Superstore Sales Prediction using Machine Learning

February 5, 2024

0.1 Importing the relevant libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings, joblib, gc, re, time
warnings.filterwarnings('ignore')
from scipy.stats import probplot
from sklearn.model_selection import train_test_split, RandomizedSearchCV,␣
 ↪cross_val_score
from sklearn.metrics import mean_squared_error, mean_absolute_error,␣
 ↪mean_absolute_percentage_error, r2_score
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor,␣
 ↪AdaBoostRegressor, ExtraTreesRegressor, HistGradientBoostingRegressor,␣
 ↪BaggingRegressor, VotingRegressor, StackingRegressor
from sklearn.linear_model import LinearRegression, Lasso, Ridge, ElasticNet,␣
 ↪BayesianRidge, HuberRegressor, PassiveAggressiveRegressor, SGDRegressor,␣
 ↪PoissonRegressor, RANSACRegressor, TheilSenRegressor, GammaRegressor,␣
 ↪ARDRegression, TweedieRegressor
from sklearn.neighbors import KNeighborsRegressor, RadiusNeighborsRegressor
from sklearn.svm import SVR, LinearSVR, NuSVR
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor, XGBRFRegressor
from lightgbm import LGBMRegressor
from catboost import CatBoostRegressor
from ngboost import NGBRegressor
from sklearn.neural_network import MLPRegressor
from skopt import BayesSearchCV
from skopt.space import Real, Categorical, Integer
from sklearn.preprocessing import StandardScaler, FunctionTransformer,␣
 ↪PowerTransformer, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import SelectKBest, f_regression,␣
 ↪SelectPercentile, SelectFromModel, RFE, SequentialFeatureSelector
from mlxtend.plotting import plot_decision_regions
```

```
from feature_engine.outliers import Winsorizer
from sklearn.pipeline import Pipeline
from feature_engine.selection import DropConstantFeatures,␣
 ↪DropDuplicateFeatures, DropCorrelatedFeatures
from sklearn.impute import SimpleImputer
from yellowbrick.regressor import PredictionError, ResidualsPlot
```

0.2 Loading the dataset

[2]:
```
df = pd.read_excel('Superstore.xlsx')
df.head()
```

[2]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	…
0	1	CA-2013-152156	2013-11-09	2013-11-12	Second Class	CG-12520	…
1	2	CA-2013-152156	2013-11-09	2013-11-12	Second Class	CG-12520	…
2	3	CA-2013-138688	2013-06-13	2013-06-17	Second Class	DV-13045	…
3	4	US-2012-108966	2012-10-11	2012-10-18	Standard Class	SO-20335	…
4	5	US-2012-108966	2012-10-11	2012-10-18	Standard Class	SO-20335	…

	Customer Name	Segment	Country	City	…
0	Claire Gute	Consumer	United States	Henderson	…
1	Claire Gute	Consumer	United States	Henderson	…
2	Darrin Van Huff	Corporate	United States	Los Angeles	…
3	Sean O'Donnell	Consumer	United States	Fort Lauderdale	…
4	Sean O'Donnell	Consumer	United States	Fort Lauderdale	…

	Postal Code	Region	Product ID	Category	Sub-Category
0	42420	South	FUR-BO-10001798	Furniture	Bookcases
1	42420	South	FUR-CH-10000454	Furniture	Chairs
2	90036	West	OFF-LA-10000240	Office Supplies	Labels
3	33311	South	FUR-TA-10000577	Furniture	Tables
4	33311	South	OFF-ST-10000760	Office Supplies	Storage

	Product Name	Sales	Quantity
0	Bush Somerset Collection Bookcase	261.9600	2
1	Hon Deluxe Fabric Upholstered Stacking Chairs,…	731.9400	3
2	Self-Adhesive Address Labels for Typewriters b…	14.6200	2
3	Bretford CR4500 Series Slim Rectangular Table	957.5775	5
4	Eldon Fold 'N Roll Cart System	22.3680	2

	Discount	Profit
0	0.00	41.9136
1	0.00	219.5820
2	0.00	6.8714
3	0.45	-383.0310
4	0.20	2.5164

```
[5 rows x 21 columns]
```

0.3 Data Exploration

[3]:
```
df.shape
```

[3]:
```
(9994, 21)
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.6+ MB
```

[5]:
```
df.drop('Row ID', axis=1, inplace=True)
```

[6]:
```
df.nunique() / len(df)
```

[6]:
```
Order ID        0.501201
Order Date      0.123874
Ship Date       0.133480
Ship Mode       0.000400
Customer ID     0.079348
```

```
Customer Name    0.079348
Segment          0.000300
Country          0.000100
City             0.053132
State            0.004903
Postal Code      0.063138
Region           0.000400
Product ID       0.186312
Category         0.000300
Sub-Category     0.001701
Product Name     0.184211
Sales            0.614769
Quantity         0.001401
Discount         0.001201
Profit           0.754953
dtype: float64
```

[7]:
```
df.isna().sum()
```

[7]:
```
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

[8]:
```
df.duplicated().sum()
```

[8]:
```
1
```

[9]:
```
df = df.drop_duplicates()
df.shape
```

[9]: (9993, 20)

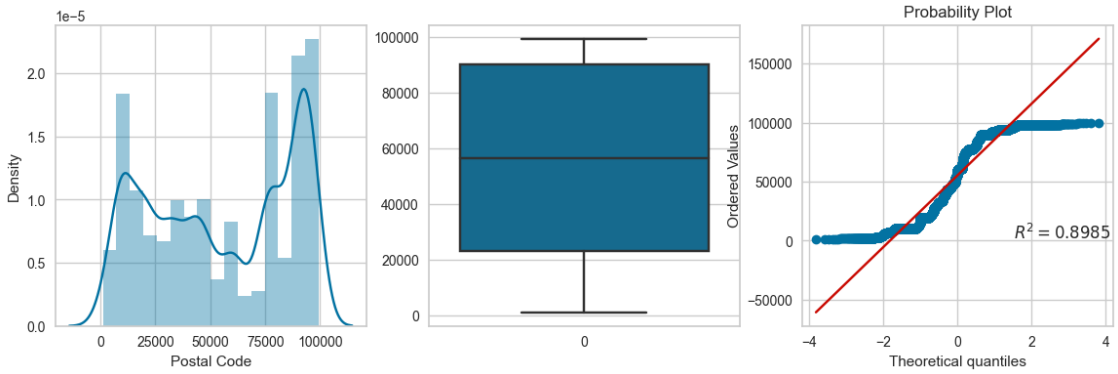
[10]:
```
df.drop(['Order ID','Customer ID','Customer Name','Product␣
 ↪ID'],axis=1,inplace=True)
```

0.4 Exploratory Data Analysis

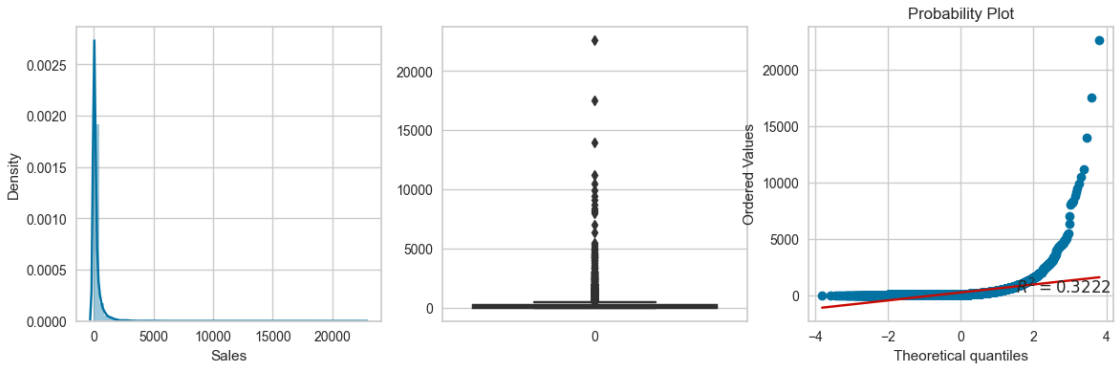
0.4.1 Univariate Analysis

[11]:
```
for col in df.select_dtypes(np.number).columns:
    print(f"Skewness of {col}:",df[col].skew())
    print(f"Kurtosis of {col}:",df[col].kurt())
    plt.subplots(nrows=1,ncols=2,figsize=(14,4))
    plt.subplot(1,3,1)
    sns.distplot(df[col])
    plt.subplot(1,3,2)
    sns.boxplot(df[col])
    plt.subplot(1,3,3)
    probplot(df[col],plot=plt,dist='norm',rvalue=True)
    plt.show()
```

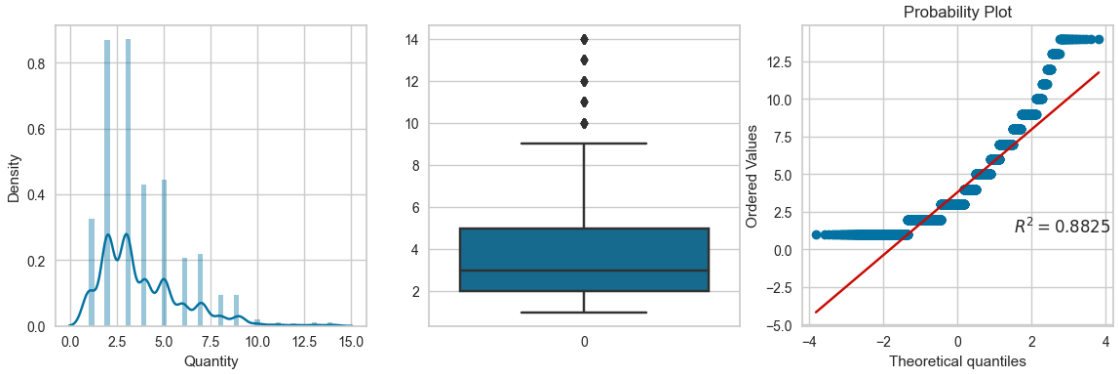
```
Skewness of Postal Code: -0.12862858810335934
Kurtosis of Postal Code: -1.4931118549547797
```



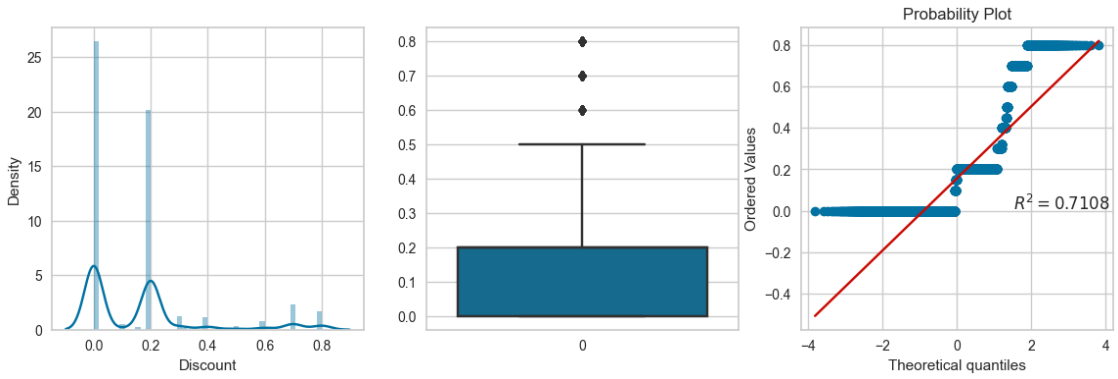

```
Skewness of Sales: 12.972141558363262
Kurtosis of Sales: 305.28176970634985
```




```
Skewness of Quantity: 1.2784155399852233
Kurtosis of Quantity: 1.9915827160047757
```

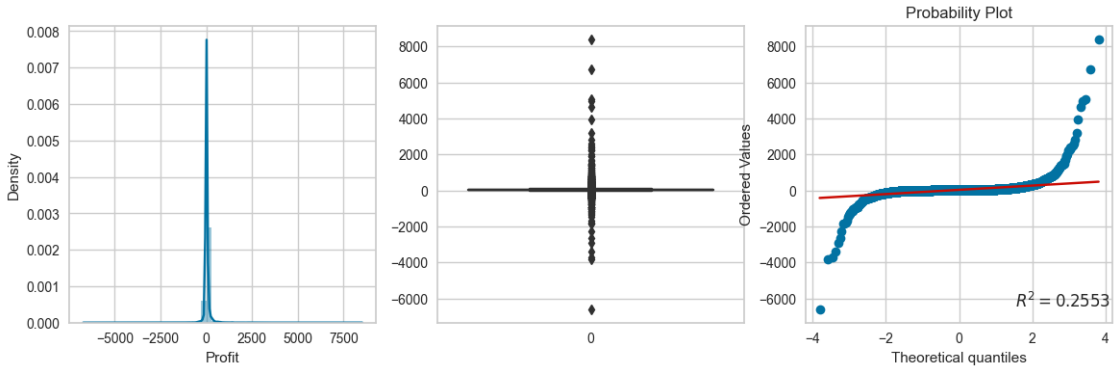



```
Skewness of Discount: 1.6845084876700462
Kurtosis of Discount: 2.409976524801519
```




```
Skewness of Profit: 7.561035996041436
```

```
Kurtosis of Profit: 397.15038474389223
```



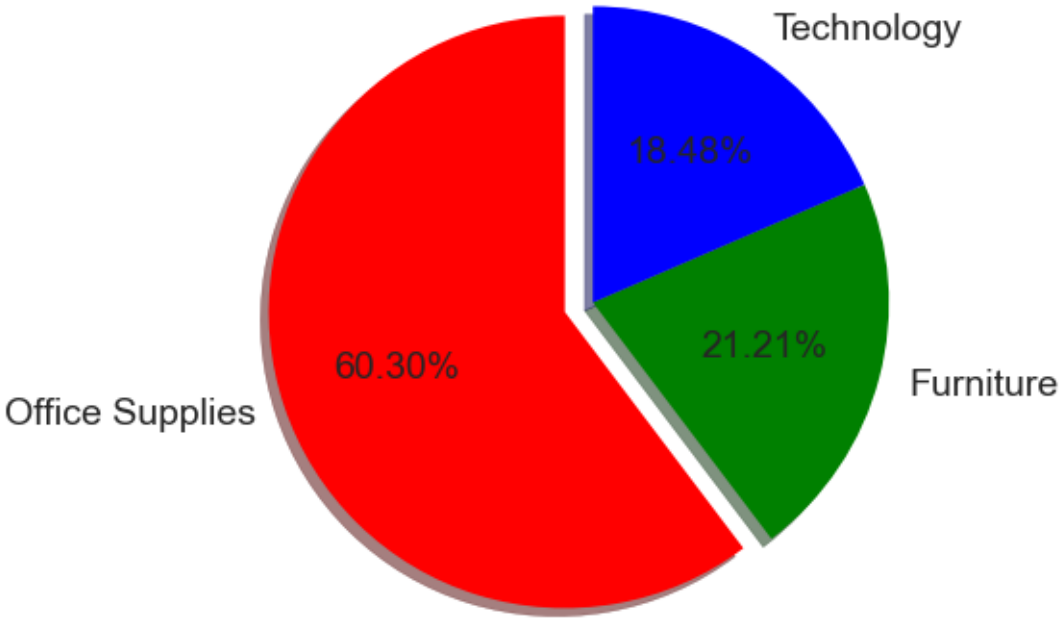
[12]:
```
skewed_cols = ['Quantity','Discount','Profit']
```

[13]:
```
df['Country'].value_counts()
```

[13]:
```
United States    9993
Name: Country, dtype: int64
```

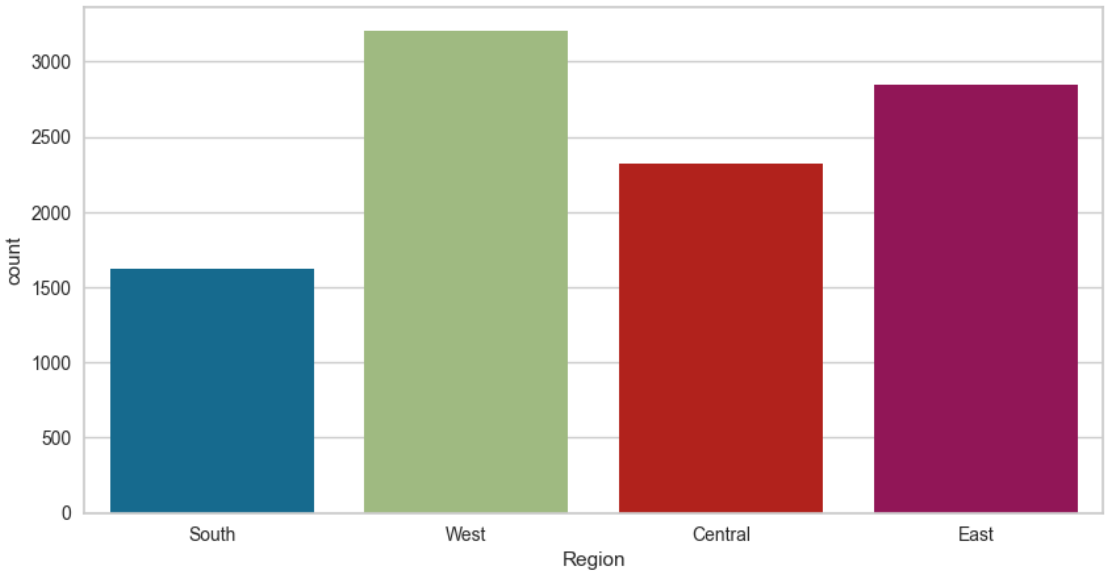
[14]:
```
df.drop('Country', axis=1, inplace=True)
```

[15]:
```
plt.figure(figsize=(10,5))
labels = df['Category'].value_counts().keys()
values = df['Category'].value_counts().values

plt.pie(values,labels=labels,autopct='%1.2f%%',shadow=True,explode=[0.
 ↪1,0,0],startangle=90,textprops={'fontsize':␣
 ↪14},colors=['red','green','blue'])
plt.show()
```



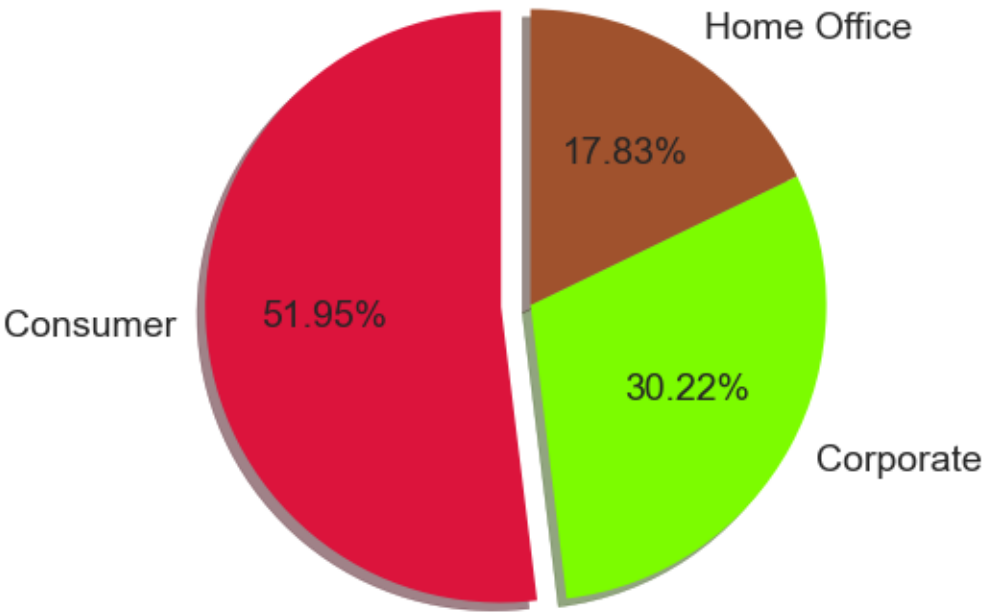
Majority of the products are in Office Supplies category whereas the least number of products are in Technology category.

[16]:
```
plt.figure(figsize=(10,5))
sns.countplot(x='Region',data=df)
plt.show()
```



A significant number of customers belong to the Western region while the Southern region has the least number of customers.

[17]:
```
plt.figure(figsize=(10,5))
labels = df['Segment'].value_counts().keys()
values = df['Segment'].value_counts().values

plt.pie(values,labels=labels,autopct='%1.2f%%',shadow=True,explode=[0.
 ↪1,0,0],startangle=90,textprops={'fontsize':␣
 ↪14},colors=['crimson','lawngreen','sienna'])
plt.show()
```

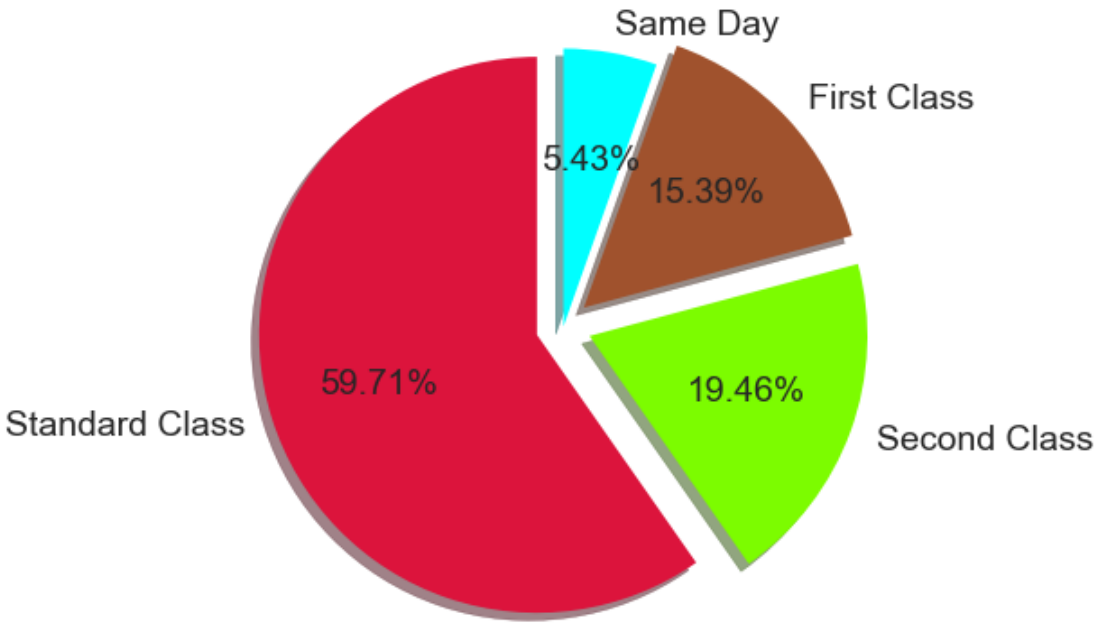


More than half of the entire population belongs to the Consumer segment.

[18]:
```
df['Ship Mode'].value_counts()
```

[18]:
```
Standard Class    5967
Second Class      1945
First Class       1538
Same Day           543
Name: Ship Mode, dtype: int64
```

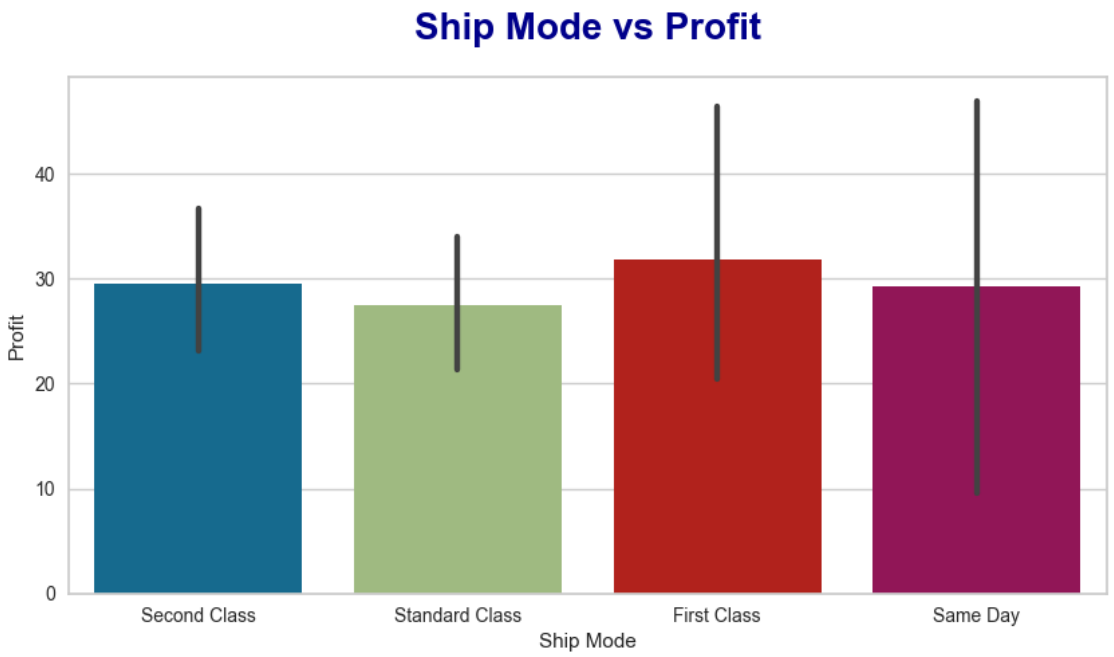
[19]:
```
plt.figure(figsize=(10,5))
labels = df['Ship Mode'].value_counts().keys()
values = df['Ship Mode'].value_counts().values

plt.pie(values,labels=labels,autopct='%1.2f%%',shadow=True,explode=[0.1,0.1,0.
↪1,0],startangle=90,textprops={'fontsize':␣
↪14},colors=['crimson','lawngreen','sienna','cyan'])
plt.show()
```



A significant number of orders were delivered via the Standard shipping mode whereas First Class and Second Class shipping modes were less frequently used.

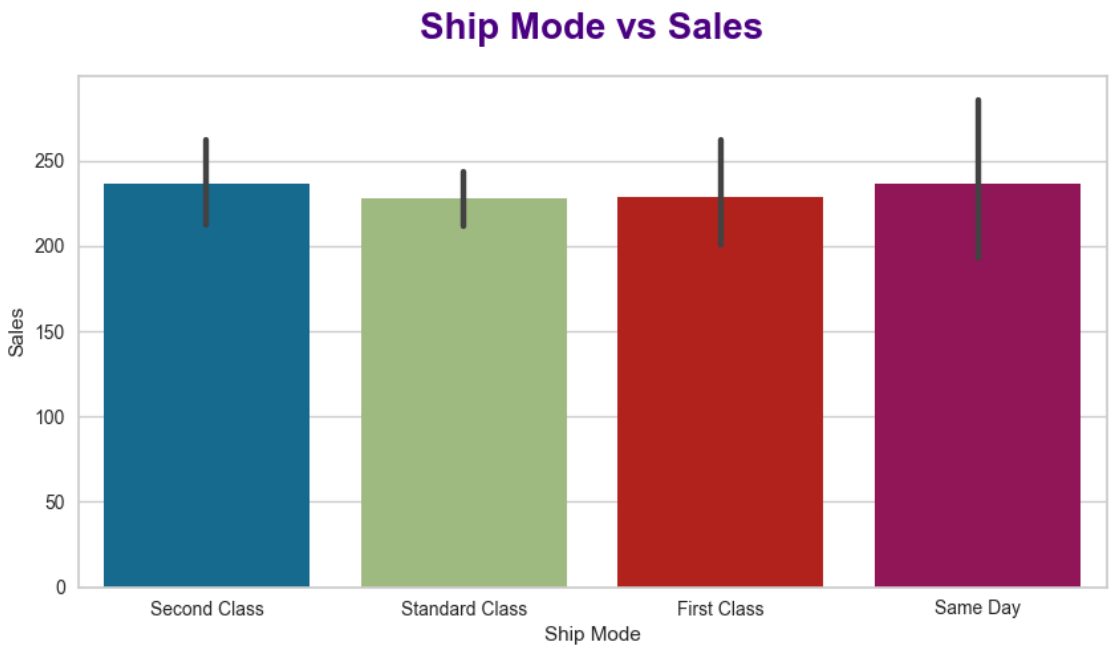
0.4.2 Bivariate Analysis

[20]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Ship Mode',y='Profit',data=df)
plt.title('Ship Mode vs␣
↪Profit',pad=20,fontsize=20,color='darkblue',fontweight='bold')
plt.show()
```



The First Class was the most profitable shipping mode while the Standard shipping mode was the least profitable.

[21]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Ship Mode',y='Sales',data=df)
plt.title('Ship Mode vs␣
 ↪Sales',pad=20,fontsize=20,fontweight='bold',color='indigo')
plt.show()
```

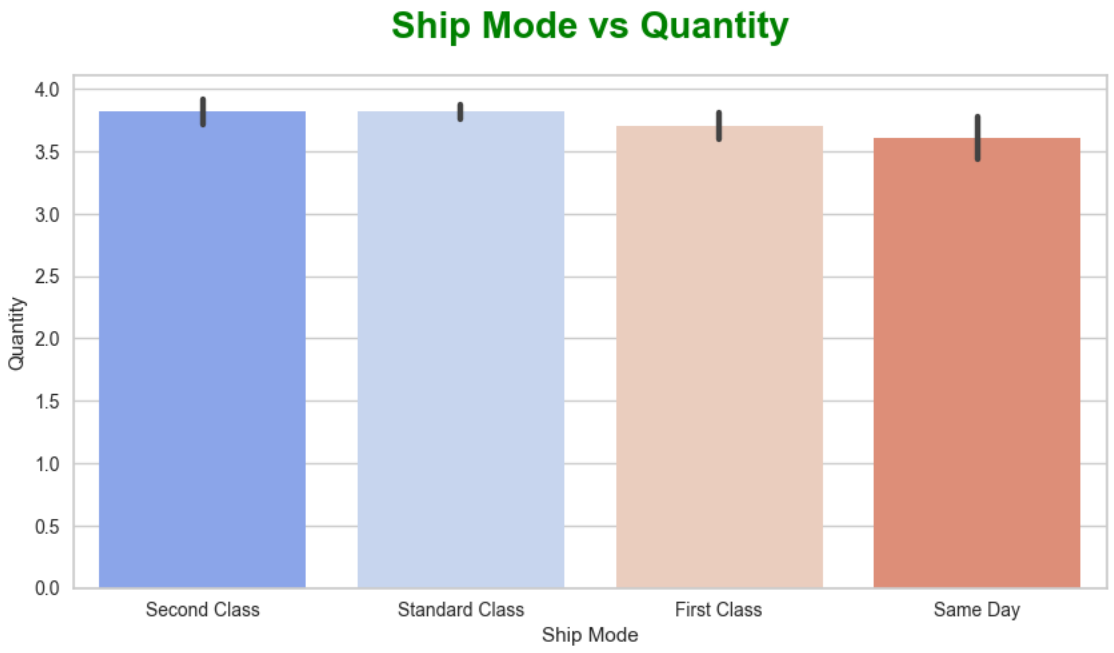


The shipping modes that generated the highest sales are Second Class and Same Day whereas Standard Class and First Class generated the least sales.

[22]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Ship Mode',y='Discount',data=df,palette='rocket')
plt.title('Ship Mode vs␣
  ↪Discount',pad=20,fontsize=20,fontweight='bold',color='tomato')
plt.show()
```

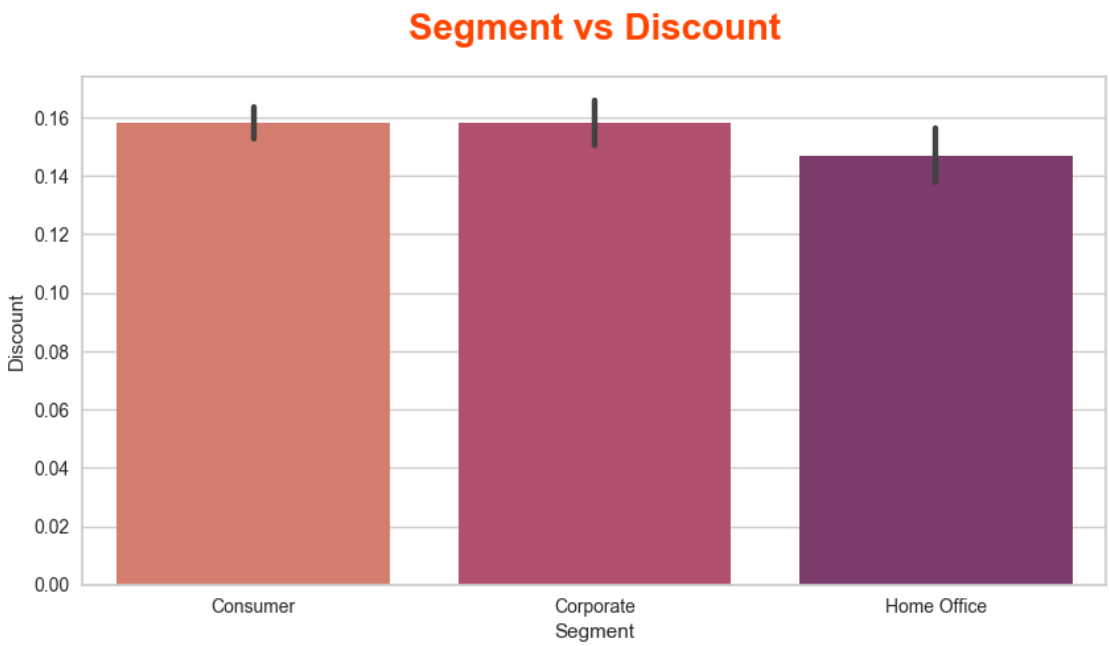
The maximum discount was offered on products that were delivered via the First Class shipping mode whereas the minimum discount was given on products that were shipped via the Second Class.

[23]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Ship Mode',y='Quantity',data=df,palette='coolwarm')
plt.title('Ship Mode vs␣
  ↪Quantity',pad=20,fontsize=20,fontweight='bold',color='green')
plt.show()
```



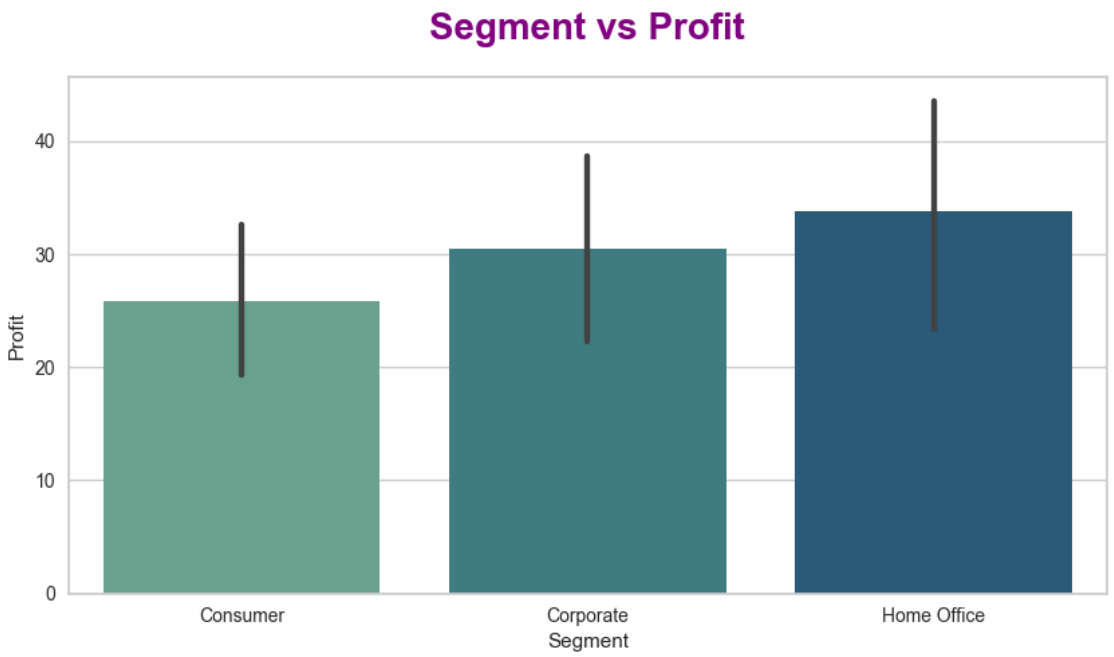
The highest quantities of products were shipped via the Second Class and Standard Class shipping modes whereas the least quantities were shipped through the Same Day shipping mode.

[24]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Segment',y='Discount',data=df,palette='flare')
plt.title('Segment vs␣
  ↪Discount',pad=20,fontsize=20,fontweight='bold',color='orangered')
plt.show()
```



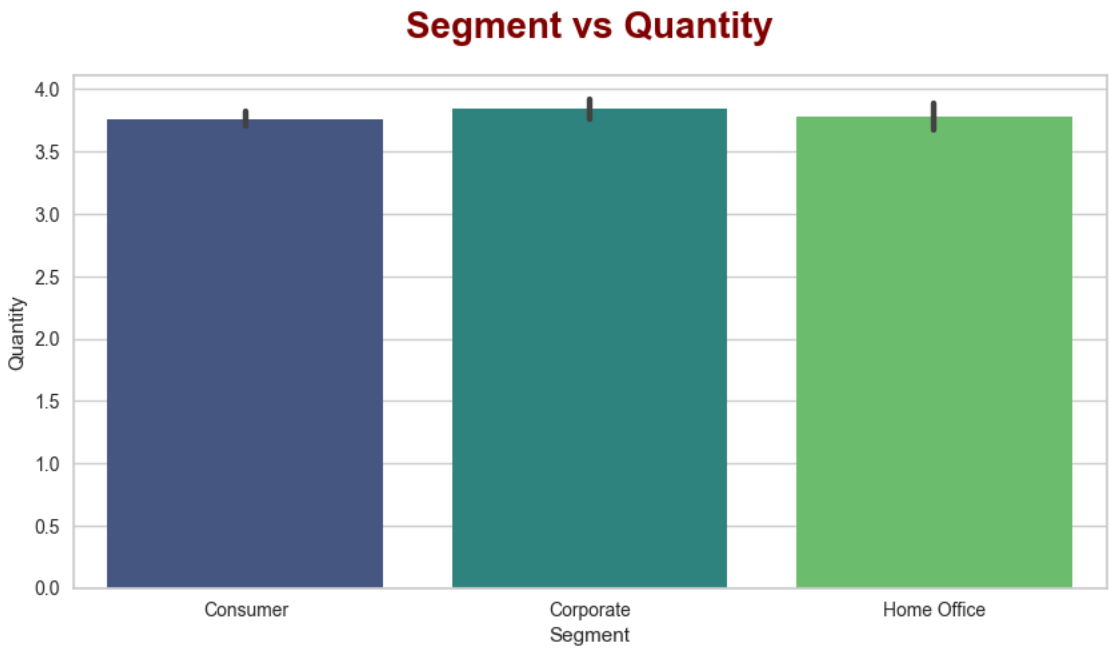
The highest discount was given on the orders made by Consumer and Corporate segments whereas those made by Home Office provided the least discount.

[25]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Segment',y='Profit',data=df,palette='crest')
plt.title('Segment vs␣
  ↪Profit',pad=20,fontsize=20,fontweight='bold',color='purple')
plt.show()
```



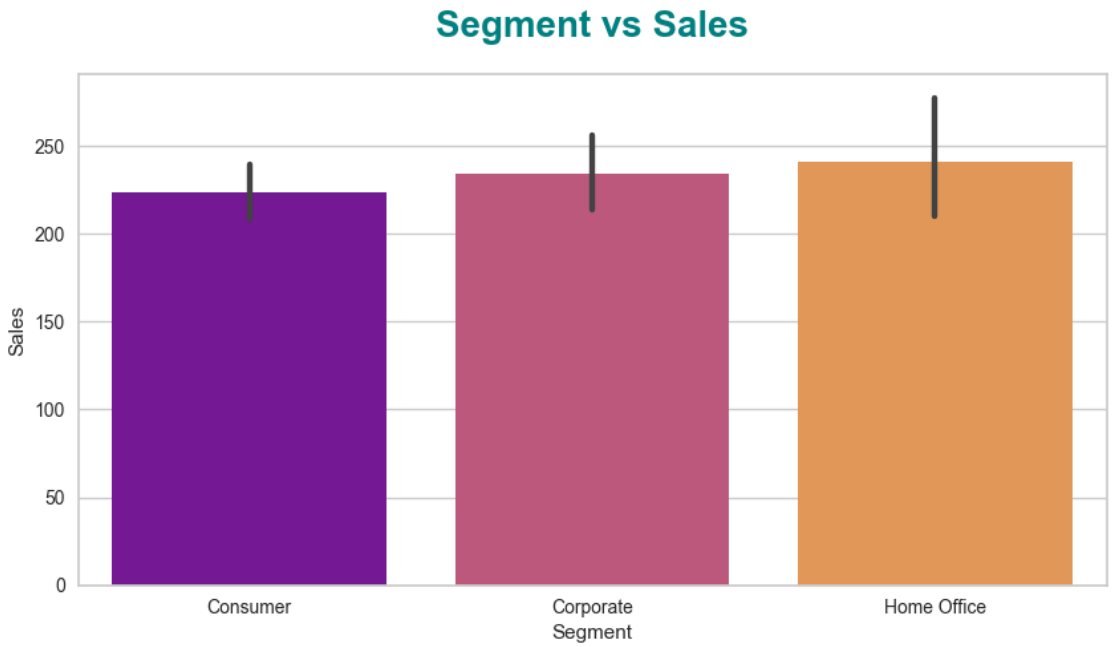
The orders made by Home Office segment generated the highest profit whereas those made by Consumer and Corporate segments generated the least profit.

[26]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Segment',y='Quantity',data=df,palette='viridis')
plt.title('Segment vs␣
  ↪Quantity',pad=20,fontsize=20,fontweight='bold',color='maroon')
plt.show()
```



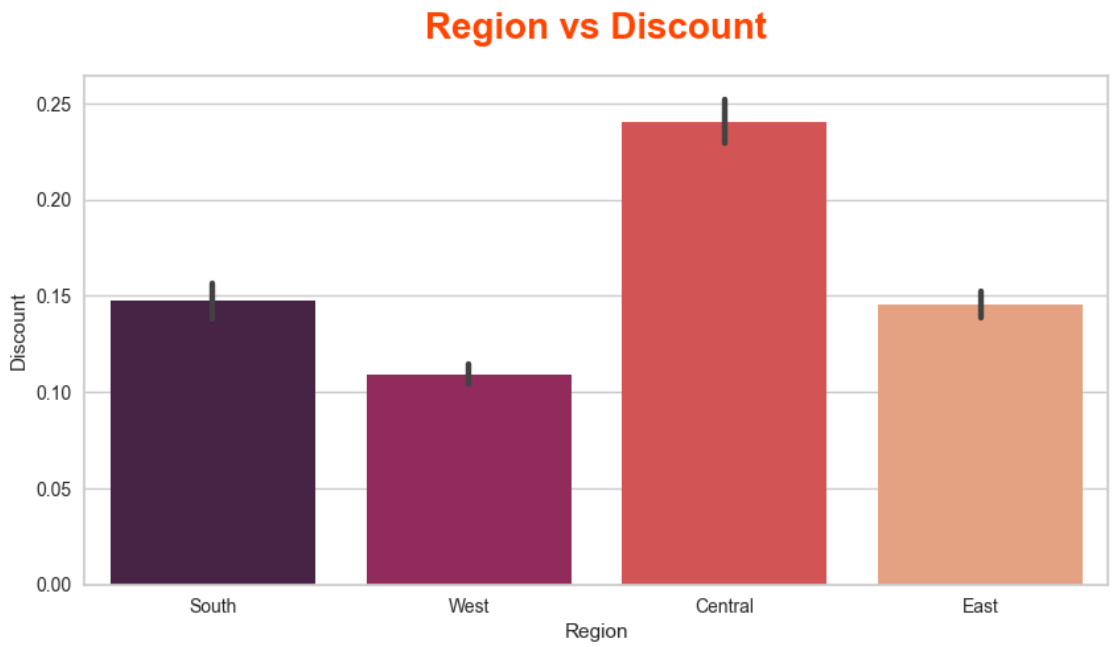
The highest quantities of products were purchased by the Corporate segment whereas the lowest quantities were purchased by the Consumer segment.

[27]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Segment',y='Sales',data=df,palette='plasma')
plt.title('Segment vs Sales',pad=20,fontsize=20,fontweight='bold',color='teal')
plt.show()
```



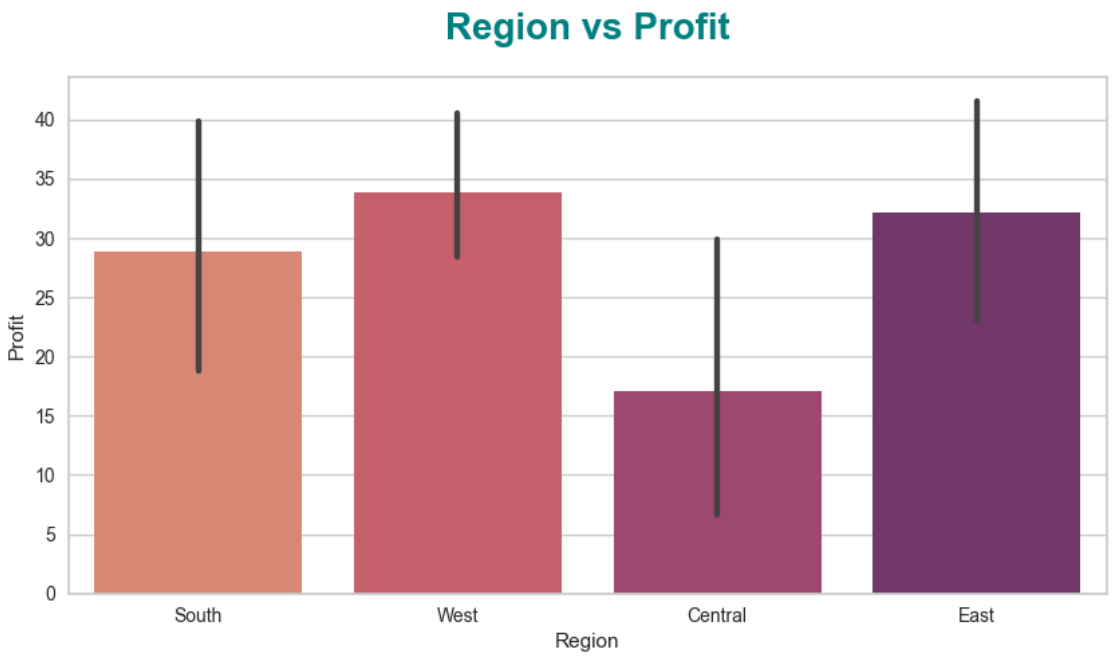
The orders made by the Home Office segment generated the maximum sales whereas those made by the Consumer segment generated the minimum sales.

[28]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Region',y='Discount',data=df,palette='rocket')
plt.title('Region vs␣
 ↪Discount',pad=20,fontsize=20,fontweight='bold',color='orangered')
plt.show()
```



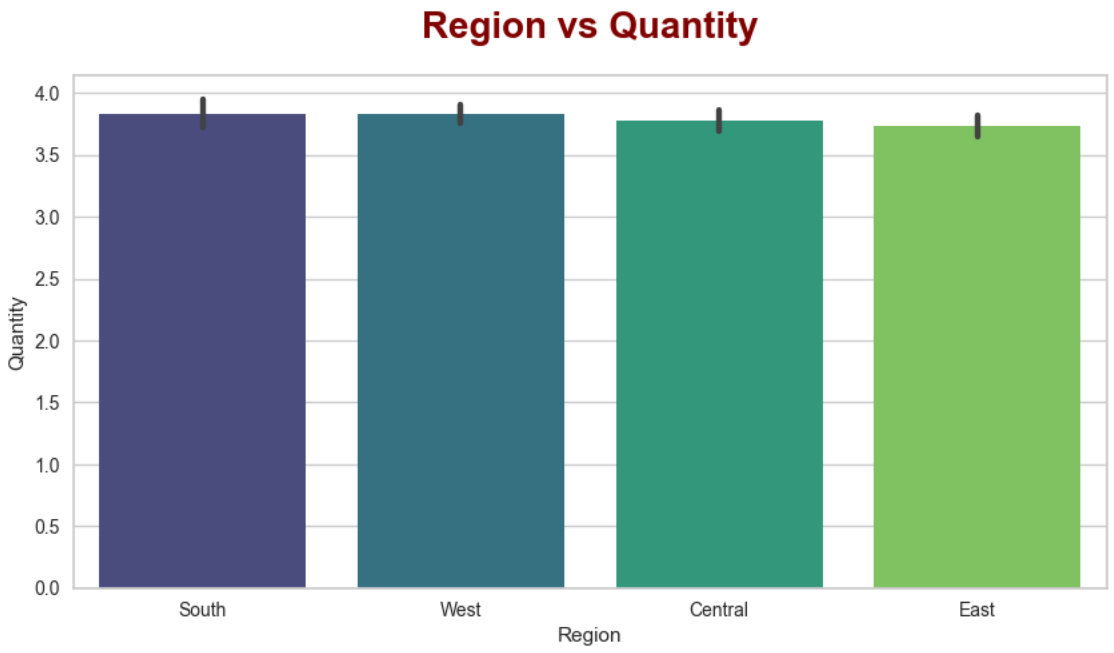
A substantial discount was offered on the orders made in the Central region whereas the least discount was given on those made in the Western region.

[29]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Region',y='Profit',data=df,palette='flare')
plt.title('Region vs Profit',pad=20,fontsize=20,fontweight='bold',color='teal')
plt.show()
```



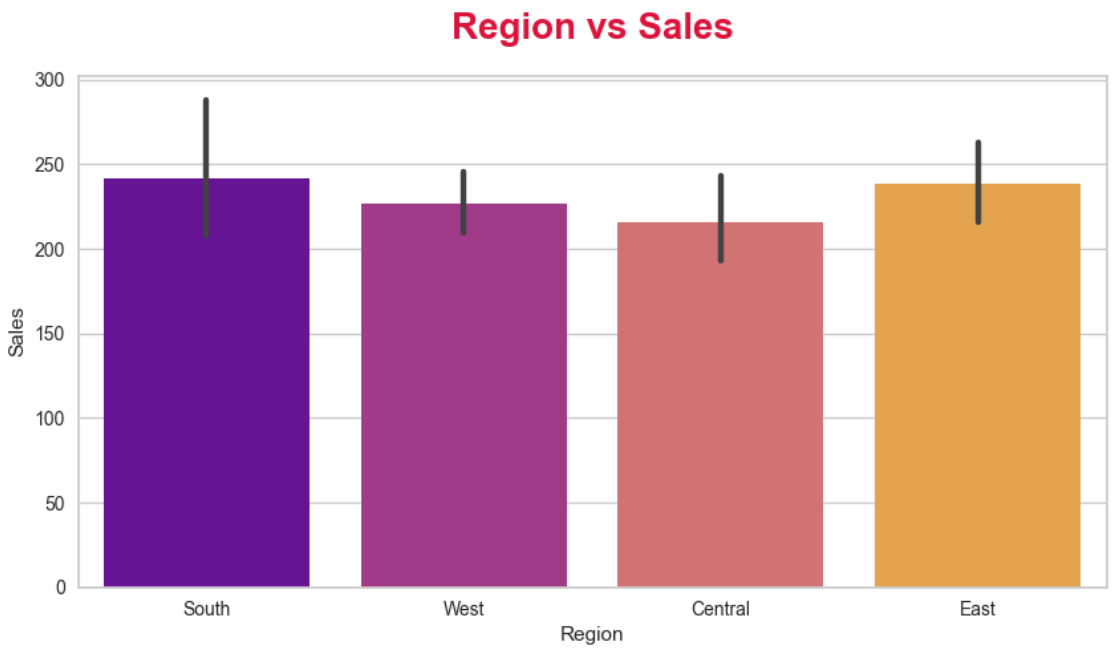
The largest profit was recorded in the Western region while the lowest profit was obtained in the Central region.

[30]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Region',y='Quantity',data=df,palette='viridis')
plt.title('Region vs␣
  ↪Quantity',pad=20,fontsize=20,fontweight='bold',color='maroon')
plt.show()
```



The highest quantities of products were purchased in the Southern and Western regions whereas the lowest quantities were ordered in the Eastern region. However, there is overall a minimal difference in the quantities of products ordered across all regions.

[31]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Region',y='Sales',data=df,palette='plasma')
plt.title('Region vs␣
  ↪Sales',pad=20,fontsize=20,fontweight='bold',color='crimson')
plt.show()
```

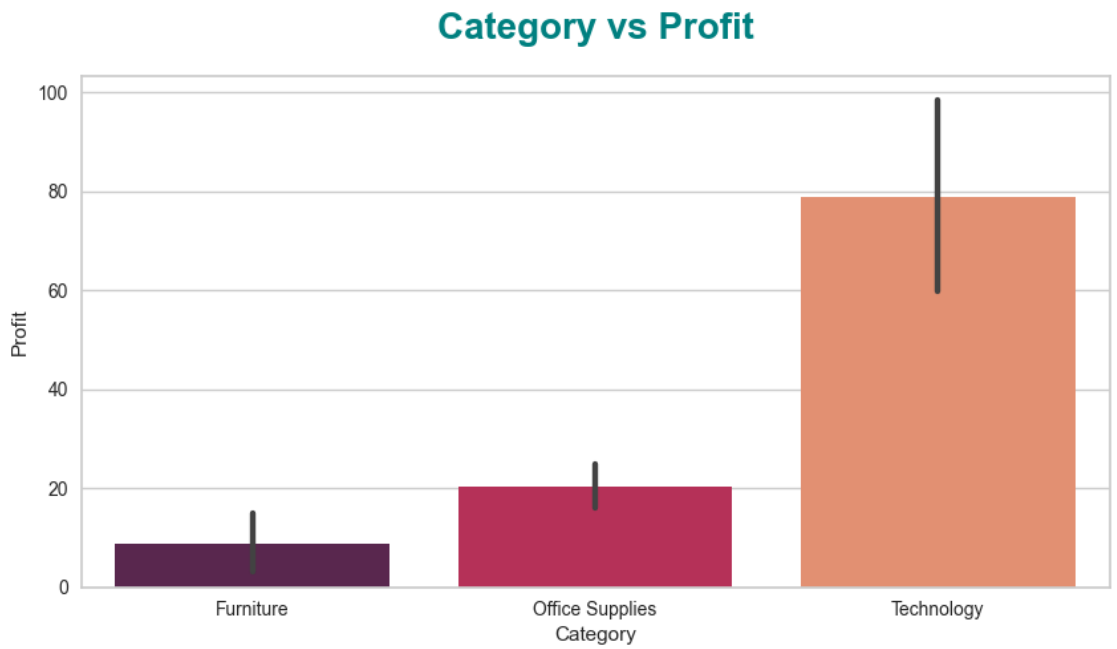


The highest sales were recorded in the South and East regions while the lowest sales were observed in the Central region.

[32]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Category',y='Discount',data=df,palette='flare')
plt.title('Category vs␣
  ↪Discount',pad=20,fontsize=20,fontweight='bold',color='orangered')
plt.show()
```

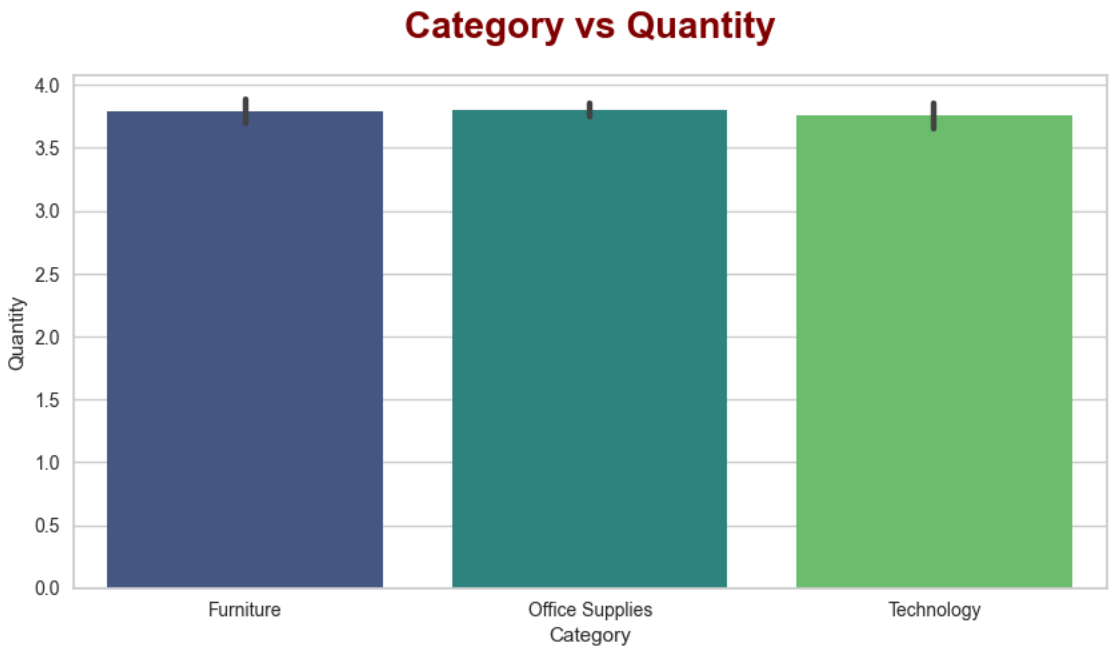
A significant discount was provided on furniture products whereas the least discount was given on technology products.

[33]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Category',y='Profit',data=df,palette='rocket')
plt.title('Category vs␣
 ↪Profit',pad=20,fontsize=20,fontweight='bold',color='teal')
plt.show()
```



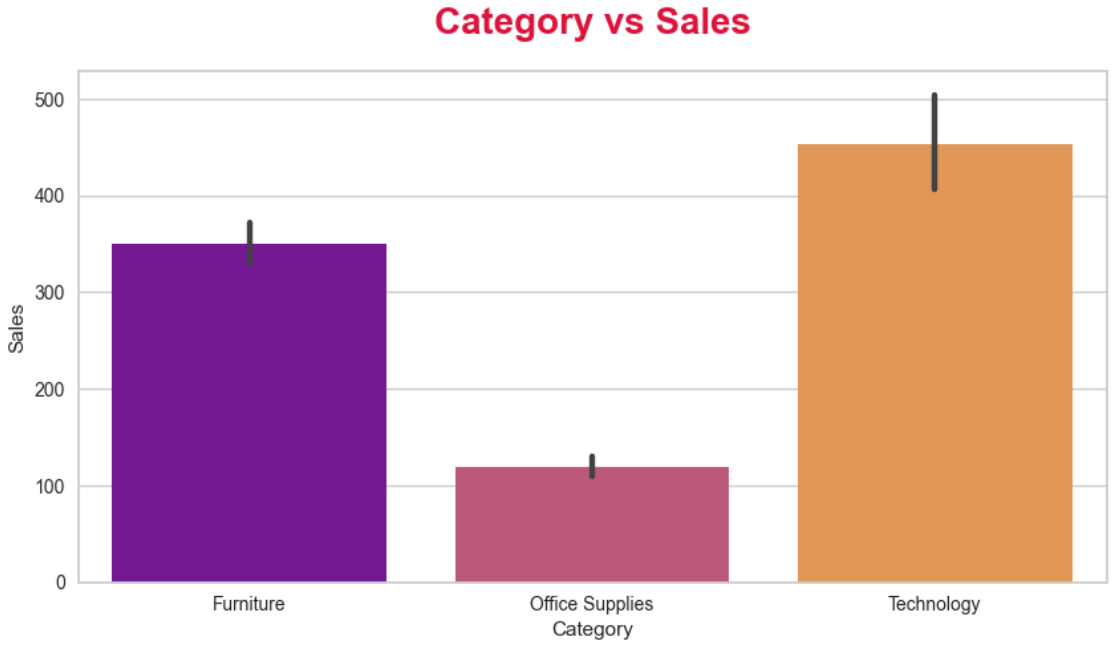
The products belonging to the Technology category are the most profitable while the products belonging to the Furniture category are the least profitable.

[34]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Category',y='Quantity',data=df,palette='viridis')
plt.title('Category vs␣
  ↪Quantity',pad=20,fontsize=20,fontweight='bold',color='maroon')
plt.show()
```



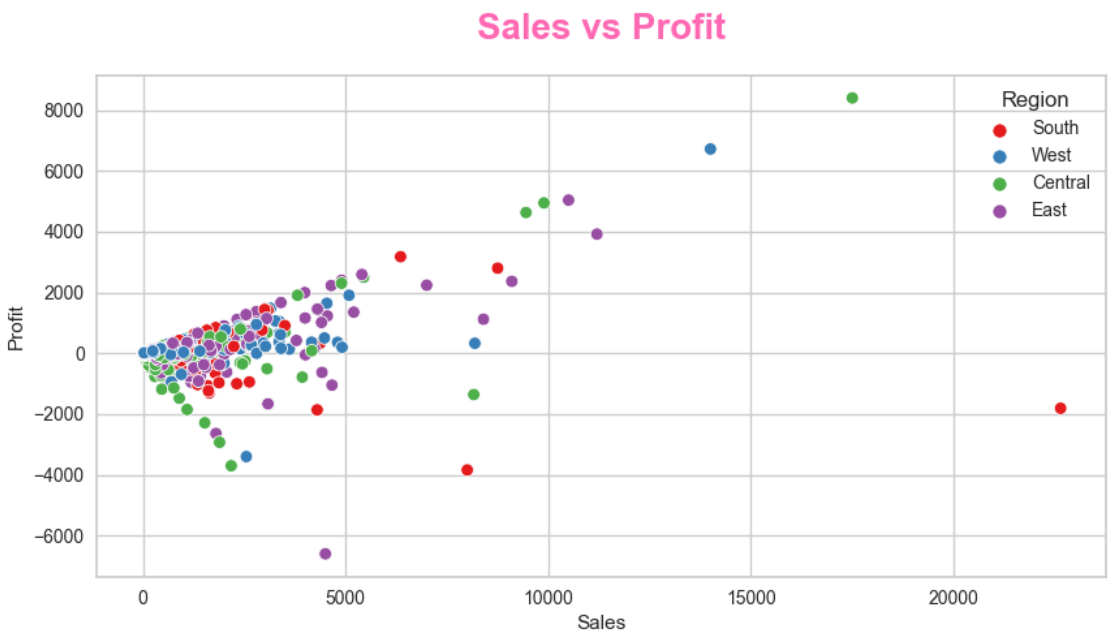
All categories have almost similar quantities of products ordered.

[35]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='Category',y='Sales',data=df,palette='plasma')
plt.title('Category vs␣
 ↪Sales',pad=20,fontsize=20,fontweight='bold',color='crimson')
plt.show()
```



The maximum sales were recorded in the Technology category while the lowest sales were obtained in the Office Supplies category.

[36]:
```
plt.figure(figsize=(10,5))
fig = sns.scatterplot(x='Sales',y='Profit',data=df,hue='Region',palette='Set1')
plt.title('Sales vs␣
 ↪Profit',pad=20,fontsize=20,fontweight='bold',color='hotpink')
plt.show()
plt.close('all')
del fig
gc.collect()
```

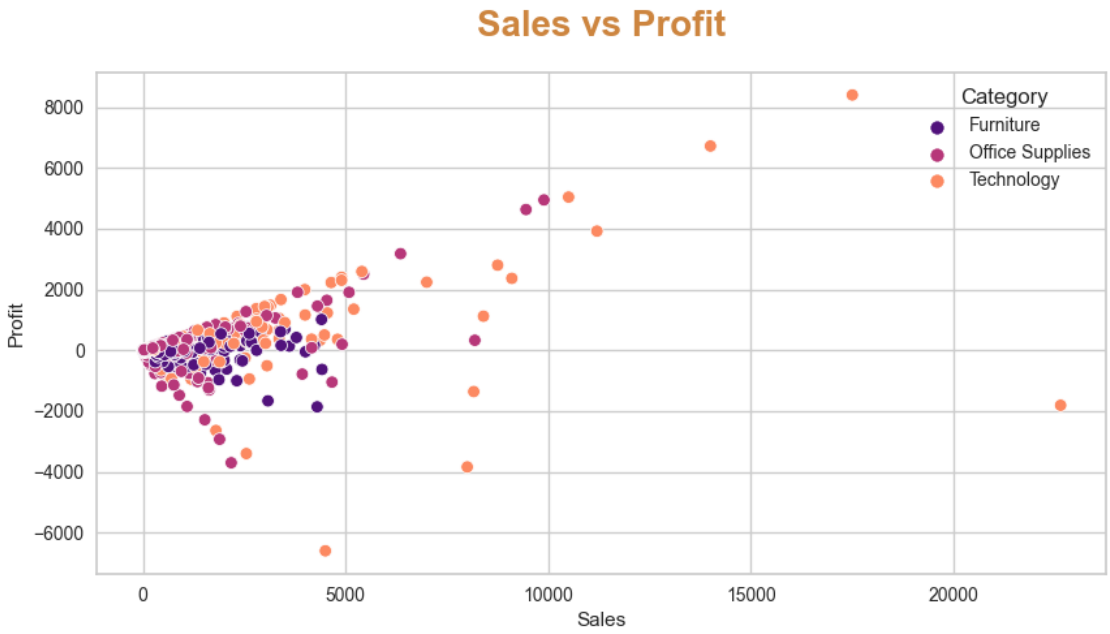


[36]: 68914

The Western region recorded the lowest sales and profit among all regions whereas the Eastern region generated a considerable profit in general across all regions.

[37]:
```
plt.figure(figsize=(10,5))
fig = sns.
 ↪scatterplot(x='Sales',y='Profit',data=df,hue='Category',palette='magma')
plt.title('Sales vs Profit',pad=20,fontsize=20,fontweight='bold',color='peru')
plt.show()
plt.close('all')
del fig
```

```
gc.collect()
```



[37]: 4385

It is clearly evident from the above plot that the highest profit was obtained in the Technology category whereas the lowest profit was obtained in the Office Supplies category.

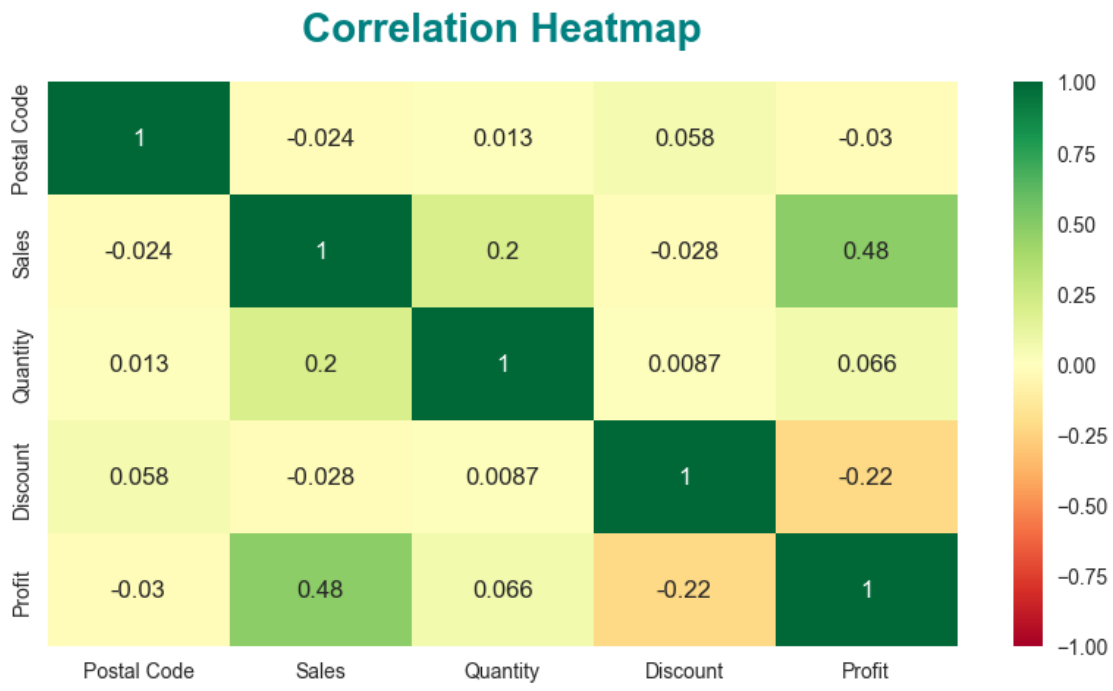
[38]:
```
plt.figure(figsize=(10,5))
fig = sns.scatterplot(x='Sales',y='Profit',data=df,hue='Ship␣
 ↪Mode',palette='rainbow')
plt.title('Sales vs␣
 ↪Profit',pad=20,fontsize=20,fontweight='bold',color='hotpink')
plt.show()
plt.close('all')
del fig
gc.collect()
```

[38]: 4189

The highest sales and profit were recorded in the orders delivered via the Standard Class shipping mode while the Second Class shipping mode was the least profitable.

0.4.3 Multivariate Analysis

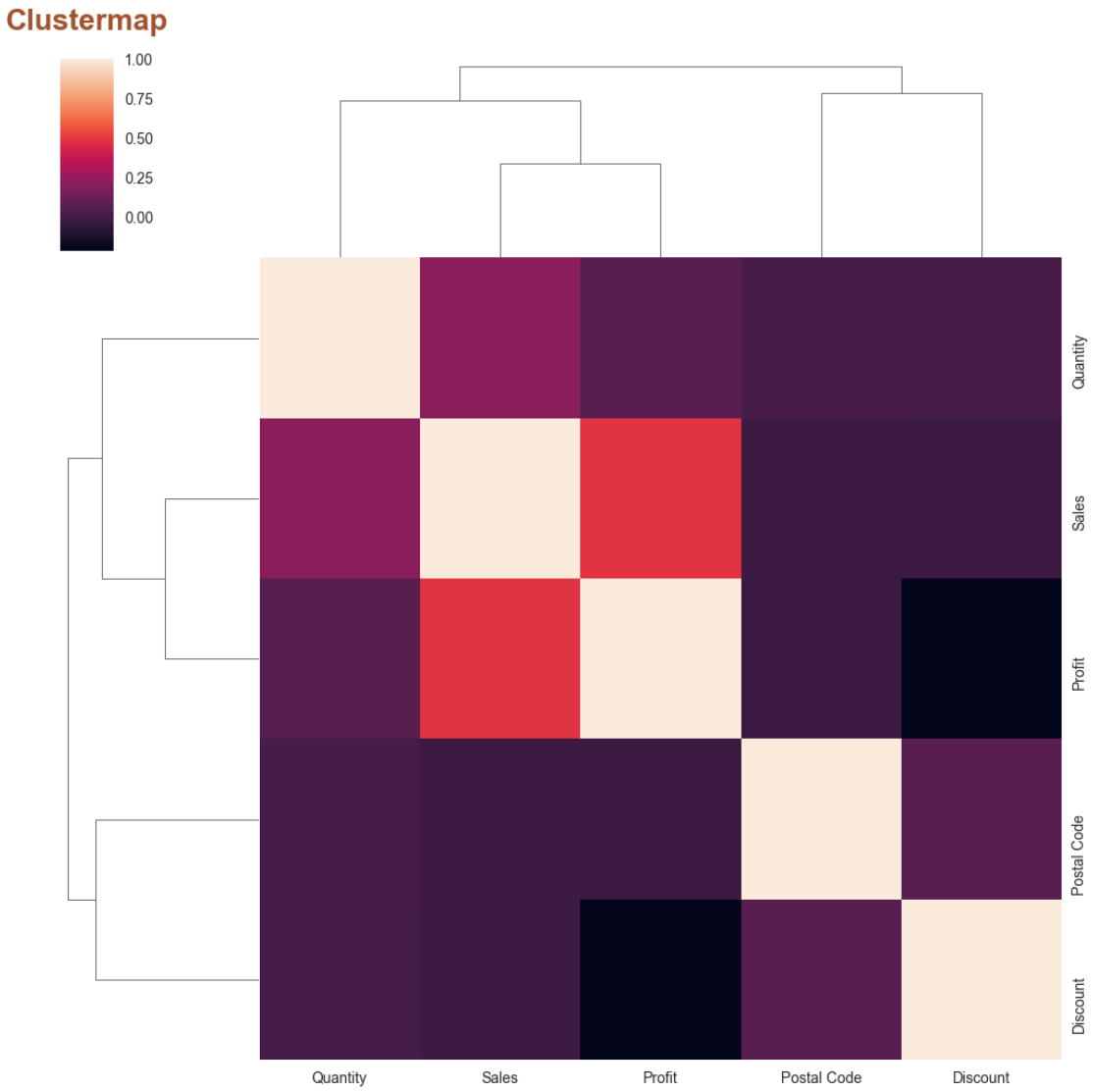
[39]:
```
plt.figure(figsize=(10,5))
fig = sns.heatmap(df.corr(),annot=True,cmap='RdYlGn',vmin=-1.0,vmax=1.0)
plt.title('Correlation␣
 ↪Heatmap',pad=20,fontsize=20,fontweight='bold',color='teal')
plt.show()
plt.close('all')
del fig
gc.collect()
```



[39]: 11336

[40]:
```
plt.figure(figsize=(10,5))
fig = sns.clustermap(df.corr())
plt.title('Clustermap',pad=20,fontsize=20,fontweight='bold',color='sienna')
plt.show()
plt.close('all')
del fig
gc.collect()
```

```
<Figure size 1000x500 with 0 Axes>
```



[40]: 9817

[41]:
```
plt.figure(figsize=(10,5))
fig = sns.pairplot(df.sample(n=5000),hue='Ship Mode',palette='RdBu')
plt.show()
plt.close('all')
del fig
gc.collect()
```

```
<Figure size 1000x500 with 0 Axes>
```

[41]: 969

0.5 Feature Engineering

0.5.1 Feature Extraction

[42]:
```
df['Profit Margin'] = df['Profit']/df['Sales']
df['Discounted Profit'] = df['Profit']-df['Profit']*df['Discount']
df['Discount Percentage'] = df['Discount']/df['Sales']*100
df['Operating Expenses'] = df['Sales'] - df['Profit']
df['Net Profit'] = df['Profit'] - df['Discount']
```

[43]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'],errors='coerce')
df['Ship Date'] = pd.to_datetime(df['Ship Date'],errors='coerce')
```

[44]:
```
df['Order Year'] = df['Order Date'].dt.year
df['Order Month'] = df['Order Date'].dt.month
```

```
df['Order Day'] = df['Order Date'].dt.day
df['Order Weekday'] = df['Order Date'].dt.dayofweek
df['Ship Year'] = df['Ship Date'].dt.year
df['Ship Month'] = df['Ship Date'].dt.month
df['Ship Day'] = df['Ship Date'].dt.day
df['Ship Weekday'] = df['Ship Date'].dt.dayofweek
df.drop(['Order Date','Ship Date'],axis=1,inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9993 entries, 0 to 9993
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Ship Mode            9993 non-null   object
 1   Segment              9993 non-null   object
 2   City                 9993 non-null   object
 3   State                9993 non-null   object
 4   Postal Code          9993 non-null   int64
 5   Region               9993 non-null   object
 6   Category             9993 non-null   object
 7   Sub-Category         9993 non-null   object
 8   Product Name         9993 non-null   object
 9   Sales                9993 non-null   float64
 10  Quantity             9993 non-null   int64
 11  Discount             9993 non-null   float64
 12  Profit               9993 non-null   float64
 13  Profit Margin        9993 non-null   float64
 14  Discounted Profit    9993 non-null   float64
 15  Discount Percentage  9993 non-null   float64
 16  Operating Expenses   9993 non-null   float64
 17  Net Profit           9993 non-null   float64
 18  Order Year           9993 non-null   int64
 19  Order Month          9993 non-null   int64
 20  Order Day            9993 non-null   int64
 21  Order Weekday        9993 non-null   int64
 22  Ship Year            9993 non-null   int64
 23  Ship Month           9993 non-null   int64
 24  Ship Day             9993 non-null   int64
 25  Ship Weekday         9993 non-null   int64
dtypes: float64(8), int64(10), object(8)
memory usage: 2.3+ MB
```

0.5.2 Categorical Encoding

[45]:
```
df['Ship Mode'].unique()
```

[45]:
```
array(['Second Class', 'Standard Class', 'First Class', 'Same Day'],
      dtype=object)
```

[46]:
```
df['Segment'].unique()
```

[46]:
```
array(['Consumer', 'Corporate', 'Home Office'], dtype=object)
```

[47]:
```
df['Category'].unique()
```

[47]:
```
array(['Furniture', 'Office Supplies', 'Technology'], dtype=object)
```

[48]:
```
df['Sub-Category'].unique()
```

[48]:
```
array(['Bookcases', 'Chairs', 'Labels', 'Tables', 'Storage',
       'Furnishings', 'Art', 'Phones', 'Binders', 'Appliances', 'Paper',
       'Accessories', 'Envelopes', 'Fasteners', 'Supplies', 'Machines',
       'Copiers'], dtype=object)
```

[49]:
```
df['Product Name'].nunique()
```

[49]:
```
1841
```

[50]:
```
df['City'].nunique()
```

[50]:
```
531
```

[51]:
```
df['State'].nunique()
```

[51]:
```
49
```

[52]:
```
df['Region'].unique()
```

[52]:
```
array(['South', 'West', 'Central', 'East'], dtype=object)
```

[53]:
```
def onehotencode(data: pd.DataFrame,col: str) -> pd.DataFrame:
    encoder = OneHotEncoder(drop='first',sparse_output=False,max_categories=10)
    encoded_data = encoder.fit_transform(data[[col]])
    encoded_data = pd.DataFrame(encoded_data,columns=encoder.
 ↪get_feature_names_out())
    return encoded_data
```

[54]:
```
for col in ['Ship␣
 ↪Mode','Segment','State','City','Region','Category','Sub-Category','Product␣
 ↪Name']:
    encoded_data = onehotencode(df,col)
    df = pd.concat([df,encoded_data],axis=1)
    df.drop(col,axis=1,inplace=True)
```

[55]:
```
transformer = ColumnTransformer(transformers=[
    ('log_transform',FunctionTransformer(np.
 ↪log1p),['Quantity','Profit','Discounted Profit','Net Profit']),
    ('sqrt_transform',FunctionTransformer(np.sqrt),['Discount']),
    ('power_transform',PowerTransformer(),['Profit Margin','Discount␣
 ↪Percentage','Operating Expenses'])
],remainder='passthrough')
transformer
```

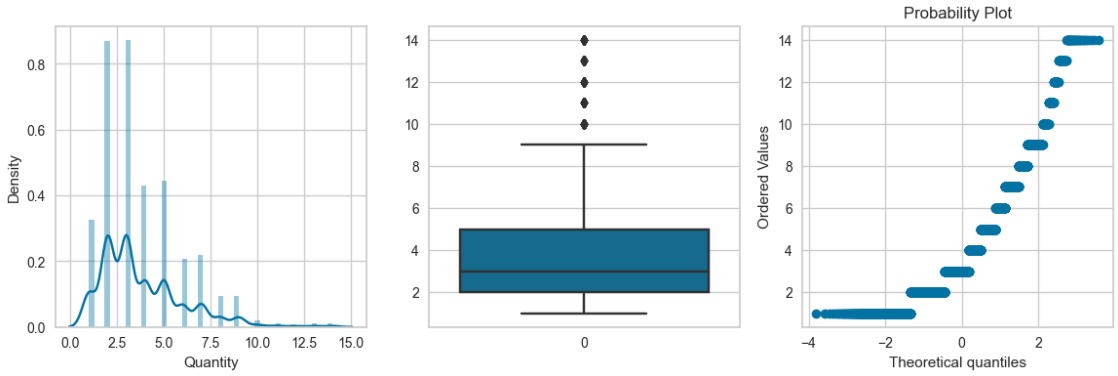
[55]:
```
ColumnTransformer(remainder='passthrough',
                  transformers=[('log_transform',
                                 FunctionTransformer(func=<ufunc 'log1p'>),
                                 ['Quantity', 'Profit', 'Discounted Profit',
                                  'Net Profit']),
                                ('sqrt_transform',
                                 FunctionTransformer(func=<ufunc 'sqrt'>),
                                 ['Discount']),
                                ('power_transform', PowerTransformer(),
                                 ['Profit Margin', 'Discount Percentage',
                                  'Operating Expenses'])])
```

0.5.3 Feature Transformation

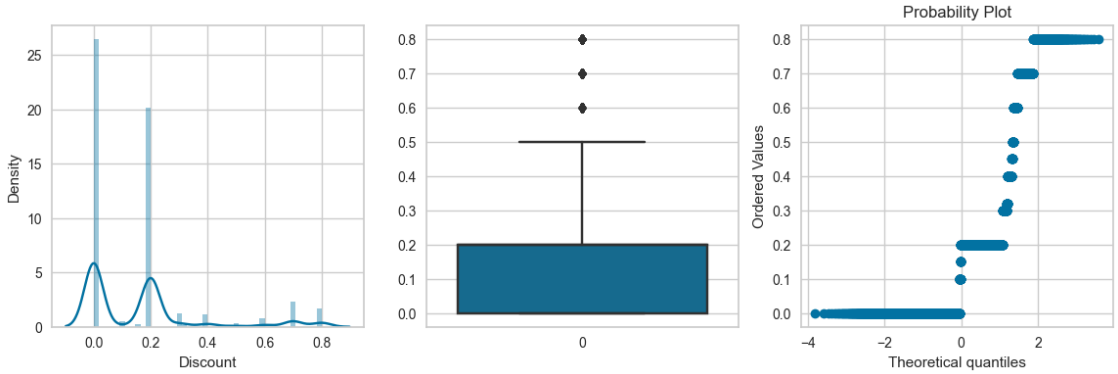
[56]:
```
skewed_cols = ['Quantity',
 'Discount',
 'Profit',
 'Profit Margin',
 'Discounted Profit',
 'Discount Percentage',
 'Operating Expenses',
 'Net Profit',
 'Order Day',
 'Order Weekday',
 'Order Month',
 'Ship Day']
```

[57]:
```
for col in skewed_cols:
    print(f"Skewness of {col}:",df[col].skew())
    print(f"Kurtosis of {col}:",df[col].kurt())
    plt.subplots(nrows=1,ncols=2,figsize=(14,4))
    plt.subplot(1,3,1)
    sns.distplot(df[col])
    plt.subplot(1,3,2)
    sns.boxplot(df[col])
    plt.subplot(1,3,3)
    probplot(df[col],plot=plt,dist='norm',rvalue=True)
    plt.show()
```

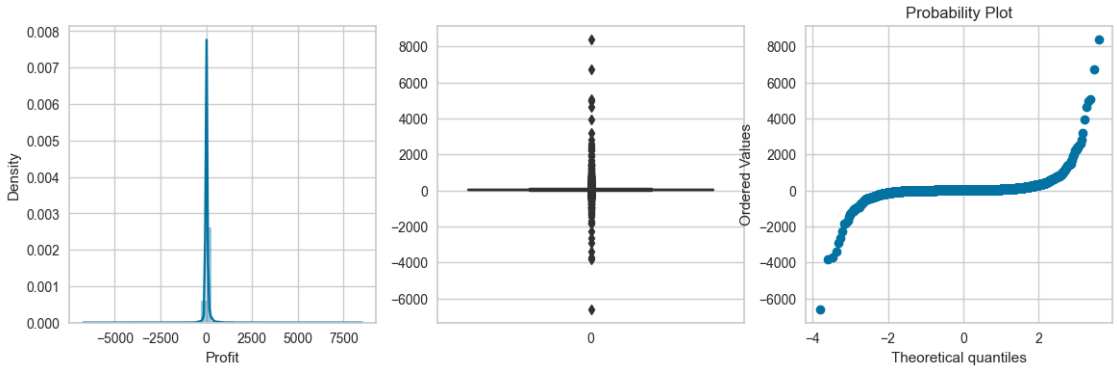
```
Skewness of Quantity: 1.2784155399852233
Kurtosis of Quantity: 1.9915827160047757

posx and posy should be finite values
posx and posy should be finite values
```




```
Skewness of Discount: 1.6845084876700462
Kurtosis of Discount: 2.409976524801519

posx and posy should be finite values
posx and posy should be finite values
```

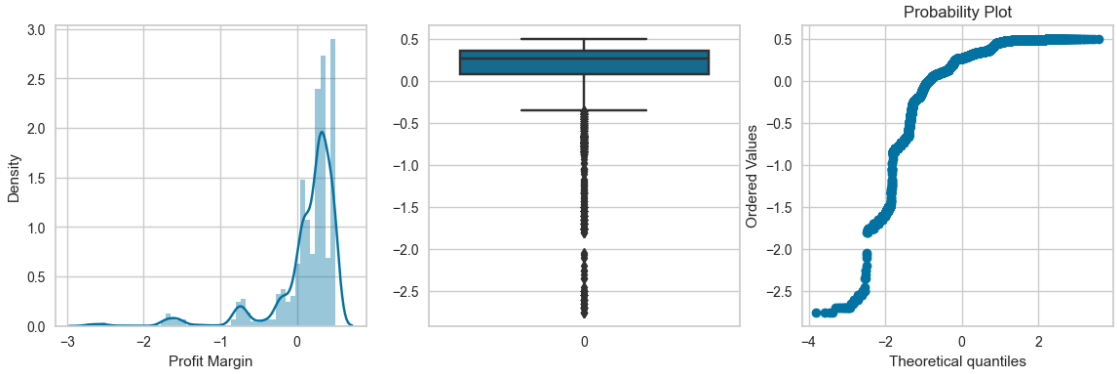



```
Skewness of Profit: 7.561035996041436
Kurtosis of Profit: 397.15038474389223

posx and posy should be finite values
posx and posy should be finite values
```



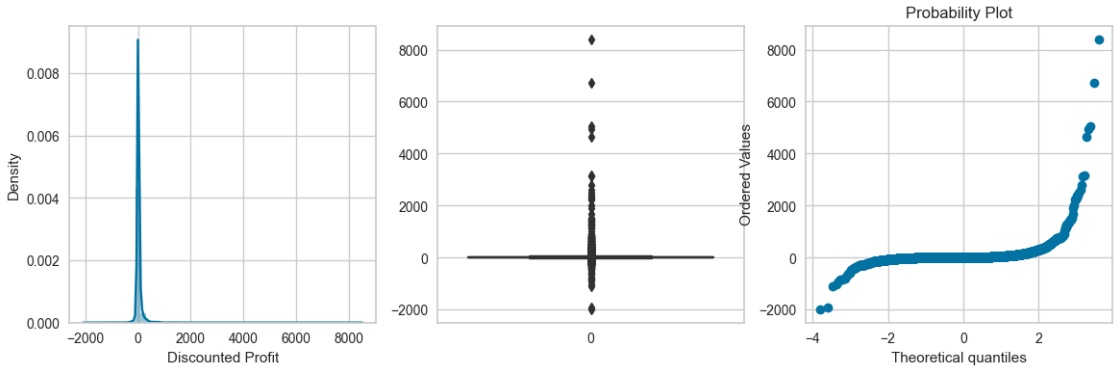

```
Skewness of Profit Margin: -2.894835315865789
Kurtosis of Profit Margin: 10.172752622828092
```

```
posx and posy should be finite values
posx and posy should be finite values
```

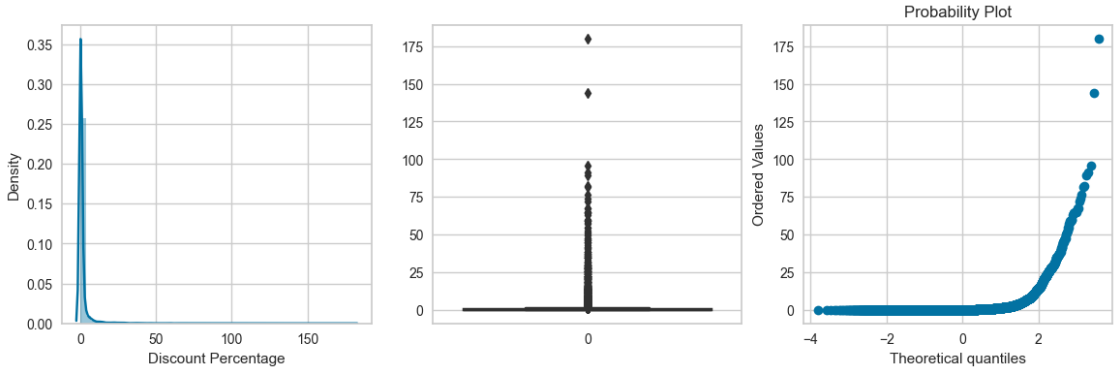



```
Skewness of Discounted Profit: 19.35659799419806
Kurtosis of Discounted Profit: 617.6369152241313
```

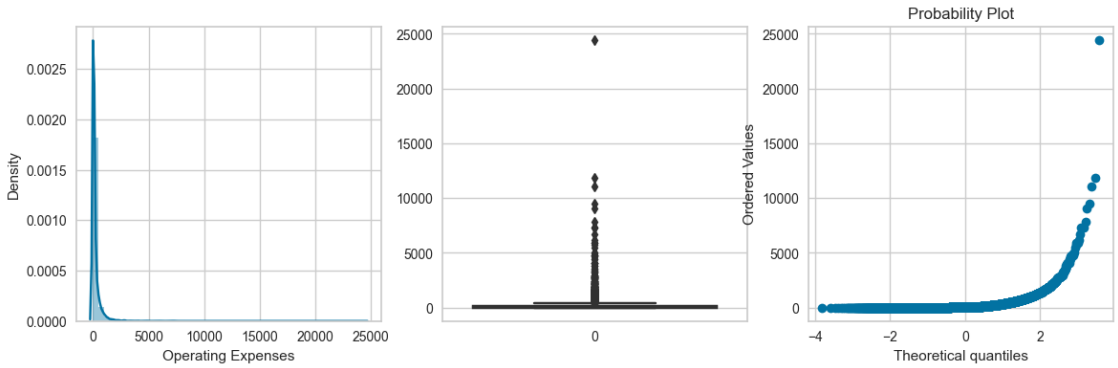
```
posx and posy should be finite values
posx and posy should be finite values
```




```
Skewness of Discount Percentage: 10.828151381652273
Kurtosis of Discount Percentage: 188.85920639655157

posx and posy should be finite values
posx and posy should be finite values
```

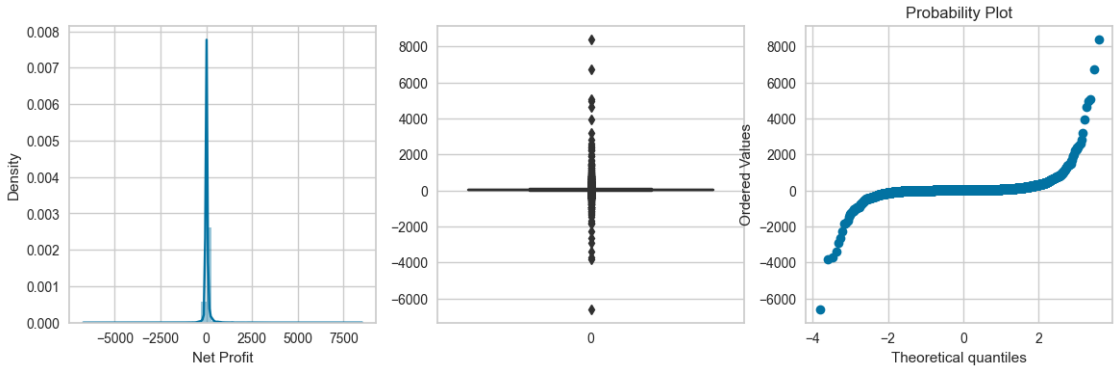



