P. Mishra, *Explainable AI Recipes*
https://doi.org/10.1007/978-1-4842-9029-3_6

6. Explainability for Time-Series Models

Pradeepta Mishra[1]
(1) Bangalore, Karnataka, India

A time series, as the name implies, has a time stamp and a variable that we are observing over time, such as stock prices, sales, revenue, profit over time, etc. Time-series modeling is a set of techniques that can be used to generate multistep predictions for a future time period, which will help a business to plan better and will help decision-makers to plan according to the future estimations. There are machine learning–based techniques that can be applied to generate future forecasting; also, there is a need to explain the predictions about the future.

The most commonly used techniques for time-series forecasting are auto-regressive methods, moving average methods, autoregressive and moving average methods, and deep learning–based techniques such as LSTM, etc. The time-series model requires the data to be at frequent time intervals. If there is any gap in recording, it requires a different process to address the gap in the time series. The time-series model can be looked at from two ways: univariate, which is completely dependent on time, and multivariate, which takes into account various factors. Those factors are called *causal factors*, which impact the predictions. In the time-series model, the time is an independent variable, so we can compute various features from the time as an independent feature. Time-series modeling has various components such as trend, seasonality, and cyclicity.

Recipe 6-1. Explain Time-Series Models Using LIME

Problem

You want to explain a time-series model using LIME.

Solution

We are taking into consideration a sample dataset that has dates and prices, and we are going to consider only the univariate analysis. We will be using the LIME library to explain the predictions.

How It Works

Let's take a look at the following example (see Figure 6-1):

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
df = pd.read_csv('https://raw.githubusercontent.com/pradmishra1/PublicDatasets/main/monthly_csv
# seasonal difference
differenced = df.diff(12)
# trim off the first year of empty data
differenced = differenced[12:]
# save differenced dataset to file
differenced.to_csv('seasonally_adjusted.csv', index=False)
# plot differenced dataset
differenced.plot()
plt.show()
```

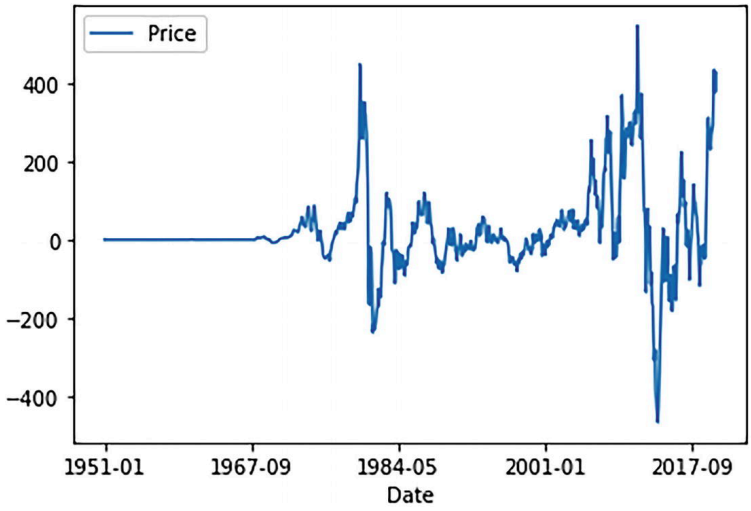

Figure 6-1 Seasonally adjusted difference plot

```
# reframe as supervised learning
dataframe = pd.DataFrame()
for i in range(12,0,-1):
    dataframe['t-'+str(i)] = df.shift(i).values[:,0]
dataframe['t'] = df.values[:,0]
print(dataframe.head(13))
dataframe = dataframe[13:]
# save to new file
dataframe.to_csv('lags_12months_features.csv', index=False)
```

For the last 12 months, lagged features will be used as training features to forecast the future time-series sales values.

```
# split into input and output
df = pd.read_csv('lags_12months_features.csv')
data = df.values
X = data[:,0:-1]
y = data[:,-1]
from sklearn.ensemble import RandomForestRegressor
# fit random forest model
model = RandomForestRegressor(n_estimators=500, random_state=1)
model.fit(X, y)
```

We are using a random forest regressor to consider the importance of each feature in a subset scenario. See Figure 6-2.

