

Monthly Expense Tracking and Forecasting Using Regression Models and Interactive Visualization

Dr. R. Prasad rao¹, A. V. S. Deepak², V. Reshma³, V. Jyoshnavi⁴, M. Pavan Ganesh⁵

¹Professor: Department of Electronics and Communication Engineering Avanthi Institute of Engineering & Technology, Tamaram, Makavarapalem, Anakapalle – 531113, Andhra Pradesh, India

²Assistant Professor: Department of Electronics and Communication Engineering Avanthi Institute of Engineering & Technology, Tamaram, Makavarapalem, Anakapalle – 531113, Andhra Pradesh, India

³⁻⁵UG Students: Department of Electronics and Communication Engineering Avanthi Institute of Engineering & Technology, Tamaram, Makavarapalem, Anakapalle – 531113, Andhra Pradesh, India

Abstract

Managing personal finances has become harder as living costs rise and spending spreads across more categories. Most people track money in a notebook, a spreadsheet or a basic budgeting app, all of which record what has already been spent but say nothing about what is coming. This paper describes a monthly expense tracking, analysis and visualisation system that adds forecasting to the usual record-keeping. The system is built in Python with Streamlit for the interface, Pandas for the data handling, Matplotlib for the charts and Scikit-learn for the models. A user enters monthly income and expenses in five categories over a twelve-month window, and the system computes the monthly total, the savings and the category-wise breakdown, and plots income against expenditure over time. For forecasting, three regressors are fitted to each series independently: linear regression, a random forest and support vector regression. Each is scored on a held-out tail of the series by mean absolute error, and the best-scoring model is used for the forecast of the following month. On the twelve-month test series, linear regression produced a mean absolute error of Rs. 2321 on income against Rs. 5095 for the random forest and Rs. 7784 for support vector regression, and was selected automatically. The system then reports the predicted income, the predicted total expense, the expected savings and a target income derived from the predicted expense, so the user is given a figure to plan against rather than only a record of the past.

Index Terms— Expense tracking, personal finance, financial forecasting, linear regression, random forest, support vector regression, mean absolute error, model selection, Streamlit, data visualisation.

I. Introduction

Personal financial management has grown more complicated as the cost of living has risen and as spending has spread across more and more distinct categories. Keeping track of what comes in, where it goes and what is left over takes more effort than it once did, and the tools most people use have not kept pace.

The common methods are all descriptive. A notebook records what was spent. A spreadsheet records the same and adds the arithmetic. A budgeting application does the same on a phone with a nicer interface. Each tells the user what has already happened. None of them says what next month is likely to look like, which is the question a person actually needs answered when deciding whether a commitment is affordable. Financial planning done this way is reactive: the user discovers a problem after the money has gone.

The system described here keeps the tracking function but adds a predictive one. A user records monthly income and category-wise expenses across a twelve-month window. The system computes totals and savings, shows the income and expenditure trends as charts, and then fits three different regression

models to the recorded series in order to forecast the following month. Rather than committing to one model in advance, it scores all three on held-out data and uses whichever performs best on that particular user's history. From the forecast it derives the expected savings and a target income figure, which turns the output from a description into something the user can plan against.

The system is built from standard open-source components: Python for the language, Streamlit for the web interface, Pandas for the data structures, Matplotlib for the charts and Scikit-learn for the models. The remainder of this paper reviews the relevant background, analyses the requirements, describes the design and implementation, presents the results obtained on a twelve-month test series, and sets out the limitations and the directions for further work.

II. Literature Survey

A. Traditional Expense Tracking

Financial record keeping began with manual ledgers. Writing down daily expenses and monthly income maintains awareness but demands continuous effort and offers no automation. Spreadsheet tools improved

on this by supporting formulas and simple charts, but they still depend on the user's own skill in building and maintaining the model, and they provide no predictive capability. Mobile budgeting applications made digital recording convenient but kept the same descriptive focus on historical data.

The limitations common to all three are consistent: no predictive analysis, no automated decision support, limited category-wise insight, no machine-learning component, and manual interpretation of whatever the tool displays. These are what motivated the present work.

B. Regression Models in Financial Forecasting

Linear regression, described in the standard treatment by Montgomery, Peck and Vining, fits a linear relationship between a predictor and a response and is widely used for trend-based financial forecasting. It is simple, interpretable and computationally cheap, and it performs well when the underlying series carries a clear trend. Its weakness is that it cannot represent non-linear structure.

