retail-pricing-optimization

March 2, 2024

Retail Pricing Optimization

Introduction:

Retail price optimization is crucial for maximizing profits while maintaining competitiveness in the market. It involves setting prices strategically based on various factors such as competitor analysis, customer segmentation, and price testing. By analyzing historical data and market trends, retailers can identify the most appropriate price points that attract customers and drive sales while maximizing profit margins.

Let's Get Started:

In this Analysis, we will cover:

- Exploratory Data Analysis (EDA) on Retail data with several products, including:
  - Competitor Analysis
  - Correlation Analysis
  - Month-wise Sales Analysis
- Feature Engineering
- Regression Model for predicting Optimal Prices

[153]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio

from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler

import eli5
from eli5.sklearn import PermutationImportance
import shap
pio.templates.default = "plotly_white"
```

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

Loading Data

[154]:
```
data = pd.read_csv('retail_price.csv')
data.head(5).T
```

[154]:
```
                                        0               1               2  \
product_id                           bed1            bed1            bed1
product_category_name      bed_bath_table  bed_bath_table  bed_bath_table
month_year                     01-05-2017      01-06-2017      01-07-2017
qty                                     1               3               6
total_price                         45.95          137.85           275.7
freight_price                        15.1       12.933333           14.84
unit_price                          45.95           45.95           45.95
product_name_lenght                    39              39              39
product_description_lenght            161             161             161
product_photos_qty                      2               2               2
product_weight_g                      350             350             350
product_score                         4.0             4.0             4.0
customers                              57              61             123
weekday                                23              22              21
weekend                                 8               8              10
holiday                                 1               1               1
month                                   5               6               7
year                                 2017            2017            2017
s                               10.267394        6.503115       12.071651
volume                               3800            3800            3800
comp_1                               89.9            89.9            89.9
ps1                                   3.9             3.9             3.9
fp1                             15.011897       14.769216       13.993833
comp_2                              215.0           209.0           205.0
ps2                                   4.4             4.4             4.4
fp2                                  8.76          21.322       22.195932
comp_3                              45.95           45.95           45.95
ps3                                   4.0             4.0             4.0
fp3                                  15.1       12.933333           14.84
lag_price                            45.9           45.95           45.95

                                        3               4
product_id                           bed1            bed1
product_category_name      bed_bath_table  bed_bath_table
month_year                     01-08-2017      01-09-2017
qty                                     4               2
total_price                         183.8            91.9
freight_price                     14.2875            15.1
unit_price                          45.95           45.95
```

```
product_name_lenght                   39            39
product_description_lenght           161           161
product_photos_qty                     2             2
product_weight_g                     350           350
product_score                        4.0           4.0
customers                             90            54
weekday                               23            21
weekend                                8             9
holiday                                1             1
month                                  8             9
year                                2017          2017
s                               9.293873      5.555556
volume                              3800          3800
comp_1                              89.9          89.9
ps1                                  3.9           3.9
fp1                            14.656757     18.776522
comp_2                        199.509804     163.39871
ps2                                  4.4           4.4
fp2                            19.412885     24.324687
comp_3                             45.95         45.95
ps3                                  4.0           4.0
fp3                              14.2875          15.1
lag_price                          45.95         45.95
```

[155]:
```python
# Getting information about the dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 676 entries, 0 to 675
Data columns (total 30 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   product_id                  676 non-null    object
 1   product_category_name       676 non-null    object
 2   month_year                  676 non-null    object
 3   qty                         676 non-null    int64
 4   total_price                 676 non-null    float64
 5   freight_price               676 non-null    float64
 6   unit_price                  676 non-null    float64
 7   product_name_lenght         676 non-null    int64
 8   product_description_lenght  676 non-null    int64
 9   product_photos_qty          676 non-null    int64
 10  product_weight_g            676 non-null    int64
 11  product_score               676 non-null    float64
 12  customers                   676 non-null    int64
 13  weekday                     676 non-null    int64
 14  weekend                     676 non-null    int64
```

