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

```
df = pd.read_csv('/content/drive/MyDrive/pizza_sales.csv')
df.head()
```

	pizza_id	order_id	pizza_name_id	quantity	order_date	order_time	unit_price	t
0	1.0	1.0	hawaiian_m	1.0	1/1/2015	11:38:36	13.25	
1	2.0	2.0	classic_dlx_m	1.0	1/1/2015	11:57:40	16.00	
2	3.0	2.0	five_cheese_l	1.0	1/1/2015	11:57:40	18.50	

Next steps: View recommended plots

```
import pandas as pd

# Assuming 'df' is your DataFrame
rows, columns = df.shape
attributes = df.columns

print(f"Number of rows: {rows}")
print(f"Number of columns: {columns}")
print(f"Attributes: {attributes}")
```

```
Number of rows: 48620
Number of columns: 12
Attributes: Index(['pizza_id', 'order_id', 'pizza_name_id', 'quantity', 'order_date',
       'order_time', 'unit_price', 'total_price', 'pizza_size',
       'pizza_category', 'pizza_ingredients', 'pizza_name'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48620 entries, 0 to 48619
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   pizza_id           48620 non-null  float64
```

```
  1   order_id           48620 non-null  float64
  2   pizza_name_id      48620 non-null  object 
  3   quantity           48620 non-null  float64
  4   order_date         48620 non-null  object 
  5   order_time         48620 non-null  object 
  6   unit_price         48620 non-null  float64
  7   total_price        48620 non-null  float64
  8   pizza_size         48620 non-null  object 
  9   pizza_category     48620 non-null  object 
  10  pizza_ingredients  48620 non-null  object 
  11  pizza_name         48620 non-null  object 
dtypes: float64(5), object(7)
memory usage: 4.5+ MB
```

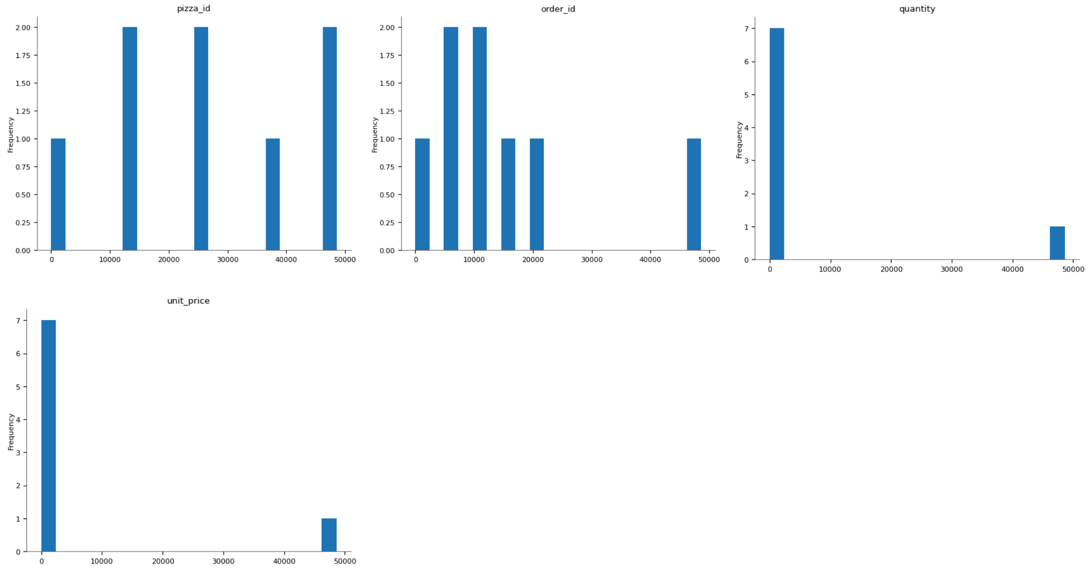
```
df.isna().sum()
```

```
pizza_id             0
order_id             0
pizza_name_id        0
quantity             0
order_date           0
order_time           0
unit_price           0
total_price          0
pizza_size           0
pizza_category       0
pizza_ingredients    0
pizza_name           0
dtype: int64
```

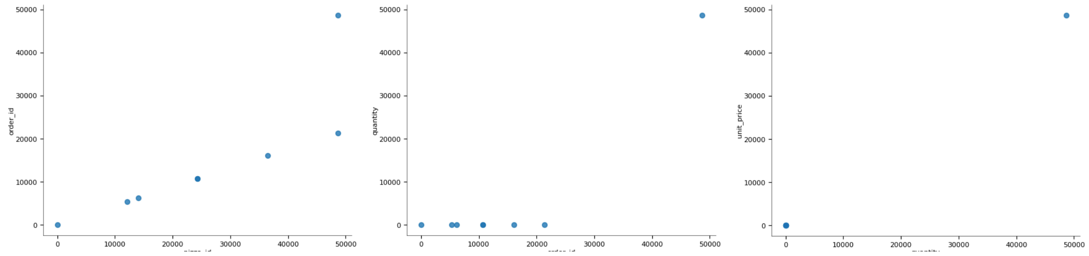
```
df.describe()
```

	pizza_id	order_id	quantity	unit_price	total_price
count	48620.000000	48620.000000	48620.000000	48620.000000	48620.000000
mean	24310.500000	10701.479761	1.019622	16.494132	16.821474
std	14035.529381	6180.119770	0.143077	3.621789	4.437398
min	1.000000	1.000000	1.000000	9.750000	9.750000
25%	12155.750000	5337.000000	1.000000	12.750000	12.750000
50%	24310.500000	10682.500000	1.000000	16.500000	16.500000
75%	36465.250000	16100.000000	1.000000	20.250000	20.500000
max	48620.000000	21350.000000	4.000000	35.950000	83.000000

