iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82639

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Autonomous Feature Engineering Agent for Time-Series Tabular Data

Sakshi Bari[1], Rashmi Borse[2], Sarthak Patil[3], Meghana Patil[4], Manisha Patil[5]
[1, 2, 3, 4, 5]*Students, Department of Data Science, R. C. Patel Institute of Technology, Shirpur, Maharashtra, India*

Abstract: ***Existing time-series datasets from most industries contain complex sequential patterns that are difficult to analyze with conventional machine learning methods. These methods are heavily reliant on manual feature engineering. This approach is time-consuming, domain-critical, resource-intensive, and is at risk of temporal data leakage. This study introduces AutoFeat Agent, a system for automated feature engineering and classification for time-series forecasting using the tsfresh library. This system provides a complete feature extraction pipeline, including the ingestion of data, preprocessing, rolling-window feature extraction, statistical model training, evaluation, and the visualization of results. The integrated framework combines automated statistical feature extraction and engineered features to augment the prediction capability, while still maintaining temporal aspects of the data. A rolling-window approach prevents leakage in the system, which traditional feature engineering is prone to. Features are screened using the False Discovery Rate (FDR) to retain features of significance. Br os, 2nd ed. The system gives a variety of machine learning models to choose from, such as Grad Boost, RF, and Log Classification. Testing shows the system successfully reduced approximately 4698 extracted features to nearly 460, which still retain their classification capabilities and accuracy (75% to 88%) evaluating predictive models formed with financial time-series datasets. The proposed system brings a flexible, scalable, and intelligent solution to the realm of automated time series analysis using machine learning.***
Keywords: ***Time-Series Analysis, Automated Feature Engineering, tsfresh, Machine Learning, Predictive Analytics, Gradient Boosting, Feature Selection, Data Leakage Prevention.***

I. INTRODUCTION

The growth of modern sensor-based computational systems like fintech, industrial automation, healthcare analytics, and climate monitoring, means that time-series datasets is even more prevalent. A dataset contains a multitude of statistical patterns that need specific analytical techniques for effective parsing. Traditional machine learning models are not able to utilize the sequential ordering of data points and temporal dependencies. Thus, feature engineering has become a key component of the data science workflow. Traditional feature engineering is highly reliant upon the analyst's experience and resulted in the development of temporal and statistical models. However, manually engineering features presents many trade-offs such as extensive development hours, subpar performance, and inflexible feature quality across various models.

Temporal leakage and lookahead bias pose significant challenges for learning systems that operate on a time-series basis. During the training phase, future data may become unintentionally accessible to the system due to suboptimal feature construction, resulting in misleadingly high accuracy and poor performance during application. The present study proposes AutoFeat Agent, a time-series domain automating feature engineering and classification framework, in order to tackle such challenges. The proposed system houses feature extraction, domain-based feature construction, feature selection, machine learning classification, and a unified leakage-free system.

The framework has a wide application range, including forecasting in stock markets, crypto analysis, anomaly detection in IoT, predicting climate changes, and predictive maintenance in industry. Its modular design facilitates seamless integration, scalability, and extensibility with contemporary deep learning structures and distributed analytics systems.

A. Research Objectives

The major objectives of the proposed work are as follows:

1) To design an automated framework for time-series feature engineering and predictive analytics.
2) To implement a leakage-free rolling-window feature extraction mechanism.
3) To integrate automated statistical extraction with domain-specific handcrafted features.
4) To reduce feature dimensionality using FDR-controlled statistical hypothesis testing.

5) To evaluate multiple machine learning classifiers on extracted temporal features.
6) To develop an interactive visualization dashboard for experimentation and analysis.

II. LITERATURE SURVEY

Machine learning and data analysis are really important. People like Christ and others have been working on this. They made something called the framework. This framework is used to get lots of information from time-series data. It can get hundreds of things like how something happens and how strong it is. The tsfresh framework also helps pick the useful information by using special tests to see if something is important. The tsfresh framework is a help, for machine learning and data analysis.

The people who made scikit-learn, like Pedregosa et al. came up with this framework that has good versions of supervised and unsupervised machine learning algorithms. They have things like Random Forest and Gradient Boosting that work well for predicting things when you have a lot of features to look at. Several studies have said that it is very important to prevent leakage in systems that predict what will happen in the future. If you do not prepare your data correctly or if you make mistakes when you are creating your features you might accidentally give your models information about the future. This can make your models not work well in the real world. People are doing research now on how to make sure this does not happen they are looking at ways to compute things in a time window without leaking any information. They are working on strategies to validate their models over time so they can make good predictions. Recent research is focused on scikit-learn and machine learning algorithms, like Random Forest and Gradient Boosting to make predictive models. The systems we have now can automatically find features but they do not really understand the area they are working in. They also do not have ways to show us what they are doing and they do not always make sure that cause comes before effect. The AutoFeat Agent framework is a way of doing things that solves these problems. It combines feature finding, knowledge of the specific area and ways to prevent mistakes all, in one system. The AutoFeat Agent framework does this by bringing automatic extraction, domain-specific engineering and leakage prevention, which makes it a complete system.

