Pizza Price Prediction

May 30, 2023

[214]:

```
#import libraries

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
%matplotlib inline

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor

from sklearn.model_selection import GridSearchCV

from sklearn.metrics import mean_absolute_error
import sklearn.metrics as metrics
import math
import os
import warnings
warnings.filterwarnings("ignore")
```

[215]:

```
# Load dataset
df = pd.read_csv("pizza_v2.csv")
```

[216]:

```
df
```

[216]:

```
    company price_rupiah  diameter      topping           variant     size  \
0         A    Rp235,000   22 inch      chicken  double_signature    jumbo
1         A    Rp198,000   20 inch    papperoni  double_signature    jumbo
2         A    Rp120,000   16 inch    mushrooms  double_signature  reguler
3         A    Rp155,000   14 inch  smoked_beef  double_signature  reguler
4         A    Rp248,000   18 inch   mozzarella  double_signature    jumbo
..      ...          ...       ...          ...               ...      ...
124       E     Rp39,000  8.5 inch         tuna        spicy tuna    small
125       E     Rp72,000   12 inch         tuna        spicy tuna   medium
```

```
126        E     Rp99,000   14 inch         tuna        spicy tuna    large
127        E     Rp44,000  8.5 inch         meat    BBQ_meat_fiesta    small
128        E     Rp78,000   12 inch         meat    BBQ_meat_fiesta   medium

    extra_sauce extra_cheese extra_mushrooms
0           yes          yes              no
1           yes          yes              no
2           yes          yes             yes
3           yes           no             yes
4           yes           no             yes
..          ...          ...             ...
124         yes          yes             yes
125         yes          yes             yes
126         yes          yes             yes
127         yes           no             yes
128          no           no             yes

[129 rows x 9 columns]
```

[217]:

```
# Getting the information about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129 entries, 0 to 128
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   company          129 non-null    object
 1   price_rupiah     129 non-null    object
 2   diameter         129 non-null    object
 3   topping          129 non-null    object
 4   variant          129 non-null    object
 5   size             129 non-null    object
 6   extra_sauce      129 non-null    object
 7   extra_cheese     129 non-null    object
 8   extra_mushrooms  129 non-null    object
dtypes: object(9)
memory usage: 9.2+ KB
```

[218]:

```
#pre processing

# Removing Rp
df['price_rupiah'] = df['price_rupiah'].str.replace('Rp', '').str.replace(',',␣
 ↪'').astype('float64')
# Removing Inch
df['diameter'] = df['diameter'].str.replace('inch', '').str.replace(',', '').
 ↪astype('float64')
```

```
#here we convert objects type numeric values into numbers.

df.head()
```

[218]:

```
  company  price_rupiah  diameter      topping           variant     size  \
0       A      235000.0      22.0      chicken  double_signature    jumbo
1       A      198000.0      20.0    papperoni  double_signature    jumbo
2       A      120000.0      16.0    mushrooms  double_signature  reguler
3       A      155000.0      14.0   smoked_beef  double_signature  reguler
4       A      248000.0      18.0   mozzarella  double_signature    jumbo

  extra_sauce extra_cheese extra_mushrooms
0         yes          yes              no
1         yes          yes              no
2         yes          yes             yes
3         yes           no             yes
4         yes           no             yes
```

[219]:

```
#pre processing
df.isnull().sum()

#here we can see no missing values.
```

[219]:

```
company            0
price_rupiah       0
diameter           0
topping            0
variant            0
size               0
extra_sauce        0
extra_cheese       0
extra_mushrooms    0
dtype: int64
```

[220]:

```
df.describe()
```

[220]:

```
        price_rupiah    diameter
count     129.000000  129.000000
mean    87151.162791   12.976744
std     44706.097732    3.272674
min     23500.000000    8.000000
25%     51000.000000   12.000000
50%     78000.000000   12.000000
75%    105000.000000   14.000000
max    248000.000000   22.000000
```

```
[221]: df.dtypes
```

```
[221]: company              object
       price_rupiah        float64
       diameter            float64
       topping              object
       variant              object
       size                 object
       extra_sauce          object
       extra_cheese         object
       extra_mushrooms      object
       dtype: object
```

