



A Combined Finite State Machine and PlantUML Approach to Machine Learning Applications

Mircea Trifan, Bogdan Ionescu and Dan Ionescu

School of Electrical Engineering and Computer Science, University of Ottawa,
Ottawa, Ontario, Canada

{mircea, bogdan, dan}@ncct.uottawa.ca



Agenda

1. Requirements
2. A FSM Architecture for Explanatory AI
3. Frameworks: AI, AutoML and AutoGluon
4. Automated Tools: PlantUML DSL, PyParsing and FSM/Miros
5. An Example: AutoGluon Time Series Automation
6. Datasets: PlantUML Diagrams
7. Conclusion



Requirements

I. Requirements for the **FSM-driven AutoML**

1. Exchange data files with any of the existing AutoML platforms;
2. Manipulate PlantUML diagrams in any shape possible;
3. Apply logic operations on PlantUML diagrams;
4. Apply the reasoning logic on FSM diagrams represented in PlantUML;
5. Run the code associated with any PlantUML diagrams;
6. Apply ElasticSearch on the PlantUML diagram repositories;



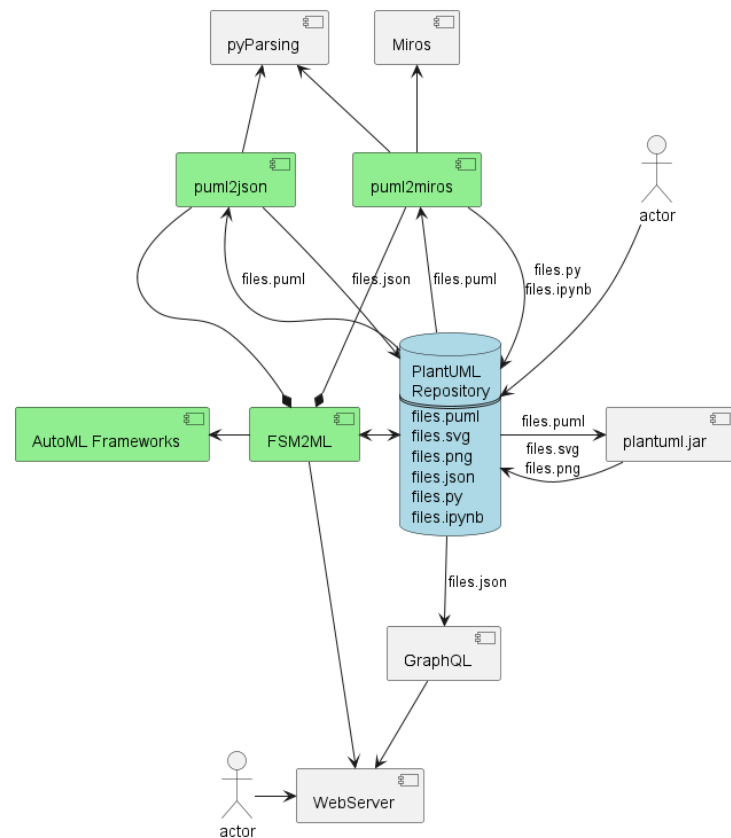
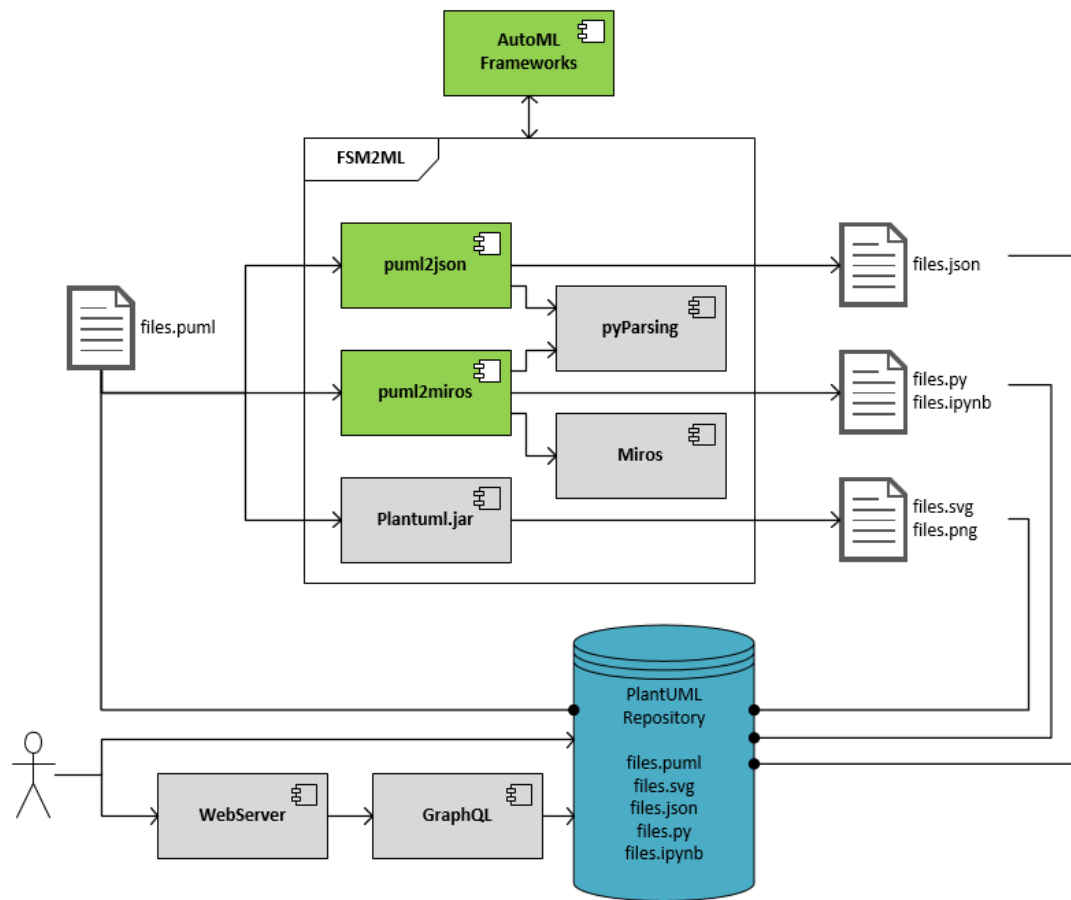
Requirements