```
 15   holiday                     676 non-null    int64
 16   month                       676 non-null    int64
 17   year                        676 non-null    int64
 18   s                           676 non-null    float64
 19   volume                      676 non-null    int64
 20   comp_1                      676 non-null    float64
 21   ps1                         676 non-null    float64
 22   fp1                         676 non-null    float64
 23   comp_2                      676 non-null    float64
 24   ps2                         676 non-null    float64
 25   fp2                         676 non-null    float64
 26   comp_3                      676 non-null    float64
 27   ps3                         676 non-null    float64
 28   fp3                         676 non-null    float64
 29   lag_price                   676 non-null    float64
dtypes: float64(15), int64(12), object(3)
memory usage: 158.6+ KB
```

[156]:
```
# Checking for missing values
any(data.isna().sum() > 0)
```

[156]:
```
False
```

[157]:
```
data.describe().T
```

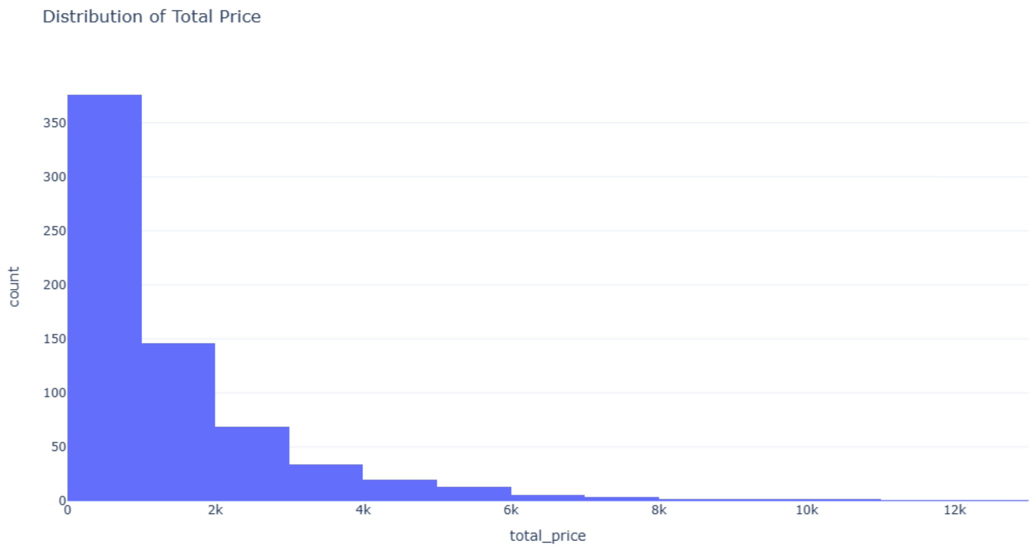
[157]:
```
                            count          mean          std          min  \
qty                         676.0     14.495562    15.443421     1.000000
total_price                 676.0   1422.708728  1700.123100    19.900000
freight_price               676.0     20.682270    10.081817     0.000000
unit_price                  676.0    106.496800    76.182972    19.900000
product_name_lenght         676.0     48.720414     9.420715    29.000000
product_description_lenght  676.0    767.399408   655.205015   100.000000
product_photos_qty          676.0      1.994083     1.420473     1.000000
product_weight_g            676.0   1847.498521  2274.808483   100.000000
product_score               676.0      4.085503     0.232021     3.300000
customers                   676.0     81.028107    62.055560     1.000000
weekday                     676.0     21.773669     0.986104    20.000000
weekend                     676.0      8.658284     0.705600     8.000000
holiday                     676.0      1.494083     0.940430     0.000000
month                       676.0      6.192308     3.243455     1.000000
year                        676.0   2017.525148     0.499737  2017.000000
s                           676.0     14.644970    11.930276     0.484262
volume                      676.0  10664.627219  9172.801850   640.000000
comp_1                      676.0     79.452054    47.933358    19.900000
ps1                         676.0      4.159467     0.121652     3.700000
fp1                         676.0     18.597610     9.406537     0.095439
comp_2                      676.0     92.930079    49.481269    19.900000
```

ps2	676.0	4.123521	0.207189	3.300000
fp2	676.0	18.620644	6.424174	4.410000
comp_3	676.0	84.182642	47.745789	19.900000
