

Prova eEDB-002/2024-3

Aluno: João Guilherme Squinelato de Melo

Código: Disponível no repositório github Squinelato/eedb-002-2024-3-exam

Configurando ambiente

Importando bibliotecas necessárias para o desenvolvimento da prova. Os principais pacotes são `sklearn` para treinamento do modelo; `numpy` e `pandas` para manipulação de dados e; `seaborn` e `matplotlib` para geração de visualizações.

```
In [ ]: from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import FunctionTransformer
from sklearn.linear_model import Lasso
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score

from datetime import datetime

import joblib
import numpy as np
import pandas as pd
import helper as hp
import seaborn as sns
import matplotlib.pyplot as plt
```

OBS: Para mais informações sobre todas as bibliotecas utilizadas para o desenvolvimento desta prova, consulte o arquivo `environment.yml` disponível do repositório github mencionado acima

Leitura dos dados

Importando dados em formato `.xlsx` para realizar a prova, considerando:

- Importando apenas a primeira pasta do arquivo "Question 1";
- ignorando as primeiras 18 linhas do arquivo (contando a partir de 0);
- Discriminando os valores faltantes "<NA>" e "?";
- Renomeando as colunas para *snake case*;
- Selectionando apenas as colunas desejadas: "date", "total_sales" e "temperature_fahrenheit";
- Convertendo o tipo das colunas para um mais apropriado.

```
In [ ]: ice_cream_sales = pd.read_excel(
    io='../../data/eDB-002 - Exercicio - REG LINENAR.xlsx',
    sheet_name='Question 1',
    skiprows=18,
    na_filter=True,
    na_values=['<NA>', '?'],
    names=['date', 'temperature_fahrenheit', 'total_sales', 'temperature_celsius'],
    usecols=['date', 'total_sales', 'temperature_fahrenheit'],
    dtype={
        'date': 'datetime64[ns]',
        'total_sales': pd.Float64Dtype(),
        'temperature_fahrenheit': pd.Float64Dtype(),
    },
)
```

Removendo linhas com dados faltantes:

```
In [ ]: ice_cream_sales.dropna(inplace=True)
```

Visualizando dados lidos do arquivo `.xlsx`:

```
In [ ]: ice_cream_sales.head()
```

	date	temperature_fahrenheit	total_sales
0	2004-03-01	65.100035	38911.0
1	2004-03-02	62.832313	36337.258058
2	2004-03-03	62.088516	35728.110238
3	2004-03-04	65.105432	36945.386701
4	2004-03-05	63.011221	36335.166206

Conferindo tipo dos dados:

```
In [ ]: ice_cream_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7175 entries, 0 to 7174
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   date              7175 non-null    datetime64[ns]
 1   temperature_fahrenheit 7175 non-null    float64 
 2   total_sales        7175 non-null    float64 
dtypes: float64(2), datetime64[ns](1)
memory usage: 238.2 KB
```

Engenharia de características

1) Criação de variáveis

1.1) Criando variável chamada `day_of_week` e populando-a com os dias da semana (*Monday, Tuesday etc*)

```
In [ ]: ice_cream_sales['day_of_week'] = ice_cream_sales['date'].dt.strftime('%A')
```

Bônus: Criando variável `week_of_month` para calcular a semana do mês (1th, 2th, 3th, 4th ou 5th Week), a fim de localizar os dados entre o começo, meio, ou fim de mês.

```
In [ ]: ice_cream_sales['week_of_month'] = ice_cream_sales['date'].dt.day // 7 + 1
ice_cream_sales['week_of_month'] = ice_cream_sales['week_of_month'].astype(str) + 'th Week'
```

1.2) Criando variável `month_of_year` e populando-a com (*January, February, March etc*)

```
In [ ]: ice_cream_sales['month_of_year'] = ice_cream_sales['date'].dt.strftime('%B')
```

Bônus: Criando variável `season` para determinar a estação (*Autumn, Winter, Spring ou Summer*) com base na data de aferição.

```
In [ ]: ice_cream_sales['season'] = ice_cream_sales['date'].apply(hp.get_season_as_string)
```

Bônus: Criando a variável `temperature_celsius` com a temperatura em graus celsius (a partir da temperatura em fahrenheit) sem arredondamentos

```
In [ ]: ice_cream_sales['temperature_celsius'] = (ice_cream_sales['temperature_fahrenheit'] - 32) / 1.8
```

Bônus: Criando a variável `30_days_rolling_mean_sales` que calcula a média móvel de 30 dias do valor de vendas de sorvete.

```
In [ ]: ice_cream_sales['30_days_rolling_mean_sales'] = ice_cream_sales['total_sales'].rolling(30).mean()
```

Bônus: Criando variável `30_days_rolling_mean_temperature` que calcula a média móvel de 30 dias da temperatura em celsius.

```
In [ ]: ice_cream_sales['30_days_rolling_mean_temperature'] = ice_cream_sales['temperature_celsius'].rolling(30).mean()
```

Conjunto de dados com novas colunas criadas:

```
In [ ]: ice_cream_sales.tail()
```

```
Out[ ]:   date  temperature_fahrenheit  total_sales  day_of_week  week_of_month  month_of_year  season  temperature_celsius  30_days_rolling_mean_sales
7170  2023-10-18          65.003948  38978.352433  Wednesday      3th Week    October  Spring          18.335526            38678.442478
7171  2023-10-19          66.153564  38974.683053  Thursday       3th Week    October  Spring          18.974202            38675.949082
7172  2023-10-20          63.062093  38346.656668  Friday        3th Week    October  Spring          17.256718            38652.604553
7173  2023-10-21          65.856644  38949.900053  Saturday       4th Week    October  Spring          18.809247            38649.442049
7174  2023-10-22          71.161566  40465.584049  Sunday        4th Week    October  Spring          21.756425            38717.201231
```

Análise exploratória

2) Análises univariadas

2.1) Análises descritivas

Calculando a quantidade de registros, média, desvio padrão, mínimo, máximo e os quartis para as variáveis quantitativas contínuas, i.e., `total_sales` e `temperature_celsius`.

```
In [ ]: ice_cream_sales.drop(
    columns=[
        'date',
        'day_of_week',
        'month_of_year',
        'week_of_month',
        'temperature_fahrenheit',
        'season',
        '30_days_rolling_mean_sales',
        '30_days_rolling_mean_temperature'
    ]
).describe()
```

```
Out[ ]:
```

	total_sales	temperature_celsius
count	7175.0	7175.0
mean	40154.737143	23.282843
std	5642.698863	7.590269
min	28238.199588	5.454535
25%	36365.003731	16.723736
50%	39052.56443	23.50444
75%	45451.687339	30.056367
max	55022.578795	37.155757

Análise: vê-se que quanto ao total de vendas, os valores se concentram em uma média de 40154.737143 com um desvio padrão de 5642.698863, com uma mediana próxima à media, com o valor de 39052.56443. O terceiro Quartil $Q3$ encontra-se dentro da variabilidade descrita pelo desvio padrão, assim como o o primeiro quartil $Q1$.

Já quanto à temperatura em graus celcius, observa-se uma média de 23.282843 com um desvio padrão de 7.590269. Idem ao total de vendas, vê-se uma mediana (23.50444) próxima à média, com o primeiro e terceiro quartis $Q1$ e $Q3$ dentro da variabilidade descrita pelo desvio padrão.

Em ambas as variáveis apresentadas, vê-se baixa influência de valores extremos (*outliers*), uma vez que os valores de média e mediada são próximos. Por fim, ambas as variáveis apresenam valores de máximo e mínimo, respectiva e relativamente próximos a $Q3$ e $Q1$.

2.2) Análises graficas

Visualização sobre a distribuição de vendas de sorvete por meio de um gráfico de violino com boxplot.