```
# show importance scores
print(model.feature_importances_)
# plot importance scores
names = dataframe.columns.values[0:-1]
ticks = [i for i in range(len(names))]
plt.bar(ticks, model.feature_importances_)
```

```
plt.xticks(ticks, names)
plt.show()
```

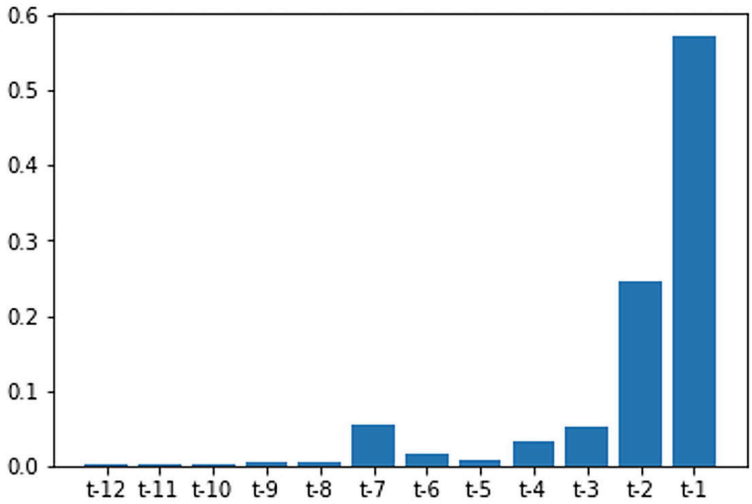

Figure 6-2 Feature importance for lagged features from the 12 lagged features

```
from sklearn.feature_selection import RFE
```

Recursive feature elimination is a technique usually used to fine-tune relevant features from the available list of features so that only important features can go into the inference generation process.

```
# perform feature selection
rfe = RFE(RandomForestRegressor(n_estimators=500, random_state=1), n_features_to_select=4)
fit = rfe.fit(X, y)
# report selected features
print('Selected Features:')
names = dataframe.columns.values[0:-1]
for i in range(len(fit.support_)):
    if fit.support_[i]:
        print(names[i])
Selected Features:
t-7
t-3
t-2
t-1
```

We can rank the time-aware important features, which are lags. See Figure 6-3 and Figure 6-4.

```
# plot feature rank
names = dataframe.columns.values[0:-1]
ticks = [i for i in range(len(names))]
plt.bar(ticks, fit.ranking_)
plt.xticks(ticks, names)
plt.show()
```

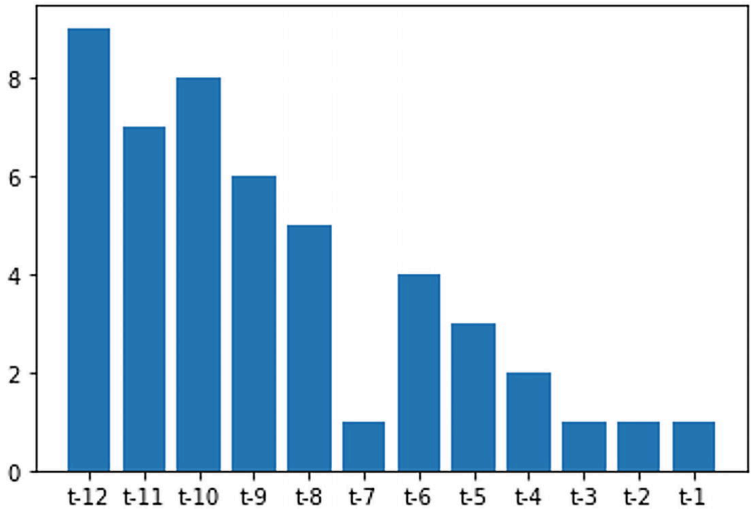

Figure 6-3 Feature ranking from all available lags

```
!pip install Lime
import lime
import lime.lime_tabular
explainer = lime.lime_tabular.LimeTabularExplainer(np.array(X),
                                                   mode='regression',
                                                  feature_names=X.columns,
                                                  class_names=['t'],
                                                  verbose=True)
explainer.feature_frequencies
{0: array([0.25659472, 0.24340528, 0.24940048, 0.25059952]), 1: array([0.25539568, 0.24460432, 0
# asking for explanation for LIME model
i = 60
exp = explainer.explain_instance(np.array(X)[i],
                                 new_model.predict,
                                 num_features=12
                                )
Intercept 524.1907857658252
Prediction_local [76.53408383]
Right: 35.77034850521053
X does not have valid feature names, but LinearRegression was fitted with feature names
exp.show_in_notebook(show_table=True)
```

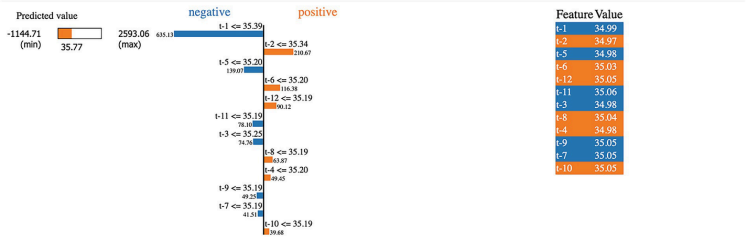

Figure 6-4 Local interpretation for time series

For the 60th record from the dataset, the predicted value is 35.77, for which lag 1 is the most important feature.

