A dissertation submitted to the **University of Greenwich**   
in partial fulfilment of the requirements for the Degree of

Master of Science

*in*

**Data Science**

**Julia For Data Science**

**Name:** Ajai Daye   
**Student ID:** 001145148

**Supervisor:**  Timothy Reis  
**Submission Date:** January 2023

**Word count:** 12,087

Julia for Data Science

Computing & Mathematical Sciences, University of Greenwich, 30 Park Row, Greenwich, UK.

*(Submitted 17 January 2023)*

**ABSTRACT**. Data science is the process of extracting information from raw data and adapting it into understandable insights, recommendations, or predictions. Data science entails a combination of skills that can range from statistics, mathematics, computer science to business analytics and machine learning. It often combines multiple topics together and focuses on producing the most accurate predictions possible through data analysis. In today's world, data science is used to examine areas such as stock market analysis, customer relationship management and sales performance analysis. However, it is not limited to these examples. Data science is helpful for anyone with a business or organization who needs to understand their customer base better. It is also used to analyse trends and new technologies, and evaluate the risks associated with new products or business strategies. While statistical mathematics is not always relevant to data science, the use of statistics in data science allows you to determine what's going on with your data. Statistics lets you determine whether a sample is representative of the whole population or not.

The aim of this project is to see if Julia has any advantages over python in Data science, specifically in machine learning algorithms. In this project, Julia will be compared to python in terms of ease of use and speed. The algorithms that are used in this project is Decision Tree, Random Forest, Logistic Regression and Linear Regression. A decision tree, a type of classification algorithm, helps you to find the most appropriate classification candidate. A random forest is built using decision trees and also helps you to predict the outcome of new cases while linear regression uses ordinary least squares as a model to predict the outcome of new cases. Logistic regression uses a linear model to predict the outcome of new cases.

The algorithms will be tested on two different datasets one being the Tesla stock price dataset and the other dataset from AutoScout24 with information about new and used cars. The datasets from AutoScout24 will contain information about different cars grouped into categories. They also have information about the average price of these cars. All the algorithms will be implemented on both python and Julia. The results between both languages will be evaluated based on four categories: Ease of Implementation, Speed, Memory Usage and readability. It was found that Julia has some major advantages compared to Python in the specified categories above, especially in term of speed.

**Keywords:** Julia, Python, Machine Learning, Supervised Learning, Linear Regression, Logistic Regression, Decision Tree.

**Preface**

This is the final project report for the University of Greenwich's MSc in Data Science programme. First and foremost, I want to express my gratitude to God Almighty and to my family and friends for their unwavering support and assistance throughout the entire journey. I also want to express my gratitude to my supervisor for his constant encouragement and suggestions throughout this project. This project was one of the project ideas proposed by the staff in the Moodle page.

As part of this project, I aimed to apply machine learning algorithms in Julia and Python and checked if Julia has some advantages over python in the field of Data science. In this project, Julia will be compared to python in terms of ease of use and speed. This project provided in-depth understanding regarding the application of the course's theoretical modules as well as the best practises to be followed when performing machine learning. A good understanding of the presenting and reporting skills needed when working on the project was gained via the supervisor's regular contact.

The report will include a thorough account of the project's various phases along with the advantages and disadvantages of Julia in the field of Data Science.

**Acknowledgements**

I would like to express my gratitude to the following people for their support and assistance during my MSc project.

First and foremost, I would like to thank my supervisor, Professor Timothy Reis, for agreeing to be my supervisor and for his invaluable guidance and support throughout the project. His insights and expertise were essential to the success of this work.

Finally, I would like to thank my family and friends for their encouragement and support during the project.

The views expressed in this report are my own and do not necessarily reflect the views of the people and organizations acknowledged above.

