03_prediction

November 5, 2024

0.0.1 Integrantes do Grupo

- André Mattos - RM358905
- Aurelio Thomasi Jr - RM358104
- Leonardo Ramires - RM358190
- Lucas Arruda - RM358628
- Pedro Marins - RM356883

Link dos dados utilizados

Apresentação do trabalho

0.0.2 Tech Challenge Fase 1

O objetivo é o desenvolvimento de um **modelo preditivo de regressão** para estimar os **custos de planos de saúde individuais**.

Os dados utilizados para a análise possuem as seguintes informações:

- Idade (age)
- Gênero (sex)
- IMC (bmi)
- Número de filhos (children)
- Se a pessoa é fumante (smoker)
- Região onde reside (region)
- Custo do plano de saúde (charges)

O processo de desenvolvimento do modelo passará pelas fases de:

1) **Preparação do ambiente**
2) **Exploração de Dados**
3) **Pré-processamento de Dados**
4) **Treinamento e Avaliação dos Modelos**

1) Preparação do ambiente Nessa fase iremos fazer a **importação das bibliotecas** e **leitura dos dados** que serão utilizados para a análise.

[1]:
```
# Importação das bibliotecas necessárias
import pandas as pd
import numpy as np
```

```
import time
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
from sklearn.model_selection import train_test_split

from sklearn.metrics import r2_score, mean_absolute_error,␣
 ↪mean_absolute_percentage_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import ElasticNet, Lasso, Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import cross_val_score

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df = pd.read_csv("02_insurance.csv")
df.head()
```

[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

0.0.3 2) Exploração dos Dados

Para entender a estrutura, os padrões do dataset e possíveis problemas no dataset, serão feitas algumas análises que envolve:

2.1) Estatísticas descritivas

2.2) Distribuição das variáveis

2.3) Análise de correlação

2.4) Análise bivariada

2.1) Estatísticas descritivas

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

[4]:
```
df.describe()
```

[4]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

Com base no resumo apresentado acima, podemos identificar as seguintes informações:

Valores de Cobrança:

- O valor máximo cobrado de um paciente é de 63.770,43;
- O valor mínimo cobrado de um paciente é de 1.121,87.

Idade:

- A idade dos pacientes está compreendida entre 18 e 64 anos com uma média de aprox. 39 anos.

Número de Filhos:

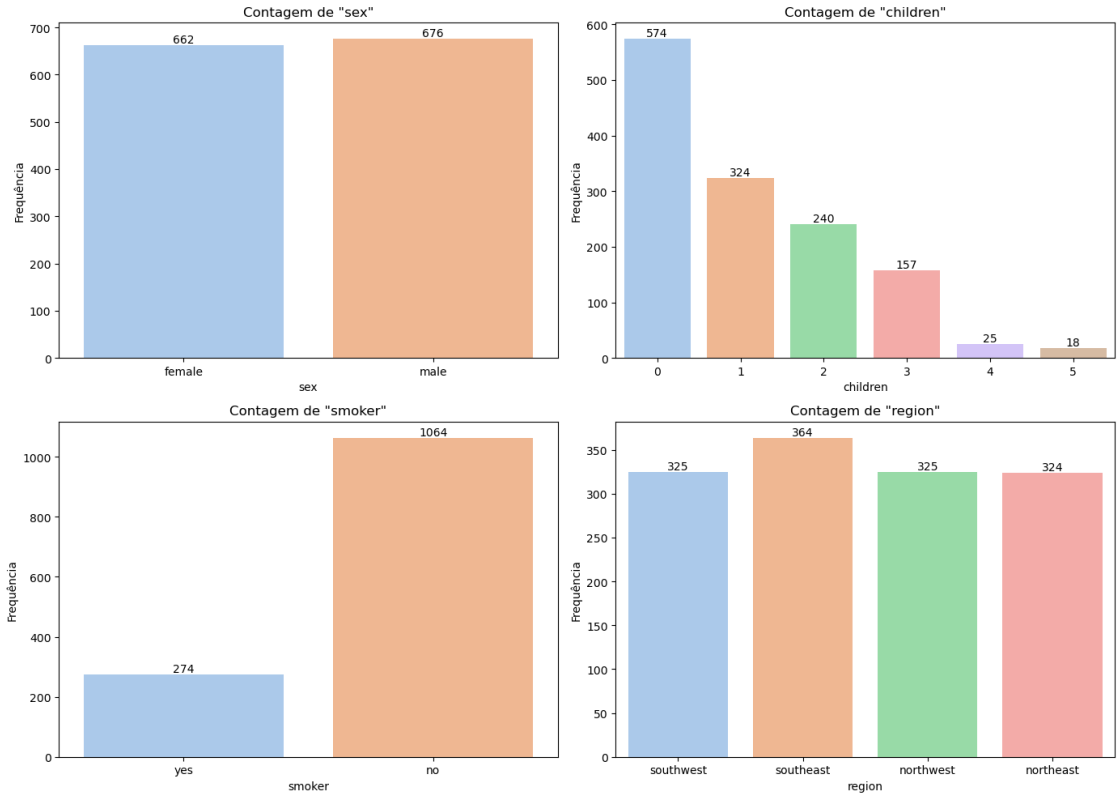
- O número de filhos por paciente varia de 0 a 5.

Índice de Massa Corporal (IMC):

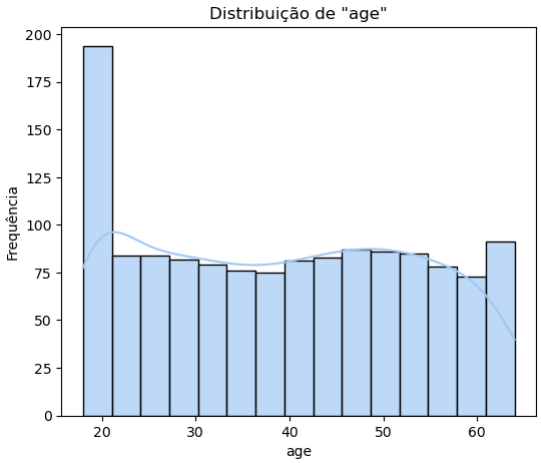
- O IMC máximo registrado é de 53;
- O IMC mínimo registrado é de 15.

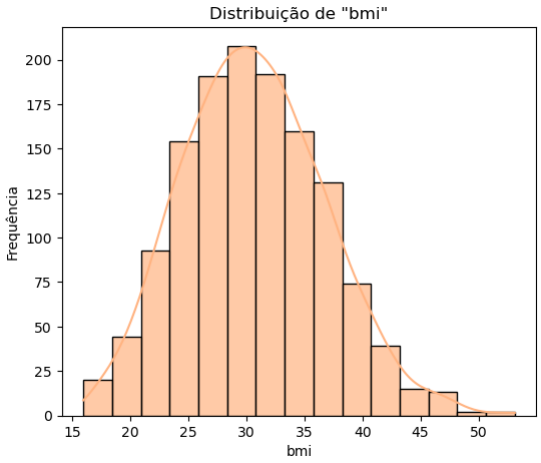
2.2) Distribuição das variáveis

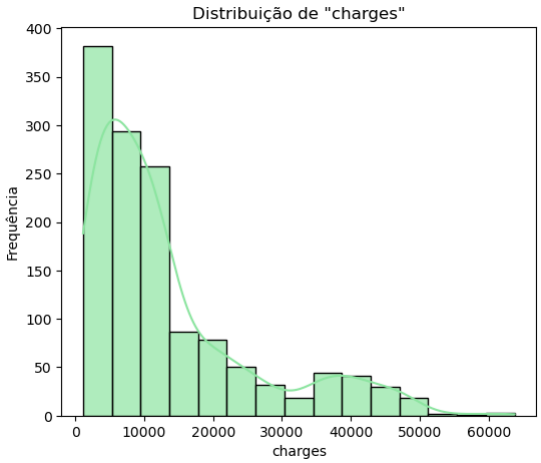
[5]:
```
# Colunas categóricas
cat_cols = ['sex', 'children', 'smoker', 'region']

# Configurando os subplots
fig, axes = plt.subplots(2, 2, figsize=(14, 10))
axes = axes.flatten()  # Achata a matriz de eixos para facilitar a iteração

# Configurando a paleta de cores
palette = sns.color_palette('pastel')

# Gerando os gráficos de barras com rótulos
for i, col in enumerate(cat_cols):
    ax = sns.countplot(data=df, x=col, ax=axes[i], palette=palette)
    ax.set_title(f'Contagem de "{col}"')
    ax.set_ylabel('Frequência')
    ax.set_xlabel(col)

    # Adicionando rótulos de contagem em cada barra
    for p in ax.patches:
        ax.annotate(f'{int(p.get_height())}',
                    (p.get_x() + p.get_width() / 2, p.get_height()),
                    ha='center', va='bottom', fontsize=10, color='black')

plt.tight_layout()  # Ajusta os subplots para não sobrepor
plt.show()
```



