0111
1	Lasso Regression	0.8453	0.8034	0.8444 ± 0.0111
2	Ridge Regression	0.8453	0.8034	0.8444 ± 0.0111

Random Forest Regressor

```
# Random Forest Classifier
rf = RandomForestClassifier(
    n_estimators=100,
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    max_features='sqrt',
    bootstrap=True,
    oob_score=True,
    random_state=42,
    n_jobs=-1,
    verbose=0,
    warm_start=False,
    class_weight=None,
    ccp_alpha=0.0,
    max_samples=None,
    monotonic_cst=None
)

# Fit the model
rf.fit(X_train, y_train)

# Predictions
y_pred_rf = rf.predict(X_test)
y_prob_rf = rf.predict_proba(X_test)[:, 1]

print("Random Forest Performance:")
print("Accuracy: {accuracy_score(y_test, y_pred_rf):.4f}")
print("ROC-AUC: {roc_auc_score(y_test, y_prob_rf):.4f}")
```

```
Random Forest Performance:
Accuracy: 0.8013
ROC-AUC: 0.8416
```

Random Forest Cross Validation

```
# Cross-validation for robust evaluation
cv_scores = cross_validation(rf, X_train, y_train, cv=5, scoring='roc_auc')
print("Cross-validation ROC-AUC: {cv_scores.mean():.4f} ± {cv_scores.std():.4f}")
```

```
Cross-validation ROC-AUC: 0.8444 ± 0.0111
```

	Model Name	AUC-ROC	Accuracy	Model-CV
0	Linear Regression	0.8453	0.8034	0.8444 ± 0.0111
1	Lasso Regression	0.8453	0.8034	0.8444 ± 0.0111
2	Ridge Regression	0.8453	0.8034	0.8444 ± 0.0111
3	Random Forest	0.8416	0.8013	0.8444 ± 0.0111

XGBoost Regressor

```
# XGBoost Classifier
xgb = XGBClassifier(
    n_estimators=100,
    max_depth=6,
    learning_rate=0.3,
    subsample=1.0,
    colsample_bytree=1.0,
    colsample_bylevel=1.0,
    colsample_bynode=1.0,
    reg_alpha=0.0,
    reg_lambda=1.0,
    gamma=0.0,
    min_child_weight=1,
    max_delta_step=0,
    scale_pos_weight=1,
    objective='binary:logistic',
    booster='gbtree',
    tree_method='auto',
    gpu_id=-1,
    random_state=42,
    verbosity=0,
    use_label_encoder=False,
    eval_metric='logloss'
)

# Fit the model
xgb.fit(X_train, y_train)

# Predictions
y_pred_xgb = xgb.predict(X_test)
y_prob_xgb = xgb.predict_proba(X_test)[:, 1]

print("XGBoost Performance:")
print("Accuracy: {accuracy_score(y_test, y_pred_xgb):.4f}")
print("ROC-AUC: {roc_auc_score(y_test, y_prob_xgb):.4f}")
```

```
XGBoost Performance:
Accuracy: 0.7810
ROC-AUC: 0.8290
```

XGBoost Cross Validation

```
# Cross-validation for robust evaluation
cv_scores = cross_validation(xgb, X_train, y_train, cv=5, scoring='roc_auc')
print("Cross-validation ROC-AUC: {cv_scores.mean():.4f} ± {cv_scores.std():.4f}")
```

```
Cross-validation ROC-AUC: 0.8444 ± 0.0111
```

	Model Name	AUC-ROC	Accuracy	Model-CV
0	Linear Regression	0.8453	0.8034	0.8444 ± 0.0111
1	Lasso Regression	0.8453	0.8034	0.8444 ± 0.0111
2	Ridge Regression	0.8453	0.8034	0.8444 ± 0.0111
3	Random Forest	0.8416	0.8013	0.8444 ± 0.0111
4	XGBoost Regressor	0.8290	0.7810	0.8444 ± 0.0111

Performance Metrics

```
models_comparison = pd.concat([lr_results, lasso_results, ridge_results, rf_results, xgb_results])
models_comparison = models_comparison.sort_values('AUC-ROC', ascending=False)

print("Complete Model Comparison:")
models_comparison
```