```
exp.as_list()
[('t-1 <= 35.39', -635.1332339969734), ('t-2 <= 35.34', 210.66614528187935), ('t-5 <= 35.20', -1
# Code for SP-LIME
import warnings
from lime import submodular_pick
# Remember to convert the dataframe to matrix values
# SP-LIME returns exaplanations on a sample set to provide a non redundant global decision bound
sp_obj = submodular_pick.SubmodularPick(explainer, np.array(X),
                                        new_model.predict,
                                        num_features=12,
```

```
                                        num_exps_desired=10)
```

The SP-LIME module from the LIME library provides explanations on a sample set to provide a global decision boundary about the prediction. In the previous script, we are considering the time-series model as a supervised learning model and using 12 lags as features. From the LIME library, we are using the LIME tabular explainer. The following script shows the explanation of record number 60. The predicted value is 35.77, and the lower threshold value and upper threshold value reflect the confidence band of the predicted outcome. Figure 6-5 shows the positive factors and negative factors contributing toward the prediction.

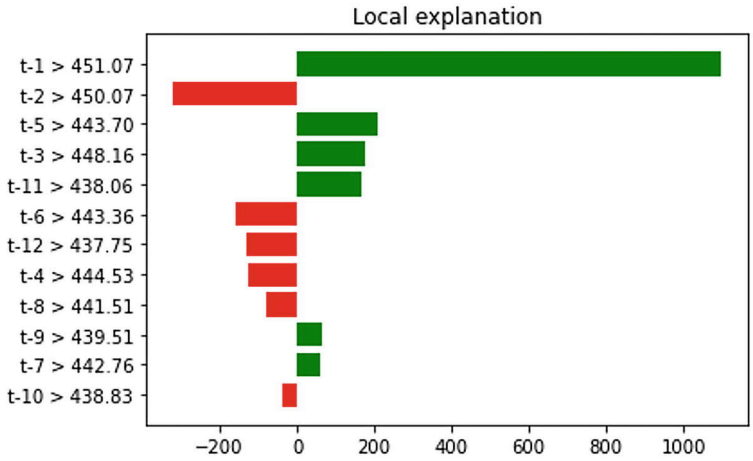

Figure 6-5 The local explanation shows positive features in green and negative in red

Recipe 6-2. Explain Time-Series Models Using SHAP

Problem

You want to explain the time-series model using SHAP.

Solution

We are taking into consideration a sample dataset that has dates and prices, and we are going to consider only the univariate analysis. We will be using the SHAP library to explain the predictions.

How It Works

Let's take a look at the following example (Figure 6-6):

```
import shap
from sklearn.ensemble import RandomForestRegressor
rforest = RandomForestRegressor(n_estimators=100, random_state=0)
rforest.fit(X, y)
# explain all the predictions in the test set
explainer = shap.TreeExplainer(rforest)
shap_values = explainer.shap_values(X)
shap.summary_plot(shap_values, X)
```

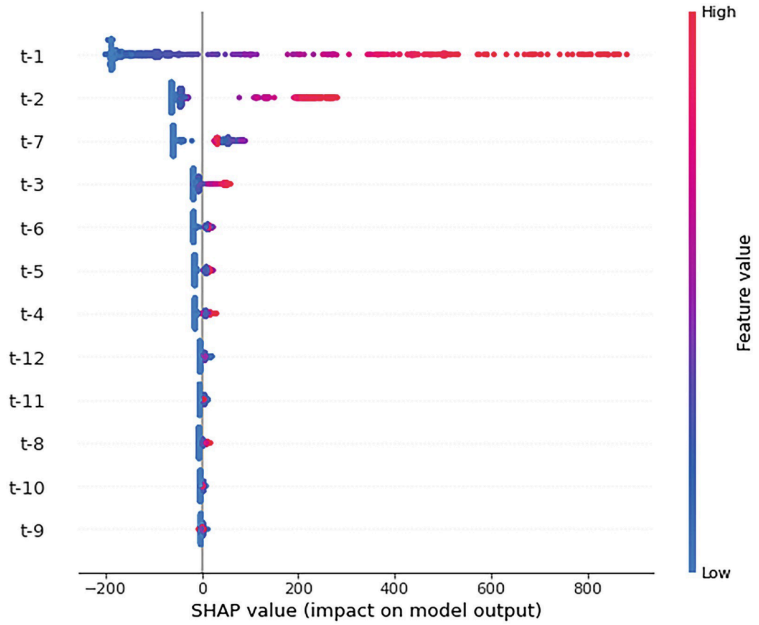

Figure 6-6 Summary plot of SHAP feature values

t-1, t-2, and t-7 are the three important features that impact the predictions. t-1 means a lag of the last time period, t-2 means a lag of the past two time periods, and t-7 means a lag of the seventh time period. Let's say data is available at a monthly level, so t-1 means last month, t-2 means the second month in the past, and t-7 means the seventh month in the past. These values impact the predictions. See Figure 6-7 and Figure 6-8.