```
Skewness of Operating Expenses: 14.752446377055431
Kurtosis of Operating Expenses: 454.5499992208234

posx and posy should be finite values
posx and posy should be finite values
```



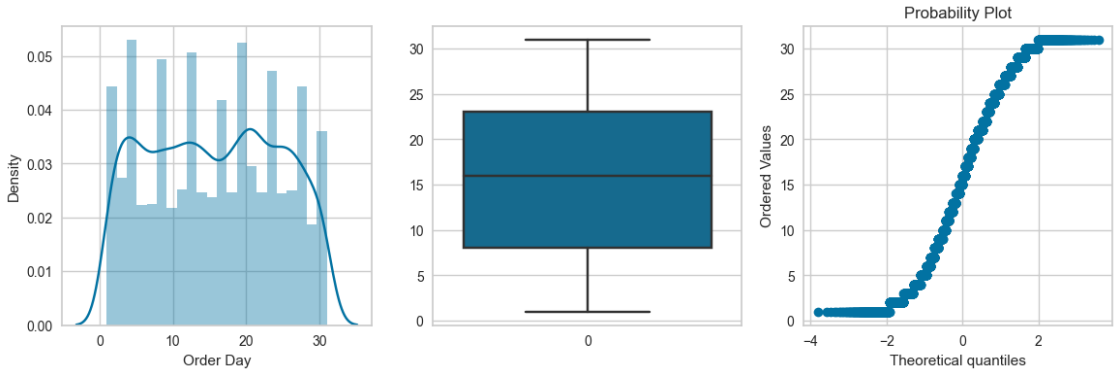

```
Skewness of Net Profit: 7.555768996026929
Kurtosis of Net Profit: 396.9091881919584
```

```
posx and posy should be finite values
posx and posy should be finite values
```



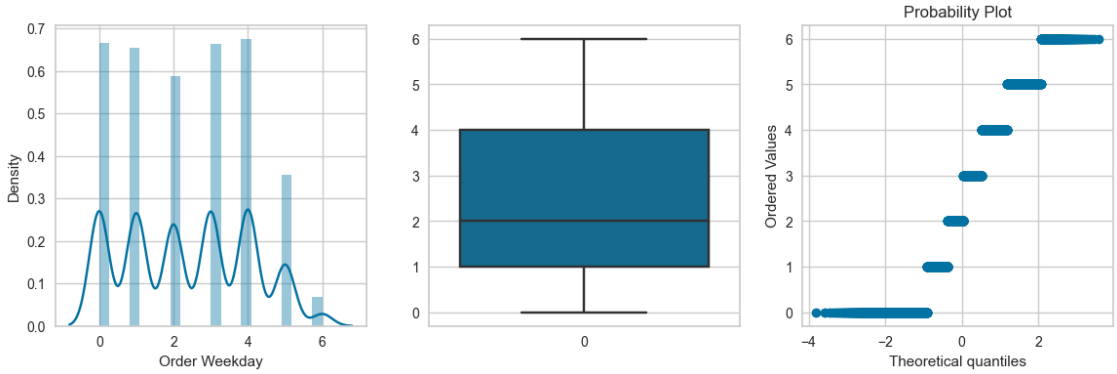

```
Skewness of Order Day: 0.013550065823461663
Kurtosis of Order Day: -1.1861898468255059
```

```
posx and posy should be finite values
posx and posy should be finite values
```



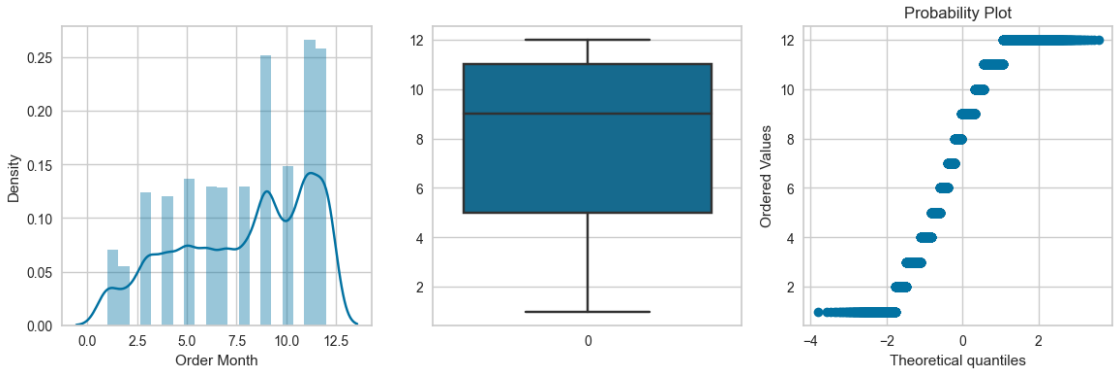

```
Skewness of Order Weekday: 0.12340952691336028
Kurtosis of Order Weekday: -1.0869179921056922
```

```
posx and posy should be finite values
posx and posy should be finite values
```



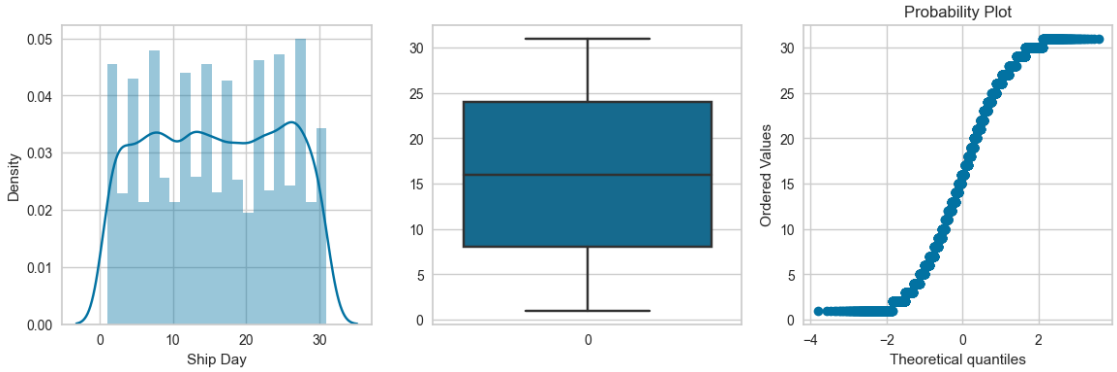

```
Skewness of Order Month: -0.4329511345177871
Kurtosis of Order Month: -0.9846637321437535
```

```
posx and posy should be finite values
posx and posy should be finite values
```




```
Skewness of Ship Day: -0.012355797229445813
Kurtosis of Ship Day: -1.2070053450291567
```

```
posx and posy should be finite values
posx and posy should be finite values
```

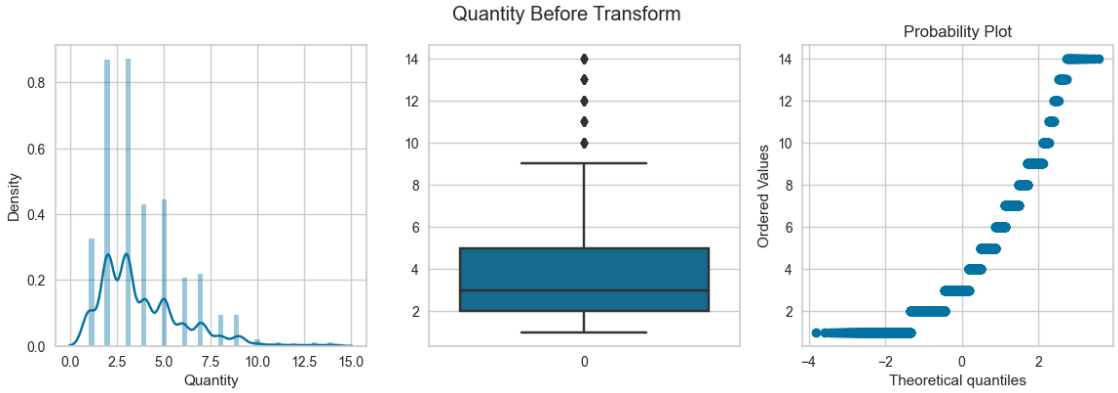


[61]:
```python
def apply_transform(data,col,transformer):
    plt.figure(figsize=(14,4))
    plt.subplot(131)
    sns.distplot(data[col])
    plt.subplot(132)
    sns.boxplot(data[col])
    plt.subplot(133)
    probplot(data[col],rvalue=True,dist='norm',plot=plt)
    plt.suptitle(f"{col} Before Transform")
    plt.show()
    col_tf = transformer.fit_transform(data[[col]])
    col_tf = np.array(col_tf).reshape(col_tf.shape[0])
    plt.figure(figsize=(14,4))
    plt.subplot(131)
```

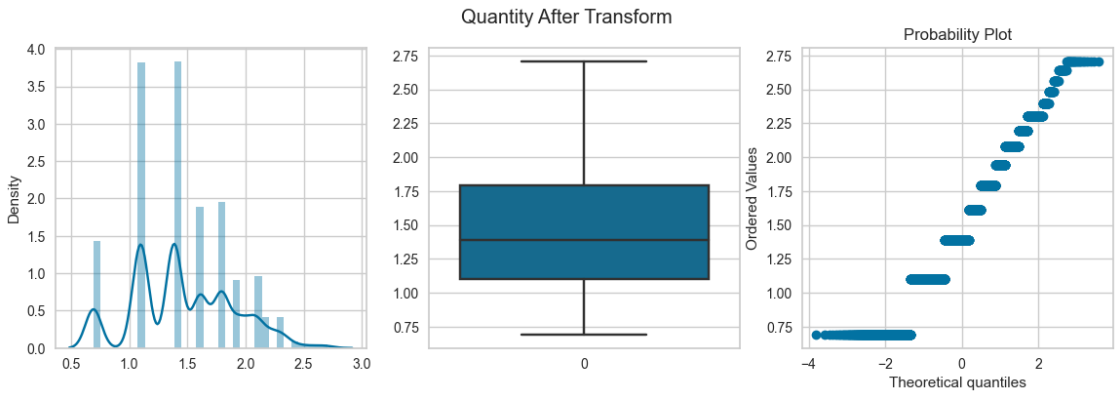
```
    sns.distplot(col_tf)
    plt.subplot(132)
    sns.boxplot(col_tf)
    plt.subplot(133)
    probplot(col_tf,rvalue=True,dist='norm',plot=plt)
    plt.suptitle(f"{col} After Transform")
    plt.show()
    gc.collect();
```

[62]:
```
for col in skewed_cols:
    apply_transform(df,col,FunctionTransformer(np.log1p))
```

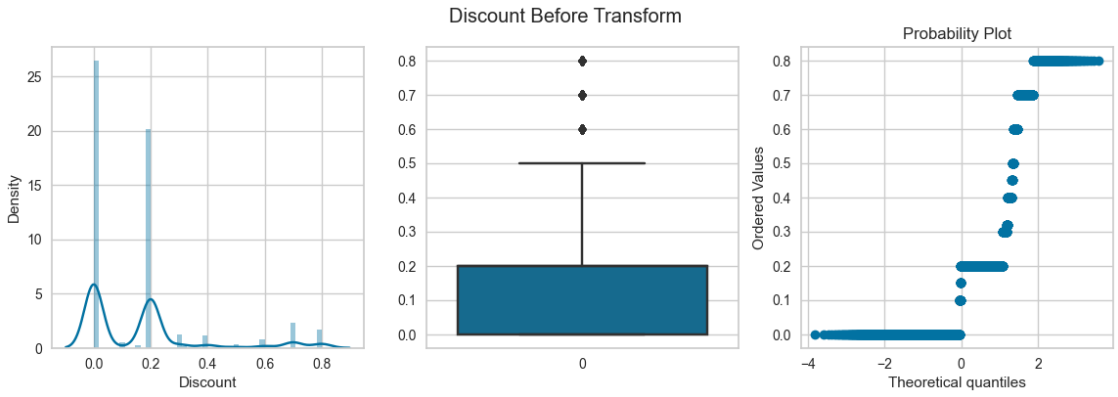
```
posx and posy should be finite values
posx and posy should be finite values
```



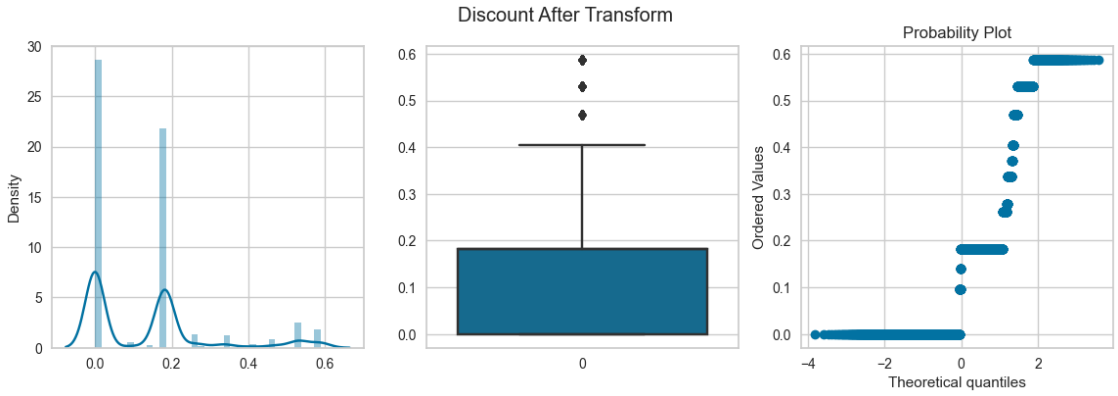

```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

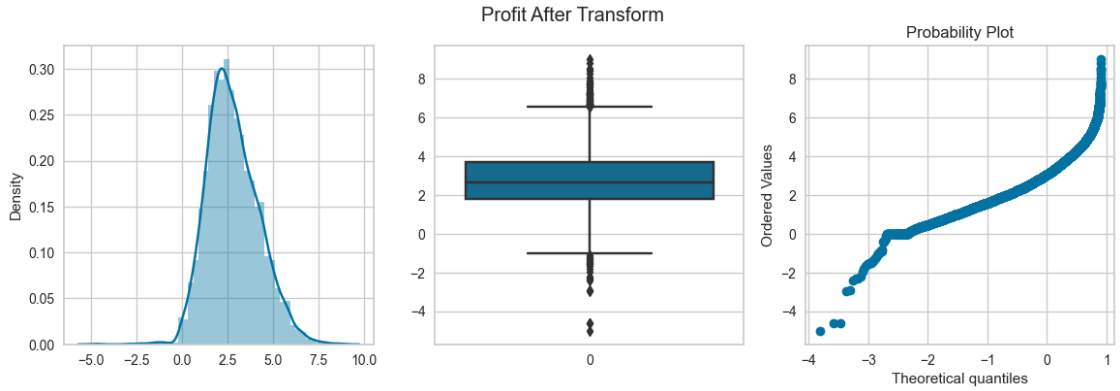



```
posx and posy should be finite values
posx and posy should be finite values
```

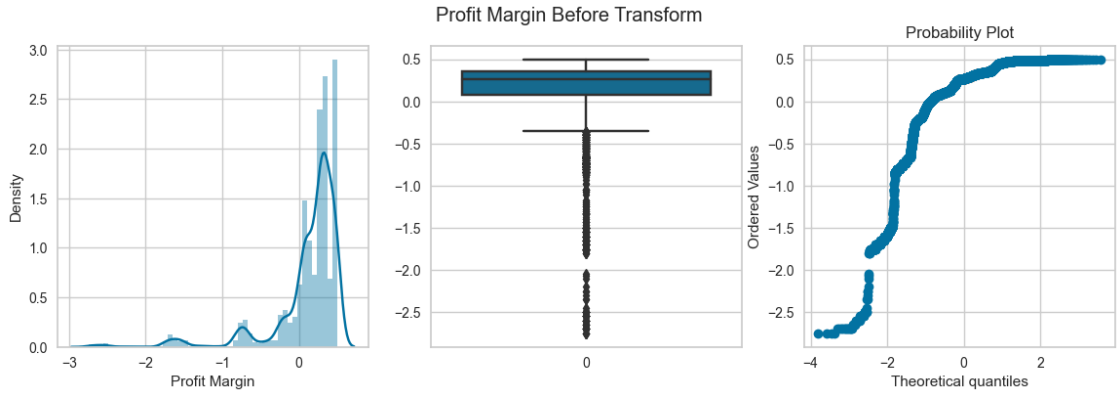



```
posx and posy should be finite values
posx and posy should be finite values
```

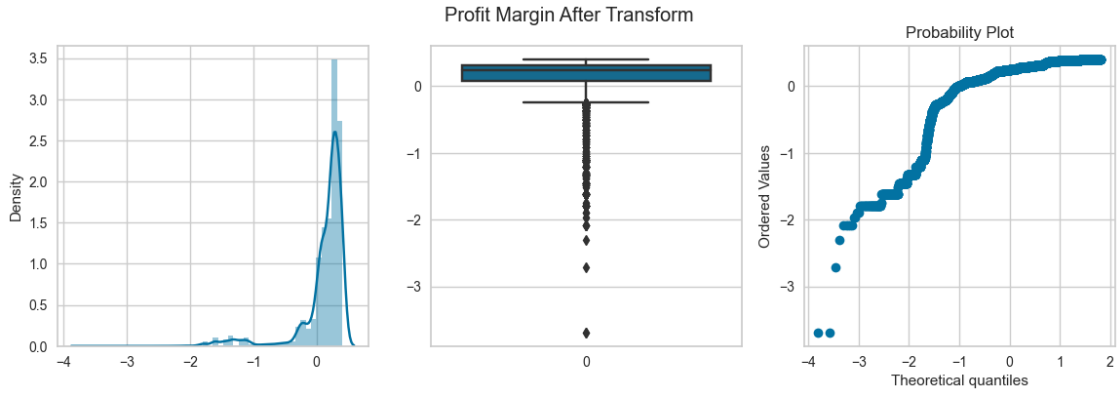
```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

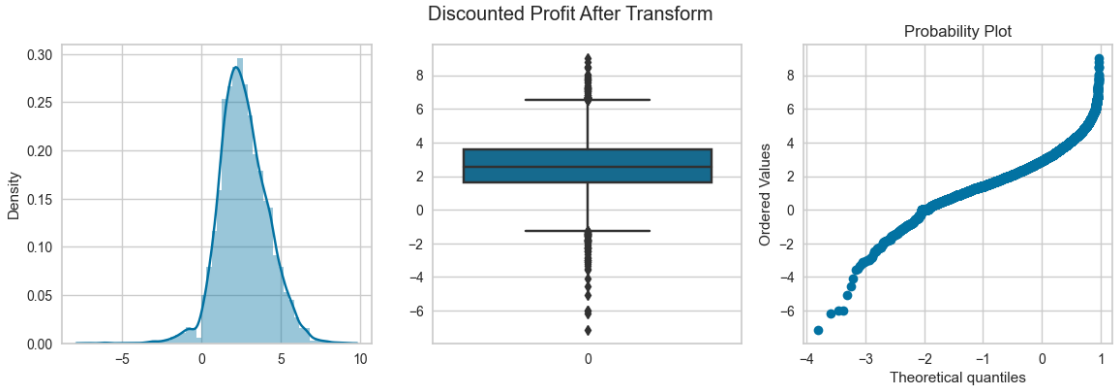



```
posx and posy should be finite values
posx and posy should be finite values
```

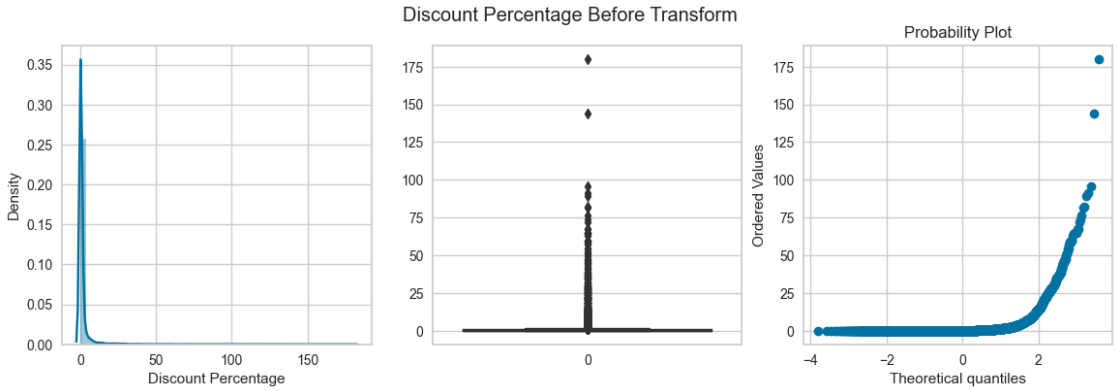


```
posx and posy should be finite values
posx and posy should be finite values
```

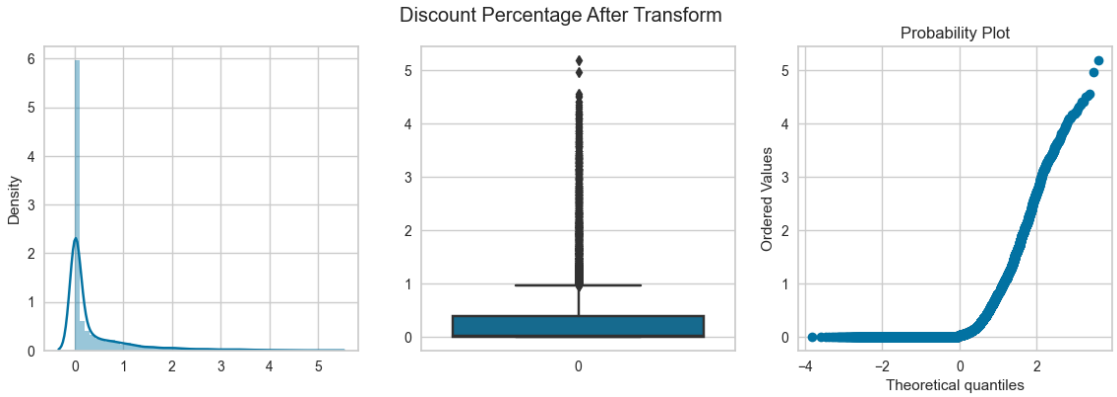

```
posx and posy should be finite values
posx and posy should be finite values
```



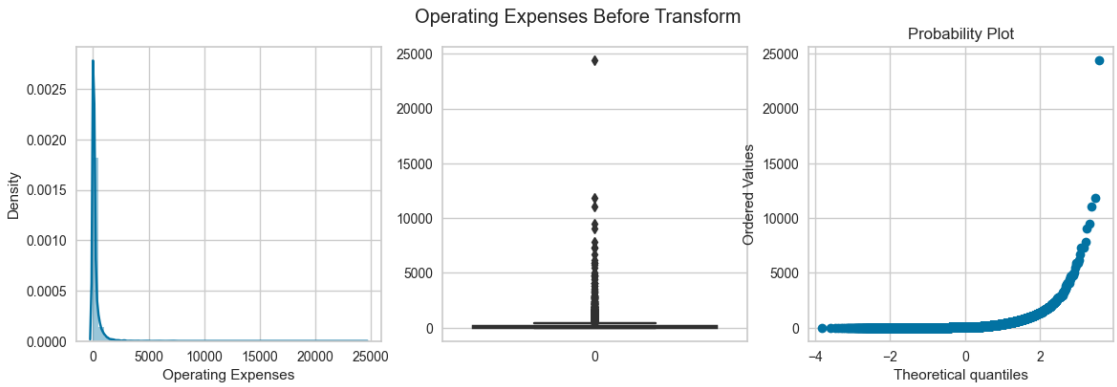

```
posx and posy should be finite values
posx and posy should be finite values
```



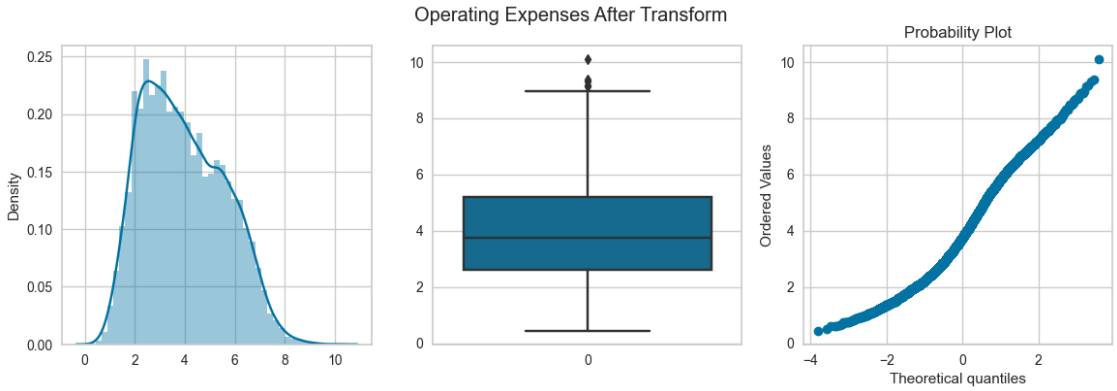

```
posx and posy should be finite values
posx and posy should be finite values
```



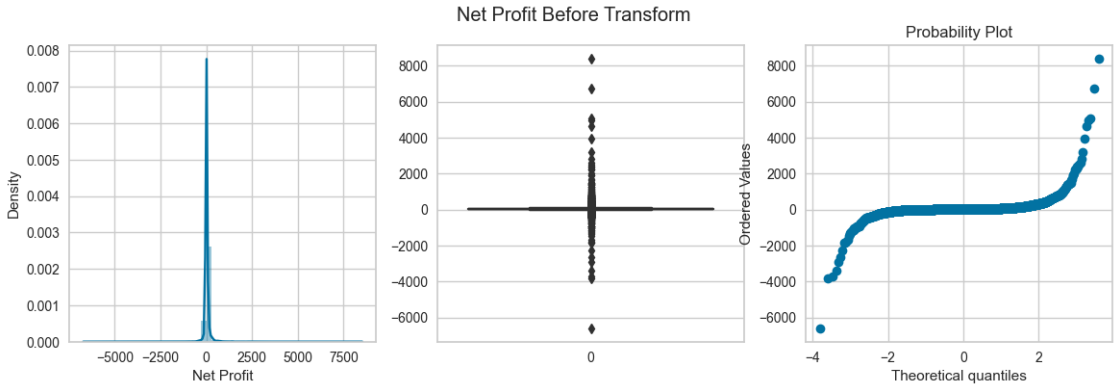

```
posx and posy should be finite values
posx and posy should be finite values
```



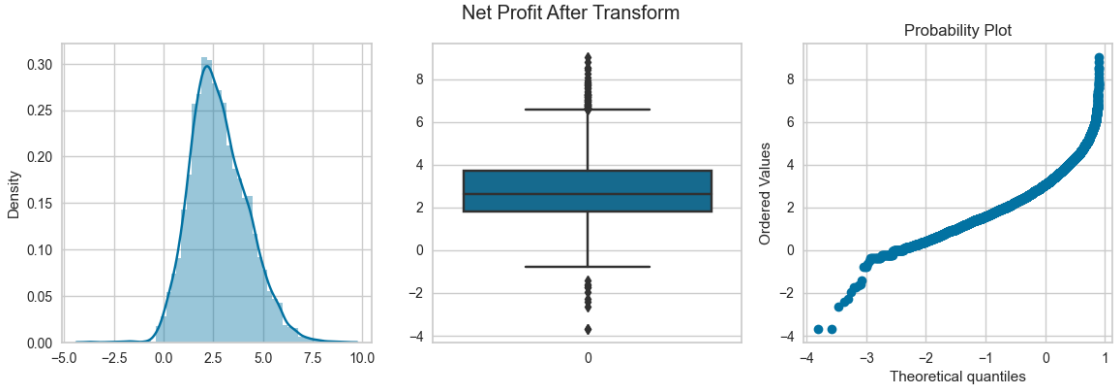
```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

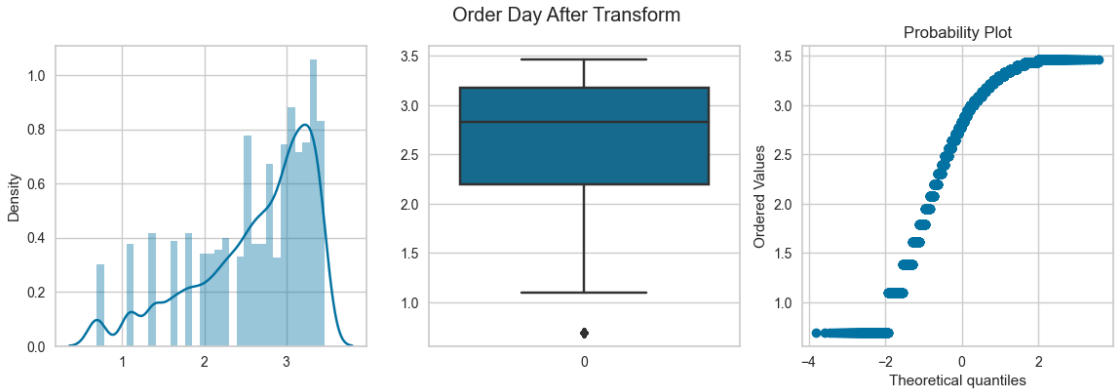



```
posx and posy should be finite values
posx and posy should be finite values
```



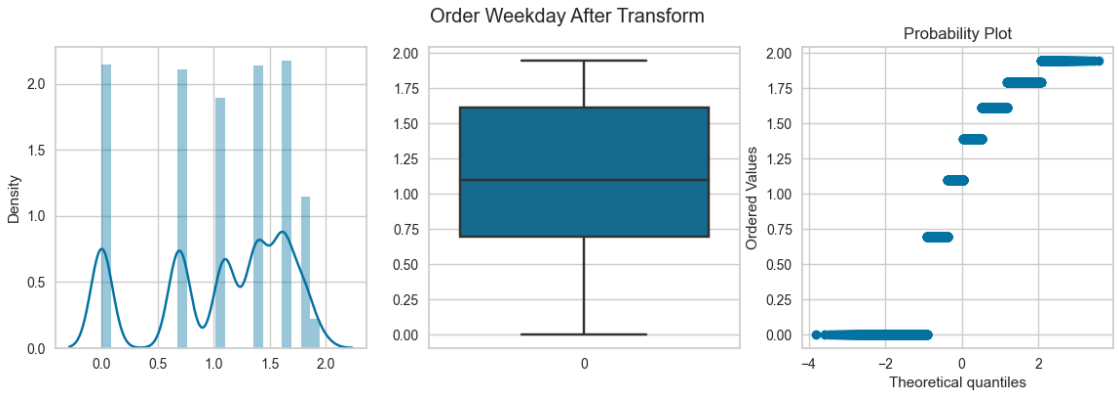
```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```



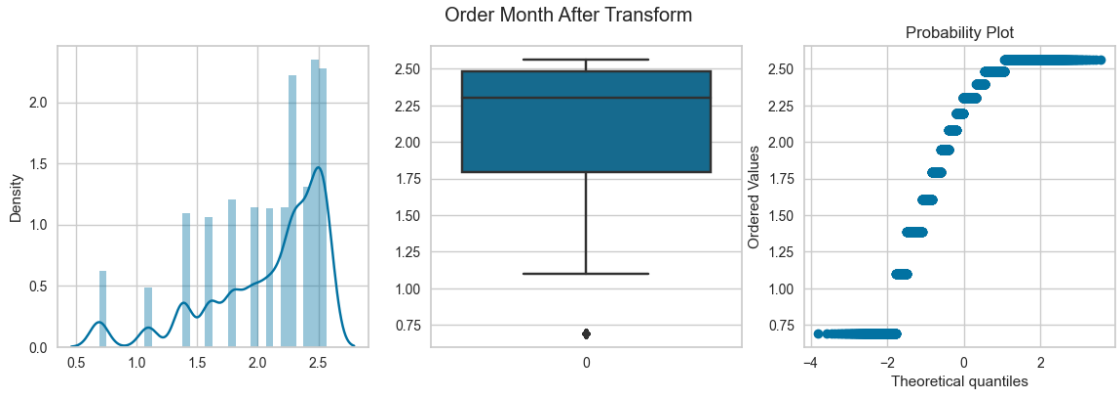

```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```



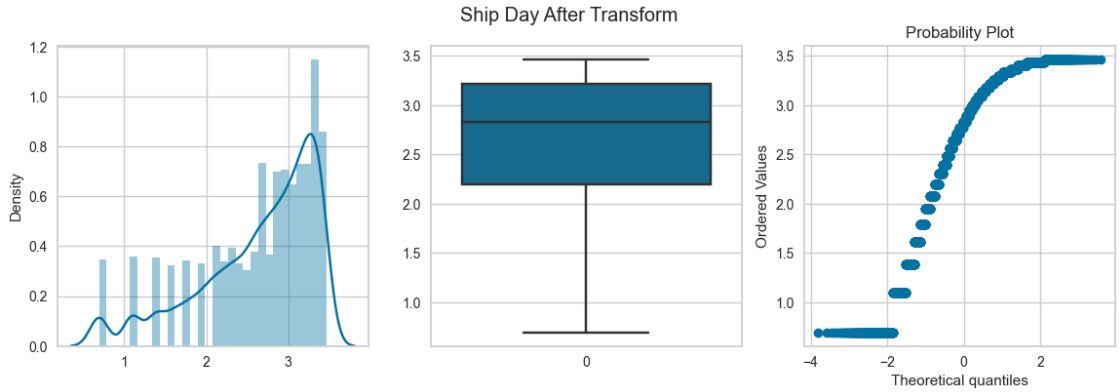

```
posx and posy should be finite values
posx and posy should be finite values
```

```
posx and posy should be finite values
posx and posy should be finite values
```



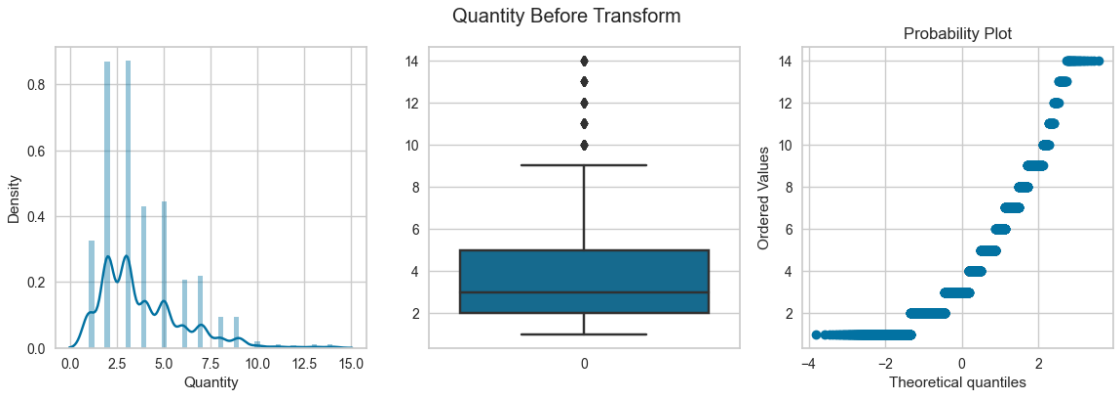

```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```

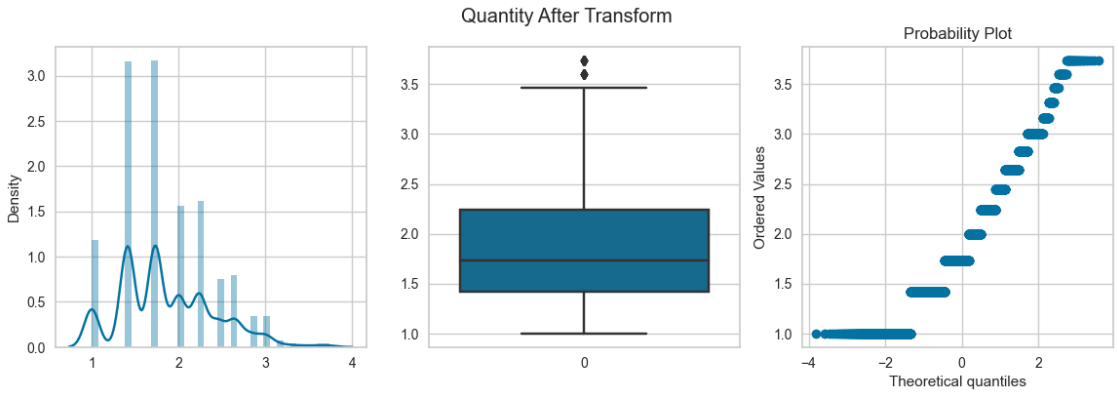


[63]:
```
for col in skewed_cols:
    apply_transform(df,col,FunctionTransformer(np.sqrt))
```

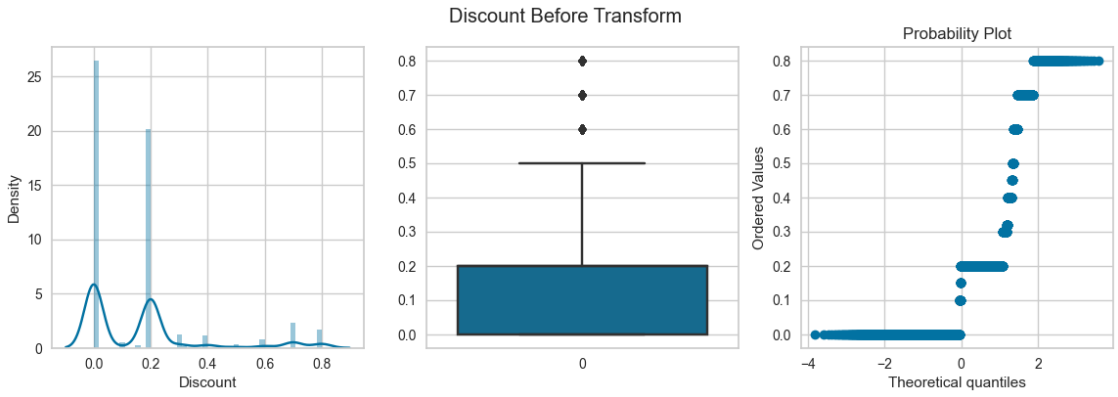
```
posx and posy should be finite values
posx and posy should be finite values
```



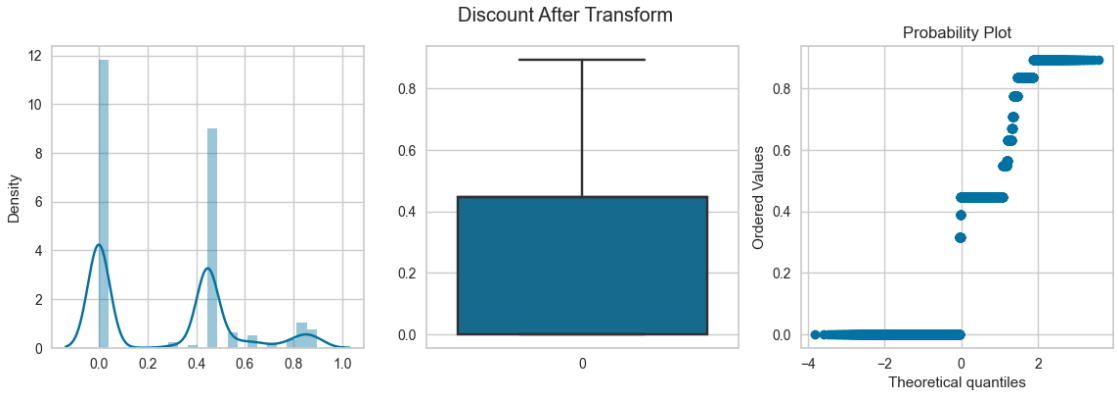

```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

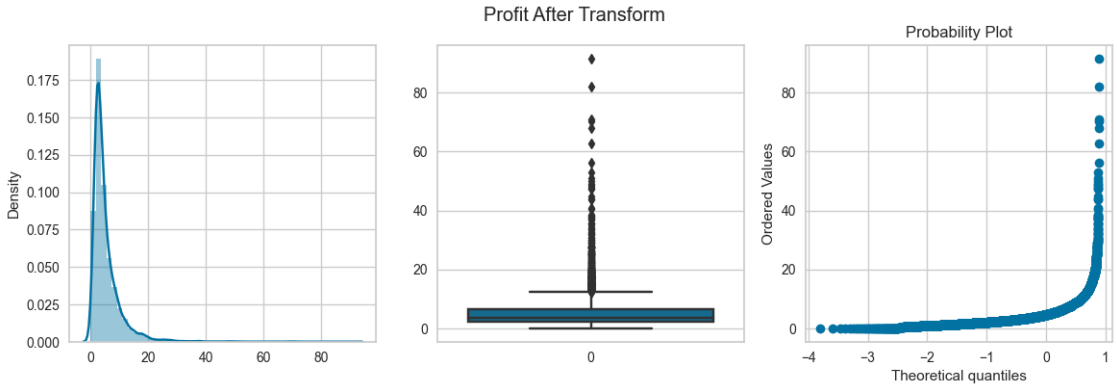



```
posx and posy should be finite values
posx and posy should be finite values
```

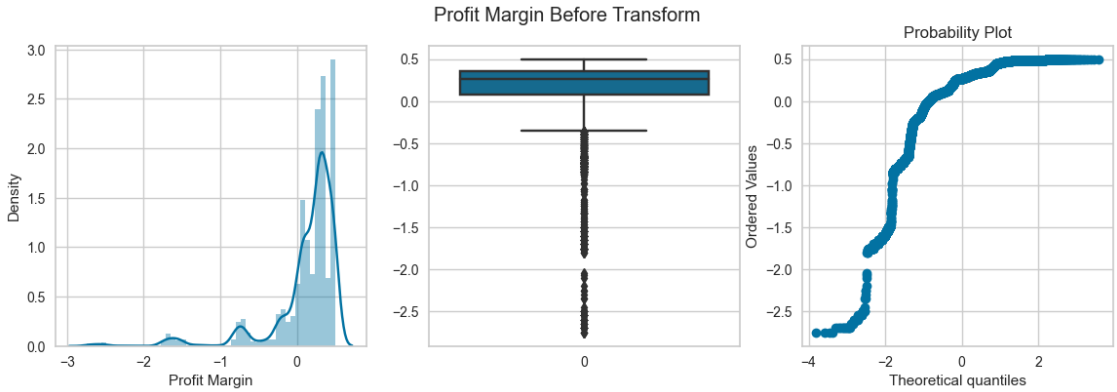



```
posx and posy should be finite values
posx and posy should be finite values
```

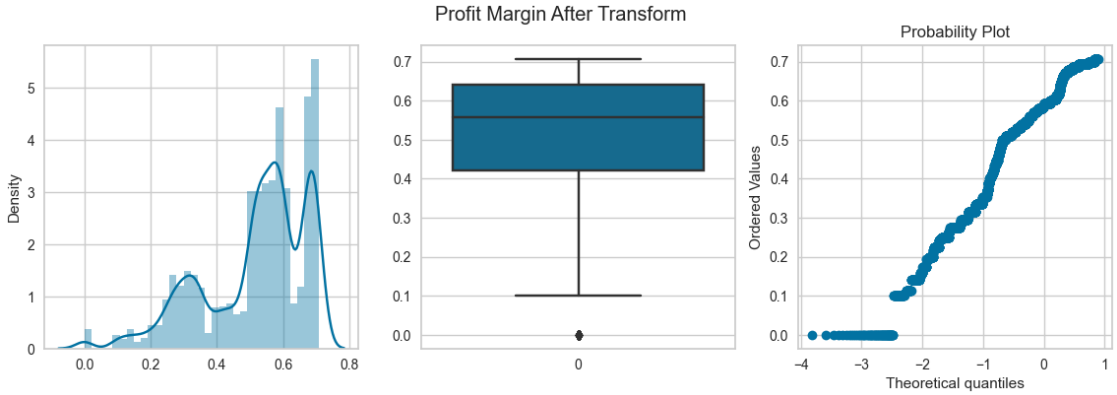
```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

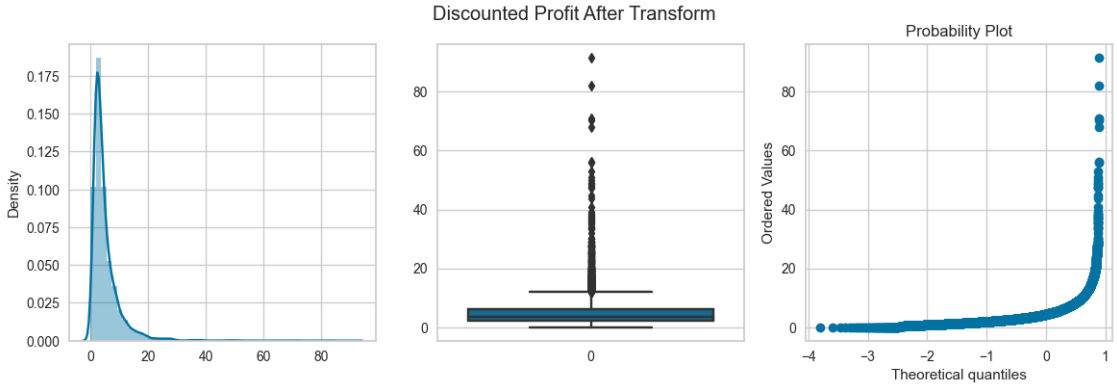



```
posx and posy should be finite values
posx and posy should be finite values
```

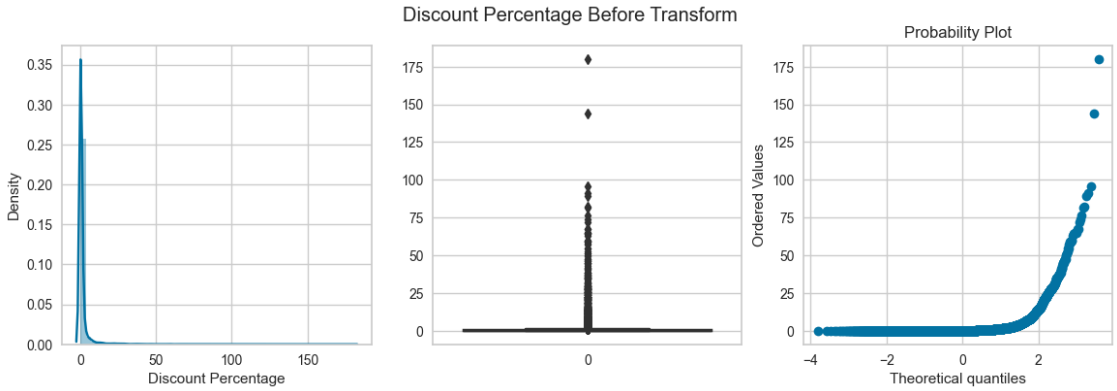


```
posx and posy should be finite values
posx and posy should be finite values
```

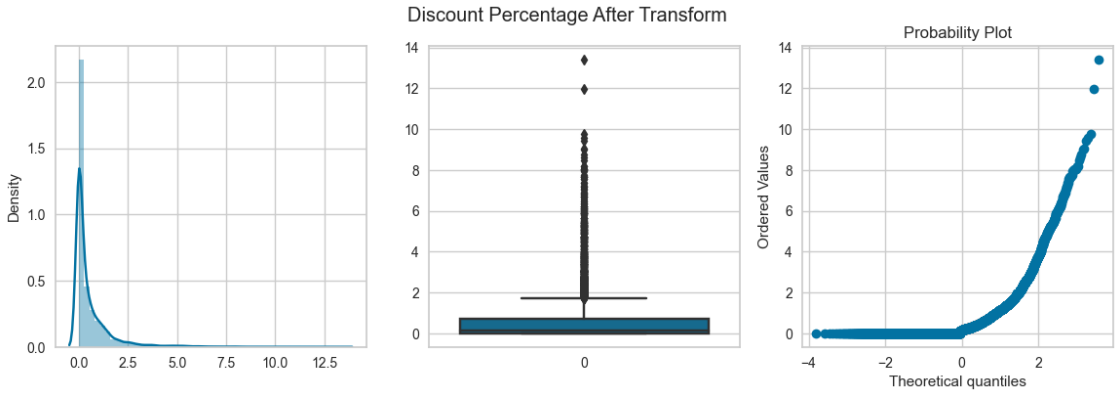

```
posx and posy should be finite values
posx and posy should be finite values
```



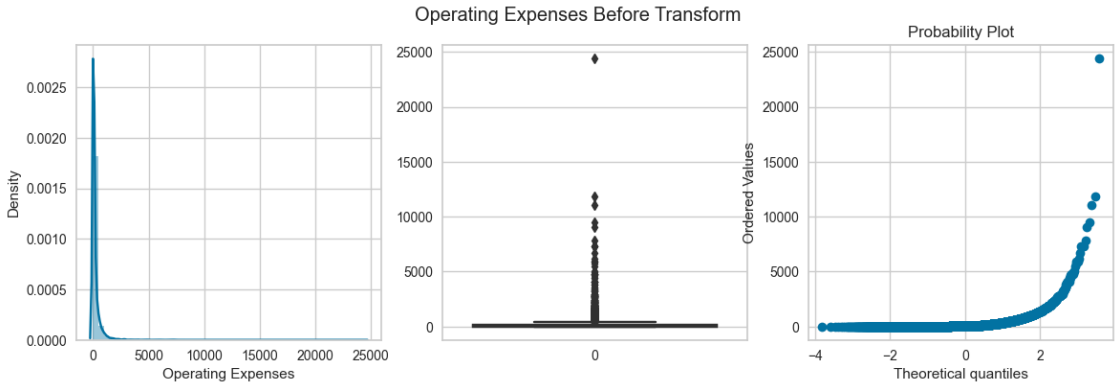

```
posx and posy should be finite values
posx and posy should be finite values
```



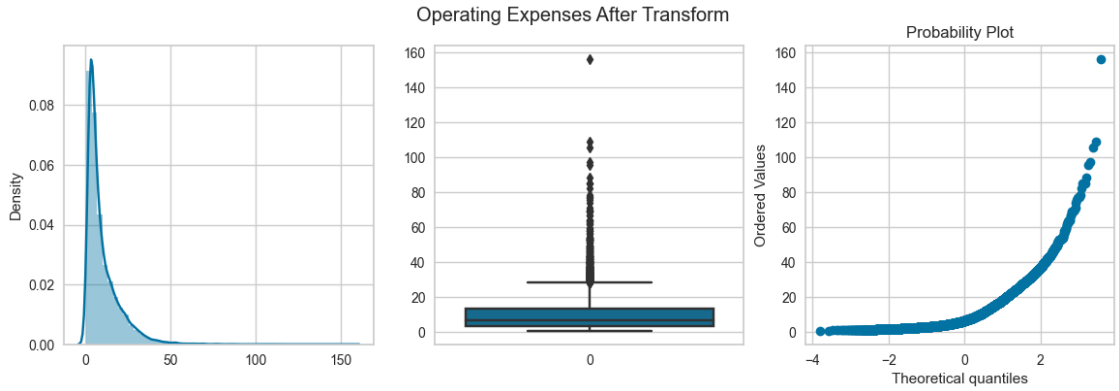

```
posx and posy should be finite values
posx and posy should be finite values
```



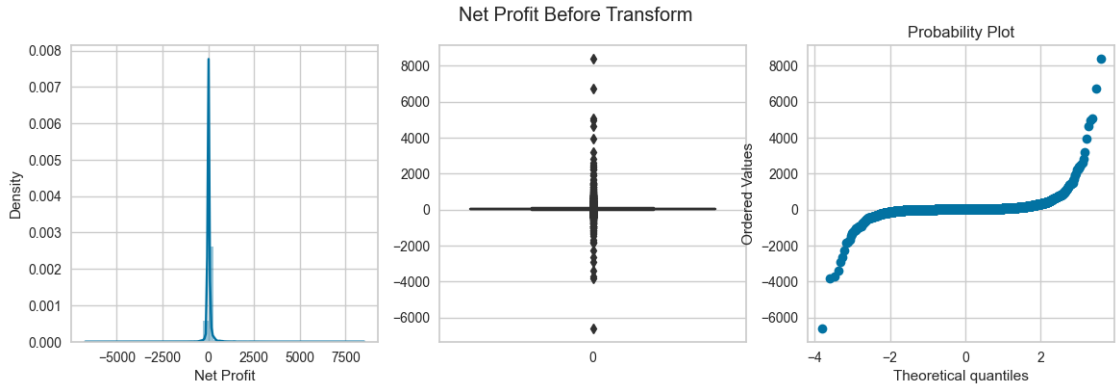

```
posx and posy should be finite values
posx and posy should be finite values
```



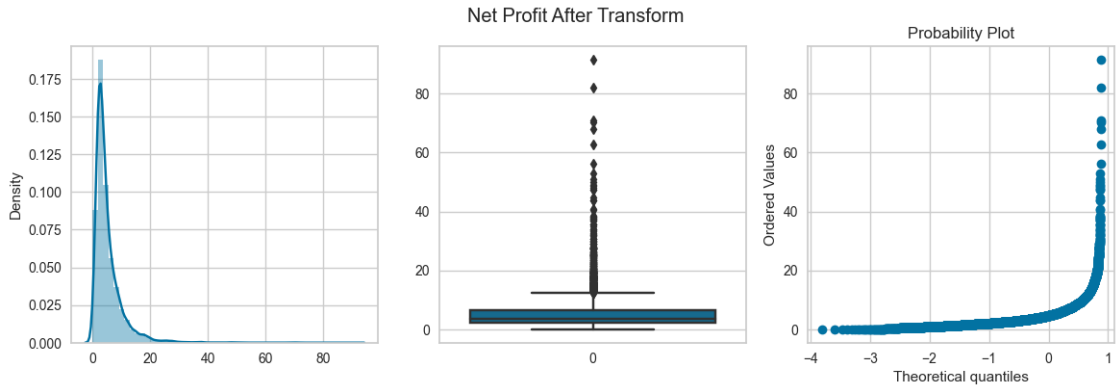
```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

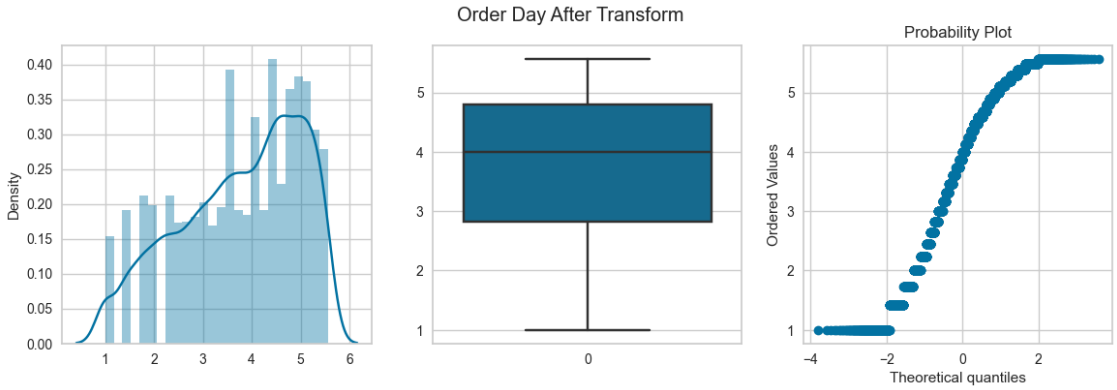



```
posx and posy should be finite values
posx and posy should be finite values
```



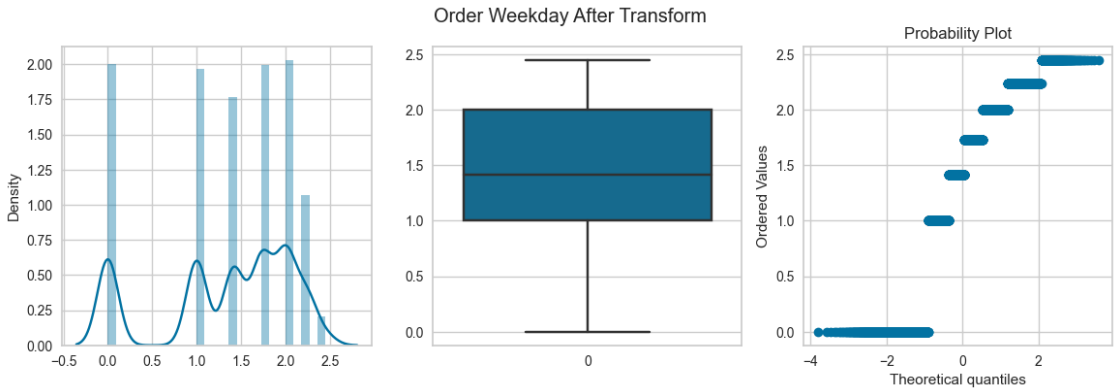
```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```



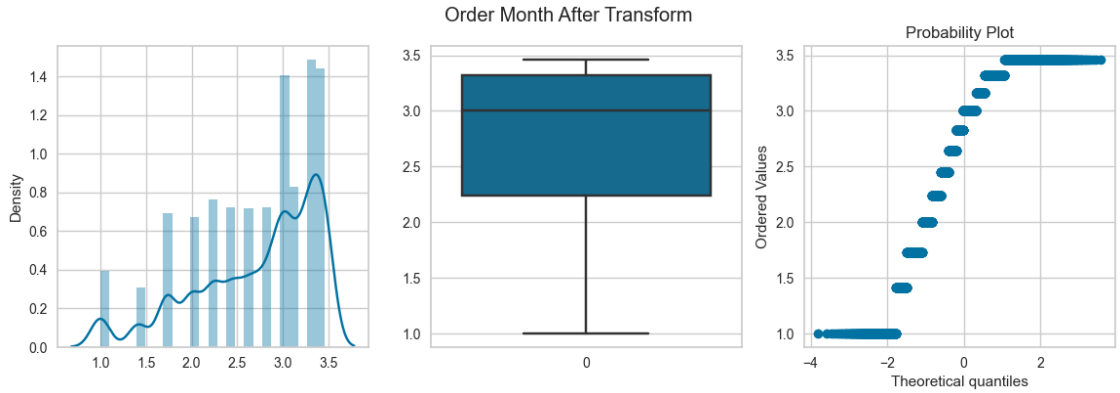

```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```



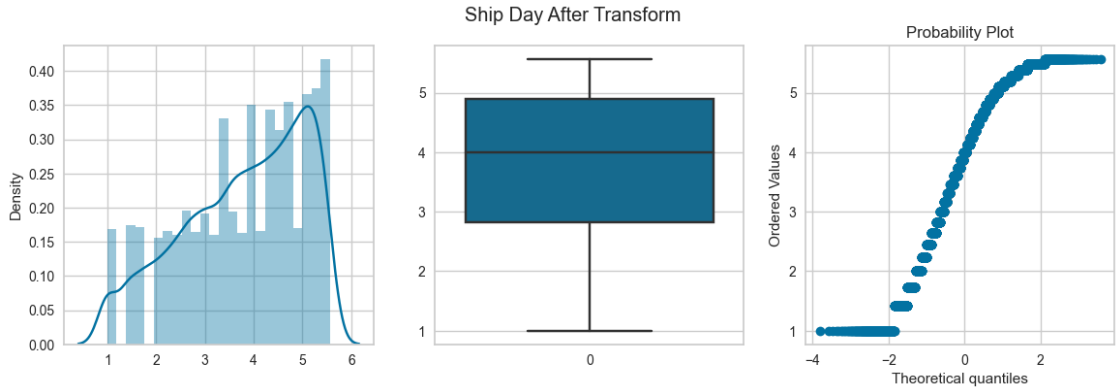

```
posx and posy should be finite values
posx and posy should be finite values
```

```
posx and posy should be finite values
posx and posy should be finite values
```



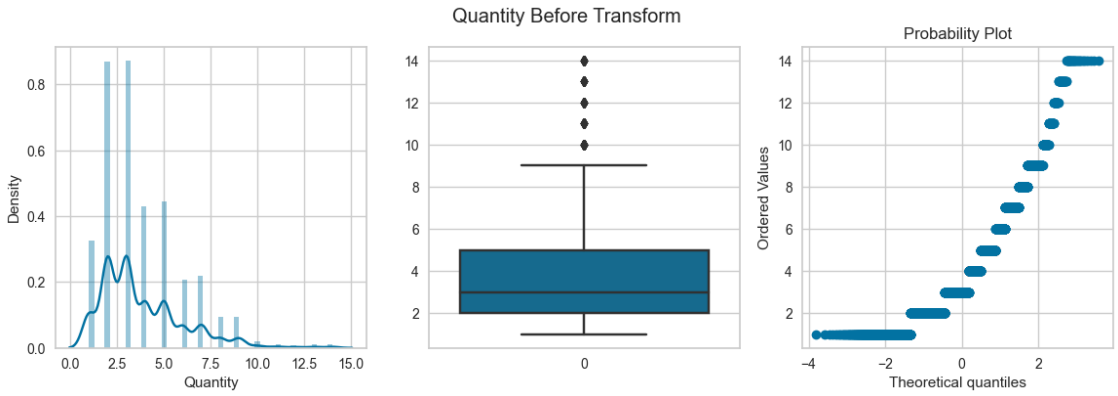

```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```

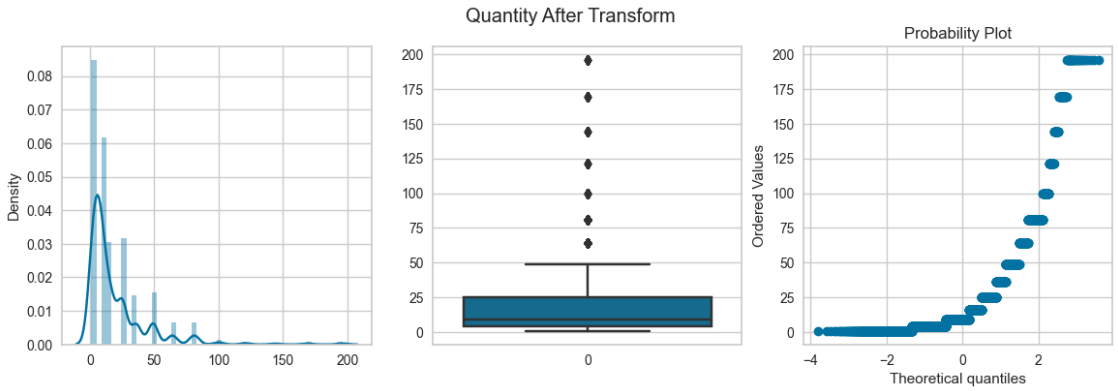


[64]:
```
for col in skewed_cols:
    apply_transform(df,col,FunctionTransformer(np.square))
```

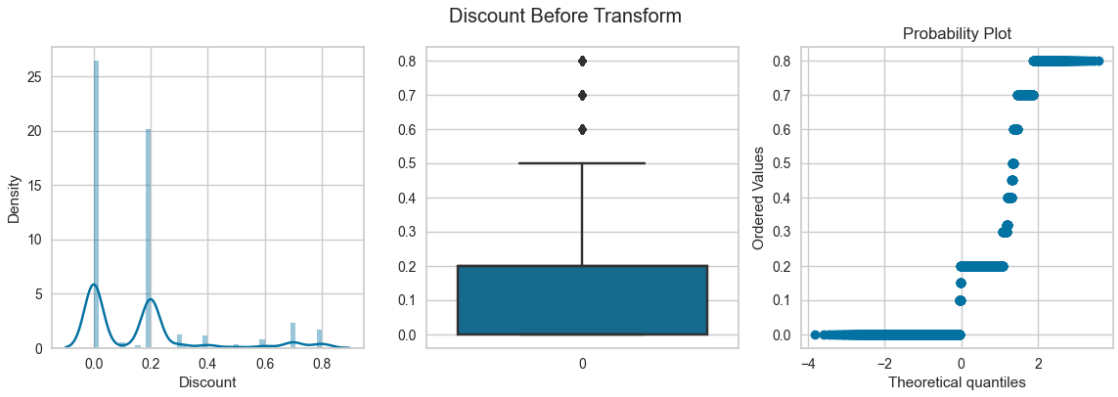
```
posx and posy should be finite values
posx and posy should be finite values
```



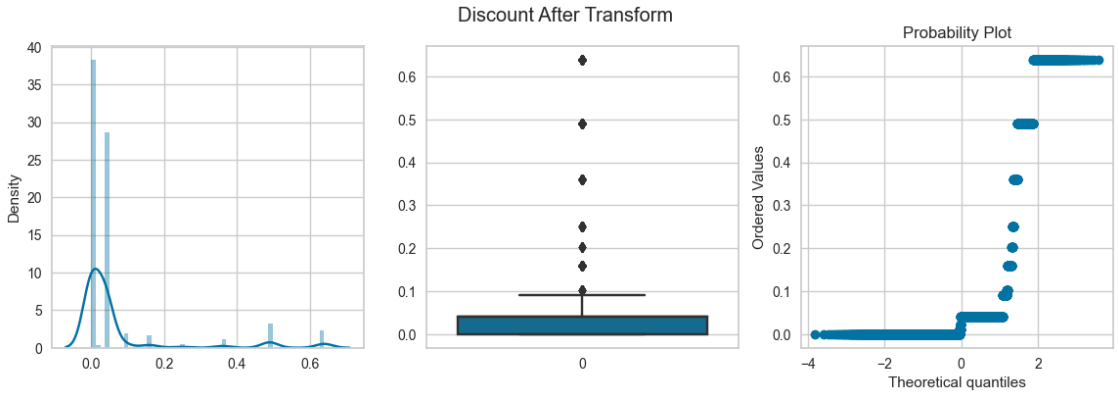

```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

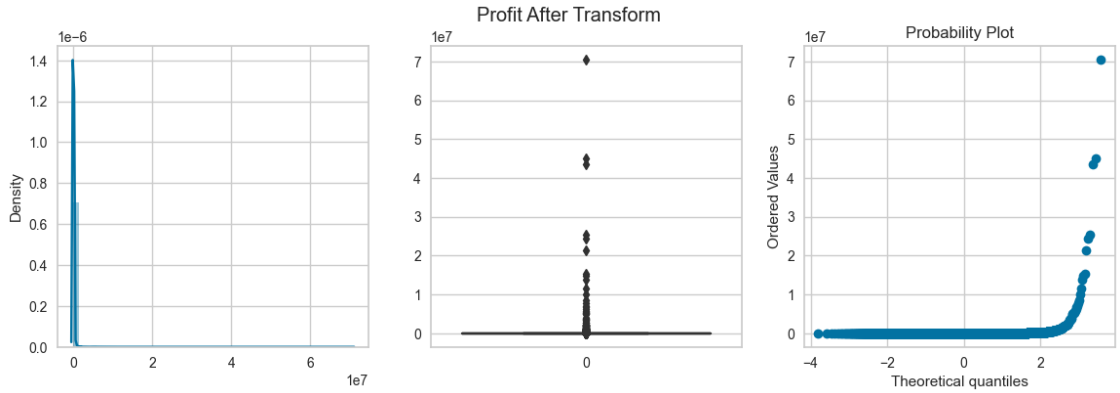



```
posx and posy should be finite values
posx and posy should be finite values
```

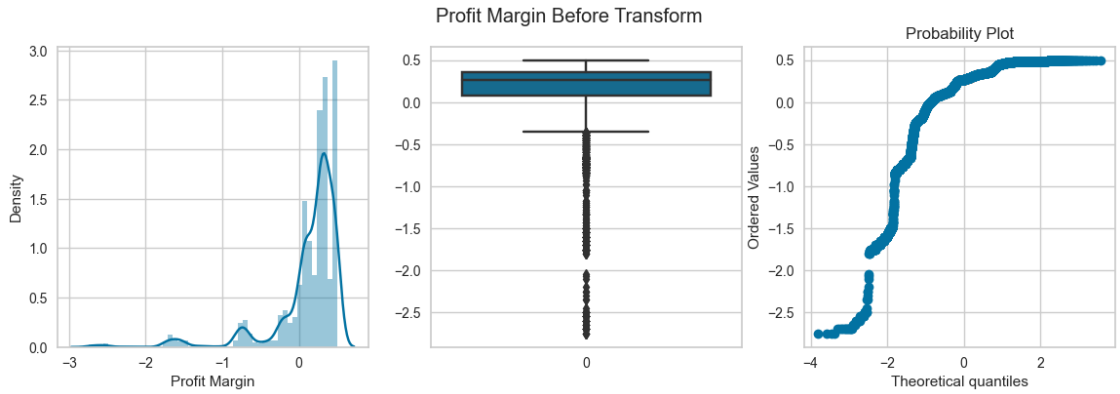



```
posx and posy should be finite values
posx and posy should be finite values
```

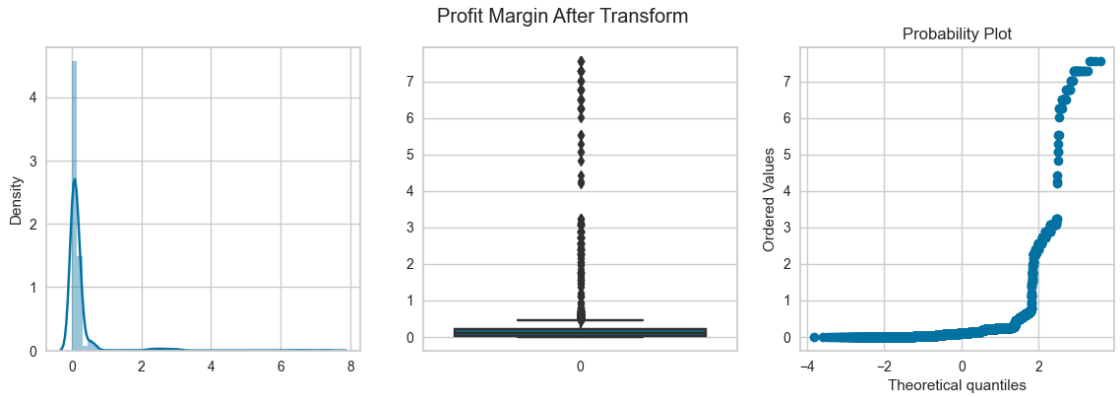
```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

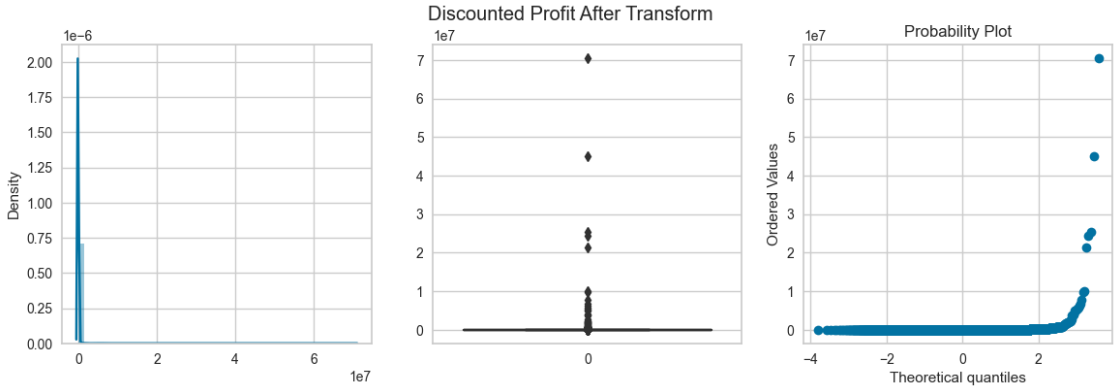



```
posx and posy should be finite values
posx and posy should be finite values
```

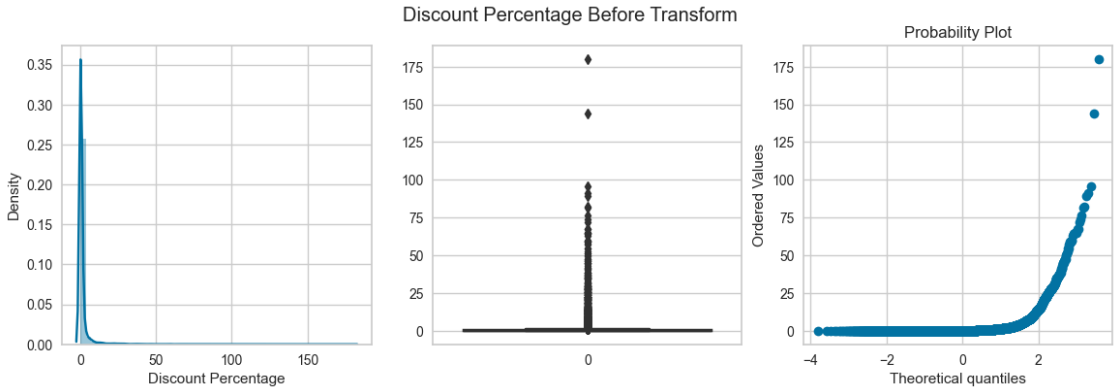


```
posx and posy should be finite values
posx and posy should be finite values
```

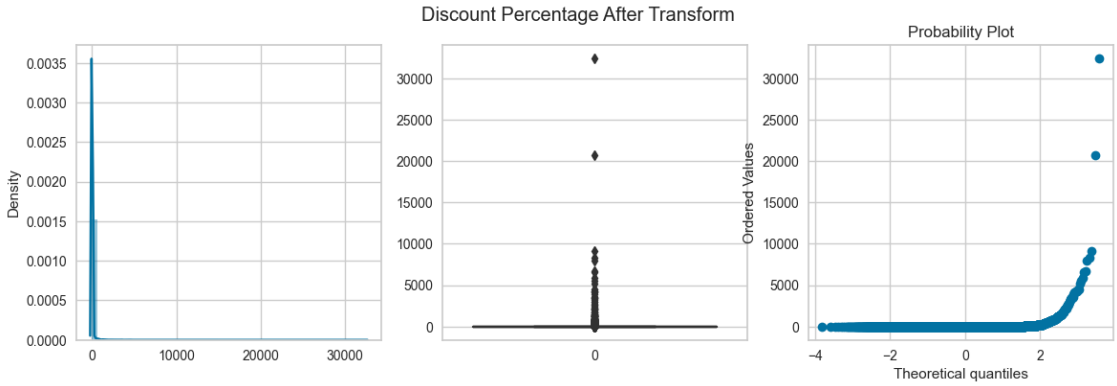

```
posx and posy should be finite values
posx and posy should be finite values
```



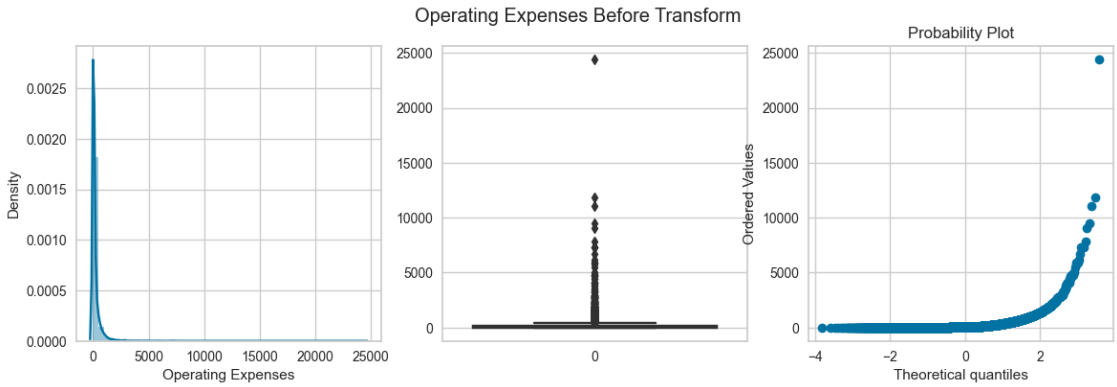

```
posx and posy should be finite values
posx and posy should be finite values
```



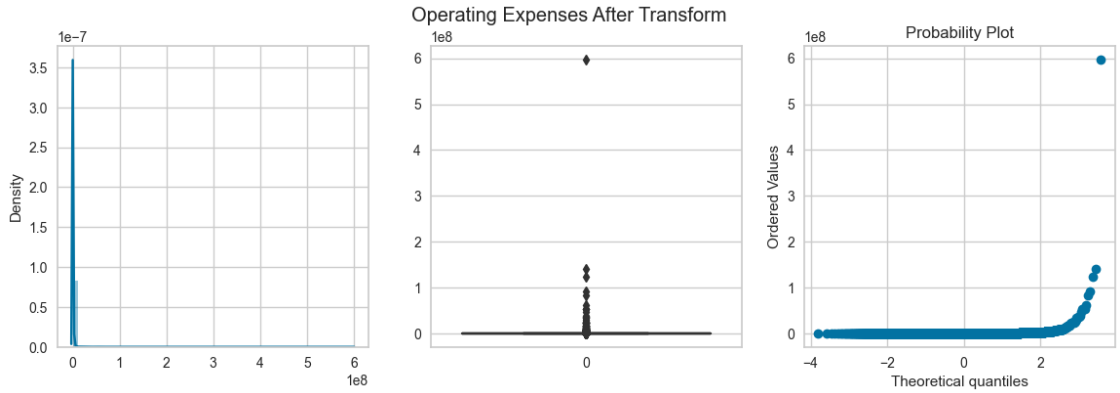

```
posx and posy should be finite values
posx and posy should be finite values
```



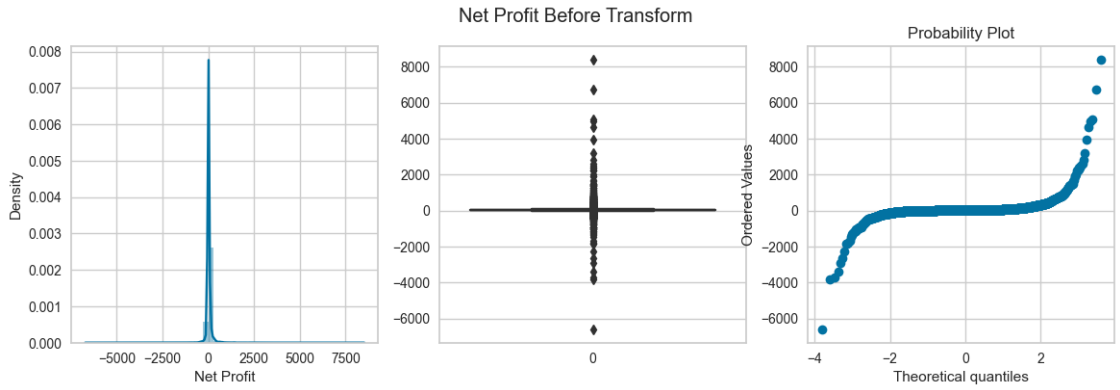

```
posx and posy should be finite values
posx and posy should be finite values
```



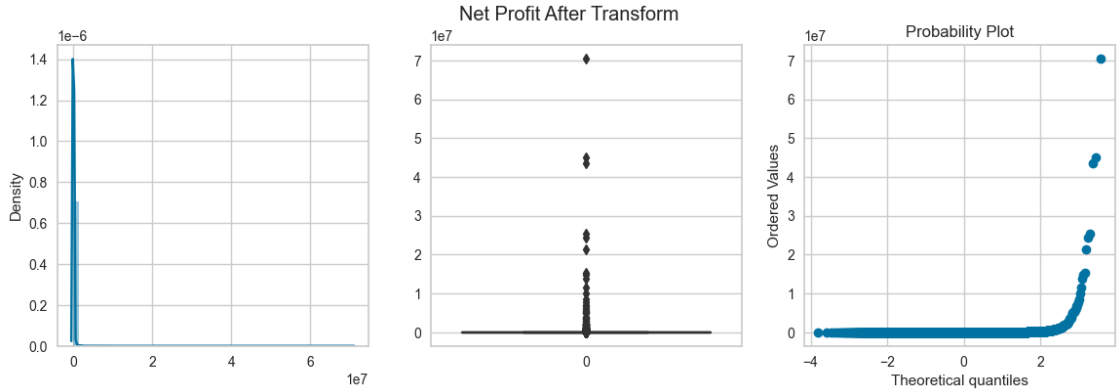
```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```



```
posx and posy should be finite values
posx and posy should be finite values
```

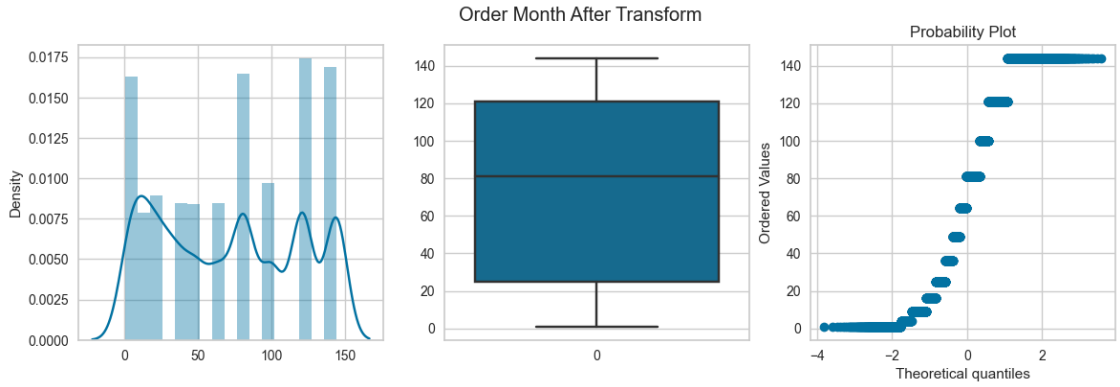

```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```

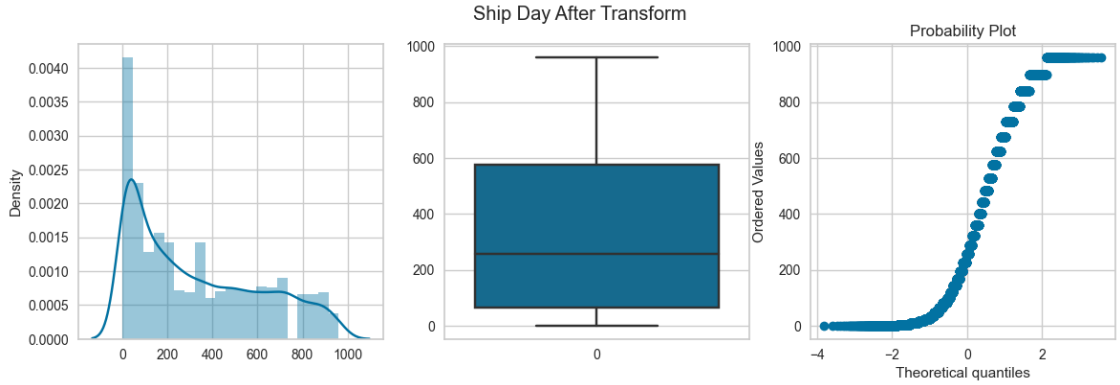

```
posx and posy should be finite values
posx and posy should be finite values
```

```
posx and posy should be finite values
posx and posy should be finite values
```



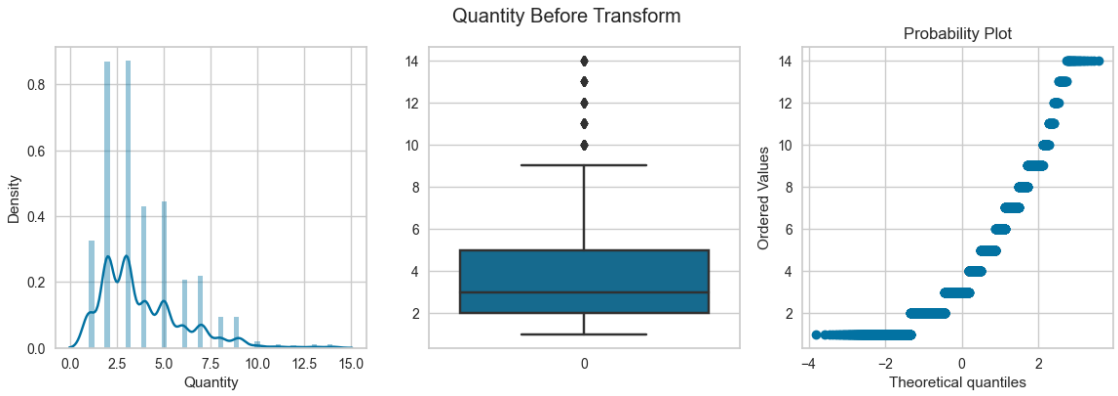

```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```

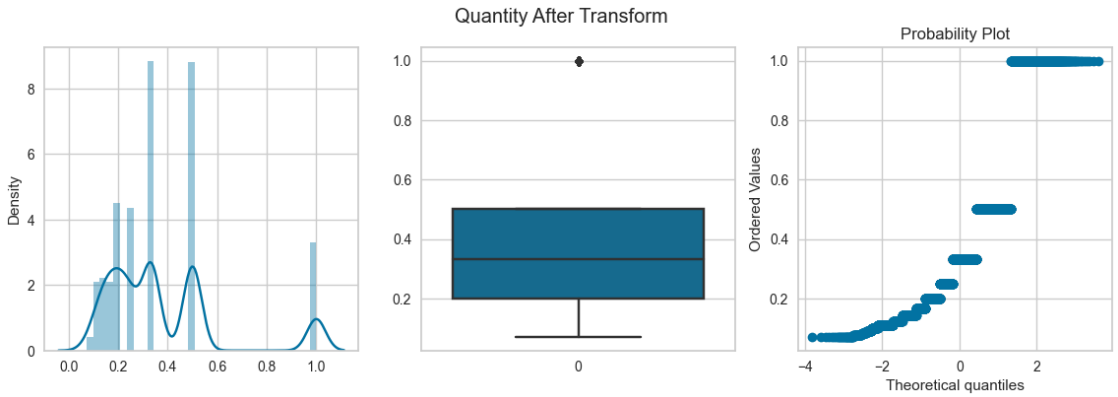


[65]:
```
for col in skewed_cols:
    apply_transform(df,col,FunctionTransformer(lambda x: 1/(x+0.000001)))
```

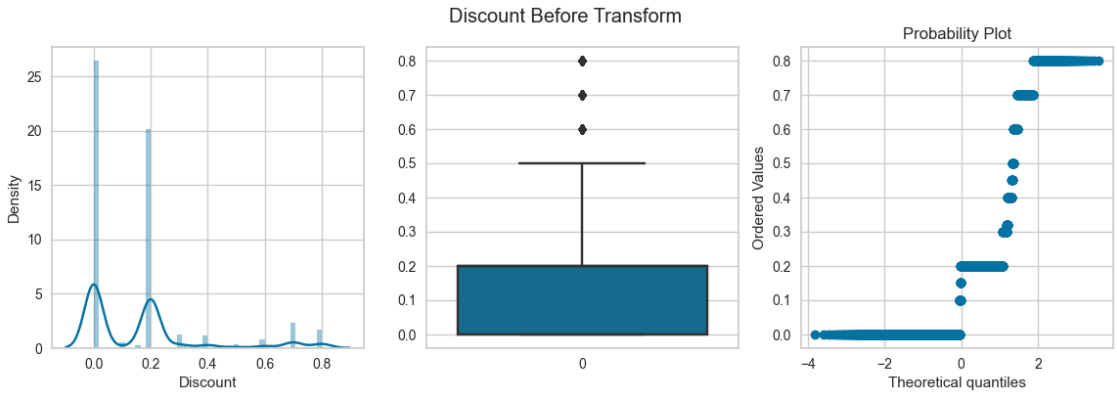
```
posx and posy should be finite values
posx and posy should be finite values
```



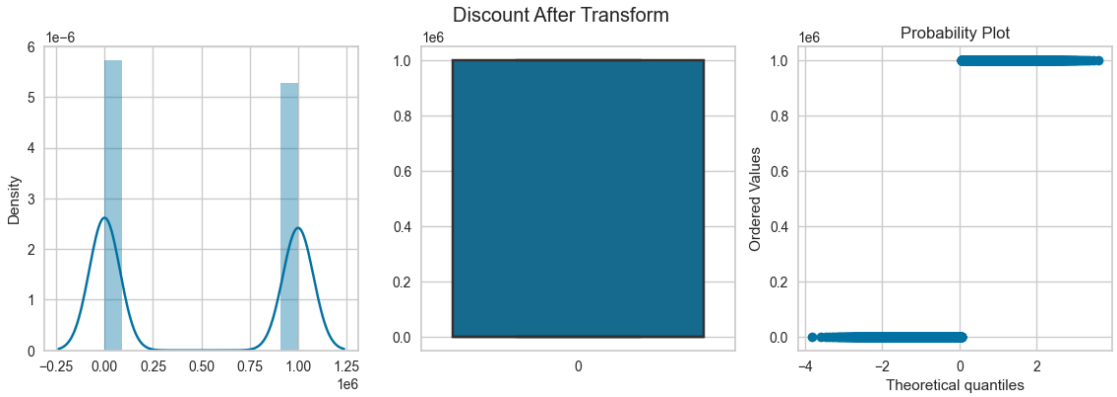

```
posx and posy should be finite values
posx and posy should be finite values
```



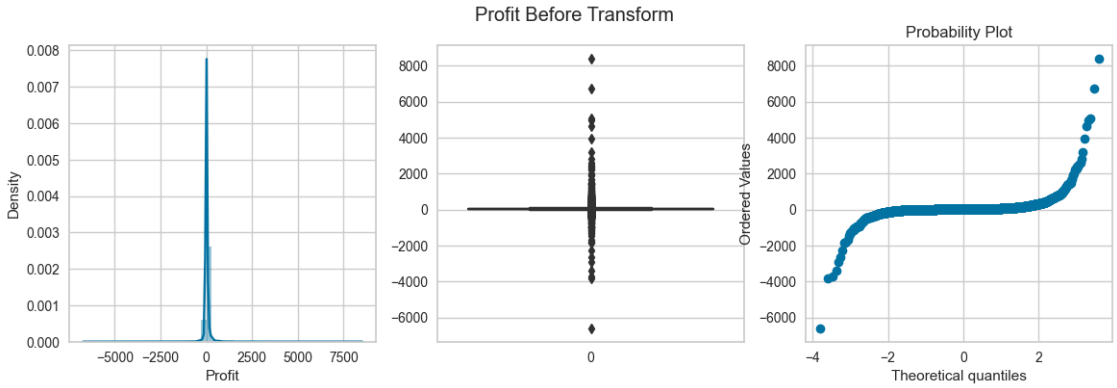

```
posx and posy should be finite values
posx and posy should be finite values
```