ps3	676.0	4.002071	0.233292	3.500000
fp3	676.0	17.965007	5.533256	7.670000
lag_price	676.0	107.399684	76.974657	19.850000

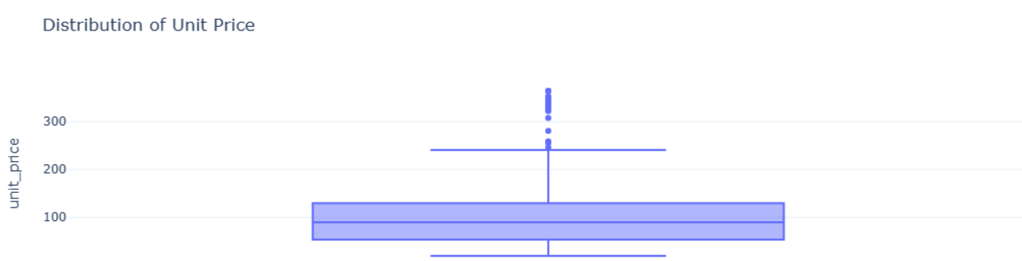
	25%	50%	75%	max
qty	4.000000	10.000000	18.000000	122.00
total_price	333.700000	807.890000	1887.322500	12095.00
freight_price	14.761912	17.518472	22.713558	79.76
unit_price	53.900000	89.900000	129.990000	364.00
product_name_lenght	40.000000	51.000000	57.000000	60.00
product_description_lenght	339.000000	501.000000	903.000000	3006.00
product_photos_qty	1.000000	1.500000	2.000000	8.00
product_weight_g	348.000000	950.000000	1850.000000	9750.00
product_score	3.900000	4.100000	4.200000	4.50
customers	34.000000	62.000000	116.000000	339.00
weekday	21.000000	22.000000	23.000000	23.00
weekend	8.000000	9.000000	9.000000	10.00
holiday	1.000000	1.000000	2.000000	4.00
month	3.000000	6.000000	8.000000	12.00
year	2017.000000	2018.000000	2018.000000	2018.00
s	7.510204	11.316760	17.745704	100.00
volume	3510.000000	8000.000000	15750.000000	32736.00
comp_1	49.910000	69.900000	104.256549	349.90
ps1	4.100000	4.200000	4.200000	4.50
fp1	13.826429	16.618984	19.732500	57.23
comp_2	53.900000	89.990000	117.888889	349.90
ps2	4.100000	4.200000	4.200000	4.40
fp2	14.485000	16.811765	21.665238	57.23
comp_3	53.785714	59.900000	99.990000	255.61
ps3	3.900000	4.000000	4.100000	4.40
fp3	15.042727	16.517110	19.447778	57.23
lag_price	55.668750	89.900000	129.990000	364.00

Exploratory Data Analysis (EDA)

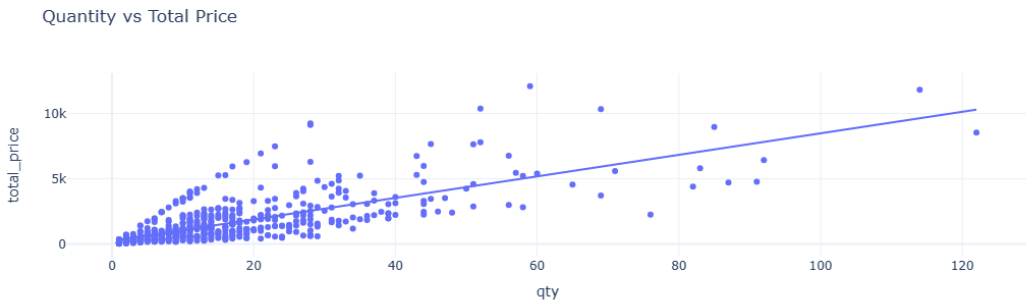
[158]:
```
# Distribution of Total Price
fig = px.histogram(data, x='total_price', nbins=20, title='Distribution of Total Price',width=1000,height=600)
fig.show()
```



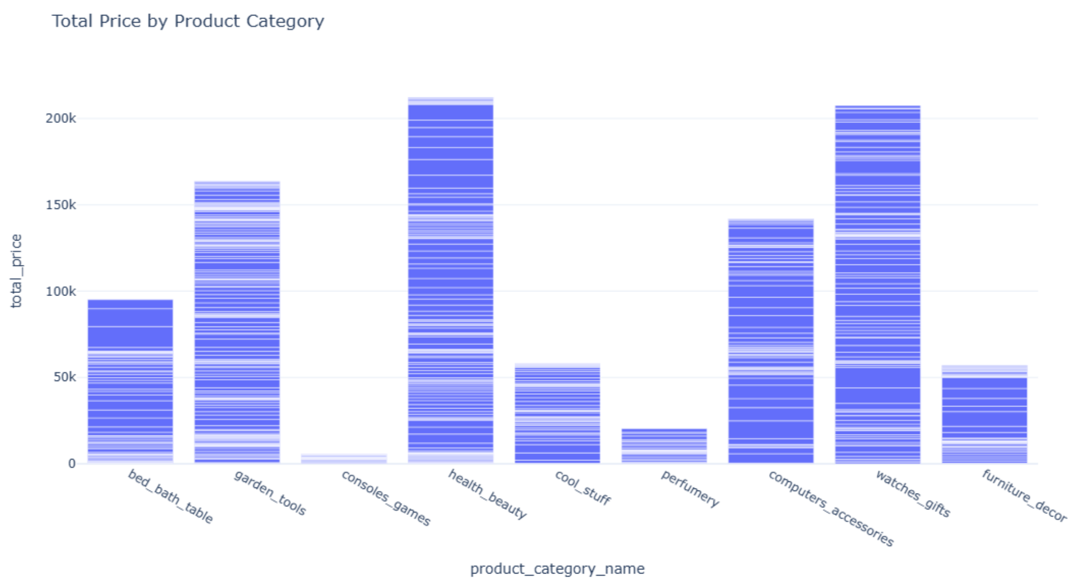
[159]:
```
# Distribution of Unit Price
fig = px.box(data, y='unit_price', title='Distribution of Unit Price')
fig.show()
```



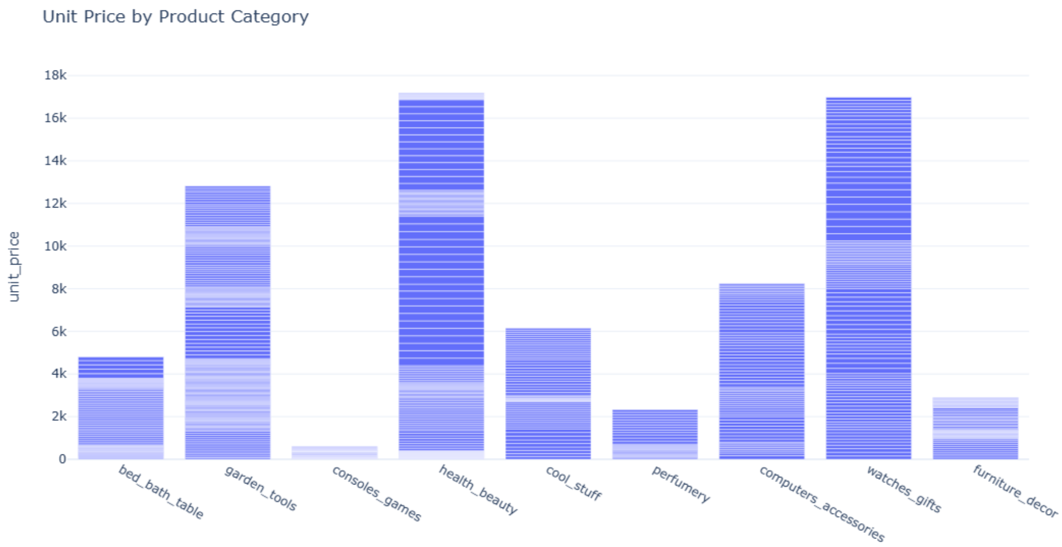
[160]:
```
# Scatter plot of Quantity vs Total Price
fig = px.scatter(data, x='qty', y='total_price', trendline='ols',␣
 ↪title='Quantity vs Total Price')
fig.show()
```



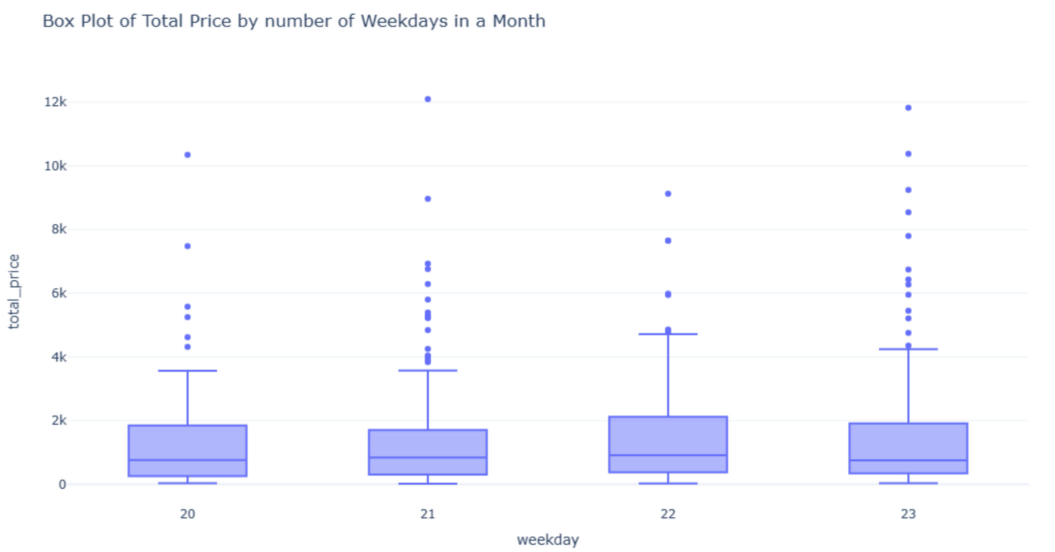
[161]:
```
# Bar plot of Total Price by Product Category
fig = px.bar(data, x='product_category_name', y='total_price', title='Total␣
↪Price by Product Category',width=1000,height=600)
fig.show()
```



[162]:
```
# Bar plot of Unit Price by Product Category
fig = px.bar(data, x='product_category_name', y='unit_price', title='Unit Price␣
↪by Product Category',width=1000,height=600)
fig.show()
```

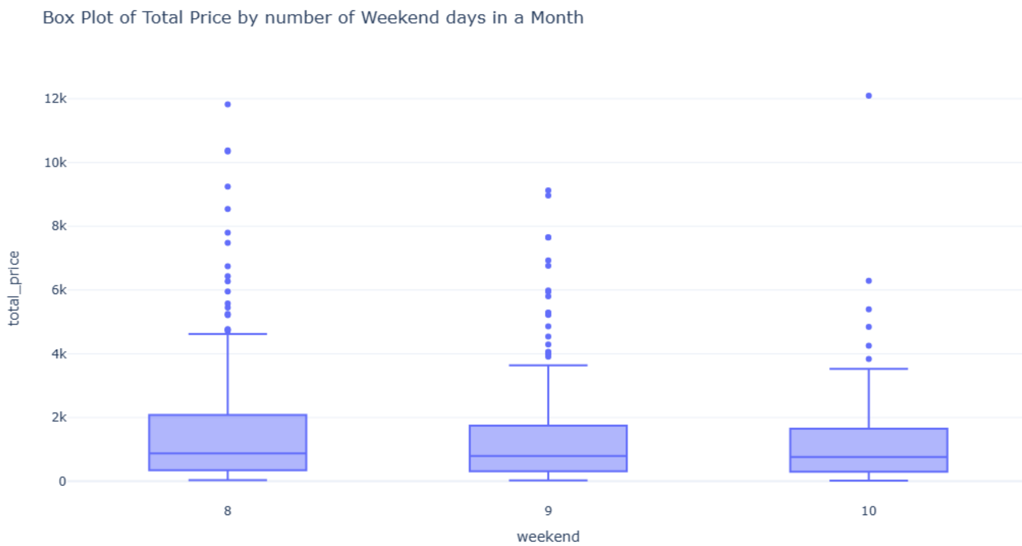


[163]:
```python
# Box plot of Total Price by number of Weekdays in a Month
fig = px.box(data, x='weekday', y='total_price', title='Box Plot of Total Price␣
 ↪by number of Weekdays in a Month',width=1000,height=600)
fig.show()
```



[164]:
```python
# Box plot of Total Price by number of Weekend days in a Month
```