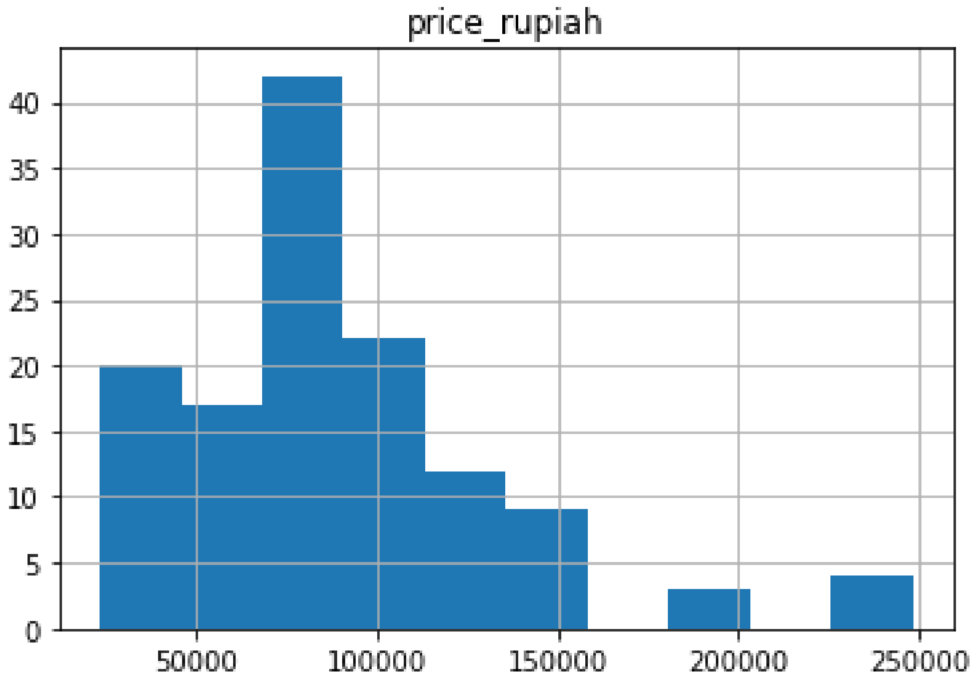
```
[222]: # check unique values

       df.nunique(axis=0)
```

```
[222]: company             5
       price_rupiah       43
       diameter           11
       topping            12
       variant            20
       size                6
       extra_sauce         2
       extra_cheese        2
       extra_mushrooms     2
       dtype: int64
```

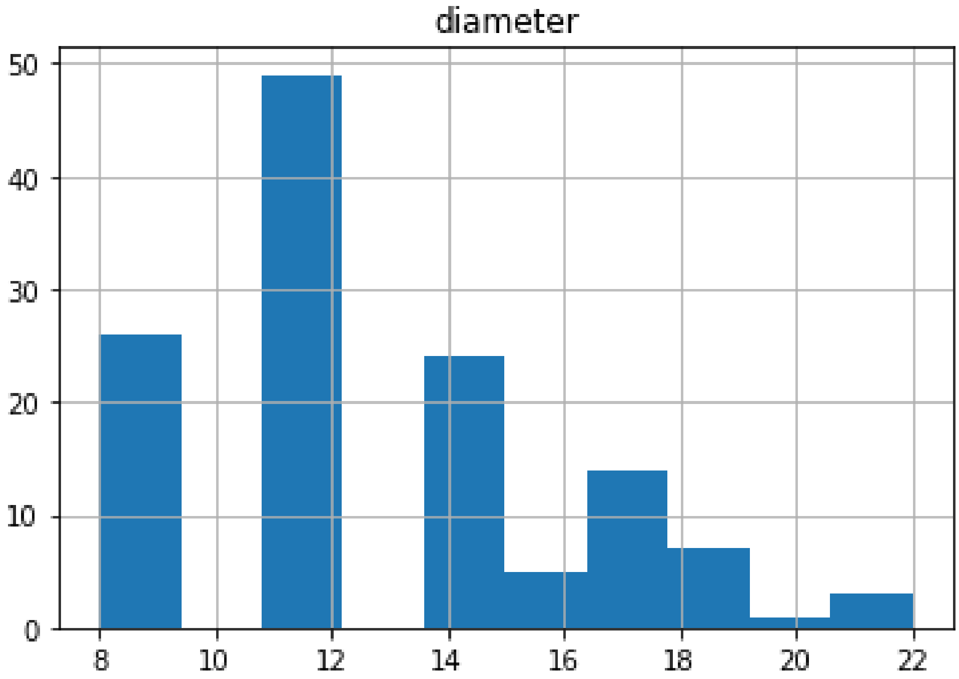
```
[223]: df.hist(column='price_rupiah')
```

```
[223]: array([[<AxesSubplot:title={'center':'price_rupiah'}>]], dtype=object)
```



```
[224]: df.hist(column='diameter')
```

```
[224]: array([[<AxesSubplot:title={'center':'diameter'}>]], dtype=object)
```