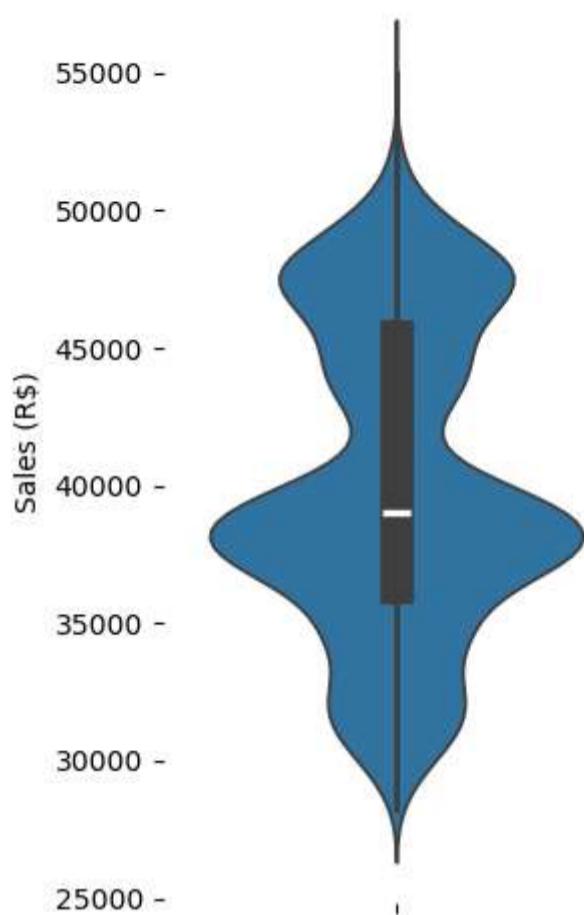
```
In [ ]: plt.subplots(figsize=(3, 6))

sns.violinplot(
    data=ice_cream_sales,
    y='total_sales',
    inner_kws=dict(box_width=12, whis_width=2)
)

plt.ylabel('Sales (R$)')
plt.title('Distribution of Ice Cream Sales')

sns.despine(left=True, bottom=True)
```

Distribution of Ice Cream Sales



Análise: observa-se concentração de valores nas faixas de 35000 a 40000, assim como entre 45000 e 50000 reais.

Visualização sobre a distribuição de temperaturas por meio de um gráfico de violino com boxplot.

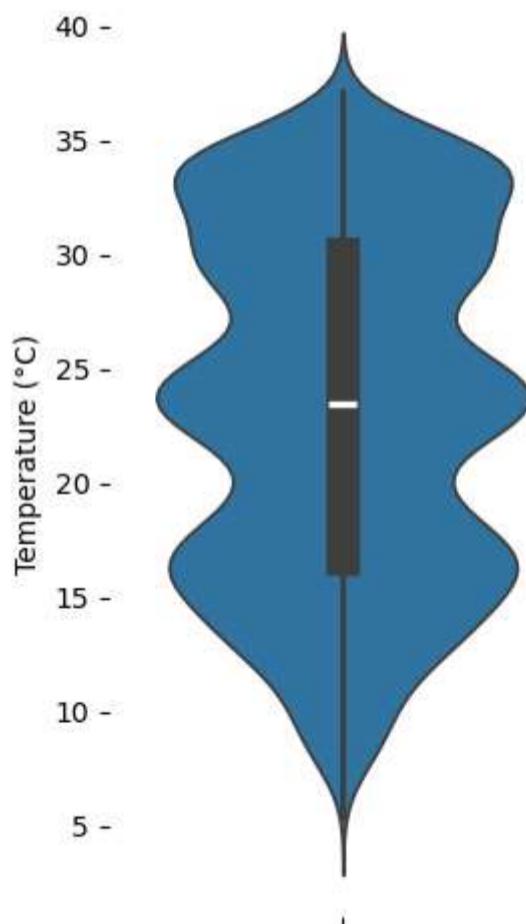
```
In [ ]: plt.subplots(figsize=(3, 6))

sns.violinplot(
    data=ice_cream_sales,
    y='temperature_celsius',
    inner_kws=dict(box_width=12, whis_width=2)
)

plt.ylabel('Temperature (°C)')
plt.suptitle('Distribution of Temperature')

sns.despine(left=True, bottom=True)
```

Distribution of Temperature



Análise: observa-se concentração de valores nas faixas de 15 a 20, 22 a 25 e 32 a 35 °C.

3) Análises bivariadas

3.1) Análise de vendas em relação a datas do ano

Visualização sobre total de vendas em relação ao mês do ano. Neste contexto foi utilizado um gráfico de violino em que a linha tracejada ao meio representa a mediana; linha a cima, o terceiro quartil; linha abaixo, primeiro quartil.

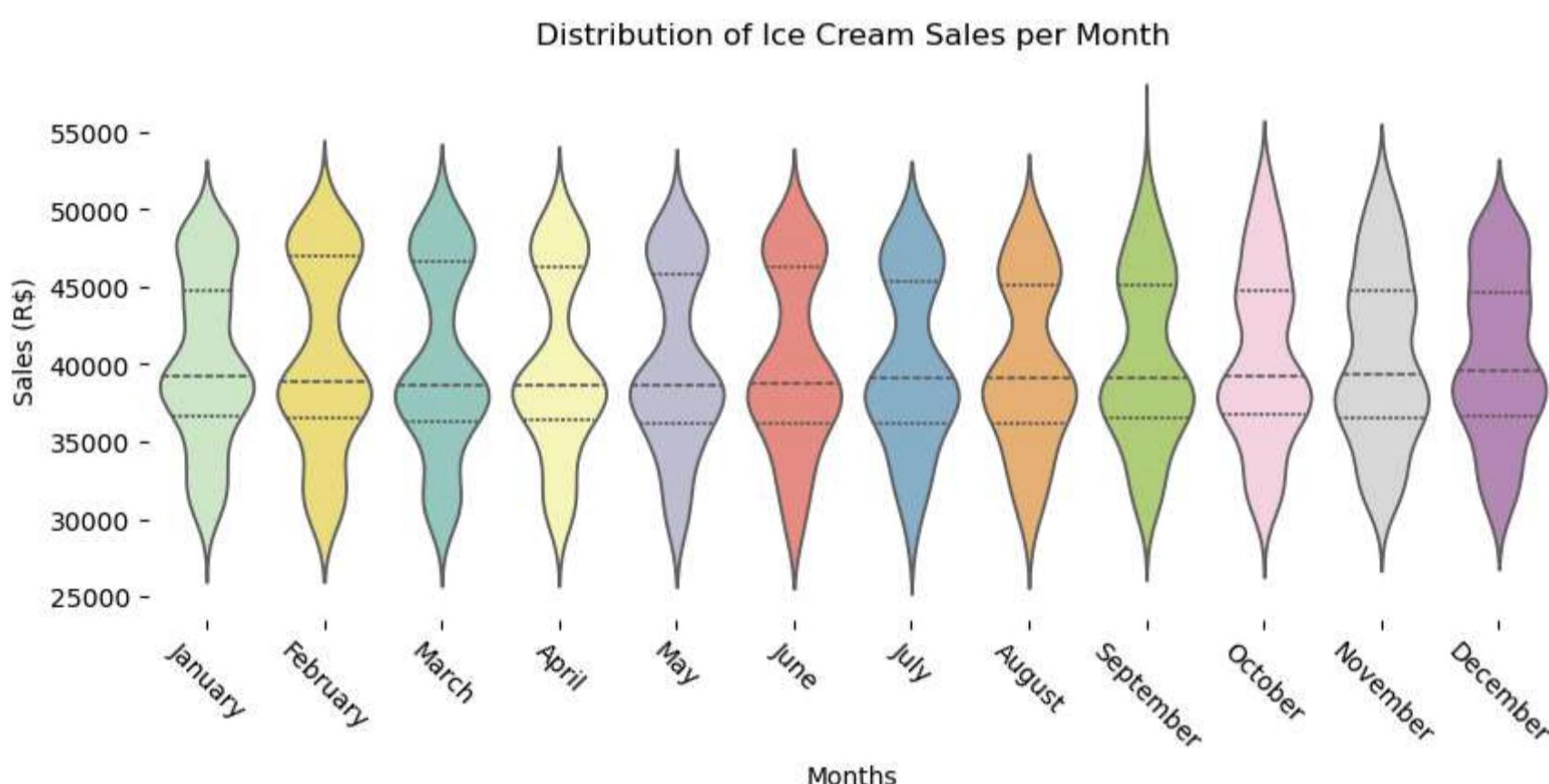
```
In [ ]: plt.subplots(figsize=(10, 4))

sns.violinplot(
    data=ice_cream_sales,
    x='month_of_year',
    y='total_sales',
    hue='month_of_year',
    palette='Set3',
    legend=False,
    inner='quart',
    order=[

        'January',
        'February',
        'March',
        'April',
        'May',
        'June',
        'July',
        'August',
        'September',
        'October',
        'November',
        'December'
    ]
)

plt.xticks(rotation=-45)
plt.xlabel('Months')
plt.ylabel('Sales (R$)')
plt.title('Distribution of Ice Cream Sales per Month')

sns.despine(left=True, bottom=True)
```