Table of Contents

[List of Figures 7](#_Toc124897661)

[List of Tables 8](#_Toc124897662)

[1. Introduction 8](#_Toc124897663)

[1.1. Overview 8](#_Toc124897664)

[1.2. Roadmap of Report 10](#_Toc124897665)

[2. Literature Review 10](#_Toc124897666)

[2.1. Overview 10](#_Toc124897667)

[2.2. Findings 11](#_Toc124897668)

[2.3. Background Research 12](#_Toc124897669)

[2.4. Summary 13](#_Toc124897670)

[3. Analysis of the System 14](#_Toc124897671)

[3.1. Legal, Social, Ethical and Professional issues 14](#_Toc124897672)

[3.2. Cost Analysis 14](#_Toc124897673)

[4. What is Julia 15](#_Toc124897674)

[5. Advantages of Julia 16](#_Toc124897675)

[6. Machine Learning in Julia 17](#_Toc124897676)

[6.1. Supervised Learning 18](#_Toc124897677)

[6.2. Unsupervised Learning 18](#_Toc124897678)

[6.3. Artificial Neural Networks 19](#_Toc124897679)

[6.4. Deep Learning 19](#_Toc124897680)

[7. Data Selection and EDA 20](#_Toc124897681)

[7.1. Overview 20](#_Toc124897682)

[7.2. Data Structure and Distribution 21](#_Toc124897683)

[7.3. Data Splitting 25](#_Toc124897684)

[7.4. Data Pre-processing 26](#_Toc124897685)

[8. Model Selection and Training 28](#_Toc124897686)

[8.1. Overview 28](#_Toc124897687)

[8.2. Linear Regression 29](#_Toc124897688)

[8.3. Logistic Regression 30](#_Toc124897689)

[8.4. Decision Tree Classifier 31](#_Toc124897690)

[8.5. Random Forest Classifier 32](#_Toc124897691)

[8.6. Model Training 34](#_Toc124897692)

[9. Model Evaluation 35](#_Toc124897693)

[9.1. Overview 35](#_Toc124897694)

[9.2. Evaluation Matrix 35](#_Toc124897695)

[9.3. Evaluation 36](#_Toc124897696)

[10. Conclusion 38](#_Toc124897697)

[10.1. Overview 38](#_Toc124897698)

[10.2. Findings and Recommendations 38](#_Toc124897699)

[10.3. Areas of Future Work 39](#_Toc124897700)

[10.4. Personal Evaluation 39](#_Toc124897701)

[10.5. Future of Julia 40](#_Toc124897702)

[10.6. Julia vs Python 40](#_Toc124897703)

[References 42](#_Toc124897704)

[Bibliography 43](#_Toc124897705)

**List of Figures**

[Figure 1 Names of columns 23](#_Toc124897640)

[Figure 2 Updated names 23](#_Toc124897641)

[Figure 3 Info cars dataset 23](#_Toc124897642)

[Figure 4 Dropping missing values 24](#_Toc124897643)

[Figure 5 Number of Gear types 24](#_Toc124897644)

[Figure 6 Different makes 24](#_Toc124897645)

[Figure 7 All make vs HP 25](#_Toc124897646)

[Figure 8 One hot encoding in python 27](#_Toc124897647)

[Figure 9 One hot encoding the dataset 27](#_Toc124897648)

[Figure 10 Julia load tesla 36](#_Toc124897649)

[Figure 11 Python load tesla 36](#_Toc124897650)

[Figure 12 Julia load time cars 36](#_Toc124897651)

[Figure 13 Python load time cars 36](#_Toc124897652)

**List of Tables**

[Table 1: Dataset 1 Structure 22](#_Toc124900010)

[Table 2: Dataset 2 Structure 23](#_Toc124900011)

[Table 3: Tesla output 37](#_Toc124900012)

[Table 4: German cars outputs 37](#_Toc124900013)

1. **Introduction**
   1. **Overview**

Data Science is currently a hot new field of study, with big companies like Google and Facebook competing for top talent. It is necessary for any professional to have at least a basic understanding of how it works, but plenty of people are still scratching their heads as to what that might actually entail. One thing we've learned over the years is that if you want to get ahead in data science, you're going to need a solid background in math, statistics and computer science. (D, 2019)

Data science has become a general term that includes everything from analyzing your sales numbers to training an Artificial Intelligence algorithm to categorize your customers by zip code. But there are different facets of data science, and one of them does require a decent understanding of computer programming. But it's not all about coding. There are also a lot of mathematical and statistical concepts that are integral to the field. For example, you should know how to analyze the data you've gathered, identify correlations and make predictions from your analysis. This is where data science intersects with statistics and computer science. (Jake VanderPlas, 2016).