[6]:
```
# Lista de variáveis numéricas
num_cols = ['age', 'bmi', 'charges']

# Configuração do grid para as subplots
fig, axes = plt.subplots(2, 2, figsize=(12, 10))  # grade 2x2
fig.tight_layout(pad=4.0)

# Configurando a paleta de cores
palette = sns.color_palette('pastel')

# Gera os histogramas para cada variável
for i, col in enumerate(num_cols):
    ax = sns.histplot(data=df, x=col, ax=axes[i // 2, i % 2], kde=True,␣
 ↪color=palette[i % len(palette)], bins=15, alpha=0.7)
    ax.set_title(f'Distribuição de "{col}"')
    ax.set_xlabel(col)
    ax.set_ylabel('Frequência')

# Remover o eixo não utilizado
fig.delaxes(axes[-1][1])

plt.show()
```







[7]:
```
# Avaliando a presença de valores nulos
(
    df.isnull()
    .sum()
    .to_frame()
    .reset_index()
    .rename({"index": "column", 0: "valores_nulos"}, axis=1)
)
```

[7]:

	column	valores_nulos
0	age	0
1	sex	0
2	bmi	0
3	children	0
4	smoker	0
5	region	0
6	charges	0

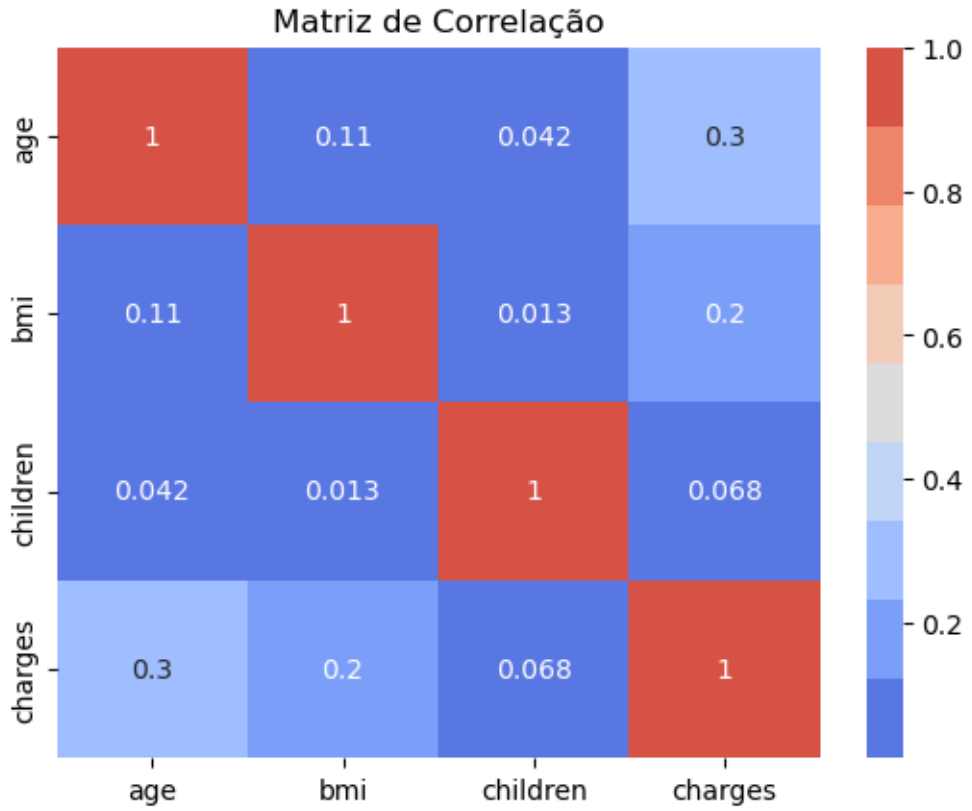
[8]:
```
# Criando coluna boolena para identificar se tem filho ou não
df.loc[df["children"] > 0, "has_children"] = "yes"
df["has_children"].fillna("no", inplace=True)
df.head()
```

[8]:

	age	sex	bmi	children	smoker	region	charges	has_children
0	19	female	27.900	0	yes	southwest	16884.92400	no
1	18	male	33.770	1	no	southeast	1725.55230	yes
2	28	male	33.000	3	no	southeast	4449.46200	yes
3	33	male	22.705	0	no	northwest	21984.47061	no
4	32	male	28.880	0	no	northwest	3866.85520	no

2.3 Análise de correlação

[9]:
```
# Seleciona apenas as colunas numéricas para o cálculo da correlação
numerical_df = df.select_dtypes(include=['number'])
div_palette = sns.color_palette("coolwarm", 9)
# Criação da matriz de correlação entre as variáveis numéricas
sns.heatmap(numerical_df.corr(), annot=True, cmap=div_palette)
plt.title('Matriz de Correlação')
plt.show()
```



De acordo com o resultado obtido com a matriz de correlação das variáveis numéricas, podemos concluir que:

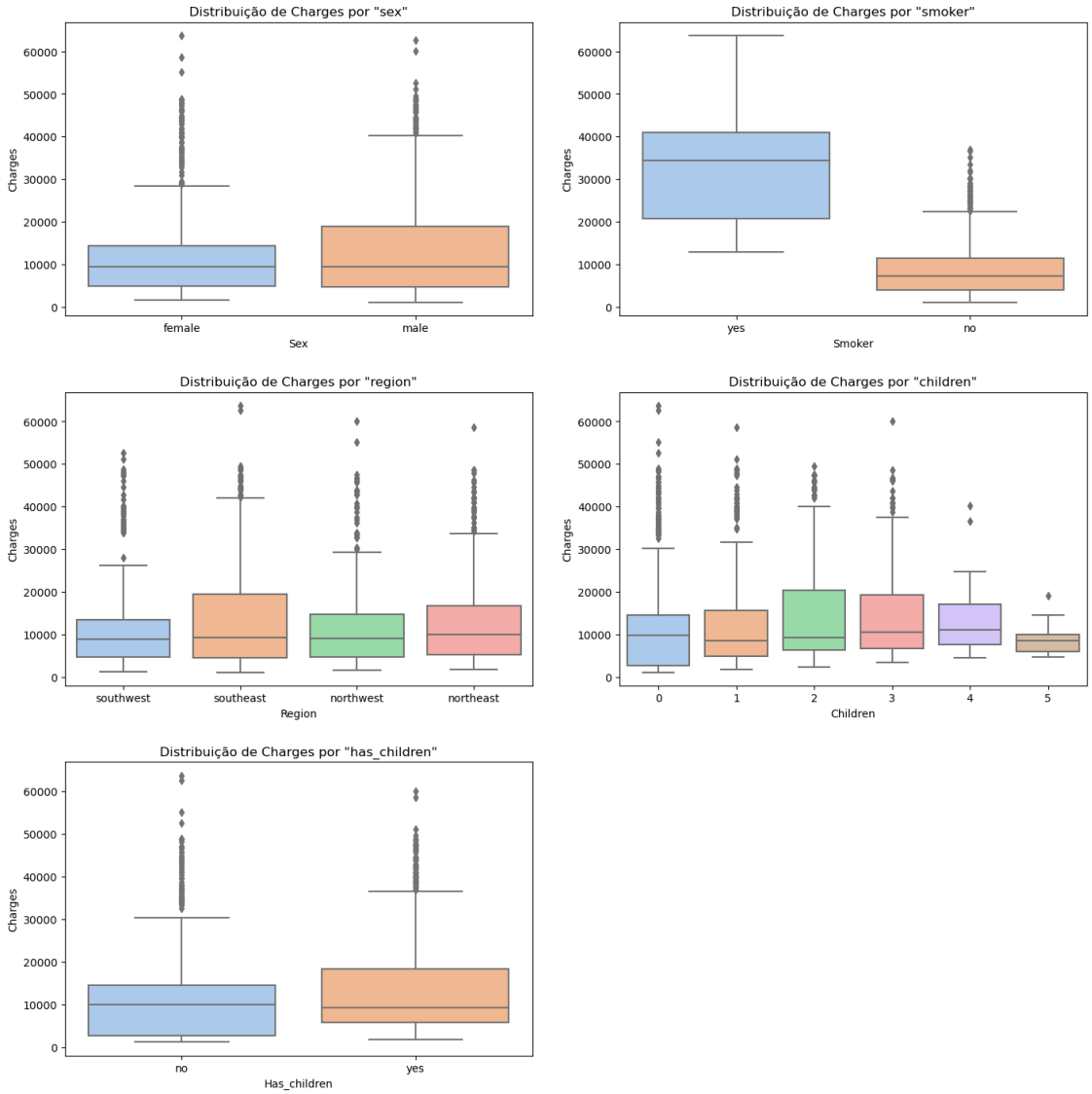
- A correlação entre **custos e IMC é positiva, mas relativamente baixa**, com um coeficiente em torno de 0.2.
- A correlação entre **custos e idade é positiva e moderada**, com um coeficiente em torno de 0.3.
- A correlação entre **custos e número de filhos é muito baixa**, com um coeficiente de apenas 0.068.