Análise: Não observou-se variação significativa de $Q1$ nem mediana no total de vendas ao longo dos meses, apenas variação de $Q3$ decrescendo levemente de Fevereiro à Janeiro.

Visualização sobre total de vendas em relação ao dia da semana. Neste contexto foi utilizado um gráfico de violino em que a linha tracejada ao meio representa a mediana; linha a cima, o terceiro quartil; linha abaixo, primeiro quartil.

```
In [ ]: plt.subplots(figsize=(8, 4))

sns.violinplot(
    data=ice_cream_sales,
    x='day_of_week',
    y='total_sales',
    hue='day_of_week',
    palette='Set3',
    legend=False,
    inner='quart',
    order=[

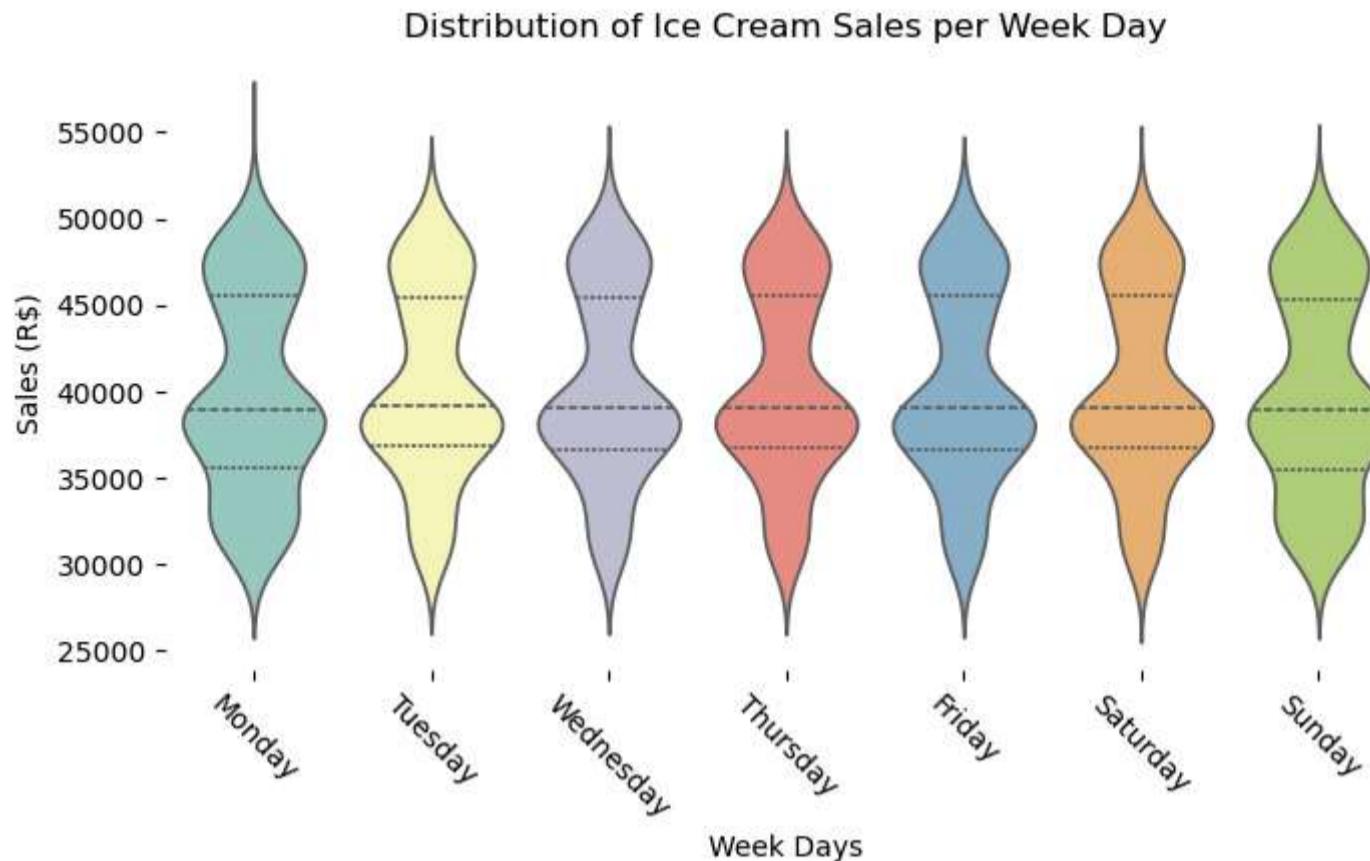
        'Monday',
        'Tuesday',
        'Wednesday',
        'Thursday',
        'Friday',
        'Saturday',
        'Sunday'
    ]
)
```

```

)
plt.xticks(rotation=-45)
plt.xlabel('Week Days')
plt.ylabel('Sales (R$)')
plt.title('Distribution of Ice Cream Sales per Week Day')

sns.despine(left=True, bottom=True)

```



Análise: Não observou-se variação significativa de Q_1 , mediana nem Q_3 no total de vendas ao longo dos dias da semana.

Analise sobre total de vendas em relação às semanas de mês. Neste contexto foi utilizado um gráfico de violino em que a linha tracejada ao meio representa a mediana; linha a cima o terceiro quartil; linha abaixo, primeiro quartil.

```

In [ ]: plt.subplots(figsize=(8, 4))

sns.violinplot(
    data=ice_cream_sales,
    x='week_of_month',
    y='total_sales',
    hue='week_of_month',
    palette='Set3',
    legend=False,
    inner='quart'
)

plt.xlabel('Weeks in a Month')
plt.ylabel('Sales (R$)')
plt.title('Distribution of Ice Cream Sales per Weeks in a Month')

sns.despine(left=True, bottom=True)

