ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

Leveraging PyCaret for Time Series Analysis-A Low Code Approach

Karthika Gopalakrishnan

Data Scientist, USA

ABSTRACT

Time series analysis is a fundamental task in various domains, ranging from finance to healthcare and beyond. Traditional methods for time series analysis often require significant manual effort and expertise. PyCaret, a low-code machine learning library, offers a simplified approach to time series analysis, enabling practitioners to build robust models with minimal code. In this paper, we delve into PyCaret's capabilities for time series analysis, exploring its methods and comparing them with traditional Python packages. Through examples and case studies, we demonstrate how PyCaret streamlines the time series analysis workflow, making it accessible to a broader audience.

***Corresponding author**

Karthika Gopalakrishnan, Data Scientist, USA.

Received: September 15, 2023; **Accepted:** September 21, 2023; **Published:** September 29, 2023

Keywords: PyCaret, Time Series, Banking Industry, Credit Risk Assessment, Fraud Detection, Machine Learning Automation

Introduction

Time series analysis is a critical aspect of data science, involving the exploration, modeling, and prediction of data points collected over time. From financial markets to weather patterns and healthcare trends, time series data is ubiquitous across various domains. Traditionally, performing time series analysis demanded a profound understanding of statistical methods, programming languages, and domain-specific knowledge. Moreover, the process often involved laborious manual effort, making it inaccessible to those without specialized expertise.

In recent years, the rise of machine learning has revolutionized the way we approach time series analysis. Machine learning techniques offer powerful tools for extracting patterns, making forecasts, and uncovering insights from time-varying data. However, adopting these techniques typically requires a steep learning curve and significant investment of time and resources.

PyCaret is an open-source Python library designed to democratize machine learning and simplify the time series analysis process. PyCaret stands out as a low-code alternative to traditional methods, enabling users to perform complex analytical tasks with minimal coding. By abstracting away the complexities of model building and evaluation, PyCaret empowers practitioners of all skill levels to harness the power of machine learning for time series analysis.

With PyCaret, users can leverage a comprehensive suite of tools and algorithms to explore, model, and predict time series data. Whether you're a seasoned data scientist or a novice analyst, PyCaret provides an intuitive interface for tackling diverse time series challenges. From data preprocessing and feature engineering to model selection and evaluation, PyCaret streamlines the entire analytical workflow, allowing users to focus on extracting insights rather than wrestling with code.

The accessibility and ease of use offered by PyCaret have democratized time series analysis, opening doors for a wider range of practitioners to harness the power of machine learning. By lowering the barrier to entry and accelerating the pace of analysis, PyCaret is reshaping the landscape of time series forecasting, driving innovation, and empowering individuals and organizations to make data-driven decisions with confidence.

PyCaret: Streamlining Machine Learning Workflows

PyCaret revolutionizes machine learning by offering a simplified and automated approach. It tackles the challenges of traditional workflows, where success often hinges on manual expertise. PyCaret automates numerous tasks, transforming the data-to-insights journey.

Focus on User Efficiency

Reduced Manual Work: PyCaret automates data preprocessing, feature selection, model training, hyperparameter tuning, and evaluation. This frees users from tedious tasks, allowing them to focus on analysis and interpretation. Users can achieve complex tasks with minimal code, saving valuable time.

Extensive Toolkit

Variety of Algorithms: PyCaret offers a comprehensive library of supervised and unsupervised learning algorithms. Users can choose from traditional models like linear regression and decision trees to cutting-edge techniques like gradient boosting and deep learning. PyCaret's constantly evolving arsenal ensures it stays current with the latest advancements in machine learning.

Unified Interface

Consistent Workflow: PyCaret provides a unified interface across various machine learning tasks, including classification, regression, clustering, and anomaly detection. Users interact with the library using a consistent set of commands, regardless of the chosen algorithm. This consistency simplifies learning, promotes agility, and allows users to switch between techniques effortlessly.

Democratizing Machine Learning

Power for All: PyCaret empowers individuals and organizations by simplifying machine learning. It removes complexity and provides automation, making data analysis accessible to a wider audience. Whether you're a data science expert or a beginner, PyCaret unlocks the potential of your data, paving the way for a new era of discovery.