Julia is essentially Julia the language, while the primary focus of JuliaSoft is developing and distributing Jupyter Notebook. It provides functions for converting between different data types, both standard ones like numbers and strings as well as more exotic ones like lists or maps. Julia also provides a variety of error-handling mechanisms, that can be programmed for in any way, including exceptions or explicit error-handling with try/catch blocks. Julia is defined as a language rather than a run-time environment. It is not provided as part of the standard distribution and must be installed separately; Julia is freely available to anyone who wants to install it on their own computer. Julia provides code introspection macros so that way you can see what your code actually compiles to. You can write Matlab, Python, R, C++, and more, right inside a Julia file. (Voulgaris,Z, 2020).

Machine learning can be divided into two major branches, being supervised and unsupervised. The most common type of machine learning is supervised learning, where data has been labeled by humans as either a positive (e.g., an image containing a dog) or negative (e.g., an image that does not contain a dog). Unsupervised machine learning — where there are no labels at all — is less common (Han Liu, 2017). Supervised learning is a way of performing machine learning where the machine-learner is provided with a set of "training" examples and their "right" outputs. The computer uses what it learns from these examples as input to calculate an output for new, or "unseen", inputs. The aim is that this should result in a system that correctly classifies the unseen instances; whereas those instances not in the training data will be predicted incorrectly. Unsupervised learning is a type of machine learning where the learner is not given any guidance and develops its own strategy based on past experience. unsupervised learning is often used in reinforcement learning systems, where the ultimate goal is to maximize some quantity such as expected reward. Unsupervised learning algorithms usually employ two basic objectives: determine which actions are part of the optimal solution and learn to generalize from past experiences.

This project aims to provide an in depth analysis of Julia and its applications in Data Science. Different Machine Learning models were taken for the tasks based on the selected dataset. The project upon completion will provide the advantages and disadvantages Julia has over other programming languages including Python. And finally see if Julia is a better option for doing data science tasks

* 1. **Roadmap of Report**

The report's initial emphasis will be on the background investigation and related work that has already been completed. This will reveal details on the papers and books that were used as references for the project and how they were helpful. The next stage of the report includes a brief summary of Julia and its advantages. It will also include the machine learning techniques used in Julia.

Following that, the report will concentrate on the data set used and how the data loading and pre-processing procedures that were completed, including information about the data loading, data splitting tasks, and observations made during these phases.

The machine learning models used in the experiment, their training procedures, the outcomes, and a comparison of the various models' capabilities will be the main topics of the next chapters. During this, the many experiments carried out when evaluating the model will also be examined.

And the last chapters will be about the advantages and some disadvantages of the Julia when compared other programming languages like python.

Finally the report will conclude with the summary and conclusion which will give a brief summary of the whole experiment in the report.

1. **Literature Review**
   1. **Overview**

Machine learning is a field of study in artificial intelligence which is concerned with algorithms that allow computers to "learn" and make decisions based on information. Machine learning systems use mathematical methods to allow them to construct very large models from using a lot of data, as well as the ability to make predictions based on new data. The ability for machines to learn is an important development in the field of artificial intelligence because it can lead to better predictions, or even the creation of new machines that are smarter than humans. The field of machine learning is still very new and is still being used by companies like Google, but there are a few researchers who are trying to bring machine learning into the public eye. (Raschka,2015).