```
shap.dependence_plot("t-1", shap_values, X)
```

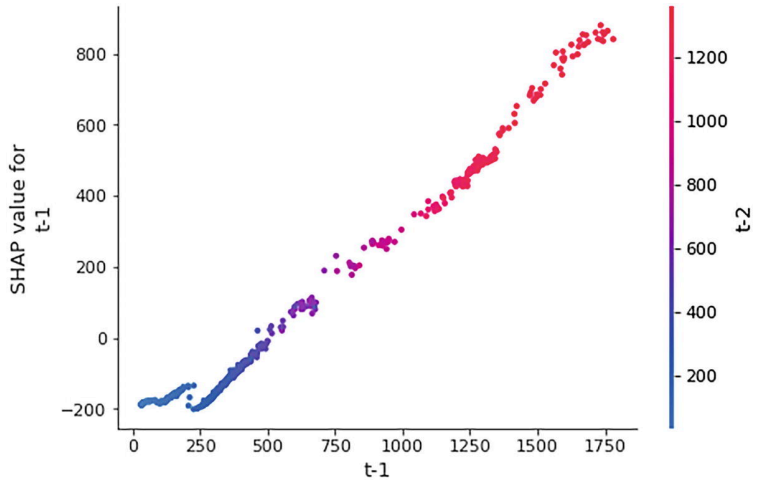

Figure 6-7 SHAP dependence plot

```
shap.partial_dependence_plot(
    "t-1", rforest.predict, X, ice=False,
    model_expected_value=True, feature_expected_value=True
)
```

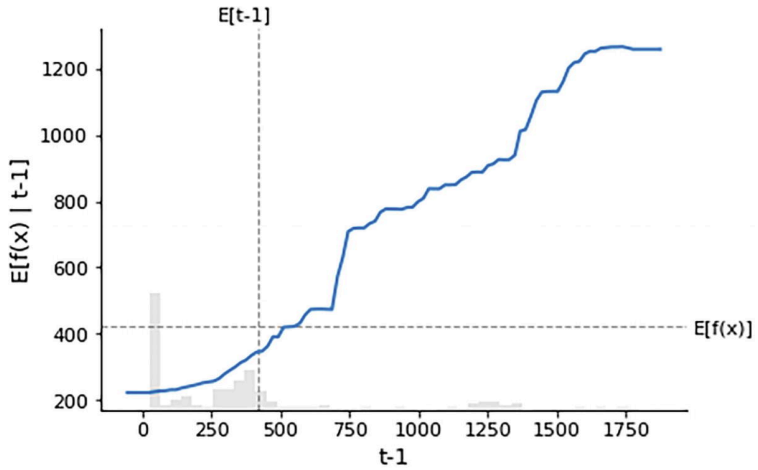

Figure 6-8 Partial dependence plot for feature t-1

Conclusion

In this chapter, we covered how to interpret a time-series model to generate a forecast. To interpret a univariate time-series model, we considered it as a supervised learning problem by taking the lags as trainable features. These features are then trained using a linear regressor, and the regression model is used to generate explanations at a global level as well as at a local level using both the SHAP and LIME libraries. A similar explanation can be generated using more complex algorithms such as the nonlinear and ensemble techniques, and finally similar kinds of graphs and charts can be generated using SHAP and LIME as in the previous chapters. The next chapter contains recipes to explain deep neural network models.