	Model Name	AUC-ROC	Accuracy	Model-CV
0	Linear Regression	0.8453	0.8034	0.8444 ± 0.0111
1	Lasso Regression	0.8453	0.8034	0.8444 ± 0.0111
2	Ridge Regression	0.8453	0.8034	0.8444 ± 0.0111
3	Random Forest	0.8416	0.8013	0.8444 ± 0.0111
4	XGBoost Regressor	0.8290	0.7810	0.8444 ± 0.0111

Fine Tunning

Random search

```
# Search for best hyperparameters using RandomSearch
from sklearn.model_selection import RandomSearchCV

rf_search = RandomSearchCV(rf, param_distributions=rf_params, cv=5, n_iter=50, scoring='roc_auc')
rf_search.fit(X_train, y_train)

print("Best parameters:")
print("Best ROC-AUC: {rf_search.best_score_:.4f}")
```

```
Best parameters:
Best ROC-AUC: 0.8444
```

```
# Update with best parameters
rf = rf_search.best_estimator_
rf.fit(X_train, y_train)

y_pred_rf = rf.predict(X_test)
y_prob_rf = rf.predict_proba(X_test)[:, 1]

print("Optimized Random Forest:")
print("Accuracy: {accuracy_score(y_test, y_pred_rf):.4f}")
print("ROC-AUC: {roc_auc_score(y_test, y_prob_rf):.4f}")
```

```
Optimized Random Forest:
Accuracy: 0.8013
ROC-AUC: 0.8416
```

```
# Add to results
rf_results = {
    'Model Name': 'Random Forest',
    'AUC-ROC': roc_auc_score(y_test, y_prob_rf),
    'Accuracy': accuracy_score(y_test, y_pred_rf),
    'Model-CV': '{rf_search.best_score_:.4f} ± {rf_search.cv_results_[\'std_test_score\'][rf_search.best_index_]:.4f}'
}

models_comparison = pd.concat([models_comparison, pd.DataFrame([rf_results])])
models_comparison = models_comparison.sort_values('AUC-ROC', ascending=False)

models_comparison
```

	Model Name	AUC-ROC	Accuracy	Model-CV
0	Linear Regression	0.8453	0.8034	0.8444 ± 0.0111
1	Lasso Regression	0.8453	0.8034	0.8444 ± 0.0111
2	Ridge Regression	0.8453	0.8034	0.8444 ± 0.0111
3	Random Forest	0.8416	0.8013	0.8444 ± 0.0111
4	XGBoost Regressor	0.8290	0.7810	0.8444 ± 0.0111

```
# Create comparison table
comparison_table = pd.DataFrame(models_comparison)
comparison_table = comparison_table.reset_index(drop=True)

comparison_table
```

	Model Name	AUC-ROC	Accuracy	Model-CV
0	Linear Regression	0.8453	0.8034	0.8444 ± 0.0111
1	Lasso Regression	0.8453	0.8034	0.8444 ± 0.0111
2	Ridge Regression	0.8453	0.8034	0.8444 ± 0.0111
3	Random Forest	0.8416	0.8013	0.8444 ± 0.0111
4	XGBoost Regressor	0.8290	0.7810	0.8444 ± 0.0111

Final Model

Final Model Random Forest

```
# Final Random Forest Classifier
rf = RandomForestClassifier(
    n_estimators=100,
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    max_features='sqrt',
    bootstrap=True,
    oob_score=True,
    random_state=42,
    n_jobs=-1,
    verbose=0,
    warm_start=False,
    class_weight=None,
    ccp_alpha=0.0,
    max_samples=None,
    monotonic_cst=None
)

# Fit the model
rf.fit(X_train, y_train)

# Predictions
y_pred_rf = rf.predict(X_test)
y_prob_rf = rf.predict_proba(X_test)[:, 1]

print("Final Random Forest Performance:")
print("Accuracy: {accuracy_score(y_test, y_pred_rf):.4f}")
print("ROC-AUC: {roc_auc_score(y_test, y_prob_rf):.4f}")
```

```
Final Random Forest Performance:
Accuracy: 0.8013
ROC-AUC: 0.8416
```

Translation and interpretation of the error

```
# Create comparison table
comparison_table = pd.DataFrame(models_comparison)
comparison_table = comparison_table.reset_index(drop=True)

comparison_table
```