```



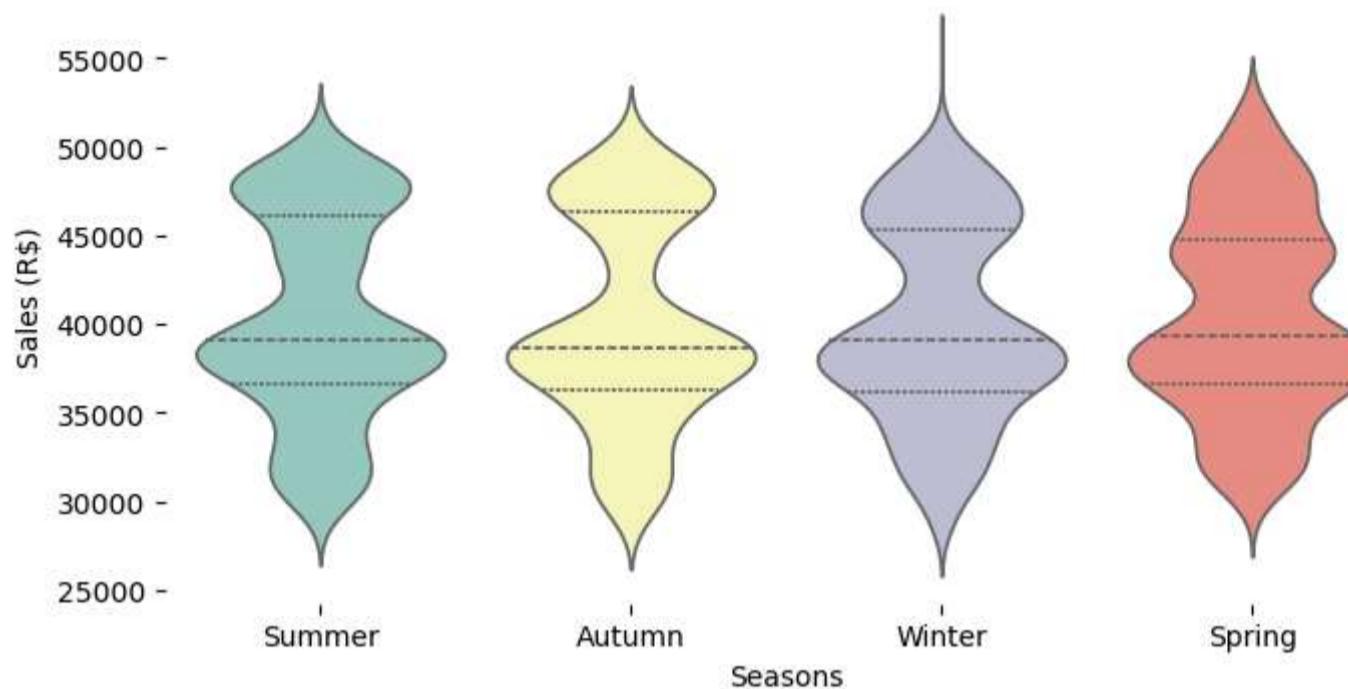
Análise: Não observou-se variação significativa de Q_1 , mediana nem Q_3 no total de vendas ao longo das semanas.

Visualização sobre total de vendas em relação às estações do ano. Neste contexto foi utilizado um gráfico de violino em que a linha tracejada ao meio representa a mediana; linha a cima o terceiro quartil; linha abaixo, primeiro quartil.

```
In [ ]: plt.subplots(figsize=(8, 4))
```

```
sns.violinplot(  
    data=ice_cream_sales,  
    x='season',  
    y='total_sales',  
    hue='season',  
    palette='Set3',  
    legend=False,  
    inner='quart'  
)  
  
plt.xlabel('Seasons')  
plt.ylabel('Sales (R$)')  
plt.title('Distribution of Ice Cream Sales per Season')  
  
sns.despine(left=True, bottom=True)
```

Distribution of Ice Cream Sales per Season



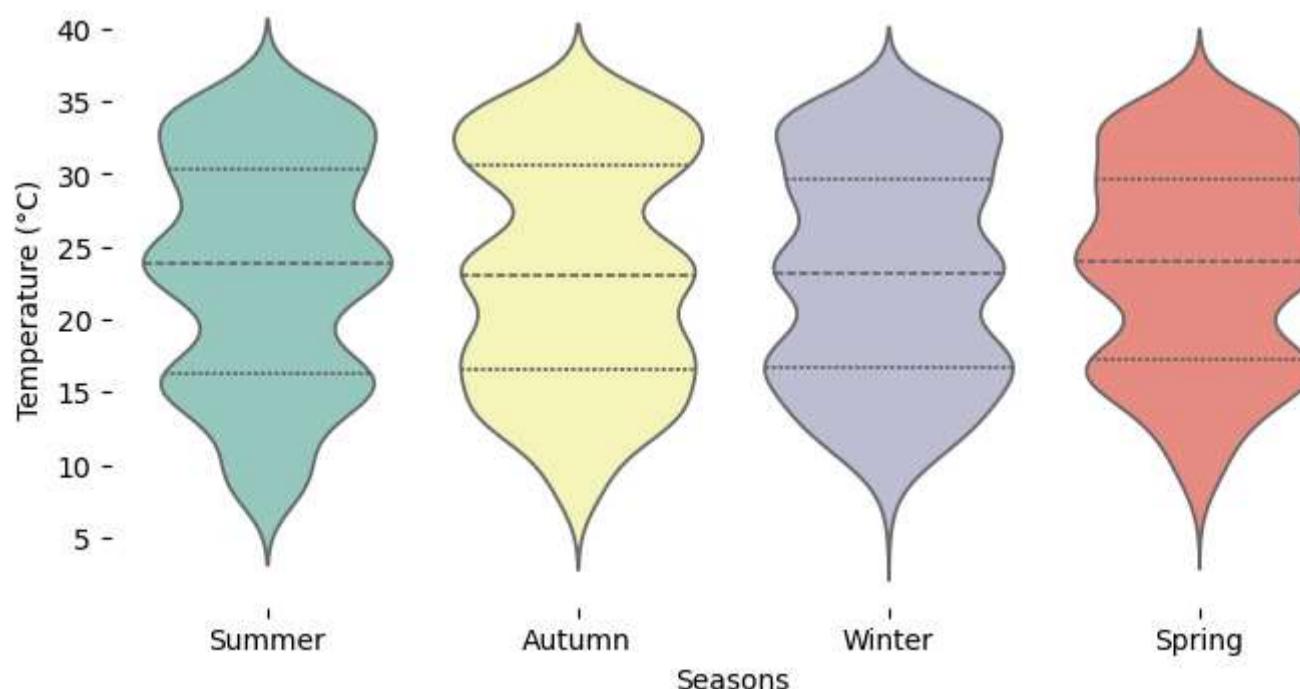
Análise: Não observou-se variação significativa de $Q1$ nem mediana no total de vendas ao longo das estações, apenas um maior valor de $Q3$ no verão e outono, ante a um menor valor para o inverno e primavera.

Visualização sobre a temperatura em relação às estações do ano. Neste contexto foi utilizado um gráfico de violino em que a linha tracejada ao meio representa a mediana; linha a cima o terceiro quartil; linha abaixo, primeiro quartil.

```
In [ ]: plt.subplots(figsize=(8, 4))
```

```
sns.violinplot(  
    data=ice_cream_sales,  
    x='season',  
    y='temperature_celsius',  
    hue='season',  
    palette='Set3',  
    legend=False,  
    inner='quart'  
)  
  
plt.xlabel('Seasons')  
plt.ylabel('Temperature (°C)')  
plt.title('Distribution of Temperature per Season')  
  
sns.despine(left=True, bottom=True)
```