Conclusão:

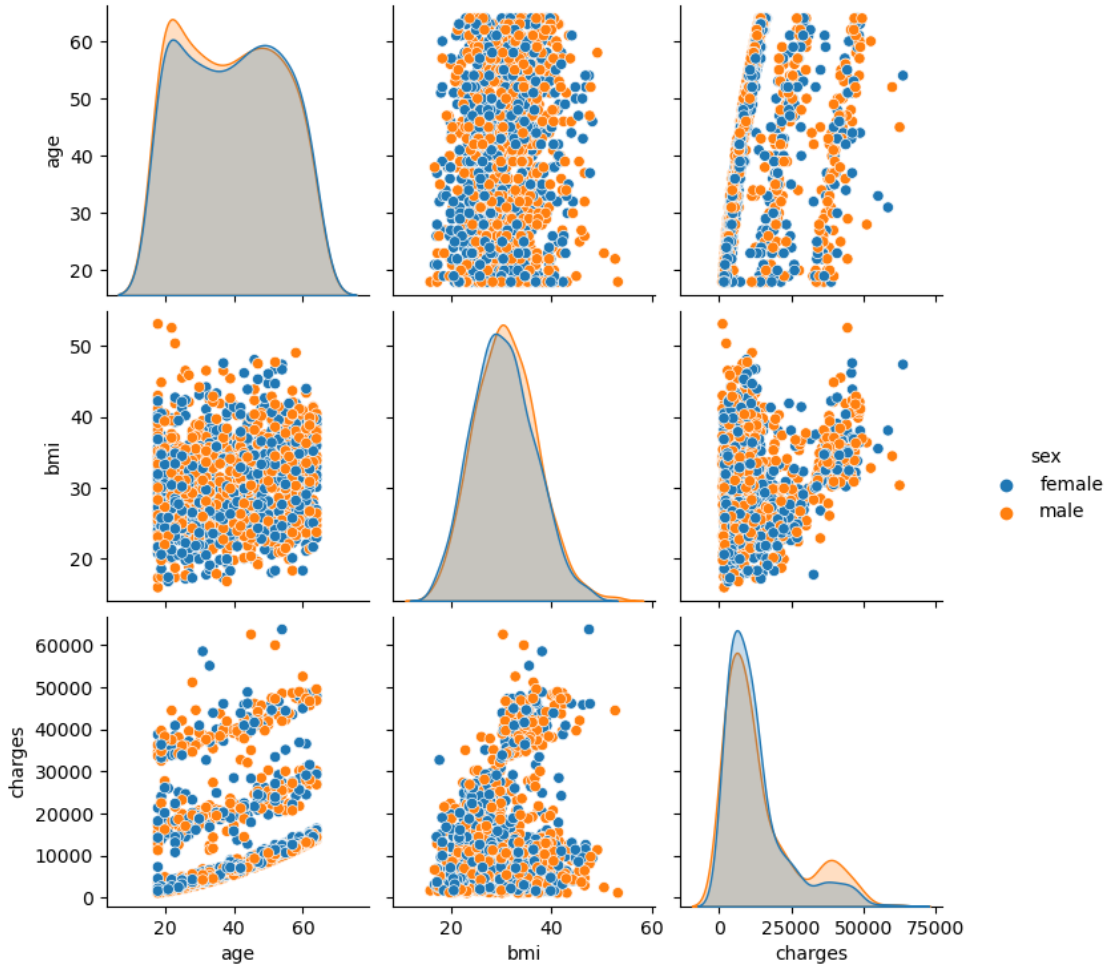
A matriz de correlação sugere que **idade e IMC são mais relevantes para prever custos**, enquanto **número de filhos** pode ter uma **menor prioridade na modelagem** preditiva.

2.4 Análise bivariada

[10]:
```
# Configura o tamanho da figura
fig, axes = plt.subplots(3, 2, figsize=(15, 15))
fig.tight_layout(pad=5.0)

# Lista de variáveis categóricas
cat_cols = ['sex', 'smoker', 'region', 'children', 'has_children']
titles = ['Distribuição de Charges por "sex"', 'Distribuição de Charges por␣
 ↪"smoker"',
          'Distribuição de Charges por "region"', 'Distribuição de Charges por␣
 ↪"children"',
          'Distribuição de Charges por "has_children"']

# Cria os boxplots em subplots com a paleta pastel
for i, col in enumerate(cat_cols):
    sns.boxplot(data=df, x=col, y='charges', ax=axes[i // 2, i % 2],␣
 ↪palette='pastel')
    axes[i // 2, i % 2].set_title(titles[i])
    axes[i // 2, i % 2].set_xlabel(col.capitalize())
    axes[i // 2, i % 2].set_ylabel('Charges')

# Remover o eixo não utilizado
fig.delaxes(axes[-1][1])

plt.show()
```



[11]:
```
# Pair plot das variáveis numéricas
sns.pairplot(df, vars=['age', 'bmi', 'charges'], hue='sex')
plt.show()
```



Com base nos boxplots fornecidos, podemos obter alguns insights sobre a relação entre **charges** (custos de planos de saúde) e as **variáveis categóricas** (sex, smoker, region):

smoker (fumante):

- Há uma **diferença clara nos custos** para pessoas **fumantes** em comparação com **não fumantes**.
- Os **fumantes tendem** a ter **custos médios muito mais altos**, com uma **mediana** de charges significativamente **superior à dos não fumantes**.
- A **variabilidade** dos custos entre **fumantes** também é **maior**, indicando que ser fumante é um **fator importante no aumento dos custos** dos planos de saúde.

sex (gênero):

- Há uma **pequena diferença** nos custos de planos de saúde **entre homens e mulheres**, com os **homens tendendo** a ter custos um pouco **mais altos em média**.
- Importante ressaltar que **essa diferença é bem menor em comparação** com o impacto observado em relação à variável fumante.

region (região):

- A **região Sudeste** (southeast) **tem uma mediana** de custos um pouco **mais alta** do que as **outras regiões**.
- A **região Sudoeste** (southwest) apresenta a **mediana** de custos **mais baixa**.
- Vale destacar que a **variabilidade** dos custos é **semelhante entre as regiões**, o que indica que a **localização pode ter um efeito moderado** sobre os custos, mas é menos relevante do que a variável fumante.

children (número de filhos)

- A **mediana dos custos** de planos é **relativamente consistente**, sem grandes mudanças visíveis.
- O grupo com **4 e 5 filhos** tem uma **menor quantidade de dados**, o que pode **afetar a confiabilidade da análise**. Esses grupos mostram uma **mediana mais baixa e maior dispersão** em **relação ao restante**.
- Existem **outliers em todos os grupos**, isso indica que **outros fatores** além do número de filhos **influenciam os custos mais altos** dos planos de saúde.
- O número de filhos **parece ter um impacto limitado** e **não é um fator determinante** isolado para os **custos dos planos**.

has_children (tem filhos?): - A **mediana dos custos** é bem próxima entre as pessoas quem **tem ou não tem filhos** e o custos são relativamente maiores para as pessoas que tem filhos, porém, **não parece ter grande impacto nos custos**.

Conclusão

A variável **fumante (smoker) é o fator categórico mais impactante nos custos**, com uma diferença clara na mediana e na distribuição geral. A variável **region pode ter uma influência moderada**, e o número de filhos tem um impacto menos evidente. O gênero, embora mostre algumas diferenças na dispersão, não parece ser um fator crucial na variação dos custos dos planos.

0.0.4 3) Pré-processamento dos dados

Nessa etapa, o objetivo é preparar os dados para realização da análise e serão realizadas as seguintes transformações:

- Codificação das variáveis categóricas utilizando label encoding e one-hot encoding;
- Separação da variável dependente da variável independente;
- Normalização dos dados através das técnicas:
  - Min Max Scaler;
  - Standard Scaler.

[12]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

df_coded = df.copy()
df_coded['sex'] = label_encoder.fit_transform(df_coded['sex'])
df_coded['smoker'] = label_encoder.fit_transform(df_coded['smoker'])
df_coded['has_children'] = label_encoder.fit_transform(df_coded['has_children'])

dummy_region = pd.get_dummies(df_coded['region'], prefix='dummy')
df_coded = pd.concat([df_coded, dummy_region], axis=1)
```

```
df_coded.drop("region", axis=1, inplace=True)