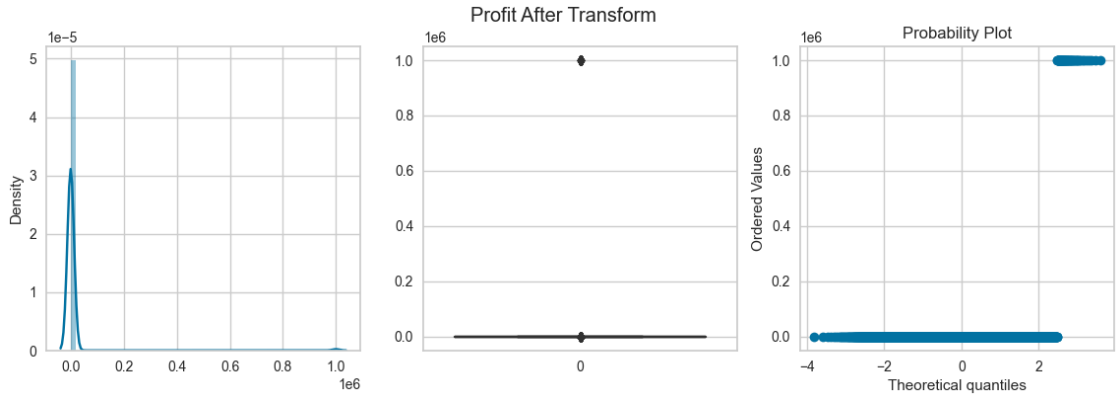

```
posx and posy should be finite values
posx and posy should be finite values
```



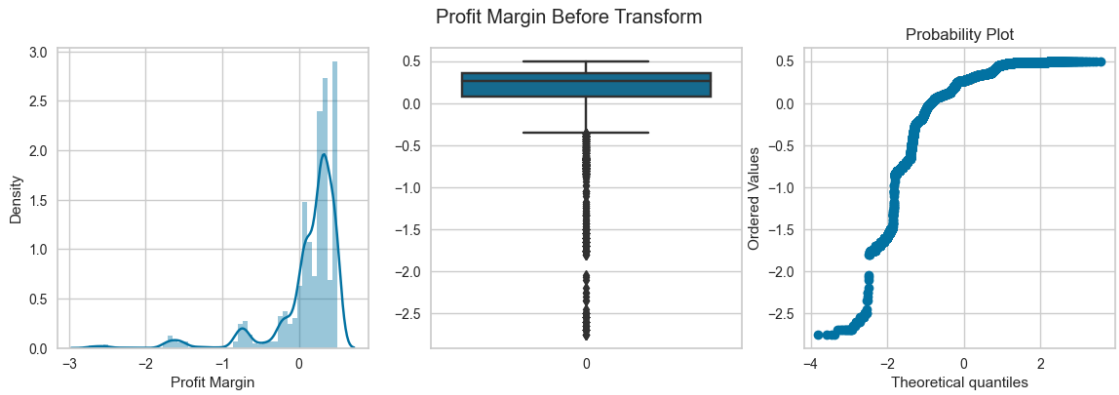

```
posx and posy should be finite values
posx and posy should be finite values
```



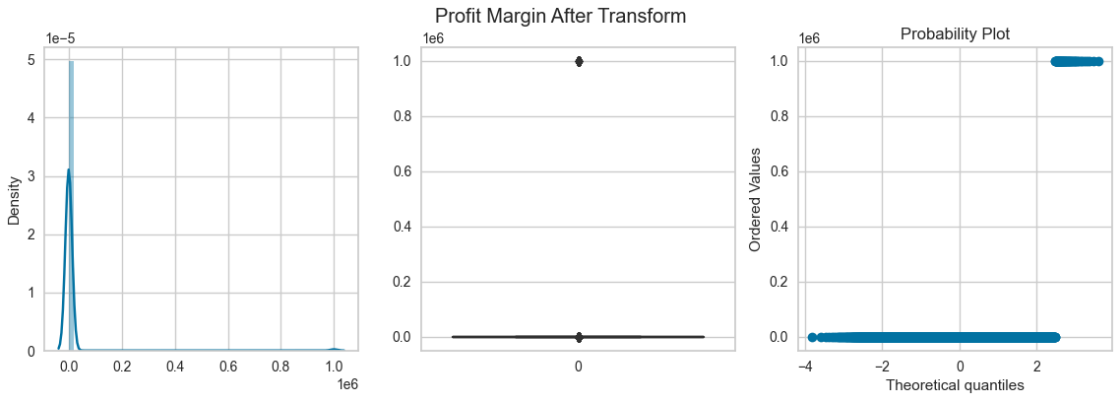
```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```



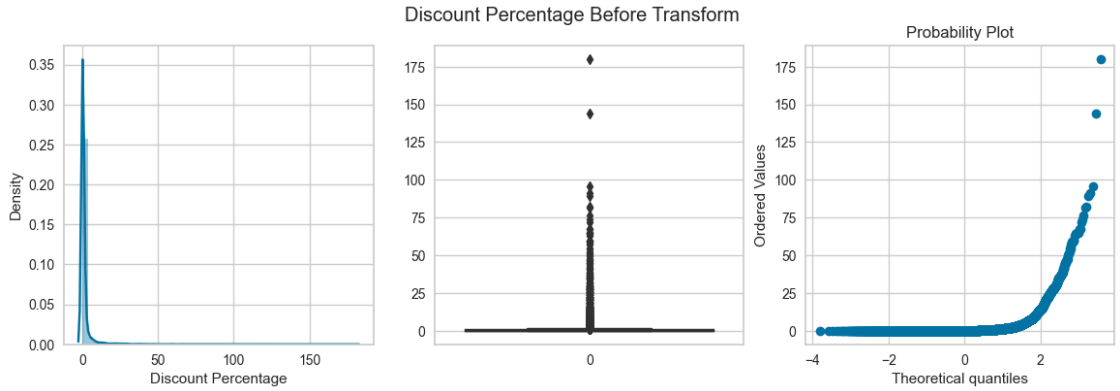

```
posx and posy should be finite values
posx and posy should be finite values
```



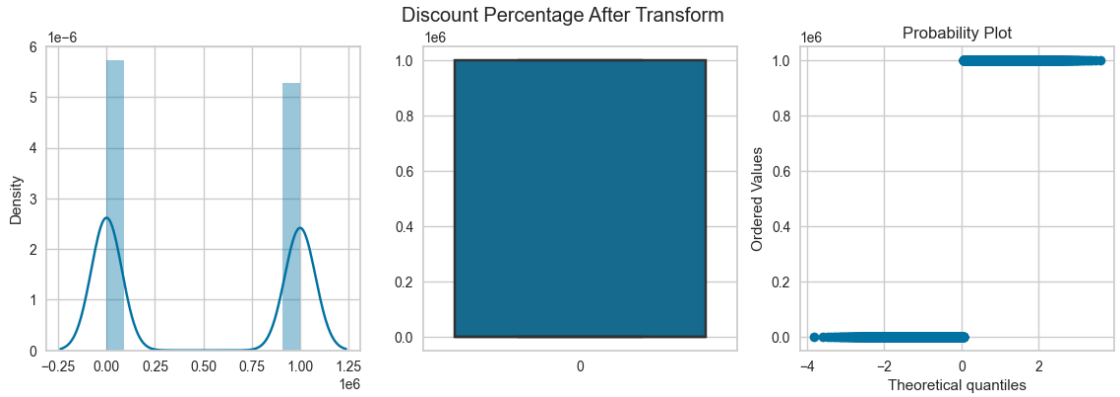

```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```

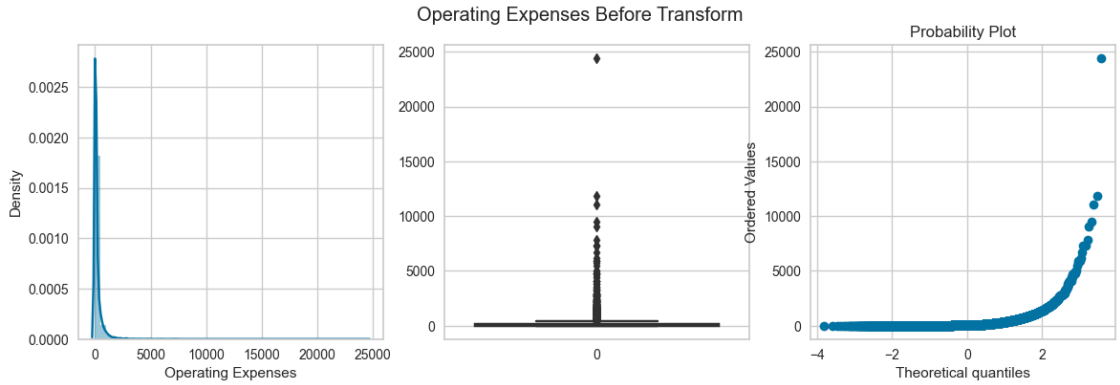
```
posx and posy should be finite values
posx and posy should be finite values
```



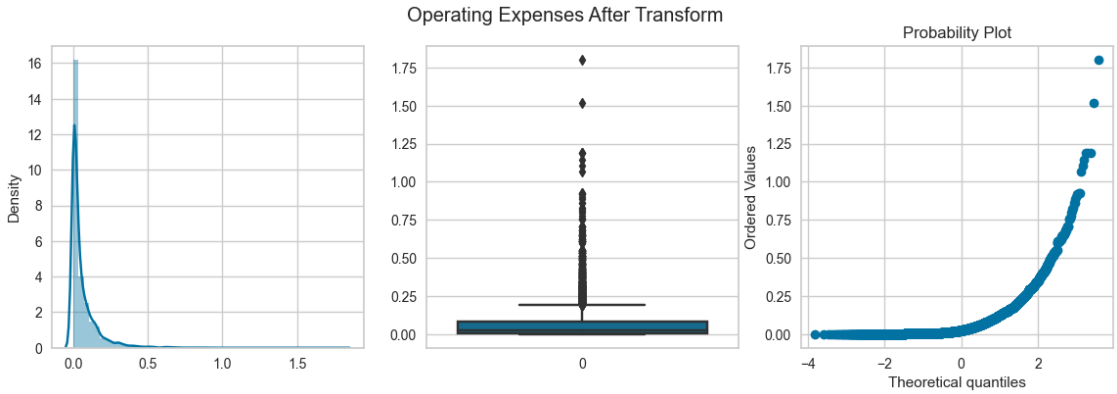

```
posx and posy should be finite values
posx and posy should be finite values
```



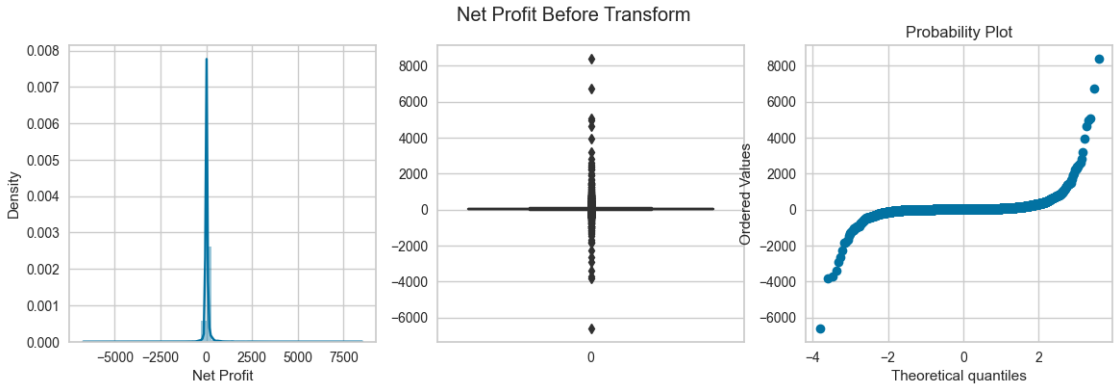

```
posx and posy should be finite values
posx and posy should be finite values
```



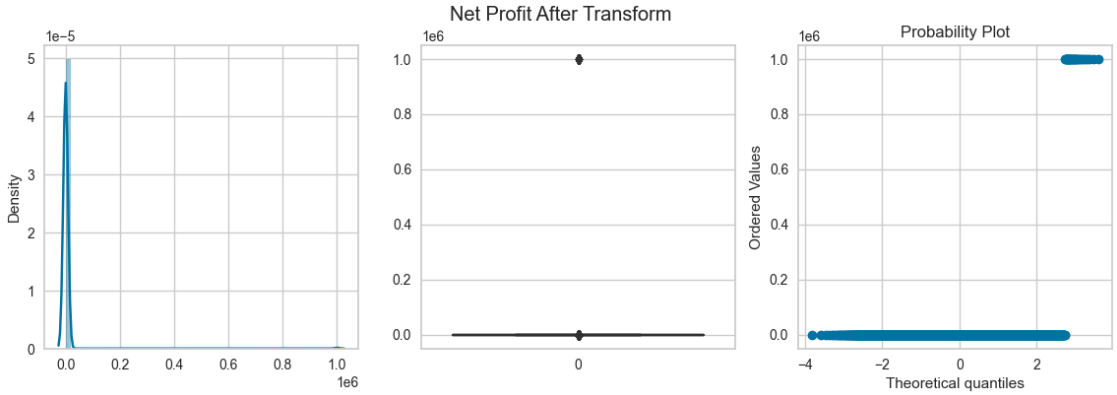
```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

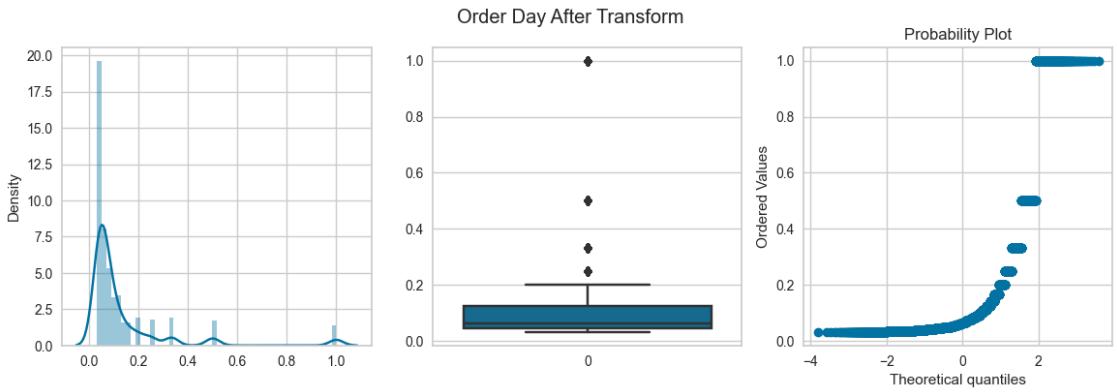



```
posx and posy should be finite values
posx and posy should be finite values
```



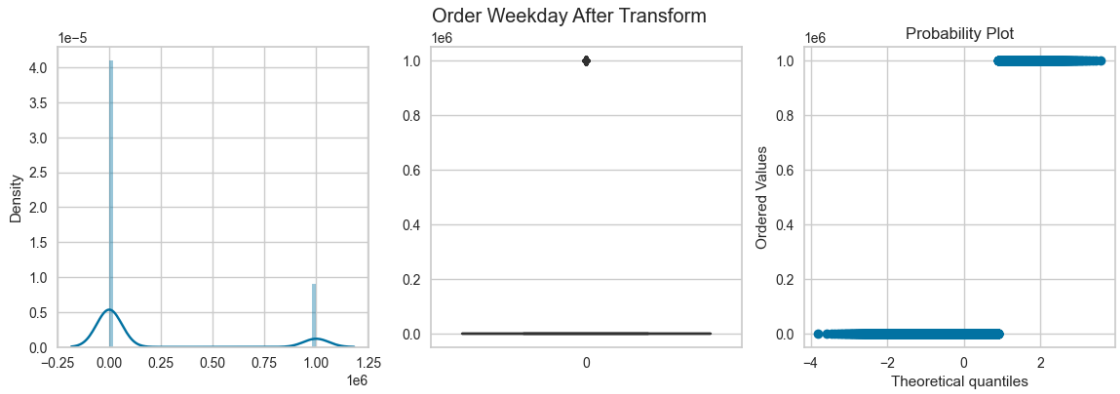

```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```

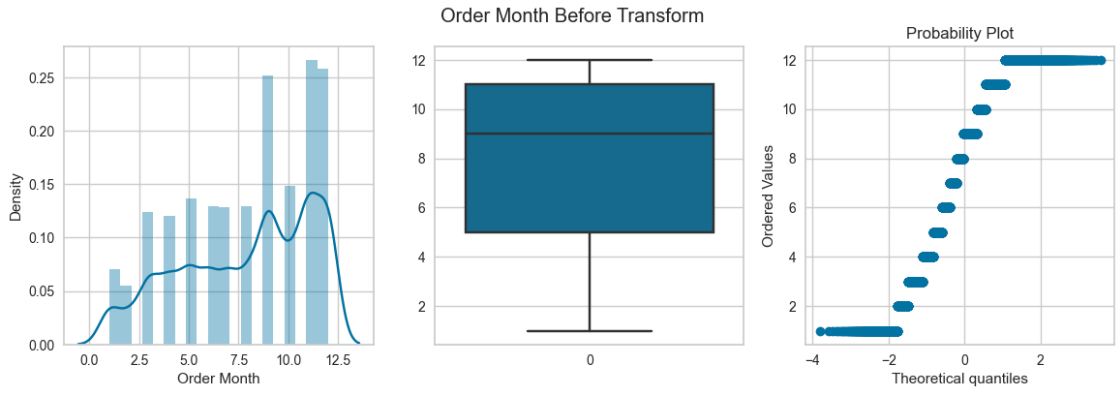


```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```

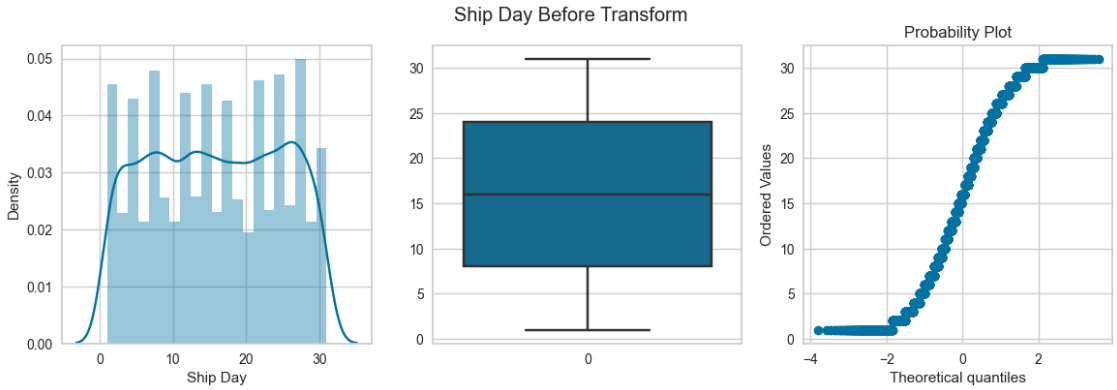



```
posx and posy should be finite values
posx and posy should be finite values
```

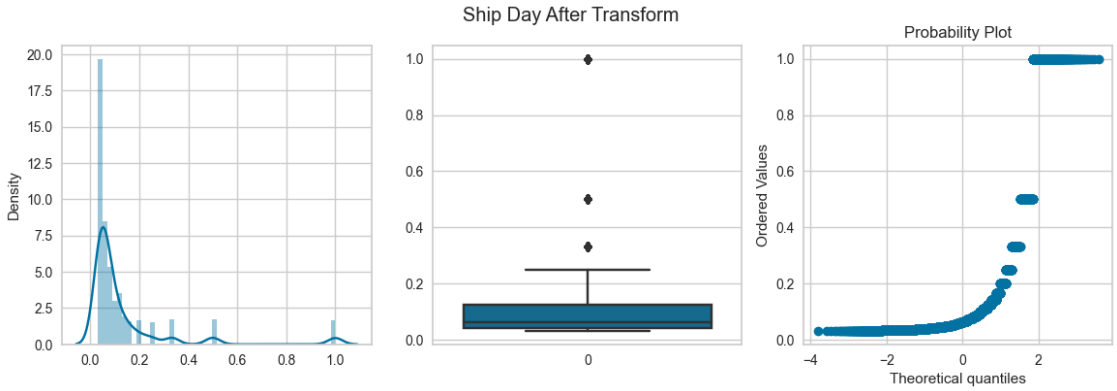


```
posx and posy should be finite values
posx and posy should be finite values
```


```
posx and posy should be finite values
posx and posy should be finite values
```

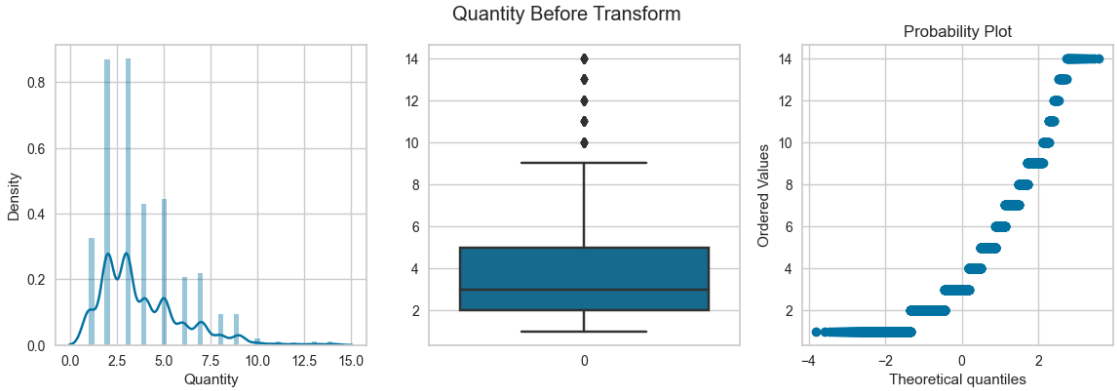



```
posx and posy should be finite values
posx and posy should be finite values
```

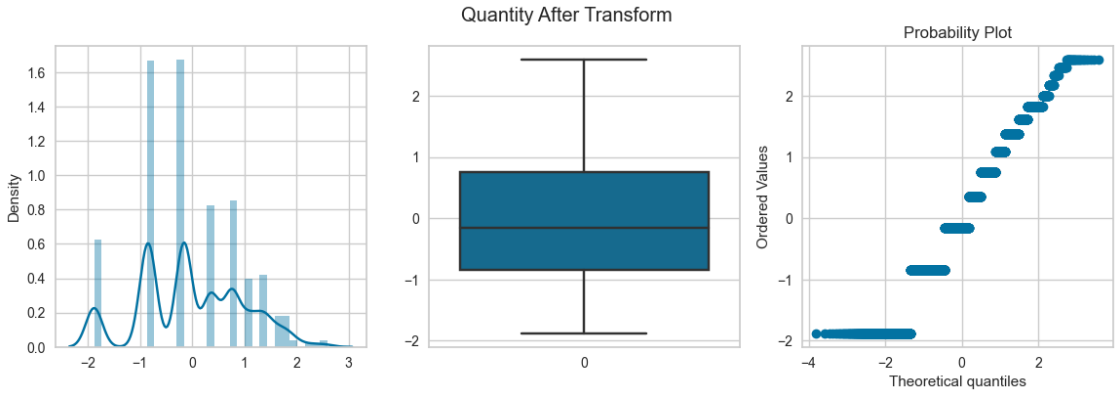


[66]:
```
for col in skewed_cols:
    apply_transform(df,col,PowerTransformer())
```

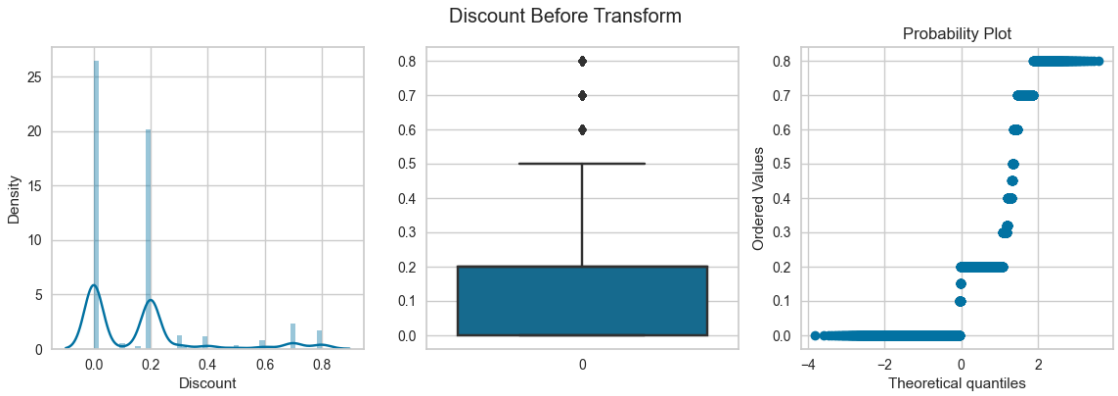
```
posx and posy should be finite values
posx and posy should be finite values
```



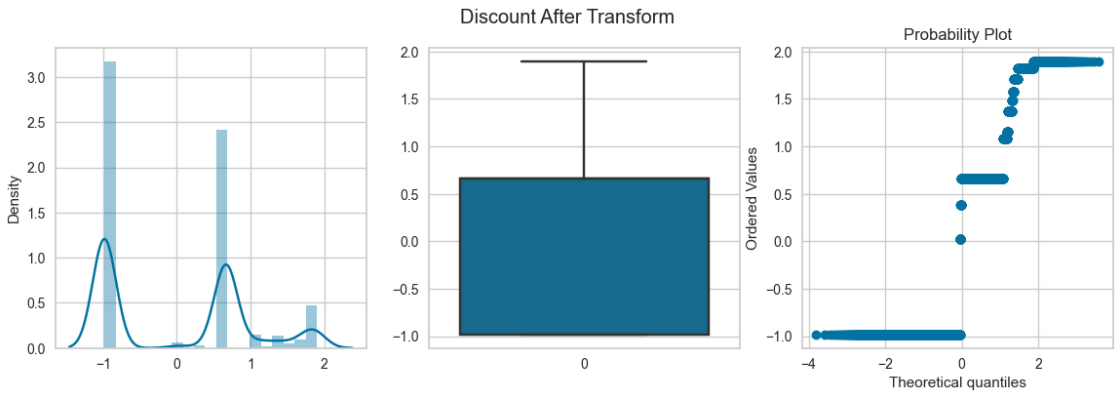

```
posx and posy should be finite values
posx and posy should be finite values
```



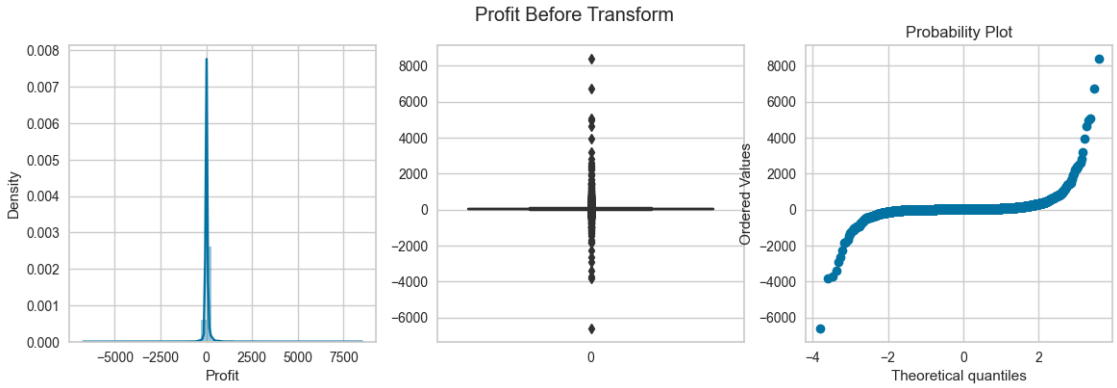

```
posx and posy should be finite values
posx and posy should be finite values
```



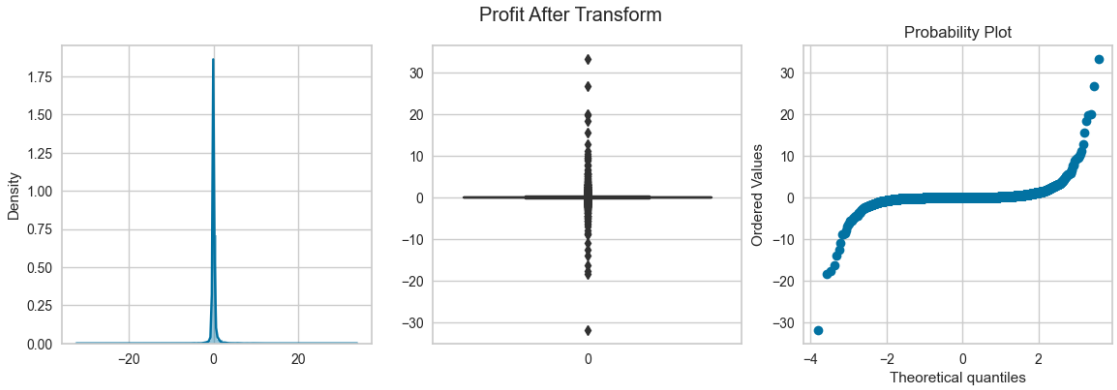

```
posx and posy should be finite values
posx and posy should be finite values
```



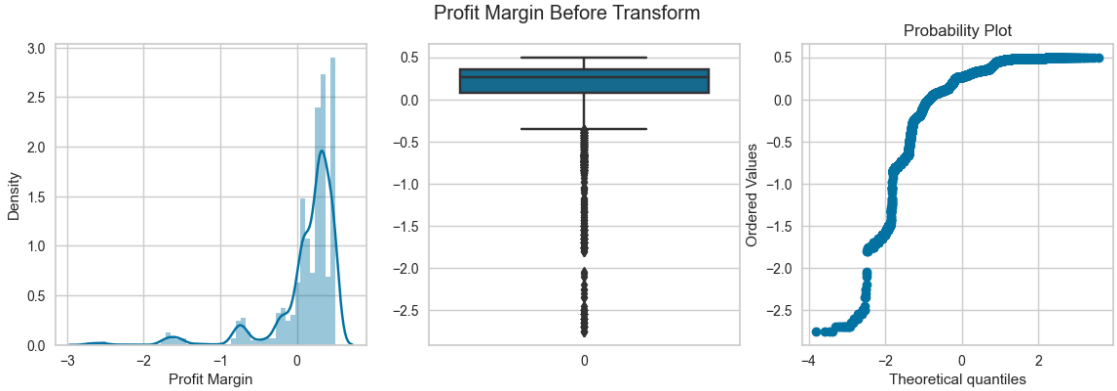
```
posx and posy should be finite values
posx and posy should be finite values
```



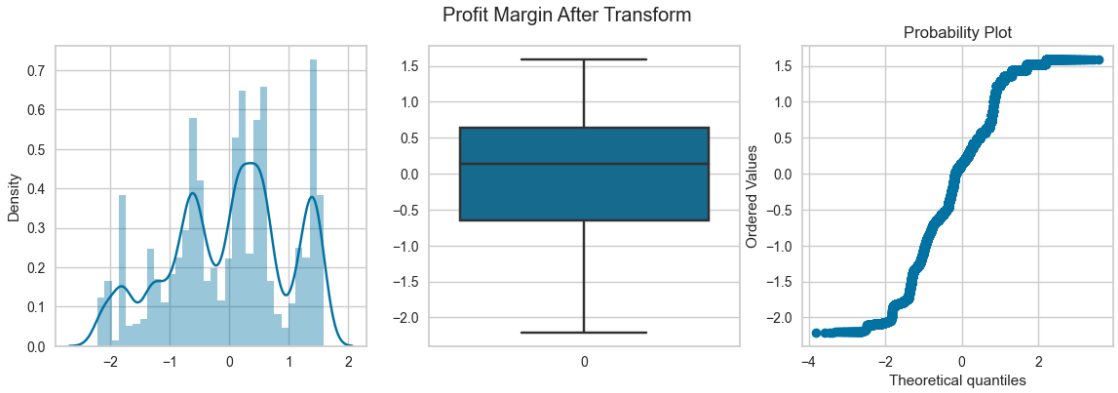

```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

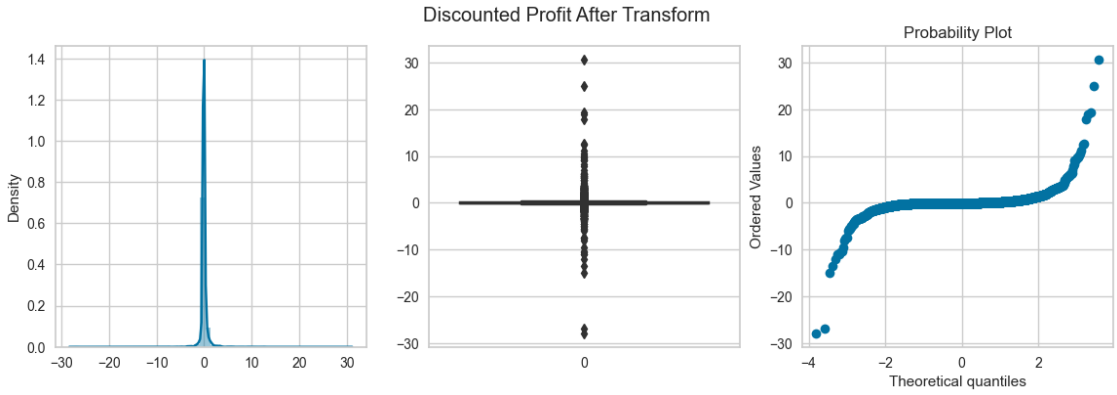


```
posx and posy should be finite values
posx and posy should be finite values
```

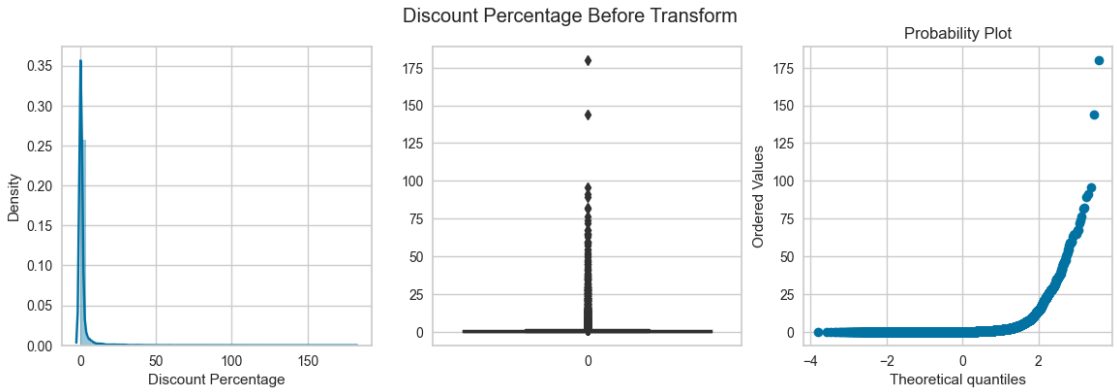



```
posx and posy should be finite values
posx and posy should be finite values
```

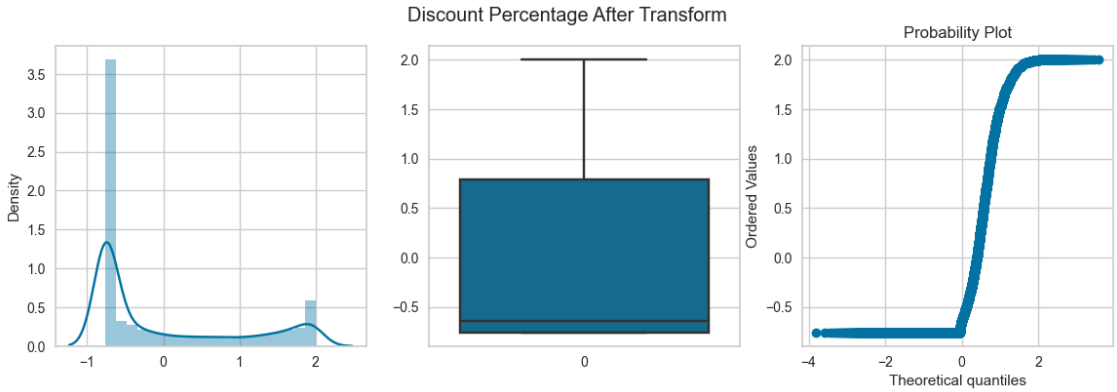

```
posx and posy should be finite values
posx and posy should be finite values
```



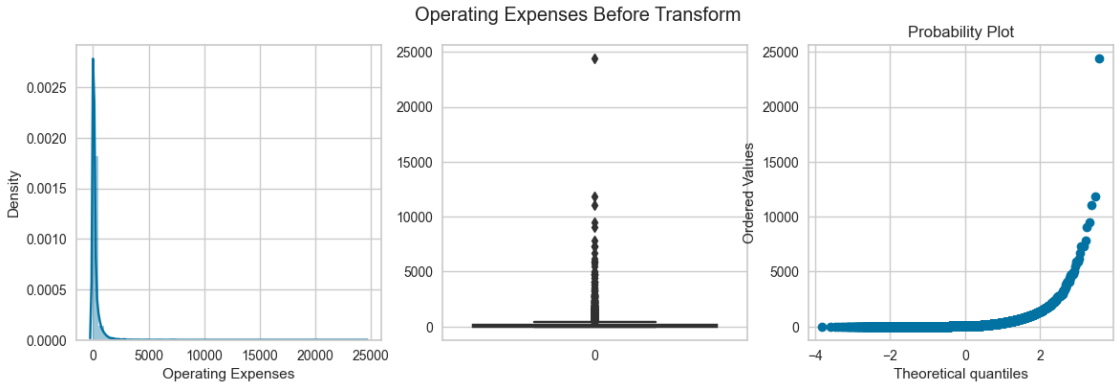

```
posx and posy should be finite values
posx and posy should be finite values
```



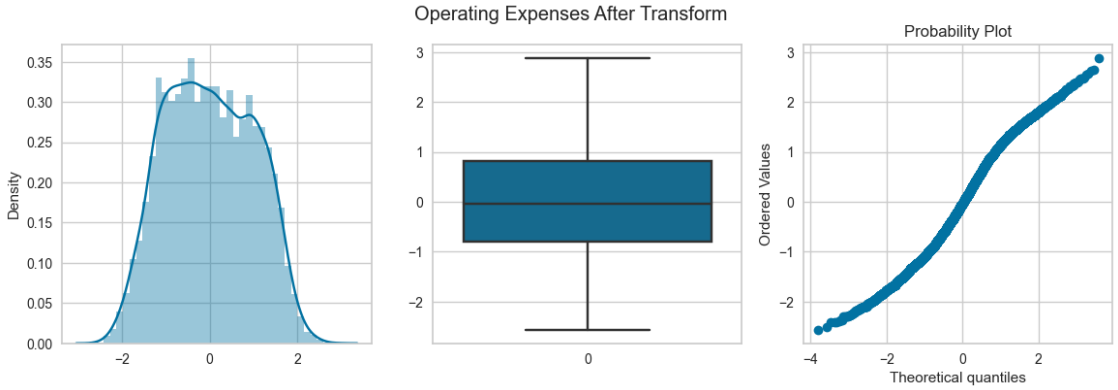

```
posx and posy should be finite values
posx and posy should be finite values
```



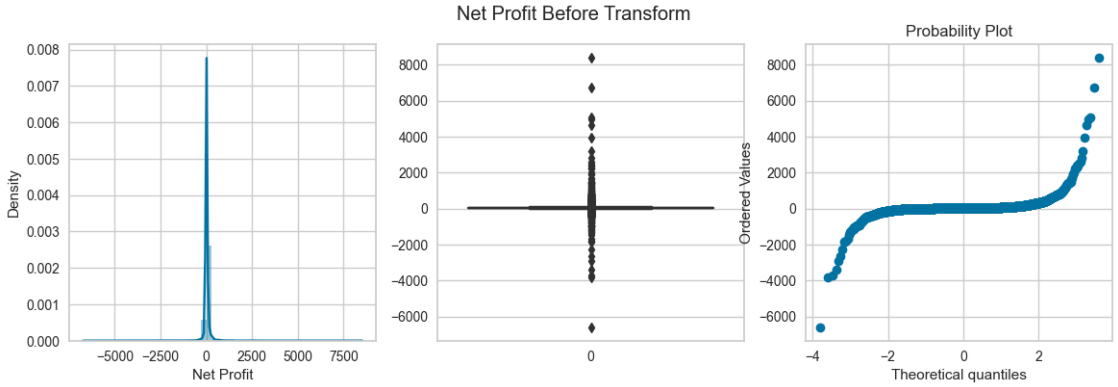
```
posx and posy should be finite values
posx and posy should be finite values
```



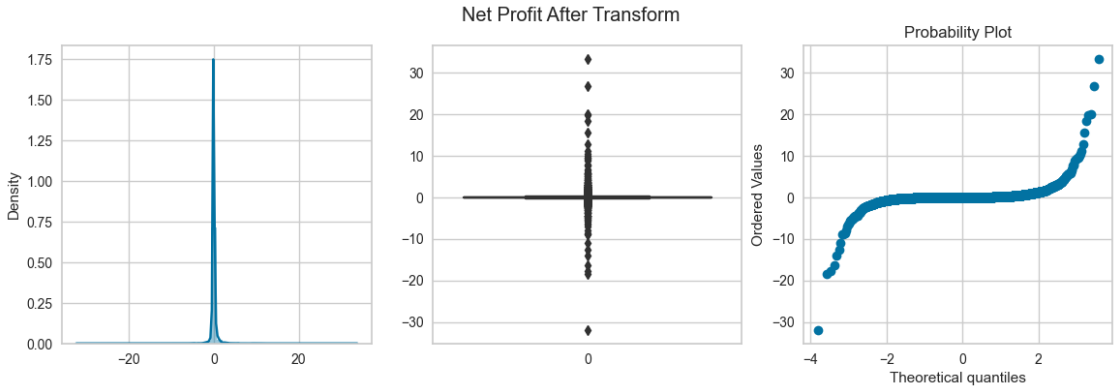

```
posx and posy should be finite values
posx and posy should be finite values
```



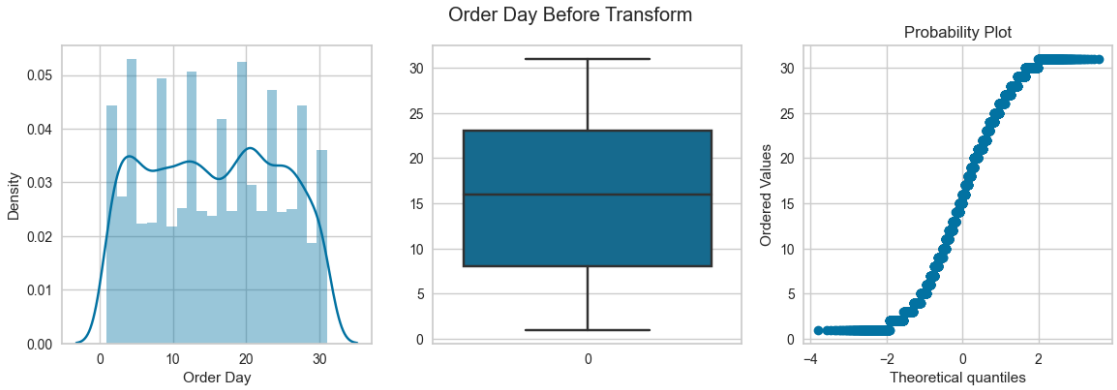

```
posx and posy should be finite values
posx and posy should be finite values
```



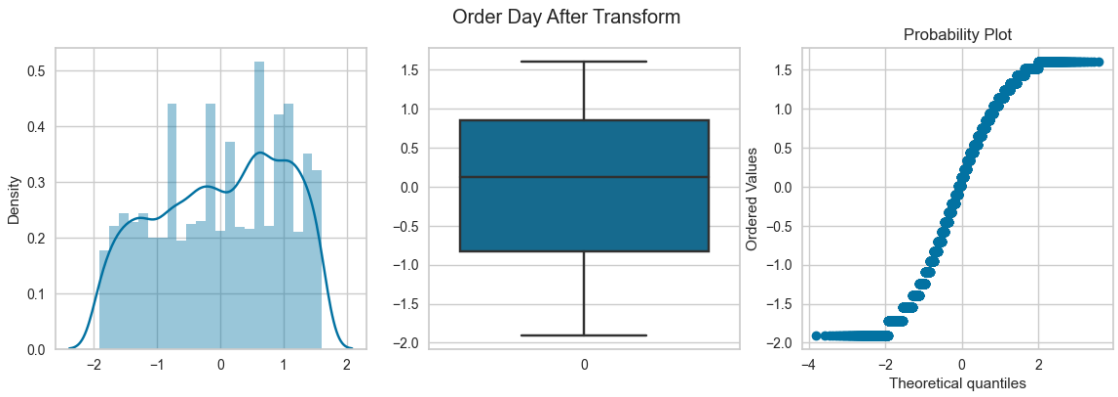
```
posx and posy should be finite values
posx and posy should be finite values
```




```
posx and posy should be finite values
posx and posy should be finite values
```

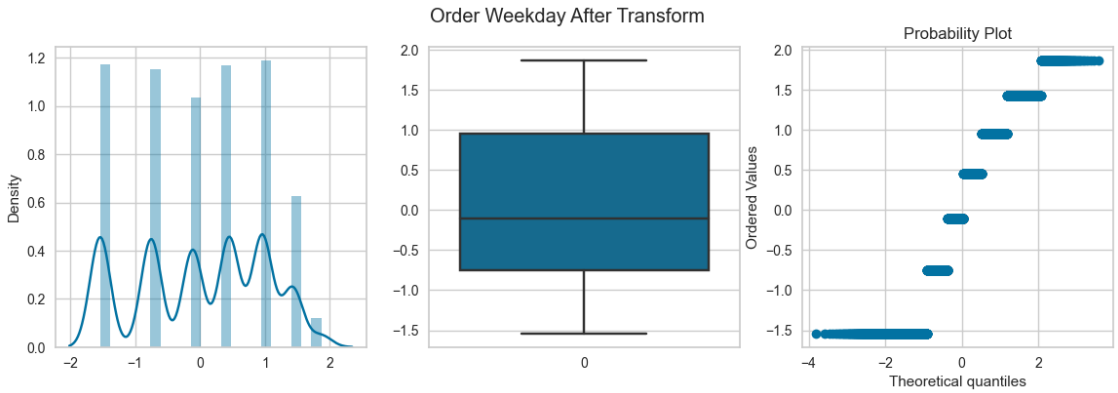



```
posx and posy should be finite values
posx and posy should be finite values
```

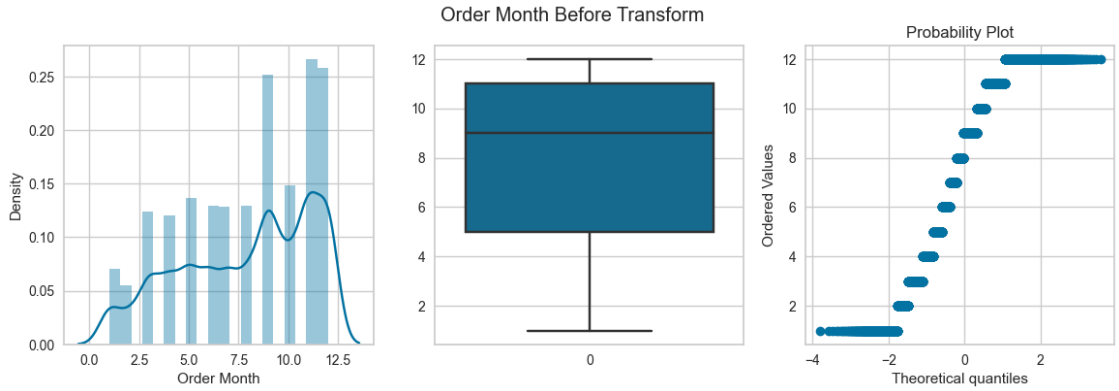



```
posx and posy should be finite values
posx and posy should be finite values
```

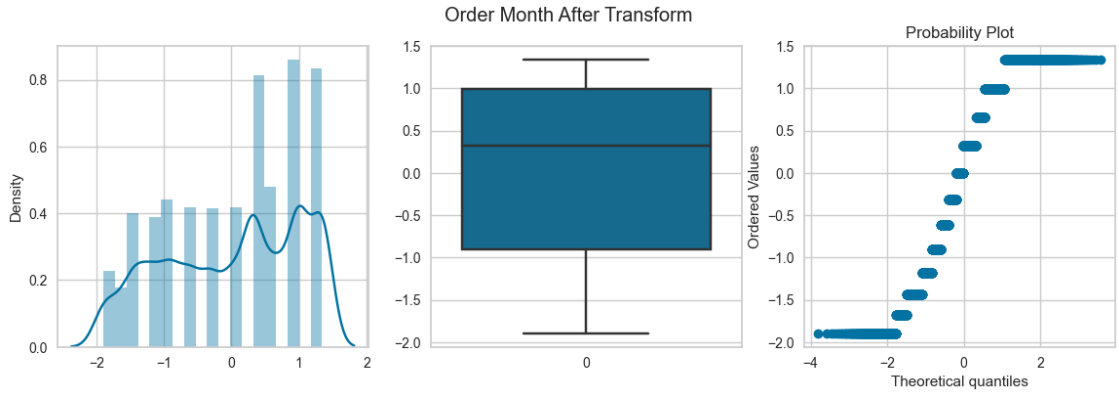

```
posx and posy should be finite values
posx and posy should be finite values
```



```
posx and posy should be finite values
posx and posy should be finite values
```

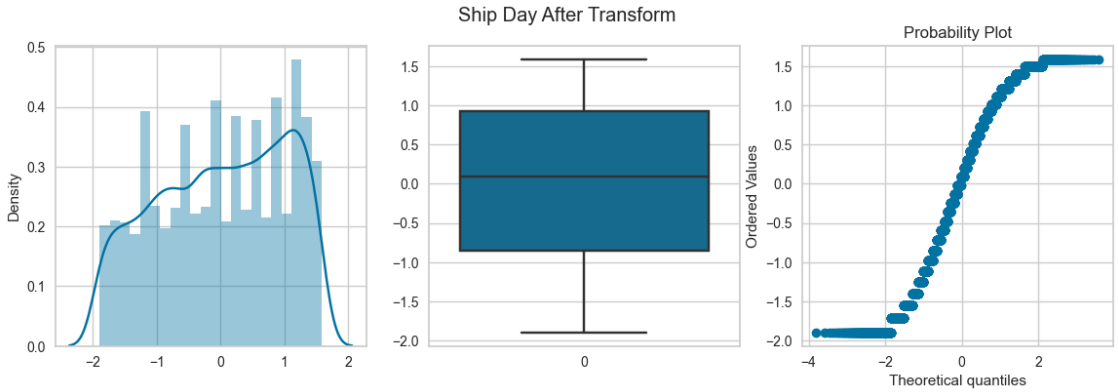



```
posx and posy should be finite values
posx and posy should be finite values
```



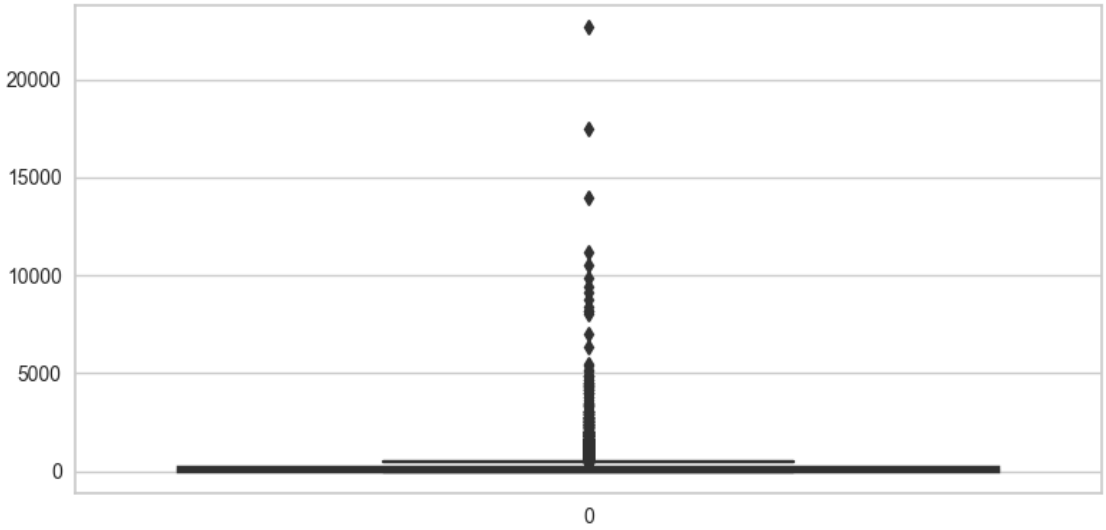

```
posx and posy should be finite values
posx and posy should be finite values
```

```
posx and posy should be finite values
posx and posy should be finite values
```



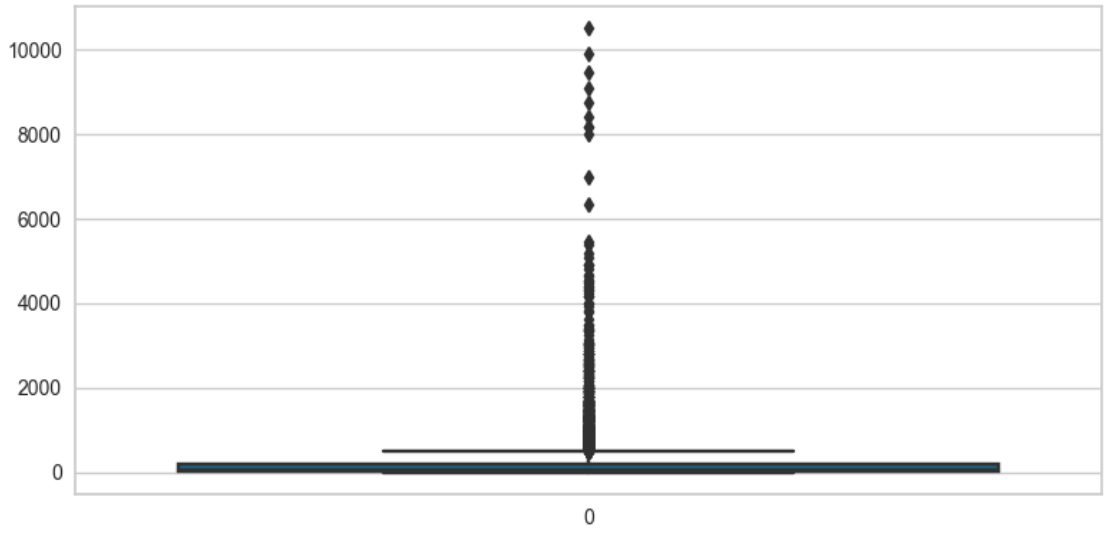
Quantity: Log Transform Discount: Sqrt Transform Profit: Log Transform Profit Margin: Power Transform Discounted Profit: Log Transform Discount Percentage: Power Transform Operating Expenses: Power Transform Net Profit: Log Transform

[67]:
```
plt.figure(figsize=(8,4))
fig = sns.boxplot(df['Sales'])
plt.tight_layout()
plt.show(fig)
plt.close('all')
del fig
gc.collect();
```



[68]:
```
for idx in list(df[df['Sales'] > 10500].index):
    df.drop(idx,axis=0,inplace=True)
```

[69]:
```
plt.figure(figsize=(8,4))
fig = sns.boxplot(df['Sales'])
plt.tight_layout()
plt.show(fig)
plt.close('all')
del fig
gc.collect();
```



0.5.4 Feature Splitting

[75]:
```
df.isnull().sum()
```

[75]:
```
Postal Code                                                          1
Sales                                                                1
Quantity                                                             1
Discount                                                             1
Profit                                                               1
                                                                    ..
Product Name_Logitech 910-002974 M325 Wireless Mouse for Web Scrolling    0
Product Name_Situations Contoured Folding Chairs, 4/Set              0
Product Name_Staples                                                 0
Product Name_Storex Dura Pro Binders                                 0
Product Name_infrequent_sklearn                                      0
Length: 67, dtype: int64
```

[76]:
```
df = df.dropna()
df.shape
```

[76]:
```
(9988, 67)
```

[77]:
```
X = df.drop('Sales',axis=1)
y = df['Sales']
```

[78]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
```

0.5.5 Feature Selection

[79]:
```
X_train.shape, X_test.shape
```

[79]:
```
((6991, 66), (2997, 66))
```

[80]:
```
pipeline = Pipeline(steps=[
    ('constant',DropConstantFeatures()),
    ('duplicate',DropDuplicateFeatures()),
    ('correlated',DropCorrelatedFeatures())
])

X_train = pipeline.fit_transform(X_train)
X_test = pipeline.transform(X_test)
```

[81]:
```
X_train.shape, X_test.shape
```

[81]:
```
((6991, 55), (2997, 55))
```

[82]:
```
kbest = SelectKBest(k=10,score_func=f_regression)
kbest.fit(X_train,y_train)
```

[82]:
```
SelectKBest(score_func=<function f_regression at 0x00000193B33160E0>)
```

[83]:
```
selected_features = kbest.get_feature_names_out()
selected_features
```

[83]:
```
array(['Quantity', 'Discount', 'Profit', 'Discount Percentage',
       'Operating Expenses', 'Category_Office Supplies',
       'Category_Technology', 'Sub-Category_Chairs', 'Sub-Category_Paper',
       'Sub-Category_infrequent_sklearn'], dtype=object)
```

[84]:
```
perc = SelectPercentile(percentile=20,score_func=f_regression)
perc.fit(X_train,y_train)
```

[84]:
```
SelectPercentile(percentile=20,
                 score_func=<function f_regression at 0x00000193B33160E0>)
```

[85]:
```
selected_features = perc.get_feature_names_out()
selected_features
```

[85]:
```
array(['Quantity', 'Discount', 'Profit', 'Discount Percentage',
       'Operating Expenses', 'Category_Office Supplies',
       'Category_Technology', 'Sub-Category_Binders',
       'Sub-Category_Chairs', 'Sub-Category_Paper',
       'Sub-Category_infrequent_sklearn'], dtype=object)
```

[86]:
```
ridge = SelectFromModel(estimator=Ridge(),max_features=10)
ridge.fit(X_train,y_train)
```

[86]:
```
SelectFromModel(estimator=Ridge(), max_features=10)
```

[87]:
```
selected_features = ridge.get_feature_names_out()
selected_features
```

[87]:
```
array(['Profit', 'Operating Expenses'], dtype=object)
```

[88]:
```
rf = SelectFromModel(estimator=RandomForestRegressor(),max_features=10)
rf.fit(X_train,y_train)
```

[88]:
```
SelectFromModel(estimator=RandomForestRegressor(), max_features=10)
```

[89]:
```
selected_features = rf.get_feature_names_out()
selected_features
```

[89]:
```
array(['Profit', 'Discount Percentage', 'Operating Expenses'],
      dtype=object)
```

[90]:
```
rfe =␣
↪RFE(estimator=RandomForestRegressor(),n_features_to_select=10,step=5,verbose=1)
rfe.fit(X_train,y_train)
```

```
Fitting estimator with 55 features.
Fitting estimator with 50 features.
Fitting estimator with 45 features.
Fitting estimator with 40 features.
Fitting estimator with 35 features.
Fitting estimator with 30 features.
Fitting estimator with 25 features.
Fitting estimator with 20 features.
Fitting estimator with 15 features.
```

[90]:
```
RFE(estimator=RandomForestRegressor(), n_features_to_select=10, step=5,
    verbose=1)
```

[91]:
```
selected_features = rfe.get_feature_names_out()
selected_features
```

[91]:
```
array(['Postal Code', 'Discount', 'Profit', 'Discount Percentage',
       'Operating Expenses', 'Order Year', 'Order Month', 'Order Day',
       'Order Weekday', 'Ship Weekday'], dtype=object)
```

[92]:
```
sfs =␣
 ↪SequentialFeatureSelector(estimator=Ridge(),n_features_to_select=10,direction='backward')
sfs.fit(X_train,y_train)
```

[92]:
```
SequentialFeatureSelector(direction='backward', estimator=Ridge(),
                          n_features_to_select=10)
```

[93]:
```
selected_features = sfs.get_feature_names_out()
selected_features
```

[93]:
```
array(['Profit', 'Operating Expenses',
       'Product Name_Global High-Back Leather Tilter, Burgundy',
       "Product Name_Global Wood Trimmed Manager's Task Chair, Khaki",
       'Product Name_KI Adjustable-Height Table',
       'Product Name_Logitech 910-002974 M325 Wireless Mouse for Web Scrolling',
       'Product Name_Situations Contoured Folding Chairs, 4/Set',
       'Product Name_Staples', 'Product Name_Storex Dura Pro Binders',
       'Product Name_infrequent_sklearn'], dtype=object)
```

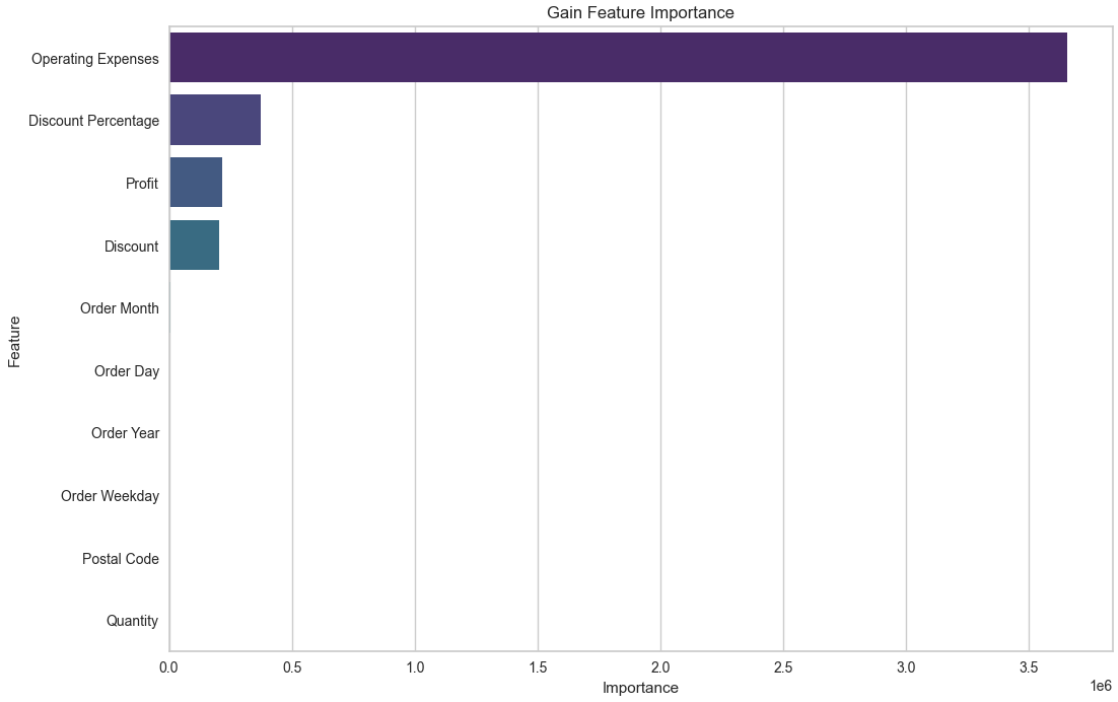
[94]:
```
xgb = XGBRegressor()
xgb.fit(X_train,y_train)
```

[94]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=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,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, …)
```

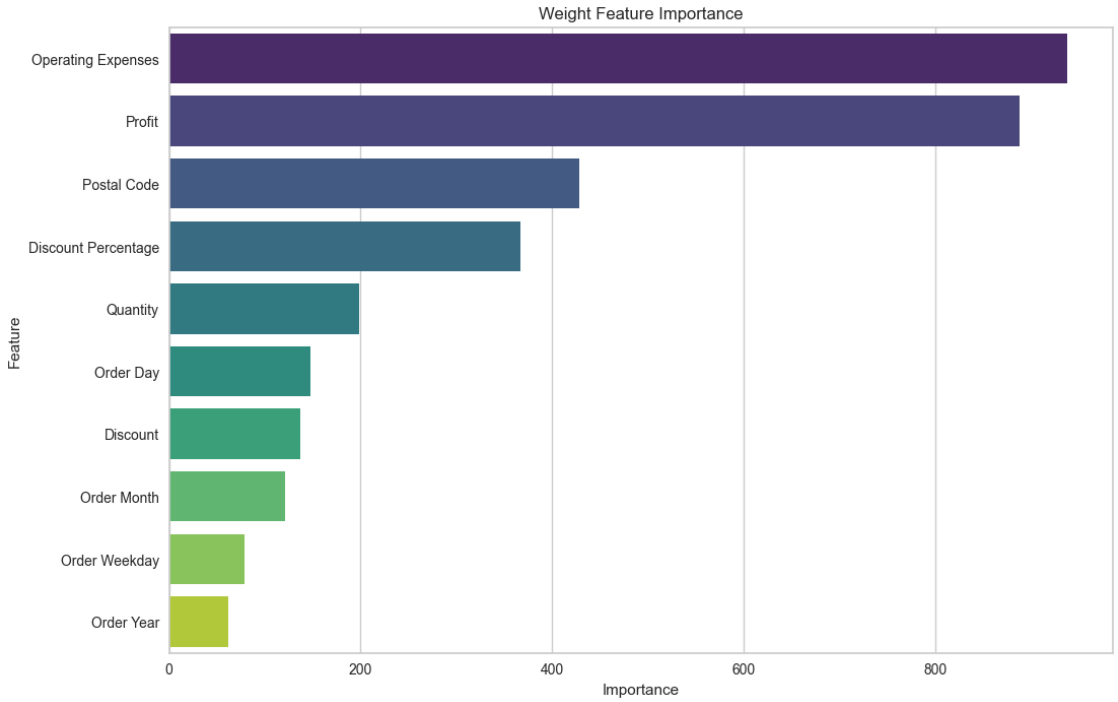
[95]:
```
def plot_feature_importances(feat_imp_type: str):
    feat_imps = xgb.get_booster().get_score(importance_type=feat_imp_type)
    keys = list(feat_imps.keys())[:10]
    values = list(feat_imps.values())[:10]
    feat_imps_df = pd.DataFrame(data=values, index=keys,␣
 ↪columns=["Importance"]).sort_values(by="Importance", ascending=False).
 ↪reset_index()
```

```
    feat_imps_df.rename({'index': 'Feature'},axis=1,inplace=True)
    plt.figure(figsize=(12,8))
    fig = sns.
↪barplot(x='Importance',y='Feature',data=feat_imps_df,orient='horizontal',palette='viridis')
    plt.title(f"{feat_imp_type.title()} Feature Importance")
    plt.show(fig)
    plt.close('all')
    del fig
    gc.collect();
```

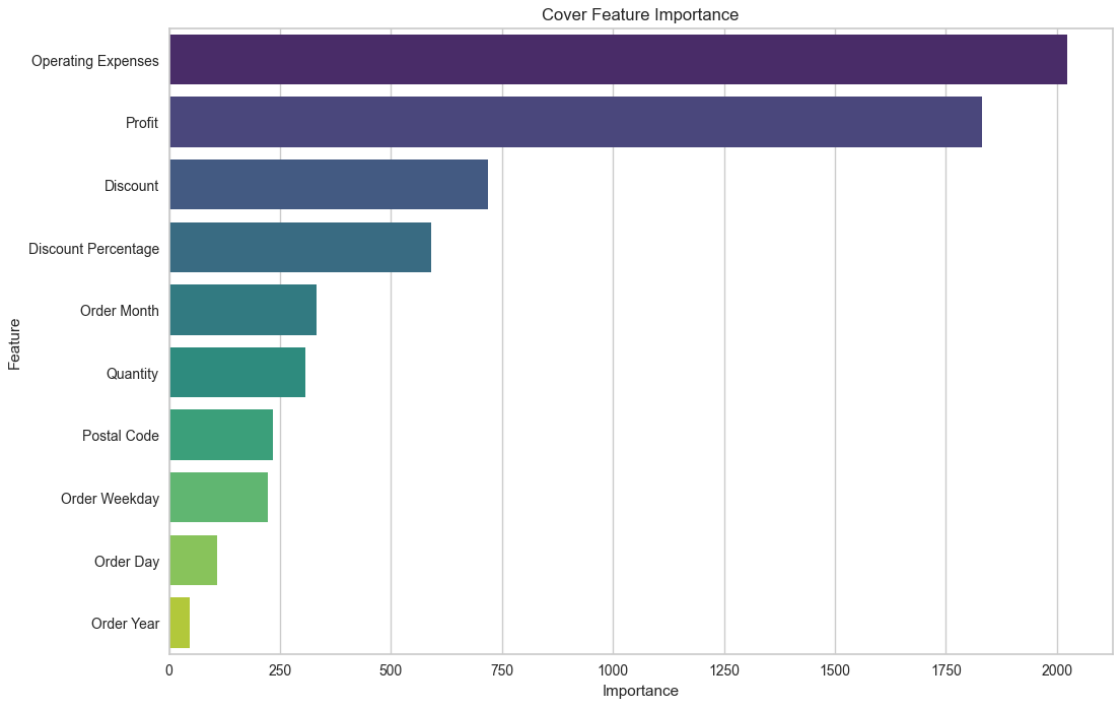
[96]:
```
plot_feature_importances('gain')
```



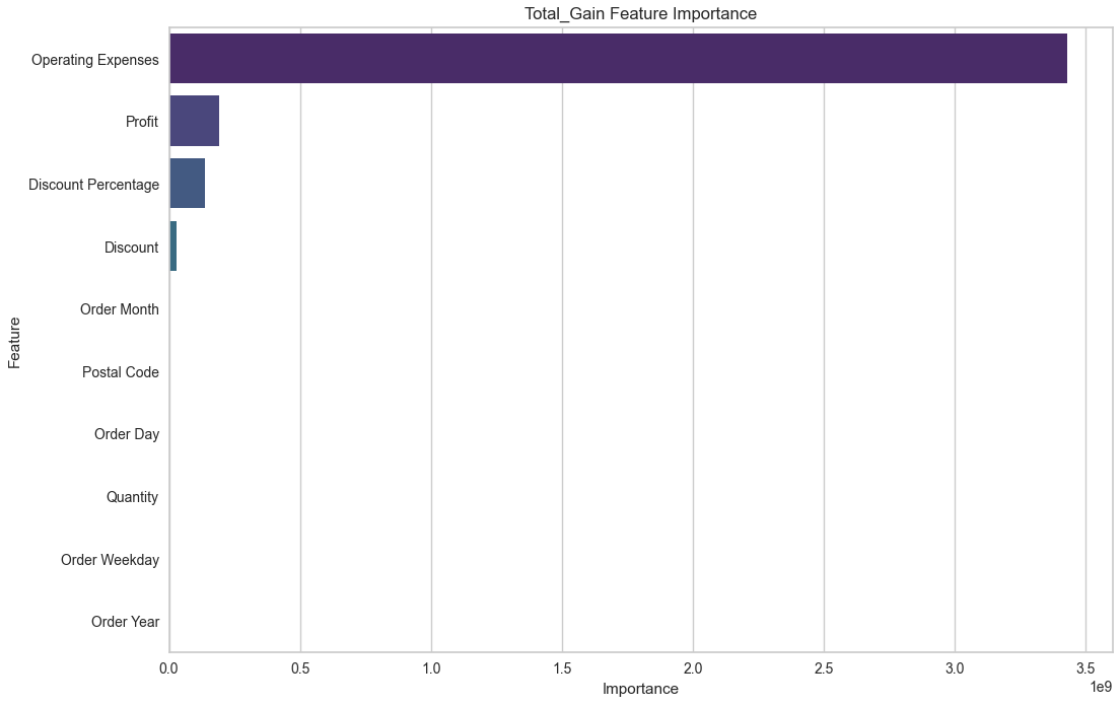
[97]:
```
plot_feature_importances('weight')
```



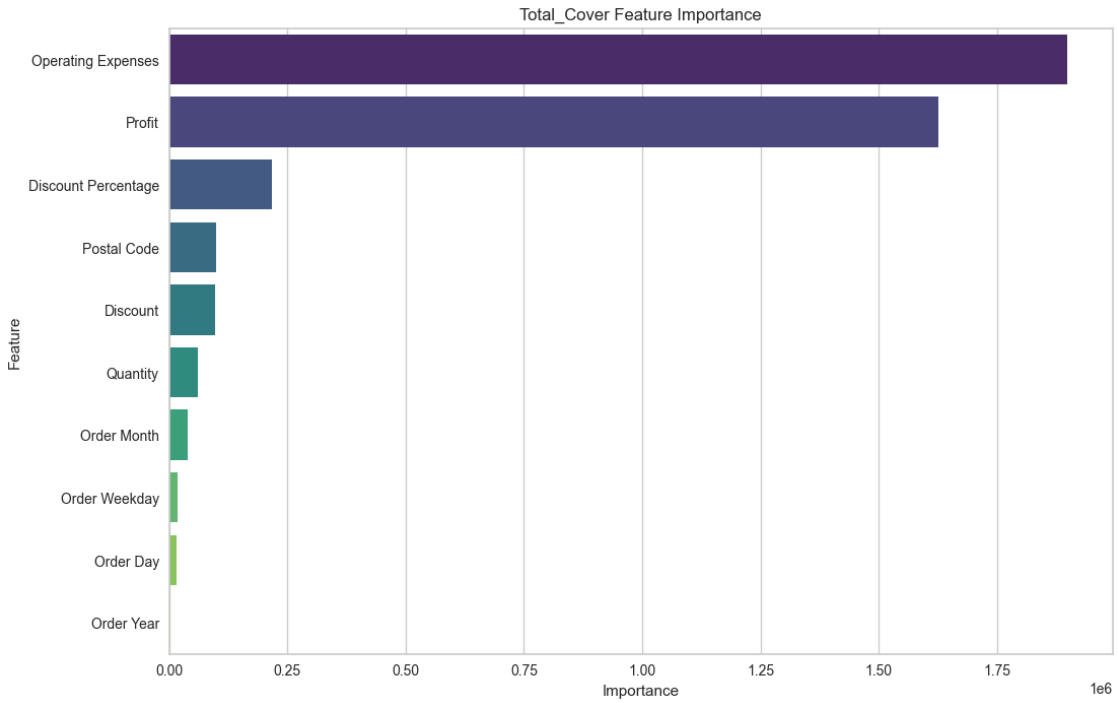
[98]:
```
plot_feature_importances('cover')
```



[99]:
```
plot_feature_importances('total_gain')
```



[100]:
```
plot_feature_importances('total_cover')
```



[101]:
```
final_selected_features = ['Profit',
                           'Discount Percentage',
                           'Postal Code',
                           'Discount',
                           'Order Day',
                           'Order Month',
                           'Quantity',
                           'Operating Expenses',
                           'Ship Day',
                           'Order Weekday']

final_X_train = X_train[final_selected_features]
final_X_test = X_test[final_selected_features]
```

[102]:
```
gc.collect()
```

[102]:
```
0
```

[103]:
```
final_X_train.head()
```

[103]:

	Profit	Discount Percentage	Postal Code	Discount	Order Day	Order Month	Quantity	Operating Expenses	Ship Day	Order Weekday
5579	-15.2450	0.081994	90004.0	0.2	10.0	9.0	5.0	259.1650	15.0	2.0
2728	94.4937	0.063496	43615.0	0.4	23.0	12.0	7.0	535.4643	25.0	1.0
477	7.1820	0.208855	90008.0	0.2	13.0	7.0	6.0	88.5780	20.0	5.0
1329	16.3020	0.000000	94109.0	0.0	24.0	5.0	3.0	20.7480	28.0	4.0
4544	16.7040	0.134698	60610.0	0.2	12.0	7.0	2.0	131.7760	17.0	3.0

[104]:
```
final_X_test.head()
```

[104]:

	Profit	Discount Percentage	Postal Code	Discount	Order Day	Order Month	Quantity	Operating Expenses	Ship Day	Order Weekday
1558	44.7096	0.000000	98103.0	0.0	16.0	3.0	2.0	127.2504	18.0	4.0
3059	-3.3506	0.820749	94110.0	0.2	20.0					
8234	0.4160	6.009615	32216.0	0.2	2.0					
5357	1.6360	0.000000	94122.0	0.0	5.0					
35	123.4737	0.018223	75080.0	0.2	9.0					

```
3059          10.0       2.0             27.7186      25.0            0.0
8234           5.0       2.0              2.9120       7.0            4.0
5357           8.0       1.0             14.7240      11.0            4.0
35            12.0       7.0            974.0703      11.0            0.0
```

[105]:
```
transformer = ColumnTransformer(transformers=[
    ('log_transform',FunctionTransformer(np.log1p),['Quantity','Profit']),
    ('sqrt_transform',FunctionTransformer(np.sqrt),['Discount']),
    ('power_transform',PowerTransformer(),['Discount Percentage','Operating␣
 ↪Expenses'])
],remainder='passthrough')
transformer
```

[105]:
```
ColumnTransformer(remainder='passthrough',
                  transformers=[('log_transform',
                                 FunctionTransformer(func=<ufunc 'log1p'>),
                                 ['Quantity', 'Profit']),
                                ('sqrt_transform',
                                 FunctionTransformer(func=<ufunc 'sqrt'>),
                                 ['Discount']),
                                ('power_transform', PowerTransformer(),
                                 ['Discount Percentage',
                                  'Operating Expenses'])])
```

[106]:
```
final_X_train = transformer.fit_transform(final_X_train)
final_X_train = pd.DataFrame(final_X_train,columns=final_selected_features)
final_X_test = transformer.transform(final_X_test)
final_X_test = pd.DataFrame(final_X_test,columns=final_selected_features)
final_X_train.head()
```

[106]:
```
     Profit  Discount Percentage  Postal Code  Discount  Order Day  \
0  1.791759                  NaN     0.447214 -0.324229   1.009995
1  2.079442             4.559060     0.632456 -0.412458   1.378259
2  1.945910             2.101937     0.447214  0.171010   0.414783
3  1.386294             2.850822     0.000000 -0.755604  -0.483757
4  1.098612             2.873791     0.447214 -0.097214   0.642144

   Order Month  Quantity  Operating Expenses  Ship Day  Order Weekday
0      90004.0      10.0                 9.0      15.0            2.0
1      43615.0      23.0                12.0      25.0            1.0
2      90008.0      13.0                 7.0      20.0            5.0
3      94109.0      24.0                 5.0      28.0            4.0
4      60610.0      12.0                 7.0      17.0            3.0
```

[107]:
```
final_X_test.head()
```

[107]:

	Profit	Discount Percentage	Postal Code	Discount	Order Day	\
0	1.098612	3.822308	0.000000	-0.755604	0.622479	
1	1.098612	NaN	0.447214	1.246647	-0.296776	
2	1.098612	0.347836	0.447214	1.963591	-1.764067	
3	0.693147	0.969263	0.000000	-0.755604	-0.708771	
4	2.079442	4.824094	0.447214	-0.650184	1.662262	

	Order Month	Quantity	Operating Expenses	Ship Day	Order Weekday
0	98103.0	16.0	3.0	18.0	4.0
1	94110.0	20.0	10.0	25.0	0.0
2	32216.0	2.0	5.0	7.0	4.0
3	94122.0	5.0	8.0	11.0	4.0
4	75080.0	9.0	12.0	11.0	0.0

0.5.6 Missing Value Imputation

[108]:
```
final_X_train.isna().sum()
```

[108]:
```
Profit                   0
Discount Percentage    1283
Postal Code               0
Discount                  0
Order Day                 0
Order Month               0
Quantity                  0
Operating Expenses        0
Ship Day                  0
Order Weekday             0
dtype: int64
```

[109]:
```
final_X_test.isnull().sum()
```

[109]:
```
Profit                   0
Discount Percentage    551
Postal Code              0
Discount                 0
Order Day                0
Order Month              0
Quantity                 0
Operating Expenses       0
Ship Day                 0
Order Weekday            0
dtype: int64
```

[110]:
```
imputer = SimpleImputer(strategy='median')
final_X_train['Discount Percentage'] = imputer.
 ↪fit_transform(final_X_train[['Discount Percentage']])
```

```
final_X_test['Discount Percentage'] = imputer.
 ↪fit_transform(final_X_test[['Discount Percentage']])
```

[111]:
```
for col in final_X_train.columns:
    imputer = SimpleImputer(strategy='median')
    final_X_train[col] = imputer.fit_transform(final_X_train[[col]])
```

[112]:
```
final_X_train.isna().sum()
```

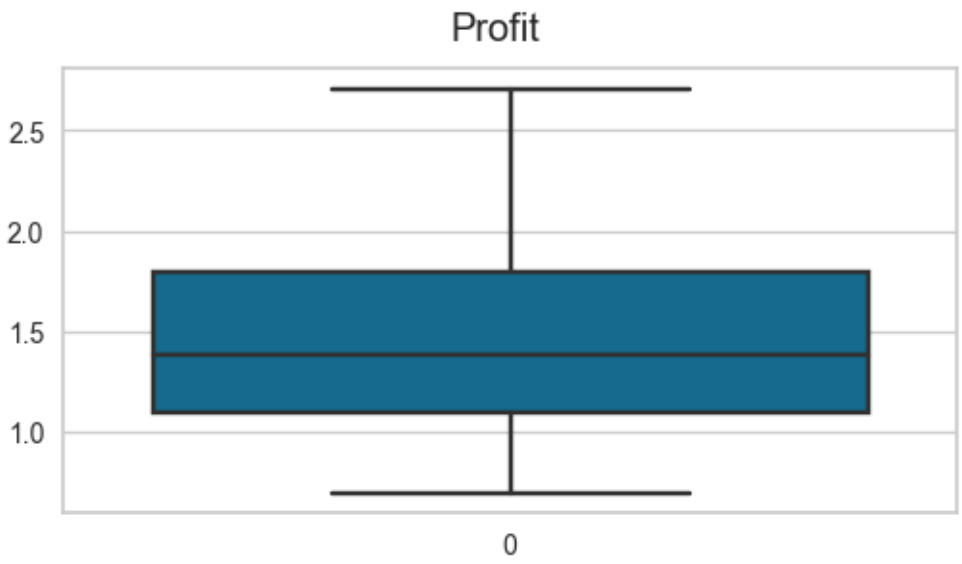
[112]:
```
Profit                0
Discount Percentage   0
Postal Code           0
Discount              0
Order Day             0
Order Month           0
Quantity              0
Operating Expenses    0
Ship Day              0
Order Weekday         0
dtype: int64
```

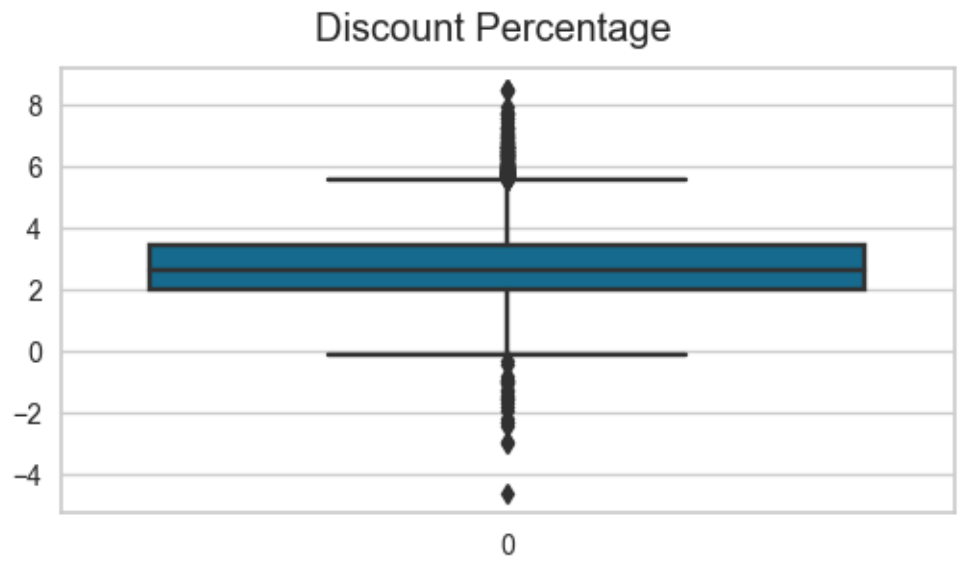
[113]:
```
final_X_test.isnull().sum()
```

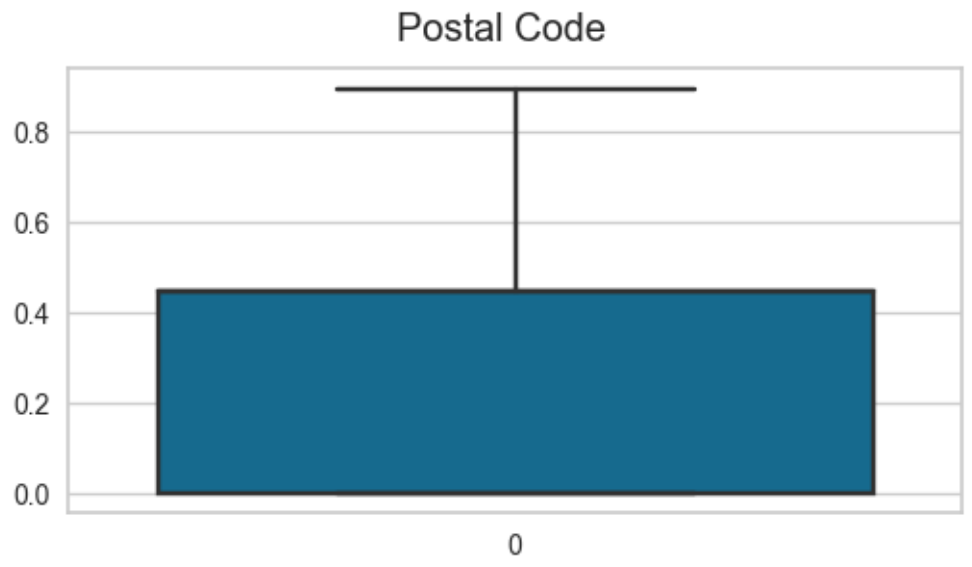
[113]:
```
Profit                0
Discount Percentage   0
Postal Code           0
Discount              0
Order Day             0
Order Month           0
Quantity              0
Operating Expenses    0
Ship Day              0
Order Weekday         0
dtype: int64
```

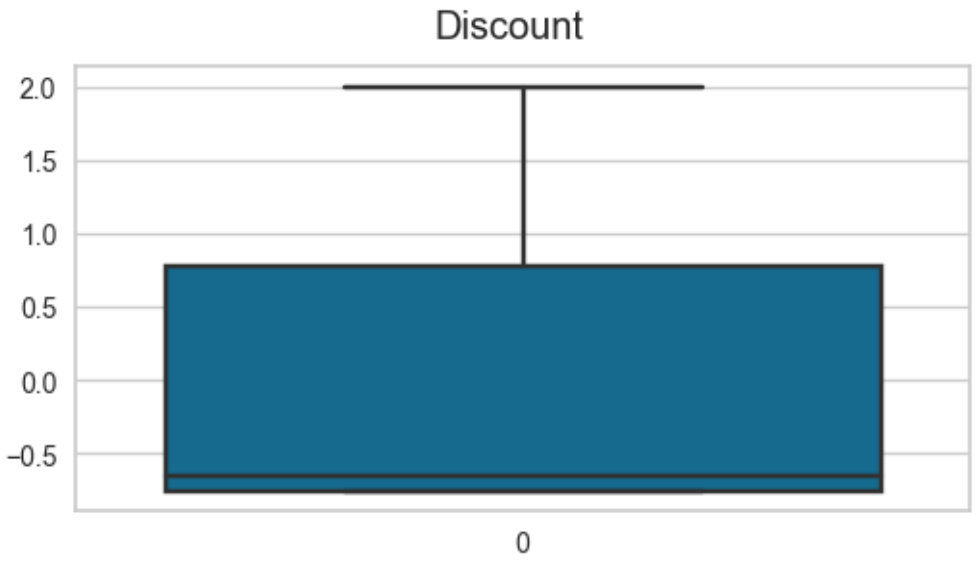
0.5.7 Outlier Treatment

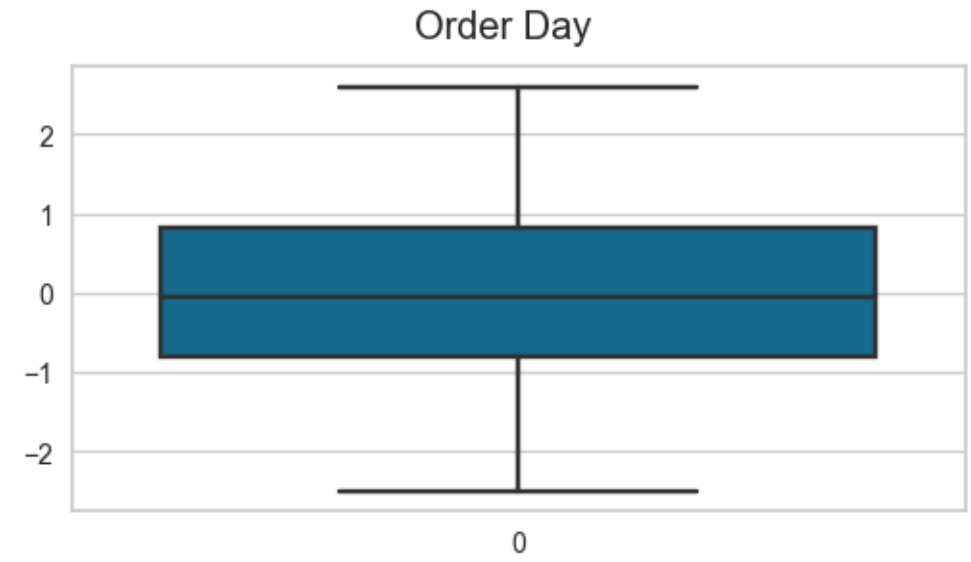
[114]:
```
for col in final_X_train.columns:
    plt.figure(figsize=(6,3))
    sns.boxplot(final_X_train[col])
    plt.suptitle(col)
    plt.show()
```

