Walmart Sales Forecasting using Machine Learning

April 18, 2023

0.1 Importing the necessary libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings, re, joblib
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split, RandomizedSearchCV,␣
 ↪RepeatedKFold, cross_val_score
from sklearn.linear_model import LinearRegression, PassiveAggressiveRegressor,␣
 ↪SGDRegressor, ARDRegression, RidgeCV, LassoCV, ElasticNetCV,␣
 ↪TweedieRegressor, HuberRegressor, RANSACRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR, LinearSVR, NuSVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, BaggingRegressor,␣
 ↪ExtraTreesRegressor, GradientBoostingRegressor, AdaBoostRegressor,␣
 ↪HistGradientBoostingRegressor, VotingRegressor
from sklearn.neural_network import MLPRegressor
from xgboost import XGBRegressor, XGBRFRegressor
from catboost import CatBoostRegressor
from lightgbm import LGBMRegressor
from scipy.stats import probplot
from sklearn.metrics import mean_absolute_percentage_error, mean_squared_error,␣
 ↪r2_score
from sklearn.preprocessing import StandardScaler, FunctionTransformer,␣
 ↪PowerTransformer, OrdinalEncoder
from yellowbrick.regressor import AlphaSelection, ResidualsPlot, PredictionError
from yellowbrick.features import Rank2D
import plotly.express as px
from sklearn.feature_selection import SelectKBest, SelectPercentile,␣
 ↪SelectFromModel, f_regression, mutual_info_regression, RFECV,␣
 ↪SequentialFeatureSelector
```

0.2 Loading the dataset

[2]:
```
df = pd.read_csv('walmart-sales-dataset-of-45stores.csv')
df.head()
```

[2]:
```
   Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0      1  05-02-2010    1643690.90             0        42.31       2.572
1      1  12-02-2010    1641957.44             1        38.51       2.548
2      1  19-02-2010    1611968.17             0        39.93       2.514
3      1  26-02-2010    1409727.59             0        46.63       2.561
4      1  05-03-2010    1554806.68             0        46.50       2.625

           CPI  Unemployment
0  211.096358         8.106
1  211.242170         8.106
2  211.289143         8.106
3  211.319643         8.106
4  211.350143         8.106
```

0.3 Understanding and assessing the shape and structure of the dataset

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

[3]:
```
(6435, 8)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   object
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

[5]:
```
df.describe()
```

[5]:
```
              Store  Weekly_Sales  Holiday_Flag  Temperature   Fuel_Price  \
count  6435.000000  6.435000e+03   6435.000000  6435.000000  6435.000000
mean     23.000000  1.046965e+06      0.069930    60.663782     3.358607
```

```
std      12.988182  5.643666e+05      0.255049    18.444933     0.459020
min       1.000000  2.099862e+05      0.000000    -2.060000     2.472000
25%      12.000000  5.533501e+05      0.000000    47.460000     2.933000
50%      23.000000  9.607460e+05      0.000000    62.670000     3.445000
75%      34.000000  1.420159e+06      0.000000    74.940000     3.735000
max      45.000000  3.818686e+06      1.000000   100.140000     4.468000

                CPI  Unemployment
count  6435.000000   6435.000000
mean    171.578394      7.999151
std      39.356712      1.875885
min     126.064000      3.879000
25%     131.735000      6.891000
50%     182.616521      7.874000
75%     212.743293      8.622000
max     227.232807     14.313000
```

[6]:
```
df.isna().sum()
```

[6]:
```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

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

[7]:
```
0
```

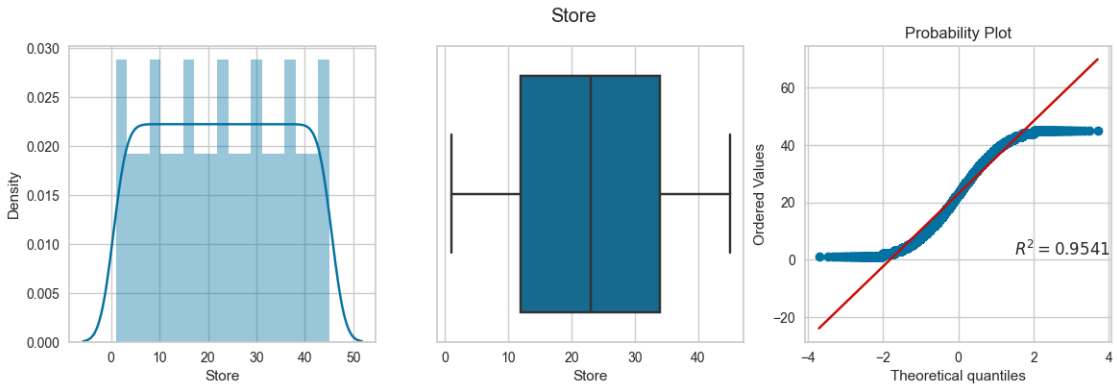
0.4 Exploratory Data Analysis (EDA)

0.4.1 Univariate Analysis

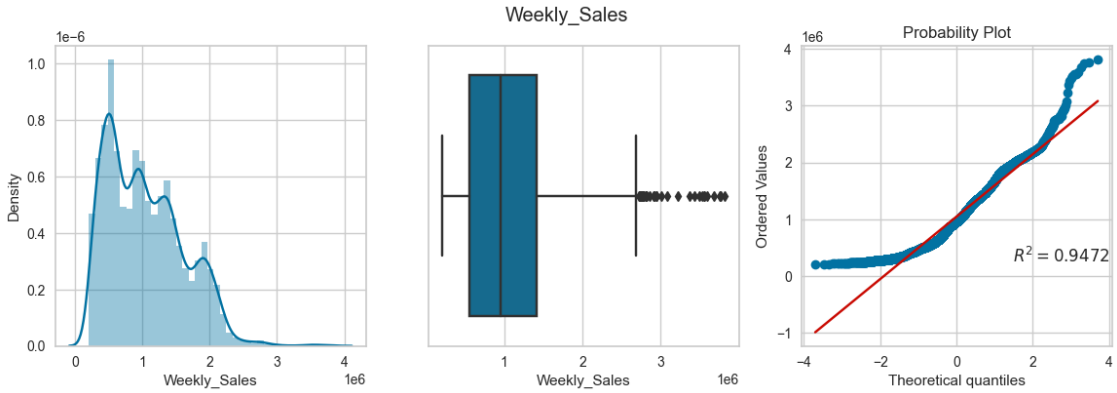
[8]:
```
for col in df.select_dtypes(np.number).columns:
    plt.figure(figsize=(14,4))
    print(f"Skewness of {col}:",df[col].skew())
    print(f"Kurtosis of {col}:",df[col].kurtosis())
    plt.subplot(131)
    sns.distplot(df[col])
    plt.subplot(132)
    sns.boxplot(df[col])
    plt.subplot(133)
    probplot(df[col],rvalue=True,plot=plt,dist='norm')
    plt.suptitle(col)
```

```
    plt.show();
```

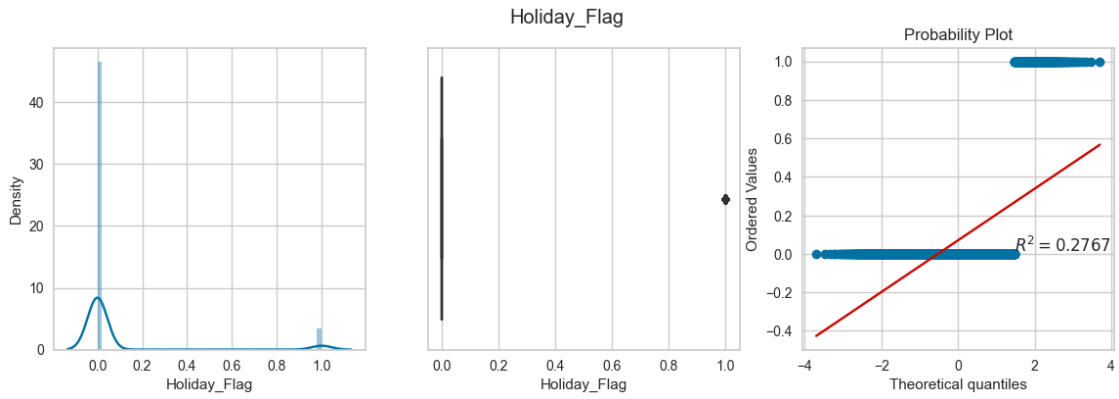
```
Skewness of Store: 0.0
Kurtosis of Store: -1.2011866346083275
```



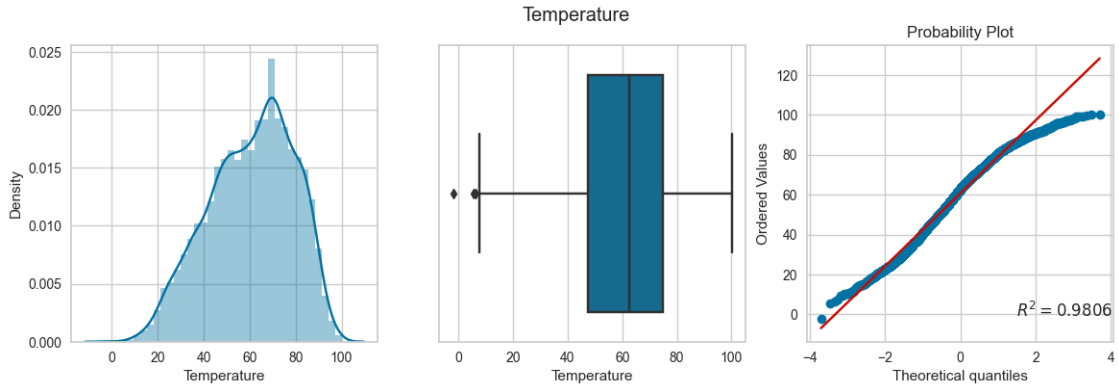

```
Skewness of Weekly_Sales: 0.6683617974864524
Kurtosis of Weekly_Sales: 0.05314092741722032
```



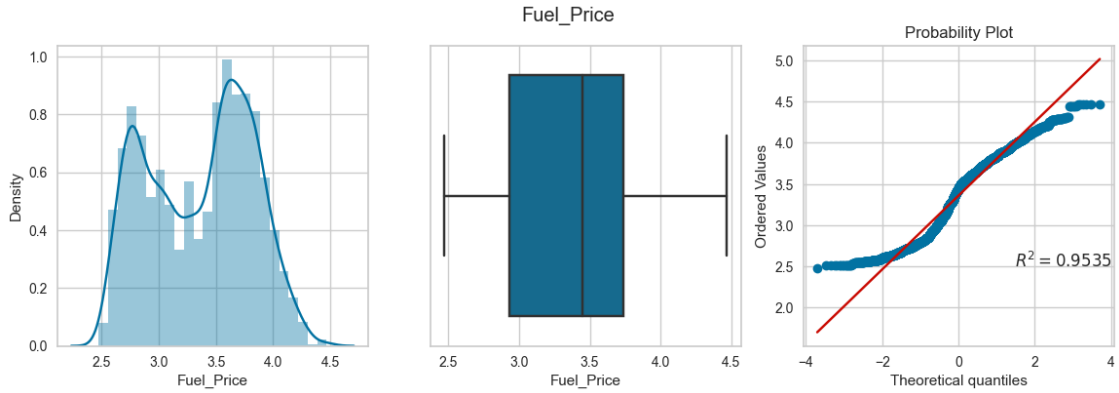

```
Skewness of Holiday_Flag: 3.3734986714578485
Kurtosis of Holiday_Flag: 9.383409556463409
```



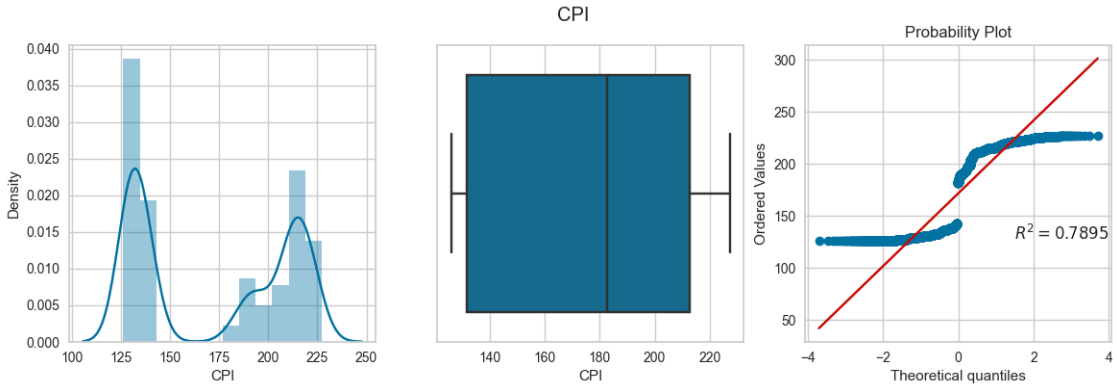

```
Skewness of Temperature: -0.3367676011075799
Kurtosis of Temperature: -0.6128009588453383
```



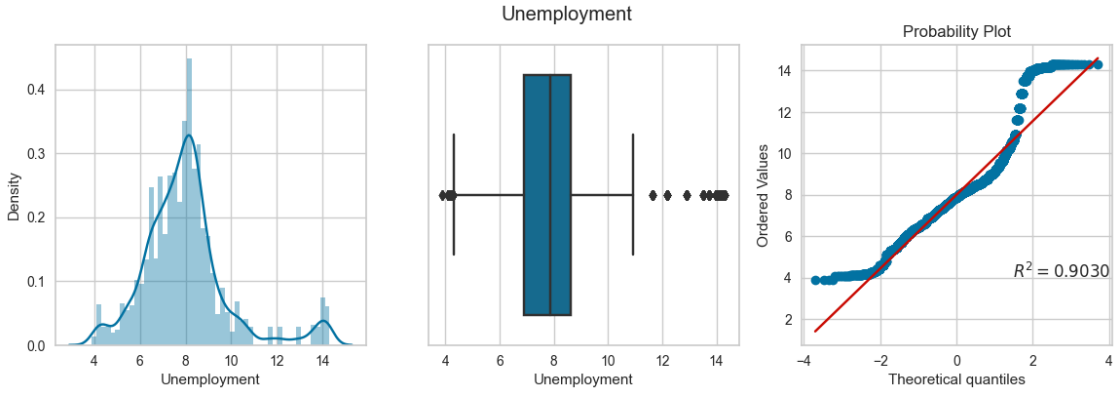

```
Skewness of Fuel_Price: -0.09615830011865549
Kurtosis of Fuel_Price: -1.1773777964906604
```



```
Skewness of CPI: 0.06349184988549494
Kurtosis of CPI: -1.8398133421838165
```

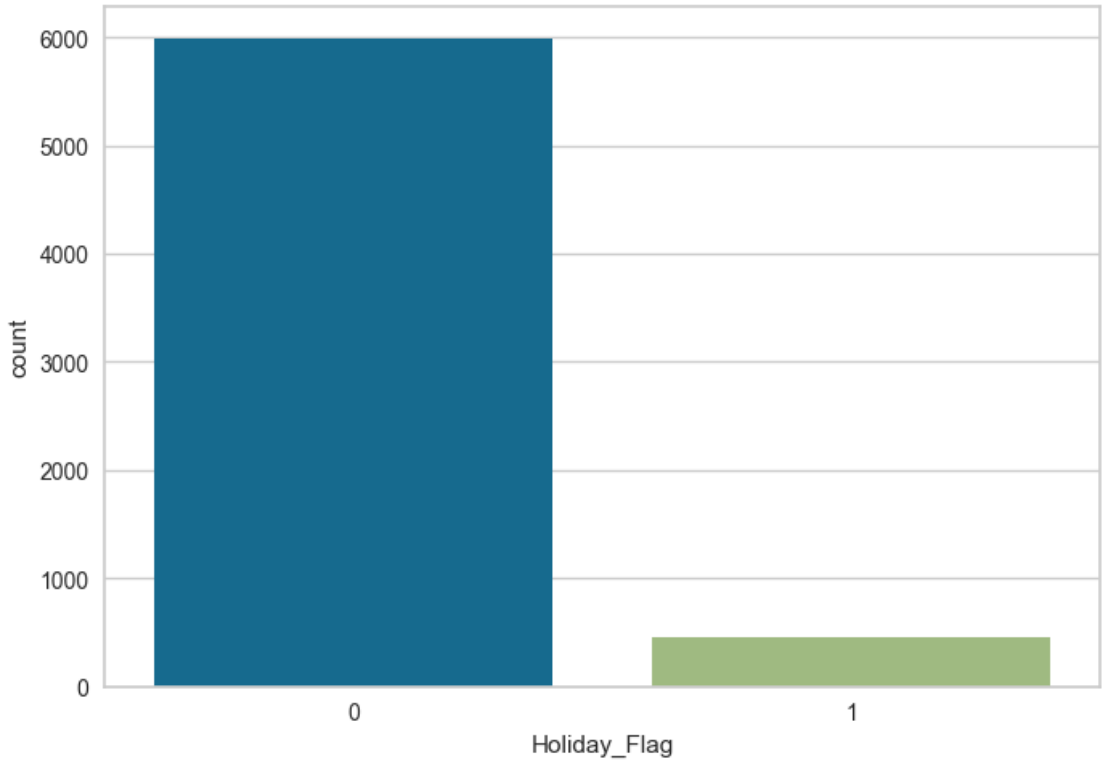



```
Skewness of Unemployment: 1.1881439334843265
Kurtosis of Unemployment: 2.639711784234234
```



Normal Distribution: Store Left-Skewed Distribution: Temperature Right-Skewed Distribution: Weekly_Sales, Unemployment, Fuel_Price Bimodal Distribution: CPI Outlier Columns: Weekly_Sales, Temperature, Unemployment

[9]:
```
sns.countplot(df.Holiday_Flag);
```



There are very few holidays across the given time period.

0.4.2 Bivariate Analysis

[10]:
```
df.corr()['Weekly_Sales'].sort_values(ascending=False)[1:]
```

[10]:
```
Holiday_Flag     0.036891
Fuel_Price       0.009464
Temperature     -0.063810
CPI             -0.072634
Unemployment    -0.106176
Store           -0.335332
Name: Weekly_Sales, dtype: float64
```

[11]:
```
px.scatter(df,x='CPI',y='Weekly_Sales',color='Holiday_Flag',title='2D Scatter␣
↪Plot of CPI and Weekly Sales')
```

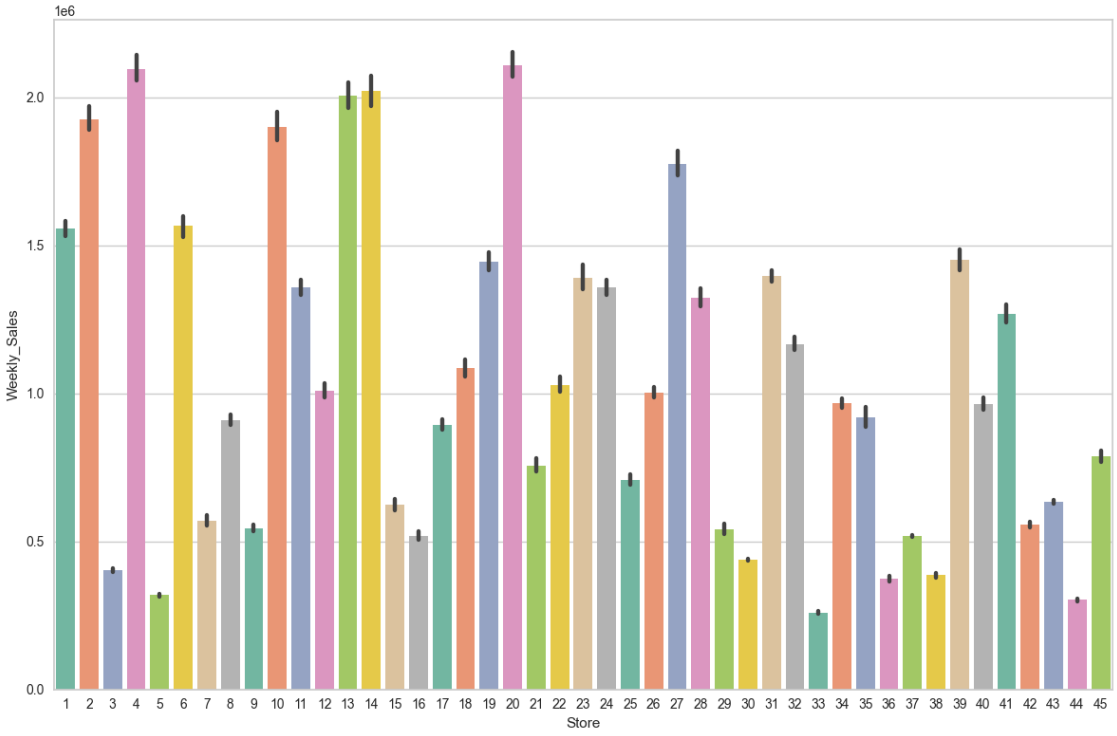
[12]:
```
px.scatter(df,x='Unemployment',y='Weekly_Sales',color='Holiday_Flag',title='2D␣
↪Scatter Plot of Unemployment and Weekly Sales')
```

There is a very weak negative correlation between unemployment and weekly sales.

[13]:
```
px.scatter(df,x='Temperature',y='Weekly_Sales',color='Holiday_Flag',title='2D␣
↪Scatter Plot of Temperature and Weekly Sales')
```

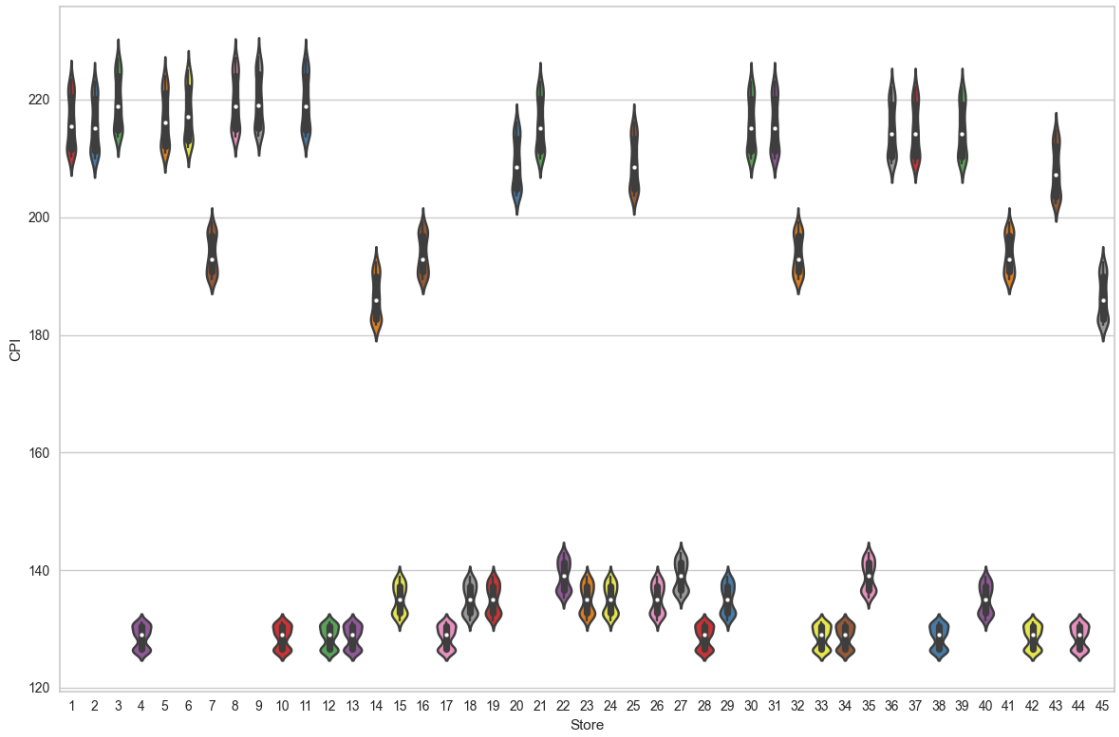
There are burgeoning weekly sales during the sunny days and clear weather when temperature ranges between 25 deg C to 45 deg C.

[14]:
```
plt.figure(figsize=(12,8))
sns.barplot(x='Store',y='Weekly_Sales',data=df,palette='Set2')
plt.tight_layout();
```



Stores 4 and 20 have had the most flourishing sales whereas store 33 has had really poor sales.

[15]:
```
plt.figure(figsize=(12,8))
sns.violinplot(x='Store',y='CPI',data=df,palette='Set1')
plt.tight_layout();
```



Stores 3,8,9 and 11 have the highest Consumer Price Index (CPI) i.e. they have raised the selling price by a significant amount in the past few years. In simple words, these stores have undergone the most inflation from 2010 to 2012. On the other hand, there are many stores which have the lowest CPI such as stores 4,10,12,13,17,28,33,34,38,42 and 44.

[16]:
```
holidays = df[df.Holiday_Flag == 1]
workdays = df[df.Holiday_Flag == 0]
```

[17]:
```
workdays.describe()
```

[17]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	\
count	5985.000000	5.985000e+03	5985.0	5985.000000	5985.000000	
mean	23.000000	1.041256e+06	0.0	61.448124	3.368467	
std	12.988258	5.589574e+05	0.0	18.076892	0.461158	
min	1.000000	2.099862e+05	0.0	-2.060000	2.472000	
25%	12.000000	5.513784e+05	0.0	48.730000	2.932000	
50%	23.000000	9.562112e+05	0.0	63.560000	3.475000	
75%	34.000000	1.414344e+06	0.0	75.320000	3.744000	
max	45.000000	3.818686e+06	0.0	100.140000	4.468000	

	CPI	Unemployment
count	5985.000000	5985.000000
mean	171.601725	7.993514
std	39.364794	1.875664

```
min     126.064000      3.879000
25%     131.784000      6.891000
50%     182.622509      7.874000
75%     212.861131      8.595000
max     227.232807     14.313000
```

[18]:
```
holidays.describe()
```

[18]:
```
              Store  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
count   450.000000  4.500000e+02         450.0   450.000000  450.000000
mean     23.000000  1.122888e+06           1.0    50.232044    3.227464
std      13.001627  6.276849e+05           0.0    20.071118    0.407934
min       1.000000  2.153592e+05           1.0    10.240000    2.513000
25%      12.000000  5.758655e+05           1.0    33.367500    2.943000
50%      23.000000  1.018538e+06           1.0    47.870000    3.179000
75%      34.000000  1.555213e+06           1.0    67.335000    3.546000
max      45.000000  3.004702e+06           1.0    96.220000    4.124000

                CPI  Unemployment
count   450.000000    450.000000
mean    171.268092      8.074127
std      39.291449      1.879293
min     126.114581      4.077000
25%     131.586613      7.015000
50%     182.598178      7.890000
75%     211.760416      8.625000
max     226.210354     14.313000
```

Holidays have slightly more weekly sales as compared to normal days. This may be due to the festive seasons when a large number of consumers visit the retail stores for purchasing their desired goods, products and services at reasonable discounted rates. Temperatures are relatively lower in the holiday season in comparison to the normal working season.

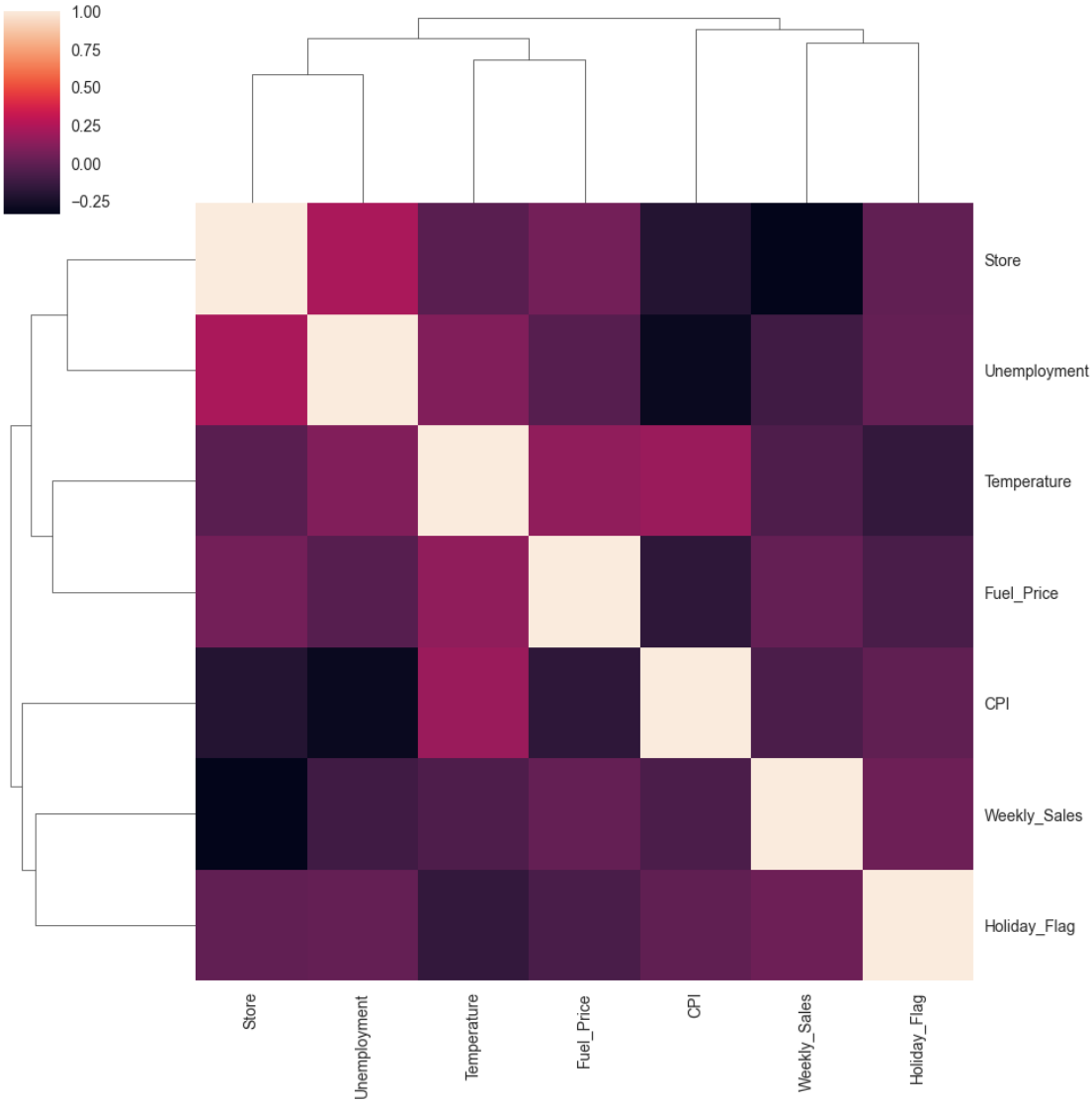
0.4.3 Multivariate Analysis

[19]:
```
fig = px.
 ↪scatter_3d(df,x='CPI',y='Weekly_Sales',z='Fuel_Price',color='Holiday_Flag',title='3D␣
 ↪Scatter Plot of CPI and Weekly Sales')
fig.show()
```

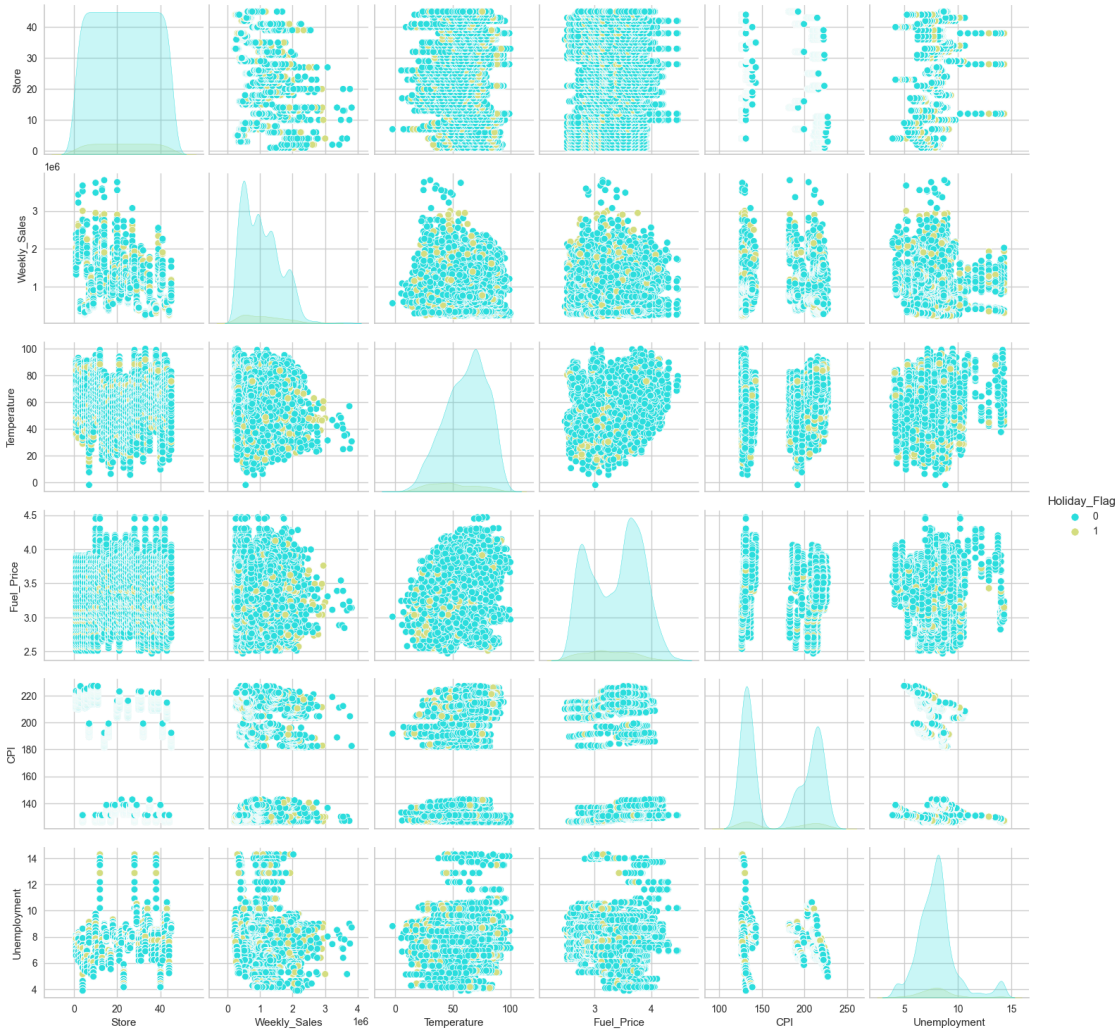
Fuel price is generally lower during the holidays.

[20]:
```
sns.heatmap(df.corr(),annot=True,cmap='viridis',vmin=-1.0,vmax=1.0);
```

[21]:
```
sns.clustermap(df.corr());
```



[22]:
```
sns.pairplot(df,hue='Holiday_Flag',palette='rainbow');
```



0.5 Feature Engineering

0.5.1 Outlier Treatment

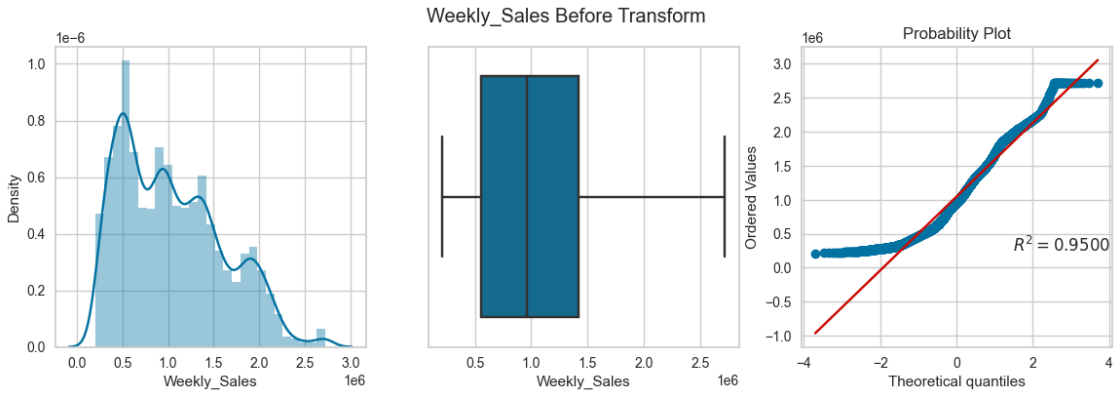
[23]:
```
def remove_outliers(col):
    lower_limit, upper_limit = df[col].quantile([0.25,0.75])
    IQR = upper_limit - lower_limit
    lower_whisker = lower_limit - 1.5 * IQR
    upper_whisker = upper_limit + 1.5 * IQR
    return np.where(df[col]>upper_whisker,upper_whisker,np.
↪where(df[col]<lower_whisker,lower_whisker,df[col]))
```

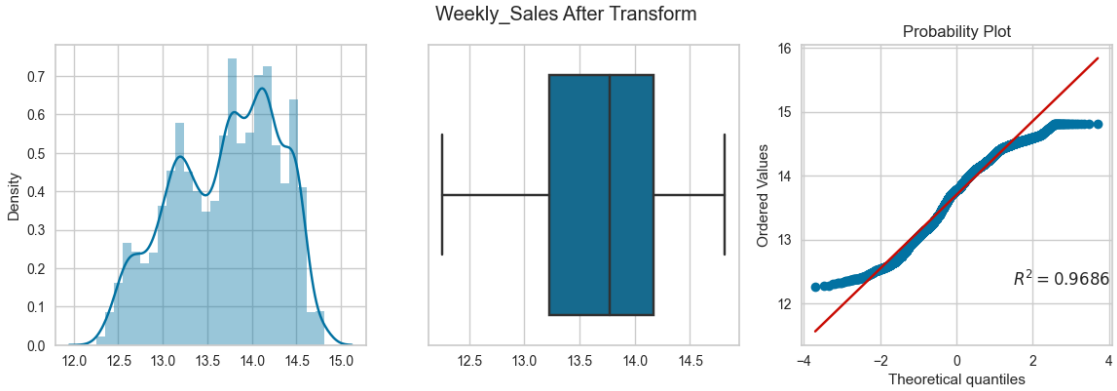
[24]:
```
outlier_cols = ['Weekly_Sales','Temperature','Unemployment']

for col in outlier_cols:
    df[col] = remove_outliers(col)
```

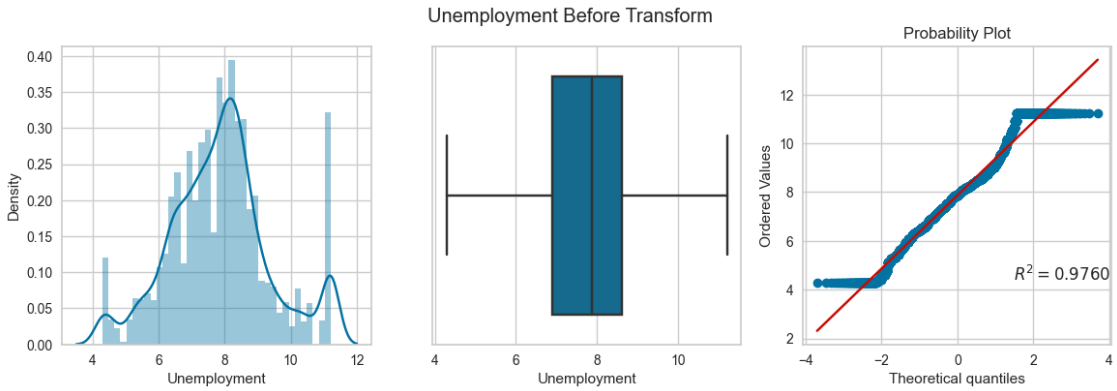
0.5.2 Feature Transformation

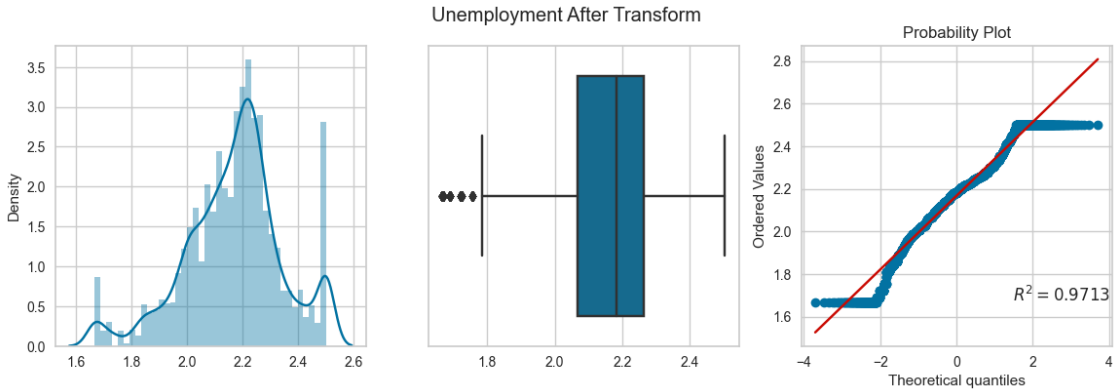
[25]:
```
def apply_transform(transform,col):
    plt.figure(figsize=(14,4))
    plt.subplot(131)
    sns.distplot(df[col])
    plt.subplot(132)
    sns.boxplot(df[col])
    plt.subplot(133)
    probplot(df[col],rvalue=True,dist='norm',plot=plt)
    plt.suptitle(f'{col} Before Transform')
    plt.show()
    col_tf = transform.fit_transform(df[[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();
```

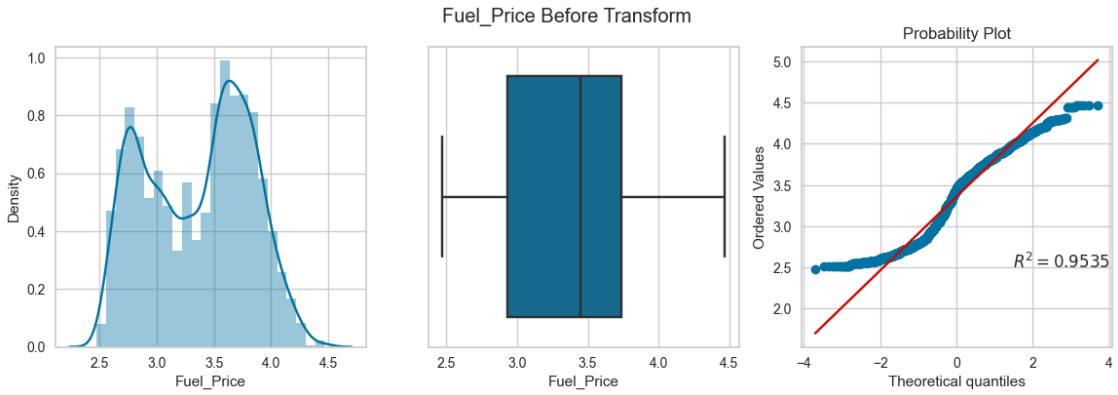
[26]:
```
skewed_cols = ['Weekly_Sales','Unemployment','Fuel_Price','CPI','Temperature']