The random forest, introduced by Breiman, is an ensemble method that builds many decision trees on resampled data and averages their predictions. It handles non-linear relationships well and resists overfitting through the averaging, at a higher computational cost than a linear model. In financial prediction it is valued for its ability to capture structure a linear model would miss.

Support vector regression extends the support vector machine of Cortes and Vapnik to regression. It uses a kernel to model non-linear relationships and is effective on small datasets and robust to outliers, but it is sensitive to the choice of its parameters and is computationally demanding.

The literature consistently reports that no one of these dominates across problems, and that fitting several and selecting on validated performance is more reliable than committing to one in advance. That finding is the basis for the model-selection strategy adopted here.

C. Tools

Python is the standard language for this class of work because of the breadth of its data and modelling libraries. Streamlit allows a data application with interactive controls and embedded charts to be written without separate front-end code. Pandas supplies the DataFrame used to hold the monthly series and the operations used to aggregate it. NumPy provides the numerical array handling underneath. Matplotlib renders the trend and distribution charts. Scikit-learn supplies the three regressors, the train-test split and the error metric, all behind a single consistent interface, which is what makes fitting and comparing three different models straightforward.

III. System Analysis

A. Existing System

Existing expense trackers, whether notebooks, spreadsheets or applications, let the user record income and expenses, compute totals and view a summary. They do not forecast, they do not compare models, they do not detect category-wise trends automatically, and they do not suggest a savings target. Their reporting is static and oriented entirely towards the past, so the user carries the whole burden of interpretation.

B. Proposed System

The proposed system manages twelve months of financial data entered through the interface, tracks expenses across five categories, computes total expense and savings for each month, charts income against expenditure over the window, fits three regression models to each series, compares them on held-out data, selects the best automatically, forecasts the next month for income and for each expense category, and reports the resulting expected savings together with a target income figure.

C. Requirements

Functionally the system must accept monthly income and category-wise expense entries, reject negative values, compute per-month totals and savings, display the full twelve-month table, plot the income and expenditure trends, fit and evaluate the three models, select the best, produce the forecast and the derived figures, and warn the user when too little data has been entered for a forecast to mean anything.

Non-functionally the interface must be usable without training, predictions must be produced within a few seconds of the request, charts must render correctly for any valid dataset, and the system must run on ordinary hardware. The requirements are summarised in Table I.

Table I. System Requirements

Item	Requirement
Processor	Intel i3 or above
RAM	4 GB minimum
Storage	10 GB free
Operating system	Windows, Linux or macOS
Language	Python 3
Interface	Streamlit
Data handling	Pandas, NumPy
Charts	Matplotlib
Models	Scikit-learn

IV. System Design

A. Data Flow

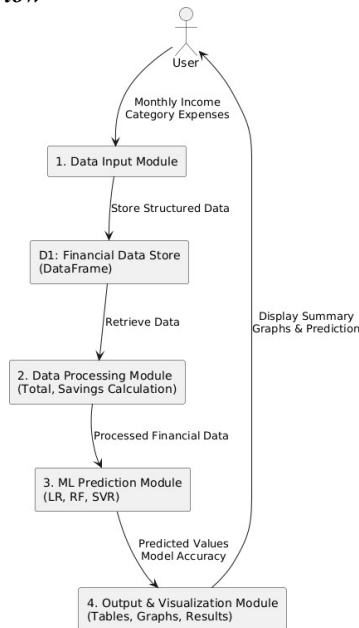


Fig. 1. Data flow diagram of the system.

Fig. 1 shows the movement of data. Income and category expenses entered in the sidebar are assembled into a DataFrame indexed by month. Derived columns for total expense and savings are computed from it. The same DataFrame feeds both the visualisation path, which produces the trend chart, and the prediction path, which fits and evaluates the models and returns the forecast.