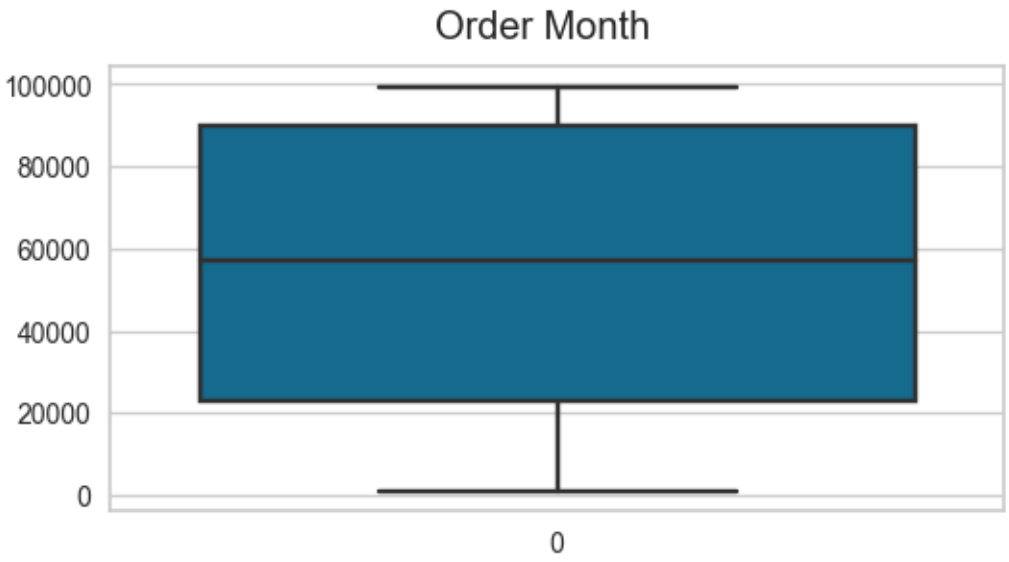

Profit
2.5
2.0
1.5
1.0
0

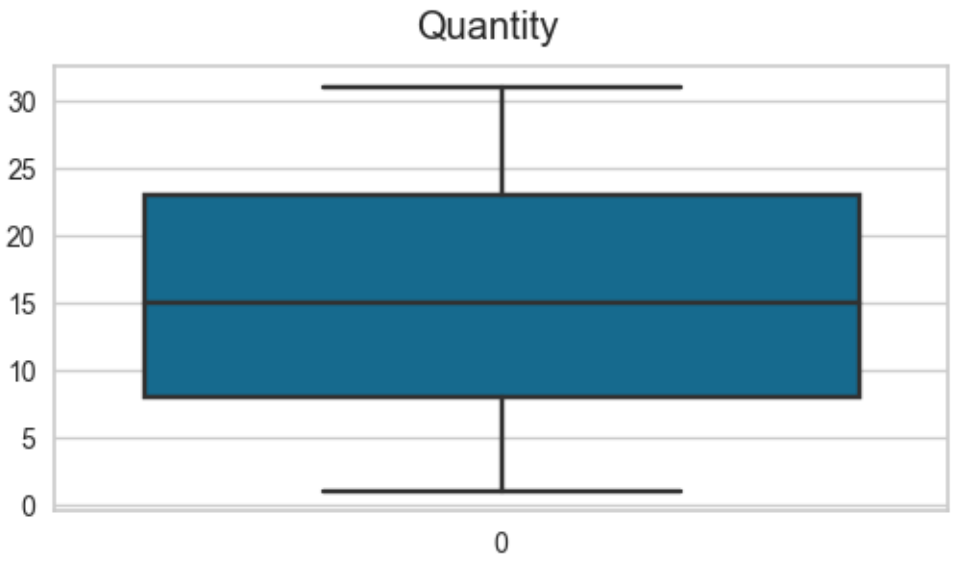

Discount Percentage
8
6
4
2
0
−2
−4
0

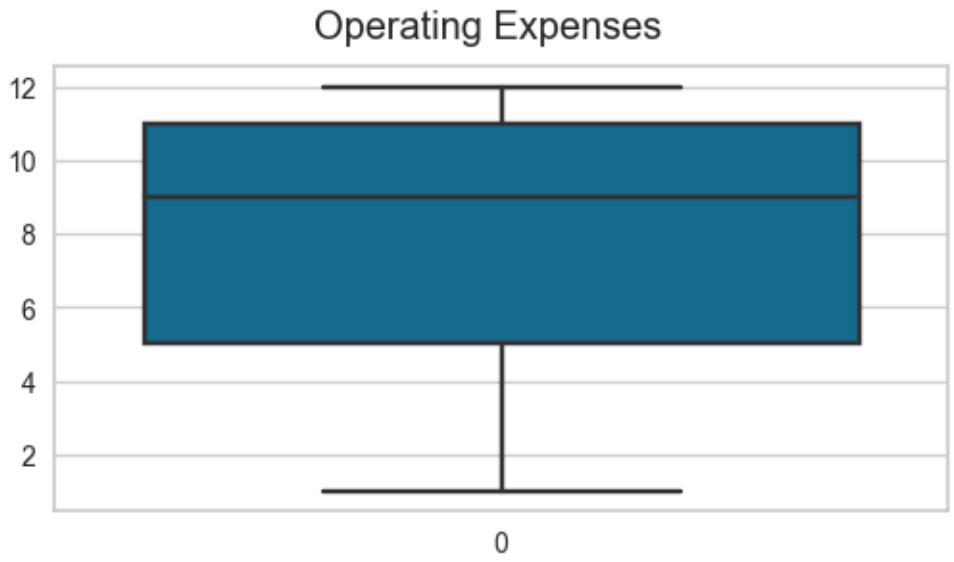

Postal Code
0.8
0.6
0.4
0.2
0.0
0

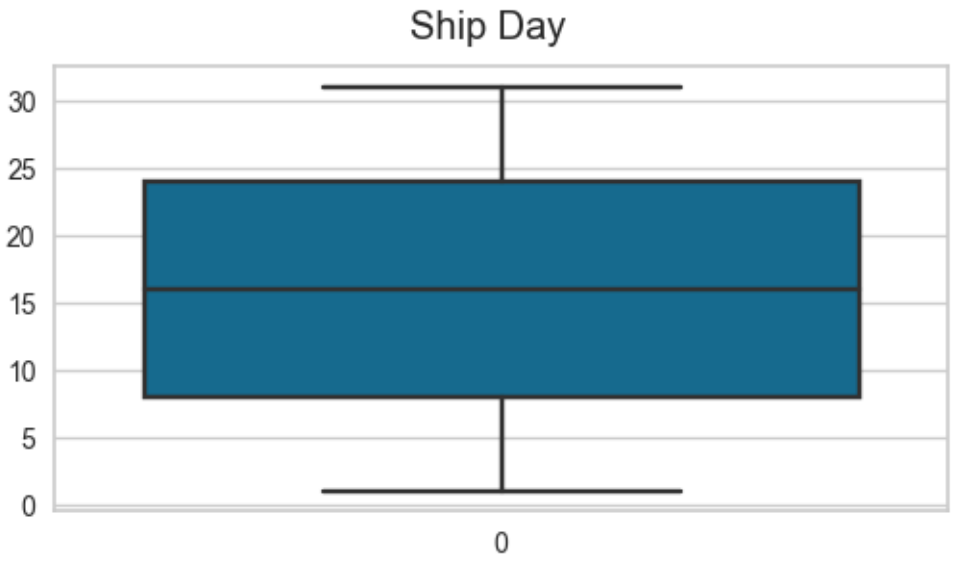

Discount
2.0
1.5
1.0
0.5
0.0
−0.5
0

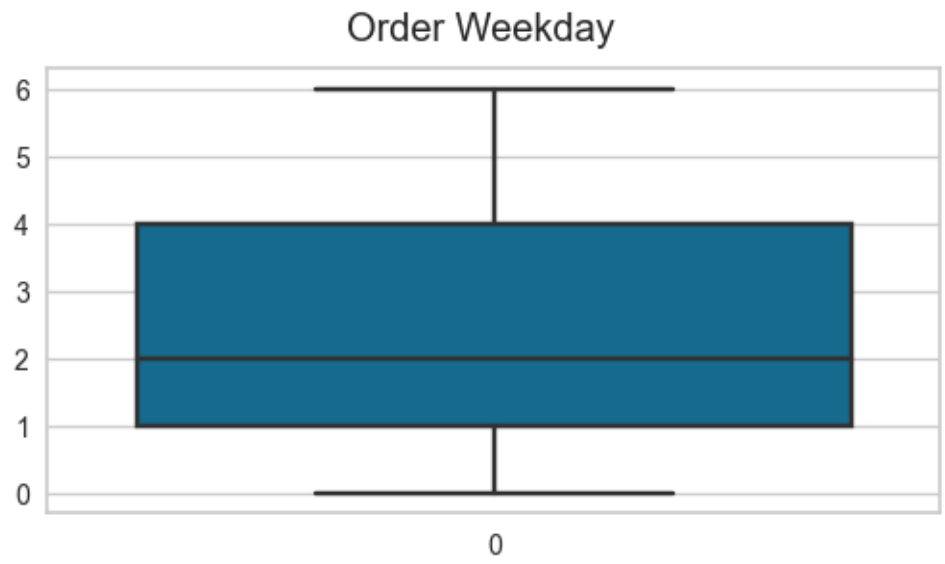

Order Day
2
1
0
−1
−2
0


Order Month
100000
80000
60000
40000
20000
0
0

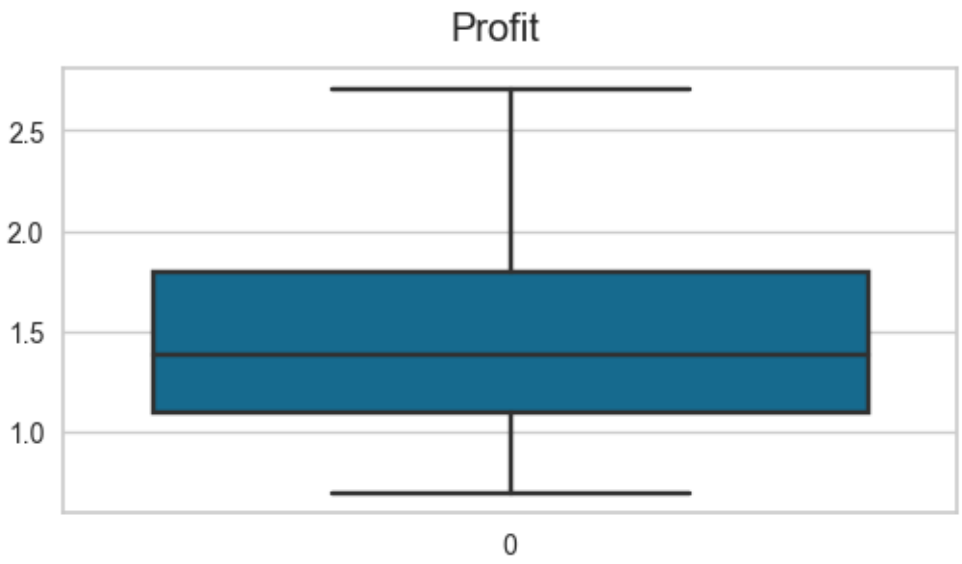

Quantity
30
25
20
15
10
5
0
0

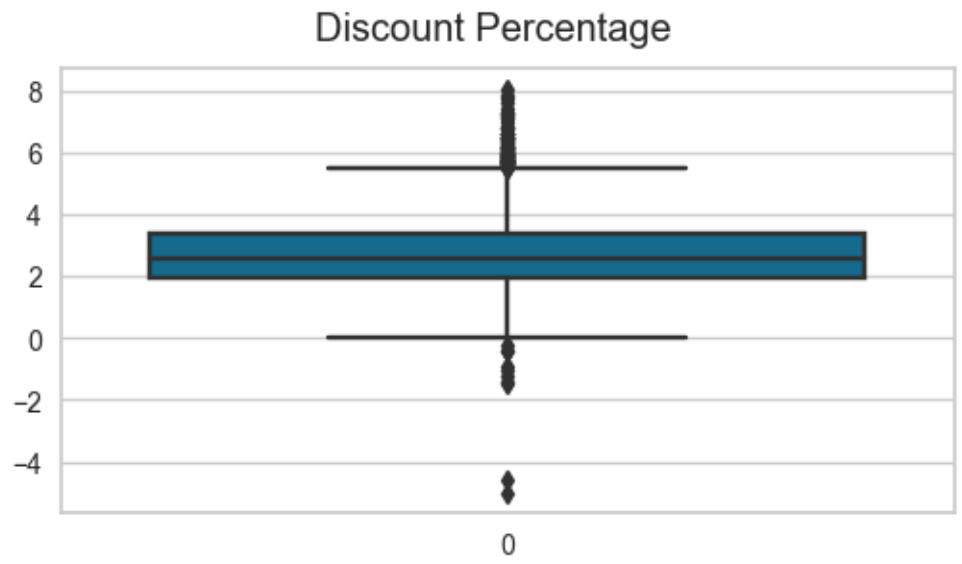

Operating Expenses
12
10
8
6
4
2
0

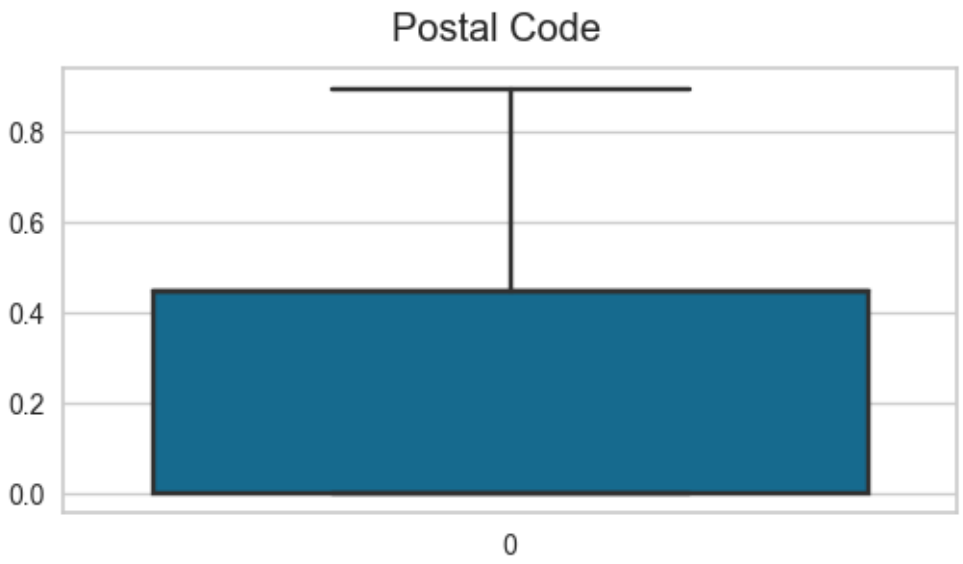


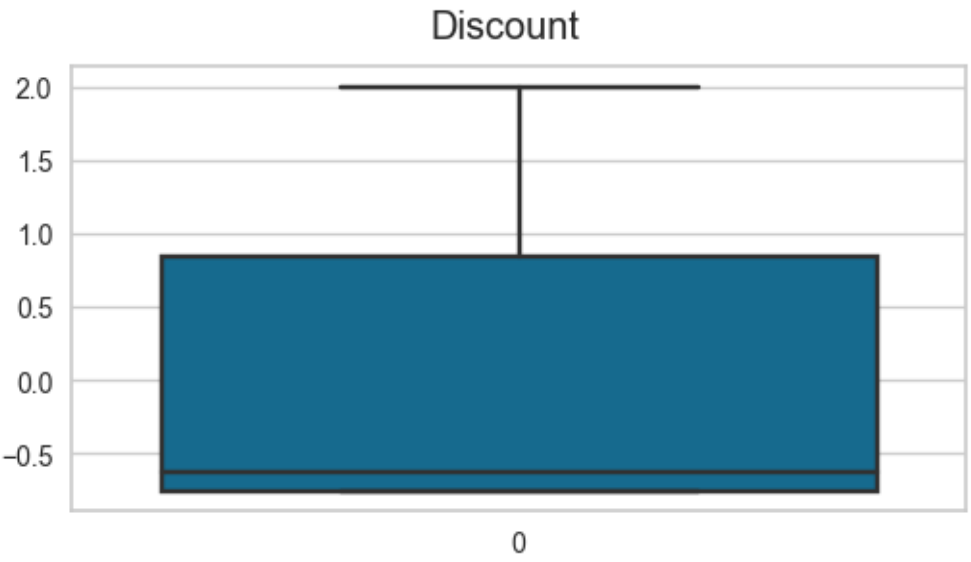


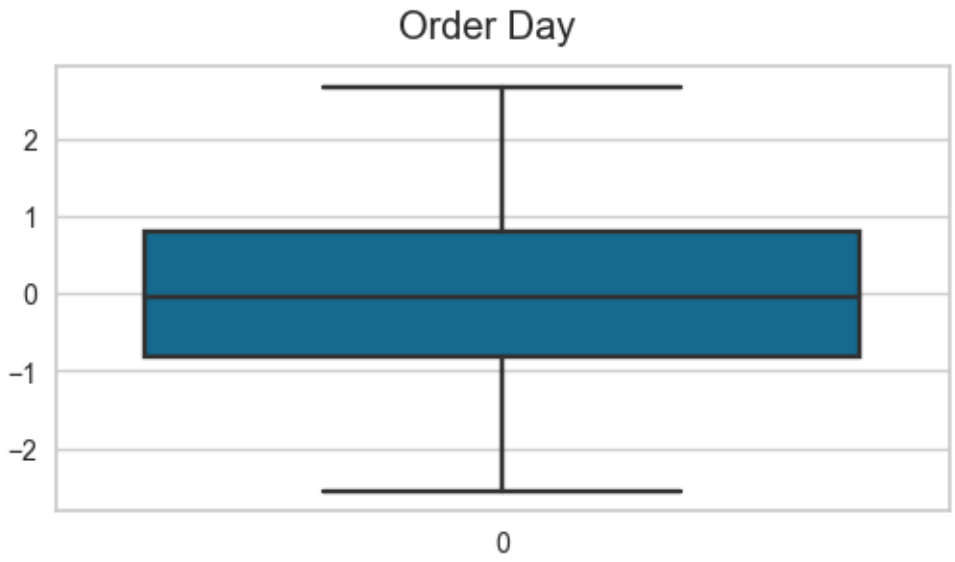
[115]:
```
for col in final_X_test.columns:
    plt.figure(figsize=(6,3))
    sns.boxplot(final_X_test[col])
    plt.suptitle(col)
    plt.show()
```

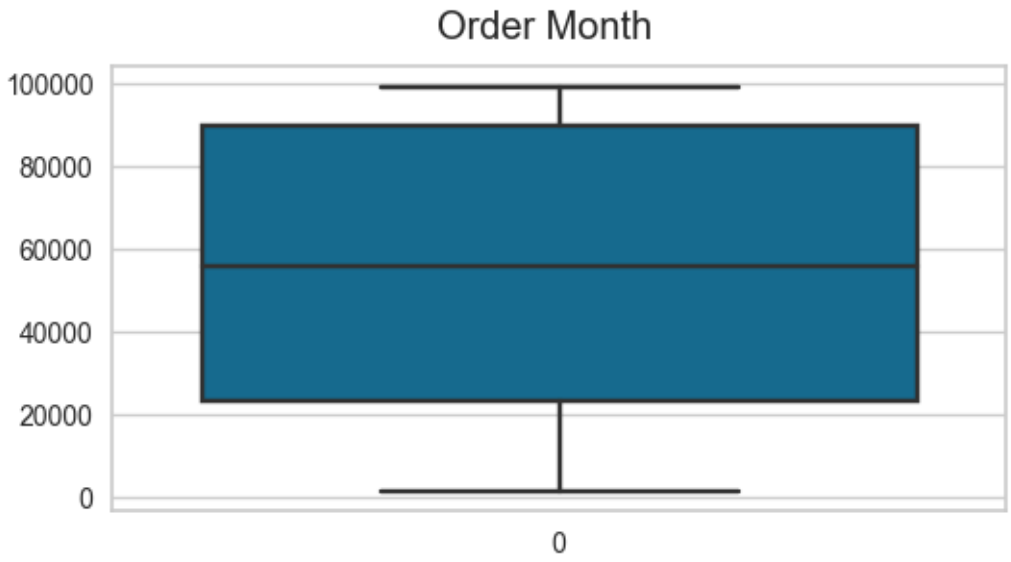

Profit
2.5
2.0
1.5
1.0
0

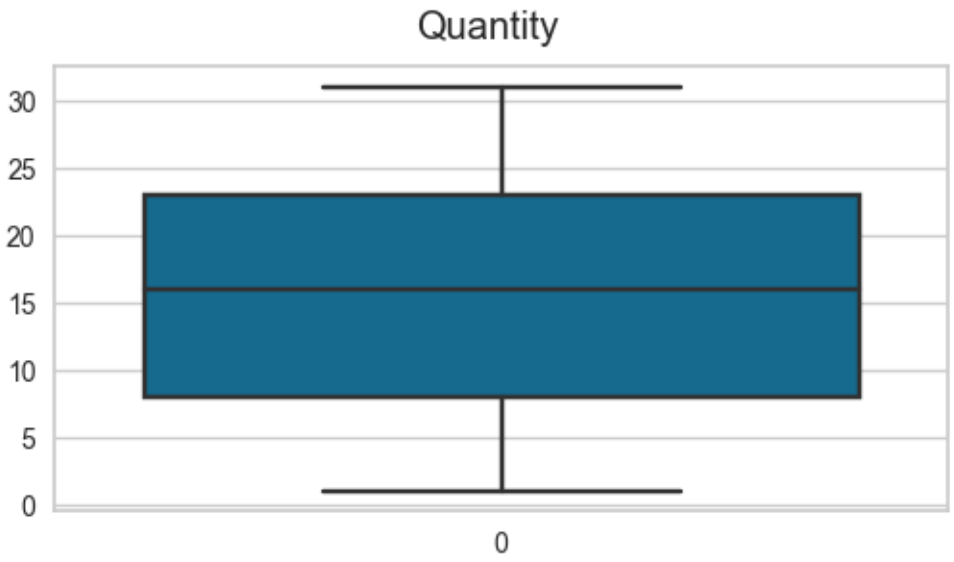

Discount Percentage
8
6
4
2
0
−2
−4
0

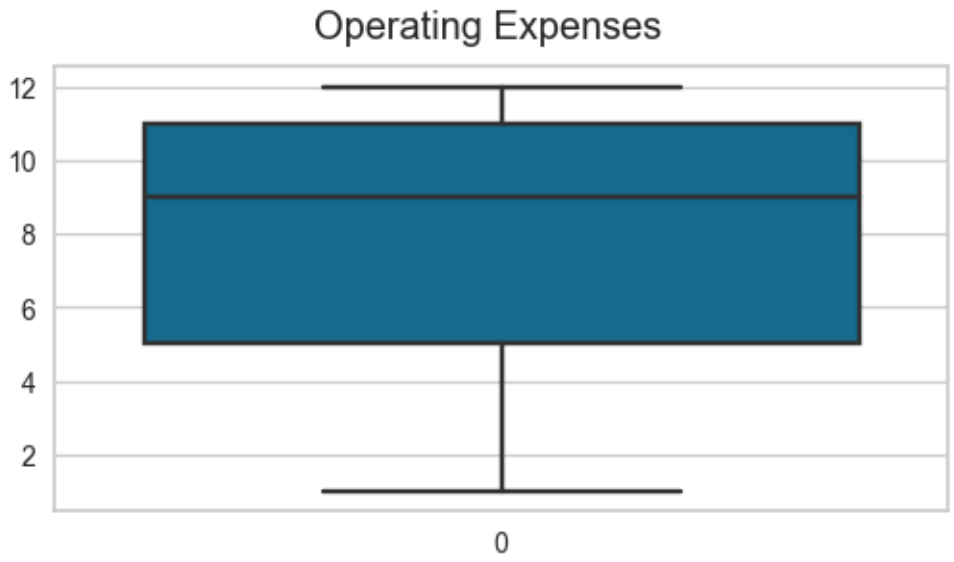

Postal Code
0.8
0.6
0.4
0.2
0.0
0

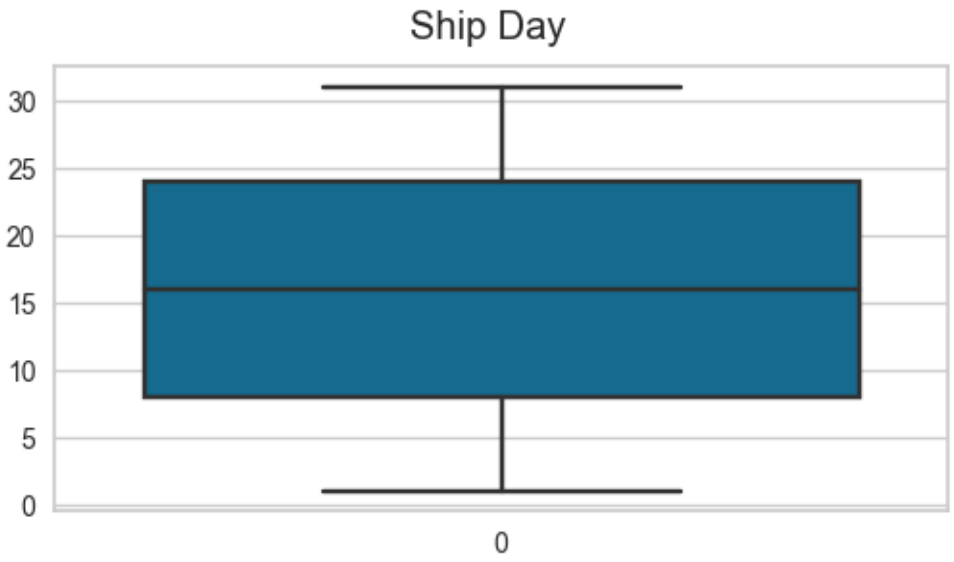

Discount
2.0
1.5
1.0
0.5
0.0
−0.5
0

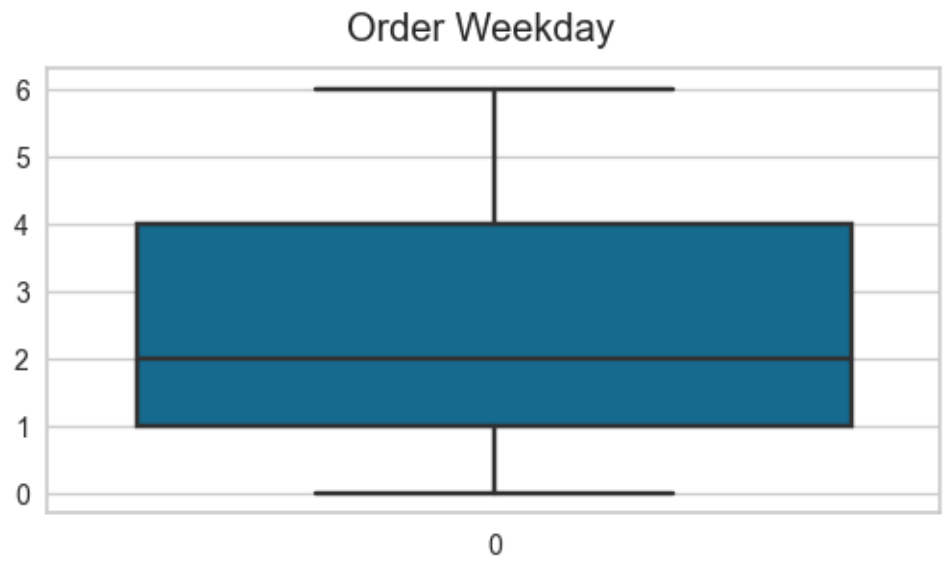

Order Day
2
1
0
−1
−2
0


Order Month
100000
80000
60000
40000
20000
0
0


Quantity
30
25
20
15
10
5
0
0

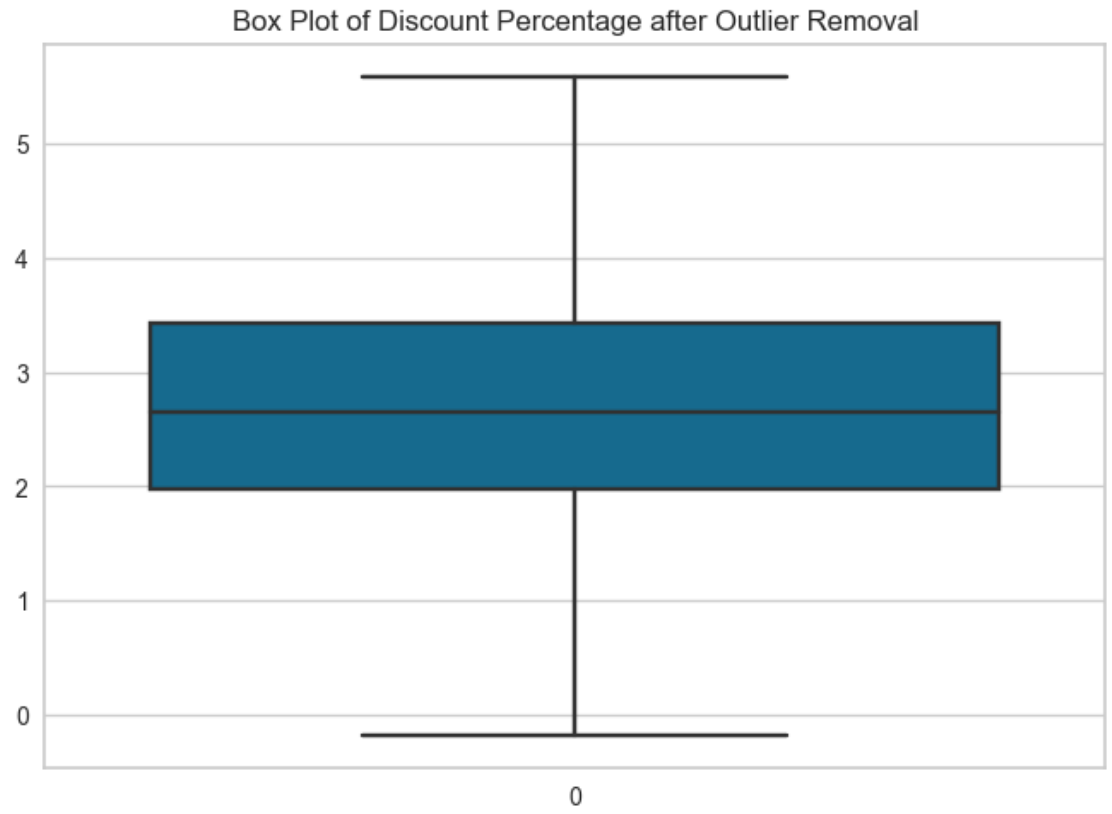

Operating Expenses
12
10
8
6
4
2
0





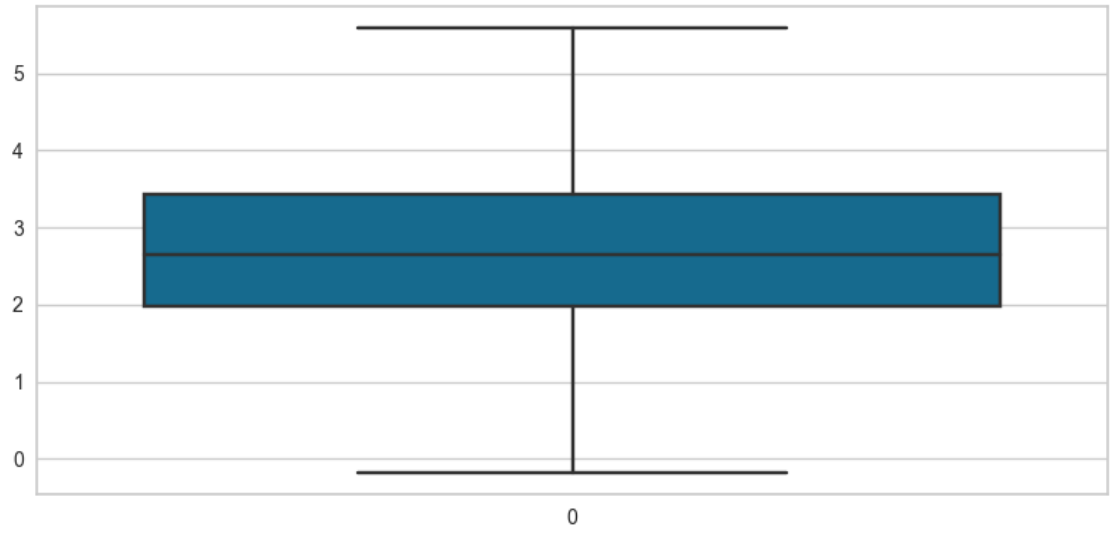
[116]:
```
def impute_outliers(data,col):
    winsorizer = Winsorizer(capping_method='iqr',fold=1.5,tail='both')
    fig = sns.boxplot(winsorizer.fit_transform(data[[col]]).values)
    plt.title(f"Box Plot of {col} after Outlier Removal")
    plt.show(fig)
    plt.close('all')
    del fig
    gc.collect()
```

[117]:
```
impute_outliers(final_X_train,'Discount Percentage')
```

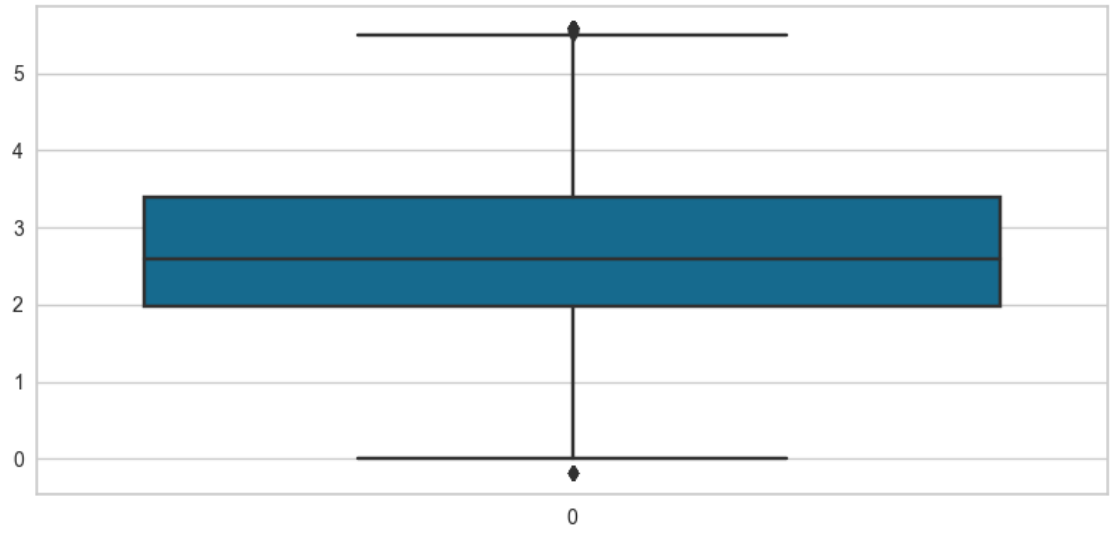


[118]:
```
winsorizer = Winsorizer(capping_method='iqr',fold=1.5,tail='both')
final_X_train['Discount Percentage'] = winsorizer.
 ↪fit_transform(final_X_train[['Discount Percentage']])
final_X_test['Discount Percentage'] = winsorizer.
 ↪transform(final_X_test[['Discount Percentage']])
```

[119]:
```
plt.figure(figsize=(8,4))
fig = sns.boxplot(final_X_train['Discount Percentage'])
plt.tight_layout()
plt.show(fig)
plt.close('all')
del fig
gc.collect();
```



[120]:
```
plt.figure(figsize=(8,4))
fig = sns.boxplot(final_X_test['Discount Percentage'])
plt.tight_layout()
plt.show(fig)
plt.close('all')
del fig
gc.collect();
```



0.5.8 Feature Scaling

[121]:
```
scaler = StandardScaler()
features = final_X_train.columns
final_X_train = scaler.fit_transform(final_X_train)
final_X_train = pd.DataFrame(final_X_train,columns=features)
final_X_test = scaler.transform(final_X_test)
final_X_test = pd.DataFrame(final_X_test,columns=features)
final_X_train.head()
```

[121]:

	Profit	Discount Percentage	Postal Code	Discount	Order Day	\
0	0.738777	-0.085201	0.620435	-0.324229	1.009995	
1	1.398690	1.488356	1.271876	-0.412458	1.378259	
2	1.092383	-0.540392	0.620435	0.171010	0.414783	
3	-0.191318	0.077933	-0.952285	-0.755604	-0.483757	
4	-0.851232	0.096897	0.620435	-0.097214	0.642144	

	Order Month	Quantity	Operating Expenses	Ship Day	Order Weekday
0	1.083957	-0.650256	0.361982	-0.093987	-0.231513
1	-0.361693	0.840780	1.277878	1.043587	-0.823287
2	1.084082	-0.306171	-0.248614	0.474800	1.543812
3	1.211884	0.955475	-0.859211	1.384859	0.952037
4	0.167933	-0.420866	-0.248614	0.133528	0.360262

[122]:
```
final_X_test.head()
```

[122]:

	Profit	Discount Percentage	Postal Code	Discount	Order Day	\
0	-0.851232	0.880050	-0.952285	-0.755604	0.622479	
1	-0.851232	-0.130519	0.620435	1.246647	-0.296776	
2	-0.851232	-1.988682	0.620435	1.963591	-1.764067	
3	-1.781327	-1.475595	-0.952285	-0.755604	-0.708771	
4	1.398690	1.707184	0.620435	-0.650184	1.662262	

	Order Month	Quantity	Operating Expenses	Ship Day	Order Weekday
0	1.336351	0.037914	-1.469808	0.247285	0.952037
1	1.211915	0.496695	0.667281	1.043587	-1.415062
2	-0.716927	-1.567816	-0.859211	-1.004046	0.952037
3	1.212289	-1.223731	0.056684	-0.549016	0.952037
4	0.618871	-0.764951	1.277878	-0.549016	-1.415062

[123]:
```
imputer = SimpleImputer(strategy='mean')
y_train = imputer.fit_transform(y_train.values.reshape(-1,1))
y_test = imputer.transform(y_test.values.reshape(-1,1))
```

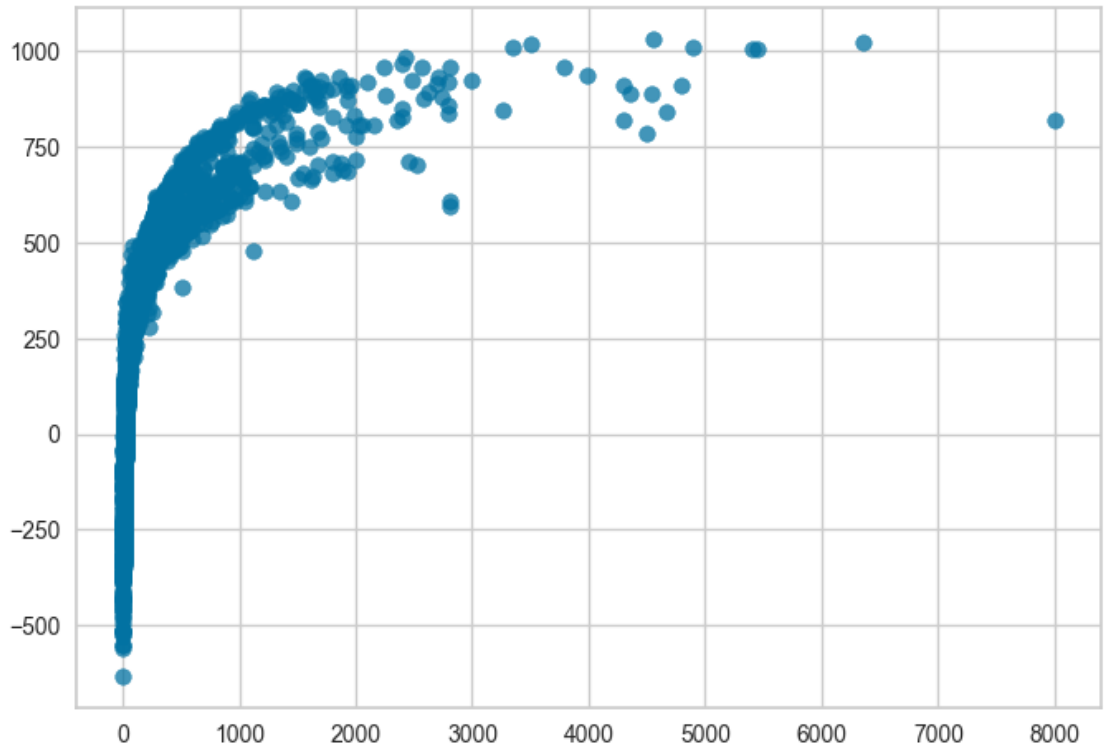
0.6 Model Training & Evaluation

[124]:
```
models = []
mae_scores = []
mse_scores = []
rmse_scores = []
mape_scores = []
r2_scores = []
training_times = []
```

[125]:
```
def train_and_evaluate_model(model):
    start_time = time.time()
    model.fit(final_X_train,y_train)
    duration = time.time() - start_time
    y_pred = model.predict(final_X_test)
    r2 = r2_score(y_test,y_pred)
    mae = mean_absolute_error(y_test,y_pred)
    mse = mean_squared_error(y_test,y_pred)
    rmse = np.sqrt(mean_squared_error(y_test,y_pred))
    mape = mean_absolute_percentage_error(y_test,y_pred)
    print("Mean Absolute Error:",mae)
    print("Mean Squared Error:",mse)
    print("Root Mean Squared Error:",rmse)
    print("Mean Absolute Percentage Error:",mape)
    print("R2 Score:",r2)
    print("Training Time:",duration)
    training_times.append(duration)
    mae_scores.append(mae)
    mse_scores.append(mse)
    rmse_scores.append(rmse)
    mape_scores.append(mape)
    r2_scores.append(r2)
    models.append(model)

    try:
        prediction_error = PredictionError(estimator=model)
        prediction_error.score(final_X_test,y_test)
        prediction_error.show()
        res_plot = ResidualsPlot(estimator=model)
        res_plot.score(final_X_test,y_test)
        res_plot.show()
        del r2, mae, mape, rmse, duration, prediction_error, res_plot
    except Exception as e:
        pass

    gc.collect()
```

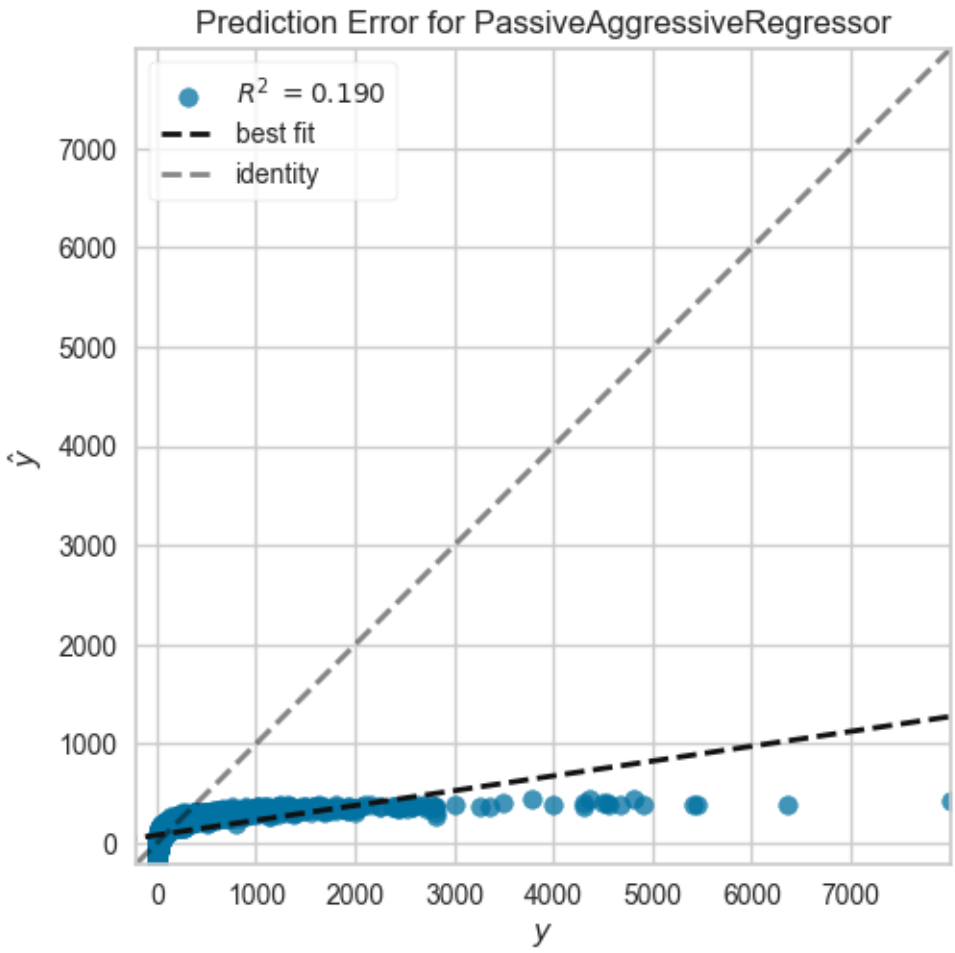
[126]:
```
train_and_evaluate_model(LinearRegression())
```

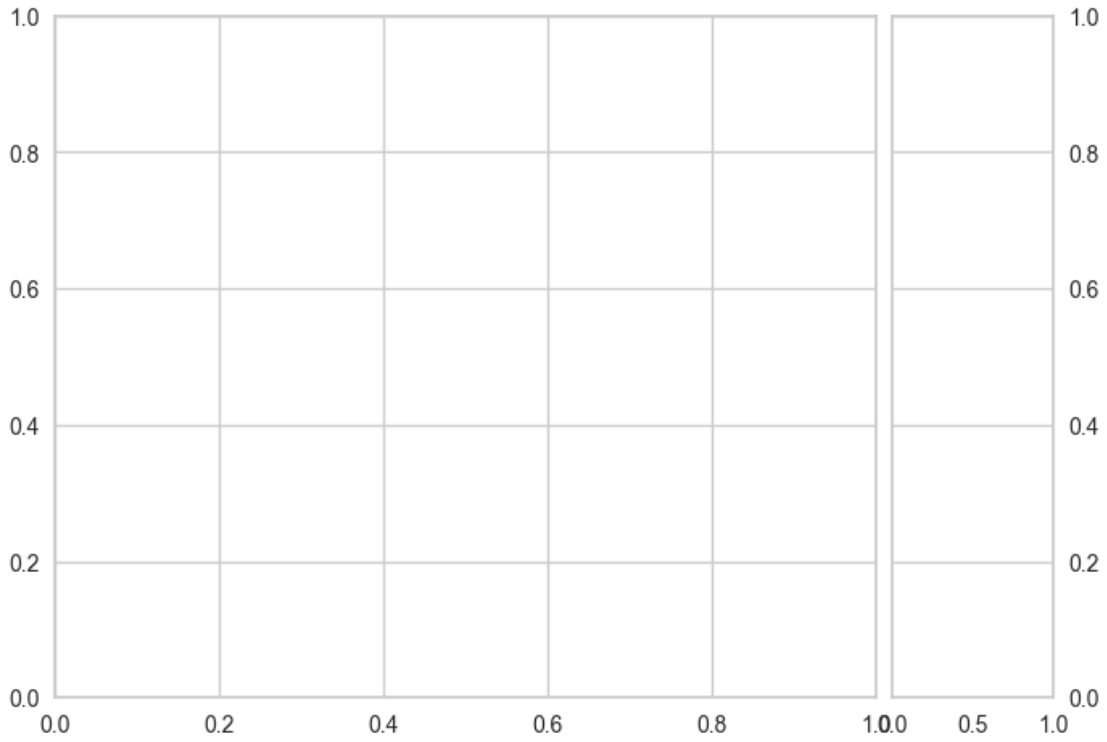
```
Mean Absolute Error: 213.42072661640157
Mean Squared Error: 167198.14332373216
Root Mean Squared Error: 408.8986956737966
Mean Absolute Percentage Error: 11.133867517277029
R2 Score: 0.37788964949952375
Training Time: 0.18094563484191895
```



[127]:
```
train_and_evaluate_model(PassiveAggressiveRegressor())
```

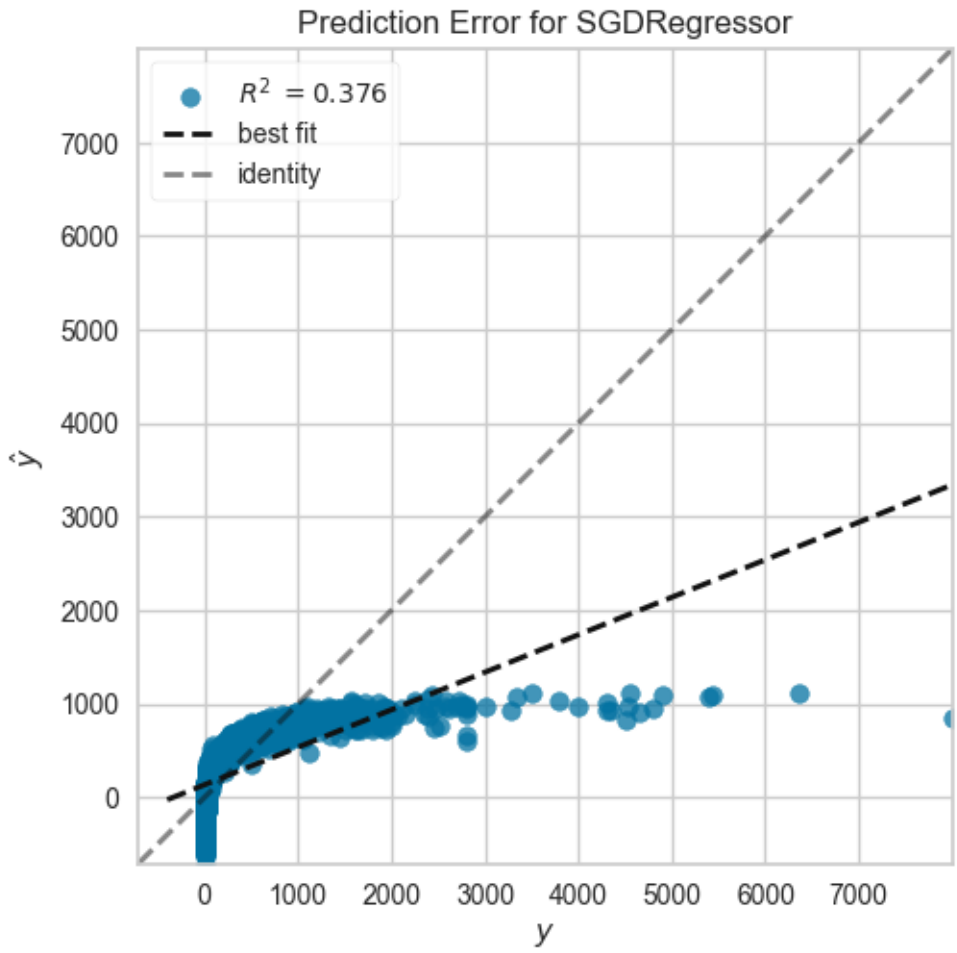
```
Mean Absolute Error: 155.77207693600002
Mean Squared Error: 217567.34541287803
Root Mean Squared Error: 466.44114892757693
Mean Absolute Percentage Error: 3.579509948557775
R2 Score: 0.19047607334852512
Training Time: 0.008995294570922852
```

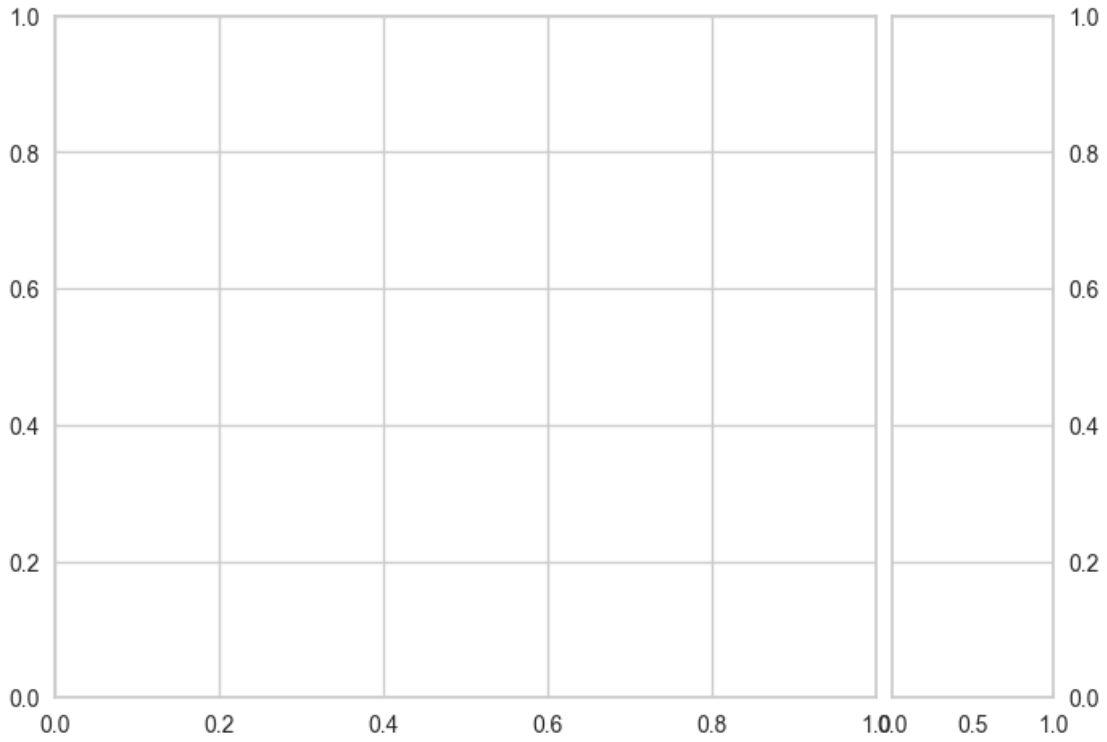

Prediction Error for PassiveAggressiveRegressor
R^2 = 0.190
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



[128]:
```
train_and_evaluate_model(SGDRegressor())
```

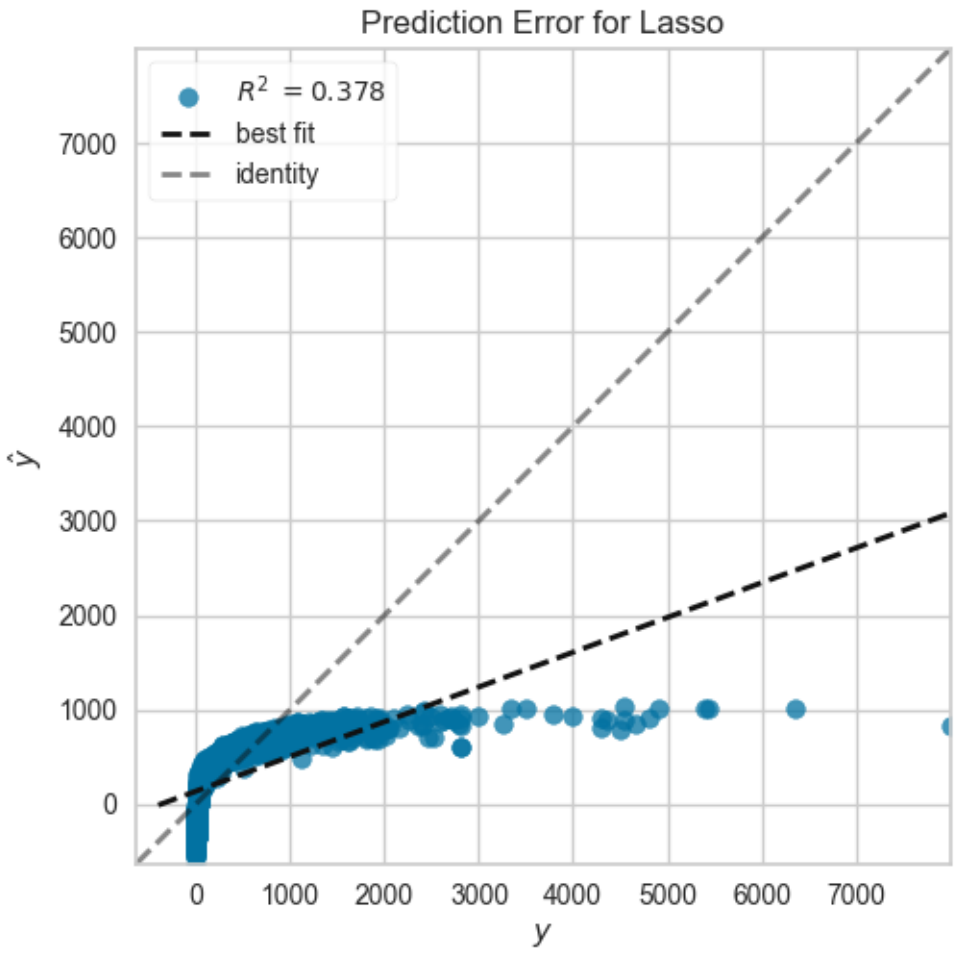
```
Mean Absolute Error: 226.2981992287408
Mean Squared Error: 167638.0696884139
Root Mean Squared Error: 409.4362828187237
Mean Absolute Percentage Error: 12.399212391767085
R2 Score: 0.3762527728005005
Training Time: 0.011340618133544922
```

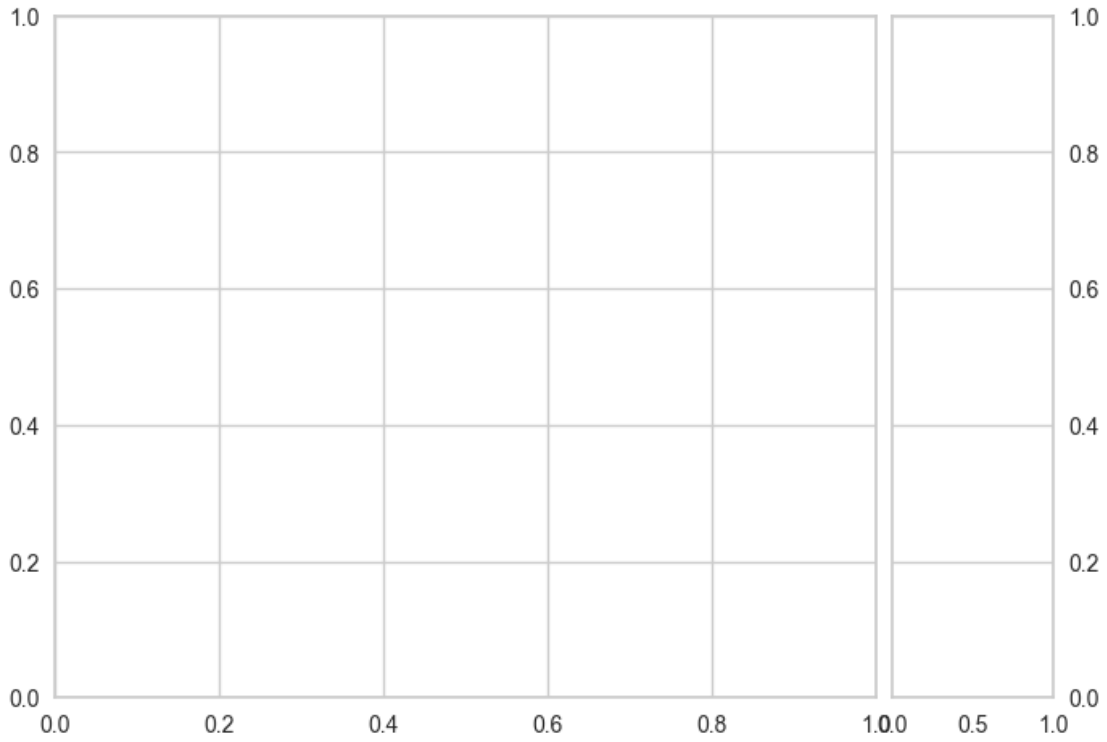

Prediction Error for SGDRegressor
R^2 = 0.376
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



[129]:
```
train_and_evaluate_model(Lasso(alpha=0.01))
```

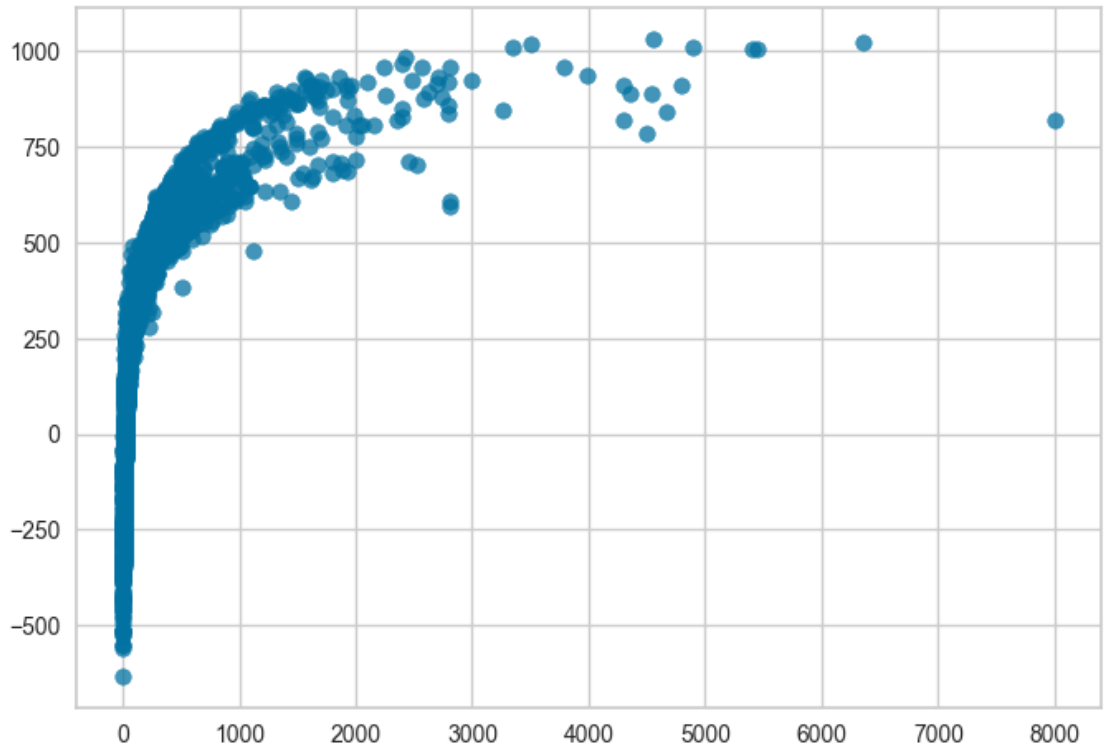
```
Mean Absolute Error: 213.41409828516774
Mean Squared Error: 167196.71701420474
Root Mean Squared Error: 408.896951583409
Mean Absolute Percentage Error: 11.132392060645758
R2 Score: 0.3778949565077381
Training Time: 0.012717962265014648
```


Prediction Error for Lasso
$R^2 = 0.378$
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



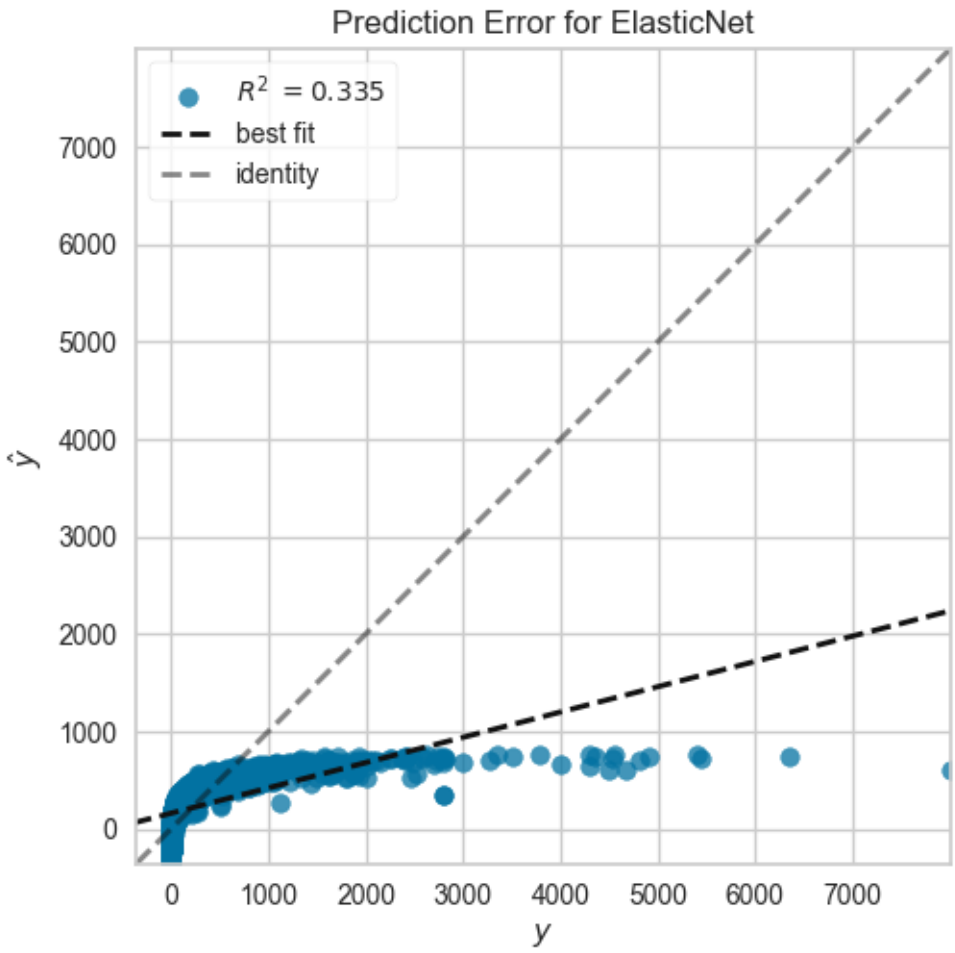
[130]:
```
train_and_evaluate_model(Ridge(alpha=0.001))
```

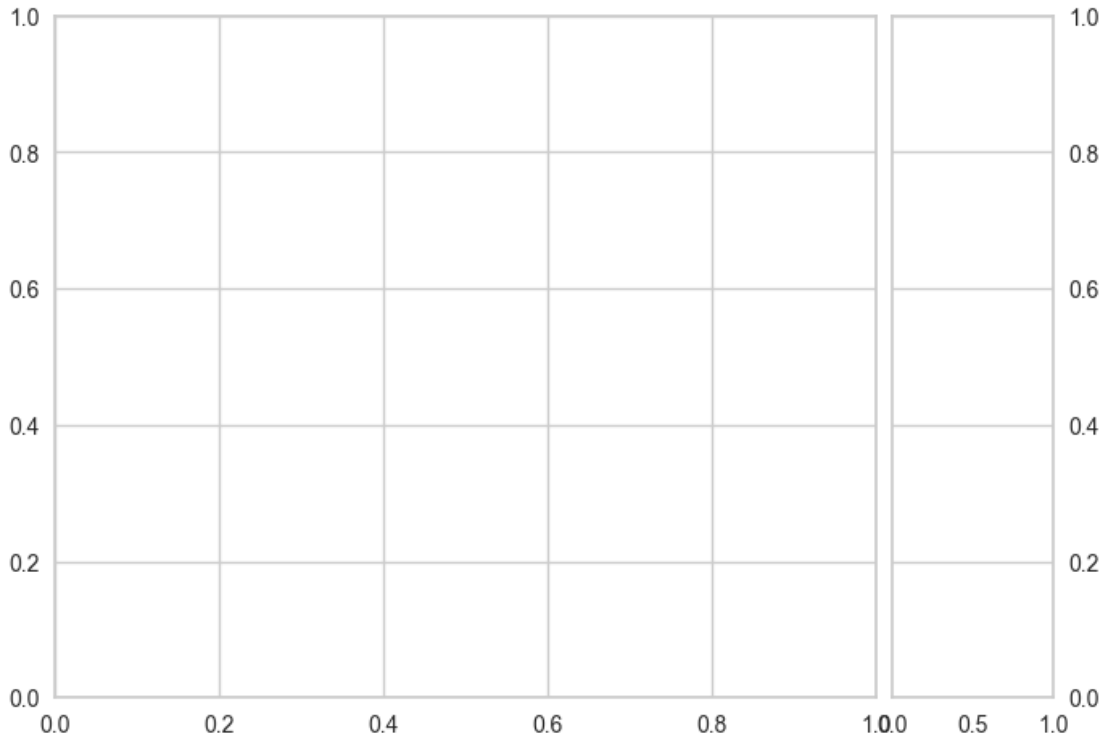
```
Mean Absolute Error: 213.42070819423265
Mean Squared Error: 167198.1421355761
Root Mean Squared Error: 408.8986942209233
Mean Absolute Percentage Error: 11.133863779419864
R2 Score: 0.3778896539204112
Training Time: 0.0029976367950439453
```



[131]:
```
train_and_evaluate_model(ElasticNet(l1_ratio=0.3))
```

```
Mean Absolute Error: 188.9466970519418
Mean Squared Error: 178713.9706704762
Root Mean Squared Error: 422.7457518065394
Mean Absolute Percentage Error: 6.127734848886094
R2 Score: 0.3350415936265909
Training Time: 0.007821083068847656
```


Prediction Error for ElasticNet
$R^2 = 0.335$
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



[132]:
```
train_and_evaluate_model(PoissonRegressor())
```

```
Mean Absolute Error: 36.553124318329665
Mean Squared Error: 20701.894731829234
Root Mean Squared Error: 143.88153019699655
Mean Absolute Percentage Error: 0.27578118260862816
R2 Score: 0.9229724521359908
Training Time: 0.01234579086303711
```

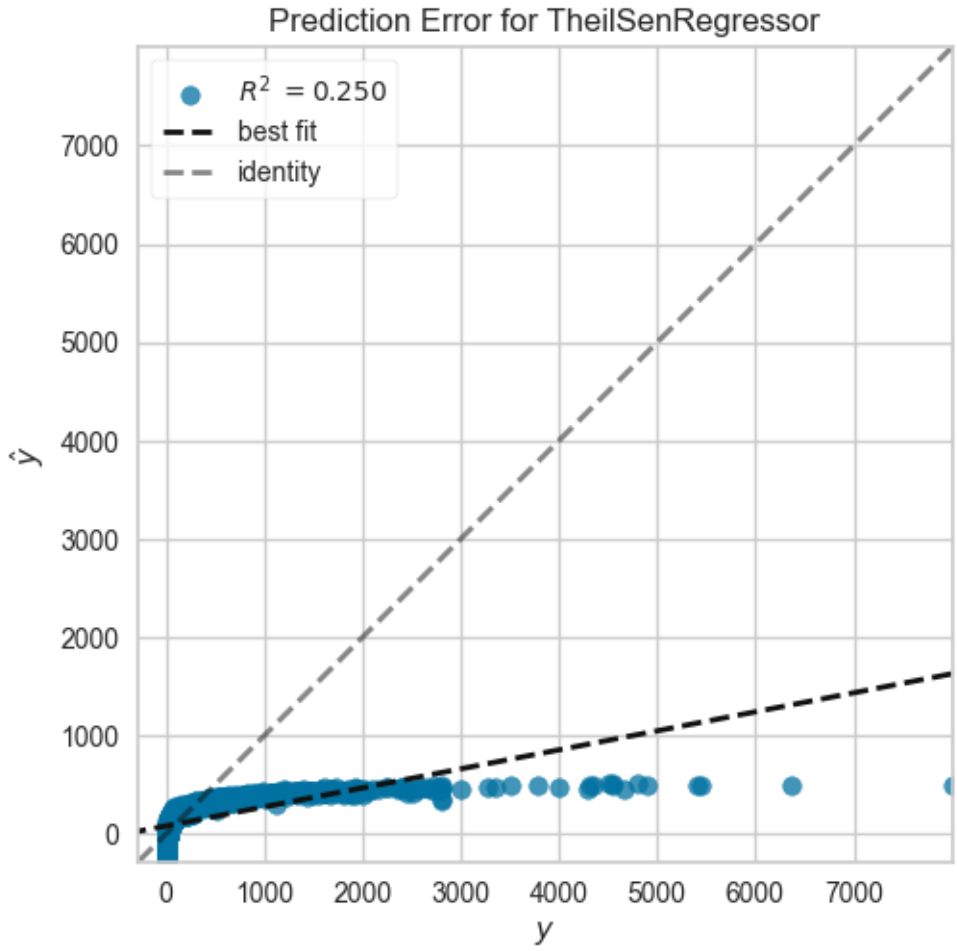
[133]:
```
train_and_evaluate_model(TweedieRegressor())
```

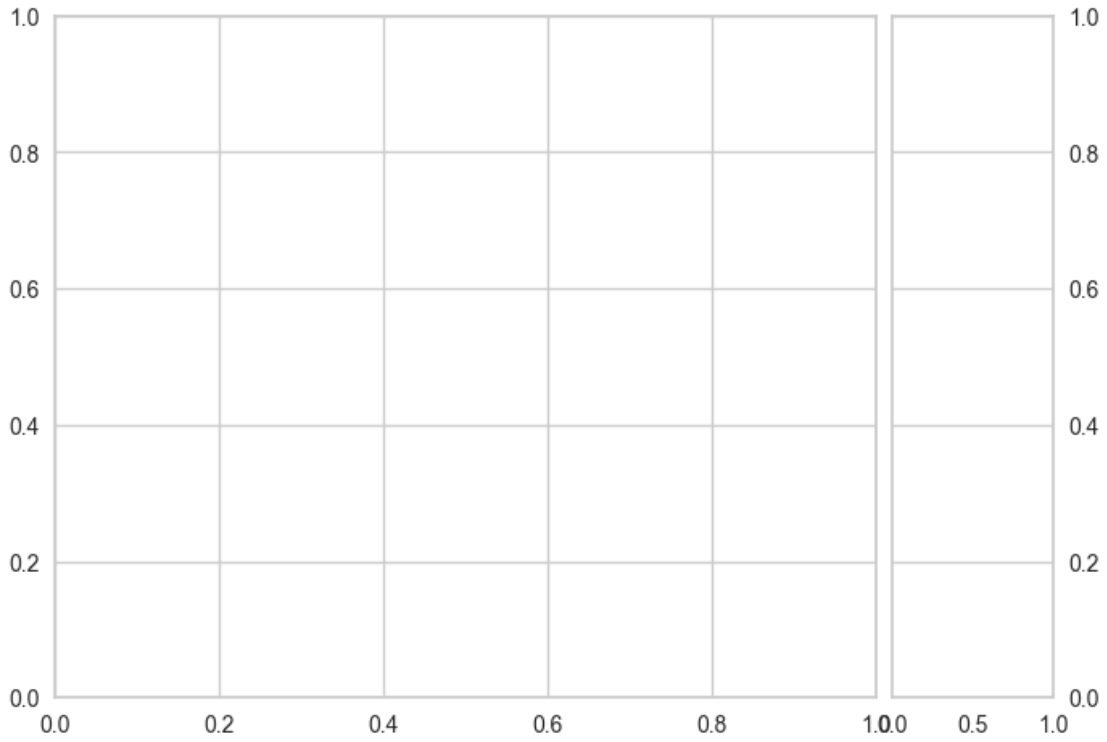
```
Mean Absolute Error: 186.83389369559367
Mean Squared Error: 183663.88195077595
Root Mean Squared Error: 428.56024308231855
Mean Absolute Percentage Error: 5.287975827607536
R2 Score: 0.31662397857227087
Training Time: 0.010033607482910156
```

[134]:
```
train_and_evaluate_model(TheilSenRegressor())
```

```
Mean Absolute Error: 157.56289226059502
Mean Squared Error: 201580.96400338018
Root Mean Squared Error: 448.97768764536636
Mean Absolute Percentage Error: 5.2122163687035545
```

```
R2 Score: 0.24995815337761218
Training Time: 2.2058563232421875
```



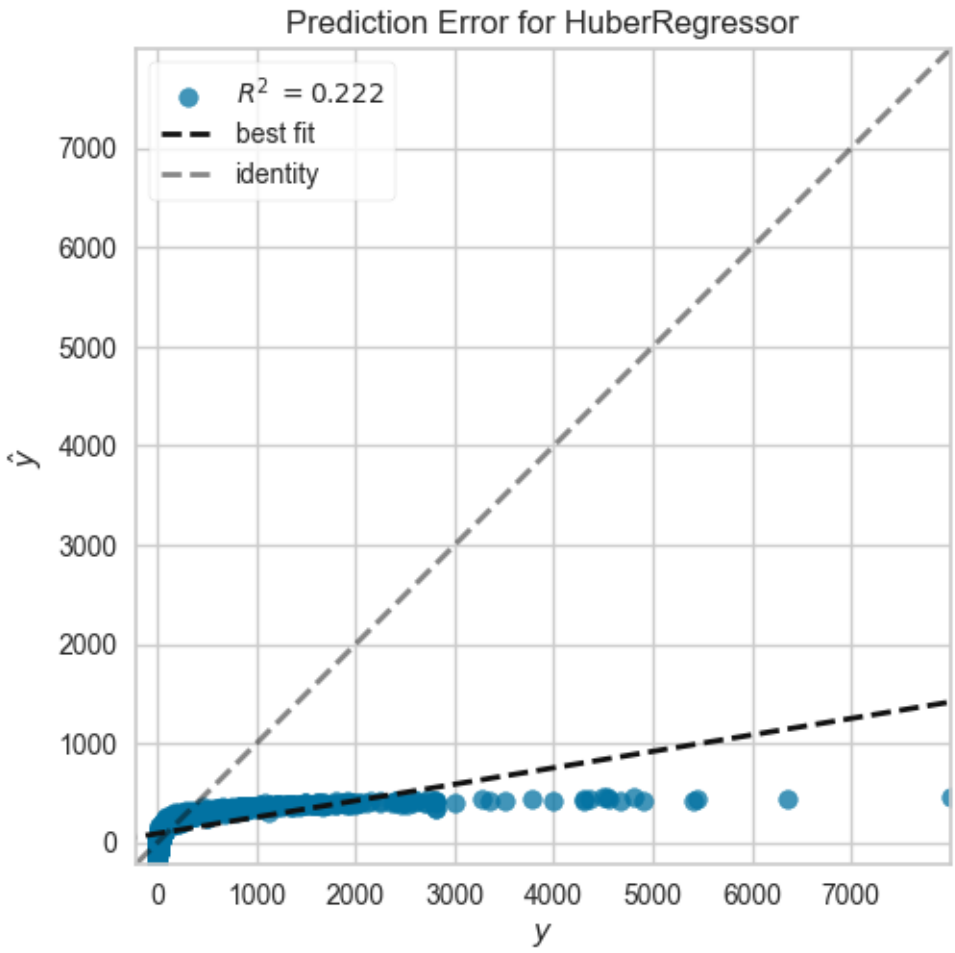


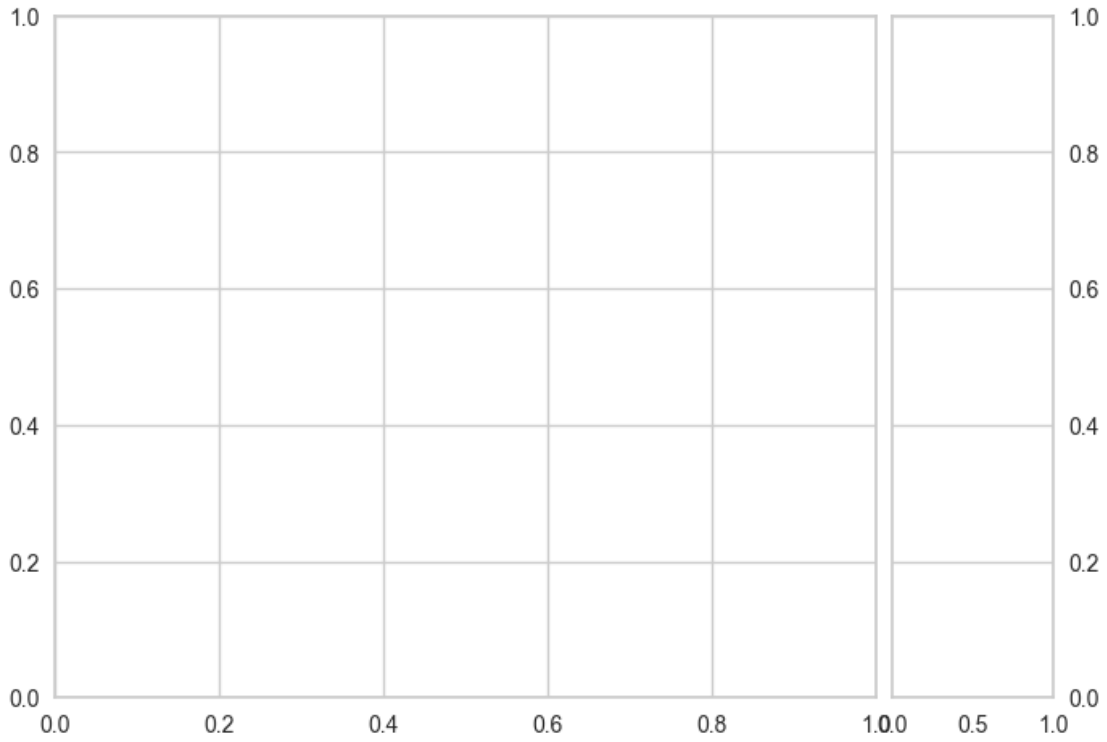
[135]:
```
train_and_evaluate_model(GammaRegressor())
```

```
Mean Absolute Error: 121.88585710962785
Mean Squared Error: 171386.88686612254
Root Mean Squared Error: 413.988993653361
Mean Absolute Percentage Error: 0.8647014371499202
R2 Score: 0.36230418508280726
Training Time: 0.009127140045166016
```

[136]:
```
train_and_evaluate_model(HuberRegressor())
```

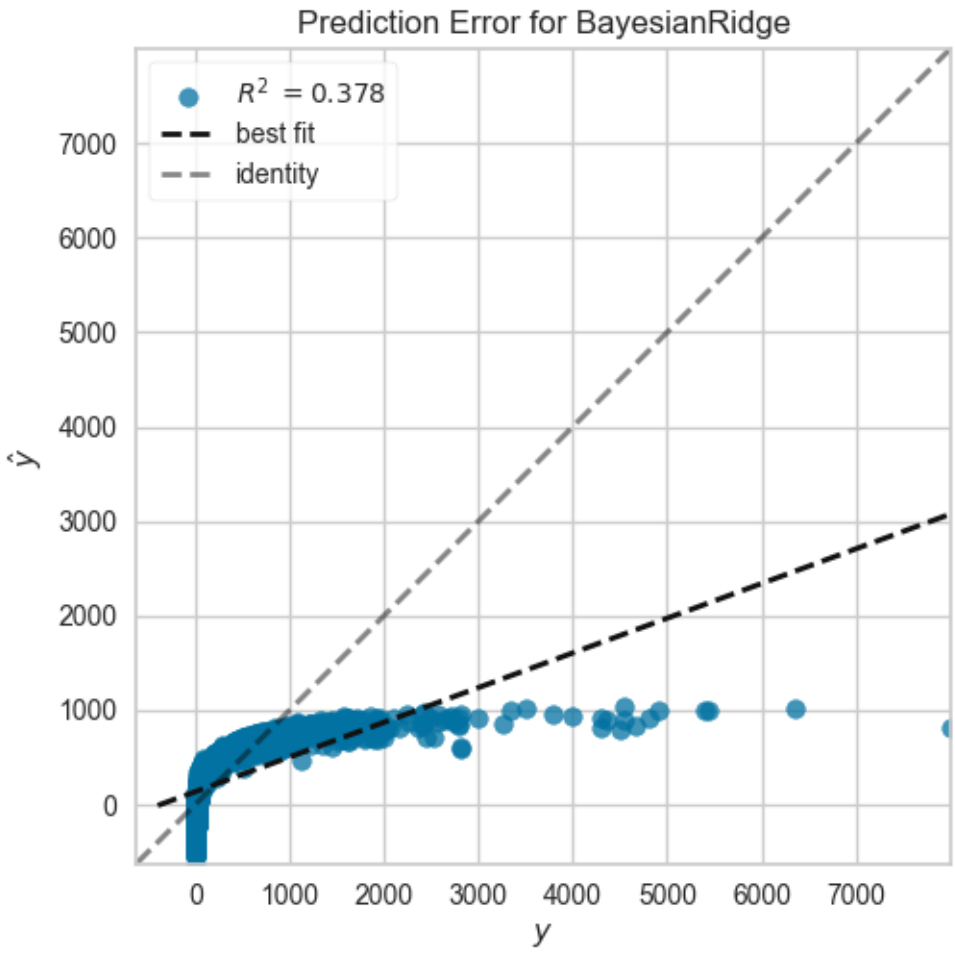
```
Mean Absolute Error: 154.62668125087058
Mean Squared Error: 209097.50277297903
Root Mean Squared Error: 457.27180404326157
Mean Absolute Percentage Error: 3.857416125931706
R2 Score: 0.2219906384546051
Training Time: 0.053121089935302734
```

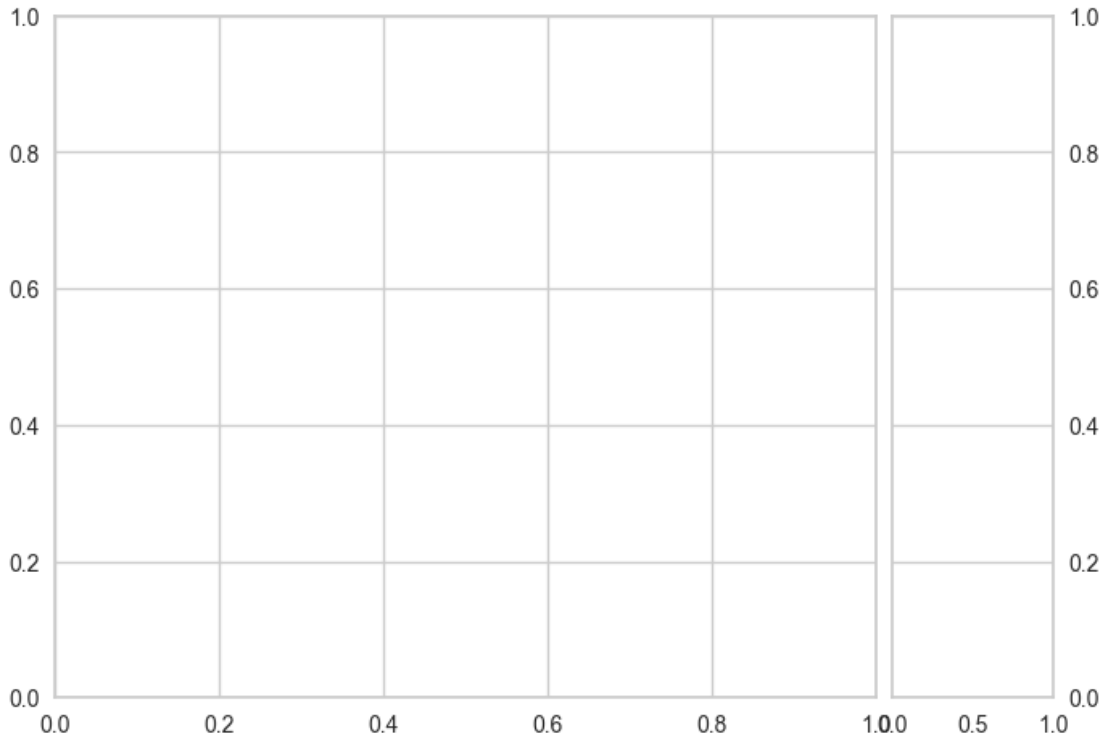

Prediction Error for HuberRegressor
R^2 = 0.222
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



[137]:
```
train_and_evaluate_model(BayesianRidge())
```

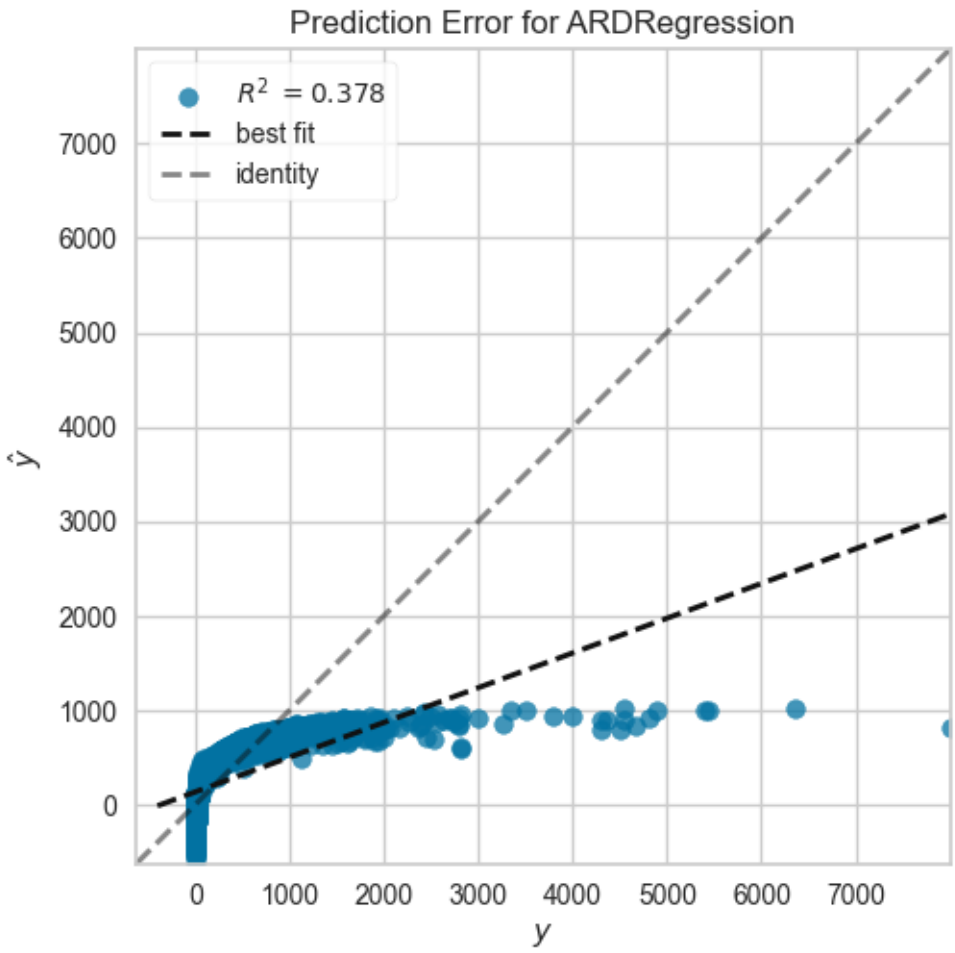
```
Mean Absolute Error: 213.06072026616184
Mean Squared Error: 167179.35522633916
Root Mean Squared Error: 408.87572100375337
Mean Absolute Percentage Error: 11.060482984111333
R2 Score: 0.3779595561959861
Training Time: 0.13362860679626465
```