for col in skewed_cols:
    apply_transform(FunctionTransformer(np.log1p),col)
```

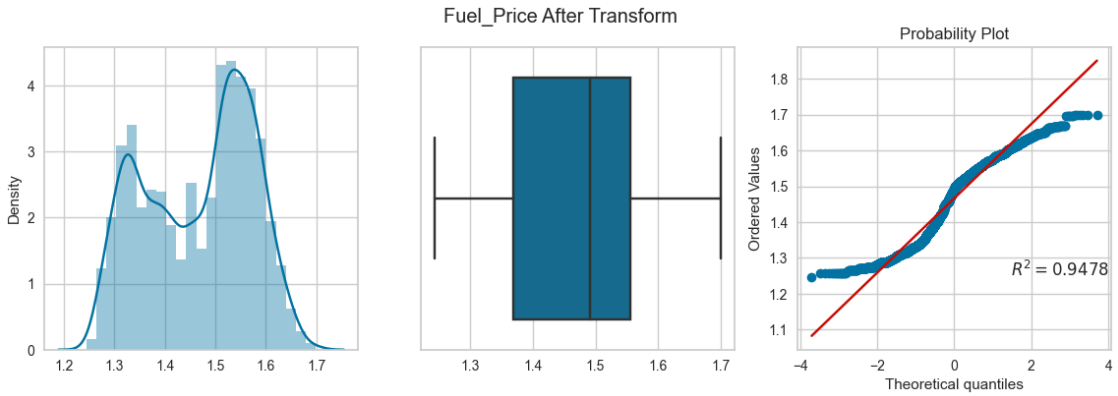


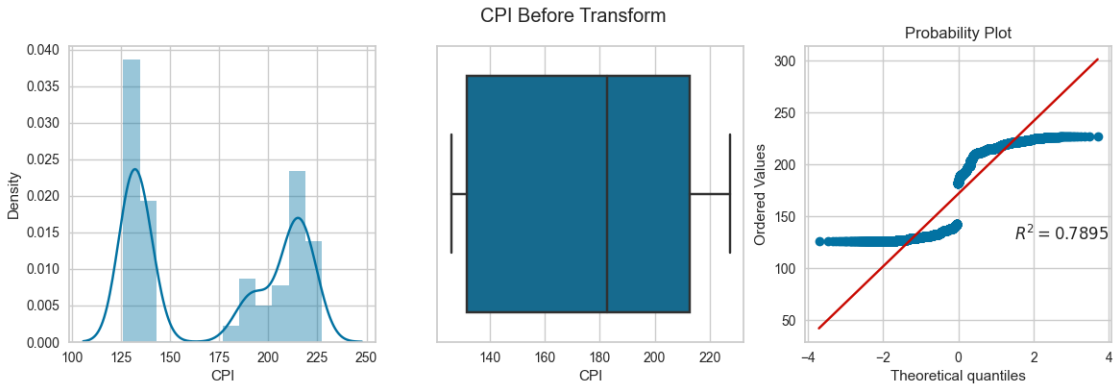

Weekly_Sales After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
R^2 = 0.9686

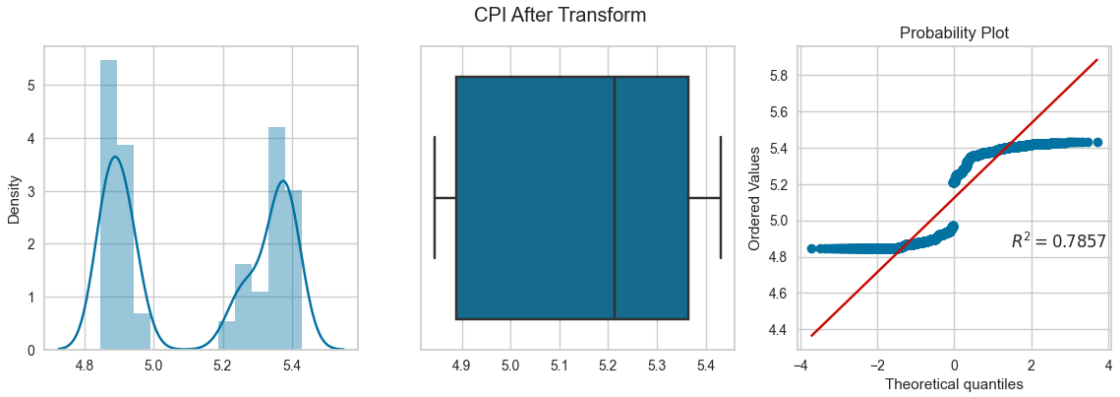

Unemployment Before Transform
Density
Unemployment
Probability Plot
Ordered Values
Theoretical quantiles
R^2 = 0.9760

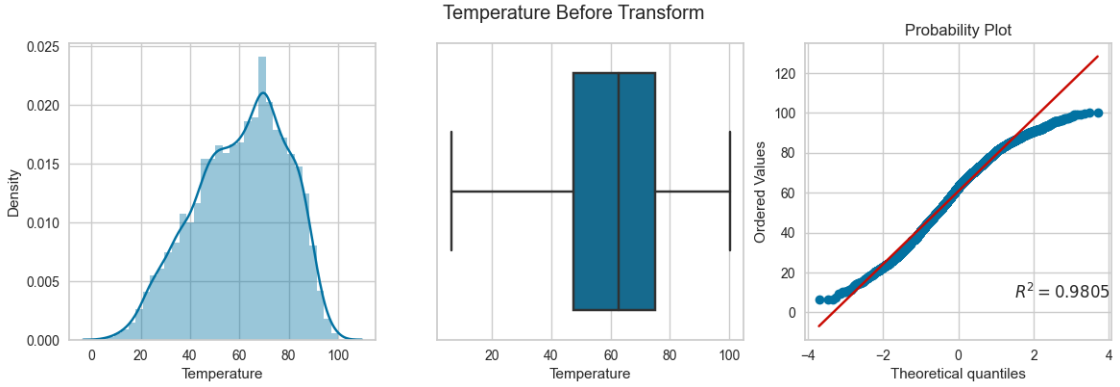

Unemployment After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
R^2 = 0.9713

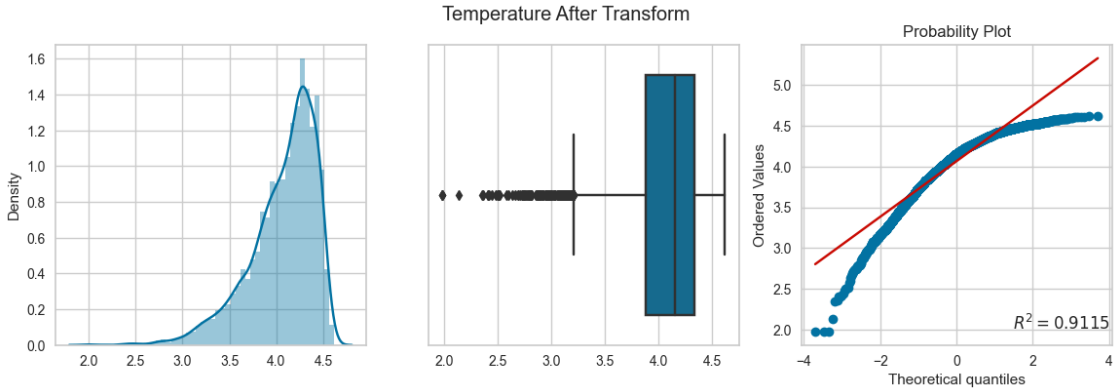

Fuel_Price Before Transform
Density
Fuel_Price
Fuel_Price
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9535$


Fuel_Price After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9478$

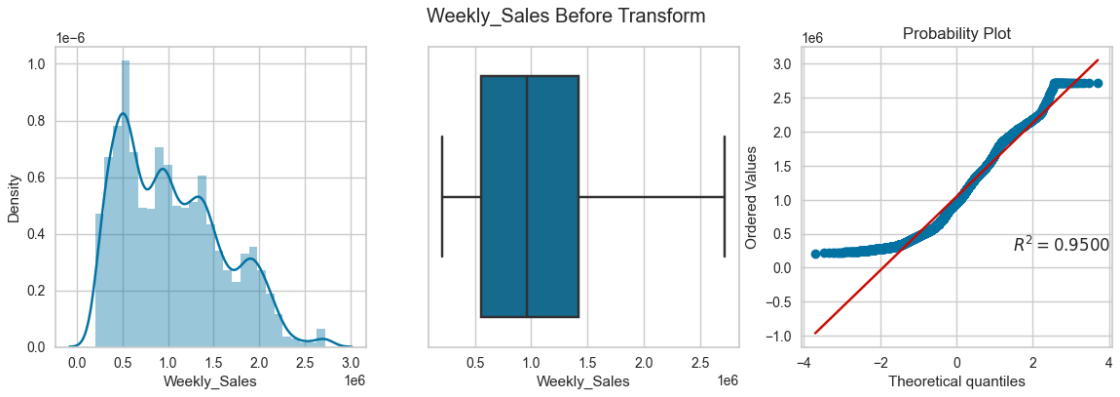

CPI Before Transform
Density
CPI
CPI
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.7895$



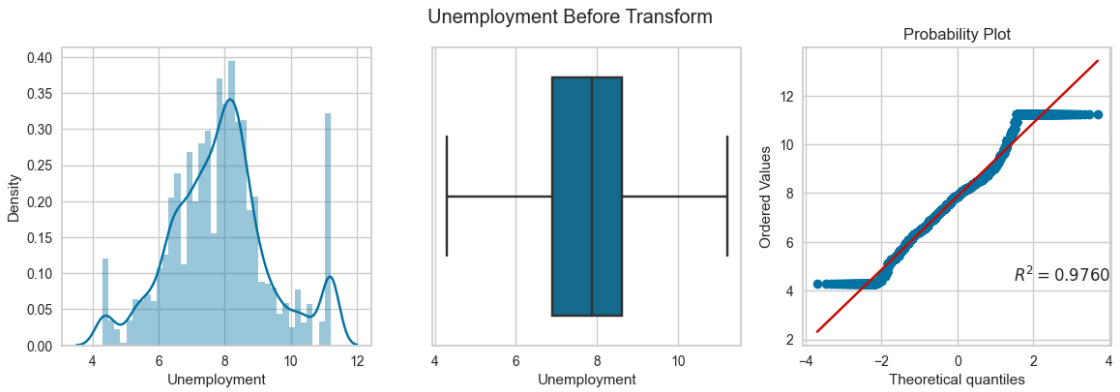


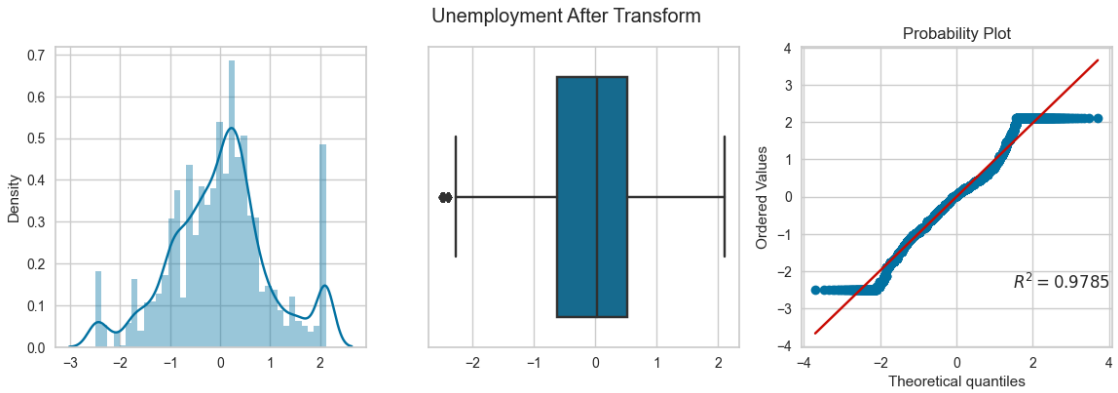


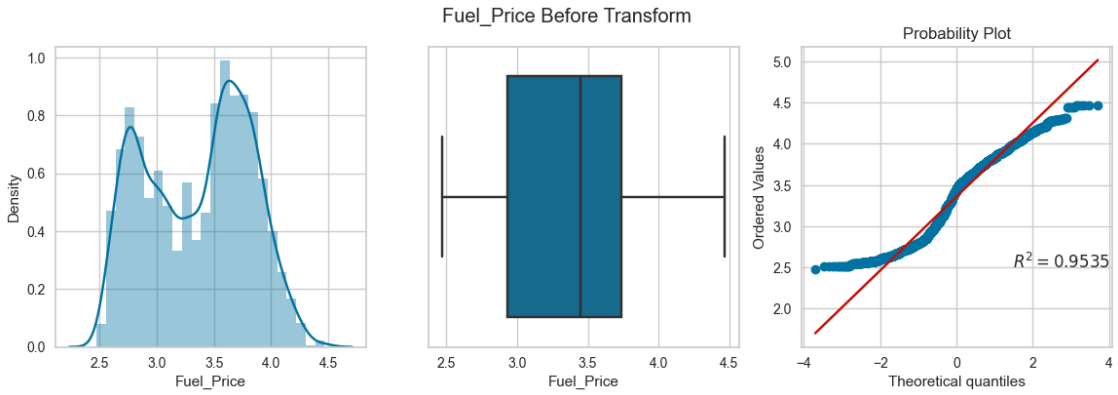
[27]:
```
for col in skewed_cols:
    apply_transform(PowerTransformer(),col)
```

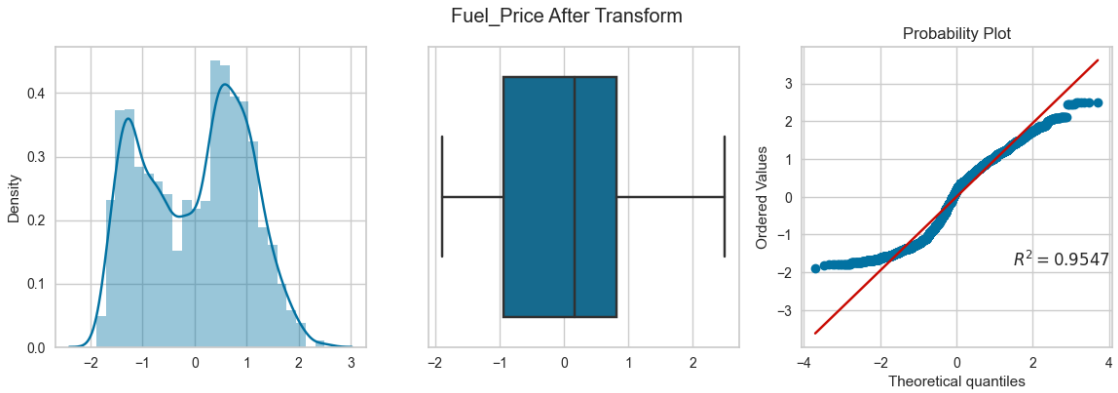

Weekly_Sales Before Transform
1e−6
Density
Weekly_Sales
1e6
Ordered Values
Theoretical quantiles
Probability Plot
$R^2 = 0.9500$

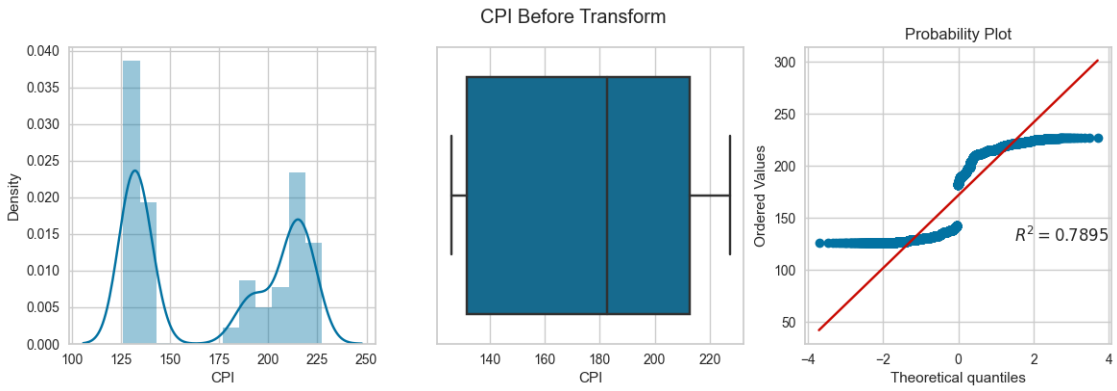

Weekly_Sales After Transform
Density
Ordered Values
Theoretical quantiles
Probability Plot
$R^2 = 0.9762$

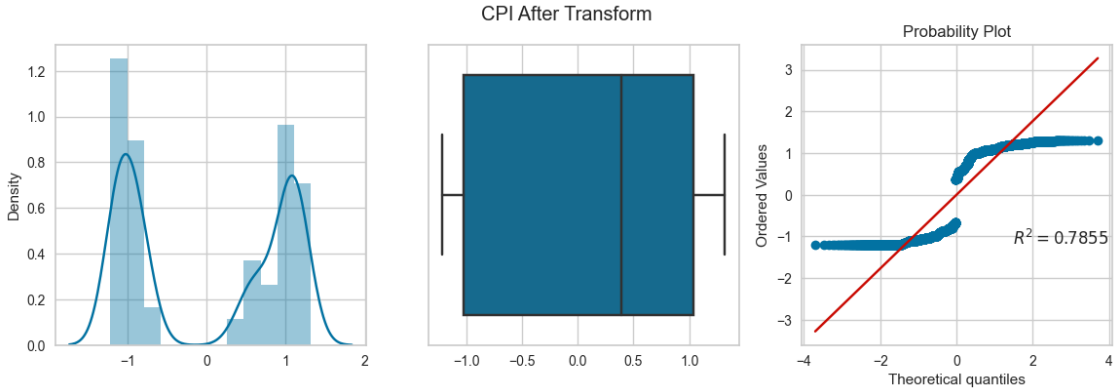

Unemployment Before Transform
Density
Unemployment
Ordered Values
Theoretical quantiles
Probability Plot
$R^2 = 0.9760$

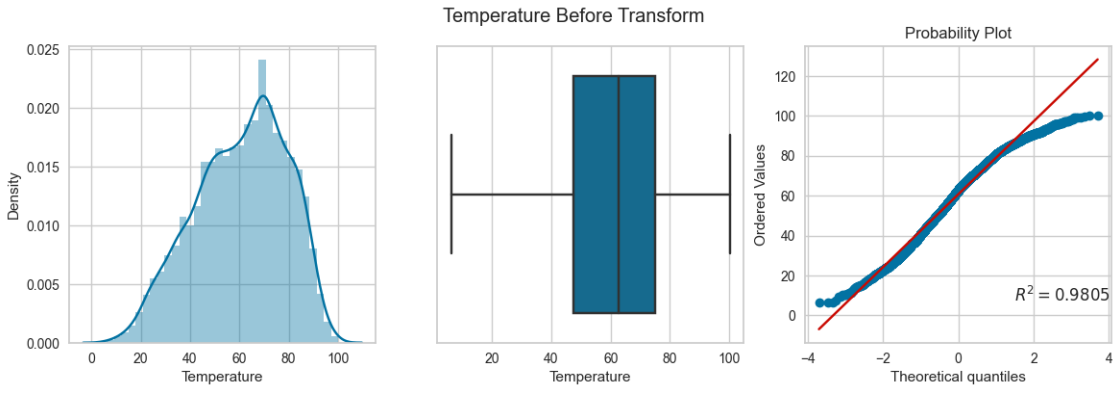

Unemployment After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9785$

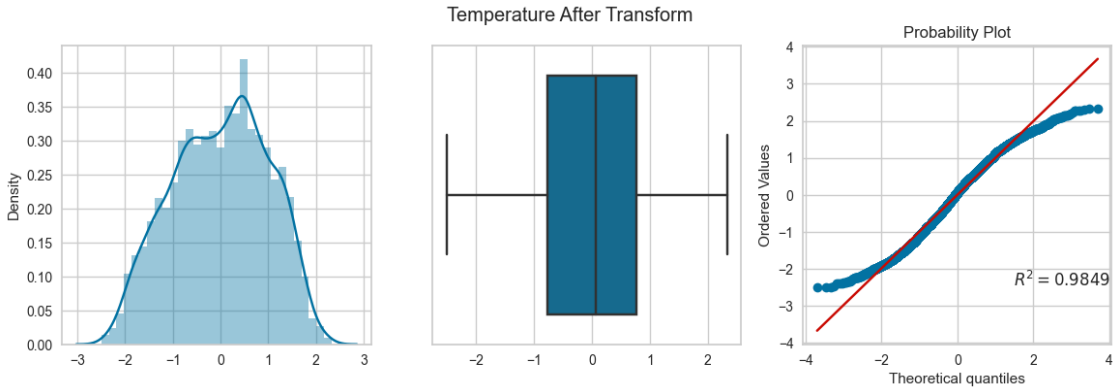

Fuel_Price Before Transform
Density
Fuel_Price
Fuel_Price
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9535$


Fuel_Price After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9547$

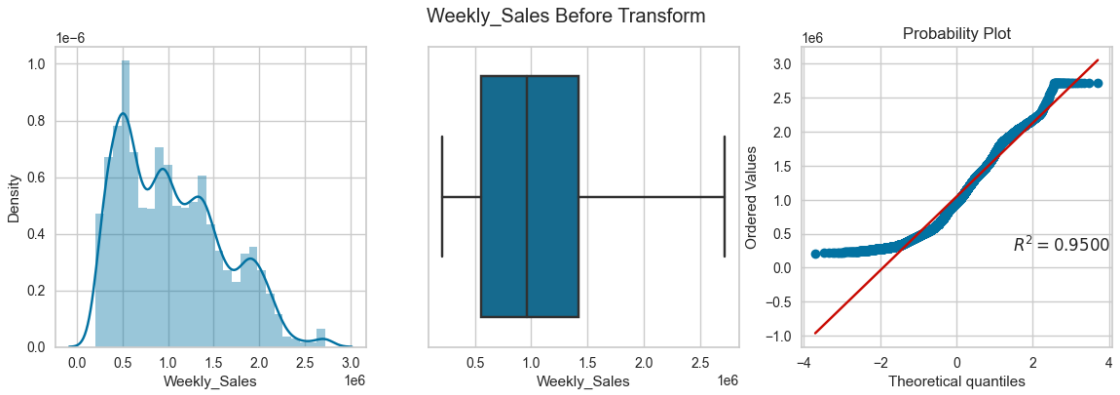

CPI Before Transform
Density
CPI
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.7895$

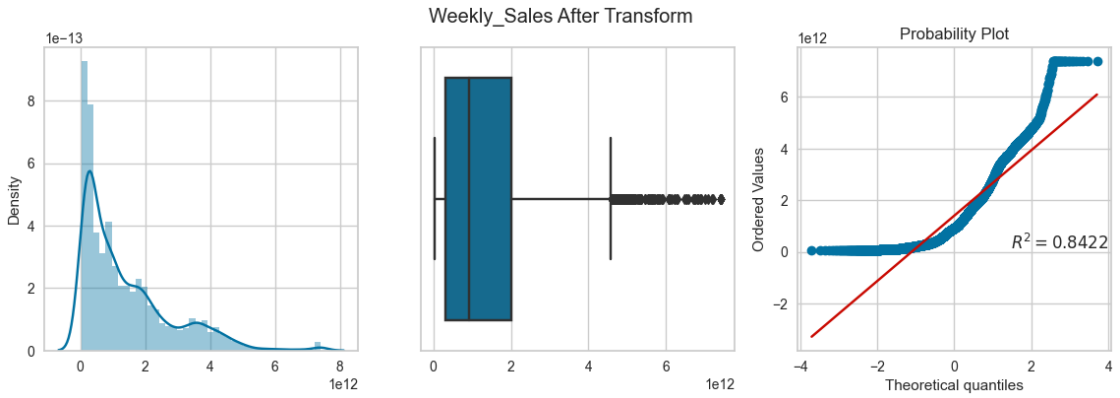

CPI After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.7855$

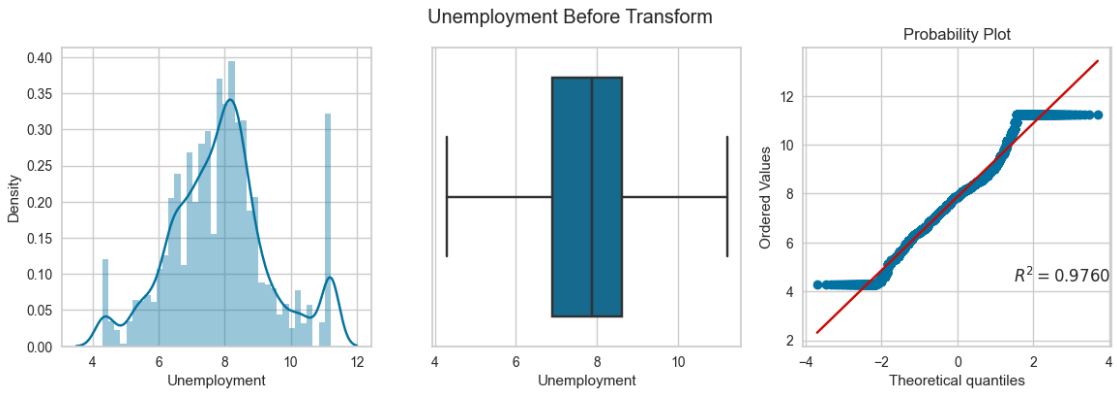

Temperature Before Transform
Density
Temperature
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9805$

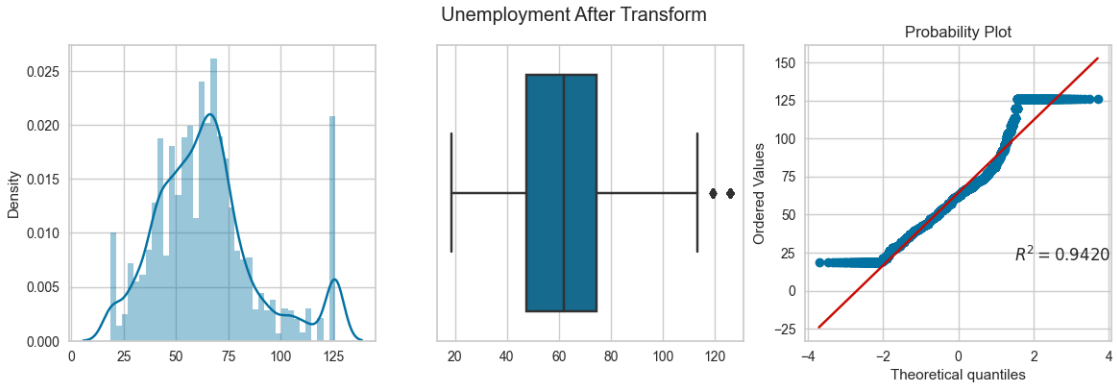


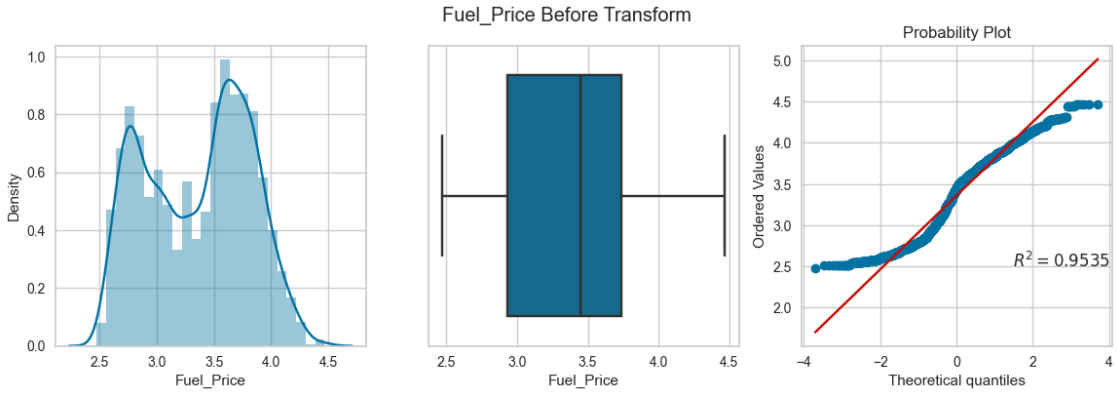
[28]:
```
for col in skewed_cols:
    apply_transform(FunctionTransformer(lambda x: x**2),col)
```

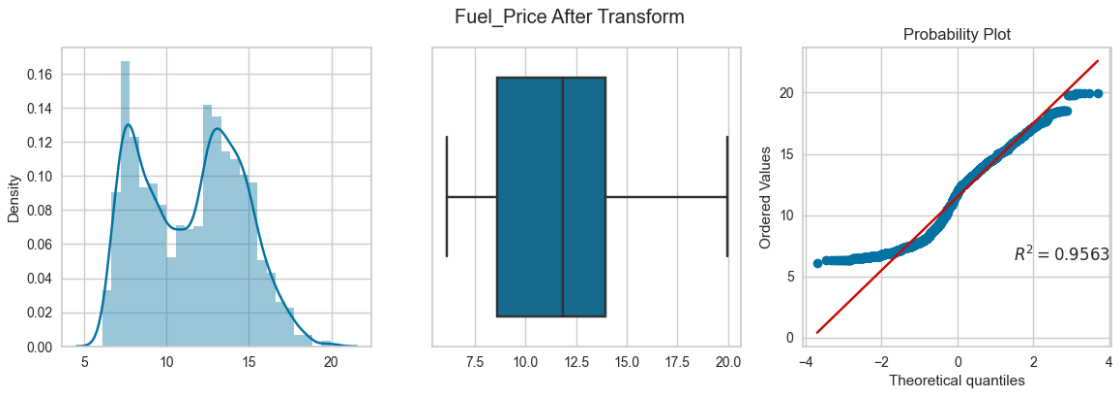


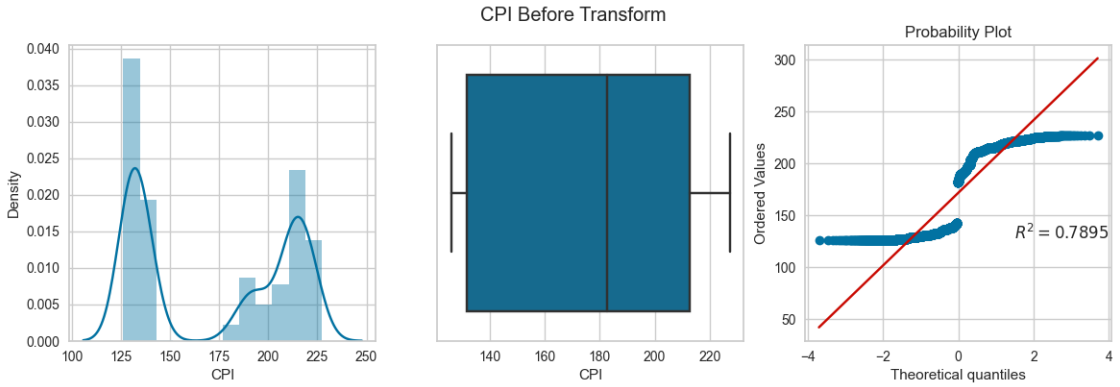


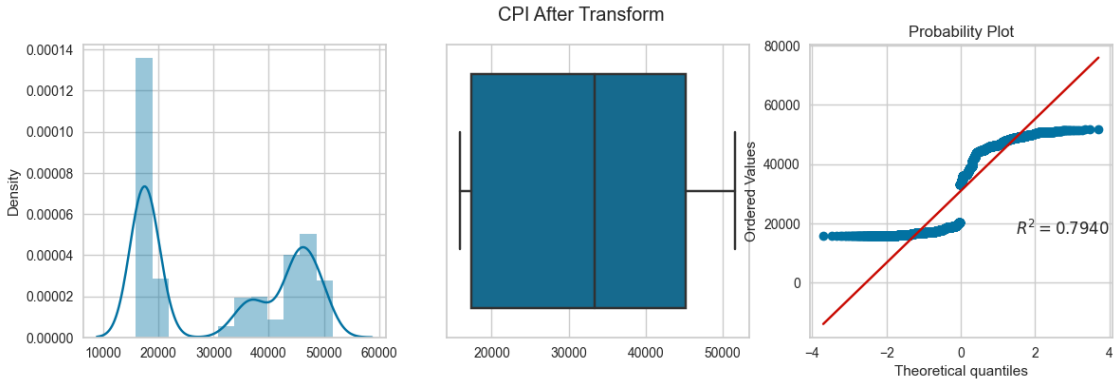

Unemployment Before Transform
Density
Unemployment
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9760$

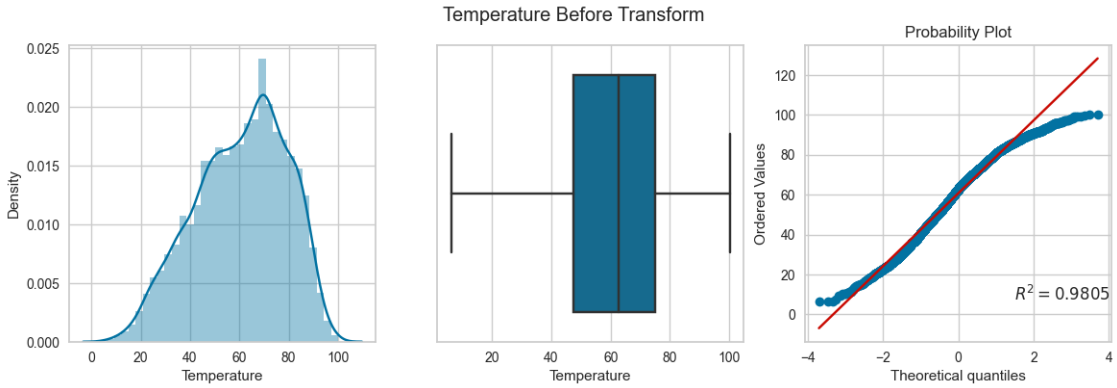

Unemployment After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9420$

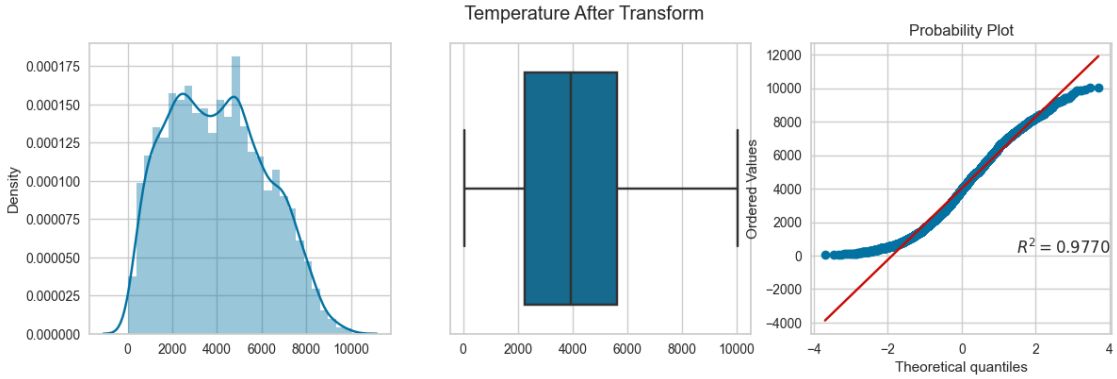

Fuel_Price Before Transform
Density
Fuel_Price
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9535$


Fuel_Price After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
R^2 = 0.9563

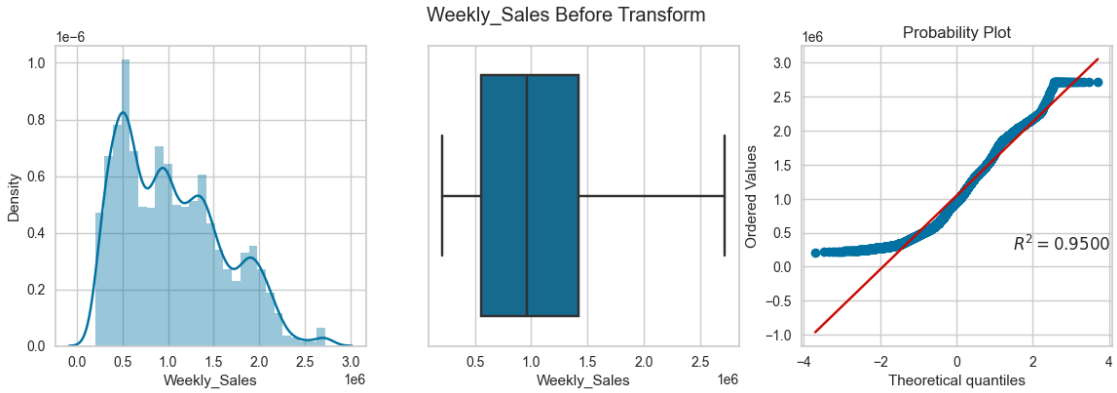

CPI Before Transform
Density
CPI
Probability Plot
Ordered Values
Theoretical quantiles
R^2 = 0.7895

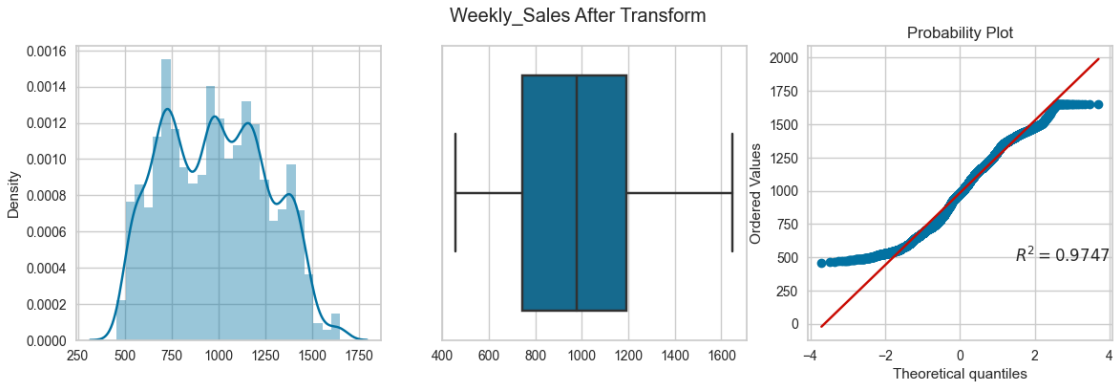

CPI After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
R^2 = 0.7940

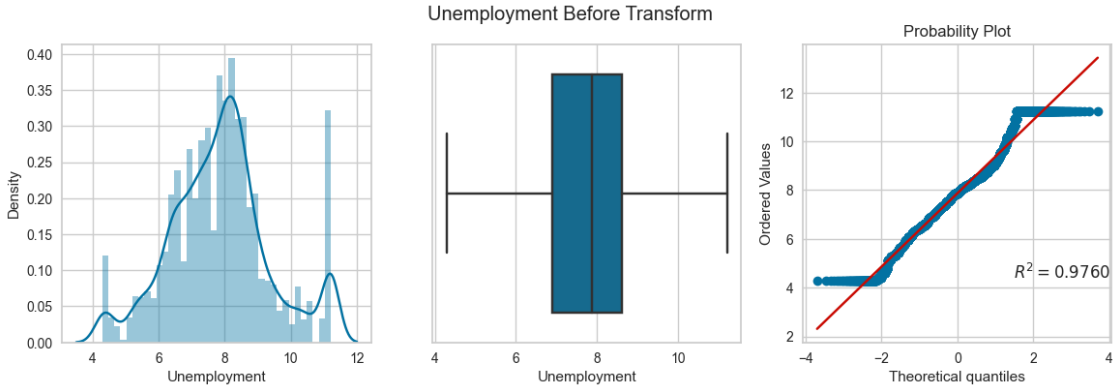


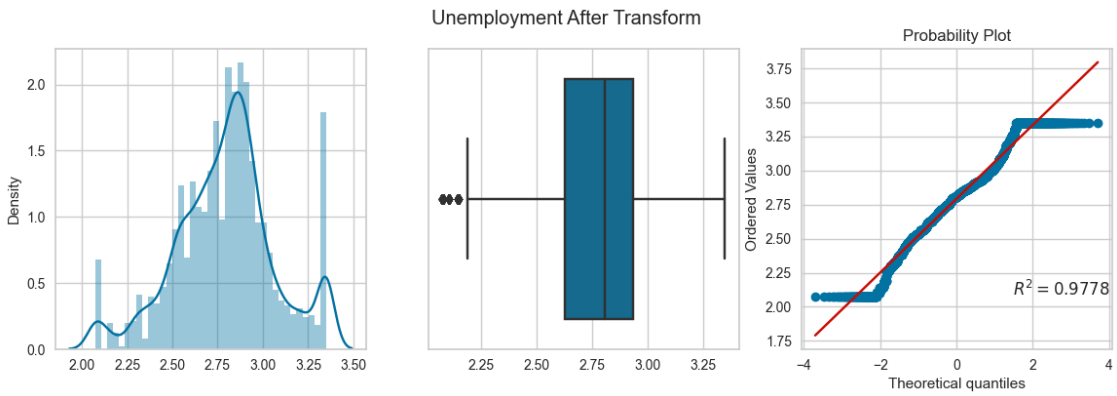


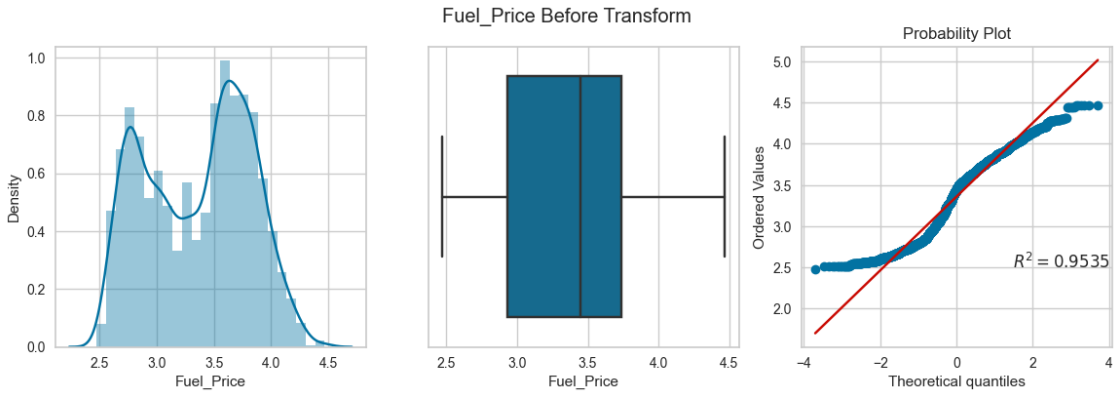
[29]:
```
for col in skewed_cols:
    apply_transform(FunctionTransformer(np.sqrt),col)
```

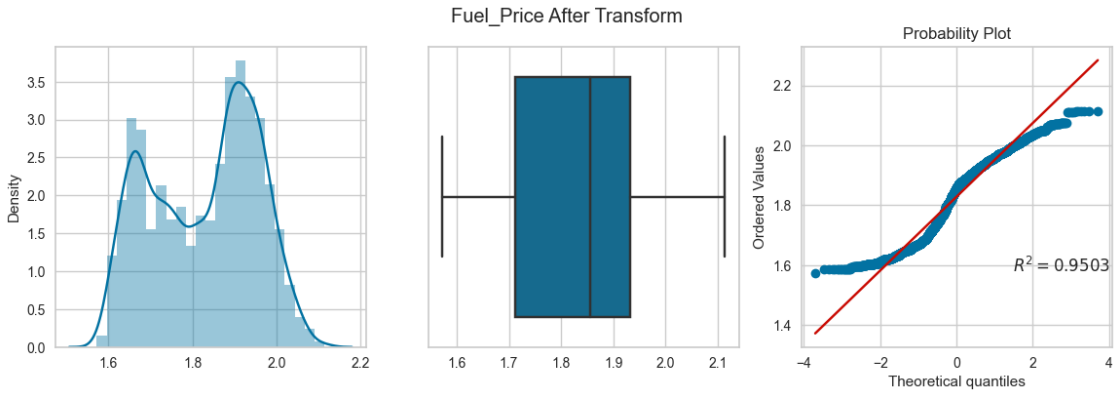


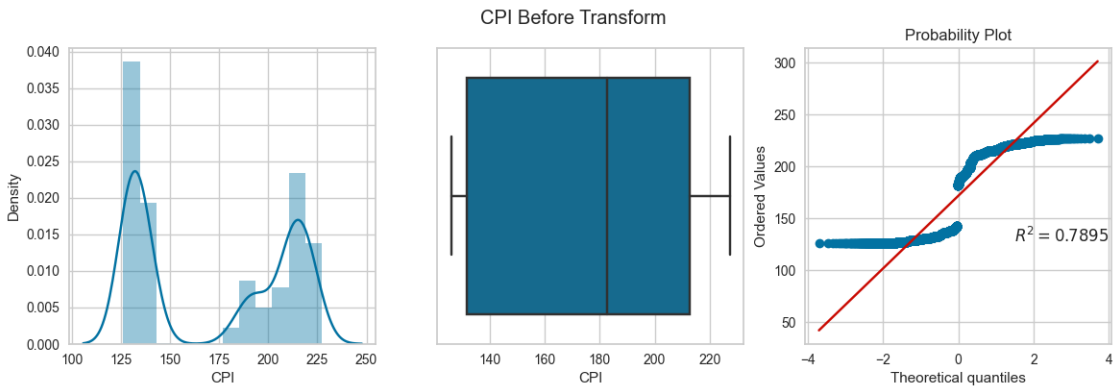

Weekly_Sales After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9747$

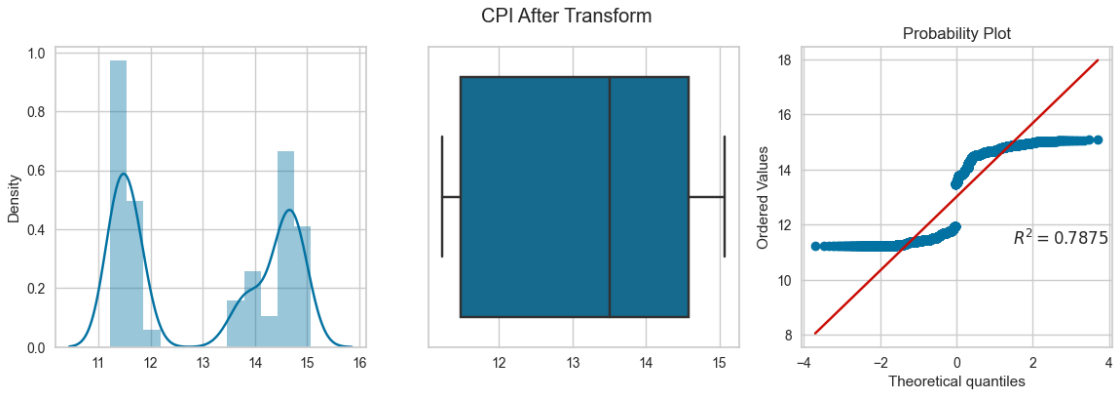

Unemployment Before Transform
Density
Unemployment
Unemployment
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9760$

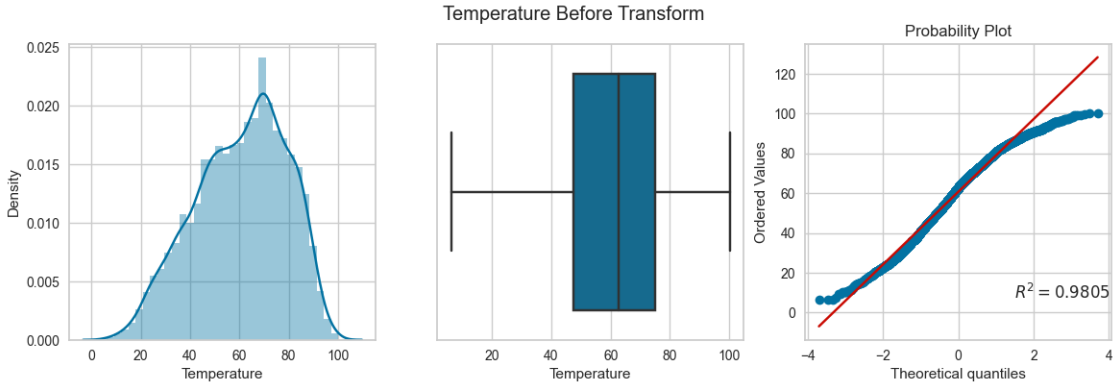

Unemployment After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9778$

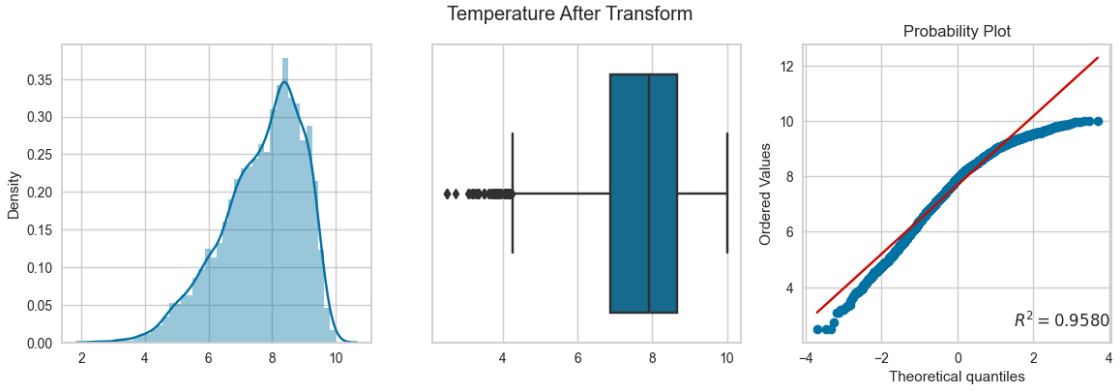

Fuel_Price Before Transform
Density
Fuel_Price
Fuel_Price
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9535$


Fuel_Price After Transform
Density
Fuel_Price
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9503$

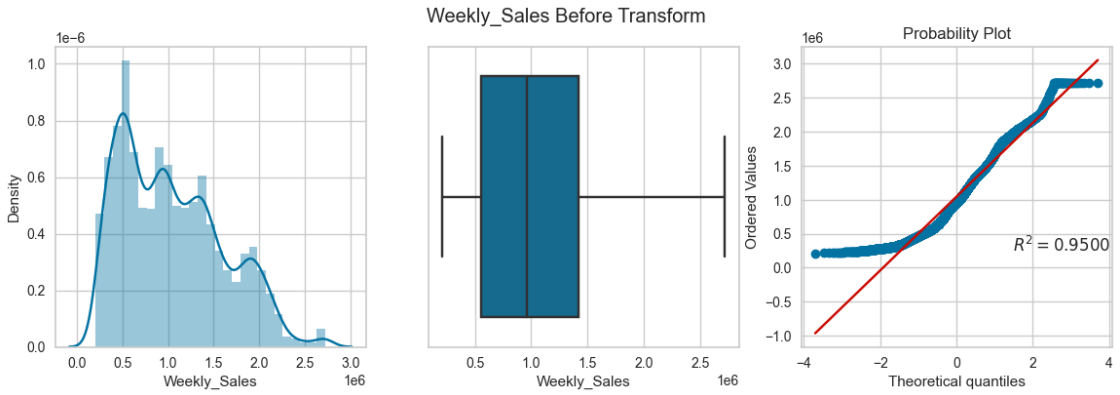

CPI Before Transform
Density
CPI
CPI
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.7895$

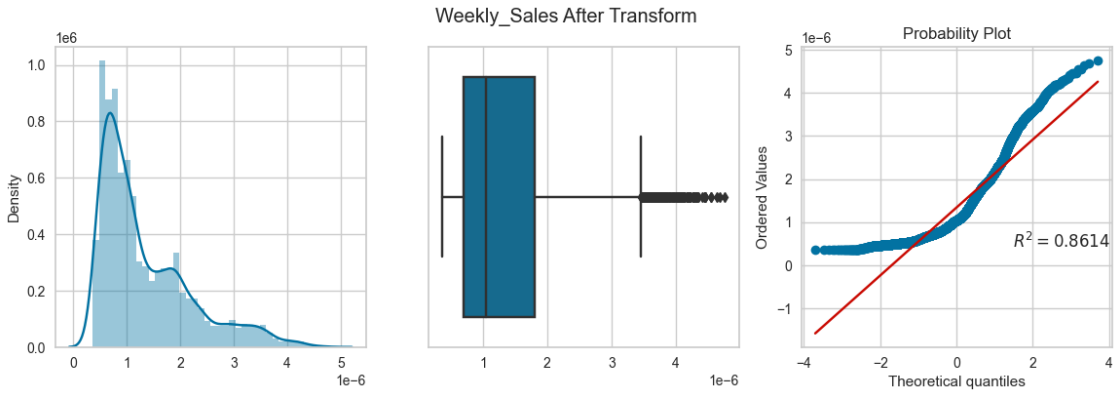


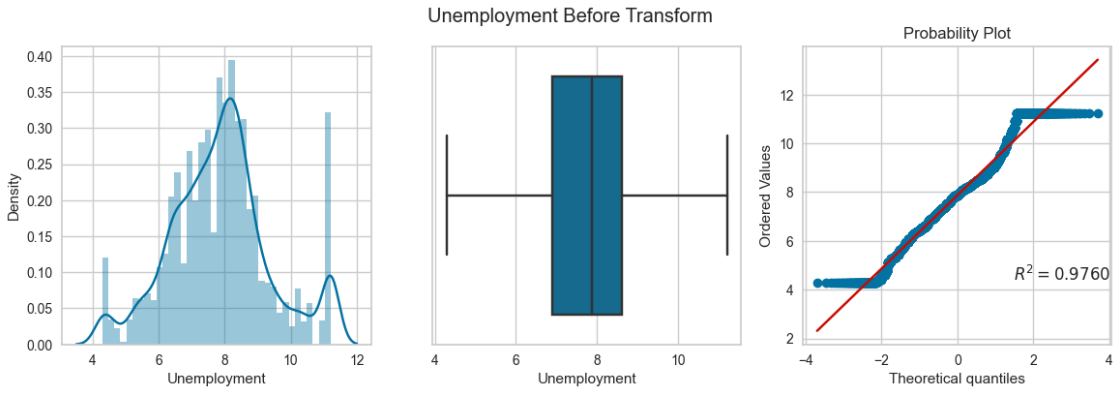


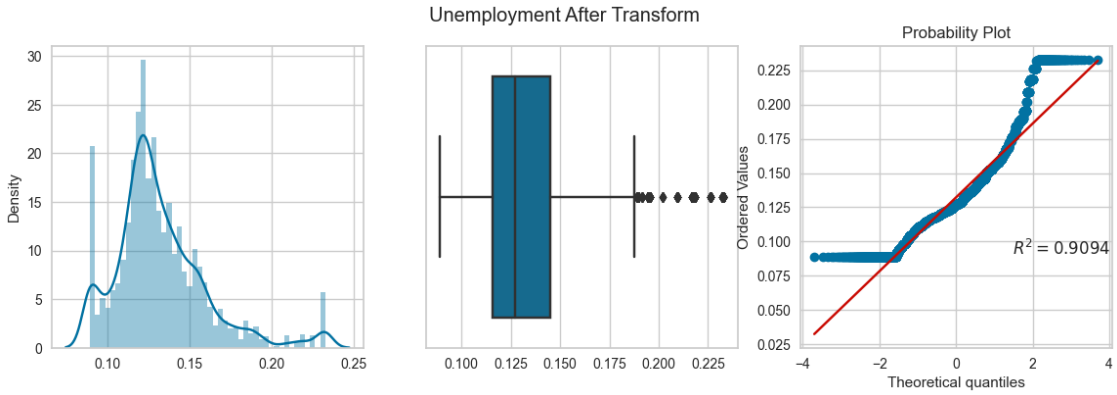


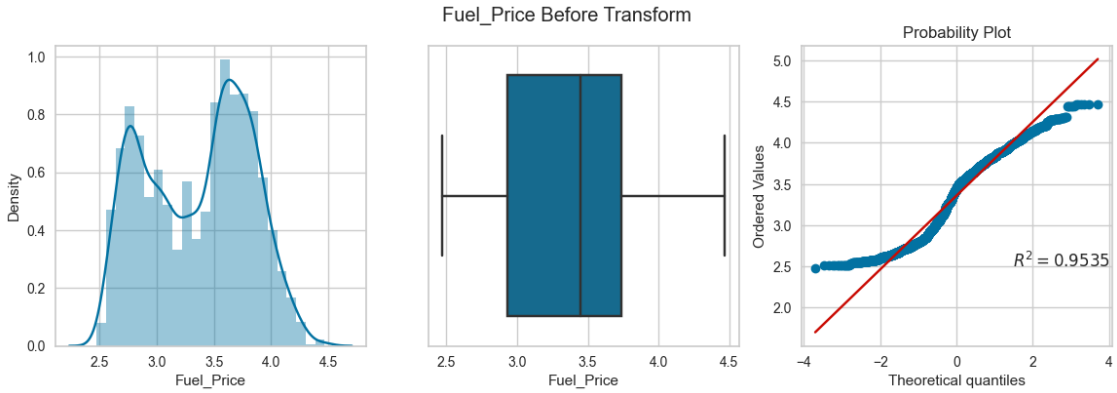
[30]:
```
for col in skewed_cols:
    apply_transform(FunctionTransformer(lambda x: 1/(x+0.0000001)),col)
```

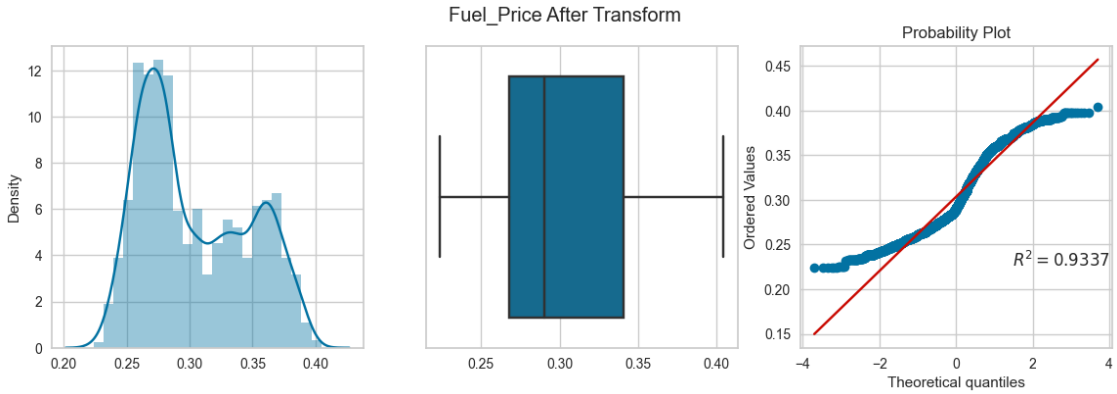

Weekly_Sales Before Transform
1e−6
Density
Weekly_Sales
1e6
Ordered Values
Probability Plot
$R^2 = 0.9500$
Theoretical quantiles

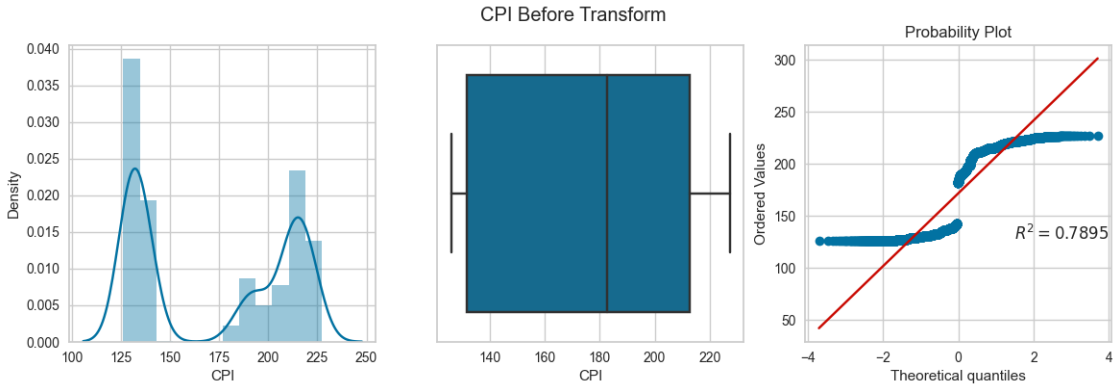

Weekly_Sales After Transform
1e6
Density
1e−6
Ordered Values
Probability Plot
$R^2 = 0.8614$
Theoretical quantiles

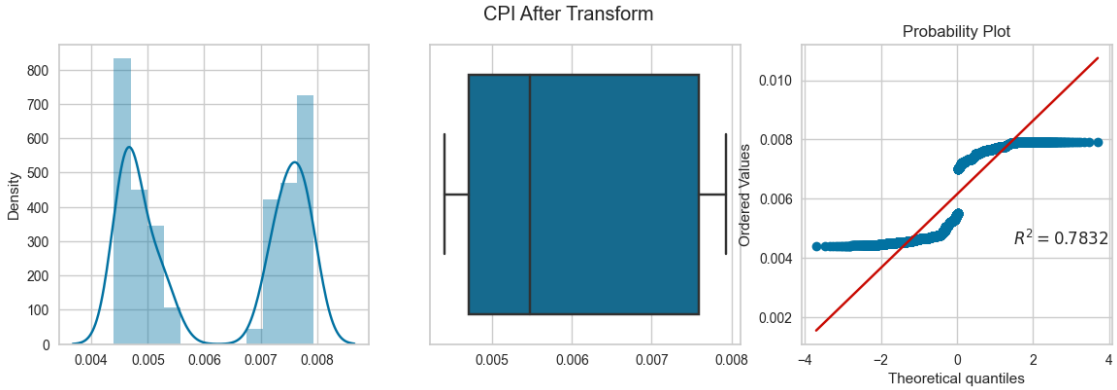

Unemployment Before Transform
Density
Unemployment
Ordered Values
Probability Plot
$R^2 = 0.9760$
Theoretical quantiles

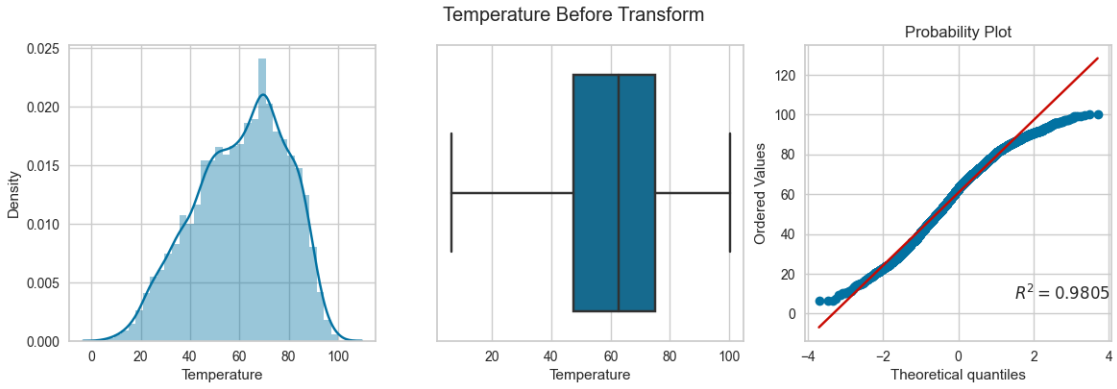

Unemployment After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9094$

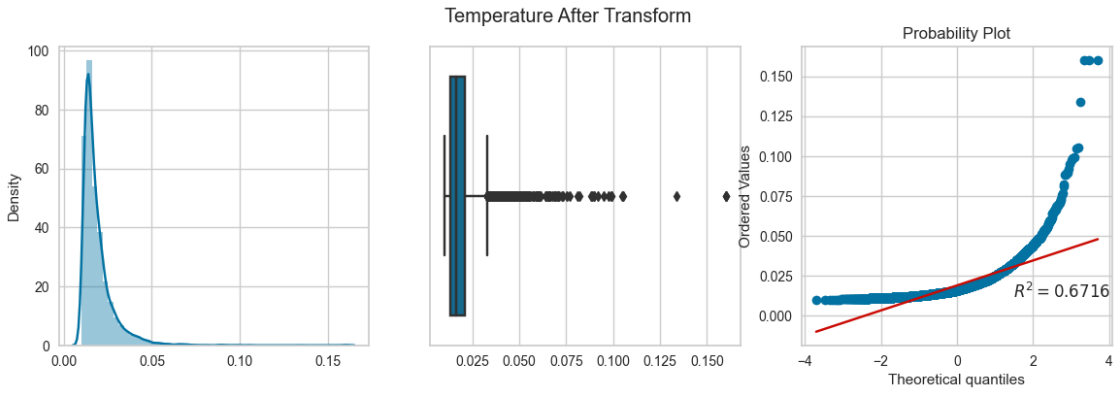

Fuel_Price Before Transform
Density
Fuel_Price
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9535$


Fuel_Price After Transform
Density
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9337$


CPI Before Transform
Density
CPI
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.7895$


CPI After Transform
Density
CPI
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.7832$


Temperature Before Transform
Density
Temperature
Probability Plot
Ordered Values
Theoretical quantiles
$R^2 = 0.9805$



Log Transform: None Power Transform: Weekly_Sales, Unemployment, Temperature Sqrt Transform: None Square Transform: Fuel_Price, CPI Reciprocal Transform: None

[31]:
```
pt = PowerTransformer()
pt_cols = ['Weekly_Sales','Unemployment','Temperature']

for col in pt_cols:
    col_tf = pt.fit_transform(df[[col]])
    col_tf = np.array(col_tf).reshape(col_tf.shape[0])
    df[col] = col_tf
```

[32]:
```
ft = FunctionTransformer(lambda x: x**2)
ft_cols = ['Fuel_Price','CPI']

for col in ft_cols:
    col_tf = ft.fit_transform(df[[col]])
    col_tf = np.array(col_tf).reshape(col_tf.shape[0])
    df[col] = col_tf
```

0.5.3 Feature Extraction

[33]:
```
df.Date = pd.to_datetime(df.Date,errors='coerce')
```

[34]:
```
df['Year'] = df.Date.dt.year
df['Month'] = df.Date.dt.month_name()
df['Day'] = df.Date.dt.day_name()
df['Days_In_Month'] = df.Date.dt.daysinmonth
df['Is_Leap_Year'] = df.Date.dt.is_leap_year
df['Is_Quarter_Start'] = df.Date.dt.is_quarter_start
df['Is_Quarter_End'] = df.Date.dt.is_quarter_end
df['Is_Year_End'] = df.Date.dt.is_year_end
df['Is_Month_Start'] = df.Date.dt.is_month_start
df['Is_Month_End'] = df.Date.dt.is_month_end
```

```
df['Day_Of_Year'] = df.Date.dt.day_of_year
df['Day_Of_Week'] = df.Date.dt.day_of_week
```

[35]:
```
df.drop('Date',axis=1,inplace=True)
```

0.5.4 Categorical Encoding

[36]:
```
oe = OrdinalEncoder(categories=[['January','February','March','April','May','June','July','Augus
df.Month = oe.fit_transform(df[['Month']])
df.Month = df.Month.astype(int)
```

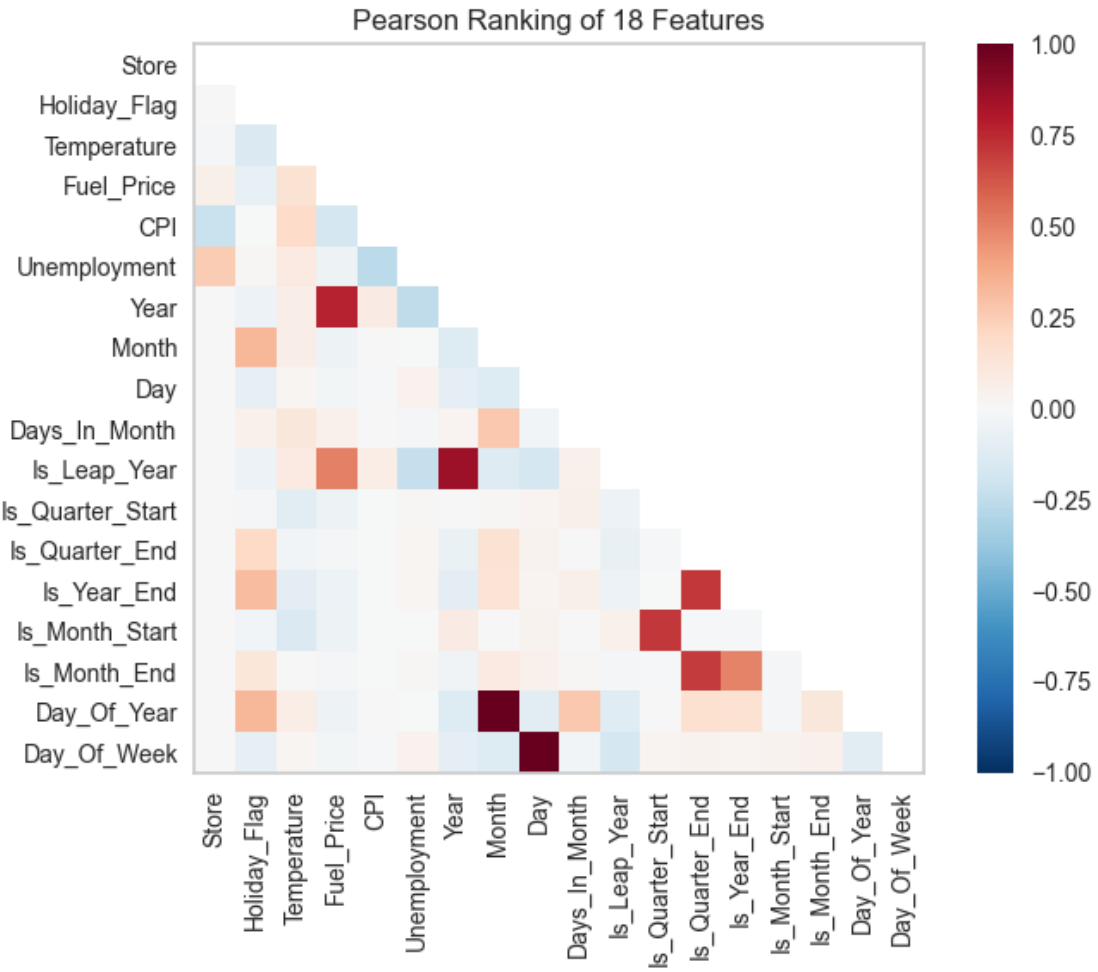
[37]:
```
oe = OrdinalEncoder(categories=[['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','
df.Day = oe.fit_transform(df[['Day']])
df.Day = df.Day.astype(int)
```

[38]:
```
df.Is_Leap_Year = df.Is_Leap_Year.astype(int)
df.Is_Quarter_Start = df.Is_Quarter_Start.astype(int)
df.Is_Quarter_End = df.Is_Quarter_End.astype(int)
df.Is_Year_End = df.Is_Year_End.astype(int)
df.Is_Month_Start = df.Is_Month_Start.astype(int)
df.Is_Month_End = df.Is_Month_End.astype(int)
```

0.5.5 Feature Splitting

[39]:
```
X = df.drop('Weekly_Sales',axis=1)
y = df.Weekly_Sales
```

[40]:
```
ranking_2d = Rank2D(algorithm='pearson')
ranking_2d.fit(X,y)
ranking_2d.transform(X)
ranking_2d.show();
```



[41]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.
↪3,random_state=101,shuffle=True)
```

0.5.6 Feature Selection

[42]:
```
kbest = SelectKBest(k=6,score_func=f_regression)
kbest.fit(X_train,y_train)
```

[42]:
```
SelectKBest(k=6, score_func=<function f_regression at 0x000001BE01581090>)
```

[43]:
```
selected_features = kbest.get_feature_names_out()
selected_features
```

[43]:
```
array(['Store', 'Temperature', 'CPI', 'Unemployment', 'Month',
       'Day_Of_Year'], dtype=object)
```

[44]:
```
percentile = SelectPercentile(percentile=30,score_func=f_regression)
percentile.fit(X_train,y_train)
```

[44]:
```
SelectPercentile(percentile=30,
                 score_func=<function f_regression at 0x000001BE01581090>)
```

[45]:
```
selected_features = percentile.get_feature_names_out()
selected_features
```

[45]:
```
array(['Store', 'Temperature', 'CPI', 'Unemployment', 'Month',
       'Day_Of_Year'], dtype=object)
```

[46]:
```
sfm = SelectFromModel(RandomForestRegressor(),threshold='median')
sfm.fit(X_train,y_train)
```

[46]:
```
SelectFromModel(estimator=RandomForestRegressor(), threshold='median')
```

[47]:
```
selected_features = sfm.get_feature_names_out()
selected_features
```

[47]:
```
array(['Store', 'Holiday_Flag', 'Temperature', 'Fuel_Price', 'CPI',
       'Unemployment', 'Month', 'Day_Of_Year', 'Day_Of_Week'],
      dtype=object)
```

[48]:
```
rfe = ␣
 ↪RFECV(estimator=RandomForestRegressor(),min_features_to_select=6,step=2,cv=5,verbose=1)
rfe.fit(X_train,y_train)
```

```
Fitting estimator with 18 features.
Fitting estimator with 16 features.
Fitting estimator with 14 features.
Fitting estimator with 12 features.
Fitting estimator with 10 features.
Fitting estimator with 8 features.
Fitting estimator with 18 features.
Fitting estimator with 16 features.
Fitting estimator with 14 features.
Fitting estimator with 12 features.
Fitting estimator with 10 features.
Fitting estimator with 8 features.
Fitting estimator with 18 features.
Fitting estimator with 16 features.
Fitting estimator with 14 features.
Fitting estimator with 12 features.
Fitting estimator with 10 features.
Fitting estimator with 8 features.
Fitting estimator with 18 features.
Fitting estimator with 16 features.
```

```
Fitting estimator with 14 features.
Fitting estimator with 12 features.
Fitting estimator with 10 features.
Fitting estimator with 8 features.
Fitting estimator with 18 features.
Fitting estimator with 16 features.
Fitting estimator with 14 features.
Fitting estimator with 12 features.
Fitting estimator with 10 features.
Fitting estimator with 8 features.
Fitting estimator with 18 features.
Fitting estimator with 16 features.
Fitting estimator with 14 features.
```

[48]:
```
RFECV(cv=5, estimator=RandomForestRegressor(), min_features_to_select=6, step=2,
      verbose=1)
```

[49]:
```
selected_features = rfe.get_feature_names_out()
selected_features
```

[49]:
```
array(['Store', 'Holiday_Flag', 'Temperature', 'Fuel_Price', 'CPI',
       'Unemployment', 'Year', 'Month', 'Day', 'Days_In_Month',
       'Day_Of_Year', 'Day_Of_Week'], dtype=object)
```

[50]:
```
sfs =␣
 ↪SequentialFeatureSelector(estimator=RandomForestRegressor(),n_features_to_select=6,directio
sfs.fit(X_train,y_train)
```

[50]:
```
SequentialFeatureSelector(cv=3, estimator=RandomForestRegressor(),
                          n_features_to_select=6)
```

[51]:
```
selected_features = sfs.get_feature_names_out()
selected_features
```

[51]:
```
array(['Store', 'Holiday_Flag', 'CPI', 'Unemployment', 'Day',
       'Day_Of_Year'], dtype=object)
```

[52]:
```
lasso = SelectFromModel(estimator=LassoCV(),threshold='0.95*mean')
lasso.fit(X_train,y_train)
```

[52]:
```
SelectFromModel(estimator=LassoCV(), threshold='0.95*mean')
```

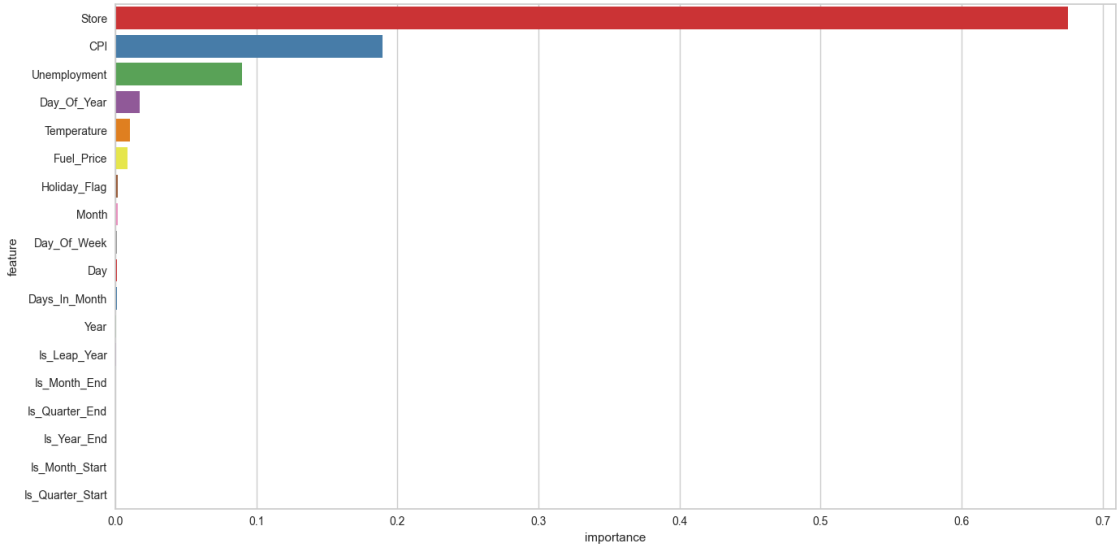
[53]:
```
selected_features = lasso.get_feature_names_out()
selected_features
```

[53]:
```
array(['Store'], dtype=object)
```

[54]:
```
rf = RandomForestRegressor()
rf.fit(X_train,y_train)
```

[54]:
```
RandomForestRegressor()
```

[55]:
```
feat_imps = pd.DataFrame({'feature': X_train.columns, 'importance': rf.
 ↪feature_importances_}).sort_values('importance',ascending=False)
plt.figure(figsize=(14,7))
sns.
 ↪barplot(x='importance',y='feature',data=feat_imps,orient='horizontal',palette='Set1')
plt.tight_layout();
```



[56]:
```
df.corr()['Weekly_Sales'].sort_values(ascending=False)[1:]
```

[56]:
```
Month              0.050730
Day_Of_Year        0.048769
Holiday_Flag       0.029496
Fuel_Price         0.016294
Is_Leap_Year      -0.006772
Days_In_Month     -0.008107
Year              -0.009913
Is_Month_Start    -0.011720
Is_Quarter_Start  -0.011894
Is_Year_End       -0.020678
Day               -0.020999
Day_Of_Week       -0.020999
Is_Month_End      -0.022571
Is_Quarter_End    -0.025321
```

```
CPI                -0.082511
Temperature        -0.085931
Unemployment       -0.087557
Store              -0.308973
Name: Weekly_Sales, dtype: float64
```

Final selected features: Store, CPI, Unemployment, Day_Of_Year, Temperature, Month, Holiday_Flag

[57]:
```
selected_features =␣
↪['Store','CPI','Unemployment','Day_Of_Year','Temperature','Month','Holiday_Flag']

X_train = X_train[selected_features]
X_test = X_test[selected_features]
```

0.5.7 Feature Scaling

[58]:
```
scaler = StandardScaler()
features = X_train.columns
X_train = scaler.fit_transform(X_train)
X_train = pd.DataFrame(X_train,columns=features)
X_test = scaler.transform(X_test)
X_test = pd.DataFrame(X_test,columns=features)
X_train.head()
```

[58]:

	Store	CPI	Unemployment	Day_Of_Year	Temperature	Month	Holiday_Flag
0	1.304226	-0.974622	-1.870001	-0.865484	-2.253026	-0.728108	-0.268562
1	1.458359	-1.013378	-0.580989	1.183872	0.511933	1.078226	-0.268562
2	-1.470166	-1.054563	-1.270860	-0.234913	0.109179	-0.125997	-0.268562
3	-1.470166	-1.109167	-0.292376	-1.397529	0.695575	-1.330220	-0.268562
4	-1.470166	-1.039096	-1.848716	0.740502	-1.509232	0.777170	-0.268562

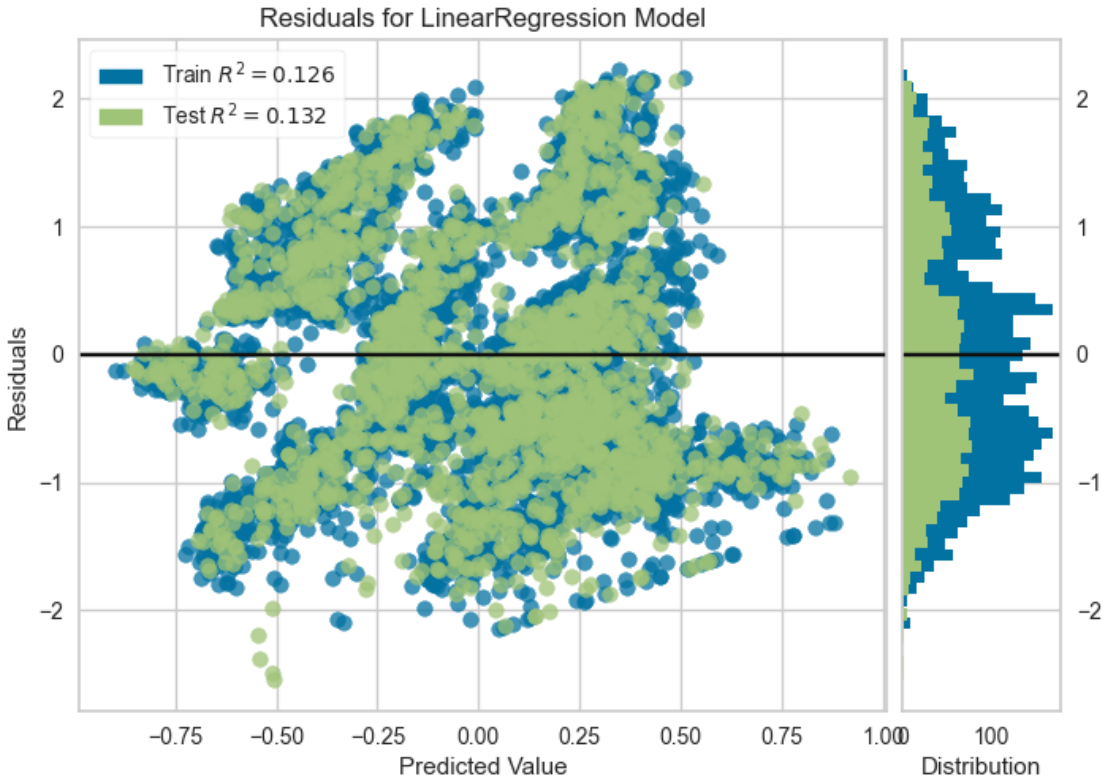
0.6 Model Training & Evaluation

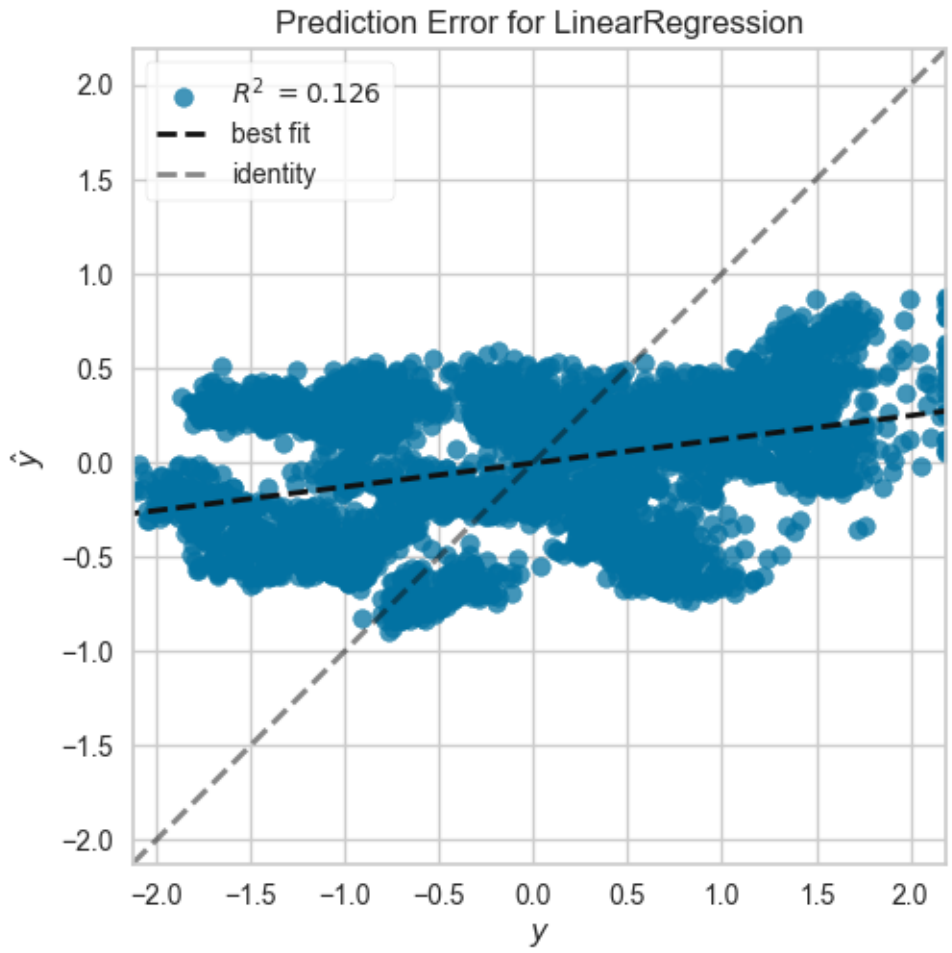
[59]:
```
models = []
mape_scores = []
rmse_scores = []
r2_scores = []
```

[144]:
```
def train_and_evaluate_model(model):
    model.fit(X_train,y_train)
    pred = model.predict(X_test)
    mape = mean_absolute_percentage_error(y_test,pred)
    print("Mean Absolute Percentage Error(MAPE):",mape)
    mape_scores.append(mape)
    rmse = np.sqrt(mean_squared_error(y_test,pred))
    print("Root Mean Squared Error(RMSE):",rmse)
    rmse_scores.append(rmse)
    r2 = r2_score(y_test,pred)
    print("R2 Score:",r2)
    r2_scores.append(r2)
    models.append(model)

    if re.search('catboost',str(model)) == None or re.
 ↪search('Voting',str(model)) != None:
        residuals_plot = ResidualsPlot(model)
        residuals_plot.fit(X_train,y_train)
        residuals_plot.score(X_test,y_test)
        residuals_plot.show()
        predictions_error = PredictionError(model)
        predictions_error.score(X_train,y_train)
        predictions_error.fit(X_train,y_train)
        predictions_error.show()

        if str(model) == 'LassoCV()':
            visualizer = AlphaSelection(LassoCV(alphas=np.logspace(-10,10,400)))
            visualizer.fit(X_train,y_train)
            visualizer.show()
        elif str(model) == 'RidgeCV()':
            visualizer = AlphaSelection(RidgeCV(alphas=np.logspace(-10,10,400)))
            visualizer.fit(X_train,y_train)
            visualizer.show()
        elif str(model) == 'ElasticNetCV()':
            visualizer = AlphaSelection(ElasticNetCV(alphas=np.
 ↪logspace(-10,10,400)))
            visualizer.fit(X_train,y_train)
            visualizer.show()
```

[61]:
```
train_and_evaluate_model(LinearRegression())
```

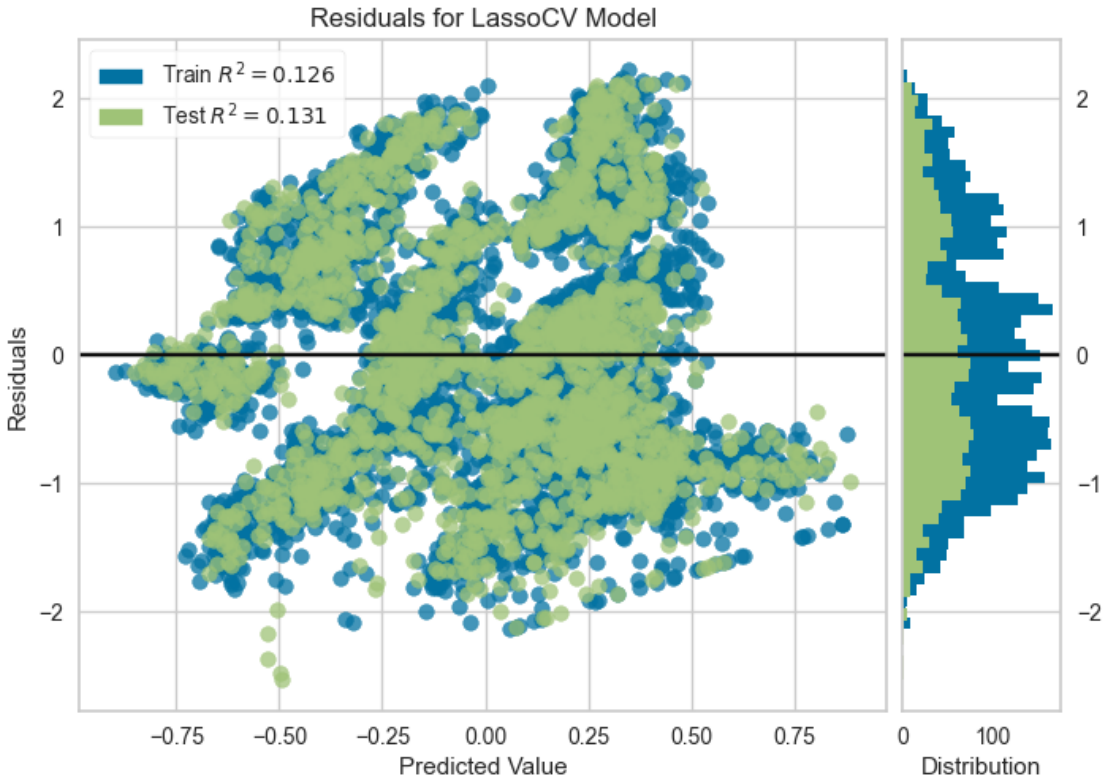
```
Mean Absolute Percentage Error(MAPE): 1.8588602855362242
Root Mean Squared Error(RMSE): 0.9428056032781673
R2 Score: 0.13166572867067572
```

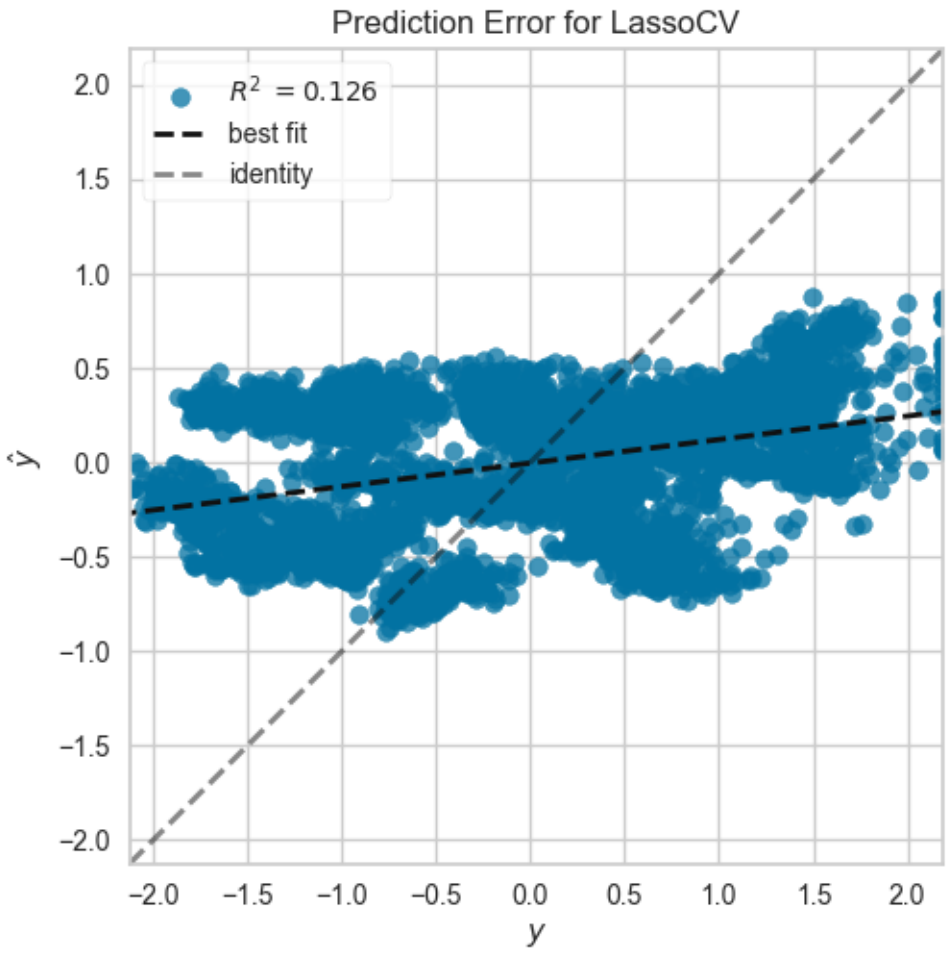

Residuals for LinearRegression Model
Train $R^2 = 0.126$
Test $R^2 = 0.132$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
100