```
fig = px.box(data, x='weekend', y='total_price', title='Box Plot of Total Price␣
↪by number of Weekend days in a Month',width=1000,height=600)
fig.show()
```



Competitor Analysis

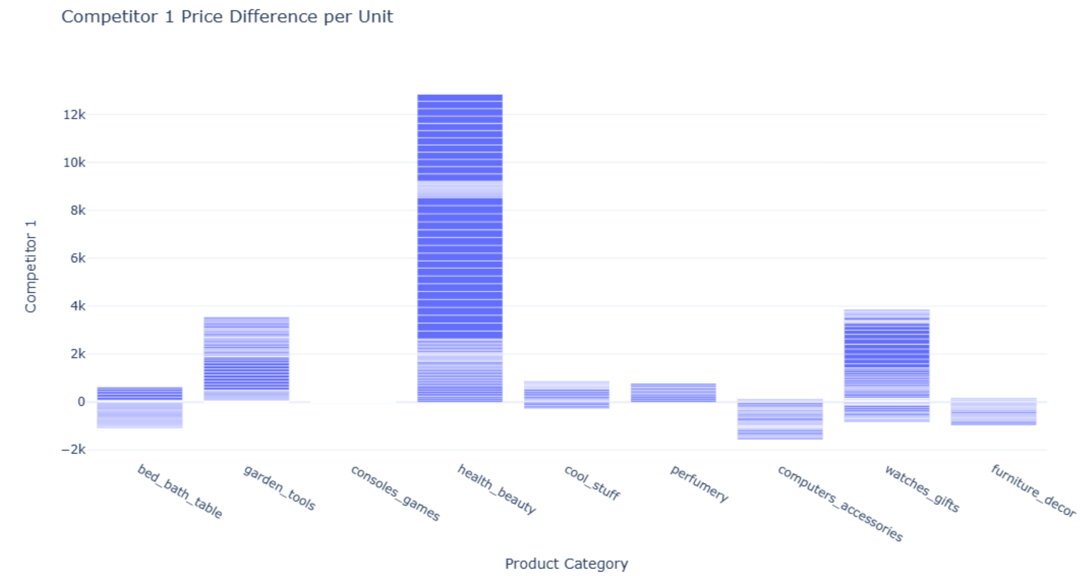
Competitor analysis is an essential part of retail pricing optimization. In this section, we will: - Compare our product prices with those of competitors to identify pricing gaps. - Visualize competitor price differences using various charts and plots.

[165]:

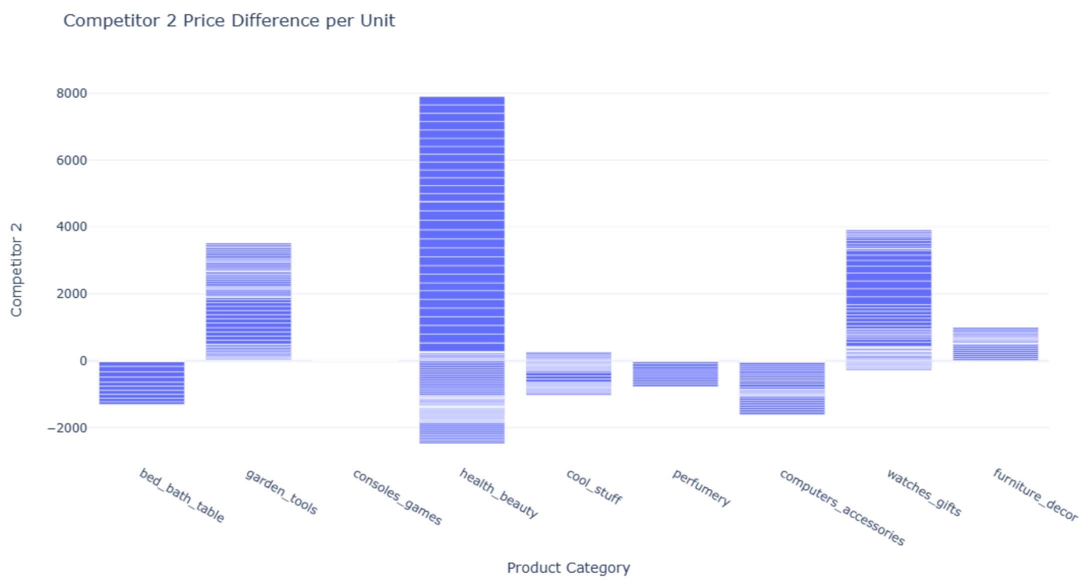
```
# Calculating price differences with competitors
data['comp1_diff'] = data['unit_price'] - data['comp_1']
data['comp2_diff'] = data['unit_price'] - data['comp_2']
data['comp3_diff'] = data['unit_price'] - data['comp_3']
```

[166]:

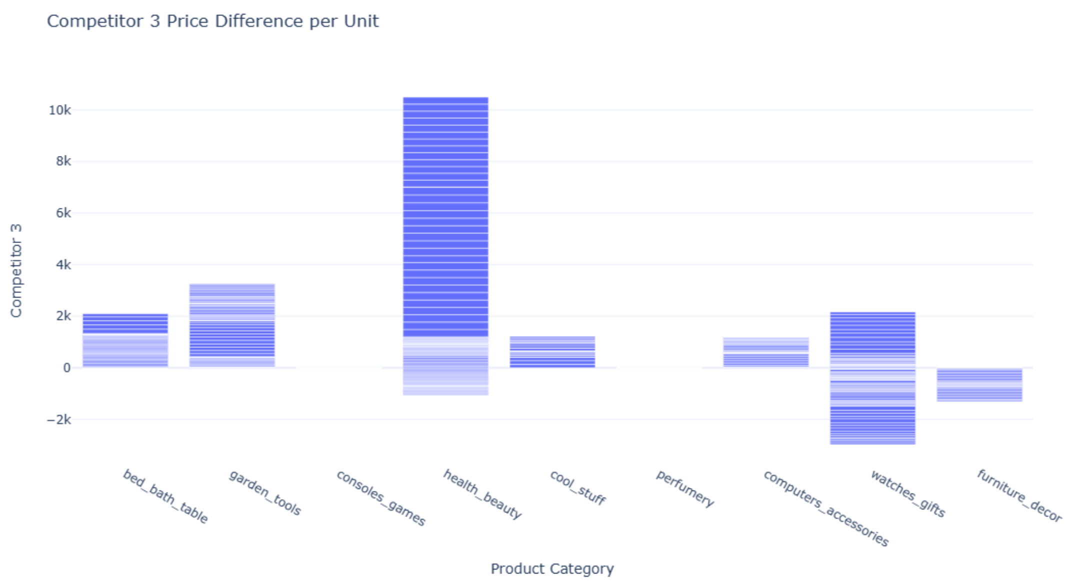
```
# Visualizing competitor price differences
for i in range(1, 4):
    comp = f"comp{i}_diff"
    fig = px.bar(x=data['product_category_name'], y=data[comp],␣
↪title=f"Competitor {i} Price Difference per Unit",
                 labels={'x': 'Product Category', 'y': f'Competitor {i}'},␣
↪width=800, height=600)
    fig.show()
```



Competitor 1 Price Difference per Unit
12k
10k
8k
6k
4k
2k
0
−2k
Competitor 1
bed_bath_table
garden_tools
consoles_games
health_beauty
cool_stuff
perfumery
computers_accessories
watches_gifts
furniture_decor
Product Category



Competitor 2 Price Difference per Unit
8000
6000
4000
2000
0
−2000
Competitor 2
bed_bath_table
garden_tools
consoles_games
health_beauty
cool_stuff
perfumery
computers_accessories
watches_gifts
furniture_decor
Product Category



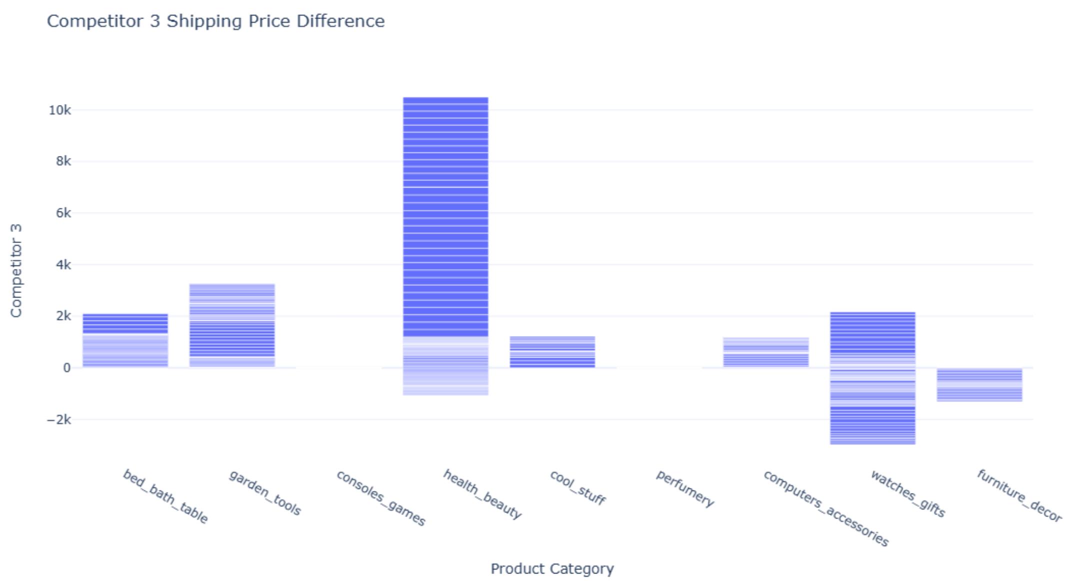
Freight Price

[167]:
```
data['fp1_diff'] = data['freight_price'] - data['fp1']
data['fp2_diff'] = data['freight_price'] - data['fp2']
data['fp3_diff'] = data['freight_price'] - data['fp3']