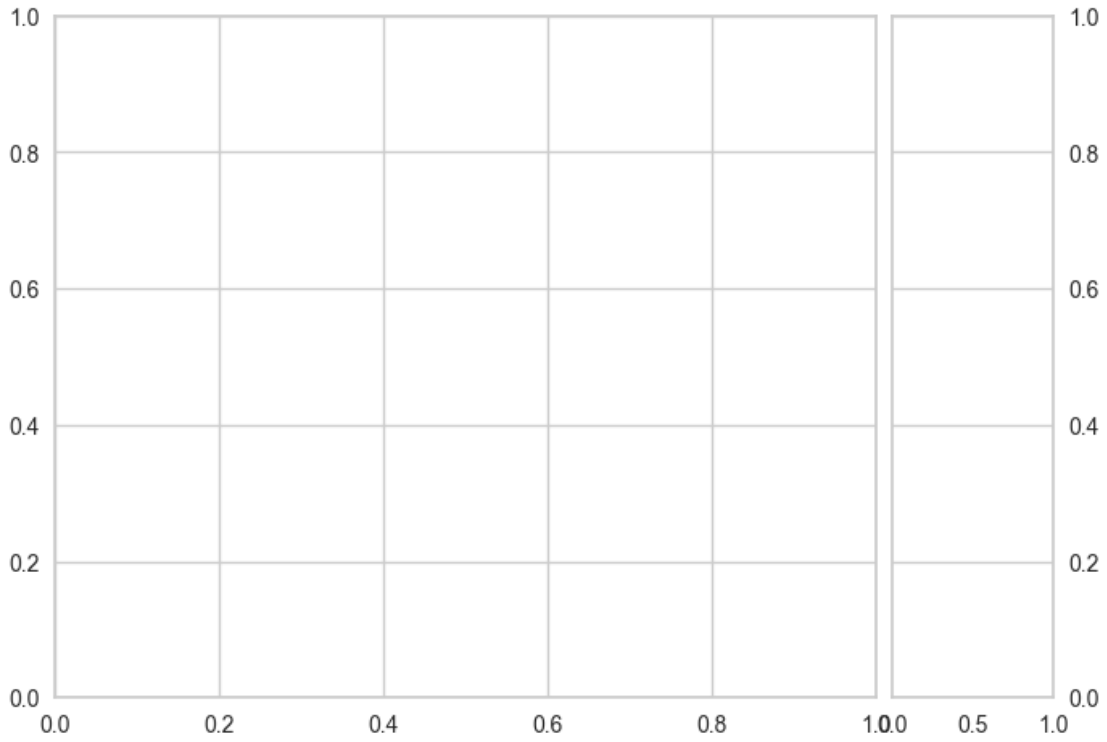

Prediction Error for BayesianRidge
R^2 = 0.378
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



[138]:
```
train_and_evaluate_model(ARDRegression())
```

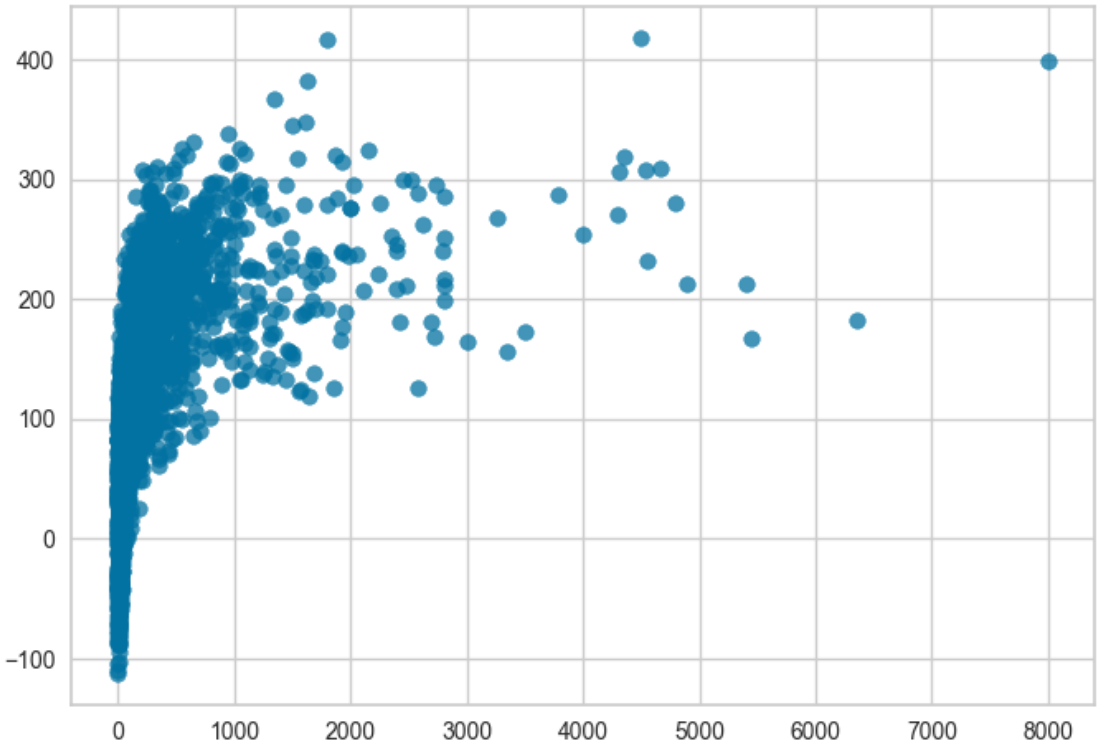
```
Mean Absolute Error: 213.15828090679744
Mean Squared Error: 167180.76873513605
Root Mean Squared Error: 408.87744953119636
Mean Absolute Percentage Error: 11.07274183117957
R2 Score: 0.3779542968166917
Training Time: 0.24023795127868652
```


Prediction Error for ARDRegression
R^2 = 0.378
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



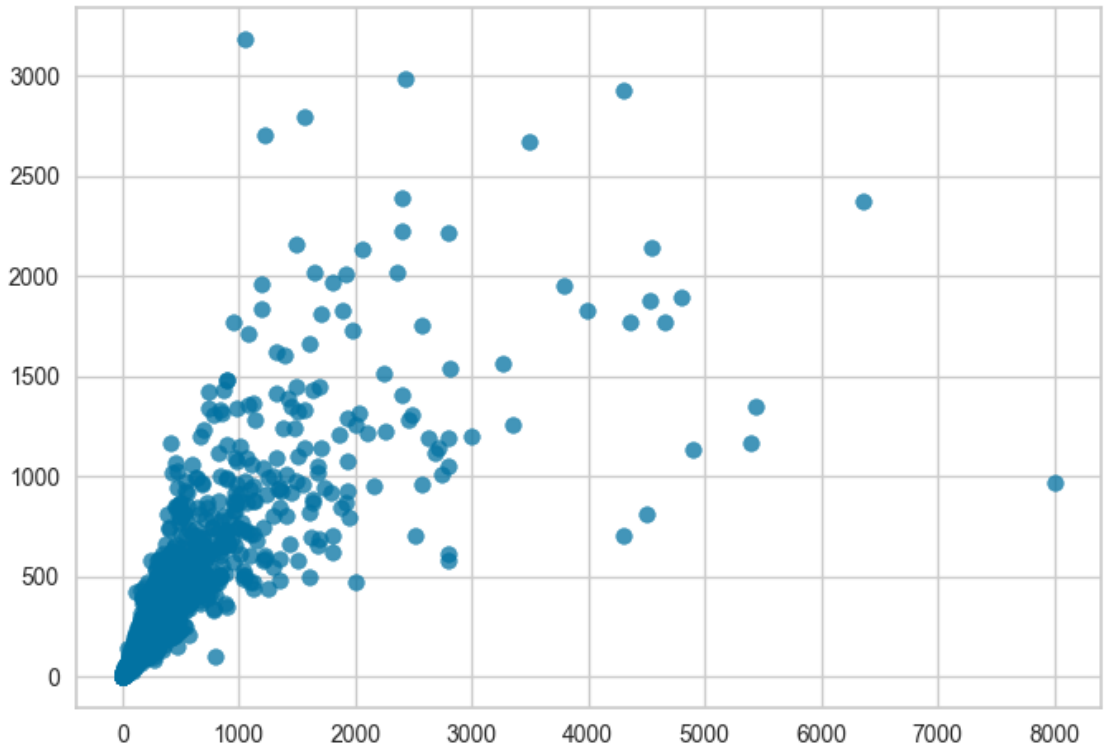
[139]:
```
train_and_evaluate_model(RANSACRegressor())
```

```
Mean Absolute Error: 168.3735449878491
Mean Squared Error: 245549.03867027044
Root Mean Squared Error: 495.52904926983894
Mean Absolute Percentage Error: 2.1964195128270743
R2 Score: 0.08636187295188502
Training Time: 0.18381214141845703
```



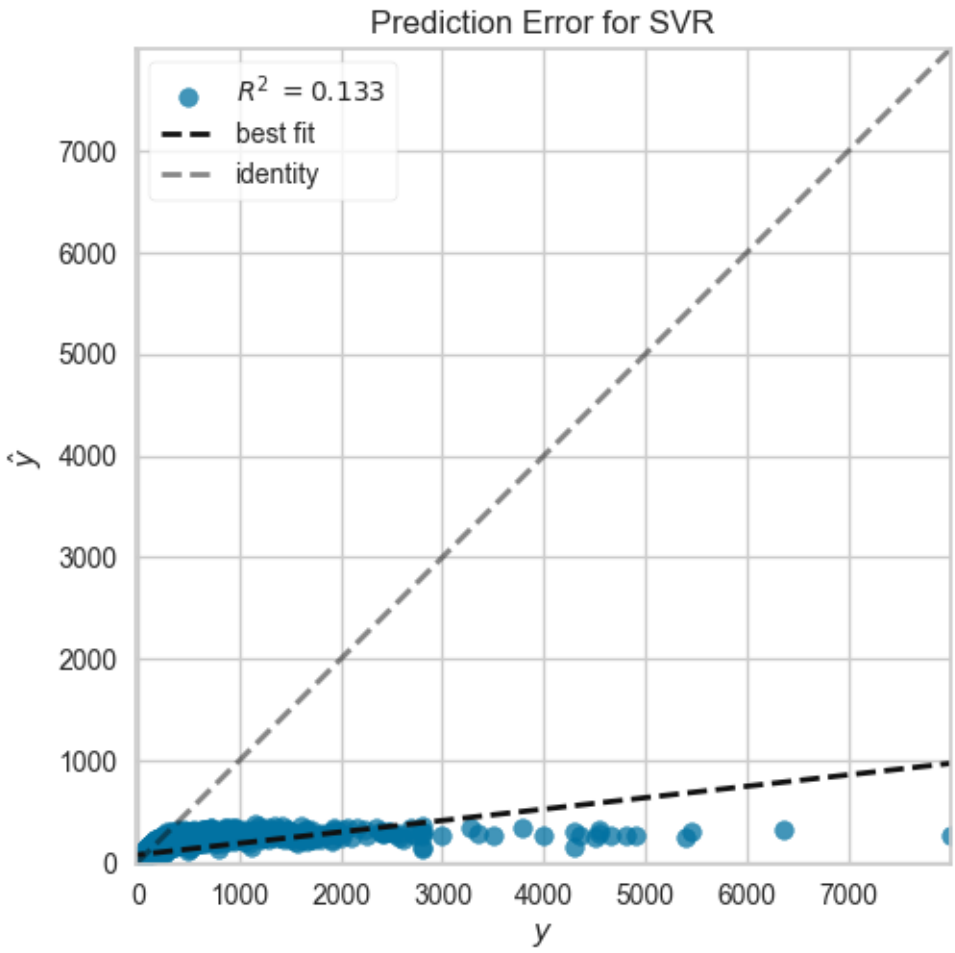
[140]:
```
train_and_evaluate_model(KNeighborsRegressor())
```

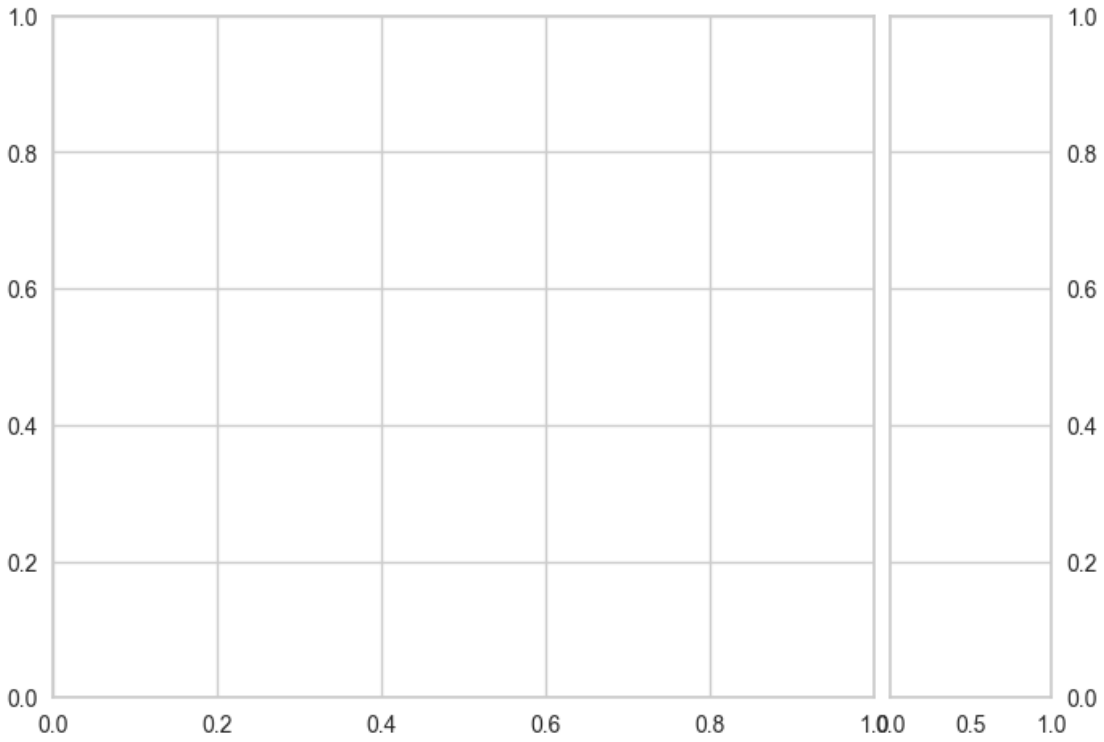
```
Mean Absolute Error: 81.89178158158158
Mean Squared Error: 101780.86219585092
Root Mean Squared Error: 319.0311304494452
Mean Absolute Percentage Error: 0.30069908637151366
R2 Score: 0.6212940730310492
Training Time: 0.015630245208740234
```



[141]:
```
train_and_evaluate_model(SVR())
```

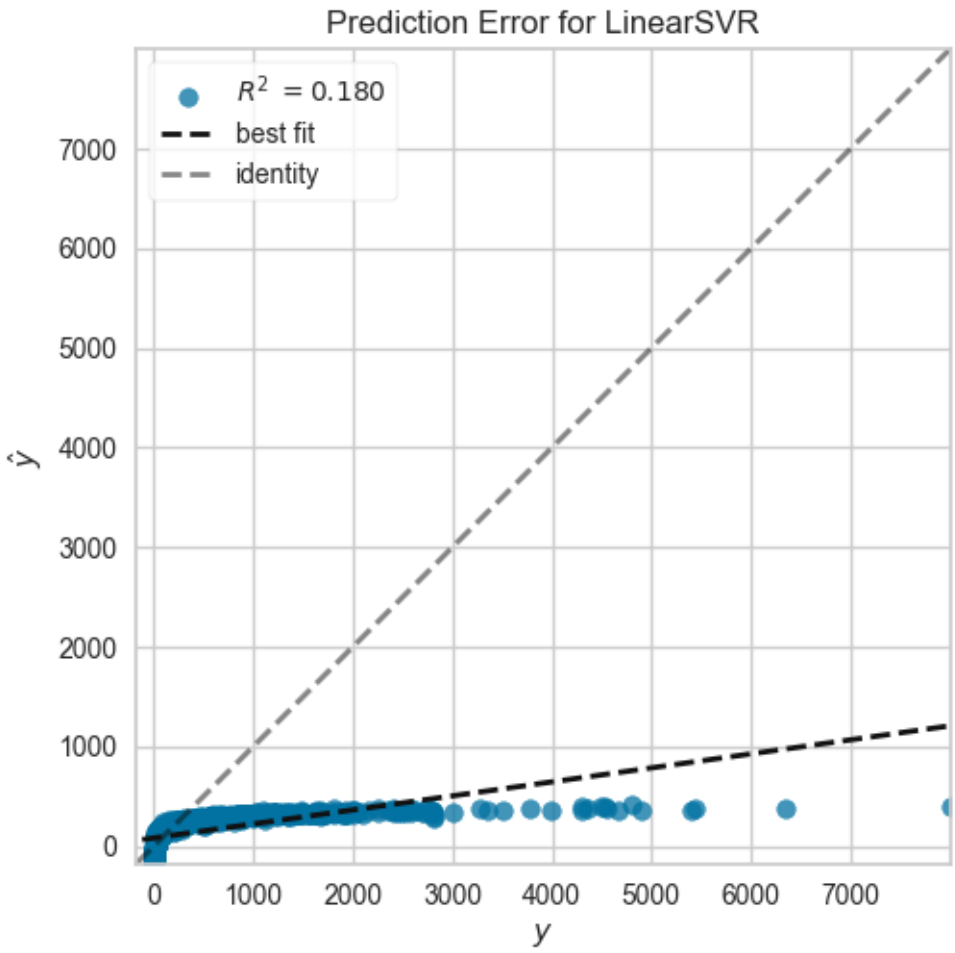
```
Mean Absolute Error: 139.8005942518609
Mean Squared Error: 233141.03728634914
Root Mean Squared Error: 482.8468051943071
Mean Absolute Percentage Error: 0.7278048080498585
R2 Score: 0.13252952730804457
Training Time: 2.899280548095703
```

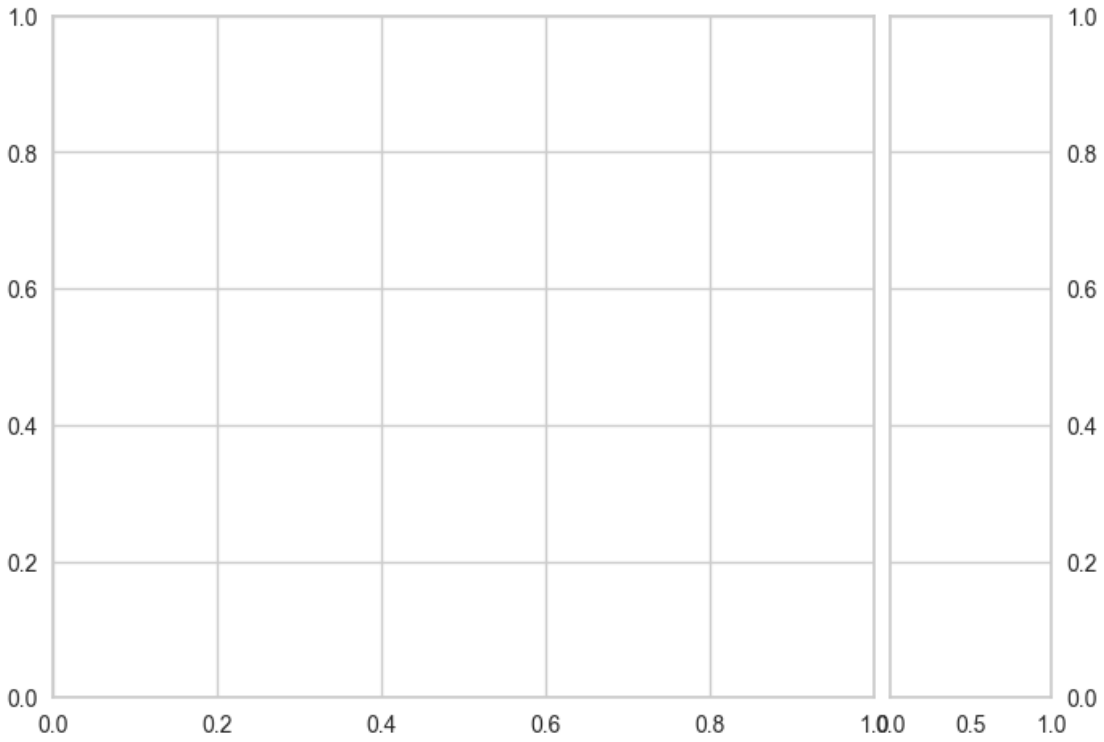

Prediction Error for SVR
$R^2 = 0.133$
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



[142]:
```
train_and_evaluate_model(LinearSVR())
```

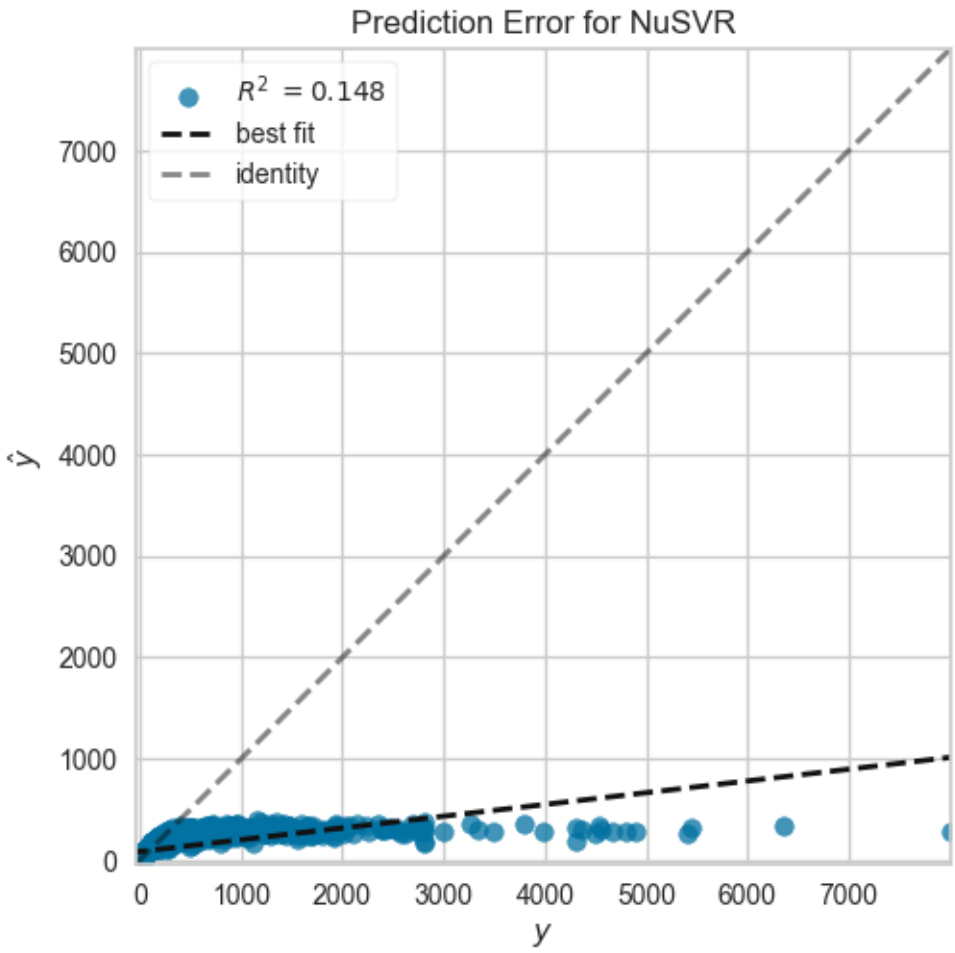
```
Mean Absolute Error: 153.58099230063056
Mean Squared Error: 220511.77549708562
Root Mean Squared Error: 469.5868135894423
Mean Absolute Percentage Error: 3.0897301047163146
R2 Score: 0.1795204467171705
Training Time: 0.008998870849609375
```

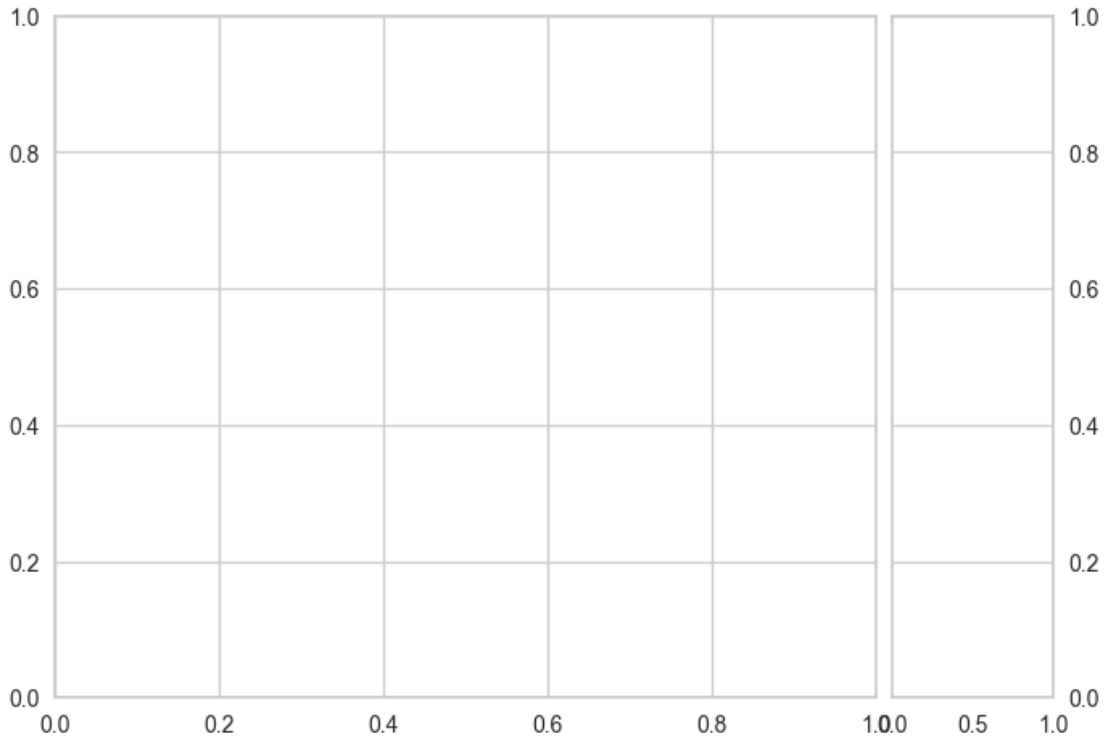

Prediction Error for LinearSVR
R^2 = 0.180
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



[143]:
```
train_and_evaluate_model(NuSVR())
```

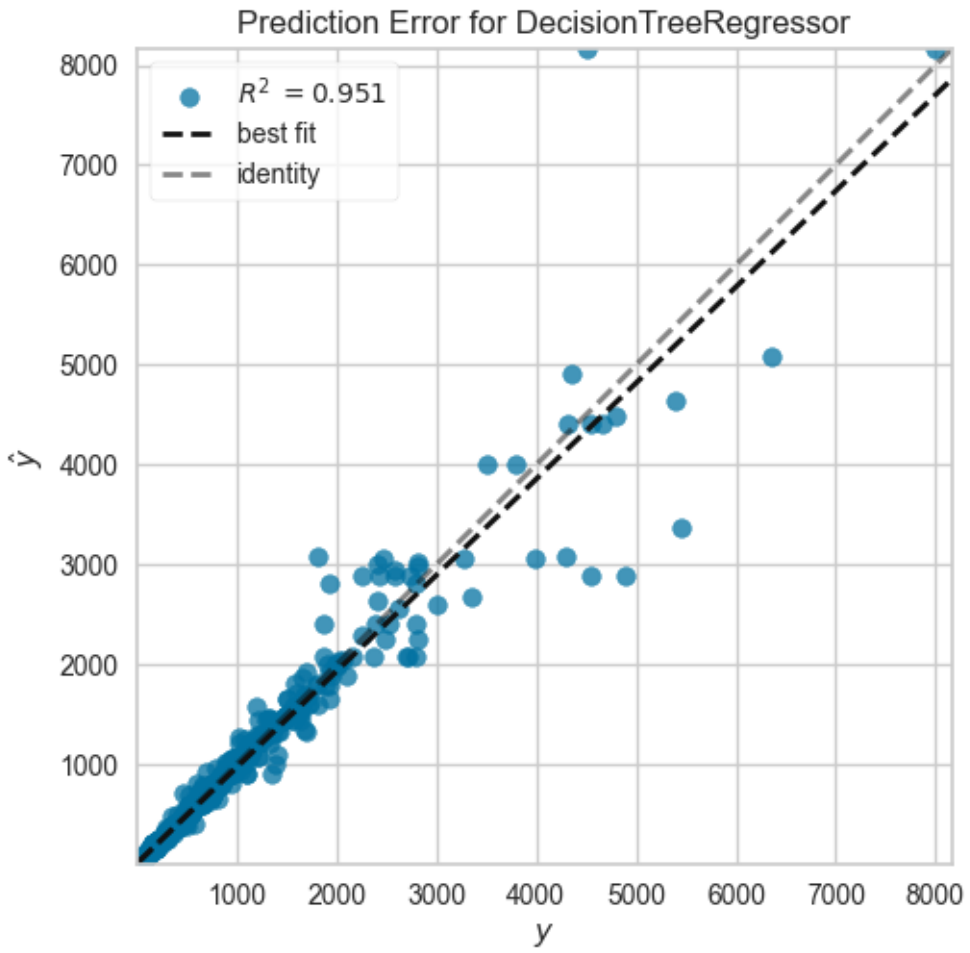
```
Mean Absolute Error: 142.5806770398449
Mean Squared Error: 228972.92921490036
Root Mean Squared Error: 478.51115892411576
Mean Absolute Percentage Error: 1.287698415285431
R2 Score: 0.14803821132633654
Training Time: 1.5845506191253662
```

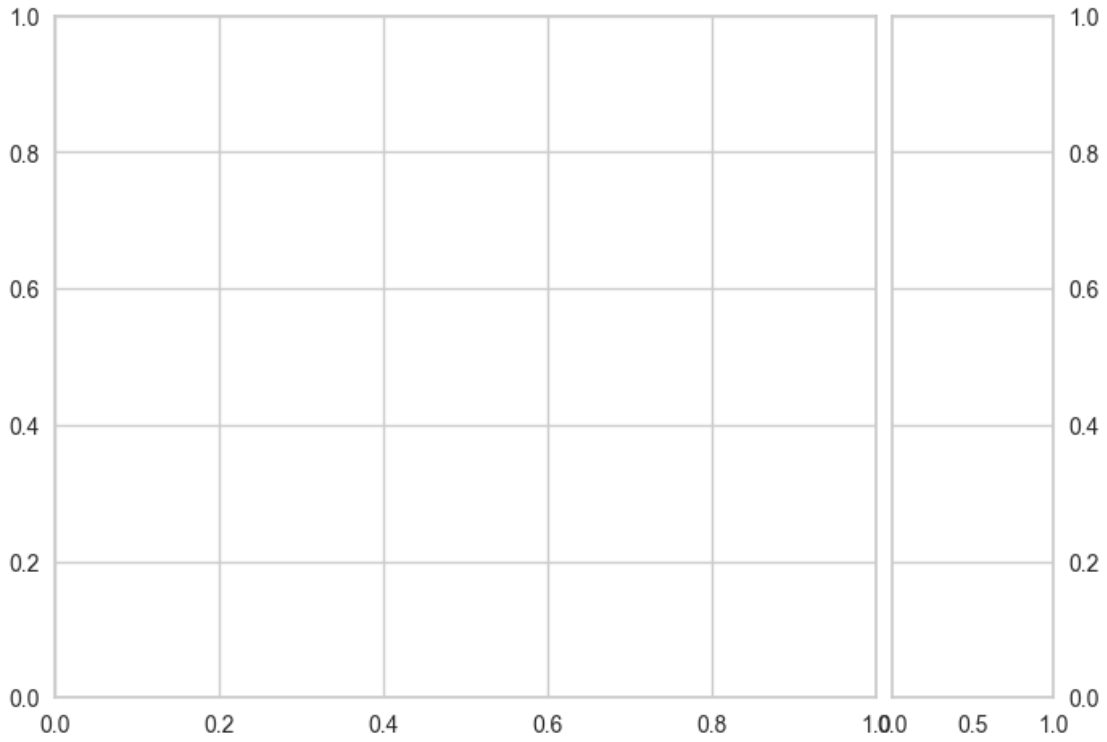

Prediction Error for NuSVR
R^2 = 0.148
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



[144]:
```
train_and_evaluate_model(DecisionTreeRegressor())
```

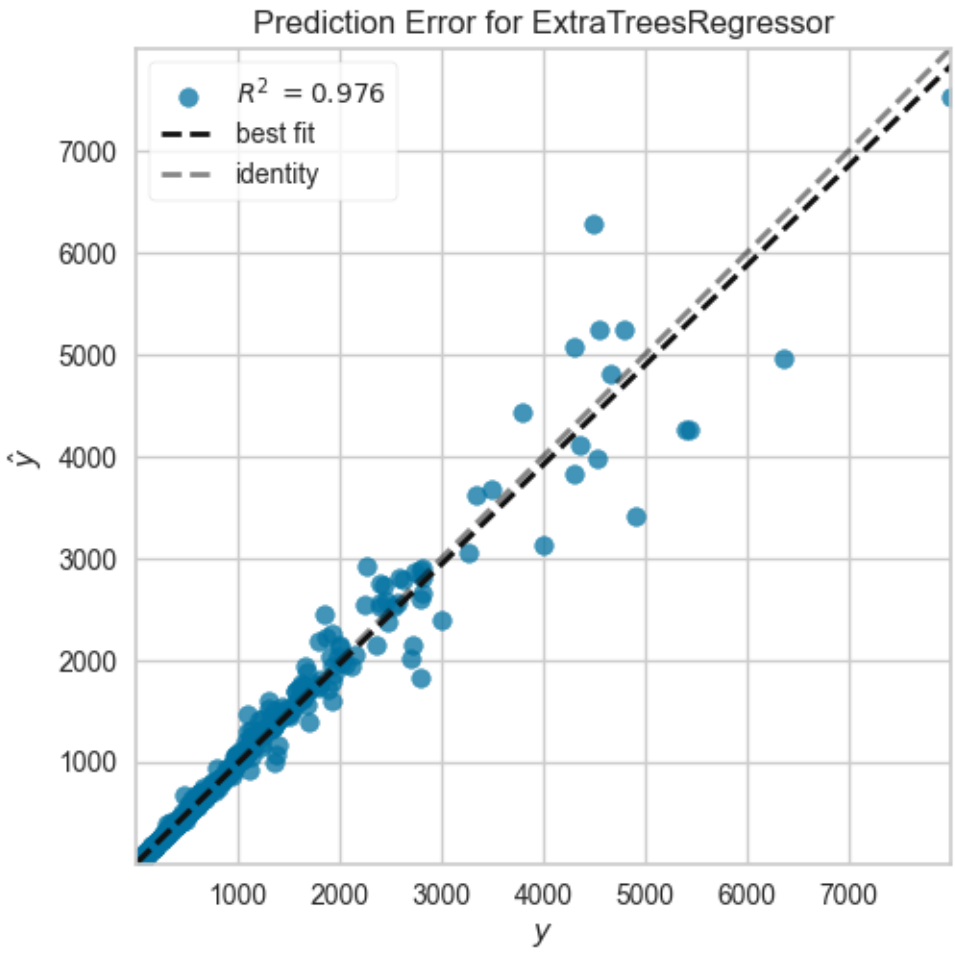
```
Mean Absolute Error: 16.70341918585252
Mean Squared Error: 13191.118209429198
Root Mean Squared Error: 114.85259339444276
Mean Absolute Percentage Error: 0.029597701858030862
R2 Score: 0.9509185269068929
Training Time: 0.06267595291137695
```

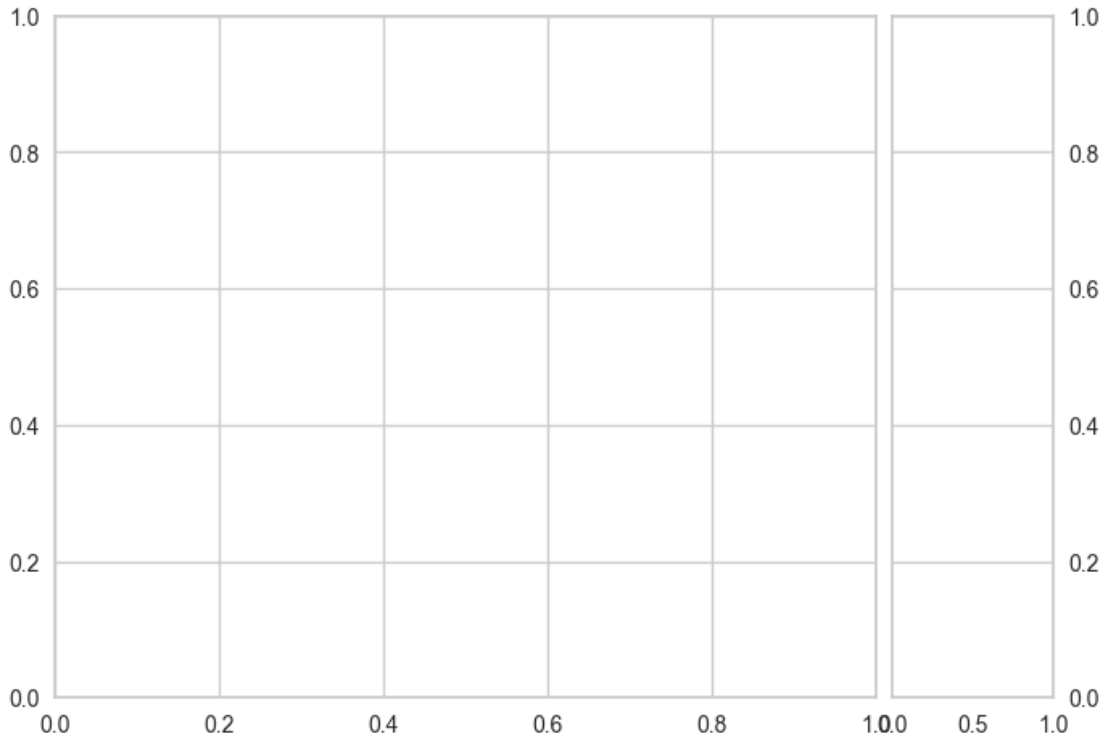

Prediction Error for DecisionTreeRegressor
R^2 = 0.951
best fit
identity
8000
7000
6000
5000
4000
3000
2000
1000
$\hat{y}$
1000
2000
3000
4000
5000
6000
7000
8000
y



[145]:
```
train_and_evaluate_model(ExtraTreesRegressor())
```

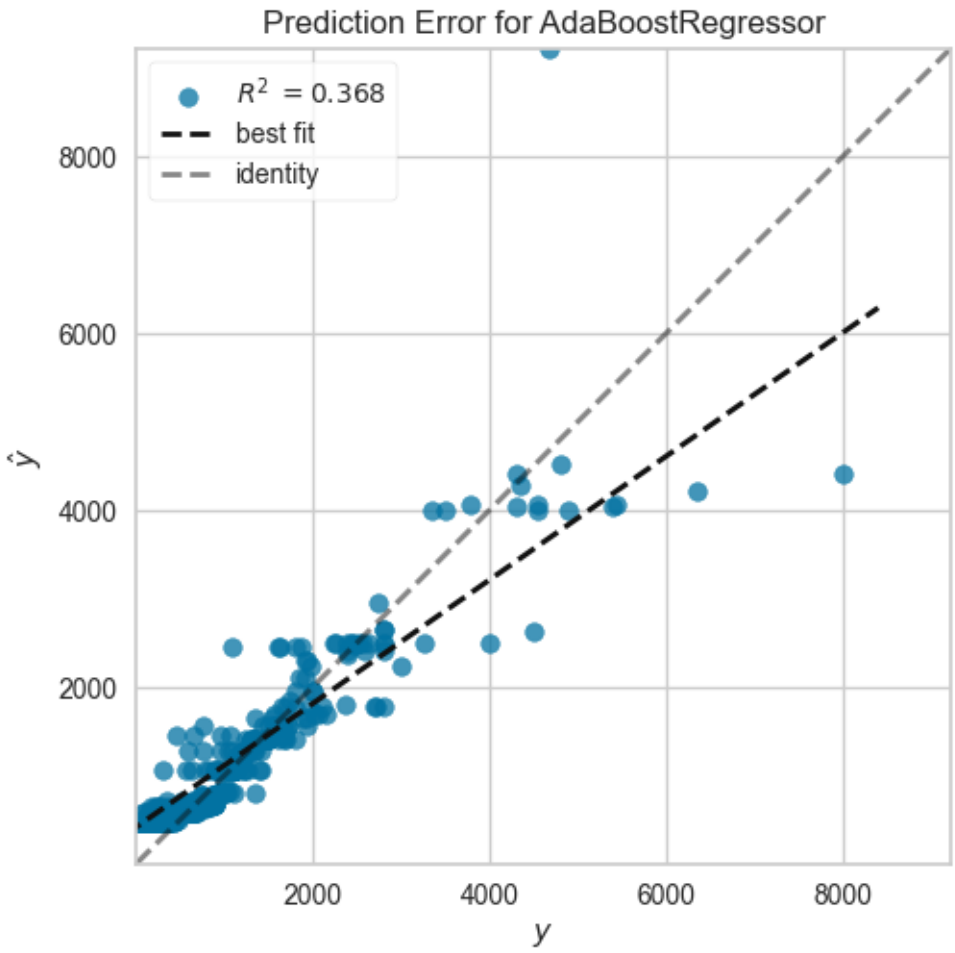
```
Mean Absolute Error: 11.767405445445448
Mean Squared Error: 6359.725944094039
Root Mean Squared Error: 79.74788990370867
Mean Absolute Percentage Error: 0.014177441652607172
R2 Score: 0.9763367507705707
Training Time: 1.8951308727264404
```

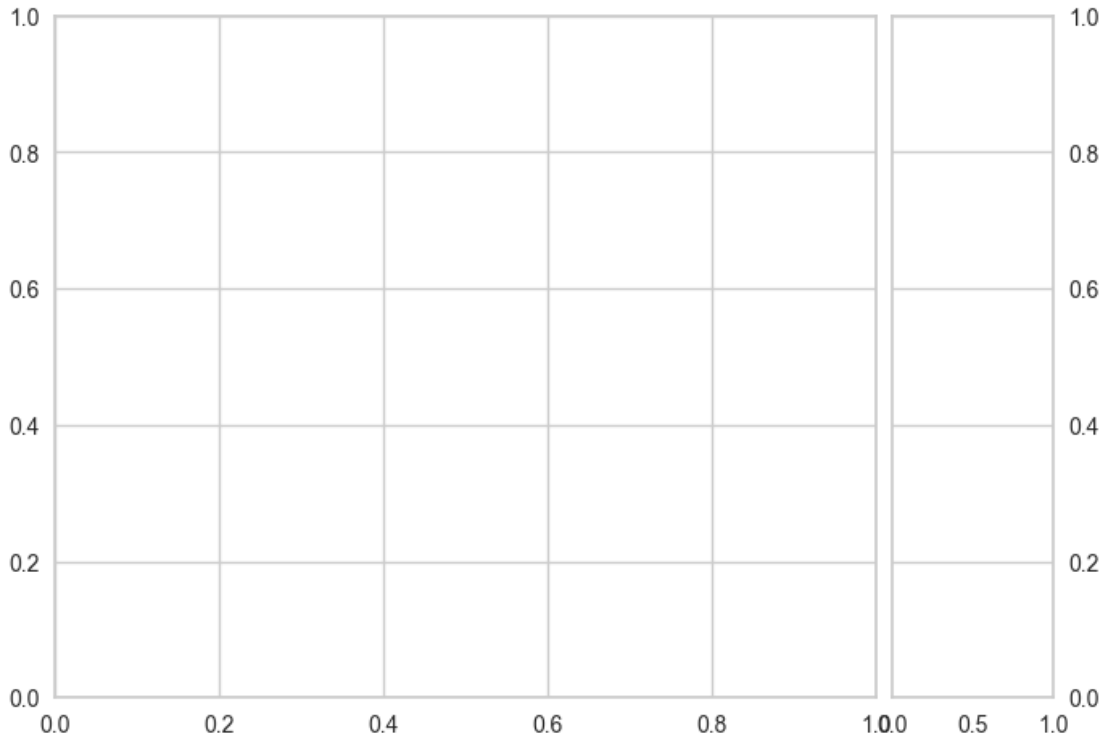

Prediction Error for ExtraTreesRegressor
R^2 = 0.976
best fit
identity
7000
6000
5000
4000
3000
2000
1000
ŷ
1000
2000
3000
4000
5000
6000
7000
y



[146]:
```
train_and_evaluate_model(AdaBoostRegressor())
```

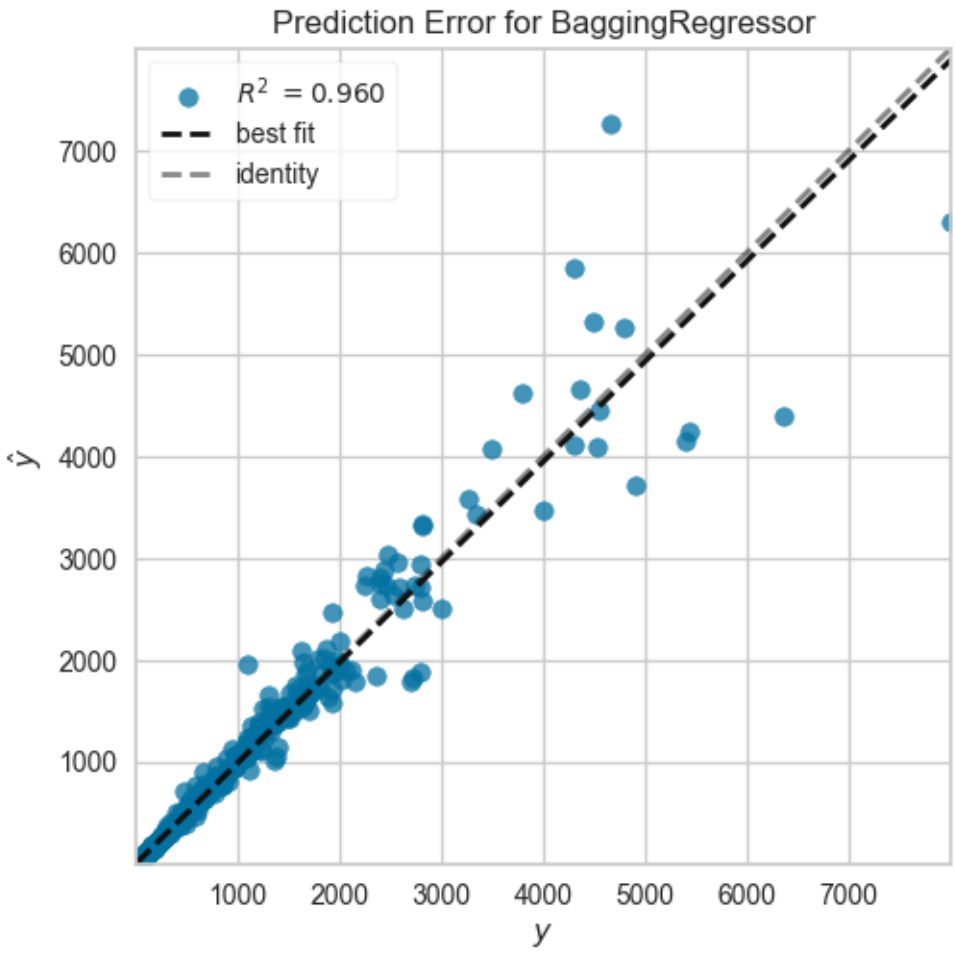
```
Mean Absolute Error: 373.0618256230851
Mean Squared Error: 169901.10125363184
Root Mean Squared Error: 412.19061276748147
Mean Absolute Percentage Error: 24.094041649226078
R2 Score: 0.36783249173610233
Training Time: 0.37844395637512207
```

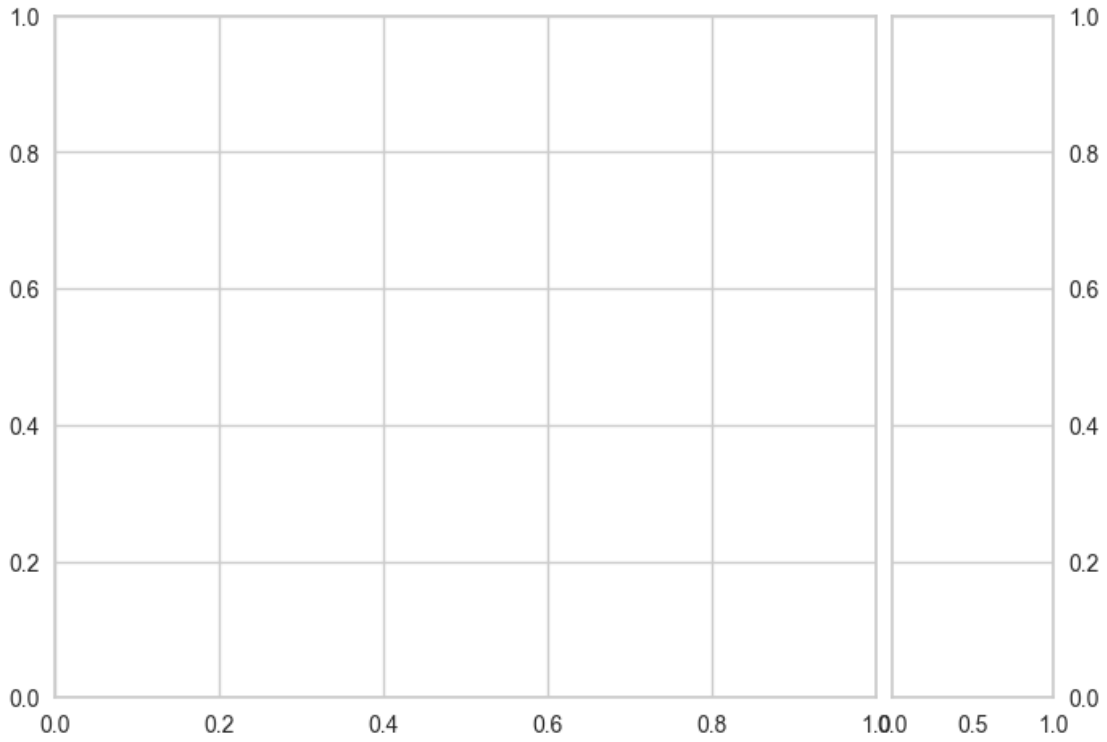

Prediction Error for AdaBoostRegressor
R^2 = 0.368
best fit
identity
8000
6000
4000
2000
$\hat{y}$
2000
4000
6000
8000
y



[147]:
```
train_and_evaluate_model(BaggingRegressor())
```

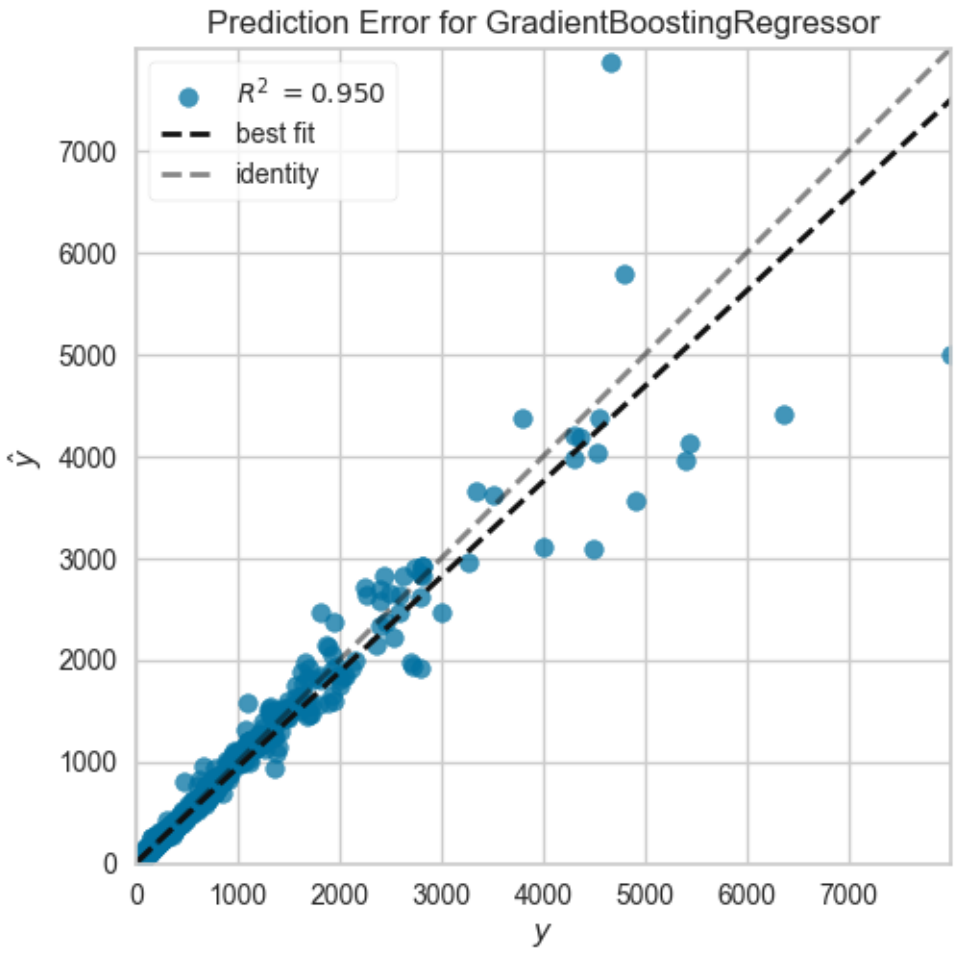
```
Mean Absolute Error: 15.934095178511846
Mean Squared Error: 10662.775435948053
Root Mean Squared Error: 103.26071584076905
Mean Absolute Percentage Error: 0.022795975968923358
R2 Score: 0.9603259771197235
Training Time: 0.45496463775634766
```

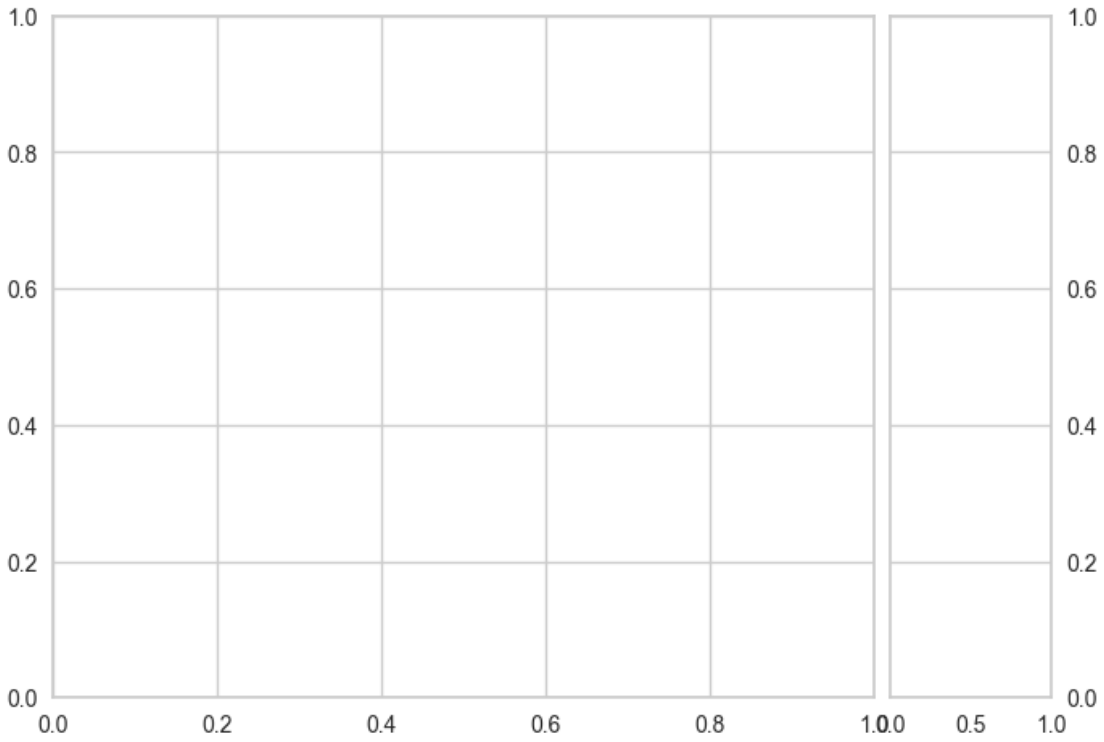

Prediction Error for BaggingRegressor
$R^2 = 0.960$
best fit
identity
7000
6000
5000
4000
3000
2000
1000
$\hat{y}$
1000
2000
3000
4000
5000
6000
7000
y



[148]:
```
train_and_evaluate_model(GradientBoostingRegressor())
```

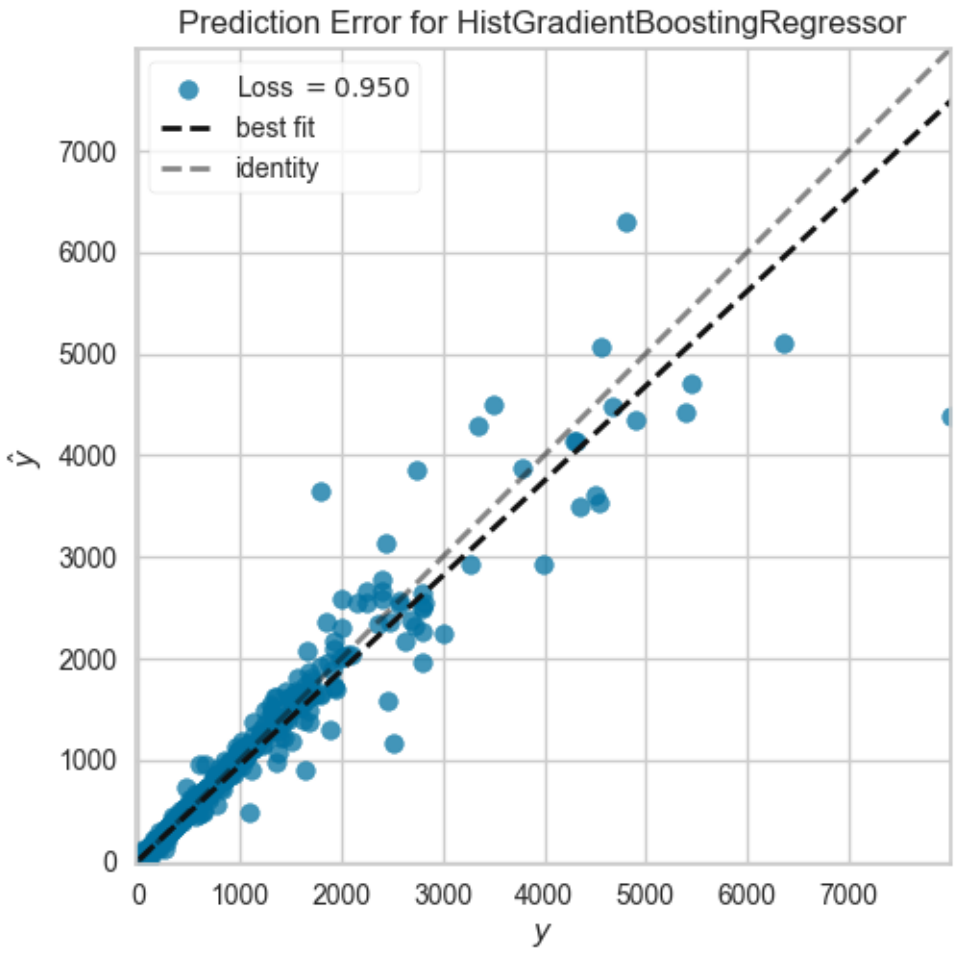
```
Mean Absolute Error: 20.64486002830925
Mean Squared Error: 13493.017227189108
Root Mean Squared Error: 116.1594474297683
Mean Absolute Percentage Error: 0.18874401808912583
R2 Score: 0.9497952219465579
Training Time: 1.1912915706634521
```

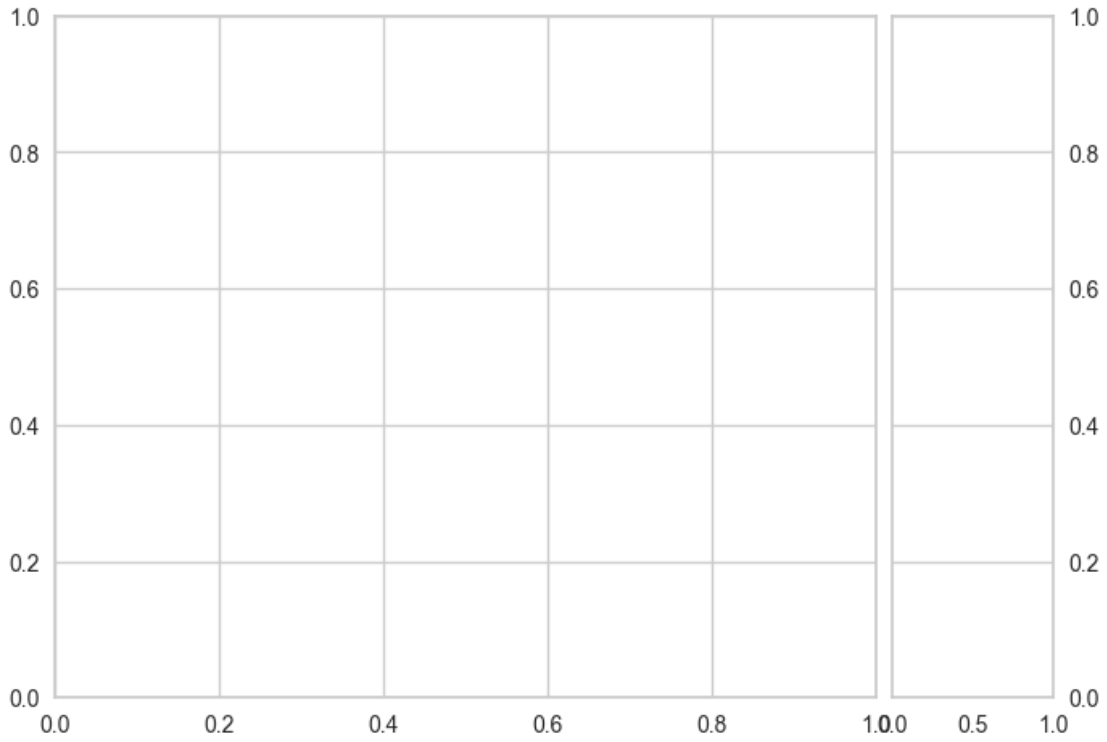

Prediction Error for GradientBoostingRegressor
$R^2 = 0.950$
best fit
identity
$\hat{y}$
y
0
1000
2000
3000
4000
5000
6000
7000



[149]:
```
train_and_evaluate_model(HistGradientBoostingRegressor())
```

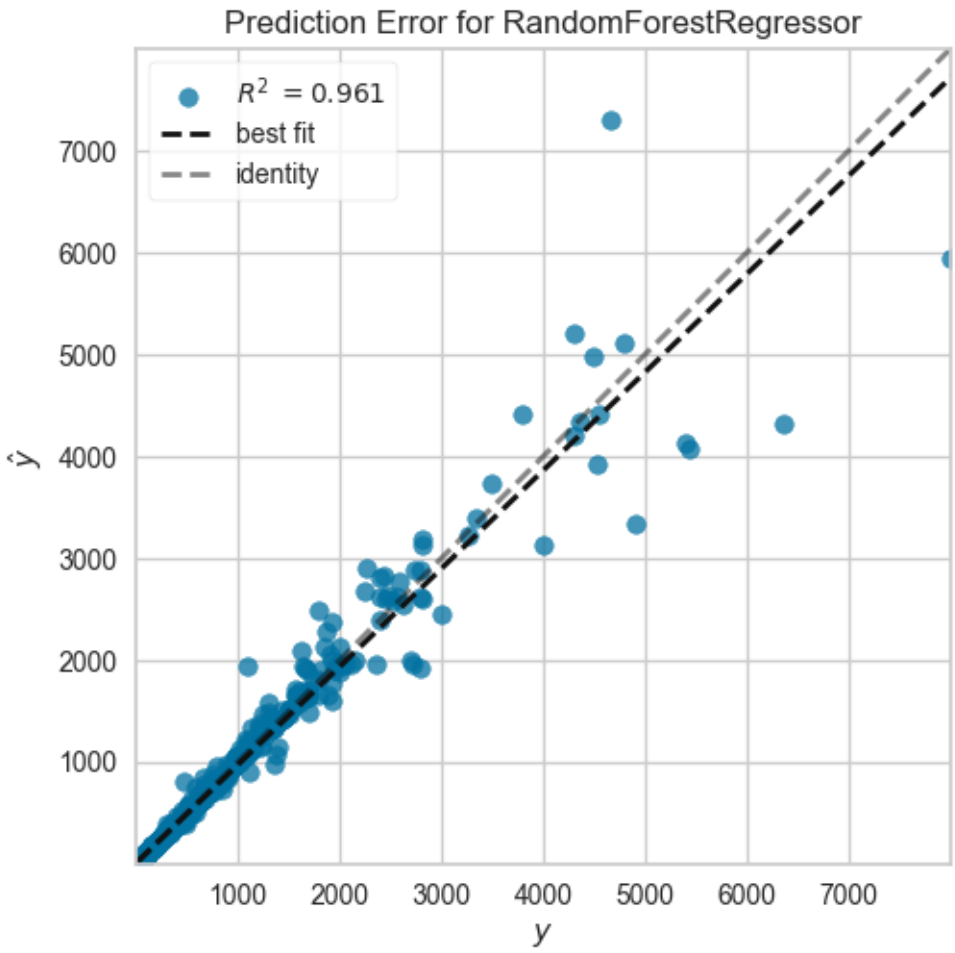
```
Mean Absolute Error: 20.29512863279213
Mean Squared Error: 13431.09317927563
Root Mean Squared Error: 115.89259328911244
Mean Absolute Percentage Error: 0.10049273279292983
R2 Score: 0.9500256287584162
Training Time: 0.5650954246520996
```

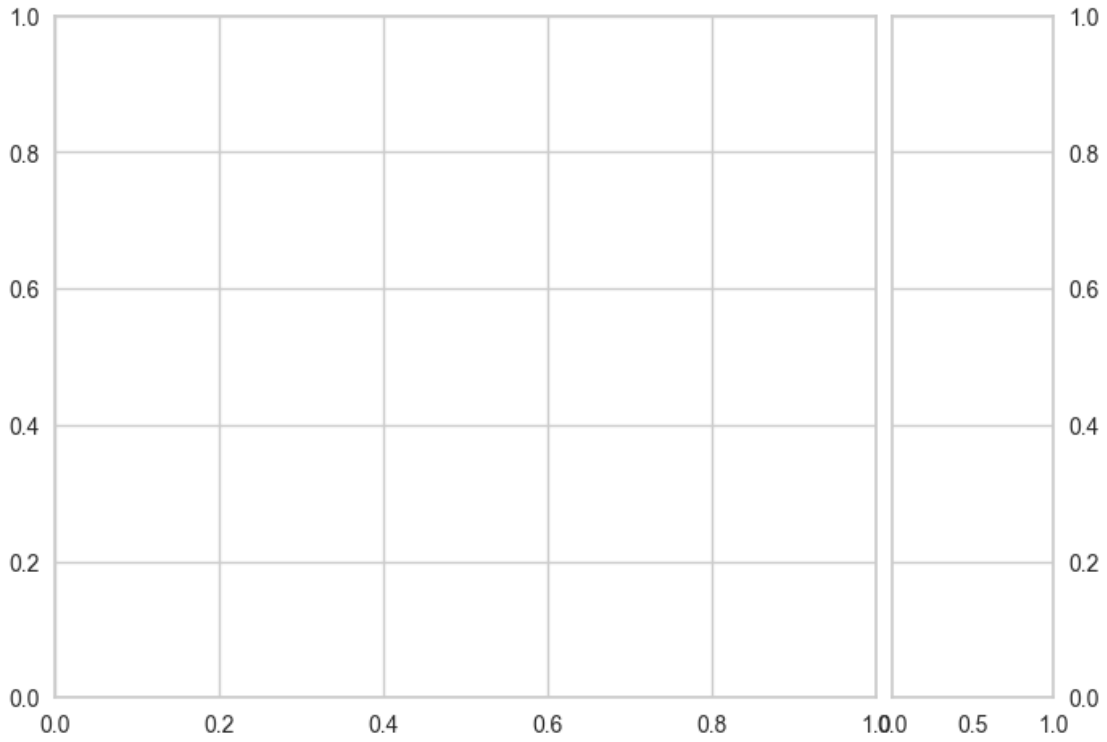

Prediction Error for HistGradientBoostingRegressor
Loss = 0.950
best fit
identity
ŷ
y
0
1000
2000
3000
4000
5000
6000
7000



[150]:
```
train_and_evaluate_model(RandomForestRegressor())
```

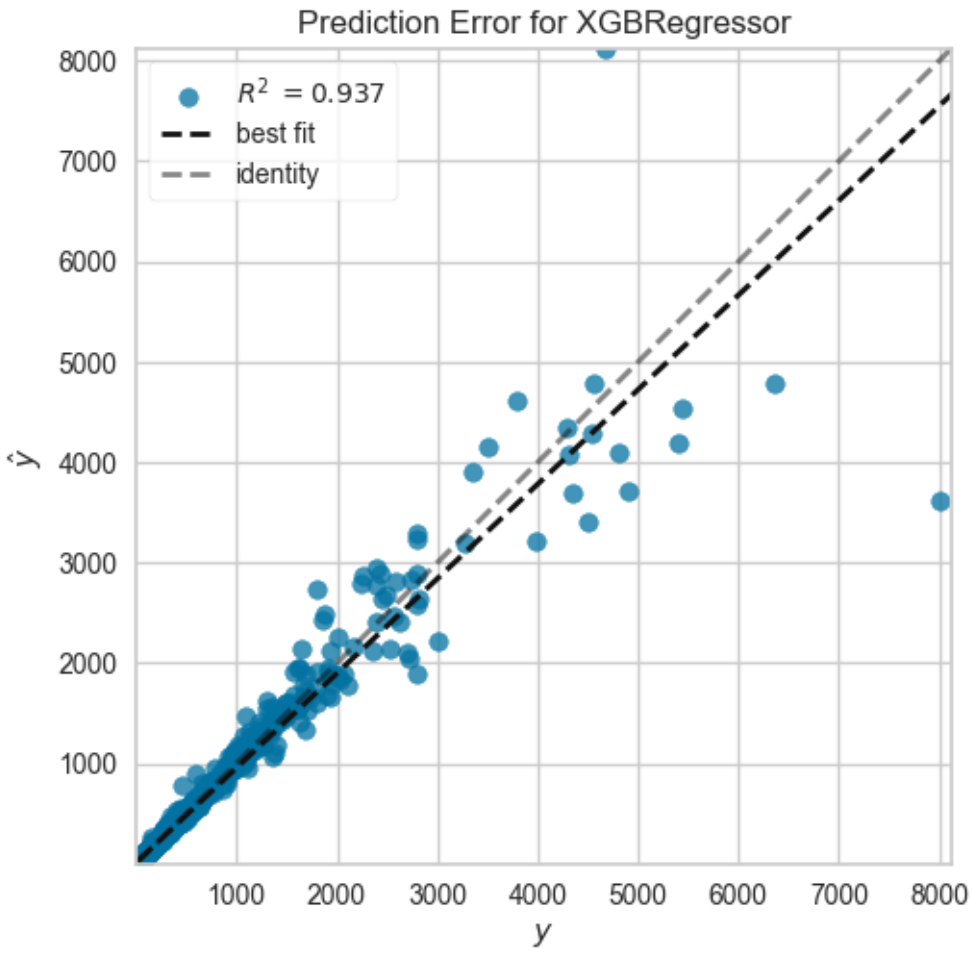
```
Mean Absolute Error: 14.07189562729396
Mean Squared Error: 10465.381974943984
Root Mean Squared Error: 102.30044953441791
Mean Absolute Percentage Error: 0.019037353428839998
R2 Score: 0.9610604381177381
Training Time: 4.4870476722717285
```

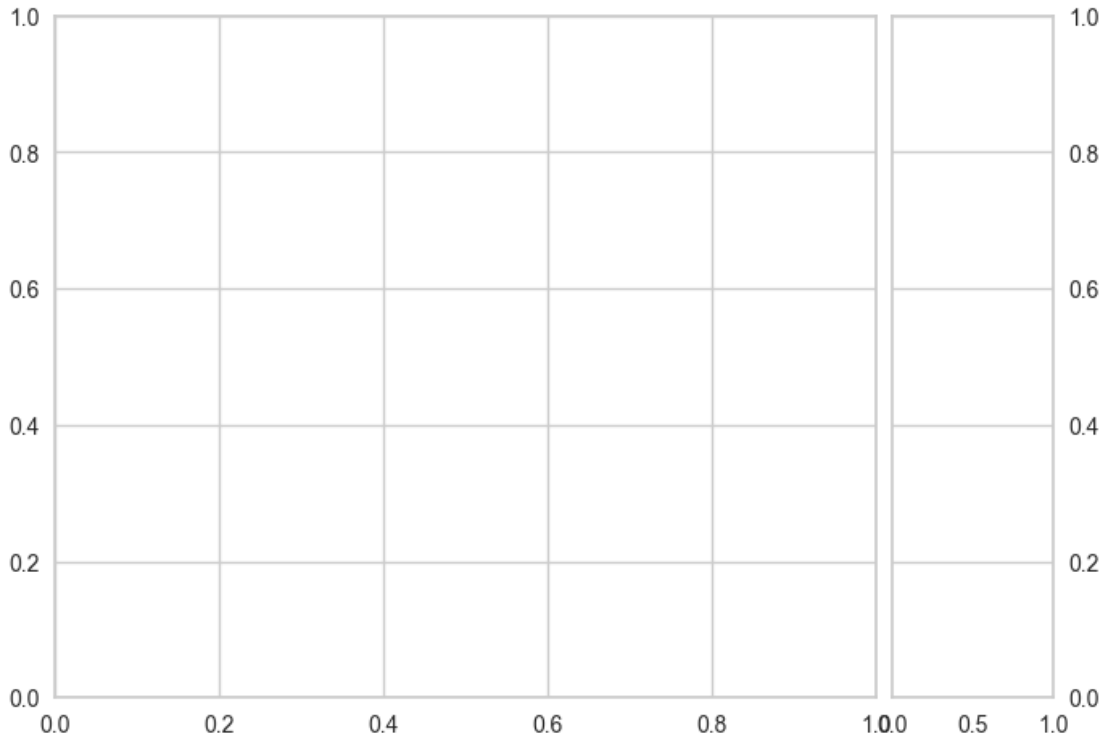

Prediction Error for RandomForestRegressor
R^2 = 0.961
best fit
identity
7000
6000
5000
4000
3000
2000
1000
$\hat{y}$
1000
2000
3000
4000
5000
6000
7000
y



[151]:
```
train_and_evaluate_model(XGBRegressor())
```

```
Mean Absolute Error: 19.70806272383624
Mean Squared Error: 16990.802533077545
Root Mean Squared Error: 130.34877265658295
Mean Absolute Percentage Error: 0.08365042218846372
R2 Score: 0.9367806728650622
Training Time: 0.27833032608032227
```


Prediction Error for XGBRegressor
R^2 = 0.937
best fit
identity
$\hat{y}$
y
8000
7000
6000
5000
4000
3000
2000
1000
1000
2000
3000
4000
5000
6000
7000
8000

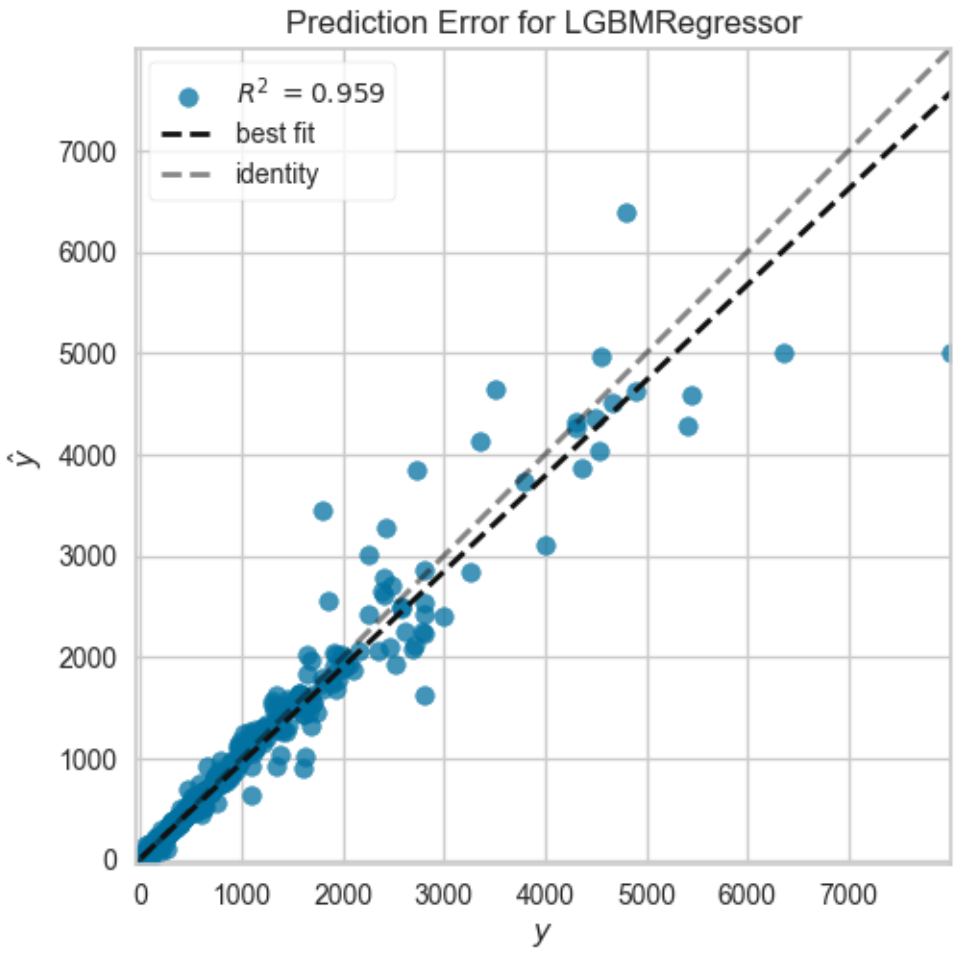


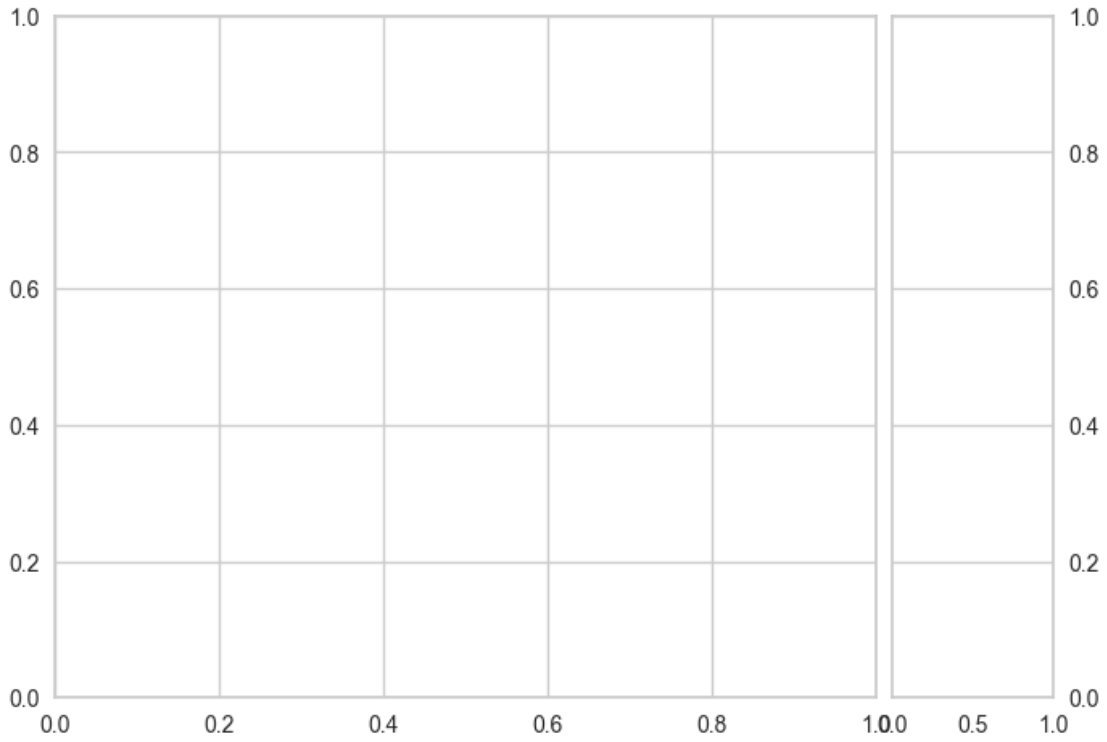
[152]:
```
train_and_evaluate_model(CatBoostRegressor(silent=True))
```

```
Mean Absolute Error: 17.901773051320596
Mean Squared Error: 10572.891101993584
Root Mean Squared Error: 102.82456468176068
Mean Absolute Percentage Error: 0.11931038240296606
R2 Score: 0.9606604184800718
Training Time: 4.6550819873809814
```

[153]:
```
train_and_evaluate_model(LGBMRegressor())
```

```
Mean Absolute Error: 18.645762437566177
Mean Squared Error: 10966.364507214426
Root Mean Squared Error: 104.72041112989591
Mean Absolute Percentage Error: 0.10556997511798226
R2 Score: 0.9591963838133675
Training Time: 0.2079143524169922
```


Prediction Error for LGBMRegressor
R^2 = 0.959
best fit
identity
$\hat{y}$
y
0
1000
2000
3000
4000
5000
6000
7000

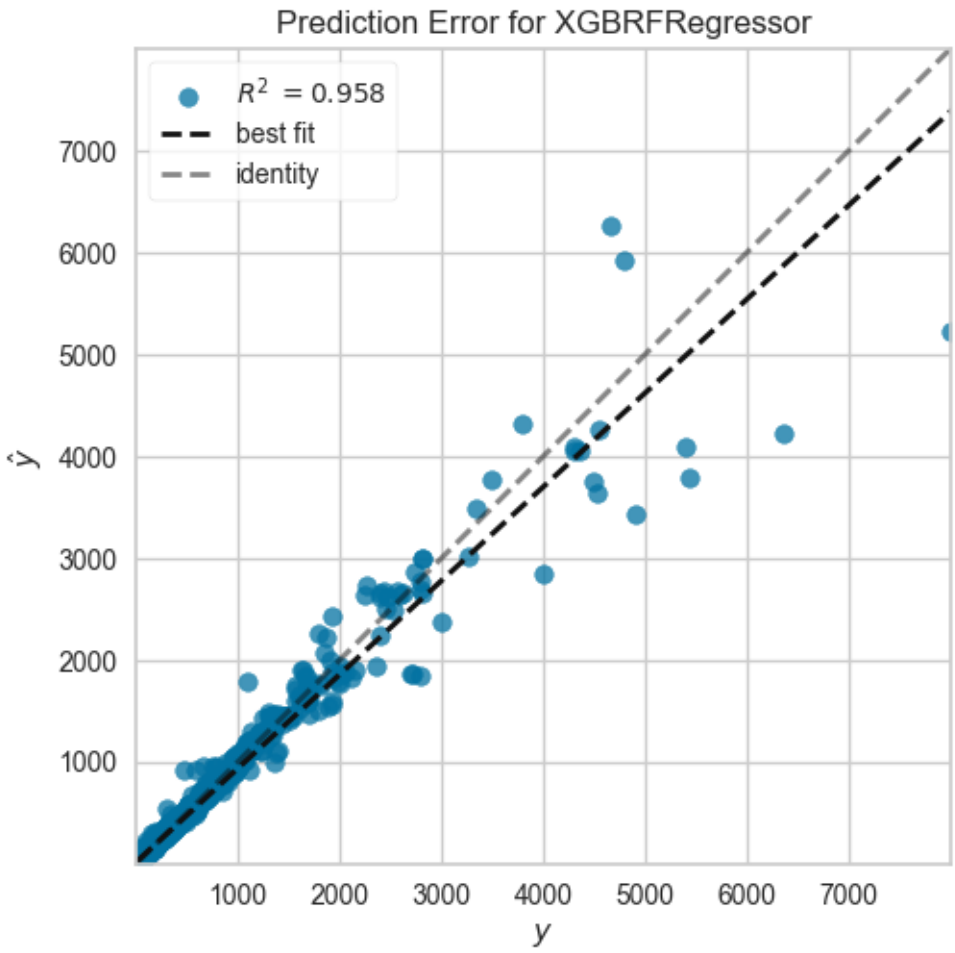


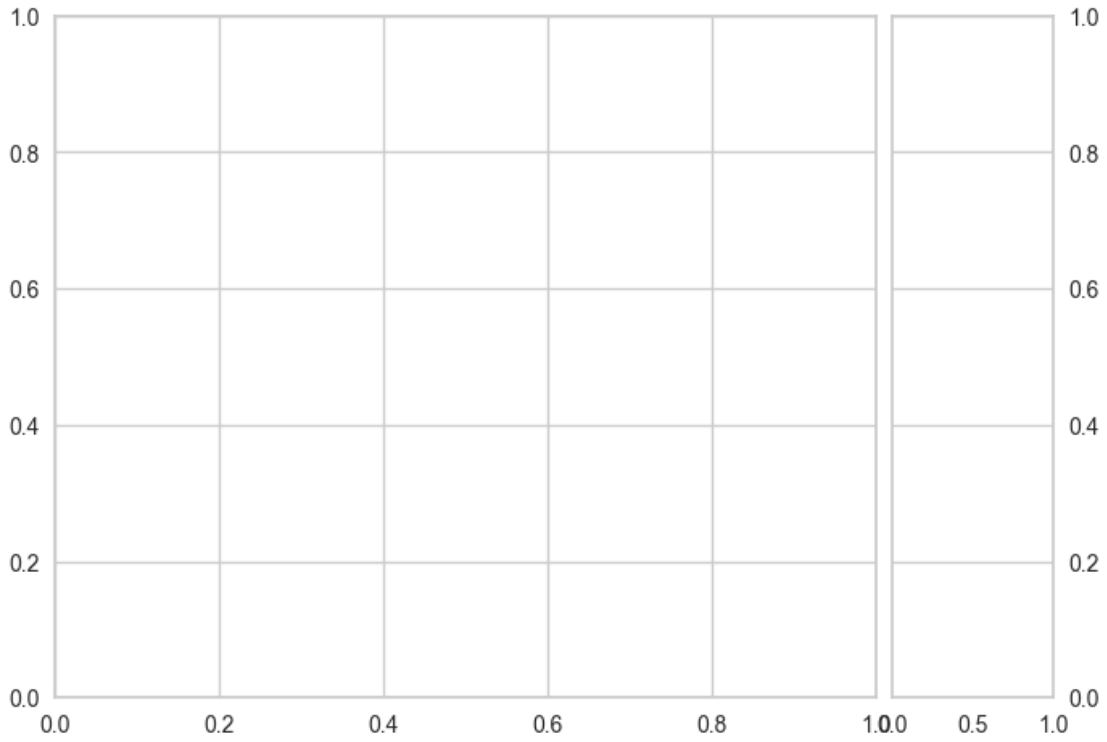
[154]:
```
train_and_evaluate_model(NGBRegressor())
```

```
[iter 0] loss=7.6922 val_loss=0.0000 scale=1.0000 norm=255.5840
[iter 100] loss=6.3421 val_loss=0.0000 scale=2.0000 norm=90.9302
[iter 200] loss=5.3874 val_loss=0.0000 scale=2.0000 norm=35.4227
[iter 300] loss=4.4015 val_loss=0.0000 scale=2.0000 norm=28.3314
[iter 400] loss=3.7229 val_loss=0.0000 scale=2.0000 norm=24.6798
Mean Absolute Error: 19.90904591492414
Mean Squared Error: 13592.158649983363
Root Mean Squared Error: 116.58541353867285
Mean Absolute Percentage Error: 0.1683398921671155
R2 Score: 0.9494263368378026
Training Time: 16.19578266143799
```

[155]:
```
train_and_evaluate_model(XGBRFRegressor())
```

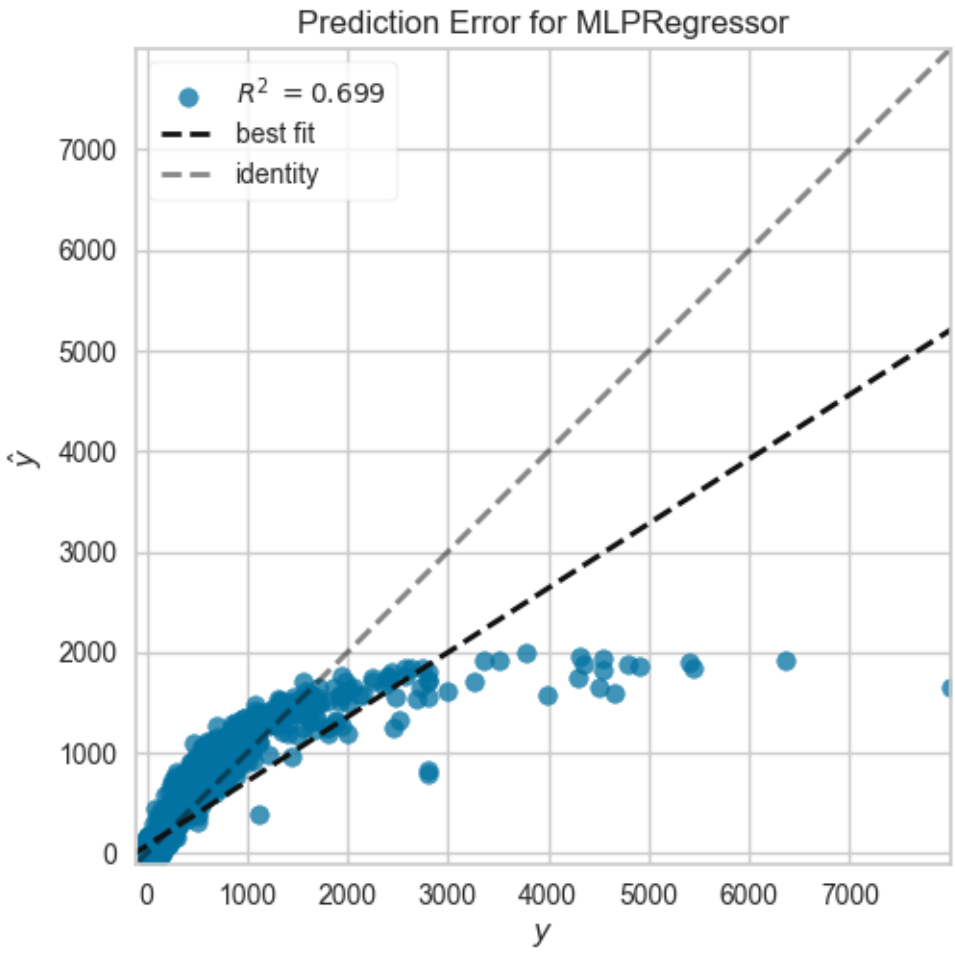
```
Mean Absolute Error: 21.889506133542397
Mean Squared Error: 11377.995442041978
Root Mean Squared Error: 106.6676869630254
Mean Absolute Percentage Error: 0.32693641229313736
R2 Score: 0.9576647886649298
Training Time: 0.261350154876709
```

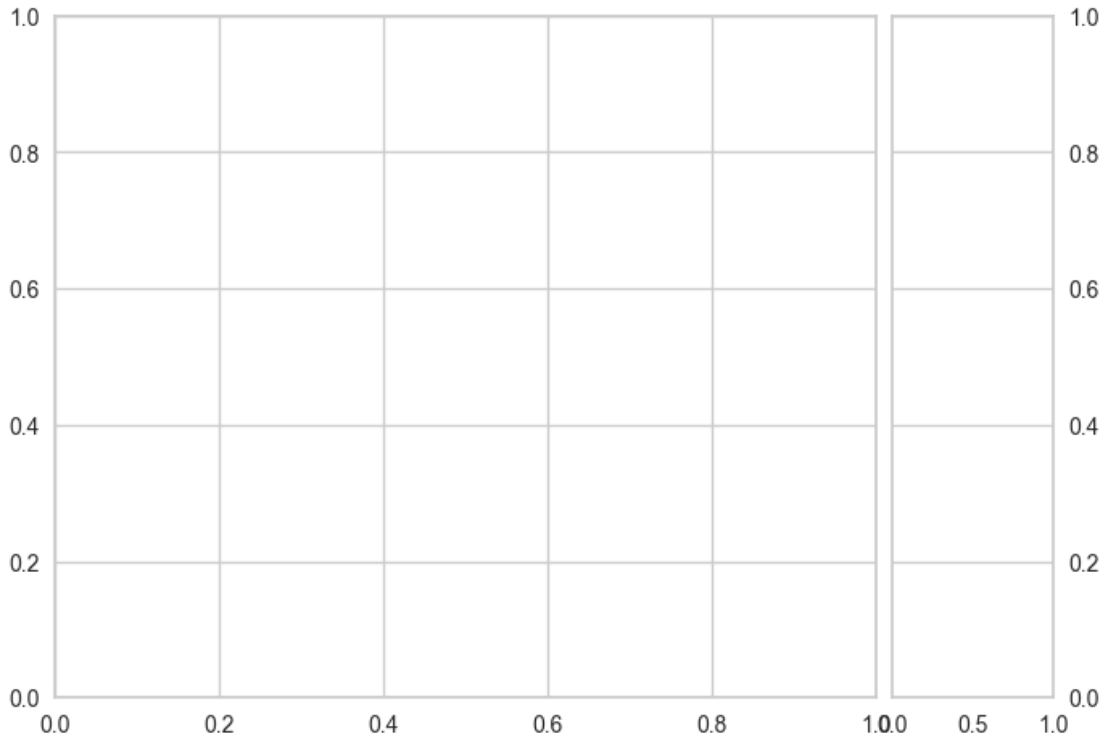

Prediction Error for XGBRFRegressor
R^2 = 0.958
best fit
identity
7000
6000
5000
4000
3000
2000
1000
$\hat{y}$
1000
2000
3000
4000
5000
6000
7000
y



[156]:
```
train_and_evaluate_model(MLPRegressor())
```

```
Mean Absolute Error: 108.39773640230997
Mean Squared Error: 80946.67987005829
Root Mean Squared Error: 284.511300074458
Mean Absolute Percentage Error: 2.3932217522358696
R2 Score: 0.6988138361781437
Training Time: 4.646028757095337
```


Prediction Error for MLPRegressor
$R^2 = 0.699$
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



0.7 Baseline Models Performance Comparison

[157]:
```
model_perfs = pd.DataFrame({'Model': models,
                            'R2': r2_scores,
                            'MAE': mape_scores,
                            'MSE': mse_scores,
                            'RMSE': rmse_scores,
                            'Training Time': training_times}).
 ↪sort_values('R2',ascending=False).reset_index(drop=True)
model_perfs
```

[157]:

	Model	R2	MAE	\
0	(ExtraTreeRegressor(random_state=1029196382), …	0.976337	0.014177	
1	(DecisionTreeRegressor(max_features=1.0, rando…	0.961060	0.019037	
2	<catboost.core.CatBoostRegressor object at 0x0…	0.960660	0.119310	
3	(DecisionTreeRegressor(random_state=2064953117…	0.960326	0.022796	
4	LGBMRegressor()	0.959196	0.105570	
5	XGBRFRegressor(base_score=None, booster=None, …	0.957665	0.326936	
6	DecisionTreeRegressor()	0.950919	0.029598	
7	HistGradientBoostingRegressor()	0.950026	0.100493	
8	([DecisionTreeRegressor(criterion='friedman_ms…	0.949795	0.188744	
9	NGBRegressor(random_state=RandomState(MT19937)…	0.949426	0.168340	
10	XGBRegressor(base_score=None, booster=None, ca…	0.936781	0.083650	

11	PoissonRegressor()	0.922972	0.275781
12	MLPRegressor()	0.698814	2.393222
13	KNeighborsRegressor()	0.621294	0.300699
14	BayesianRidge()	0.377960	11.060483
15	ARDRegression()	0.377954	11.072742
16	Lasso(alpha=0.01)	0.377895	11.132392
17	Ridge(alpha=0.001)	0.377890	11.133864
18	LinearRegression()	0.377890	11.133868
19	SGDRegressor()	0.376253	12.399212
20	(DecisionTreeRegressor(max_depth=3, random_sta…	0.367832	24.094042
21	GammaRegressor()	0.362304	0.864701
22	ElasticNet(l1_ratio=0.3)	0.335042	6.127735
23	TweedieRegressor()	0.316624	5.287976
24	TheilSenRegressor()	0.249958	5.212216
25	HuberRegressor()	0.221991	3.857416
26	PassiveAggressiveRegressor()	0.190476	3.579510
27	LinearSVR()	0.179520	3.089730
28	NuSVR()	0.148038	1.287698
29	SVR()	0.132530	0.727805
30	RANSACRegressor()	0.086362	2.196420

	MSE	RMSE	Training Time
0	6359.725944	79.747890	1.895131
1	10465.381975	102.300450	4.487048
2	10572.891102	102.824565	4.655082
3	10662.775436	103.260716	0.454965
4	10966.364507	104.720411	0.207914
5	11377.995442	106.667687	0.261350
6	13191.118209	114.852593	0.062676
7	13431.093179	115.892593	0.565095
8	13493.017227	116.159447	1.191292
9	13592.158650	116.585414	16.195783
10	16990.802533	130.348773	0.278330
11	20701.894732	143.881530	0.012346
12	80946.679870	284.511300	4.646029
13	101780.862196	319.031130	0.015630
14	167179.355226	408.875721	0.133629
15	167180.768735	408.877450	0.240238
16	167196.717014	408.896952	0.012718
17	167198.142136	408.898694	0.002998
18	167198.143324	408.898696	0.180946
19	167638.069688	409.436283	0.011341
20	169901.101254	412.190613	0.378444
21	171386.886866	413.988994	0.009127
22	178713.970670	422.745752	0.007821
23	183663.881951	428.560243	0.010034
24	201580.964003	448.977688	2.205856

```
25  209097.502773  457.271804       0.053121
26  217567.345413  466.441149       0.008995
27  220511.775497  469.586814       0.008999
28  228972.929215  478.511159       1.584551
29  233141.037286  482.846805       2.899281
30  245549.038670  495.529049       0.183812
```

The Extra Trees Regressor is the best performing model which achieved an incredible r2 score of more than 97% on the test set. It's closely followed by the Decision Trees Regressor model which produced an amazing r2 score of approximately 96.2%.