Julia is an open-source language with a unique syntax that makes it easy for expressive programmers everywhere to write code. It's commonly used for web development and can be seen in JavaScript and Python, among other languages. Julia is meant to provide power users with the ability to write extremely efficient code, rather than spending hours trying to map out all their data structures before writing a single line of code, as is normally the case when writing low-level languages like C++ or Java.

There were many publications, books, and papers that can offered knowledge about the project and was clinical in providing me with the information about data pre-processing, data loading and about the various machine learning models implemented in the project.

Kaggle is an excellent location to explore for work that has been done on comparable topics. People all around the world often upload fresh datasets and test out novel machine learning techniques on various datasets in this online community.

As the aim of the project was to compare Julia with other programming language like python.

Also, as a master’s student books on Machine Learning can provide more concrete knowledge about the different steps in creating a machine learning model. It can also provide information about the different models such as the traditional machine learning models which can act as baseline models along with neural network models which can be built on top of the baseline models. A final accuracy comparison between the different models will provide the best model that can be used depending on the conditions.

* 1. **Findings**

The book “Data Science Using Python and R” gives a complete introduction to the field of data science, teaching you how to process and analyze large and complex datasets. The book begins with a historical perspective on the development of statistics and data science, including an overview of programming concepts in Python and R. Using real-world examples and hands-on explanations, the book teaches you how to process and analyze your data using Python and R. In the second part of the book, you will learn how to apply machine learning to your datasets using Python, R, and scikit-learn. You will also learn how to develop your own machine learning algorithms using scikit-learn. As most of the models created in this project involves scikit learn it was vital in completion of the whole project.

“Python Data Science Handbook: Essential Tools for Working with Data” by Jake VanderPlas provides a focus on the most important tasks in data science such as importing, filtering, aggregating, calculating mean and standard deviation, and transforming data from one format to another. The book also covers essential tools used by data scientists such as NumPy arrays, Pandas objects, Matplotlib graphs and messages. This practical guide is also appropriate for those with little experience who are not familiar with programming.

Nazarathy, Y., Klok, H.'s “Linear Regression and Extensions. In: Statistics with Julia” talks about linear regression, which is a technique for predicting continuous responses. It will help you predict outputs from an input. Linear regression is one of the most popular techniques used in social sciences and assists with data analysis. It can be used with many kinds of different datasets, including survey information, economic analyses, and medical research such as gene mapping or determining cancer risks. Linear regression can be used to estimate the mean, variance, and slope of a regression line. It is also used for prediction where there is known data about the dependent variable. The book gave a lot of insights for doing machine learning models using Julia, from splitting dataset using lathe function to understanding other hyperparameters in Julia model.

The platform Kaggle was really helpful in this project as the two datasets taken in this project is from Kaggle. It was also helpful in other parts of the project like data pre-processing.

* 1. **Background Research**
     1. **Primary Research**

The main input was the data being used and the useful insights that can be found out from the data itself. A fixed amount of time was dedicated to do the EDA analysis of the data. This provided useful information about the data distribution and the imbalances that are present in the data. This analysis also resulted in the addition of a new dataset to resolve the imbalances in the first dataset. As a result, the final dataset was a combination of two datasets which meant that the model was generalizing well to new data that was being fed.

The discussions that took place with the supervisor was also helpful as this opened new possibilities to explore with respect to models being considered, the models that need to be further analysed using hyperparameters, the distribution of dataset, the design of application etc. These discussions also enabled to discuss and solve the challenges that occurred along with making sure that the project is moving on the right path.

The initial model experiments and the associated results also showed that both datasets need to be mixed to resolve the lower test accuracy associated with model predictions.