II. Requirements for the UI/UX

7. Browse existing diagrams;
8. Select components via a pin-down mechanism;
9. Display running results inside PlantUML diagrams;
10. Reuse fragments of existing PlantUML diagrams;
11. Expand/collapse PlantUML nested components;
12. Expand/collapse headings sections within a component;
13. Expand/collapse special "more" nodes in order to zoom in/out parts;
14. The above requirements apply verbatim to FSM diagrams;



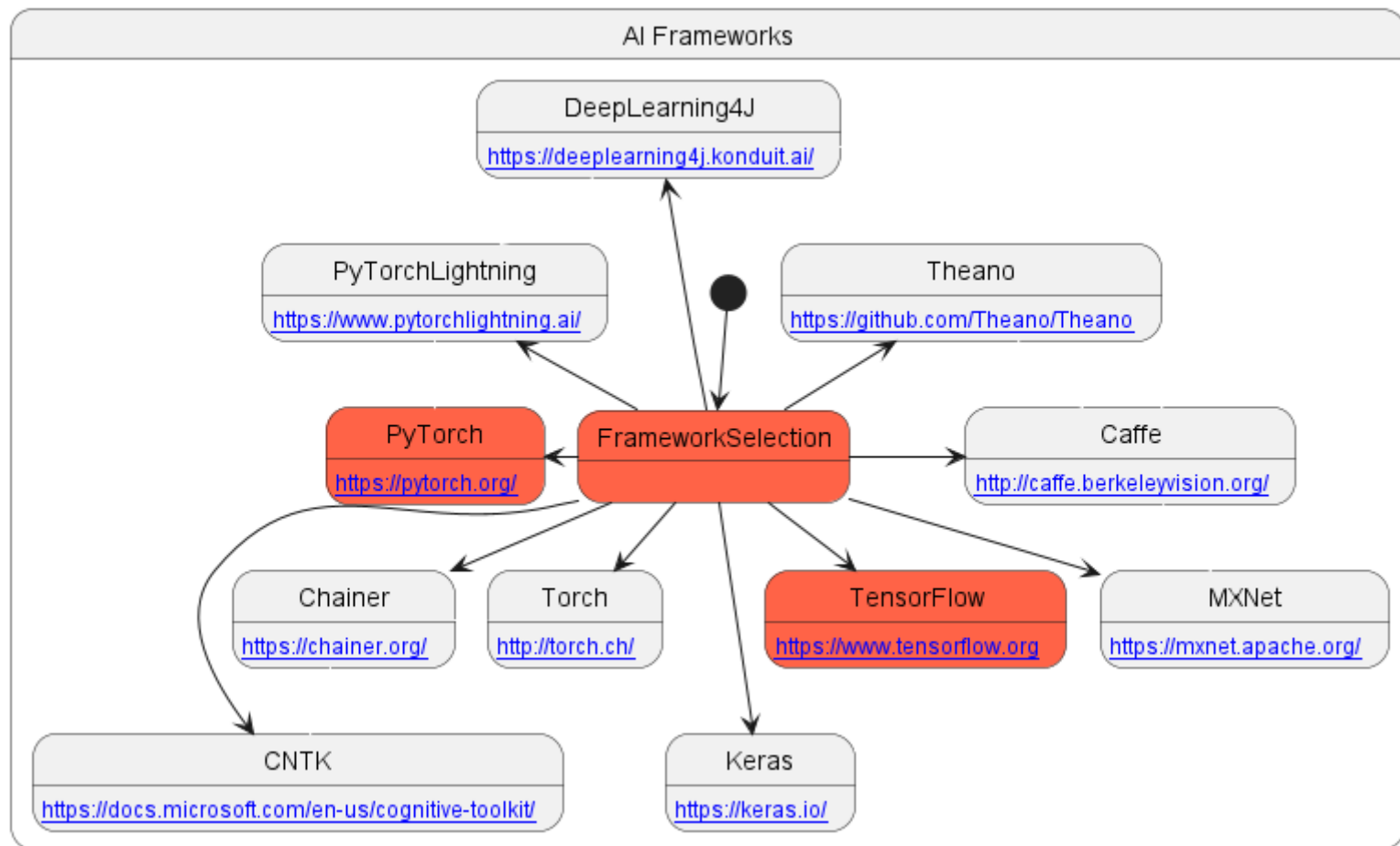
A FSM Architecture for Explanatory AI



PlantUML



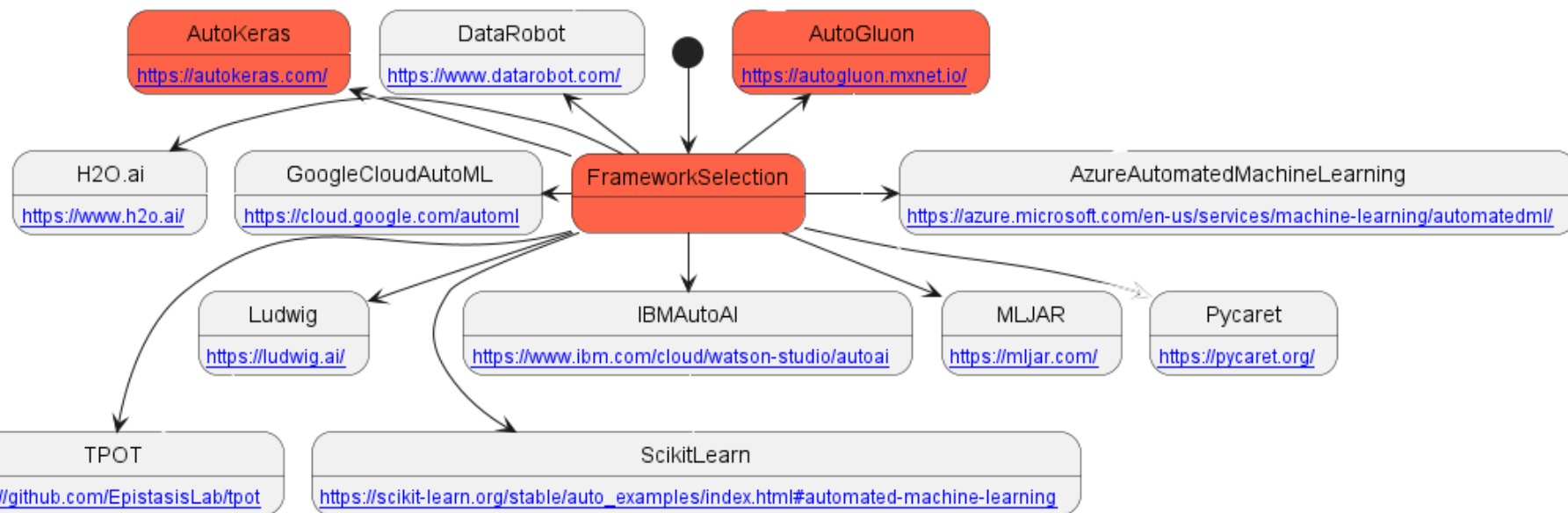
AI Frameworks





AutoML Frameworks

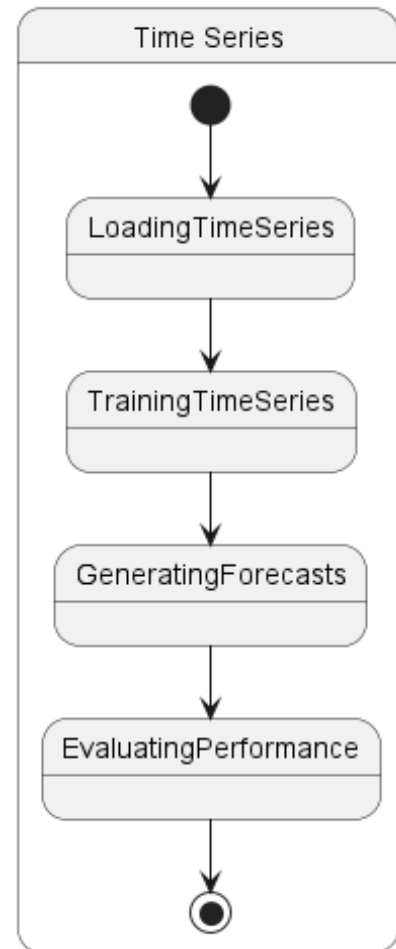
AutoML Frameworks





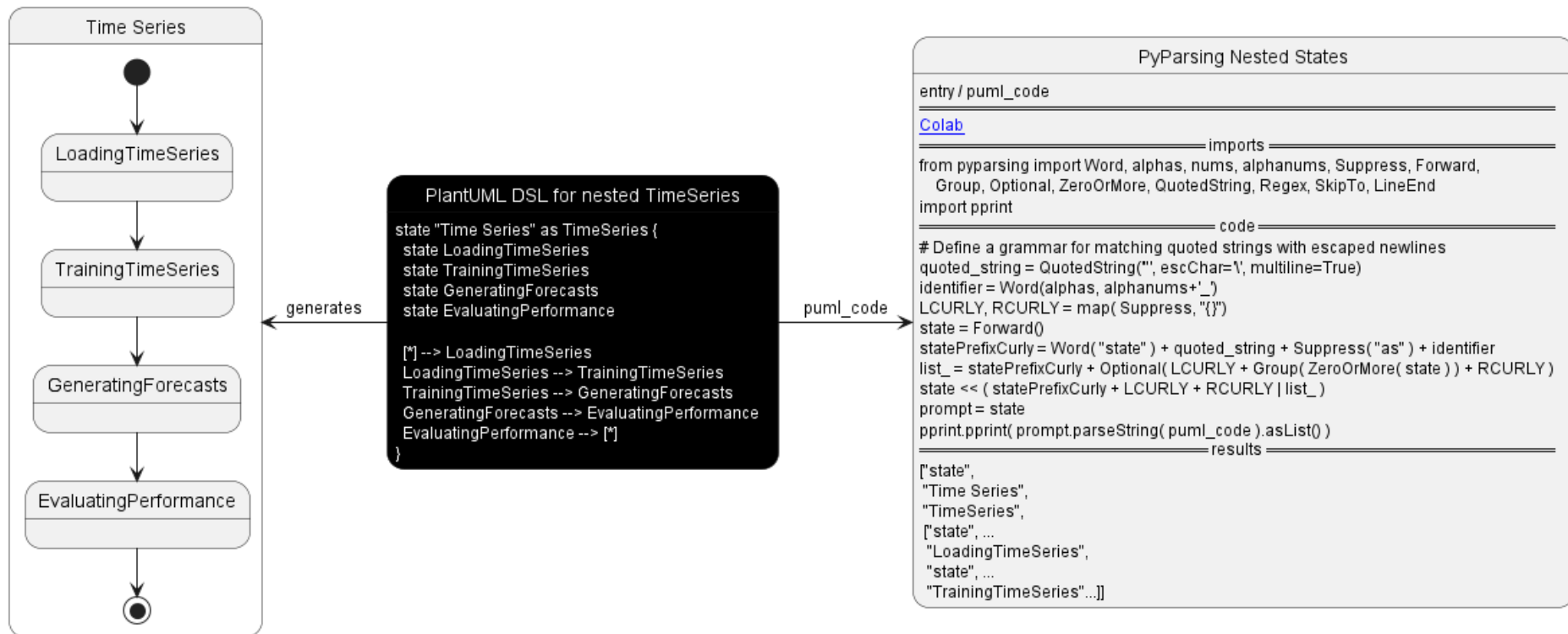
Tools :: PlantUML DSL