for i in range(1,4):
    comp = f"comp{i}_diff"
    fig = px.
 ↪bar(x=data['product_category_name'],y=data[comp],title=f"Competitor {i}␣
 ↪Shipping Price Difference",
                 labels={'x': 'Product Category','y': f'Competitor {i}'},␣
 ↪width=800,height=600)
    fig.show()
```

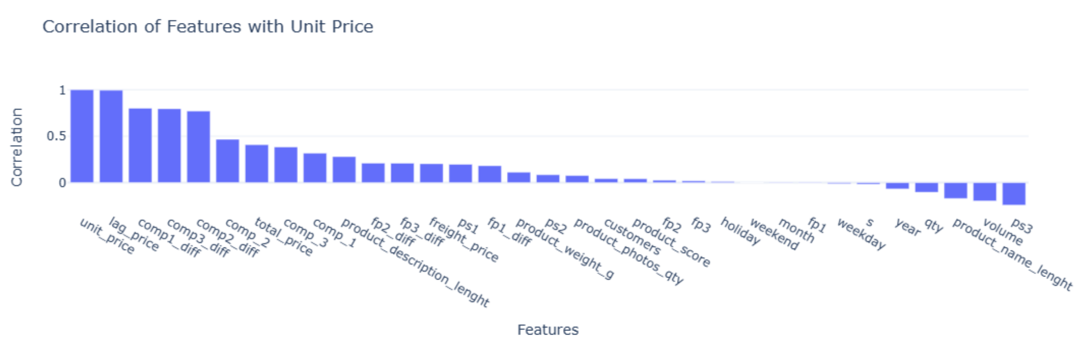

Competitor 1 Shipping Price Difference
12k
10k
8k
6k
4k
2k
0
−2k
Competitor 1
bed_bath_table
garden_tools
consoles_games
health_beauty
cool_stuff
perfumery
computers_accessories
watches_gifts
furniture_decor
Product Category


Competitor 2 Shipping Price Difference
8000
6000
4000
2000
0
−2000
Competitor 2
bed_bath_table
garden_tools
consoles_games
health_beauty
cool_stuff
perfumery
computers_accessories
watches_gifts
furniture_decor
Product Category



Correlation Analysis with Unit Price

[168]:
```python
# Correlation Analysis with Unit Price
numeric_data = data.select_dtypes(include='number')
corrs = numeric_data.corr()['unit_price'].sort_values(ascending=False)
fig = px.bar(x=corrs.keys(), y=corrs.values, title='Correlation of Features␣
↪with Unit Price',
             labels={'x': 'Features', 'y': 'Correlation'})
fig.show()
```



Month-wise Sales Analysis

[169]:
```python
# Aggregating data on a monthly basis
monthly_data = data.groupby(by='month_year').agg({
```

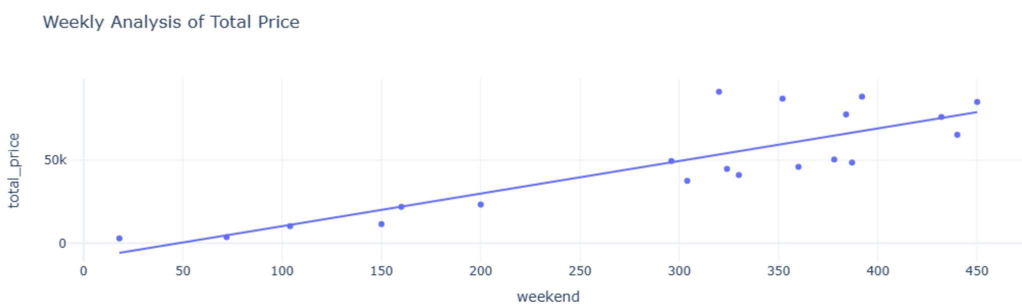
```
    'unit_price': 'mean', 'total_price': 'sum', 'freight_price': 'sum',
    'qty': 'sum', 'weekday': 'sum', 'weekend': 'sum', 'customers': 'sum'
}).reset_index()
monthly_data['month_year'] = pd.to_datetime(monthly_data['month_year'],␣
 ↪format='%d-%m-%Y')
monthly_data = monthly_data.sort_values(by='month_year')
```

[170]:
```
monthly_data.head()
```

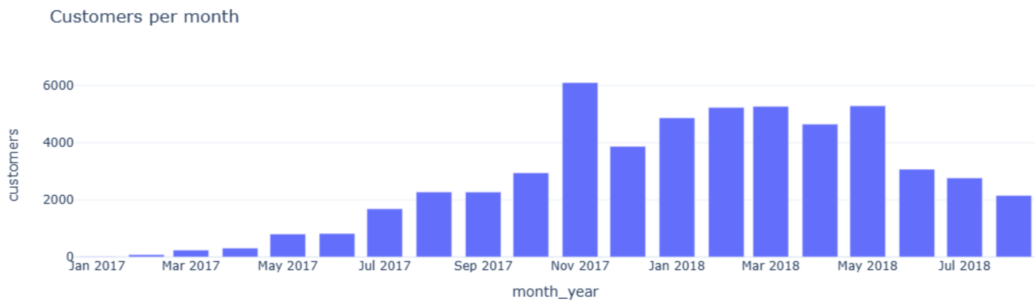
[170]:
```
   month_year  unit_price  total_price  freight_price  qty  weekday  weekend  \
0 2017-01-01  207.445000      2864.19      33.961250    9       44       18
2 2017-02-01  127.827143      3584.11     217.847838   35      180       72
4 2017-03-01  122.586615     10204.38     282.314965  101      299      104
6 2017-04-01  119.288667     11524.62     335.440132  121      300      150
8 2017-05-01  104.785769     21843.33     393.828633  222      460      160