Distribution of Temperature per Season

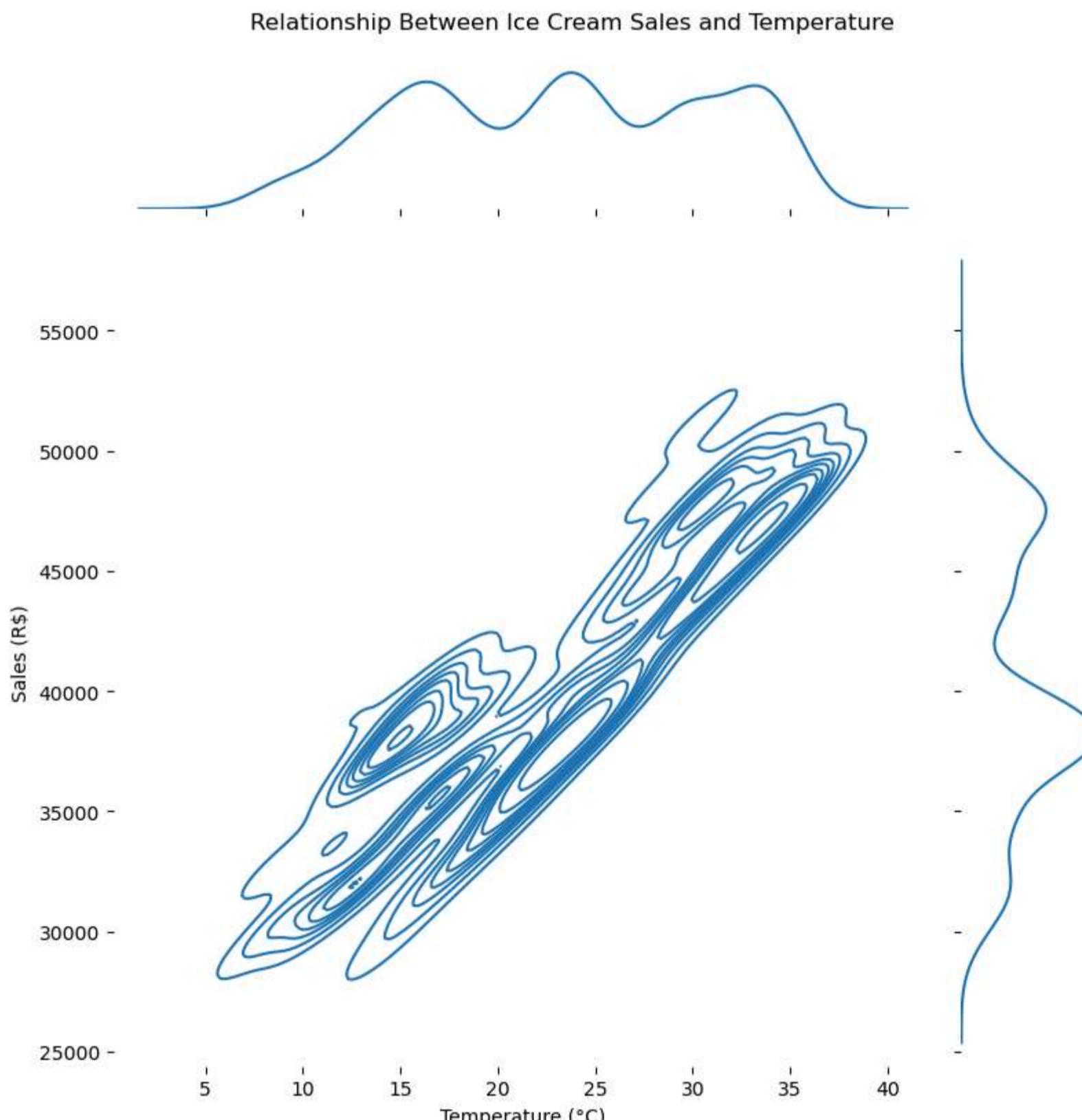


Análise: Não observou-se variação significativa de $Q1$, mediana nem $Q3$ na temperatura ao longo das estações, o que indica que os dados foram coletados em uma região próxima à linha do equador.

3.2) Analise graficas bivariadas quanto à vendas e temperatura

Visualização por meio de um gráfico de dispersão, com sua variante por contornos, a qual representa os pontos (venda por temperatura) com curvas de nível; ao passo que quanto maior a quantidade de contornos internos, maior concentração de pontos. Além disso, cada variável contínua tem sua função de densidade calculada para adicionar informações que facilitam a análise uni e bivariada.

```
In [ ]: sns.jointplot(  
    data=ice_cream_sales,  
    y='total_sales',  
    x='temperature_celsius',  
    kind='kde',  
    height=8,  
)  
  
plt.xlabel('Temperature (°C)')  
plt.ylabel('Sales (R$)')  
plt.suptitle('Relationship Between Ice Cream Sales and Temperature', y=1.03)  
  
sns.despine(left=True, bottom=True)
```



Análise: Observa-se claramente uma correlação positiva entre a temperatura e a venda de sorvetes, um vez que à medida que a temperatura cresce, também cresce o faturamento da venda de sorvetes. Além disso, por meio das curvas de nível, é possível perceber regiões onde há um grande número de amostras, ocorrendo justamente nas regiões de coincidência de temperaturas e faturamentos, num processo que culmina na formação de picos. Isto significa que nestas regiões certas temperaturas tem maior probabilidade de coincidirem com certos faturamentos.

bônus: análise bivariada da quantidade vendas em relação ao tempo.

```
In [ ]: plt.subplots(figsize=(20,5))  
  
sns.lineplot(  
    data=ice_cream_sales,  
    y='total_sales',  
    x='date',
```

```

        label='Ice Cream Sales',
    )

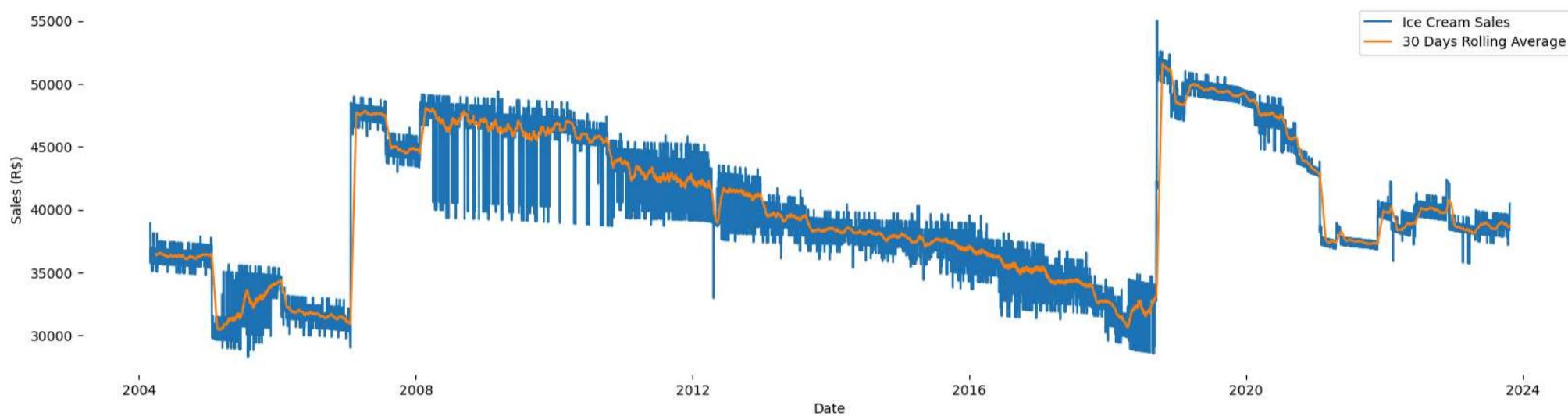
sns.lineplot(
    data=ice_cream_sales,
    y='30_days_rolling_mean_sales',
    x='date',
    label='30 Days Rolling Average',
)

plt.xlabel('Date')
plt.ylabel('Sales (R$)')
plt.suptitle('Ice Cream Sales over the years', y=1.03)

sns.despine(left=True, bottom=True)

```

Ice Cream Sales over the years



Análise: Com a media móvel de 30 dias, observa-se que o total de vendas apresenta alguns padrões ao longo dos anos, como por exemplo, que de 2007 a 2019 houve um declínio constante nas vendas. Em contrapartida, em outros momentos, certas lacunas são observadas, como entre 2019 e 2020, assim como entre 2006 e 2007, onde saltos positivos no faturamento foram identificados.