0.8 Cross Validation & Hyperparameter Tuning

[158]:
```
print(f"Mean cross validation training R2 score of Extra Trees Regressor:␣
↪{round(np.mean(cross_val_score(model_perfs['Model'].
↪iloc[0],final_X_train,y_train,cv=5,scoring='r2',verbose=0))*100,2)}%")
```

```
Mean cross validation training R2 score of Extra Trees Regressor: 96.07%
```

[159]:
```
print(f"Mean cross validation test R2 score of Extra Trees Regressor: {round(np.
↪mean(cross_val_score(model_perfs['Model'].
↪iloc[0],final_X_test,y_test,cv=5,scoring='r2',verbose=0))*100,2)}%")
```

```
Mean cross validation test R2 score of Extra Trees Regressor: 95.59%
```

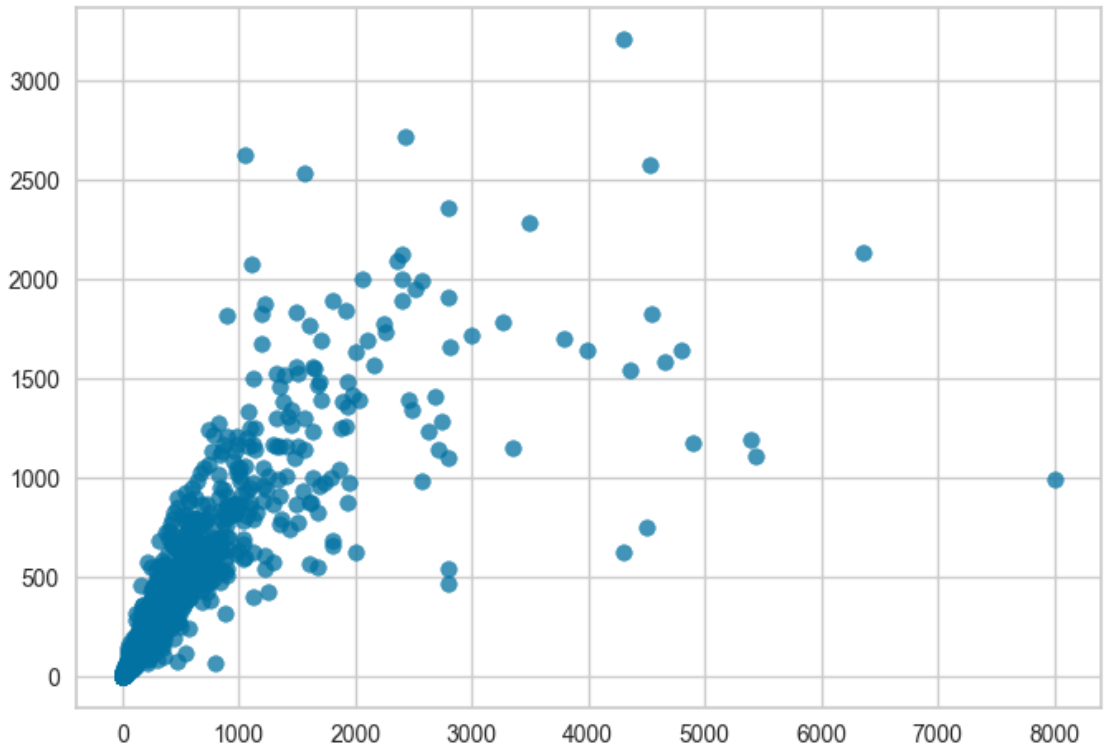
The cross validation R2 score of the Extra Trees Regressor model on the test set is comparable to the training cross validation R2 score so there seems to be no overfitting.

[160]:
```
param_grid = {'n_neighbors': [2,8,12,20],
              'weights': ['uniform','distance'],
              'algorithm': ['ball_tree', 'brute', 'kd_tree'],
              'metric': ['minkowski','manhattan','euclidean','chebyshev'],
              'p': [1,2]
             }

grid_knn = RandomizedSearchCV(KNeighborsRegressor(),param_grid,cv=5,verbose=2)
train_and_evaluate_model(grid_knn)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END algorithm=brute, metric=minkowski, n_neighbors=8, p=1,
weights=distance; total time=   0.1s
[CV] END algorithm=brute, metric=minkowski, n_neighbors=8, p=1,
weights=distance; total time=   0.0s
[CV] END algorithm=brute, metric=minkowski, n_neighbors=8, p=1,
weights=distance; total time=   0.0s
[CV] END algorithm=brute, metric=minkowski, n_neighbors=8, p=1,
weights=distance; total time=   0.0s
[CV] END algorithm=brute, metric=minkowski, n_neighbors=8, p=1,
weights=distance; total time=   0.0s
[CV] END algorithm=kd_tree, metric=minkowski, n_neighbors=12, p=1,
```

```
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=minkowski, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=minkowski, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=minkowski, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=minkowski, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=euclidean, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=euclidean, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=euclidean, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=euclidean, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=euclidean, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, p=2,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, p=2,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, p=2,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, p=2,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, p=2,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=manhattan, n_neighbors=20, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=manhattan, n_neighbors=20, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=manhattan, n_neighbors=20, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=manhattan, n_neighbors=20, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=manhattan, n_neighbors=20, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=brute, metric=minkowski, n_neighbors=12, p=2,
weights=distance; total time=   0.0s
[CV] END algorithm=brute, metric=minkowski, n_neighbors=12, p=2,
weights=distance; total time=   0.0s
[CV] END algorithm=brute, metric=minkowski, n_neighbors=12, p=2,
weights=distance; total time=   0.0s
[CV] END algorithm=brute, metric=minkowski, n_neighbors=12, p=2,
weights=distance; total time=   0.0s
[CV] END algorithm=brute, metric=minkowski, n_neighbors=12, p=2,
```

```
weights=distance; total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=8, p=1,
weights=distance; total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=8, p=1,
weights=distance; total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=8, p=1,
weights=distance; total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=8, p=1,
weights=distance; total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=8, p=1,
weights=distance; total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=20, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=20, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=20, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=20, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=20, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=kd_tree, metric=euclidean, n_neighbors=12, p=2,
weights=distance; total time=   0.1s
[CV] END algorithm=kd_tree, metric=euclidean, n_neighbors=12, p=2,
weights=distance; total time=   0.1s
[CV] END algorithm=kd_tree, metric=euclidean, n_neighbors=12, p=2,
weights=distance; total time=   0.1s
[CV] END algorithm=kd_tree, metric=euclidean, n_neighbors=12, p=2,
weights=distance; total time=   0.1s
[CV] END algorithm=kd_tree, metric=euclidean, n_neighbors=12, p=2,
weights=distance; total time=   0.1s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=12, p=1,
weights=uniform; total time=   0.0s
Mean Absolute Error: 71.65283456492334
Mean Squared Error: 93634.44927347753
Root Mean Squared Error: 305.99746612264215
Mean Absolute Percentage Error: 0.2875659800825623
R2 Score: 0.6516052218136443
Training Time: 4.524564027786255
```

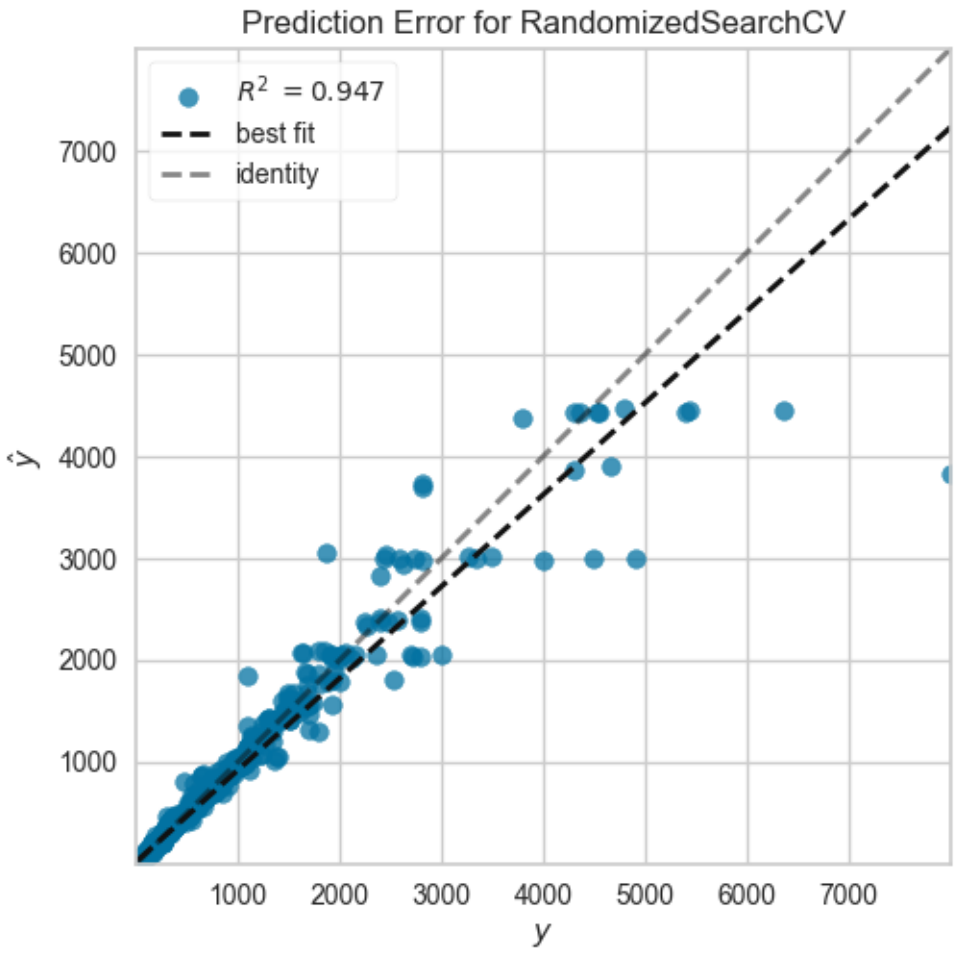


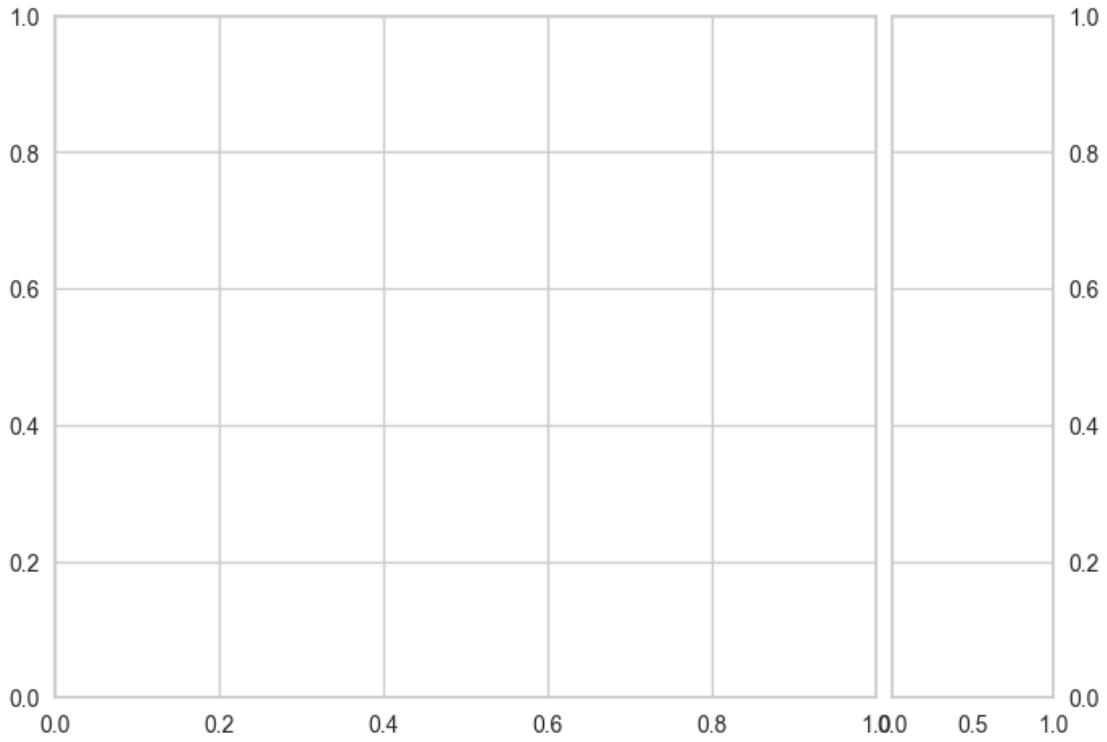
[161]:
```
param_grid = {'learning_rate': [0.2,0.4,0.5,0.8,1.0],
              'loss': ['squared_error','absolute_error','poisson','quantile'],
              'max_bins': np.arange(0,255,50),
              'interaction_cst': ['pairwise','no_interaction']}

grid_hgb = ␣
 ↪RandomizedSearchCV(HistGradientBoostingRegressor(),param_grid,cv=5,verbose=2)
train_and_evaluate_model(grid_hgb)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=quantile,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=quantile,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=quantile,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=quantile,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=quantile,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=1.0, loss=quantile,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=1.0, loss=quantile,
```

```
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=1.0, loss=quantile,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=1.0, loss=quantile,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=1.0, loss=quantile,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.4, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.4, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.4, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.4, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.4, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.8, loss=squared_error,
max_bins=100; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.8, loss=squared_error,
max_bins=100; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.8, loss=squared_error,
max_bins=100; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.8, loss=squared_error,
max_bins=100; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.8, loss=squared_error,
max_bins=100; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=quantile,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=quantile,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=quantile,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=quantile,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=quantile,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=absolute_error,
```

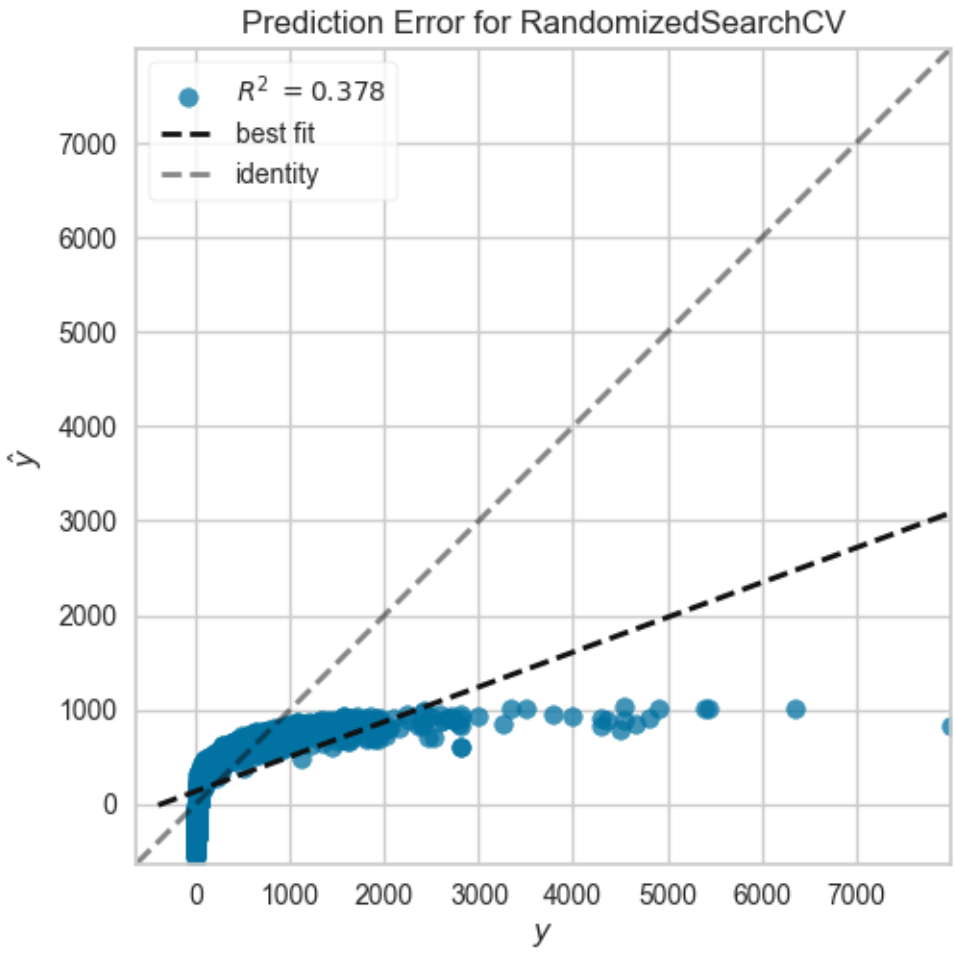
```
max_bins=150; total time=   0.9s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=absolute_error,
max_bins=150; total time=   1.3s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=absolute_error,
max_bins=150; total time=   0.6s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=absolute_error,
max_bins=150; total time=   0.8s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=absolute_error,
max_bins=150; total time=   1.3s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.4, loss=absolute_error,
max_bins=200; total time=   0.8s
[CV] END interaction_cst=pairwise, learning_rate=0.4, loss=absolute_error,
max_bins=200; total time=   1.2s
[CV] END interaction_cst=pairwise, learning_rate=0.4, loss=absolute_error,
max_bins=200; total time=   1.0s
[CV] END interaction_cst=pairwise, learning_rate=0.4, loss=absolute_error,
max_bins=200; total time=   0.7s
[CV] END interaction_cst=pairwise, learning_rate=0.4, loss=absolute_error,
max_bins=200; total time=   1.7s
[CV] END interaction_cst=pairwise, learning_rate=1.0, loss=squared_error,
max_bins=250; total time=   0.3s
[CV] END interaction_cst=pairwise, learning_rate=1.0, loss=squared_error,
max_bins=250; total time=   0.6s
[CV] END interaction_cst=pairwise, learning_rate=1.0, loss=squared_error,
max_bins=250; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=1.0, loss=squared_error,
max_bins=250; total time=   0.6s
[CV] END interaction_cst=pairwise, learning_rate=1.0, loss=squared_error,
max_bins=250; total time=   0.6s
Mean Absolute Error: 19.09897761697773
Mean Squared Error: 14334.490768016569
Root Mean Squared Error: 119.72673372316046
Mean Absolute Percentage Error: 0.07168080706910085
R2 Score: 0.9466642697181742
Training Time: 15.922370910644531
```

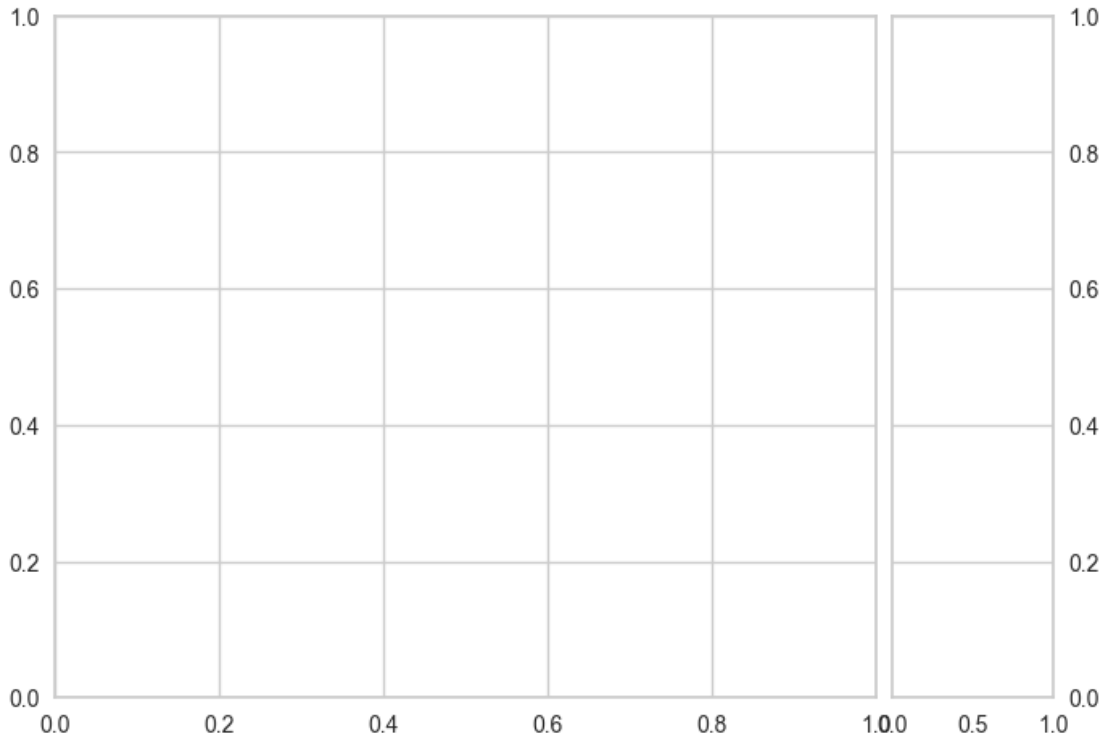

Prediction Error for RandomizedSearchCV
R^2 = 0.947
best fit
identity
7000
6000
5000
4000
3000
2000
1000
$\hat{y}$
1000
2000
3000
4000
5000
6000
7000
y



[162]:
```
param_grid = {'loss': ['epsilon_insensitive','squared_epsilon_insensitive'],
              'C': [0.0001,0.001,0.01,0.1,1],
              'epsilon': np.linspace(0.001,1,5)}

grid_lsvr = RandomizedSearchCV(LinearSVR(),param_grid,cv=5,verbose=2,n_jobs=10)
train_and_evaluate_model(grid_lsvr)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Mean Absolute Error: 213.51384394076788
Mean Squared Error: 167184.22055764636
Root Mean Squared Error: 408.8816706061136
Mean Absolute Percentage Error: 11.136663604608701
R2 Score: 0.37794145328584217
Training Time: 59.68140125274658
```


Prediction Error for RandomizedSearchCV
R^2 = 0.378
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y

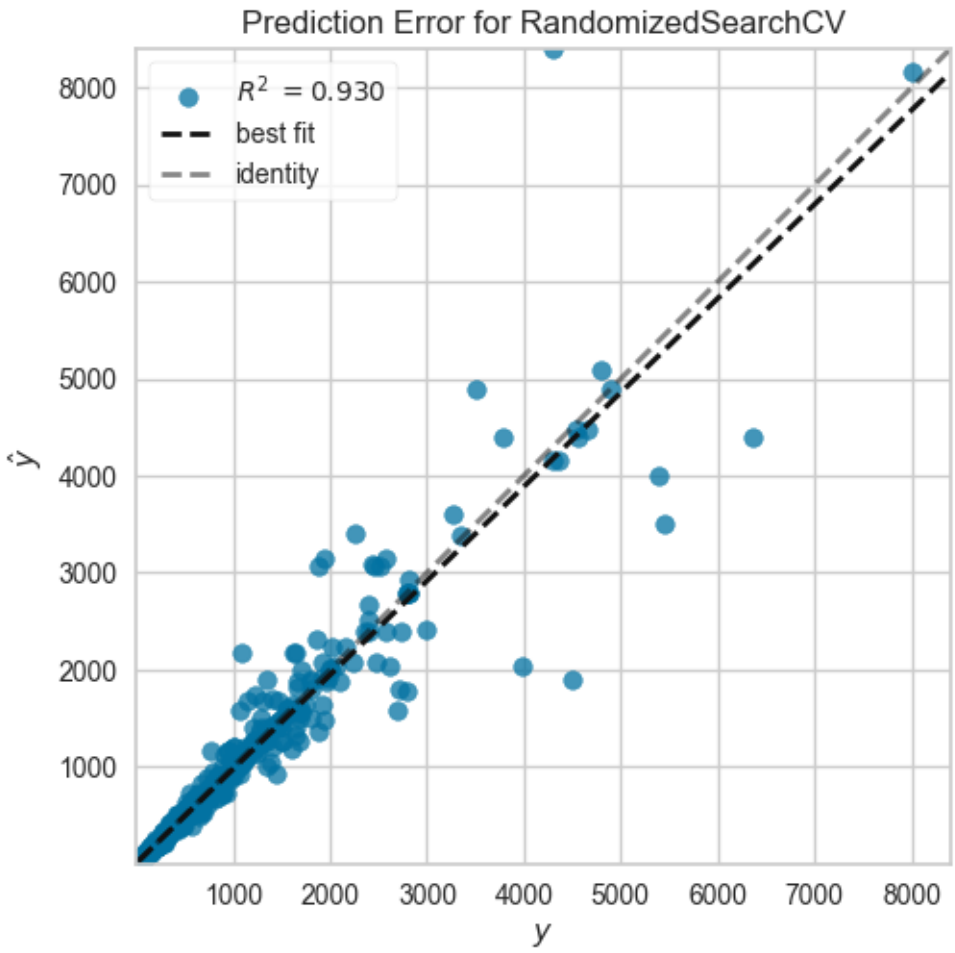


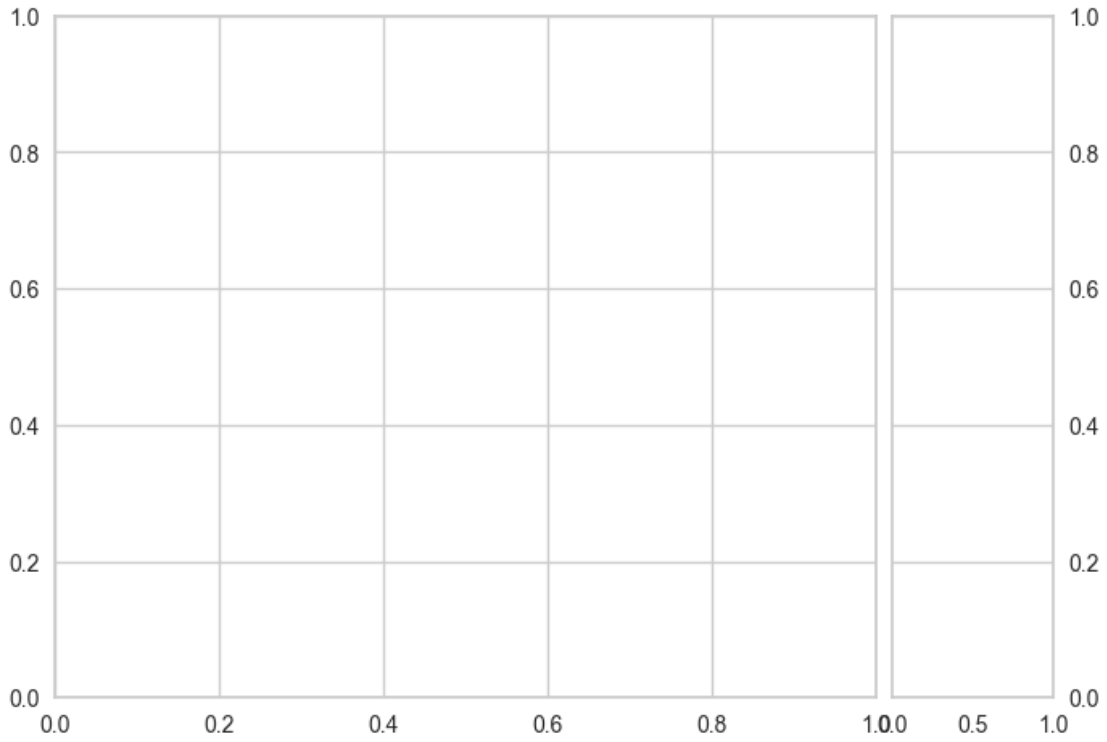
[163]:
```
param_grid = {'criterion':␣
 ↪['squared_error','friedman_mse','absolute_error','poisson'],
              'splitter': ['best','random'],
              'max_features': ['auto','sqrt','log2']
             }

grid_dt = RandomizedSearchCV(DecisionTreeRegressor(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_dt)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.4s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.4s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.4s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.8s
[CV] END criterion=friedman_mse, max_features=sqrt, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=sqrt, splitter=random; total time=
```

```
0.0s
[CV] END criterion=friedman_mse, max_features=sqrt, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=sqrt, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=sqrt, splitter=random; total time=
0.0s
[CV] END criterion=poisson, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=poisson, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=poisson, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=poisson, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=poisson, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=log2, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=log2, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=log2, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=log2, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=log2, splitter=random; total time=
0.0s
[CV] END criterion=squared_error, max_features=sqrt, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=sqrt, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=sqrt, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=sqrt, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=sqrt, splitter=random; total
time=   0.0s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.4s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.4s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.5s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.6s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.4s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=random; total
```

```
time=   0.1s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=random; total
time=   0.4s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=random; total
time=   0.3s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=random; total
time=   0.5s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=random; total
time=   0.2s
[CV] END criterion=squared_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=friedman_mse, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=poisson, max_features=log2, splitter=random; total time=
0.0s
[CV] END criterion=poisson, max_features=log2, splitter=random; total time=
0.0s
[CV] END criterion=poisson, max_features=log2, splitter=random; total time=
0.0s
[CV] END criterion=poisson, max_features=log2, splitter=random; total time=
0.0s
[CV] END criterion=poisson, max_features=log2, splitter=random; total time=
0.0s
Mean Absolute Error: 20.00465809142476
Mean Squared Error: 18876.17098466418
Root Mean Squared Error: 137.39057822377842
Mean Absolute Percentage Error: 0.030517763205018887
R2 Score: 0.9297655995818135
Training Time: 8.734698295593262
```


Prediction Error for RandomizedSearchCV
$R^2 = 0.930$
best fit
identity
$\hat{y}$
y
1000
2000
3000
4000
5000
6000
7000
8000

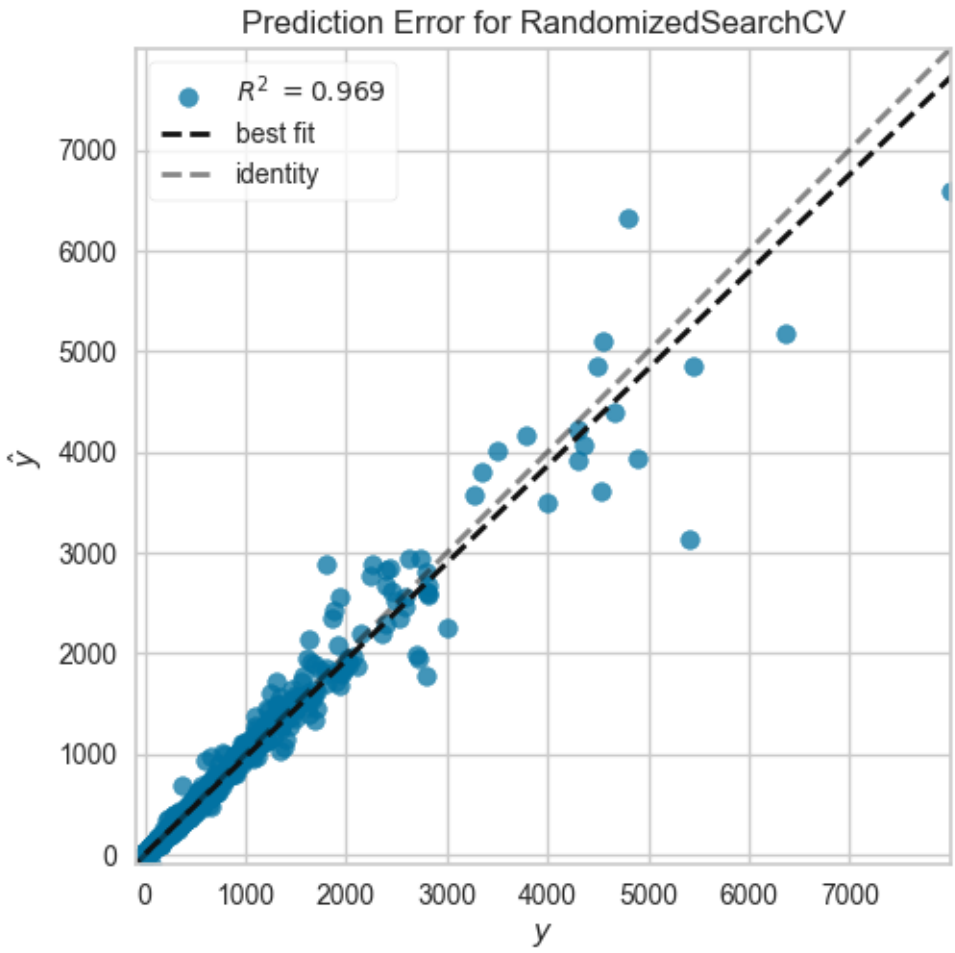


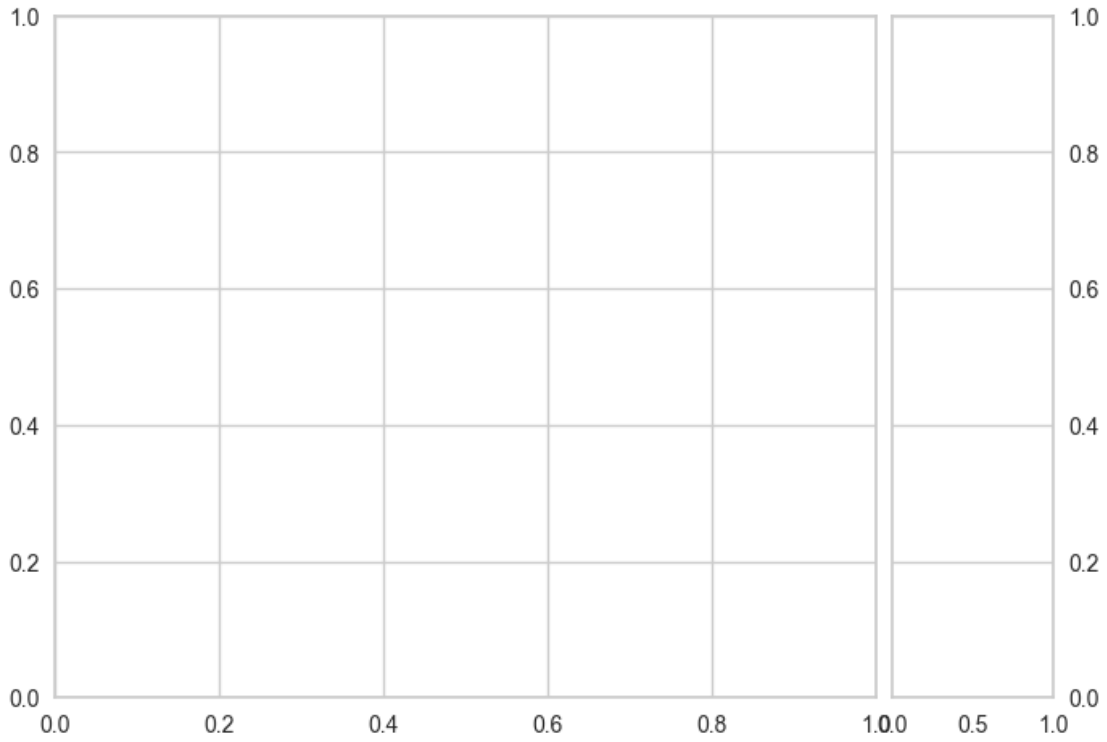
[164]:
```
param_grid =  {'loss': ['squared_error','absolute_error','huber','quantile'],
               'n_estimators': [100,400,700,1000],
               'learning_rate': [0.2,0.4,0.7,1],
               'criterion': ['friedman_mse','squared_error'],
               'max_features': ['auto','sqrt','log2']
               }

grid_gb =␣
 ↪RandomizedSearchCV(GradientBoostingRegressor(),param_grid,verbose=4,cv=5)
train_and_evaluate_model(grid_gb)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END criterion=friedman_mse, learning_rate=0.2, loss=absolute_error,
max_features=log2, n_estimators=700;, score=0.964 total time=   4.3s
[CV 2/5] END criterion=friedman_mse, learning_rate=0.2, loss=absolute_error,
max_features=log2, n_estimators=700;, score=0.943 total time=   5.0s
[CV 3/5] END criterion=friedman_mse, learning_rate=0.2, loss=absolute_error,
max_features=log2, n_estimators=700;, score=0.947 total time=   5.8s
[CV 4/5] END criterion=friedman_mse, learning_rate=0.2, loss=absolute_error,
max_features=log2, n_estimators=700;, score=0.922 total time=   4.9s
[CV 5/5] END criterion=friedman_mse, learning_rate=0.2, loss=absolute_error,
max_features=log2, n_estimators=700;, score=0.952 total time=   5.9s
[CV 1/5] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
```

```
max_features=auto, n_estimators=400;, score=0.960 total time=   5.9s
[CV 2/5] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
max_features=auto, n_estimators=400;, score=0.934 total time=   5.8s
[CV 3/5] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
max_features=auto, n_estimators=400;, score=0.984 total time=   5.0s
[CV 4/5] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
max_features=auto, n_estimators=400;, score=0.949 total time=   5.8s
[CV 5/5] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
max_features=auto, n_estimators=400;, score=0.933 total time=   4.9s
[CV 1/5] END criterion=friedman_mse, learning_rate=1, loss=huber,
max_features=auto, n_estimators=400;, score=0.954 total time=   7.3s
[CV 2/5] END criterion=friedman_mse, learning_rate=1, loss=huber,
max_features=auto, n_estimators=400;, score=0.798 total time=   8.1s
[CV 3/5] END criterion=friedman_mse, learning_rate=1, loss=huber,
max_features=auto, n_estimators=400;, score=0.967 total time=   7.1s
[CV 4/5] END criterion=friedman_mse, learning_rate=1, loss=huber,
max_features=auto, n_estimators=400;, score=0.897 total time=   7.3s
[CV 5/5] END criterion=friedman_mse, learning_rate=1, loss=huber,
max_features=auto, n_estimators=400;, score=0.941 total time=   7.3s
[CV 1/5] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=log2, n_estimators=400;, score=0.976 total time=   2.2s
[CV 2/5] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=log2, n_estimators=400;, score=0.953 total time=   1.4s
[CV 3/5] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=log2, n_estimators=400;, score=0.969 total time=   1.8s
[CV 4/5] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=log2, n_estimators=400;, score=0.958 total time=   1.9s
[CV 5/5] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=log2, n_estimators=400;, score=0.935 total time=   2.4s
[CV 1/5] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
max_features=sqrt, n_estimators=700;, score=0.951 total time=   5.0s
[CV 2/5] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
max_features=sqrt, n_estimators=700;, score=0.968 total time=   5.5s
[CV 3/5] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
max_features=sqrt, n_estimators=700;, score=0.961 total time=   6.2s
[CV 4/5] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
max_features=sqrt, n_estimators=700;, score=0.911 total time=   6.4s
[CV 5/5] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
max_features=sqrt, n_estimators=700;, score=0.958 total time=   4.7s
[CV 1/5] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
max_features=auto, n_estimators=700;, score=0.945 total time=  10.3s
[CV 2/5] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
max_features=auto, n_estimators=700;, score=0.909 total time=  11.1s
[CV 3/5] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
max_features=auto, n_estimators=700;, score=0.949 total time=  10.8s
[CV 4/5] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
max_features=auto, n_estimators=700;, score=0.900 total time=  11.1s
[CV 5/5] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
```

```
max_features=auto, n_estimators=700;, score=0.897 total time=  10.3s
[CV 1/5] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.785 total time=   3.3s
[CV 2/5] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.918 total time=   3.2s
[CV 3/5] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.951 total time=   3.4s
[CV 4/5] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.843 total time=   3.2s
[CV 5/5] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.887 total time=   3.9s
[CV 1/5] END criterion=squared_error, learning_rate=1, loss=absolute_error,
max_features=sqrt, n_estimators=1000;, score=0.872 total time=   8.0s
[CV 2/5] END criterion=squared_error, learning_rate=1, loss=absolute_error,
max_features=sqrt, n_estimators=1000;, score=0.824 total time=   7.7s
[CV 3/5] END criterion=squared_error, learning_rate=1, loss=absolute_error,
max_features=sqrt, n_estimators=1000;, score=0.969 total time=   8.5s
[CV 4/5] END criterion=squared_error, learning_rate=1, loss=absolute_error,
max_features=sqrt, n_estimators=1000;, score=0.953 total time=   8.4s
[CV 5/5] END criterion=squared_error, learning_rate=1, loss=absolute_error,
max_features=sqrt, n_estimators=1000;, score=0.898 total time=   8.4s
[CV 1/5] END criterion=friedman_mse, learning_rate=1, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.921 total time=   1.0s
[CV 2/5] END criterion=friedman_mse, learning_rate=1, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.906 total time=   1.8s
[CV 3/5] END criterion=friedman_mse, learning_rate=1, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.963 total time=   0.9s
[CV 4/5] END criterion=friedman_mse, learning_rate=1, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.953 total time=   1.8s
[CV 5/5] END criterion=friedman_mse, learning_rate=1, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.869 total time=   1.0s
[CV 1/5] END criterion=squared_error, learning_rate=0.2, loss=squared_error,
max_features=log2, n_estimators=400;, score=0.973 total time=   1.3s
[CV 2/5] END criterion=squared_error, learning_rate=0.2, loss=squared_error,
max_features=log2, n_estimators=400;, score=0.933 total time=   2.2s
[CV 3/5] END criterion=squared_error, learning_rate=0.2, loss=squared_error,
max_features=log2, n_estimators=400;, score=0.968 total time=   1.5s
[CV 4/5] END criterion=squared_error, learning_rate=0.2, loss=squared_error,
max_features=log2, n_estimators=400;, score=0.960 total time=   1.9s
[CV 5/5] END criterion=squared_error, learning_rate=0.2, loss=squared_error,
max_features=log2, n_estimators=400;, score=0.976 total time=   2.5s
Mean Absolute Error: 21.98267635185396
Mean Squared Error: 8284.362773282255
Root Mean Squared Error: 91.01847490088073
Mean Absolute Percentage Error: 0.20159016879956287
R2 Score: 0.9691755678256495
Training Time: 262.9485516548157
```


Prediction Error for RandomizedSearchCV
R^2 = 0.969
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y

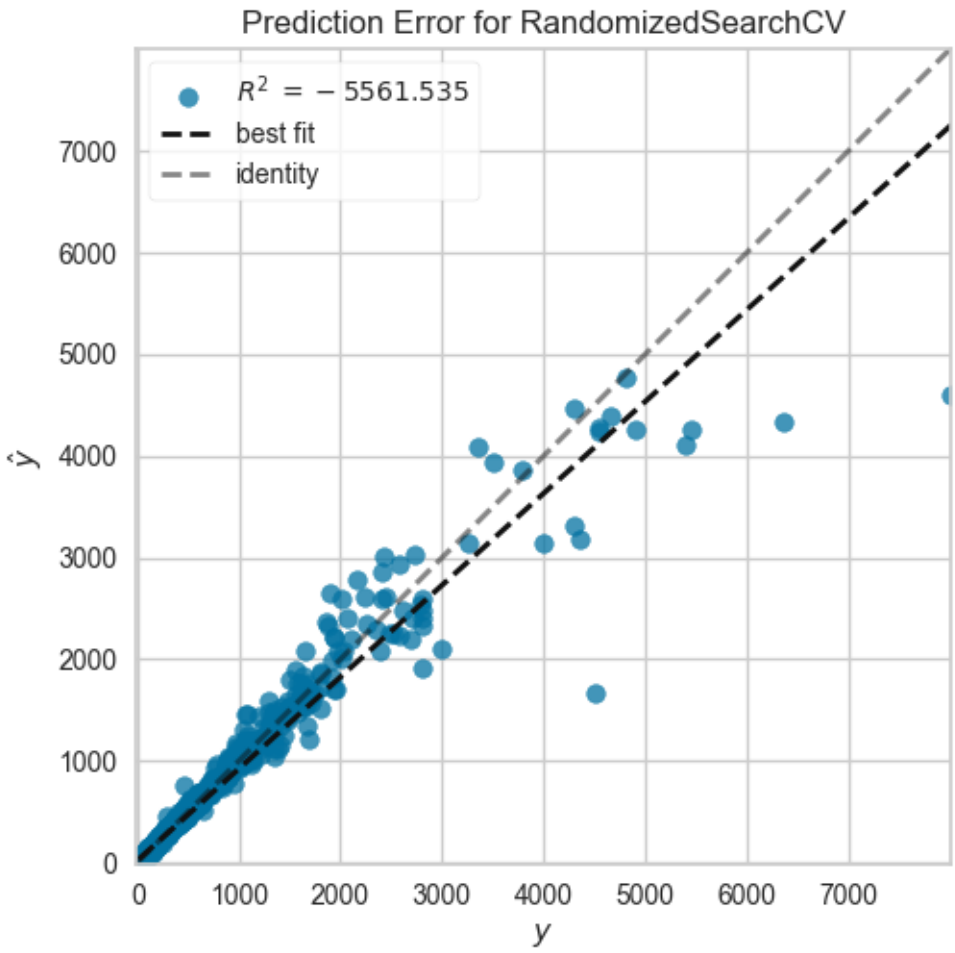


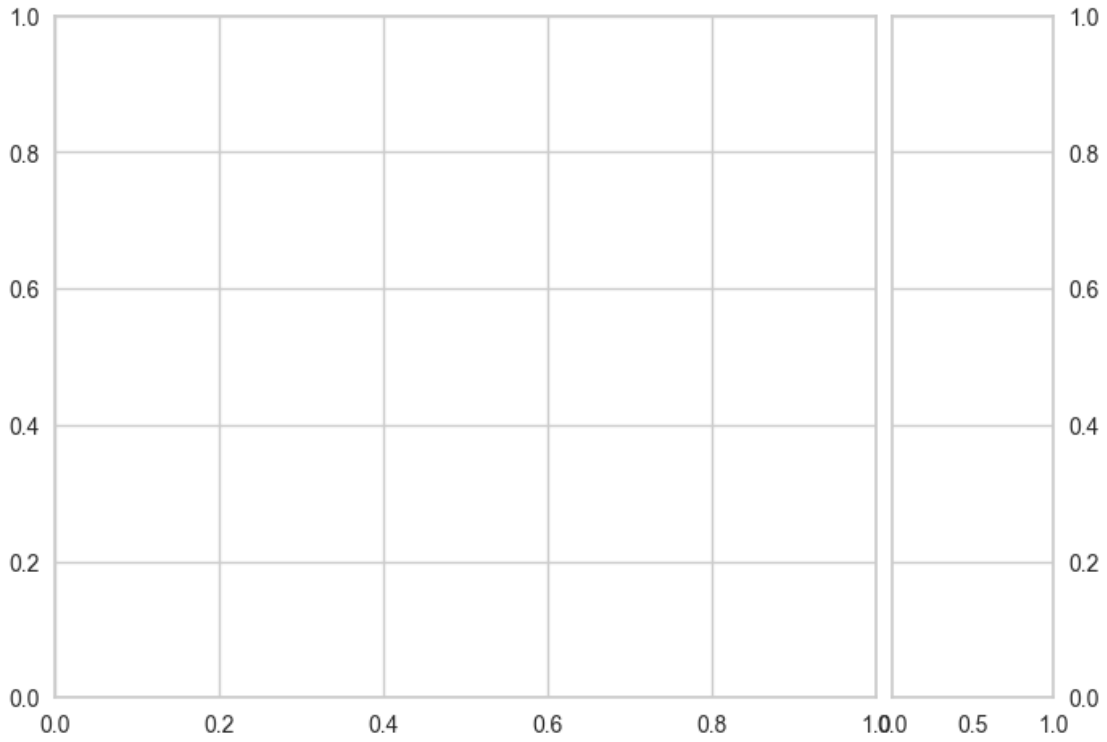
[165]:
```
param_grid = {'learning_rate': [0.2,0.4,0.5,0.7,1],
'n_estimators': [100,400,700,800,1000]
}

grid_cat =␣
  ↪RandomizedSearchCV(CatBoostRegressor(silent=True),param_grid,verbose=5,cv=5)
train_and_evaluate_model(grid_cat)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END learning_rate=0.5, n_estimators=800;, score=-2757.847 total time=
3.4s
[CV 2/5] END learning_rate=0.5, n_estimators=800;, score=-2592.367 total time=
2.4s
[CV 3/5] END learning_rate=0.5, n_estimators=800;, score=-2315.575 total time=
3.5s
[CV 4/5] END learning_rate=0.5, n_estimators=800;, score=-2898.043 total time=
2.5s
[CV 5/5] END learning_rate=0.5, n_estimators=800;, score=-3009.212 total time=
4.4s
[CV 1/5] END learning_rate=0.5, n_estimators=100;, score=-2753.978 total time=
0.7s
[CV 2/5] END learning_rate=0.5, n_estimators=100;, score=-2589.350 total time=
0.7s
```

```
[CV 3/5] END learning_rate=0.5, n_estimators=100;, score=-2314.087 total time=
0.9s
[CV 4/5] END learning_rate=0.5, n_estimators=100;, score=-2892.918 total time=
0.9s
[CV 5/5] END learning_rate=0.5, n_estimators=100;, score=-3001.728 total time=
0.9s
[CV 1/5] END learning_rate=0.4, n_estimators=400;, score=-2755.052 total time=
1.5s
[CV 2/5] END learning_rate=0.4, n_estimators=400;, score=-2541.720 total time=
2.0s
[CV 3/5] END learning_rate=0.4, n_estimators=400;, score=-2436.039 total time=
1.6s
[CV 4/5] END learning_rate=0.4, n_estimators=400;, score=-2946.224 total time=
1.7s
[CV 5/5] END learning_rate=0.4, n_estimators=400;, score=-3114.339 total time=
2.7s
[CV 1/5] END learning_rate=1, n_estimators=100;, score=-2899.196 total time=
0.7s
[CV 2/5] END learning_rate=1, n_estimators=100;, score=-2746.633 total time=
0.8s
[CV 3/5] END learning_rate=1, n_estimators=100;, score=-2317.243 total time=
0.9s
[CV 4/5] END learning_rate=1, n_estimators=100;, score=-2808.648 total time=
0.8s
[CV 5/5] END learning_rate=1, n_estimators=100;, score=-2869.722 total time=
0.9s
[CV 1/5] END learning_rate=0.2, n_estimators=1000;, score=-2844.557 total time=
3.2s
[CV 2/5] END learning_rate=0.2, n_estimators=1000;, score=-2609.014 total time=
3.3s
[CV 3/5] END learning_rate=0.2, n_estimators=1000;, score=-2497.227 total time=
2.4s
[CV 4/5] END learning_rate=0.2, n_estimators=1000;, score=-2882.528 total time=
3.2s
[CV 5/5] END learning_rate=0.2, n_estimators=1000;, score=-3002.589 total time=
3.0s
[CV 1/5] END learning_rate=0.5, n_estimators=400;, score=-2758.058 total time=
1.8s
[CV 2/5] END learning_rate=0.5, n_estimators=400;, score=-2591.846 total time=
1.4s
[CV 3/5] END learning_rate=0.5, n_estimators=400;, score=-2315.094 total time=
1.6s
[CV 4/5] END learning_rate=0.5, n_estimators=400;, score=-2897.910 total time=
1.4s
[CV 5/5] END learning_rate=0.5, n_estimators=400;, score=-3009.620 total time=
1.6s
[CV 1/5] END learning_rate=0.4, n_estimators=100;, score=-2752.508 total time=
0.7s
```

```
[CV 2/5] END learning_rate=0.4, n_estimators=100;, score=-2537.884 total time=
0.6s
[CV 3/5] END learning_rate=0.4, n_estimators=100;, score=-2435.489 total time=
0.9s
[CV 4/5] END learning_rate=0.4, n_estimators=100;, score=-2946.788 total time=
0.7s
[CV 5/5] END learning_rate=0.4, n_estimators=100;, score=-3107.809 total time=
0.8s
[CV 1/5] END learning_rate=0.7, n_estimators=1000;, score=-2823.702 total time=
2.5s
[CV 2/5] END learning_rate=0.7, n_estimators=1000;, score=-2604.896 total time=
2.9s
[CV 3/5] END learning_rate=0.7, n_estimators=1000;, score=-2463.886 total time=
3.7s
[CV 4/5] END learning_rate=0.7, n_estimators=1000;, score=-2898.326 total time=
3.0s
[CV 5/5] END learning_rate=0.7, n_estimators=1000;, score=-3106.916 total time=
2.6s
[CV 1/5] END learning_rate=1, n_estimators=700;, score=-2902.830 total time=
2.1s
[CV 2/5] END learning_rate=1, n_estimators=700;, score=-2749.698 total time=
2.4s
[CV 3/5] END learning_rate=1, n_estimators=700;, score=-2319.440 total time=
2.0s
[CV 4/5] END learning_rate=1, n_estimators=700;, score=-2810.855 total time=
3.5s
[CV 5/5] END learning_rate=1, n_estimators=700;, score=-2866.923 total time=
2.1s
[CV 1/5] END learning_rate=0.4, n_estimators=700;, score=-2755.201 total time=
1.8s
[CV 2/5] END learning_rate=0.4, n_estimators=700;, score=-2542.123 total time=
3.6s
[CV 3/5] END learning_rate=0.4, n_estimators=700;, score=-2436.202 total time=
2.0s
[CV 4/5] END learning_rate=0.4, n_estimators=700;, score=-2946.212 total time=
2.0s
[CV 5/5] END learning_rate=0.4, n_estimators=700;, score=-3114.956 total time=
2.3s
Mean Absolute Error: 21.810524291672234
Mean Squared Error: 13507.323308578432
Root Mean Squared Error: 116.22101061588835
Mean Absolute Percentage Error: 0.21254411346167956
R2 Score: 0.9497419919218071
Training Time: 104.70100808143616
```


Prediction Error for RandomizedSearchCV
$R^2 = -5561.535$
best fit
identity
$\hat{y}$
y
0
1000
2000
3000
4000
5000
6000
7000



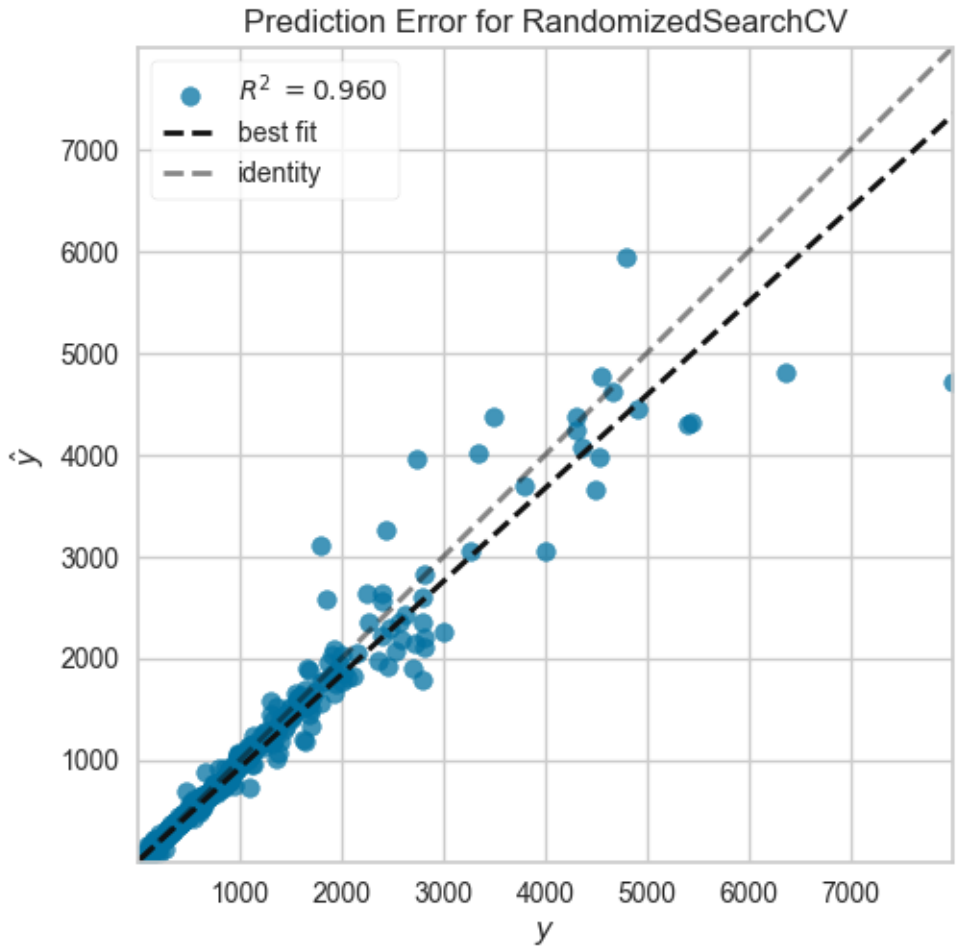
[166]:
```
param_grid = {'boosting_type': ['gbdt','dart','goss','rf'],
              'learning_rate': np.linspace(0,1,6)[1:],
              'n_estimators': [100,300,500,800,1000],
              'importance_type': ['split','gain'],
              'min_split_gain': [0.68,0.79,0.87,1]}

grid_lgbm = RandomizedSearchCV(LGBMRegressor(),param_grid,verbose=3,cv=5)
train_and_evaluate_model(grid_lgbm)
```

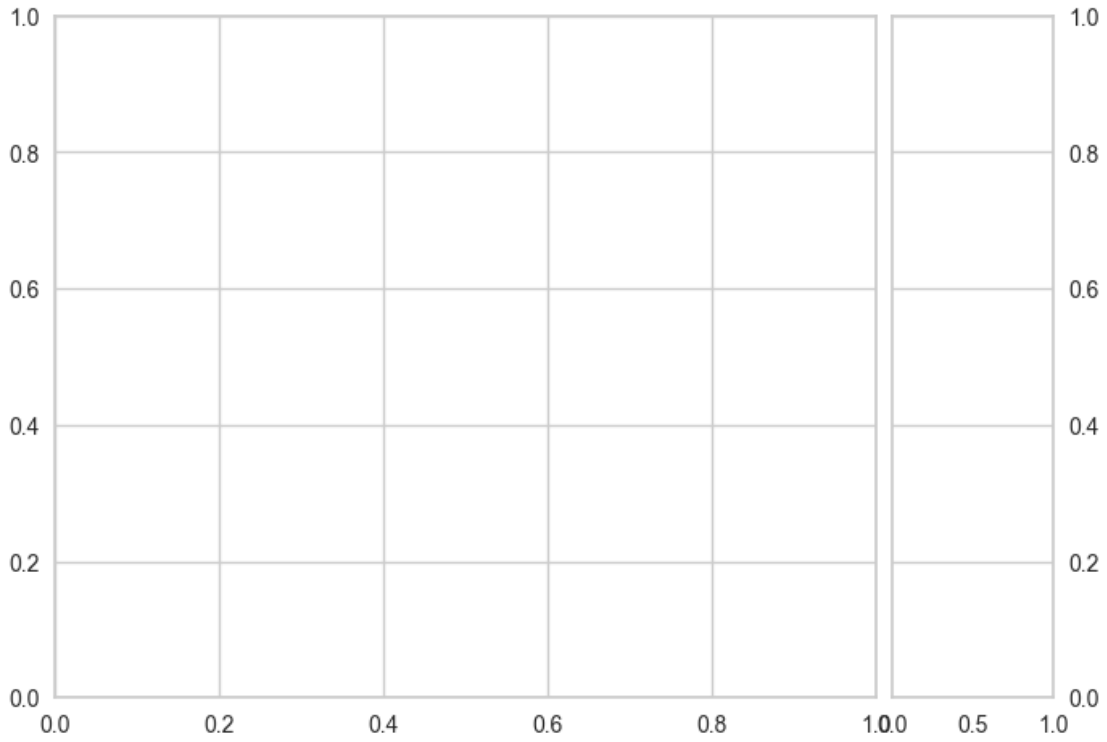
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END boosting_type=rf, importance_type=split, learning_rate=1.0,
min_split_gain=0.68, n_estimators=300;, score=nan total time=   0.1s
[CV 2/5] END boosting_type=rf, importance_type=split, learning_rate=1.0,
min_split_gain=0.68, n_estimators=300;, score=nan total time=   0.0s
[CV 3/5] END boosting_type=rf, importance_type=split, learning_rate=1.0,
min_split_gain=0.68, n_estimators=300;, score=nan total time=   0.0s
[CV 4/5] END boosting_type=rf, importance_type=split, learning_rate=1.0,
min_split_gain=0.68, n_estimators=300;, score=nan total time=   0.0s
[CV 5/5] END boosting_type=rf, importance_type=split, learning_rate=1.0,
min_split_gain=0.68, n_estimators=300;, score=nan total time=   0.0s
[CV 1/5] END boosting_type=dart, importance_type=gain, learning_rate=0.2,
min_split_gain=0.79, n_estimators=100;, score=0.939 total time=   0.0s
[CV 2/5] END boosting_type=dart, importance_type=gain, learning_rate=0.2,
```

```
min_split_gain=0.79, n_estimators=100;, score=0.876 total time=   0.0s
[CV 3/5] END boosting_type=dart, importance_type=gain, learning_rate=0.2,
min_split_gain=0.79, n_estimators=100;, score=0.926 total time=   0.0s
[CV 4/5] END boosting_type=dart, importance_type=gain, learning_rate=0.2,
min_split_gain=0.79, n_estimators=100;, score=0.875 total time=   0.0s
[CV 5/5] END boosting_type=dart, importance_type=gain, learning_rate=0.2,
min_split_gain=0.79, n_estimators=100;, score=0.971 total time=   0.1s
[CV 1/5] END boosting_type=gbdt, importance_type=split, learning_rate=0.8,
min_split_gain=1, n_estimators=500;, score=0.804 total time=   0.8s
[CV 2/5] END boosting_type=gbdt, importance_type=split, learning_rate=0.8,
min_split_gain=1, n_estimators=500;, score=0.818 total time=   0.5s
[CV 3/5] END boosting_type=gbdt, importance_type=split, learning_rate=0.8,
min_split_gain=1, n_estimators=500;, score=0.909 total time=   0.5s
[CV 4/5] END boosting_type=gbdt, importance_type=split, learning_rate=0.8,
min_split_gain=1, n_estimators=500;, score=0.873 total time=   0.6s
[CV 5/5] END boosting_type=gbdt, importance_type=split, learning_rate=0.8,
min_split_gain=1, n_estimators=500;, score=0.915 total time=   0.6s
[CV 1/5] END boosting_type=rf, importance_type=gain, learning_rate=0.8,
min_split_gain=0.87, n_estimators=300;, score=nan total time=   0.0s
[CV 2/5] END boosting_type=rf, importance_type=gain, learning_rate=0.8,
min_split_gain=0.87, n_estimators=300;, score=nan total time=   0.0s
[CV 3/5] END boosting_type=rf, importance_type=gain, learning_rate=0.8,
min_split_gain=0.87, n_estimators=300;, score=nan total time=   0.0s
[CV 4/5] END boosting_type=rf, importance_type=gain, learning_rate=0.8,
min_split_gain=0.87, n_estimators=300;, score=nan total time=   0.0s
[CV 5/5] END boosting_type=rf, importance_type=gain, learning_rate=0.8,
min_split_gain=0.87, n_estimators=300;, score=nan total time=   0.0s
[CV 1/5] END boosting_type=goss, importance_type=gain,
learning_rate=0.6000000000000001, min_split_gain=0.87, n_estimators=300;,
score=0.852 total time=   0.4s
[CV 2/5] END boosting_type=goss, importance_type=gain,
learning_rate=0.6000000000000001, min_split_gain=0.87, n_estimators=300;,
score=0.875 total time=   0.4s
[CV 3/5] END boosting_type=goss, importance_type=gain,
learning_rate=0.6000000000000001, min_split_gain=0.87, n_estimators=300;,
score=0.879 total time=   0.4s
[CV 4/5] END boosting_type=goss, importance_type=gain,
learning_rate=0.6000000000000001, min_split_gain=0.87, n_estimators=300;,
score=0.848 total time=   0.4s
[CV 5/5] END boosting_type=goss, importance_type=gain,
learning_rate=0.6000000000000001, min_split_gain=0.87, n_estimators=300;,
score=0.887 total time=   0.4s
[CV 1/5] END boosting_type=gbdt, importance_type=gain,
learning_rate=0.6000000000000001, min_split_gain=0.87, n_estimators=500;,
score=0.847 total time=   0.6s
[CV 2/5] END boosting_type=gbdt, importance_type=gain,
learning_rate=0.6000000000000001, min_split_gain=0.87, n_estimators=500;,
score=0.839 total time=   0.6s
```

```
[CV 3/5] END boosting_type=gbdt, importance_type=gain,
learning_rate=0.6000000000000001, min_split_gain=0.87, n_estimators=500;,
score=0.905 total time=   0.6s
[CV 4/5] END boosting_type=gbdt, importance_type=gain,
learning_rate=0.6000000000000001, min_split_gain=0.87, n_estimators=500;,
score=0.853 total time=   0.6s
[CV 5/5] END boosting_type=gbdt, importance_type=gain,
learning_rate=0.6000000000000001, min_split_gain=0.87, n_estimators=500;,
score=0.929 total time=   0.7s
[CV 1/5] END boosting_type=gbdt, importance_type=gain, learning_rate=0.2,
min_split_gain=1, n_estimators=800;, score=0.859 total time=   1.3s
[CV 2/5] END boosting_type=gbdt, importance_type=gain, learning_rate=0.2,
min_split_gain=1, n_estimators=800;, score=0.814 total time=   1.2s
[CV 3/5] END boosting_type=gbdt, importance_type=gain, learning_rate=0.2,
min_split_gain=1, n_estimators=800;, score=0.915 total time=   0.4s
[CV 4/5] END boosting_type=gbdt, importance_type=gain, learning_rate=0.2,
min_split_gain=1, n_estimators=800;, score=0.873 total time=   0.5s
[CV 5/5] END boosting_type=gbdt, importance_type=gain, learning_rate=0.2,
min_split_gain=1, n_estimators=800;, score=0.934 total time=   0.8s
[CV 1/5] END boosting_type=goss, importance_type=gain, learning_rate=0.4,
min_split_gain=1, n_estimators=100;, score=0.883 total time=   0.0s
[CV 2/5] END boosting_type=goss, importance_type=gain, learning_rate=0.4,
min_split_gain=1, n_estimators=100;, score=0.862 total time=   0.1s
[CV 3/5] END boosting_type=goss, importance_type=gain, learning_rate=0.4,
min_split_gain=1, n_estimators=100;, score=0.903 total time=   0.0s
[CV 4/5] END boosting_type=goss, importance_type=gain, learning_rate=0.4,
min_split_gain=1, n_estimators=100;, score=0.910 total time=   0.1s
[CV 5/5] END boosting_type=goss, importance_type=gain, learning_rate=0.4,
min_split_gain=1, n_estimators=100;, score=0.925 total time=   0.2s
[CV 1/5] END boosting_type=goss, importance_type=split, learning_rate=1.0,
min_split_gain=0.79, n_estimators=300;, score=0.667 total time=   0.4s
[CV 2/5] END boosting_type=goss, importance_type=split, learning_rate=1.0,
min_split_gain=0.79, n_estimators=300;, score=0.727 total time=   0.5s
[CV 3/5] END boosting_type=goss, importance_type=split, learning_rate=1.0,
min_split_gain=0.79, n_estimators=300;, score=0.743 total time=   0.6s
[CV 4/5] END boosting_type=goss, importance_type=split, learning_rate=1.0,
min_split_gain=0.79, n_estimators=300;, score=0.719 total time=   0.4s
[CV 5/5] END boosting_type=goss, importance_type=split, learning_rate=1.0,
min_split_gain=0.79, n_estimators=300;, score=0.599 total time=   0.5s
[CV 1/5] END boosting_type=rf, importance_type=split, learning_rate=0.2,
min_split_gain=0.79, n_estimators=100;, score=nan total time=   0.0s
[CV 2/5] END boosting_type=rf, importance_type=split, learning_rate=0.2,
min_split_gain=0.79, n_estimators=100;, score=nan total time=   0.0s
[CV 3/5] END boosting_type=rf, importance_type=split, learning_rate=0.2,
min_split_gain=0.79, n_estimators=100;, score=nan total time=   0.0s
[CV 4/5] END boosting_type=rf, importance_type=split, learning_rate=0.2,
min_split_gain=0.79, n_estimators=100;, score=nan total time=   0.0s
[CV 5/5] END boosting_type=rf, importance_type=split, learning_rate=0.2,
```

```
min_split_gain=0.79, n_estimators=100;, score=nan total time=   0.0s
Mean Absolute Error: 19.390607890516893
Mean Squared Error: 10832.651486810042
Root Mean Squared Error: 104.08002443701693
Mean Absolute Percentage Error: 0.12477714262082172
R2 Score: 0.9596939028188817
Training Time: 19.581051349639893
```





[167]:
```
param_grid = {'n_estimators': [100,400,700,1000],
        'grow_policy': [0,1],
        'learning_rate': [0.1,0.4,0.7,1],
        'booster': ['gbtree','gblinear','dart'],
        'sampling_method': ['uniform','gradient_based'],
        'importance_type': ['gain','weight','cover','total_gain','total_cover']
}

grid_xgb = RandomizedSearchCV(XGBRegressor(),param_grid,verbose=3,cv=5)
train_and_evaluate_model(grid_xgb)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END booster=gbtree, grow_policy=0, importance_type=total_gain,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform;, score=nan total
time=   0.2s
[CV 2/5] END booster=gbtree, grow_policy=0, importance_type=total_gain,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 3/5] END booster=gbtree, grow_policy=0, importance_type=total_gain,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 4/5] END booster=gbtree, grow_policy=0, importance_type=total_gain,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform;, score=nan total
```

```
time=   0.0s
[CV 5/5] END booster=gbtree, grow_policy=0, importance_type=total_gain,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 1/5] END booster=dart, grow_policy=0, importance_type=total_gain,
learning_rate=0.7, n_estimators=400, sampling_method=gradient_based;, score=nan
total time=   0.0s
[CV 2/5] END booster=dart, grow_policy=0, importance_type=total_gain,
learning_rate=0.7, n_estimators=400, sampling_method=gradient_based;, score=nan
total time=   0.0s
[CV 3/5] END booster=dart, grow_policy=0, importance_type=total_gain,
learning_rate=0.7, n_estimators=400, sampling_method=gradient_based;, score=nan
total time=   0.0s
[CV 4/5] END booster=dart, grow_policy=0, importance_type=total_gain,
learning_rate=0.7, n_estimators=400, sampling_method=gradient_based;, score=nan
total time=   0.0s
[CV 5/5] END booster=dart, grow_policy=0, importance_type=total_gain,
learning_rate=0.7, n_estimators=400, sampling_method=gradient_based;, score=nan
total time=   0.0s
[CV 1/5] END booster=dart, grow_policy=1, importance_type=total_gain,
learning_rate=0.1, n_estimators=700, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 2/5] END booster=dart, grow_policy=1, importance_type=total_gain,
learning_rate=0.1, n_estimators=700, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 3/5] END booster=dart, grow_policy=1, importance_type=total_gain,
learning_rate=0.1, n_estimators=700, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 4/5] END booster=dart, grow_policy=1, importance_type=total_gain,
learning_rate=0.1, n_estimators=700, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 5/5] END booster=dart, grow_policy=1, importance_type=total_gain,
learning_rate=0.1, n_estimators=700, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 1/5] END booster=gbtree, grow_policy=1, importance_type=weight,
learning_rate=0.7, n_estimators=700, sampling_method=gradient_based;, score=nan
total time=   0.0s
[CV 2/5] END booster=gbtree, grow_policy=1, importance_type=weight,
learning_rate=0.7, n_estimators=700, sampling_method=gradient_based;, score=nan
total time=   0.0s
[CV 3/5] END booster=gbtree, grow_policy=1, importance_type=weight,
learning_rate=0.7, n_estimators=700, sampling_method=gradient_based;, score=nan
total time=   0.0s
[CV 4/5] END booster=gbtree, grow_policy=1, importance_type=weight,
learning_rate=0.7, n_estimators=700, sampling_method=gradient_based;, score=nan
total time=   0.0s
[CV 5/5] END booster=gbtree, grow_policy=1, importance_type=weight,
learning_rate=0.7, n_estimators=700, sampling_method=gradient_based;, score=nan
```

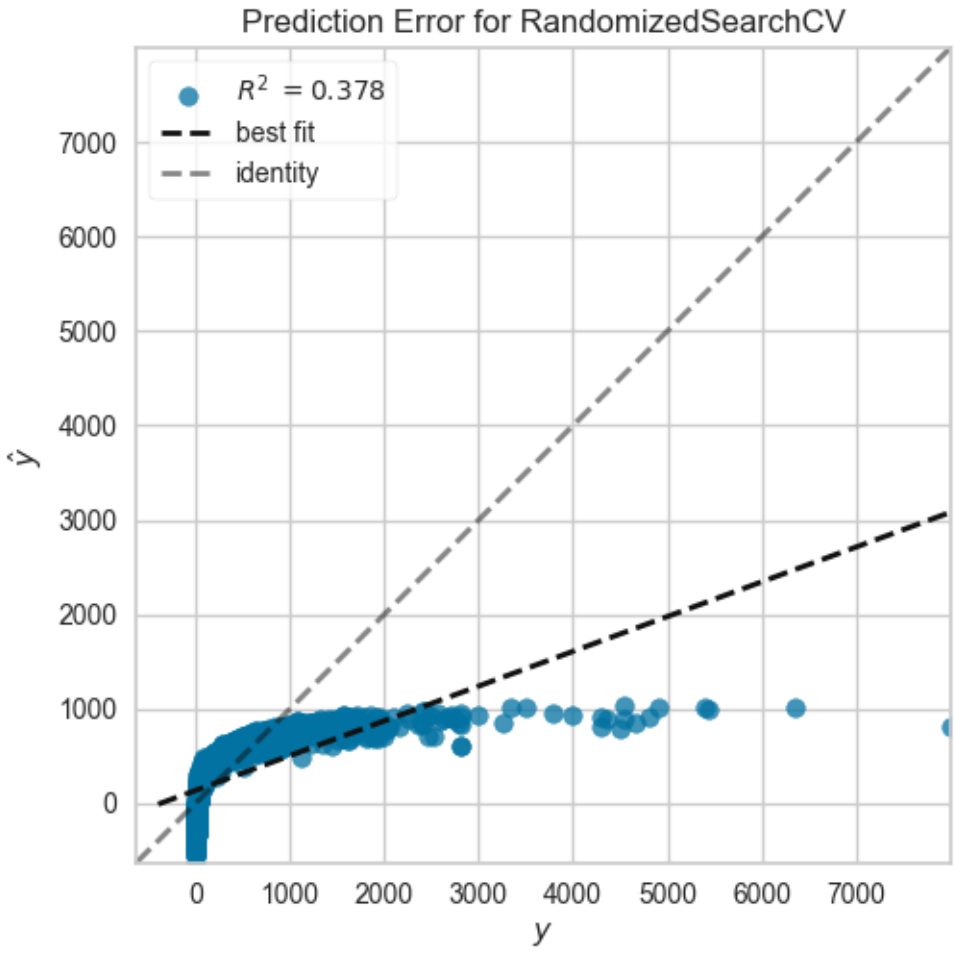
```
total time=   0.0s
[02:56:11] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 1/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=700, sampling_method=uniform;, score=0.381 total
time=   0.2s
[02:56:11] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 2/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=700, sampling_method=uniform;, score=0.306 total
time=   0.1s
[02:56:11] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 3/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=700, sampling_method=uniform;, score=0.297 total
time=   0.1s
[02:56:12] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 4/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=700, sampling_method=uniform;, score=0.328 total
time=   0.1s
[02:56:12] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 5/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=700, sampling_method=uniform;, score=0.423 total
time=   0.4s
[02:56:12] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 1/5] END booster=gblinear, grow_policy=0, importance_type=total_cover,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform;, score=0.381
total time=   0.6s
[02:56:13] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 2/5] END booster=gblinear, grow_policy=0, importance_type=total_cover,
```

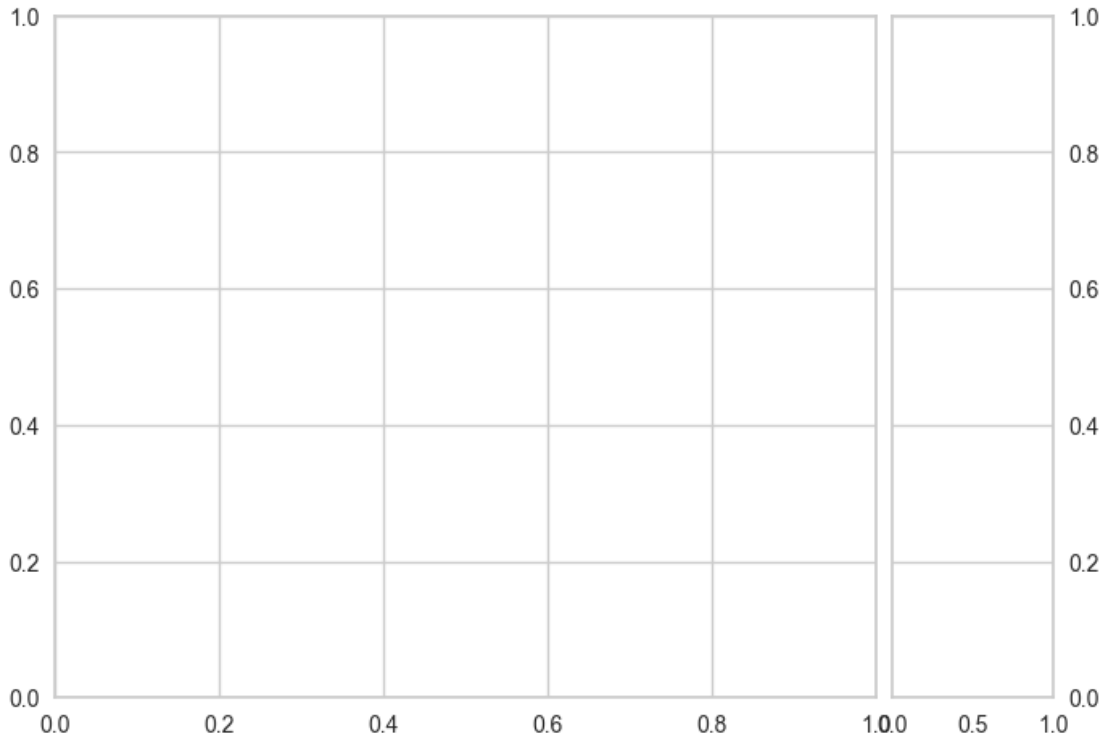
```
learning_rate=0.4, n_estimators=1000, sampling_method=uniform;, score=0.306
total time=   0.6s
[02:56:14] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 3/5] END booster=gblinear, grow_policy=0, importance_type=total_cover,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform;, score=0.297
total time=   0.6s
[02:56:15] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 4/5] END booster=gblinear, grow_policy=0, importance_type=total_cover,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform;, score=0.328
total time=   0.4s
[02:56:15] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 5/5] END booster=gblinear, grow_policy=0, importance_type=total_cover,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform;, score=0.423
total time=   0.2s
[02:56:15] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 1/5] END booster=gblinear, grow_policy=0, importance_type=gain,
learning_rate=0.7, n_estimators=700, sampling_method=gradient_based;,
score=0.381 total time=   0.1s
[02:56:16] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 2/5] END booster=gblinear, grow_policy=0, importance_type=gain,
learning_rate=0.7, n_estimators=700, sampling_method=gradient_based;,
score=0.306 total time=   0.1s
[02:56:16] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 3/5] END booster=gblinear, grow_policy=0, importance_type=gain,
learning_rate=0.7, n_estimators=700, sampling_method=gradient_based;,
score=0.297 total time=   0.1s
[02:56:16] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.
```

```
[CV 4/5] END booster=gblinear, grow_policy=0, importance_type=gain,
learning_rate=0.7, n_estimators=700, sampling_method=gradient_based;,
score=0.328 total time=   0.1s
[02:56:16] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 5/5] END booster=gblinear, grow_policy=0, importance_type=gain,
learning_rate=0.7, n_estimators=700, sampling_method=gradient_based;,
score=0.423 total time=   0.4s
[CV 1/5] END booster=dart, grow_policy=1, importance_type=total_cover,
learning_rate=1, n_estimators=100, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 2/5] END booster=dart, grow_policy=1, importance_type=total_cover,
learning_rate=1, n_estimators=100, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 3/5] END booster=dart, grow_policy=1, importance_type=total_cover,
learning_rate=1, n_estimators=100, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 4/5] END booster=dart, grow_policy=1, importance_type=total_cover,
learning_rate=1, n_estimators=100, sampling_method=uniform;, score=nan total
time=   0.0s
[CV 5/5] END booster=dart, grow_policy=1, importance_type=total_cover,
learning_rate=1, n_estimators=100, sampling_method=uniform;, score=nan total
time=   0.0s
[02:56:17] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 1/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=100, sampling_method=gradient_based;,
score=0.381 total time=   0.0s
[02:56:17] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 2/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=100, sampling_method=gradient_based;,
score=0.306 total time=   0.0s
[02:56:17] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 3/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=100, sampling_method=gradient_based;,
score=0.297 total time=   0.0s
[02:56:17] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
```

```
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 4/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=100, sampling_method=gradient_based;,
score=0.328 total time=   0.0s
[02:56:17] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 5/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=100, sampling_method=gradient_based;,
score=0.423 total time=   0.0s
[02:56:17] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 1/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=1000, sampling_method=gradient_based;,
score=0.381 total time=   0.6s
[02:56:18] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 2/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=1000, sampling_method=gradient_based;,
score=0.306 total time=   0.6s
[02:56:19] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 3/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=1000, sampling_method=gradient_based;,
score=0.297 total time=   0.6s
[02:56:19] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 4/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=1000, sampling_method=gradient_based;,
score=0.328 total time=   0.8s
[02:56:20] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV 5/5] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.4, n_estimators=1000, sampling_method=gradient_based;,
score=0.423 total time=   0.6s
[02:56:21] WARNING: C:\Users\dev-admin\croot2\xgboost-
```

