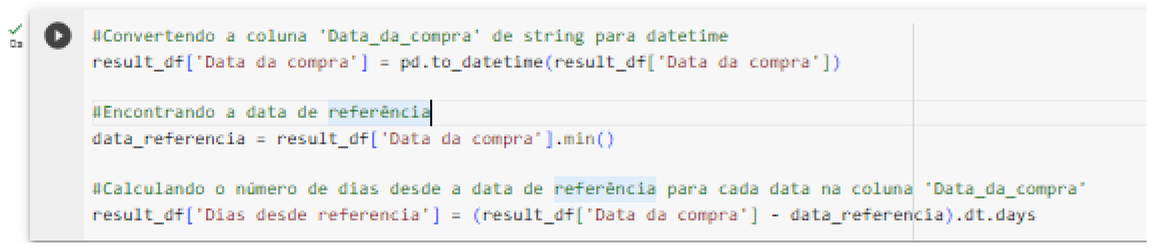
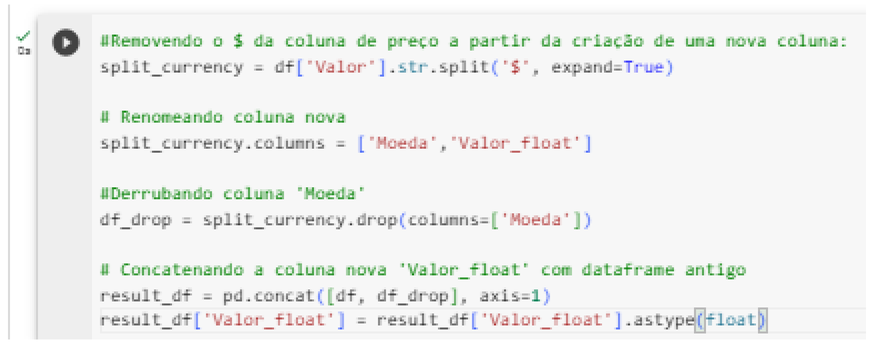
Creating a Multivariate Linear Regression

With the provided dataset, we processed some variables to make them usable in the model. The first adjustment was to the purchase date variable, which was transformed into the "Reference Days" variable, representing the difference in days between the earliest purchase date in the dataset and the specific date.



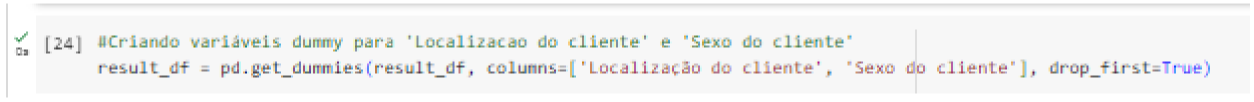
```python
#Convertendo a coluna 'Data_da_compra' de string para datetime
result_df['Data da compra'] = pd.to_datetime(result_df['Data da compra'])

#Encontrando a data de referência
data_referencia = result_df['Data da compra'].min()

#Calculando o número de dias desde a data de referência para cada data na coluna 'Data_da_compra'
result_df['Dias desde referencia'] = (result_df['Data da compra'] - data_referencia).dt.days
```

The second adjustment was for the "Value" variable, which was stored as a string. We converted it to a float.



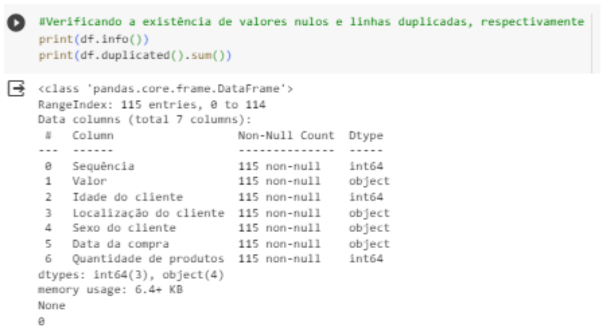
```python
#Removendo o $ da coluna de preço a partir da criação de uma nova coluna:
split_currency = df['Valor'].str.split('$', expand=True)

# Renomeando coluna nova
split_currency.columns = ['Moeda','Valor_float']

#Derrubando coluna 'Moeda'
df_drop = split_currency.drop(columns=['Moeda'])

# Concatenando a coluna nova 'Valor_float' com dataframe antigo
result_df = pd.concat([df, df_drop], axis=1)
result_df['Valor_float'] = result_df['Valor_float'].astype(float)
```