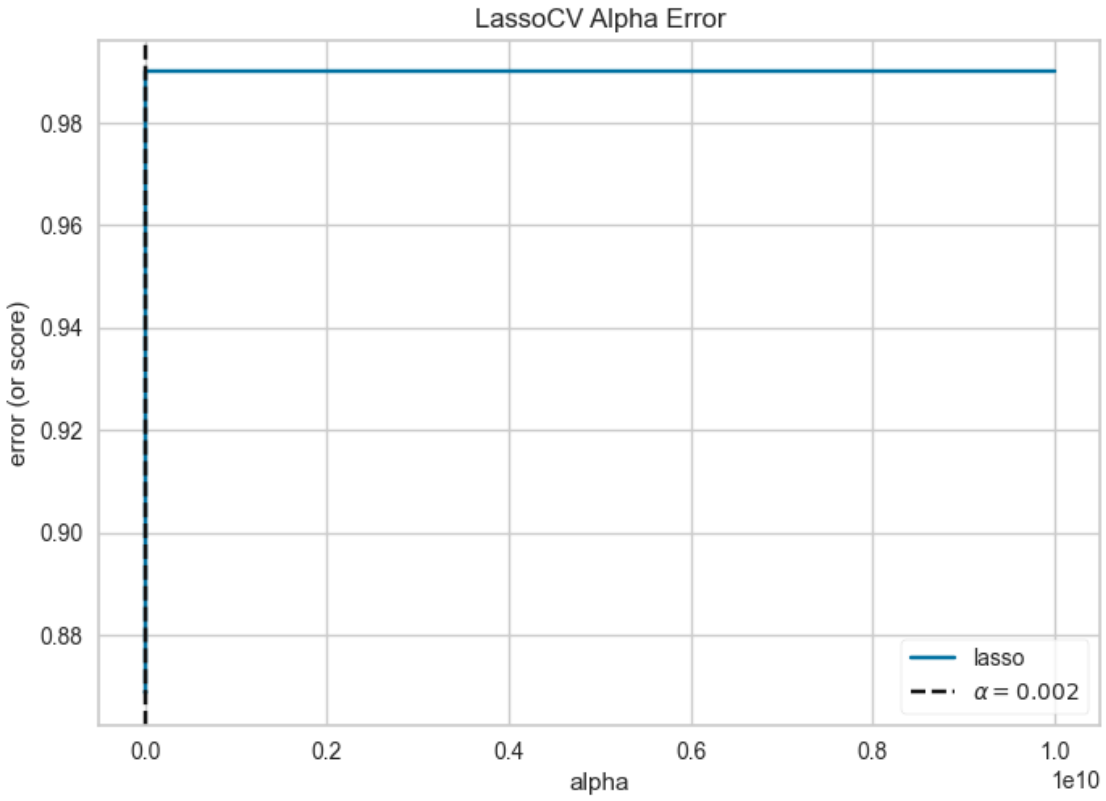


[62]:
```
train_and_evaluate_model(LassoCV())
```

```
Mean Absolute Percentage Error(MAPE): 1.87217359396011
Root Mean Squared Error(RMSE): 0.9430963176756088
R2 Score: 0.131130143830821
```

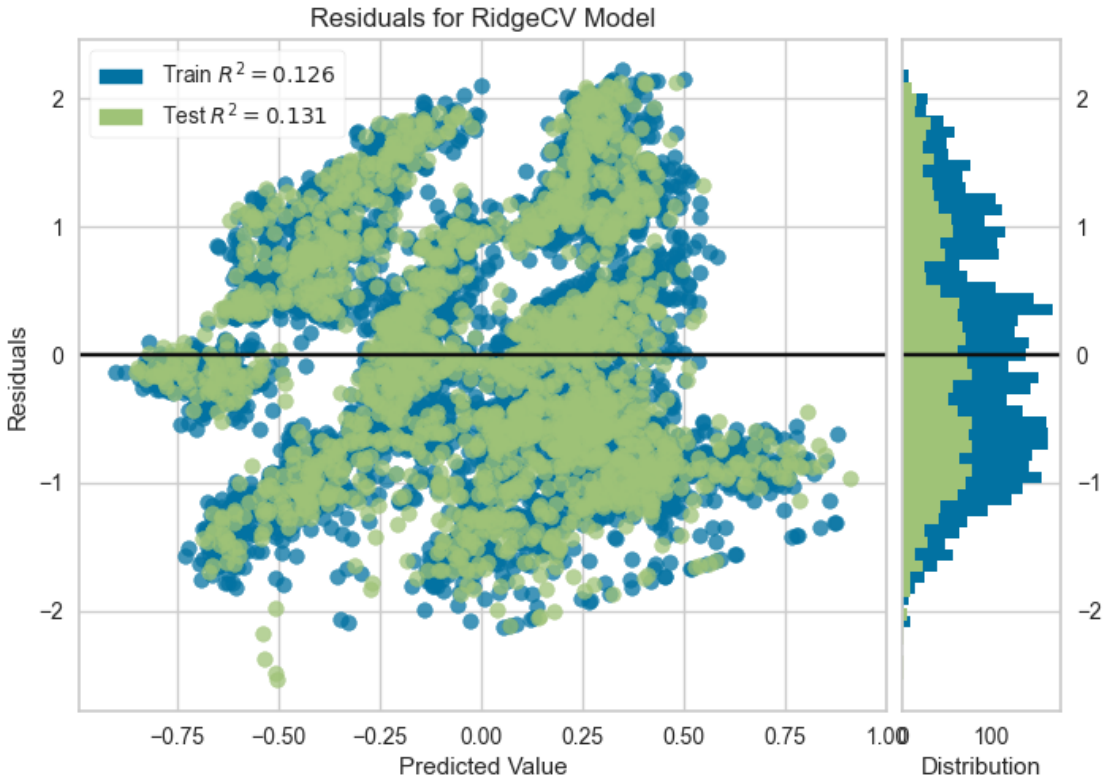

Residuals for LassoCV Model
Train $R^2 = 0.126$
Test $R^2 = 0.131$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
0
100

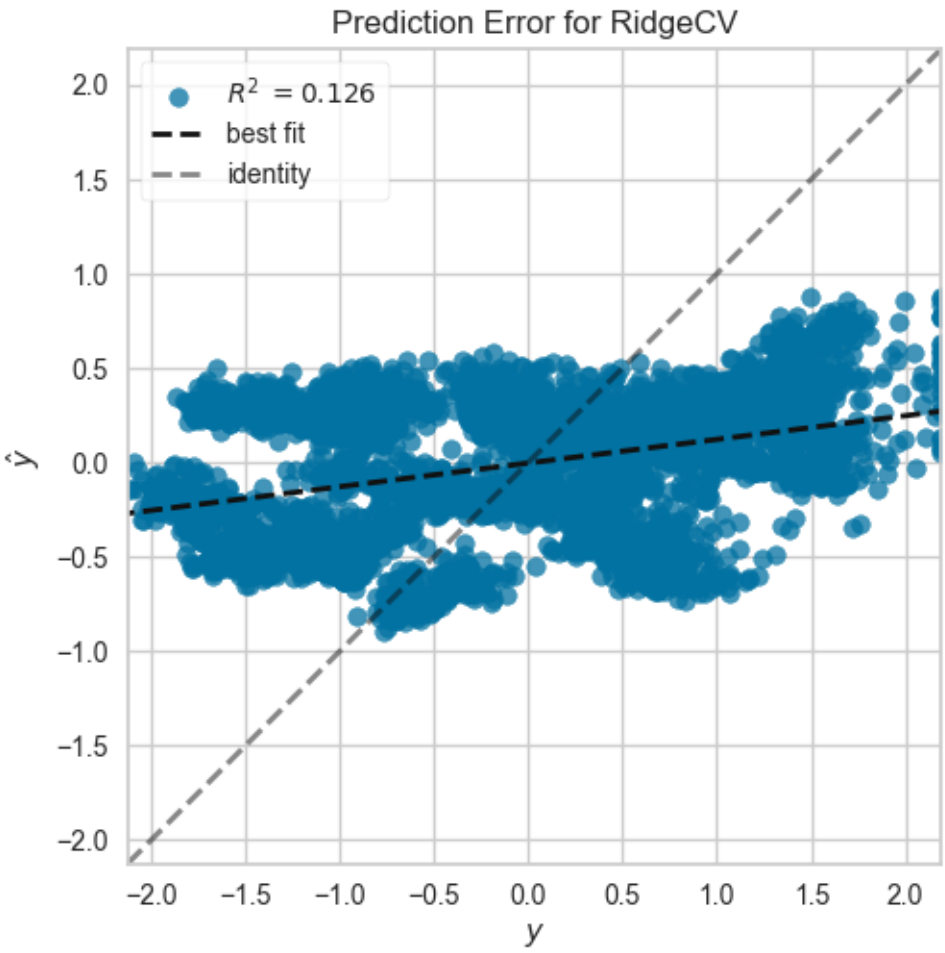

Prediction Error for LassoCV
R^2 = 0.126
best fit
identity
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
$\hat{y}$
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
y

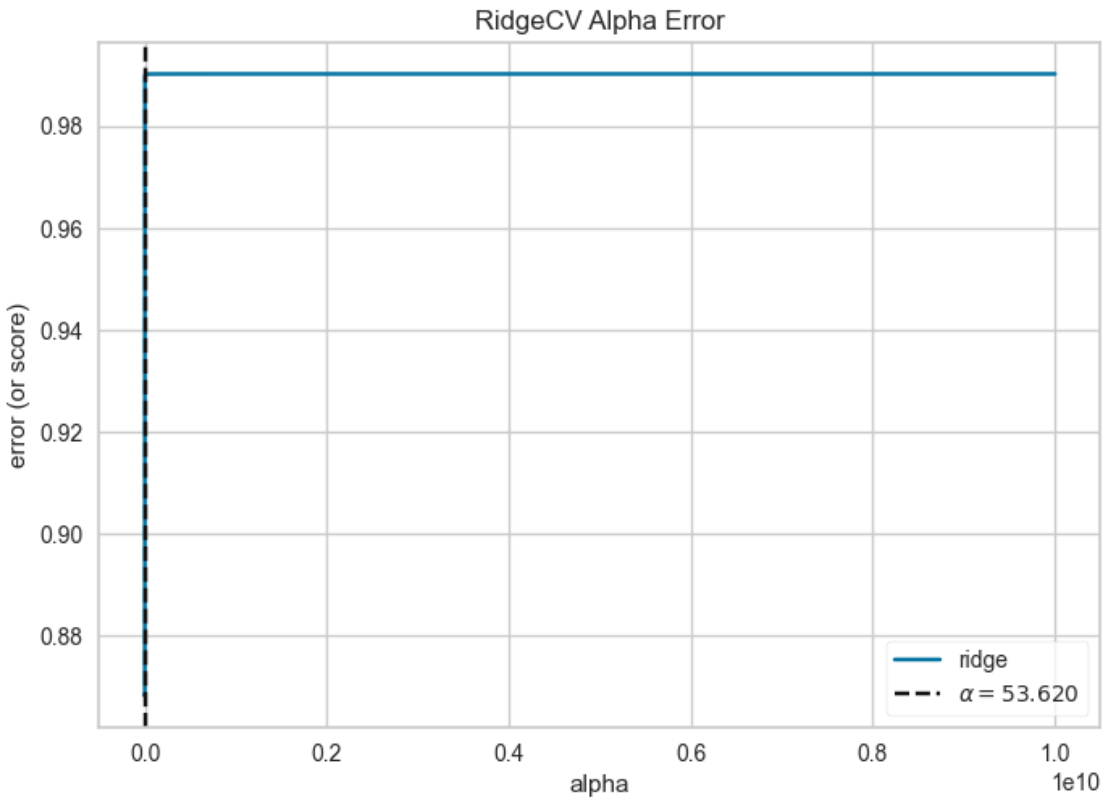


[63]:
```
train_and_evaluate_model(RidgeCV())
```

```
Mean Absolute Percentage Error(MAPE): 1.8583366137333945
Root Mean Squared Error(RMSE): 0.9429127460413069
R2 Score: 0.1314683581433027
```

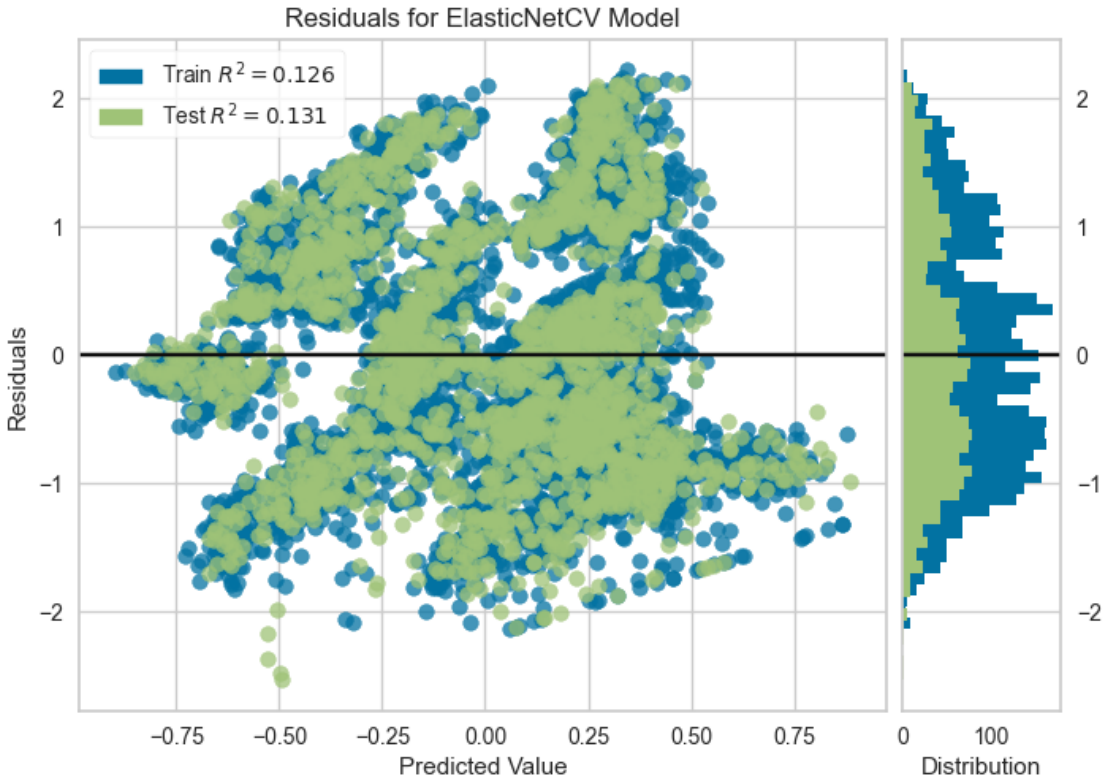

Residuals for RidgeCV Model
Train $R^2 = 0.126$
Test $R^2 = 0.131$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
100

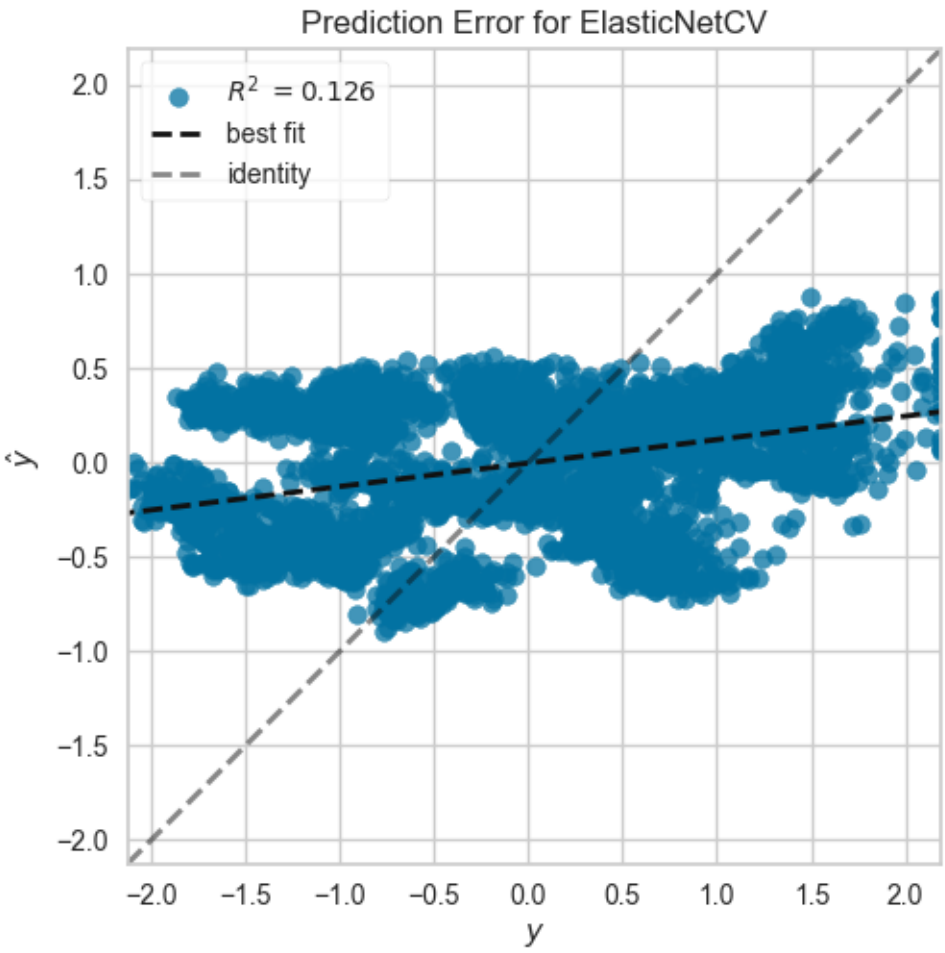

Prediction Error for RidgeCV
R^2 = 0.126
best fit
identity
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
$\hat{y}$
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
y

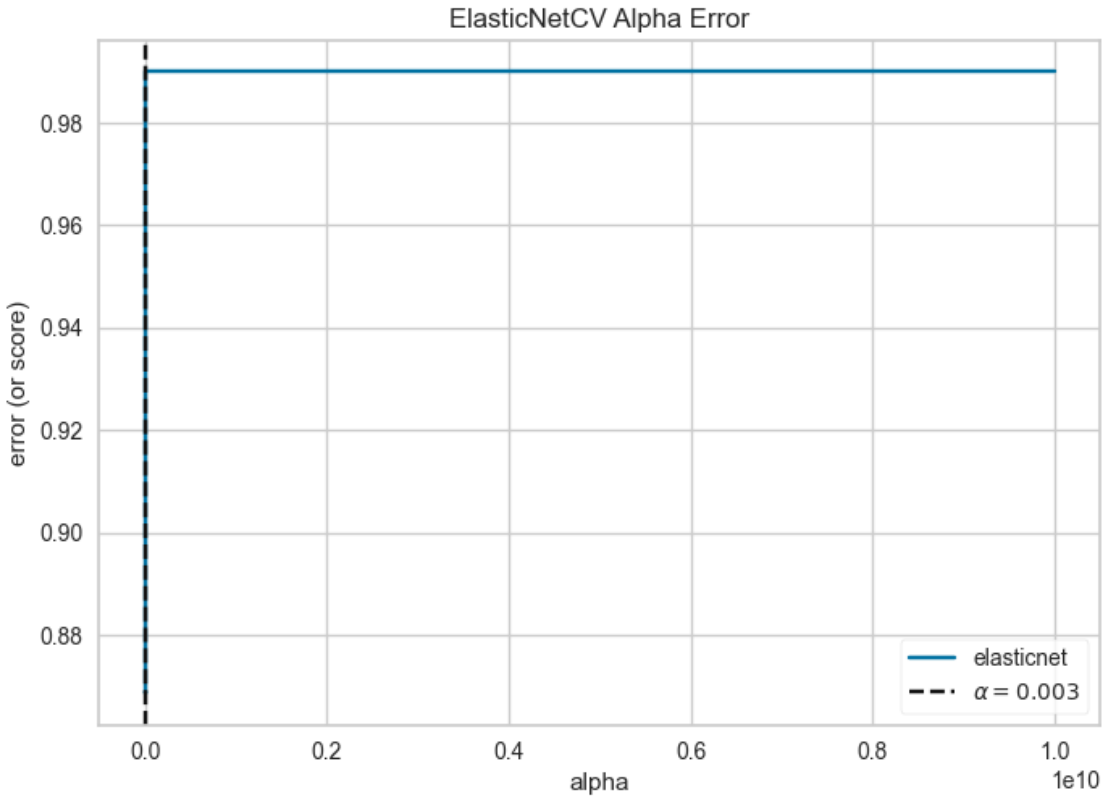


[64]:
```
train_and_evaluate_model(ElasticNetCV())
```

```
Mean Absolute Percentage Error(MAPE): 1.8694173949342796
Root Mean Squared Error(RMSE): 0.9431117395338787
R2 Score: 0.13110172743880832
```

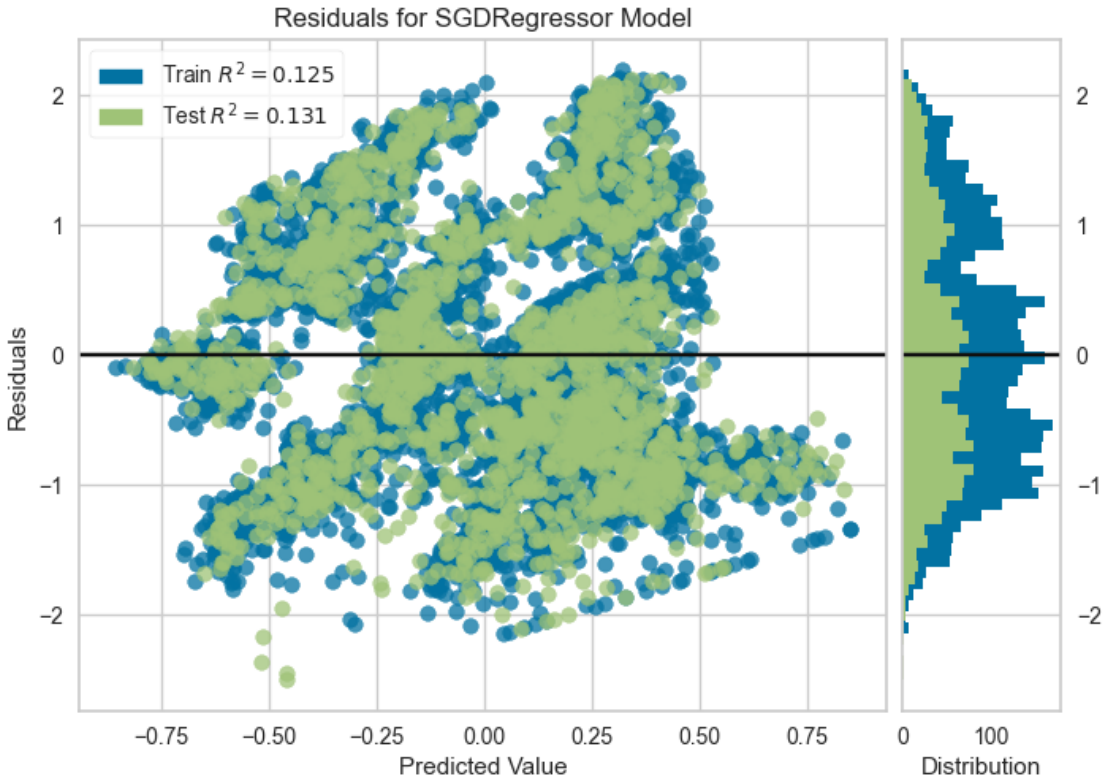

Residuals for ElasticNetCV Model
Train $R^2 = 0.126$
Test $R^2 = 0.131$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
0
100

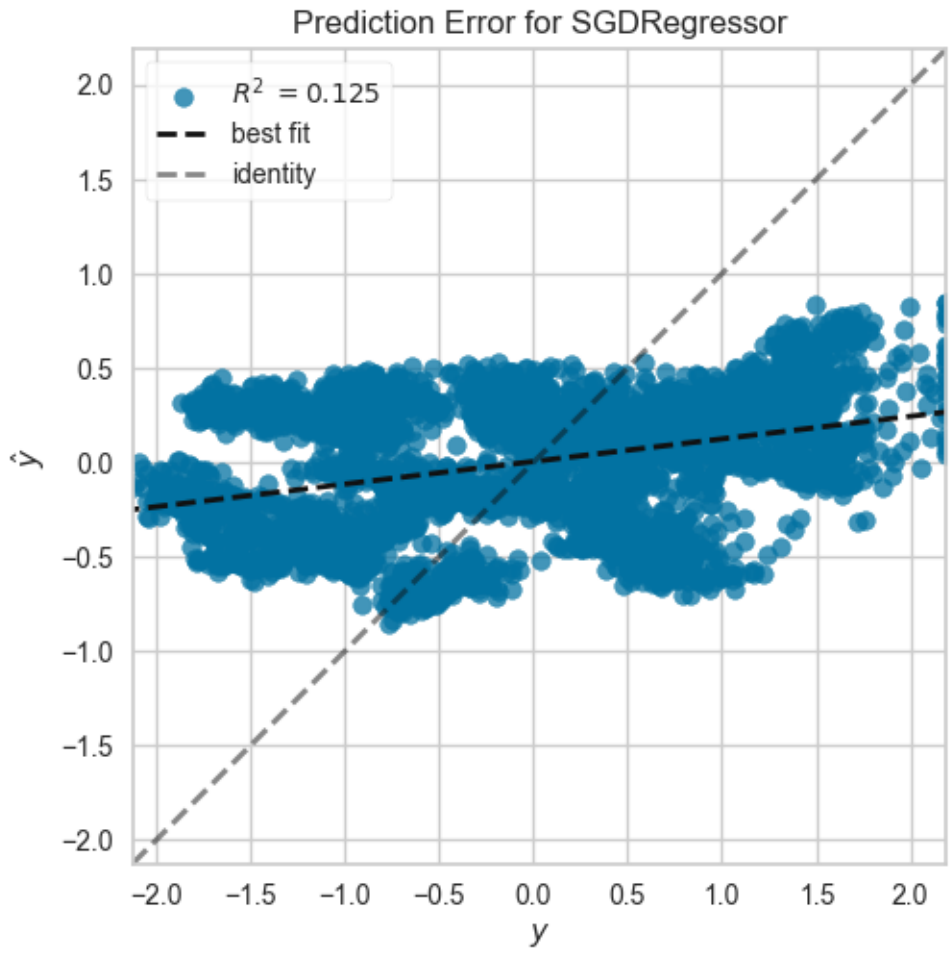

Prediction Error for ElasticNetCV
R^2 = 0.126
best fit
identity
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
$\hat{y}$
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
y



[65]:
```
train_and_evaluate_model(SGDRegressor())
```

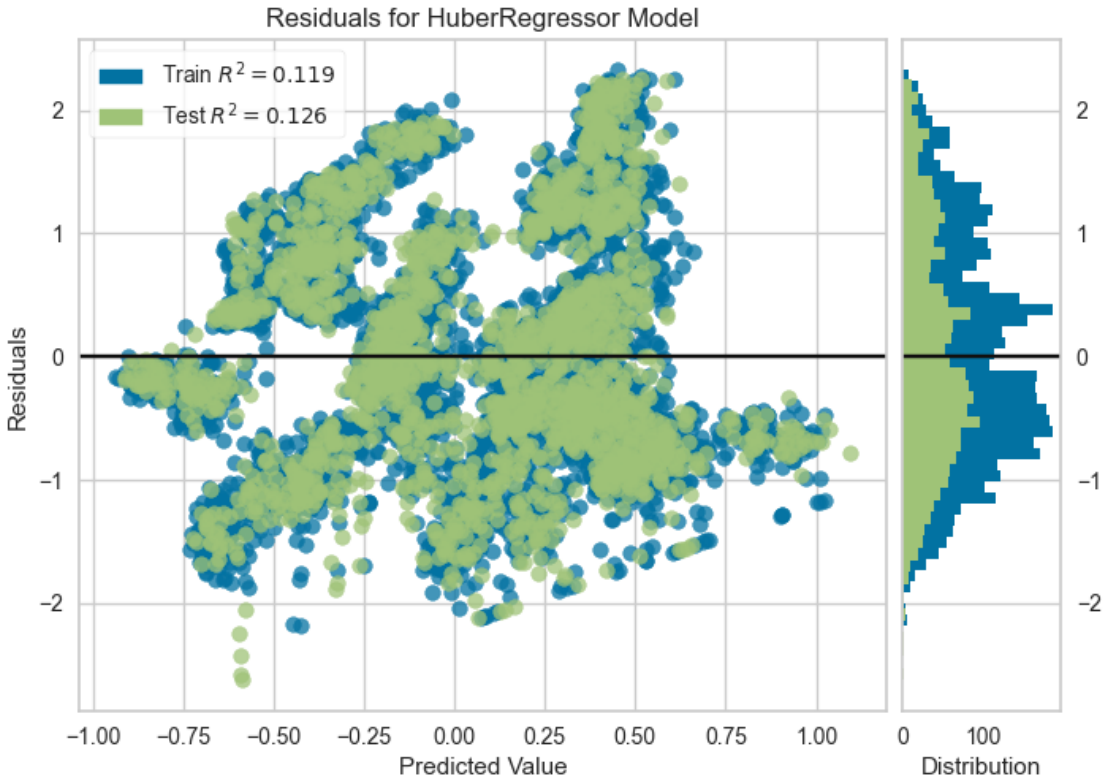
```
Mean Absolute Percentage Error(MAPE): 1.8248079631075012
Root Mean Squared Error(RMSE): 0.9429675988096315
R2 Score: 0.13136730372257965
```

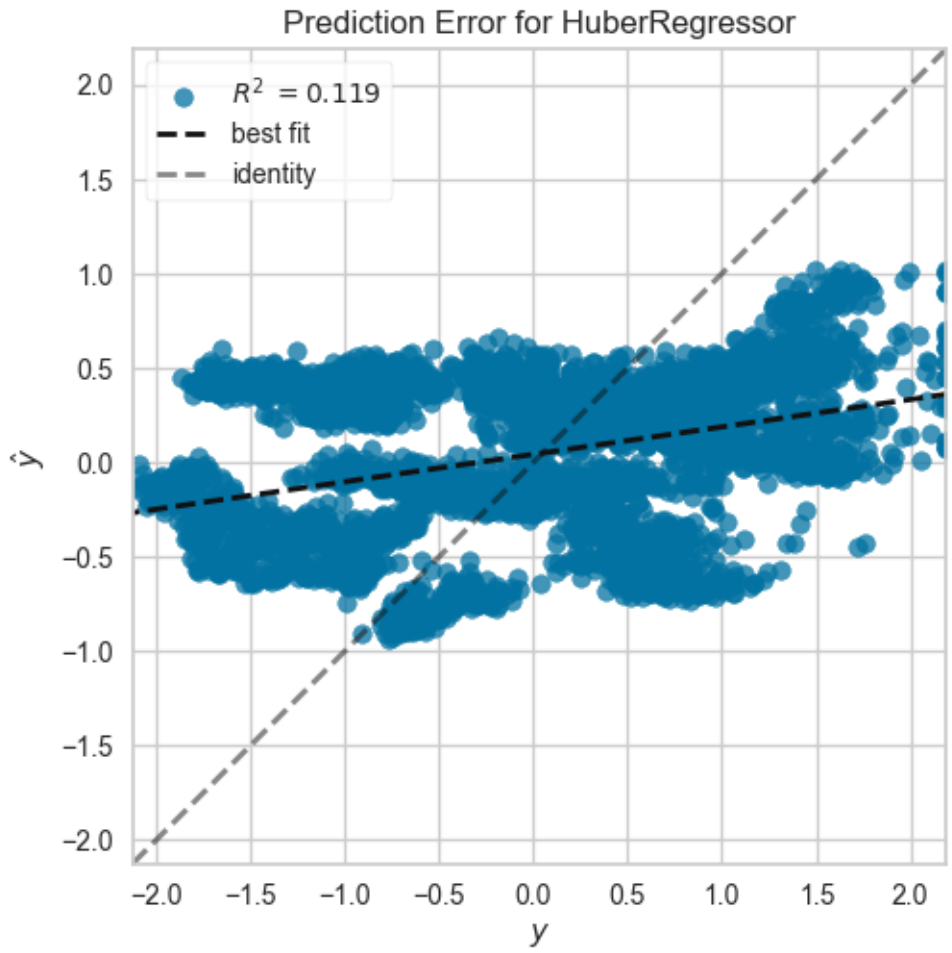

Residuals for SGDRegressor Model
Train $R^2 = 0.125$
Test $R^2 = 0.131$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
0
100



[66]:
```
train_and_evaluate_model(HuberRegressor())
```

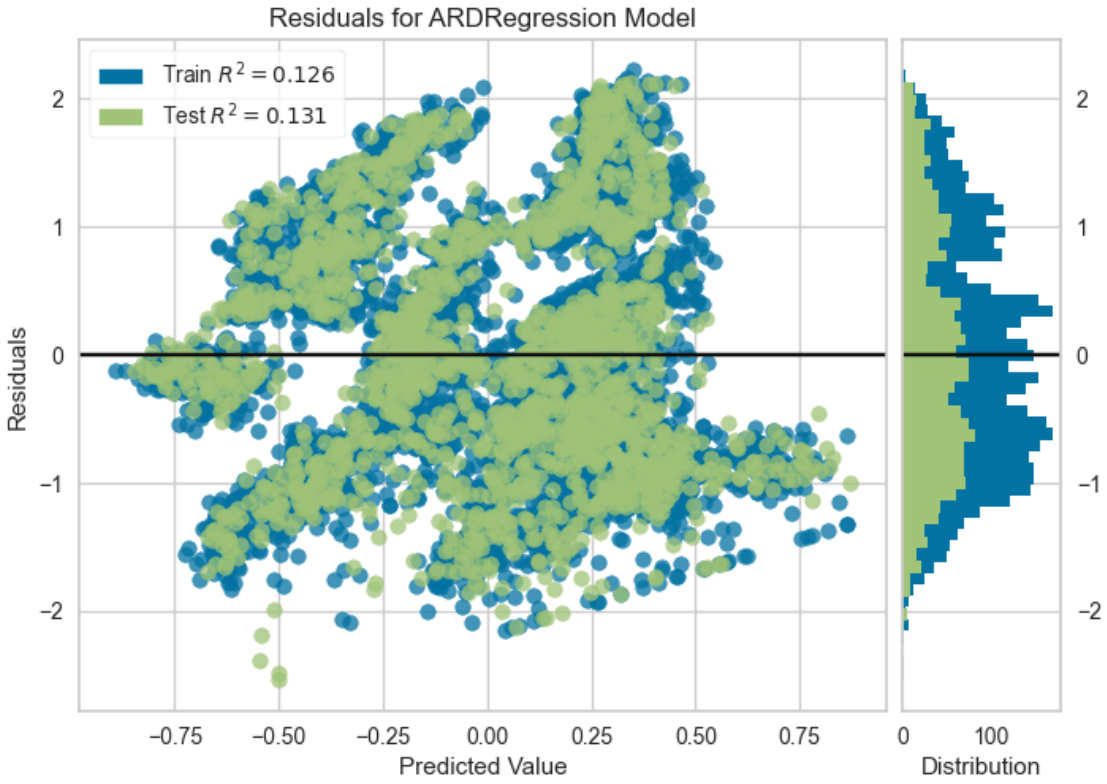
```
Mean Absolute Percentage Error(MAPE): 2.047510142946776
Root Mean Squared Error(RMSE): 0.9456807950172547
R2 Score: 0.12636148697284888
```

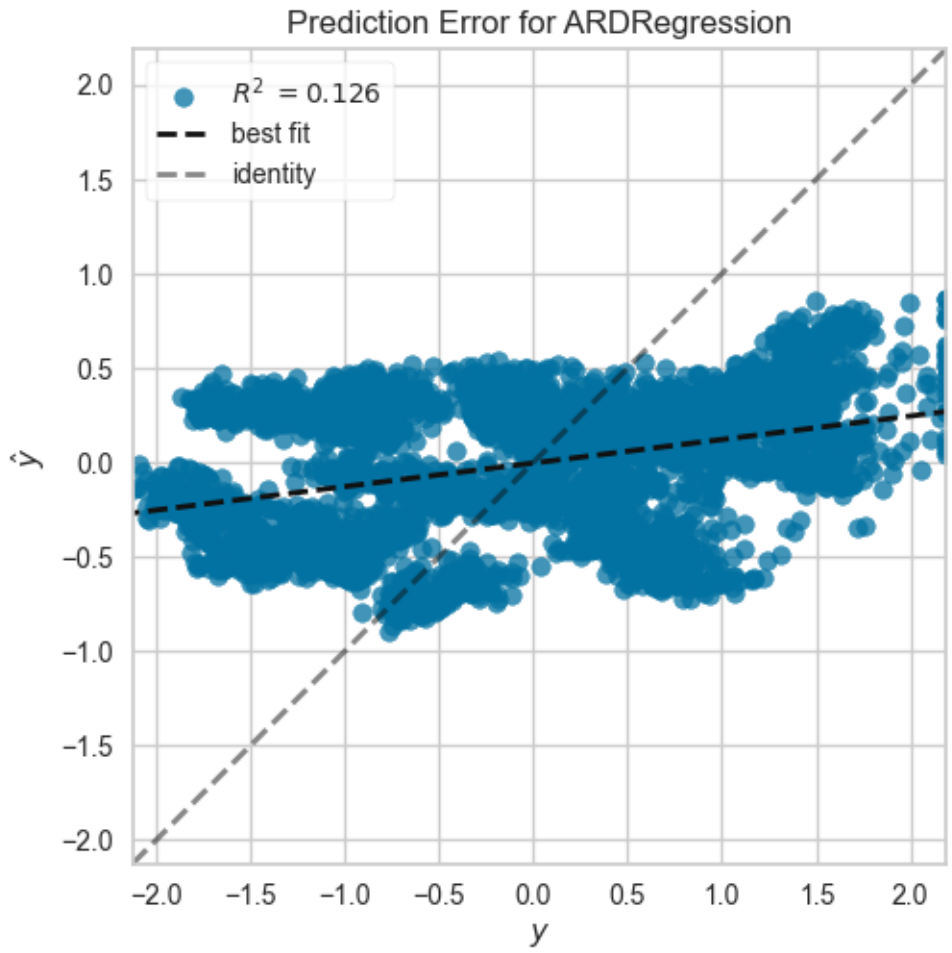

Residuals for HuberRegressor Model
Train $R^2 = 0.119$
Test $R^2 = 0.126$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
0
100



[67]:
```
train_and_evaluate_model(ARDRegression())
```

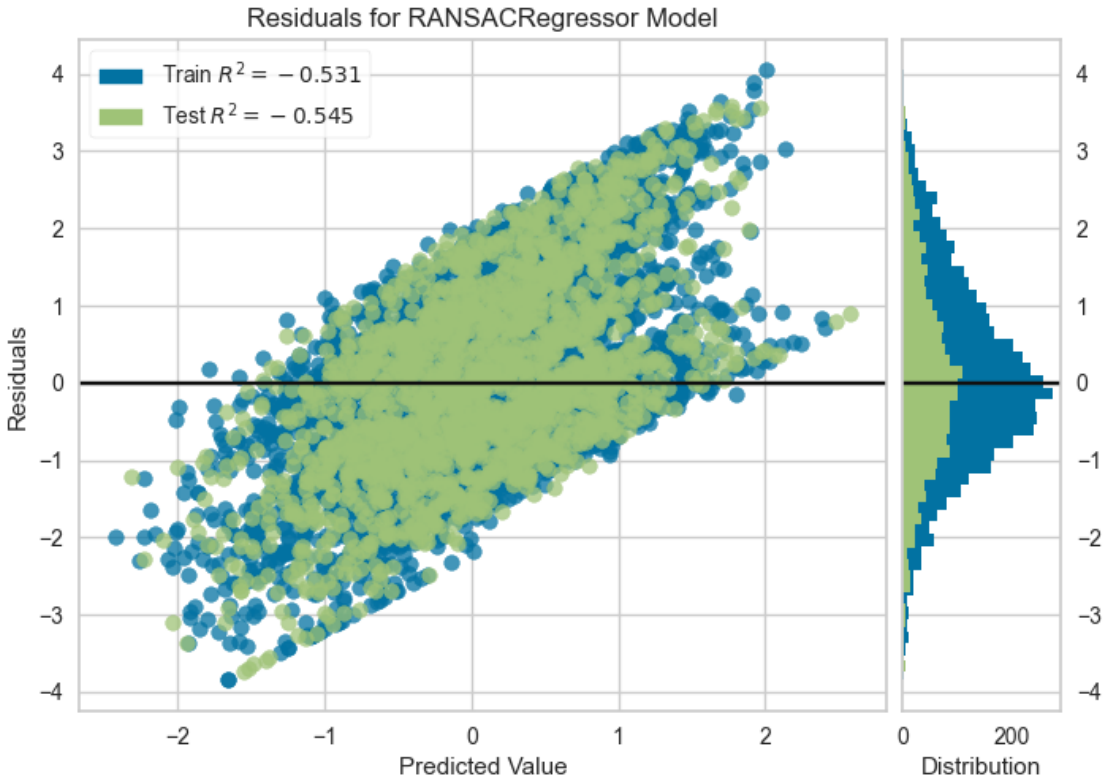
```
Mean Absolute Percentage Error(MAPE): 1.8962842173507548
Root Mean Squared Error(RMSE): 0.9431246081769947
R2 Score: 0.13107801525797047
```

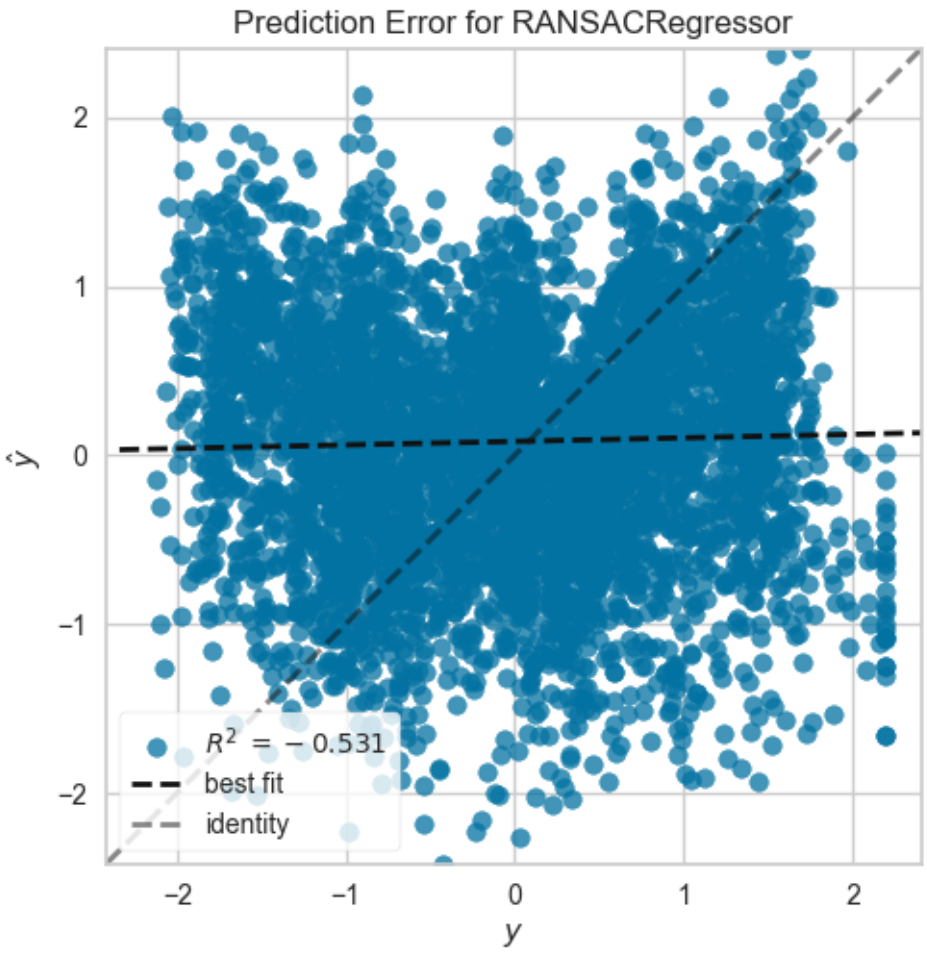

Residuals for ARDRegression Model
Train $R^2 = 0.126$
Test $R^2 = 0.131$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
0
100



[68]:
```
train_and_evaluate_model(RANSACRegressor())
```

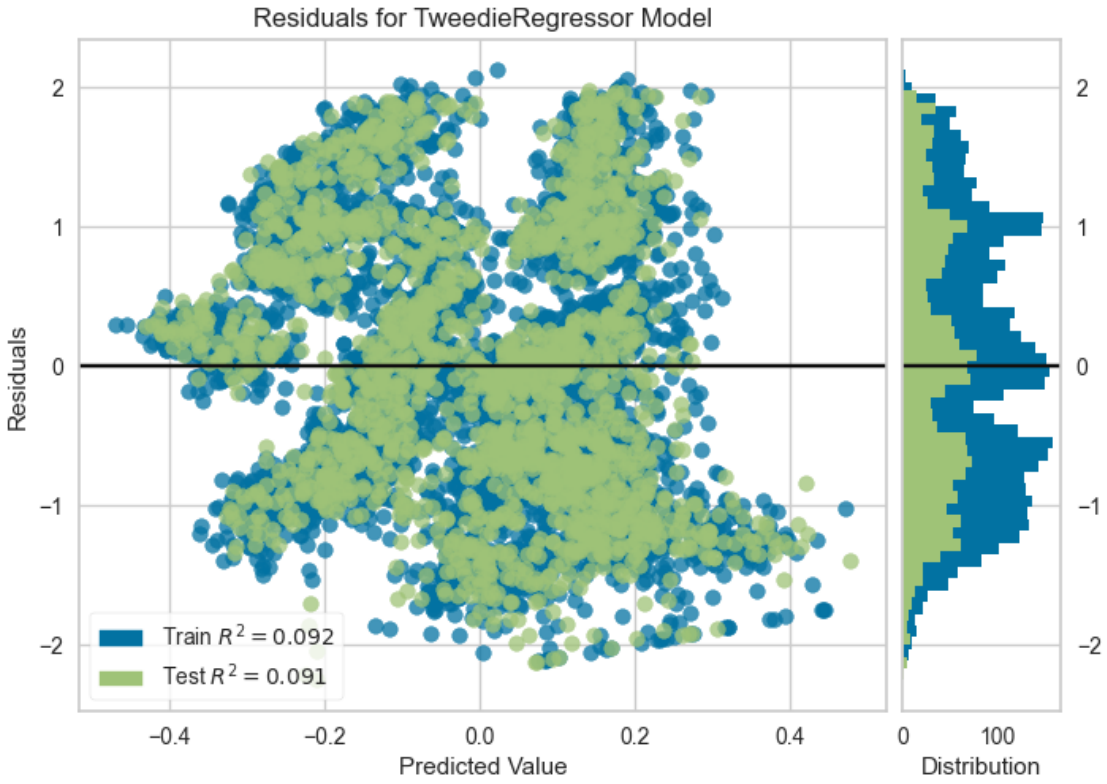
```
Mean Absolute Percentage Error(MAPE): 4.191927385484206
Root Mean Squared Error(RMSE): 1.2575291396130108
R2 Score: -0.5448230714503632
```

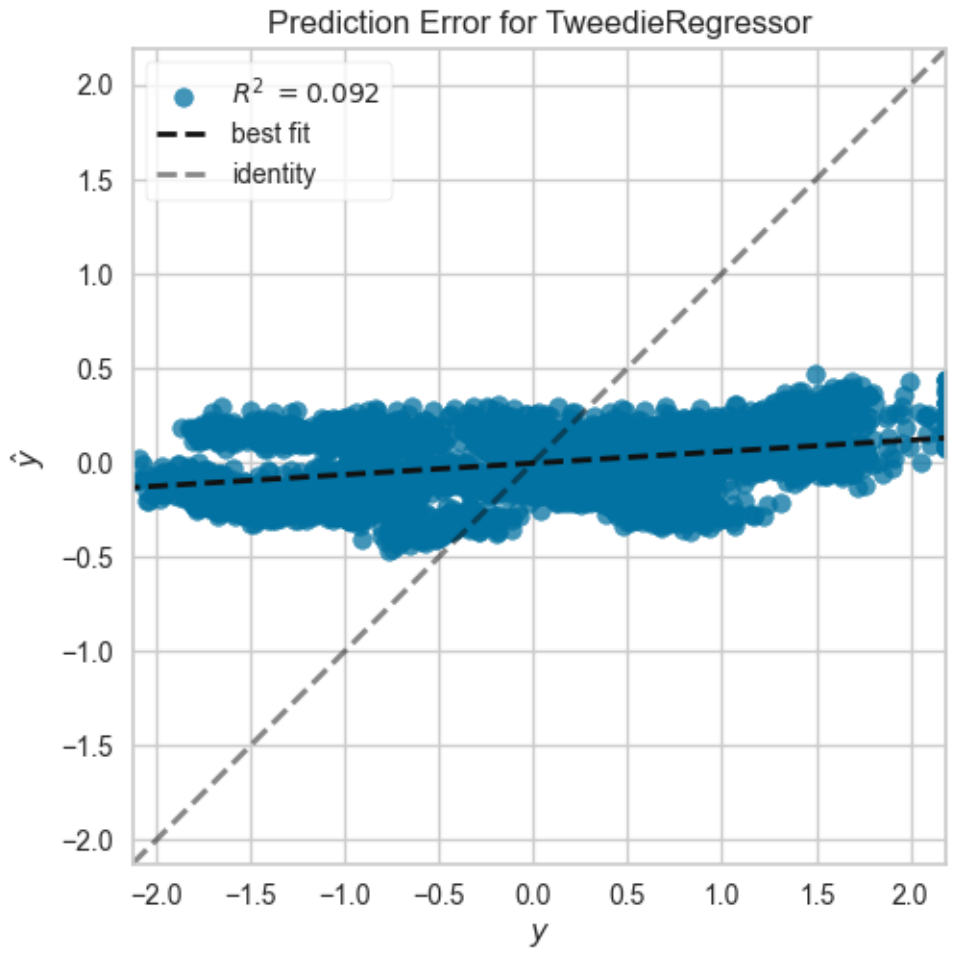

Residuals for RANSACRegressor Model
Train $R^2 = -0.531$
Test $R^2 = -0.545$
Residuals
Predicted Value
Distribution
4
3
2
1
0
−1
−2
−3
−4
−2
−1
0
1
2
0
200



[69]:
```
train_and_evaluate_model(TweedieRegressor())
```

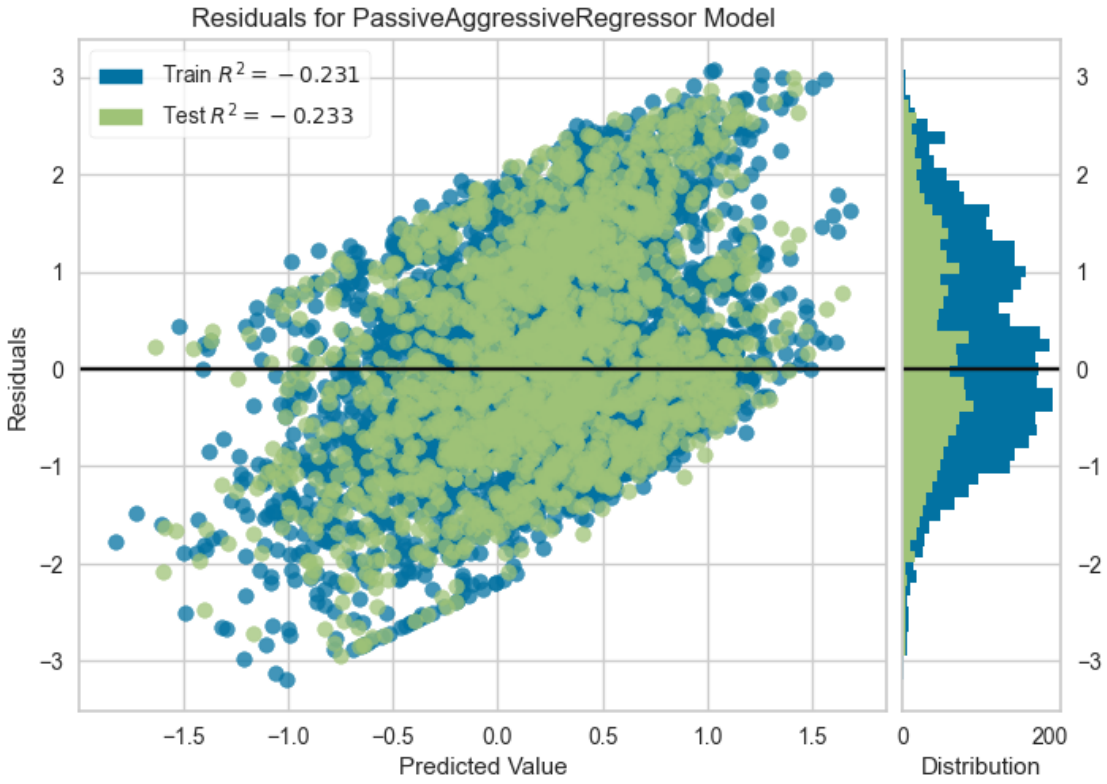
```
Mean Absolute Percentage Error(MAPE): 1.30032699724825
Root Mean Squared Error(RMSE): 0.9645206708385978
R2 Score: 0.09120545570586358
```

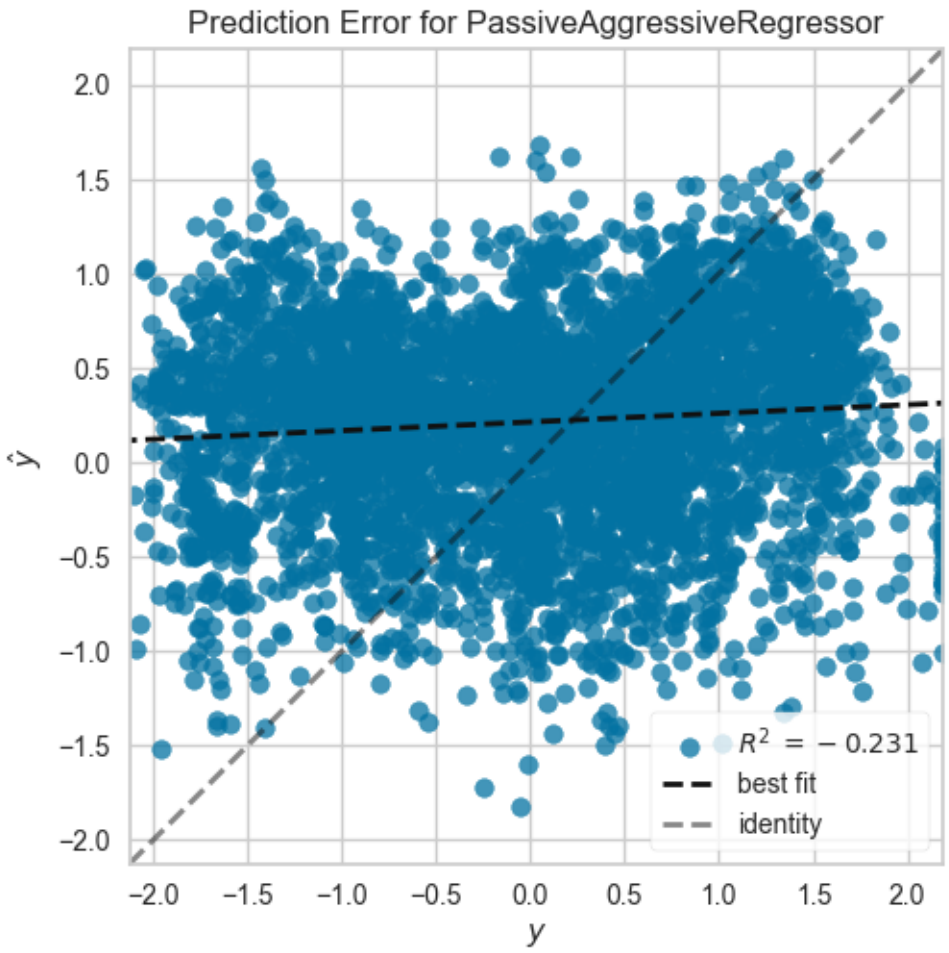

Residuals for TweedieRegressor Model
Residuals
Predicted Value
Distribution
Train $R^2 = 0.092$
Test $R^2 = 0.091$
2
1
0
−1
−2
−0.4
−0.2
0.0
0.2
0.4
0
100



[70]:
```
train_and_evaluate_model(PassiveAggressiveRegressor())
```

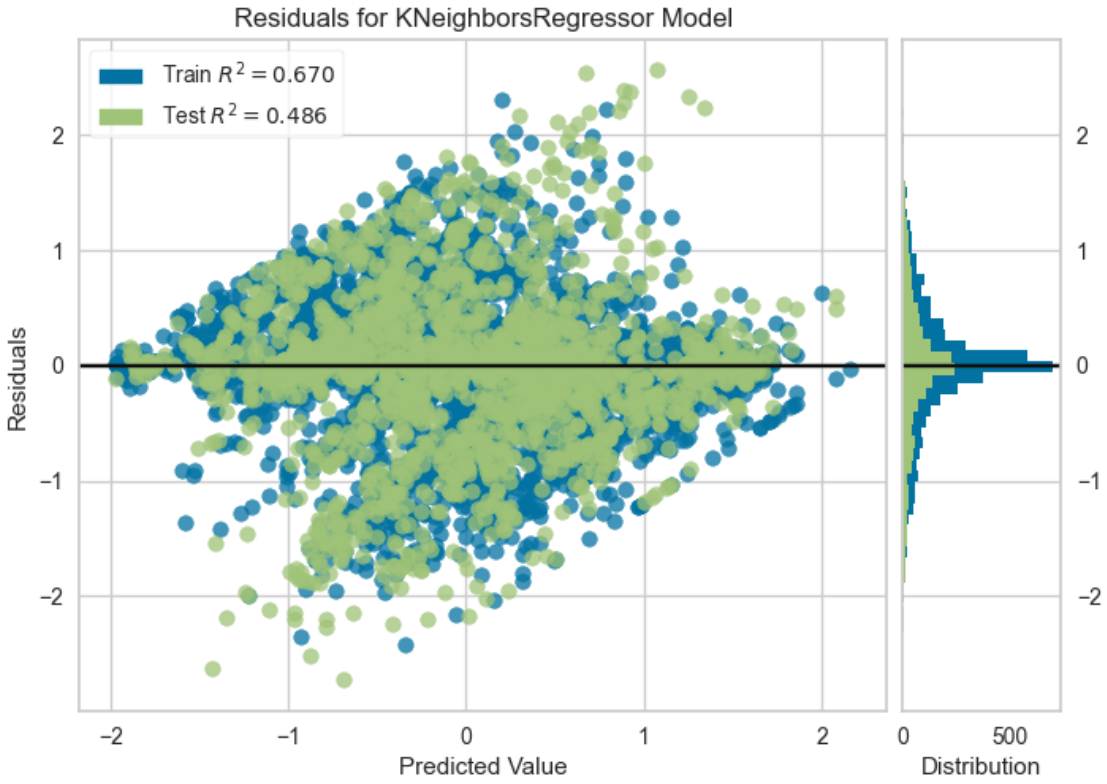
```
Mean Absolute Percentage Error(MAPE): 5.203633425717179
Root Mean Squared Error(RMSE): 1.1235222151525137
R2 Score: -0.23312175640425492
```

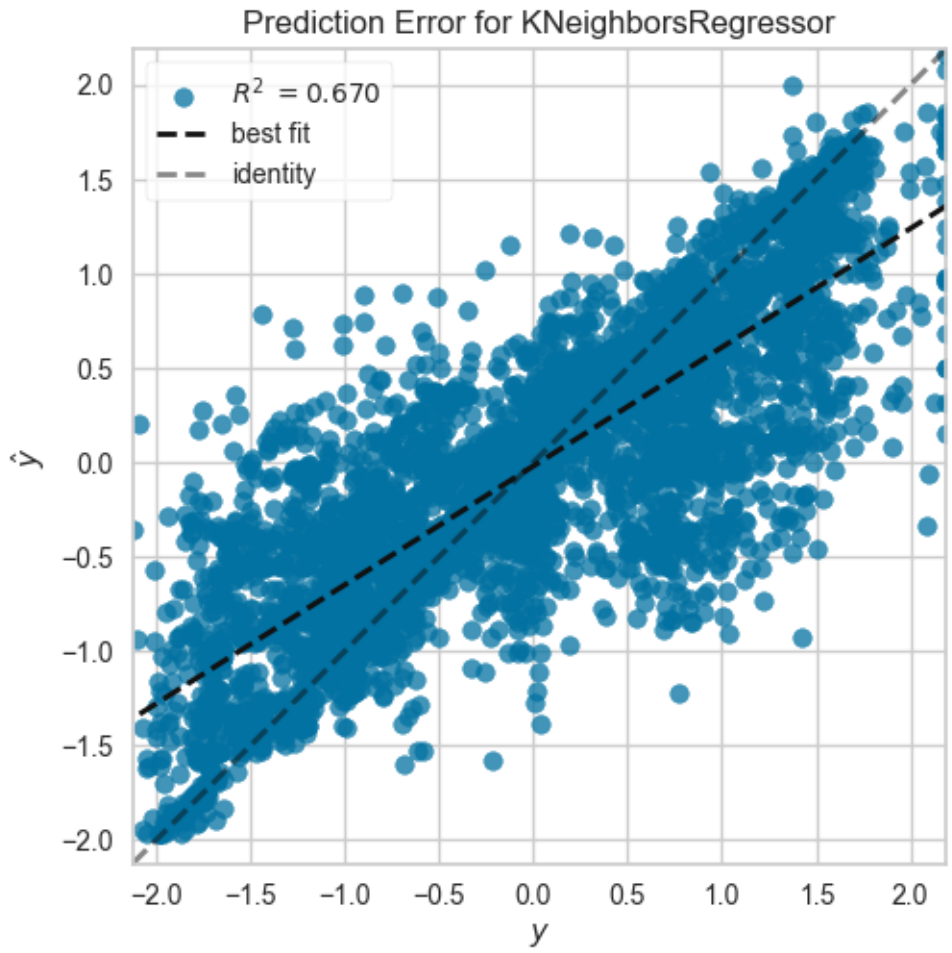

Residuals for PassiveAggressiveRegressor Model
Train $R^2 = -0.231$
Test $R^2 = -0.233$
Residuals
Predicted Value
Distribution
3
2
1
0
−1
−2
−3
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
0
200



[71]:
```
train_and_evaluate_model(KNeighborsRegressor())
```

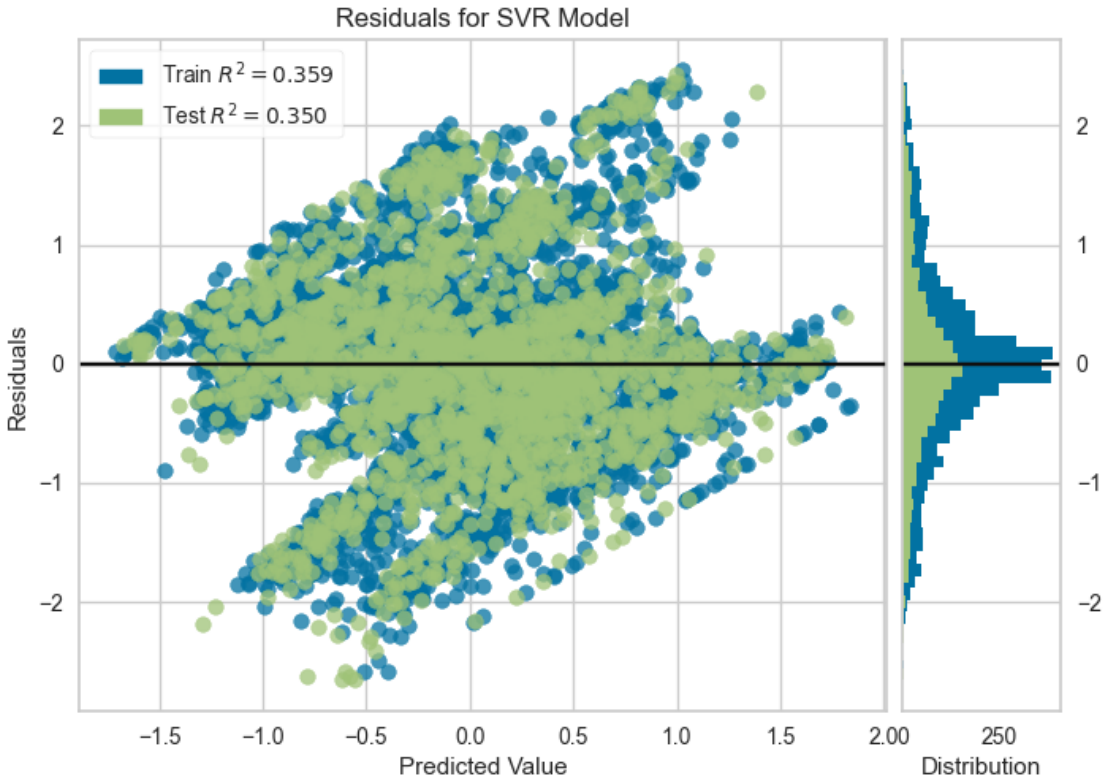
```
Mean Absolute Percentage Error(MAPE): 1.8324259921256374
Root Mean Squared Error(RMSE): 0.725657170052076
R2 Score: 0.485594487792688
```