   customers
0         18
2         78
4        242
6        309
8        803
```

[171]:
```
# Scatter plot of Weekly Analysis of Total Price
fig = px.scatter(monthly_data, x='weekend', y='total_price', trendline='ols',
                 title='Weekly Analysis of Total Price')
fig.show()
```



[172]:
```
# Bar plot of Customers per month
fig = px.bar(monthly_data, x='month_year', y='customers', title='Customers per␣
 ↪month')
fig.show()
```



Feature Engineering

[173]:
```
# Extracting date features
data['month'] = pd.to_datetime(data['month_year'], format='%d-%m-%Y').dt.month
data['year'] = pd.to_datetime(data['month_year'], format='%d-%m-%Y').dt.year
data['weekday'] = pd.to_datetime(data['month_year'], format='%d-%m-%Y').dt.
 ↪weekday
data['weekend'] = (pd.to_datetime(data['month_year'], format='%d-%m-%Y').dt.
 ↪weekday >= 5).astype(int)

# Calculate difference from competitors
data['comp_diff_mean'] = data[['comp_1', 'comp_2', 'comp_3']].mean(axis=1) -␣
 ↪data['unit_price']
# Total number of customers per product
total_customers_per_product = data.groupby('product_id')['customers'].sum()
data = data.merge(total_customers_per_product, on='product_id', suffixes=('',␣
 ↪'_total_customers'))
```

[174]:
```
# Select relevant features
features = ['qty', 'total_price', 'freight_price', 'product_score',
            'comp_diff_mean', 'month', 'year', 'weekday',␣
 ↪'weekend',"customers_total_customers" ]
```

[175]:
```
# Split data into features (X) and target variable (y)
X = data[features]
y = data['unit_price']
```

[176]:
```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[177]:
```
# Scaling features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