```
1  @startuml
2
3  state "Time Series" as TimeSeries {
4      state LoadingTimeSeries
5      state TrainingTimeSeries
6      state GeneratingForecasts
7      state EvaluatingPerformance
8
9      [*] --> LoadingTimeSeries
10     LoadingTimeSeries --> TrainingTimeSeries
11     TrainingTimeSeries --> GeneratingForecasts
12     GeneratingForecasts --> EvaluatingPerformance
13     EvaluatingPerformance --> [*]
14 }
15
16 @enduml
```





Tools :: PlantUML DSL and PyParsing Transformation





Tools :: PyParsing Nested States

Headings
(Sections)

Results: Nested Python List

Python Code

Input

Link to Colab

Imports

PyParsing Nested States

entry / puml_code

[Colab](#)

```
===== imports =====  
from pyarsing import Word, alphas, nums, alphanums, Suppress, Forward,  
    Group, Optional, ZeroOrMore, QuotedString, Regex, SkipTo, LineEnd  
import pprint
```

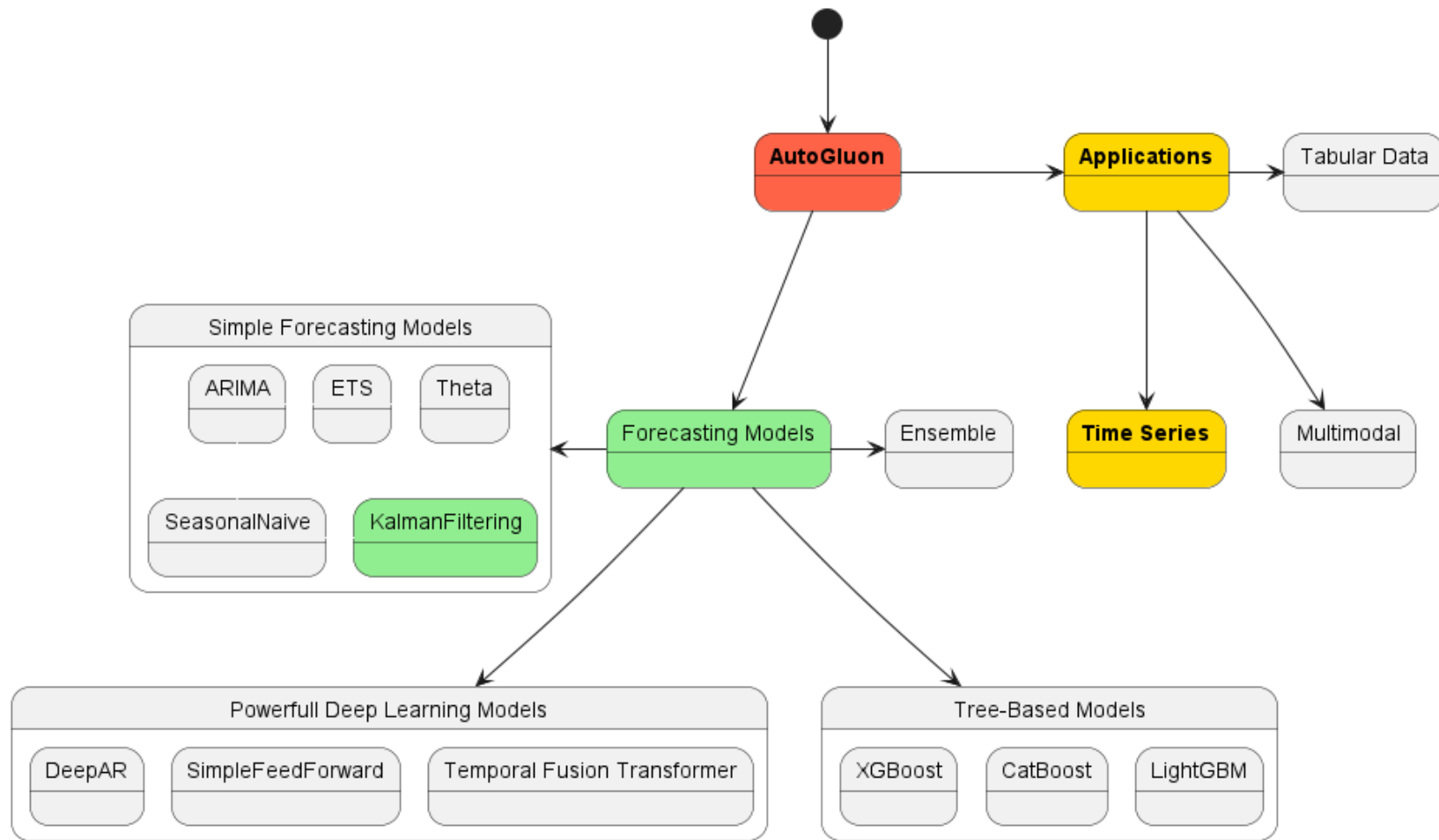
```
===== code =====  
# Define a grammar for matching quoted strings with escaped newlines  
quoted_string = QuotedString("'", escChar='\\', multiline=True)  
identifier = Word(alphas, alphanums+'_')  
LCURLY, RCURLY = map( Suppress, "{}")  
state = Forward()  
statePrefixCurly = Word("state") + quoted_string + Suppress("as") + identifier  
list_ = statePrefixCurly + Optional( LCURLY + Group( ZeroOrMore( state ) ) + RCURLY )  
state << ( statePrefixCurly + LCURLY + RCURLY | list_ )  
prompt = state  
pprint.pprint( prompt.parseString( puml_code ).asList() )
```

===== results =====

```
['state',  
 "Time Series",  
 "TimeSeries",  
 ['state', ...  
  "LoadingTimeSeries",  
  "state", ...  
  "TrainingTimeSeries"...]]
```

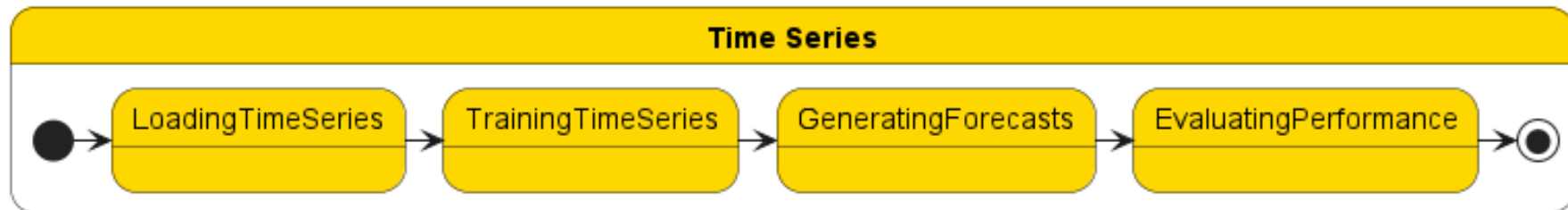


FSM for AutoGluon Models and Applications





AutoGluon :: Applications :: TimeSeries





AutoGluon :: Applications :: TimeSeries :: LoadingTimeSeries

Time Series

Loading Time Series

data as a TimeSeriesDataFrame

```
entry / "https://autogluon.s3.amazonaws.com/datasets/timeseries/m4_hourly_subset/train.csv"
exit / train_data
```

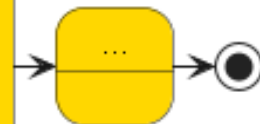
[Colab](#)

```
!pip install autogluon.timeseries
```