```
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

Mean Absolute Error: 213.3631469830626
Mean Squared Error: 167185.65088499957
Root Mean Squared Error: 408.8834196748501
Mean Absolute Percentage Error: 11.117363786337505
R2 Score: 0.37793613132811377
Training Time: 10.735273599624634
```





[168]:
```
param_grid = {
                'loss':␣
 ↪['squared_error','huber','epsilon_insensitive','squared_epsilon_insensitive'],
                'penalty': ['l2', 'l1', 'elasticnet'],
                'l1_ratio': [0.15,0.45,0.68,0.81,0.97],
                'alpha': [0.0001,0.001,0.01,0.1,1],
                'shuffle': [True,False],
                'learning_rate': ['adaptive','constant','optimal','invscaling'],
                'epsilon': np.linspace(0.001,100,10),
                'average': [True,False]
            }

grid_sgd = RandomizedSearchCV(SGDRegressor(),param_grid,verbose=3,cv=5)
train_and_evaluate_model(grid_sgd)
```

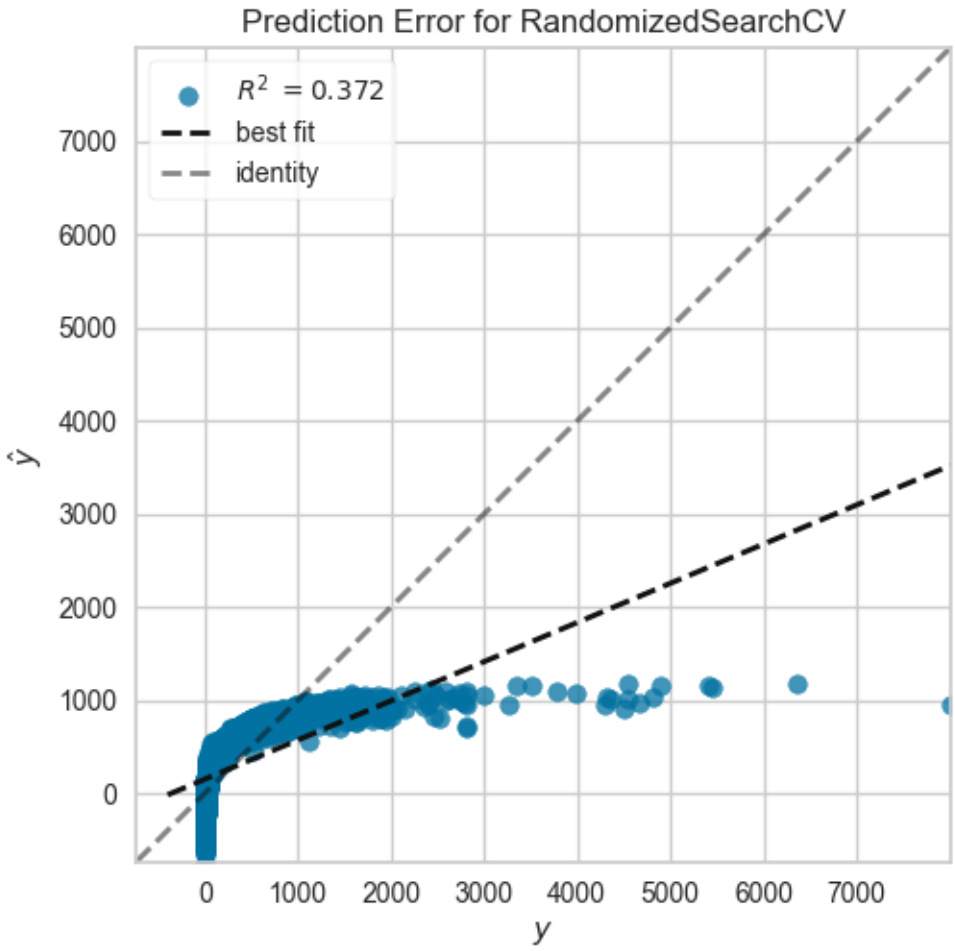
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END alpha=0.0001, average=False, epsilon=44.44499999999999,
l1_ratio=0.97, learning_rate=invscaling, loss=huber, penalty=elasticnet,
shuffle=True;, score=0.203 total time=   0.0s
[CV 2/5] END alpha=0.0001, average=False, epsilon=44.44499999999999,
l1_ratio=0.97, learning_rate=invscaling, loss=huber, penalty=elasticnet,
shuffle=True;, score=0.159 total time=   0.0s
[CV 3/5] END alpha=0.0001, average=False, epsilon=44.44499999999999,
```

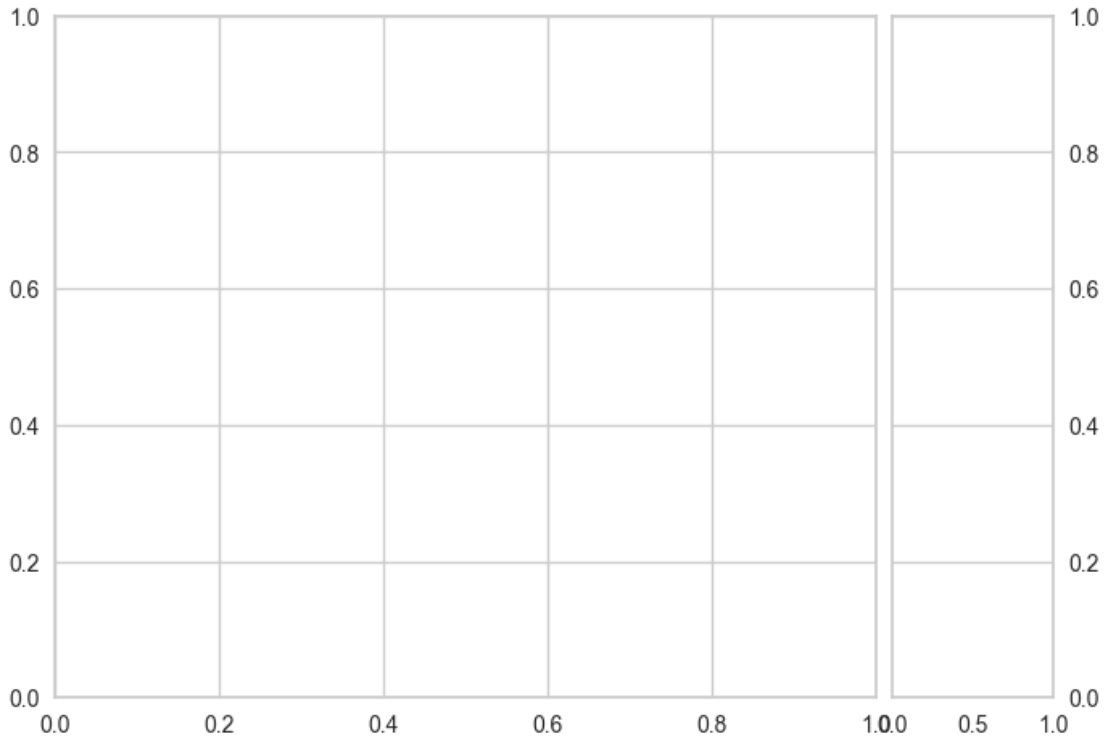
```
l1_ratio=0.97, learning_rate=invscaling, loss=huber, penalty=elasticnet,
shuffle=True;, score=0.174 total time=   0.0s
[CV 4/5] END alpha=0.0001, average=False, epsilon=44.44499999999999,
l1_ratio=0.97, learning_rate=invscaling, loss=huber, penalty=elasticnet,
shuffle=True;, score=0.163 total time=   0.0s
[CV 5/5] END alpha=0.0001, average=False, epsilon=44.44499999999999,
l1_ratio=0.97, learning_rate=invscaling, loss=huber, penalty=elasticnet,
shuffle=True;, score=0.258 total time=   0.0s
[CV 1/5] END alpha=0.001, average=True, epsilon=77.77799999999999,
l1_ratio=0.45, learning_rate=optimal, loss=squared_error, penalty=l2,
shuffle=True;, score=-376493500737332.000 total time=   0.0s
[CV 2/5] END alpha=0.001, average=True, epsilon=77.77799999999999,
l1_ratio=0.45, learning_rate=optimal, loss=squared_error, penalty=l2,
shuffle=True;, score=-90699043708121.375 total time=   0.0s
[CV 3/5] END alpha=0.001, average=True, epsilon=77.77799999999999,
l1_ratio=0.45, learning_rate=optimal, loss=squared_error, penalty=l2,
shuffle=True;, score=-139182928634662.406 total time=   0.0s
[CV 4/5] END alpha=0.001, average=True, epsilon=77.77799999999999,
l1_ratio=0.45, learning_rate=optimal, loss=squared_error, penalty=l2,
shuffle=True;, score=-33589735038042.703 total time=   0.0s
[CV 5/5] END alpha=0.001, average=True, epsilon=77.77799999999999,
l1_ratio=0.45, learning_rate=optimal, loss=squared_error, penalty=l2,
shuffle=True;, score=-266640783688842.938 total time=   0.0s
[CV 1/5] END alpha=0.001, average=True, epsilon=66.667, l1_ratio=0.45,
learning_rate=constant, loss=huber, penalty=l2, shuffle=False;, score=0.219
total time=   0.0s
[CV 2/5] END alpha=0.001, average=True, epsilon=66.667, l1_ratio=0.45,
learning_rate=constant, loss=huber, penalty=l2, shuffle=False;, score=0.170
total time=   0.0s
[CV 3/5] END alpha=0.001, average=True, epsilon=66.667, l1_ratio=0.45,
learning_rate=constant, loss=huber, penalty=l2, shuffle=False;, score=0.185
total time=   0.0s
[CV 4/5] END alpha=0.001, average=True, epsilon=66.667, l1_ratio=0.45,
learning_rate=constant, loss=huber, penalty=l2, shuffle=False;, score=0.173
total time=   0.0s
[CV 5/5] END alpha=0.001, average=True, epsilon=66.667, l1_ratio=0.45,
learning_rate=constant, loss=huber, penalty=l2, shuffle=False;, score=0.275
total time=   0.0s
[CV 1/5] END alpha=1, average=True, epsilon=44.44499999999999, l1_ratio=0.68,
learning_rate=invscaling, loss=huber, penalty=l2, shuffle=True;, score=0.007
total time=   0.0s
[CV 2/5] END alpha=1, average=True, epsilon=44.44499999999999, l1_ratio=0.68,
learning_rate=invscaling, loss=huber, penalty=l2, shuffle=True;, score=0.014
total time=   0.0s
[CV 3/5] END alpha=1, average=True, epsilon=44.44499999999999, l1_ratio=0.68,
learning_rate=invscaling, loss=huber, penalty=l2, shuffle=True;, score=0.029
total time=   0.0s
[CV 4/5] END alpha=1, average=True, epsilon=44.44499999999999, l1_ratio=0.68,
```

```
learning_rate=invscaling, loss=huber, penalty=l2, shuffle=True;, score=0.012
total time=   0.0s
[CV 5/5] END alpha=1, average=True, epsilon=44.44499999999999, l1_ratio=0.68,
learning_rate=invscaling, loss=huber, penalty=l2, shuffle=True;, score=0.019
total time=   0.0s
[CV 1/5] END alpha=0.001, average=False, epsilon=66.667, l1_ratio=0.81,
learning_rate=optimal, loss=huber, penalty=l2, shuffle=True;, score=0.220 total
time=   0.0s
[CV 2/5] END alpha=0.001, average=False, epsilon=66.667, l1_ratio=0.81,
learning_rate=optimal, loss=huber, penalty=l2, shuffle=True;, score=0.163 total
time=   0.0s
[CV 3/5] END alpha=0.001, average=False, epsilon=66.667, l1_ratio=0.81,
learning_rate=optimal, loss=huber, penalty=l2, shuffle=True;, score=0.189 total
time=   0.0s
[CV 4/5] END alpha=0.001, average=False, epsilon=66.667, l1_ratio=0.81,
learning_rate=optimal, loss=huber, penalty=l2, shuffle=True;, score=0.177 total
time=   0.0s
[CV 5/5] END alpha=0.001, average=False, epsilon=66.667, l1_ratio=0.81,
learning_rate=optimal, loss=huber, penalty=l2, shuffle=True;, score=0.277 total
time=   0.0s
[CV 1/5] END alpha=0.001, average=True, epsilon=77.77799999999999,
l1_ratio=0.45, learning_rate=adaptive, loss=huber, penalty=elasticnet,
shuffle=False;, score=-2758632435854.854 total time=   0.6s
[CV 2/5] END alpha=0.001, average=True, epsilon=77.77799999999999,
l1_ratio=0.45, learning_rate=adaptive, loss=huber, penalty=elasticnet,
shuffle=False;, score=-1137325344824.067 total time=   0.5s
[CV 3/5] END alpha=0.001, average=True, epsilon=77.77799999999999,
l1_ratio=0.45, learning_rate=adaptive, loss=huber, penalty=elasticnet,
shuffle=False;, score=-36207.628 total time=   0.2s
[CV 4/5] END alpha=0.001, average=True, epsilon=77.77799999999999,
l1_ratio=0.45, learning_rate=adaptive, loss=huber, penalty=elasticnet,
shuffle=False;, score=-31072933711.059 total time=   0.9s
[CV 5/5] END alpha=0.001, average=True, epsilon=77.77799999999999,
l1_ratio=0.45, learning_rate=adaptive, loss=huber, penalty=elasticnet,
shuffle=False;, score=-73931582341.137 total time=   0.6s
[CV 1/5] END alpha=0.1, average=False, epsilon=55.55599999999999, l1_ratio=0.15,
learning_rate=invscaling, loss=huber, penalty=l2, shuffle=False;, score=0.168
total time=   0.2s
[CV 2/5] END alpha=0.1, average=False, epsilon=55.55599999999999, l1_ratio=0.15,
learning_rate=invscaling, loss=huber, penalty=l2, shuffle=False;, score=0.129
total time=   0.1s
[CV 3/5] END alpha=0.1, average=False, epsilon=55.55599999999999, l1_ratio=0.15,
learning_rate=invscaling, loss=huber, penalty=l2, shuffle=False;, score=0.147
total time=   0.2s
[CV 4/5] END alpha=0.1, average=False, epsilon=55.55599999999999, l1_ratio=0.15,
learning_rate=invscaling, loss=huber, penalty=l2, shuffle=False;, score=0.132
total time=   0.1s
[CV 5/5] END alpha=0.1, average=False, epsilon=55.55599999999999, l1_ratio=0.15,
```

```
learning_rate=invscaling, loss=huber, penalty=l2, shuffle=False;, score=0.210
total time=   0.1s
[CV 1/5] END alpha=0.01, average=False, epsilon=100.0, l1_ratio=0.97,
learning_rate=adaptive, loss=squared_epsilon_insensitive, penalty=l1,
shuffle=True;, score=0.360 total time=   0.0s
[CV 2/5] END alpha=0.01, average=False, epsilon=100.0, l1_ratio=0.97,
learning_rate=adaptive, loss=squared_epsilon_insensitive, penalty=l1,
shuffle=True;, score=0.307 total time=   0.0s
[CV 3/5] END alpha=0.01, average=False, epsilon=100.0, l1_ratio=0.97,
learning_rate=adaptive, loss=squared_epsilon_insensitive, penalty=l1,
shuffle=True;, score=0.279 total time=   0.0s
[CV 4/5] END alpha=0.01, average=False, epsilon=100.0, l1_ratio=0.97,
learning_rate=adaptive, loss=squared_epsilon_insensitive, penalty=l1,
shuffle=True;, score=0.334 total time=   0.0s
[CV 5/5] END alpha=0.01, average=False, epsilon=100.0, l1_ratio=0.97,
learning_rate=adaptive, loss=squared_epsilon_insensitive, penalty=l1,
shuffle=True;, score=0.380 total time=   0.0s
[CV 1/5] END alpha=0.01, average=True, epsilon=55.55599999999999, l1_ratio=0.15,
learning_rate=optimal, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=True;, score=-7508189375417.478 total time=   0.0s
[CV 2/5] END alpha=0.01, average=True, epsilon=55.55599999999999, l1_ratio=0.15,
learning_rate=optimal, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=True;, score=-81528061655566.766 total time=   0.0s
[CV 3/5] END alpha=0.01, average=True, epsilon=55.55599999999999, l1_ratio=0.15,
learning_rate=optimal, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=True;, score=-15823453020813.086 total time=   0.0s
[CV 4/5] END alpha=0.01, average=True, epsilon=55.55599999999999, l1_ratio=0.15,
learning_rate=optimal, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=True;, score=-2471491025461.659 total time=   0.0s
[CV 5/5] END alpha=0.01, average=True, epsilon=55.55599999999999, l1_ratio=0.15,
learning_rate=optimal, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=True;, score=-14085520142506.691 total time=   0.0s
[CV 1/5] END alpha=0.001, average=True, epsilon=100.0, l1_ratio=0.81,
learning_rate=optimal, loss=squared_error, penalty=l1, shuffle=True;,
score=-1063754092627493.750 total time=   0.0s
[CV 2/5] END alpha=0.001, average=True, epsilon=100.0, l1_ratio=0.81,
learning_rate=optimal, loss=squared_error, penalty=l1, shuffle=True;,
score=-792252346495577.250 total time=   0.0s
[CV 3/5] END alpha=0.001, average=True, epsilon=100.0, l1_ratio=0.81,
learning_rate=optimal, loss=squared_error, penalty=l1, shuffle=True;,
score=-442904399636929.812 total time=   0.0s
[CV 4/5] END alpha=0.001, average=True, epsilon=100.0, l1_ratio=0.81,
learning_rate=optimal, loss=squared_error, penalty=l1, shuffle=True;,
score=-44974798835061.047 total time=   0.0s
[CV 5/5] END alpha=0.001, average=True, epsilon=100.0, l1_ratio=0.81,
learning_rate=optimal, loss=squared_error, penalty=l1, shuffle=True;,
score=-323078557331346.125 total time=   0.0s
Mean Absolute Error: 239.00472412724383
```

```
Mean Squared Error: 168728.18103813217
Root Mean Squared Error: 410.7653600757155
Mean Absolute Percentage Error: 12.91891045713944
R2 Score: 0.37219668975809084
Training Time: 6.264957904815674
```

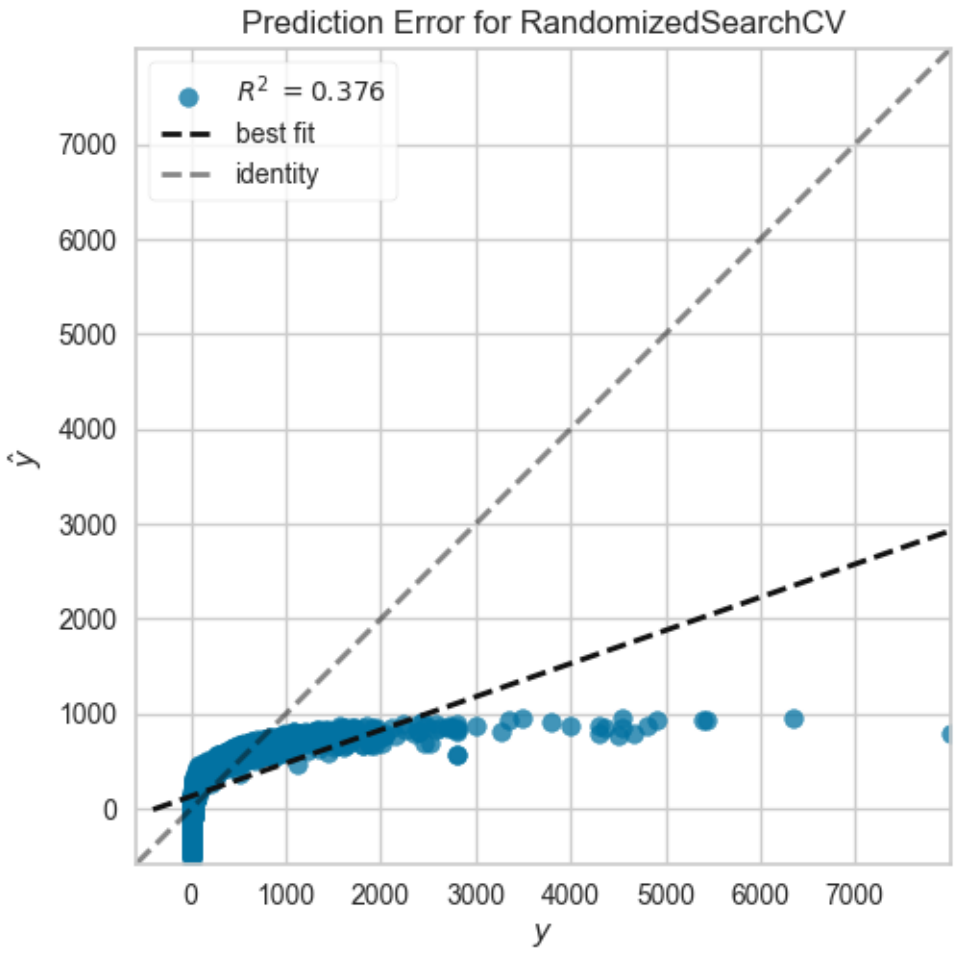


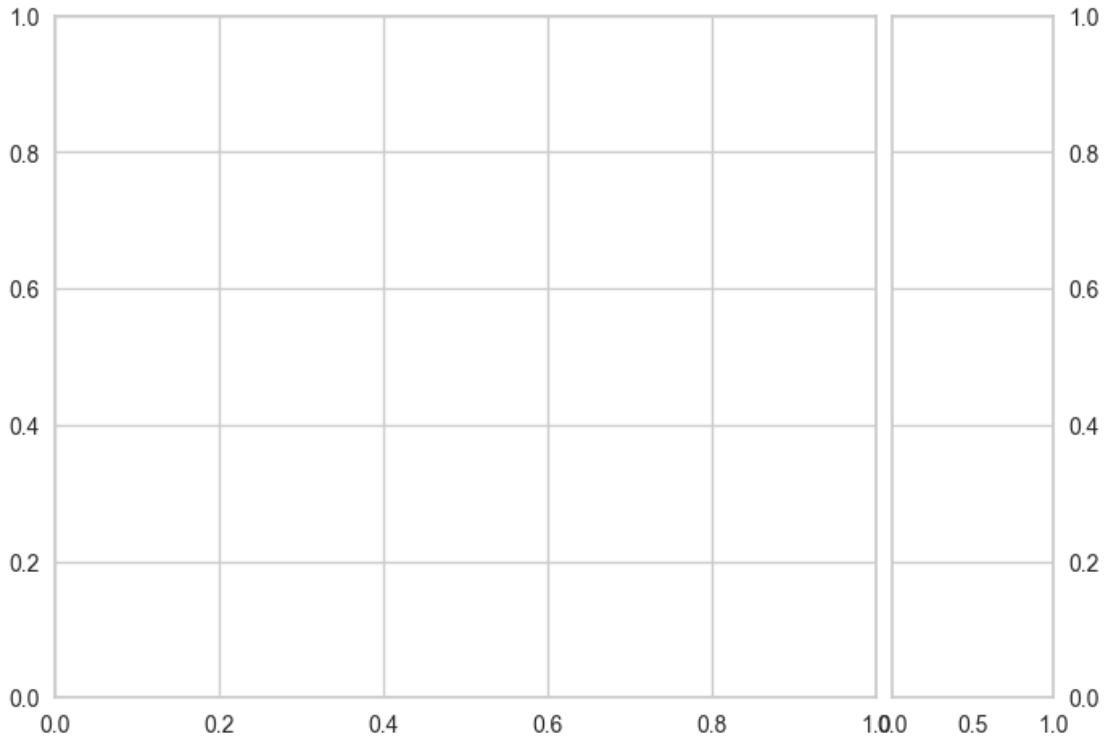


[169]:
```
param_grid = {'epsilon': np.linspace(1,10,10),
'alpha': np.linspace(0.0001,10,10)}

grid_huber = RandomizedSearchCV(HuberRegressor(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_huber)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END …alpha=2.2223, epsilon=1.0; total time=   0.0s
[CV] END …alpha=2.2223, epsilon=1.0; total time=   0.0s
[CV] END …alpha=2.2223, epsilon=1.0; total time=   0.0s
[CV] END …alpha=2.2223, epsilon=1.0; total time=   0.0s
[CV] END …alpha=2.2223, epsilon=1.0; total time=   0.0s
[CV] END …alpha=1.1112, epsilon=8.0; total time=   0.0s
[CV] END …alpha=1.1112, epsilon=8.0; total time=   0.0s
[CV] END …alpha=1.1112, epsilon=8.0; total time=   0.0s
[CV] END …alpha=1.1112, epsilon=8.0; total time=   0.0s
[CV] END …alpha=1.1112, epsilon=8.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=1.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=1.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=1.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=1.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=1.0; total time=   0.0s
[CV] END …alpha=7.777799999999999, epsilon=9.0; total time=   0.0s
```

```
[CV] END …alpha=7.777799999999999, epsilon=9.0; total time=   0.0s
[CV] END …alpha=7.777799999999999, epsilon=9.0; total time=   0.0s
[CV] END …alpha=7.777799999999999, epsilon=9.0; total time=   0.0s
[CV] END …alpha=7.777799999999999, epsilon=9.0; total time=   0.0s
[CV] END …alpha=4.4445, epsilon=7.0; total time=   0.0s
[CV] END …alpha=4.4445, epsilon=7.0; total time=   0.0s
[CV] END …alpha=4.4445, epsilon=7.0; total time=   0.0s
[CV] END …alpha=4.4445, epsilon=7.0; total time=   0.0s
[CV] END …alpha=4.4445, epsilon=7.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=9.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=9.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=9.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=9.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=9.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=1.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=1.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=1.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=1.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=1.0; total time=   0.0s
[CV] END …alpha=0.0001, epsilon=10.0; total time=   0.0s
[CV] END …alpha=0.0001, epsilon=10.0; total time=   0.0s
[CV] END …alpha=0.0001, epsilon=10.0; total time=   0.0s
[CV] END …alpha=0.0001, epsilon=10.0; total time=   0.0s
[CV] END …alpha=0.0001, epsilon=10.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=2.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=2.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=2.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=2.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=2.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=4.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=4.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=4.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=4.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=4.0; total time=   0.0s
Mean Absolute Error: 205.90303286825855
Mean Squared Error: 167804.9273423752
Root Mean Squared Error: 409.63999724437946
Mean Absolute Percentage Error: 10.283807031979869
R2 Score: 0.37563192934179923
Training Time: 1.359616994857788
```


Prediction Error for RandomizedSearchCV
R^2 = 0.376
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



[170]:
```
param_grid = {'link': ['auto', 'identity', 'log'],
'solver': ['lbfgs', 'newton-cholesky'],
'alpha': np.linspace(0.0001,10,10)}

grid_tweedie = RandomizedSearchCV(TweedieRegressor(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_tweedie)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END …alpha=2.2223, link=log, solver=newton-cholesky; total time=   0.1s
[CV] END …alpha=2.2223, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=2.2223, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=2.2223, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=2.2223, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=1.1112, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=1.1112, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=1.1112, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=1.1112, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=1.1112, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=6.6667, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=6.6667, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=6.6667, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=6.6667, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=6.6667, link=identity, solver=newton-cholesky; total time=   0.0s
```

```
[CV] END …alpha=10.0, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=10.0, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=10.0, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=10.0, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=10.0, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=2.2223, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=2.2223, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=2.2223, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=2.2223, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=2.2223, link=auto, solver=lbfgs; total time=   0.0s
[CV] END alpha=4.4445, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=4.4445, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=4.4445, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=4.4445, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=4.4445, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=4.4445, link=identity, solver=lbfgs; total time=   0.0s
[CV] END …alpha=4.4445, link=identity, solver=lbfgs; total time=   0.0s
[CV] END …alpha=4.4445, link=identity, solver=lbfgs; total time=   0.0s
[CV] END …alpha=4.4445, link=identity, solver=lbfgs; total time=   0.0s
[CV] END …alpha=4.4445, link=identity, solver=lbfgs; total time=   0.0s
[CV] END ..alpha=10.0, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END ..alpha=10.0, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END ..alpha=10.0, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END ..alpha=10.0, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END ..alpha=10.0, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=5.5556, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=5.5556, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=5.5556, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=5.5556, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=5.5556, link=auto, solver=lbfgs; total time=   0.0s
[CV] END alpha=3.3334, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=3.3334, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=3.3334, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=3.3334, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=3.3334, link=identity, solver=newton-cholesky; total time=   0.0s
Mean Absolute Error: 48.77439941680776
Mean Squared Error: 11536.538348777854
Root Mean Squared Error: 107.40827877206605
Mean Absolute Percentage Error: 0.5650010180314439
R2 Score: 0.9570748826927813
Training Time: 0.8718183040618896
```

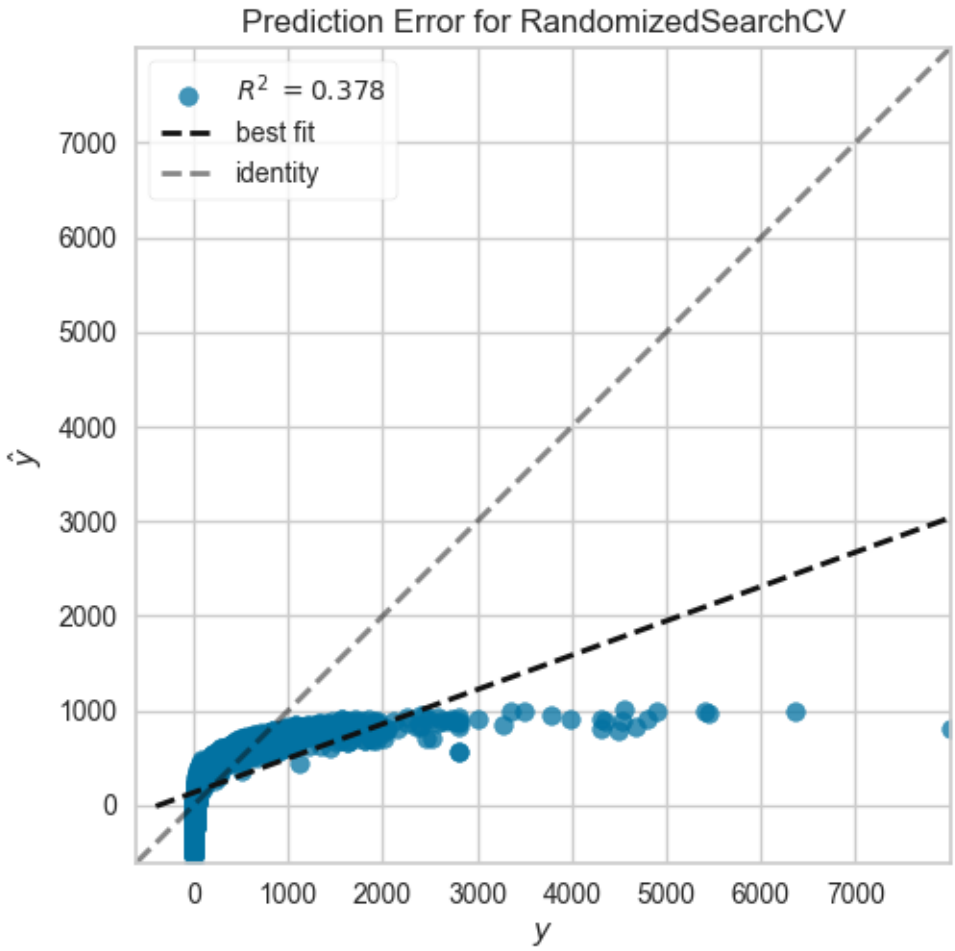
[171]:
```
param_grid = {'C': [0.0001,0.001,0.01,0.1,1,10],
              'loss': ['epsilon_insensitive','squared_epsilon_insensitive'],
              'epsilon': np.linspace(0.001,1,5),
              'shuffle': [True,False],
              'average': [True,False]}
```

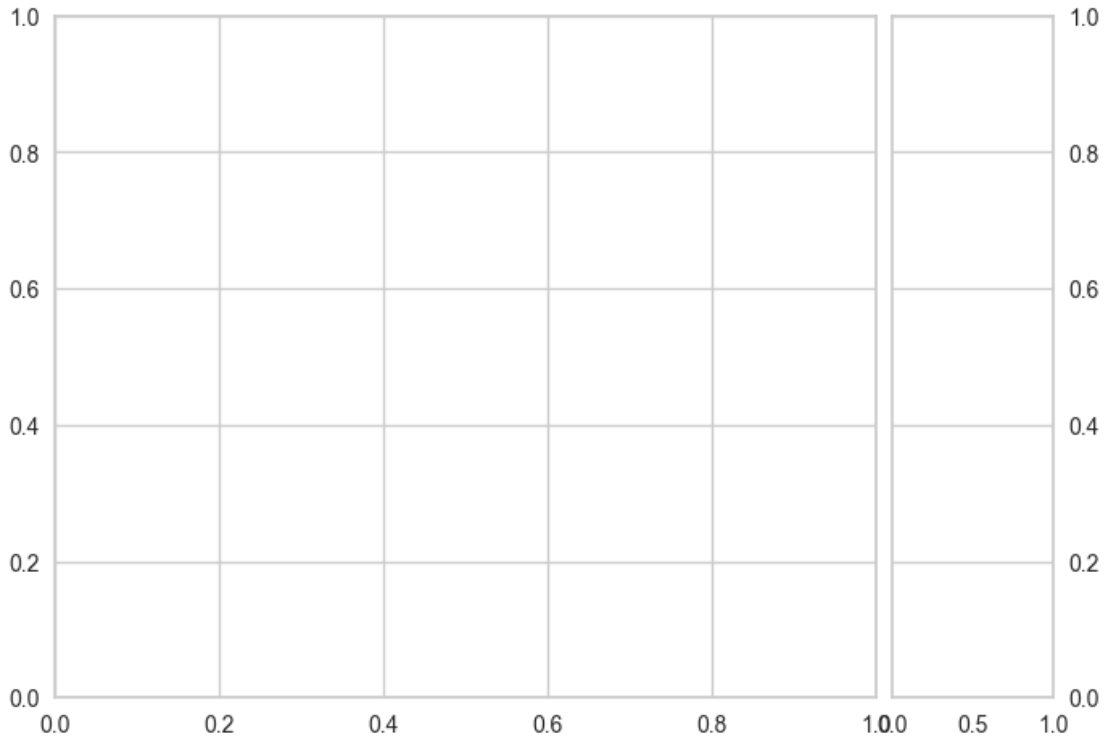
```
grid_pa = RandomizedSearchCV(PassiveAggressiveRegressor(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_pa)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END C=0.1, average=False, epsilon=0.5005, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.1, average=False, epsilon=0.5005, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.1, average=False, epsilon=0.5005, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.1, average=False, epsilon=0.5005, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.1, average=False, epsilon=0.5005, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=1, average=False, epsilon=0.75025, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=False, epsilon=0.75025, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=False, epsilon=0.75025, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=False, epsilon=0.75025, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=False, epsilon=0.75025, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.01, average=False, epsilon=0.001, loss=squared_epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.01, average=False, epsilon=0.001, loss=squared_epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.01, average=False, epsilon=0.001, loss=squared_epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.01, average=False, epsilon=0.001, loss=squared_epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.01, average=False, epsilon=0.001, loss=squared_epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=False, epsilon=0.5005, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
```

```
[CV] END C=1, average=False, epsilon=0.5005, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=False, epsilon=0.5005, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=False, epsilon=0.5005, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=False, epsilon=0.5005, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.0001, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.2s
[CV] END C=0.0001, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.5s
[CV] END C=0.0001, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.7s
[CV] END C=0.0001, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.4s
[CV] END C=0.0001, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.3s
[CV] END C=10, average=False, epsilon=0.001, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.001, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.001, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.001, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.001, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.5005, loss=squared_epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.5005, loss=squared_epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.5005, loss=squared_epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.5005, loss=squared_epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.5005, loss=squared_epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.5005, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.5005, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.5005, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.5005, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.5005, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
```

```
[CV] END C=0.001, average=True, epsilon=0.75025,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.75025,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.75025,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.75025,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.75025,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
Mean Absolute Error: 210.89409846923607
Mean Squared Error: 167298.68872978995
Root Mean Squared Error: 409.0216237924224
Mean Absolute Percentage Error: 10.834506036309525
R2 Score: 0.37751554045405
Training Time: 3.723102331161499
```



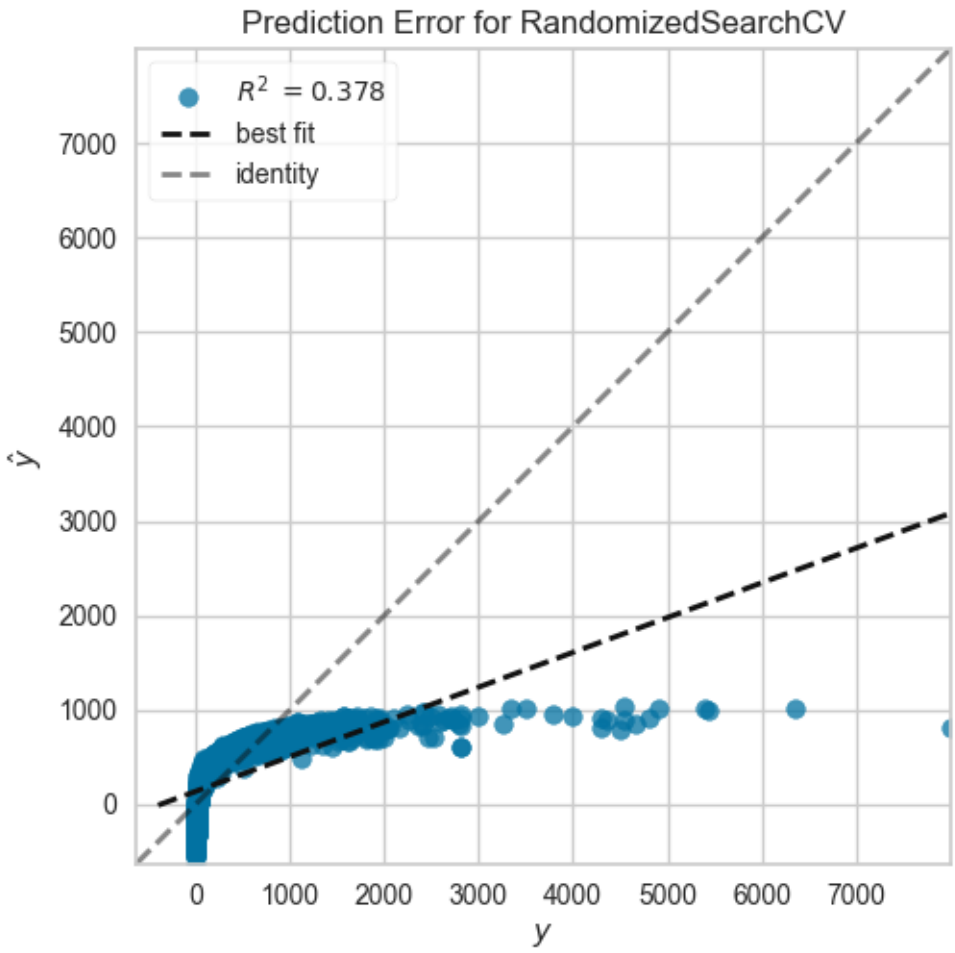


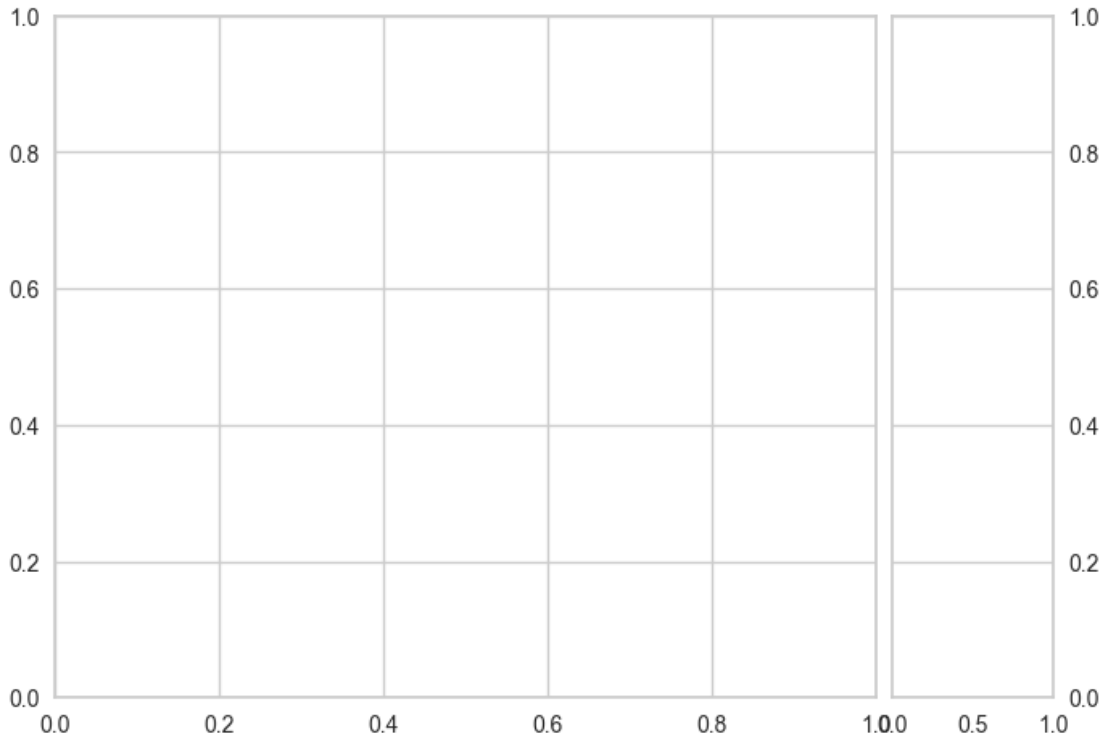
[172]:
```
param_grid = {
             'alpha': [0.001,0.01,0.1],
             'positive': [True,False],
             'selection': ['cyclic','random'],
             'fit_intercept': [True,False]
}

grid_lasso = RandomizedSearchCV(Lasso(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_lasso)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END alpha=0.1, fit_intercept=True, positive=False, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=False, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=False, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=False, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=False, selection=random; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=False, selection=random;
total time=   0.0s
```

```
[CV] END alpha=0.001, fit_intercept=True, positive=False, selection=random;
total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=False, selection=random;
total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=False, selection=random;
total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=False, selection=random;
total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=False, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=False, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=False, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=False, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=False, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=True, selection=cyclic; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=True, selection=cyclic; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=True, selection=cyclic; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=True, selection=cyclic; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=True, selection=cyclic; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, selection=random; total
time=   0.0s
```

```
[CV] END alpha=0.01, fit_intercept=False, positive=True, selection=cyclic; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, selection=cyclic; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, selection=cyclic; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, selection=cyclic; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, selection=cyclic; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=False, selection=cyclic;
total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=False, selection=cyclic;
total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=False, selection=cyclic;
total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=False, selection=cyclic;
total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=False, selection=cyclic;
total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, selection=random; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, positive=False, selection=random;
total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, positive=False, selection=random;
total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, positive=False, selection=random;
total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, positive=False, selection=random;
total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, positive=False, selection=random;
total time=   0.0s
Mean Absolute Error: 213.35509963735015
Mean Squared Error: 167184.2051732798
Root Mean Squared Error: 408.88165179337625
Mean Absolute Percentage Error: 11.119135979312638
R2 Score: 0.3779415105279471
Training Time: 0.5212137699127197
```


Prediction Error for RandomizedSearchCV
R^2 = 0.378
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y

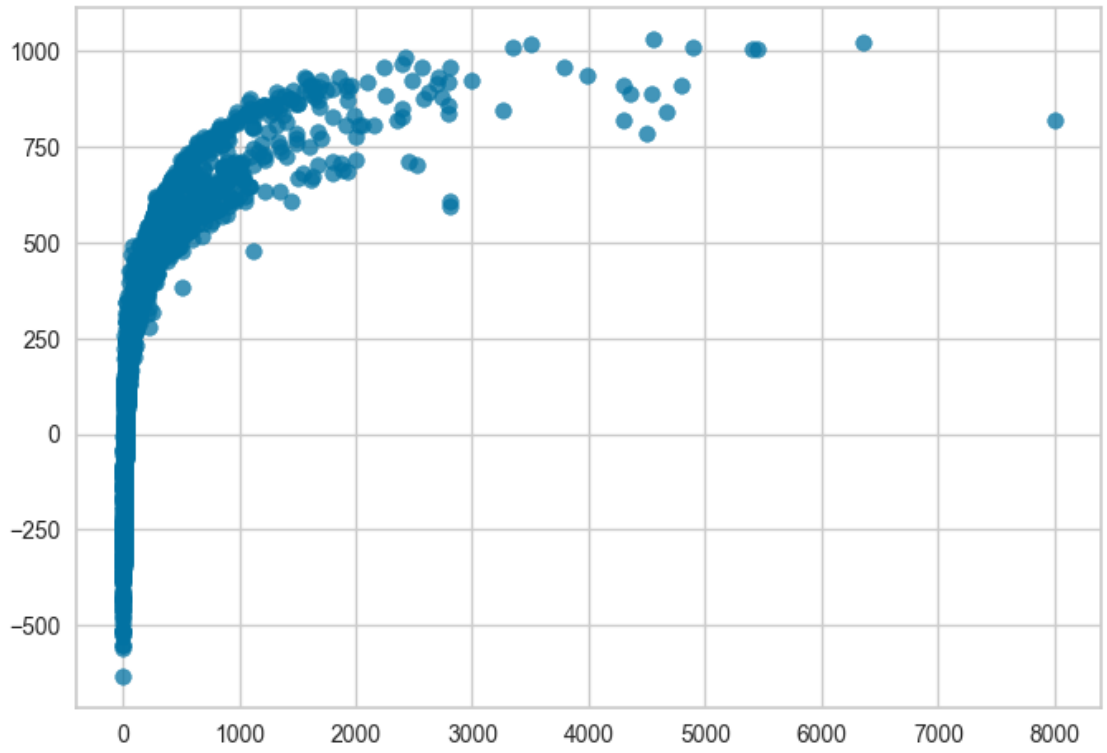


[173]:
```
param_grid = {
             'alpha': [0.001,0.01,0.1],
             'positive': [True,False],
             'fit_intercept': [True,False],
             'solver': ['svd','cholesky','lsqr','sparse_cg','sag','saga','lbfgs']
}

grid_ridge = RandomizedSearchCV(Ridge(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_ridge)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END alpha=0.001, fit_intercept=False, positive=False, solver=lsqr; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, positive=False, solver=lsqr; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, positive=False, solver=lsqr; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, positive=False, solver=lsqr; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, positive=False, solver=lsqr; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=False, solver=cholesky; total
time=   0.0s
```

```
[CV] END alpha=0.1, fit_intercept=True, positive=False, solver=cholesky; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=False, solver=cholesky; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=False, solver=cholesky; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=False, solver=cholesky; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=False, solver=svd; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=False, solver=svd; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=False, solver=svd; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=False, solver=svd; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=False, positive=False, solver=svd; total time=
0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, solver=sparse_cg; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, solver=sparse_cg; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, solver=sparse_cg; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, solver=sparse_cg; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, solver=sparse_cg; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=lbfgs; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=lbfgs; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=lbfgs; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=lbfgs; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=lbfgs; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=sparse_cg; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=sparse_cg; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=sparse_cg; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=sparse_cg; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=sparse_cg; total
time=   0.0s
```

```
[CV] END alpha=0.01, fit_intercept=False, positive=True, solver=lsqr; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, solver=lsqr; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, solver=lsqr; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, solver=lsqr; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, positive=True, solver=lsqr; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=sag; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=sag; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=sag; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=sag; total time=
0.0s
[CV] END alpha=0.1, fit_intercept=True, positive=True, solver=sag; total time=
0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=True, solver=saga; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=True, solver=saga; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=True, solver=saga; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=True, solver=saga; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=True, solver=saga; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=False, solver=cholesky; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=False, solver=cholesky; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=False, solver=cholesky; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=False, solver=cholesky; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, positive=False, solver=cholesky; total
time=   0.0s
Mean Absolute Error: 213.4188968476167
Mean Squared Error: 167198.02463907766
Root Mean Squared Error: 408.89855054656
Mean Absolute Percentage Error: 11.133493789711713
R2 Score: 0.37789009110103045
Training Time: 0.30089831352233887
```

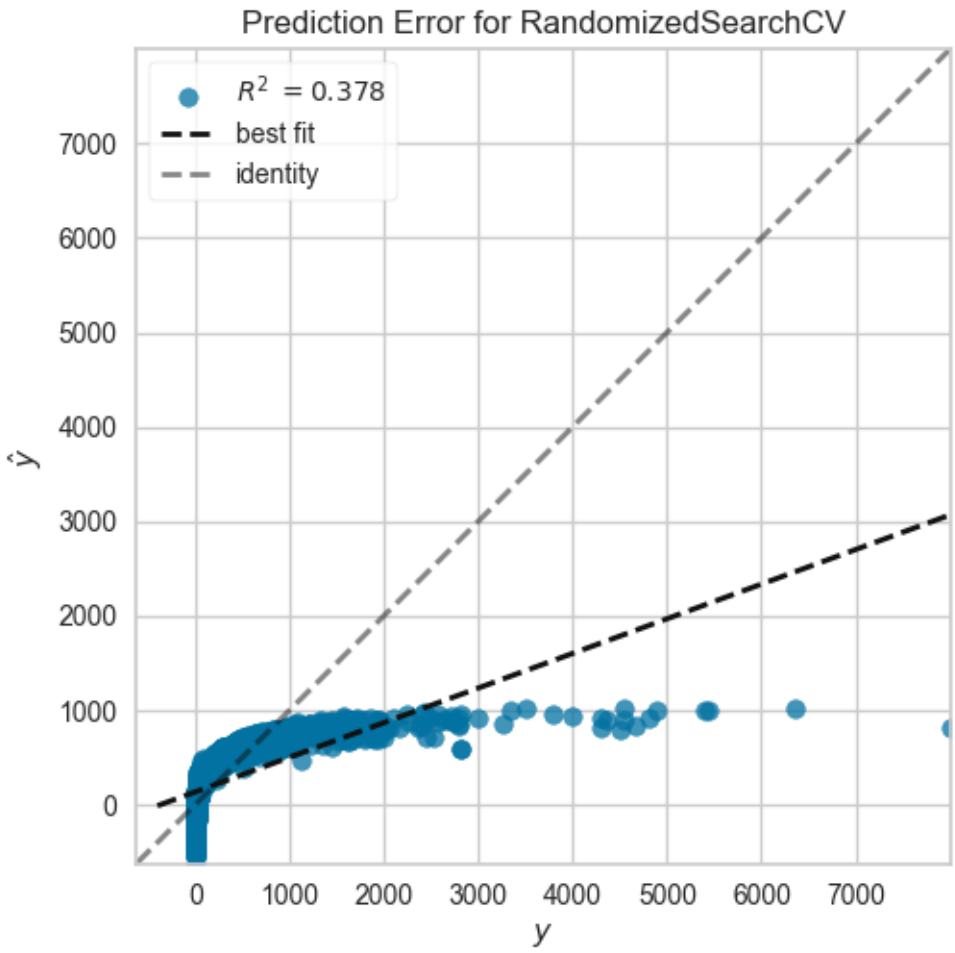


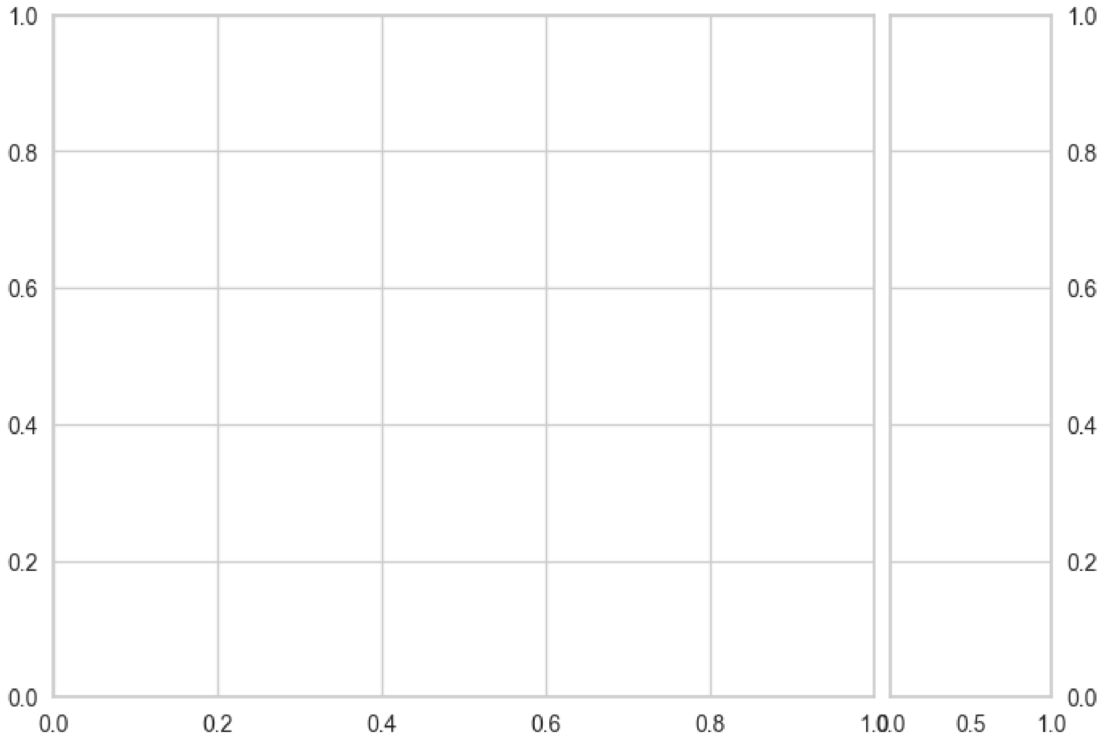
[174]:
```
param_grid = {
    'alpha': [0.001,0.01,0.1],
    'selection': ['cyclic','random'],
    'l1_ratio': [0.1,0.3,0.5,0.8,1],
    'positive': [True,False],
    'fit_intercept': [True,False]
}

grid_elasticnet = RandomizedSearchCV(ElasticNet(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_elasticnet)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END alpha=0.1, fit_intercept=False, l1_ratio=0.8, positive=True,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, l1_ratio=0.8, positive=True,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, l1_ratio=0.8, positive=True,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, l1_ratio=0.8, positive=True,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, l1_ratio=0.8, positive=True,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, l1_ratio=0.3, positive=True,
```

```
selection=random; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, l1_ratio=0.3, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, l1_ratio=0.3, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, l1_ratio=0.3, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, l1_ratio=0.3, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.1, positive=False,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.1, positive=False,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.1, positive=False,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.1, positive=False,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.1, positive=False,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, l1_ratio=0.5, positive=False,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, l1_ratio=0.5, positive=False,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, l1_ratio=0.5, positive=False,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, l1_ratio=0.5, positive=False,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, l1_ratio=0.5, positive=False,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, l1_ratio=0.8, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, l1_ratio=0.8, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, l1_ratio=0.8, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, l1_ratio=0.8, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, l1_ratio=0.8, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, l1_ratio=0.5, positive=False,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, l1_ratio=0.5, positive=False,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, l1_ratio=0.5, positive=False,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, l1_ratio=0.5, positive=False,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, l1_ratio=0.5, positive=False,
```

```
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, l1_ratio=0.8, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, l1_ratio=0.8, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, l1_ratio=0.8, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, l1_ratio=0.8, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, l1_ratio=0.8, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.5, positive=False,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.5, positive=False,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.5, positive=False,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.5, positive=False,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.5, positive=False,
selection=cyclic; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, l1_ratio=0.3, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, l1_ratio=0.3, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, l1_ratio=0.3, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, l1_ratio=0.3, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, l1_ratio=0.3, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.5, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.5, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.5, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.5, positive=True,
selection=random; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, l1_ratio=0.5, positive=True,
selection=random; total time=   0.0s
Mean Absolute Error: 212.79970831241474
Mean Squared Error: 167170.8219541828
Root Mean Squared Error: 408.86528582674123
Mean Absolute Percentage Error: 11.007336749098856
R2 Score: 0.3779913067694469
Training Time: 0.4041624069213867
```


Prediction Error for RandomizedSearchCV
R^2 = 0.378
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y

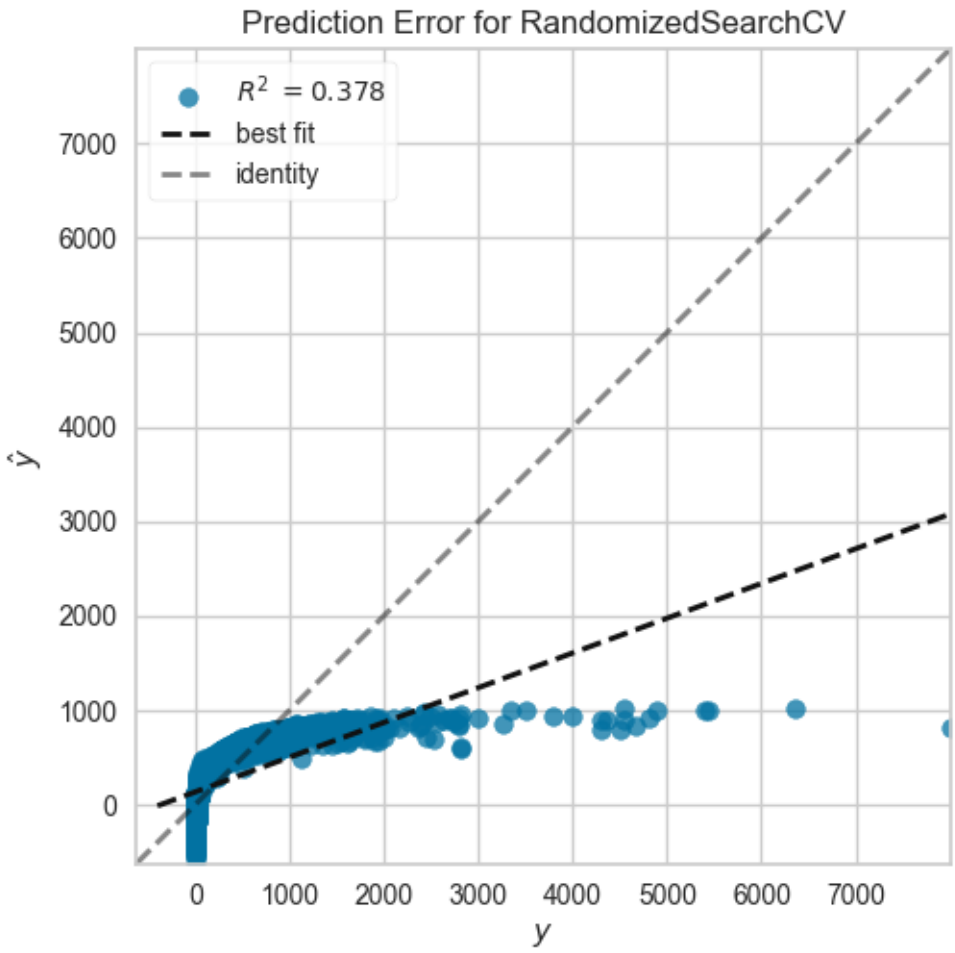


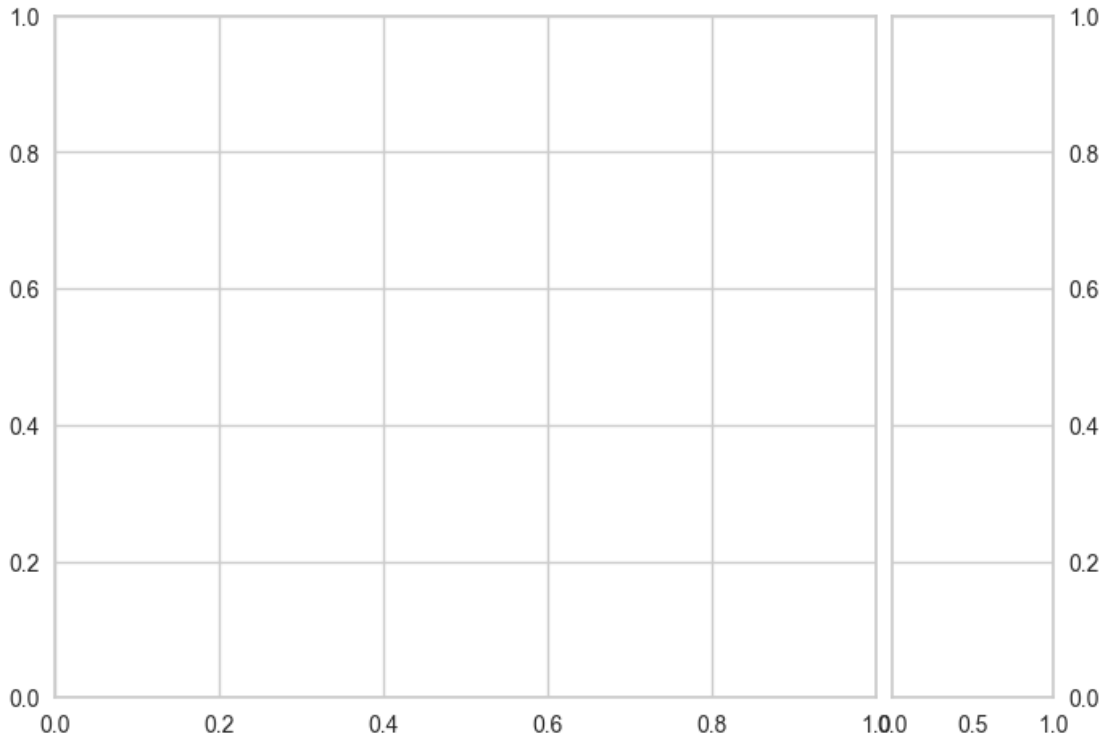
[175]:
```
param_grid = {'alpha_1': [1e-5,1e-6,1e-7,1e-8],
'alpha_2': [1e-5,1e-6,1e-7,1e-8],
'lambda_1': [1e-5,1e-6,1e-7,1e-8],
'lambda_2': [1e-5,1e-6,1e-7,1e-8],
'fit_intercept': [True,False],
'compute_score': [True,False]}

grid_ard = RandomizedSearchCV(ARDRegression(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_ard)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END alpha_1=1e-08, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-08, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-08, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-08, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-08, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-08, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-08, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-08, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-08, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-08, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-06; total time=   0.0s
```

```
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-07, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-07, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-07, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-07, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-07, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-08, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-08, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-08, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-08, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-08, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-08; total time=   0.0s
```

```
[CV] END alpha_1=1e-06, alpha_2=1e-07, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-07, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-07, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-07, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-07, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=False, fit_intercept=False,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=False, fit_intercept=False,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=False, fit_intercept=False,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=False, fit_intercept=False,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=False, fit_intercept=False,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=False, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=False, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=False, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=False, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=False, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-07; total time=   0.0s
Mean Absolute Error: 213.15824638780975
Mean Squared Error: 167180.73114621072
Root Mean Squared Error: 408.87740356518935
Mean Absolute Percentage Error: 11.072741945216503
R2 Score: 0.3779544366774523
Training Time: 2.080883264541626
```

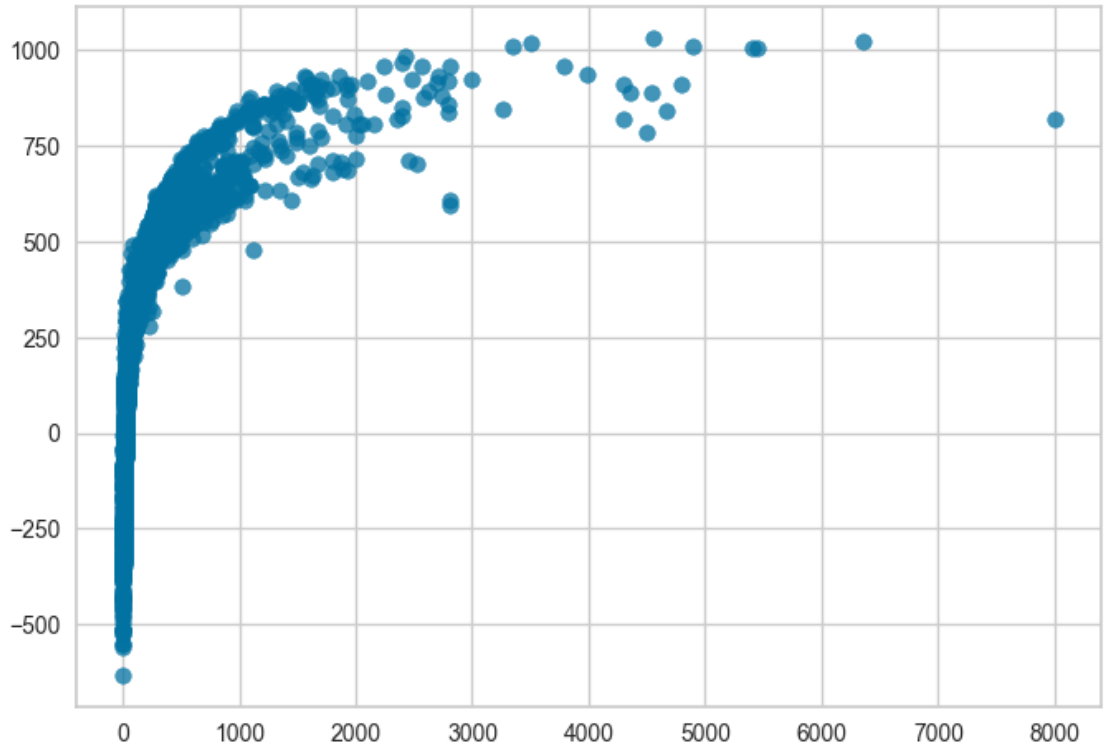

Prediction Error for RandomizedSearchCV
R^2 = 0.378
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y



[176]:
```
param_grid = {'fit_intercept': [True,False],
'positive': [True,False]}

grid_lr = RandomizedSearchCV(LinearRegression(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_lr)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
[CV] END …fit_intercept=True, positive=True; total time=   0.0s
[CV] END …fit_intercept=True, positive=True; total time=   0.0s
[CV] END …fit_intercept=True, positive=True; total time=   0.0s
[CV] END …fit_intercept=True, positive=True; total time=   0.0s
[CV] END …fit_intercept=True, positive=True; total time=   0.0s
[CV] END …fit_intercept=True, positive=False; total time=   0.0s
[CV] END …fit_intercept=True, positive=False; total time=   0.0s
[CV] END …fit_intercept=True, positive=False; total time=   0.0s
[CV] END …fit_intercept=True, positive=False; total time=   0.0s
[CV] END …fit_intercept=True, positive=False; total time=   0.0s
[CV] END …fit_intercept=False, positive=True; total time=   0.0s
[CV] END …fit_intercept=False, positive=True; total time=   0.0s
[CV] END …fit_intercept=False, positive=True; total time=   0.0s
[CV] END …fit_intercept=False, positive=True; total time=   0.0s
[CV] END …fit_intercept=False, positive=True; total time=   0.0s
[CV] END …fit_intercept=False, positive=False; total time=   0.0s
```

```
[CV] END …fit_intercept=False, positive=False; total time=   0.0s
[CV] END …fit_intercept=False, positive=False; total time=   0.0s
[CV] END …fit_intercept=False, positive=False; total time=   0.0s
[CV] END …fit_intercept=False, positive=False; total time=   0.0s
Mean Absolute Error: 213.42072661640157
Mean Squared Error: 167198.14332373216
Root Mean Squared Error: 408.8986956737966
Mean Absolute Percentage Error: 11.133867517277029
R2 Score: 0.37788964949952375
Training Time: 0.17040324211120605
```



[177]:
```
param_grid = {
    'alpha': [0.001,0.01,0.1],
    'fit_intercept': [True,False],
    'solver': ['lbfgs','newton-cholesky'],
    'warm_start': [True,False]
}

grid_poisson =␣
 ↪RandomizedSearchCV(estimator=PoissonRegressor(),param_distributions=param_grid,cv=5,verbose
train_and_evaluate_model(grid_poisson)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
```

```
[CV] END alpha=0.01, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
```

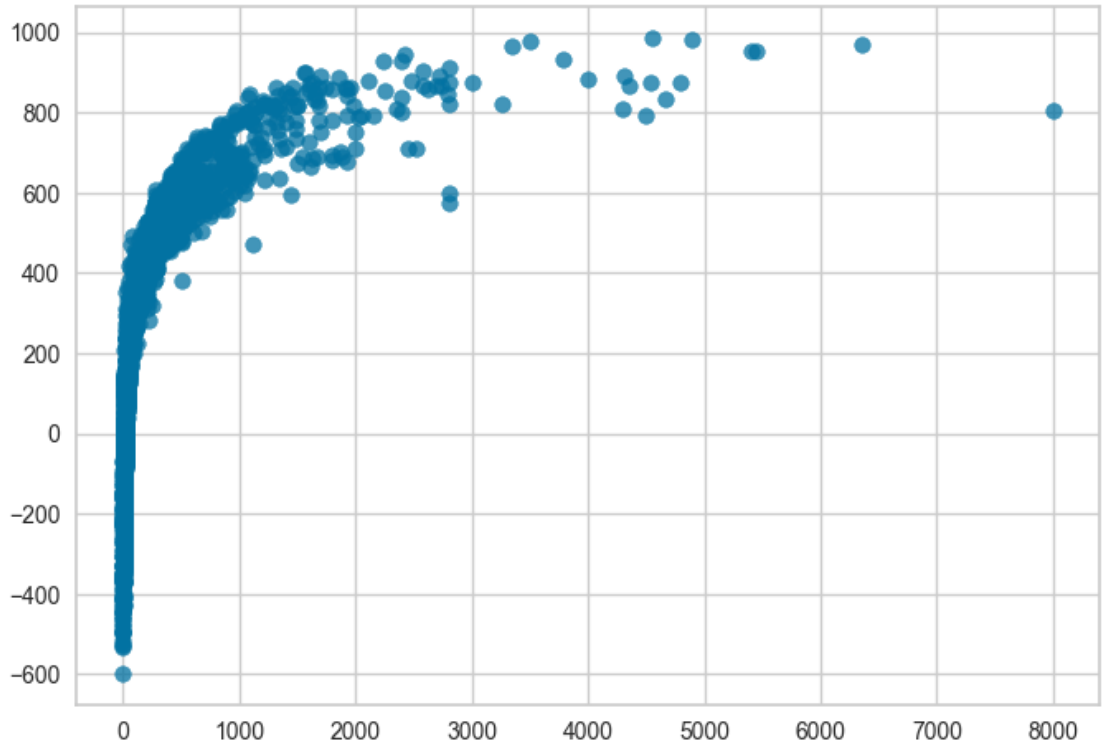
```
[CV] END alpha=0.1, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=False, solver=lbfgs, warm_start=False; total
time=   0.0s
Mean Absolute Error: 92.94608390684164
Mean Squared Error: 172508.4106162349
Root Mean Squared Error: 415.3413182145919
Mean Absolute Percentage Error: 0.7253631686578623
R2 Score: 0.3581312228751703
Training Time: 1.0594820976257324
```

[178]:
```
param_grid = {
    'stop_probability': np.linspace(0,1,5),
    'loss': ['absolute_error','squared_error'],
    'min_samples': np.linspace(0,1,5)[1:]
}

grid_ransac =␣
 ↪RandomizedSearchCV(estimator=RANSACRegressor(),param_distributions=param_grid,cv=5,verbose=
train_and_evaluate_model(grid_ransac)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END loss=squared_error, min_samples=0.25, stop_probability=1.0; total time=
0.4s
[CV] END loss=squared_error, min_samples=0.25, stop_probability=1.0; total time=
0.4s
[CV] END loss=squared_error, min_samples=0.25, stop_probability=1.0; total time=
0.2s
[CV] END loss=squared_error, min_samples=0.25, stop_probability=1.0; total time=
0.2s
[CV] END loss=squared_error, min_samples=0.25, stop_probability=1.0; total time=
0.2s
[CV] END loss=squared_error, min_samples=0.25, stop_probability=0.75; total
time=   0.2s
[CV] END loss=squared_error, min_samples=0.25, stop_probability=0.75; total
time=   0.1s
[CV] END loss=squared_error, min_samples=0.25, stop_probability=0.75; total
time=   0.2s
[CV] END loss=squared_error, min_samples=0.25, stop_probability=0.75; total
time=   0.1s
[CV] END loss=squared_error, min_samples=0.25, stop_probability=0.75; total
time=   0.2s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=0.0; total time=
0.0s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=0.0; total time=
0.0s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=0.0; total time=
0.0s
```

```
[CV] END loss=squared_error, min_samples=0.75, stop_probability=0.0; total time=
0.0s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=0.0; total time=
0.0s
[CV] END loss=squared_error, min_samples=1.0, stop_probability=1.0; total time=
0.0s
[CV] END loss=squared_error, min_samples=1.0, stop_probability=1.0; total time=
0.0s
[CV] END loss=squared_error, min_samples=1.0, stop_probability=1.0; total time=
0.0s
[CV] END loss=squared_error, min_samples=1.0, stop_probability=1.0; total time=
0.0s
[CV] END loss=squared_error, min_samples=1.0, stop_probability=1.0; total time=
0.0s
[CV] END loss=absolute_error, min_samples=0.75, stop_probability=0.75; total
time=   0.2s
[CV] END loss=absolute_error, min_samples=0.75, stop_probability=0.75; total
time=   0.3s
[CV] END loss=absolute_error, min_samples=0.75, stop_probability=0.75; total
time=   0.2s
[CV] END loss=absolute_error, min_samples=0.75, stop_probability=0.75; total
time=   0.2s
[CV] END loss=absolute_error, min_samples=0.75, stop_probability=0.75; total
time=   0.2s
[CV] END loss=absolute_error, min_samples=0.5, stop_probability=0.0; total time=
0.0s
[CV] END loss=absolute_error, min_samples=0.5, stop_probability=0.0; total time=
0.0s
[CV] END loss=absolute_error, min_samples=0.5, stop_probability=0.0; total time=
0.0s
[CV] END loss=absolute_error, min_samples=0.5, stop_probability=0.0; total time=
0.0s
[CV] END loss=absolute_error, min_samples=0.5, stop_probability=0.0; total time=
0.0s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=0.75; total
time=   0.3s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=0.75; total
time=   0.2s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=0.75; total
time=   0.2s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=0.75; total
time=   0.3s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=0.75; total
time=   0.2s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=1.0; total time=
0.2s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=1.0; total time=
0.2s
```

```
[CV] END loss=squared_error, min_samples=0.75, stop_probability=1.0; total time=
0.3s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=1.0; total time=
0.3s
[CV] END loss=squared_error, min_samples=0.75, stop_probability=1.0; total time=
0.2s
[CV] END loss=absolute_error, min_samples=0.75, stop_probability=0.25; total
time=   0.2s
[CV] END loss=absolute_error, min_samples=0.75, stop_probability=0.25; total
time=   0.2s
[CV] END loss=absolute_error, min_samples=0.75, stop_probability=0.25; total
time=   0.2s
[CV] END loss=absolute_error, min_samples=0.75, stop_probability=0.25; total
time=   0.3s
[CV] END loss=absolute_error, min_samples=0.75, stop_probability=0.25; total
time=   0.2s
[CV] END loss=squared_error, min_samples=1.0, stop_probability=0.5; total time=
0.0s
[CV] END loss=squared_error, min_samples=1.0, stop_probability=0.5; total time=
0.0s
[CV] END loss=squared_error, min_samples=1.0, stop_probability=0.5; total time=
0.0s
[CV] END loss=squared_error, min_samples=1.0, stop_probability=0.5; total time=
0.0s
[CV] END loss=squared_error, min_samples=1.0, stop_probability=0.5; total time=
0.0s
Mean Absolute Error: 208.56311483198877
Mean Squared Error: 167686.7539067618
Root Mean Squared Error: 409.495731243638
Mean Absolute Percentage Error: 10.658372041596438
R2 Score: 0.3760716287068032
Training Time: 11.20222282409668
```



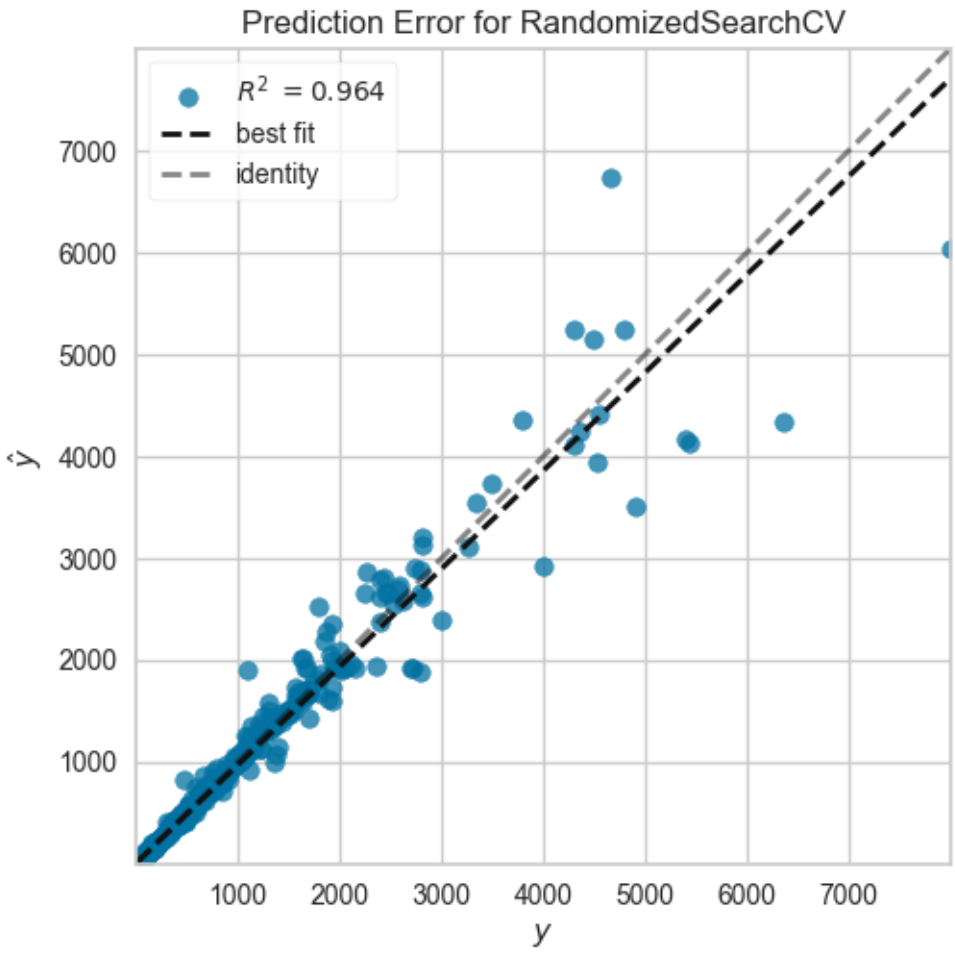
[179]:
```
param_grid = {
    'n_estimators': [200,500,800,1000],
    'criterion': ['squared_error','absolute_error','friedman_mse','poisson'],
    'max_features': ['auto','sqrt','log2'],
    'bootstrap': [True,False],
    'oob_score': [True,False],
    'warm_start': [True,False],
    'max_samples': [0.25,0.5,0.75,1.0]
}

grid_rf = ␣
 ↪RandomizedSearchCV(estimator=RandomForestRegressor(),param_distributions=param_grid,verbose
train_and_evaluate_model(grid_rf)
```

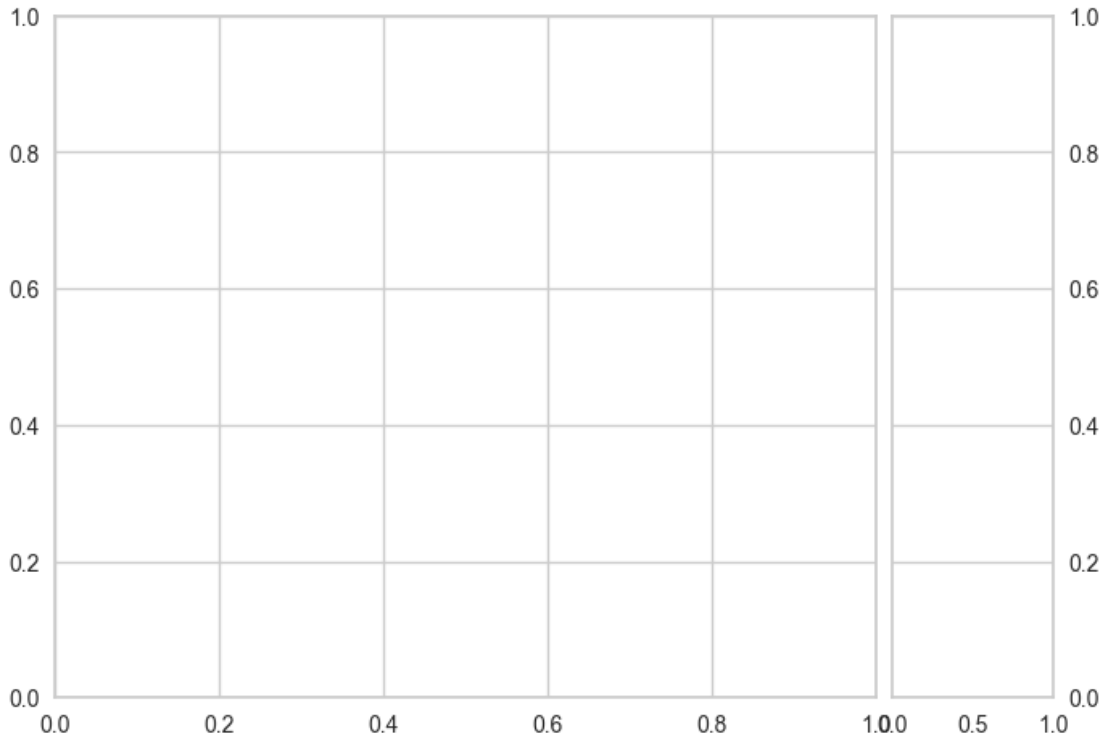
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
max_samples=0.5, n_estimators=1000, oob_score=True, warm_start=False; total
time=  15.6s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
max_samples=0.5, n_estimators=1000, oob_score=True, warm_start=False; total
time=  14.2s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
max_samples=0.5, n_estimators=1000, oob_score=True, warm_start=False; total
```

```
time=  14.1s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
max_samples=0.5, n_estimators=1000, oob_score=True, warm_start=False; total
time=  13.2s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
max_samples=0.5, n_estimators=1000, oob_score=True, warm_start=False; total
time=  13.2s
[CV] END bootstrap=False, criterion=poisson, max_features=sqrt,
max_samples=0.25, n_estimators=1000, oob_score=True, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=sqrt,
max_samples=0.25, n_estimators=1000, oob_score=True, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=sqrt,
max_samples=0.25, n_estimators=1000, oob_score=True, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=sqrt,
max_samples=0.25, n_estimators=1000, oob_score=True, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=sqrt,
max_samples=0.25, n_estimators=1000, oob_score=True, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=False; total
time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=False; total
time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=False; total
time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=False; total
time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=False; total
time=   0.0s
[CV] END bootstrap=True, criterion=poisson, max_features=auto, max_samples=0.25,
n_estimators=800, oob_score=True, warm_start=True; total time=  14.2s
[CV] END bootstrap=True, criterion=poisson, max_features=auto, max_samples=0.25,
n_estimators=800, oob_score=True, warm_start=True; total time=  14.1s
[CV] END bootstrap=True, criterion=poisson, max_features=auto, max_samples=0.25,
n_estimators=800, oob_score=True, warm_start=True; total time=  16.0s
[CV] END bootstrap=True, criterion=poisson, max_features=auto, max_samples=0.25,
n_estimators=800, oob_score=True, warm_start=True; total time=  15.2s
[CV] END bootstrap=True, criterion=poisson, max_features=auto, max_samples=0.25,
n_estimators=800, oob_score=True, warm_start=True; total time=  16.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
```

```
max_samples=0.25, n_estimators=1000, oob_score=False, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=0.25, n_estimators=1000, oob_score=False, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=0.25, n_estimators=1000, oob_score=False, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=0.25, n_estimators=1000, oob_score=False, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=0.25, n_estimators=1000, oob_score=False, warm_start=True; total
time=   0.0s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=auto,
max_samples=0.75, n_estimators=500, oob_score=True, warm_start=False; total
time=  17.1s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=auto,
max_samples=0.75, n_estimators=500, oob_score=True, warm_start=False; total
time=  17.7s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=auto,
max_samples=0.75, n_estimators=500, oob_score=True, warm_start=False; total
time=  16.5s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=auto,
max_samples=0.75, n_estimators=500, oob_score=True, warm_start=False; total
time=  16.8s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=auto,
max_samples=0.75, n_estimators=500, oob_score=True, warm_start=False; total
time=  17.7s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, max_samples=0.25,
n_estimators=500, oob_score=True, warm_start=True; total time=   3.9s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, max_samples=0.25,
n_estimators=500, oob_score=True, warm_start=True; total time=   5.7s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, max_samples=0.25,
n_estimators=500, oob_score=True, warm_start=True; total time=   3.9s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, max_samples=0.25,
n_estimators=500, oob_score=True, warm_start=True; total time=   4.8s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, max_samples=0.25,
n_estimators=500, oob_score=True, warm_start=True; total time=   4.8s
[CV] END bootstrap=False, criterion=poisson, max_features=log2, max_samples=1.0,
n_estimators=200, oob_score=True, warm_start=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2, max_samples=1.0,
n_estimators=200, oob_score=True, warm_start=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2, max_samples=1.0,
n_estimators=200, oob_score=True, warm_start=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2, max_samples=1.0,
n_estimators=200, oob_score=True, warm_start=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2, max_samples=1.0,
```

```
n_estimators=200, oob_score=True, warm_start=True; total time=   0.0s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
max_samples=1.0, n_estimators=500, oob_score=True, warm_start=True; total time=
8.8s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
max_samples=1.0, n_estimators=500, oob_score=True, warm_start=True; total time=
8.8s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
max_samples=1.0, n_estimators=500, oob_score=True, warm_start=True; total time=
10.4s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
max_samples=1.0, n_estimators=500, oob_score=True, warm_start=True; total time=
8.7s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
max_samples=1.0, n_estimators=500, oob_score=True, warm_start=True; total time=
7.7s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=1.0, n_estimators=200, oob_score=True, warm_start=False; total time=
0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=1.0, n_estimators=200, oob_score=True, warm_start=False; total time=
0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=1.0, n_estimators=200, oob_score=True, warm_start=False; total time=
0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=1.0, n_estimators=200, oob_score=True, warm_start=False; total time=
0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=1.0, n_estimators=200, oob_score=True, warm_start=False; total time=
0.0s
Mean Absolute Error: 14.343881435969275
Mean Squared Error: 9595.681569495793
Root Mean Squared Error: 97.95754983407758
Mean Absolute Percentage Error: 0.019700597726653345
R2 Score: 0.9642964167793922
Training Time: 339.90159940719604
```


Prediction Error for RandomizedSearchCV
$R^2 = 0.964$
best fit
identity
7000
6000
5000
4000
3000
2000
1000
$\hat{y}$
1000
2000
3000
4000
5000
6000
7000
y



[180]:
```
param_grid = {
    'alpha': [0.001,0.01,0.1],
    'fit_intercept': [True,False],
    'solver': ['lbfgs','newton-cholesky'],
    'warm_start': [True,False]
}

grid_gamma =␣
 ↪RandomizedSearchCV(estimator=GammaRegressor(),param_distributions=param_grid,verbose=2,cv=5
train_and_evaluate_model(grid_gamma)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END alpha=0.01, fit_intercept=True, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=lbfgs, warm_start=True; total
```

```
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=newton-cholesky,
warm_start=False; total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=newton-cholesky, warm_start=True;
total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=newton-cholesky, warm_start=True;
total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=newton-cholesky, warm_start=True;
total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=newton-cholesky, warm_start=True;
total time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=newton-cholesky, warm_start=True;
total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=True, solver=lbfgs, warm_start=False; total
```

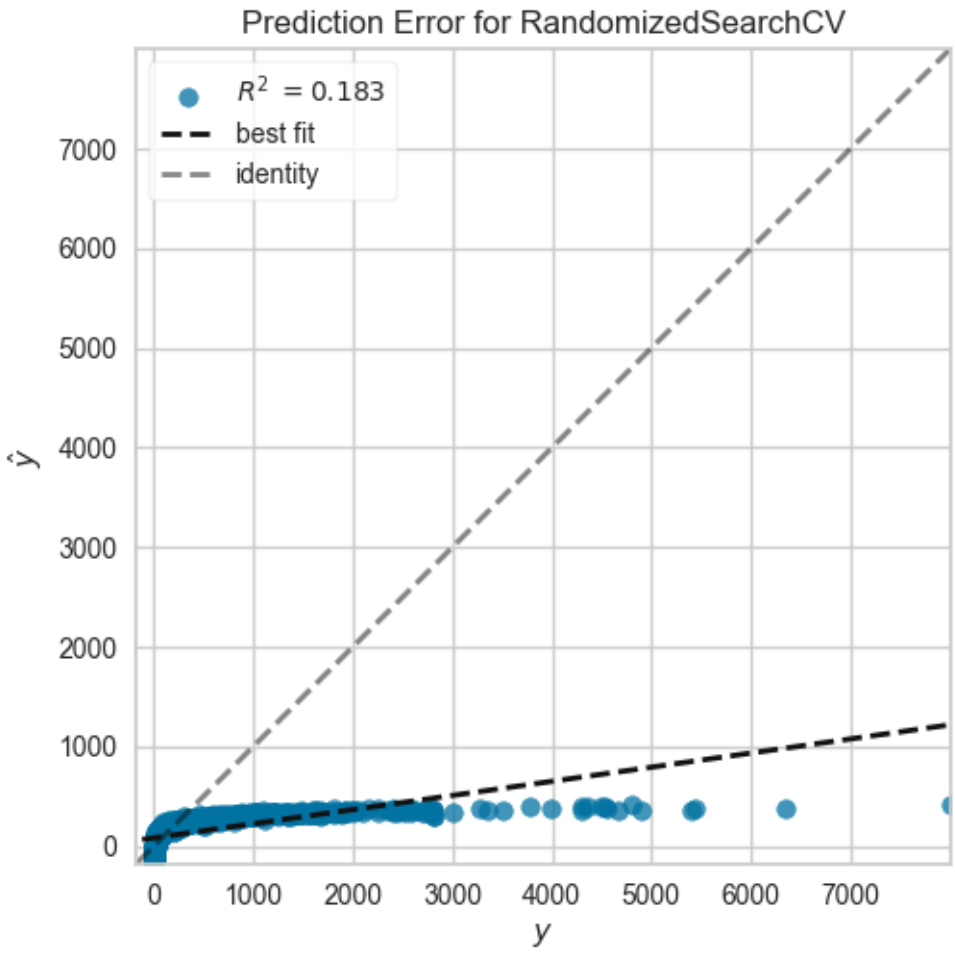
```
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.001, fit_intercept=False, solver=lbfgs, warm_start=True; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.1, fit_intercept=True, solver=lbfgs, warm_start=False; total
time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, solver=newton-cholesky,
warm_start=True; total time=   0.0s
[CV] END alpha=0.01, fit_intercept=True, solver=newton-cholesky,
warm_start=True; total time=   0.0s
Mean Absolute Error: 53.455642523101204
Mean Squared Error: 59054.11516827237
Root Mean Squared Error: 243.01052480967232
Mean Absolute Percentage Error: 0.21712500186824626
R2 Score: 0.78027162529732
Training Time: 0.765162467956543
```

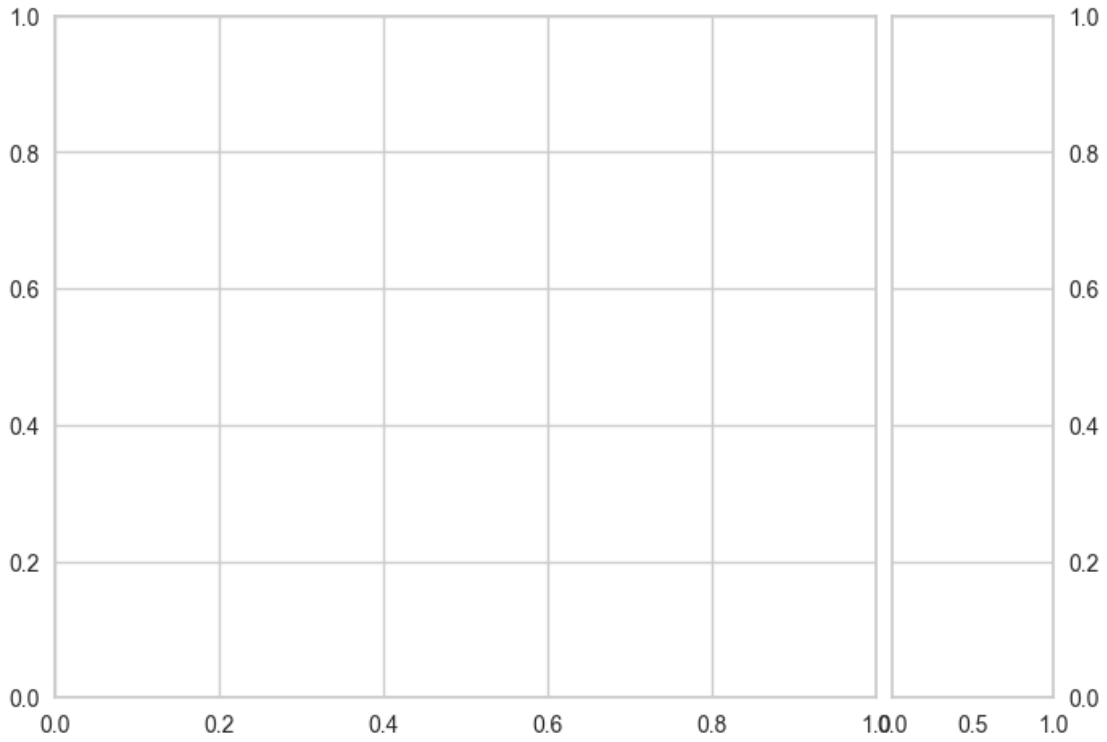
[181]:
```
param_grid = {
    'kernel': ['linear','poly','rbf','sigmoid'],
    'degree': [1,2,3,4],
    'C': [0.01,0.1,1,10],
    'gamma': ['scale','auto'],
    'shrinking': [True,False],
    'epsilon': [0.001,0.01,0.1,1]
}

grid_svr = RandomizedSearchCV(estimator=SVR(),param_distributions=param_grid,cv=5,verbose=2)
train_and_evaluate_model(grid_svr)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END C=1, degree=3, epsilon=0.1, gamma=scale, kernel=sigmoid,
shrinking=True; total time=   2.5s
[CV] END C=1, degree=3, epsilon=0.1, gamma=scale, kernel=sigmoid,
shrinking=True; total time=   2.5s
[CV] END C=1, degree=3, epsilon=0.1, gamma=scale, kernel=sigmoid,
shrinking=True; total time=   2.4s
[CV] END C=1, degree=3, epsilon=0.1, gamma=scale, kernel=sigmoid,
shrinking=True; total time=   2.1s
[CV] END C=1, degree=3, epsilon=0.1, gamma=scale, kernel=sigmoid,
shrinking=True; total time=   2.3s
[CV] END C=0.1, degree=1, epsilon=1, gamma=scale, kernel=sigmoid,
shrinking=True; total time=   2.6s
[CV] END C=0.1, degree=1, epsilon=1, gamma=scale, kernel=sigmoid,
shrinking=True; total time=   2.5s
[CV] END C=0.1, degree=1, epsilon=1, gamma=scale, kernel=sigmoid,
shrinking=True; total time=   2.5s
[CV] END C=0.1, degree=1, epsilon=1, gamma=scale, kernel=sigmoid,
shrinking=True; total time=   2.6s
[CV] END C=0.1, degree=1, epsilon=1, gamma=scale, kernel=sigmoid,
shrinking=True; total time=   2.2s
[CV] END C=0.01, degree=1, epsilon=0.001, gamma=auto, kernel=poly,
shrinking=True; total time=   1.5s
[CV] END C=0.01, degree=1, epsilon=0.001, gamma=auto, kernel=poly,
shrinking=True; total time=   1.9s
[CV] END C=0.01, degree=1, epsilon=0.001, gamma=auto, kernel=poly,
shrinking=True; total time=   1.2s
[CV] END C=0.01, degree=1, epsilon=0.001, gamma=auto, kernel=poly,
shrinking=True; total time=   2.1s
[CV] END C=0.01, degree=1, epsilon=0.001, gamma=auto, kernel=poly,
shrinking=True; total time=   1.1s
[CV] END C=10, degree=3, epsilon=0.1, gamma=auto, kernel=sigmoid,
shrinking=False; total time=   2.5s
[CV] END C=10, degree=3, epsilon=0.1, gamma=auto, kernel=sigmoid,
shrinking=False; total time=   2.5s
```