Time Series Analysis using PyCaret

In PyCaret, the TimeSeriesForecaster module serves as a comprehensive toolkit for time series forecasting, offering a rich array of methods to streamline the analytical workflow. Below, we elaborate on the main methods available within this module.

This pivotal method initializes the environment for time series forecasting, setting the stage for subsequent analysis. Upon invocation, it automatically detects the time column within the dataset and orchestrates essential preprocessing steps. This includes handling missing values, generating lag features to capture temporal dependencies, and partitioning the data into training and testing sets. By automating these preparatory tasks, the setup method expedites the analytical process and ensures that the data is primed for modeling. Figure 1 shows the usage of setup method of PyCaret and Figure 2 shows the output of the setup method.

```
exp = TSForecastingExperiment()
exp.setup(data=df, target="Amount", fh=12, session_id=42, fig_kwargs = {"template": "simple_white","hoverinfo":"none"})
```

Figure 1: Setup Method – PyCaret

	Description	Value
0	session_id	42
1	Target	Amount
2	Approach	Univariate
3	Exogenous Variables	Present
4	Original data shape	(1769, 2)
5	Transformed data shape	(1769, 2)
6	Transformed train set shape	(1757, 2)
7	Transformed test set shape	(12, 2)
8	Rows with missing values	0.0%
9	Fold Generator	ExpandingWindowSplitter
10	Fold Number	3
11	Enforce Prediction Interval	False
12	Splits used for hyperparameters	all
13	User Defined Seasonal Period(s)	None
14	Ignore Seasonality Test	False
15	Seasonality Detection Algo	auto
16	Max Period to Consider	60
17	Seasonal Period(s) Tested	[5]
18	Significant Seasonal Period(s)	[5]
19	Significant Seasonal Period(s) without Harmonics	[5]
20	Remove Harmonics	False
21	Harmonics Order Method	harmonic_max
22	Num Seasonalities to Use	1
23	All Seasonalities to Use	[5]
24	Primary Seasonality	5
25	Seasonality Present	True
26	Seasonality Type	add
27	Target Strictly Positive	False
28	Target White Noise	No
29	Recommended d	1
30	Recommended Seasonal D	0

Figure 2: Output - Setup Method

PyCaret Models

compare_models

This method facilitates informed decision-making by systematically evaluating the performance of various time series forecasting models. Leveraging cross-validation techniques, it trains and assesses multiple models on the training data, yielding a comprehensive summary of performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared. By providing a holistic view of model performance, the compare_models method empowers users to identify the most suitable algorithm for their specific forecasting task.

Figure 3 shows the usage of the compare_models method

```
top4 = exp.compare_models(n_select = 4)
```

	Model	MASE	RMSSE	MAE	RMSE	MAPE	SMAPE	R2	TT (Sec)
croston	Croston	1.0333	0.7167	1369492.1052	1595792.3138	35.0588	0.9976	-0.4870	0.0533

Figure 3: Compare_models - PyCaret

create_models

Building upon the insights gleaned from model comparison, the create_model method enables users to instantiate a specific time series forecasting model using a designated algorithm. By fitting the model to the training data, it constructs a trained model object that encapsulates the learned patterns and relationships within the temporal dataset. This trained model serves as a powerful tool for making accurate predictions on new, unseen data.

```
croston_ml = exp.create_model('croston')
```

	cutoff	MASE	RMSSE	MAE	RMSE	MAPE	SMAPE	R2
0	1720.0000	1.3071	0.8902	1723125.2959	1954578.6975	43.4648	1.0343	-0.4557
1	1732.0000	1.0976	0.8154	1459943.7898	1833595.7281	47.1541	1.0326	-0.0740
2	1744.0000	0.6951	0.4445	925407.2297	999202.5159	14.5576	0.9259	-0.9314
Mean	nan	1.0333	0.7167	1369492.1052	1595792.3138	35.0588	0.9976	-0.4870
SD	nan	0.2539	0.1949	331888.1928	424734.2385	14.5746	0.0507	0.3507