bônus: análise bivariada da temperature ($^{\circ}\text{C}$) em relação ao tempo.

```
In [ ]: plt.subplots(figsize=(20,5))

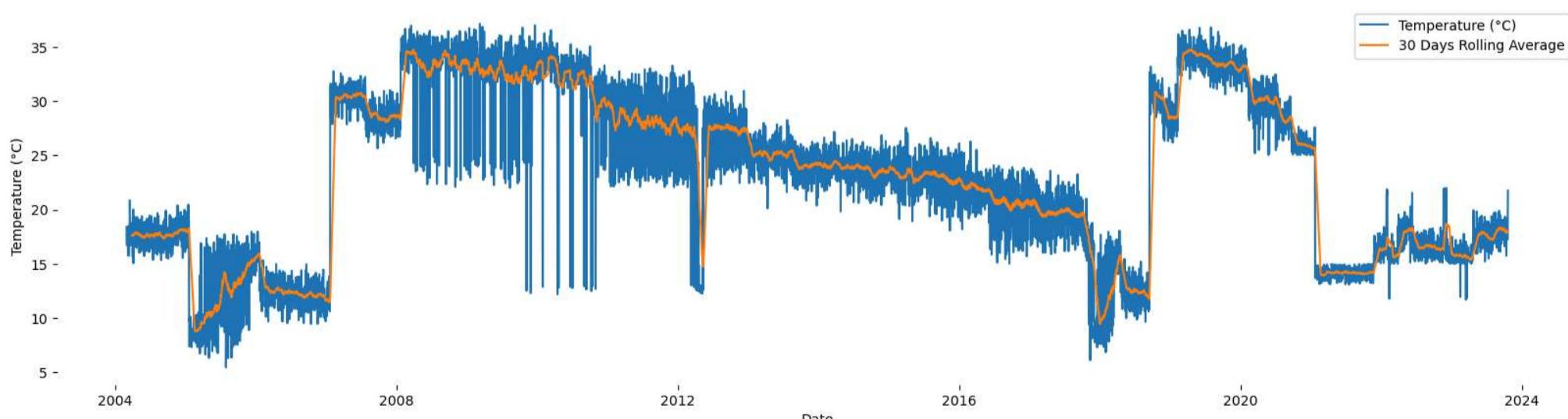
sns.lineplot(
    data=ice_cream_sales,
    y='temperature_celsius',
    x='date',
    label='Temperature ( $^{\circ}\text{C}$ )',
)

sns.lineplot(
    data=ice_cream_sales,
    y='30_days_rolling_mean_temperature',
    x='date',
    label='30 Days Rolling Average',
)

plt.xlabel('Date')
plt.ylabel('Temperature ( $^{\circ}\text{C}$ )')
plt.suptitle('Temperature over the years', y=1.03)

sns.despine(left=True, bottom=True)
```

Temperature over the years



Análise: Com a media móvel de 30 dias, observa-se com temperatura ao longo dos anos um comportamento parecido com o faturamento, com momentos de declínio e lacunas em períodos de tempo similares aos observados com o faturamento.

3.3) Calculando correlação entre as variaveis `temperature_celsius` e `total_sales`

```
In [ ]: numeric_variabels = ice_cream_sales.drop(
    columns=[
        'day_of_week',
        'month_of_year',
        'week_of_month',
        'season',
        '30_days_rolling_mean_sales'
    ]
)
correlations = numeric_variabels.corr(method='pearson')
correlations.loc['total_sales', 'temperature_celsius']
```

Out[]: 0.9040588653750967

Análise: A correlação entre a temperatura e o total de vendas foi de 0.9040588653750967, a qual é considerada uma correlação positiva e forte entre estas duas variáveis, isto é, se uma cresce, a outra a acompanha.

Análise preditiva

4) Escolha do modelo de regressão linear

Foi utilizado o modelo de **regressão Lasso** (*Least Absolute Shrinkage and Selection Operator*), o qual consiste em aplicar uma regressão linear adicionando em sua função custo uma penalização ℓ_1 para os coeficientes β . Isto faz com que os coeficientes lineares β não significativos para predizer a variável dependente y sejam reduzidos à zero. Isto acontece pois a penalização ℓ_1 penaliza valores altos para os coeficientes β , o que pode ser intensificado pela constante α ; isto é, quanto maior o valor de α , maior a penalização. Além disso, caso o valor de α tenda ao infinito, assintoticamente a função tenderá a um valor constante, o que representa, na prática, que o modelo se torna insensível à mudanças nas variáveis independentes (x).

Dito isto, a escolha pela regressão Lasso deu-se um vez que o modelo de predição receberá cinco variáveis independentes, quais sejam: temperatura ($^{\circ}\text{C}$), dia da semana, semana do mês, mês do ano e estação. Logo, a fim de encontrar quais variáveis independentes são mais significativas para o modelo, optou-se pela regressão Lasso em virtude de sua capacidade de selecionar as variáveis mais significativas, descartando as variáveis menos significativas, ou reduzindo o valor de seus pesos β para o cálculo da variável dependente.

Assim, para além de encontrar a função que preverá o total de vendas de sorvete, a regressão Lasso se utiliza da equação abaixo para encontrar as variáveis independentes mais significativas para o modelo, por meio da otimização dos coeficientes β que minimizem o valor do custo J . Este custo é então calculado pela soma dos resíduos ao quadrado, isto é, a diferença entre o valor predito \hat{y} e o valor real y , acrescido da penalização ℓ_1 dos coeficientes lineares β . Além disso, especificamente na implementação da regressão Lasso da biblioteca [sklearn](#), a soma dos resíduos ao quadrado é normalizada pela quantidade n de instâncias das variáveis independentes x , para que o custo não divirja entre conjuntos de dados com tamanhos distintos. Outrossim, a quantidade de instâncias n é multiplicada por 2 para simplificar o cálculo da derivada da função custo J .

Por fim, a quantidade de coeficientes lineares β é representada por p , o qual é 5, isto é, um coeficiente para cada variável independente; em que o coeficiente linear β_0 não é penalizado e, portanto, não é considerado pela penalização ℓ_1 .

$$J(\beta) = \min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2n} \|y - \beta_0 - X\beta\|_2^2 + \alpha \|\beta\|_1 \right\}$$

Treinando modelo de regressão Lasso

Selecionando a matriz de variáveis independentes X , contendo a temperatura em graus celsius e data de aferição, e o vetor com a variável dependente y representando o total de vendas de sorvetes.

```
In [ ]: X = ice_cream_sales.loc[:, ['date', 'temperature_celsius']]
y = ice_cream_sales.loc[:, ['total_sales']].values.ravel()
```

Dividindo dados entre conjunto de treinamento, com 80% dos dados; teste, 20%.

```
In [ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Criando função para extrair dia da semana, semana do mês, mês e estação do ano da data de aferição.

```
In [ ]: def get_date_features(X: pd.DataFrame) -> pd.DataFrame:
    """
    A function to extract the day of week, week of the month, month of
    the year and the season from a given date.
    """
    X['day_of_week'] = X['date'].dt.day_of_week + 1
    X['week_of_month'] = X['date'].dt.day // 7 + 1
    X['month_of_year'] = X['date'].dt.month
    X['season'] = X['date'].apply(hp.get_season_as_number)
    X = X.drop(columns=['date'])

    return X

date_features = FunctionTransformer(get_date_features, validate=False)
```

Criando pipeline de transformações para os dados, para automatizar a extração de características baseadas na data de aferição.

```
In [ ]: data_pipeline = Pipeline([
    ('date_features', date_features)
])
```

Ajustando dados de treinamento e aplicando transformações, posteriormente transformando dados de teste.

```
In [ ]: X_train_pre = data_pipeline.fit_transform(X_train)
X_test_pre = data_pipeline.transform(X_test)
```

Treinando modelo de regressão Lasso com penalização ℓ_1 , com α variando de 0.1 a 1000 com intervalos de 0.1 resultando em 10000 valores de α para testar.

```
In [ ]: grid_serch = GridSearchCV(
    estimator=Lasso(),
    param_grid={'alpha': np.arange(0.1, 1000.1, 0.1)},
    cv=10,
    scoring='neg_root_mean_squared_error',
    verbose=3,
    n_jobs=-1,
    refit=True
)
```

Para obter-se o modelo mais otimizado variando α , a técnica de *cross validation* foi empregada com 10 *folds*, isto é, para cada valor de α treinou-se o modelo com $\frac{9}{10}$ do conjunto de treinamento e o testou-se com o $\frac{1}{10}$ restante dos dados, realizando este processo 10 vezes.

```
In [ ]: grid_serch.fit(X_train_pre, y_train)
```

```
Fitting 10 folds for each of 10000 candidates, totalling 100000 fits
Out[ ]: > GridSearchCV ⓘ ⓘ
      > best_estimator_: Lasso
          > Lasso ⓘ
```

Melhor α encontrado:

```
In [ ]: grid_serch.best_params_
```

```
Out[ ]: {'alpha': 160.6}
```

Salvando melhor modelo.

```
In [ ]: optimal_lasso = grid_serch.best_estimator_
joblib.dump(optimal_lasso, '../model/optimal-lasso.joblib')
```

```
Out[ ]: ['../model/optimal-lasso.joblib']
```

Realizando predições sobre total de vendas:

```
In [ ]: y_train_pred = optimal_lasso.predict(X_train_pre)
y_test_pred = optimal_lasso.predict(X_test_pre)
```

Métrica de R^2 para conjunto de treinamento:

```
In [ ]: r2_score(y_train, y_train_pred)
```

```
Out[ ]: 0.8158148364525213
```

Métrica de R^2 para conjunto de teste:

```
In [ ]: r2_score(y_test, y_test_pred)
```

```
Out[ ]: 0.8230665165155082
```

Valor final dos coeficientes β para cada característica (x) juntamente com coeficiente de linear β_0 :

```
In [ ]: feature_coef = pd.DataFrame({
    'feature_names': ['temperature_celsius', 'day_of_week', 'week_of_month', 'month_of_year', 'season', 'y_intercept'],
    'coefficient': np.append(optimal_lasso.coef_, optimal_lasso.intercept_)
})
feature_coef
```

	feature_names	coefficient
0	temperature_celsius	667.689705
1	day_of_week	-0.000000
2	week_of_month	0.000000
3	month_of_year	-0.000000
4	season	0.000000
5	y_intercept	24616.057692

Modelo obtido

Assim, a função que prevê o valor de vendas \hat{y} foi modelada por meio dos coeficientes angulares: β_t que representa o coeficiente de temperatura; β_d , coeficiente de dia da semana e; β_w , coeficiente de semana do mês; β_m , coeficiente de mês do ano e; β_s , coeficiente de estação do ano. Já o coeficiente linear é representado por β_0 . Naturalmente, as variáveis utilizadas como entrada desta função são a temperatura (x_t), dia da semana (x_d), semana do mês (x_w), mês do ano (x_m) e estação do ano (x_s).

$$\hat{y} = \beta_t * x_t + \beta_d * x_d + \beta_w * x_w + \beta_m * x_m + \beta_s * x_s + \beta_0$$

Após o treinamento do modelo, por meio da técnica de regressão Lasso, os coeficientes obtidos foram:

$$\begin{aligned}\beta_t &= 667.689705 \\ \beta_d &= 0 \\ \beta_w &= 0 \\ \beta_m &= 0 \\ \beta_s &= 0 \\ \beta_0 &= 24616.057692\end{aligned}$$

Ao fim do treinamento, foi obtido um R^2 de 0.8158148364525213 para o conjunto de treinamento e um R^2 de 0.8230665165155082 para o conjunto de teste. O maior valor de R^2 para o conjunto de teste denota que o modelo não apresentou *overfitting*, isto é, o modelo não se especializou no conjunto de treinamento, conseguindo superar sua performance em um conjunto de dados não considerado durante seu treinamento, o que denota boa generalização do modelo.

Inferências

5) Inferindo o faturado para 30/out/23, com a previsão do tempo apontando 18 °C

Criando um função para automatizar a transformação e inferência dos dados.

```
In [ ]: loaded_model = joblib.load('../model/optimal-lasso.joblib')

In [ ]: def predict_sales(input_data: list[list]) -> list[float]:
    """
    A function to prepare the raw data to a proper way for prediction,
    and then predicts.
    """
    instances = pd.DataFrame(data=input_data, columns=['date', 'temperature_celsius'])

    instances['date'] = instances['date'].astype('datetime64[ns]')
    instances['temperature_celsius'] = instances['temperature_celsius'].astype(pd.Float64Dtype())

    instances_preprocessed = data_pipeline.transform(instances)
    prediction = loaded_model.predict(instances_preprocessed)

    return prediction.tolist()
```

```
In [ ]: input_data = [['30-10-2023', 18]]
predict_sales(input_data)
```

```
Out[ ]: [36634.472382105494]
```

Inferência: O faturamento previsto para a data de 30/out/23, considerando uma temperatura de 18 °C, é de 36634.472382105494 reais.

Visualizando a inferência

Visualização da inferência em relação ao faturamento mensal dos últimos 12 meses.

```
In [ ]: plt.subplots(figsize=(20,5))

sns.scatterplot(
    x=[datetime(2023, 10, 30)],
    y=[36634.472382105494],
    color='red',
    marker='X',
```

```

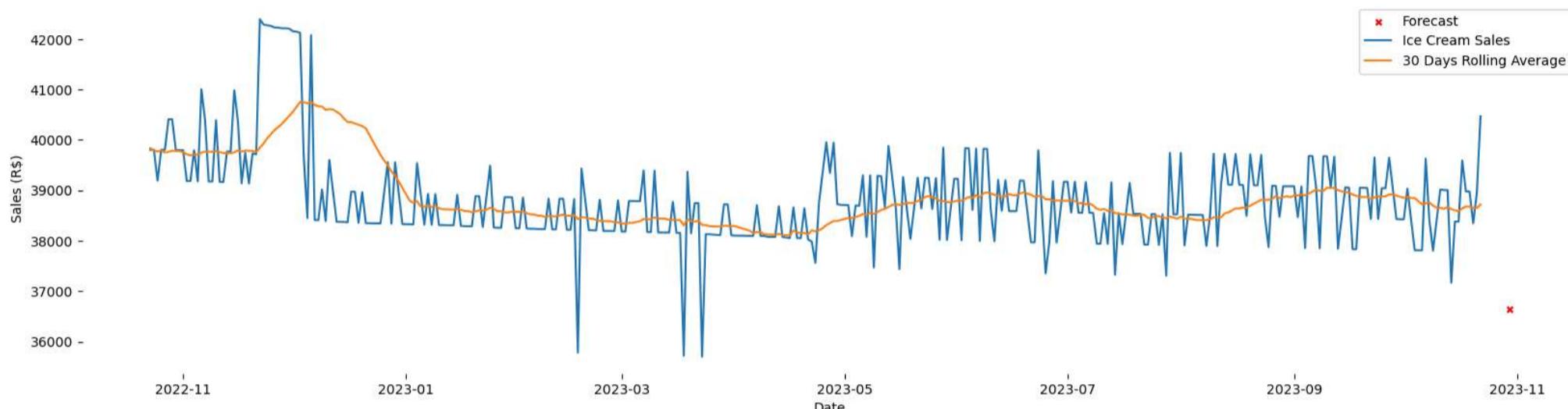
        label='Forecast'
    )

sns.lineplot(
    data=ice_cream_sales.tail(365),
    y='total_sales',
    x='date',
    label='Ice Cream Sales',
)
sns.lineplot(
    data=ice_cream_sales.tail(365),
    y='30_days_rolling_mean_sales',
    x='date',
    label='30 Days Rolling Average',
)
plt.xlabel('Date')
plt.ylabel('Sales (R$)')
plt.suptitle('Ice Cream Sales Forecast for October 30, 2023')

sns.despine(left=True, bottom=True)

```

Ice Cream Sales Forecast for October 30, 2023



Análise: Vê-se que o total de vendas previsto (forecast) para o dia 30 de Outubro de 2023 apresenta um valor à baixo da média móvel de 30 dias, comparando-se com as vendas nos últimos 12 meses.

Visualização da inferência em relação a períodos de tempo específicos.

In []: plt.subplots(figsize=(15, 7))