Another necessary change involved the location and customer gender variables. We transformed these into "dummy" variables, as string variables cannot be directly included in a mathematical model. This resulted in several boolean variables stored as integers.



```python
[24] #Criando variáveis dummy para 'Localizacao do cliente' e 'Sexo do cliente'
     result_df = pd.get_dummies(result_df, columns=['Localização do cliente', 'Sexo do cliente'], drop_first=True)
```

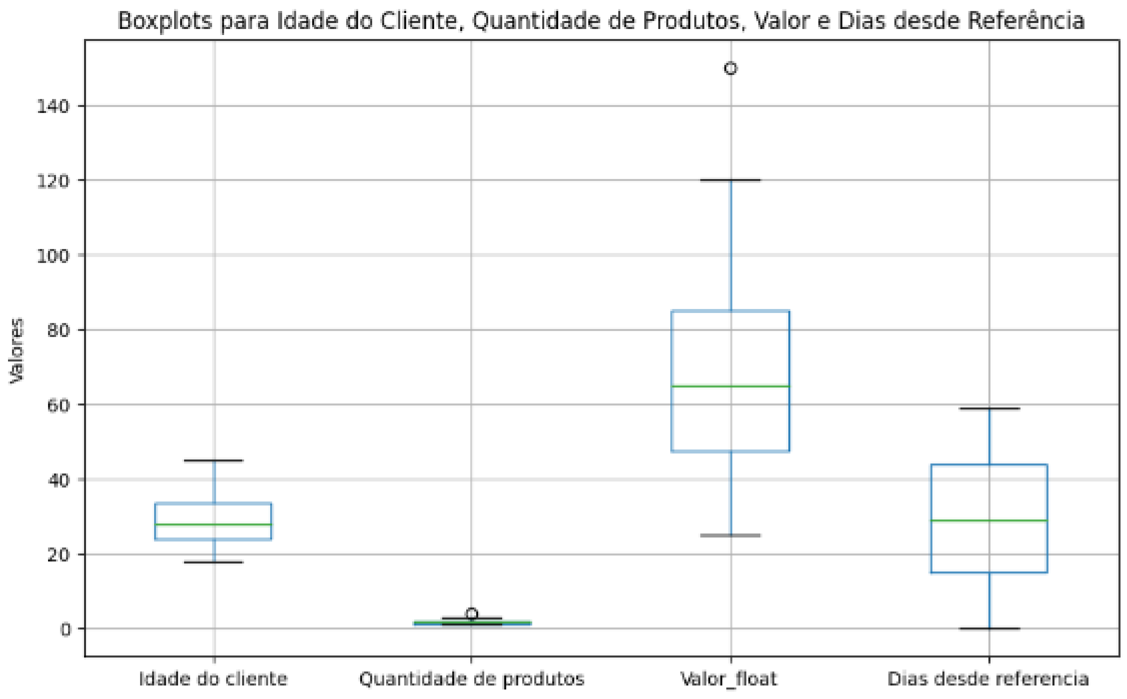
We also verified the dataset for null or duplicate values, finding none.



```
#Verificando a existência de valores nulos e linhas duplicadas, respectivamente
print(df.info())
print(df.duplicated().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115 entries, 0 to 114
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Sequência               115 non-null    int64 
 1   Valor                   115 non-null    object
 2   Idade do cliente        115 non-null    int64 
 3   Localização do cliente  115 non-null    object
 4   Sexo do cliente         115 non-null    object
 5   Data da compra          115 non-null    object
 6   Quantidade de produtos  115 non-null    int64 
dtypes: int64(3), object(4)
memory usage: 6.4+ KB
None
0
```

Next, we analyzed the need to address outliers.