B. Use Cases and Structure

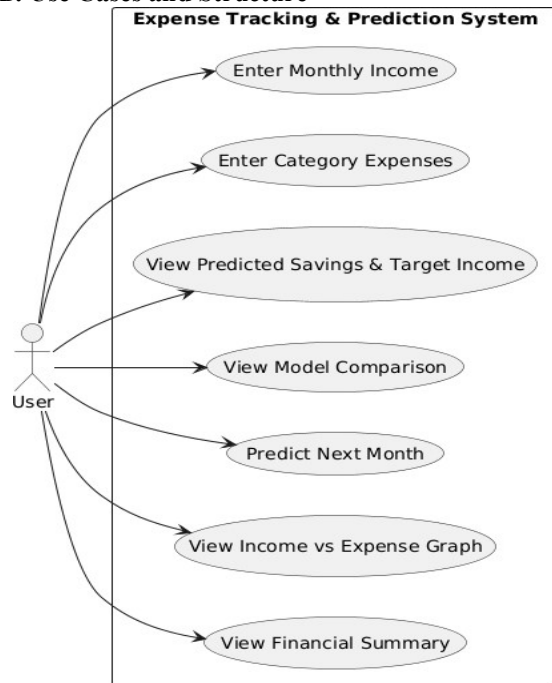


Fig. 2. Use case diagram.

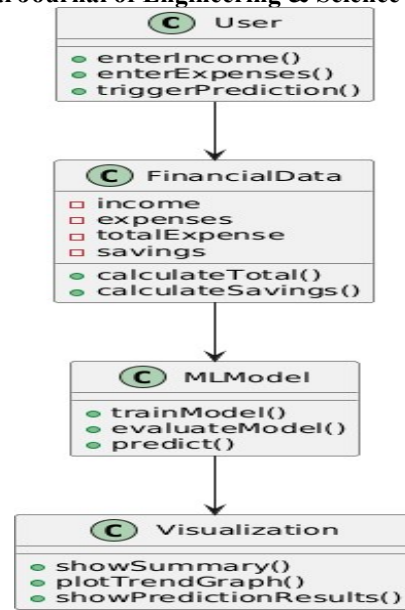


Fig. 3. Class diagram.

The user is the single actor. The use cases are entering monthly data, viewing the financial summary, viewing the trend chart, requesting a prediction, and viewing the model comparison. The class structure separates the data layer that holds the monthly series, the analysis layer that computes totals and savings, the prediction layer that wraps the three models and the evaluation, and the presentation layer that renders tables and charts.

C. Interaction and Workflow

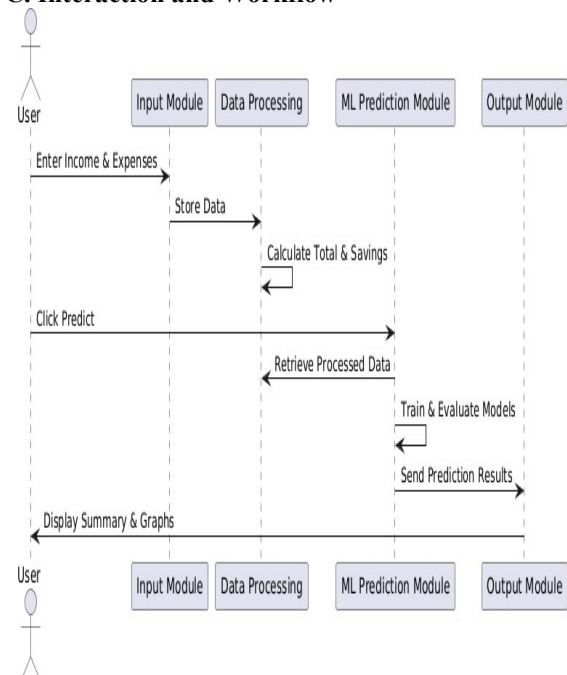


Fig. 4. Sequence diagram for a prediction request.