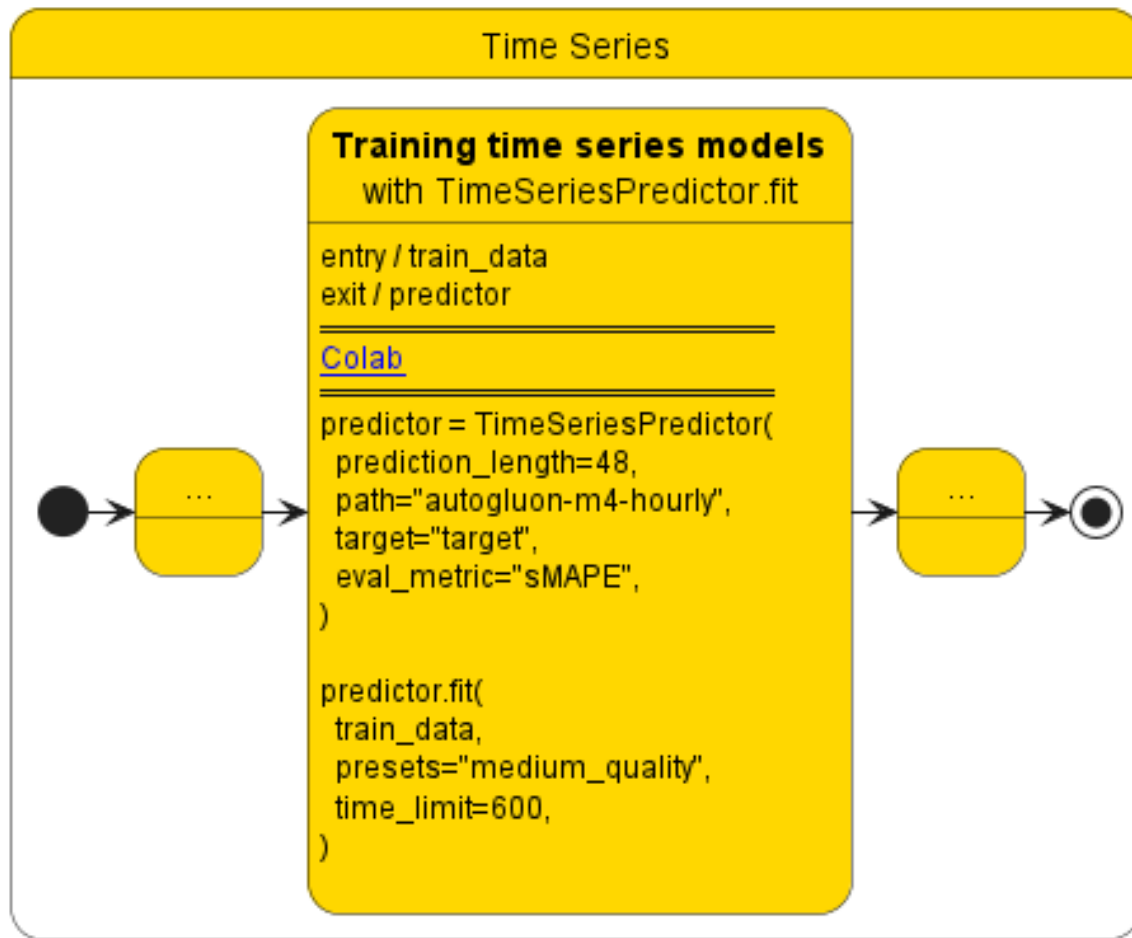
```
import pandas as pd
from autogluon.timeseries import TimeSeriesDataFrame, TimeSeriesPredictor
```

```
df = pd.read_csv("https://autogluon.s3.amazonaws.com/datasets/timeseries/m4_hourly_subset/train.csv")
df.head()
```

```
train_data = TimeSeriesDataFrame.from_data_frame(
    df,
    id_column="item_id",
    timestamp_column="timestamp"
)
train_data.head()
```



AutoGluon :: Applications :: Time Series :: Training



AutoGluon :: Applications :: Time Series :: GeneratingForecasts

Time Series

GeneratingForecasts

```
entry / predictor
entry / "https://autogluon.s3.amazonaws.com/datasets/timeseries/m4_hourly_subset/test.csv"
exit / predictor
exit / test_data
```