Distributions



2-d distributions



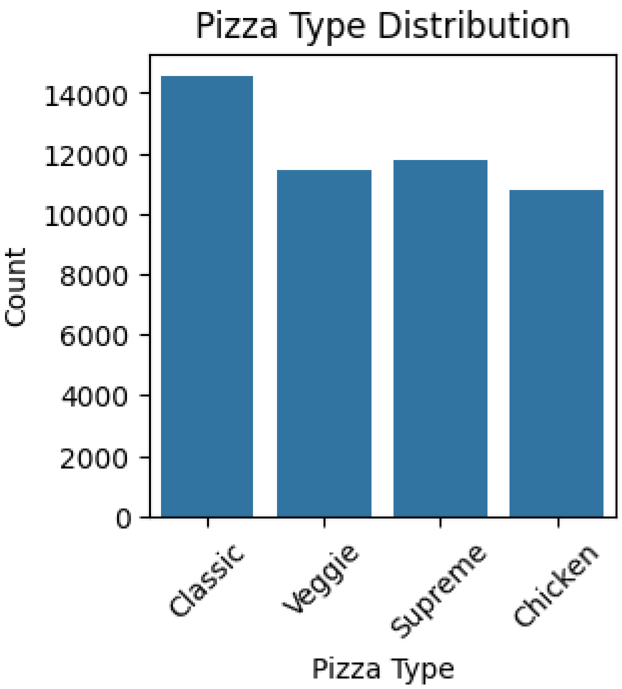
```
df['pizza_size'].value_counts()
```

```
pizza_size
L      18526
M      15385
S      14137
XL       544
XXL       28
Name: count, dtype: int64
```

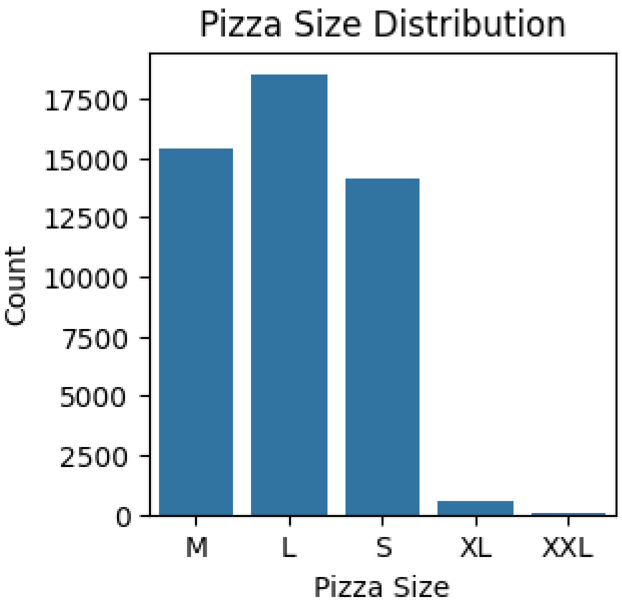
```
df['pizza_name'].value_counts()
```

```
pizza_name
The Classic Deluxe Pizza                      2416
The Barbecue Chicken Pizza                    2372
The Hawaiian Pizza                            2370
The Pepperoni Pizza                           2369
The Thai Chicken Pizza                        2315
The California Chicken Pizza                  2302
The Sicilian Pizza                            1887
The Spicy Italian Pizza                       1887
The Southwest Chicken Pizza                   1885
The Four Cheese Pizza                         1850
The Italian Supreme Pizza                     1849
The Big Meat Pizza                            1811
The Vegetables + Vegetables Pizza             1510
The Mexicana Pizza                            1456
The Napolitana Pizza                          1451
The Spinach and Feta Pizza                    1432
The Prosciutto and Arugula Pizza              1428
The Pepper Salami Pizza                       1422
The Italian Capocollo Pizza                   1414
The Greek Pizza                               1406
The Five Cheese Pizza                         1359
The Pepperoni, Mushroom, and Peppers Pizza    1342
The Green Garden Pizza                         987
The Chicken Alfredo Pizza                      980
The Italian Vegetables Pizza                   975
The Chicken Pesto Pizza                        961
The Spinach Pesto Pizza                        957
The Soppressata Pizza                          957
The Spinach Supreme Pizza                      940
The Calabrese Pizza                            927
The Mediterranean Pizza                        923
The Brie Carre Pizza                           480
Name: count, dtype: int64
```