df_coded.head()
```

[12]:

	age	sex	bmi	children	smoker	charges	has_children	\
0	19	0	27.900	0	1	16884.92400	0	
1	18	1	33.770	1	0	1725.55230	1	
2	28	1	33.000	3	0	4449.46200	1	
3	33	1	22.705	0	0	21984.47061	0	
4	32	1	28.880	0	0	3866.85520	0	

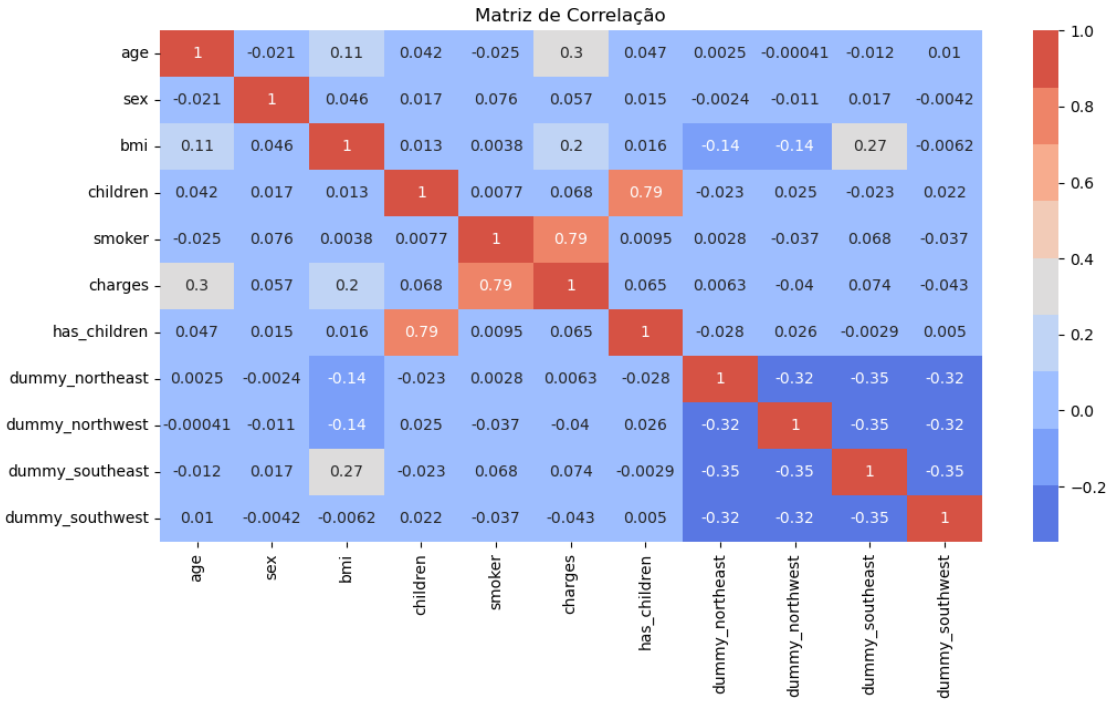
	dummy_northeast	dummy_northwest	dummy_southeast	dummy_southwest
0	0	0	0	1
1	0	0	1	0
2	0	0	1	0
3	0	1	0	0
4	0	1	0	0

[13]:
```
(
    df_coded.corr()
    .round(2)["charges"]
    .sort_values(ascending=False)
    .to_frame()
    .reset_index()
    .rename({'index': 'variable', 'charges': 'pearson'}, axis=1)
)
```

[13]:

	variable	pearson
0	charges	1.00
1	smoker	0.79
2	age	0.30
3	bmi	0.20
4	children	0.07
5	dummy_southeast	0.07
6	sex	0.06
7	has_children	0.06
8	dummy_northeast	0.01
9	dummy_northwest	-0.04
10	dummy_southwest	-0.04

[14]:
```
# Criação da matriz de correlação entre as variáveis numéricas
plt.figure(figsize=(12, 6))
div_palette = sns.color_palette("coolwarm", 9)
sns.heatmap(df_coded.corr(), annot=True, cmap=div_palette)
plt.title('Matriz de Correlação')
plt.show()
```



Já foi observado que as variáveis IMC e idade possuem correlação (mesmo que não muito significativa) em relação ao custo. Em relação às demais variáveis pode-se observar o seguinte resultado:

- **Ser fumante**: A **correlação é muito forte** e negativa entre não fumantes e custos (-0,79) e positiva entre fumantes e charges (0,79). Isso **indica que fumar está fortemente associado a custos médicos mais altos**.
- **Regiões**: As **correlações entre as regiões e charges são fracas**, mas a região **sudeste** apresenta a **maior correlação positiva** com charges (0,07), sugerindo um **leve aumento de custos** nessa região.
- **Sexo**: **Não há correlações significativas** entre o sexo e os custos, o que sugere que a **influência do gênero nos custos médicos é mínima** em comparação com outras variáveis.

Conclusão

A variável **ser fumante** (smoker) possui maior significância, apresentando **maior correlação com os custos**.

0.0.5 4) Treinamento e Avaliação dos Modelos

Para avaliação dos possíveis modelos a serem utilizados, serão aplicas diferentes técnicas para os datasets com e sem scaler, considerando as variáveis que possuem maior impacto de acordo com a análise realizada.

Premissas

1) Diferentes métodos de **feature scaling** para os datasets:
   - Sem scaler

- Min Max Scaler
- Standard Scaler

2) Diferentes **técnicas para regressão**:
   - SVR;
   - Decision Tree Regressor;
   - MLP Regressor;
   - Ridge;
   - Elastic Net;
   - Linear Regression;
   - Lasso;
   - Random Forest Regressor.
3) Variáveis independentes:
   - age
   - bmi
   - smoker

Métricas para Avaliação dos Modelos Abaixo temos as métricas que serão avaliadas para cada um dos modelos a serem testados:

1. MAE (Mean Absolute Error) - **Definição:** Média das diferenças absolutas entre valores reais e previstos. - **Interpretação:** Indica o erro médio em unidades da variável. Fácil de entender e não penaliza erros grandes mais que pequenos.

---

2. MSE (Mean Squared Error) - **Definição:** Média dos erros ao quadrado entre valores reais e previstos. - **Interpretação:** Penaliza erros grandes mais severamente. Sensível a outliers.

---

3. RMSE (Root Mean Squared Error) - **Definição:** Raiz quadrada do MSE. - **Interpretação:** Representa a magnitude média do erro nas mesmas unidades da variável. Penaliza erros grandes.

---

4. R^2 (Coeficiente de Determinação) - **Definição:** Proporção da variabilidade dos dados explicada pelo modelo. - **Interpretação:** Varia de 0 a 1; 0 significa que o modelo não explica a variabilidade, enquanto 1 significa que explica toda a variabilidade. Um R^2 alto não garante um bom modelo.

---

5. Duração da execução do Modelo - Tempo necessário para executar a separação dos dados em treino e teste e para treinar, ajustar e executar o modelo.

[15]:
```
# Função destinada a fazer a divisão dos datasets de treino e teste

def split_datasets(df, variaveis_independentes, variavel_dependente, scaler,␣
↪test_size=0.2, random_state=42):
    # Seleciona as variáveis independentes e a dependente
```

```
    X = df[variaveis_independentes]
    y = df[variavel_dependente]

    if scaler:
        # Aplica o scaler nos dados
        X = scaler.fit_transform(X)

    # Separa o dataset
    X_train, X_test, y_train, y_test = train_test_split(X, y,␣
 ↪test_size=test_size, random_state=random_state)
    return X_train, X_test, y_train, y_test
```

[16]:
```
# Printa na tela o padrao de scaler sendo analisado
def print_scaler(scaler):
    print("-----------------------------------------------------")
    print(f"- Avaliando modelos lineares usando  {scaler}-")
    print("-----------------------------------------------------")
```

[17]:
```
# Avalia os diferentes modelos
def evaluate_models(models, df, variaveis_independentes, variavel_dependente,␣
 ↪scaler):
    print_scaler(scaler)
    results = []
    for name, model in models:
        # Registra o tempo de início
        begin = time.time()

        X_train, X_test, y_train, y_test = split_datasets(df,␣
 ↪variaveis_independentes, variavel_dependente, scaler)

        # Ajusta o modelo para os datasets de treino
        model.fit(X_train, y_train)

        # Realiza as predições nos datasets de teste
        y_pred = model.predict(X_test)

        # Avalia o desempenho do modelo usando MAE, MSE, RMSE e R²
        mae = mean_absolute_error(y_test, y_pred)
        mse = mean_squared_error(y_test, y_pred)
        rmse = np.sqrt(mse)
        r2 = r2_score(y_test, y_pred)

        # Calcula o tempo de execução
        duration = time.time() - begin

        # Calcula p-valores usando statsmodels (apenas para modelos lineares)
        try:
```

```
            X_train_const = sm.add_constant(X_train)  # Adiciona a constante␣
↪para o intercepto
            modelo_stats = sm.OLS(y_train, X_train_const).fit()  # Ajusta o␣
↪modelo estatístico
            p_values = modelo_stats.pvalues[1:]  # Ignora o p-valor do␣
↪intercepto
        except Exception as e:
            p_values = "N/A"
            print(f"Não foi possível calcular os p-valores para o modelo {name}:
↪ {e}")

        # Print das estatísticas do modelo
        print(f'{name}: MAE = {mae:.3f} - MSE = {mse:.3f} - RMSE = {rmse:.3f} -␣
↪R2 = {r2:.3f} [{duration:.3f}sec]')

        # Cria um dict com os resultados
        result = {
            'Scaler': scaler,
            'Modelo': name,
            'R²': r2,
            'RMSE': rmse,
            'MAE': mae,
            'MSE': mse,
            'Duração': duration,
            'P-valor F': p_values,
            'Real': y_test.values,
            'Previsto': y_pred,
        }
        results.append(result)
    return results
```

[18]:
```
# Seleção das variáveis independentes
variaveis_independentes = ['age', 'bmi', 'smoker']

# Configurando modelos
regressors = [
     ('SVR                    ', SVR(kernel='linear', C=4000.0))
    ,('Decision Tree Regressor', DecisionTreeRegressor(max_depth=33))
    ,('MLP Regressor          ', MLPRegressor(hidden_layer_sizes=(1000,),␣
↪max_iter=2000))
    ,('Ridge                  ', Ridge(alpha=1.0))
    ,('Elastic Net            ', ElasticNet(alpha=0.01, l1_ratio=0.5))
    ,('Linear Regression      ', LinearRegression())
    ,('Lasso                  ', Lasso(alpha=0.1))
    ,('Random Forest Regressor', RandomForestRegressor(n_estimators=30,␣
↪random_state=42))
```

```
]
```

[19]:
```
results_standard = evaluate_models(regressors, df_coded,␣
 ↪variaveis_independentes, variavel_dependente='charges',␣
 ↪scaler=StandardScaler())
results_minmax = evaluate_models(regressors, df_coded, variaveis_independentes,␣
 ↪variavel_dependente='charges', scaler=MinMaxScaler())
results = evaluate_models(regressors, df_coded, variaveis_independentes,␣
 ↪variavel_dependente='charges', scaler=None)
```