```
X_test_scaled = scaler.transform(X_test)
```

Model Creation

[178]:
```
# Initialize and fit the RandomForestRegressor model
model = RandomForestRegressor(n_estimators=50, random_state=42)
model.fit(X_train_scaled, y_train)
```

[178]:
```
RandomForestRegressor(n_estimators=50, random_state=42)
```

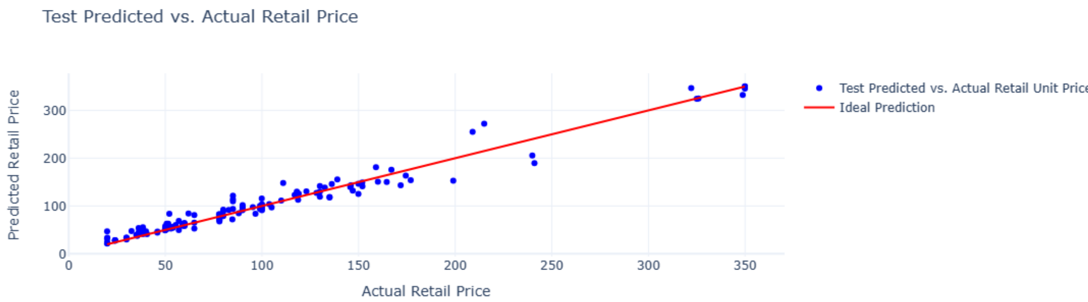
[179]:
```
# Making predictions
y_pred_train = model.predict(X_train_scaled)
y_pred_test = model.predict(X_test_scaled)
```

[180]:
```
# Evaluating the model
train_r2_score = r2_score(y_train, y_pred_train)
test_r2_score = r2_score(y_test, y_pred_test)
train_mae = mean_absolute_error(y_train, y_pred_train)
test_mae = mean_absolute_error(y_test, y_pred_test)