```
[CV] END C=10, degree=3, epsilon=0.1, gamma=auto, kernel=sigmoid,
shrinking=False; total time=   2.6s
[CV] END C=10, degree=3, epsilon=0.1, gamma=auto, kernel=sigmoid,
shrinking=False; total time=   1.7s
[CV] END C=10, degree=3, epsilon=0.1, gamma=auto, kernel=sigmoid,
shrinking=False; total time=   1.7s
[CV] END C=1, degree=1, epsilon=0.1, gamma=auto, kernel=poly, shrinking=False;
total time=   2.0s
[CV] END C=1, degree=1, epsilon=0.1, gamma=auto, kernel=poly, shrinking=False;
total time=   1.2s
[CV] END C=1, degree=1, epsilon=0.1, gamma=auto, kernel=poly, shrinking=False;
total time=   2.0s
[CV] END C=1, degree=1, epsilon=0.1, gamma=auto, kernel=poly, shrinking=False;
total time=   1.4s
[CV] END C=1, degree=1, epsilon=0.1, gamma=auto, kernel=poly, shrinking=False;
total time=   1.8s
[CV] END C=1, degree=3, epsilon=0.01, gamma=auto, kernel=rbf, shrinking=False;
total time=   2.9s
[CV] END C=1, degree=3, epsilon=0.01, gamma=auto, kernel=rbf, shrinking=False;
total time=   2.9s
[CV] END C=1, degree=3, epsilon=0.01, gamma=auto, kernel=rbf, shrinking=False;
total time=   3.0s
[CV] END C=1, degree=3, epsilon=0.01, gamma=auto, kernel=rbf, shrinking=False;
total time=   2.8s
[CV] END C=1, degree=3, epsilon=0.01, gamma=auto, kernel=rbf, shrinking=False;
total time=   2.0s
[CV] END C=1, degree=1, epsilon=0.001, gamma=scale, kernel=linear,
shrinking=False; total time=   2.1s
[CV] END C=1, degree=1, epsilon=0.001, gamma=scale, kernel=linear,
shrinking=False; total time=   1.3s
[CV] END C=1, degree=1, epsilon=0.001, gamma=scale, kernel=linear,
shrinking=False; total time=   1.9s
[CV] END C=1, degree=1, epsilon=0.001, gamma=scale, kernel=linear,
shrinking=False; total time=   1.9s
[CV] END C=1, degree=1, epsilon=0.001, gamma=scale, kernel=linear,
shrinking=False; total time=   1.9s
[CV] END C=0.01, degree=4, epsilon=0.001, gamma=scale, kernel=sigmoid,
shrinking=False; total time=   2.7s
[CV] END C=0.01, degree=4, epsilon=0.001, gamma=scale, kernel=sigmoid,
shrinking=False; total time=   2.3s
[CV] END C=0.01, degree=4, epsilon=0.001, gamma=scale, kernel=sigmoid,
shrinking=False; total time=   2.2s
[CV] END C=0.01, degree=4, epsilon=0.001, gamma=scale, kernel=sigmoid,
shrinking=False; total time=   1.8s
[CV] END C=0.01, degree=4, epsilon=0.001, gamma=scale, kernel=sigmoid,
shrinking=False; total time=   1.8s
[CV] END C=0.1, degree=4, epsilon=0.1, gamma=auto, kernel=poly, shrinking=False;
total time=   2.1s
```

```
[CV] END C=0.1, degree=4, epsilon=0.1, gamma=auto, kernel=poly, shrinking=False;
total time=   1.3s
[CV] END C=0.1, degree=4, epsilon=0.1, gamma=auto, kernel=poly, shrinking=False;
total time=   1.9s
[CV] END C=0.1, degree=4, epsilon=0.1, gamma=auto, kernel=poly, shrinking=False;
total time=   1.5s
[CV] END C=0.1, degree=4, epsilon=0.1, gamma=auto, kernel=poly, shrinking=False;
total time=   1.6s
[CV] END C=1, degree=1, epsilon=1, gamma=auto, kernel=rbf, shrinking=True; total
time=   1.8s
[CV] END C=1, degree=1, epsilon=1, gamma=auto, kernel=rbf, shrinking=True; total
time=   2.7s
[CV] END C=1, degree=1, epsilon=1, gamma=auto, kernel=rbf, shrinking=True; total
time=   2.7s
[CV] END C=1, degree=1, epsilon=1, gamma=auto, kernel=rbf, shrinking=True; total
time=   2.7s
[CV] END C=1, degree=1, epsilon=1, gamma=auto, kernel=rbf, shrinking=True; total
time=   1.8s
Mean Absolute Error: 153.56680614156429
Mean Squared Error: 219699.17722882665
Root Mean Squared Error: 468.72078813385974
Mean Absolute Percentage Error: 3.0943761423114085
R2 Score: 0.18254395991793526
Training Time: 112.78347778320312
```


Prediction Error for RandomizedSearchCV
R^2 = 0.183
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y

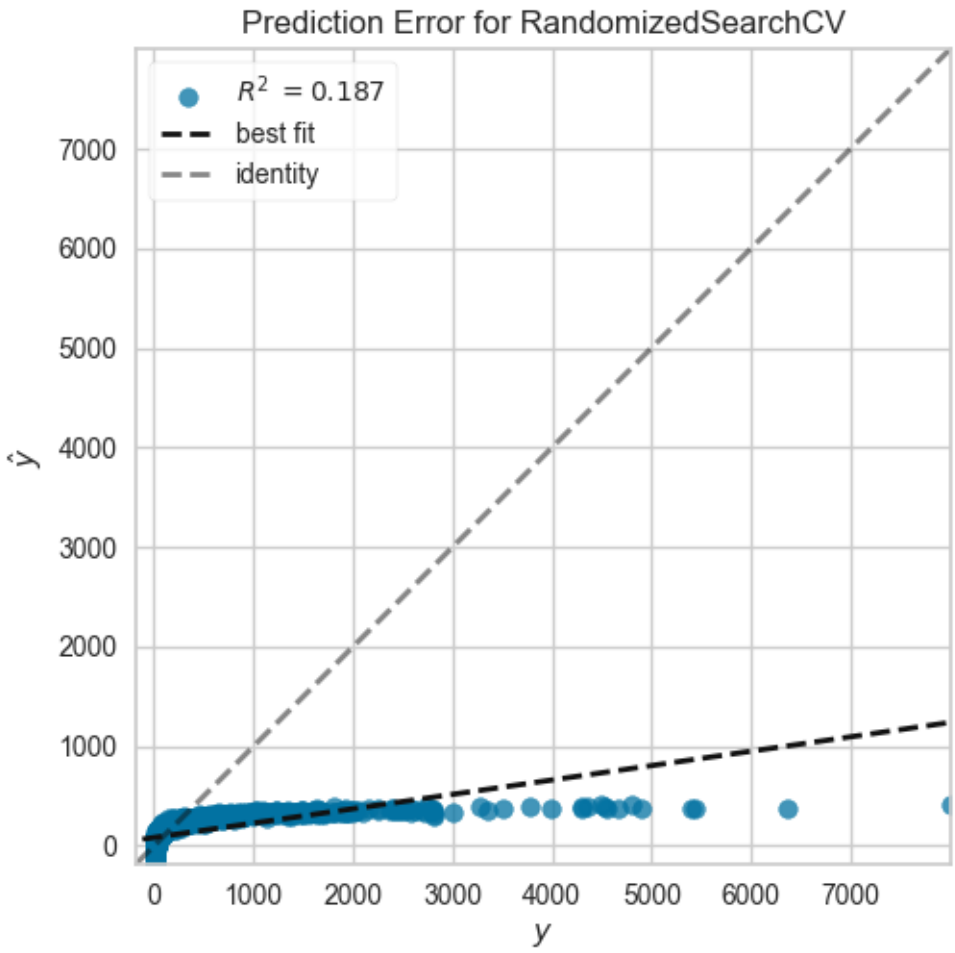


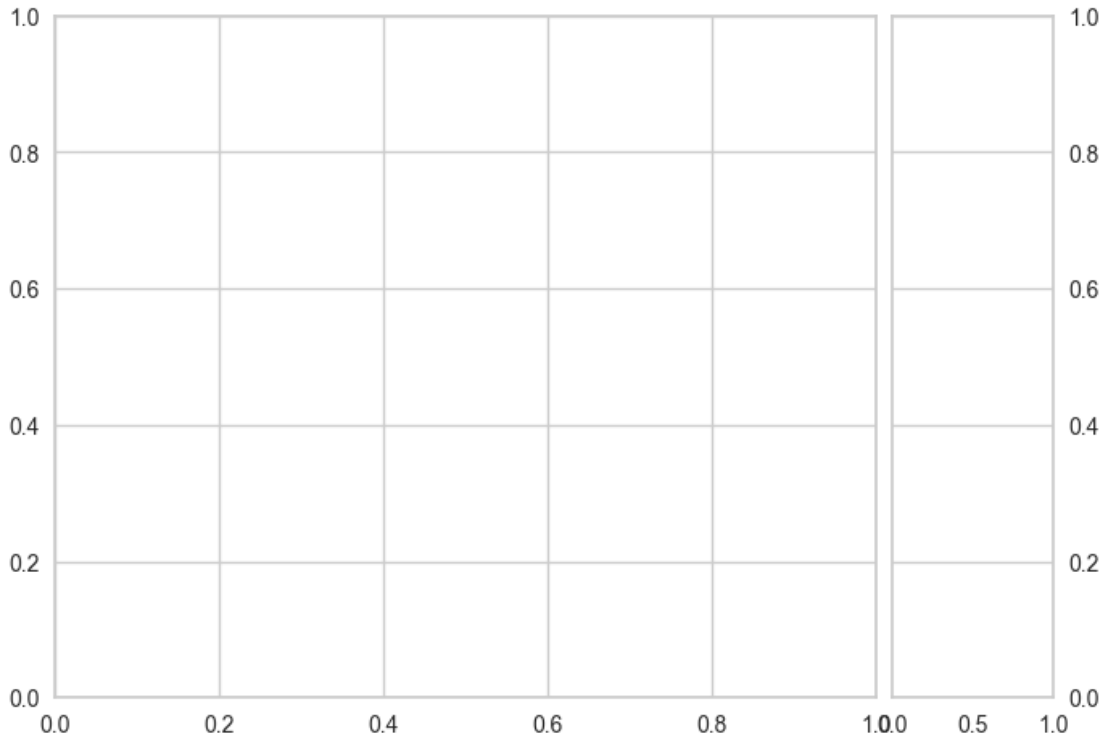
[182]:
```
param_grid = {
    'nu': [0.001,0.01,0.1,0.5,1],
    'C': [0.1,1,10],
    'kernel': ['linear','poly','rbf','sigmoid'],
    'degree': [1,2,3,4],
    'shrinking': [True,False]
}

grid_nusvr = ␣
 ↪RandomizedSearchCV(estimator=NuSVR(),param_distributions=param_grid,cv=5,verbose=2)
train_and_evaluate_model(grid_nusvr)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END C=0.1, degree=1, kernel=rbf, nu=0.1, shrinking=True; total time=   0.4s
[CV] END C=0.1, degree=1, kernel=rbf, nu=0.1, shrinking=True; total time=   0.5s
[CV] END C=0.1, degree=1, kernel=rbf, nu=0.1, shrinking=True; total time=   0.8s
[CV] END C=0.1, degree=1, kernel=rbf, nu=0.1, shrinking=True; total time=   0.4s
[CV] END C=0.1, degree=1, kernel=rbf, nu=0.1, shrinking=True; total time=   0.3s
[CV] END C=10, degree=1, kernel=poly, nu=0.01, shrinking=True; total time=
0.0s
[CV] END C=10, degree=1, kernel=poly, nu=0.01, shrinking=True; total time=
0.0s
[CV] END C=10, degree=1, kernel=poly, nu=0.01, shrinking=True; total time=
```

```
0.0s
[CV] END C=10, degree=1, kernel=poly, nu=0.01, shrinking=True; total time=
0.0s
[CV] END C=10, degree=1, kernel=poly, nu=0.01, shrinking=True; total time=
0.0s
[CV] END ..C=1, degree=1, kernel=rbf, nu=0.1, shrinking=True; total time=   0.2s
[CV] END ..C=1, degree=1, kernel=rbf, nu=0.1, shrinking=True; total time=   0.5s
[CV] END ..C=1, degree=1, kernel=rbf, nu=0.1, shrinking=True; total time=   0.7s
[CV] END ..C=1, degree=1, kernel=rbf, nu=0.1, shrinking=True; total time=   0.4s
[CV] END ..C=1, degree=1, kernel=rbf, nu=0.1, shrinking=True; total time=   0.2s
[CV] END C=10, degree=4, kernel=linear, nu=1, shrinking=False; total time=
7.1s
[CV] END C=10, degree=4, kernel=linear, nu=1, shrinking=False; total time=
3.4s
[CV] END C=10, degree=4, kernel=linear, nu=1, shrinking=False; total time=
2.7s
[CV] END C=10, degree=4, kernel=linear, nu=1, shrinking=False; total time=
6.1s
[CV] END C=10, degree=4, kernel=linear, nu=1, shrinking=False; total time=
7.3s
[CV] END C=0.1, degree=1, kernel=linear, nu=0.01, shrinking=True; total time=
0.0s
[CV] END C=0.1, degree=1, kernel=linear, nu=0.01, shrinking=True; total time=
0.0s
[CV] END C=0.1, degree=1, kernel=linear, nu=0.01, shrinking=True; total time=
0.0s
[CV] END C=0.1, degree=1, kernel=linear, nu=0.01, shrinking=True; total time=
0.0s
[CV] END C=0.1, degree=1, kernel=linear, nu=0.01, shrinking=True; total time=
0.0s
[CV] END C=1, degree=3, kernel=sigmoid, nu=0.1, shrinking=False; total time=
0.2s
[CV] END C=1, degree=3, kernel=sigmoid, nu=0.1, shrinking=False; total time=
0.2s
[CV] END C=1, degree=3, kernel=sigmoid, nu=0.1, shrinking=False; total time=
0.6s
[CV] END C=1, degree=3, kernel=sigmoid, nu=0.1, shrinking=False; total time=
0.7s
[CV] END C=1, degree=3, kernel=sigmoid, nu=0.1, shrinking=False; total time=
0.3s
[CV] END C=0.1, degree=2, kernel=rbf, nu=0.1, shrinking=True; total time=   0.3s
[CV] END C=0.1, degree=2, kernel=rbf, nu=0.1, shrinking=True; total time=   0.3s
[CV] END C=0.1, degree=2, kernel=rbf, nu=0.1, shrinking=True; total time=   0.3s
[CV] END C=0.1, degree=2, kernel=rbf, nu=0.1, shrinking=True; total time=   0.4s
[CV] END C=0.1, degree=2, kernel=rbf, nu=0.1, shrinking=True; total time=   0.3s
[CV] END C=1, degree=2, kernel=poly, nu=0.01, shrinking=True; total time=   0.0s
[CV] END C=1, degree=2, kernel=poly, nu=0.01, shrinking=True; total time=   0.0s
[CV] END C=1, degree=2, kernel=poly, nu=0.01, shrinking=True; total time=   0.0s
```

```
[CV] END C=1, degree=2, kernel=poly, nu=0.01, shrinking=True; total time=   0.0s
[CV] END C=1, degree=2, kernel=poly, nu=0.01, shrinking=True; total time=   0.0s
[CV] END C=0.1, degree=1, kernel=poly, nu=0.1, shrinking=False; total time=
0.2s
[CV] END C=0.1, degree=1, kernel=poly, nu=0.1, shrinking=False; total time=
0.2s
[CV] END C=0.1, degree=1, kernel=poly, nu=0.1, shrinking=False; total time=
0.1s
[CV] END C=0.1, degree=1, kernel=poly, nu=0.1, shrinking=False; total time=
0.1s
[CV] END C=0.1, degree=1, kernel=poly, nu=0.1, shrinking=False; total time=
0.2s
[CV] END C=0.1, degree=1, kernel=linear, nu=0.1, shrinking=True; total time=
0.1s
[CV] END C=0.1, degree=1, kernel=linear, nu=0.1, shrinking=True; total time=
0.2s
[CV] END C=0.1, degree=1, kernel=linear, nu=0.1, shrinking=True; total time=
0.4s
[CV] END C=0.1, degree=1, kernel=linear, nu=0.1, shrinking=True; total time=
0.5s
[CV] END C=0.1, degree=1, kernel=linear, nu=0.1, shrinking=True; total time=
0.3s
Mean Absolute Error: 153.54629396352811
Mean Squared Error: 218627.6025002441
Root Mean Squared Error: 467.5763066070009
Mean Absolute Percentage Error: 3.1743114306521045
R2 Score: 0.18653107195598695
Training Time: 48.15500235557556
```


Prediction Error for RandomizedSearchCV
R^2 = 0.187
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y

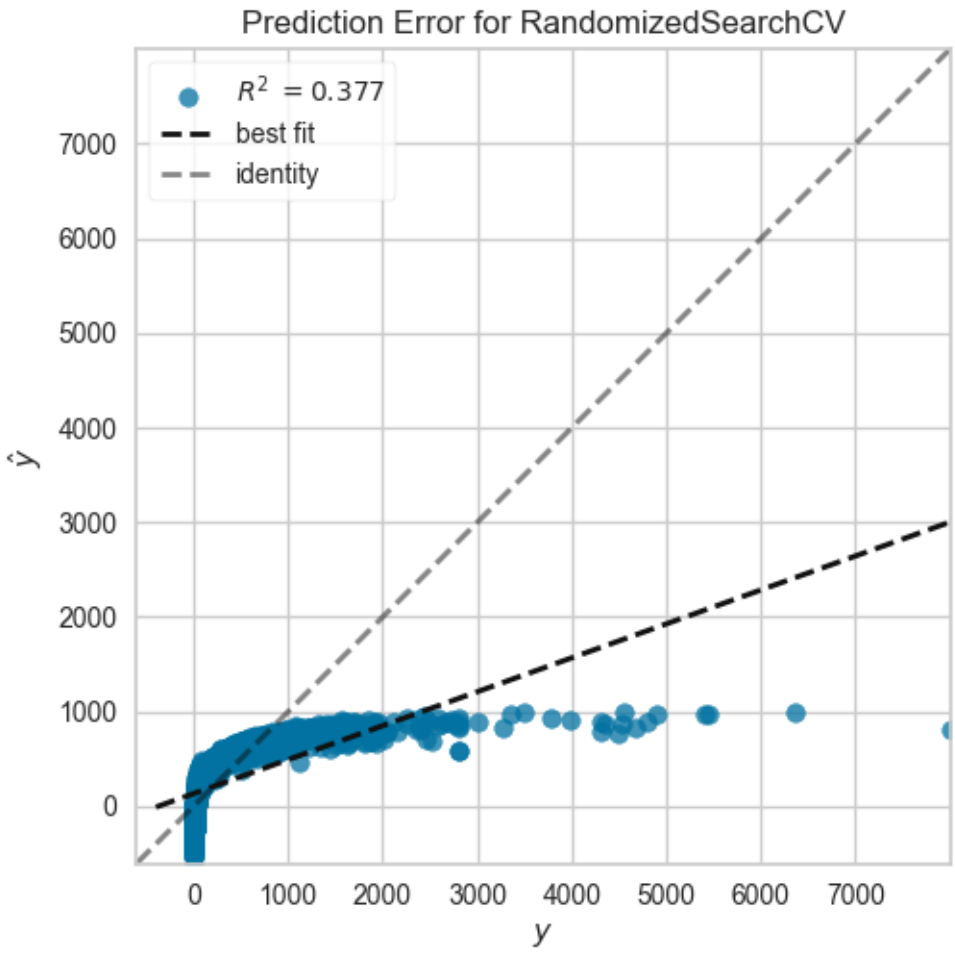


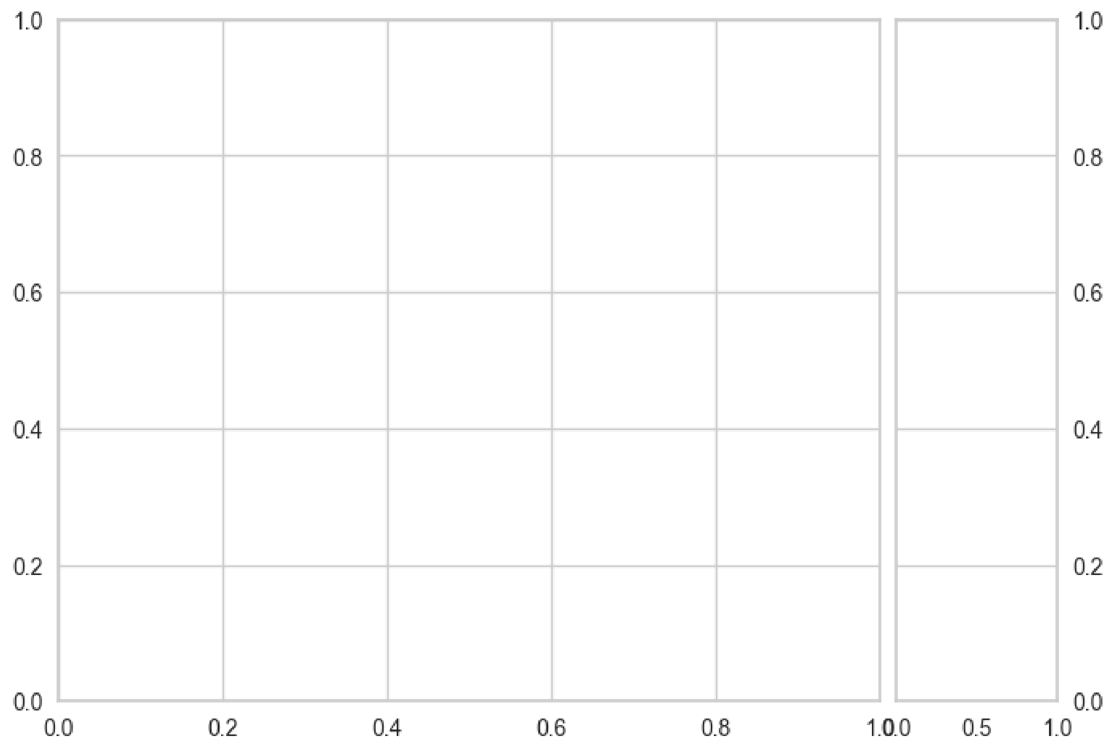
[183]:
```
param_grid = {
    'fit_intercept': [True,False],
    'copy_X': [True,False],
    'max_subpopulation': [1,10,100,1000,10000,100000],
    'n_subsamples': [300,500,753,None]
}

grid_tsr =␣
  ↪RandomizedSearchCV(estimator=TheilSenRegressor(),param_distributions=param_grid,cv=5,verbos
train_and_evaluate_model(grid_tsr)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END copy_X=True, fit_intercept=False, max_subpopulation=100,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=True, fit_intercept=False, max_subpopulation=100,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=True, fit_intercept=False, max_subpopulation=100,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=True, fit_intercept=False, max_subpopulation=100,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=True, fit_intercept=False, max_subpopulation=100,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100000,
```

```
n_subsamples=500; total time=  57.3s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100000,
n_subsamples=500; total time=  48.5s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100000,
n_subsamples=500; total time=  45.6s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100000,
n_subsamples=500; total time=  45.3s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100000,
n_subsamples=500; total time=  46.9s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=1,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=1,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=1,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=1,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=1,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=100000,
n_subsamples=None; total time=  36.7s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=100000,
n_subsamples=None; total time=  35.2s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=100000,
n_subsamples=None; total time=  36.3s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=100000,
n_subsamples=None; total time=  36.2s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=100000,
n_subsamples=None; total time=  38.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100000,
n_subsamples=300; total time=  42.4s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100000,
n_subsamples=300; total time=  43.4s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100000,
n_subsamples=300; total time=  43.4s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100000,
n_subsamples=300; total time=  43.2s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100000,
```

```
n_subsamples=300; total time=  44.7s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=False, fit_intercept=True, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=1,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=1,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=1,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=1,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=False, fit_intercept=False, max_subpopulation=1,
n_subsamples=None; total time=   0.0s
[CV] END copy_X=True, fit_intercept=False, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=True, fit_intercept=False, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=True, fit_intercept=False, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=True, fit_intercept=False, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=True, fit_intercept=False, max_subpopulation=100,
n_subsamples=300; total time=   0.0s
[CV] END copy_X=True, fit_intercept=True, max_subpopulation=10000,
n_subsamples=None; total time=   2.4s
[CV] END copy_X=True, fit_intercept=True, max_subpopulation=10000,
n_subsamples=None; total time=   1.5s
[CV] END copy_X=True, fit_intercept=True, max_subpopulation=10000,
n_subsamples=None; total time=   1.7s
[CV] END copy_X=True, fit_intercept=True, max_subpopulation=10000,
n_subsamples=None; total time=   2.1s
[CV] END copy_X=True, fit_intercept=True, max_subpopulation=10000,
n_subsamples=None; total time=   2.5s
Mean Absolute Error: 209.42373977666145
Mean Squared Error: 167449.4982389903
Root Mean Squared Error: 409.2059362215928
Mean Absolute Percentage Error: 10.734662613380062
R2 Score: 0.3769544088842707
Training Time: 709.5463767051697
```


Prediction Error for RandomizedSearchCV
R^2 = 0.377
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y

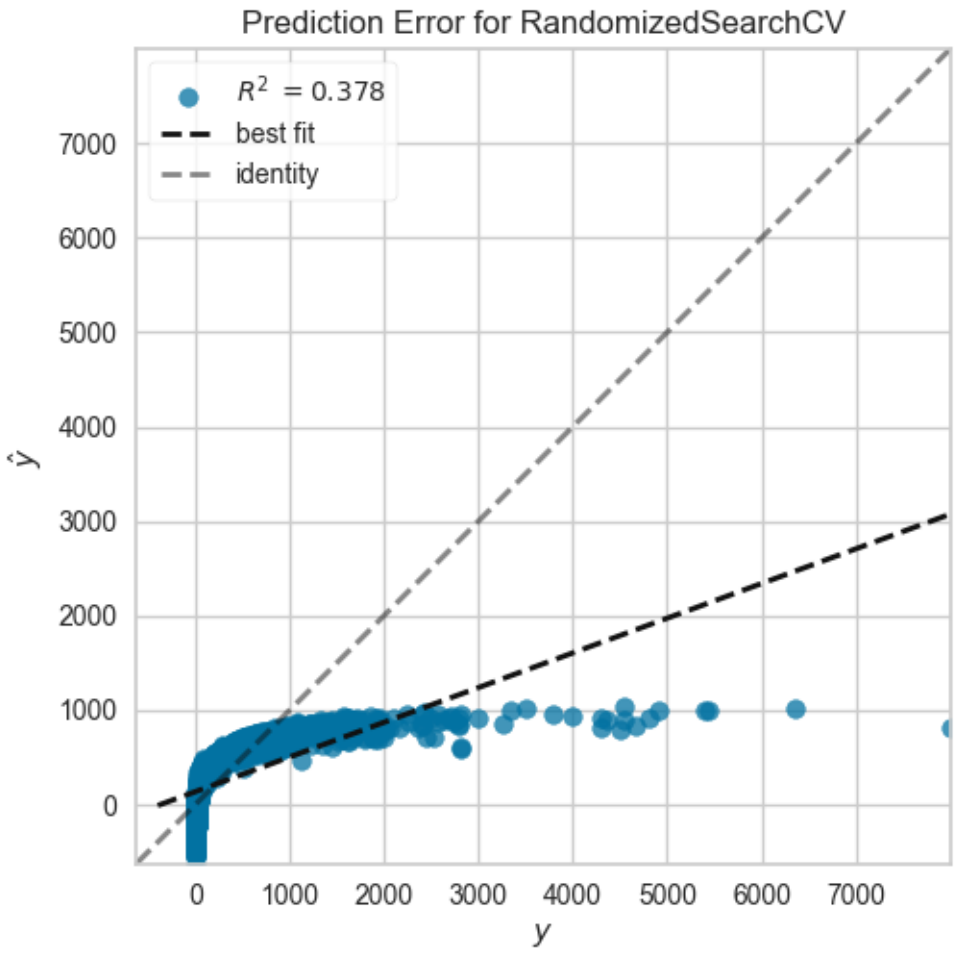


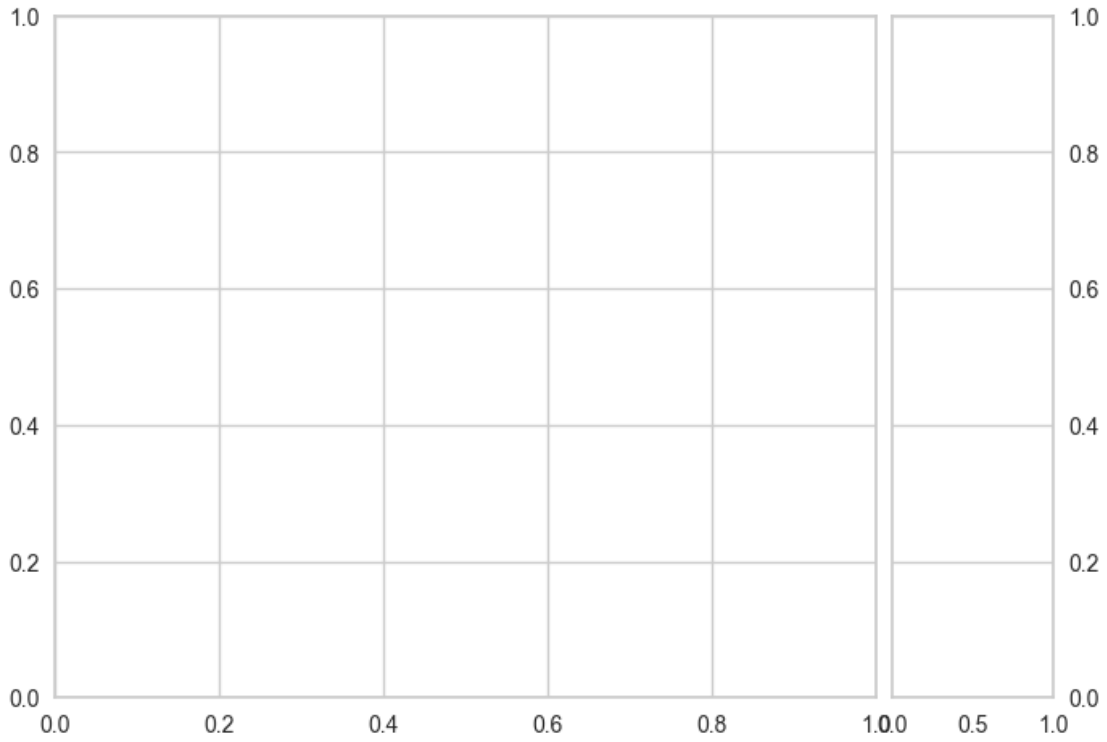
[184]:
```
param_grid = {'alpha_1': [1e-5,1e-6,1e-7,1e-8],
'alpha_2': [1e-5,1e-6,1e-7,1e-8],
'lambda_1': [1e-5,1e-6,1e-7,1e-8],
'lambda_2': [1e-5,1e-6,1e-7,1e-8],
'fit_intercept': [True,False],
'compute_score': [True,False]
             }

grid_bayesian_ridge =␣
 ↪RandomizedSearchCV(BayesianRidge(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_bayesian_ridge)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END alpha_1=1e-07, alpha_2=1e-05, compute_score=True, fit_intercept=True,
lambda_1=1e-05, lambda_2=1e-08; total time=   0.5s
[CV] END alpha_1=1e-07, alpha_2=1e-05, compute_score=True, fit_intercept=True,
lambda_1=1e-05, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-05, compute_score=True, fit_intercept=True,
lambda_1=1e-05, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-05, compute_score=True, fit_intercept=True,
lambda_1=1e-05, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-05, compute_score=True, fit_intercept=True,
lambda_1=1e-05, lambda_2=1e-08; total time=   0.0s
```

```
[CV] END alpha_1=1e-06, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-08, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=True, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-06, compute_score=True, fit_intercept=False,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-06, compute_score=True, fit_intercept=False,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-06, compute_score=True, fit_intercept=False,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-06, compute_score=True, fit_intercept=False,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-06, compute_score=True, fit_intercept=False,
lambda_1=1e-06, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=True, fit_intercept=True,
lambda_1=1e-08, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=True, fit_intercept=True,
lambda_1=1e-08, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=True, fit_intercept=True,
lambda_1=1e-08, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=True, fit_intercept=True,
lambda_1=1e-08, lambda_2=1e-05; total time=   0.0s
```

```
[CV] END alpha_1=1e-05, alpha_2=1e-06, compute_score=True, fit_intercept=True,
lambda_1=1e-08, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-06, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-06, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-06, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-06, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-06, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-05, compute_score=True, fit_intercept=False,
lambda_1=1e-05, lambda_2=1e-07; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-07, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-05; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
[CV] END alpha_1=1e-05, alpha_2=1e-07, compute_score=False, fit_intercept=True,
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
Mean Absolute Error: 213.06071960991235
Mean Squared Error: 167179.35520036993
Root Mean Squared Error: 408.8757209719965
Mean Absolute Percentage Error: 11.060482850937637
R2 Score: 0.37795955629261224
Training Time: 1.9563612937927246
```


Prediction Error for RandomizedSearchCV
R^2 = 0.378
best fit
identity
7000
6000
5000
4000
3000
2000
1000
0
$\hat{y}$
0
1000
2000
3000
4000
5000
6000
7000
y

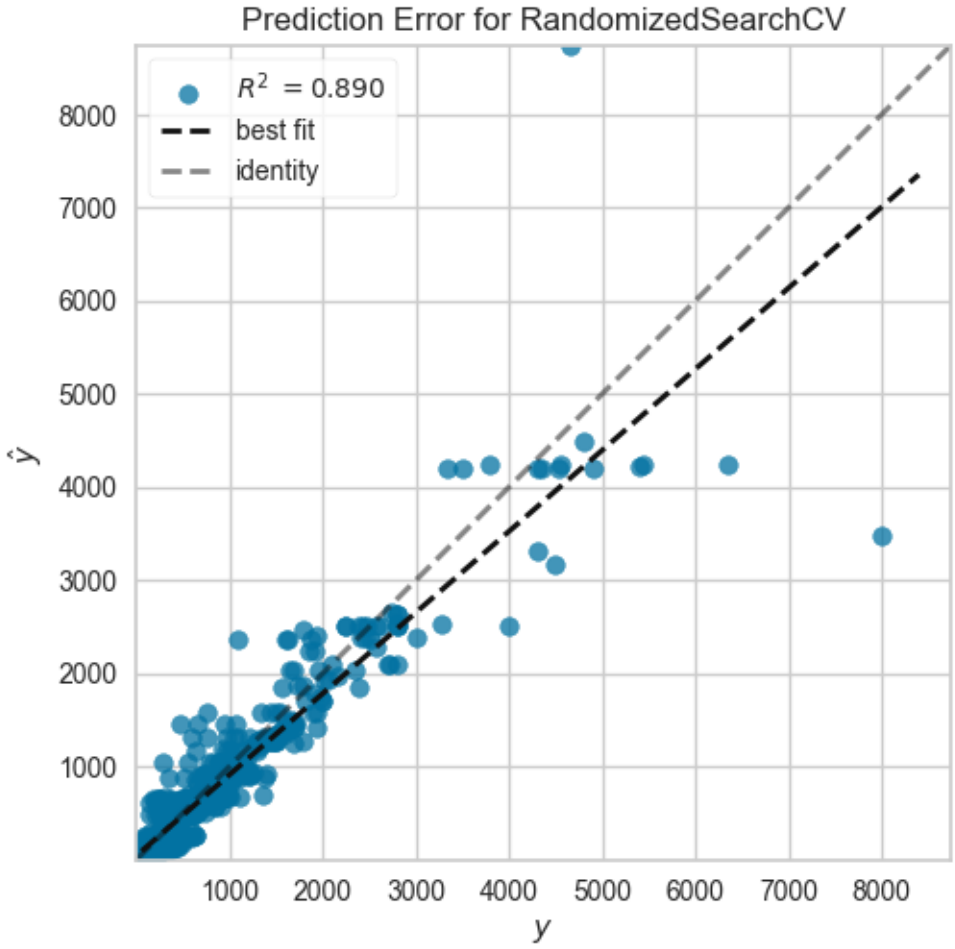


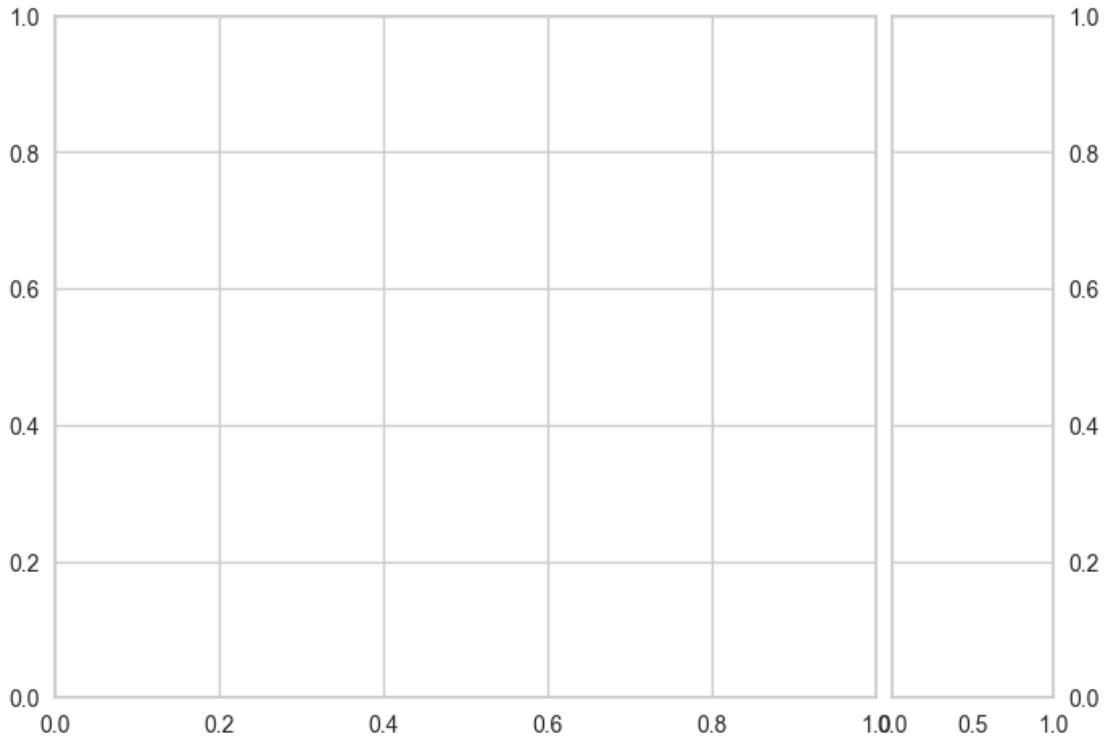
[185]:
```
param_grid = {
    'n_estimators': [200,500,800,1000],
    'learning_rate': [0.001,0.01,0.1,1],
    'loss': ['linear','square','exponential']
}

grid_ab =␣
  ↪RandomizedSearchCV(estimator=AdaBoostRegressor(),param_distributions=param_grid,cv=5,verbos
train_and_evaluate_model(grid_ab)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END ..learning_rate=0.01, loss=linear, n_estimators=200; total time=   4.0s
[CV] END ..learning_rate=0.01, loss=linear, n_estimators=200; total time=   3.5s
[CV] END ..learning_rate=0.01, loss=linear, n_estimators=200; total time=   3.4s
[CV] END ..learning_rate=0.01, loss=linear, n_estimators=200; total time=   3.4s
[CV] END ..learning_rate=0.01, loss=linear, n_estimators=200; total time=   3.5s
[CV] END …learning_rate=1, loss=square, n_estimators=1000; total time=  12.8s
[CV] END …learning_rate=1, loss=square, n_estimators=1000; total time=  13.7s
[CV] END …learning_rate=1, loss=square, n_estimators=1000; total time=  12.6s
[CV] END …learning_rate=1, loss=square, n_estimators=1000; total time=  12.7s
[CV] END …learning_rate=1, loss=square, n_estimators=1000; total time=  12.7s
[CV] END learning_rate=0.001, loss=exponential, n_estimators=800; total time=
14.9s
```

```
[CV] END learning_rate=0.001, loss=exponential, n_estimators=800; total time=
15.3s
[CV] END learning_rate=0.001, loss=exponential, n_estimators=800; total time=
14.7s
[CV] END learning_rate=0.001, loss=exponential, n_estimators=800; total time=
14.8s
[CV] END learning_rate=0.001, loss=exponential, n_estimators=800; total time=
14.8s
[CV] END …learning_rate=1, loss=square, n_estimators=200; total time=   2.5s
[CV] END …learning_rate=1, loss=square, n_estimators=200; total time=   2.6s
[CV] END …learning_rate=1, loss=square, n_estimators=200; total time=   2.5s
[CV] END …learning_rate=1, loss=square, n_estimators=200; total time=   2.5s
[CV] END …learning_rate=1, loss=square, n_estimators=200; total time=   2.6s
[CV] END ..learning_rate=0.1, loss=linear, n_estimators=1000; total time=  14.1s
[CV] END ..learning_rate=0.1, loss=linear, n_estimators=1000; total time=  13.6s
[CV] END ..learning_rate=0.1, loss=linear, n_estimators=1000; total time=  13.9s
[CV] END ..learning_rate=0.1, loss=linear, n_estimators=1000; total time=  11.8s
[CV] END ..learning_rate=0.1, loss=linear, n_estimators=1000; total time=   3.5s
[CV] END learning_rate=0.01, loss=exponential, n_estimators=800; total time=
15.7s
[CV] END learning_rate=0.01, loss=exponential, n_estimators=800; total time=
14.2s
[CV] END learning_rate=0.01, loss=exponential, n_estimators=800; total time=
14.2s
[CV] END learning_rate=0.01, loss=exponential, n_estimators=800; total time=
14.3s
[CV] END learning_rate=0.01, loss=exponential, n_estimators=800; total time=
14.9s
[CV] END .learning_rate=0.001, loss=linear, n_estimators=500; total time=   9.3s
[CV] END .learning_rate=0.001, loss=linear, n_estimators=500; total time=   8.6s
[CV] END .learning_rate=0.001, loss=linear, n_estimators=500; total time=   8.5s
[CV] END .learning_rate=0.001, loss=linear, n_estimators=500; total time=   8.8s
[CV] END .learning_rate=0.001, loss=linear, n_estimators=500; total time=   8.8s
[CV] END …learning_rate=1, loss=linear, n_estimators=800; total time=  10.2s
[CV] END …learning_rate=1, loss=linear, n_estimators=800; total time=  10.7s
[CV] END …learning_rate=1, loss=linear, n_estimators=800; total time=  10.6s
[CV] END …learning_rate=1, loss=linear, n_estimators=800; total time=   1.1s
[CV] END …learning_rate=1, loss=linear, n_estimators=800; total time=   0.6s
[CV] END .learning_rate=0.001, loss=linear, n_estimators=200; total time=   3.4s
[CV] END .learning_rate=0.001, loss=linear, n_estimators=200; total time=   3.5s
[CV] END .learning_rate=0.001, loss=linear, n_estimators=200; total time=   3.4s
[CV] END .learning_rate=0.001, loss=linear, n_estimators=200; total time=   3.4s
[CV] END .learning_rate=0.001, loss=linear, n_estimators=200; total time=   3.4s
[CV] END ..learning_rate=0.01, loss=square, n_estimators=800; total time=  13.3s
[CV] END ..learning_rate=0.01, loss=square, n_estimators=800; total time=  13.2s
[CV] END ..learning_rate=0.01, loss=square, n_estimators=800; total time=  13.2s
[CV] END ..learning_rate=0.01, loss=square, n_estimators=800; total time=  13.6s
[CV] END ..learning_rate=0.01, loss=square, n_estimators=800; total time=  13.3s
```

```
Mean Absolute Error: 83.75897579016292
Mean Squared Error: 29672.7299463715
Root Mean Squared Error: 172.25774277625808
Mean Absolute Percentage Error: 3.7471147820762267
R2 Score: 0.8895937953599096
Training Time: 479.21917819976807
```





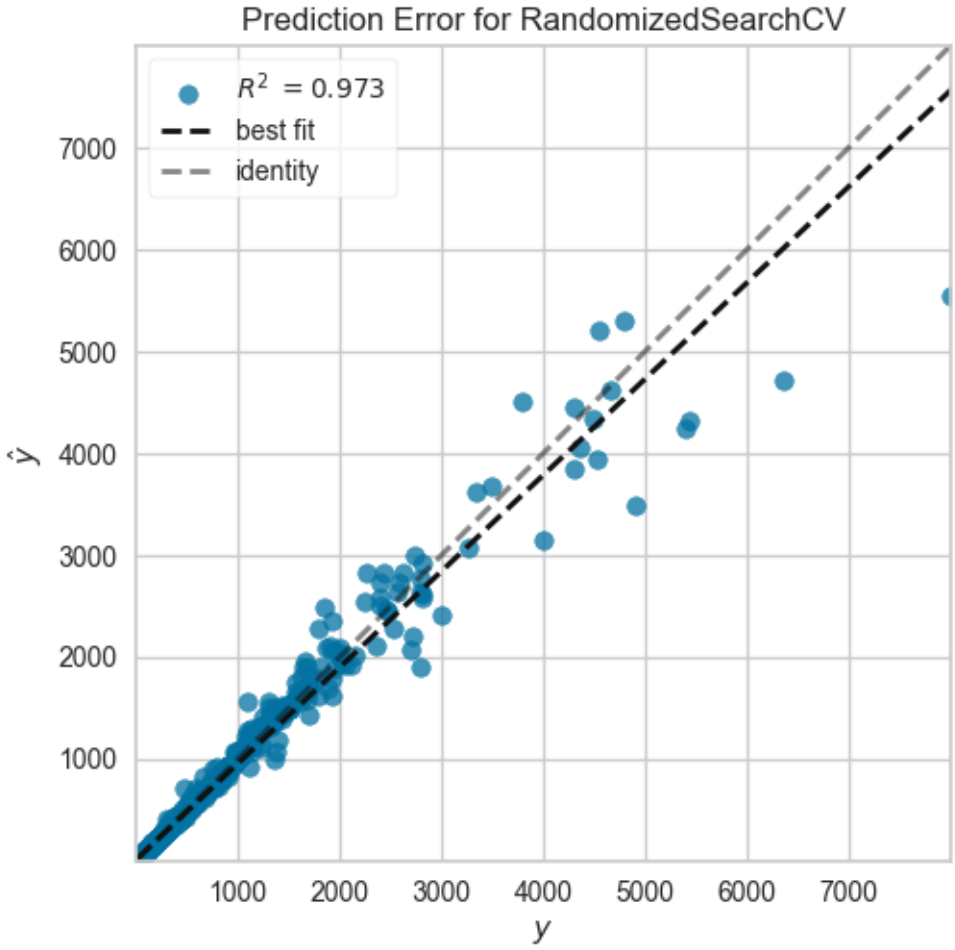
[186]:
```
param_grid = {
    'n_estimators': [200,500,800,1000],
    'criterion': ['squared_error','absolute_error','friedman_mse','poisson'],
    'max_features': ['auto','sqrt','log2'],
    'bootstrap': [True,False],
    'oob_score': [True,False],
    'max_samples': [0.5,0.75,0.9,1]
}

grid_et = ␣
  ↪RandomizedSearchCV(estimator=ExtraTreesRegressor(),param_distributions=param_grid,cv=5,verb
train_and_evaluate_model(grid_et)
```

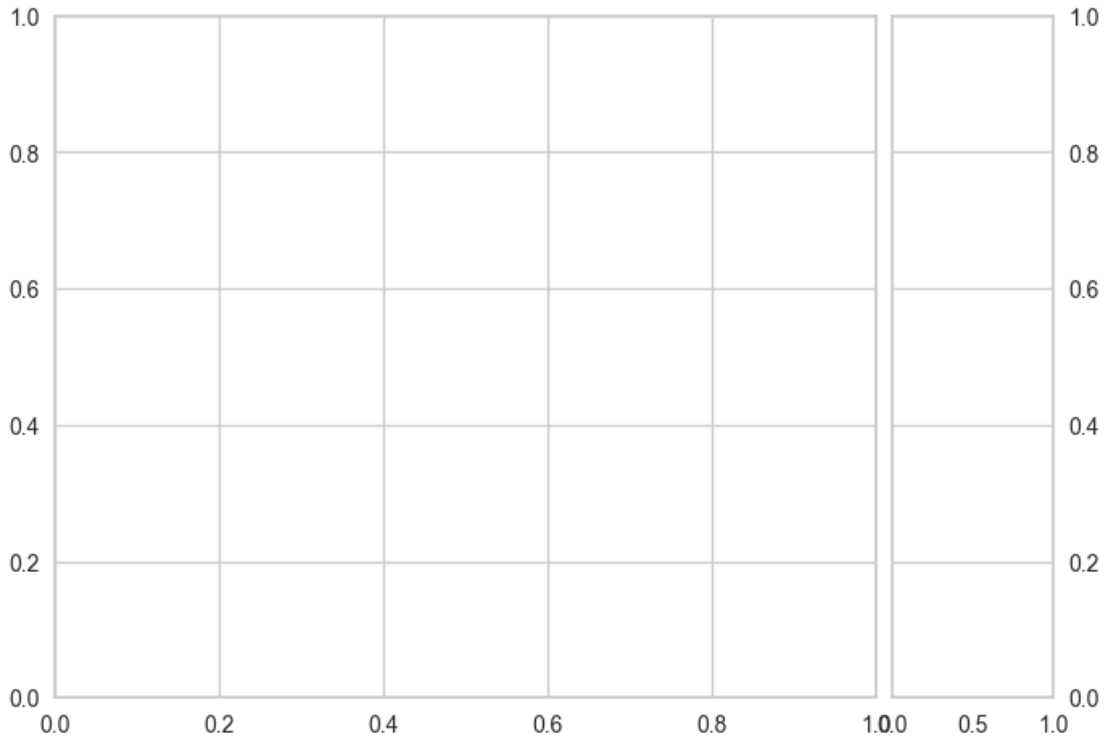
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END bootstrap=False, criterion=poisson, max_features=auto, max_samples=0.9,
n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=auto, max_samples=0.9,
n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=auto, max_samples=0.9,
n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=auto, max_samples=0.9,
n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=auto, max_samples=0.9,
```

```
n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=sqrt,
max_samples=0.5, n_estimators=1000, oob_score=True; total time=   9.0s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=sqrt,
max_samples=0.5, n_estimators=1000, oob_score=True; total time=   8.1s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=sqrt,
max_samples=0.5, n_estimators=1000, oob_score=True; total time=   7.9s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=sqrt,
max_samples=0.5, n_estimators=1000, oob_score=True; total time=   8.1s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=sqrt,
max_samples=0.5, n_estimators=1000, oob_score=True; total time=   7.2s
[CV] END bootstrap=True, criterion=poisson, max_features=sqrt, max_samples=0.5,
n_estimators=1000, oob_score=False; total time=   7.8s
[CV] END bootstrap=True, criterion=poisson, max_features=sqrt, max_samples=0.5,
n_estimators=1000, oob_score=False; total time=   6.9s
[CV] END bootstrap=True, criterion=poisson, max_features=sqrt, max_samples=0.5,
n_estimators=1000, oob_score=False; total time=   8.0s
[CV] END bootstrap=True, criterion=poisson, max_features=sqrt, max_samples=0.5,
n_estimators=1000, oob_score=False; total time=   7.9s
[CV] END bootstrap=True, criterion=poisson, max_features=sqrt, max_samples=0.5,
n_estimators=1000, oob_score=False; total time=   8.1s
[CV] END bootstrap=True, criterion=poisson, max_features=sqrt, max_samples=1,
n_estimators=500, oob_score=True; total time=   1.4s
[CV] END bootstrap=True, criterion=poisson, max_features=sqrt, max_samples=1,
n_estimators=500, oob_score=True; total time=   0.9s
[CV] END bootstrap=True, criterion=poisson, max_features=sqrt, max_samples=1,
n_estimators=500, oob_score=True; total time=   0.8s
[CV] END bootstrap=True, criterion=poisson, max_features=sqrt, max_samples=1,
n_estimators=500, oob_score=True; total time=   1.6s
[CV] END bootstrap=True, criterion=poisson, max_features=sqrt, max_samples=1,
n_estimators=500, oob_score=True; total time=   0.7s
[CV] END bootstrap=False, criterion=poisson, max_features=log2,
max_samples=0.75, n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2,
max_samples=0.75, n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2,
max_samples=0.75, n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2,
max_samples=0.75, n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2,
max_samples=0.75, n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=sqrt,
max_samples=0.75, n_estimators=200, oob_score=True; total time=   1.4s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=sqrt,
max_samples=0.75, n_estimators=200, oob_score=True; total time=   1.4s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=sqrt,
max_samples=0.75, n_estimators=200, oob_score=True; total time=   1.4s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=sqrt,
```

```
max_samples=0.75, n_estimators=200, oob_score=True; total time=   1.4s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=sqrt,
max_samples=0.75, n_estimators=200, oob_score=True; total time=   2.4s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
max_samples=0.75, n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
max_samples=0.75, n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
max_samples=0.75, n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
max_samples=0.75, n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
max_samples=0.75, n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=True, criterion=poisson, max_features=auto, max_samples=0.75,
n_estimators=200, oob_score=False; total time=   3.8s
[CV] END bootstrap=True, criterion=poisson, max_features=auto, max_samples=0.75,
n_estimators=200, oob_score=False; total time=   3.5s
[CV] END bootstrap=True, criterion=poisson, max_features=auto, max_samples=0.75,
n_estimators=200, oob_score=False; total time=   2.6s
[CV] END bootstrap=True, criterion=poisson, max_features=auto, max_samples=0.75,
n_estimators=200, oob_score=False; total time=   3.4s
[CV] END bootstrap=True, criterion=poisson, max_features=auto, max_samples=0.75,
n_estimators=200, oob_score=False; total time=   3.0s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=auto,
max_samples=1, n_estimators=800, oob_score=True; total time=   1.8s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=auto,
max_samples=1, n_estimators=800, oob_score=True; total time=   2.2s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=auto,
max_samples=1, n_estimators=800, oob_score=True; total time=   1.5s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=auto,
max_samples=1, n_estimators=800, oob_score=True; total time=   2.0s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=auto,
max_samples=1, n_estimators=800, oob_score=True; total time=   2.1s
[CV] END bootstrap=False, criterion=squared_error, max_features=log2,
max_samples=0.9, n_estimators=800, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=log2,
max_samples=0.9, n_estimators=800, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=log2,
max_samples=0.9, n_estimators=800, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=log2,
max_samples=0.9, n_estimators=800, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=log2,
max_samples=0.9, n_estimators=800, oob_score=False; total time=   0.0s
Mean Absolute Error: 12.540924698365034
Mean Squared Error: 7296.108216823835
Root Mean Squared Error: 85.41725947853769
Mean Absolute Percentage Error: 0.017916232540110155
R2 Score: 0.972852662417015
```

```
Training Time: 125.16977620124817
```





[187]:
```
param_grid = {
    'n_estimators': [200,500,800,1000],
    'learning_rate': [0.001,0.01,0.1,1],
    'booster': ['gbtree','dart','gblinear'],
    'importance_type': ['gain','weight','cover','total_gain','total_cover']
}

grid_xgbrf = ␣
  ↪RandomizedSearchCV(estimator=XGBRFRegressor(),param_distributions=param_grid,cv=5,verbose=2
train_and_evaluate_model(grid_xgbrf)
```

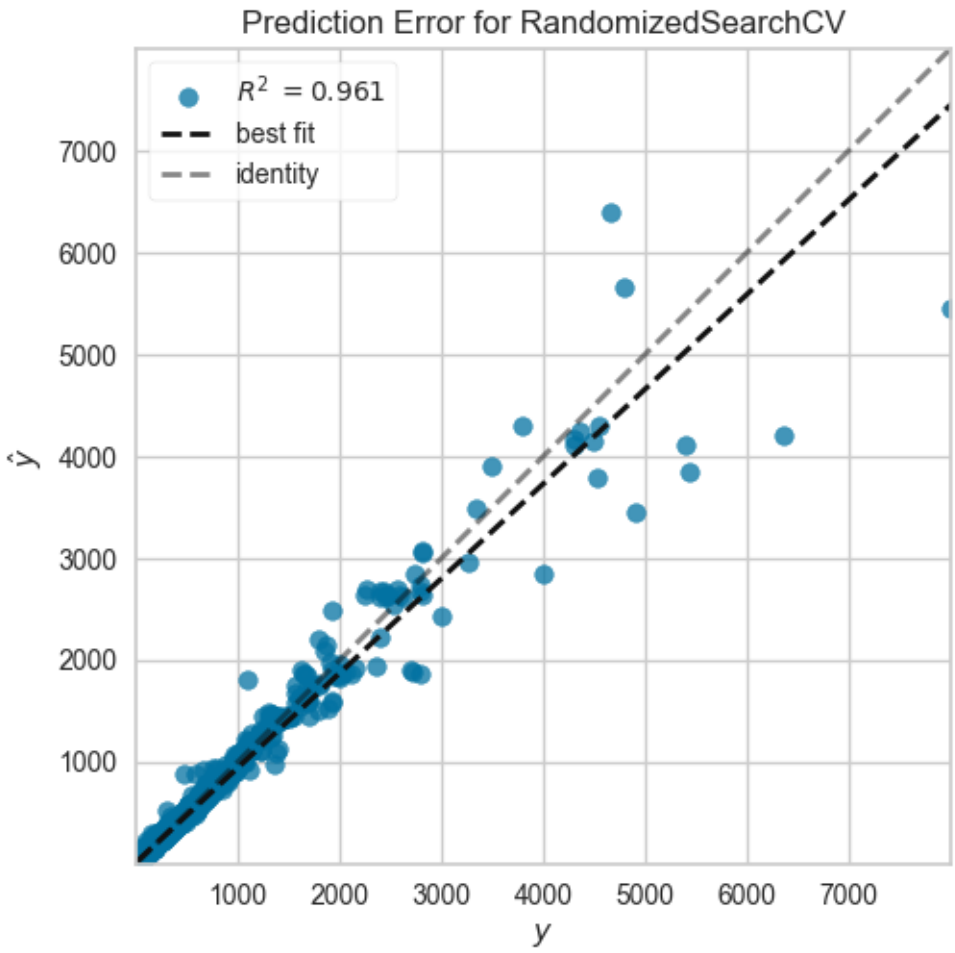
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[03:27:34] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=total_cover, learning_rate=0.01,
n_estimators=500; total time=   0.2s
[03:27:34] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.
```

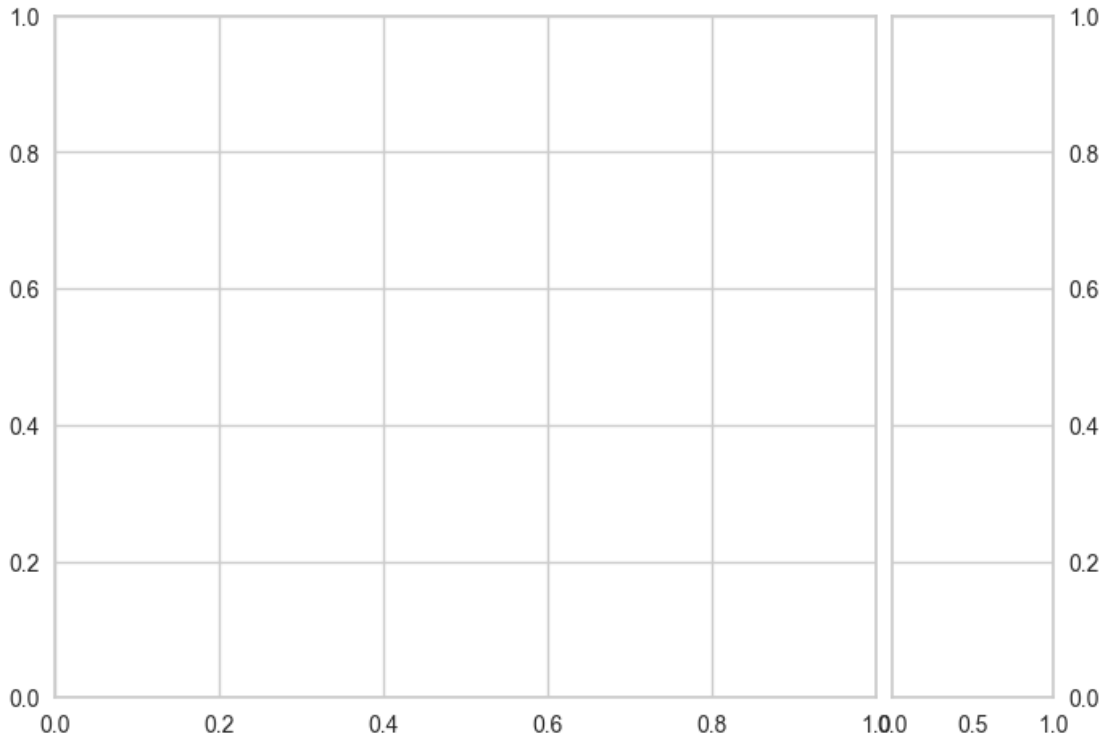
```
[CV] END booster=gblinear, importance_type=total_cover, learning_rate=0.01,
n_estimators=500; total time=   0.0s
[03:27:34] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=total_cover, learning_rate=0.01,
n_estimators=500; total time=   0.0s
[03:27:34] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=total_cover, learning_rate=0.01,
n_estimators=500; total time=   0.0s
[03:27:34] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=total_cover, learning_rate=0.01,
n_estimators=500; total time=   0.0s
[CV] END booster=dart, importance_type=gain, learning_rate=1, n_estimators=200;
total time=   0.6s
[CV] END booster=dart, importance_type=gain, learning_rate=1, n_estimators=200;
total time=   0.6s
[CV] END booster=dart, importance_type=gain, learning_rate=1, n_estimators=200;
total time=   1.0s
[CV] END booster=dart, importance_type=gain, learning_rate=1, n_estimators=200;
total time=   1.0s
[CV] END booster=dart, importance_type=gain, learning_rate=1, n_estimators=200;
total time=   0.6s
[03:27:39] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=weight, learning_rate=0.01,
n_estimators=200; total time=   0.0s
[03:27:39] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=weight, learning_rate=0.01,
n_estimators=200; total time=   0.0s
```

```
[03:27:39] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=weight, learning_rate=0.01,
n_estimators=200; total time=   0.0s
[03:27:39] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=weight, learning_rate=0.01,
n_estimators=200; total time=   0.0s
[03:27:39] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=weight, learning_rate=0.01,
n_estimators=200; total time=   0.0s
[CV] END booster=dart, importance_type=gain, learning_rate=1, n_estimators=800;
total time=   2.4s
[CV] END booster=dart, importance_type=gain, learning_rate=1, n_estimators=800;
total time=   2.7s
[CV] END booster=dart, importance_type=gain, learning_rate=1, n_estimators=800;
total time=   3.3s
[CV] END booster=dart, importance_type=gain, learning_rate=1, n_estimators=800;
total time=   3.5s
[CV] END booster=dart, importance_type=gain, learning_rate=1, n_estimators=800;
total time=   3.4s
[CV] END booster=gbtree, importance_type=weight, learning_rate=1,
n_estimators=500; total time=   2.3s
[CV] END booster=gbtree, importance_type=weight, learning_rate=1,
n_estimators=500; total time=   1.5s
[CV] END booster=gbtree, importance_type=weight, learning_rate=1,
n_estimators=500; total time=   0.9s
[CV] END booster=gbtree, importance_type=weight, learning_rate=1,
n_estimators=500; total time=   2.4s
[CV] END booster=gbtree, importance_type=weight, learning_rate=1,
n_estimators=500; total time=   2.5s
[CV] END booster=gbtree, importance_type=total_gain, learning_rate=0.001,
n_estimators=500; total time=   1.4s
[CV] END booster=gbtree, importance_type=total_gain, learning_rate=0.001,
n_estimators=500; total time=   1.1s
[CV] END booster=gbtree, importance_type=total_gain, learning_rate=0.001,
n_estimators=500; total time=   2.6s
[CV] END booster=gbtree, importance_type=total_gain, learning_rate=0.001,
```

```
n_estimators=500; total time=   2.9s
[CV] END booster=gbtree, importance_type=total_gain, learning_rate=0.001,
n_estimators=500; total time=   1.1s
[CV] END booster=dart, importance_type=weight, learning_rate=0.001,
n_estimators=500; total time=   2.3s
[CV] END booster=dart, importance_type=weight, learning_rate=0.001,
n_estimators=500; total time=   2.4s
[CV] END booster=dart, importance_type=weight, learning_rate=0.001,
n_estimators=500; total time=   2.4s
[CV] END booster=dart, importance_type=weight, learning_rate=0.001,
n_estimators=500; total time=   1.2s
[CV] END booster=dart, importance_type=weight, learning_rate=0.001,
n_estimators=500; total time=   1.7s
[CV] END booster=dart, importance_type=total_cover, learning_rate=1,
n_estimators=1000; total time=   4.8s
[CV] END booster=dart, importance_type=total_cover, learning_rate=1,
n_estimators=1000; total time=   3.1s
[CV] END booster=dart, importance_type=total_cover, learning_rate=1,
n_estimators=1000; total time=   4.5s
[CV] END booster=dart, importance_type=total_cover, learning_rate=1,
n_estimators=1000; total time=   3.8s
[CV] END booster=dart, importance_type=total_cover, learning_rate=1,
n_estimators=1000; total time=   4.9s
[CV] END booster=dart, importance_type=cover, learning_rate=1, n_estimators=800;
total time=   2.2s
[CV] END booster=dart, importance_type=cover, learning_rate=1, n_estimators=800;
total time=   4.0s
[CV] END booster=dart, importance_type=cover, learning_rate=1, n_estimators=800;
total time=   2.3s
[CV] END booster=dart, importance_type=cover, learning_rate=1, n_estimators=800;
total time=   4.6s
[CV] END booster=dart, importance_type=cover, learning_rate=1, n_estimators=800;
total time=   4.5s
[03:29:04] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=total_cover, learning_rate=0.001,
n_estimators=200; total time=   0.0s
[03:29:04] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=total_cover, learning_rate=0.001,
n_estimators=200; total time=   0.0s
[03:29:05] WARNING: C:\Users\dev-admin\croot2\xgboost-
```

```
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=total_cover, learning_rate=0.001,
n_estimators=200; total time=   0.0s
[03:29:05] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=total_cover, learning_rate=0.001,
n_estimators=200; total time=   0.0s
[03:29:05] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "colsample_bynode", "num_parallel_tree", "subsample" } are not
used.

[CV] END booster=gblinear, importance_type=total_cover, learning_rate=0.001,
n_estimators=200; total time=   0.0s
Mean Absolute Error: 21.32384239605155
Mean Squared Error: 10522.195561941786
Root Mean Squared Error: 102.57775373803906
Mean Absolute Percentage Error: 0.3268403518343805
R2 Score: 0.9608490462935361
Training Time: 92.0232880115509
```


Prediction Error for RandomizedSearchCV
$R^2 = 0.961$
best fit
identity
7000
6000
5000
4000
3000
2000
1000
$\hat{y}$
1000
2000
3000
4000
5000
6000
7000
y



[188]:
```
param_grid = {
    'n_estimators': [200,500,800,1000],
    'max_samples': [0.25,0.5,0.75,1],
    'max_features': [0.5,0.65,0.85,1],
    'bootstrap': [True,False],
    'oob_score': [True,False],
    'bootstrap_features': [True,False],
    'warm_start': [True,False]
}

grid_bag =␣
 ↪RandomizedSearchCV(estimator=BaggingRegressor(),param_distributions=param_grid,cv=5,verbose
train_and_evaluate_model(grid_bag)
```

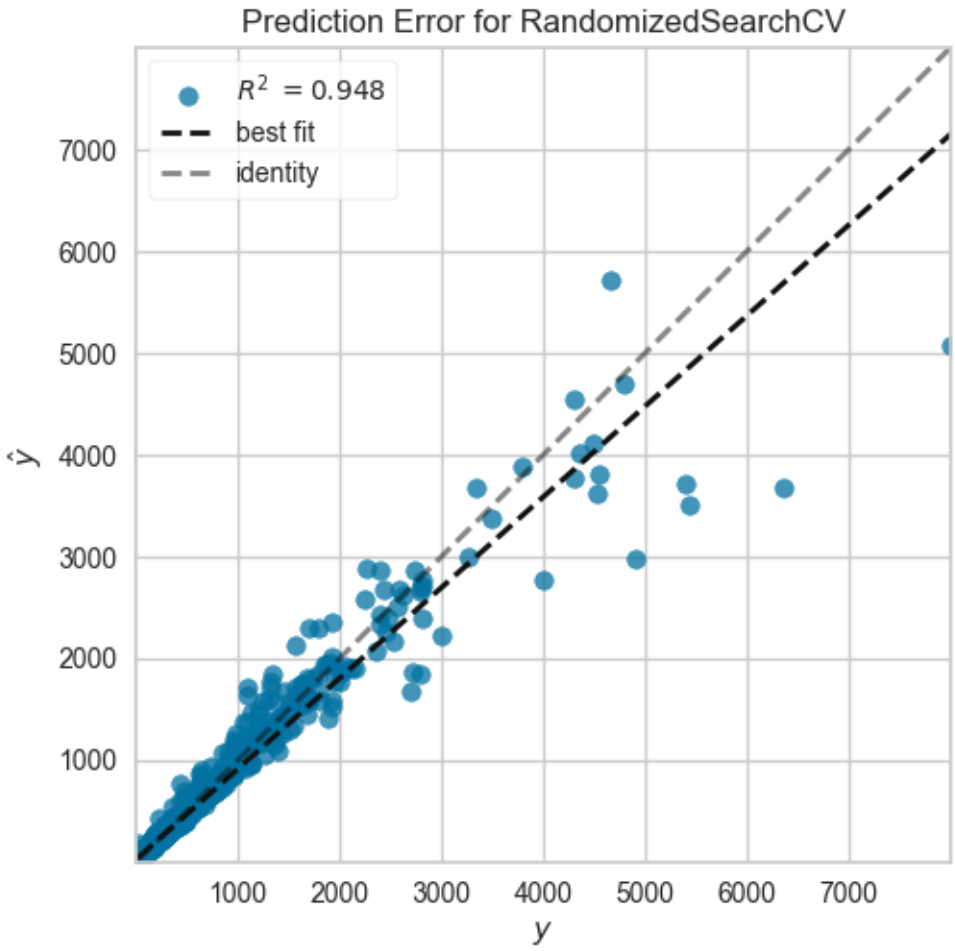
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.5,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=True; total time=
4.4s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.5,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=True; total time=
4.1s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.5,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=True; total time=
```

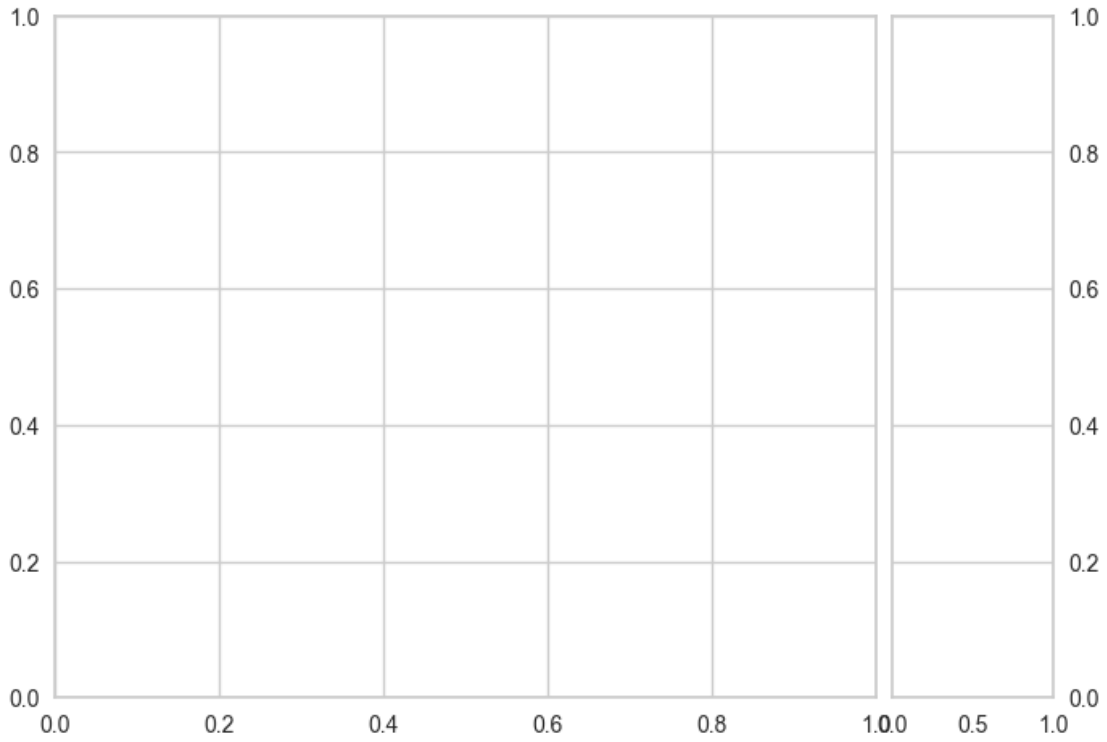
```
5.0s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.5,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=True; total time=
4.0s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.5,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=True; total time=
3.9s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.65,
max_samples=0.5, n_estimators=1000, oob_score=False, warm_start=True; total
time=  22.8s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.65,
max_samples=0.5, n_estimators=1000, oob_score=False, warm_start=True; total
time=  21.8s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.65,
max_samples=0.5, n_estimators=1000, oob_score=False, warm_start=True; total
time=  21.2s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.65,
max_samples=0.5, n_estimators=1000, oob_score=False, warm_start=True; total
time=  20.9s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.65,
max_samples=0.5, n_estimators=1000, oob_score=False, warm_start=True; total
time=  22.1s
[CV] END bootstrap=True, bootstrap_features=True, max_features=0.65,
max_samples=0.25, n_estimators=200, oob_score=True, warm_start=True; total time=
0.0s
[CV] END bootstrap=True, bootstrap_features=True, max_features=0.65,
max_samples=0.25, n_estimators=200, oob_score=True, warm_start=True; total time=
0.0s
[CV] END bootstrap=True, bootstrap_features=True, max_features=0.65,
max_samples=0.25, n_estimators=200, oob_score=True, warm_start=True; total time=
0.0s
[CV] END bootstrap=True, bootstrap_features=True, max_features=0.65,
max_samples=0.25, n_estimators=200, oob_score=True, warm_start=True; total time=
0.0s
[CV] END bootstrap=True, bootstrap_features=True, max_features=0.65,
max_samples=0.25, n_estimators=200, oob_score=True, warm_start=True; total time=
0.0s
[CV] END bootstrap=True, bootstrap_features=False, max_features=0.5,
max_samples=0.75, n_estimators=200, oob_score=False, warm_start=True; total
time=   4.0s
[CV] END bootstrap=True, bootstrap_features=False, max_features=0.5,
max_samples=0.75, n_estimators=200, oob_score=False, warm_start=True; total
time=   3.6s
[CV] END bootstrap=True, bootstrap_features=False, max_features=0.5,
max_samples=0.75, n_estimators=200, oob_score=False, warm_start=True; total
time=   3.5s
[CV] END bootstrap=True, bootstrap_features=False, max_features=0.5,
max_samples=0.75, n_estimators=200, oob_score=False, warm_start=True; total
```

```
time=   4.6s
[CV] END bootstrap=True, bootstrap_features=False, max_features=0.5,
max_samples=0.75, n_estimators=200, oob_score=False, warm_start=True; total
time=   4.2s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.65,
max_samples=0.25, n_estimators=1000, oob_score=False, warm_start=True; total
time=  10.7s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.65,
max_samples=0.25, n_estimators=1000, oob_score=False, warm_start=True; total
time=  10.8s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.65,
max_samples=0.25, n_estimators=1000, oob_score=False, warm_start=True; total
time=  11.8s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.65,
max_samples=0.25, n_estimators=1000, oob_score=False, warm_start=True; total
time=  13.0s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.65,
max_samples=0.25, n_estimators=1000, oob_score=False, warm_start=True; total
time=  10.7s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.5,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=False; total
time=   3.4s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.5,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=False; total
time=   4.9s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.5,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=False; total
time=   3.0s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.5,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=False; total
time=   4.0s
[CV] END bootstrap=False, bootstrap_features=True, max_features=0.5,
max_samples=0.5, n_estimators=200, oob_score=False, warm_start=False; total
time=   3.0s
[CV] END bootstrap=False, bootstrap_features=False, max_features=0.5,
max_samples=1, n_estimators=200, oob_score=True, warm_start=True; total time=
0.0s
[CV] END bootstrap=False, bootstrap_features=False, max_features=0.5,
max_samples=1, n_estimators=200, oob_score=True, warm_start=True; total time=
0.0s
[CV] END bootstrap=False, bootstrap_features=False, max_features=0.5,
max_samples=1, n_estimators=200, oob_score=True, warm_start=True; total time=
0.0s
[CV] END bootstrap=False, bootstrap_features=False, max_features=0.5,
max_samples=1, n_estimators=200, oob_score=True, warm_start=True; total time=
0.0s
[CV] END bootstrap=False, bootstrap_features=False, max_features=0.5,
max_samples=1, n_estimators=200, oob_score=True, warm_start=True; total time=
```

```
0.0s
[CV] END bootstrap=True, bootstrap_features=False, max_features=0.85,
max_samples=0.75, n_estimators=200, oob_score=True, warm_start=False; total
time=   6.1s
[CV] END bootstrap=True, bootstrap_features=False, max_features=0.85,
max_samples=0.75, n_estimators=200, oob_score=True, warm_start=False; total
time=   6.0s
[CV] END bootstrap=True, bootstrap_features=False, max_features=0.85,
max_samples=0.75, n_estimators=200, oob_score=True, warm_start=False; total
time=   6.3s
[CV] END bootstrap=True, bootstrap_features=False, max_features=0.85,
max_samples=0.75, n_estimators=200, oob_score=True, warm_start=False; total
time=   6.0s
[CV] END bootstrap=True, bootstrap_features=False, max_features=0.85,
max_samples=0.75, n_estimators=200, oob_score=True, warm_start=False; total
time=   6.1s
[CV] END bootstrap=False, bootstrap_features=False, max_features=0.5,
max_samples=0.75, n_estimators=1000, oob_score=True, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, bootstrap_features=False, max_features=0.5,
max_samples=0.75, n_estimators=1000, oob_score=True, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, bootstrap_features=False, max_features=0.5,
max_samples=0.75, n_estimators=1000, oob_score=True, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, bootstrap_features=False, max_features=0.5,
max_samples=0.75, n_estimators=1000, oob_score=True, warm_start=True; total
time=   0.0s
[CV] END bootstrap=False, bootstrap_features=False, max_features=0.5,
max_samples=0.75, n_estimators=1000, oob_score=True, warm_start=True; total
time=   0.0s
[CV] END bootstrap=True, bootstrap_features=True, max_features=0.5,
max_samples=1, n_estimators=500, oob_score=True, warm_start=False; total time=
1.9s
[CV] END bootstrap=True, bootstrap_features=True, max_features=0.5,
max_samples=1, n_estimators=500, oob_score=True, warm_start=False; total time=
0.9s
[CV] END bootstrap=True, bootstrap_features=True, max_features=0.5,
max_samples=1, n_estimators=500, oob_score=True, warm_start=False; total time=
0.9s
[CV] END bootstrap=True, bootstrap_features=True, max_features=0.5,
max_samples=1, n_estimators=500, oob_score=True, warm_start=False; total time=
1.0s
[CV] END bootstrap=True, bootstrap_features=True, max_features=0.5,
max_samples=1, n_estimators=500, oob_score=True, warm_start=False; total time=
1.6s
Mean Absolute Error: 25.651986015467866
Mean Squared Error: 14047.290302725823
```

```
Root Mean Squared Error: 118.52126519205667
Mean Absolute Percentage Error: 0.3433791575828548
R2 Score: 0.9477328843485411
Training Time: 272.380309343338
```





0.9 Optimized Models Performance Comparison

[194]:
```
model_perfs = pd.DataFrame({'Model': models,
                            'R2': r2_scores,
                            'RMSE': rmse_scores,
                            'MAPE': mape_scores,
                            'MSE': mse_scores,
                            'Training Time': training_times}).
 ↪sort_values('R2',ascending=False).reset_index(drop=True)
model_perfs[:20]
```

[194]:

	Model	R2	RMSE	\
0	(ExtraTreeRegressor(random_state=1029196382), …	0.976337	79.747890	
1	RandomizedSearchCV(cv=5, estimator=ExtraTreesR…	0.972853	85.417259	
2	RandomizedSearchCV(cv=5, estimator=GradientBoo…	0.969176	91.018475	
3	RandomizedSearchCV(estimator=RandomForestRegre…	0.964296	97.957550	
4	(DecisionTreeRegressor(max_features=1.0, rando…	0.961060	102.300450	
5	RandomizedSearchCV(cv=5,\n e…	0.960849	102.577754	
6	<catboost.core.CatBoostRegressor object at 0x0…	0.960660	102.824565	
7	(DecisionTreeRegressor(random_state=2064953117…	0.960326	103.260716	
8	RandomizedSearchCV(cv=5, estimator=LGBMRegress…	0.959694	104.080024	
9	LGBMRegressor()	0.959196	104.720411	
10	XGBRFRegressor(base_score=None, booster=None, …	0.957665	106.667687	

```
11  RandomizedSearchCV(cv=5, estimator=TweedieRegr…  0.957075  107.408279
12                            DecisionTreeRegressor()  0.950919  114.852593
13                  HistGradientBoostingRegressor()  0.950026  115.892593
14  ([DecisionTreeRegressor(criterion='friedman_ms…  0.949795  116.159447
15  RandomizedSearchCV(cv=5,\n                    e…  0.949742  116.221011
16  NGBRegressor(random_state=RandomState(MT19937)…  0.949426  116.585414
17  RandomizedSearchCV(cv=5, estimator=BaggingRegr…  0.947733  118.521265
18  RandomizedSearchCV(cv=5, estimator=HistGradien…  0.946664  119.726734
19  XGBRegressor(base_score=None, booster=None, ca…  0.936781  130.348773
```

	MAPE	MSE	Training Time
0	0.014177	6359.725944	1.895131
1	0.017916	7296.108217	125.169776
2	0.201590	8284.362773	262.948552
3	0.019701	9595.681569	339.901599
4	0.019037	10465.381975	4.487048
5	0.326840	10522.195562	92.023288
6	0.119310	10572.891102	4.655082
7	0.022796	10662.775436	0.454965
8	0.124777	10832.651487	19.581051
9	0.105570	10966.364507	0.207914
10	0.326936	11377.995442	0.261350
11	0.565001	11536.538349	0.871818
12	0.029598	13191.118209	0.062676
13	0.100493	13431.093179	0.565095
14	0.188744	13493.017227	1.191292
15	0.212544	13507.323309	104.701008
16	0.168340	13592.158650	16.195783
17	0.343379	14047.290303	272.380309
18	0.071681	14334.490768	15.922371
19	0.083650	16990.802533	0.278330

Even after tuning the hyperparameters of all the models, the Extra Trees Regressor model stays intact as the best performing model with an outstanding R2 score of more than 97.6%.

0.10 Saving the best performing model for deployment into production

[195]:
```
best_model = model_perfs['Model'].iloc[0]
best_model
```

[195]:
```
ExtraTreesRegressor()
```

[196]:
```
pipeline = Pipeline(steps=[
        ('transformer',transformer),
        ('scaler',scaler),
        ('model',best_model)
])
pipeline
```

[196]:
```
Pipeline(steps=[('transformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('log_transform',
FunctionTransformer(func=<ufunc 'log1p'>),
                                                  ['Quantity', 'Profit']),
                                                 ('sqrt_transform',
FunctionTransformer(func=<ufunc 'sqrt'>),
                                                  ['Discount']),
                                                 ('power_transform',
                                                  PowerTransformer(),
                                                  ['Discount Percentage',
                                                   'Operating Expenses'])])),
                ('scaler', StandardScaler()),
                ('model', ExtraTreesRegressor())])
```

[197]:
```
joblib.dump(pipeline,'pipeline.pkl')
```

[197]:
```
['pipeline.pkl']
```

[198]:
```
joblib.load('pipeline.pkl')
```

[198]:
```
Pipeline(steps=[('transformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('log_transform',
FunctionTransformer(func=<ufunc 'log1p'>),
                                                  ['Quantity', 'Profit']),
                                                 ('sqrt_transform',
FunctionTransformer(func=<ufunc 'sqrt'>),
                                                  ['Discount']),
                                                 ('power_transform',
                                                  PowerTransformer(),
                                                  ['Discount Percentage',
                                                   'Operating Expenses'])])),
                ('scaler', StandardScaler()),
                ('model', ExtraTreesRegressor())])
```

[200]:
```
for col in final_X_train.columns:
    print(f"Min value of {col}:",df[col].min())
    print(f"Max value of {col}:",df[col].max())
```

```
Min value of Profit: -6599.978000000001
Max value of Profit: 5039.9856
Min value of Discount Percentage: 0.0
Max value of Discount Percentage: 180.1801801801802
Min value of Postal Code: 1040.0
Max value of Postal Code: 99301.0
Min value of Discount: 0.0
Max value of Discount: 0.8
```

```
Min value of Order Day: 1.0
Max value of Order Day: 31.0
Min value of Order Month: 1.0
Max value of Order Month: 12.0
Min value of Quantity: 1.0
Max value of Quantity: 14.0
Min value of Operating Expenses: 0.5543999999999999
Max value of Operating Expenses: 11839.970399999998
Min value of Ship Day: 1.0
Max value of Ship Day: 31.0
Min value of Order Weekday: 0.0
Max value of Order Weekday: 6.0
```

[205]:
```
list(final_X_train.columns)
```

[205]:
```
['Profit',
 'Discount Percentage',
 'Postal Code',
 'Discount',
 'Order Day',
 'Order Month',
 'Quantity',
 'Operating Expenses',
 'Ship Day',
 'Order Weekday']
```

0.10.1 Making a test prediction

[208]:
```
pipeline.predict(pd.DataFrame([[3846.91,79.42479138,5937,0.5,8.0,5.0,9.0,6293.
↪46,29.0,3.0]],columns=list(final_X_train.columns)))
```

[208]:
```
array([6367.70122])
```

[210]:
```
gc.collect()
```

[210]:
```
0
```

[2]:
```
import sklearn
print(sklearn.__version__)
```

```
1.2.2
```