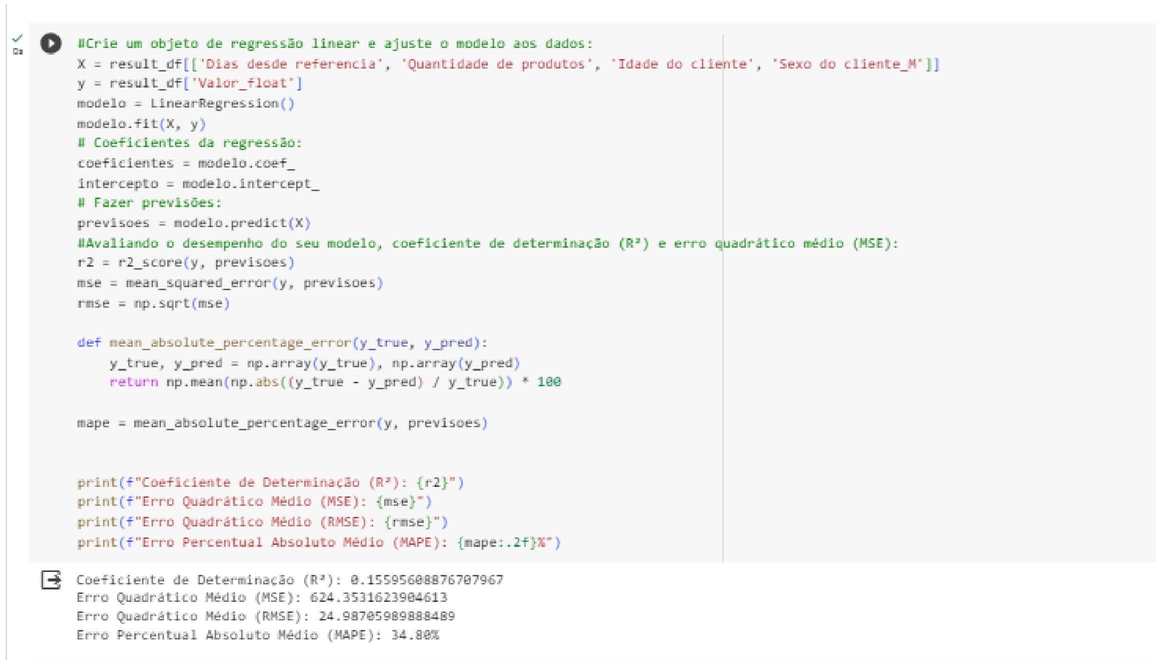


For numerical variables with meaningful variations, we found few outliers, making them negligible for the model.

Building the Models

Model 1 (without location variables):

Variables included: 'Reference Days', 'Product Quantity', 'Customer Age', 'Customer Gender_M'.



```python
#Crie um objeto de regressão linear e ajuste o modelo aos dados:
X = result_df[['Dias desde referencia', 'Quantidade de produtos', 'Idade do cliente', 'Sexo do cliente_M']]
y = result_df['Valor_float']
modelo = LinearRegression()
modelo.fit(X, y)
# Coeficientes da regressão:
coeficientes = modelo.coef_
intercepto = modelo.intercept_
# Fazer previsões:
previsoes = modelo.predict(X)
#Avaliando o desempenho do seu modelo, coeficiente de determinação (R²) e erro quadrático médio (MSE):
r2 = r2_score(y, previsoes)
mse = mean_squared_error(y, previsoes)
rmse = np.sqrt(mse)

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(y, previsoes)


print(f"Coeficiente de Determinação (R²): {r2}")
print(f"Erro Quadrático Médio (MSE): {mse}")
print(f"Erro Quadrático Médio (RMSE): {rmse}")
print(f"Erro Percentual Absoluto Médio (MAPE): {mape:.2f}%")
```

```
Coeficiente de Determinação (R²): 0.15595608876707967
Erro Quadrático Médio (MSE): 624.3531623904613
Erro Quadrático Médio (RMSE): 24.98705989888489
Erro Percentual Absoluto Médio (MAPE): 34.80%
```