Table 1: Existing System Performance Comparison

Existing System / Method	Accuracy	ROC-AUC
Traditional Statistical Models	60–70%	0.65–0.72
Basic Machine Learning Pipeline	70–78%	0.75–0.80
tsfresh + Classical ML Models	78–82%	0.80–0.84
Random Forest-Based Systems	82–85%	High
Conventional Gradient Boosting Systems	83–86%	0.84+

III. SYSTEM DESIGN AND METHODOLOGY

A. *Overall System Architecture*

The proposed AutoFeat Agent framework follows a modular multi-layer architecture designed specifically for automated time-series analytics and predictive modeling.

The architecture consists of four major layers:

1) Data Acquisition and Preprocessing Layer
2) Automated Feature Engineering Layer
3) Machine Learning and Evaluation Layer
4) Visualization and Interaction Layer

The complete workflow of the framework is illustrated through interconnected processing stages that transform raw sequential data into predictive insights.

Table 2: Core System Modules

Module	Description
csv loader.py –	Handles CSV dataset ingestion, timestamp parsing, preprocessing, normalization, and missing value handling.
tsfresh extractor.py –	Performs automated statistical feature extraction and domain-specific feature engineering.
run pipeline.py –	Executes the complete machine learning workflow including feature selection, model training, and evaluation.
streamlit app.py –	Provides an interactive web-based dashboard for the visualization and experimentation.

B. Data Acquisition and Preprocessing

The preprocessing layer changes data into a structured format that works well with time-series and machine learning. Time-series data often has some problems like missing information or wrong timing. The preprocessing step is very important to make the data better and more accurate for machine learning analysis of time-series data. It helps to deal with issues like missing timestamps and noisy observations, in time-series data. So preprocessing of time-series data is a step to get good results from machine learning analysis of time-series data.

The framework supports multiple application domains including:

1) Financial Market Forecasting
2) Cryptocurrency Trend Prediction
3) IoT Sensor Analytics
4) Climate Monitoring Systems
5) Industrial Predictive Maintenance

The first step is to get everything in order. This is where we sort things by time change the time format pick the numbers fill in the missing information and make sure everything is aligned. We use a set of tools based on pandas to do all of this. The preprocessing stage is really important, for the data it does the sorting it does the timestamp conversion it does the numerical feature selection it does the missing value imputation and it does the temporal alignment.

C. Automated Feature Engineering

The feature engineering layer represents the core intelligence of the proposed system. Instead of manually constructing features, the framework automatically extracts a large number of statistical and temporal descriptors using tsfresh.

For each observation indexed at i, features are generated using historical values from the interval:

$$[i - window\ size,\ i - 1]$$

This method of doing calculations over a moving time period makes sure that we only use information from the past and do not look at what's going to happen next.

The system that pulls out information calculates types of features like numbers that show how things change over time results from special math operations measures of how much things are mixed up how things are related to each other what is happening over time how energy is spread out and what is not normal.

The automatic part of the system that pulls out information creates 4698 pieces of information, from just a few basic pieces of data. To avoid repeating the information over and over and to make the system work faster the system uses a special method called False Discovery Rate to test ideas and it uses a specific way of doing this called the Benjamini–Hochberg procedure:

$$p_{adjusted} \leq 0.05$$

After we pick the features we keep around 460 features that really matter for making predictions. The framework also uses features that people with experience in the field have created, such as things that show how prices are moving how much prices are changing what is different, about certain times of the year how unusual something is and how signals are changing over time.

D. Machine Learning and Evaluation

The machine learning layer is responsible for classification and predictive analysis. Extracted features are normalized using standard scaling techniques before model training.

The framework supports the following classifiers:

1) Gradient Boosting Classifier
2) Random Forest Classifier
3) Logistic Regression Classifier

When we look at these models Gradient Boosting does a great job. This is because Gradient Boosting is very good at understanding relationships, between lots of different features.

The things we used to set up the Gradient Boosting model are:

$$n_{estimators} = 200,\ learning_{rate} = 0.1,\ max_{depth} = 6$$

The dataset is divided using an 80:20 train-test split strategy with stratified sampling.

The system evaluates model performance using Accuracy, Precision, Recall, F1-Score, and ROC-AUC metrics.