Business Performance

```
# Business impact analysis
cost_per_customer = 100  # Average cost to retain customer
revenue_per_customer = 500  # Average revenue per customer

print("Business Performance:")
```

```
Business Performance:
```

	actual_churn	pred_churn	predicted_churn
0	0	0.0	0.0
1	0	0.0	0.0
2	1	0.0	0.0
3	1	0.0	0.0
4	1	0.0	0.0

```
# Calculate business metrics
df_test = df_test.copy()
df_test['prob_churn'] = y_prob_rf
df_test['pred_churn'] = y_pred_rf
df_test['actual_churn'] = y_test

# Confusion matrix components
tp = (df_test['actual_churn'] == 1) & (df_test['pred_churn'] == 1)
fp = (df_test['actual_churn'] == 0) & (df_test['pred_churn'] == 1)
tn = (df_test['actual_churn'] == 0) & (df_test['pred_churn'] == 0)
fn = (df_test['actual_churn'] == 1) & (df_test['pred_churn'] == 0)

# Business metrics
print("True Positives (correctly predicted churn): {tp.sum()}")
print("False Positives (incorrectly predicted churn): {fp.sum()}")
print("True Negatives (correctly predicted loyal): {tn.sum()}")
print("False Negatives (missed churn): {fn.sum()}")

# Financial impact
print("\nFinancial Impact:")
print("Cost of False Positives: $" + str(fp.sum() * cost_per_customer)")
print("Revenue from True Positives: $" + str(tp.sum() * revenue_per_customer)")
print("Missed revenue (False Negatives): $" + str(fn.sum() * revenue_per_customer)")
```

```
True Positives (correctly predicted churn): 0
False Positives (incorrectly predicted churn): 0
True Negatives (correctly predicted loyal): 1033
False Negatives (missed churn): 374

Financial Impact:
Cost of False Positives: $0
Revenue from True Positives: $0
Missed revenue (False Negatives): $187000
```

```
# Customer segmentation analysis
df_test['risk_segment'] = pd.cut(df_test['prob_churn'], bins=[0, 0.3, 0.7, 1.0], labels=['Low', 'Medium', 'High'])

segment_analysis = df_test.groupby('risk_segment').agg{
    'actual_churn': ['count', 'sum'],
    'prob_churn': 'mean'
}

segment_analysis
```

risk_segment	actual_churn		prob_churn
	count	sum	mean
Low	1033	0	0.000000
Medium	0	0	NaN
High	374	187	0.500000

```
# Save model and results
import joblib
joblib.dump(rf, 'churn_model.pkl')
comparison_table.to_csv('model_comparison.csv')

print("Model saved as churn_model.pkl")
print("Results saved as model_comparison.csv")
```

```
Model saved as churn_model.pkl
Results saved as model_comparison.csv
```

```
# Summary statistics
print("Dataset shape: {df.shape}")
print("Churn rate: {df['Churn_Yes'].mean()*100:.2f}%")
print("Best model: {comparison_table.iloc[0]['Model Name']}")
print("Best AUC-ROC: {comparison_table.iloc[0]['AUC-ROC']:.4f}")
```

```
Dataset shape: (7043, 45)
Churn rate: 26.54%
Best model: Linear Regression
Best AUC-ROC: 0.8453
```


Machine Learning Performance

```
# ROC Curves
plt.figure(figure_size=(8, 6))
for i, model_name in enumerate(['Linear Regression', 'Lasso Regression', 'Random Forest']):
    if model_name == 'Linear Regression':
        y_prob = y_prob_lr
    elif model_name == 'Lasso Regression':
        y_prob = y_prob_lasso
    else:
        y_prob = y_prob_rf
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    plt.plot(fpr, tpr, label='{model_name}')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves')
plt.legend()
plt.grid(True)
plt.show()
```



```
# Feature importance comparison
fig, axes = plt.subplots(1, 2, figure_size=(15, 6))

# Top 10 features
top_features = coefficients.head(10)
axes[0].barh(top_features['Feature'], top_features['Abs_Coefficient'])
axes[0].set_title('Top 10 Feature Importance')
axes[0].set_xlabel('Absolute Coefficient')

# Cumulative importance
cumulative = coefficients['Abs_Coefficient'].cumsum() / coefficients['Abs_Coefficient'].sum()
axes[1].plot(range(1, 11), cumulative[:10])
axes[1].set_title('Cumulative Feature Importance')
axes[1].set_xlabel('Number of Features')
axes[1].set_ylabel('Cumulative Importance')
axes[1].grid(True)

plt.tight_layout()
plt.show
```