Figure 4: Create_model - PyCaret

tune_model

To optimize the predictive performance of a time series forecasting model, the tune_model method facilitates hyperparameter tuning through rigorous experimentation. Employing techniques such as grid search or random search, it systematically explores a predefined space of hyperparameters, evaluating each combination's efficacy through cross-validation. By iteratively refining the model's configuration, the tune_model method enhances its predictive capabilities, maximizing its utility in real-world applications.

evaluate_model

Once a time series forecasting model has been trained and tuned, the evaluate_model method offers a comprehensive assessment of its performance on the test data. Through the generation of evaluation plots such as actual vs. predicted values, residual plots, and error distributions, it provides valuable insights into the model's predictive accuracy and reliability. By visualizing the model's behavior and identifying areas for improvement, the evaluate_model method empowers users to refine their forecasting strategies iteratively.

plot_model

Complementing the quantitative evaluation provided by the evaluate_model method, the plot_model method offers intuitive visualizations of the model's predictions on the test data. By

generating interactive plots such as time series plots with actual vs. predicted values, residual plots, and trend plots, it facilitates a deeper understanding of the model's behavior and performance. These visualizations serve as invaluable aids in interpreting the model's predictions and guiding decision-making processes. Figure 5 shows the output of Plot_models

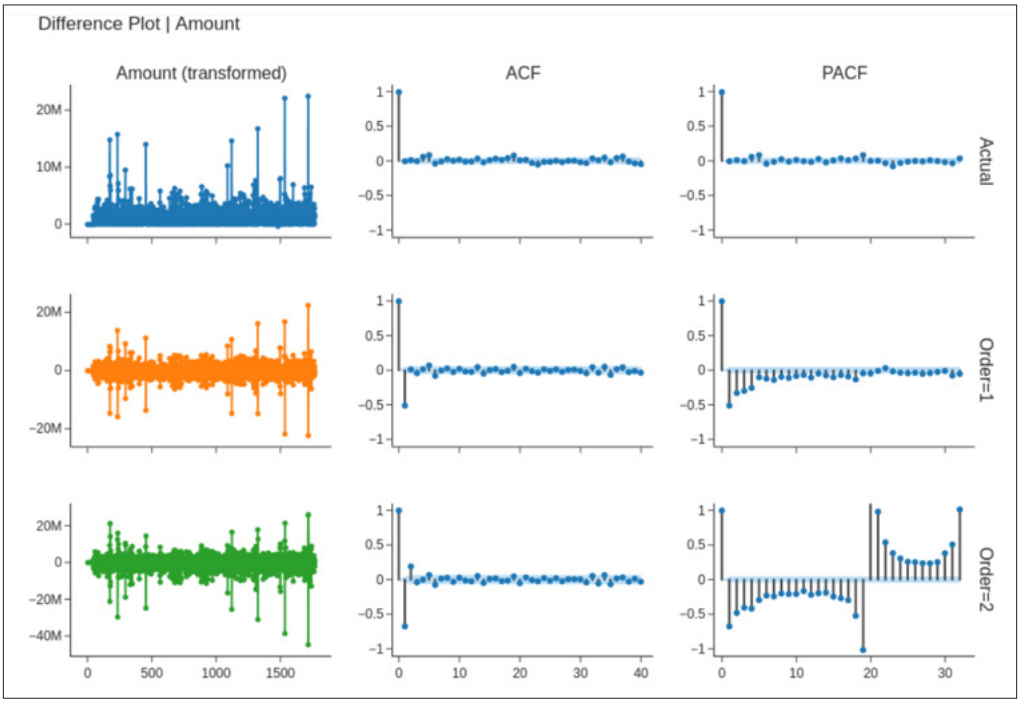

Figure 5: Plot_model – PyCaret

finalize_model

To leverage the full predictive power of a trained time series forecasting model, the finalize_model method consolidates its learning by retraining it on the entire dataset. By incorporating the entire data corpus into the model training process, it ensures that the model is optimally calibrated to capture underlying patterns and dynamics. This finalization step enhances the model's predictive accuracy and robustness, paving the way for confident deployment in real-world forecasting scenarios.

predict_model

Upon finalization, the predict_model method facilitates the generation of predictions using the trained time series forecasting model on new, unseen data. By leveraging the learned patterns and relationships encoded within the model, it generates predictions for future time points, accompanied by confidence intervals where applicable. These predictions serve as valuable insights into future trends and facilitate informed decision-making in a variety of domains.

Comparison with Traditional Python Packages

Traditional approaches to time series analysis in Python often relied on a combination of libraries such as Pandas, NumPy, and Statsmodels. While these libraries are powerful and versatile, they typically require users to write custom code for each step of the analysis process. This manual approach can be time-consuming and error-prone, particularly for users who lack a strong background in programming and statistics.

Pandas and NumPy provide foundational tools for data manipulation and numerical computation, allowing users to preprocess and analyze time series data. However, performing advanced analytical tasks such as model selection, hyperparameter tuning, and evaluation often requires additional libraries or custom implementations. Statsmodels offers a wide range of statistical models and tests for time series analysis, but using these models effectively requires a deep understanding of statistical concepts and methodologies.

In contrast, PyCaret revolutionizes the time series analysis workflow by abstracting away the complexity of model building and evaluation. It provides a higher-level interface that automates many of the tedious tasks involved in time series analysis, allowing users to focus on the problem at hand rather than the implementation details. Here's how PyCaret simplifies the process compared to traditional Python packages:

• **Automation:** PyCaret automates various aspects of time series analysis, including data preprocessing, feature engineering, model selection, hyperparameter tuning, and model evaluation. This automation streamlines the analytical workflow and reduces the need for manual intervention, enabling users to achieve results more efficiently.

• **Higher-Level Interface:** PyCaret offers a unified and intuitive interface for performing time series analysis tasks. Instead of writing custom code for each step, users can leverage PyCaret's pre-built functions and methods to accomplish common analytical tasks with minimal effort. This higher-level interface abstracts away the complexity of underlying implementations, making it accessible to users with diverse backgrounds and skill levels.

• **Model Selection and Evaluation:** PyCaret provides a wide range of algorithms for time series forecasting, along with automated model selection and hyperparameter tuning capabilities. Users can compare the performance of different models using cross-validation and select the best-performing model for their dataset. PyCaret also offers tools for evaluating model performance and interpreting model predictions, helping users gain insights into the underlying patterns in their data.

• **Ease of Use:** PyCaret's low-code approach makes it easy for users to perform complex time series analysis tasks without writing extensive code. By abstracting away the implementation details, PyCaret allows users to focus on the problem at hand rather than getting bogged down in the intricacies of programming and statistics.

Conclusion

PyCaret offers a low-code solution for time series analysis, making it accessible to a wider audience of practitioners. By automating many of the tedious tasks involved in model building and evaluation, PyCaret streamlines the time series analysis workflow, allowing users to focus on solving real-world problems rather than wrestling with code. As machine learning continues to become increasingly important across various domains, tools like PyCaret play a crucial role in democratizing access to advanced analytical techniques [1-6].

References

1. Low-Code Machine Learning. PyCaret Documentation: https://pycaret.org/.
2. Time Series Analysis in Python with Statsmodels. Statsmodels 0.14.1 https://www.statsmodels.org/stable/tsa.html.
3. Malik N, Agarwal B (2022) Time Series Nowcasting of India's GDP with Machine Learning. 2022 International Conference on Artificial Intelligence of Things (ICAIoT), Istanbul, Turkey 1-6.
4. Annamalai R, Deena S, Venkatakrishnan Ramesh, Harrieni Shankar,Y Sree Harshitha, et al. (2023) Automating Machine Learning Model Development: An OperationalML Approach with PyCARET and Streamlit. 2023 Innovations in Power and Advanced Computing Technologies (i-PACT), Kuala Lumpur, Malaysia 1-6.
5. Sarangpure N, Dhamde V, Roge A, Doye J, Patle S (2023) Automating the Machine Learning Process using PyCaret and Streamlit. 2023 2nd International Conference for Innovation in Technology (INOCON), Bangalore, India 1-5.
6. Sihombing DJC, Dexius JU, Manurung J, Aritonang M, Adinata HS (2022) Design and Analysis of Automated Machine Learning (AutoML) in PowerBI Application Using PyCaret. 2022 International Conference of Science and Information Technology in Smart Administration (ICSINTESA), Denpasar, Bali, Indonesia 89-94.