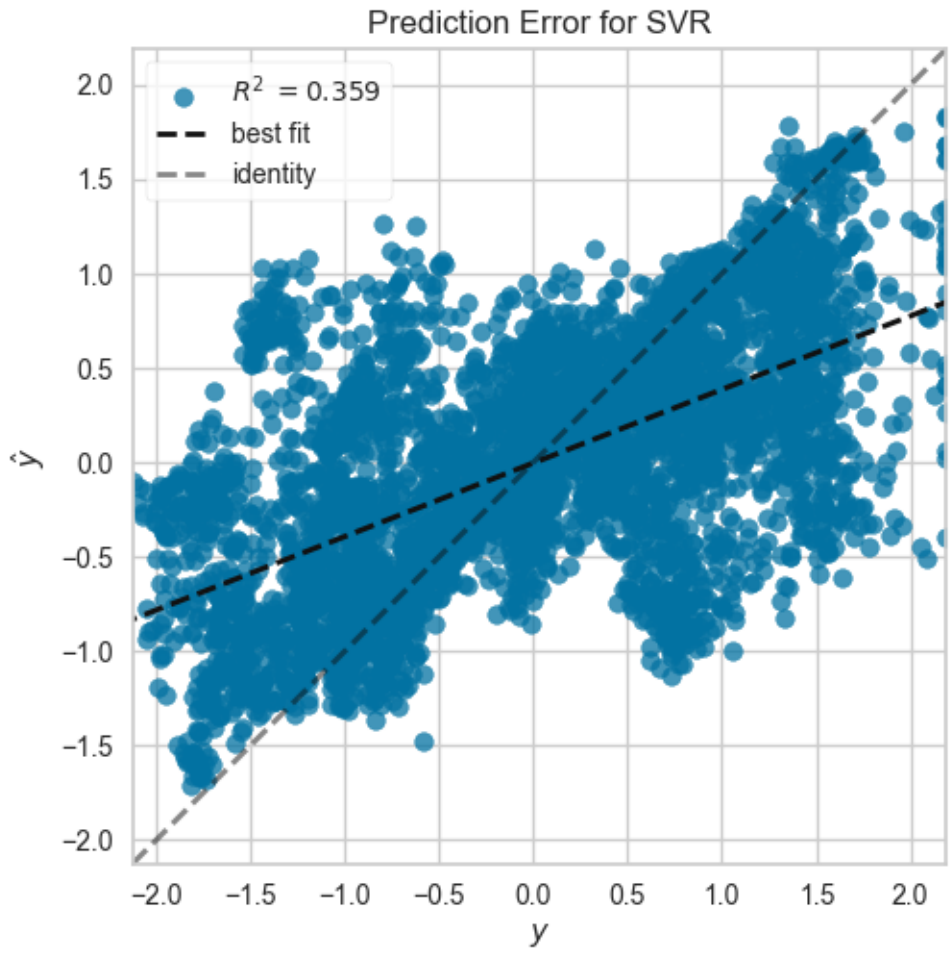

Residuals for KNeighborsRegressor Model
Train $R^2 = 0.670$
Test $R^2 = 0.486$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−2
−1
0
1
2
0
500



[72]:
```
train_and_evaluate_model(SVR())
```

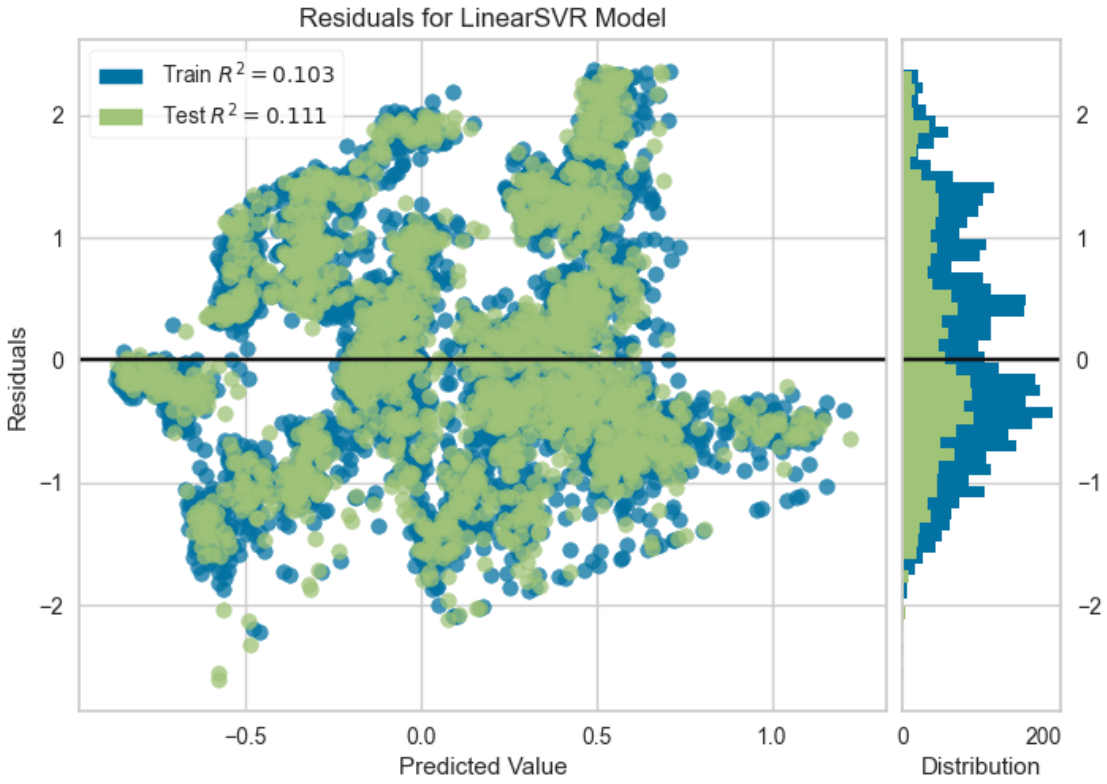
```
Mean Absolute Percentage Error(MAPE): 1.7786440447853233
Root Mean Squared Error(RMSE): 0.8159376019038274
R2 Score: 0.3496359488709865
```

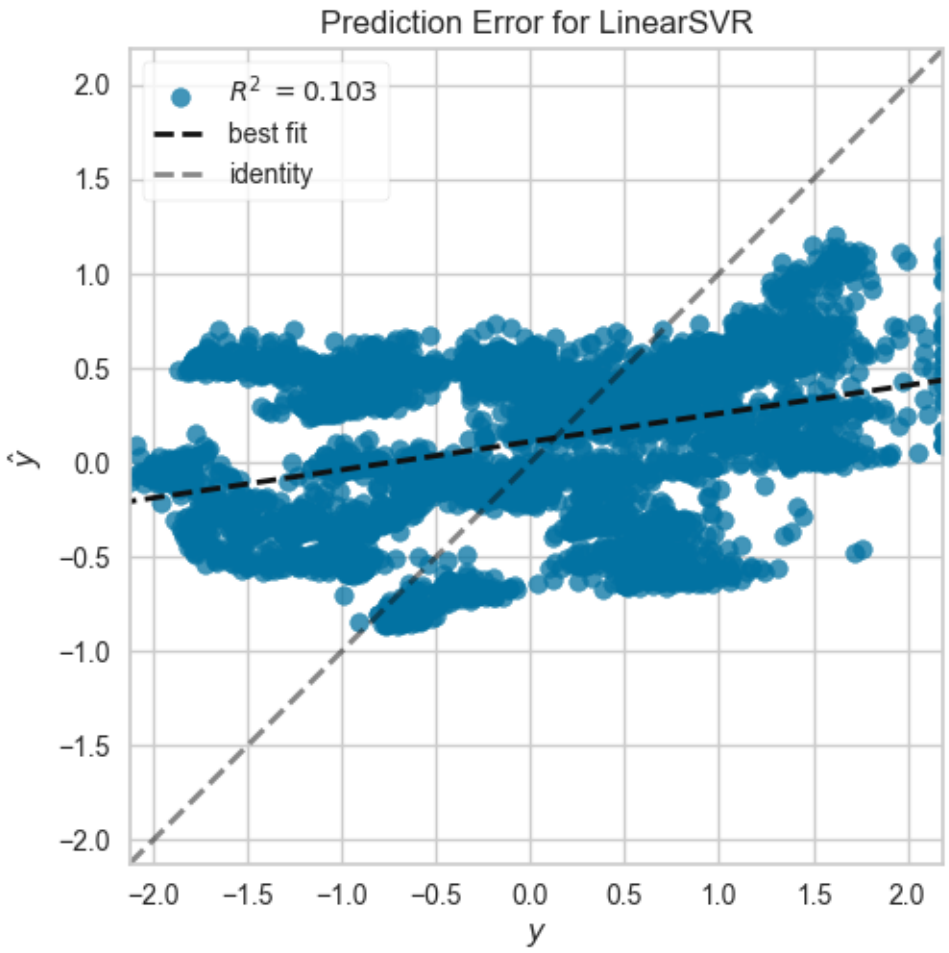

Residuals for SVR Model
Train $R^2 = 0.359$
Test $R^2 = 0.350$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.00
250



[73]:
```
train_and_evaluate_model(LinearSVR())
```

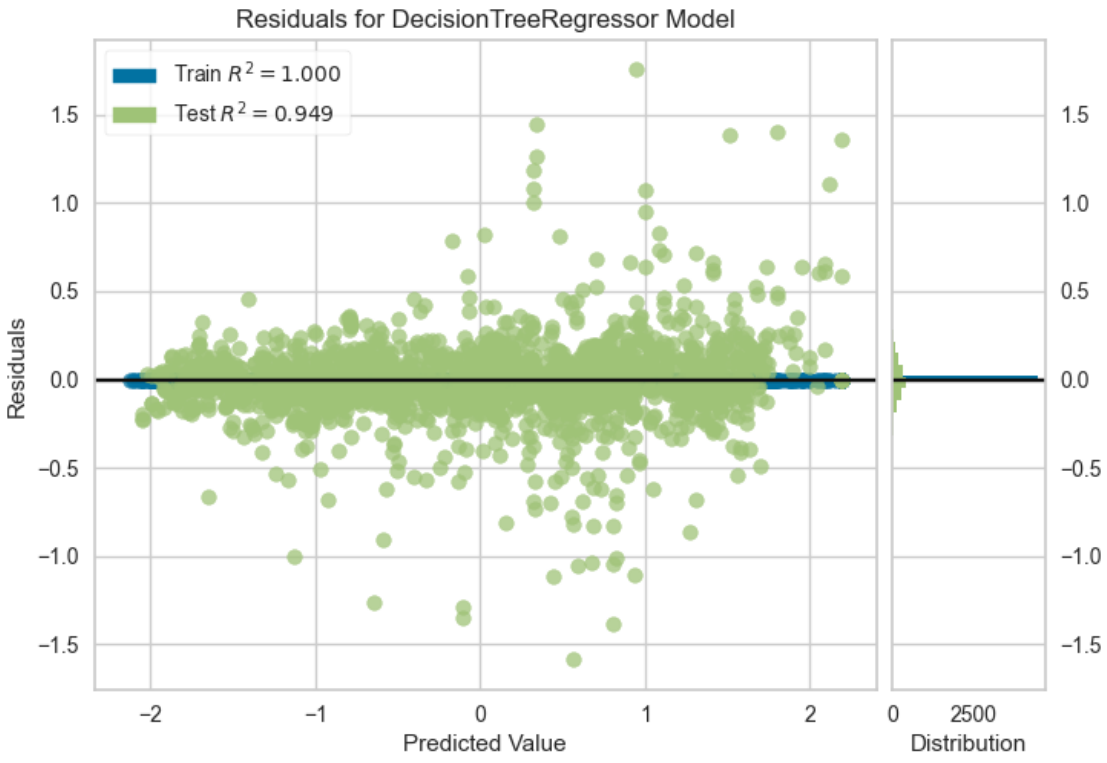
```
Mean Absolute Percentage Error(MAPE): 2.058692310166648
Root Mean Squared Error(RMSE): 0.9537043048035928
R2 Score: 0.11147404624017132
```

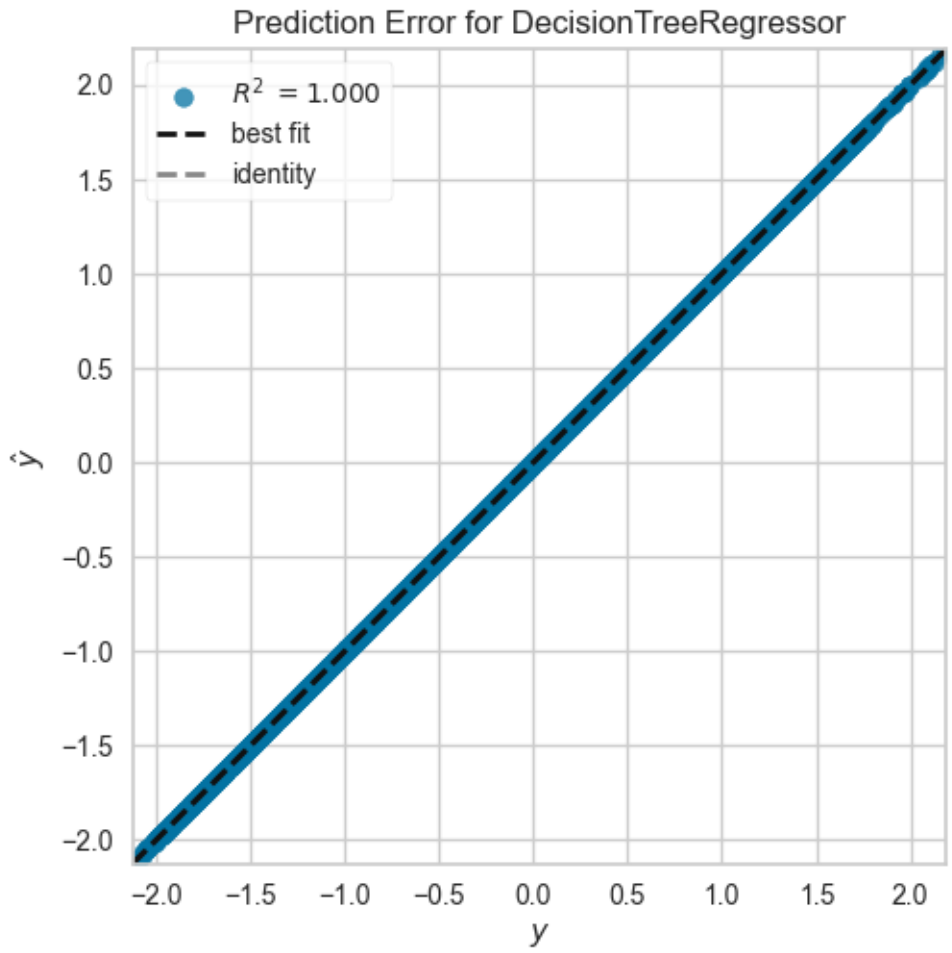

Residuals for LinearSVR Model
Train $R^2 = 0.103$
Test $R^2 = 0.111$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−0.5
0.0
0.5
1.0
0
200



[74]:
```
train_and_evaluate_model(DecisionTreeRegressor())
```

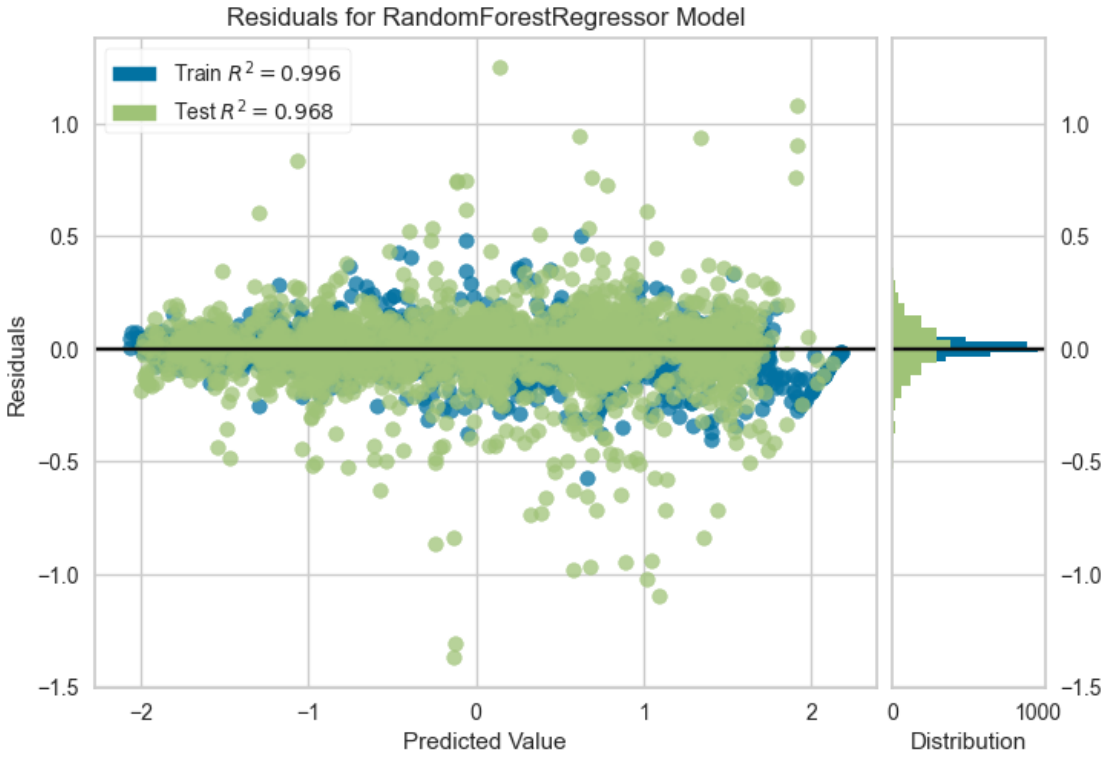
```
Mean Absolute Percentage Error(MAPE): 0.6886173929889536
Root Mean Squared Error(RMSE): 0.22928363764923226
R2 Score: 0.9486442875189682
```

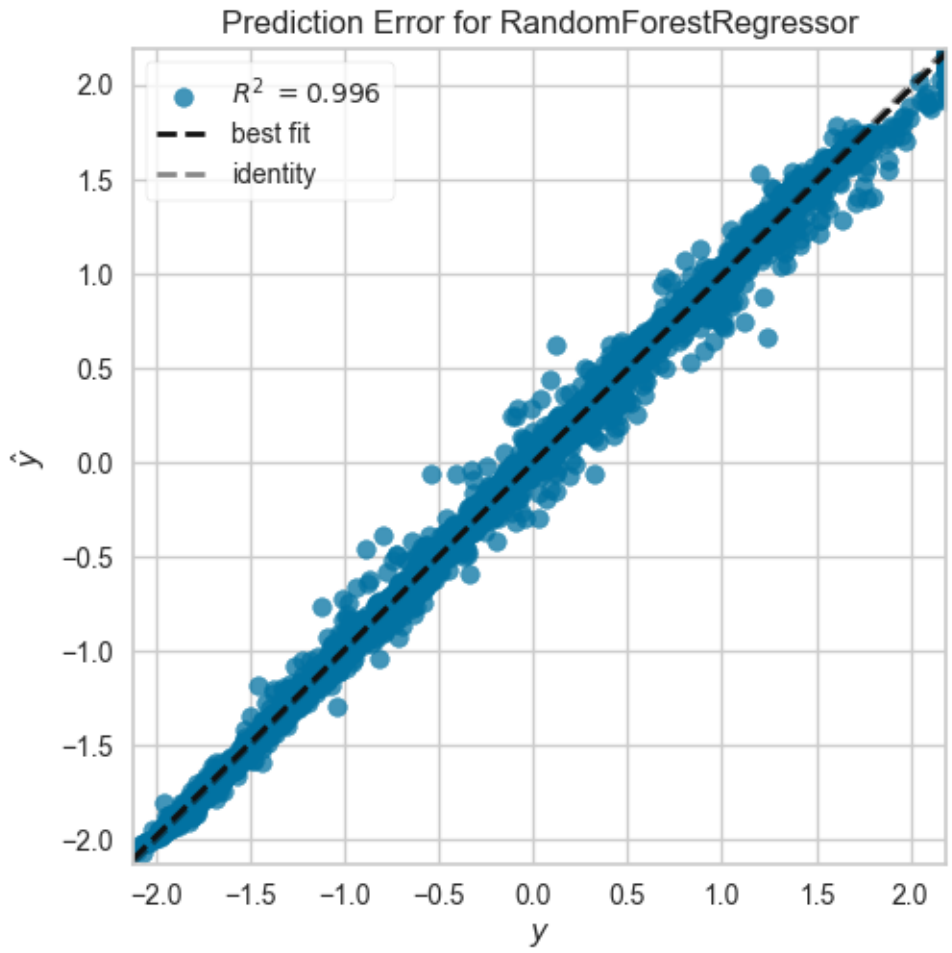

Residuals for DecisionTreeRegressor Model
Train $R^2 = 1.000$
Test $R^2 = 0.949$
Residuals
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2
Predicted Value
0
2500
Distribution



[75]:
```
train_and_evaluate_model(RandomForestRegressor())
```

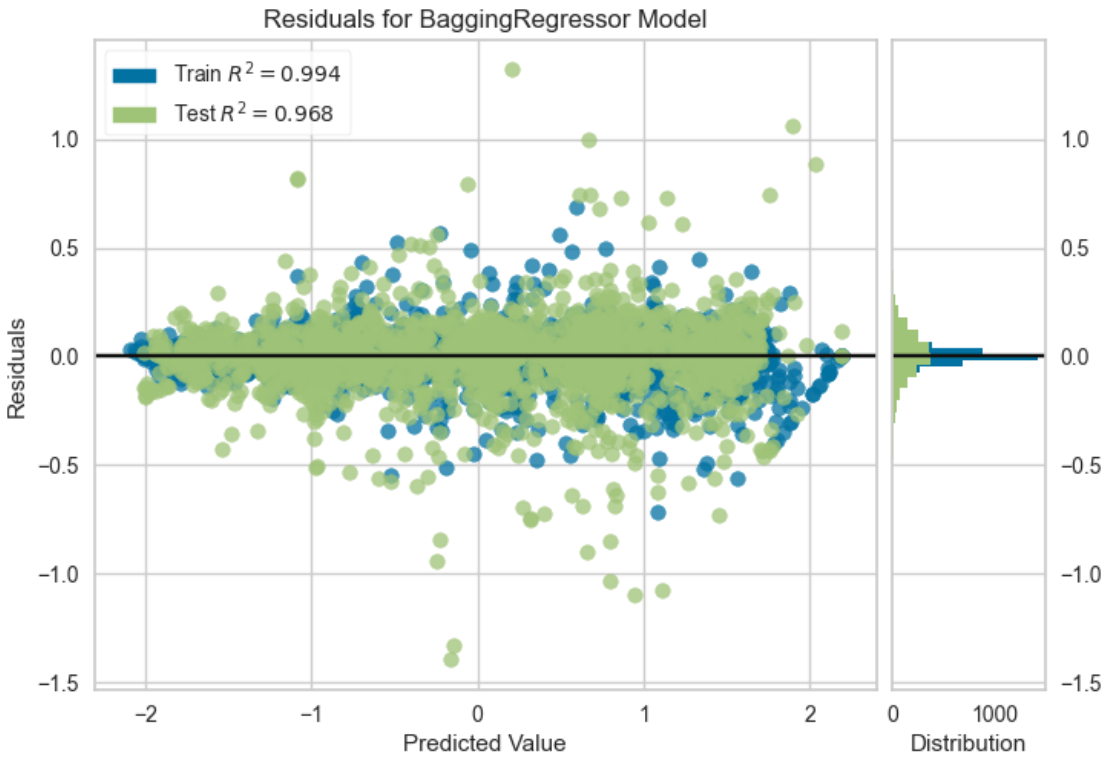
```
Mean Absolute Percentage Error(MAPE): 0.6316357845454231
Root Mean Squared Error(RMSE): 0.18088904075470633
R2 Score: 0.9680355568084091
```

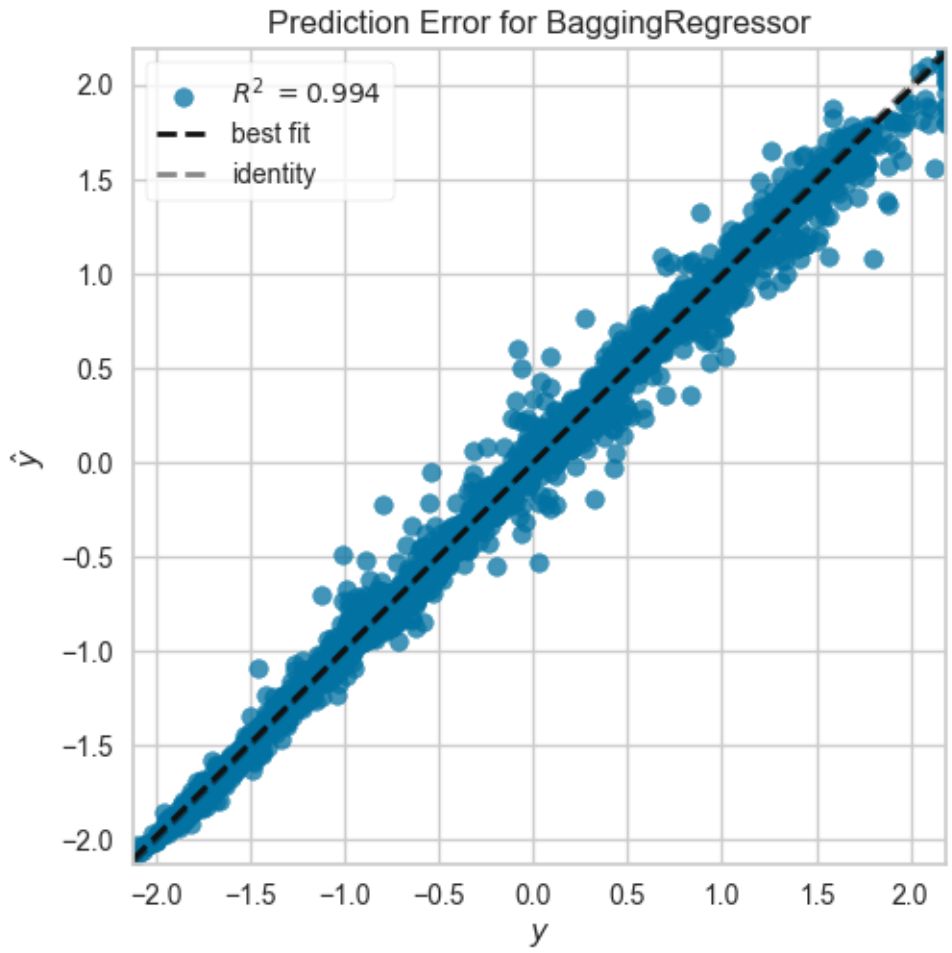

Residuals for RandomForestRegressor Model
Train $R^2 = 0.996$
Test $R^2 = 0.968$
Residuals
Predicted Value
Distribution
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2
0
1000



[76]:
```
train_and_evaluate_model(BaggingRegressor())
```

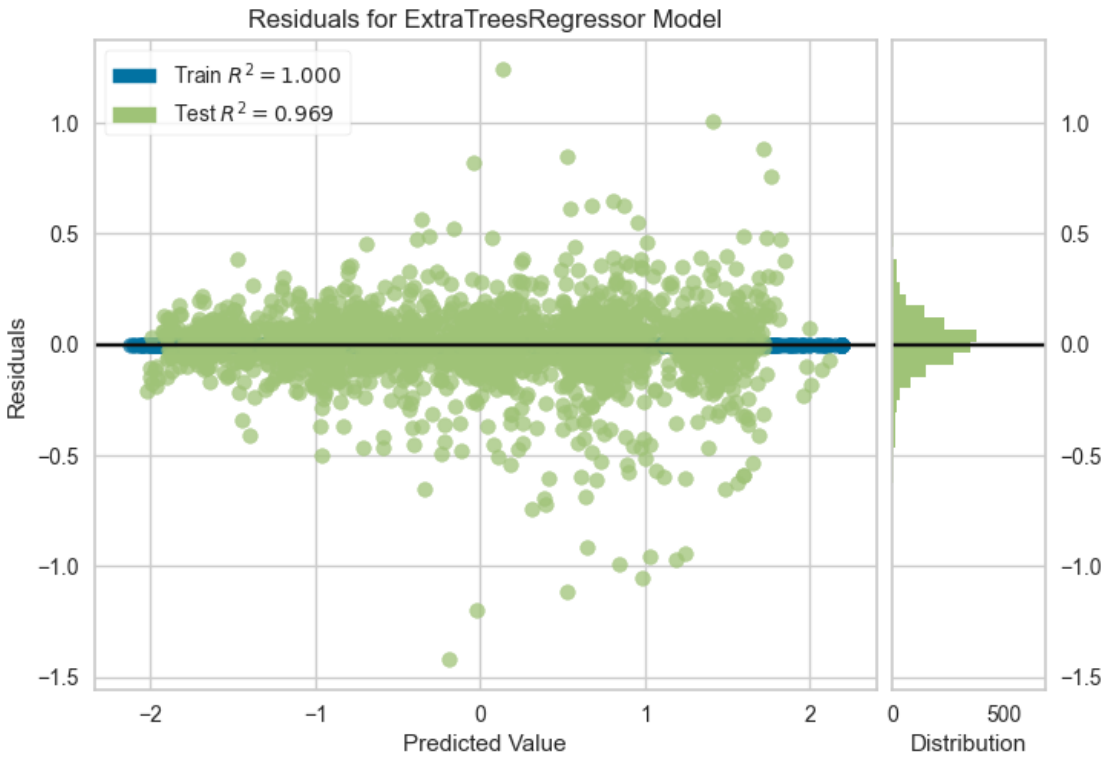
```
Mean Absolute Percentage Error(MAPE): 0.6288842943145447
Root Mean Squared Error(RMSE): 0.18181709881010605
R2 Score: 0.9677067258609293
```

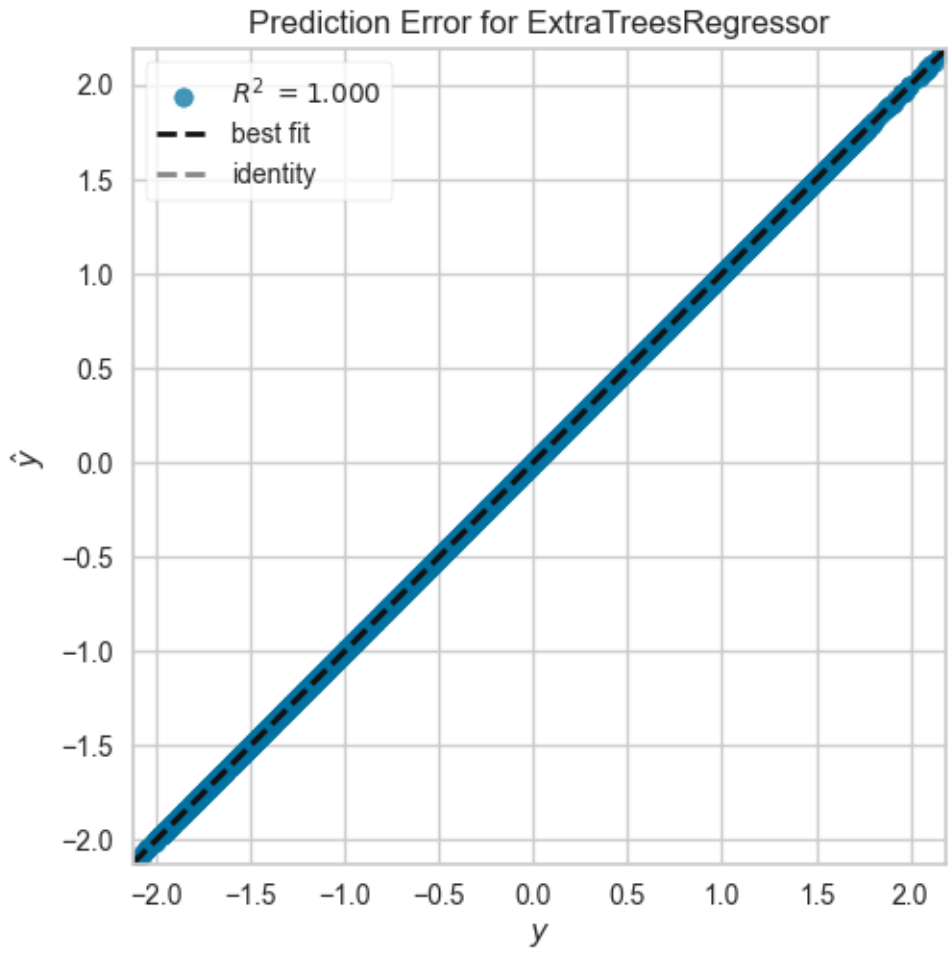

Residuals for BaggingRegressor Model
Train $R^2 = 0.994$
Test $R^2 = 0.968$
Residuals
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2
Predicted Value
0
1000
Distribution



[77]:
```
train_and_evaluate_model(ExtraTreesRegressor())
```

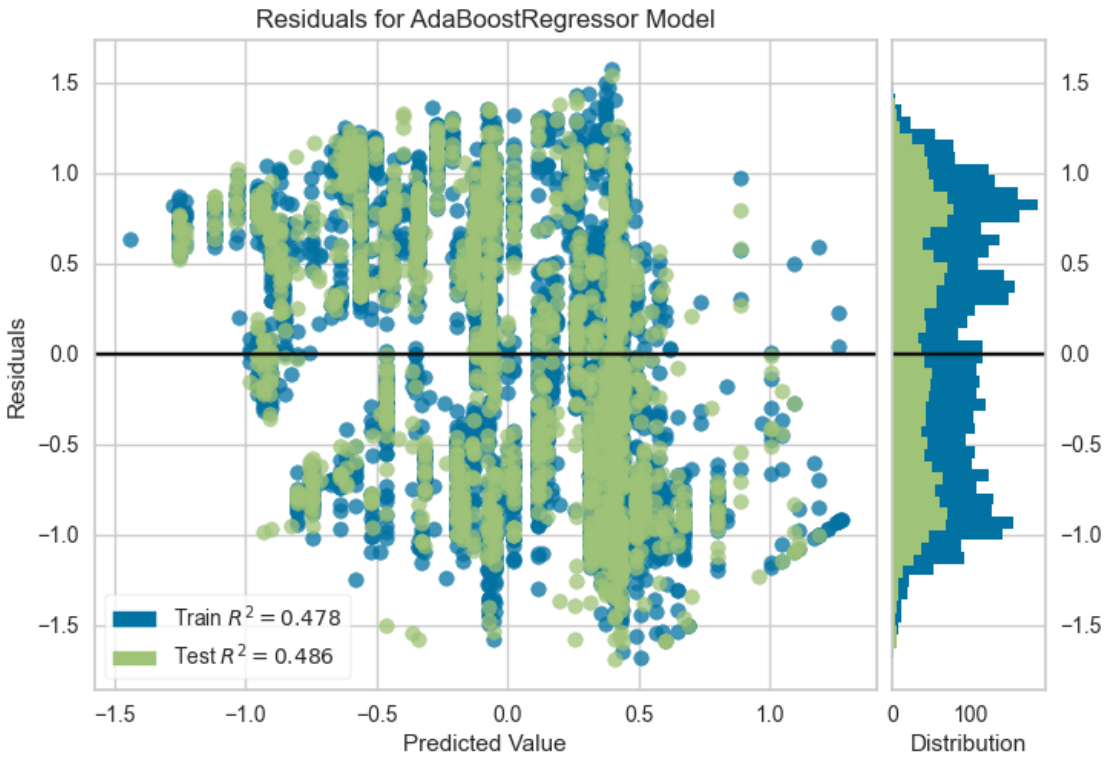
```
Mean Absolute Percentage Error(MAPE): 0.5968460838198617
Root Mean Squared Error(RMSE): 0.17849433508980103
R2 Score: 0.9688762794770702
```


Residuals for ExtraTreesRegressor Model
Train $R^2 = 1.000$
Test $R^2 = 0.969$
Residuals
Predicted Value
Distribution
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2
0
500



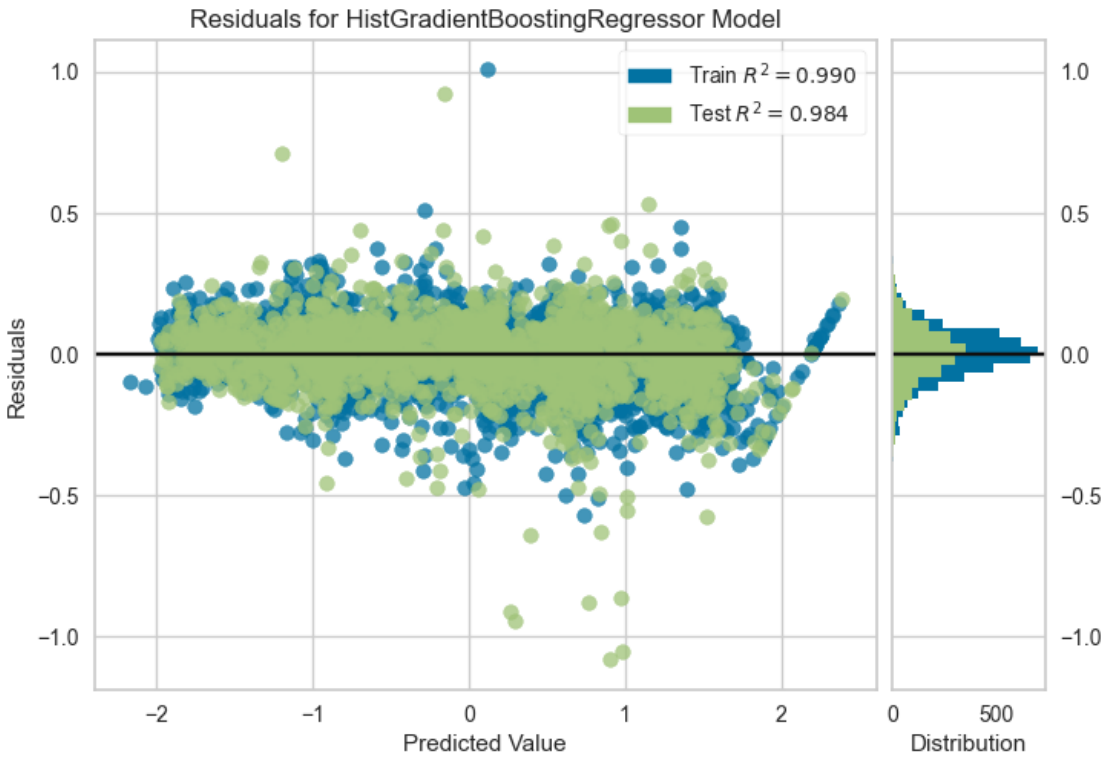
[78]:
```
train_and_evaluate_model(AdaBoostRegressor())
```

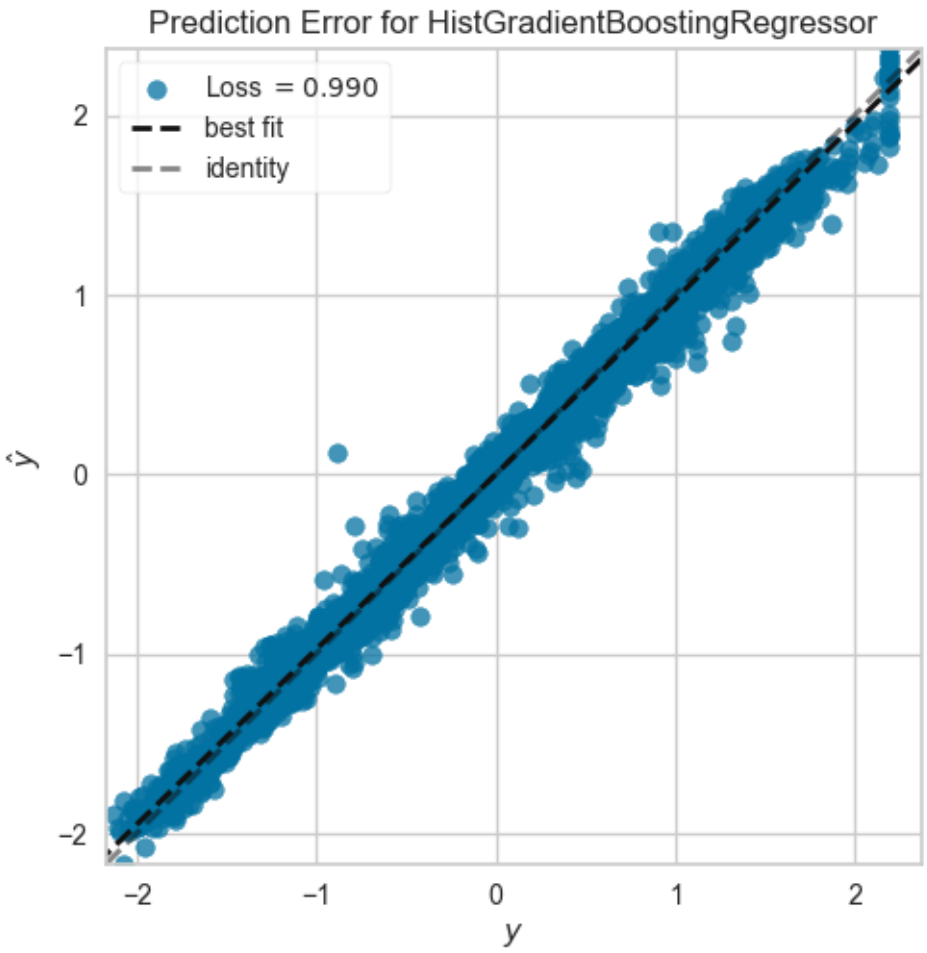
```
Mean Absolute Percentage Error(MAPE): 2.57532882324867
Root Mean Squared Error(RMSE): 0.7250431693889414
R2 Score: 0.4864646278959307
```


Residuals for AdaBoostRegressor Model
Residuals
Predicted Value
Distribution
Train $R^2 = 0.478$
Test $R^2 = 0.486$
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
100

[79]:
```
train_and_evaluate_model(HistGradientBoostingRegressor())
```

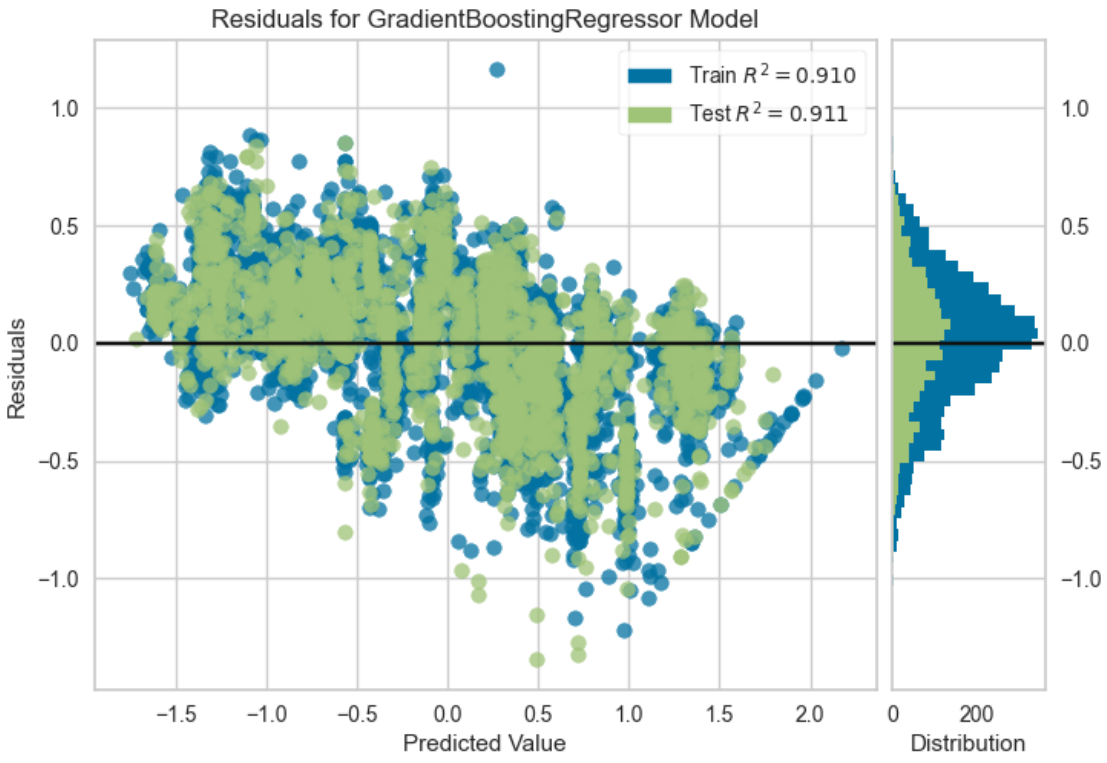
```
Mean Absolute Percentage Error(MAPE): 0.51087336067734
Root Mean Squared Error(RMSE): 0.1294334736270051
R2 Score: 0.9836342525411399
```

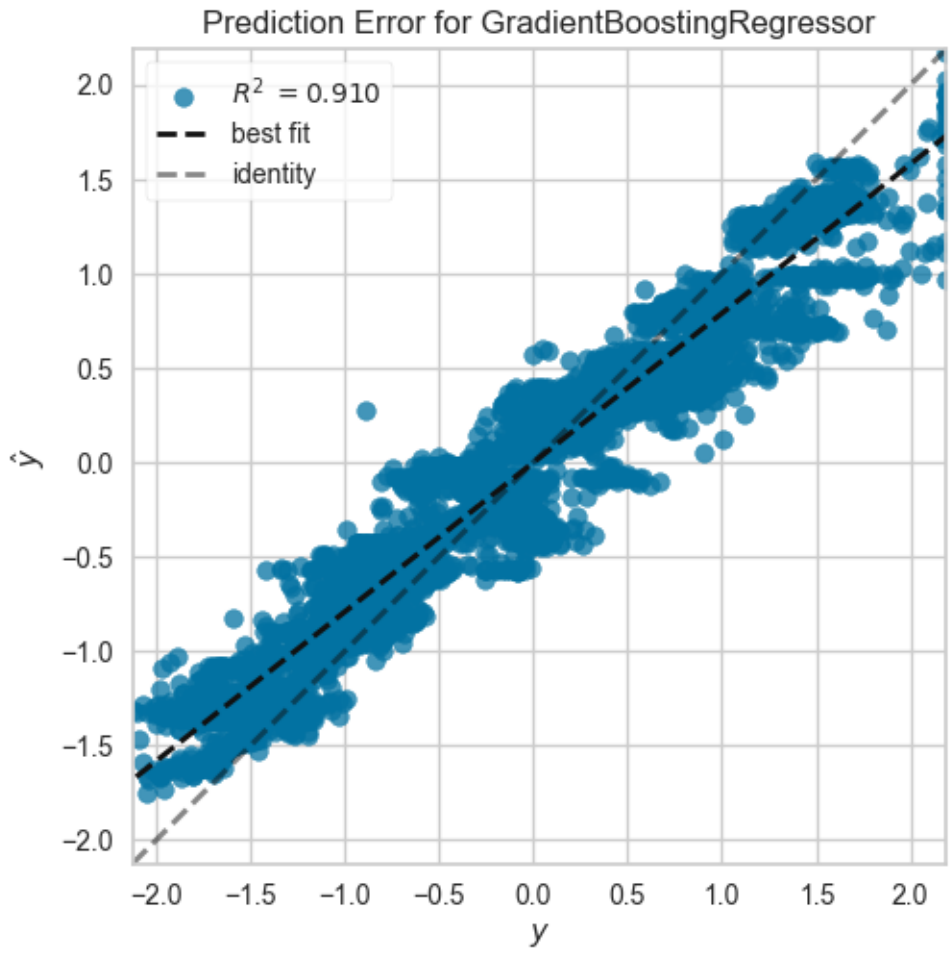

Residuals for HistGradientBoostingRegressor Model
Train $R^2 = 0.990$
Test $R^2 = 0.984$
Residuals
Predicted Value
Distribution
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2
0
500



[80]:
```
train_and_evaluate_model(GradientBoostingRegressor())
```

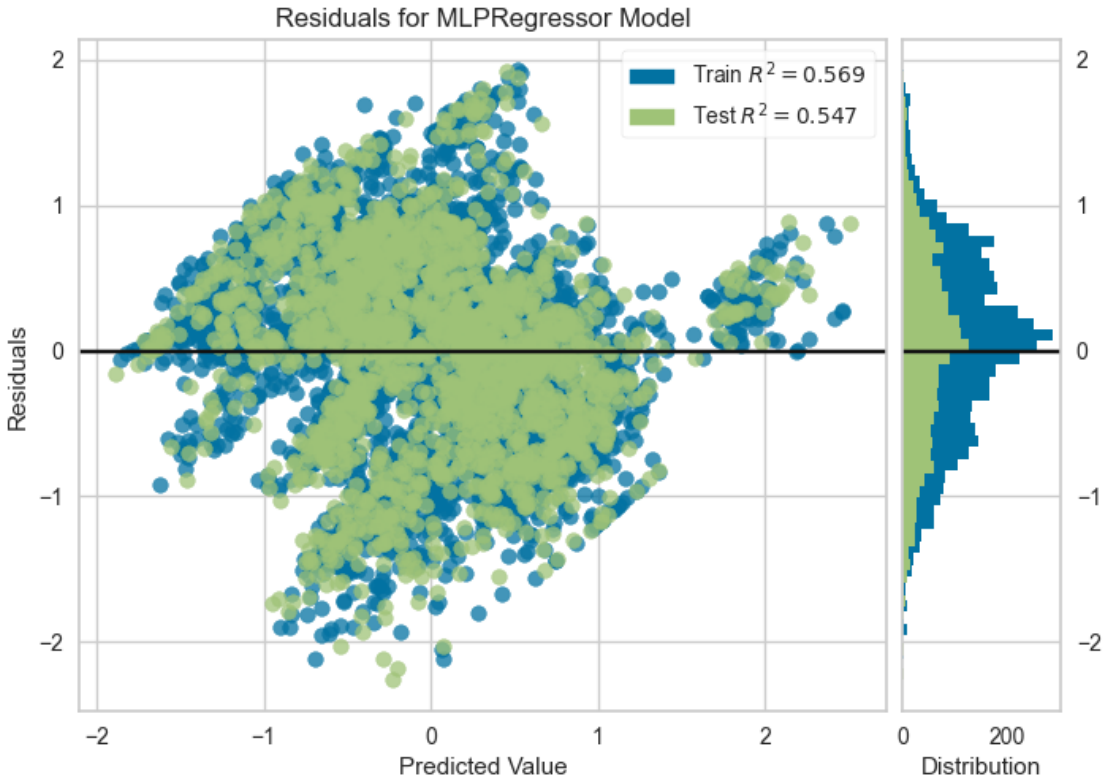
```
Mean Absolute Percentage Error(MAPE): 1.5252582037997267
Root Mean Squared Error(RMSE): 0.30209150905999316
R2 Score: 0.910850344465775
```

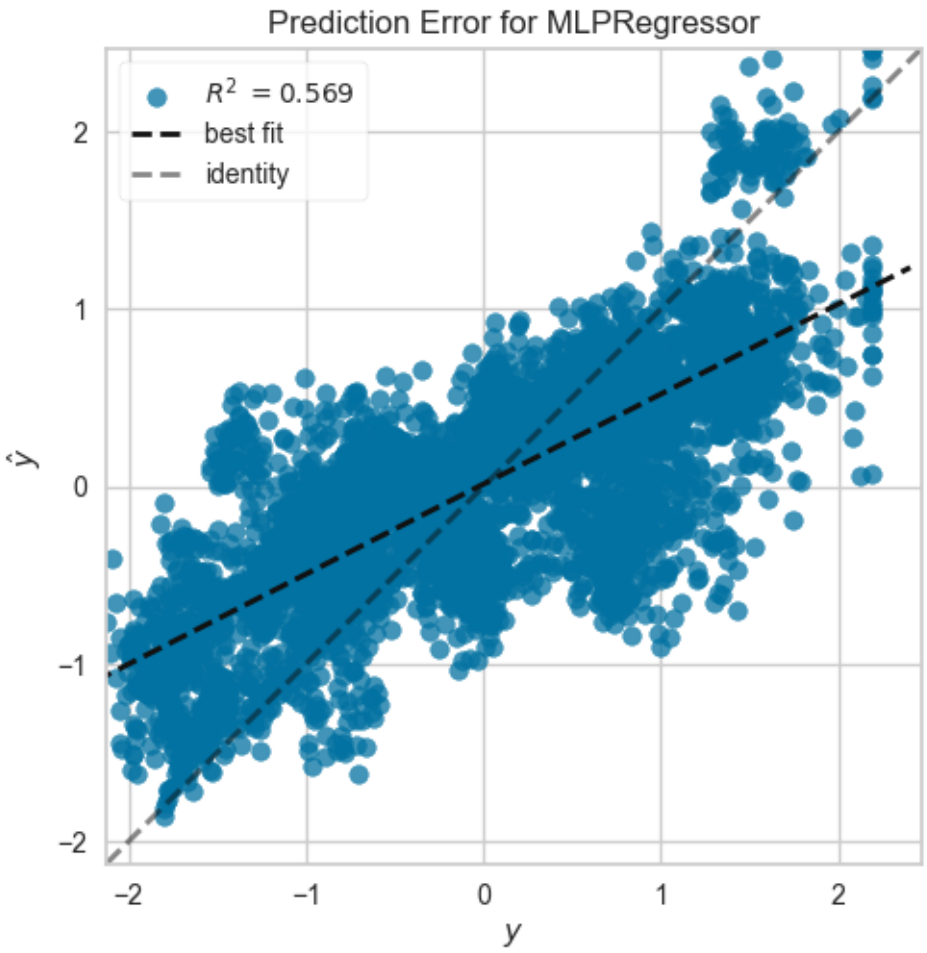

Residuals for GradientBoostingRegressor Model
Train $R^2 = 0.910$
Test $R^2 = 0.911$
Residuals
Predicted Value
Distribution
1.0
0.5
0.0
−0.5
−1.0
−1.5
2.0
0
200



[81]:
```
train_and_evaluate_model(MLPRegressor())
```

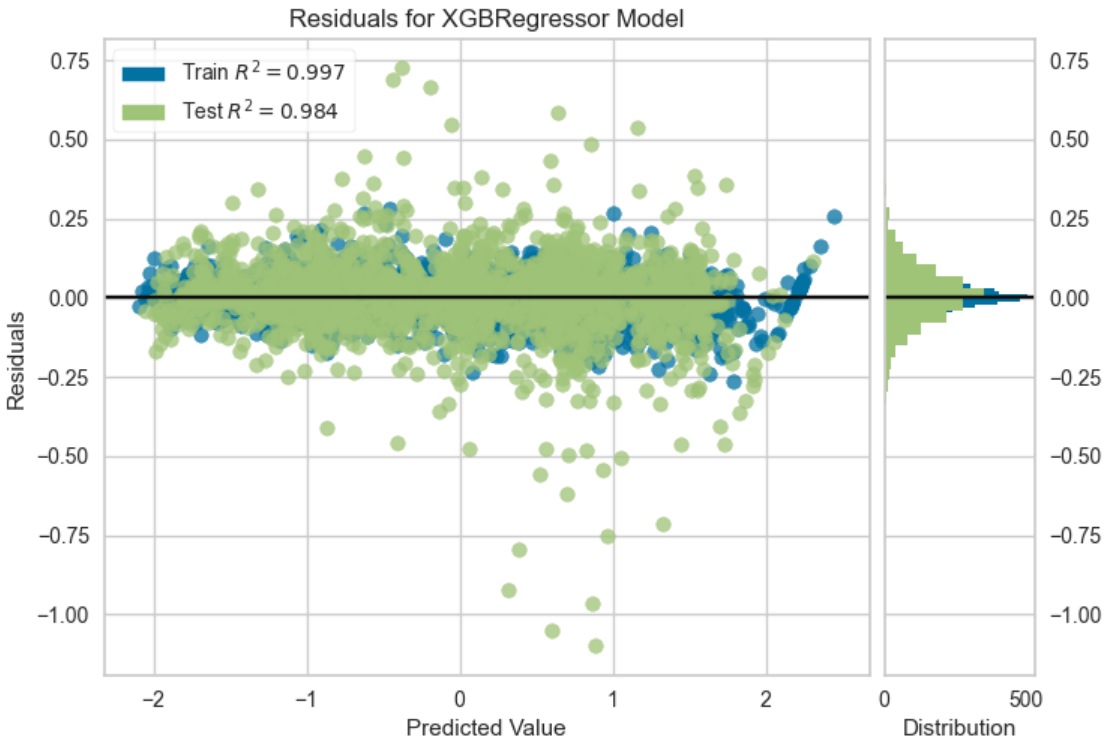
```
Mean Absolute Percentage Error(MAPE): 2.302538030869336
Root Mean Squared Error(RMSE): 0.6807573916414449
R2 Score: 0.5472824159249126
```

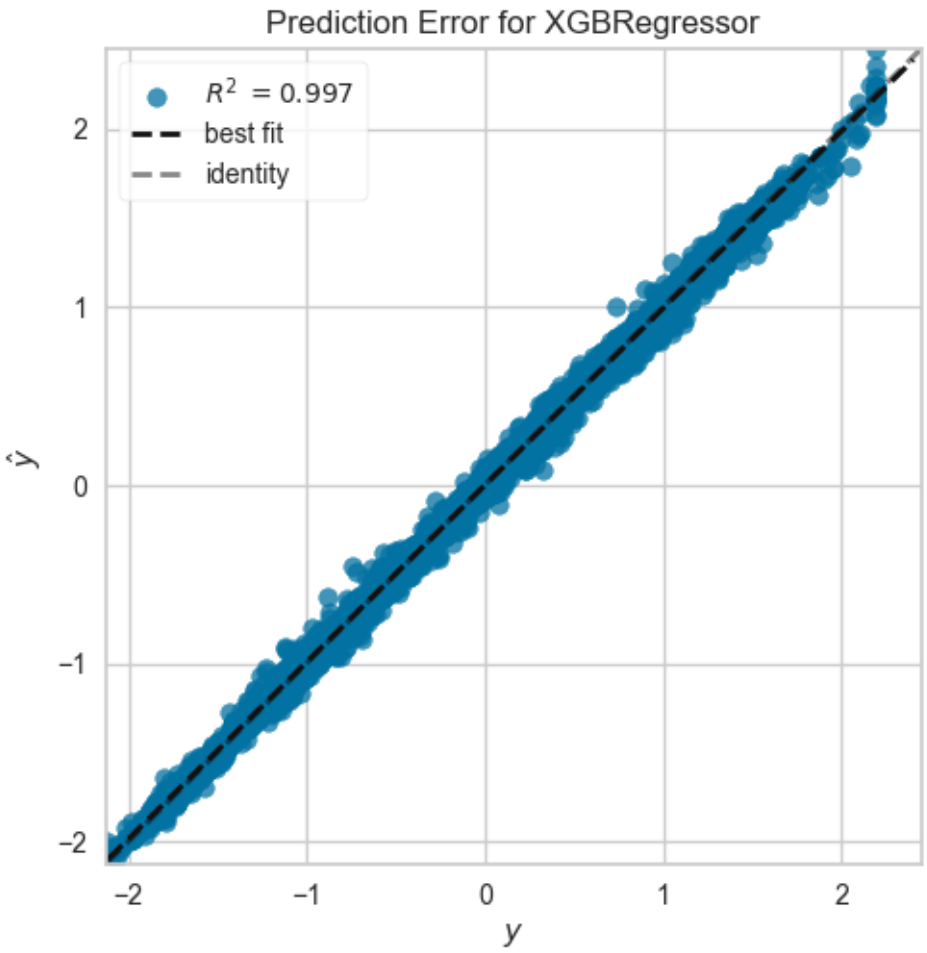

Residuals for MLPRegressor Model
Train $R^2 = 0.569$
Test $R^2 = 0.547$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−2
−1
0
1
2
0
200



[82]:
```
train_and_evaluate_model(XGBRegressor())
```

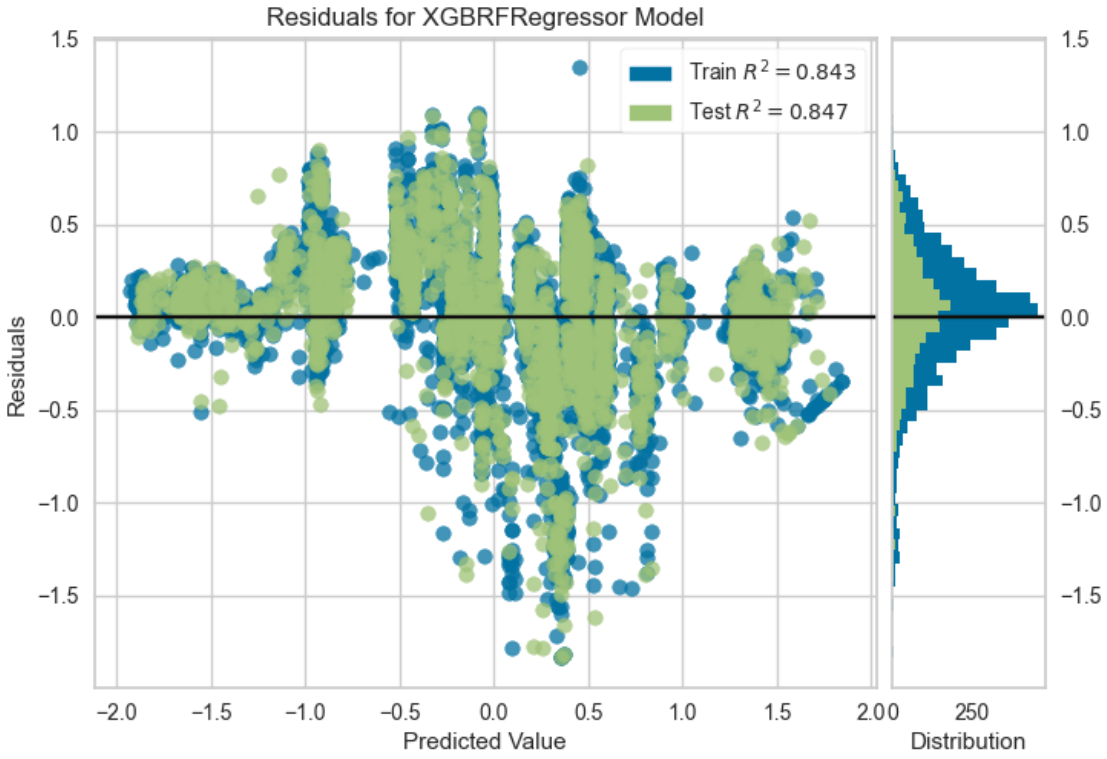
```
Mean Absolute Percentage Error(MAPE): 0.4484239317276413
Root Mean Squared Error(RMSE): 0.1279829255613287
R2 Score: 0.9839990156994279
```

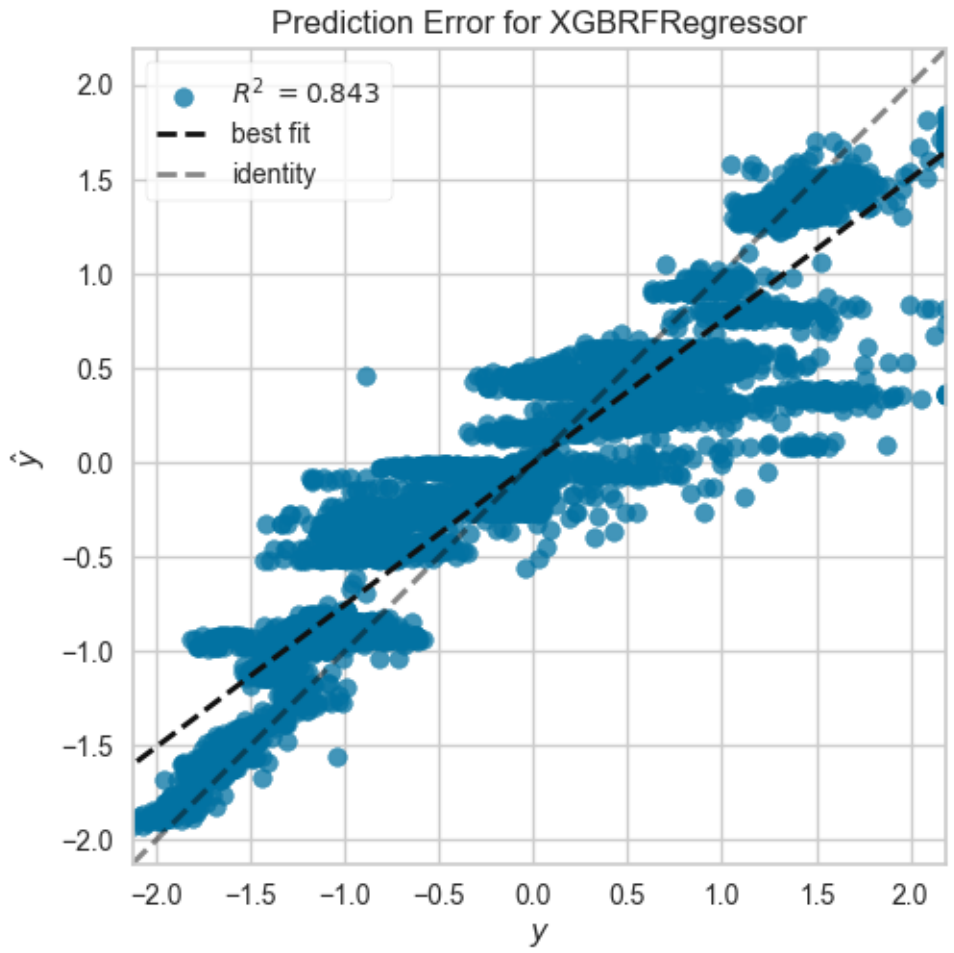

Residuals for XGBRegressor Model
Train $R^2 = 0.997$
Test $R^2 = 0.984$
Residuals
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−2
−1
0
1
2
Predicted Value
0
500
Distribution



[83]:
```
train_and_evaluate_model(XGBRFRegressor())
```

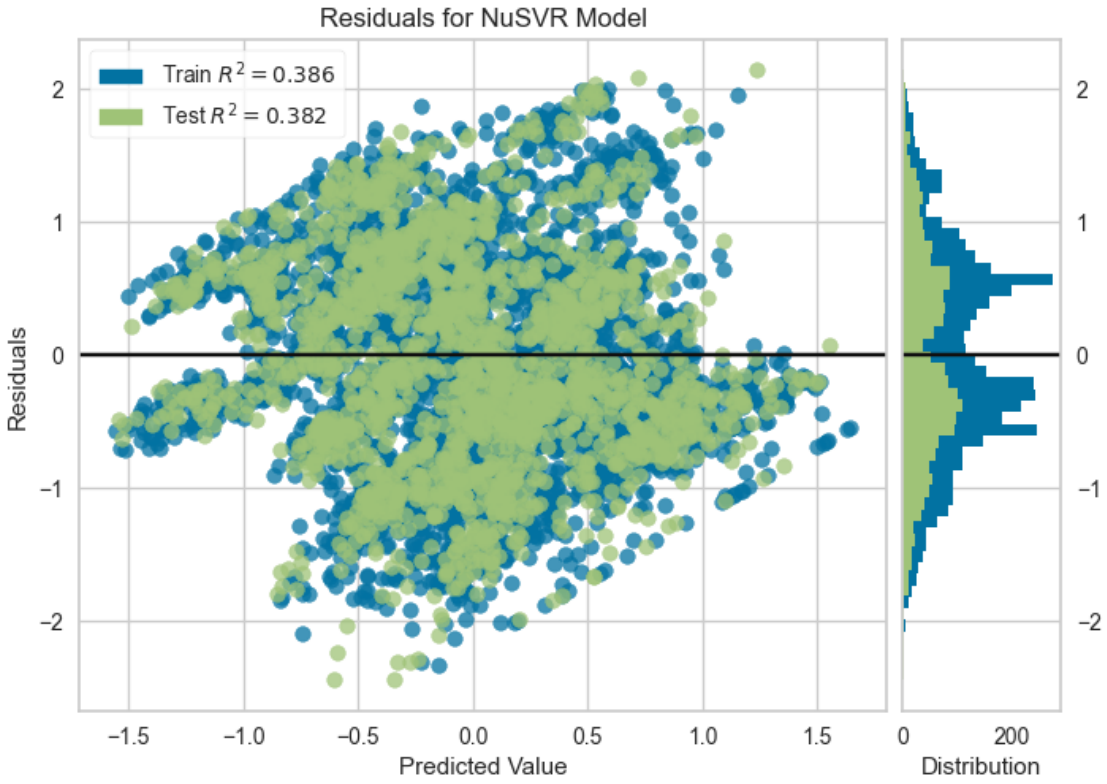
```
Mean Absolute Percentage Error(MAPE): 1.4958698963663275
Root Mean Squared Error(RMSE): 0.3956003717003603
R2 Score: 0.8471181201865408
```