E. Visualization and Dashboard Layer

An interactive visualization interface is developed using Streamlit to simplify experimentation and improve interpretability.
The dashboard provides:

1) Dataset selection
2) Model selection
3) Performance visualization
4) Feature importance analysis
5) Comparative evaluation graphs
6) Real-time pipeline execution

The visualization layer improves usability for researchers, students, and practitioners working with time-series predictive analytics systems.

IV. RESULTS AND ANALYSIS

A. Feature Extraction Results

Table 3: Feature Extraction and Selection Results

Stage	Feature Count
Raw Input Features	6
Extracted Features	4698
Selected Features After FDR	460

The proposed system achieved nearly 90% dimensionality reduction while preserving predictive performance.

B. Classification Performance

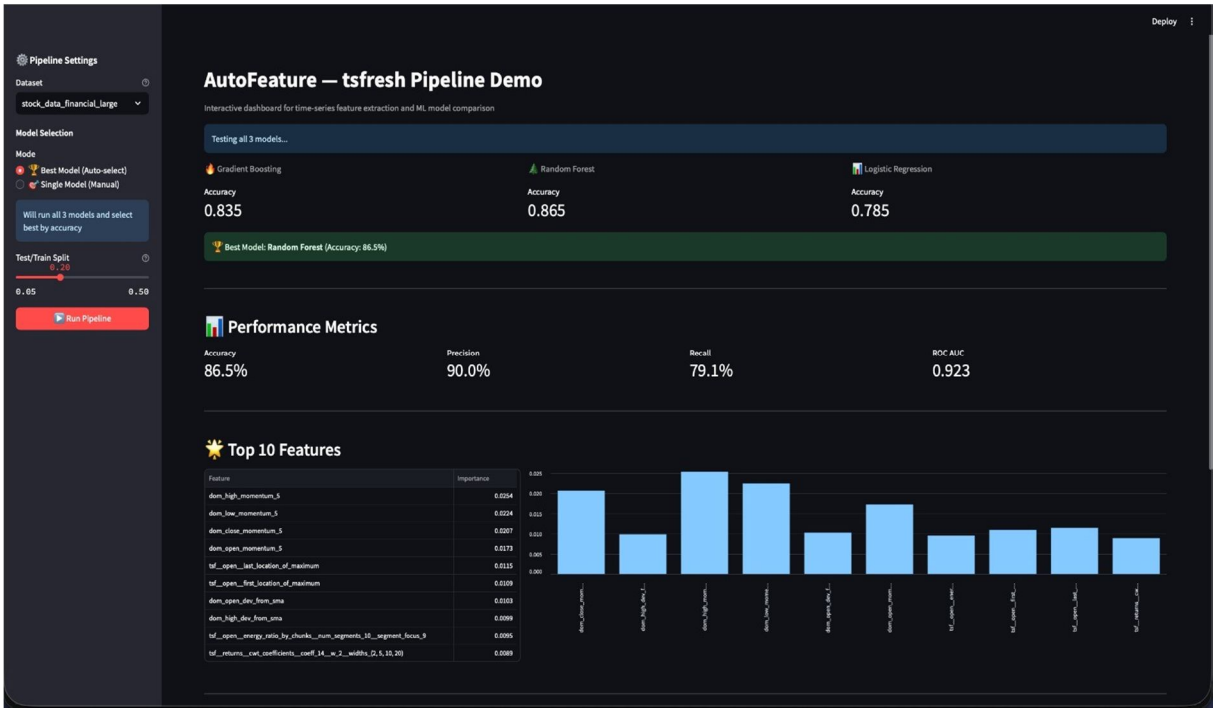


Figure 1: Classification performance and feature engineering results generated by AutoFeat Agent.

Table 4: Model Performance Comparison

Model	Accuracy	ROC-AUC
Gradient Boosting	75–88%	0.85+
Random Forest	86%	High
Logistic Regression	80%	Moderate

Gradient Boosting achieved the best overall classification performance across multiple synthetic stock datasets.