# Output model evaluation metrics
print(f"Train R2 score: {train_r2_score}")
print(f"Test R2 score: {test_r2_score}")
print(f"Train Mean Absolute Error: {train_mae}")
print(f"Test Mean Absolute Error: {test_mae}")
```

```
Train R2 score: 0.9920140370973357
Test R2 score: 0.9619318132952844
Train Mean Absolute Error: 4.058761611799262
Test Mean Absolute Error: 9.333377510832356
```

[181]:
```
# Scatter plot of Predicted vs Actual Retail Price for test set
fig = go.Figure()
fig.add_trace(go.Scatter(x=y_test, y=y_pred_test, mode='markers',␣
 ↪marker=dict(color='blue'),
                         name='Test Predicted vs. Actual Retail Unit Price'))
fig.add_trace(go.Scatter(x=[min(y_test), max(y_test)], y=[min(y_test),␣
 ↪max(y_test)], mode='lines',
                         marker=dict(color='red'), name='Ideal Prediction'))
fig.update_layout(
    title='Test Predicted vs. Actual Retail Price',
    xaxis_title='Actual Retail Price',
    yaxis_title='Predicted Retail Price'
)
fig.show()
```



Model Interpretation

After training the random forest regressor model with 50 estimators and using the scaled training data, we evaluated its performance using various metrics.

0.0.1 R2 Score:

The R2 score, also known as the coefficient of determination, measures the proportion of the variance in the dependent variable (unit price) that is predictable from the independent variables (features). A score of 0.992 on the training dataset and 0.962 on the testing dataset indicates that the model explains approximately 99.2% and 96.2% of the variance in unit price, respectively. These high R2 scores suggest that the model captures a significant amount of information about the target variable.

0.0.2 Mean Absolute Error (MAE):

The MAE measures the average absolute difference between the predicted unit prices and the actual unit prices. A lower MAE indicates better model performance. The model achieved a MAE of 4.059 on the training dataset and 9.333 on the testing dataset. These relatively low MAE values suggest that the model's predictions are close to the actual unit prices, on average.

0.0.3 Interpretation:

The high R2 scores and low MAE values demonstrate that the random forest regressor model performs exceptionally well in predicting optimal prices for retail products. The similar performance on both the training and testing datasets indicates that the model generalizes well to unseen data. Overall, the model's strong performance makes it a reliable tool for retail price optimization, providing valuable insights for pricing strategies and maximizing profitability.