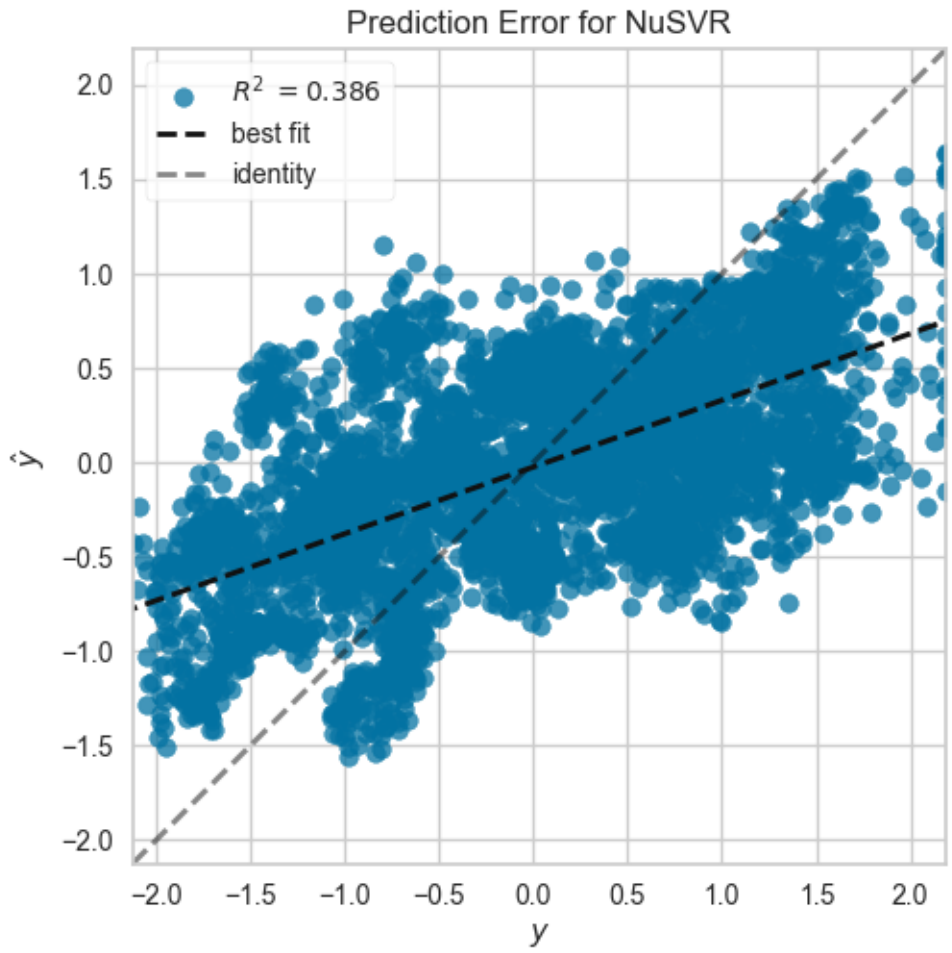

Residuals for XGBRFRegressor Model
Train $R^2 = 0.843$
Test $R^2 = 0.847$
Residuals
Predicted Value
Distribution
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.00
250



[84]:
```
train_and_evaluate_model(NuSVR())
```

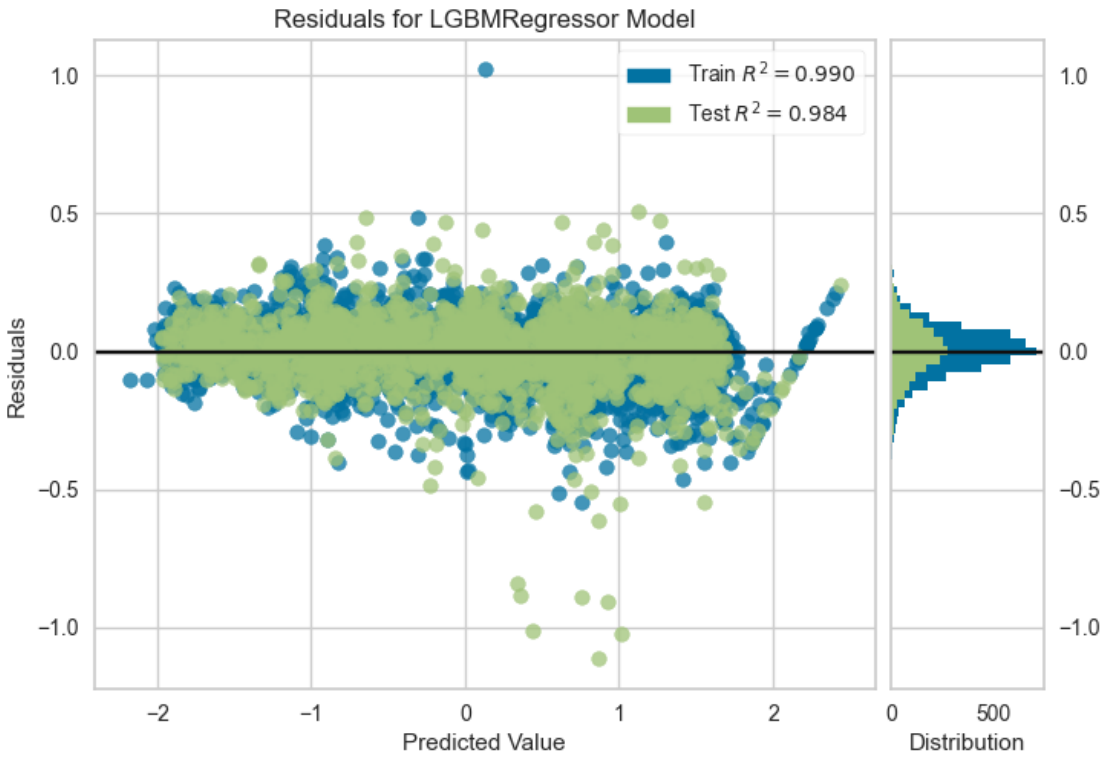
```
Mean Absolute Percentage Error(MAPE): 2.7676772849788933
Root Mean Squared Error(RMSE): 0.7953092639783247
R2 Score: 0.3821049519963209
```

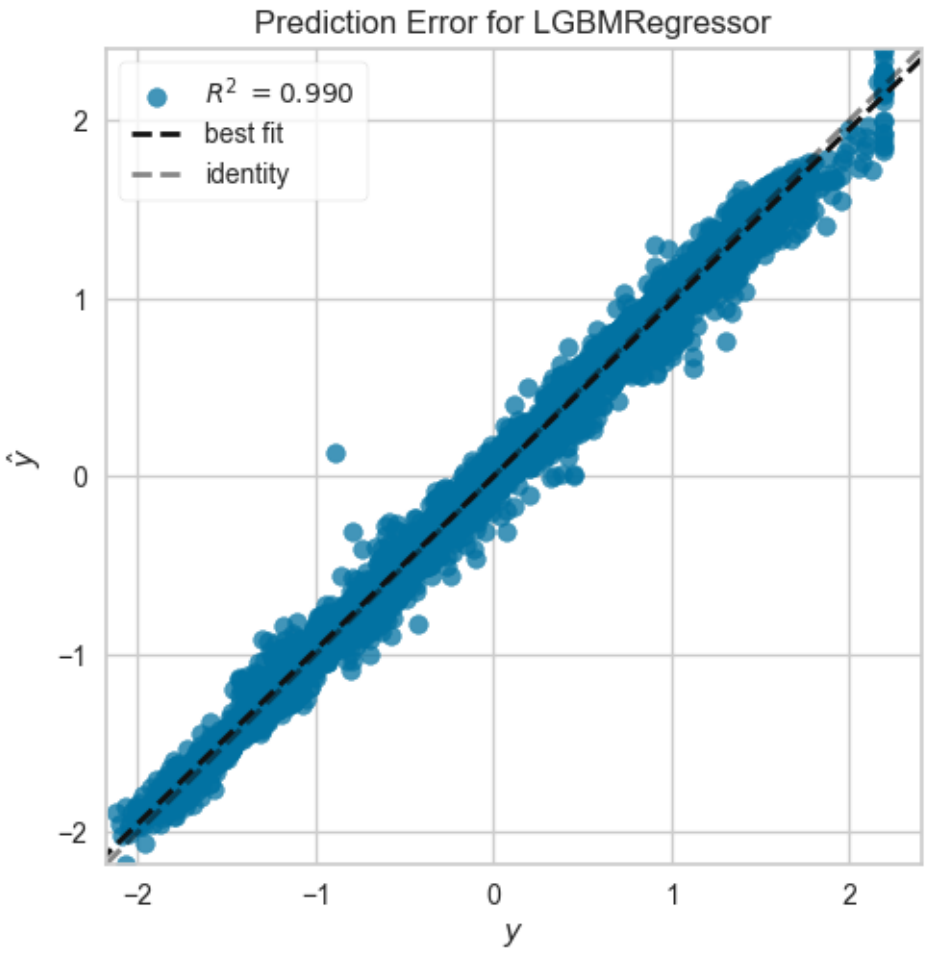

Residuals for NuSVR Model
Train $R^2 = 0.386$
Test $R^2 = 0.382$
Residuals
Predicted Value
Distribution
2
1
0
−1
−2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
0
200



[85]:
```
train_and_evaluate_model(LGBMRegressor())
```

```
Mean Absolute Percentage Error(MAPE): 0.4836466164864434
Root Mean Squared Error(RMSE): 0.1289760326076023
R2 Score: 0.9837497270767015
```


Residuals for LGBMRegressor Model
Train $R^2 = 0.990$
Test $R^2 = 0.984$
Residuals
Predicted Value
Distribution
1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1
2
0
500



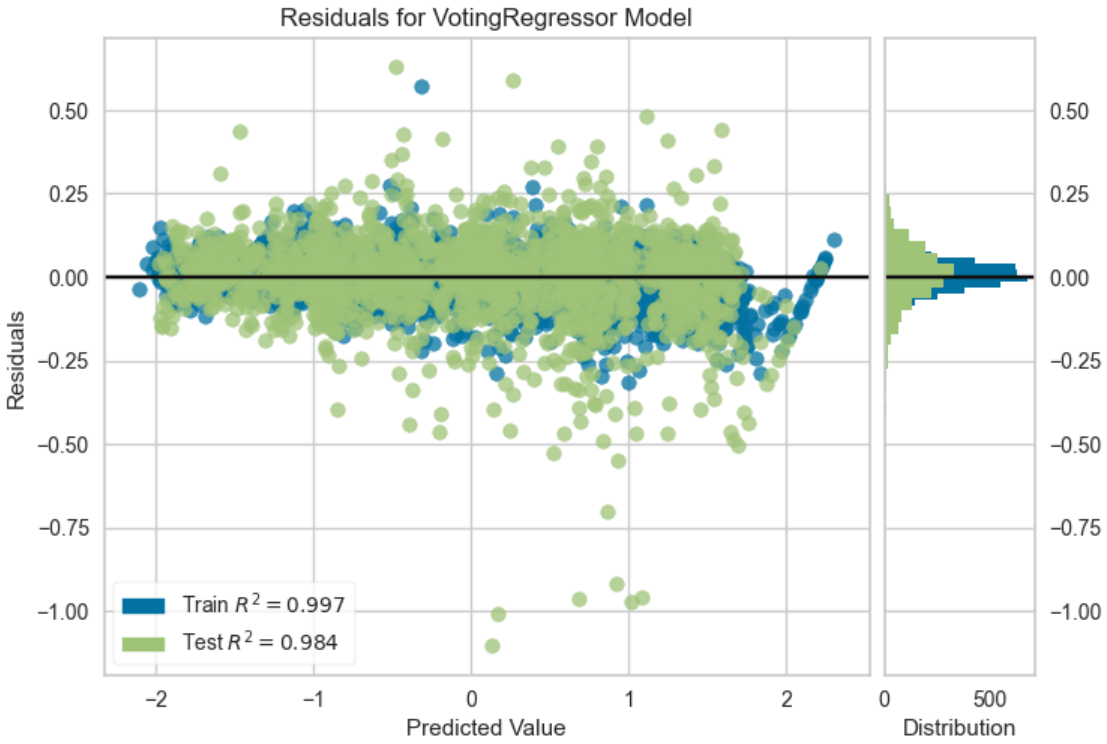
[86]:
```
train_and_evaluate_model(CatBoostRegressor(silent=True))
```

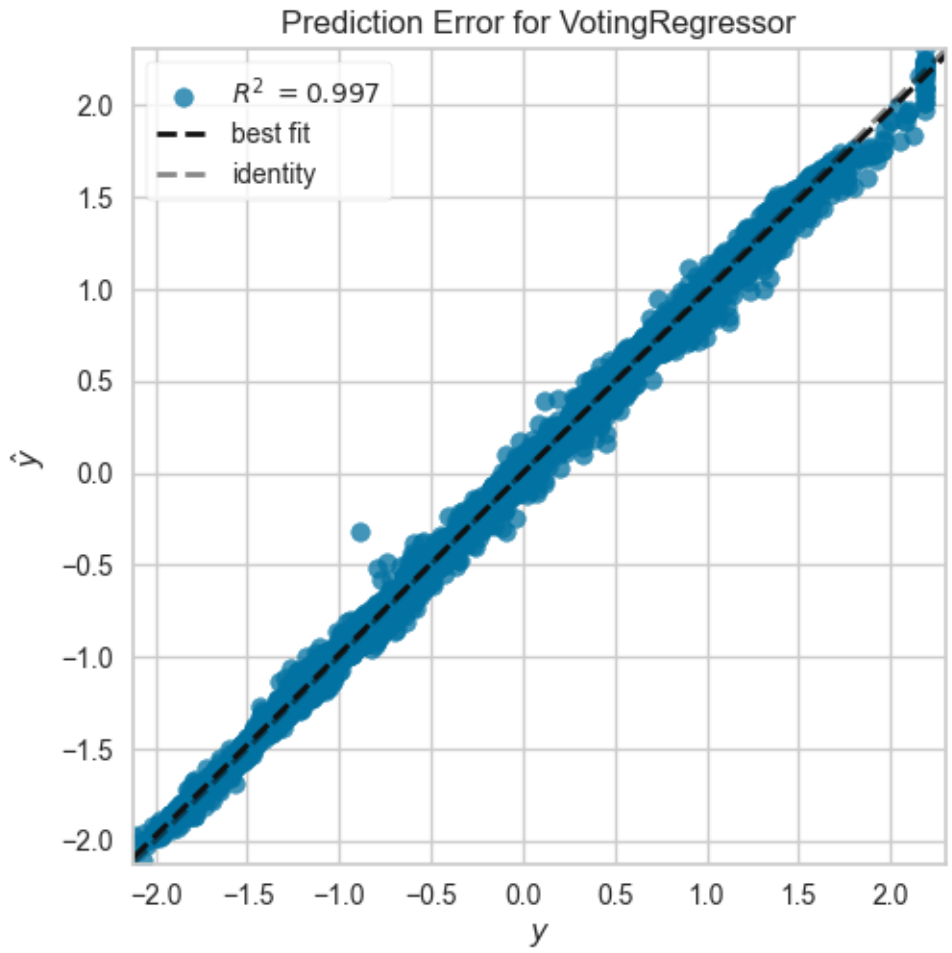
```
Mean Absolute Percentage Error(MAPE): 0.5341453587985909
Root Mean Squared Error(RMSE): 0.12290887063346671
R2 Score: 0.9852426258469904
```

[145]:
```
train_and_evaluate_model(VotingRegressor(estimators=[
    ('CAT',CatBoostRegressor(silent=True)),
    ('XGB',XGBRegressor()),
    ('LGBM',LGBMRegressor()),
    ('HGB',HistGradientBoostingRegressor()),
    ('RF',RandomForestRegressor()),
    ('BAG',BaggingRegressor()),
    ('ET',ExtraTreesRegressor())
]))
```

```
Mean Absolute Percentage Error(MAPE): 0.4845606374418606
Root Mean Squared Error(RMSE): 0.12994806863698197
```

```
R2 Score: 0.9835038616447688
```





0.7 Baseline Models Performance Comparison

[88]:
```
model_perfs = pd.DataFrame({'Model': models, 'MAPE': mape_scores, 'RMSE':␣
 ↪rmse_scores, 'R2': r2_scores})
model_perfs = model_perfs.sort_values('R2',ascending=False)
model_perfs
```

[88]:

	Model	MAPE	RMSE	\
25	<catboost.core.CatBoostRegressor object at 0x0…	0.534145	0.122909	
21	XGBRegressor(base_score=None, booster=None, ca…	0.448424	0.127983	
24	LGBMRegressor()	0.483647	0.128976	
18	HistGradientBoostingRegressor()	0.510873	0.129433	
26	VotingRegressor(estimators=[('CAT',\n …	0.483340	0.131280	
16	(ExtraTreeRegressor(random_state=1040630546), …	0.596846	0.178494	
14	(DecisionTreeRegressor(max_features=1.0, rando…	0.631636	0.180889	
15	(DecisionTreeRegressor(random_state=1012004194…	0.628884	0.181817	
13	DecisionTreeRegressor()	0.688617	0.229284	

19	([DecisionTreeRegressor(criterion='friedman_ms…	1.525258	0.302092
22	XGBRFRegressor(base_score=None, booster=None, …	1.495870	0.395600
20	MLPRegressor()	2.302538	0.680757
17	(DecisionTreeRegressor(max_depth=3, random_sta…	2.575329	0.725043
10	KNeighborsRegressor()	1.832426	0.725657
23	NuSVR()	2.767677	0.795309
11	SVR()	1.778644	0.815938
0	LinearRegression()	1.858860	0.942806
2	RidgeCV()	1.858337	0.942913
4	SGDRegressor()	1.824808	0.942968
1	LassoCV()	1.872174	0.943096
3	ElasticNetCV()	1.869417	0.943112
6	ARDRegression()	1.896284	0.943125
5	HuberRegressor()	2.047510	0.945681
12	LinearSVR()	2.058692	0.953704
8	TweedieRegressor()	1.300327	0.964521
9	PassiveAggressiveRegressor()	5.203633	1.123522
7	RANSACRegressor()	4.191927	1.257529

	R2
25	0.985243
21	0.983999
24	0.983750
18	0.983634
26	0.983164
16	0.968876
14	0.968036
15	0.967707
13	0.948644
19	0.910850
22	0.847118
20	0.547282
17	0.486465
10	0.485594
23	0.382105
11	0.349636
0	0.131666
2	0.131468
4	0.131367
1	0.131130
3	0.131102
6	0.131078
5	0.126361
12	0.111474
8	0.091205
9	-0.233122
7	-0.544823

Among the baseline models, Cat Boost Regressor produced the best performance by obtaining an astonishing r2 score of more than 98.5%.

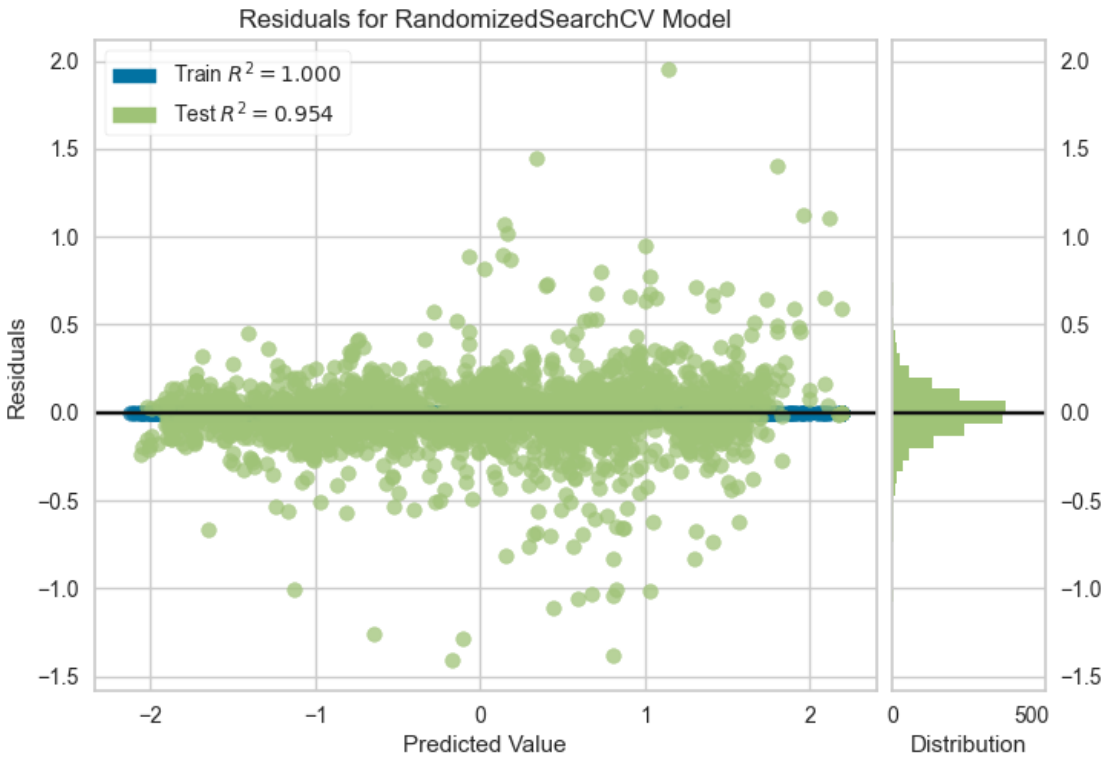
0.8 Hyperparameter Optimization and Cross Validation

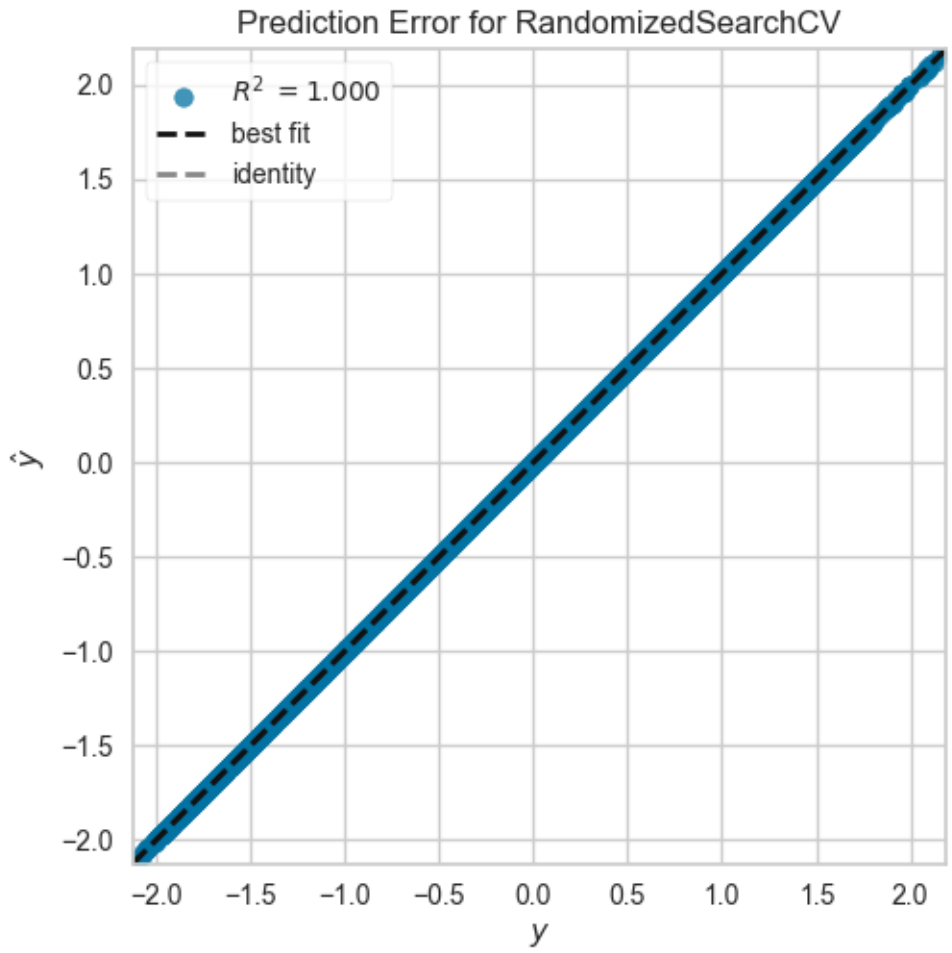
[89]:
```
param_grid = {'n_estimators': [200,400,600,800,1000],
              'criterion': ['squared_error', 'absolute_error', 'friedman_mse',␣
 ↪'poisson'],
              'max_features': ['auto','sqrt','log2'],
              'bootstrap': [True,False],
              'oob_score': [True,False]}

grid_rf = RandomizedSearchCV(RandomForestRegressor(),param_grid,verbose=2,cv=5)
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,
n_estimators=600, oob_score=True; total time=   5.2s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
n_estimators=600, oob_score=True; total time=   5.1s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
n_estimators=600, oob_score=True; total time=   5.2s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
n_estimators=600, oob_score=True; total time=   5.2s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
n_estimators=600, oob_score=True; total time=   6.0s
[CV] END bootstrap=True, criterion=absolute_error, max_features=log2,
n_estimators=200, oob_score=False; total time=  12.6s
[CV] END bootstrap=True, criterion=absolute_error, max_features=log2,
n_estimators=200, oob_score=False; total time=  12.8s
[CV] END bootstrap=True, criterion=absolute_error, max_features=log2,
n_estimators=200, oob_score=False; total time=  12.9s
[CV] END bootstrap=True, criterion=absolute_error, max_features=log2,
n_estimators=200, oob_score=False; total time=  12.5s
[CV] END bootstrap=True, criterion=absolute_error, max_features=log2,
n_estimators=200, oob_score=False; total time=  14.5s
[CV] END bootstrap=False, criterion=poisson, max_features=log2,
n_estimators=200, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2,
n_estimators=200, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2,
n_estimators=200, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2,
n_estimators=200, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=log2,
n_estimators=200, oob_score=False; total time=   0.0s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, n_estimators=400,
oob_score=True; total time=   0.0s
```

```
[CV] END bootstrap=True, criterion=poisson, max_features=log2, n_estimators=400,
oob_score=True; total time=   0.0s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, n_estimators=400,
oob_score=True; total time=   0.0s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, n_estimators=400,
oob_score=True; total time=   0.0s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, n_estimators=400,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=auto,
n_estimators=1000, oob_score=False; total time=  18.2s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=auto,
n_estimators=1000, oob_score=False; total time=  18.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=auto,
n_estimators=1000, oob_score=False; total time=  18.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=auto,
n_estimators=1000, oob_score=False; total time=  19.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=auto,
n_estimators=1000, oob_score=False; total time=  18.2s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
n_estimators=400, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
n_estimators=400, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
n_estimators=400, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
n_estimators=400, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
n_estimators=400, oob_score=True; total time=   0.0s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
n_estimators=800, oob_score=True; total time=   7.1s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
n_estimators=800, oob_score=True; total time=   7.1s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
n_estimators=800, oob_score=True; total time=   7.0s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
n_estimators=800, oob_score=True; total time=   7.0s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
n_estimators=800, oob_score=True; total time=   7.1s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
n_estimators=400, oob_score=False; total time=   7.9s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
n_estimators=400, oob_score=False; total time=   7.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
n_estimators=400, oob_score=False; total time=   7.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
n_estimators=400, oob_score=False; total time=   7.1s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
n_estimators=400, oob_score=False; total time=   7.1s
```

```
[CV] END bootstrap=False, criterion=absolute_error, max_features=sqrt,
n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=sqrt,
n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=sqrt,
n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=sqrt,
n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=sqrt,
n_estimators=1000, oob_score=True; total time=   0.0s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
n_estimators=600, oob_score=True; total time=   5.2s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
n_estimators=600, oob_score=True; total time=   5.3s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
n_estimators=600, oob_score=True; total time=   5.2s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
n_estimators=600, oob_score=True; total time=   5.3s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
n_estimators=600, oob_score=True; total time=   5.3s
Mean Absolute Percentage Error(MAPE): 0.6704343334863478
Root Mean Squared Error(RMSE): 0.21668601661566092
R2 Score: 0.9541325703008455
```





[90]:
```
grid_rf.best_params_
```

[90]:
```
{'oob_score': False,
 'n_estimators': 400,
 'max_features': 'auto',
 'criterion': 'squared_error',
 'bootstrap': False}
```

[91]:
```
param_grid = {'n_neighbors': [2,5,8,12,20],
              'weights': ['uniform','distance'],
              'algorithm': ['ball_tree', 'kd_tree', 'brute'],
              'metric': ['minkowski','manhattan','euclidean','chebyshev']
             }

grid_knn =␣
 ↪RandomizedSearchCV(KNeighborsRegressor(),param_grid,cv=RepeatedKFold(n_splits=5,n_repeats=3
train_and_evaluate_model(grid_knn)
```

```
Fitting 15 folds for each of 10 candidates, totalling 150 fits
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
```

```
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=euclidean, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
```

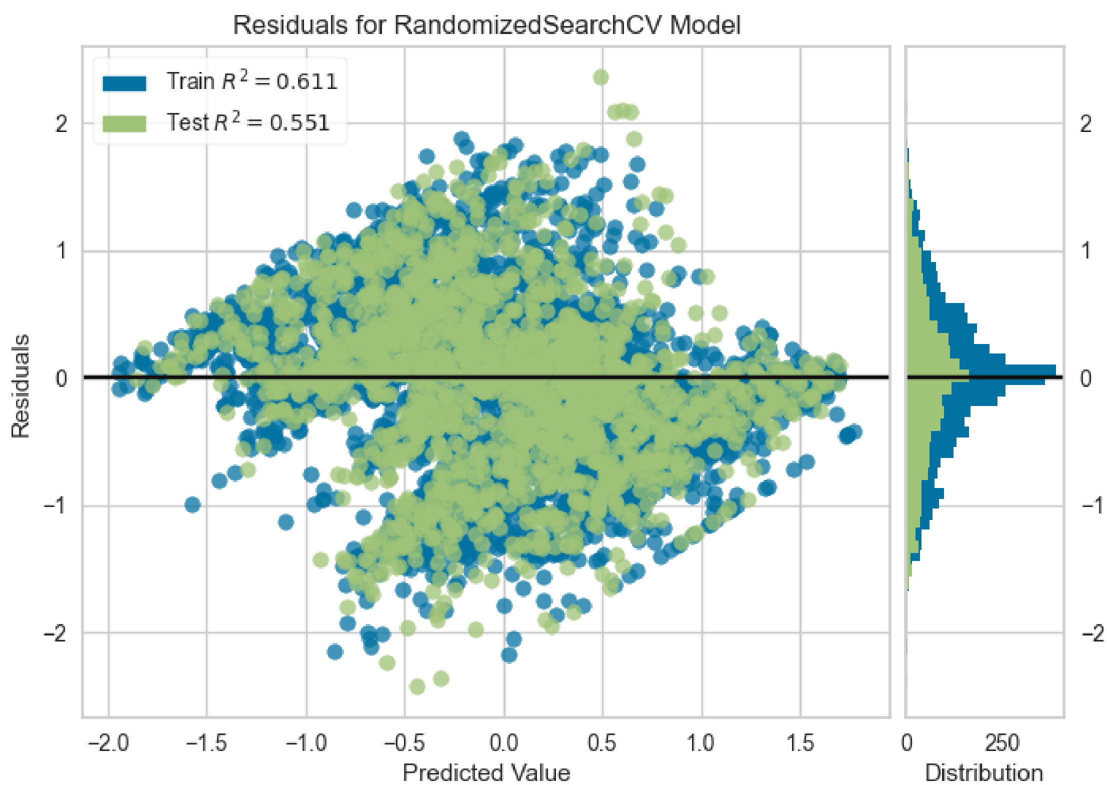
```
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
```

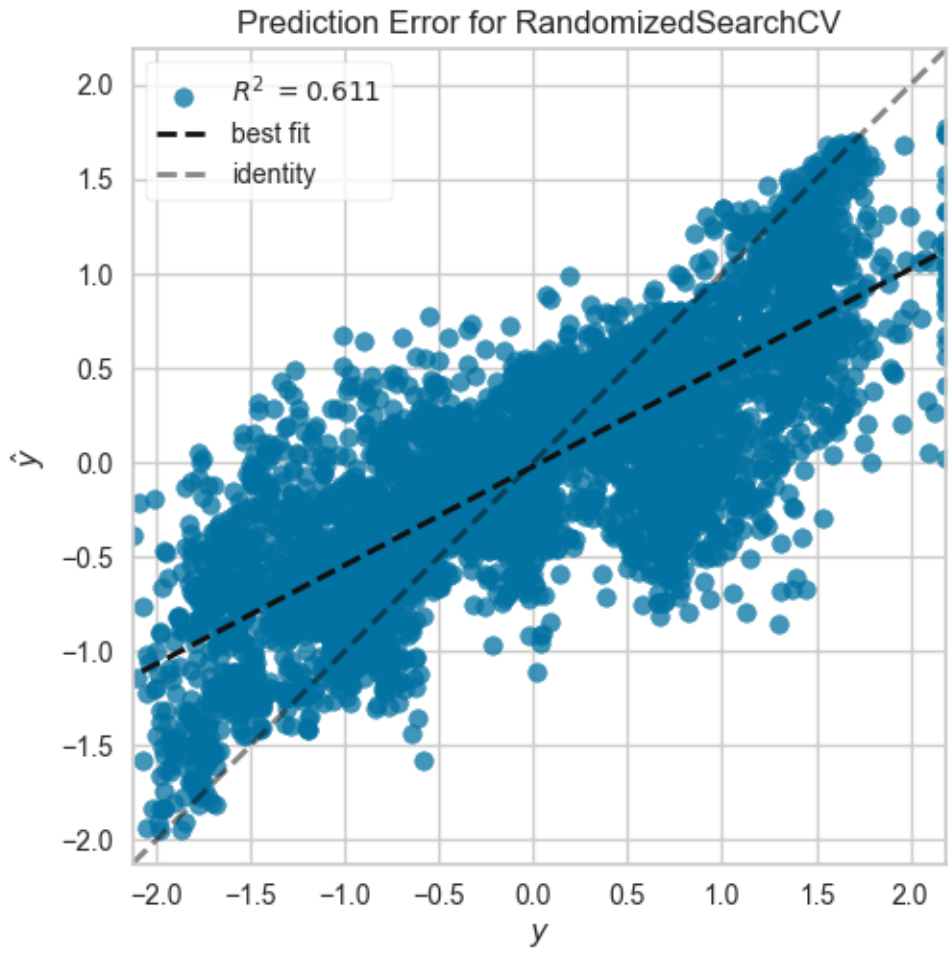
```
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, metric=manhattan, n_neighbors=20, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
```

```
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=manhattan, n_neighbors=12, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
weights=distance; total time=   0.0s
[CV] END algorithm=ball_tree, metric=chebyshev, n_neighbors=20,
```

```
weights=distance; total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, metric=chebyshev, n_neighbors=12, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
```

```
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, metric=chebyshev, n_neighbors=20, weights=distance;
total time=   0.0s
Mean Absolute Percentage Error(MAPE): 1.8270423271050913
Root Mean Squared Error(RMSE): 0.6781109353399526
R2 Score: 0.5507954691967909
```





[92]:
```
grid_knn.best_params_
```

[92]:
```
{'weights': 'uniform',
 'n_neighbors': 12,
 'metric': 'manhattan',
 'algorithm': 'ball_tree'}
```

[93]:
```
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=RepeatedKFold(n_splits=5,n
train_and_evaluate_model(grid_hgb)
```

```
Fitting 15 folds for each of 10 candidates, totalling 150 fits
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
```

```
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.8, loss=poisson, max_bins=0;
total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.9s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.9s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   1.3s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.6s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
```

```
max_bins=200; total time=   0.9s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   1.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   1.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   1.3s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   1.3s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=absolute_error,
max_bins=200; total time=   1.3s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=absolute_error,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
```

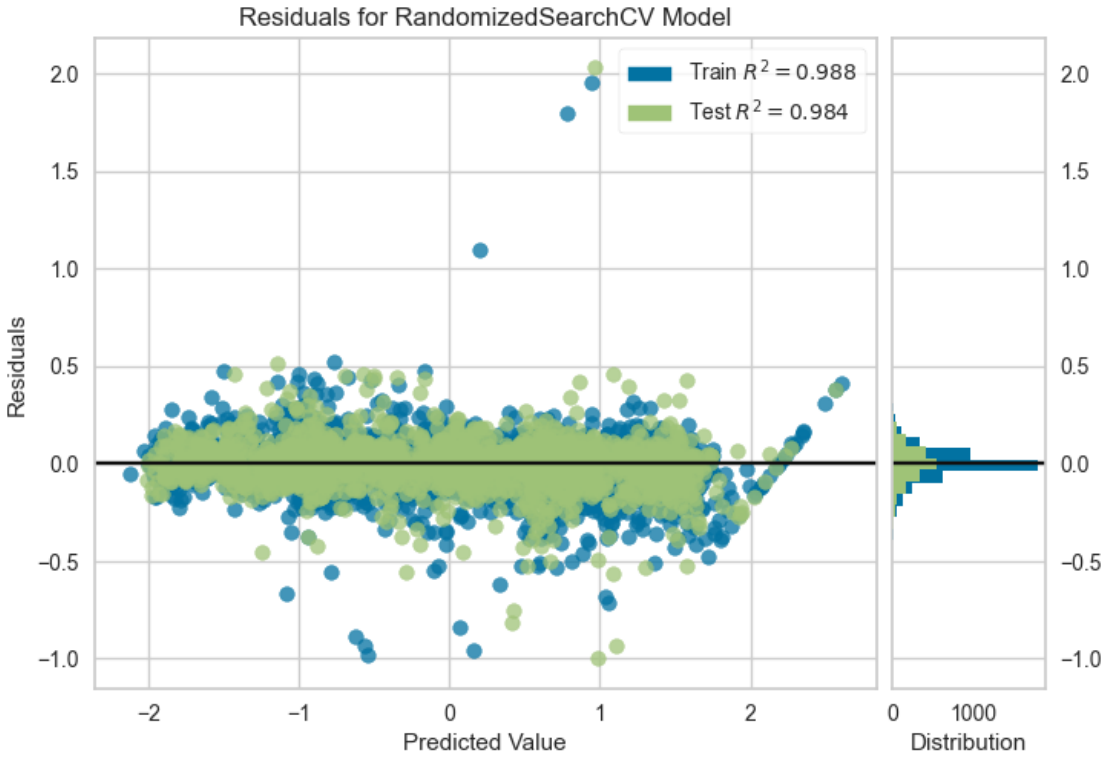
```
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=poisson,
max_bins=200; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
```

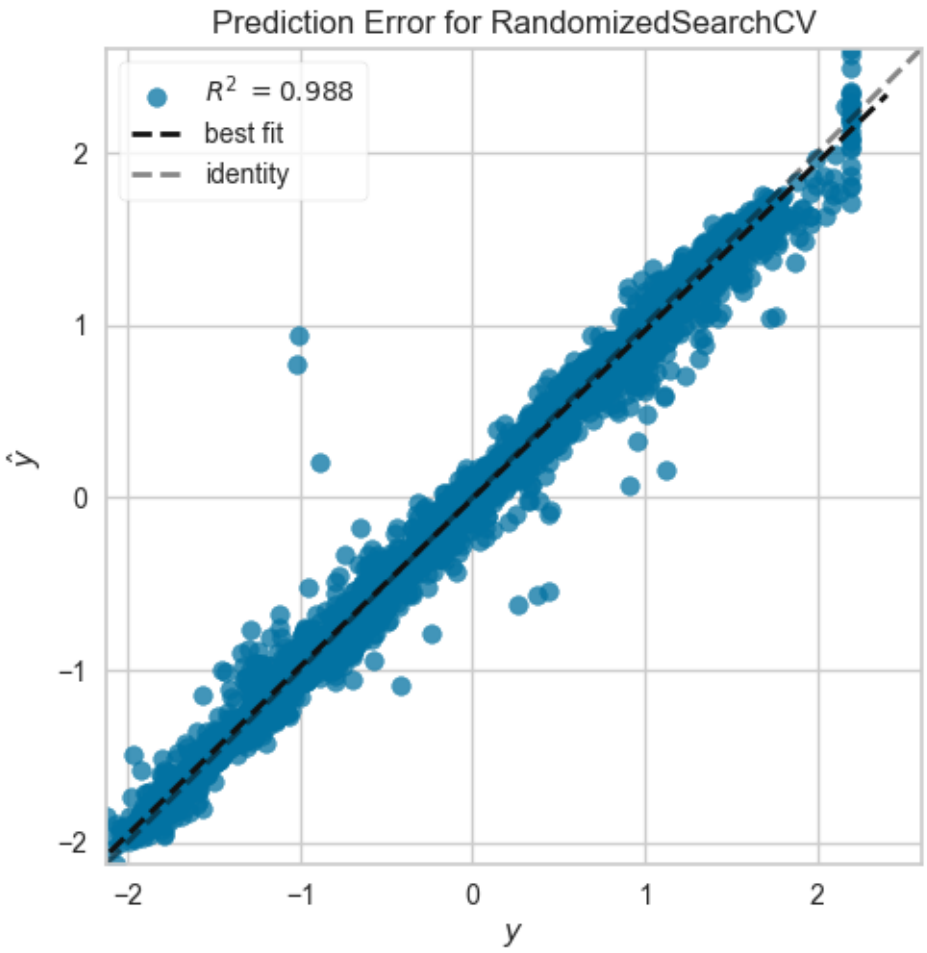
```
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.2, loss=poisson,
max_bins=0; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=1.0, loss=absolute_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   1.3s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   1.3s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   1.3s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   1.3s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   0.9s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
```

```
max_bins=250; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   1.1s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   0.8s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   1.2s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   0.6s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   0.5s
[CV] END interaction_cst=pairwise, learning_rate=0.5, loss=absolute_error,
max_bins=250; total time=   0.8s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=poisson,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
```

```
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=no_interaction, learning_rate=0.5, loss=squared_error,
max_bins=50; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
```

```
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
[CV] END interaction_cst=pairwise, learning_rate=0.2, loss=quantile,
max_bins=250; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 0.353993990408274
Root Mean Squared Error(RMSE): 0.1286878540246637
R2 Score: 0.9838222637922769
```





[94]:
```
grid_hgb.best_params_
```

[94]:
```
{'max_bins': 200,
 'loss': 'absolute_error',
 'learning_rate': 0.2,
 'interaction_cst': 'pairwise'}
```

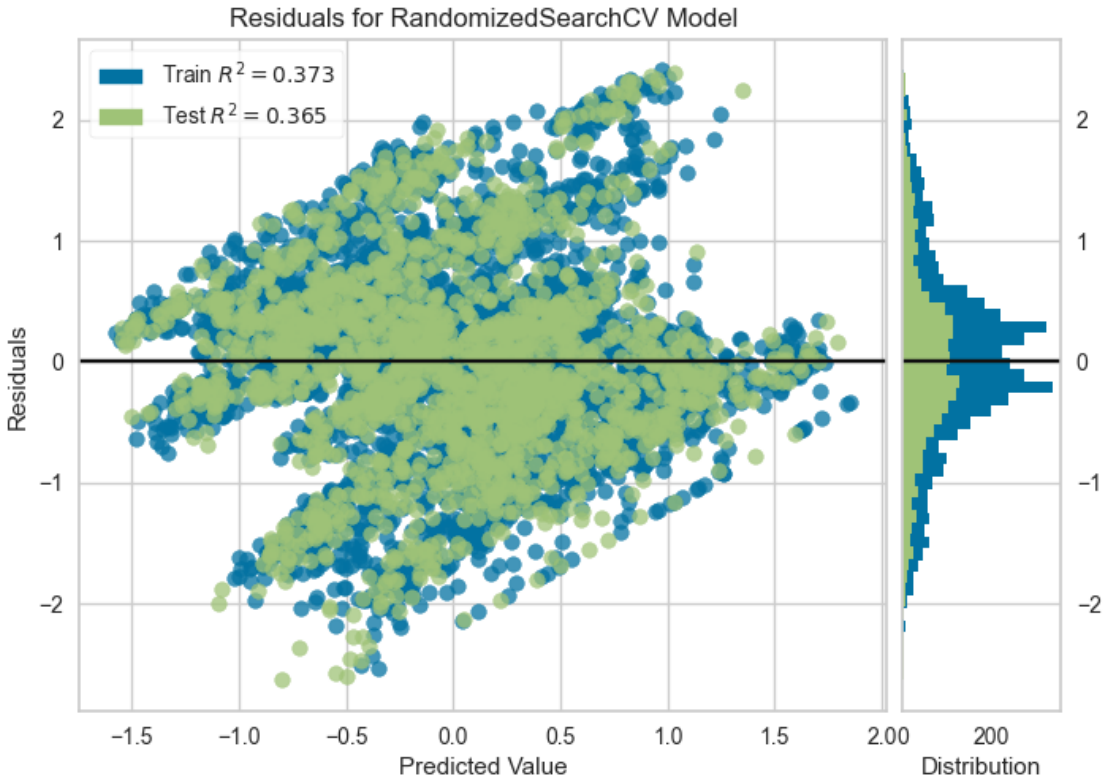
[95]:
```
param_grid = {'C': [0.001,0.01,0.1,1,5],
              'gamma': ['scale','auto'],
              'epsilon': np.linspace(0.001,1,5),
              'kernel': ['linear','poly','rbf','sigmoid'],
              'degree': [2,3,4,5],
              'shrinking': [True,False]
              }
grid_svr = RandomizedSearchCV(SVR(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_svr)
```

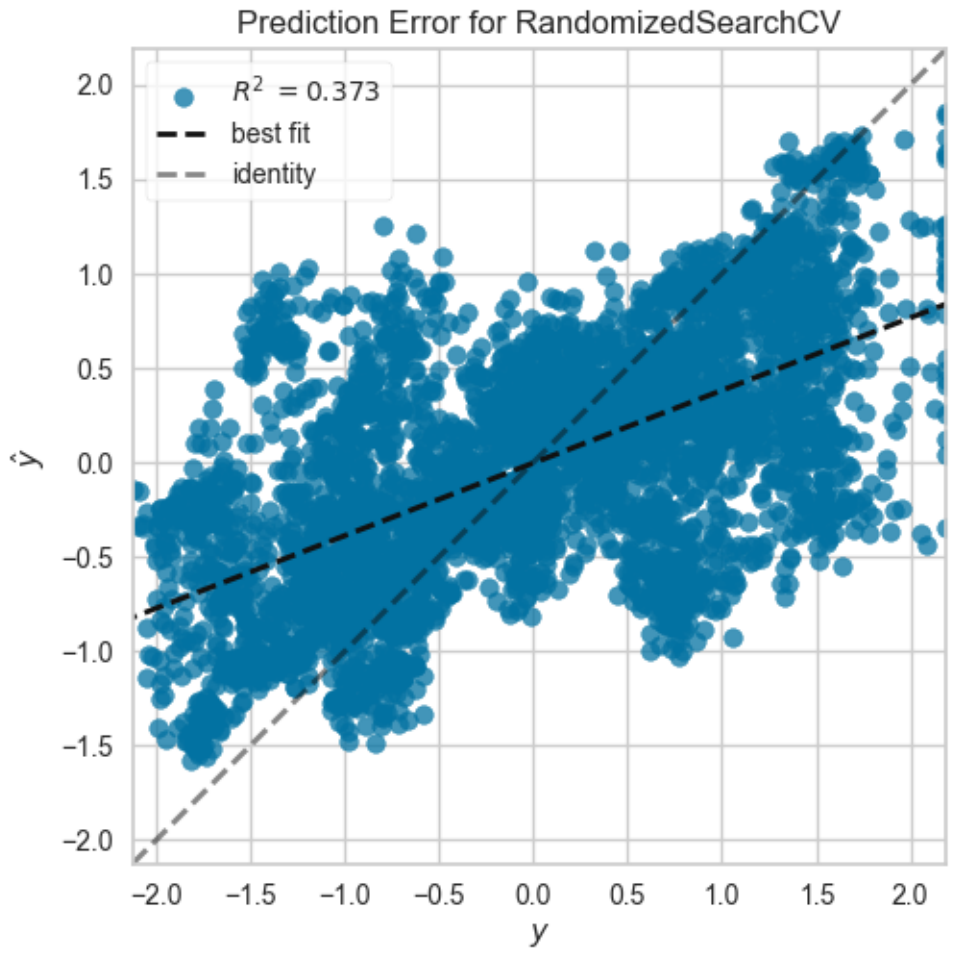
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[CV] END C=0.1, degree=5, epsilon=1.0, gamma=scale, kernel=linear,
shrinking=True; total time=   0.1s
[CV] END C=0.1, degree=5, epsilon=1.0, gamma=scale, kernel=linear,
shrinking=True; total time=   0.1s
[CV] END C=0.1, degree=5, epsilon=1.0, gamma=scale, kernel=linear,
shrinking=True; total time=   0.1s
[CV] END C=0.1, degree=5, epsilon=1.0, gamma=scale, kernel=linear,
shrinking=True; total time=   0.1s
[CV] END C=0.1, degree=5, epsilon=1.0, gamma=scale, kernel=linear,
shrinking=True; total time=   0.1s
[CV] END C=1, degree=3, epsilon=0.25075, gamma=scale, kernel=rbf,
shrinking=True; total time=   0.5s
[CV] END C=1, degree=3, epsilon=0.25075, gamma=scale, kernel=rbf,
shrinking=True; total time=   0.5s
[CV] END C=1, degree=3, epsilon=0.25075, gamma=scale, kernel=rbf,
shrinking=True; total time=   0.5s
[CV] END C=1, degree=3, epsilon=0.25075, gamma=scale, kernel=rbf,
shrinking=True; total time=   0.6s
[CV] END C=1, degree=3, epsilon=0.25075, gamma=scale, kernel=rbf,
shrinking=True; total time=   0.6s
[CV] END C=0.1, degree=3, epsilon=0.5005, gamma=auto, kernel=linear,
shrinking=True; total time=   0.2s
[CV] END C=0.1, degree=3, epsilon=0.5005, gamma=auto, kernel=linear,
shrinking=True; total time=   0.2s
[CV] END C=0.1, degree=3, epsilon=0.5005, gamma=auto, kernel=linear,
shrinking=True; total time=   0.4s
[CV] END C=0.1, degree=3, epsilon=0.5005, gamma=auto, kernel=linear,
shrinking=True; total time=   0.6s
[CV] END C=0.1, degree=3, epsilon=0.5005, gamma=auto, kernel=linear,
shrinking=True; total time=   0.6s
[CV] END C=0.1, degree=4, epsilon=0.25075, gamma=scale, kernel=sigmoid,
shrinking=False; total time=   0.7s
[CV] END C=0.1, degree=4, epsilon=0.25075, gamma=scale, kernel=sigmoid,
shrinking=False; total time=   0.7s
[CV] END C=0.1, degree=4, epsilon=0.25075, gamma=scale, kernel=sigmoid,
shrinking=False; total time=   0.7s
[CV] END C=0.1, degree=4, epsilon=0.25075, gamma=scale, kernel=sigmoid,
shrinking=False; total time=   0.7s
[CV] END C=0.1, degree=4, epsilon=0.25075, gamma=scale, kernel=sigmoid,
shrinking=False; total time=   0.7s
[CV] END C=1, degree=5, epsilon=0.75025, gamma=auto, kernel=poly,
shrinking=False; total time=   3.4s
[CV] END C=1, degree=5, epsilon=0.75025, gamma=auto, kernel=poly,
shrinking=False; total time=   2.7s
[CV] END C=1, degree=5, epsilon=0.75025, gamma=auto, kernel=poly,
shrinking=False; total time=   2.9s
[CV] END C=1, degree=5, epsilon=0.75025, gamma=auto, kernel=poly,
shrinking=False; total time=   3.4s
```

```
[CV] END C=1, degree=5, epsilon=0.75025, gamma=auto, kernel=poly,
shrinking=False; total time=   2.9s
[CV] END C=1, degree=2, epsilon=0.75025, gamma=auto, kernel=poly,
shrinking=False; total time=   0.4s
[CV] END C=1, degree=2, epsilon=0.75025, gamma=auto, kernel=poly,
shrinking=False; total time=   0.3s
[CV] END C=1, degree=2, epsilon=0.75025, gamma=auto, kernel=poly,
shrinking=False; total time=   0.3s
[CV] END C=1, degree=2, epsilon=0.75025, gamma=auto, kernel=poly,
shrinking=False; total time=   0.4s
[CV] END C=1, degree=2, epsilon=0.75025, gamma=auto, kernel=poly,
shrinking=False; total time=   0.3s
[CV] END C=0.1, degree=3, epsilon=0.001, gamma=scale, kernel=rbf,
shrinking=True; total time=   0.7s
[CV] END C=0.1, degree=3, epsilon=0.001, gamma=scale, kernel=rbf,
shrinking=True; total time=   0.7s
[CV] END C=0.1, degree=3, epsilon=0.001, gamma=scale, kernel=rbf,
shrinking=True; total time=   0.7s
[CV] END C=0.1, degree=3, epsilon=0.001, gamma=scale, kernel=rbf,
shrinking=True; total time=   0.7s
[CV] END C=0.1, degree=3, epsilon=0.001, gamma=scale, kernel=rbf,
shrinking=True; total time=   1.2s
[CV] END C=0.1, degree=2, epsilon=0.5005, gamma=auto, kernel=rbf,
shrinking=False; total time=   0.7s
[CV] END C=0.1, degree=2, epsilon=0.5005, gamma=auto, kernel=rbf,
shrinking=False; total time=   0.4s
[CV] END C=0.1, degree=2, epsilon=0.5005, gamma=auto, kernel=rbf,
shrinking=False; total time=   0.4s
[CV] END C=0.1, degree=2, epsilon=0.5005, gamma=auto, kernel=rbf,
shrinking=False; total time=   0.4s
[CV] END C=0.1, degree=2, epsilon=0.5005, gamma=auto, kernel=rbf,
shrinking=False; total time=   0.4s
[CV] END C=0.001, degree=5, epsilon=0.25075, gamma=auto, kernel=linear,
shrinking=False; total time=   0.2s
[CV] END C=0.001, degree=5, epsilon=0.25075, gamma=auto, kernel=linear,
shrinking=False; total time=   0.2s
[CV] END C=0.001, degree=5, epsilon=0.25075, gamma=auto, kernel=linear,
shrinking=False; total time=   0.4s
[CV] END C=0.001, degree=5, epsilon=0.25075, gamma=auto, kernel=linear,
shrinking=False; total time=   0.7s
[CV] END C=0.001, degree=5, epsilon=0.25075, gamma=auto, kernel=linear,
shrinking=False; total time=   0.7s
[CV] END C=0.01, degree=5, epsilon=0.25075, gamma=auto, kernel=poly,
shrinking=True; total time=   0.4s
[CV] END C=0.01, degree=5, epsilon=0.25075, gamma=auto, kernel=poly,
shrinking=True; total time=   0.3s
[CV] END C=0.01, degree=5, epsilon=0.25075, gamma=auto, kernel=poly,
shrinking=True; total time=   0.3s
```

```
[CV] END C=0.01, degree=5, epsilon=0.25075, gamma=auto, kernel=poly,
shrinking=True; total time=   0.3s
[CV] END C=0.01, degree=5, epsilon=0.25075, gamma=auto, kernel=poly,
shrinking=True; total time=   0.3s
Mean Absolute Percentage Error(MAPE): 2.0159023659059336
Root Mean Squared Error(RMSE): 0.8061338824665089
R2 Score: 0.36517067123023694
```





[96]:
```
grid_svr.best_params_
```

[96]:
```
{'shrinking': True,
 'kernel': 'rbf',
 'gamma': 'scale',
 'epsilon': 0.25075,
 'degree': 3,
 'C': 1}
```

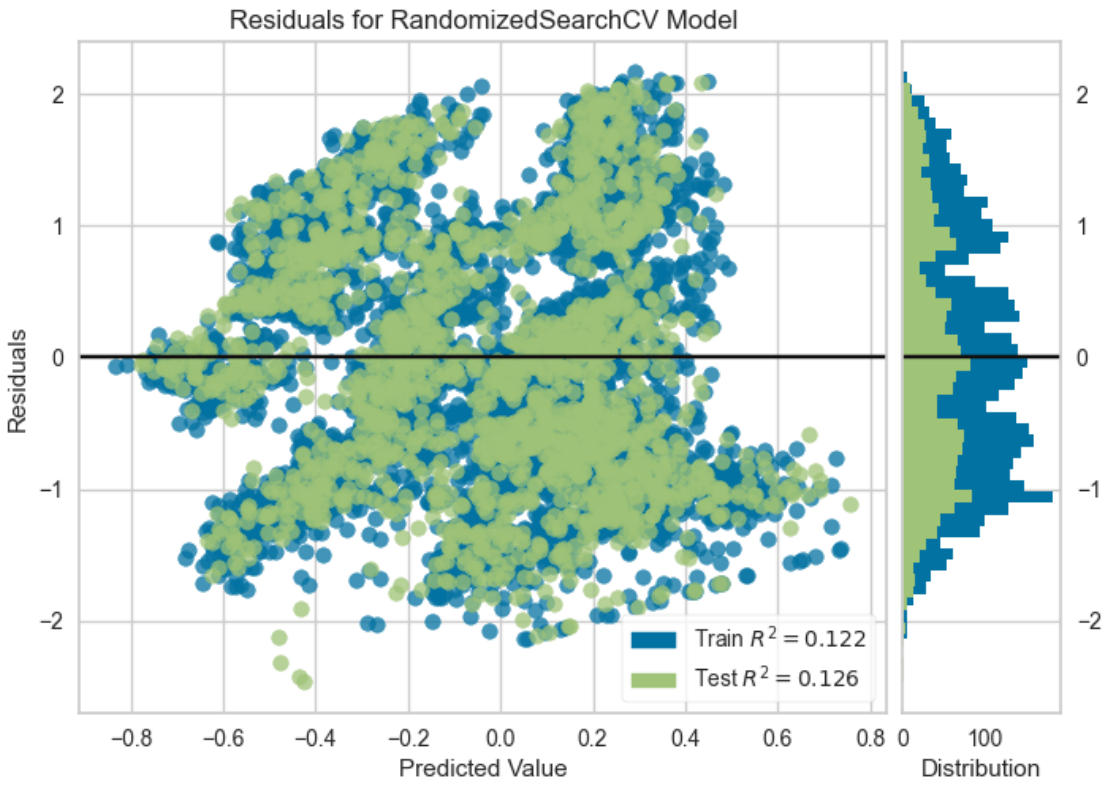
[97]:
```
param_grid = {'loss': ['epsilon_insensitive','squared_epsilon_insensitive'],
              'C': [0.0001,0.001,0.01,0.1,1],
              'epsilon': np.linspace(0.001,1,5)}

grid_lsvr =␣
 ↪RandomizedSearchCV(LinearSVR(),param_grid,cv=RepeatedKFold(n_splits=6,n_repeats=2),verbose=
train_and_evaluate_model(grid_lsvr)
```

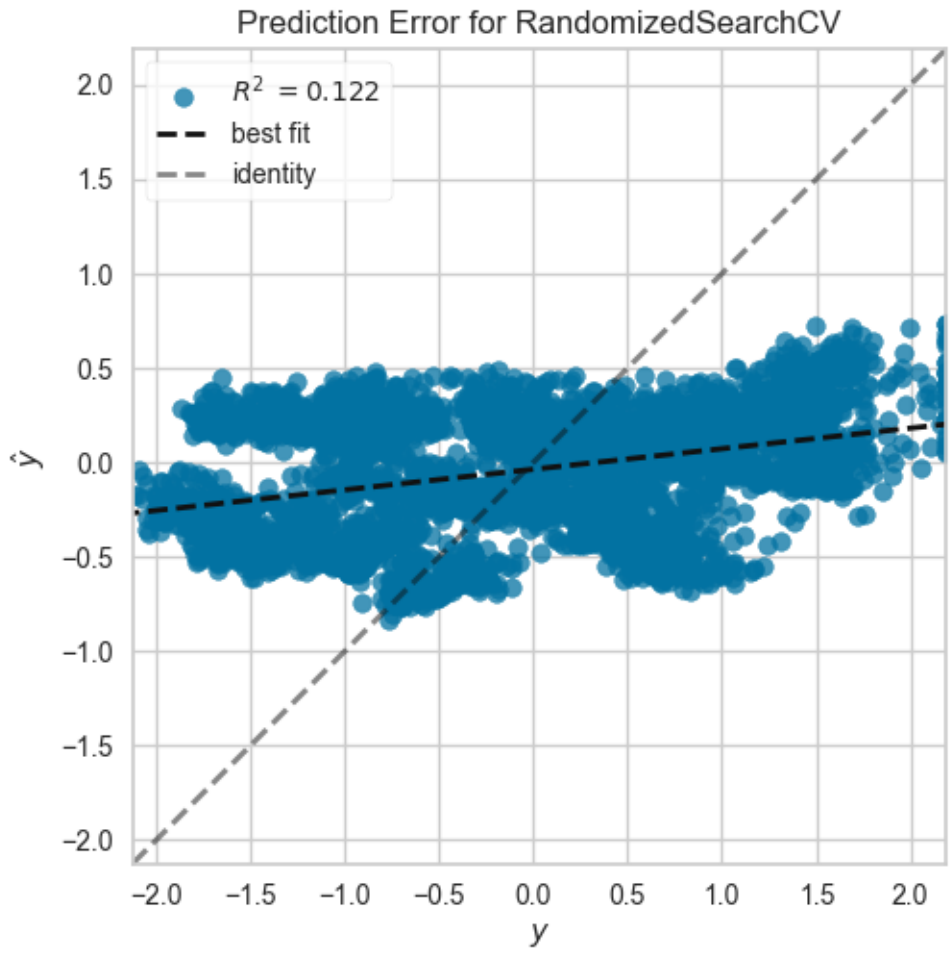
```
Fitting 12 folds for each of 10 candidates, totalling 120 fits
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.0001, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.001, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.25075, loss=squared_epsilon_insensitive; total time=
```

```
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END ..C=0.01, epsilon=0.25075, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
```

```
[CV] END …C=0.1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.1, epsilon=0.75025, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=0.5005, loss=epsilon_insensitive; total time=   0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
```

```
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END C=0.001, epsilon=0.5005, loss=squared_epsilon_insensitive; total time=
0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
[CV] END …C=1, epsilon=1.0, loss=epsilon_insensitive; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 1.8027668839748314
Root Mean Squared Error(RMSE): 0.9457010405217784
R2 Score: 0.12632408018212493
```


Residuals for RandomizedSearchCV Model
Residuals
Predicted Value
Distribution
Train $R^2 = 0.122$
Test $R^2 = 0.126$
2
1
0
−1
−2
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
0
100



[98]:
```
grid_lsvr.best_params_
```

[98]:
```
{'loss': 'squared_epsilon_insensitive', 'epsilon': 0.5005, 'C': 0.1}
```

[99]:
```
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=RepeatedKFold(n_splits=6
train_and_evaluate_model(grid_dt)
```

```
Fitting 18 folds for each of 10 candidates, totalling 180 fits
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
```

```
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=log2, splitter=best; total time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
```

```
[CV] END criterion=squared_error, max_features=log2, splitter=random; total
time=   0.0s
[CV] END criterion=squared_error, 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=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=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=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=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
```

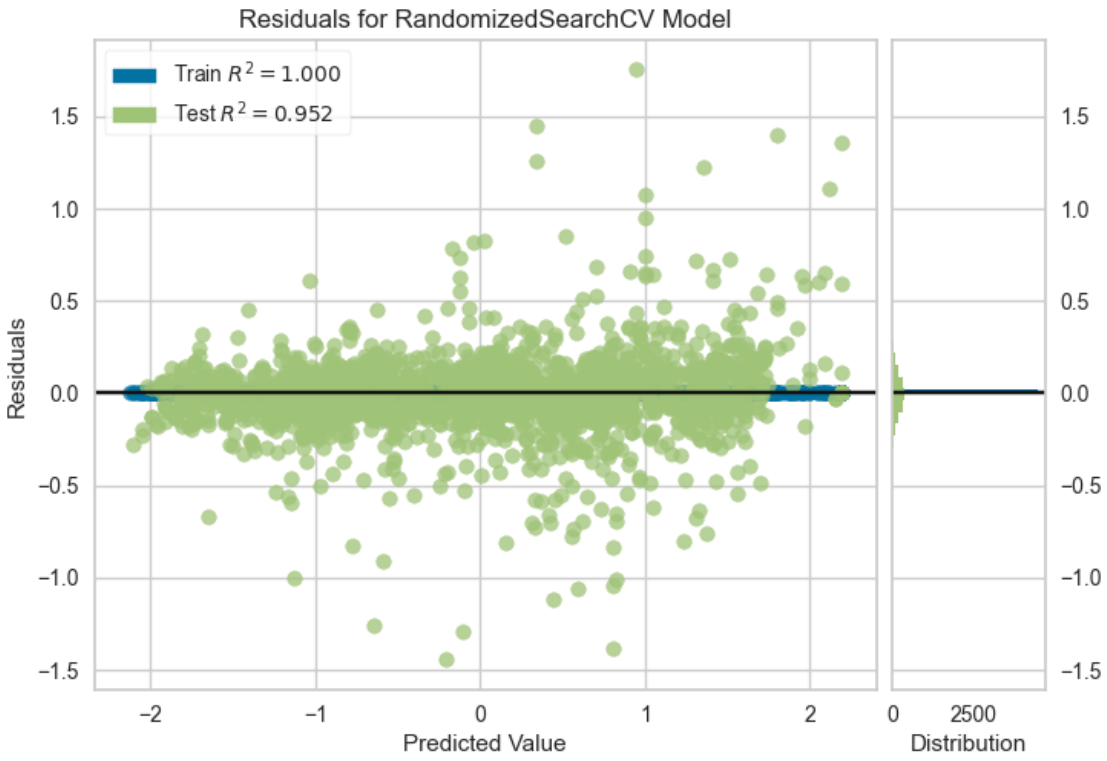
```
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.1s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.1s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.1s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.1s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.1s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.1s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.1s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.1s
```

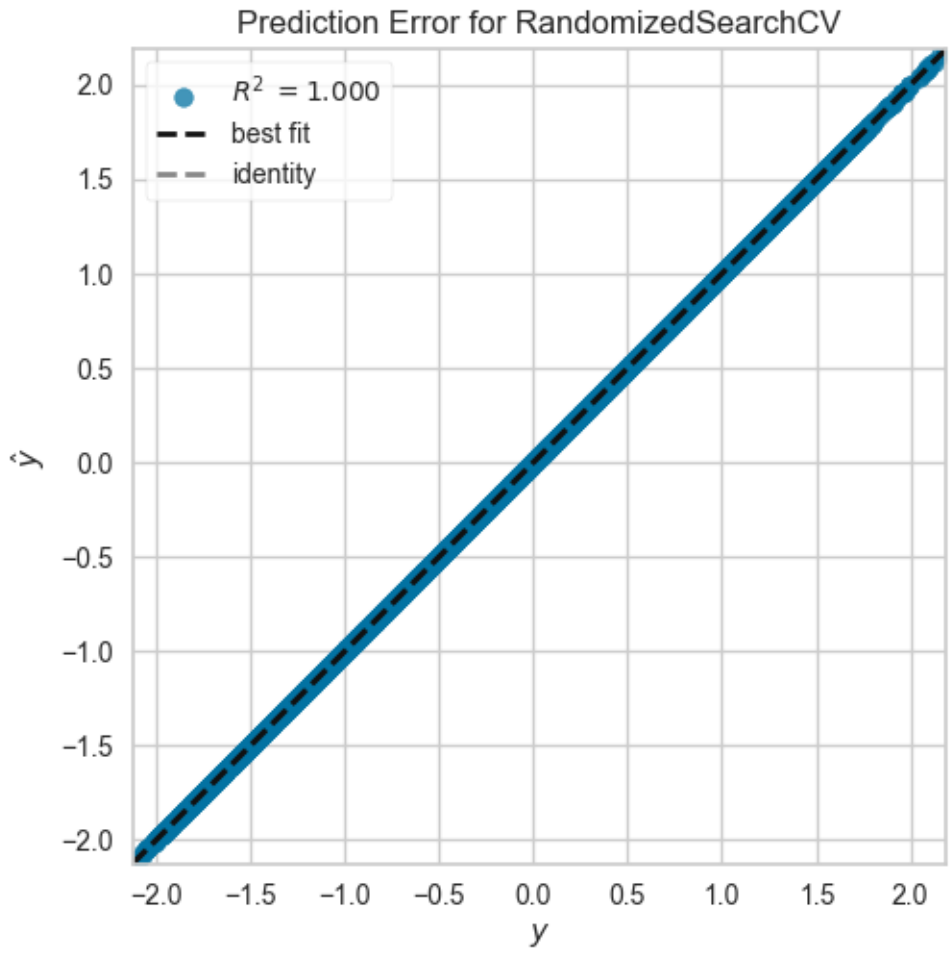
```
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.1s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=absolute_error, max_features=auto, splitter=random; total
time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; total time=   0.0s
[CV] END criterion=poisson, max_features=sqrt, splitter=best; 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=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=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=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=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.2s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.2s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.2s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.2s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
```