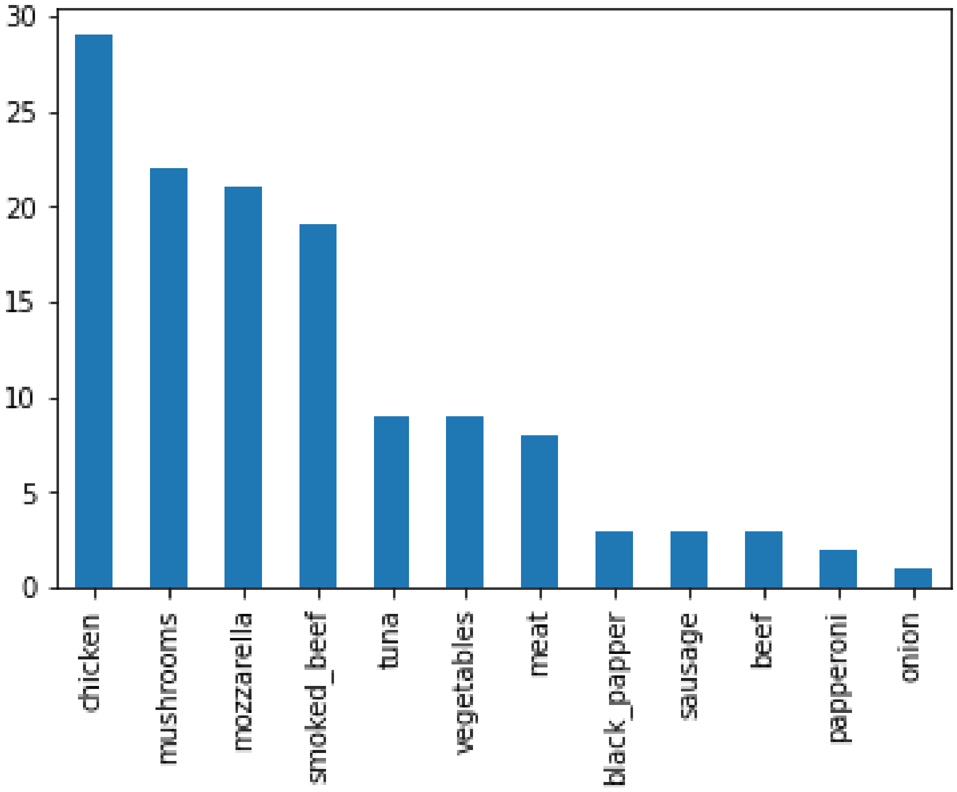


[225]:

```
df['topping'].value_counts().plot(kind='bar')
```

[225]:

```
<AxesSubplot:>
```

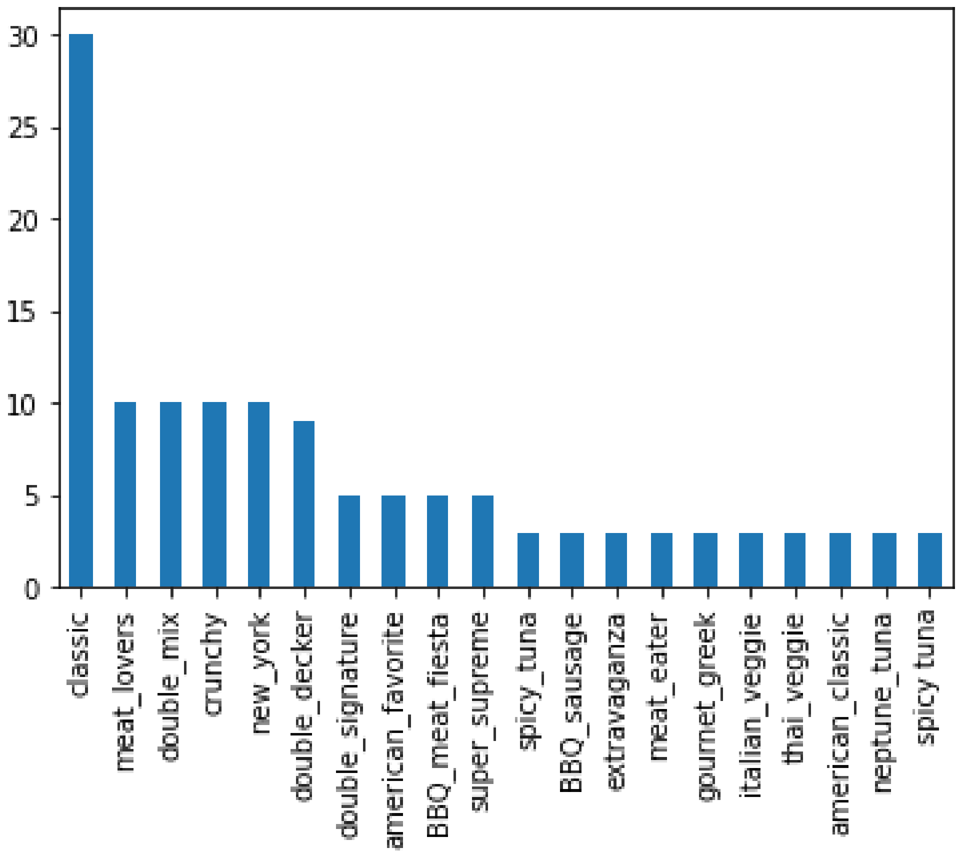


[226]:

```
df['variant'].value_counts().plot(kind='bar')