[Colab](#)

```
predictions = predictor.predict(train_data)
predictions.head()
```

```
import matplotlib.pyplot as plt

# TimeSeriesDataFrame can also be loaded directly from a file
test_data = TimeSeriesDataFrame.from_path("https://autogluon.s3.amazonaws.com/datasets/timeseries/m4_hourly_subset/test.csv")

plt.figure(figsize=(20, 3))

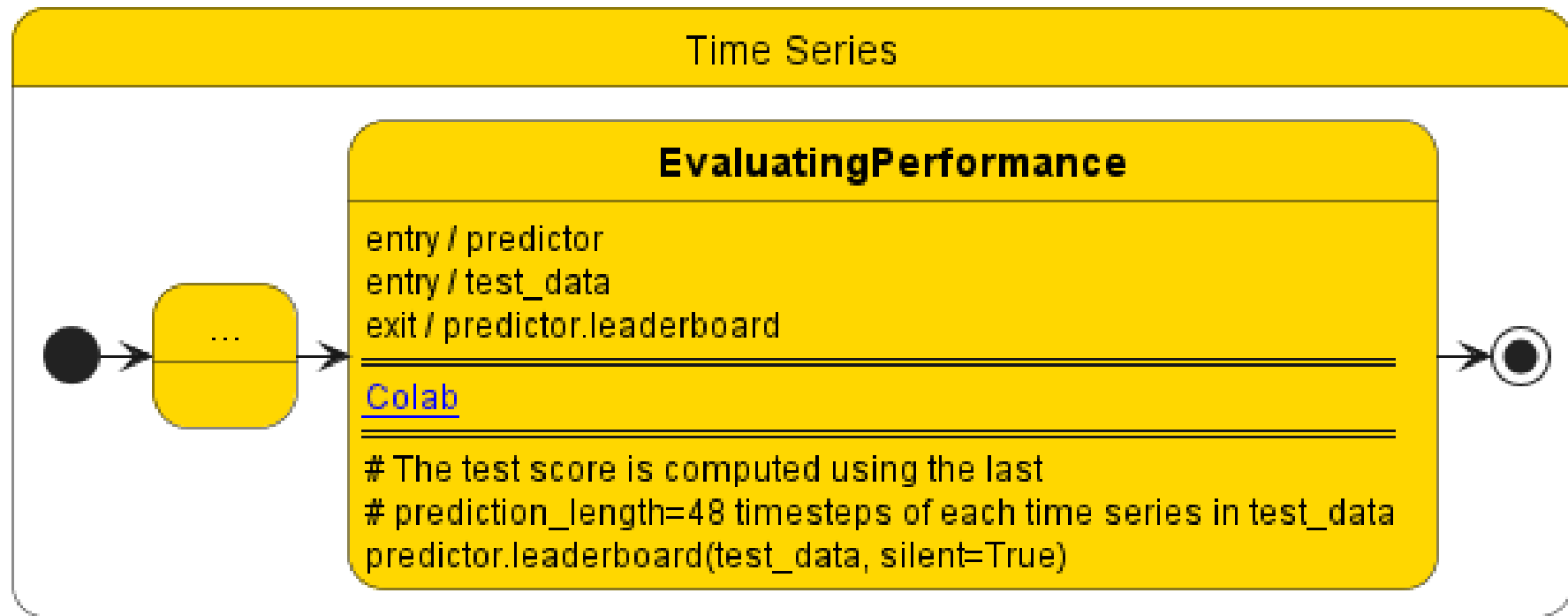
item_id = "H1"
y_past = train_data.loc[item_id]["target"]
y_pred = predictions.loc[item_id]
y_test = test_data.loc[item_id]["target"][-48:]

plt.plot(y_past[-200:], label="Past time series values")
plt.plot(y_pred["mean"], label="Mean forecast")
plt.plot(y_test, label="Future time series values")

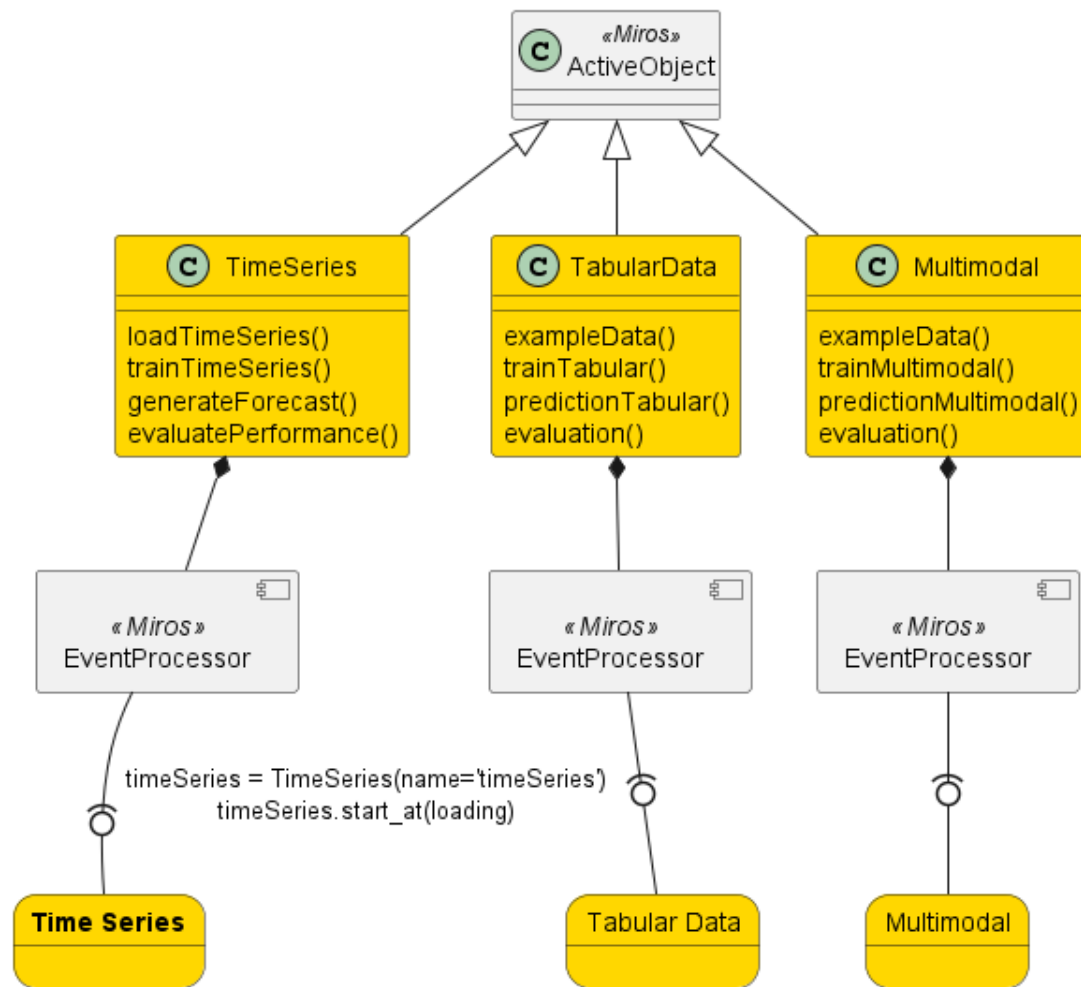
plt.fill_between(
    y_pred.index, y_pred["0.1"], y_pred["0.9"], color="red", alpha=0.1, label=f"10%-90% confidence interval"
)

plt.legend();
```

AutoGluon :: Applications :: TimeSeries :: EvaluatingPerformance

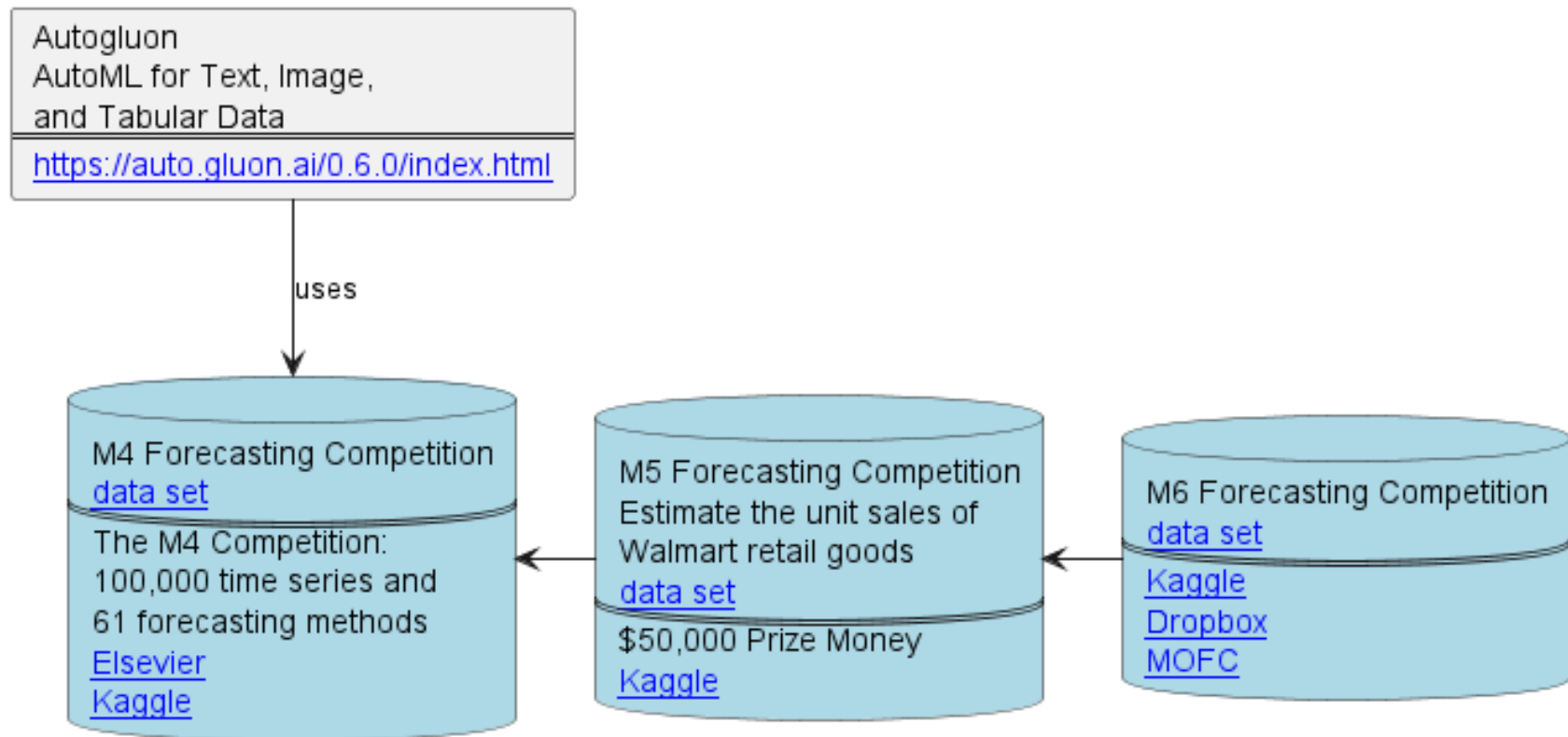


Tools :: Miros





Datasets





More Datasets

- Generating PlantUML diagrams = A Generative FSM Tool
- Online Diagrams Corpus Counter:

236,999,568 - diagrams

```
@startuml
state "A" as stateA
state "C" as stateC {
  state B
}
json foo1 {
  "foo2": "foo3"
}
@enduml
```

<http://www.plantuml.com/plantuml/png/SolwkIimgAStDuG8oIb8Lb1oL51AB5S0> Decode URL

Submit [Pure Javascript](#) NEW!

[PNG](#) [SVG](#) [ASCII Art](#)

chat 39 online online diagrams 236,999,568 current rate 174 diag. per minute

peak rate 1004 diag. per minute

```
graph LR
    A((A))
    subgraph C [C]
        B((B))
    end
    foo1[foo1  
foo2: foo3]
```



Conclusion

1. FSM frameworks and tools are built for AI and AutoML (**AutoGluon**) development.
2. A **FSM architecture** for explanatory AI provides powerful insights into model decision-making.
3. Tools such as **PlantUML**, **PyParsing**, and **Miros** are used to build complex FSM models.
4. **Datasets** are an essential part of the development process.
5. The **AutoGluon Time Series** used for forecasting are represented in FSM form.

References

- S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “The m4 competition: 100,000 time series and 61 forecasting methods,” *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020.
- R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, D. Pedreschi, and F. Giannotti, “A Survey Of Methods For Explaining Black Box Models.” *arXiv*, Jun. 21, 2018. Accessed: Apr. 12, 2023. [Online]. Available: <http://arxiv.org/abs/1802.01933>
- “AutoGluon 0.7.0 documentation,” <https://auto.gluon.ai/stable/index.html>, accessed: Apr. 19, 2023.
- F. Hutter, L. Kotthoff, and J. Vanschoren, “Automated Machine Learning: Methods, Systems, Challenges”, 1st ed. 2019 edition. Springer, 2019.