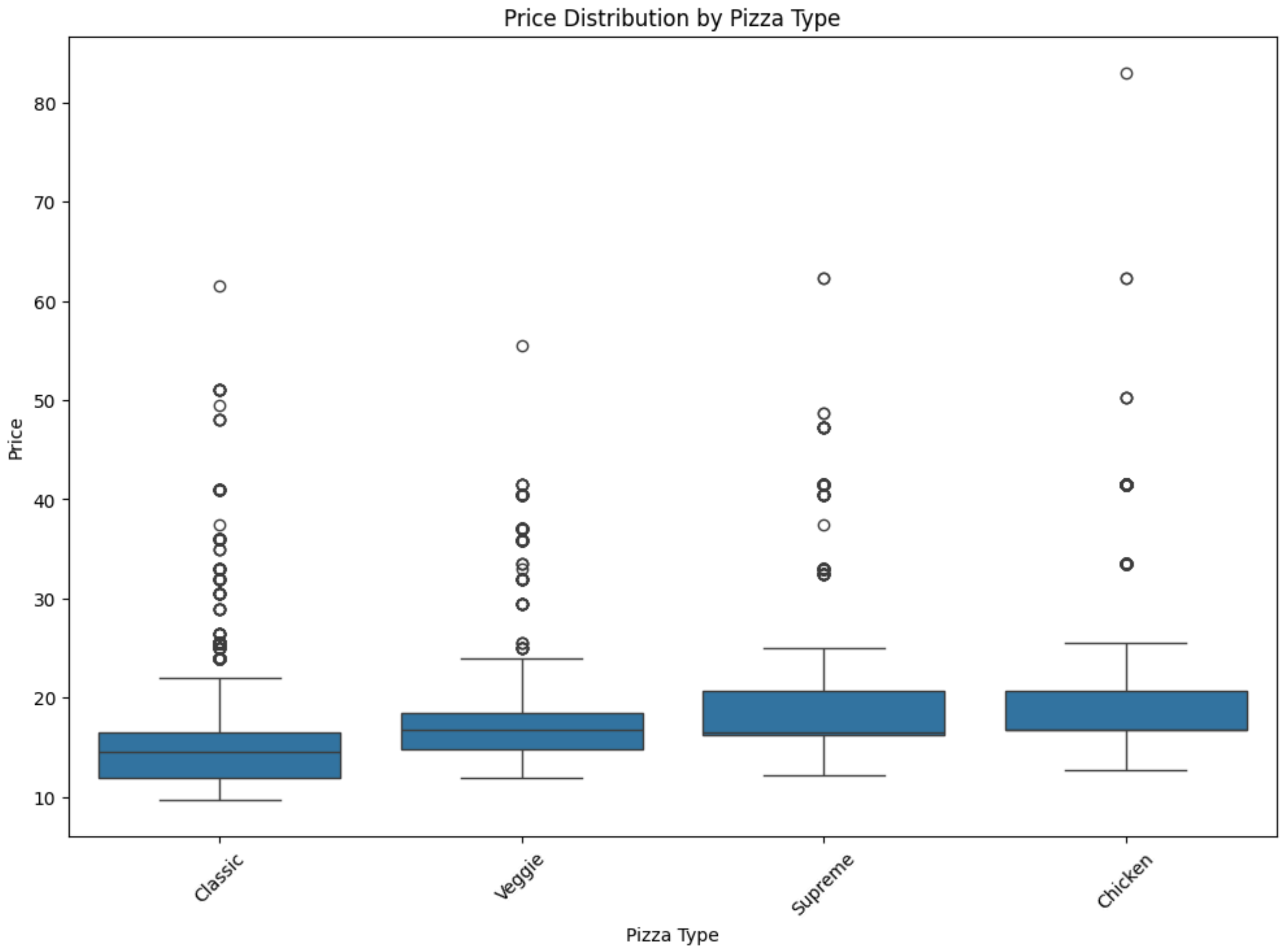
```
plt.figure(figsize=(3, 3))
sns.countplot(data=df, x='pizza_category')
plt.title('Pizza Type Distribution')
plt.xlabel('Pizza Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



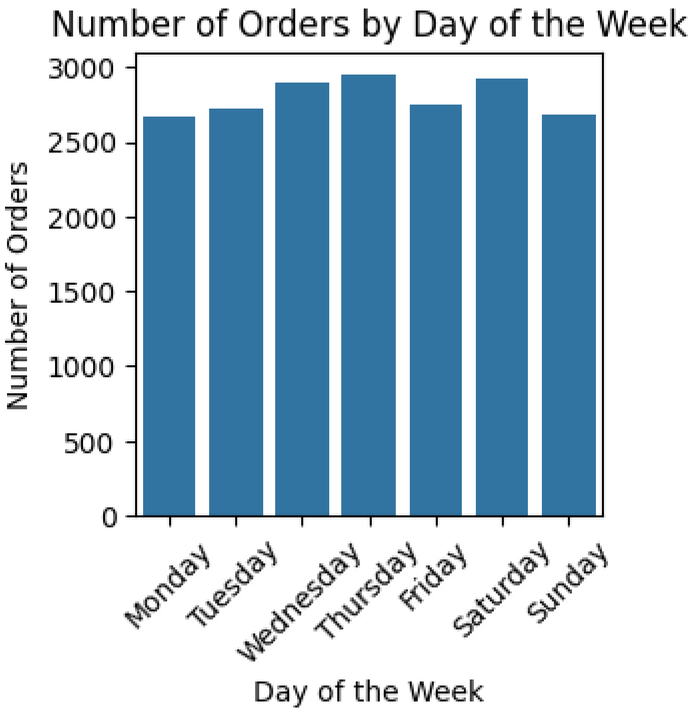
```
plt.figure(figsize=(3, 3))
sns.countplot(data=df, x='pizza_size')
plt.title('Pizza Size Distribution')
plt.xlabel('Pizza Size')
plt.ylabel('Count')
plt.show()
```



```
plt.figure(figsize=(12, 8))
sns.boxplot(data=df, x='pizza_category', y='total_price')
plt.title('Price Distribution by Pizza Type')
plt.xlabel('Pizza Type')
plt.ylabel('Price')
plt.xticks(rotation=45)
plt.show()
```

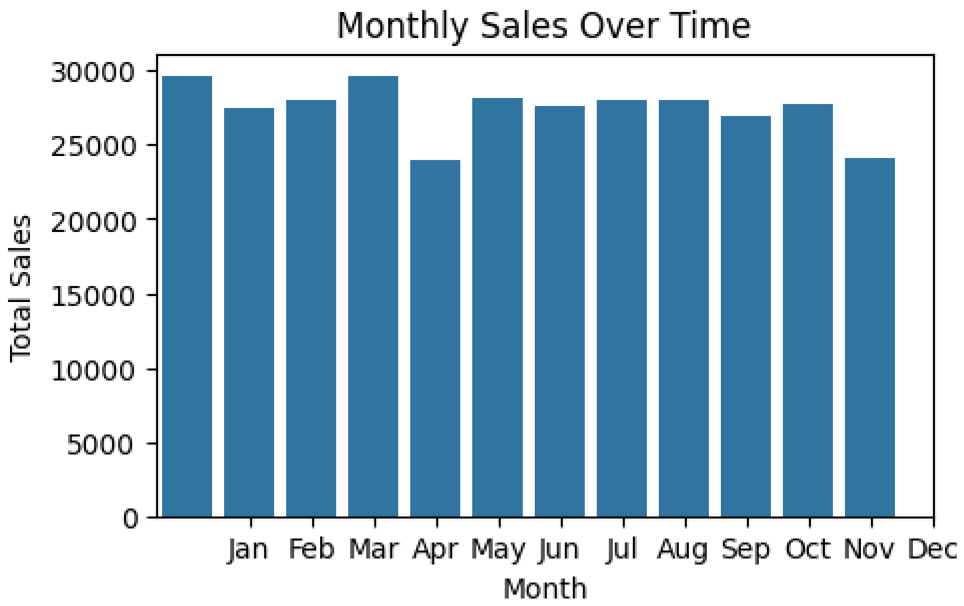

Price Distribution by Pizza Type
Price
80
70
60
50
40
30
20
10
Classic
Veggie
Supreme
Chicken
Pizza Type

```
df['order_date'] = pd.to_datetime(df['order_date'],errors='coerce')

# Extractday of the week (0 = Monday, 6 = Sunday)
df['day_of_week'] = df['order_date'].dt.dayofweek

days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sund

#  DataFrame to store daily order counts
daily_orders = df['day_of_week'].value_counts().sort_index().reindex(range(7), fill_value

plt.figure(figsize=(3, 3))
sns.barplot(x=days_of_week, y=daily_orders.values)
plt.title('Number of Orders by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45)
plt.show()
```



```
df['order_date'] = pd.to_datetime(df['order_date'],errors='coerce')
df['month'] = df['order_date'].dt.month
monthly_sales = df.groupby('month')['total_price'].sum()
```

```
plt.figure(figsize=(5, 3))
sns.barplot(x=monthly_sales.index, y=monthly_sales.values)
plt.title('Monthly Sales Over Time')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
plt.show()
```



```
favorite_pizza = df.groupby(['pizza_name', 'pizza_size'])['order_id'].count().idxmax()
print("Favorite Pizza:", favorite_pizza)
```

```
Favorite Pizza: ('The Big Meat Pizza', 'S')
```

Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_absolute_error
```

```
from sklearn.model_selection import train_test_split
```

```
le = LabelEncoder()
df['pizza_id_encoded'] = le.fit_transform(df['pizza_id'])
df['pizza_name_encoded'] = le.fit_transform(df['pizza_name'])
df['order_date_encoded'] = le.fit_transform(df['order_date'])
df['order_time_encoded'] = le.fit_transform(df['order_time'])
df['pizza_size_encoded'] = le.fit_transform(df['pizza_size'])
df['pizza_category_encoded'] = le.fit_transform(df['pizza_category'])
df['pizza_ingredients_encoded'] = le.fit_transform(df['pizza_ingredients'])
```

```
X = df[['pizza_size_encoded','pizza_category_encoded', 'pizza_ingredients_encoded']]
y = df['total_price']
```

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

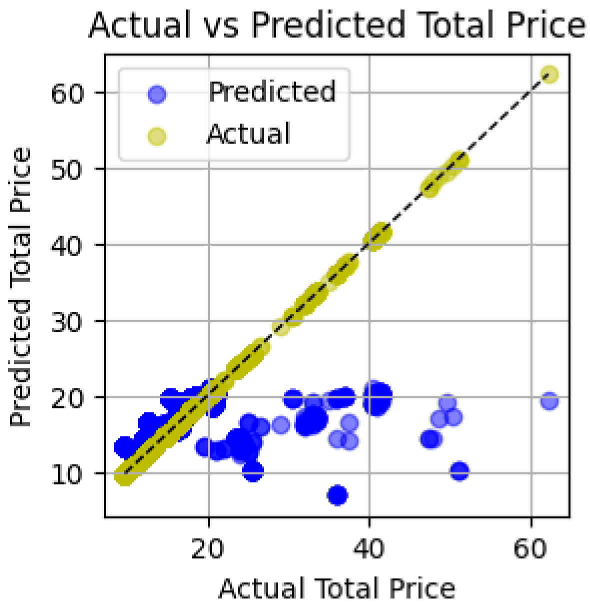
```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred = model.predict(X_test)
predicted_prices = model.predict(X)
```

```
most_sold_pizza = df['pizza_name'].value_counts().idxmax()
print(f"Predicted Price: {predicted_prices.mean()}")
print(f"Most Sold Pizza: {most_sold_pizza}")
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mean_actual = y_test.mean()
mse_percentage = (mse / mean_actual) * 100
mape = mean_absolute_error(y_test, y_pred) / mean_actual * 100
rmse=np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")
print(f"Mean Squared Error Percentage: {mse_percentage:.2f}%")
print(f"R-squared: {r2}")
import matplotlib.pyplot as plt
plt.figure(figsize=(3, 3))
plt.scatter(y_test, y_pred, color='b', label='Predicted', alpha=0.5)
plt.scatter(y_test, y_test, color='y', label='Actual', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=1)
plt.xlabel('Actual Total Price')
plt.ylabel('Predicted Total Price')
plt.title('Actual vs Predicted Total Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
Predicted Price: 16.831657327045967
Most Sold Pizza: The Classic Deluxe Pizza
Root Mean Squared Error (RMSE): 3.4847568887628606
Mean Squared Error: 12.143530573780213
Mean Absolute Error (MAE): 1.6426313470600975
Mean Absolute Error Percentage (MAPE): 9.79%
Mean Squared Error Percentage: 72.37%
R-squared: 0.393204246713203
```



Decision Trees

```
from sklearn.tree import DecisionTreeRegressor
```

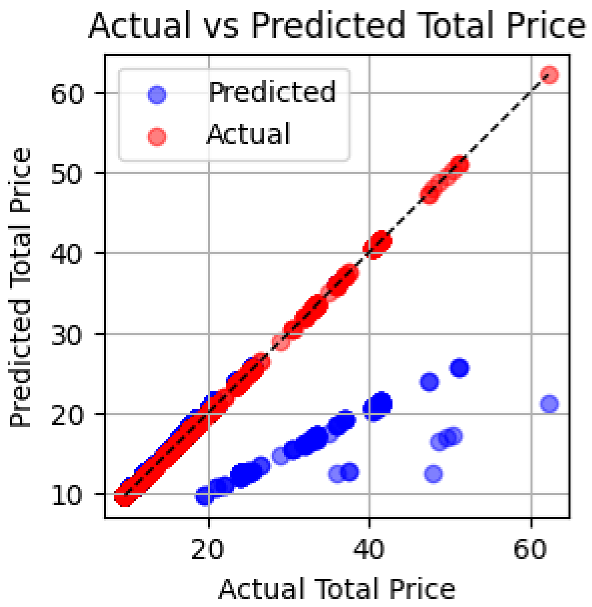
```
model = DecisionTreeRegressor(random_state=42)
model.fit(X_train, y_train)
```

```
▾         DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

```
y_pred = model.predict(X_test)
predicted_prices = model.predict(X)
```

```
most_sold_pizza = df['pizza_name'].value_counts().idxmax()
print(f"Predicted Price: {predicted_prices.mean()}")
print(f"Most Sold Pizza: {most_sold_pizza}")
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mean_actual = y_test.mean()
mse_percentage = (mse / mean_actual) * 100
mape = mean_absolute_error(y_test, y_pred) / mean_actual * 100
rmse=np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")
print(f"Mean Squared Error Percentage: {mse_percentage:.2f}%")
print(f"R-squared: {r2}")
import matplotlib.pyplot as plt
plt.figure(figsize=(3, 3))
plt.scatter(y_test, y_pred, color='b', label='Predicted', alpha=0.5)
plt.scatter(y_test, y_test, color='r', label='Actual', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=1)
plt.xlabel('Actual Total Price')
plt.ylabel('Predicted Total Price')
plt.title('Actual vs Predicted Total Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
Predicted Price: 16.820613402256335
Most Sold Pizza: The Classic Deluxe Pizza
Root Mean Squared Error (RMSE): 2.470449559058465
Mean Squared Error: 6.103121023852l645
Mean Absolute Error (MAE): 0.6428497160727743
Mean Absolute Error Percentage (MAPE): 3.83%
Mean Squared Error Percentage: 36.37%
R-squared: 0.6950353197063652
```



Random Forests