```
-----------------------------------------------------
- Avaliando modelos lineares usando  StandardScaler()-
-----------------------------------------------------
SVR                    : MAE = 3308.206 - MSE = 43804691.573 - RMSE = 6618.511 -
R2 = 0.718 [0.350sec]
Decision Tree Regressor: MAE = 3082.418 - MSE = 38713215.830 - RMSE = 6221.995 -
R2 = 0.751 [0.031sec]
MLP Regressor          : MAE = 3196.727 - MSE = 24651523.308 - RMSE = 4965.030 -
R2 = 0.841 [186.218sec]
Ridge                  : MAE = 4261.978 - MSE = 34517967.935 - RMSE = 5875.199 -
R2 = 0.778 [0.064sec]
Elastic Net            : MAE = 4268.118 - MSE = 34542561.148 - RMSE = 5877.292 -
R2 = 0.778 [0.026sec]
Linear Regression      : MAE = 4260.560 - MSE = 34512843.880 - RMSE = 5874.763 -
R2 = 0.778 [0.043sec]
Lasso                  : MAE = 4260.553 - MSE = 34512883.490 - RMSE = 5874.767 -
R2 = 0.778 [0.023sec]
Random Forest Regressor: MAE = 2811.365 - MSE = 26651682.167 - RMSE = 5162.527 -
R2 = 0.828 [0.407sec]
-----------------------------------------------------
- Avaliando modelos lineares usando  MinMaxScaler()-
-----------------------------------------------------
SVR                    : MAE = 3308.943 - MSE = 43111031.075 - RMSE = 6565.899 -
R2 = 0.722 [0.379sec]
Decision Tree Regressor: MAE = 3132.140 - MSE = 39839394.071 - RMSE = 6311.846 -
R2 = 0.743 [0.040sec]
MLP Regressor          : MAE = 3996.940 - MSE = 32709425.838 - RMSE = 5719.215 -
R2 = 0.789 [144.224sec]
Ridge                  : MAE = 4251.319 - MSE = 34539984.103 - RMSE = 5877.073 -
R2 = 0.778 [0.022sec]
Elastic Net            : MAE = 4241.810 - MSE = 34844600.763 - RMSE = 5902.932 -
R2 = 0.776 [0.016sec]
Linear Regression      : MAE = 4260.560 - MSE = 34512843.880 - RMSE = 5874.763 -
R2 = 0.778 [0.021sec]
Lasso                  : MAE = 4260.423 - MSE = 34512857.371 - RMSE = 5874.764 -
R2 = 0.778 [0.014sec]
Random Forest Regressor: MAE = 2826.220 - MSE = 26666173.323 - RMSE = 5163.930 -
```

```
R2 = 0.828 [0.361sec]
-------------------------------------------------------
- Avaliando modelos lineares usando  None-
-------------------------------------------------------
SVR                    : MAE = 3309.514 - MSE = 43113963.570 - RMSE = 6566.122 -
R2 = 0.722 [2.033sec]
Decision Tree Regressor: MAE = 3140.748 - MSE = 40038343.551 - RMSE = 6327.586 -
R2 = 0.742 [0.025sec]
MLP Regressor          : MAE = 4189.132 - MSE = 35268258.588 - RMSE = 5938.708 -
R2 = 0.773 [122.743sec]
Ridge                  : MAE = 4271.802 - MSE = 34548347.036 - RMSE = 5877.784 -
R2 = 0.777 [0.015sec]
Elastic Net            : MAE = 4321.054 - MSE = 34763618.541 - RMSE = 5896.068 -
R2 = 0.776 [0.012sec]
Linear Regression      : MAE = 4260.560 - MSE = 34512843.880 - RMSE = 5874.763 -
R2 = 0.778 [0.015sec]
Lasso                  : MAE = 4260.606 - MSE = 34512988.167 - RMSE = 5874.776 -
R2 = 0.778 [0.018sec]
Random Forest Regressor: MAE = 2820.420 - MSE = 26613362.823 - RMSE = 5158.814 -
R2 = 0.829 [0.357sec]
```

[20]:
```
df_results = [
    pd.DataFrame(results_standard),
    pd.DataFrame(results_minmax),
    pd.DataFrame(results)]

df_results[0].head()
```

[20]:

	Scaler	Modelo	R^2	RMSE	\
0	StandardScaler()	SVR	0.717842	6618.511281	
1	StandardScaler()	Decision Tree Regressor	0.750637	6221.994522	
2	StandardScaler()	MLP Regressor	0.841213	4965.030041	
3	StandardScaler()	Ridge	0.777660	5875.199395	
4	StandardScaler()	Elastic Net	0.777502	5877.291991	

	MAE	MSE	Duração	\
0	3308.205744	4.380469e+07	0.349532	
1	3082.418354	3.871322e+07	0.030529	
2	3196.727328	2.465152e+07	186.218219	
3	4261.978020	3.451797e+07	0.064207	
4	4268.118427	3.454256e+07	0.026004	

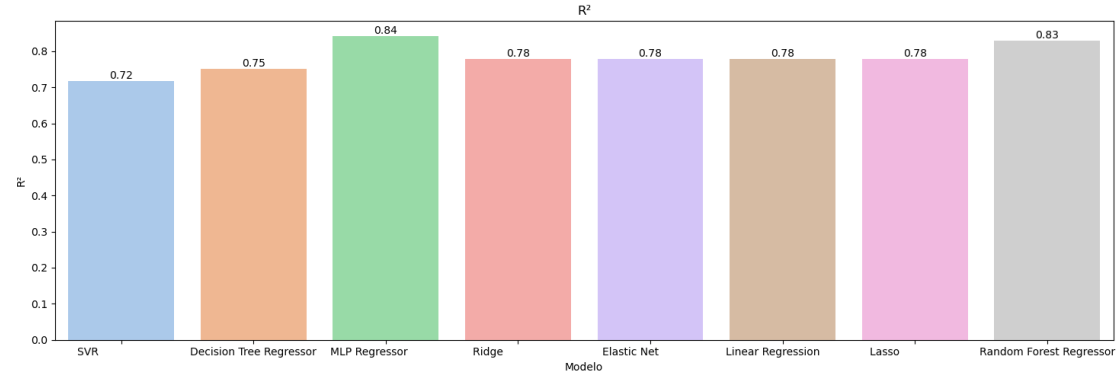
		P-valor F	\
0	x1	3.714304e-71	
	x2	2.792135e-24	
	x3	…	
1	x1	3.714304e-71	

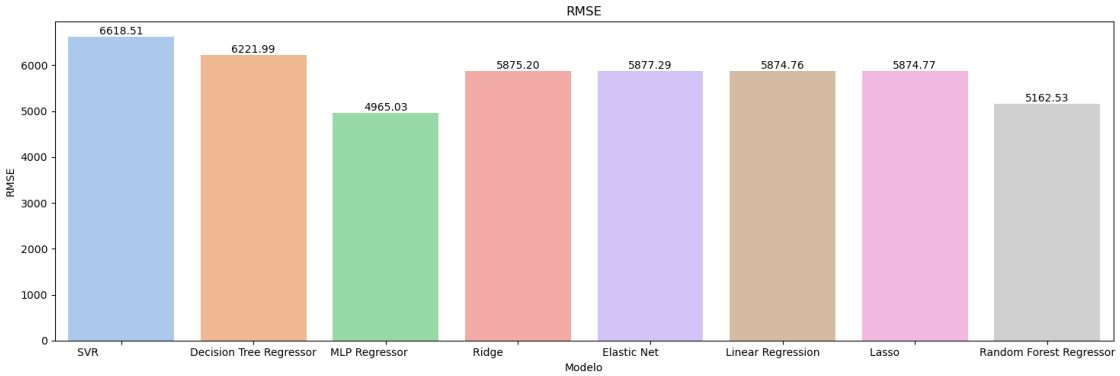
```
x2     2.792135e-24
x3    …
2  x1     3.714304e-71
x2     2.792135e-24
x3    …
3  x1     3.714304e-71
x2     2.792135e-24
x3    …
4  x1     3.714304e-71
x2     2.792135e-24
x3    …

                                                  Real  \
0  [9095.06825, 5272.1758, 29330.98315, 9301.8935…
1  [9095.06825, 5272.1758, 29330.98315, 9301.8935…
2  [9095.06825, 5272.1758, 29330.98315, 9301.8935…
3  [9095.06825, 5272.1758, 29330.98315, 9301.8935…
4  [9095.06825, 5272.1758, 29330.98315, 9301.8935…