* + 1. **Secondary Research**

The resources that were referred for the project include but not limited to the following ones:

* Julia Introduction by Edelman.
* Julia Language in Machine Learning: Algorithms, Applications, and Open Issues by Gao, Kaifeng & Mei, Gang & Piccialli, Francesco & Cuomo, Salvatore & Tu, Jingzhi & Huo, Zenan.
* “Data cleaning”. Ilyas, Ihab F and Chu, Xu.
* Julia Data Science by Jose Storopoli, Rik Huijzer, Lazaro Alonso.
* Stock Market Analysis Using Linear Regression and Decision Tree Regression by R. Karim, M. K. Alam and M. R. Hossain.
  1. **Summary**

When working on a project, it can be helpful to do research and learn about related work that has been done in order to identify potential problems that could occur during the project as well as to learn more about the road map or the project structure that needs to be followed when working on the project. This makes it possible to work and learn more about others' creative projects, as well as to interact with and critique their work.

The many discoveries made it possible for the project to advance without experiencing any significant setbacks. Additionally, it made it possible to specify precisely the actions that must be taken at each project stage, such as the models that must be chosen and tested as well as the necessary ratio of dataset partition.

1. **Analysis of the System**
   1. **Legal, Social, Ethical and Professional issues**

My project is Julia for data science, which is about the advantages julia have over other programming languages when doing data science techniques like machine learning. For this project no legal, social or ethical issues are violated. In the course of this project, extra effort was taken to ensure that no such issues develop and, if they do, that their impact is as minimal as possible. I want to make sure that in the event of legal issues I can provide a solid explanation and proof that this project is not violating any laws, social regulations or ethics. For example, I do not want to release any sensitive data which is not my property.

As Julia is free and open source and it works on every operating system, with no need to install anything. You can download julia from github. Julia is a language that lets people take advantage of the power of a modern computing machine at an unprecedented level without sacrificing ease-of-use or speed. As it is open source there will not be any copyright issues.

As for the dataset , the two datasets taken for the project are the german auto cars dataset and tesla stock price dataset which is available online in Kaggle for public. No changes were made to the initial dataset

* 1. **Cost Analysis**

The project was completed using existing resources and no additional expenses were incurred. The cost of using Julia for a data science project will depend on several factors such as the size and complexity of the project, the resources required etc. Julia is an open-source software, so it is free to download and use. Depending on the size and complexity of the project, additional hardware resources may be needed, such as high-performance computers or clusters. But no additional hardware was required for this project and was done using my personal pc. Julia has a large ecosystem of third-party libraries and packages that can be used for specific tasks. Some of these packages may be free, while others may have associated costs. As all the packages used in this project were free and open source no cost was required for the packages. All the packages in python was also free.

1. **What is Julia**

Julia programming language is a new language, designed to be "friendly" and "non-intimidating" for those who are new to programming. Julia was designed by a team from MIT and UC Berkeley, with the purpose of being faster than Python and R, but easier to learn than Matlab. Julia is open-source (MIT license), so it's available for anyone to use. Julia has also been used in many academic programs to teach kids how to code.

At the time of its development, Julia was being developed as a way to measure its potential by setting goals. To be successful, Julia would have to reach an industry-wide adoption level and inspire more interest among students in computer science. These goals have been achieved and surpassed. Julia has become a programming language used by students and professionals around the world, including industry leaders such as Uber, Coinbase, Dropbox and NetSuite.

In the initial stages of Julia, there were different goals to achieve. These goals were:

Julia has a wide variety of language constructs and data structures, including:

Julia's text format (called "Julia-code") is similar to Python's and R's. Julia's syntax is inspired by Haskell, but does not use any of Haskell's type classes or dependent types. Julia also lacks some features of Haskell, like infix notation or higher-order functions, as well as a module system and class inheritance. (Edelman, 2015)

It has a manual memory management system. Julia also includes a package manager, "Pkg".

As far as the development is concerned, Julia was started in 2009. However, Julia is still under development and there are some features that are still being implemented to enhance the language further. Julia has a very active community which is appreciated by its users. On GitHub there are 6,483 contributors while on StackOverflow the number of users asking questions and receiving answers is around 4,000 per month. These numbers show the number of popular and active users who are using Julia. Julia's website was launched on December 9, 2013. The website includes tutorials and the "how to" sections of the project documentation. It also includes links to users' projects which utilize Julia as part of their codebase.

Julia is faster than Python and can be used for solving a multitude of problems. It’s not limited to one particular domain and can be used for a variety of applications. It’s also one of the languages adopting fastest in recent years with an increase in growth rate from 0-5%. This is an indication that Julia is gaining popularity across the globe as it’s value-for-money investment. Julia has its problems though; there's little availability of books on Julia programming and few tutorials to learn the language from scratch.

1. **Advantages of Julia**

Julia, Python, and R are all popular programming languages that are commonly used for scientific and technical computing. There are several advantages of Julia over other programming languages that are commonly used for scientific and technical computing, including:

1.Speed: Julia is designed to be fast. It can perform complex computations at speeds that are similar to those of lower-level languages like C and Fortran, while still being easy to use and read.

2. Ease of use: Julia's syntax is similar to that of other high-level languages, such as Python and MATLAB, making it easy to learn for users familiar with these languages.

3.Flexibility: Julia is a high-level, high-performance dynamic programming language, allowing user to easily switch between the high-level and low-level operations as per their need

4.Built-in support for parallel computing: Julia has built-in support for parallel computing, which allows it to easily take advantage of multiple processors or cores to speed up computations (Edelman, 2015).

5.Extensive ecosystem: Julia has a large and growing ecosystem of packages and libraries, which expands its functionality and makes it easy to perform a wide range of tasks without having to write complex code.

6.Open-source: Julia is open-source, which means that it is free to use and that anyone can contribute to its development.

7.Interoperability: Julia's LLVM-based JIT compiler allows it to easily call C and Fortran libraries, and libraries from other languages via C-compatible interfaces.

8.It's growing popularity: Julia is being increasingly adopted by a diverse range of organizations from research and academia to industry, giving a room for further developments and improvements as well as more users support.

Julia is a compiled language, which means that the code is translated into machine code before it is executed. This can result in faster performance than interpreted languages like Python, because the overhead of interpreting the code at runtime is avoided. Julia has a built-in just-in-time (JIT) compiler that can dynamically optimize the code at runtime. This can further improve performance by generating machine code that is optimized for the specific data types and operations that are used in the code. Julia has a number of built-in features for numerical computing and data manipulation, such as support for multi-dimensional arrays and linear algebra operations. These features are implemented in a way that is highly optimized for performance, and they can be used through the built-in Julia libraries to take advantage of Julia's performance characteristics.

1. **Machine Learning in Julia**

Machine learning can be classified generally into two learning methods: supervised and unsupervised learning. Unsupervised learning relies on data that is unlabeled without any training examples. Supervised learning uses unlabeled and labeled data to train a model. In this blog post, we will be discussing unsupervised learning methods.

* 1. **Supervised Learning**

Supervised learning is a type of machine learning that requires a human to set ground rules for what the machine is supposed to learn from data and then provide feedback on how well it does, so it can adjust its behaviour accordingly. This method has been used for teaching machines about language, hand-writing identification and more.

In supervised learning, data comes paired with labels that specify what should be done with it. For example, it could be used to teach a computer how to identify what's in an image or whether someone is in a video. When you take a photo, the camera app will inform you if there is any red-eye in the image so that you can quickly remove it. This is accomplished by analyzing an image and feeding its findings back into the system so that it recognizes red-eye next time and automatically corrects it. (Gao, 2020)

The goal of this type of machine learning is to provide computers with the ability to learn from small amounts of data and make decisions on the data provided. It's different from unsupervised learning, where a computer can recognize patterns in data but doesn't have any guidance about what it means. The main practical advantage of supervised learning is that it can be used even in cases where no labels are available.

The different supervised learning algorithms that can be used in Julia include: Linear regression Logistic regression, Decision Tree, Random forest, Gradient Boosting, SVM, Naïve Bays etc.

Linear regression and Logistic regression can be done using GLM.jl package. Decision Tree can be created using DecisionTree.jl package. The Random forest can be reated using RadomForest.jl and DecisionTree.jl packages. Gradient Boosting using the packages XGBoost.jl or LightGBM.jl or MLJ.jl and Naive Bayes using MLJ.jl package. It is worth noting that some of the packages might contain multiple algorithms and not just the one specified. All of the above packages are open-source and can be installed by running the command using Pkg; Pkg.add("PackageName") in the Julia environment.

* 1. **Unsupervised Learning**

Unsupervised learning is the process of an algorithm finding patterns in data without any human intervention, and it can be thought of as a form of machine learning. Algorithms learn patterns in unlabelled data and are found through exploratory analysis, clustering algorithms such as k-means and hierarchical clustering, feature selection based on both distance measurement functions and non-metric similarity functions. (Raschka, Python Machine Learning, 2015)

There are many different unsupervised learning algorithms that can be used in Julia. Some popular options include:

Clustering: Clustering is the process of grouping similar data points together. Common clustering algorithms include k-means, hierarchical clustering, and density-based clustering.

Dimensionality reduction: Dimensionality reduction is the process of reducing the number of features in a dataset while preserving as much information as possible possible (Raschka, Python Machine Learning, 2015). Common dimensionality reduction techniques include principal component analysis (PCA) and singular value decomposition (SVD).

Anomaly detection: Anomaly detection is the process of identifying data points that are unusual or different from the majority of the data. Common anomaly detection techniques include density-based methods and distance-based methods.

All of the above algorithms are available in Julia through various machine learning packages such as Clustering.jl, DimensionalityReduction.jl, AnomalyDetection.jl

* 1. **Artificial Neural Networks**

An artificial neural network is a type of machine learning model that consists of multiple layers of interconnected neurons. Additionally, they are inspired by the structure and function of the human brain. This means that they contain a large number of hidden layers and loops made up of self-learning units, just like the human brain; allowing them to deal with sensory data like that found in images or audio recordings.

The basic building block of an ANN is the artificial neuron, which is a mathematical function that receives one or more inputs, applies a set of weights to them, and produces an output. The output is then passed through an activation function, which is a non-linear function that introduces non-linearity in the model.

In python Artificial neural networks (ANNs) can be implemented using a number of libraries including TenserFlow, Keras, PyTorch, Theano, Scikit-Learn etc.,

In Julia, there are a number of packages available for working with artificial neural networks, including Flux.jl, Knet.jl, and MLJ.jl. These packages provide a range of tools for building, training, and evaluating neural networks, and support a variety of architectures and algorithms.

* 1. **Deep Learning**

Deep learning is a subset of machine learning that is concerned with neural networks that have a large number of layers, typically more than one hidden layer. These neural networks are called deep neural networks (DNNs) or deep networks. Deep learning is often used to solve sequential data problems, such as language translation and image recognition. The concept of deep learning is not new and is broadly used in computer vision, robotics and speech recognition. Several recent successes in Deep Learning are due to mathematical breakthroughs, but also due to improvements in hardware performance, that occurred at the same time. This combination of hardware and maths has led to the renaissance of Deep Learning. Today's deep learning libraries are more mature, provide more features and do more with less computation than before. (R. Gonzalez, 2022)

Deep learning algorithms are typically implemented using neural networks, and they are often trained using large amounts of data and powerful computing resources. Deep Learning provides a new way of solving complex problems using deep learning methods such as Convolutional Neural Networks, Recurrent Neural Networks, Long Short Term Memory (LSTM), and Generative Adversarial Nets. Deep learning is currently a hot research topic in academia and industry.

TenserFlow, Keras, PyTorch, Theano, Scikit-Learn etc can be used to implement deep learning algorithms in python. Some of the popular library options for implementing Deep learning Julia includes: Flux.jl,Knet.jl,MLJ.jl, JuliaDL. JuliaDL is a package for deep learning in Julia that provides an easy-to-use interface for building, training, and evaluating deep neural networks. It provides a high-performance implementation of the most popular deep learning models and frameworks such as TensorFlow, Keras, Pytorch and more.