```
from sklearn.ensemble import RandomForestRegressor
```

```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

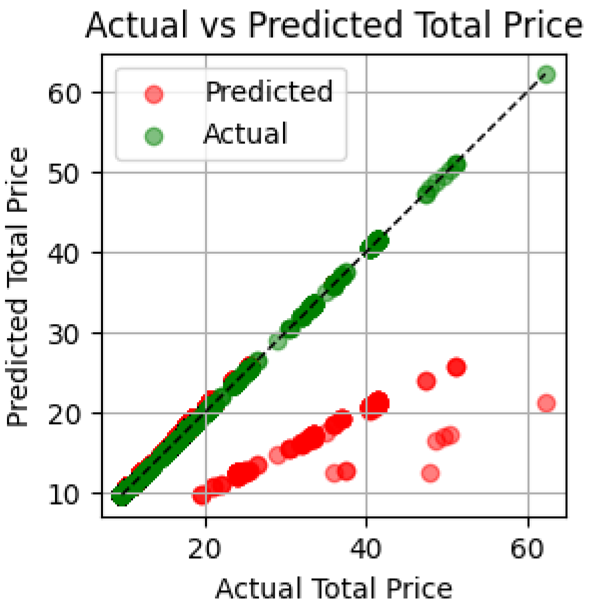
```
▾        RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
y_pred = model.predict(X_test)
predicted_prices = model.predict(X)
```

```
from sklearn.preprocessing import LabelEncoder

# Assuming 'data' is your DataFrame containing the pizza sales data
le = LabelEncoder()
df['pizza_id_encoded'] = le.fit_transform(df['pizza_id'])
```

```
most_sold_pizza = df['pizza_name'].value_counts().idxmax()
print(f"Predicted Price: {predicted_prices.mean()}")
print(f"Most Sold Pizza: {most_sold_pizza}")
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mean_actual = y_test.mean()
mse_percentage = (mse / mean_actual) * 100
mape = mean_absolute_error(y_test, y_pred) / mean_actual * 100
rmse=np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")
print(f"Mean Squared Error Percentage: {mse_percentage:.2f}%")
print(f"R-squared: {r2}")
import matplotlib.pyplot as plt
plt.figure(figsize=(3, 3))
plt.scatter(y_test, y_pred, color='r', label='Predicted', alpha=0.5)
plt.scatter(y_test, y_test, color='g', label='Actual', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=1)
plt.xlabel('Actual Total Price')
plt.ylabel('Predicted Total Price')
plt.title('Actual vs Predicted Total Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
Predicted Price: 16.821224685681262
Most Sold Pizza: The Classic Deluxe Pizza
Root Mean Squared Error (RMSE): 2.470468603990518
Mean Squared Error: 6.103215123302858
Mean Absolute Error (MAE): 0.6436547903754081
Mean Absolute Error Percentage (MAPE): 3.84%
Mean Squared Error Percentage: 36.37%
R-squared: 0.6950306176844351
```



KNN Regression

```
from sklearn.neighbors import KNeighborsRegressor
```

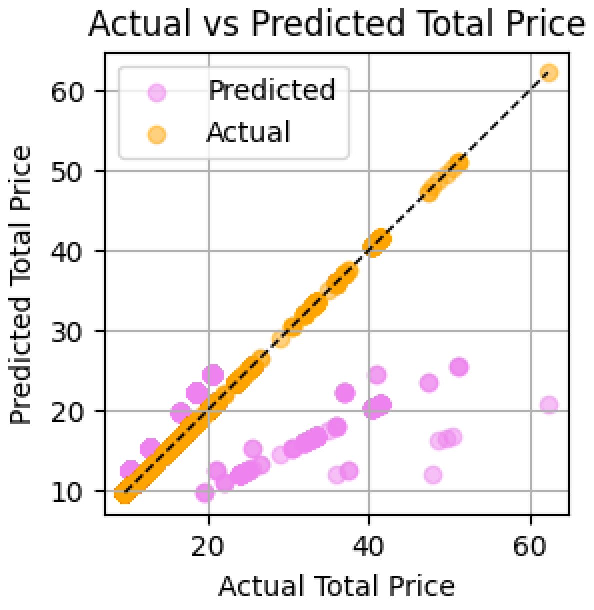
```
model = KNeighborsRegressor(n_neighbors=5)
model.fit(X_train, y_train)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

```
y_pred = model.predict(X_test)
```

```
most_sold_pizza = df['pizza_name'].value_counts().idxmax()
print(f"Predicted Price: {predicted_prices.mean()}")
print(f"Most Sold Pizza: {most_sold_pizza}")
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mean_actual = y_test.mean()
mse_percentage = (mse / mean_actual) * 100
mape = mean_absolute_error(y_test, y_pred) / mean_actual * 100
rmse=np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")
print(f"Mean Squared Error Percentage: {mse_percentage:.2f}%")
print(f"R-squared: {r2}")
import matplotlib.pyplot as plt
plt.figure(figsize=(3, 3))
plt.scatter(y_test, y_pred, color='Violet', label='Predicted', alpha=0.5)
plt.scatter(y_test, y_test, color='Orange', label='Actual', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=1)
plt.xlabel('Actual Total Price')
plt.ylabel('Predicted Total Price')
plt.title('Actual vs Predicted Total Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
Predicted Price: 16.821224685681262
Most Sold Pizza: The Classic Deluxe Pizza
Root Mean Squared Error (RMSE): 2.6213152626675233
Mean Squared Error: 6.871293706293707
Mean Absolute Error (MAE): 0.56444878650761
Mean Absolute Error Percentage (MAPE): 3.36%
Mean Squared Error Percentage: 40.95%
R-squared: 0.6566507725876805
```



Neural Networks (Multi-layer Perceptron)

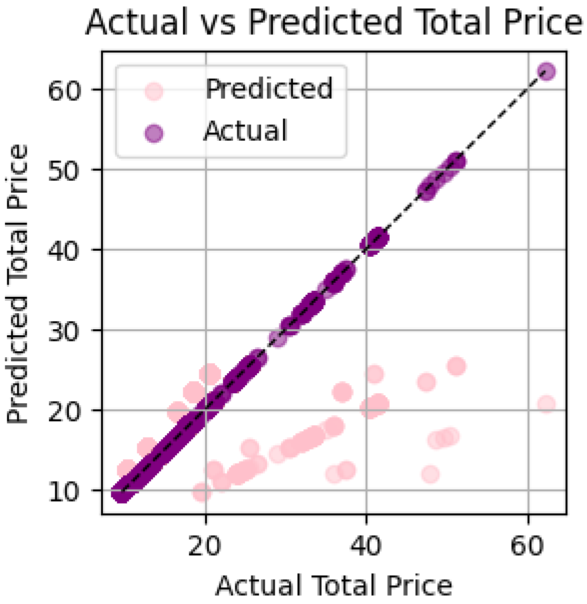
```
from sklearn.neural_network import MLPRegressor
```

```
model = MLPRegressor(hidden_layer_sizes=(100, 50), activation='relu', solver='adam', max_
model.fit(X_train, y_train)
```

```
▾                        MLPRegressor
MLPRegressor(hidden_layer_sizes=(100, 50), max_iter=500, random_state=42)
```

```
most_sold_pizza = df['pizza_name'].value_counts().idxmax()
print(f"Predicted Price: {predicted_prices.mean()}")
print(f"Most Sold Pizza: {most_sold_pizza}")
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mean_actual = y_test.mean()
mse_percentage = (mse / mean_actual) * 100
mape = mean_absolute_error(y_test, y_pred) / mean_actual * 100
rmse=np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")
print(f"Mean Squared Error Percentage: {mse_percentage:.2f}%")
print(f"R-squared: {r2}")
import matplotlib.pyplot as plt
plt.figure(figsize=(3, 3))
plt.scatter(y_test, y_pred, color='Pink', label='Predicted', alpha=0.5)
plt.scatter(y_test, y_test, color='purple', label='Actual', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=1)
plt.xlabel('Actual Total Price')
plt.ylabel('Predicted Total Price')
plt.title('Actual vs Predicted Total Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
Predicted Price: 16.821224685681262
Most Sold Pizza: The Classic Deluxe Pizza
Root Mean Squared Error (RMSE): 2.6213152626675233
Mean Squared Error: 6.871293706293707
Mean Absolute Error (MAE): 0.56444878650761
Mean Absolute Error Percentage (MAPE): 3.36%
Mean Squared Error Percentage: 40.95%
R-squared: 0.6566507725876805
```



SVR

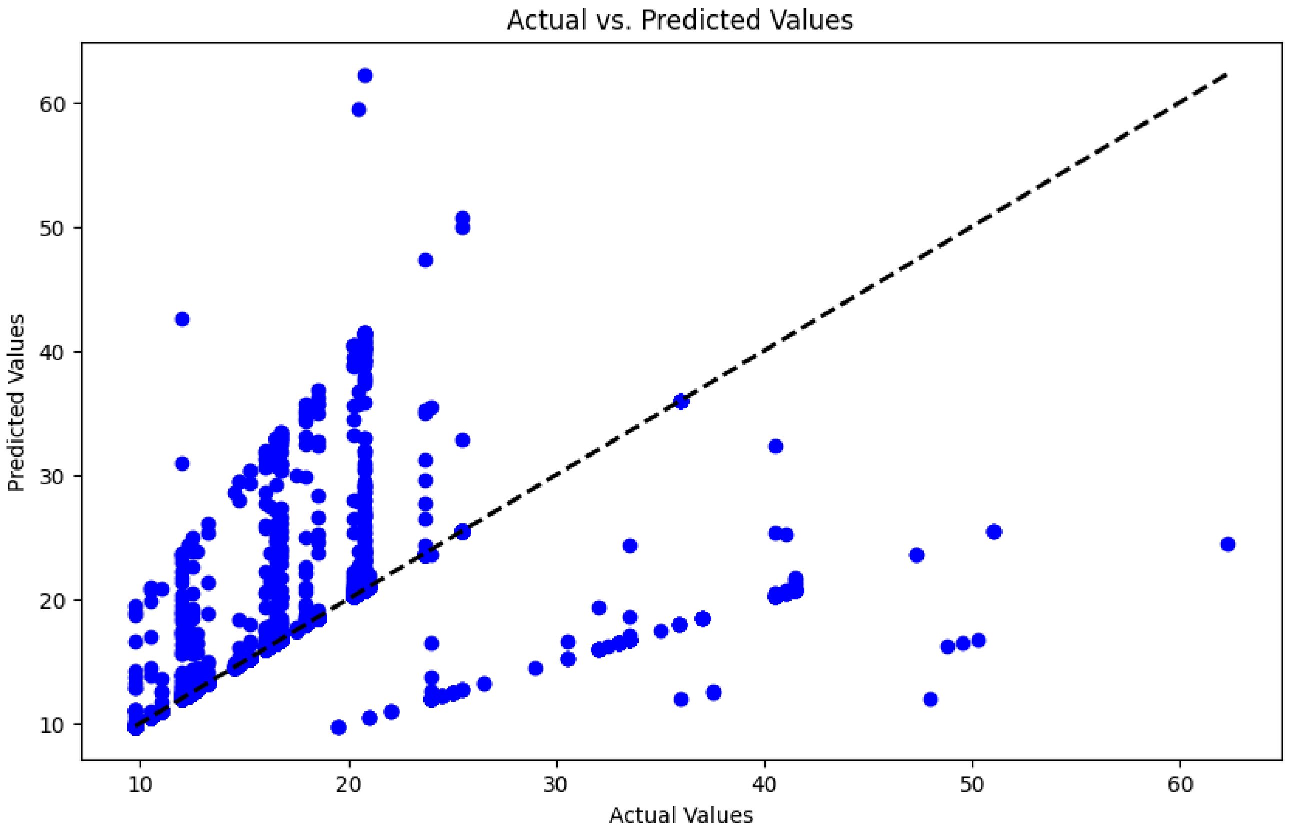
```
from sklearn.svm import SVR
```

```
model = SVR(kernel='linear')
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
```

```
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mean_actual = y_test.mean()
mse_percentage = (mse / mean_actual) * 100
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error Percentage: {mse_percentage:.2f}%")
print(f"R-squared: {r2}")
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.show()
```

```
Mean Squared Error: 11.321205793723127
Mean Absolute Error (MAE): 0.7048968345788212
Mean Squared Error Percentage: 67.47%
R-squared: 0.43429470070674936
```



EXTRA TREES

```
from sklearn.ensemble import ExtraTreesRegressor
```

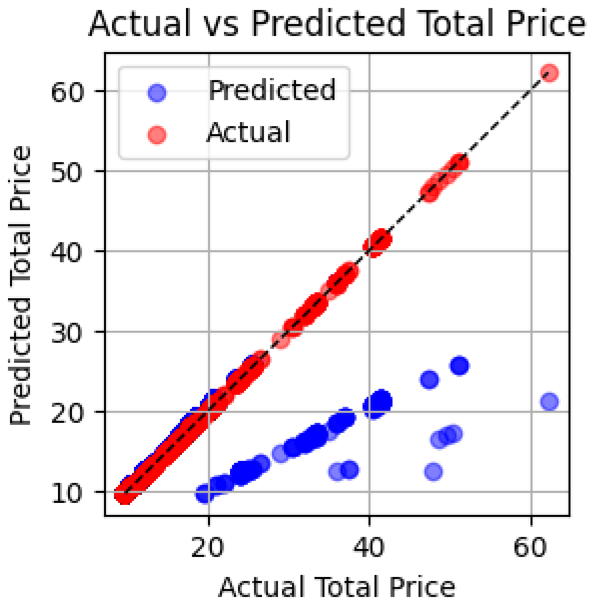
```
model = ExtraTreesRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
▾         ExtraTreesRegressor
ExtraTreesRegressor(random_state=42)
```

```
y_pred = model.predict(X_test)
```

```
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mean_actual = y_test.mean()
mse_percentage = (mse / mean_actual) * 100
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error Percentage: {mse_percentage:.2f}%")
print(f"R-squared: {r2}")
plt.figure(figsize=(3, 3))
plt.scatter(y_test, y_pred, color='b', label='Predicted', alpha=0.5)
plt.scatter(y_test, y_test, color='r', label='Actual', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=1)
plt.xlabel('Actual Total Price')
plt.ylabel('Predicted Total Price')
plt.title('Actual vs Predicted Total Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
Mean Squared Error: 6.103121023852163
Mean Absolute Error (MAE): 0.6428497160727766
Mean Squared Error Percentage: 36.37%
R-squared: 0.6950353197063653
```



RidgeRegression

```
from sklearn.linear_model import Ridge
```

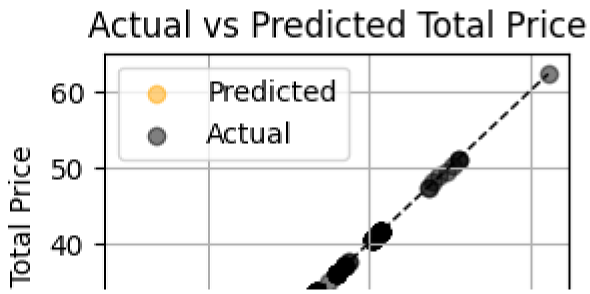
```
model = Ridge(alpha=1.0, random_state=42)
model.fit(X_train, y_train)
```

```
▾       Ridge
Ridge(random_state=42)
```

```
y_pred = model.predict(X_test)
```

```
most_sold_pizza = df['pizza_name'].value_counts().idxmax()
print(f"Predicted Price: {predicted_prices.mean()}")
print(f"Most Sold Pizza: {most_sold_pizza}")
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mean_actual = y_test.mean()
mse_percentage = (mse / mean_actual) * 100
mape = mean_absolute_error(y_test, y_pred) / mean_actual * 100
rmse=np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")
print(f"Mean Squared Error Percentage: {mse_percentage:.2f}%")
print(f"R-squared: {r2}")
import matplotlib.pyplot as plt
plt.figure(figsize=(3, 3))
plt.scatter(y_test, y_pred, color='orange', label='Predicted', alpha=0.5)
plt.scatter(y_test, y_test, color='black', label='Actual', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=1)
plt.xlabel('Actual Total Price')
plt.ylabel('Predicted Total Price')
plt.title('Actual vs Predicted Total Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
Predicted Price: 16.821224685681262
Most Sold Pizza: The Classic Deluxe Pizza
Root Mean Squared Error (RMSE): 3.4847580042757946
Mean Squared Error: 12.143538348364219
Mean Absolute Error (MAE): 1.642680742325806
Mean Absolute Error Percentage (MAPE): 9.79%
Mean Squared Error Percentage: 72.37%
R-squared: 0.39320385822778436
```



XG BOOST

```
import xgboost as xgb
```

```
model = xgb.XGBRegressor(objective ='reg:squarederror', random_state=42)
model.fit(X_train, y_train)
```

```
▾                               XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

```
y_pred = model.predict(X_test)
```

```
most_sold_pizza = df['pizza_name'].value_counts().idxmax()
print(f"Predicted Price: {predicted_prices.mean()}")
print(f"Most Sold Pizza: {most_sold_pizza}")
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mean_actual = y_test.mean()
```