                                               Previsto
0  [8389.50608196299, 6199.353759500285, 44498.06…
1  [9500.57305, 5478.0368, 28950.4692, 10096.97, …
2  [8463.143676616655, 5905.961053641169, 36918.9…
3  [8188.501136475334, 7436.646437062464, 37322.9…
4  [8207.817923947481, 7461.083075387974, 37221.3…
```

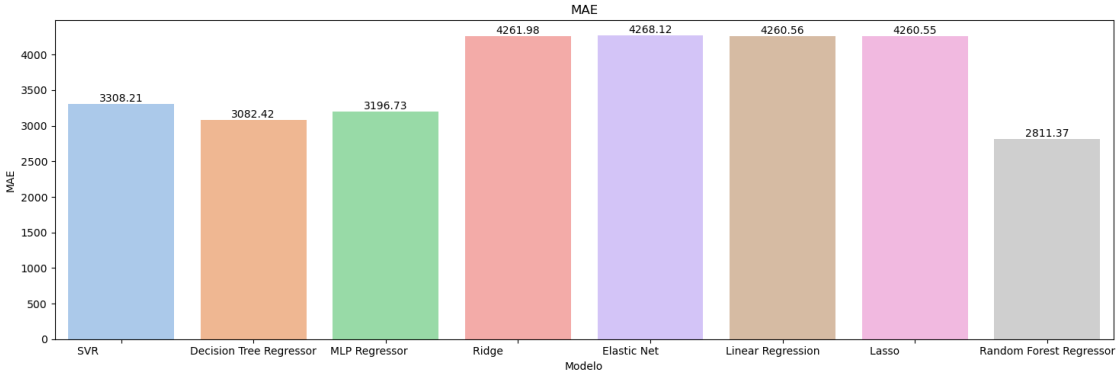
[21]:
```python
def generate_graphs(df_results):

    # Listando as métricas para iterar sobre elas
    metrics = ['R²', 'RMSE', 'MAE', 'MSE', 'Duração']

    # Configurando a paleta de cores
    palette = sns.color_palette('pastel')

    # Configurando os subplots
    fig, axes = plt.subplots(len(metrics), 1, figsize=(15, 25), sharey=False)

    # Criando um gráfico de barras para cada métrica
    for i, metric in enumerate(metrics):
        sns.barplot(data=df_results, x='Modelo', y=metric, ax=axes[i],␣
↪palette=palette)
        axes[i].set_title(metric)
        axes[i].set_xlabel('Modelo')
        for bar in axes[i].containers[0]:
            axes[i].text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
                         f'{bar.get_height():.2f}', ha='center', va='bottom')
```

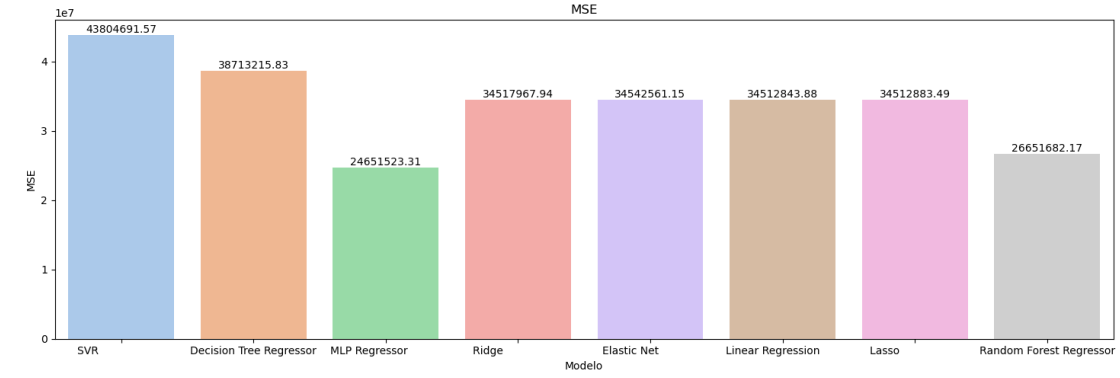
```
    if i == 0:
        axes[i].set_ylabel('Valor')

    # Ajusta o layout para evitar sobreposição
    plt.tight_layout()
    plt.show()

for r in df_results:
    r["Scaler"].fillna("SemScaler", inplace=True)
    print_scaler(r["Scaler"].unique()[0])
    generate_graphs(r)
```

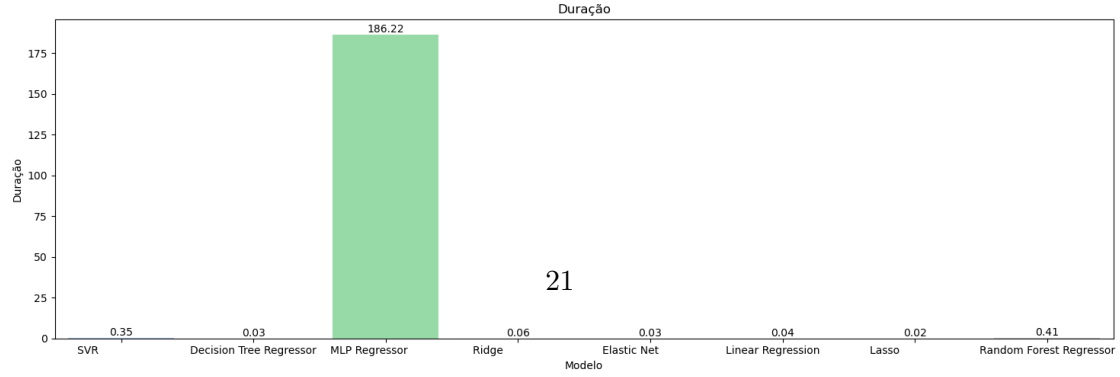
```
-------------------------------------------------------
- Avaliando modelos lineares usando  StandardScaler()-
-------------------------------------------------------
```


R²
0.72
0.75
0.84
0.78
0.78
0.78
0.78
0.83
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor
Modelo

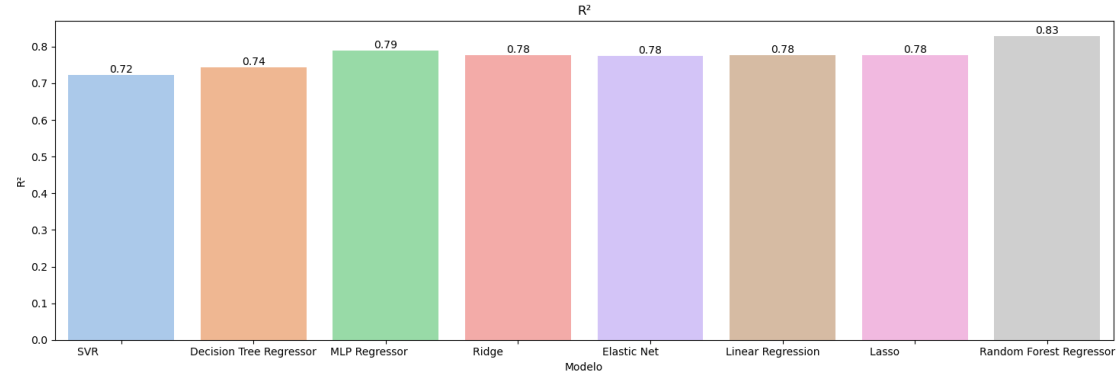

RMSE
6618.51
6221.99
4965.03
5875.20
5877.29
5874.76
5874.77
5162.53
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor
Modelo

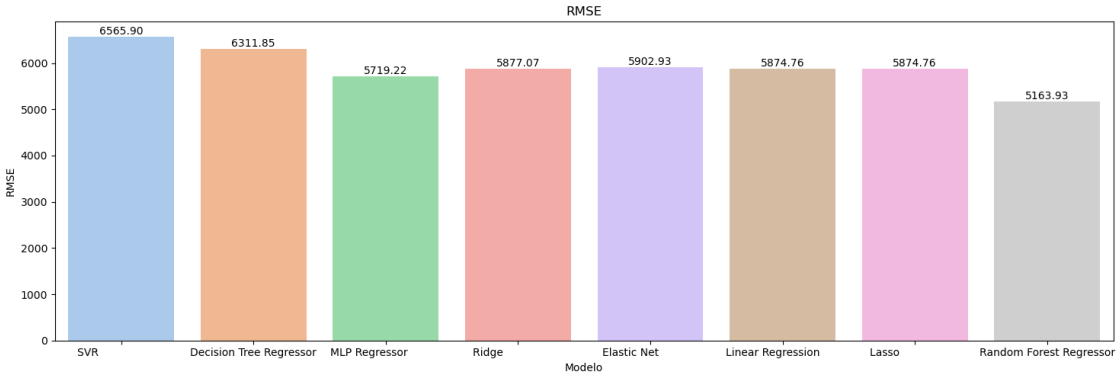

MAE
3308.21
3082.42
3196.73
4261.98
4268.12
4260.56
4260.55
2811.37
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor
Modelo

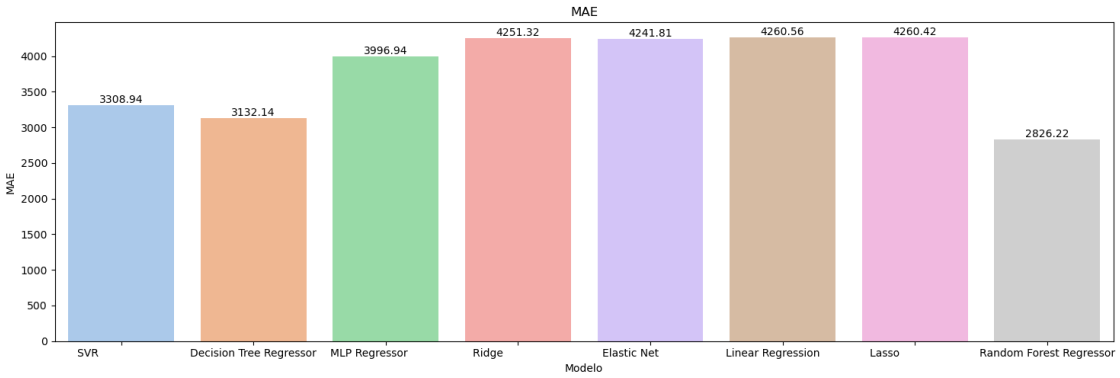

MSE
1e7
43804691.57
38713215.83
24651523.31
34517967.94
34542561.15
34512843.88
34512883.49
26651682.17
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor
Modelo

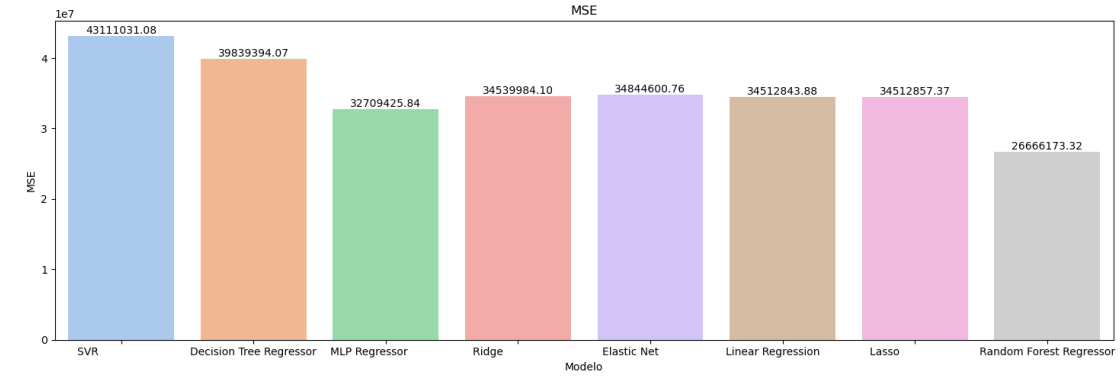

Duração
186.22
0.35
0.03
0.06
0.03
0.04
0.02
0.41
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor
Modelo

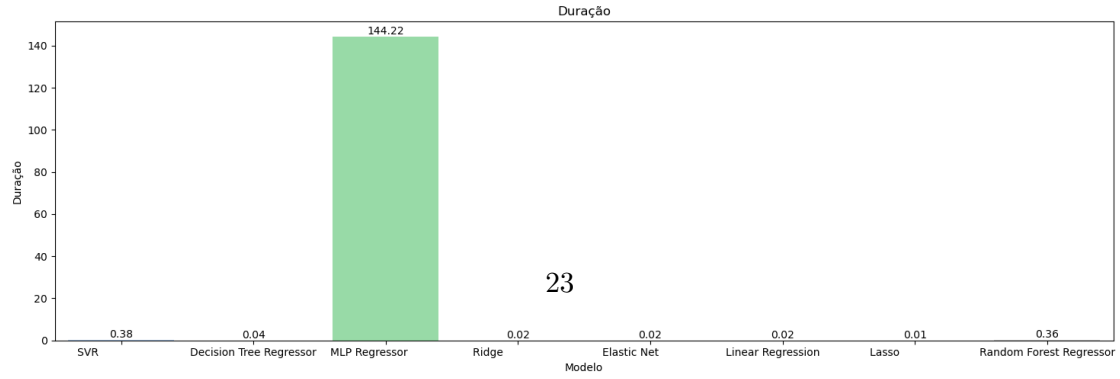
```
------------------------------------------------------
- Avaliando modelos lineares usando  MinMaxScaler()-
------------------------------------------------------
```


R²
R²
Modelo
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor
0.72
0.74
0.79
0.78
0.78
0.78
0.78
0.83

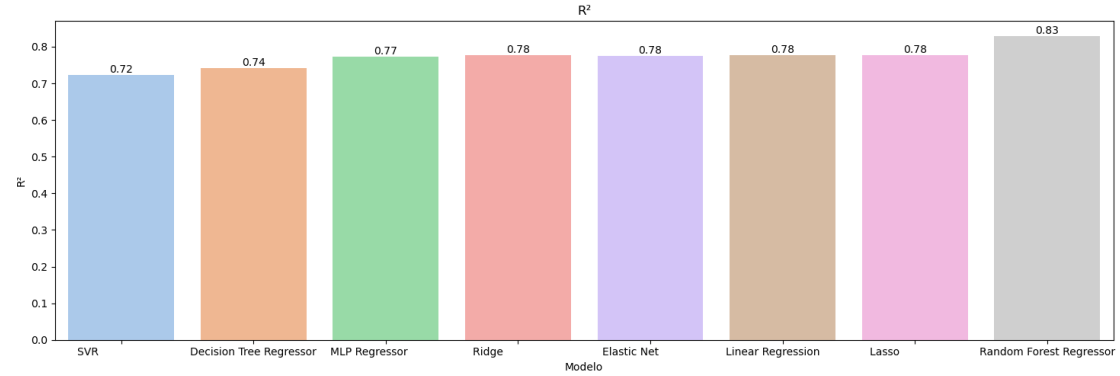

RMSE
RMSE
Modelo
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor
6565.90
6311.85
5719.22
5877.07
5902.93
5874.76
5874.76
5163.93

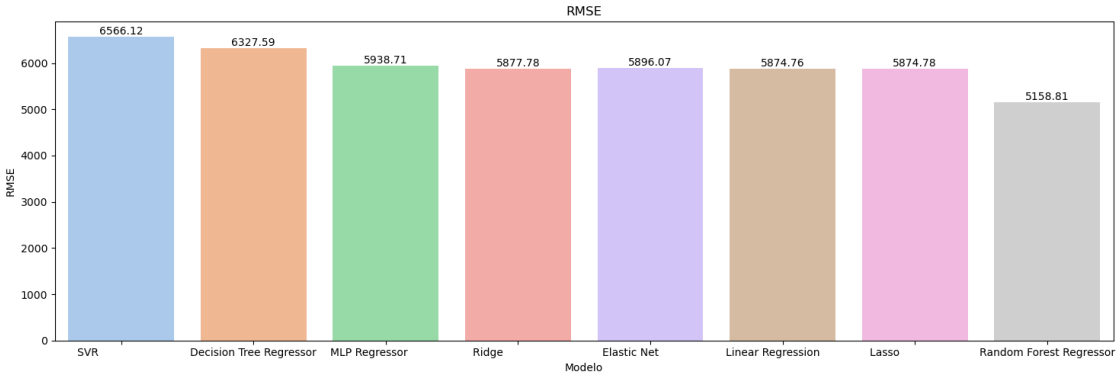

MAE
MAE
Modelo
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor
3308.94
3132.14
3996.94
4251.32
4241.81
4260.56
4260.42
2826.22

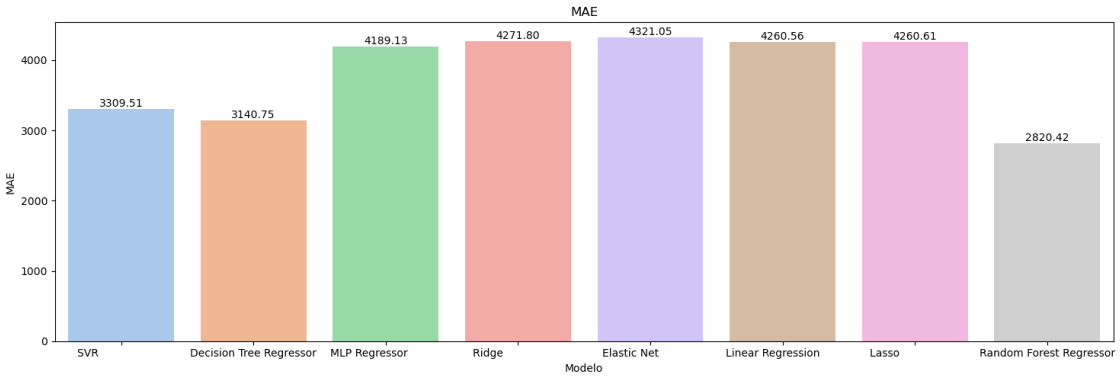

MSE
1e7
MSE
Modelo
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor
43111031.08
39839394.07
32709425.84
34539984.10
34844600.76
34512843.88
34512857.37
26666173.32

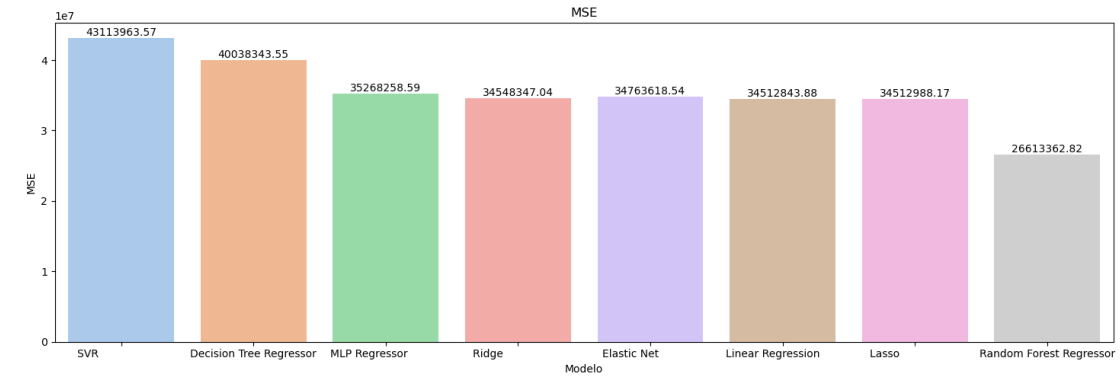

Duração
Duração
Modelo
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor
0.38
0.04
144.22
0.02
0.02
0.02
0.01
0.36

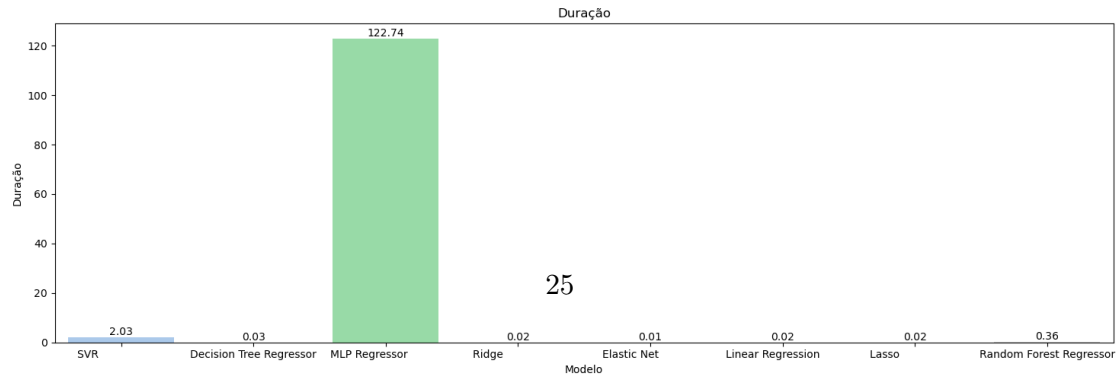
```
-------------------------------------------------------
- Avaliando modelos lineares usando  SemScaler-
-------------------------------------------------------
```


R²
0.72
0.74
0.77
0.78
0.78
0.78
0.78
0.83
R²
Modelo
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor


RMSE
6566.12
6327.59
5938.71
5877.78
5896.07
5874.76
5874.78
5158.81
RMSE
Modelo
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor


MAE
3309.51
3140.75
4189.13
4271.80
4321.05
4260.56
4260.61
2820.42
MAE
Modelo
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor


MSE
1e7
43113963.57
40038343.55
35268258.59
34548347.04
34763618.54
34512843.88
34512988.17
26613362.82
MSE
Modelo
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor

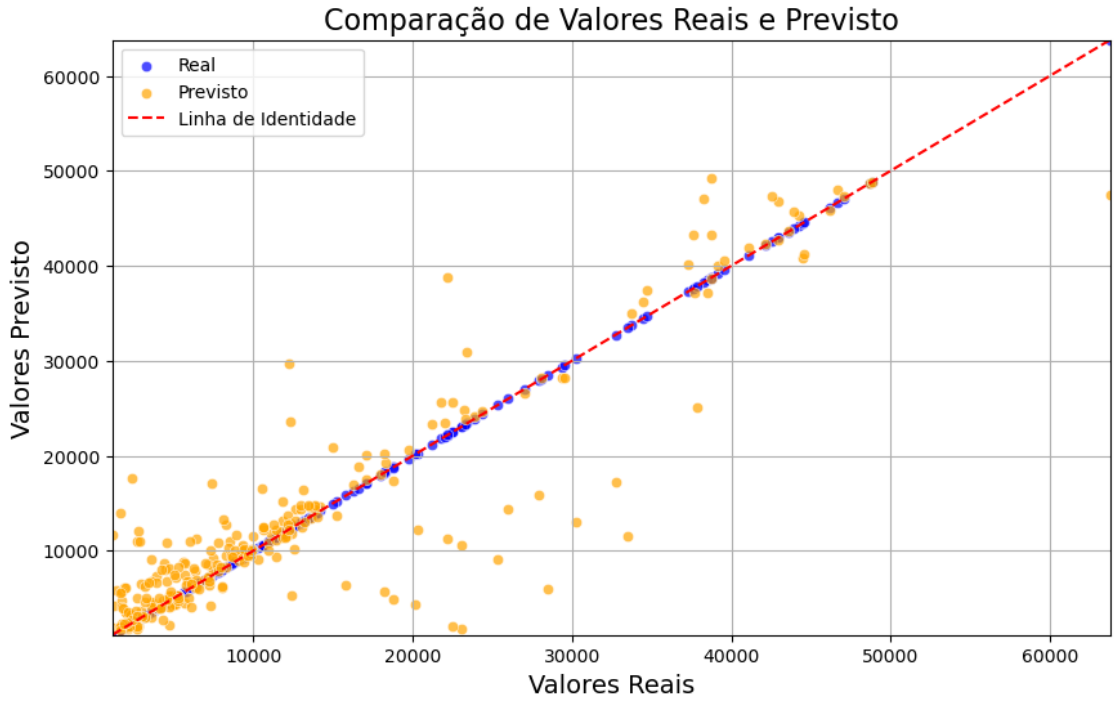

Duração
2.03
0.03
122.74
0.02
0.01
0.02
0.02
0.36
Duração
Modelo
SVR
Decision Tree Regressor
MLP Regressor
Ridge
Elastic Net
Linear Regression
Lasso
Random Forest Regressor

[22]:
```
real = df_results[2].loc[df_results[2]['Modelo'] == 'Random Forest␣
 ↪Regressor']["Real"].values[0]
previsto = df_results[2].loc[df_results[2]['Modelo'] == 'Random Forest␣
 ↪Regressor']["Previsto"].values[0]

df_real_prev = pd.DataFrame({
    'Real': real,
    'Previsto': previsto
})
df_real_prev.head()
```

[22]:

	Real	Previsto
0	9095.06825	8932.412398
1	5272.17580	5297.352157
2	29330.98315	28188.011885
3	9301.89355	10349.014236
4	33750.29180	35016.111143

[23]:
```
# Criar o gráfico de dispersão
plt.figure(figsize=(10, 6))

# Plotar os valores reais
sns.scatterplot(data=df_real_prev, x='Real', y='Real', color='blue',␣
 ↪label='Real', alpha=0.7)

# Plotar os valores previstos
sns.scatterplot(data=df_real_prev, x='Real', y='Previsto', color='orange',␣
 ↪label='Previsto', alpha=0.7)

# Adicionando a linha de identidade
plt.plot([df_real_prev['Real'].min(), df_real_prev['Real'].max()],
         [df_real_prev['Real'].min(), df_real_prev['Real'].max()],
         color='red', linestyle='--', label='Linha de Identidade')

# Personalizando o gráfico
plt.title('Comparação de Valores Reais e Previsto', fontsize=16)
plt.xlabel('Valores Reais', fontsize=14)
plt.ylabel('Valores Previsto', fontsize=14)
plt.xlim(df_real_prev['Real'].min(), df_real_prev['Real'].max())
plt.ylim(df_real_prev['Real'].min(), df_real_prev['Real'].max())
plt.legend()
plt.grid()
plt.show()
```



[24]:
```
p_values = (
    df_results[2] # Dataset com os resultados da abordagem sem Scaler
    .loc[df_results[2]['Modelo'] == 'Random Forest Regressor']['P-valor F'] # Filtrando o modelo e selecionando atributo
    .to_numpy()[0] # Converte para array e retorna a primeira posição
    .to_numpy() # Converte a primeira posição para array
)

for var, p in zip(variaveis_independentes, p_values):
    print(f'O p-valor da variável {var} é: {p}. \n')
```

```
O p-valor da variável age é: 3.714303547683245e-71.

O p-valor da variável bmi é: 2.7921352067083835e-24.

O p-valor da variável smoker é: 2.7093354572758147e-287.
```

De acordo com as métricas avaliadas, os modelos **MLP Regressor** e o **Random Forest** foram os que apresentaram os **melhores resultados**, embora o MLP tenha performado bem na primeira abordagem, o **Random Forest Regressor se destaca em todas elas** (StandardScaler, MinMaxScaler e sem scaler), se mantendo mais consistente em relação às métricas (MAE, MSE, RMSE e R^2), além de apresentar uma **melhor performance**, sendo o **modelo mais adequado dentre os testados**.

Abaixo temos as métricas do Random Forest para as 3 abordagens:

1. Random Forest com StandardScaler:

- MAE = 2811.365
- MSE = 26651682.167
- RMSE = 5162.527
- R^2 = 0.828

2. Random Forest com MinMaxScaler

- MAE = 2826.220
- MSE = 26666173.323
- RMSE = 5163.930
- R^2 = 0.828

3. Random Forest sem Scaler

- MAE = 2820.420
- MSE = 26613362.823
- RMSE = 5158.814
- R^2 = 0.829

Conclusão

A **Random Forest sem Scaler é a melhor opção**, pois tem um R^2 de 0,829, ou seja, **explica 82,9% dos dados**. Além de ter o **menor MSE e RMSE**, que indica que a **variabilidade entre os valores previstos e reais possuem o menor erro**.

Esses indicadores sugerem que o modelo está proporcionando **previsões mais precisas**.

Considerando que **75% da concentração dos valores de custos do plano** estão **entre 1.121,87 e 16639.91**, a comparação dos valores **previstos x reais** mostra **melhor resultado** quando o custo está **dentro desse intervalo**.