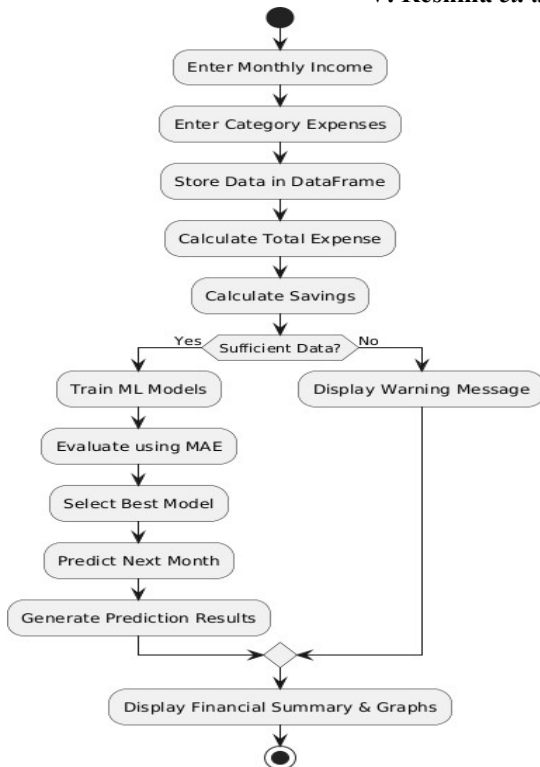


Fig. 5. Activity diagram of the overall workflow.

The sequence diagram traces a prediction request from the button press through the data-sufficiency check, the fitting and scoring of the three models, the selection of the best and the rendering of the result. The activity diagram places that within the whole session, from data entry through summary and visualisation to prediction and the derived target income.

V. Implementation

A. Data Entry and Summary

The application generates a sequence of twelve month labels from a start date and presents, for each month, a numeric input for income and one for each of the five expense categories: travel, food, household, shopping and other. Every input is constrained to a non-negative minimum, which is where the rejection of negative values is enforced. The entered values are assembled into a Pandas DataFrame indexed by month, with one column per category plus an income column. Two derived columns are then computed: the total expense as the row-wise sum of the five category columns, and the savings as income minus total expense. The complete table is displayed to the user.

B. Visualisation

A line chart plots income and total expense against the month labels on common axes, so the gap between the two curves is the monthly saving and its widening or narrowing over the year is directly visible. After a

prediction has been made, a bar chart shows the forecast expense broken down by category.

C. Prediction

Each series is treated as a univariate regression against the month index. The predictor is the integer position of the month in the series and the response is the recorded value, so the model learns the shape of the trend over the twelve points and extrapolates one step beyond the end.

Three models are fitted to every series: a linear regression, a random forest regressor with two hundred trees and a fixed random seed, and a support vector regressor with a radial basis kernel. Each is first evaluated by splitting the series into a training part and a held-out tail without shuffling, which preserves the time order and ensures the model is tested on points later than those it was trained on. The model is fitted on the training part, predictions are made for the held-out part, and the mean absolute error between prediction and actual is computed.

From that error a percentage figure is derived as one minus the ratio of the mean absolute error to the mean of the held-out values, expressed as a percentage. This is a normalised error measure rather than classification accuracy, and it is reported here under the label the application uses while noting what it actually represents: a value of 96 per cent means the average absolute error was about four per cent of the average held-out value.

Each model is then refitted on the full series and used to predict the next step. The three candidates are compared on the derived percentage figure and the best is selected. The procedure is applied independently to income and to each of the five expense categories, so different categories may end up forecast by different models, which is appropriate because they need not share the same shape of trend.

Before any of this runs, the system checks that enough non-zero values have been entered, and issues a warning instead of a forecast if the data is too sparse for the exercise to be meaningful.

D. Derived Figures

From the forecasts the system computes the predicted total expense as the sum of the five predicted categories, the expected savings as predicted income minus predicted total expense, and a target income figure obtained by scaling the predicted total expense upward by ten per cent.

VI. Results

The system was exercised on a twelve-month series running from February 2025 to January 2026, with the forecast made for February 2026.

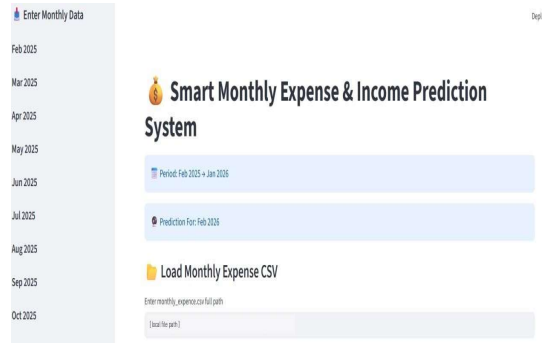


Fig. 6. Application home view showing the period and the data source.

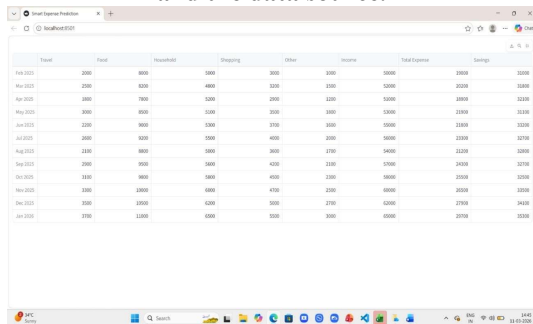


Fig. 7. Financial summary table for the twelve-month window.

Fig. 7 shows the summary table. Each row is a month, the five category columns carry the recorded expenses, and the income, total expense and savings columns are shown alongside. Over the window the income rises from Rs. 50,000 to Rs. 65,000 and the total expense from Rs. 19,000 to Rs. 29,700, so savings also rise, from Rs. 31,000 to Rs. 35,300.



Fig. 8. Income against total expenditure over the twelve-month window.

The trend chart of Fig. 8 makes the relationship between the two series immediately readable. Both rise over the year, income more steeply than expenditure, and the vertical gap between the curves at any month is that month's saving. The August dip in both series is visible at a glance in a way it is not in the table.

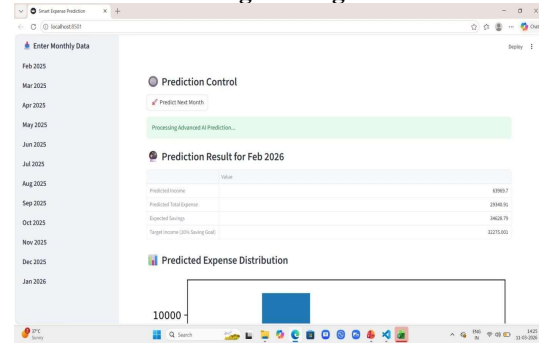


Fig. 9. Prediction result for the following month.

Fig. 9 shows the forecast. For February 2026 the system predicts an income of Rs. 63,970 and a total expense of Rs. 29,341, giving expected savings of Rs. 34,629, and reports a target income of Rs. 32,275.

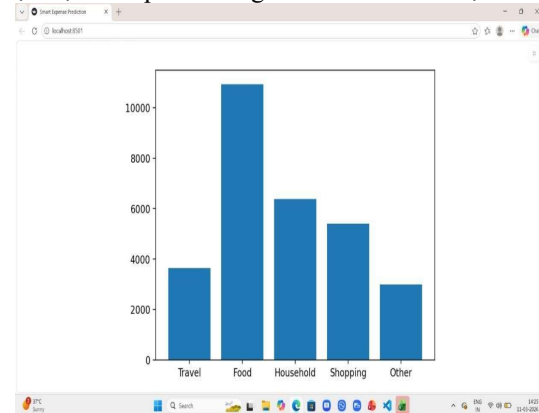


Fig. 10. Predicted expense distribution by category.

The category breakdown in Fig. 10 shows where the predicted expenditure falls. Food dominates at close to Rs. 11,000, followed by household at about Rs. 6,400 and shopping at about Rs. 5,400, with travel and other each near Rs. 3,000. This is the view that tells a user which category to act on if the total needs to come down.

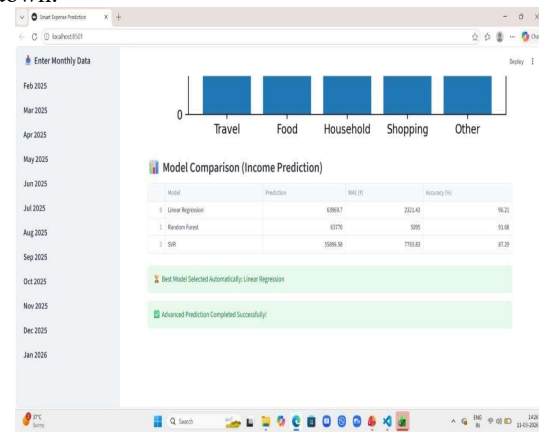


Fig. 11. Comparison of the three models on the income series.

Fig. 11 shows the model comparison for the income series, and the figures are repeated in Table II.

Table II. Model Comparison On The Income Series

Model	Prediction (Rs.)	MAE (Rs.)	Score (%)
Linear regression	63,969.70	2,321.43	96.21
Random forest	63,770.00	5,095.00	91.68
Support vector regression	55,896.58	7,783.83	87.29

Linear regression gave the lowest mean absolute error at Rs. 2,321, against Rs. 5,095 for the random forest and Rs. 7,784 for support vector regression, and was selected automatically. The ordering is what the shape of the data would lead one to expect. The income series is close to a straight line, which is exactly the structure linear regression represents; the random forest, which predicts by averaging values it has already seen, cannot extrapolate above the range of its training data and therefore undershoots a rising series; and support vector regression with an untuned radial basis kernel reverts towards the mean away from the training points, which is why its prediction of Rs. 55,897 falls so far below the others.

A. Testing

Ten functional test cases were run, covering income entry, category expense entry, rejection of negative values, total expense computation, savings computation, the warning raised on insufficient data, prediction on a complete dataset, display of the model comparison, chart rendering, and the target income calculation. All ten passed.

B. Limitations

Three limitations should be stated plainly, because they bound what the results above establish.

The evaluation rests on a single twelve-month series with a strong, nearly linear trend. With a split that holds out the tail of such a short series, each model is fitted on roughly eight points and tested on four. A comparison at that scale indicates which model suits this particular series; it does not establish a general ranking, and on a series with seasonality or an abrupt change the ordering could easily reverse. This is in fact an argument for the design adopted, since selecting per series is what allows the right model to win in each case, but it means the reported figures should not be read as a general result about the three methods.

The percentage figure reported as accuracy is one minus the normalised mean absolute error, not accuracy in the classification sense. It is a reasonable relative measure for choosing between models on the

same series, but it is not comparable across series with different means and it should not be quoted as a success rate.

The target income figure is computed as the predicted expense multiplied by 1.10. That produces an income exceeding expenditure by ten per cent of expenditure, under which savings are about 9.1 per cent of income rather than the intended 10 per cent. Achieving a saving of ten per cent of income requires dividing the predicted expense by 0.90 instead, which for the figures above gives a target income of about Rs. 32,601 rather than Rs. 32,275. The difference is small at this scale but it is systematic, and the formula should be corrected.

VII. Conclusion

A monthly expense tracking, analysis and visualisation system has been built that goes beyond recording past spending to forecasting the month ahead. The system records income and category-wise expenses over a twelve-month window, computes totals and savings, charts income against expenditure, and forecasts the following month by fitting three regression models to each series, scoring them on held-out data and selecting the best automatically.

On the twelve-month test series linear regression was selected for the income forecast with a mean absolute error of Rs. 2,321, against Rs. 5,095 for the random forest and Rs. 7,784 for support vector regression. The system reported a predicted income of Rs. 63,970 and a predicted expense of Rs. 29,341 for the following month, with the expense broken down by category so the user can see which heading to act on.

The per-series model selection is the element worth carrying forward: committing to one model in advance would have produced a materially worse forecast on this data, and the selection costs very little when the series are this short. Against that, the evaluation rests on a single short series, the reported percentage is a normalised error rather than accuracy, and the target income formula understates the required income slightly. Each of these is addressable.

Further work should evaluate the system on real multi-user transaction histories spanning more than one year, so that seasonality is present and the model selection is tested against data it was not designed around; correct the target income formula; add persistent database storage and user accounts in place of the current session-scoped data; and extend the forecast horizon beyond a single month with an indication of its uncertainty, since a point forecast with no interval attached invites more confidence than a twelve-point series can support.

References

- [1] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
- [2] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.
- [3] D. C. Montgomery, E. A. Peck and G. G. Vining, Introduction to Linear Regression Analysis, 5th ed. Hoboken, NJ: Wiley, 2012.
- [4] F. Pedregosa et al., "Scikit-learn: machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
- [5] W. McKinney, "Data structures for statistical computing in Python," in Proc. 9th Python in Science Conference, 2010, pp. 56–61.
- [6] J. D. Hunter, "Matplotlib: a 2D graphics environment," Computing in Science and Engineering, vol. 9, no. 3, pp. 90–95, 2007.
- [7] C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, pp. 357–362, 2020.
- [8] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 3rd ed. Melbourne: OTexts, 2021. [Online]. Available: <https://otexts.com/fpp3/>
- [9] Streamlit Inc., Streamlit Documentation. [Online]. Available: <https://docs.streamlit.io>
- [10] Python Software Foundation, The Python Language Reference. [Online]. Available: <https://docs.python.org>