V. DISCUSSION

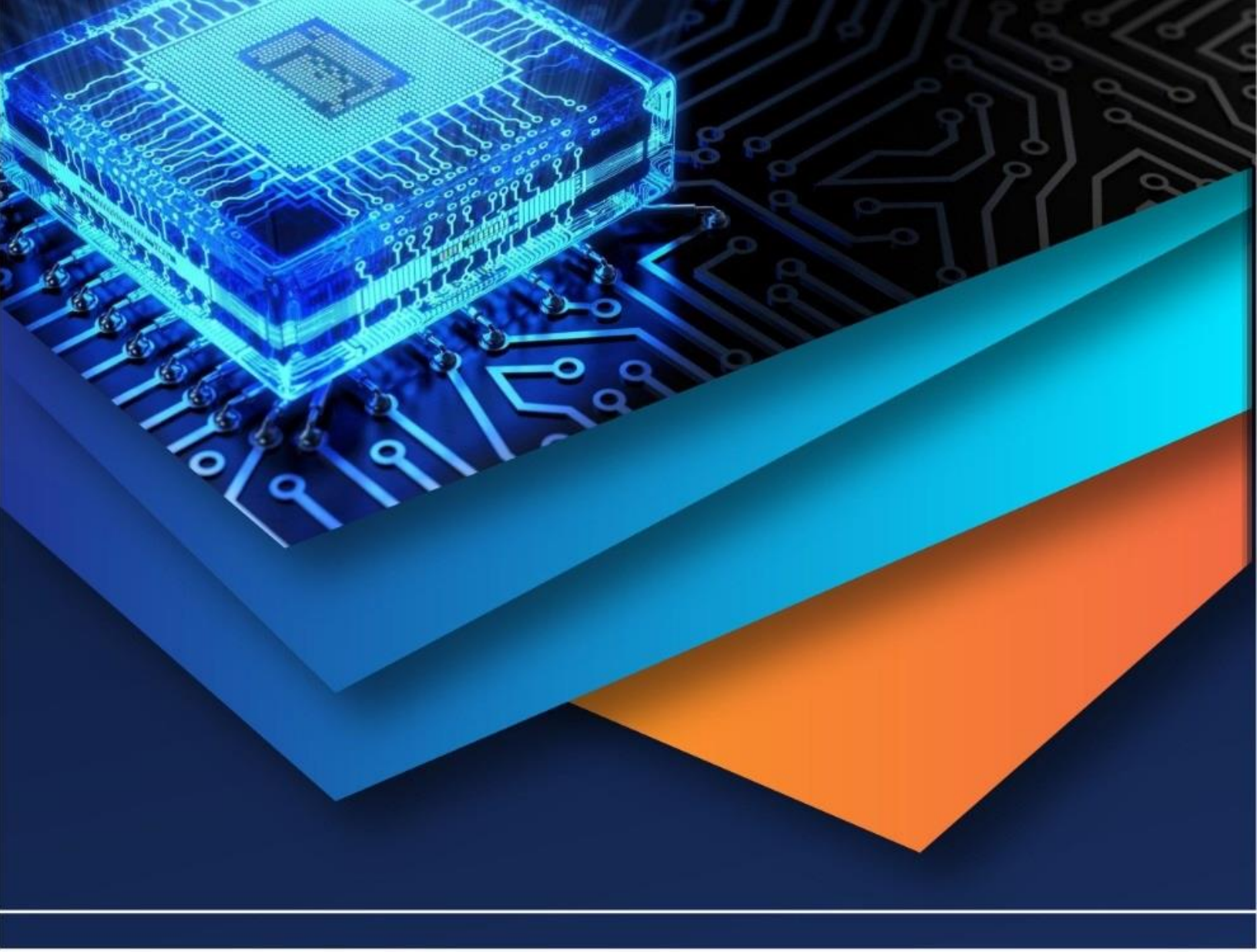
The AutoFeat Agent framework does a job of making the process of working with time series data a lot easier. It takes care of the part of finding the right features in the data without messing up the timeline. The AutoFeat Agent framework combines automatically finding features with features that people with experience, in the field have carefully chosen. This really helps when it comes to classifying data and making sure the model works with different types of data. The way the AutoFeat Agent framework is set up makes it easy to add data, new ways of finding features and new deep learning models in the future. This is really helpful because the AutoFeat Agent framework can be used in different situations.

VI. CONCLUSION AND FUTURE WORK

This paper is about the AutoFeat Agent. It is a system that helps with time-series analytics. The AutoFeat Agent can automatically look at the data. Pick out the important parts. It also uses methods to make sure the data is good and useful. The system can even show the results in a way that's easy to understand. The people who made the AutoFeat Agent tried it out on some data. They found out that it works well and can make the data smaller and easier to use. The AutoFeat Agent is an flexible system that can be used for many different time-series analytics applications. In the future the people who made the AutoFeat Agent want to make it even better. They want to add some techniques like Transformer-based architectures and real-time streaming analytics. They also want to make it possible to use the system on datasets and to understand how it makes its predictions.

REFERENCES

[1] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.
[2] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[3] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
[4] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[5] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.
[6] Streamlit Inc., "Streamlit Documentation," Available: https://streamlit.io/
[7] Python Software Foundation, "Python Language Reference," Available: https://www. python.org/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY