Modeling Popularity in Programming Languages using ARIMA

*Modeling and Simulation Report*

*Group 5*

| Mayar Shaath | Bhavik Panchal |
| --- | --- |
| *Ontario Tech University*  *Oshawa, Ontario* | *Ontario Tech University*  *Oshawa, Ontario* |
| mayar.shaath@ontariotechu.net | bhavik.panchal@ontariotechu.net |

***Abstract* — this report focuses on the implementation and results regarding a programming languages popularity model and future prediction simulation. It goes through how the model was developed and what the results mean for the future of software development.**

I. Introduction

Computer programming is critical in today's day and age, as it drives practically every device used. Machines can be told what to do using them, and they are required to bridge the gap between machines and humans' "thinking." The majority of programming languages were inspired by or built on previous programming languages' notions. Newer programming languages make programmers' work easier, but older languages continue to serve as a firm foundation for future ones. To meet all of their data, transaction, and customer service demands, businesses rely largely on programs. And science and medicine require precise and sophisticated programs for their research. To keep up with consumer needs, mobile applications must be updated. That is why learning computer programming language trends is very important in moving forward.

*A. Brief History*

The "Algorithm for the Analytical Engine" by Ada Lovelace is widely regarded as the first computer language. It was built by Ada in 1883 to assist Charles Babbage with Bernoulli number computations. Because of its potential to be programmed to tackle problems of any complexity, Lovelace views her machine as distinct from earlier calculating machines. Her contributions to computer programming are significant because they demonstrated the capabilities of computed devices about a century before the concept of a contemporary programming computer was conceived.

After that came Assembly Language, which was first introduced in 1949 and was quickly adopted by Electronic Delay Storage Automatic Calculators. Assembly was a low-level computer language that made machine code easier to understand. Quickly followed in 1952 the appearance of the first compiled programming language, Autocode, and it could be translated into machine code using a compiler.

Over time, more and more programming languages got developed and the evolution of programming and software development gained momentum and became what it is today.

II. Problem Statement and Proposed Solution

With the advancement of technology and the growth of the programming/software development industry, there are constant changes to the popularity of different programming languages. Thus, education institutions and other training institutes face the drawback of potentially teaching languages no longer desired in the rapidly changing industry. Even hiring companies may face a similar problem when looking for the “perfect” candidate, when they are in fact searching for experience in a practically dead language.

The project aims to model the current and future trends in programming language popularity. The models will be used to predict future trends for the most popular programming languages to see if they will truly withstand the test of time and remain in their positions. They can also be used to predict the general direction of where the industry is heading as different languages are mainly used for different types of programming.

It includes models for five different programming languages: C/C++, Java, JavaScript, PHP, and Python, and it shows the change in their popularity over time. There is a special group of approaches and techniques in the domain of machine learning that are particularly well suited for forecasting the value of a dependent variable over time. For this project, this approach is the AutoRegressive Integrated Moving Average (ARIMA) Model. The model below is anl ARIMA model displaying the popularity of 29 different programming languages from 2004 to 2022.

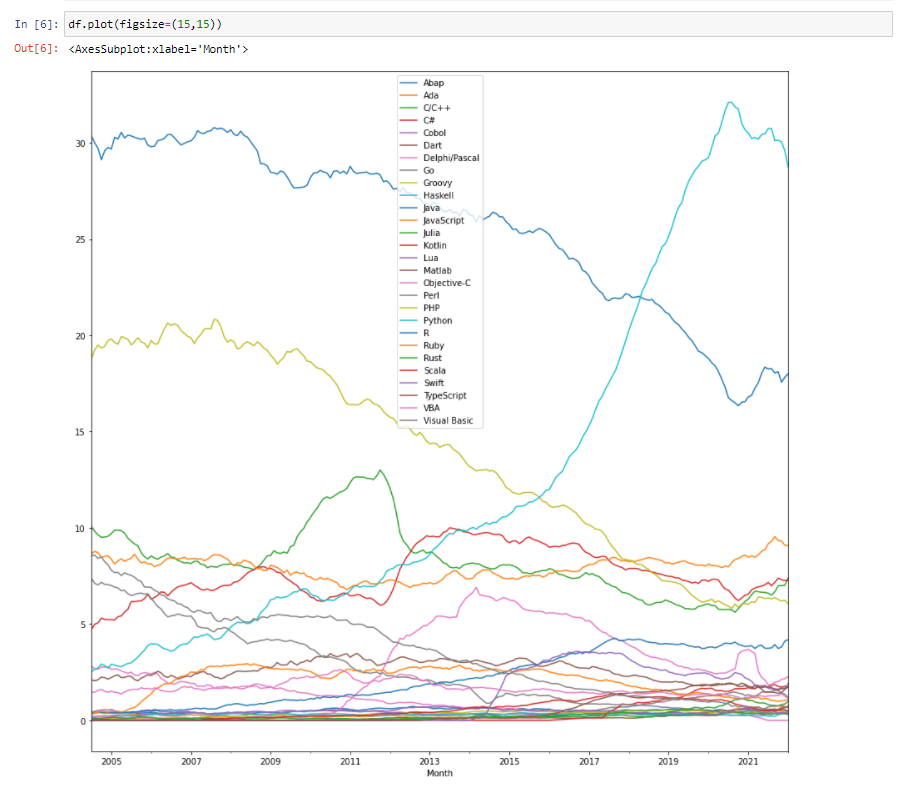


Fig. 1 Popularity of different programming languages, 2004-2022.

III. Technical Description of Solution

*A. The ARIMA Model*

A collection of quantities assembled over even time periods and ordered chronologically is referred to as *time series data*. A time series can be broken down into 3 components:

* Trend: upward and downward movement of the data over a period of time
* Seasonality: seasonal variance
* Noise: spikes at random intervals

ARIMA is a statistical analysis model that uses time series data to better understand data sets or forecast future trends. If a statistical model predicts future values based on past values, it is called autoregressive.

An ARIMA model can be understood by outlining each of its components as follows:

* Autoregression (AR): refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
* Integrated (I): represents the differencing of raw observations to allow for the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).
* Moving average (MA): incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

An ARIMA model has three component functions: AR (p), the number of lag observations or autoregressive terms in the model; I (d), the difference in the nonseasonal observations; and MA (q), the size of the moving average window. An ARIMA model order is depicted as (p,d,q) with values for the order or number of times the function occurs in running the model. Values of zero are acceptable.

The data are differenced in an autoregressive integrated moving average (ARIMA) model to make it stationary. A model that demonstrates stationarity demonstrates that the data is consistent throughout time. The goal of differencing is to remove any trends or seasonal structures found in most economic and market data.

*B. Implementation and Dataset*

In this project, the ARIMA model was used to model and predict the future popularity trends of 5 different programming languages. The dataset used as a base for the project is one outlining data about the most popular programming languages from each month during the span of 2004 to 2022. All values of each programming language are shown in percentage form out of 100%. To model each individual language, we created a new dataset for each language containing only their own statistics.

First, we imported all necessary libraries, such as numpy, pandas and matplotlib.pylab. We then imported the given languages dataset, and parsed their Months to datetime type.The data is then set, and it can be observed with the head function (displaying first statistics) or tail function (displaying last statistics). It can also be viewed on a graph with the plot function.

Next, we calculated the rolling mean, and rolling standard deviation with both being implemented with windows of size 12. This dismisses the first 11 values and takes the average of these values and assigns it to the 12th value (month). After the 12th month, each month will be assigned the value of : the average of its own value and all of its prior values. These 2 values are plotted, and then the Dickey-Fuller test was performed to determine if the data was stationary. Stationarity is shown in the Dickey-Fuller test if the Test Statistic holds a value that is similar to the values of the Critical Values and if the p-value is less than 0.05. Stationarity can be viewed in the plot if the data represents a flat line.

After this, the trend is estimated by taking the log of the dataset. This represents the same trend as the original plot but with smaller ‘y’ values. With this logged dataset, the moving average and moving standard deviation are calculated. This is followed by calculating the difference between moving average and actual value.

We then created a function that would test the stationarity and would calculate and plot the rolling statistics of the moving average and the moving standard deviation. We also included a Dickey-Fuller test at the end of the stationarity test. This test was then used to evaluate the difference between the moving average and the actual values.

The next step consisted of calculating the exponential decay weighted average to see the trend of the time series. The weighted average is then subtracted from the previously created log scale and the values are shifted by a lag of 1. Usually after this process the data will seem most stationary.

Since some stationarity is established, we proceed by viewing the components of the time series. These components are the trend, seasonality and the residuals. This just allows the user to view and evaluate the differences in the original plot vs the trend, seasonality and the residuals. After this, the stationarity of the residuals is tested with the previously created stationarity test. The noise of the residuals is then tested for stationarity right after this.

Next, we plotted the autocorrelation and the partial autocorrelation graphs. These allowed us to determine the ‘p’ and ‘q’ values for modeling the autoregressive (AR) model. The ‘p’ value is determined by evaluating the partial autocorrelation graph. When the graph reaches 0, this is the value that ‘p’ should hold. Determining the ‘q’ value is very similar to this, but it is done by examining the autocorrelation graph for when the data hits 0. This will represent the value of ‘q’.

Now that the values of p and q are obtained, we already know we have a ‘d’ value of 1, therefore we can begin modeling the ARIMA model. First, import the ARIMA function, then we model the Autoregressive (AR) model, with the indexed data and the order obtained (values of p, d, and q). Then we calculate the residual sum of squares. This exact same process is done for the Moving average (MA) model, but the order for this graph is found through trial and error. The goal is to receive the lowest residual sum of squares possible. After this is done, the exact same process is repeated in order to graph the ARIMA model, with the lowest residual sum of squares (compared to all 3 of the graphs).

Then we converted the fitted values to a series format in order to generate predictions. These values are then converted into cumulative sums to produce cumulative values. Then the predictions for the fitted values are generated, and the data is exponentiated in order for it to come back to its original format.

Finally, the results can be predicted and viewed on a graph by applying the plot predict function. Our data was represented monthly and we had 211 different entries, therefore a threshold of more than 211 will need to be provided in order to predict future values. For example, to generate 5 years of predictions, a threshold of 271 will need to be provided as 5 years is equal to 60 months. The predictions can also be viewed in array format (with floating values) with the forecast function, as well as providing how many future months want to be predicted.

IV. Evaluation and Results

The modeling and prediction of the five different programming languages is outlined below.

1. *C/C++*

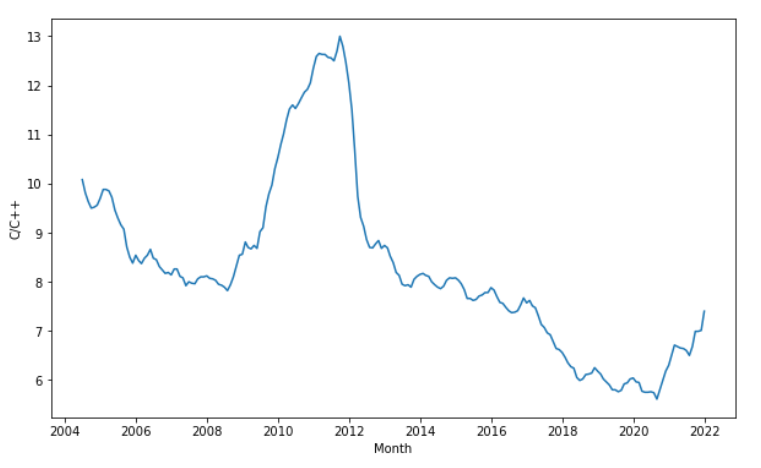


Fig. 2 C and C++ Popularity, 2004-2022

The model shows that the popularity of C and C++ has been fluctuating over the years, with its peak of popularity being between 2010 and 2012. After that, it had a significant decrease in popularity until 2020, where it started gaining some interest again.



Fig. 3 C and C++ Predicted Popularity, 2022-2026

In the ARIMA model prediction, C/C++ popularity is expected to see a slow decline over the few upcoming years. However, the confidence interval shows a very wide range of possibilities for C/C++, so it’s not a very certain negative forecast.

1. *Java*

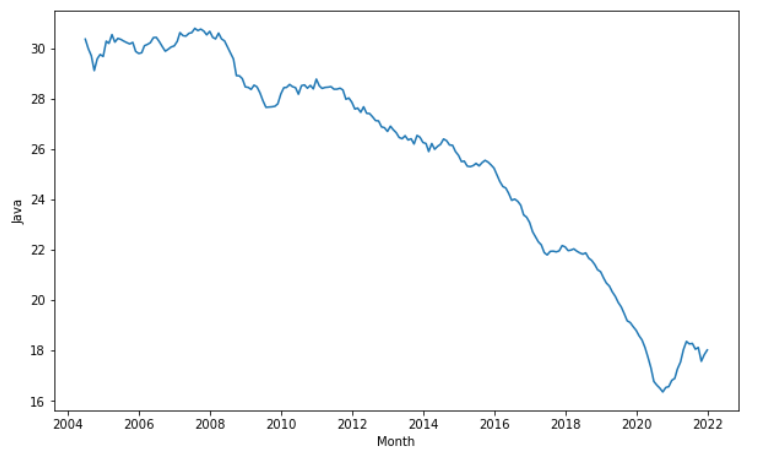
**

Fig. 4 Java Popularity, 2004-2022

The model shows that the popularity of Java has been on the decline since around 2008, with slight increases in popularity every now and then. Its popularity increased ever so slightly after 2020, but it seems to have decreased again in 2021.

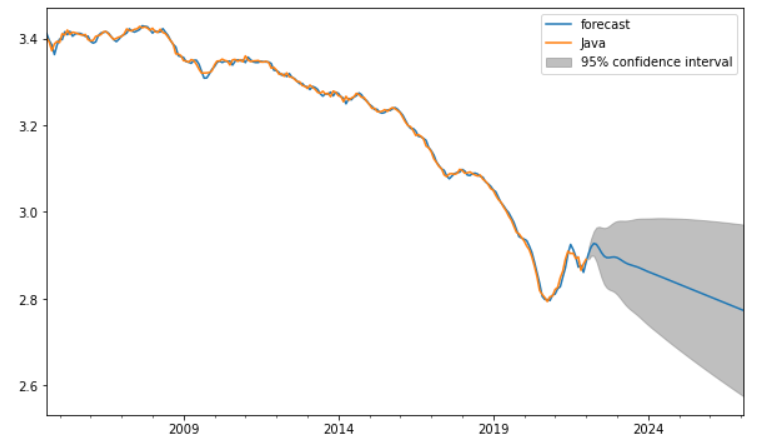


Fig. 5 Java Predicted Popularity, 2022-2026

In the ARIMA model prediction, Java popularity is expected to continue decreasing steadily over the next few years. The confidence interval range seems to also support the language’s decline.

1. *JavaScript*

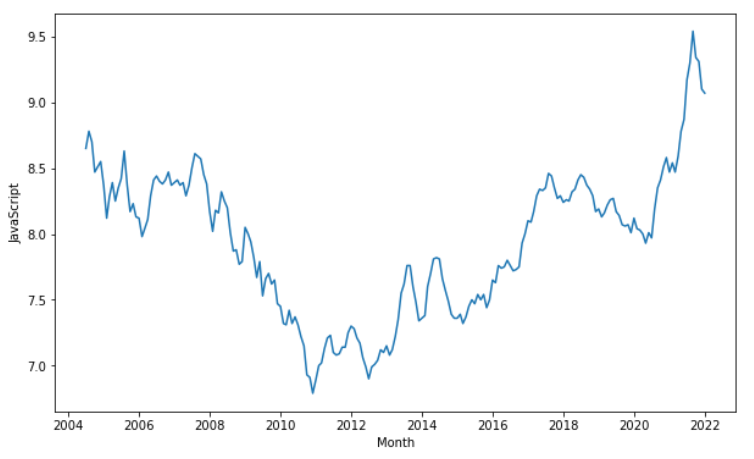


Fig. 6 JavaScript Popularity, 2004-2022

The model shows that the popularity of JavaScript has been fluctuating over the years, with various increases and decreases in popularity being seemingly random. The language saw a sharp increase in popularity between 2020 and 2022, but it seems to currently be on the decline.



Fig. 7 JavaScript Predicted Popularity, 2022-2026

The ARIMA model prediction for JavaScript seems to reflect that fluctuation in popularity through the confidence interval range, which shows a huge difference in high and low points. The forecast shows a very slight increase in popularity over the next few years, but it doesn’t seem to be a very confident prediction.

1. *PHP*

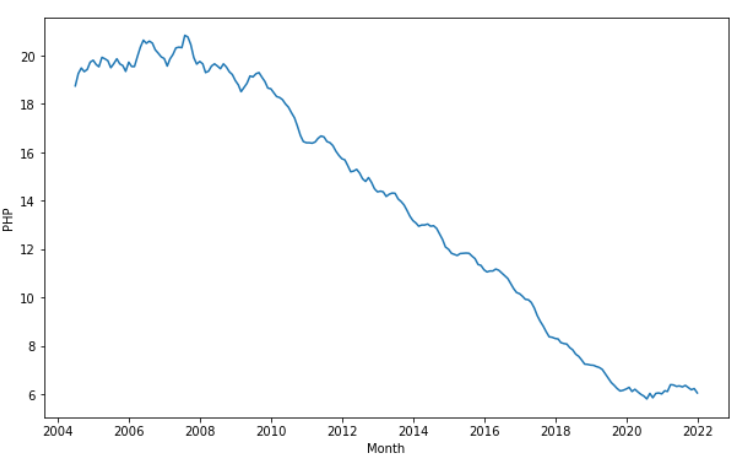


Fig. 8 PHP Popularity, 2004-2022

The model shows that PHP’s popularity has been decreasing undeniably over the years. PHP was most popular back in 2008, but has been losing popularity ever since.

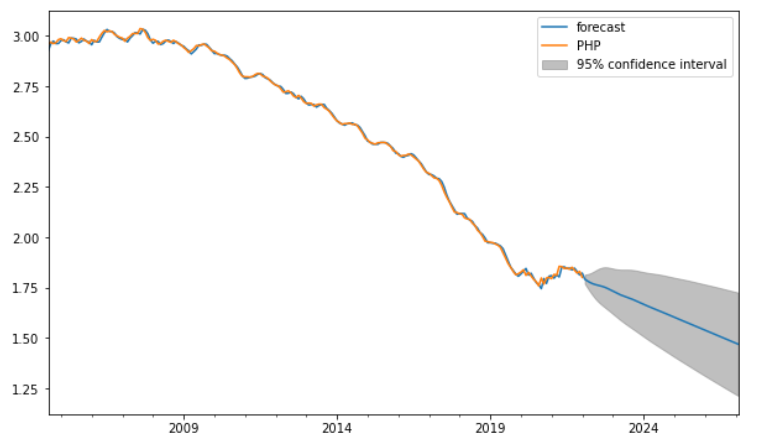


Fig. 9 PHP Predicted Popularity, 2022-2026

As expected, the ARIMA model prediction for PHP’s popularity over the coming years is surely a steady decline. Even the confidence interval concurs the forecast and shows a very small range that also points downward.

1. *Python*

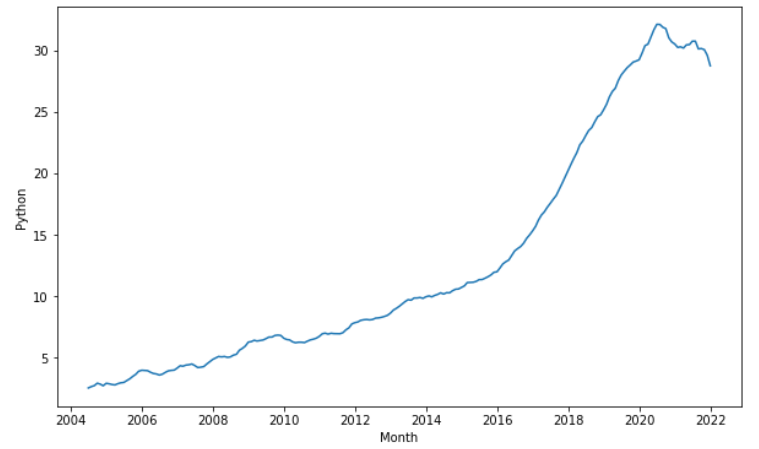


Fig. 10 Python Popularity, 2004-2022

As modeled above, Python has been significantly rising in popularity over the years. With only a slight dip in popularity in 2020, it is still extremely popular and remains the most popular out of all 5 languages outlined in this paper.

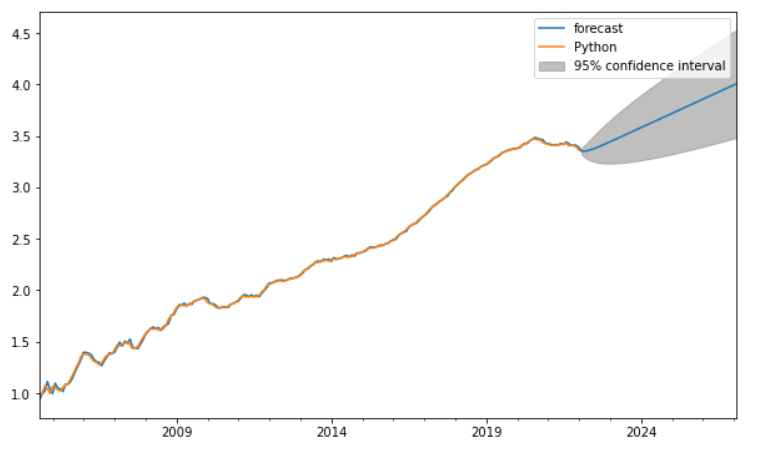


Fig. 11 Python Predicted Popularity, 2022-2026

The ARIMA model aligns itself with the original model and predicts a steady upward trend for Python’s popularity over the coming years. It is expected that Python will continue to grow in popularity even if it has a slight decline sometime in the coming months, as shown through the confidence interval range.

Results summary table:

| Language | Highest Value % | Lowest Value % | Forecast | Prediction Confidence Level |
| --- | --- | --- | --- | --- |
| C/C++ | 13 | 3 | Decrease | Uncertain |
| Java | 35 | 16 | Decrease | Confident |
| JavaScript | 9.5 | 6.5 | Slight Increase | Uncertain |
| PHP | 21 | 5 | Decrease | Confident |
| Python | 35 | 3 | Increase | Confident |

V. Challenges and The Future

There are a few challenges with trying to predict the future in that there is no way to reach 100% accuracy. Some challenges faced during this project include finding accurate, reliable, well-populated datasets to use for modeling that are also significant in the content they provide in terms of research. It was also difficult to determine the *p* & *q* values when plotting the MA and the ARIMA models. The values were finally determined using trial and error.

As for the future of this project, it could be improved further with the introduction of more frequent data collection, like having daily statistics as opposed to only monthly ones. That would greatly enhance the accuracy of the models, especially with regards to future predictions. Similarly, having research data covering a longer period of time would be very helpful for the same reasons. Another point would be the usage of more seasonal data to allow for the production of more complex results. It would cause the predictions to be less linear and as such more complex.

VI. Conclusion

The five most popular programming languages were successfully modeled in terms of past, present, and future popularity. The results show that C/C++, Java and PHP are 3 languages that are on a downwards trend in terms of popularity. JavaScript looks like it will have a very slight rise in its trend, but it is currently trending downwards which takes away our confidence in its prediction. Python, on the other hand, is showing constant growth in popularity and is expected to continue to do so. This suggests a growing interest in data analysis and visualization as well as machine learning and AI technologies since Python is most commonly used in these fields. It is safe to assume that out of the five programming languages observed, python will be the most popular in the future, possibly making it the most used language in many aspects.

References

1. C. Maklin, “Arima model python example - time series forecasting,” *Medium*, 21-Jul-2019. [Online]. Available: https://towardsdatascience.com/machine-learning-part-19-time-series-and-autoregressive-integrated-moving-average-model-arima-c1005347b0d7. [Accessed: 10-Apr-2022].
2. J. Brownlee, “A gentle introduction to autocorrelation and partial autocorrelation,” *Machine Learning Mastery*, 14-Aug-2020. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/. [Accessed: 03-Apr-2022].
3. “Stationarity,” *6.4.4.2. stationarity*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm. [Accessed: 02-Apr-2022].
4. M. Khalid, “Most popular programming languages since 2004,” *Kaggle*, 02-Jan-2022. [Online]. Available: https://www.kaggle.com/datasets/muhammadkhalid/most-popular-programming-languages-since-2004. [Accessed: 04-Apr-2022].
5. P. Roy, “Annual ticket sales,” *Kaggle*, 26-Jan-2022. [Online]. Available: https://www.kaggle.com/datasets/pratyayroy/annual-ticket-sales. [Accessed: 01-Apr-2022].
6. W. Kenton, “Why stochastic modeling is less complicated than it sounds,” *Investopedia*, 08-Feb-2022. [Online]. Available: https://www.investopedia.com/terms/s/stochastic-modeling.asp. [Accessed: 04-Apr-2022].
7. S. Prabhakaran, “Arima model - complete guide to time series forecasting in python: ML+,” *Machine Learning Plus*, 08-Mar-2022. [Online]. Available: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/. [Accessed: 06-Apr-2022].
8. A. Hayes, “Autoregressive Integrated moving average (ARIMA),” *Investopedia*, 08-Feb-2022. [Online]. Available: https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp#:~:text=An%20autoregressive%20integrated%20moving%20average%2C%20or%20ARIMA%2C%20is%20a%20statistical,values%20based%20on%20past%20values. [Accessed: 08-Apr-2022].
9. T. Kidder, "History of Programming Languages", LEARN academy - San Diego's Original Coding Bootcamp, 2019. [Online]. Available: https://www.learnacademy.org/blog/first-programming-language-use-microsoft-apple/. [Accessed: 12- Apr- 2022].
10. "A History of Computer Programming Languages", Online College Plan. [Online]. Available: https://www.onlinecollegeplan.com/computer-programming-languages/. [Accessed: 12- Apr- 2022].