```
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.3s
[CV] END criterion=absolute_error, max_features=auto, splitter=best; total time=
0.2s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=friedman_mse, max_features=auto, splitter=random; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
```

```
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.1s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.1s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.1s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.1s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
[CV] END criterion=absolute_error, max_features=sqrt, splitter=best; total time=
0.0s
Mean Absolute Percentage Error(MAPE): 0.6790385905167696
Root Mean Squared Error(RMSE): 0.22065466321116575
R2 Score: 0.9524370424226405
```


Residuals for RandomizedSearchCV Model
Train $R^2 = 1.000$
Test $R^2 = 0.952$
Residuals
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2
Predicted Value
0
2500
Distribution



[100]:
```
grid_dt.best_params_
```

[100]:
```
{'splitter': 'best', 'max_features': 'auto', 'criterion': 'friedman_mse'}
```

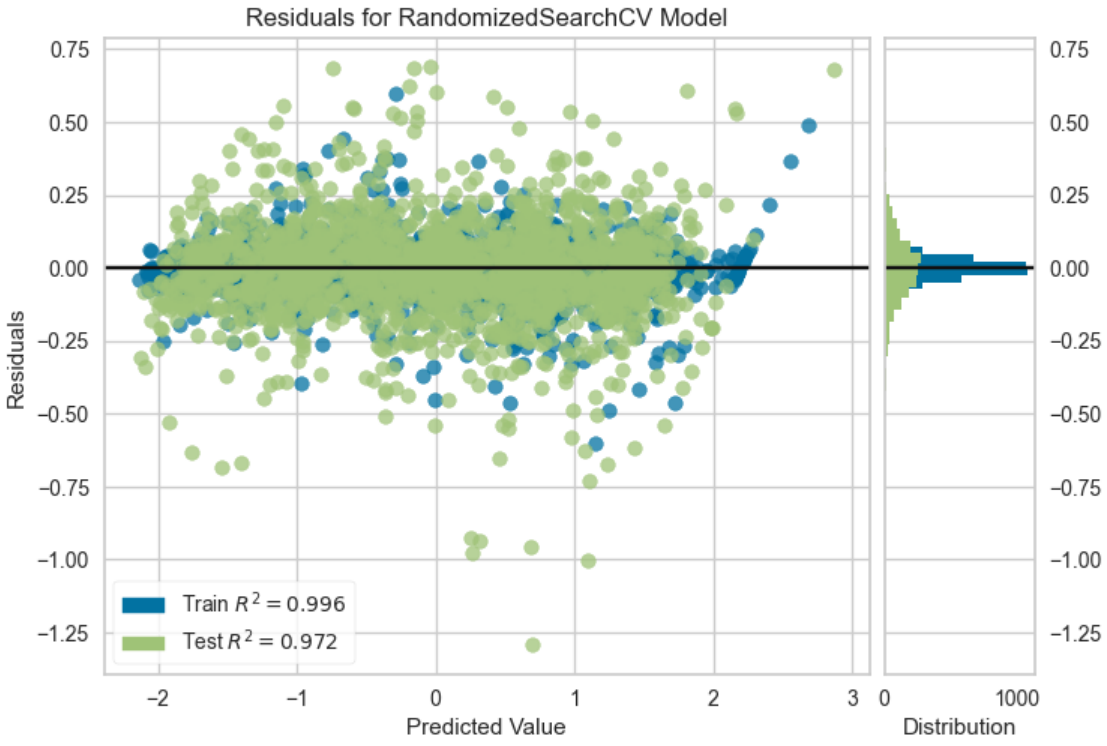
[101]:
```
param_grid = {'loss': ['squared_error','absolute_error','huber','quantile'],
              'n_estimators': [100,400,700,1000],
              'learning_rate': [0.2,0.4,0.7,1],
              'criterion': ['friedman_mse','squared_error'],
              'max_features': ['auto','sqrt','log2']
             }

grid_gb =␣
 ↪RandomizedSearchCV(GradientBoostingRegressor(),param_grid,verbose=3,cv=6)
train_and_evaluate_model(grid_gb)
```

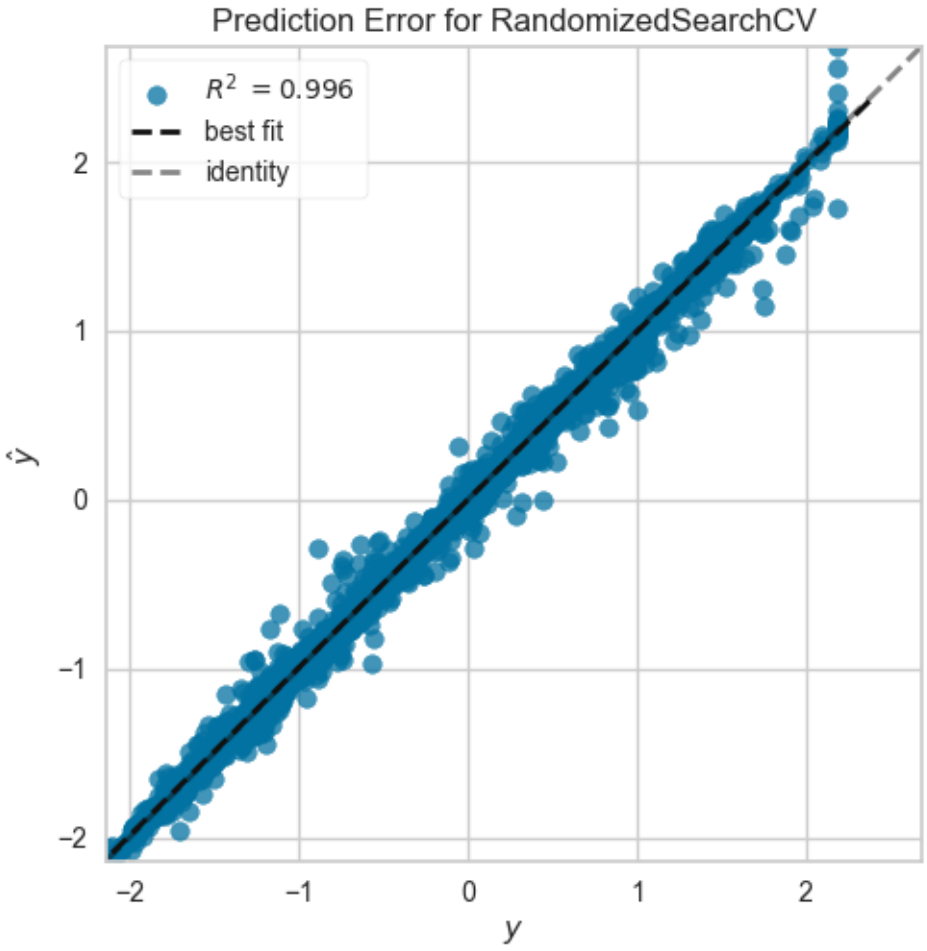
```
Fitting 6 folds for each of 10 candidates, totalling 60 fits
[CV 1/6] END criterion=friedman_mse, learning_rate=0.4, loss=absolute_error,
```

```
max_features=log2, n_estimators=1000;, score=0.969 total time=   3.5s
[CV 2/6] END criterion=friedman_mse, learning_rate=0.4, loss=absolute_error,
max_features=log2, n_estimators=1000;, score=0.976 total time=   3.4s
[CV 3/6] END criterion=friedman_mse, learning_rate=0.4, loss=absolute_error,
max_features=log2, n_estimators=1000;, score=0.960 total time=   3.6s
[CV 4/6] END criterion=friedman_mse, learning_rate=0.4, loss=absolute_error,
max_features=log2, n_estimators=1000;, score=0.962 total time=   3.6s
[CV 5/6] END criterion=friedman_mse, learning_rate=0.4, loss=absolute_error,
max_features=log2, n_estimators=1000;, score=0.967 total time=   3.5s
[CV 6/6] END criterion=friedman_mse, learning_rate=0.4, loss=absolute_error,
max_features=log2, n_estimators=1000;, score=0.960 total time=   3.6s
[CV 1/6] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.582 total time=   1.6s
[CV 2/6] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.514 total time=   1.7s
[CV 3/6] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.495 total time=   1.7s
[CV 4/6] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.628 total time=   1.6s
[CV 5/6] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.479 total time=   1.6s
[CV 6/6] END criterion=squared_error, learning_rate=0.2, loss=quantile,
max_features=log2, n_estimators=400;, score=0.543 total time=   1.4s
[CV 1/6] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.966 total time= 2.2min
[CV 2/6] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.972 total time=   0.5s
[CV 3/6] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.964 total time=   0.5s
[CV 4/6] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.962 total time=   0.7s
[CV 5/6] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.970 total time=   1.4s
[CV 6/6] END criterion=friedman_mse, learning_rate=0.2, loss=squared_error,
max_features=auto, n_estimators=100;, score=0.968 total time=   0.8s
[CV 1/6] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
max_features=auto, n_estimators=700;, score=0.934 total time=   6.4s
[CV 2/6] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
max_features=auto, n_estimators=700;, score=0.890 total time=   4.7s
[CV 3/6] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
max_features=auto, n_estimators=700;, score=0.881 total time=   4.6s
[CV 4/6] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
max_features=auto, n_estimators=700;, score=0.907 total time=   4.8s
[CV 5/6] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
max_features=auto, n_estimators=700;, score=0.922 total time=   4.6s
[CV 6/6] END criterion=friedman_mse, learning_rate=0.2, loss=quantile,
max_features=auto, n_estimators=700;, score=0.884 total time=   4.7s
[CV 1/6] END criterion=squared_error, learning_rate=1, loss=squared_error,
```

```
max_features=sqrt, n_estimators=100;, score=0.932 total time=   0.1s
[CV 2/6] END criterion=squared_error, learning_rate=1, loss=squared_error,
max_features=sqrt, n_estimators=100;, score=0.943 total time=   0.1s
[CV 3/6] END criterion=squared_error, learning_rate=1, loss=squared_error,
max_features=sqrt, n_estimators=100;, score=0.913 total time=   0.1s
[CV 4/6] END criterion=squared_error, learning_rate=1, loss=squared_error,
max_features=sqrt, n_estimators=100;, score=0.899 total time=   0.1s
[CV 5/6] END criterion=squared_error, learning_rate=1, loss=squared_error,
max_features=sqrt, n_estimators=100;, score=0.943 total time=   0.1s
[CV 6/6] END criterion=squared_error, learning_rate=1, loss=squared_error,
max_features=sqrt, n_estimators=100;, score=0.882 total time=   0.1s
[CV 1/6] END criterion=squared_error, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=1000;, score=0.969 total time=   4.4s
[CV 2/6] END criterion=squared_error, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=1000;, score=0.964 total time=   4.4s
[CV 3/6] END criterion=squared_error, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=1000;, score=0.970 total time=   4.4s
[CV 4/6] END criterion=squared_error, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=1000;, score=0.962 total time=   4.7s
[CV 5/6] END criterion=squared_error, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=1000;, score=0.975 total time=   4.4s
[CV 6/6] END criterion=squared_error, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=1000;, score=0.962 total time=   4.4s
[CV 1/6] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
max_features=log2, n_estimators=700;, score=0.967 total time=   1.2s
[CV 2/6] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
max_features=log2, n_estimators=700;, score=0.969 total time=   1.2s
[CV 3/6] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
max_features=log2, n_estimators=700;, score=0.960 total time=   1.6s
[CV 4/6] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
max_features=log2, n_estimators=700;, score=0.961 total time=   1.5s
[CV 5/6] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
max_features=log2, n_estimators=700;, score=0.971 total time=   1.5s
[CV 6/6] END criterion=squared_error, learning_rate=0.7, loss=squared_error,
max_features=log2, n_estimators=700;, score=0.969 total time=   1.4s
[CV 1/6] END criterion=friedman_mse, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=400;, score=0.967 total time=   1.7s
[CV 2/6] END criterion=friedman_mse, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=400;, score=0.959 total time=   1.7s
[CV 3/6] END criterion=friedman_mse, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=400;, score=0.953 total time=   1.7s
[CV 4/6] END criterion=friedman_mse, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=400;, score=0.963 total time=   1.7s
[CV 5/6] END criterion=friedman_mse, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=400;, score=0.966 total time=   1.7s
[CV 6/6] END criterion=friedman_mse, learning_rate=0.7, loss=huber,
max_features=sqrt, n_estimators=400;, score=0.964 total time=   1.7s
[CV 1/6] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
```

```
max_features=log2, n_estimators=700;, score=0.960 total time=   2.5s
[CV 2/6] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
max_features=log2, n_estimators=700;, score=0.966 total time=   2.5s
[CV 3/6] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
max_features=log2, n_estimators=700;, score=0.961 total time=   2.5s
[CV 4/6] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
max_features=log2, n_estimators=700;, score=0.949 total time=   2.5s
[CV 5/6] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
max_features=log2, n_estimators=700;, score=0.966 total time=   2.6s
[CV 6/6] END criterion=squared_error, learning_rate=0.2, loss=absolute_error,
max_features=log2, n_estimators=700;, score=0.964 total time=   2.5s
[CV 1/6] END criterion=squared_error, learning_rate=1, loss=huber,
max_features=log2, n_estimators=100;, score=0.931 total time=   0.4s
[CV 2/6] END criterion=squared_error, learning_rate=1, loss=huber,
max_features=log2, n_estimators=100;, score=0.894 total time=   0.4s
[CV 3/6] END criterion=squared_error, learning_rate=1, loss=huber,
max_features=log2, n_estimators=100;, score=0.935 total time=   0.4s
[CV 4/6] END criterion=squared_error, learning_rate=1, loss=huber,
max_features=log2, n_estimators=100;, score=0.927 total time=   0.4s
[CV 5/6] END criterion=squared_error, learning_rate=1, loss=huber,
max_features=log2, n_estimators=100;, score=0.922 total time=   0.4s
[CV 6/6] END criterion=squared_error, learning_rate=1, loss=huber,
max_features=log2, n_estimators=100;, score=0.951 total time=   0.4s
Mean Absolute Percentage Error(MAPE): 0.5385070919758704
Root Mean Squared Error(RMSE): 0.16830518927732563
R2 Score: 0.9723281850145602
```





[102]:
```
grid_gb.best_params_
```

[102]:
```
{'n_estimators': 1000,
 'max_features': 'sqrt',
 'loss': 'huber',
 'learning_rate': 0.7,
 'criterion': 'squared_error'}
```

[103]:
```
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=0.4, n_estimators=700;, score=0.986 total time=
1.7s
[CV 2/5] END learning_rate=0.4, n_estimators=700;, score=0.986 total time=
2.1s
[CV 3/5] END learning_rate=0.4, n_estimators=700;, score=0.984 total time=
1.6s
[CV 4/5] END learning_rate=0.4, n_estimators=700;, score=0.986 total time=
1.6s
[CV 5/5] END learning_rate=0.4, n_estimators=700;, score=0.986 total time=
1.9s
[CV 1/5] END learning_rate=0.2, n_estimators=800;, score=0.987 total time=
1.9s
[CV 2/5] END learning_rate=0.2, n_estimators=800;, score=0.986 total time=
1.8s
[CV 3/5] END learning_rate=0.2, n_estimators=800;, score=0.984 total time=
2.0s
[CV 4/5] END learning_rate=0.2, n_estimators=800;, score=0.986 total time=
1.9s
[CV 5/5] END learning_rate=0.2, n_estimators=800;, score=0.987 total time=
2.1s
[CV 1/5] END .learning_rate=1, n_estimators=700;, score=0.978 total time=   2.1s
[CV 2/5] END .learning_rate=1, n_estimators=700;, score=0.980 total time=   1.9s
[CV 3/5] END .learning_rate=1, n_estimators=700;, score=0.974 total time=   2.2s
[CV 4/5] END .learning_rate=1, n_estimators=700;, score=0.972 total time=   2.3s
[CV 5/5] END .learning_rate=1, n_estimators=700;, score=0.977 total time=   2.9s
[CV 1/5] END learning_rate=1, n_estimators=1000;, score=0.978 total time=   4.4s
[CV 2/5] END learning_rate=1, n_estimators=1000;, score=0.980 total time=   4.2s
[CV 3/5] END learning_rate=1, n_estimators=1000;, score=0.974 total time=   2.5s
[CV 4/5] END learning_rate=1, n_estimators=1000;, score=0.972 total time=   2.4s
[CV 5/5] END learning_rate=1, n_estimators=1000;, score=0.977 total time=   2.3s
[CV 1/5] END learning_rate=0.2, n_estimators=400;, score=0.986 total time=
1.5s
[CV 2/5] END learning_rate=0.2, n_estimators=400;, score=0.985 total time=
2.4s
[CV 3/5] END learning_rate=0.2, n_estimators=400;, score=0.983 total time=
2.3s
[CV 4/5] END learning_rate=0.2, n_estimators=400;, score=0.985 total time=
2.3s
[CV 5/5] END learning_rate=0.2, n_estimators=400;, score=0.986 total time=
2.3s
[CV 1/5] END learning_rate=0.2, n_estimators=700;, score=0.987 total time=
2.6s
[CV 2/5] END learning_rate=0.2, n_estimators=700;, score=0.986 total time=
2.2s
[CV 3/5] END learning_rate=0.2, n_estimators=700;, score=0.984 total time=
```

```
2.0s
[CV 4/5] END learning_rate=0.2, n_estimators=700;, score=0.986 total time=
2.0s
[CV 5/5] END learning_rate=0.2, n_estimators=700;, score=0.987 total time=
2.1s
[CV 1/5] END learning_rate=0.7, n_estimators=700;, score=0.983 total time=
2.4s
[CV 2/5] END learning_rate=0.7, n_estimators=700;, score=0.983 total time=
2.4s
[CV 3/5] END learning_rate=0.7, n_estimators=700;, score=0.978 total time=
2.2s
[CV 4/5] END learning_rate=0.7, n_estimators=700;, score=0.983 total time=
1.8s
[CV 5/5] END learning_rate=0.7, n_estimators=700;, score=0.981 total time=
1.8s
[CV 1/5] END learning_rate=0.7, n_estimators=400;, score=0.983 total time=
1.7s
[CV 2/5] END learning_rate=0.7, n_estimators=400;, score=0.983 total time=
1.6s
[CV 3/5] END learning_rate=0.7, n_estimators=400;, score=0.978 total time=
1.4s
[CV 4/5] END learning_rate=0.7, n_estimators=400;, score=0.983 total time=
2.2s
[CV 5/5] END learning_rate=0.7, n_estimators=400;, score=0.982 total time=
1.5s
[CV 1/5] END .learning_rate=1, n_estimators=100;, score=0.976 total time=   0.9s
[CV 2/5] END .learning_rate=1, n_estimators=100;, score=0.980 total time=   0.9s
[CV 3/5] END .learning_rate=1, n_estimators=100;, score=0.973 total time=   1.1s
[CV 4/5] END .learning_rate=1, n_estimators=100;, score=0.972 total time=   1.2s
[CV 5/5] END .learning_rate=1, n_estimators=100;, score=0.977 total time=   1.1s
[CV 1/5] END learning_rate=0.4, n_estimators=100;, score=0.980 total time=
1.0s
[CV 2/5] END learning_rate=0.4, n_estimators=100;, score=0.982 total time=
0.9s
[CV 3/5] END learning_rate=0.4, n_estimators=100;, score=0.978 total time=
1.2s
[CV 4/5] END learning_rate=0.4, n_estimators=100;, score=0.980 total time=
1.0s
[CV 5/5] END learning_rate=0.4, n_estimators=100;, score=0.981 total time=
0.9s
Mean Absolute Percentage Error(MAPE): 0.4397850247521942
Root Mean Squared Error(RMSE): 0.11688404578618586
R2 Score: 0.9866539390601259
```

[104]:
```
grid_cat.best_params_
```

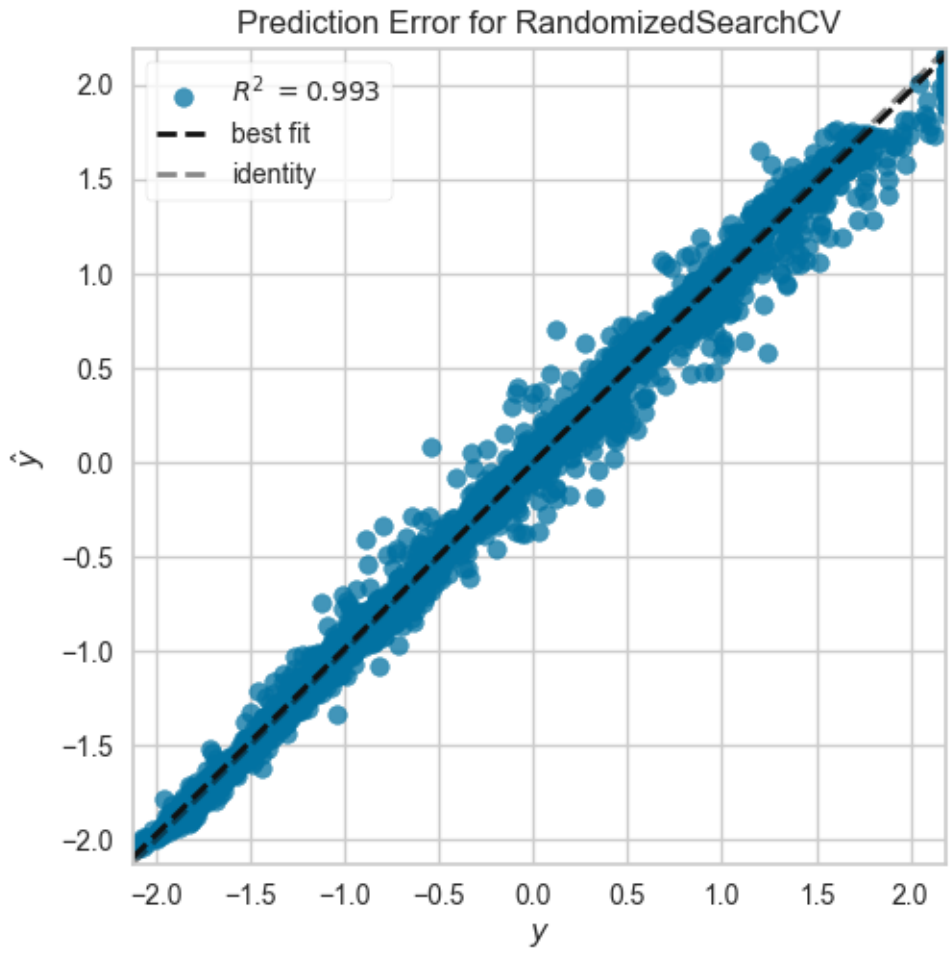
[104]:
```
{'n_estimators': 800, 'learning_rate': 0.2}
```

[105]:
```
param_grid = {'n_estimators': [100,300,500,800,1000],
              'max_samples': [0.24,0.58,0.71,0.96],
              'max_features': np.linspace(0,1,5),
              'bootstrap': [True,False],
              'oob_score': [True,False]
             }
grid_bag = RandomizedSearchCV(BaggingRegressor(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_bag)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END bootstrap=True, max_features=0.25, max_samples=0.24, n_estimators=300,
oob_score=True; total time=   0.9s
[CV] END bootstrap=True, max_features=0.25, max_samples=0.24, n_estimators=300,
oob_score=True; total time=   0.8s
[CV] END bootstrap=True, max_features=0.25, max_samples=0.24, n_estimators=300,
oob_score=True; total time=   0.8s
[CV] END bootstrap=True, max_features=0.25, max_samples=0.24, n_estimators=300,
oob_score=True; total time=   0.8s
[CV] END bootstrap=True, max_features=0.25, max_samples=0.24, n_estimators=300,
oob_score=True; total time=   0.8s
[CV] END bootstrap=False, max_features=0.5, max_samples=0.24, n_estimators=800,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, max_features=0.5, max_samples=0.24, n_estimators=800,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, max_features=0.5, max_samples=0.24, n_estimators=800,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, max_features=0.5, max_samples=0.24, n_estimators=800,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, max_features=0.5, max_samples=0.24, n_estimators=800,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, max_features=0.75, max_samples=0.96, n_estimators=500,
oob_score=False; total time=  10.5s
[CV] END bootstrap=False, max_features=0.75, max_samples=0.96, n_estimators=500,
oob_score=False; total time=  10.4s
[CV] END bootstrap=False, max_features=0.75, max_samples=0.96, n_estimators=500,
oob_score=False; total time=  10.5s
[CV] END bootstrap=False, max_features=0.75, max_samples=0.96, n_estimators=500,
oob_score=False; total time=  10.4s
[CV] END bootstrap=False, max_features=0.75, max_samples=0.96, n_estimators=500,
oob_score=False; total time=  11.5s
[CV] END bootstrap=True, max_features=1.0, max_samples=0.24, n_estimators=500,
oob_score=True; total time=   3.7s
[CV] END bootstrap=True, max_features=1.0, max_samples=0.24, n_estimators=500,
oob_score=True; total time=   3.6s
[CV] END bootstrap=True, max_features=1.0, max_samples=0.24, n_estimators=500,
oob_score=True; total time=   3.6s
[CV] END bootstrap=True, max_features=1.0, max_samples=0.24, n_estimators=500,
oob_score=True; total time=   3.6s
```

```
[CV] END bootstrap=True, max_features=1.0, max_samples=0.24, n_estimators=500,
oob_score=True; total time=   3.6s
[CV] END bootstrap=False, max_features=0.25, max_samples=0.71, n_estimators=500,
oob_score=False; total time=   2.2s
[CV] END bootstrap=False, max_features=0.25, max_samples=0.71, n_estimators=500,
oob_score=False; total time=   2.1s
[CV] END bootstrap=False, max_features=0.25, max_samples=0.71, n_estimators=500,
oob_score=False; total time=   2.2s
[CV] END bootstrap=False, max_features=0.25, max_samples=0.71, n_estimators=500,
oob_score=False; total time=   2.0s
[CV] END bootstrap=False, max_features=0.25, max_samples=0.71, n_estimators=500,
oob_score=False; total time=   1.9s
[CV] END bootstrap=False, max_features=0.75, max_samples=0.58, n_estimators=800,
oob_score=False; total time=  10.8s
[CV] END bootstrap=False, max_features=0.75, max_samples=0.58, n_estimators=800,
oob_score=False; total time=  10.5s
[CV] END bootstrap=False, max_features=0.75, max_samples=0.58, n_estimators=800,
oob_score=False; total time=  10.9s
[CV] END bootstrap=False, max_features=0.75, max_samples=0.58, n_estimators=800,
oob_score=False; total time=  11.2s
[CV] END bootstrap=False, max_features=0.75, max_samples=0.58, n_estimators=800,
oob_score=False; total time=  10.2s
[CV] END bootstrap=False, max_features=0.0, max_samples=0.24, n_estimators=100,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, max_features=0.0, max_samples=0.24, n_estimators=100,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, max_features=0.0, max_samples=0.24, n_estimators=100,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, max_features=0.0, max_samples=0.24, n_estimators=100,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, max_features=0.0, max_samples=0.24, n_estimators=100,
oob_score=True; total time=   0.0s
[CV] END bootstrap=False, max_features=0.5, max_samples=0.24, n_estimators=800,
oob_score=False; total time=   3.6s
[CV] END bootstrap=False, max_features=0.5, max_samples=0.24, n_estimators=800,
oob_score=False; total time=   3.5s
[CV] END bootstrap=False, max_features=0.5, max_samples=0.24, n_estimators=800,
oob_score=False; total time=   3.6s
[CV] END bootstrap=False, max_features=0.5, max_samples=0.24, n_estimators=800,
oob_score=False; total time=   3.6s
[CV] END bootstrap=False, max_features=0.5, max_samples=0.24, n_estimators=800,
oob_score=False; total time=   3.6s
[CV] END bootstrap=False, max_features=1.0, max_samples=0.71, n_estimators=100,
oob_score=False; total time=   1.9s
[CV] END bootstrap=False, max_features=1.0, max_samples=0.71, n_estimators=100,
oob_score=False; total time=   1.9s
[CV] END bootstrap=False, max_features=1.0, max_samples=0.71, n_estimators=100,
oob_score=False; total time=   1.9s
```

```
[CV] END bootstrap=False, max_features=1.0, max_samples=0.71, n_estimators=100,
oob_score=False; total time=   1.9s
[CV] END bootstrap=False, max_features=1.0, max_samples=0.71, n_estimators=100,
oob_score=False; total time=   1.9s
[CV] END bootstrap=True, max_features=1.0, max_samples=0.71, n_estimators=1000,
oob_score=False; total time=  14.1s
[CV] END bootstrap=True, max_features=1.0, max_samples=0.71, n_estimators=1000,
oob_score=False; total time=  15.1s
[CV] END bootstrap=True, max_features=1.0, max_samples=0.71, n_estimators=1000,
oob_score=False; total time=  14.2s
[CV] END bootstrap=True, max_features=1.0, max_samples=0.71, n_estimators=1000,
oob_score=False; total time=  14.2s
[CV] END bootstrap=True, max_features=1.0, max_samples=0.71, n_estimators=1000,
oob_score=False; total time=  14.1s
Mean Absolute Percentage Error(MAPE): 0.6257275100955686
Root Mean Squared Error(RMSE): 0.17932731459094725
R2 Score: 0.9685851115331399
```



[106]:
```
grid_bag.best_params_
```

[106]:
```
{'oob_score': False,
 'n_estimators': 1000,
 'max_samples': 0.71,
 'max_features': 1.0,
 'bootstrap': True}
```

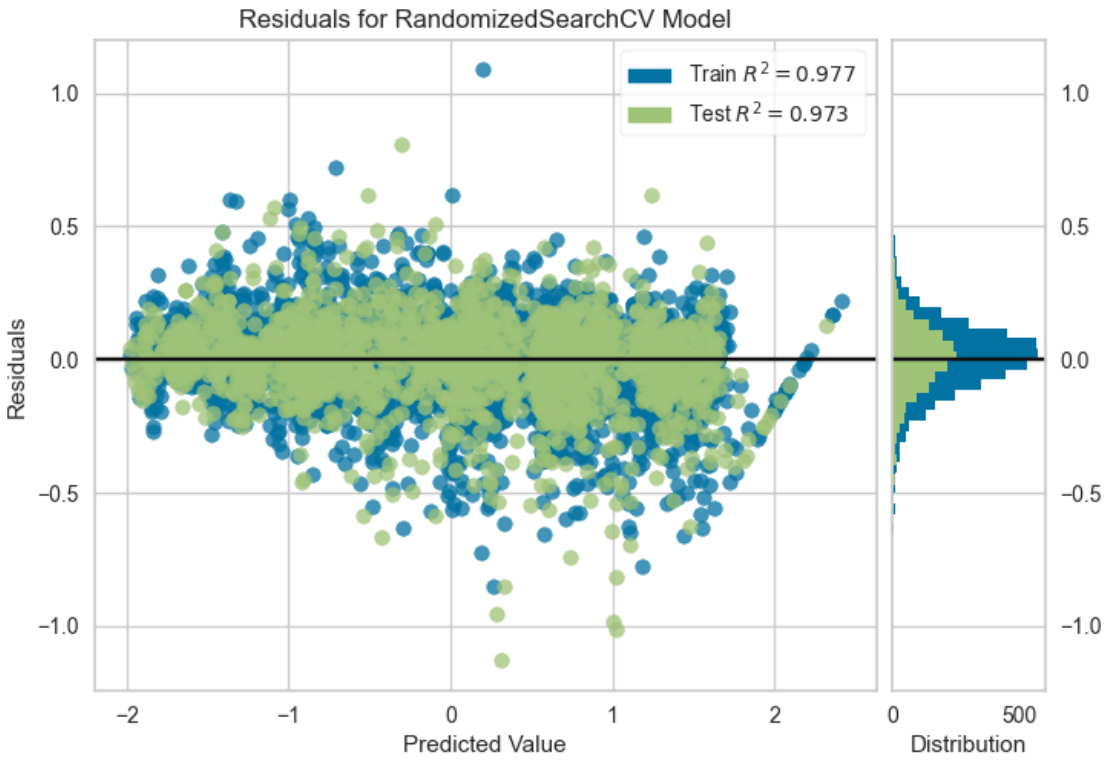
[107]:
```
param_grid = {'boosting_type': ['gbdt','dart','goss','rf'],
              'learning_rate': np.linspace(0,1,6)[1:],
              'n_estimators': [100,300,500,800,1000],
              'importance_type': ['split','gain'],
              'min_split_gain': [0.68,0.79,0.87,1]}

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

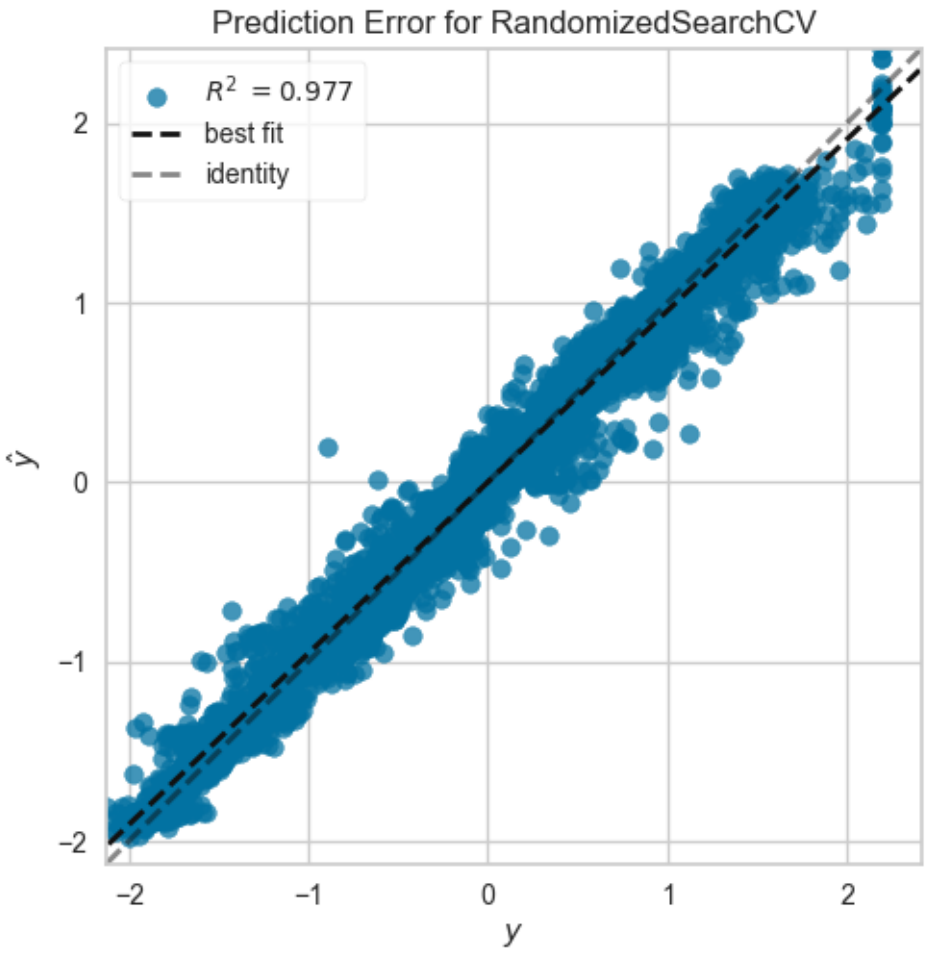
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[CV 1/5] END boosting_type=goss, importance_type=gain, learning_rate=0.2,
min_split_gain=1, n_estimators=1000;, score=0.967 total time=   0.0s
[CV 2/5] END boosting_type=goss, importance_type=gain, learning_rate=0.2,
min_split_gain=1, n_estimators=1000;, score=0.970 total time=   0.0s
[CV 3/5] END boosting_type=goss, importance_type=gain, learning_rate=0.2,
min_split_gain=1, n_estimators=1000;, score=0.960 total time=   0.0s
[CV 4/5] END boosting_type=goss, importance_type=gain, learning_rate=0.2,
min_split_gain=1, n_estimators=1000;, score=0.966 total time=   0.0s
[CV 5/5] END boosting_type=goss, importance_type=gain, learning_rate=0.2,
min_split_gain=1, n_estimators=1000;, score=0.973 total time=   0.0s
[CV 1/5] END boosting_type=goss, importance_type=split, learning_rate=0.4,
min_split_gain=0.68, n_estimators=500;, score=0.971 total time=   0.0s
[CV 2/5] END boosting_type=goss, importance_type=split, learning_rate=0.4,
min_split_gain=0.68, n_estimators=500;, score=0.970 total time=   0.0s
[CV 3/5] END boosting_type=goss, importance_type=split, learning_rate=0.4,
min_split_gain=0.68, n_estimators=500;, score=0.967 total time=   0.0s
[CV 4/5] END boosting_type=goss, importance_type=split, learning_rate=0.4,
min_split_gain=0.68, n_estimators=500;, score=0.969 total time=   0.0s
[CV 5/5] END boosting_type=goss, importance_type=split, learning_rate=0.4,
min_split_gain=0.68, n_estimators=500;, score=0.971 total time=   0.0s
[CV 1/5] END boosting_type=dart, importance_type=split, learning_rate=1.0,
min_split_gain=1, n_estimators=800;, score=0.951 total time=   0.5s
[CV 2/5] END boosting_type=dart, importance_type=split, learning_rate=1.0,
min_split_gain=1, n_estimators=800;, score=0.941 total time=   0.9s
[CV 3/5] END boosting_type=dart, importance_type=split, learning_rate=1.0,
min_split_gain=1, n_estimators=800;, score=0.935 total time=   0.9s
[CV 4/5] END boosting_type=dart, importance_type=split, learning_rate=1.0,
min_split_gain=1, n_estimators=800;, score=0.940 total time=   0.5s
[CV 5/5] END boosting_type=dart, importance_type=split, learning_rate=1.0,
min_split_gain=1, n_estimators=800;, score=0.963 total time=   0.5s
[CV 1/5] END boosting_type=goss, importance_type=split, learning_rate=0.4,
min_split_gain=0.79, n_estimators=1000;, score=0.970 total time=   0.0s
[CV 2/5] END boosting_type=goss, importance_type=split, learning_rate=0.4,
min_split_gain=0.79, n_estimators=1000;, score=0.970 total time=   0.2s
[CV 3/5] END boosting_type=goss, importance_type=split, learning_rate=0.4,
min_split_gain=0.79, n_estimators=1000;, score=0.967 total time=   0.1s
[CV 4/5] END boosting_type=goss, importance_type=split, learning_rate=0.4,
min_split_gain=0.79, n_estimators=1000;, score=0.969 total time=   0.2s
[CV 5/5] END boosting_type=goss, importance_type=split, learning_rate=0.4,
min_split_gain=0.79, n_estimators=1000;, score=0.971 total time=   0.2s
[CV 1/5] END boosting_type=goss, importance_type=gain, learning_rate=0.2,
min_split_gain=0.79, n_estimators=500;, score=0.969 total time=   0.0s
[CV 2/5] END boosting_type=goss, importance_type=gain, learning_rate=0.2,
min_split_gain=0.79, n_estimators=500;, score=0.972 total time=   0.1s
[CV 3/5] END boosting_type=goss, importance_type=gain, learning_rate=0.2,
min_split_gain=0.79, n_estimators=500;, score=0.963 total time=   0.0s
[CV 4/5] END boosting_type=goss, importance_type=gain, learning_rate=0.2,
min_split_gain=0.79, n_estimators=500;, score=0.970 total time=   0.0s
```

```
[CV 5/5] END boosting_type=goss, importance_type=gain, learning_rate=0.2,
min_split_gain=0.79, n_estimators=500;, score=0.972 total time=   0.0s
[CV 1/5] END boosting_type=goss, importance_type=split,
learning_rate=0.6000000000000001, min_split_gain=1, n_estimators=800;,
score=0.965 total time=   0.0s
[CV 2/5] END boosting_type=goss, importance_type=split,
learning_rate=0.6000000000000001, min_split_gain=1, n_estimators=800;,
score=0.965 total time=   0.0s
[CV 3/5] END boosting_type=goss, importance_type=split,
learning_rate=0.6000000000000001, min_split_gain=1, n_estimators=800;,
score=0.963 total time=   0.0s
[CV 4/5] END boosting_type=goss, importance_type=split,
learning_rate=0.6000000000000001, min_split_gain=1, n_estimators=800;,
score=0.964 total time=   0.0s
[CV 5/5] END boosting_type=goss, importance_type=split,
learning_rate=0.6000000000000001, min_split_gain=1, n_estimators=800;,
score=0.966 total time=   0.0s
[CV 1/5] END boosting_type=dart, importance_type=split, learning_rate=0.8,
min_split_gain=0.87, n_estimators=1000;, score=0.964 total time=   0.9s
[CV 2/5] END boosting_type=dart, importance_type=split, learning_rate=0.8,
min_split_gain=0.87, n_estimators=1000;, score=0.971 total time=   1.9s
[CV 3/5] END boosting_type=dart, importance_type=split, learning_rate=0.8,
min_split_gain=0.87, n_estimators=1000;, score=0.967 total time=   2.1s
[CV 4/5] END boosting_type=dart, importance_type=split, learning_rate=0.8,
min_split_gain=0.87, n_estimators=1000;, score=0.967 total time=   2.0s
[CV 5/5] END boosting_type=dart, importance_type=split, learning_rate=0.8,
min_split_gain=0.87, n_estimators=1000;, score=0.971 total time=   2.1s
[CV 1/5] END boosting_type=rf, importance_type=gain, learning_rate=0.8,
min_split_gain=0.79, n_estimators=800;, score=nan total time=   0.0s
[CV 2/5] END boosting_type=rf, importance_type=gain, learning_rate=0.8,
min_split_gain=0.79, n_estimators=800;, score=nan total time=   0.0s
[CV 3/5] END boosting_type=rf, importance_type=gain, learning_rate=0.8,
min_split_gain=0.79, n_estimators=800;, score=nan total time=   0.0s
[CV 4/5] END boosting_type=rf, importance_type=gain, learning_rate=0.8,
min_split_gain=0.79, n_estimators=800;, score=nan total time=   0.0s
[CV 5/5] END boosting_type=rf, importance_type=gain, learning_rate=0.8,
min_split_gain=0.79, n_estimators=800;, score=nan total time=   0.0s
[CV 1/5] END boosting_type=gbdt, importance_type=gain, learning_rate=0.4,
min_split_gain=0.87, n_estimators=800;, score=0.956 total time=   0.0s
[CV 2/5] END boosting_type=gbdt, importance_type=gain, learning_rate=0.4,
min_split_gain=0.87, n_estimators=800;, score=0.964 total time=   0.0s
[CV 3/5] END boosting_type=gbdt, importance_type=gain, learning_rate=0.4,
min_split_gain=0.87, n_estimators=800;, score=0.957 total time=   0.0s
[CV 4/5] END boosting_type=gbdt, importance_type=gain, learning_rate=0.4,
min_split_gain=0.87, n_estimators=800;, score=0.964 total time=   0.0s
[CV 5/5] END boosting_type=gbdt, importance_type=gain, learning_rate=0.4,
min_split_gain=0.87, n_estimators=800;, score=0.968 total time=   0.0s
[CV 1/5] END boosting_type=gbdt, importance_type=split, learning_rate=0.4,
```

```
min_split_gain=0.87, n_estimators=100;, score=0.956 total time=   0.0s
[CV 2/5] END boosting_type=gbdt, importance_type=split, learning_rate=0.4,
min_split_gain=0.87, n_estimators=100;, score=0.964 total time=   0.0s
[CV 3/5] END boosting_type=gbdt, importance_type=split, learning_rate=0.4,
min_split_gain=0.87, n_estimators=100;, score=0.957 total time=   0.0s
[CV 4/5] END boosting_type=gbdt, importance_type=split, learning_rate=0.4,
min_split_gain=0.87, n_estimators=100;, score=0.964 total time=   0.0s
[CV 5/5] END boosting_type=gbdt, importance_type=split, learning_rate=0.4,
min_split_gain=0.87, n_estimators=100;, score=0.968 total time=   0.0s
Mean Absolute Percentage Error(MAPE): 0.8123791708530981
Root Mean Squared Error(RMSE): 0.1660202071635469
R2 Score: 0.9730744529902002
```





[108]:
```
grid_lgbm.best_params_
```

[108]:
```
{'n_estimators': 1000,
 'min_split_gain': 0.79,
 'learning_rate': 0.4,
 'importance_type': 'split',
 'boosting_type': 'goss'}
```

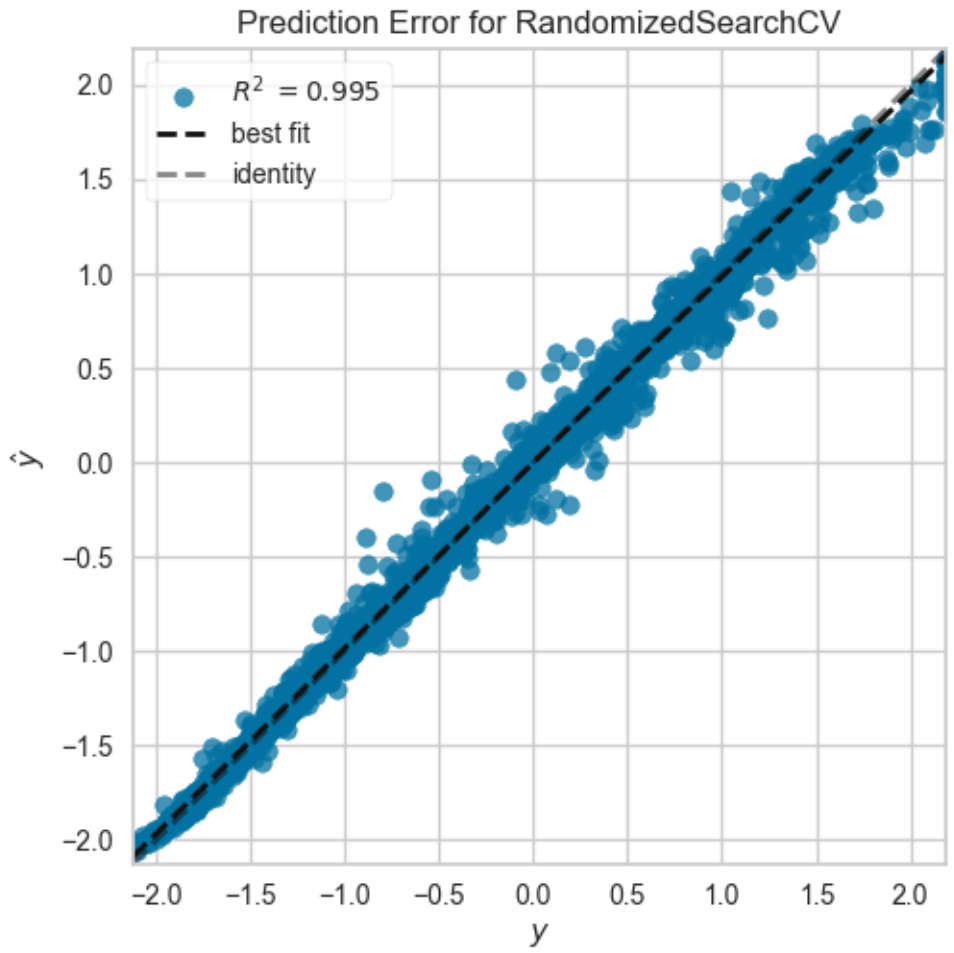
[109]:
```
param_grid = {'n_estimators': [100,300,500,800,1000],
              'criterion': ["squared_error", "absolute_error", "friedman_mse",␣
 ↪"poisson"],
              'max_features': ['auto','sqrt','log2'],
              'bootstrap': [True,False],
              'oob_score': [True,False],
              'max_samples': [0.2,0.45,0.7,0.95]
              }
```

```
grid_et = RandomizedSearchCV(ExtraTreesRegressor(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_et)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
max_samples=0.2, n_estimators=100, oob_score=True; total time=   0.6s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
max_samples=0.2, n_estimators=100, oob_score=True; total time=   0.6s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
max_samples=0.2, n_estimators=100, oob_score=True; total time=   0.6s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
max_samples=0.2, n_estimators=100, oob_score=True; total time=   0.2s
[CV] END bootstrap=True, criterion=squared_error, max_features=log2,
max_samples=0.2, n_estimators=100, oob_score=True; total time=   0.2s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
max_samples=0.45, n_estimators=1000, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
max_samples=0.45, n_estimators=1000, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
max_samples=0.45, n_estimators=1000, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
max_samples=0.45, n_estimators=1000, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=squared_error, max_features=auto,
max_samples=0.45, n_estimators=1000, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=log2,
max_samples=0.95, n_estimators=1000, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=log2,
max_samples=0.95, n_estimators=1000, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=log2,
max_samples=0.95, n_estimators=1000, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=log2,
max_samples=0.95, n_estimators=1000, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=log2,
max_samples=0.95, n_estimators=1000, oob_score=False; total time=   0.0s
[CV] END bootstrap=True, criterion=absolute_error, max_features=auto,
max_samples=0.2, n_estimators=800, oob_score=False; total time=   9.2s
[CV] END bootstrap=True, criterion=absolute_error, max_features=auto,
max_samples=0.2, n_estimators=800, oob_score=False; total time=  10.0s
[CV] END bootstrap=True, criterion=absolute_error, max_features=auto,
max_samples=0.2, n_estimators=800, oob_score=False; total time=   9.2s
[CV] END bootstrap=True, criterion=absolute_error, max_features=auto,
max_samples=0.2, n_estimators=800, oob_score=False; total time=  10.2s
[CV] END bootstrap=True, criterion=absolute_error, max_features=auto,
max_samples=0.2, n_estimators=800, oob_score=False; total time=   9.8s
[CV] END bootstrap=False, criterion=poisson, max_features=auto,
max_samples=0.45, n_estimators=800, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=auto,
max_samples=0.45, n_estimators=800, oob_score=True; total time=   0.0s
```

```
[CV] END bootstrap=False, criterion=poisson, max_features=auto,
max_samples=0.45, n_estimators=800, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=auto,
max_samples=0.45, n_estimators=800, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=poisson, max_features=auto,
max_samples=0.45, n_estimators=800, oob_score=True; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=0.7, n_estimators=800, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=0.7, n_estimators=800, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=0.7, n_estimators=800, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=0.7, n_estimators=800, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=friedman_mse, max_features=sqrt,
max_samples=0.7, n_estimators=800, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
max_samples=0.45, n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
max_samples=0.45, n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
max_samples=0.45, n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
max_samples=0.45, n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=False, criterion=absolute_error, max_features=auto,
max_samples=0.45, n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=True, criterion=squared_error, max_features=auto,
max_samples=0.95, n_estimators=100, oob_score=True; total time=   0.6s
[CV] END bootstrap=True, criterion=squared_error, max_features=auto,
max_samples=0.95, n_estimators=100, oob_score=True; total time=   0.6s
[CV] END bootstrap=True, criterion=squared_error, max_features=auto,
max_samples=0.95, n_estimators=100, oob_score=True; total time=   0.6s
[CV] END bootstrap=True, criterion=squared_error, max_features=auto,
max_samples=0.95, n_estimators=100, oob_score=True; total time=   0.6s
[CV] END bootstrap=True, criterion=squared_error, max_features=auto,
max_samples=0.95, n_estimators=100, oob_score=True; total time=   0.6s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
max_samples=0.45, n_estimators=100, oob_score=True; total time=   0.3s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
max_samples=0.45, n_estimators=100, oob_score=True; total time=   0.3s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
max_samples=0.45, n_estimators=100, oob_score=True; total time=   0.3s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
max_samples=0.45, n_estimators=100, oob_score=True; total time=   0.3s
[CV] END bootstrap=True, criterion=friedman_mse, max_features=log2,
max_samples=0.45, n_estimators=100, oob_score=True; total time=   0.3s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, max_samples=0.95,
n_estimators=500, oob_score=False; total time=   0.0s
```

```
[CV] END bootstrap=True, criterion=poisson, max_features=log2, max_samples=0.95,
n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, max_samples=0.95,
n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, max_samples=0.95,
n_estimators=500, oob_score=False; total time=   0.0s
[CV] END bootstrap=True, criterion=poisson, max_features=log2, max_samples=0.95,
n_estimators=500, oob_score=False; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 0.6428374445512449
Root Mean Squared Error(RMSE): 0.17914616108779607
R2 Score: 0.9686485490822321
```



[110]:
```
grid_et.best_params_
```

[110]:
```
{'oob_score': True,
 'n_estimators': 100,
 'max_samples': 0.95,
 'max_features': 'auto',
 'criterion': 'squared_error',
 'bootstrap': True}
```

[111]:
```
param_grid = {'n_estimators': [100,400,700,900,1000],
              'grow_policy': [0,1],
              'learning_rate': [0.1,0.4,0.6,0.8,1],
              'booster': ['gbtree','gblinear','dart'],
              'sampling_method': ['uniform','gradient_based'],
              'importance_type':␣
 ↪['gain','weight','cover','total_gain','total_cover']
              }
```

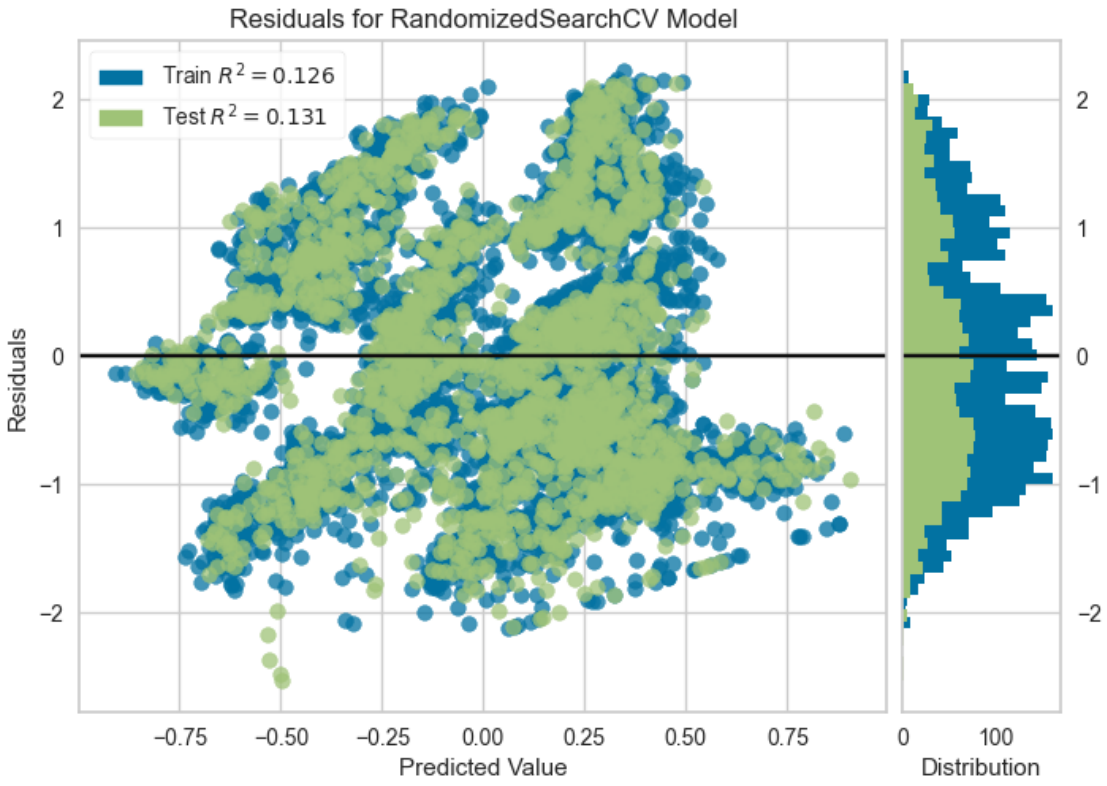
```python
grid_xgb = RandomizedSearchCV(XGBRegressor(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_xgb)
```

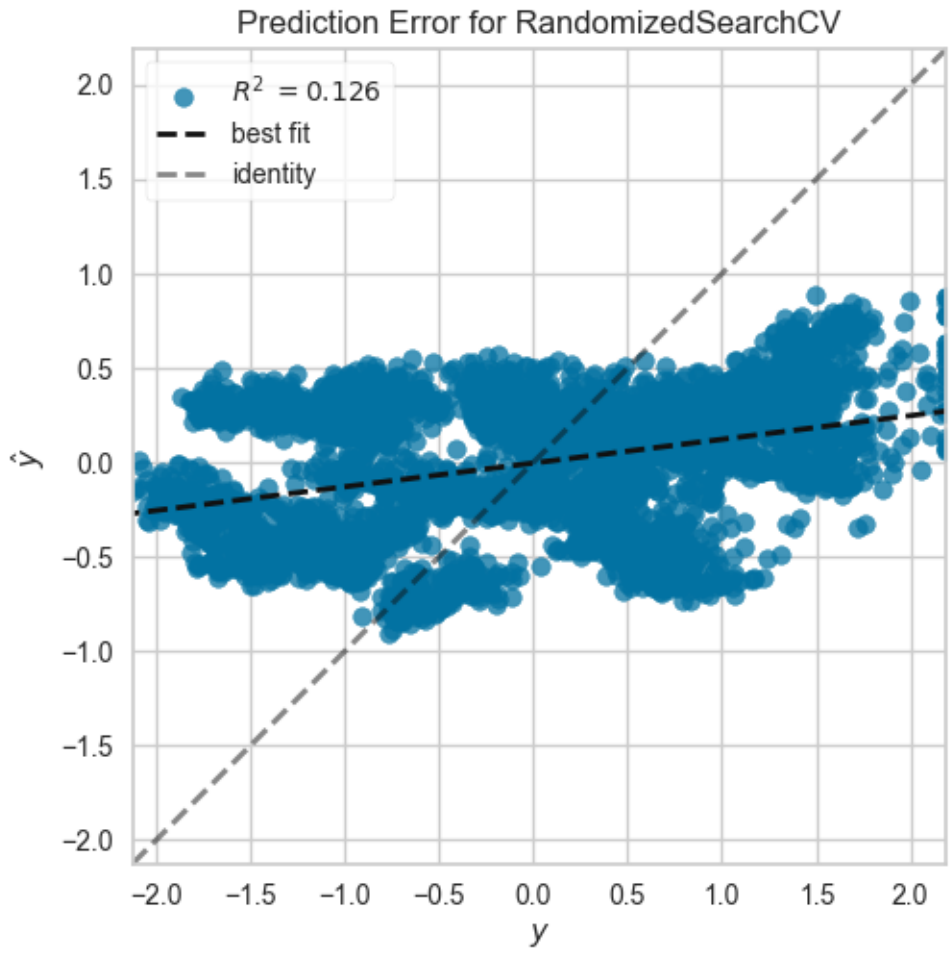
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END booster=gbtree, grow_policy=0, importance_type=gain, learning_rate=1,
n_estimators=700, sampling_method=uniform; total time=   0.0s
[CV] END booster=gbtree, grow_policy=0, importance_type=gain, learning_rate=1,
n_estimators=700, sampling_method=uniform; total time=   0.0s
[CV] END booster=gbtree, grow_policy=0, importance_type=gain, learning_rate=1,
n_estimators=700, sampling_method=uniform; total time=   0.0s
[CV] END booster=gbtree, grow_policy=0, importance_type=gain, learning_rate=1,
n_estimators=700, sampling_method=uniform; total time=   0.0s
[CV] END booster=gbtree, grow_policy=0, importance_type=gain, learning_rate=1,
n_estimators=700, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=0, importance_type=gain, learning_rate=0.6,
n_estimators=700, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=dart, grow_policy=0, importance_type=gain, learning_rate=0.6,
n_estimators=700, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=dart, grow_policy=0, importance_type=gain, learning_rate=0.6,
n_estimators=700, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=dart, grow_policy=0, importance_type=gain, learning_rate=0.6,
n_estimators=700, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=dart, grow_policy=0, importance_type=gain, learning_rate=0.6,
n_estimators=700, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=gbtree, grow_policy=1, importance_type=gain, learning_rate=0.6,
n_estimators=1000, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=gbtree, grow_policy=1, importance_type=gain, learning_rate=0.6,
n_estimators=1000, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=gbtree, grow_policy=1, importance_type=gain, learning_rate=0.6,
n_estimators=1000, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=gbtree, grow_policy=1, importance_type=gain, learning_rate=0.6,
n_estimators=1000, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=gbtree, grow_policy=1, importance_type=gain, learning_rate=0.6,
n_estimators=1000, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=gbtree, grow_policy=0, importance_type=weight, learning_rate=1,
n_estimators=100, sampling_method=uniform; total time=   0.0s
[CV] END booster=gbtree, grow_policy=0, importance_type=weight, learning_rate=1,
n_estimators=100, sampling_method=uniform; total time=   0.0s
[CV] END booster=gbtree, grow_policy=0, importance_type=weight, learning_rate=1,
n_estimators=100, sampling_method=uniform; total time=   0.0s
[CV] END booster=gbtree, grow_policy=0, importance_type=weight, learning_rate=1,
n_estimators=100, sampling_method=uniform; total time=   0.0s
[CV] END booster=gbtree, grow_policy=0, importance_type=weight, learning_rate=1,
n_estimators=100, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=total_cover,
learning_rate=0.1, n_estimators=700, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=total_cover,
```

```
learning_rate=0.1, n_estimators=700, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=total_cover,
learning_rate=0.1, n_estimators=700, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=total_cover,
learning_rate=0.1, n_estimators=700, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=total_cover,
learning_rate=0.1, n_estimators=700, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=total_gain,
learning_rate=1, n_estimators=900, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=total_gain,
learning_rate=1, n_estimators=900, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=total_gain,
learning_rate=1, n_estimators=900, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=total_gain,
learning_rate=1, n_estimators=900, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=total_gain,
learning_rate=1, n_estimators=900, sampling_method=uniform; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=gain, learning_rate=0.6,
n_estimators=1000, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=gain, learning_rate=0.6,
n_estimators=1000, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=gain, learning_rate=0.6,
n_estimators=1000, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=gain, learning_rate=0.6,
n_estimators=1000, sampling_method=gradient_based; total time=   0.0s
[CV] END booster=dart, grow_policy=1, importance_type=gain, learning_rate=0.6,
n_estimators=1000, sampling_method=gradient_based; total time=   0.0s
[01:02:53] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.8, n_estimators=100, sampling_method=uniform; total time=   0.0s
[01:02:53] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.8, n_estimators=100, sampling_method=uniform; total time=   0.0s
[01:02:53] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.8, n_estimators=100, sampling_method=uniform; total time=   0.0s
[01:02:53] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.
```

```
[CV] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.8, n_estimators=100, sampling_method=uniform; total time=   0.0s
[01:02:53] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV] END booster=gblinear, grow_policy=1, importance_type=gain,
learning_rate=0.8, n_estimators=100, sampling_method=uniform; total time=   0.0s
[01:02:53] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV] END booster=gblinear, grow_policy=0, importance_type=cover,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform; total time=
0.1s
[01:02:54] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV] END booster=gblinear, grow_policy=0, importance_type=cover,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform; total time=
0.1s
[01:02:54] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV] END booster=gblinear, grow_policy=0, importance_type=cover,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform; total time=
0.1s
[01:02:54] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV] END booster=gblinear, grow_policy=0, importance_type=cover,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform; total time=
0.1s
[01:02:54] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

[CV] END booster=gblinear, grow_policy=0, importance_type=cover,
learning_rate=0.4, n_estimators=1000, sampling_method=uniform; total time=
0.1s
[CV] END booster=gbtree, grow_policy=1, importance_type=weight,
learning_rate=0.6, n_estimators=400, sampling_method=gradient_based; total time=
0.0s
[CV] END booster=gbtree, grow_policy=1, importance_type=weight,
```

```
learning_rate=0.6, n_estimators=400, sampling_method=gradient_based; total time=
0.0s
[CV] END booster=gbtree, grow_policy=1, importance_type=weight,
learning_rate=0.6, n_estimators=400, sampling_method=gradient_based; total time=
0.0s
[CV] END booster=gbtree, grow_policy=1, importance_type=weight,
learning_rate=0.6, n_estimators=400, sampling_method=gradient_based; total time=
0.0s
[CV] END booster=gbtree, grow_policy=1, importance_type=weight,
learning_rate=0.6, n_estimators=400, sampling_method=gradient_based; total time=
0.0s
[01:02:54] WARNING: C:\Users\dev-admin\croot2\xgboost-
split_1675461376218\work\src\learner.cc:767:
Parameters: { "grow_policy", "sampling_method" } are not used.

Mean Absolute Percentage Error(MAPE): 1.8642096254763194
Root Mean Squared Error(RMSE): 0.9429977568683355
R2 Score: 0.1313117414852658
```





[112]:
```
grid_xgb.best_params_
```

[112]:
```
{'sampling_method': 'uniform',
 'n_estimators': 100,
 'learning_rate': 0.8,
 'importance_type': 'gain',
 'grow_policy': 1,
 'booster': 'gblinear'}
```

[113]:
```
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),
```

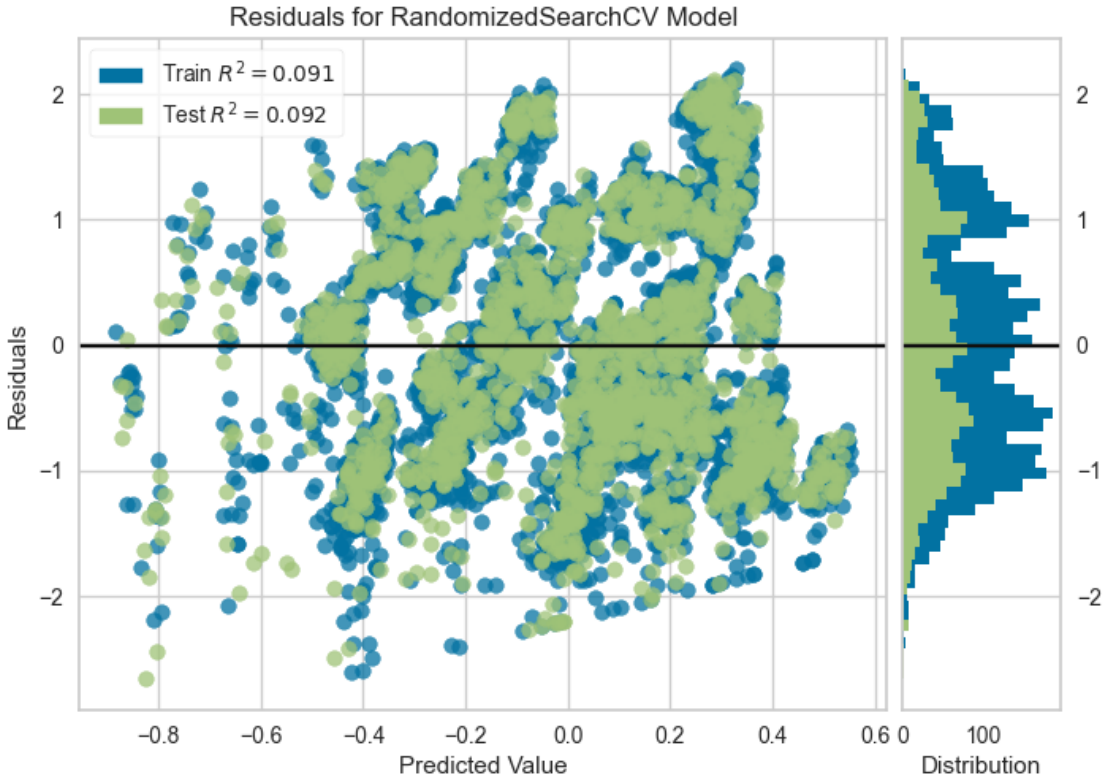
```
             'average': [True,False]
            }

grid_sgd = RandomizedSearchCV(SGDRegressor(),param_grid,verbose=2,cv=5)
train_and_evaluate_model(grid_sgd)
```

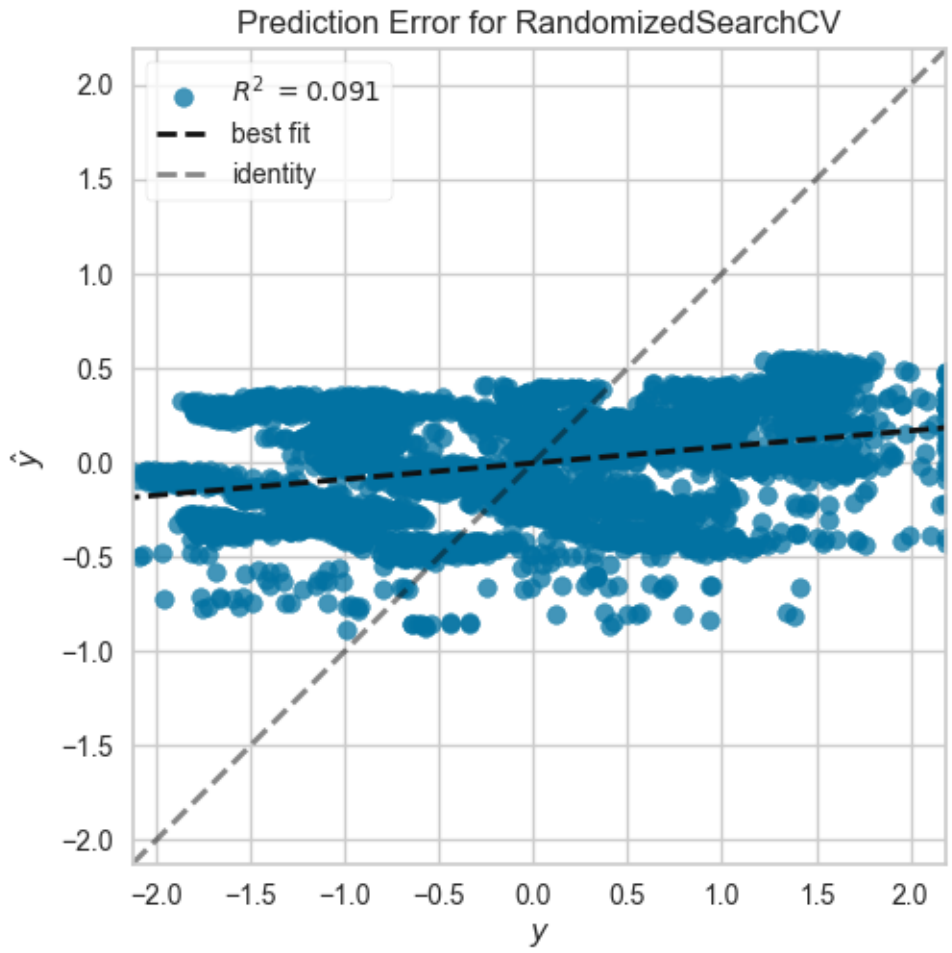
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END alpha=1, average=False, epsilon=11.111999999999998, l1_ratio=0.15,
learning_rate=adaptive, loss=squared_error, penalty=l2, shuffle=True; total
time=   0.0s
[CV] END alpha=1, average=False, epsilon=11.111999999999998, l1_ratio=0.15,
learning_rate=adaptive, loss=squared_error, penalty=l2, shuffle=True; total
time=   0.0s
[CV] END alpha=1, average=False, epsilon=11.111999999999998, l1_ratio=0.15,
learning_rate=adaptive, loss=squared_error, penalty=l2, shuffle=True; total
time=   0.0s
[CV] END alpha=1, average=False, epsilon=11.111999999999998, l1_ratio=0.15,
learning_rate=adaptive, loss=squared_error, penalty=l2, shuffle=True; total
time=   0.0s
[CV] END alpha=1, average=False, epsilon=11.111999999999998, l1_ratio=0.15,
learning_rate=adaptive, loss=squared_error, penalty=l2, shuffle=True; total
time=   0.0s
[CV] END alpha=1, average=False, epsilon=11.111999999999998, l1_ratio=0.97,
learning_rate=optimal, loss=huber, penalty=l2, shuffle=False; total time=   0.0s
[CV] END alpha=1, average=False, epsilon=11.111999999999998, l1_ratio=0.97,
learning_rate=optimal, loss=huber, penalty=l2, shuffle=False; total time=   0.0s
[CV] END alpha=1, average=False, epsilon=11.111999999999998, l1_ratio=0.97,
learning_rate=optimal, loss=huber, penalty=l2, shuffle=False; total time=   0.0s
[CV] END alpha=1, average=False, epsilon=11.111999999999998, l1_ratio=0.97,
learning_rate=optimal, loss=huber, penalty=l2, shuffle=False; total time=   0.0s
[CV] END alpha=1, average=False, epsilon=11.111999999999998, l1_ratio=0.97,
learning_rate=optimal, loss=huber, penalty=l2, shuffle=False; total time=   0.0s
[CV] END alpha=1, average=False, epsilon=100.0, l1_ratio=0.97,
learning_rate=constant, loss=squared_epsilon_insensitive, penalty=l1,
shuffle=False; total time=   0.0s
[CV] END alpha=1, average=False, epsilon=100.0, l1_ratio=0.97,
learning_rate=constant, loss=squared_epsilon_insensitive, penalty=l1,
shuffle=False; total time=   0.0s
[CV] END alpha=1, average=False, epsilon=100.0, l1_ratio=0.97,
learning_rate=constant, loss=squared_epsilon_insensitive, penalty=l1,
shuffle=False; total time=   0.0s
[CV] END alpha=1, average=False, epsilon=100.0, l1_ratio=0.97,
learning_rate=constant, loss=squared_epsilon_insensitive, penalty=l1,
shuffle=False; total time=   0.0s
[CV] END alpha=1, average=False, epsilon=100.0, l1_ratio=0.97,
learning_rate=constant, loss=squared_epsilon_insensitive, penalty=l1,
shuffle=False; total time=   0.0s
[CV] END alpha=0.01, average=False, epsilon=22.223, l1_ratio=0.15,
```

```
learning_rate=optimal, loss=squared_error, penalty=elasticnet, shuffle=False;
total time=   0.4s
[CV] END alpha=0.01, average=False, epsilon=22.223, l1_ratio=0.15,
learning_rate=optimal, loss=squared_error, penalty=elasticnet, shuffle=False;
total time=   0.2s
[CV] END alpha=0.01, average=False, epsilon=22.223, l1_ratio=0.15,
learning_rate=optimal, loss=squared_error, penalty=elasticnet, shuffle=False;
total time=   0.1s
[CV] END alpha=0.01, average=False, epsilon=22.223, l1_ratio=0.15,
learning_rate=optimal, loss=squared_error, penalty=elasticnet, shuffle=False;
total time=   0.1s
[CV] END alpha=0.01, average=False, epsilon=22.223, l1_ratio=0.15,
learning_rate=optimal, loss=squared_error, penalty=elasticnet, shuffle=False;
total time=   0.1s
[CV] END alpha=0.0001, average=False, epsilon=0.001, l1_ratio=0.81,
learning_rate=constant, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=True; total time=   0.0s
[CV] END alpha=0.0001, average=False, epsilon=0.001, l1_ratio=0.81,
learning_rate=constant, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=True; total time=   0.0s
[CV] END alpha=0.0001, average=False, epsilon=0.001, l1_ratio=0.81,
learning_rate=constant, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=True; total time=   0.0s
[CV] END alpha=0.0001, average=False, epsilon=0.001, l1_ratio=0.81,
learning_rate=constant, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=True; total time=   0.0s
[CV] END alpha=0.0001, average=False, epsilon=0.001, l1_ratio=0.81,
learning_rate=constant, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=True; total time=   0.0s
[CV] END alpha=0.0001, average=True, epsilon=100.0, l1_ratio=0.45,
learning_rate=invscaling, loss=squared_epsilon_insensitive, penalty=l2,
shuffle=False; total time=   0.0s
[CV] END alpha=0.0001, average=True, epsilon=100.0, l1_ratio=0.45,
learning_rate=invscaling, loss=squared_epsilon_insensitive, penalty=l2,
shuffle=False; total time=   0.0s
[CV] END alpha=0.0001, average=True, epsilon=100.0, l1_ratio=0.45,
learning_rate=invscaling, loss=squared_epsilon_insensitive, penalty=l2,
shuffle=False; total time=   0.0s
[CV] END alpha=0.0001, average=True, epsilon=100.0, l1_ratio=0.45,
learning_rate=invscaling, loss=squared_epsilon_insensitive, penalty=l2,
shuffle=False; total time=   0.0s
[CV] END alpha=0.0001, average=True, epsilon=100.0, l1_ratio=0.45,
learning_rate=invscaling, loss=squared_epsilon_insensitive, penalty=l2,
shuffle=False; total time=   0.0s
[CV] END alpha=1, average=True, epsilon=88.889, l1_ratio=0.81,
learning_rate=invscaling, loss=epsilon_insensitive, penalty=l1, shuffle=True;
total time=   0.0s
[CV] END alpha=1, average=True, epsilon=88.889, l1_ratio=0.81,
```

```
learning_rate=invscaling, loss=epsilon_insensitive, penalty=l1, shuffle=True;
total time=   0.0s
[CV] END alpha=1, average=True, epsilon=88.889, l1_ratio=0.81,
learning_rate=invscaling, loss=epsilon_insensitive, penalty=l1, shuffle=True;
total time=   0.0s
[CV] END alpha=1, average=True, epsilon=88.889, l1_ratio=0.81,
learning_rate=invscaling, loss=epsilon_insensitive, penalty=l1, shuffle=True;
total time=   0.0s
[CV] END alpha=1, average=True, epsilon=88.889, l1_ratio=0.81,
learning_rate=invscaling, loss=epsilon_insensitive, penalty=l1, shuffle=True;
total time=   0.0s
[CV] END alpha=0.1, average=True, epsilon=22.223, l1_ratio=0.68,
learning_rate=adaptive, loss=huber, penalty=l1, shuffle=False; total time=
0.0s
[CV] END alpha=0.1, average=True, epsilon=22.223, l1_ratio=0.68,
learning_rate=adaptive, loss=huber, penalty=l1, shuffle=False; total time=
0.0s
[CV] END alpha=0.1, average=True, epsilon=22.223, l1_ratio=0.68,
learning_rate=adaptive, loss=huber, penalty=l1, shuffle=False; total time=
0.0s
[CV] END alpha=0.1, average=True, epsilon=22.223, l1_ratio=0.68,
learning_rate=adaptive, loss=huber, penalty=l1, shuffle=False; total time=
0.0s
[CV] END alpha=0.1, average=True, epsilon=22.223, l1_ratio=0.68,
learning_rate=adaptive, loss=huber, penalty=l1, shuffle=False; total time=
0.0s
[CV] END alpha=0.001, average=True, epsilon=66.667, l1_ratio=0.15,
learning_rate=adaptive, loss=squared_error, penalty=l1, shuffle=False; total
time=   0.0s
[CV] END alpha=0.001, average=True, epsilon=66.667, l1_ratio=0.15,
learning_rate=adaptive, loss=squared_error, penalty=l1, shuffle=False; total
time=   0.0s
[CV] END alpha=0.001, average=True, epsilon=66.667, l1_ratio=0.15,
learning_rate=adaptive, loss=squared_error, penalty=l1, shuffle=False; total
time=   0.0s
[CV] END alpha=0.001, average=True, epsilon=66.667, l1_ratio=0.15,
learning_rate=adaptive, loss=squared_error, penalty=l1, shuffle=False; total
time=   0.0s
[CV] END alpha=0.001, average=True, epsilon=66.667, l1_ratio=0.15,
learning_rate=adaptive, loss=squared_error, penalty=l1, shuffle=False; total
time=   0.0s
[CV] END alpha=0.1, average=True, epsilon=88.889, l1_ratio=0.97,
learning_rate=optimal, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=False; total time=   0.0s
[CV] END alpha=0.1, average=True, epsilon=88.889, l1_ratio=0.97,
learning_rate=optimal, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=False; total time=   0.0s
[CV] END alpha=0.1, average=True, epsilon=88.889, l1_ratio=0.97,
```

```
learning_rate=optimal, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=False; total time=   0.0s
[CV] END alpha=0.1, average=True, epsilon=88.889, l1_ratio=0.97,
learning_rate=optimal, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=False; total time=   0.0s
[CV] END alpha=0.1, average=True, epsilon=88.889, l1_ratio=0.97,
learning_rate=optimal, loss=squared_epsilon_insensitive, penalty=elasticnet,
shuffle=False; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 2.2562796873375857
Root Mean Squared Error(RMSE): 0.964216413840828
R2 Score: 0.09177872177690394
```





[114]:
```
grid_sgd.best_params_
```

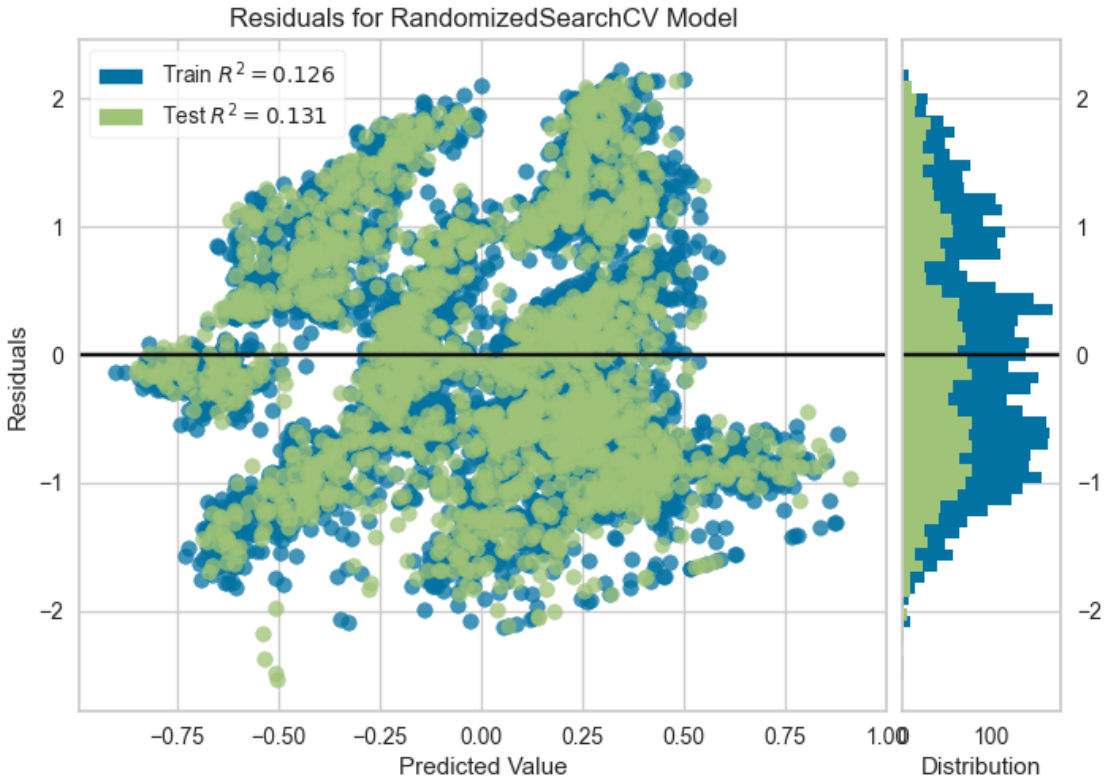
[114]:
```
{'shuffle': False,
 'penalty': 'l1',
 'loss': 'squared_error',
 'learning_rate': 'adaptive',
 'l1_ratio': 0.15,
 'epsilon': 66.667,
 'average': True,
 'alpha': 0.001}
```

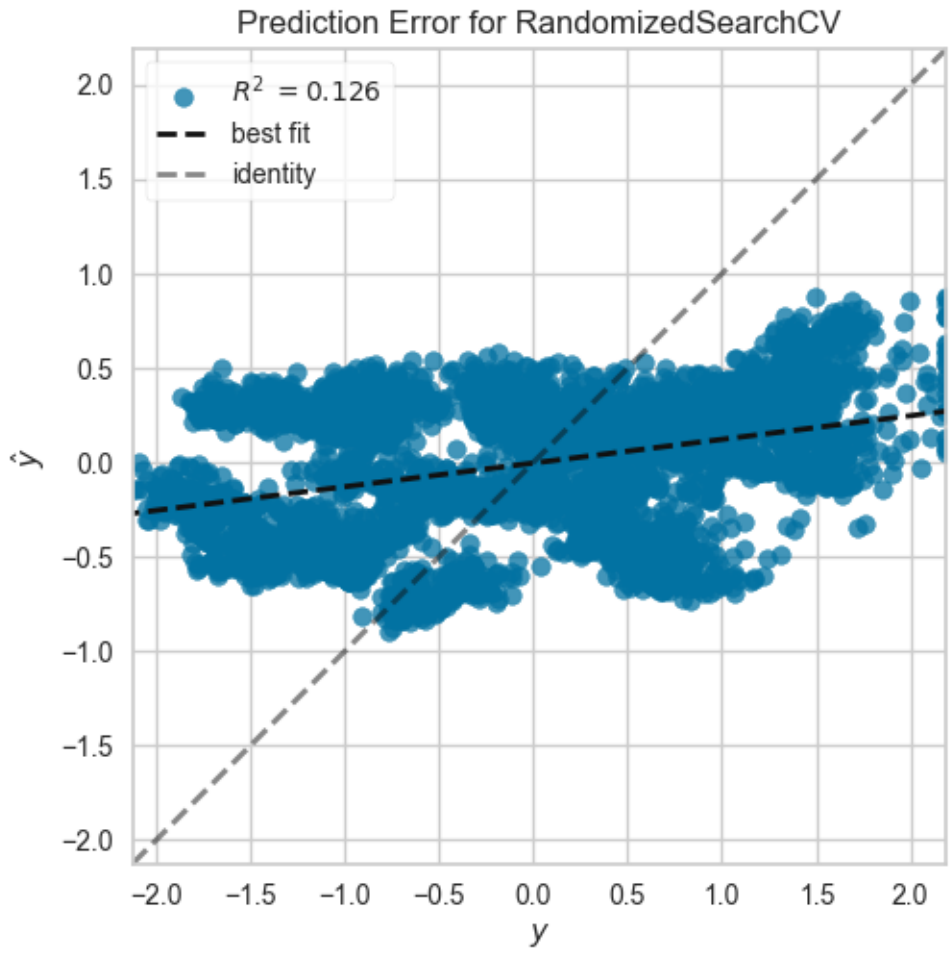
[115]:
```
param_grid = {'epsilon': np.linspace(1,10,10),
              'alpha': np.linspace(0.0001,10,10)}

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=6.6667, epsilon=3.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=3.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=3.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=3.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=3.0; 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=10.0, epsilon=5.0; total time=   0.0s
[CV] END …alpha=10.0, epsilon=5.0; total time=   0.0s
[CV] END …alpha=10.0, epsilon=5.0; total time=   0.0s
[CV] END …alpha=10.0, epsilon=5.0; total time=   0.0s
[CV] END …alpha=10.0, epsilon=5.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=5.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=5.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=5.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=5.0; total time=   0.0s
[CV] END …alpha=6.6667, epsilon=5.0; 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
[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=4.4445, epsilon=6.0; total time=   0.0s
[CV] END …alpha=4.4445, epsilon=6.0; total time=   0.0s
[CV] END …alpha=4.4445, epsilon=6.0; total time=   0.0s
[CV] END …alpha=4.4445, epsilon=6.0; total time=   0.0s
[CV] END …alpha=4.4445, epsilon=6.0; total time=   0.0s
[CV] END …alpha=1.1112, epsilon=4.0; total time=   0.0s
[CV] END …alpha=1.1112, epsilon=4.0; total time=   0.0s
[CV] END …alpha=1.1112, epsilon=4.0; total time=   0.0s
[CV] END …alpha=1.1112, epsilon=4.0; total time=   0.0s
[CV] END …alpha=1.1112, epsilon=4.0; 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=3.3334, epsilon=10.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=10.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=10.0; total time=   0.0s
```

```
[CV] END …alpha=3.3334, epsilon=10.0; total time=   0.0s
[CV] END …alpha=3.3334, epsilon=10.0; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 1.8583990965205004
Root Mean Squared Error(RMSE): 0.9429069446854338
R2 Score: 0.1314790455492416
```





[116]:
```
grid_huber.best_params_
```

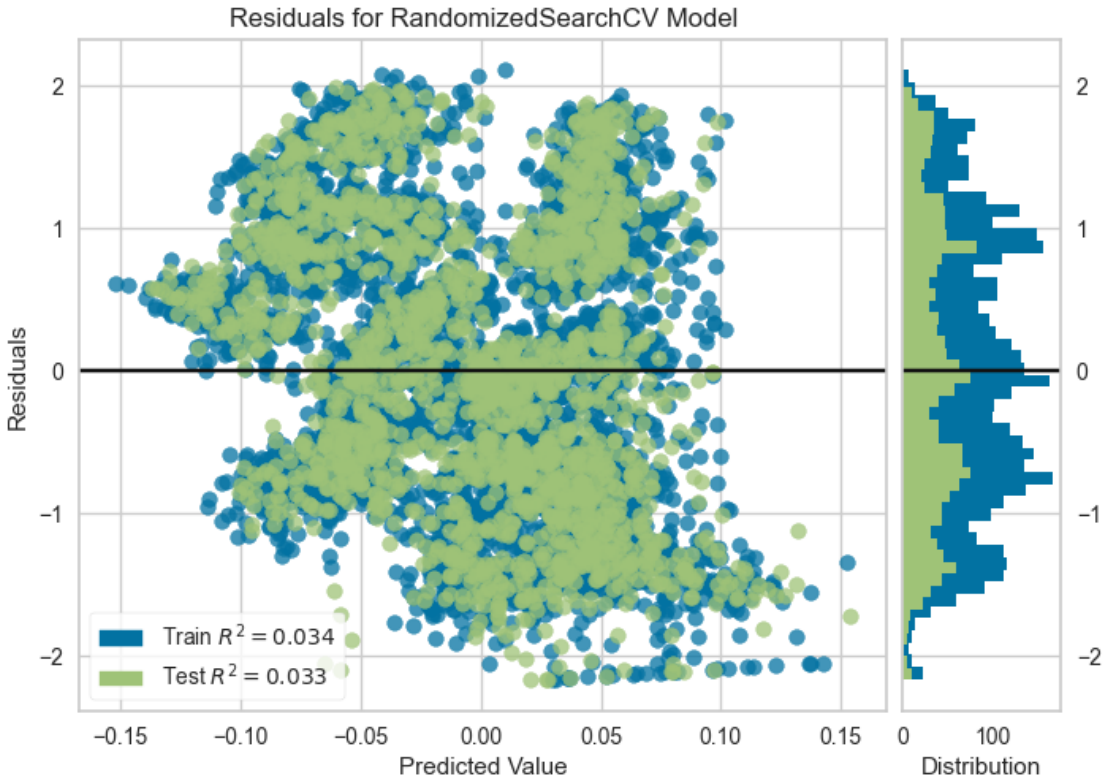
[116]:
```
{'epsilon': 5.0, 'alpha': 10.0}
```

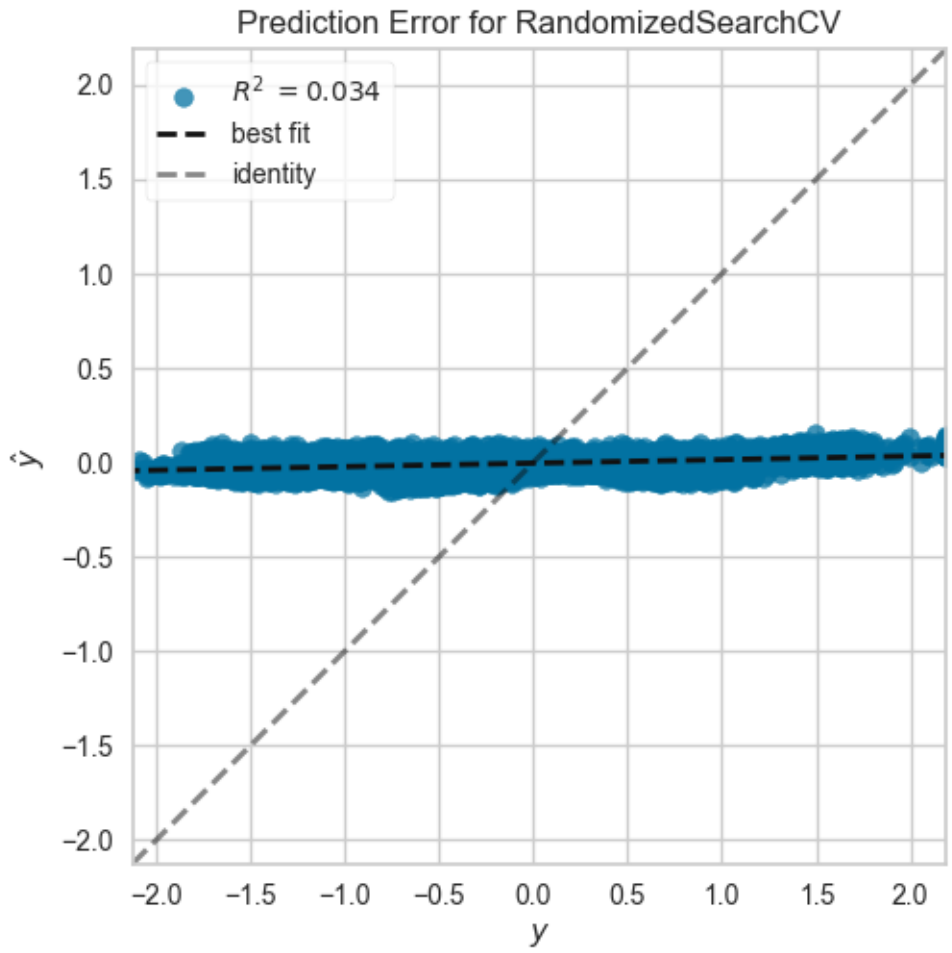
[117]:
```
param_grid = {'link': ['auto', 'identity', 'log'],
              'solver': ['lbfgs', 'newton-cholesky'],
              'alpha': np.linspace(0.0001,10,10)}

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=10.0, link=identity, solver=lbfgs; total time=   0.0s
[CV] END …alpha=10.0, link=identity, solver=lbfgs; total time=   0.0s
[CV] END …alpha=10.0, link=identity, solver=lbfgs; total time=   0.0s
[CV] END …alpha=10.0, link=identity, solver=lbfgs; total time=   0.0s
[CV] END …alpha=10.0, link=identity, solver=lbfgs; total time=   0.0s
[CV] END …alpha=1.1112, link=log, solver=lbfgs; total time=   0.0s
```

```
[CV] END …alpha=1.1112, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=1.1112, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=1.1112, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=1.1112, link=log, solver=lbfgs; 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=5.5556, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=5.5556, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=5.5556, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=5.5556, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=5.5556, link=identity, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=6.6667, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=6.6667, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=6.6667, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=6.6667, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=6.6667, link=log, solver=lbfgs; total time=   0.0s
[CV] END …alpha=6.6667, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=6.6667, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=6.6667, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=6.6667, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=6.6667, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=6.6667, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=6.6667, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=6.6667, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=6.6667, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END …alpha=6.6667, link=log, solver=newton-cholesky; total time=   0.0s
[CV] END alpha=7.777799999999999, link=identity, solver=lbfgs; total time=
0.0s
[CV] END alpha=7.777799999999999, link=identity, solver=lbfgs; total time=
0.0s
[CV] END alpha=7.777799999999999, link=identity, solver=lbfgs; total time=
0.0s
[CV] END alpha=7.777799999999999, link=identity, solver=lbfgs; total time=
0.0s
[CV] END alpha=7.777799999999999, link=identity, solver=lbfgs; total time=
0.0s
[CV] END …alpha=7.777799999999999, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=7.777799999999999, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=7.777799999999999, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=7.777799999999999, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=7.777799999999999, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=8.8889, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=8.8889, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=8.8889, link=auto, solver=lbfgs; total time=   0.0s
[CV] END …alpha=8.8889, link=auto, solver=lbfgs; total time=   0.0s
```

```
[CV] END …alpha=8.8889, link=auto, solver=lbfgs; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 1.0537279069678445
Root Mean Squared Error(RMSE): 0.9946842583367421
R2 Score: 0.03347493298077597
```





[118]:
```
grid_tweedie.best_params_
```

[118]:
```
{'solver': 'newton-cholesky', 'link': 'identity', 'alpha': 5.5556}
```

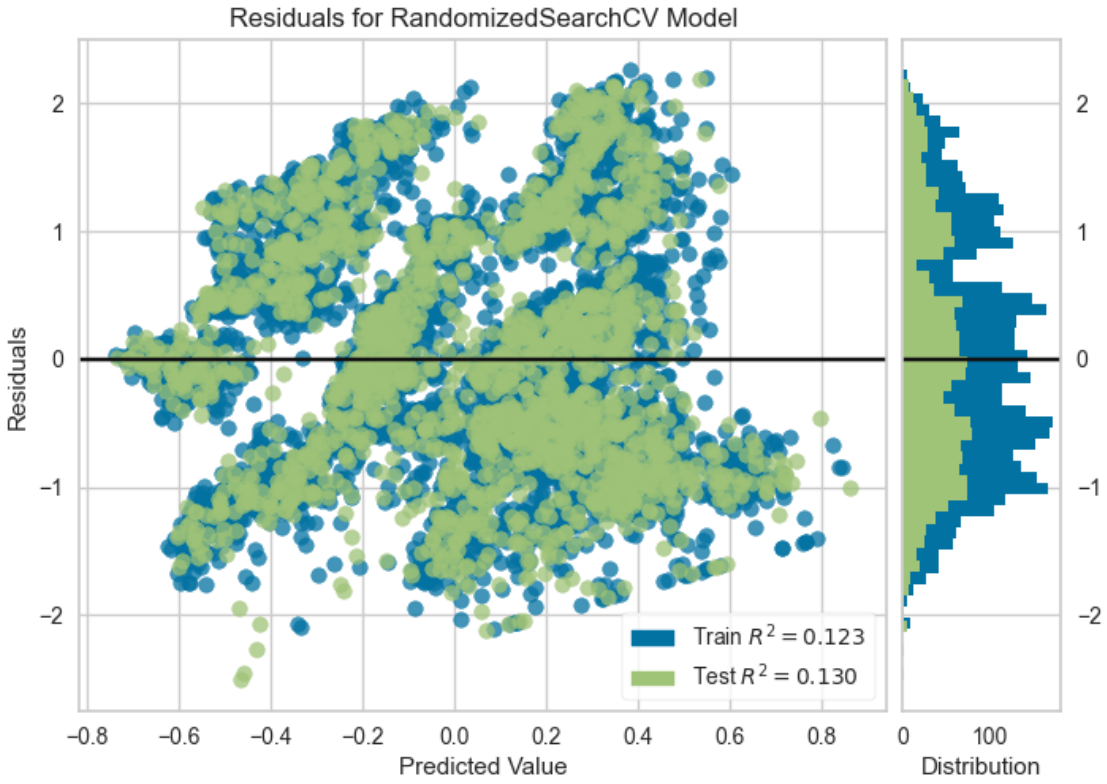
[119]:
```
param_grid = {'C': [0.0001,0.001,0.01,0.1,1,10],
              'loss': ['epsilon_insensitive','squared_epsilon_insensitive'],
              'epsilon': np.linspace(0.001,1,5),
              'shuffle': [True,False],
              'average': [True,False]}

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

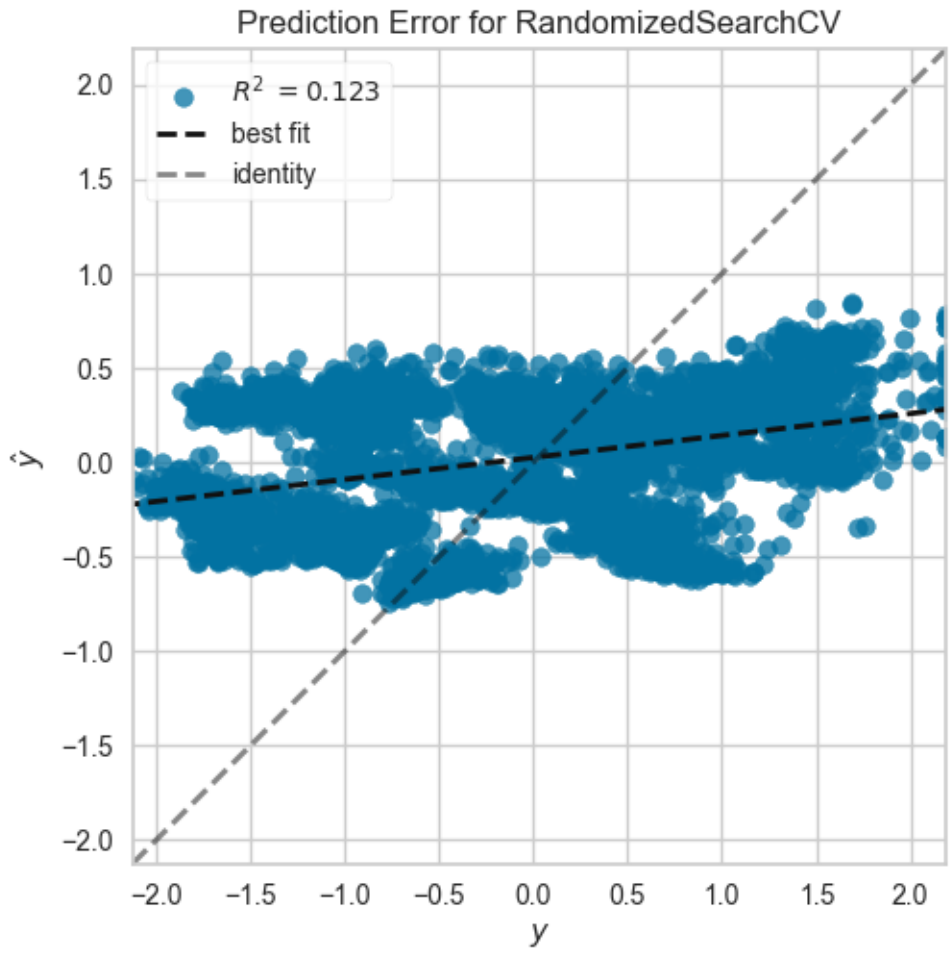
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END C=10, average=False, epsilon=0.75025, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
```

```
[CV] END C=10, average=False, epsilon=0.75025, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.75025, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.75025, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.75025, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=0.1, average=True, epsilon=0.25075, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.1, average=True, epsilon=0.25075, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.1, average=True, epsilon=0.25075, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.1, average=True, epsilon=0.25075, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.1, average=True, epsilon=0.25075, loss=squared_epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.0001, average=True, epsilon=0.001, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.0001, average=True, epsilon=0.001, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.0001, average=True, epsilon=0.001, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.0001, average=True, epsilon=0.001, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.0001, average=True, epsilon=0.001, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=1, average=True, epsilon=0.001, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=True, epsilon=0.001, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=True, epsilon=0.001, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=True, epsilon=0.001, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=1, average=True, epsilon=0.001, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.001, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.001, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.001, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.001, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=10, average=False, epsilon=0.001, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
```

```
[CV] END C=0.001, average=True, epsilon=0.5005,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.5005,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.5005,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.5005,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.001, average=True, epsilon=0.5005,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.01, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.01, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.01, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.01, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.01, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.0001, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.0001, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.0001, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.0001, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.0001, average=False, epsilon=1.0, loss=epsilon_insensitive,
shuffle=False; total time=   0.0s
[CV] END C=0.1, average=False, epsilon=0.75025,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.1, average=False, epsilon=0.75025,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.1, average=False, epsilon=0.75025,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.1, average=False, epsilon=0.75025,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=0.1, average=False, epsilon=0.75025,
loss=squared_epsilon_insensitive, shuffle=False; total time=   0.0s
[CV] END C=10, average=True, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=True, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=True, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
[CV] END C=10, average=True, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
```

```
[CV] END C=10, average=True, epsilon=1.0, loss=epsilon_insensitive,
shuffle=True; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 2.0718371806563853
Root Mean Squared Error(RMSE): 0.9436517054856786
R2 Score: 0.13010649055828494
```





[120]:
```
grid_pa.best_params_
```

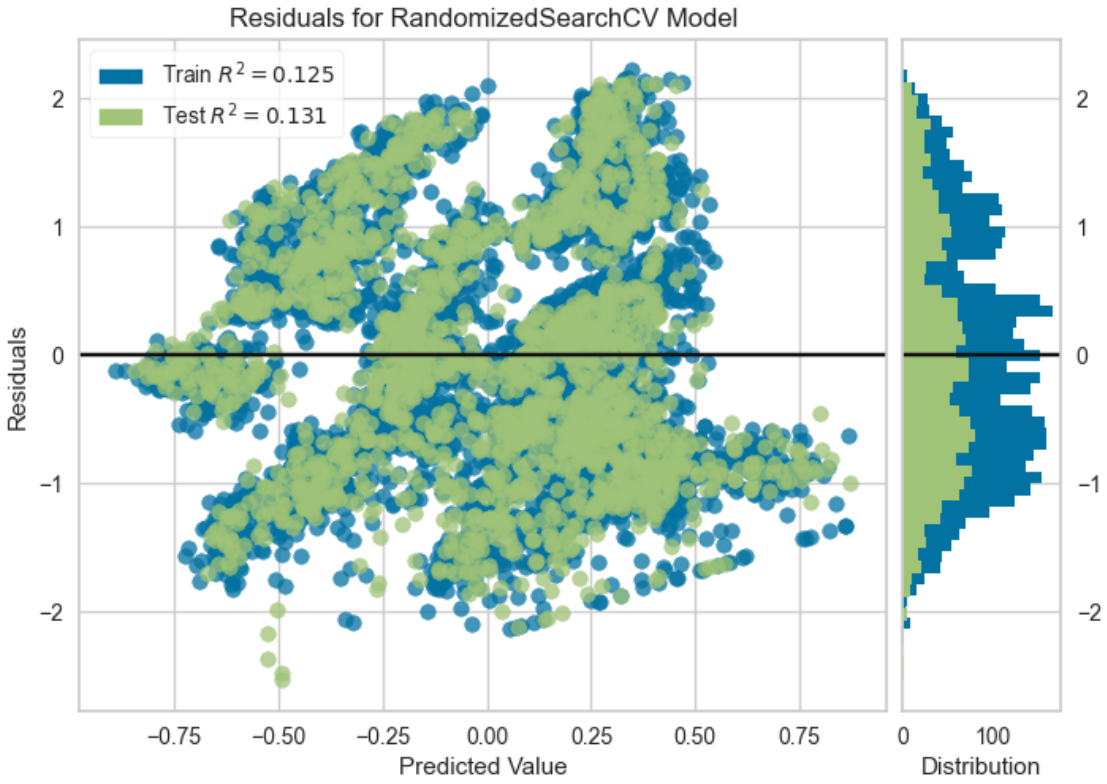
[120]:
```
{'shuffle': False,
 'loss': 'squared_epsilon_insensitive',
 'epsilon': 0.25075,
 'average': True,
 'C': 0.1}
```

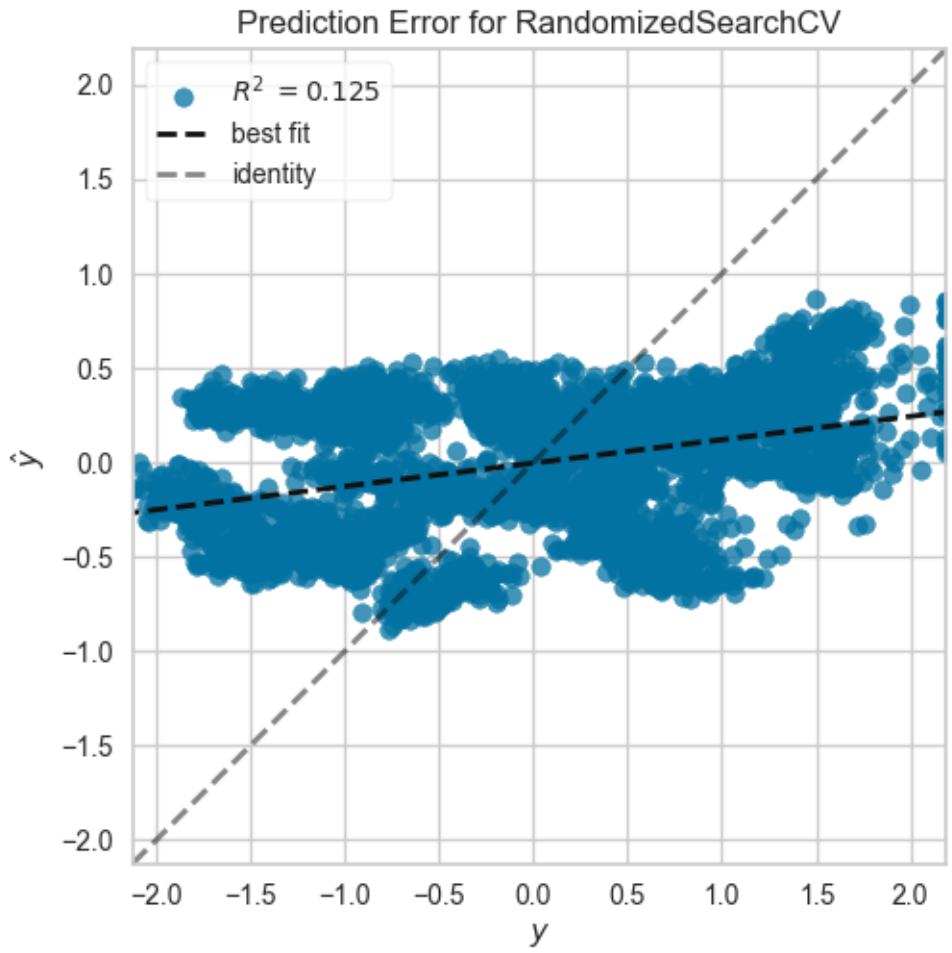
[121]:
```
param_grid = {
    'eps': [0.0001,0.001,0.01,0.1,1],
    'positive': [True,False],
    'selection': ['cyclic','random']
}

grid_lasso = RandomizedSearchCV(LassoCV(),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 …eps=0.1, positive=False, selection=random; total time=   0.0s
[CV] END …eps=0.1, positive=False, selection=random; total time=   0.0s
[CV] END …eps=0.1, positive=False, selection=random; total time=   0.0s
[CV] END …eps=0.1, positive=False, selection=random; total time=   0.0s
[CV] END …eps=0.1, positive=False, selection=random; total time=   0.0s
[CV] END …eps=0.1, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.1, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.1, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.1, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.1, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.0001, positive=False, selection=cyclic; total time=   0.0s
[CV] END …eps=0.0001, positive=False, selection=cyclic; total time=   0.0s
[CV] END …eps=0.0001, positive=False, selection=cyclic; total time=   0.0s
[CV] END …eps=0.0001, positive=False, selection=cyclic; total time=   0.0s
[CV] END …eps=0.0001, positive=False, selection=cyclic; total time=   0.1s
[CV] END …eps=0.0001, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.0001, positive=True, selection=random; total time=   0.1s
[CV] END …eps=0.0001, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.0001, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.0001, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.001, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.001, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.001, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.001, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.001, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.01, positive=False, selection=random; total time=   0.0s
[CV] END …eps=0.01, positive=False, selection=random; total time=   0.0s
[CV] END …eps=0.01, positive=False, selection=random; total time=   0.0s
[CV] END …eps=0.01, positive=False, selection=random; total time=   0.0s
[CV] END …eps=0.01, positive=False, selection=random; total time=   0.0s
[CV] END …eps=1, positive=True, selection=cyclic; total time=   0.0s
[CV] END …eps=1, positive=True, selection=cyclic; total time=   0.0s
[CV] END …eps=1, positive=True, selection=cyclic; total time=   0.0s
[CV] END …eps=1, positive=True, selection=cyclic; total time=   0.0s
[CV] END …eps=1, positive=True, selection=cyclic; total time=   0.0s
[CV] END …eps=0.001, positive=False, selection=cyclic; total time=   0.0s
[CV] END …eps=0.001, positive=False, selection=cyclic; total time=   0.0s
[CV] END …eps=0.001, positive=False, selection=cyclic; total time=   0.0s
[CV] END …eps=0.001, positive=False, selection=cyclic; total time=   0.0s
[CV] END …eps=0.001, positive=False, selection=cyclic; total time=   0.0s
[CV] END …eps=1, positive=True, selection=random; total time=   0.0s
[CV] END …eps=1, positive=True, selection=random; total time=   0.0s
[CV] END …eps=1, positive=True, selection=random; total time=   0.0s
[CV] END …eps=1, positive=True, selection=random; total time=   0.0s
[CV] END …eps=1, positive=True, selection=random; total time=   0.0s
[CV] END …eps=0.001, positive=True, selection=cyclic; total time=   0.0s
[CV] END …eps=0.001, positive=True, selection=cyclic; total time=   0.0s
[CV] END …eps=0.001, positive=True, selection=cyclic; total time=   0.0s
```

```
[CV] END …eps=0.001, positive=True, selection=cyclic; total time=   0.0s
[CV] END …eps=0.001, positive=True, selection=cyclic; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 1.8794575635430164
Root Mean Squared Error(RMSE): 0.9432143199528643
R2 Score: 0.13091270042859826
```





[122]:
```
grid_lasso.best_params_
```

[122]:
```
{'selection': 'random', 'positive': False, 'eps': 0.01}
```

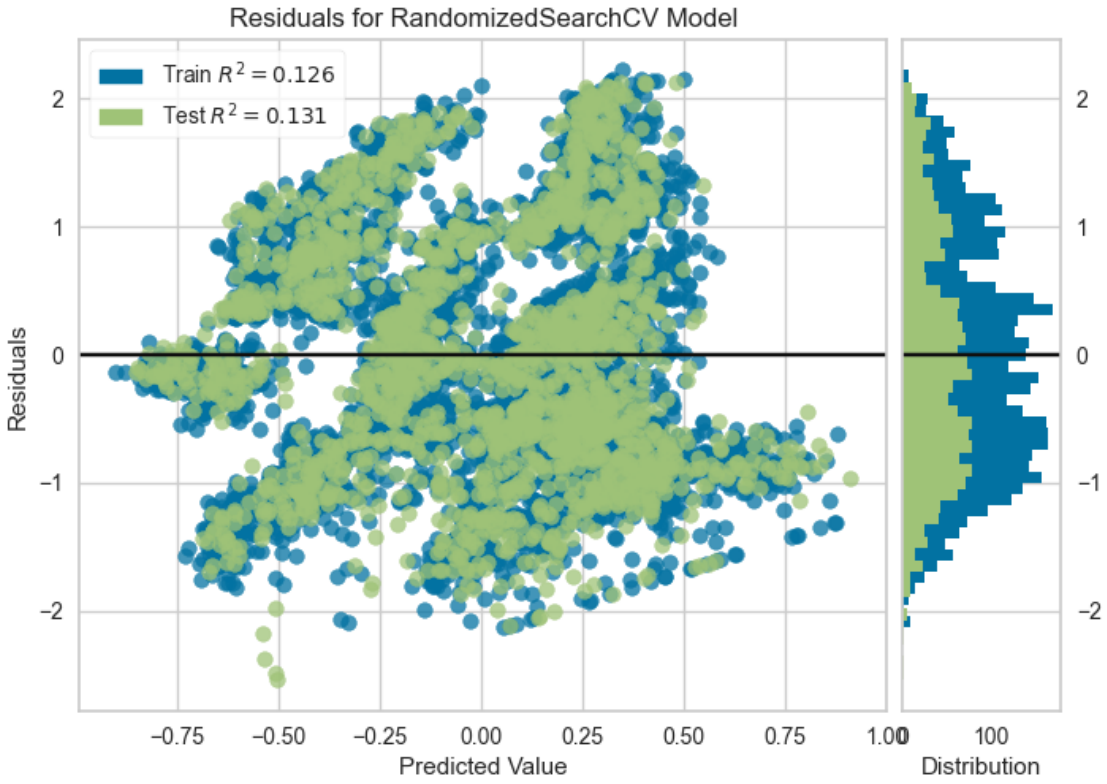
[123]:
```
param_grid = {
    'alphas': [(0.1, 1.0, 10.0),(0.01,0.1,1),(0.001,0.01,0.1)],
    'gcv_mode': ['auto', 'svd', 'eigen'],
    'store_cv_values': [True,False],
    'alpha_per_target': [True,False]
}

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

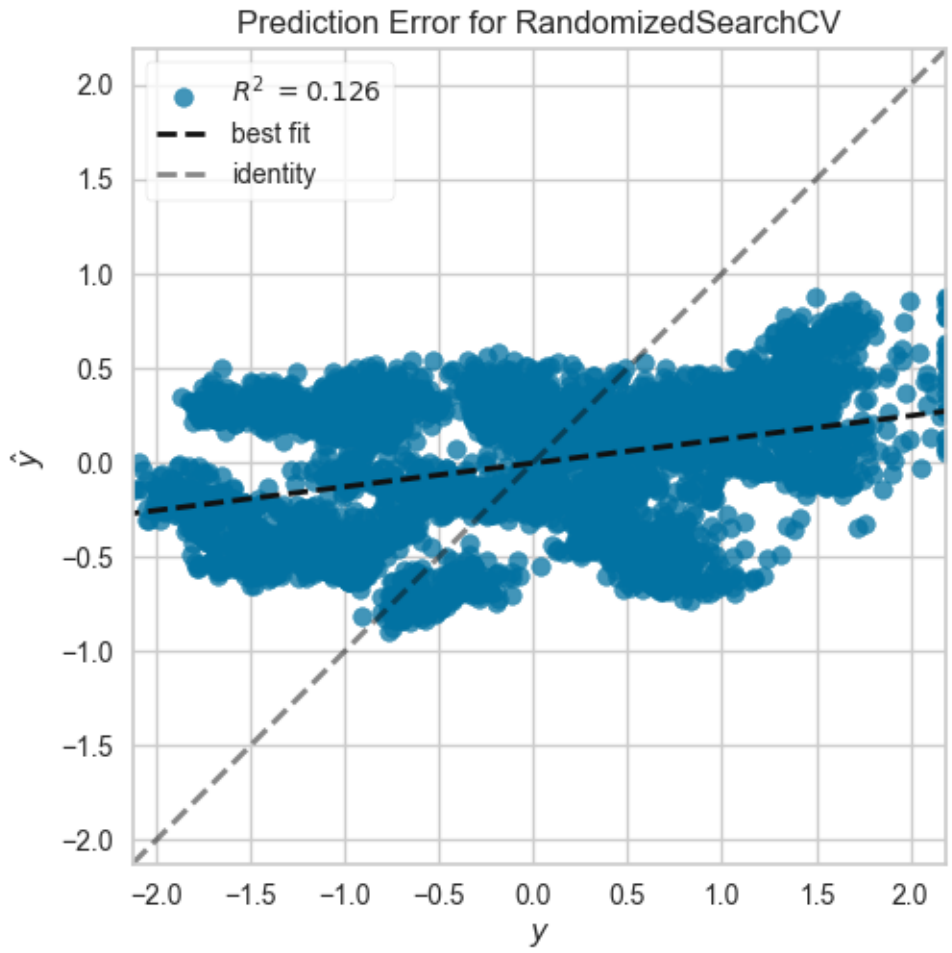
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=eigen,
store_cv_values=False; total time=  10.4s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=eigen,
```

```
store_cv_values=False; total time=  10.3s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=eigen,
store_cv_values=False; total time=  11.4s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=eigen,
store_cv_values=False; total time=  10.1s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=eigen,
store_cv_values=False; total time=  10.0s
[CV] END alpha_per_target=False, alphas=(0.01, 0.1, 1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.01, 0.1, 1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.01, 0.1, 1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.01, 0.1, 1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.01, 0.1, 1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.01, 0.1, 1), gcv_mode=eigen,
store_cv_values=True; total time=  10.0s
[CV] END alpha_per_target=False, alphas=(0.01, 0.1, 1), gcv_mode=eigen,
store_cv_values=True; total time=  10.0s
[CV] END alpha_per_target=False, alphas=(0.01, 0.1, 1), gcv_mode=eigen,
store_cv_values=True; total time=  10.1s
[CV] END alpha_per_target=False, alphas=(0.01, 0.1, 1), gcv_mode=eigen,
store_cv_values=True; total time=  10.5s
[CV] END alpha_per_target=False, alphas=(0.01, 0.1, 1), gcv_mode=eigen,
store_cv_values=True; total time=  10.5s
[CV] END alpha_per_target=True, alphas=(0.1, 1.0, 10.0), gcv_mode=svd,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.1, 1.0, 10.0), gcv_mode=svd,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.1, 1.0, 10.0), gcv_mode=svd,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.1, 1.0, 10.0), gcv_mode=svd,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.1, 1.0, 10.0), gcv_mode=svd,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.001, 0.01, 0.1), gcv_mode=auto,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.001, 0.01, 0.1), gcv_mode=auto,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.001, 0.01, 0.1), gcv_mode=auto,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.001, 0.01, 0.1), gcv_mode=auto,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.001, 0.01, 0.1), gcv_mode=auto,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.1, 1.0, 10.0), gcv_mode=eigen,
```

```
store_cv_values=True; total time=  10.1s
[CV] END alpha_per_target=False, alphas=(0.1, 1.0, 10.0), gcv_mode=eigen,
store_cv_values=True; total time=  10.2s
[CV] END alpha_per_target=False, alphas=(0.1, 1.0, 10.0), gcv_mode=eigen,
store_cv_values=True; total time=  10.0s
[CV] END alpha_per_target=False, alphas=(0.1, 1.0, 10.0), gcv_mode=eigen,
store_cv_values=True; total time=  10.0s
[CV] END alpha_per_target=False, alphas=(0.1, 1.0, 10.0), gcv_mode=eigen,
store_cv_values=True; total time=  10.1s
[CV] END alpha_per_target=True, alphas=(0.01, 0.1, 1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.01, 0.1, 1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.01, 0.1, 1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.01, 0.1, 1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.01, 0.1, 1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.1, 1.0, 10.0), gcv_mode=auto,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.1, 1.0, 10.0), gcv_mode=auto,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.1, 1.0, 10.0), gcv_mode=auto,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.1, 1.0, 10.0), gcv_mode=auto,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=True, alphas=(0.1, 1.0, 10.0), gcv_mode=auto,
store_cv_values=True; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=auto,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=auto,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=auto,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=auto,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=auto,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=svd,
store_cv_values=False; total time=   0.0s
[CV] END alpha_per_target=False, alphas=(0.001, 0.01, 0.1), gcv_mode=svd,
```

```
store_cv_values=False; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 1.8583366137333945
Root Mean Squared Error(RMSE): 0.9429127460413069
R2 Score: 0.1314683581433027
```





[124]:
```
grid_ridge.best_params_
```

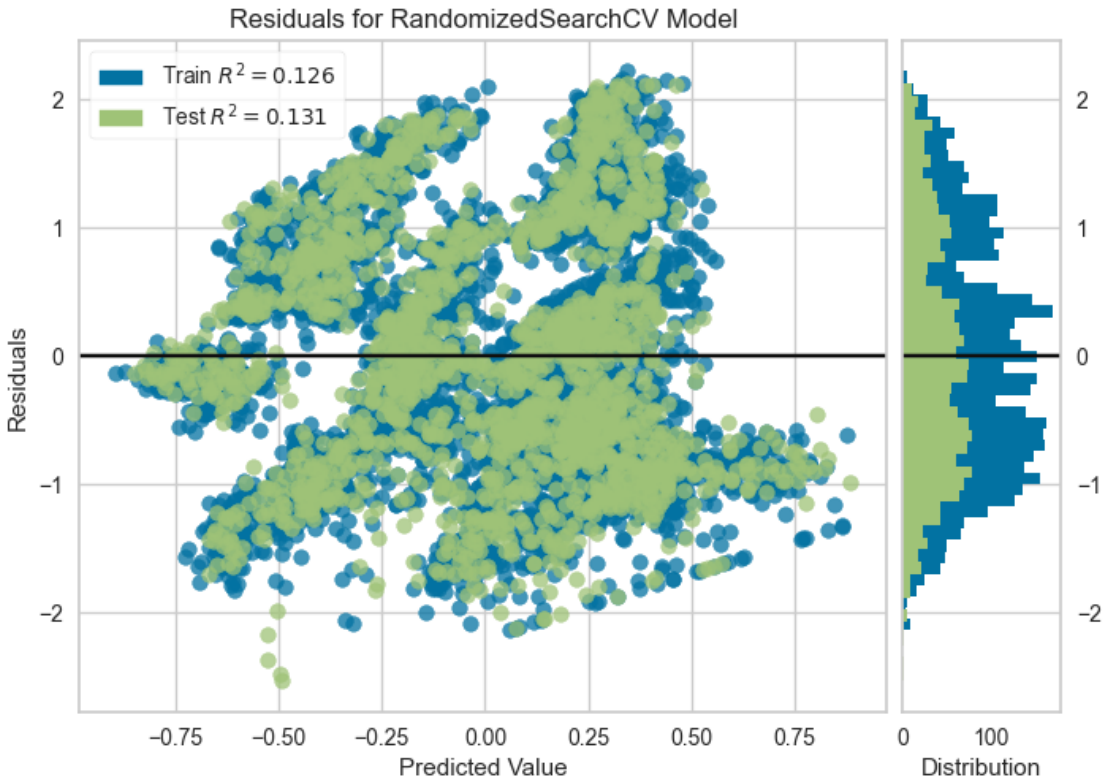
[124]:
```
{'store_cv_values': True,
 'gcv_mode': 'svd',
 'alphas': (0.1, 1.0, 10.0),
 'alpha_per_target': True}
```

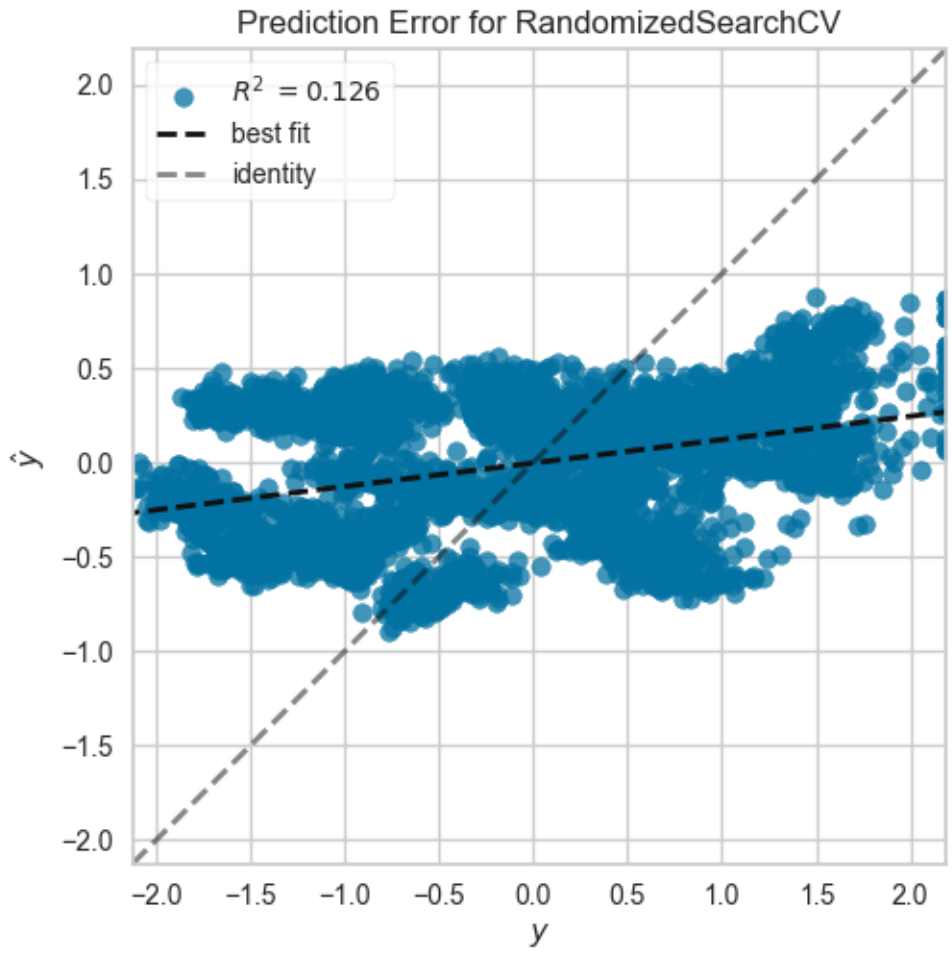
[125]:
```
param_grid = {'selection': ['cyclic','random'],
              'l1_ratio': [0.1,0.3,0.5,0.8,1],
              'positive': [True,False]}

grid_elasticnet = RandomizedSearchCV(ElasticNetCV(),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 …l1_ratio=0.1, positive=True, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=True, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=True, selection=cyclic; total time=   0.0s
```

```
[CV] END …l1_ratio=0.1, positive=True, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=True, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=1, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=1, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=1, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=1, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=1, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.8, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.8, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.8, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.8, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.8, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=1, positive=False, selection=random; total time=   0.0s
[CV] END …l1_ratio=1, positive=False, selection=random; total time=   0.0s
[CV] END …l1_ratio=1, positive=False, selection=random; total time=   0.0s
[CV] END …l1_ratio=1, positive=False, selection=random; total time=   0.0s
[CV] END …l1_ratio=1, positive=False, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.5, positive=False, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.5, positive=False, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.5, positive=False, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.5, positive=False, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.5, positive=False, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.3, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.3, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.3, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.3, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.3, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.8, positive=True, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.8, positive=True, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.8, positive=True, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.8, positive=True, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.8, positive=True, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=1, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=1, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=1, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=1, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=1, positive=False, selection=cyclic; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=True, selection=random; total time=   0.0s
[CV] END …l1_ratio=0.1, positive=True, selection=random; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 1.8664852231708287
```

```
Root Mean Squared Error(RMSE): 0.9431409062703996
R2 Score: 0.13104798339598622
```





[126]:
```
grid_elasticnet.best_params_
```

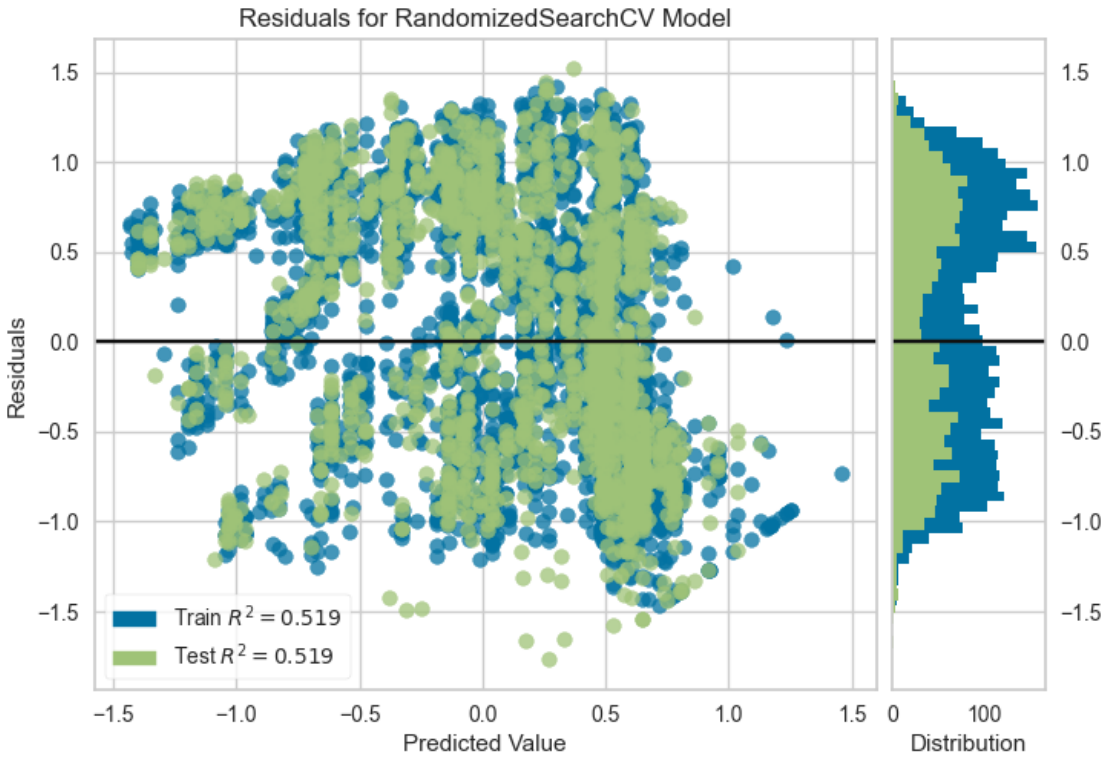
[126]:
```
{'selection': 'cyclic', 'positive': False, 'l1_ratio': 0.3}
```

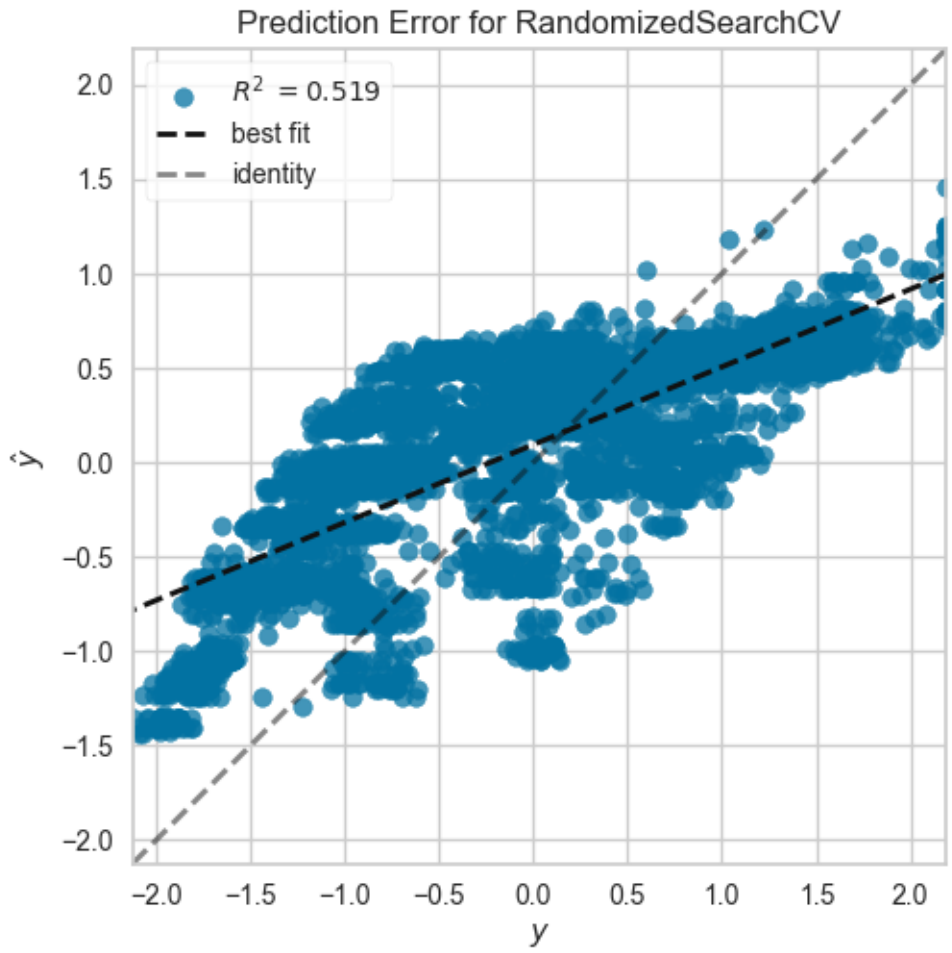
[127]:
```
param_grid = {'n_estimators': np.arange(100,1001,300),
              'learning_rate': [0.1,0.4,0.6,0.8,1],
              'loss': ['linear', 'square', 'exponential']}

grid_ab = RandomizedSearchCV(AdaBoostRegressor(),param_grid,cv=5,verbose=2)
train_and_evaluate_model(grid_ab)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END …learning_rate=0.4, loss=square, n_estimators=700; total time=   2.5s
[CV] END …learning_rate=0.4, loss=square, n_estimators=700; total time=   2.2s
[CV] END …learning_rate=0.4, loss=square, n_estimators=700; total time=   2.2s
[CV] END …learning_rate=0.4, loss=square, n_estimators=700; total time=   2.2s
[CV] END …learning_rate=0.4, loss=square, n_estimators=700; total time=   2.1s
[CV] END learning_rate=0.4, loss=exponential, n_estimators=400; total time=
```

```
1.4s
[CV] END learning_rate=0.4, loss=exponential, n_estimators=400; total time=
1.4s
[CV] END learning_rate=0.4, loss=exponential, n_estimators=400; total time=
1.4s
[CV] END learning_rate=0.4, loss=exponential, n_estimators=400; total time=
1.4s
[CV] END learning_rate=0.4, loss=exponential, n_estimators=400; total time=
1.4s
[CV] END …learning_rate=0.6, loss=linear, n_estimators=100; total time=   0.2s
[CV] END …learning_rate=0.6, loss=linear, n_estimators=100; total time=   0.3s
[CV] END …learning_rate=0.6, loss=linear, n_estimators=100; total time=   0.1s
[CV] END …learning_rate=0.6, loss=linear, n_estimators=100; total time=   0.2s
[CV] END …learning_rate=0.6, loss=linear, n_estimators=100; total time=   0.3s
[CV] END …learning_rate=0.1, loss=linear, n_estimators=400; total time=   1.7s
[CV] END …learning_rate=0.1, loss=linear, n_estimators=400; total time=   1.7s
[CV] END …learning_rate=0.1, loss=linear, n_estimators=400; total time=   1.3s
[CV] END …learning_rate=0.1, loss=linear, n_estimators=400; total time=   1.7s
[CV] END …learning_rate=0.1, loss=linear, n_estimators=400; total time=   1.4s
[CV] END …learning_rate=1, loss=square, n_estimators=100; total time=   0.3s
[CV] END …learning_rate=1, loss=square, n_estimators=100; total time=   0.3s
[CV] END …learning_rate=1, loss=square, n_estimators=100; total time=   0.3s
[CV] END …learning_rate=1, loss=square, n_estimators=100; total time=   0.3s
[CV] END …learning_rate=1, loss=square, n_estimators=100; total time=   0.2s
[CV] END learning_rate=0.8, loss=exponential, n_estimators=400; total time=
1.3s
[CV] END learning_rate=0.8, loss=exponential, n_estimators=400; total time=
1.3s
[CV] END learning_rate=0.8, loss=exponential, n_estimators=400; total time=
1.3s
[CV] END learning_rate=0.8, loss=exponential, n_estimators=400; total time=
1.3s
[CV] END learning_rate=0.8, loss=exponential, n_estimators=400; total time=
1.3s
[CV] END learning_rate=0.8, loss=exponential, n_estimators=100; total time=
0.3s
[CV] END learning_rate=0.8, loss=exponential, n_estimators=100; total time=
0.3s
[CV] END learning_rate=0.8, loss=exponential, n_estimators=100; total time=
0.3s
[CV] END learning_rate=0.8, loss=exponential, n_estimators=100; total time=
0.3s
[CV] END learning_rate=0.8, loss=exponential, n_estimators=100; total time=
0.3s
[CV] END learning_rate=1, loss=exponential, n_estimators=400; total time=   1.3s
[CV] END learning_rate=1, loss=exponential, n_estimators=400; total time=   1.3s
[CV] END learning_rate=1, loss=exponential, n_estimators=400; total time=   1.3s
[CV] END learning_rate=1, loss=exponential, n_estimators=400; total time=   1.3s
```

```
[CV] END learning_rate=1, loss=exponential, n_estimators=400; total time=   1.2s
[CV] END …learning_rate=0.4, loss=square, n_estimators=100; total time=   0.3s
[CV] END …learning_rate=0.4, loss=square, n_estimators=100; total time=   0.3s
[CV] END …learning_rate=0.4, loss=square, n_estimators=100; total time=   0.3s
[CV] END …learning_rate=0.4, loss=square, n_estimators=100; total time=   0.3s
[CV] END …learning_rate=0.4, loss=square, n_estimators=100; total time=   0.3s
[CV] END learning_rate=0.1, loss=exponential, n_estimators=1000; total time=
4.0s
[CV] END learning_rate=0.1, loss=exponential, n_estimators=1000; total time=
4.0s
[CV] END learning_rate=0.1, loss=exponential, n_estimators=1000; total time=
4.2s
[CV] END learning_rate=0.1, loss=exponential, n_estimators=1000; total time=
4.5s
[CV] END learning_rate=0.1, loss=exponential, n_estimators=1000; total time=
4.3s
Mean Absolute Percentage Error(MAPE): 3.300458874356672
Root Mean Squared Error(RMSE): 0.7017717032881801
R2 Score: 0.5189011292491548
```





[128]:
```
grid_ab.best_params_
```

[128]:
```
{'n_estimators': 100, 'loss': 'square', 'learning_rate': 1}
```

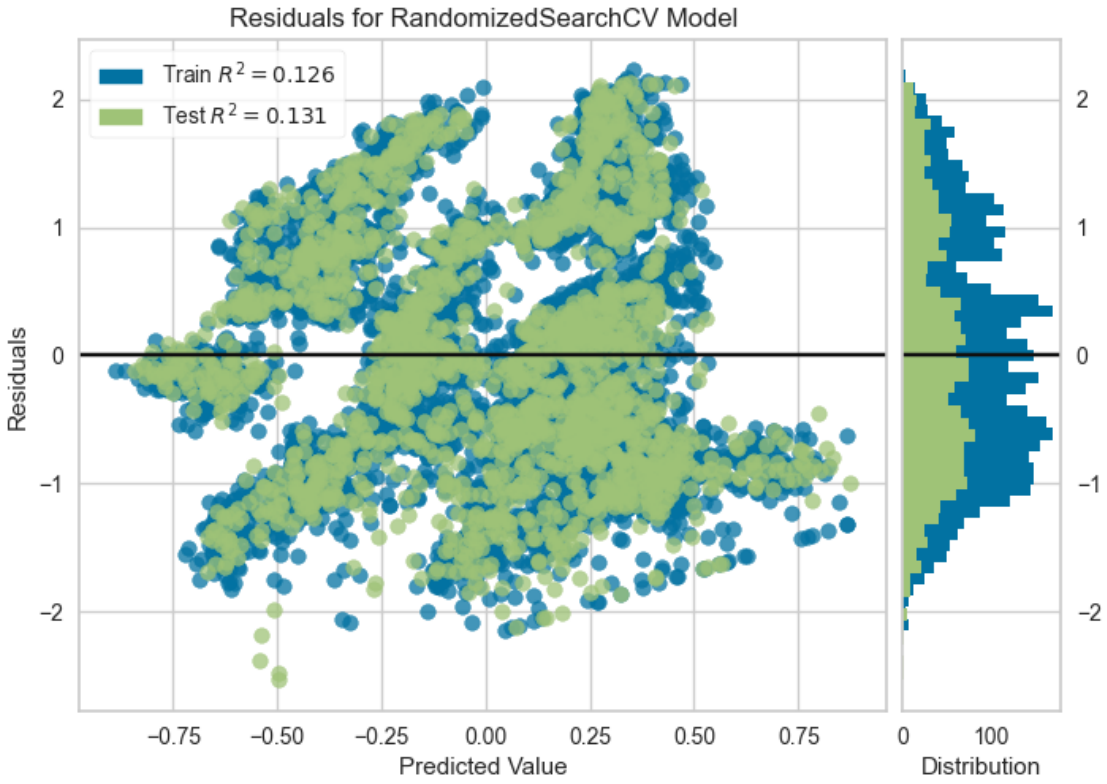
[129]:
```
param_grid = {'alpha_1': [1e-5,1e-6,1e-7,1e-8],
              'alpha_2': [1e-5,1e-6,1e-7,1e-8],
              'lambda_1': [1e-5,1e-6,1e-7,1e-8],
              'lambda_2': [1e-5,1e-6,1e-7,1e-8],
              'fit_intercept': [True,False],
              'compute_score': [True,False]}

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

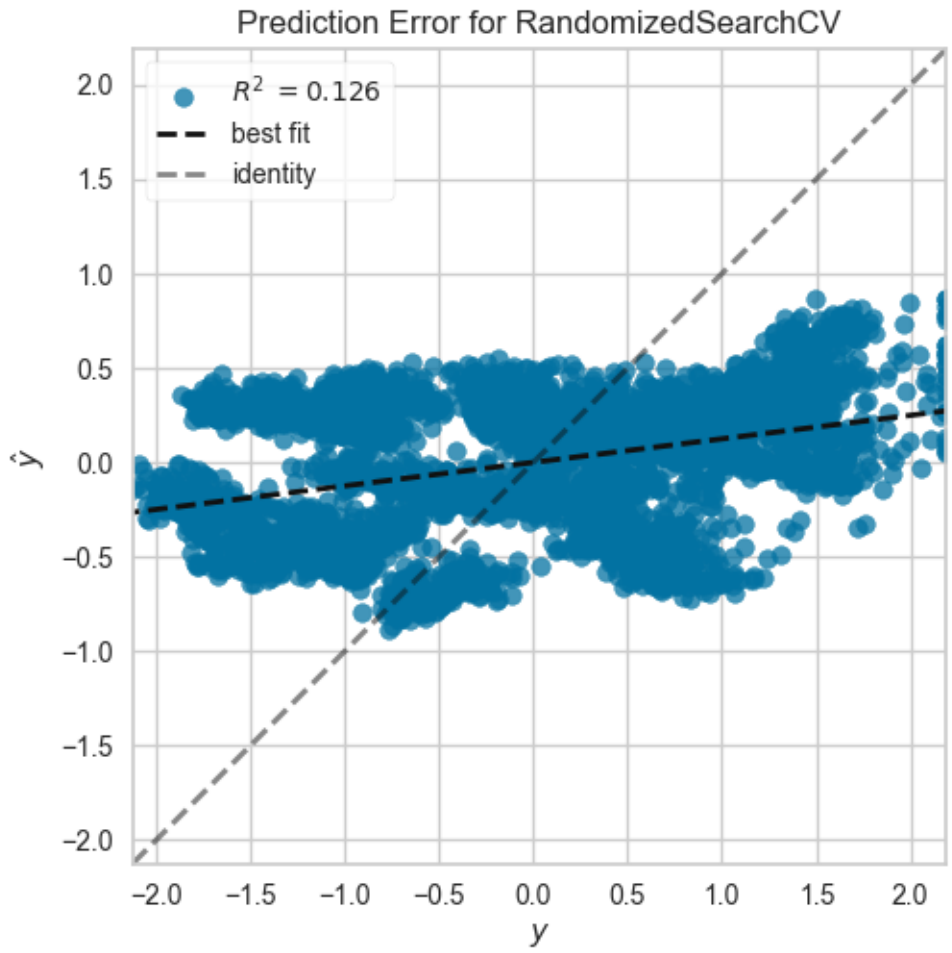
```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END alpha_1=1e-06, alpha_2=1e-06, compute_score=False, fit_intercept=False,
lambda_1=1e-07, lambda_2=1e-08; total time=   0.0s
[CV] END alpha_1=1e-06, alpha_2=1e-06, compute_score=False, fit_intercept=False,
```

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

```
lambda_1=1e-06, lambda_2=1e-06; total time=   0.0s
Mean Absolute Percentage Error(MAPE): 1.8923249460952503
Root Mean Squared Error(RMSE): 0.9431019784663416
R2 Score: 0.13111971328391103
```





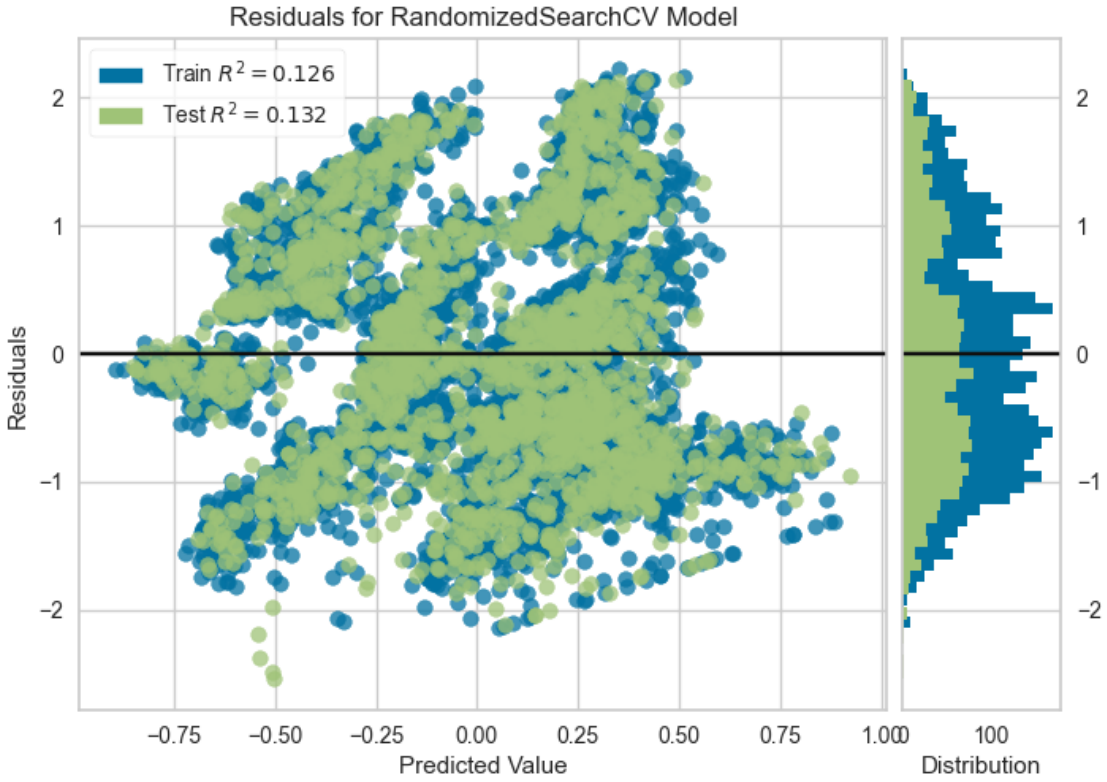
[130]:
```
grid_ard.best_params_
```

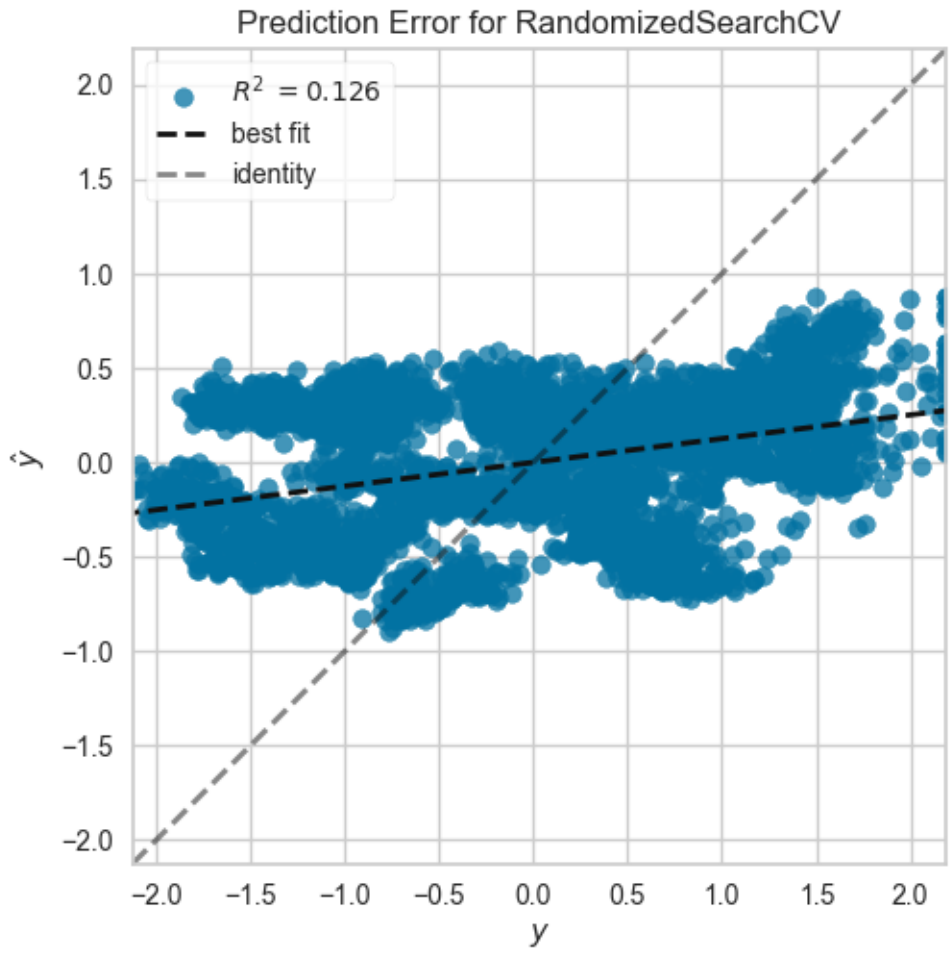
[130]:
```
{'lambda_2': 1e-08,
 'lambda_1': 1e-07,
 'fit_intercept': False,
 'compute_score': False,
 'alpha_2': 1e-06,
 'alpha_1': 1e-06}
```

[131]:
```
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 Percentage Error(MAPE): 1.8548642518330756
Root Mean Squared Error(RMSE): 0.9427826687072169
R2 Score: 0.13170797413816493
```





[132]:
```
grid_lr.best_params_
```

[132]:
```
{'positive': False, 'fit_intercept': False}
```

0.9 Optimized Models Performance Comparison

[133]:
```
model_perfs = pd.DataFrame({'Model': models, 'MAPE': mape_scores, 'RMSE':␣
 ↪rmse_scores, 'R2': r2_scores}).sort_values('R2',ascending=False)
model_perfs
```

[133]:

	Model	MAPE	RMSE	\
34	RandomizedSearchCV(cv=5,\n e…	0.439785	0.116884	
25	<catboost.core.CatBoostRegressor object at 0x0…	0.534145	0.122909	
21	XGBRegressor(base_score=None, booster=None, ca…	0.448424	0.127983	
29	RandomizedSearchCV(cv=RepeatedKFold(n_repeats=…	0.353994	0.128688	
24	LGBMRegressor()	0.483647	0.128976	
18	HistGradientBoostingRegressor()	0.510873	0.129433	

26	VotingRegressor(estimators=[('CAT',\n …	0.483340	0.131280	
36	RandomizedSearchCV(estimator=LGBMRegressor(),\…	0.812379	0.166020	
33	RandomizedSearchCV(cv=6, estimator=GradientBoo…	0.538507	0.168305	
16	(ExtraTreeRegressor(random_state=1040630546), …	0.596846	0.178494	
37	RandomizedSearchCV(cv=5, estimator=ExtraTreesR…	0.642837	0.179146	
35	RandomizedSearchCV(cv=5, estimator=BaggingRegr…	0.625728	0.179327	
14	(DecisionTreeRegressor(max_features=1.0, rando…	0.631636	0.180889	
15	(DecisionTreeRegressor(random_state=1012004194…	0.628884	0.181817	
27	RandomizedSearchCV(cv=5, estimator=RandomFores…	0.670434	0.216686	
32	RandomizedSearchCV(cv=RepeatedKFold(n_repeats=…	0.679039	0.220655	
13	DecisionTreeRegressor()	0.688617	0.229284	
19	([DecisionTreeRegressor(criterion='friedman_ms…	1.525258	0.302092	
22	XGBRFRegressor(base_score=None, booster=None, …	1.495870	0.395600	
28	RandomizedSearchCV(cv=RepeatedKFold(n_repeats=…	1.827042	0.678111	
20	MLPRegressor()	2.302538	0.680757	
46	RandomizedSearchCV(cv=5, estimator=AdaBoostReg…	3.300459	0.701772	
17	(DecisionTreeRegressor(max_depth=3, random_sta…	2.575329	0.725043	
10	KNeighborsRegressor()	1.832426	0.725657	
23	NuSVR()	2.767677	0.795309	
30	RandomizedSearchCV(cv=5, estimator=SVR(),\n …	2.015902	0.806134	
11	SVR()	1.778644	0.815938	
48	RandomizedSearchCV(cv=5, estimator=LinearRegre…	1.854864	0.942783	
0	LinearRegression()	1.858860	0.942806	
40	RandomizedSearchCV(cv=5, estimator=HuberRegres…	1.858399	0.942907	
2	RidgeCV()	1.858337	0.942913	
44	RandomizedSearchCV(cv=5, estimator=RidgeCV(),\…	1.858337	0.942913	
4	SGDRegressor()	1.824808	0.942968	
38	RandomizedSearchCV(cv=5,\n e…	1.864210	0.942998	
1	LassoCV()	1.872174	0.943096	
47	RandomizedSearchCV(cv=5, estimator=ARDRegressi…	1.892325	0.943102	
3	ElasticNetCV()	1.869417	0.943112	
6	ARDRegression()	1.896284	0.943125	
45	RandomizedSearchCV(cv=5, estimator=ElasticNetC…	1.866485	0.943141	
43	RandomizedSearchCV(cv=5, estimator=LassoCV(),\…	1.879458	0.943214	
42	RandomizedSearchCV(cv=5, estimator=PassiveAggr…	2.071837	0.943652	
5	HuberRegressor()	2.047510	0.945681	
31	RandomizedSearchCV(cv=RepeatedKFold(n_repeats=…	1.802767	0.945701	
12	LinearSVR()	2.058692	0.953704	
39	RandomizedSearchCV(cv=5, estimator=SGDRegresso…	2.256280	0.964216	
8	TweedieRegressor()	1.300327	0.964521	
41	RandomizedSearchCV(cv=5, estimator=TweedieRegr…	1.053728	0.994684	
9	PassiveAggressiveRegressor()	5.203633	1.123522	
7	RANSACRegressor()	4.191927	1.257529	

	R2
34	0.986654
25	0.985243

```
21  0.983999
29  0.983822
24  0.983750
18  0.983634
26  0.983164
36  0.973074
33  0.972328
16  0.968876
37  0.968649
35  0.968585
14  0.968036
15  0.967707
27  0.954133
32  0.952437
13  0.948644
19  0.910850
22  0.847118
28  0.550795
20  0.547282
46  0.518901
17  0.486465
10  0.485594
23  0.382105
30  0.365171
11  0.349636
48  0.131708
0   0.131666
40  0.131479
2   0.131468
44  0.131468
4   0.131367
38  0.131312
1   0.131130
47  0.131120
3   0.131102
6   0.131078
45  0.131048
43  0.130913
42  0.130106
5   0.126361
31  0.126324
12  0.111474
39  0.091779
8   0.091205
41  0.033475
9  -0.233122
7  -0.544823
```

Hyperparameter-tuned Cat Boost Regressor is the best performing model among all optimized models, achieving an outstanding r2 score of nearly 98.7%.

0.10 Saving the best model for future use

[141]:
```
# Cross validating the performance of the Cat Boost Regressor model before␣
 ↪saving it
avg_cv_score = np.
 ↪mean(cross_val_score(grid_cat,X_test,y_test,scoring='r2',cv=RepeatedKFold(n_splits=4,n_repe
print("Average cross validation R2 score:",avg_cv_score)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END learning_rate=0.4, n_estimators=700;, score=0.967 total time=
1.6s
[CV 2/5] END learning_rate=0.4, n_estimators=700;, score=0.962 total time=
1.2s
[CV 3/5] END learning_rate=0.4, n_estimators=700;, score=0.967 total time=
1.3s
[CV 4/5] END learning_rate=0.4, n_estimators=700;, score=0.957 total time=
1.2s
[CV 5/5] END learning_rate=0.4, n_estimators=700;, score=0.962 total time=
1.3s
[CV 1/5] END learning_rate=0.4, n_estimators=400;, score=0.967 total time=
1.1s
[CV 2/5] END learning_rate=0.4, n_estimators=400;, score=0.962 total time=
1.0s
[CV 3/5] END learning_rate=0.4, n_estimators=400;, score=0.967 total time=
0.9s
[CV 4/5] END learning_rate=0.4, n_estimators=400;, score=0.957 total time=
0.9s
[CV 5/5] END learning_rate=0.4, n_estimators=400;, score=0.961 total time=
0.9s
[CV 1/5] END learning_rate=0.2, n_estimators=400;, score=0.967 total time=
1.1s
[CV 2/5] END learning_rate=0.2, n_estimators=400;, score=0.968 total time=
0.9s
[CV 3/5] END learning_rate=0.2, n_estimators=400;, score=0.973 total time=
0.9s
[CV 4/5] END learning_rate=0.2, n_estimators=400;, score=0.967 total time=
0.9s
[CV 5/5] END learning_rate=0.2, n_estimators=400;, score=0.964 total time=
0.9s
[CV 1/5] END learning_rate=0.7, n_estimators=1000;, score=0.963 total time=
1.7s
[CV 2/5] END learning_rate=0.7, n_estimators=1000;, score=0.960 total time=
1.7s
```

```
[CV 3/5] END learning_rate=0.7, n_estimators=1000;, score=0.963 total time=
1.8s
[CV 4/5] END learning_rate=0.7, n_estimators=1000;, score=0.965 total time=
2.2s
[CV 5/5] END learning_rate=0.7, n_estimators=1000;, score=0.948 total time=
1.5s
[CV 1/5] END learning_rate=0.5, n_estimators=100;, score=0.961 total time=
0.6s
[CV 2/5] END learning_rate=0.5, n_estimators=100;, score=0.954 total time=
0.7s
[CV 3/5] END learning_rate=0.5, n_estimators=100;, score=0.966 total time=
0.7s
[CV 4/5] END learning_rate=0.5, n_estimators=100;, score=0.957 total time=
0.8s
[CV 5/5] END learning_rate=0.5, n_estimators=100;, score=0.958 total time=
0.5s
[CV 1/5] END .learning_rate=1, n_estimators=100;, score=0.941 total time=   0.6s
[CV 2/5] END .learning_rate=1, n_estimators=100;, score=0.933 total time=   0.5s
[CV 3/5] END .learning_rate=1, n_estimators=100;, score=0.940 total time=   0.4s
[CV 4/5] END .learning_rate=1, n_estimators=100;, score=0.941 total time=   0.8s
[CV 5/5] END .learning_rate=1, n_estimators=100;, score=0.922 total time=   0.6s
[CV 1/5] END learning_rate=0.4, n_estimators=100;, score=0.962 total time=
0.7s
[CV 2/5] END learning_rate=0.4, n_estimators=100;, score=0.955 total time=
0.7s
[CV 3/5] END learning_rate=0.4, n_estimators=100;, score=0.966 total time=
0.7s
[CV 4/5] END learning_rate=0.4, n_estimators=100;, score=0.953 total time=
0.5s
[CV 5/5] END learning_rate=0.4, n_estimators=100;, score=0.956 total time=
0.6s
[CV 1/5] END learning_rate=0.2, n_estimators=1000;, score=0.968 total time=
1.8s
[CV 2/5] END learning_rate=0.2, n_estimators=1000;, score=0.969 total time=
1.8s
[CV 3/5] END learning_rate=0.2, n_estimators=1000;, score=0.973 total time=
1.6s
[CV 4/5] END learning_rate=0.2, n_estimators=1000;, score=0.968 total time=
1.8s
[CV 5/5] END learning_rate=0.2, n_estimators=1000;, score=0.964 total time=
1.9s
[CV 1/5] END .learning_rate=1, n_estimators=400;, score=0.942 total time=   1.1s
[CV 2/5] END .learning_rate=1, n_estimators=400;, score=0.933 total time=   1.1s
[CV 3/5] END .learning_rate=1, n_estimators=400;, score=0.941 total time=   1.1s
[CV 4/5] END .learning_rate=1, n_estimators=400;, score=0.942 total time=   1.2s
[CV 5/5] END .learning_rate=1, n_estimators=400;, score=0.923 total time=   1.1s
[CV 1/5] END .learning_rate=1, n_estimators=700;, score=0.942 total time=   1.4s
[CV 2/5] END .learning_rate=1, n_estimators=700;, score=0.934 total time=   1.5s
```

```
[CV 3/5] END .learning_rate=1, n_estimators=700;, score=0.941 total time=   1.6s
[CV 4/5] END .learning_rate=1, n_estimators=700;, score=0.942 total time=   1.4s
[CV 5/5] END .learning_rate=1, n_estimators=700;, score=0.923 total time=   1.5s
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END learning_rate=0.5, n_estimators=100;, score=0.949 total time=
0.6s
[CV 2/5] END learning_rate=0.5, n_estimators=100;, score=0.966 total time=
0.8s
[CV 3/5] END learning_rate=0.5, n_estimators=100;, score=0.969 total time=
1.0s
[CV 4/5] END learning_rate=0.5, n_estimators=100;, score=0.951 total time=
0.8s
[CV 5/5] END learning_rate=0.5, n_estimators=100;, score=0.964 total time=
0.8s
[CV 1/5] END learning_rate=0.2, n_estimators=800;, score=0.958 total time=
1.5s
[CV 2/5] END learning_rate=0.2, n_estimators=800;, score=0.972 total time=
1.5s
[CV 3/5] END learning_rate=0.2, n_estimators=800;, score=0.973 total time=
1.6s
[CV 4/5] END learning_rate=0.2, n_estimators=800;, score=0.966 total time=
1.7s
[CV 5/5] END learning_rate=0.2, n_estimators=800;, score=0.972 total time=
1.4s
[CV 1/5] END learning_rate=0.4, n_estimators=400;, score=0.953 total time=
1.6s
[CV 2/5] END learning_rate=0.4, n_estimators=400;, score=0.961 total time=
1.4s
[CV 3/5] END learning_rate=0.4, n_estimators=400;, score=0.966 total time=
1.2s
[CV 4/5] END learning_rate=0.4, n_estimators=400;, score=0.952 total time=
1.1s
[CV 5/5] END learning_rate=0.4, n_estimators=400;, score=0.971 total time=
1.2s
[CV 1/5] END learning_rate=0.7, n_estimators=400;, score=0.941 total time=
1.2s
[CV 2/5] END learning_rate=0.7, n_estimators=400;, score=0.958 total time=
1.3s
[CV 3/5] END learning_rate=0.7, n_estimators=400;, score=0.956 total time=
1.1s
[CV 4/5] END learning_rate=0.7, n_estimators=400;, score=0.942 total time=
1.2s
[CV 5/5] END learning_rate=0.7, n_estimators=400;, score=0.952 total time=
1.3s
[CV 1/5] END .learning_rate=1, n_estimators=100;, score=0.952 total time=   1.0s
[CV 2/5] END .learning_rate=1, n_estimators=100;, score=0.940 total time=   0.8s
[CV 3/5] END .learning_rate=1, n_estimators=100;, score=0.935 total time=   0.8s
[CV 4/5] END .learning_rate=1, n_estimators=100;, score=0.937 total time=   0.7s
```

```
[CV 5/5] END .learning_rate=1, n_estimators=100;, score=0.943 total time=   0.7s
[CV 1/5] END learning_rate=0.4, n_estimators=1000;, score=0.952 total time=
1.9s
[CV 2/5] END learning_rate=0.4, n_estimators=1000;, score=0.961 total time=
1.8s
[CV 3/5] END learning_rate=0.4, n_estimators=1000;, score=0.967 total time=
1.9s
[CV 4/5] END learning_rate=0.4, n_estimators=1000;, score=0.952 total time=
2.2s
[CV 5/5] END learning_rate=0.4, n_estimators=1000;, score=0.971 total time=
2.0s
[CV 1/5] END learning_rate=0.7, n_estimators=700;, score=0.942 total time=
1.4s
[CV 2/5] END learning_rate=0.7, n_estimators=700;, score=0.958 total time=
1.4s
[CV 3/5] END learning_rate=0.7, n_estimators=700;, score=0.956 total time=
1.3s
[CV 4/5] END learning_rate=0.7, n_estimators=700;, score=0.942 total time=
1.4s
[CV 5/5] END learning_rate=0.7, n_estimators=700;, score=0.952 total time=
1.7s
[CV 1/5] END learning_rate=0.5, n_estimators=1000;, score=0.952 total time=
2.1s
[CV 2/5] END learning_rate=0.5, n_estimators=1000;, score=0.968 total time=
2.0s
[CV 3/5] END learning_rate=0.5, n_estimators=1000;, score=0.972 total time=
1.8s
[CV 4/5] END learning_rate=0.5, n_estimators=1000;, score=0.956 total time=
2.2s
[CV 5/5] END learning_rate=0.5, n_estimators=1000;, score=0.967 total time=
2.1s
[CV 1/5] END learning_rate=0.5, n_estimators=700;, score=0.952 total time=
1.5s
[CV 2/5] END learning_rate=0.5, n_estimators=700;, score=0.968 total time=
1.4s
[CV 3/5] END learning_rate=0.5, n_estimators=700;, score=0.972 total time=
1.8s
[CV 4/5] END learning_rate=0.5, n_estimators=700;, score=0.956 total time=
1.5s
[CV 5/5] END learning_rate=0.5, n_estimators=700;, score=0.967 total time=
1.5s
[CV 1/5] END learning_rate=0.5, n_estimators=800;, score=0.952 total time=
1.6s
[CV 2/5] END learning_rate=0.5, n_estimators=800;, score=0.968 total time=
1.8s
[CV 3/5] END learning_rate=0.5, n_estimators=800;, score=0.972 total time=
1.4s
[CV 4/5] END learning_rate=0.5, n_estimators=800;, score=0.956 total time=
```

```
1.5s
[CV 5/5] END learning_rate=0.5, n_estimators=800;, score=0.967 total time=
1.6s
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END learning_rate=0.2, n_estimators=100;, score=0.944 total time=
0.7s
[CV 2/5] END learning_rate=0.2, n_estimators=100;, score=0.952 total time=
0.7s
[CV 3/5] END learning_rate=0.2, n_estimators=100;, score=0.966 total time=
0.8s
[CV 4/5] END learning_rate=0.2, n_estimators=100;, score=0.952 total time=
0.8s
[CV 5/5] END learning_rate=0.2, n_estimators=100;, score=0.947 total time=
1.0s
[CV 1/5] END .learning_rate=1, n_estimators=100;, score=0.920 total time=   0.8s
[CV 2/5] END .learning_rate=1, n_estimators=100;, score=0.948 total time=   0.7s
[CV 3/5] END .learning_rate=1, n_estimators=100;, score=0.959 total time=   0.7s
[CV 4/5] END .learning_rate=1, n_estimators=100;, score=0.897 total time=   0.6s
[CV 5/5] END .learning_rate=1, n_estimators=100;, score=0.948 total time=   0.7s
[CV 1/5] END learning_rate=0.2, n_estimators=800;, score=0.966 total time=
1.7s
[CV 2/5] END learning_rate=0.2, n_estimators=800;, score=0.972 total time=
1.5s
[CV 3/5] END learning_rate=0.2, n_estimators=800;, score=0.976 total time=
1.7s
[CV 4/5] END learning_rate=0.2, n_estimators=800;, score=0.970 total time=
1.6s
[CV 5/5] END learning_rate=0.2, n_estimators=800;, score=0.969 total time=
1.6s
[CV 1/5] END learning_rate=0.7, n_estimators=700;, score=0.960 total time=
1.4s
[CV 2/5] END learning_rate=0.7, n_estimators=700;, score=0.963 total time=
1.5s
[CV 3/5] END learning_rate=0.7, n_estimators=700;, score=0.962 total time=
1.5s
[CV 4/5] END learning_rate=0.7, n_estimators=700;, score=0.932 total time=
1.5s
[CV 5/5] END learning_rate=0.7, n_estimators=700;, score=0.961 total time=
1.7s
[CV 1/5] END learning_rate=0.5, n_estimators=100;, score=0.956 total time=
0.8s
[CV 2/5] END learning_rate=0.5, n_estimators=100;, score=0.960 total time=
1.1s
[CV 3/5] END learning_rate=0.5, n_estimators=100;, score=0.975 total time=
0.7s
[CV 4/5] END learning_rate=0.5, n_estimators=100;, score=0.948 total time=
0.7s
[CV 5/5] END learning_rate=0.5, n_estimators=100;, score=0.959 total time=
```

```
0.7s
[CV 1/5] END learning_rate=0.2, n_estimators=1000;, score=0.966 total time=
1.9s
[CV 2/5] END learning_rate=0.2, n_estimators=1000;, score=0.972 total time=
2.1s
[CV 3/5] END learning_rate=0.2, n_estimators=1000;, score=0.977 total time=
1.7s
[CV 4/5] END learning_rate=0.2, n_estimators=1000;, score=0.971 total time=
1.7s
[CV 5/5] END learning_rate=0.2, n_estimators=1000;, score=0.969 total time=
2.1s
[CV 1/5] END learning_rate=0.4, n_estimators=400;, score=0.954 total time=
1.1s
[CV 2/5] END learning_rate=0.4, n_estimators=400;, score=0.967 total time=
1.3s
[CV 3/5] END learning_rate=0.4, n_estimators=400;, score=0.968 total time=
1.1s
[CV 4/5] END learning_rate=0.4, n_estimators=400;, score=0.961 total time=
1.7s
[CV 5/5] END learning_rate=0.4, n_estimators=400;, score=0.958 total time=
1.1s
[CV 1/5] END .learning_rate=1, n_estimators=800;, score=0.921 total time=   1.8s
[CV 2/5] END .learning_rate=1, n_estimators=800;, score=0.947 total time=   1.8s
[CV 3/5] END .learning_rate=1, n_estimators=800;, score=0.960 total time=   1.5s
[CV 4/5] END .learning_rate=1, n_estimators=800;, score=0.901 total time=   1.7s
[CV 5/5] END .learning_rate=1, n_estimators=800;, score=0.949 total time=   1.5s
[CV 1/5] END .learning_rate=1, n_estimators=700;, score=0.921 total time=   1.4s
[CV 2/5] END .learning_rate=1, n_estimators=700;, score=0.947 total time=   1.5s
[CV 3/5] END .learning_rate=1, n_estimators=700;, score=0.960 total time=   2.0s
[CV 4/5] END .learning_rate=1, n_estimators=700;, score=0.901 total time=   1.4s
[CV 5/5] END .learning_rate=1, n_estimators=700;, score=0.949 total time=   1.4s
[CV 1/5] END learning_rate=0.7, n_estimators=1000;, score=0.960 total time=
1.7s
[CV 2/5] END learning_rate=0.7, n_estimators=1000;, score=0.963 total time=
1.7s
[CV 3/5] END learning_rate=0.7, n_estimators=1000;, score=0.962 total time=
1.8s
[CV 4/5] END learning_rate=0.7, n_estimators=1000;, score=0.933 total time=
2.3s
[CV 5/5] END learning_rate=0.7, n_estimators=1000;, score=0.961 total time=
1.7s
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END learning_rate=0.5, n_estimators=800;, score=0.969 total time=
1.8s
[CV 2/5] END learning_rate=0.5, n_estimators=800;, score=0.960 total time=
1.5s
[CV 3/5] END learning_rate=0.5, n_estimators=800;, score=0.967 total time=
1.7s
```

```
[CV 4/5] END learning_rate=0.5, n_estimators=800;, score=0.957 total time=
1.5s
[CV 5/5] END learning_rate=0.5, n_estimators=800;, score=0.960 total time=
1.7s
[CV 1/5] END learning_rate=0.7, n_estimators=800;, score=0.972 total time=
1.7s
[CV 2/5] END learning_rate=0.7, n_estimators=800;, score=0.945 total time=
1.6s
[CV 3/5] END learning_rate=0.7, n_estimators=800;, score=0.962 total time=
3.0s
[CV 4/5] END learning_rate=0.7, n_estimators=800;, score=0.950 total time=
2.1s
[CV 5/5] END learning_rate=0.7, n_estimators=800;, score=0.952 total time=
1.6s
[CV 1/5] END .learning_rate=1, n_estimators=800;, score=0.939 total time=   1.6s
[CV 2/5] END .learning_rate=1, n_estimators=800;, score=0.942 total time=   1.5s
[CV 3/5] END .learning_rate=1, n_estimators=800;, score=0.938 total time=   2.2s
[CV 4/5] END .learning_rate=1, n_estimators=800;, score=0.929 total time=   1.7s
[CV 5/5] END .learning_rate=1, n_estimators=800;, score=0.936 total time=   1.6s
[CV 1/5] END .learning_rate=1, n_estimators=700;, score=0.939 total time=   1.5s
[CV 2/5] END .learning_rate=1, n_estimators=700;, score=0.942 total time=   1.4s
[CV 3/5] END .learning_rate=1, n_estimators=700;, score=0.938 total time=   1.5s
[CV 4/5] END .learning_rate=1, n_estimators=700;, score=0.929 total time=   1.4s
[CV 5/5] END .learning_rate=1, n_estimators=700;, score=0.936 total time=   1.5s
[CV 1/5] END learning_rate=0.2, n_estimators=1000;, score=0.975 total time=
1.8s
[CV 2/5] END learning_rate=0.2, n_estimators=1000;, score=0.972 total time=
2.2s
[CV 3/5] END learning_rate=0.2, n_estimators=1000;, score=0.971 total time=
1.9s
[CV 4/5] END learning_rate=0.2, n_estimators=1000;, score=0.966 total time=
2.3s
[CV 5/5] END learning_rate=0.2, n_estimators=1000;, score=0.969 total time=
1.7s
[CV 1/5] END learning_rate=0.2, n_estimators=700;, score=0.976 total time=
1.4s
[CV 2/5] END learning_rate=0.2, n_estimators=700;, score=0.971 total time=
1.4s
[CV 3/5] END learning_rate=0.2, n_estimators=700;, score=0.970 total time=
1.4s
[CV 4/5] END learning_rate=0.2, n_estimators=700;, score=0.965 total time=
1.5s
[CV 5/5] END learning_rate=0.2, n_estimators=700;, score=0.969 total time=
1.5s
[CV 1/5] END learning_rate=1, n_estimators=1000;, score=0.939 total time=   1.8s
[CV 2/5] END learning_rate=1, n_estimators=1000;, score=0.942 total time=   2.3s
[CV 3/5] END learning_rate=1, n_estimators=1000;, score=0.938 total time=   1.7s
[CV 4/5] END learning_rate=1, n_estimators=1000;, score=0.929 total time=   2.3s
```

```
[CV 5/5] END learning_rate=1, n_estimators=1000;, score=0.936 total time=   1.8s
[CV 1/5] END learning_rate=0.5, n_estimators=400;, score=0.969 total time=
1.1s
[CV 2/5] END learning_rate=0.5, n_estimators=400;, score=0.960 total time=
1.1s
[CV 3/5] END learning_rate=0.5, n_estimators=400;, score=0.967 total time=
1.2s
[CV 4/5] END learning_rate=0.5, n_estimators=400;, score=0.956 total time=
1.0s
[CV 5/5] END learning_rate=0.5, n_estimators=400;, score=0.960 total time=
1.1s
[CV 1/5] END learning_rate=0.2, n_estimators=400;, score=0.975 total time=
1.0s
[CV 2/5] END learning_rate=0.2, n_estimators=400;, score=0.970 total time=
1.0s
[CV 3/5] END learning_rate=0.2, n_estimators=400;, score=0.970 total time=
1.1s
[CV 4/5] END learning_rate=0.2, n_estimators=400;, score=0.963 total time=
1.0s
[CV 5/5] END learning_rate=0.2, n_estimators=400;, score=0.969 total time=
1.0s
[CV 1/5] END .learning_rate=1, n_estimators=400;, score=0.939 total time=   0.9s
[CV 2/5] END .learning_rate=1, n_estimators=400;, score=0.942 total time=   0.9s
[CV 3/5] END .learning_rate=1, n_estimators=400;, score=0.938 total time=   1.0s
[CV 4/5] END .learning_rate=1, n_estimators=400;, score=0.929 total time=   0.8s
[CV 5/5] END .learning_rate=1, n_estimators=400;, score=0.936 total time=   1.1s
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END .learning_rate=1, n_estimators=800;, score=0.931 total time=   1.5s
[CV 2/5] END .learning_rate=1, n_estimators=800;, score=0.932 total time=   2.0s
[CV 3/5] END .learning_rate=1, n_estimators=800;, score=0.945 total time=   2.2s
[CV 4/5] END .learning_rate=1, n_estimators=800;, score=0.928 total time=   1.4s
[CV 5/5] END .learning_rate=1, n_estimators=800;, score=0.945 total time=   1.4s
[CV 1/5] END learning_rate=0.7, n_estimators=700;, score=0.954 total time=
1.4s
[CV 2/5] END learning_rate=0.7, n_estimators=700;, score=0.941 total time=
1.4s
[CV 3/5] END learning_rate=0.7, n_estimators=700;, score=0.958 total time=
1.4s
[CV 4/5] END learning_rate=0.7, n_estimators=700;, score=0.954 total time=
1.7s
[CV 5/5] END learning_rate=0.7, n_estimators=700;, score=0.954 total time=
1.3s
[CV 1/5] END learning_rate=0.4, n_estimators=100;, score=0.954 total time=
0.6s
[CV 2/5] END learning_rate=0.4, n_estimators=100;, score=0.962 total time=
0.7s
[CV 3/5] END learning_rate=0.4, n_estimators=100;, score=0.961 total time=
0.8s
```

```
[CV 4/5] END learning_rate=0.4, n_estimators=100;, score=0.953 total time=
1.0s
[CV 5/5] END learning_rate=0.4, n_estimators=100;, score=0.958 total time=
0.8s
[CV 1/5] END learning_rate=0.2, n_estimators=1000;, score=0.967 total time=
1.8s
[CV 2/5] END learning_rate=0.2, n_estimators=1000;, score=0.970 total time=
2.0s
[CV 3/5] END learning_rate=0.2, n_estimators=1000;, score=0.971 total time=
1.7s
[CV 4/5] END learning_rate=0.2, n_estimators=1000;, score=0.965 total time=
1.7s
[CV 5/5] END learning_rate=0.2, n_estimators=1000;, score=0.969 total time=
2.3s
[CV 1/5] END learning_rate=0.5, n_estimators=800;, score=0.953 total time=
1.5s
[CV 2/5] END learning_rate=0.5, n_estimators=800;, score=0.955 total time=
1.5s
[CV 3/5] END learning_rate=0.5, n_estimators=800;, score=0.960 total time=
1.7s
[CV 4/5] END learning_rate=0.5, n_estimators=800;, score=0.961 total time=
1.5s
[CV 5/5] END learning_rate=0.5, n_estimators=800;, score=0.958 total time=
2.0s
[CV 1/5] END learning_rate=0.4, n_estimators=800;, score=0.958 total time=
1.5s
[CV 2/5] END learning_rate=0.4, n_estimators=800;, score=0.967 total time=
1.7s
[CV 3/5] END learning_rate=0.4, n_estimators=800;, score=0.965 total time=
1.6s
[CV 4/5] END learning_rate=0.4, n_estimators=800;, score=0.959 total time=
1.6s
[CV 5/5] END learning_rate=0.4, n_estimators=800;, score=0.963 total time=
2.1s
[CV 1/5] END learning_rate=0.5, n_estimators=700;, score=0.953 total time=
1.4s
[CV 2/5] END learning_rate=0.5, n_estimators=700;, score=0.955 total time=
1.3s
[CV 3/5] END learning_rate=0.5, n_estimators=700;, score=0.960 total time=
1.3s
[CV 4/5] END learning_rate=0.5, n_estimators=700;, score=0.961 total time=
1.8s
[CV 5/5] END learning_rate=0.5, n_estimators=700;, score=0.958 total time=
1.3s
[CV 1/5] END learning_rate=0.7, n_estimators=100;, score=0.955 total time=
0.5s
[CV 2/5] END learning_rate=0.7, n_estimators=100;, score=0.940 total time=
0.5s
```

```
[CV 3/5] END learning_rate=0.7, n_estimators=100;, score=0.957 total time=
0.5s
[CV 4/5] END learning_rate=0.7, n_estimators=100;, score=0.952 total time=
0.4s
[CV 5/5] END learning_rate=0.7, n_estimators=100;, score=0.952 total time=
0.5s
[CV 1/5] END learning_rate=0.2, n_estimators=400;, score=0.966 total time=
0.8s
[CV 2/5] END learning_rate=0.2, n_estimators=400;, score=0.969 total time=
0.8s
[CV 3/5] END learning_rate=0.2, n_estimators=400;, score=0.970 total time=
0.8s
[CV 4/5] END learning_rate=0.2, n_estimators=400;, score=0.963 total time=
0.9s
[CV 5/5] END learning_rate=0.2, n_estimators=400;, score=0.968 total time=
1.5s
[CV 1/5] END learning_rate=0.2, n_estimators=100;, score=0.951 total time=
0.6s
[CV 2/5] END learning_rate=0.2, n_estimators=100;, score=0.956 total time=
0.6s
[CV 3/5] END learning_rate=0.2, n_estimators=100;, score=0.960 total time=
0.5s
[CV 4/5] END learning_rate=0.2, n_estimators=100;, score=0.945 total time=
0.5s
[CV 5/5] END learning_rate=0.2, n_estimators=100;, score=0.951 total time=
0.4s
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END .learning_rate=1, n_estimators=800;, score=0.941 total time=   1.3s
[CV 2/5] END .learning_rate=1, n_estimators=800;, score=0.950 total time=   1.7s
[CV 3/5] END .learning_rate=1, n_estimators=800;, score=0.933 total time=   1.3s
[CV 4/5] END .learning_rate=1, n_estimators=800;, score=0.922 total time=   1.3s
[CV 5/5] END .learning_rate=1, n_estimators=800;, score=0.947 total time=   1.4s
[CV 1/5] END learning_rate=0.5, n_estimators=700;, score=0.958 total time=
2.0s
[CV 2/5] END learning_rate=0.5, n_estimators=700;, score=0.966 total time=
1.6s
[CV 3/5] END learning_rate=0.5, n_estimators=700;, score=0.964 total time=
1.4s
[CV 4/5] END learning_rate=0.5, n_estimators=700;, score=0.962 total time=
1.5s
[CV 5/5] END learning_rate=0.5, n_estimators=700;, score=0.965 total time=
1.4s
[CV 1/5] END learning_rate=0.4, n_estimators=800;, score=0.963 total time=
1.6s
[CV 2/5] END learning_rate=0.4, n_estimators=800;, score=0.961 total time=
1.5s
[CV 3/5] END learning_rate=0.4, n_estimators=800;, score=0.972 total time=
1.5s
```

```
[CV 4/5] END learning_rate=0.4, n_estimators=800;, score=0.971 total time=
2.0s
[CV 5/5] END learning_rate=0.4, n_estimators=800;, score=0.962 total time=
1.4s
[CV 1/5] END .learning_rate=1, n_estimators=400;, score=0.941 total time=   1.2s
[CV 2/5] END .learning_rate=1, n_estimators=400;, score=0.950 total time=   1.0s
[CV 3/5] END .learning_rate=1, n_estimators=400;, score=0.933 total time=   1.0s
[CV 4/5] END .learning_rate=1, n_estimators=400;, score=0.922 total time=   1.5s
[CV 5/5] END .learning_rate=1, n_estimators=400;, score=0.947 total time=   1.7s
[CV 1/5] END learning_rate=0.4, n_estimators=400;, score=0.963 total time=
1.1s
[CV 2/5] END learning_rate=0.4, n_estimators=400;, score=0.960 total time=
1.1s
[CV 3/5] END learning_rate=0.4, n_estimators=400;, score=0.972 total time=
1.1s
[CV 4/5] END learning_rate=0.4, n_estimators=400;, score=0.970 total time=
1.3s
[CV 5/5] END learning_rate=0.4, n_estimators=400;, score=0.962 total time=
1.0s
[CV 1/5] END learning_rate=0.4, n_estimators=700;, score=0.963 total time=
1.3s
[CV 2/5] END learning_rate=0.4, n_estimators=700;, score=0.961 total time=
2.1s
[CV 3/5] END learning_rate=0.4, n_estimators=700;, score=0.972 total time=
1.4s
[CV 4/5] END learning_rate=0.4, n_estimators=700;, score=0.971 total time=
1.2s
[CV 5/5] END learning_rate=0.4, n_estimators=700;, score=0.962 total time=
1.4s
[CV 1/5] END learning_rate=0.4, n_estimators=1000;, score=0.963 total time=
1.9s
[CV 2/5] END learning_rate=0.4, n_estimators=1000;, score=0.961 total time=
2.0s
[CV 3/5] END learning_rate=0.4, n_estimators=1000;, score=0.972 total time=
3.5s
[CV 4/5] END learning_rate=0.4, n_estimators=1000;, score=0.971 total time=
2.4s
[CV 5/5] END learning_rate=0.4, n_estimators=1000;, score=0.962 total time=
2.2s
[CV 1/5] END learning_rate=0.7, n_estimators=1000;, score=0.946 total time=
2.0s
[CV 2/5] END learning_rate=0.7, n_estimators=1000;, score=0.962 total time=
4.3s
[CV 3/5] END learning_rate=0.7, n_estimators=1000;, score=0.966 total time=
1.8s
[CV 4/5] END learning_rate=0.7, n_estimators=1000;, score=0.948 total time=
3.0s
[CV 5/5] END learning_rate=0.7, n_estimators=1000;, score=0.947 total time=
```

```
1.7s
[CV 1/5] END learning_rate=0.5, n_estimators=100;, score=0.956 total time=
0.6s
[CV 2/5] END learning_rate=0.5, n_estimators=100;, score=0.962 total time=
0.7s
[CV 3/5] END learning_rate=0.5, n_estimators=100;, score=0.961 total time=
0.8s
[CV 4/5] END learning_rate=0.5, n_estimators=100;, score=0.958 total time=
1.1s
[CV 5/5] END learning_rate=0.5, n_estimators=100;, score=0.962 total time=
0.8s
[CV 1/5] END .learning_rate=1, n_estimators=100;, score=0.940 total time=   0.8s
[CV 2/5] END .learning_rate=1, n_estimators=100;, score=0.950 total time=   0.7s
[CV 3/5] END .learning_rate=1, n_estimators=100;, score=0.934 total time=   0.6s
[CV 4/5] END .learning_rate=1, n_estimators=100;, score=0.919 total time=   0.8s
[CV 5/5] END .learning_rate=1, n_estimators=100;, score=0.946 total time=   0.6s
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END learning_rate=0.7, n_estimators=700;, score=0.962 total time=
1.5s
[CV 2/5] END learning_rate=0.7, n_estimators=700;, score=0.969 total time=
1.4s
[CV 3/5] END learning_rate=0.7, n_estimators=700;, score=0.957 total time=
1.4s
[CV 4/5] END learning_rate=0.7, n_estimators=700;, score=0.951 total time=
1.5s
[CV 5/5] END learning_rate=0.7, n_estimators=700;, score=0.959 total time=
1.5s
[CV 1/5] END learning_rate=0.7, n_estimators=100;, score=0.960 total time=
0.9s
[CV 2/5] END learning_rate=0.7, n_estimators=100;, score=0.968 total time=
0.7s
[CV 3/5] END learning_rate=0.7, n_estimators=100;, score=0.955 total time=
0.7s
[CV 4/5] END learning_rate=0.7, n_estimators=100;, score=0.950 total time=
0.8s
[CV 5/5] END learning_rate=0.7, n_estimators=100;, score=0.957 total time=
0.6s
[CV 1/5] END .learning_rate=1, n_estimators=800;, score=0.933 total time=   1.6s
[CV 2/5] END .learning_rate=1, n_estimators=800;, score=0.949 total time=   1.9s
[CV 3/5] END .learning_rate=1, n_estimators=800;, score=0.937 total time=   1.5s
[CV 4/5] END .learning_rate=1, n_estimators=800;, score=0.930 total time=   1.5s
[CV 5/5] END .learning_rate=1, n_estimators=800;, score=0.946 total time=   1.6s
[CV 1/5] END learning_rate=0.2, n_estimators=100;, score=0.948 total time=
0.7s
[CV 2/5] END learning_rate=0.2, n_estimators=100;, score=0.959 total time=
0.7s
[CV 3/5] END learning_rate=0.2, n_estimators=100;, score=0.956 total time=
0.7s
```

```
[CV 4/5] END learning_rate=0.2, n_estimators=100;, score=0.949 total time=
0.7s
[CV 5/5] END learning_rate=0.2, n_estimators=100;, score=0.954 total time=
0.7s
[CV 1/5] END learning_rate=1, n_estimators=1000;, score=0.933 total time=   1.7s
[CV 2/5] END learning_rate=1, n_estimators=1000;, score=0.949 total time=   1.8s
[CV 3/5] END learning_rate=1, n_estimators=1000;, score=0.937 total time=   2.2s
[CV 4/5] END learning_rate=1, n_estimators=1000;, score=0.930 total time=   1.9s
[CV 5/5] END learning_rate=1, n_estimators=1000;, score=0.946 total time=   2.2s
[CV 1/5] END .learning_rate=1, n_estimators=100;, score=0.932 total time=   0.9s
[CV 2/5] END .learning_rate=1, n_estimators=100;, score=0.947 total time=   1.0s
[CV 3/5] END .learning_rate=1, n_estimators=100;, score=0.935 total time=   0.8s
[CV 4/5] END .learning_rate=1, n_estimators=100;, score=0.930 total time=   0.6s
[CV 5/5] END .learning_rate=1, n_estimators=100;, score=0.945 total time=   0.8s
[CV 1/5] END learning_rate=0.2, n_estimators=800;, score=0.968 total time=
1.6s
[CV 2/5] END learning_rate=0.2, n_estimators=800;, score=0.974 total time=
1.6s
[CV 3/5] END learning_rate=0.2, n_estimators=800;, score=0.976 total time=
1.7s
[CV 4/5] END learning_rate=0.2, n_estimators=800;, score=0.966 total time=
1.5s
[CV 5/5] END learning_rate=0.2, n_estimators=800;, score=0.973 total time=
2.2s
[CV 1/5] END learning_rate=0.2, n_estimators=400;, score=0.966 total time=
1.2s
[CV 2/5] END learning_rate=0.2, n_estimators=400;, score=0.972 total time=
1.2s
[CV 3/5] END learning_rate=0.2, n_estimators=400;, score=0.974 total time=
1.3s
[CV 4/5] END learning_rate=0.2, n_estimators=400;, score=0.964 total time=
1.1s
[CV 5/5] END learning_rate=0.2, n_estimators=400;, score=0.972 total time=
1.0s
[CV 1/5] END learning_rate=0.4, n_estimators=700;, score=0.958 total time=
1.4s
[CV 2/5] END learning_rate=0.4, n_estimators=700;, score=0.970 total time=
1.7s
[CV 3/5] END learning_rate=0.4, n_estimators=700;, score=0.970 total time=
1.4s
[CV 4/5] END learning_rate=0.4, n_estimators=700;, score=0.955 total time=
1.4s
[CV 5/5] END learning_rate=0.4, n_estimators=700;, score=0.972 total time=
1.5s
[CV 1/5] END learning_rate=0.4, n_estimators=800;, score=0.958 total time=
1.6s
[CV 2/5] END learning_rate=0.4, n_estimators=800;, score=0.970 total time=
1.6s
```

```
[CV 3/5] END learning_rate=0.4, n_estimators=800;, score=0.970 total time=
1.8s
[CV 4/5] END learning_rate=0.4, n_estimators=800;, score=0.955 total time=
1.4s
[CV 5/5] END learning_rate=0.4, n_estimators=800;, score=0.972 total time=
1.6s
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END learning_rate=0.2, n_estimators=1000;, score=0.968 total time=
1.9s
[CV 2/5] END learning_rate=0.2, n_estimators=1000;, score=0.970 total time=
2.1s
[CV 3/5] END learning_rate=0.2, n_estimators=1000;, score=0.972 total time=
2.0s
[CV 4/5] END learning_rate=0.2, n_estimators=1000;, score=0.968 total time=
1.7s
[CV 5/5] END learning_rate=0.2, n_estimators=1000;, score=0.975 total time=
2.2s
[CV 1/5] END .learning_rate=1, n_estimators=700;, score=0.951 total time=   1.5s
[CV 2/5] END .learning_rate=1, n_estimators=700;, score=0.934 total time=   1.5s
[CV 3/5] END .learning_rate=1, n_estimators=700;, score=0.937 total time=   1.4s
[CV 4/5] END .learning_rate=1, n_estimators=700;, score=0.949 total time=   1.4s
[CV 5/5] END .learning_rate=1, n_estimators=700;, score=0.937 total time=   1.4s
[CV 1/5] END learning_rate=0.7, n_estimators=800;, score=0.936 total time=
2.7s
[CV 2/5] END learning_rate=0.7, n_estimators=800;, score=0.945 total time=
1.8s
[CV 3/5] END learning_rate=0.7, n_estimators=800;, score=0.966 total time=
1.5s
[CV 4/5] END learning_rate=0.7, n_estimators=800;, score=0.958 total time=
1.6s
[CV 5/5] END learning_rate=0.7, n_estimators=800;, score=0.961 total time=
3.1s
[CV 1/5] END learning_rate=0.7, n_estimators=100;, score=0.934 total time=
0.7s
[CV 2/5] END learning_rate=0.7, n_estimators=100;, score=0.945 total time=
0.7s
[CV 3/5] END learning_rate=0.7, n_estimators=100;, score=0.965 total time=
0.8s
[CV 4/5] END learning_rate=0.7, n_estimators=100;, score=0.956 total time=
0.8s
[CV 5/5] END learning_rate=0.7, n_estimators=100;, score=0.959 total time=
0.7s
[CV 1/5] END learning_rate=0.7, n_estimators=700;, score=0.937 total time=
1.4s
[CV 2/5] END learning_rate=0.7, n_estimators=700;, score=0.945 total time=
1.4s
[CV 3/5] END learning_rate=0.7, n_estimators=700;, score=0.966 total time=
1.6s
```

```
[CV 4/5] END learning_rate=0.7, n_estimators=700;, score=0.958 total time=
1.5s
[CV 5/5] END learning_rate=0.7, n_estimators=700;, score=0.961 total time=
1.9s
[CV 1/5] END .learning_rate=1, n_estimators=100;, score=0.952 total time=   0.9s
[CV 2/5] END .learning_rate=1, n_estimators=100;, score=0.934 total time=   0.9s
[CV 3/5] END .learning_rate=1, n_estimators=100;, score=0.936 total time=   0.7s
[CV 4/5] END .learning_rate=1, n_estimators=100;, score=0.945 total time=   0.7s
[CV 5/5] END .learning_rate=1, n_estimators=100;, score=0.938 total time=   0.8s
[CV 1/5] END learning_rate=0.2, n_estimators=800;, score=0.968 total time=
1.7s
[CV 2/5] END learning_rate=0.2, n_estimators=800;, score=0.970 total time=
1.5s
[CV 3/5] END learning_rate=0.2, n_estimators=800;, score=0.972 total time=
1.5s
[CV 4/5] END learning_rate=0.2, n_estimators=800;, score=0.967 total time=
1.6s
[CV 5/5] END learning_rate=0.2, n_estimators=800;, score=0.975 total time=
1.7s
[CV 1/5] END learning_rate=0.4, n_estimators=700;, score=0.951 total time=
1.4s
[CV 2/5] END learning_rate=0.4, n_estimators=700;, score=0.967 total time=
1.5s
[CV 3/5] END learning_rate=0.4, n_estimators=700;, score=0.966 total time=
1.9s
[CV 4/5] END learning_rate=0.4, n_estimators=700;, score=0.966 total time=
1.5s
[CV 5/5] END learning_rate=0.4, n_estimators=700;, score=0.967 total time=
1.4s
[CV 1/5] END learning_rate=0.2, n_estimators=100;, score=0.949 total time=
0.8s
[CV 2/5] END learning_rate=0.2, n_estimators=100;, score=0.953 total time=
0.9s
[CV 3/5] END learning_rate=0.2, n_estimators=100;, score=0.953 total time=
0.7s
[CV 4/5] END learning_rate=0.2, n_estimators=100;, score=0.952 total time=
0.7s
[CV 5/5] END learning_rate=0.2, n_estimators=100;, score=0.956 total time=
0.7s
[CV 1/5] END learning_rate=0.4, n_estimators=100;, score=0.950 total time=
0.5s
[CV 2/5] END learning_rate=0.4, n_estimators=100;, score=0.963 total time=
0.7s
[CV 3/5] END learning_rate=0.4, n_estimators=100;, score=0.963 total time=
0.9s
[CV 4/5] END learning_rate=0.4, n_estimators=100;, score=0.957 total time=
0.7s
[CV 5/5] END learning_rate=0.4, n_estimators=100;, score=0.958 total time=
```

```
0.6s
Average cross validation R2 score: 0.9753825201956607

[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed: 10.0min finished
```

[142]:
```
joblib.dump(grid_cat,'walmart_sales_predictor.h5')
```

[142]:
```
['walmart_sales_predictor.h5']
```

[143]:
```
model = joblib.load('walmart_sales_predictor.h5')
model
```

[143]:
```
RandomizedSearchCV(cv=5,
                   estimator=<catboost.core.CatBoostRegressor object at
0x000001BE0B867DF0>,
                   param_distributions={'learning_rate': [0.2, 0.4, 0.5, 0.7,
                                                          1],
                                        'n_estimators': [100, 400, 700, 800,
                                                         1000]},
                   verbose=5)
```