Evaluation metrics for Model 1:

Coefficient of Determination (R^2): 0.15595608876707967

Mean Squared Error (MSE): 624.3531623904613

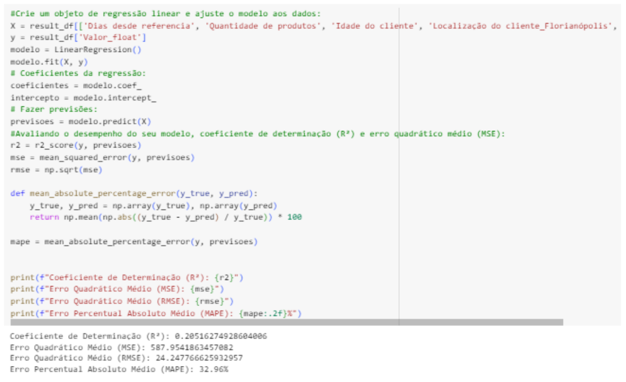
Root Mean Squared Error (RMSE): 24.98705989888489

Mean Absolute Percentage Error (MAPE): 34.80%

Model 2:

Variables included:

'Reference Days', 'Product Quantity', 'Customer Age', 'Customer Location_Florianópolis', 'Customer Location_Fortaleza', 'Customer Location_Brasília', 'Customer Location_Curitiba', 'Customer Location_Belém', 'Customer Location_Manaus', 'Customer Location_Goiânia', 'Customer Location_São Paulo', 'Customer Location_Salvador', 'Customer Location_Rio de Janeiro', 'Customer Location_Porto Alegre', 'Customer Location_Recife', 'Customer Gender_M'.



```
#Crie um objeto de regressão linear e ajuste o modelo aos dados:
X = result_df[['Dias desde referencia', 'Quantidade de produtos', 'Idade do cliente', 'Localização do cliente_Florianópolis',
y = result_df['Valor_float']
modelo = LinearRegression()
modelo.fit(X, y)
# Coeficientes da regressão:
coeficientes = modelo.coef_
intercepto = modelo.intercept_
# Fazer previsões:
previsoes = modelo.predict(X)
#Avaliando o desempenho do seu modelo, coeficiente de determinação (R²) e erro quadrático médio (MSE):
r2 = r2_score(y, previsoes)
mse = mean_squared_error(y, previsoes)
rmse = np.sqrt(mse)

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(y, previsoes)


print(f"Coeficiente de Determinação (R²): {r2}")
print(f"Erro Quadrático Médio (MSE): {mse}")
print(f"Erro Quadrático Médio (RMSE): {rmse}")
print(f"Erro Percentual Absoluto Médio (MAPE): {mape:.2f}%")
```

```
Coeficiente de Determinação (R²): 0.20516274928604006
Erro Quadrático Médio (MSE): 587.9541863457082
Erro Quadrático Médio (RMSE): 24.247766625932957
Erro Percentual Absoluto Médio (MAPE): 32.96%
```

Model 2 results:

Coefficient of Determination (R²): 0.20516274928604006

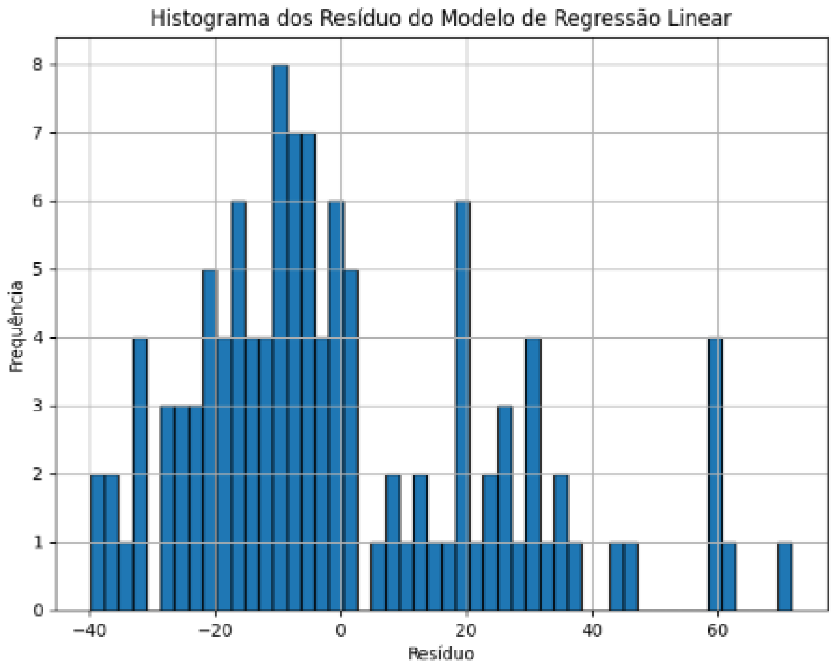
Mean Squared Error (MSE): 587.9541863457082

Root Mean Squared Error (RMSE): 24.247766625932957

Mean Absolute Percentage Error (MAPE): 32.96%

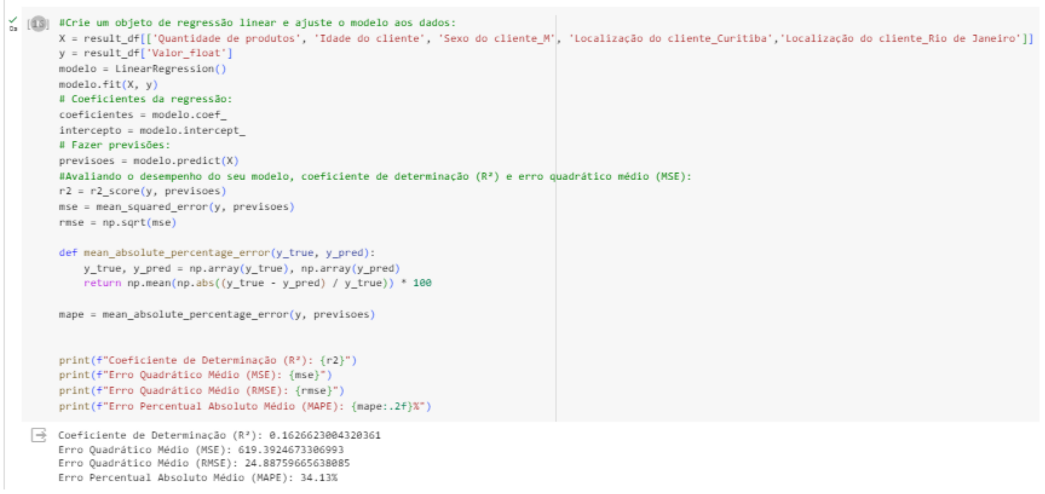
This model had the lowest error, though still inadequate for explaining the sales value variation.

A histogram of residuals was constructed.



Model 3:

Variables included: 'Product Quantity', 'Customer Age', 'Customer Gender_M', 'Customer Location_Curitiba', 'Customer Location_Rio de Janeiro'.



```
#Crie um objeto de regressão linear e ajuste o modelo aos dados:
X = result_df[['Quantidade de produtos', 'Idade do cliente', 'Sexo do cliente_M', 'Localização do cliente_Curitiba','Localização do cliente_Rio de Janeiro']]
y = result_df['Valor_float']
modelo = LinearRegression()
modelo.fit(X, y)
# Coeficientes da regressão:
coeficientes = modelo.coef_
intercepto = modelo.intercept_
# Fazer previsões:
previsoes = modelo.predict(X)
#Avaliando o desempenho do seu modelo, coeficiente de determinação (R²) e erro quadrático médio (MSE):
r2 = r2_score(y, previsoes)
mse = mean_squared_error(y, previsoes)
rmse = np.sqrt(mse)

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape = mean_absolute_percentage_error(y, previsoes)


print(f"Coeficiente de Determinação (R²): {r2}")
print(f"Erro Quadrático Médio (MSE): {mse}")
print(f"Erro Quadrático Médio (RMSE): {rmse}")
print(f"Erro Percentual Absoluto Médio (MAPE): {mape:.2f}%")
```

```
Coeficiente de Determinação (R²): 0.1626623004320361
Erro Quadrático Médio (MSE): 619.3924673306993
Erro Quadrático Médio (RMSE): 24.88759665638085
Erro Percentual Absoluto Médio (MAPE): 34.13%
```

Model 3 results:

Coefficient of Determination (R^2): 0.1626623004320361

Mean Squared Error (MSE): 619.3924673306993

Root Mean Squared Error (RMSE): 24.88759665638085

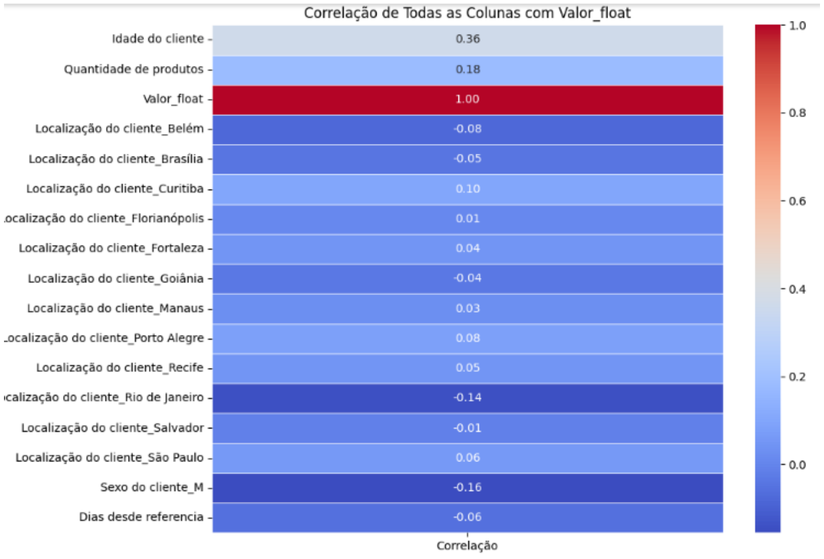
Mean Absolute Percentage Error (MAPE): 34.13%

Conclusion on the Models:

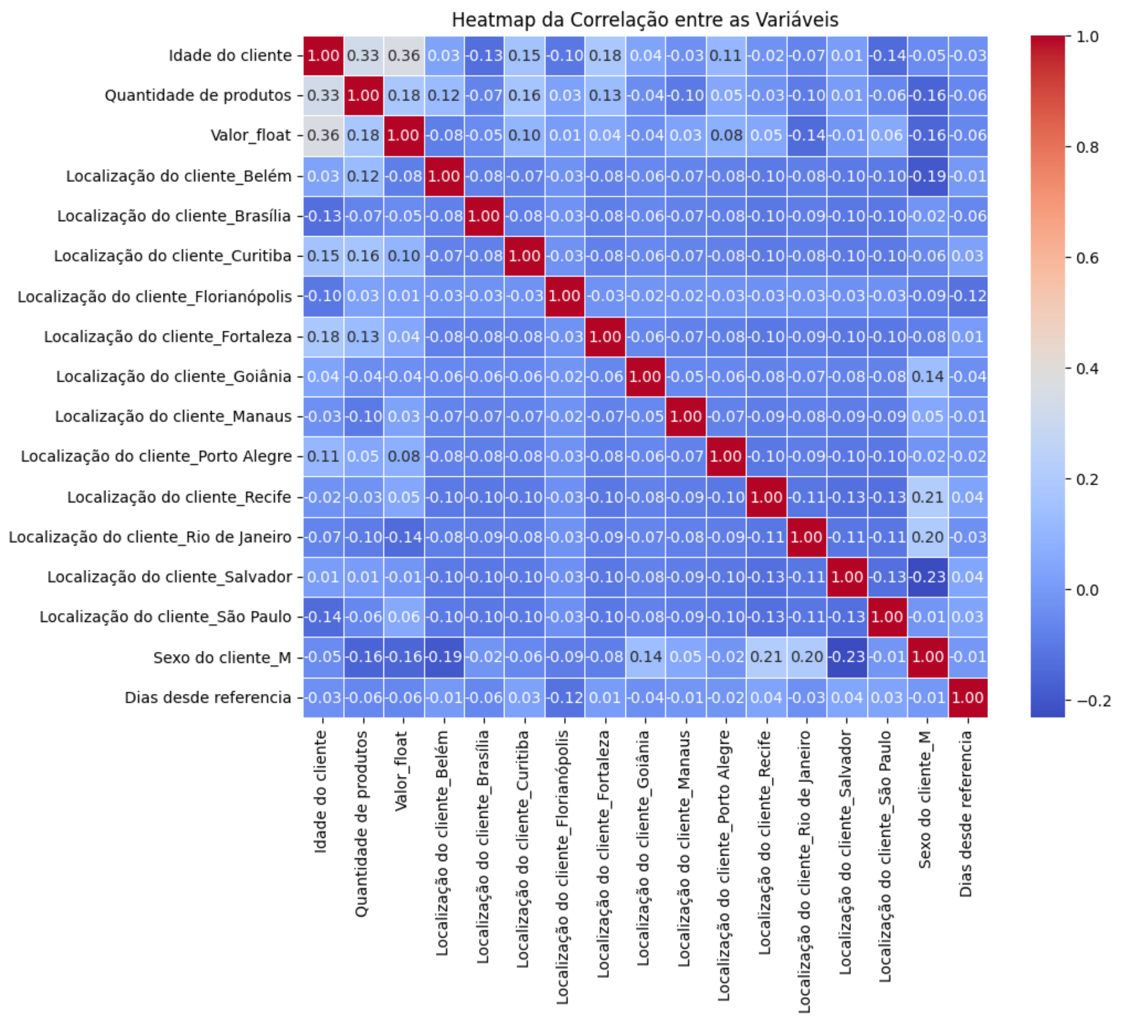
Testing various combinations of variables, we observed that, even in the best case, the provided data does not sufficiently estimate sales value. The coefficient of determination (R^2) in Model 2 explains only 20% of data variation, and the RMSE shows that predicted sales values deviate, on average, by approximately 24 units from actual values. The MAPE indicates an average deviation of 32.96%. Residual histogram analysis reveals random errors without systematic over- or under-prediction. Thus, the dataset is insufficient for an accurate sales value prediction model.

Correlation Matrix of Variables.

Pearson correlation between sales value and other variables:



Pearson correlation matrix of all variables:



Findings:

The variable most positively correlated with the sales value is "customer age," meaning older customers tend to have higher sales values. However, the correlation is 0.33, which is still relatively weak.

Additionally, the correlation matrix shows that the number of products also has a slightly notable correlation of 0.18. This indicates that a higher number of items in a sale might suggest a higher sales value. However, this variable requires careful analysis since the type and price of products influence this relationship. For instance, if the store offers a mix of low-cost products (e.g., items below R$10) and high-cost products (e.g., TVs priced above R$1,000), the correlation would likely be weaker. However, if the store only offered products with less price variation, a stronger positive correlation would be observed.

Another finding is the weak negative correlation between the sales value and male customers. This suggests a slight tendency for women to make higher-value purchases at Melhores Compras.

The correlation matrix also reveals that the "Rio de Janeiro location" variable has a negative correlation of -0.14, meaning purchases from Rio de Janeiro customers tend to have lower values. Conversely, the "Curitiba location" variable shows a positive correlation of +0.10, indicating slightly higher purchase values from Curitiba customers.

Suggestions for Next Steps:

To improve the modeling process at Melhores Compras, it would be valuable to study the behavior of new variables in relation to sales value. For example:

Does the time spent on the webpage before purchase correlate with higher sales values?

Do customers with longer registration periods tend to make higher-value purchases? This hypothesis could stem from the idea that successful transactions might increase customer trust in the platform.

There is also potential to explore other mathematical models, such as logistic regression, to analyze factors influencing purchase decisions. For instance, determining whether products with more photos or reviews are more likely to be purchased could help improve how products are displayed and marketed.