```

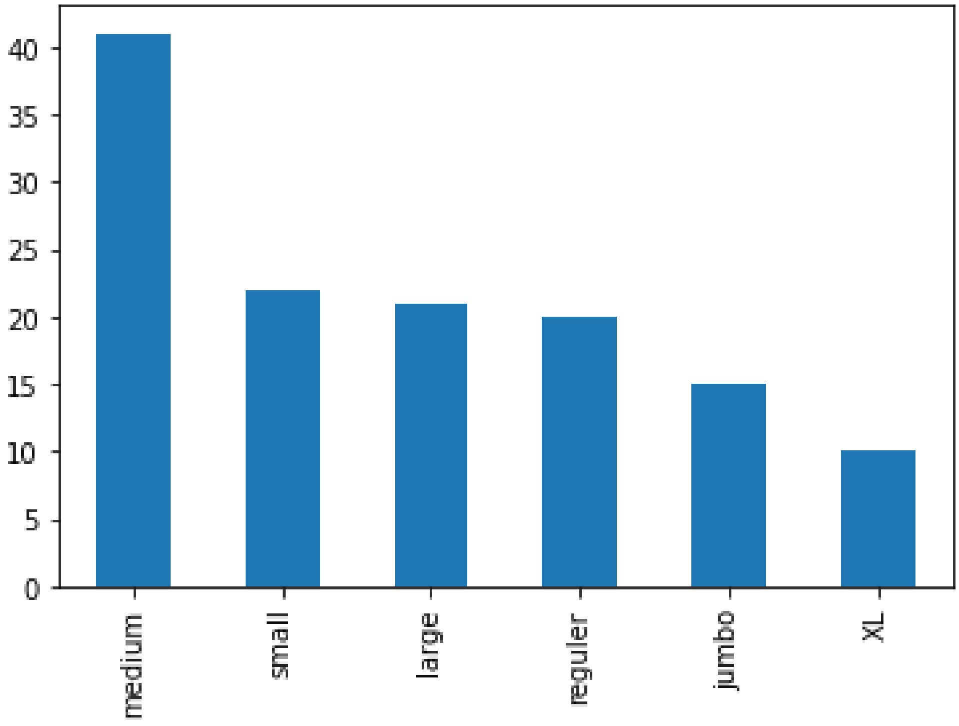
[226]:

```
<AxesSubplot:>
```



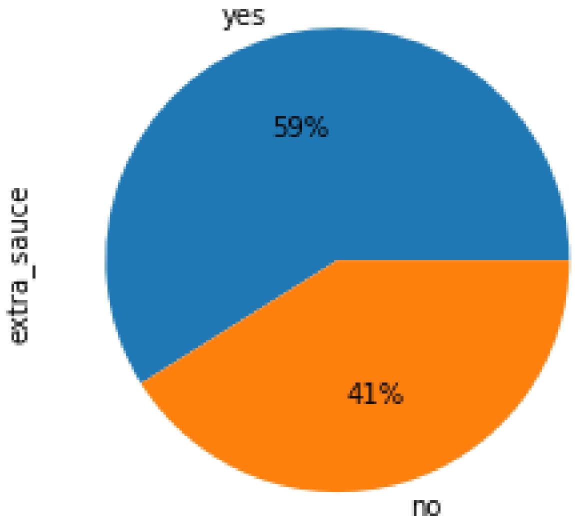
```
[227]: df['size'].value_counts().plot(kind='bar')
```

```
[227]: <AxesSubplot:>
```



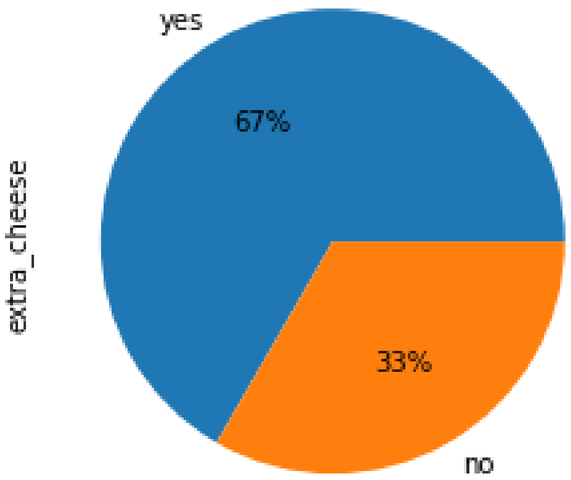
```
[228]: df['extra_sauce'].value_counts().plot(kind='pie',autopct='%1.0f%%')
```

```
[228]: <AxesSubplot:ylabel='extra_sauce'>
```



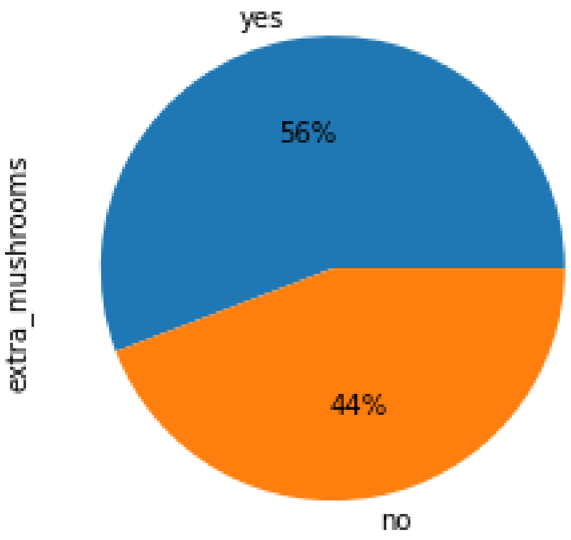
```
[229]: df['extra_cheese'].value_counts().plot(kind='pie',autopct='%1.0f%%')
```

```
[229]: <AxesSubplot:ylabel='extra_cheese'>
```



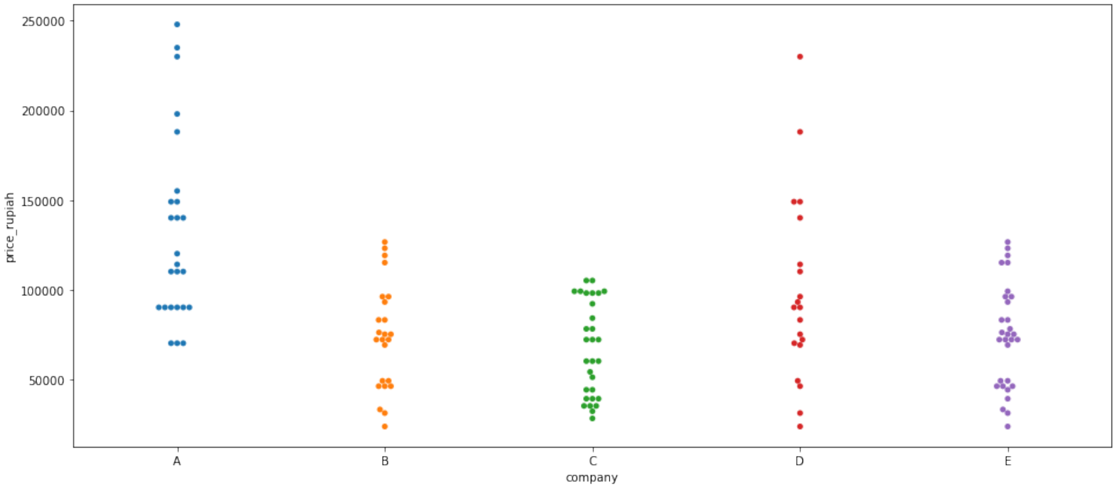
```
[230]: df['extra_mushrooms'].value_counts().plot(kind='pie',autopct='%1.0f%%')
```

```
[230]: <AxesSubplot:ylabel='extra_mushrooms'>
```



[231]:

```
plt.figure(figsize=(16, 7))
sns.swarmplot(x='company', y='price_rupiah', data=df)
plt.show()
```



[232]:

```
#replace categorical data values using dummy variables.

encoder = LabelEncoder()
cato_col = [col for col in df.columns if df[col].dtype == 'object']

for cols in cato_col:
    df[cols] = encoder.fit_transform(df[cols])
```

[233]:

```
df.head()
```

[233]:

```
   company  price_rupiah  diameter  topping  variant  size  extra_sauce  \
0        0      235000.0      22.0        2        8     1            1
1        0      198000.0      20.0        7        8     1            1
2        0      120000.0      16.0        5        8     4            1
3        0      155000.0      14.0        9        8     4            1
4        0      248000.0      18.0        4        8     1            1

   extra_cheese  extra_mushrooms
0             1                0
1             1                0
2             1                1
3             0                1
4             0                1
```

[234]:

```
#Split data into train and test using libries
```

```
y = pd.DataFrame(df["price_rupiah"])
X = df.drop("price_rupiah",axis = 1)
```

[235]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)
```

[236]:
```
# Define the model and fit
model = LinearRegression()
model.fit(X_train, y_train)
```

[236]:
```
LinearRegression()
```

[237]:
```
# Get predictions
predictions = model.predict(X_test)
```

[238]:
```
#Evaluating trainning data using these 3 metrics

# Calculate MAE
mae = mean_absolute_error(y_test,predictions)
print("Mean Absolute Error:" , mae)

#Calculating R2
r2 =  metrics.r2_score(y_test, predictions)
print("R2 score :", r2)

mse = math.sqrt(metrics.mean_squared_error(y_test, predictions))
print(f'Root MSE :',mse)
```

```
Mean Absolute Error: 17253.909449342045
R2 score : 0.4812529133111144
Root MSE : 21173.28470590215
```

[239]:
```
# Define the hyperparameters and their values for tuning
parameters = {'fit_intercept': [True, False],'normalize': [True, False]}

# Create a GridSearchCV object and fit it to the data
grid_search = GridSearchCV(model, parameters, scoring='neg_mean_squared_error',␣
 ↪cv=5)
grid_search.fit(X_test, y_test)

# Get the best hyperparameters and model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

# Evaluate the best model
y_pred = best_model.predict(X_test)
print("Best hyperparameters:", best_params)
```

```
#Evaluating the tuned model testing data using these 3 metrics

mae = mean_absolute_error(y_test,y_pred)
print("Mean Absolute Error:" , mae)
r2 =  metrics.r2_score(y_test, y_pred)
print("R2 score :", r2)
mse = math.sqrt(metrics.mean_squared_error(y_test, y_pred))
print(f'Root MSE :',mse)
```

```
Best hyperparameters: {'fit_intercept': False, 'normalize': True}
Mean Absolute Error: 10494.731035178569
R2 score : 0.7972146621329157
Root MSE : 13238.192051555461
```

[240]:

```
# Define the model
model2 = XGBRegressor()

# Fit the model
model2.fit(X_train,y_train)

# Get predictions
predictions2 = model2.predict(X_test)

#Evaluating the new model testing data using these 3 metrics

# Calculate MAE
mae = mean_absolute_error(y_test,predictions2)
print("Mean Absolute Error:" , mae)

#Calculating R2
r2 =  metrics.r2_score(y_test, predictions2)
print("R2 score :", r2)

mse = math.sqrt(metrics.mean_squared_error(y_test, predictions2))
print(f'Root MSE :',mse)
```

```
Mean Absolute Error: 6103.510817307692
R2 score : 0.8999900325696046
Root MSE : 9296.769103201883
```