```

sns.violinplot(
    data=ice_cream_sales,
    x='day_of_week',
    y='total_sales',
    inner='quart',
    order=['Monday'],
    label='Monday Sales',
    color="#8dd3c7",
)

sns.violinplot(
    data=ice_cream_sales,
    x='week_of_month',
    y='total_sales',
    inner='quart',
    order=[5th Week],
    label='5th Week Sales',
    color="#ffffb3",
)

sns.violinplot(
    data=ice_cream_sales,
    x='month_of_year',
    y='total_sales',
    inner='quart',
    order=[October],
    label='October Sales',
    color="#bebada",
)

sns.violinplot(
    data=ice_cream_sales,
    x='season',
    y='total_sales',
    inner='quart',
    order=[Spring],
    label='Spring Sales',
    color="#80b1d3"
)

sns.scatterplot(
    x=['Monday', '5th Week', 'October', 'Spring'],
    y=[36634.472382105494]*4,

```

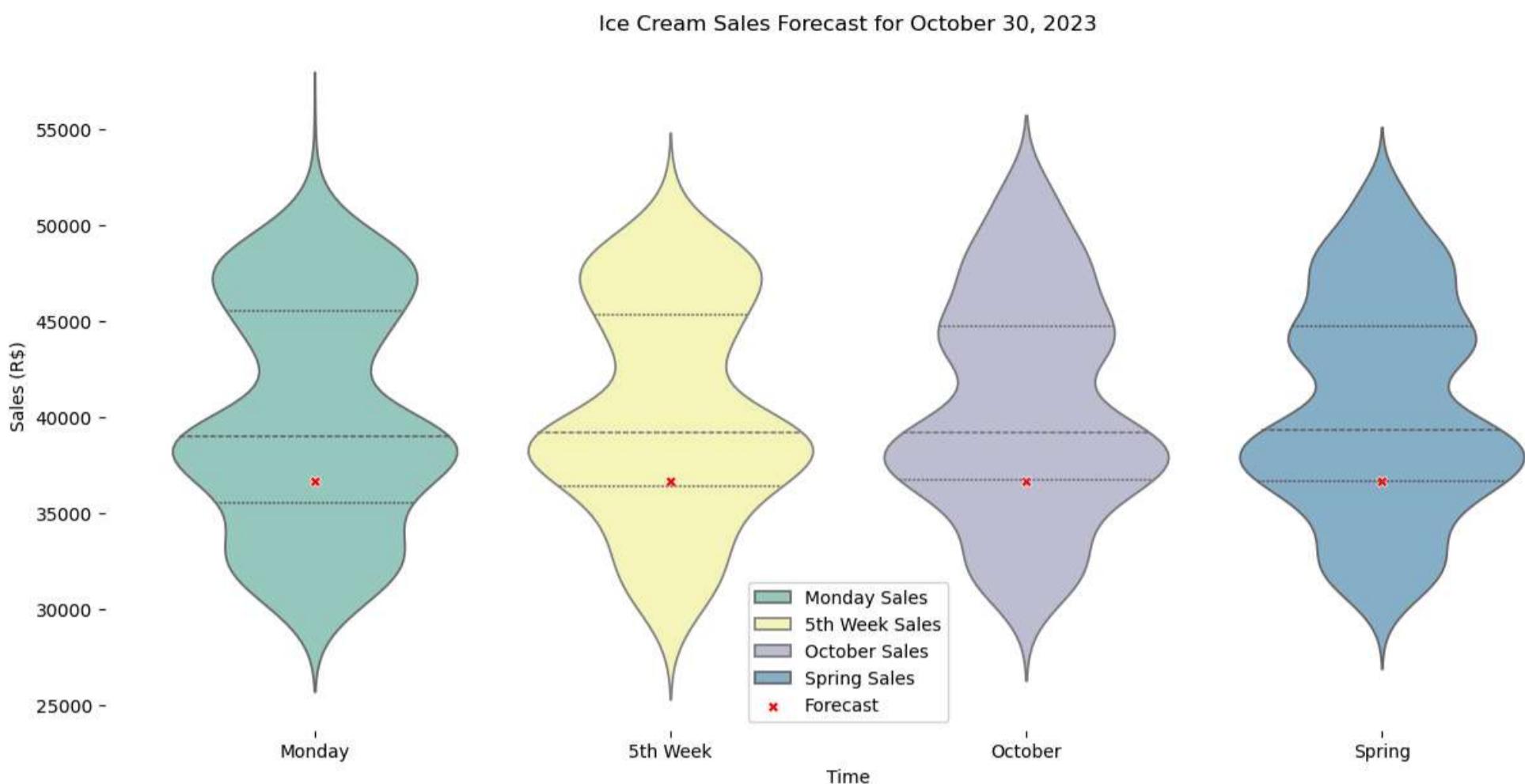
```

        color='red',
        marker='X',
        label='Forecast',
    )

plt.xlabel('Time')
plt.ylabel('Sales (R$)')
plt.title('Ice Cream Sales Forecast for October 30, 2023')

sns.despine(left=True, bottom=True)

```



Análise: Analisando as vendas para períodos de tempo comparáveis com a data de 30 de Outubro, isto é, por se tratar de uma segunda-feira, localizada na quinta semana do mês, no mês de Outubro e na estação da primavera, vê-se que o faturamento para o dia de 30 de Outubro de 2023 apresenta um valor próximo do primeiro quartil Q_1 em todos os períodos de tempo analisados.

Mais especificamente, apenas em relação às segundas-feiras e às quintas semanas dos meses que o valor predito encontra-se um pouco acima do primeiro quartil Q_1 ; porém, em relação ao mês de Outubro e à estação da primavera, o valor predito está abaixo de Q_1 . Por fim, apesar do valor preditor estar dentro da variabilidade prevista, é importante salientar que encontra-se no limiar entre os 25% dos faturamentos mais baixos observados nos períodos de tempo analisados.

Discutindo resultados

6) Conclusão

Com a prova foi possível realizar e exercitar diversos conceitos estatísticos, como análise uni e bivariada, elaboração de visualizações e, enfim, o treinamento de um modelo de regressão para prever o faturamento de vendas de sorvete. Além disso, instigando-se a buscar correlações entre tempo e faturamento, exercitou-se a criação de diversas variáveis independentes (dia da semana, semana do mês, mês e estação) que, apesar de não terem se mostrado significativas para o modelo de predição, tornaram-se úteis para realizar análises discriminando períodos de tempo específicos.

Ainda, viu-se que apenas com a temperatura, pôde-se prever o faturamento de vendas de sorvete, em virtude da alta correlação entre estas duas variáveis. Outrossim, além de selecionar as variáveis mais significativas para a predição, a regressão Lasso foi capaz de evitar o *overfitting* do modelo, o que resultou em uma boa generalização para novos conjuntos de dados.

Por fim, como a temperatura prevista para a data de inferência está abaixo da média, considerando a distribuição dos dados, não foi surpreendente que a predição de faturamento para esta mesma data resultasse, também, em um valor abaixo da média. Ainda, apoiando-se sobre as análises gráficas, viu-se que o valor predito encontra-se no limiar entre os 25% dos faturamentos mais baixos obtidos pela empresa.

7) Recomendação para o diretor financeiro

Tendo em vista que a predição do faturamento encontra-se no limiar entre os faturamentos rasoavelmente baixos da empresa, é aconselhável que se invista em campanhas de *marketing* e também promoções que busquem aumentar o número de pessoas consumindo sorvetes neste fim de mês de Outubro.

Por outro lado, uma vez que a relação entre faturamento e temperatura é alto, e tendo em vista que a temperatura é algo incontrolável pela empresa, seria interessante investir na criação de um produto que seja desejável nos períodos de frio. Assim, será mais garantido manter o faturamento, ao menos constante, independentemente da temperatura.