1. **Data Selection and EDA**
   1. **Overview**

Exploratory Data Analysis (EDA) is an approach to analyzing and summarizing data sets to identify patterns, anomalies, and relationships. It is typically the first step in the data analysis process and helps to understand the underlying structure of the data. EDA involves visualizing data using plots and charts, summarizing key statistics, and identifying outliers and missing data. The goal of EDA is to gain insights and a better understanding of the data, rather than to confirm hypotheses or make predictions. This is particularly true in Data Science and Machine Learning applications. However, you should be very familiar with every step of EDA before embarking on a project.

EDA works by calculating summary statistics or counts to identify the most important features in the data set. Then, it calculates statistical models to describe these features and their relationships in a way that is expected to be more meaningful than the raw data. Finally, it uses these statistical models to make inferences about the data – for example, whether there is a difference between two data points or what relationship exists between two variables.

There are many different types of plots that can be used in EDA. You can use these tools to start exploring and understanding your data, even before applying a particular data mining technique, like clustering or classification. EDA works best when it can be performed in parallel.

Before choosing data for the machine learning models, careful examination is required. As a result, the data being used is crucial while working on a data science project with built-in machine learning models.

* 1. **Data Structure and Distribution**

The first dataset was the Tesla stock dataset having 2416 rows and 7 columns and it had the following structure:

|  |  |
| --- | --- |
| **Columns** | **Values** |
| Date | Dates |
| Open | Float |
| High | Float |
| Low | Float |
| Close | Float |
| Adj Close | Float |
| Volume | Int |

Table 1: Dataset 1 Structure

The second dataset was the German cars dataset having 46505 rows and 9 columns and it had the following structure:

|  |  |
| --- | --- |
| **Columns** | **Values** |
| mileage | Int |
| make | String |
| model | String |
| fuel | String |
| gear | String |
| offerType | Int |
| price | Int |
| hp | Int |
| year | Int |

Table 2: Dataset 2 Structure

One of the important steps in the pre-processing of the Tesla dataset was to convert the column names which was in Symbols to vector values and it was done in Julia as follows.

Graphical user interface, text, application

Description automatically generated

Figure 1 Names of columns

Graphical user interface, text, application, email

Description automatically generated

Figure 2 Updated names

As you can see there is large difference between both the datasets. The tesla dataset didn’t had any missing values or duplicates in them and was a perfect dataset so it didn’t need any sort of pre processing.

The second dataset had lots of missing values and duplicates in them and needed preprocessing and the string values in them was needed to be converted into integers through one hot encoding before training the dataset.

Table

Description automatically generated

Figure 3 Info cars dataset

Graphical user interface, text, application

Description automatically generated

Figure 4 Dropping missing values

We can see that there is significant difference in both the datasets and visualizing these datasets gave further information about the datasets. Some of the visualizations are given below.

Chart, bar chart

Description automatically generated

Figure 5 Number of Gear types

It was found that the number of semi-automatic cars were minimal so alterations were made to this to consider it as automatic.

Chart, bar chart

Description automatically generated

Figure 6 Different makes

The different car manufactures in the dataset was found.

Chart, histogram

Description automatically generated

Figure 7 All make vs HP

We could see that some of the car models didn’t have any HP value added to them so alterations were made to the dataset accordingly.

* 1. **Data Splitting**

Train and test split is an important step in machine learning because it allows you to evaluate the performance of your model on unseen data. When you train a machine learning model on a dataset, it "learns" the patterns and relationships in the data. However, it's possible that the model may not generalize well to new, unseen data. This is known as overfitting. By splitting the data into training and test sets, you can ensure that the model is tested on data that it has not seen during training, which gives you a better estimate of its performance on unseen data.

Additionally, by splitting the data into training and test sets, you can use the test set as a "hold-out" set to evaluate the model's performance and tune its hyperparameters. This allows you to compare different models or different configurations of the same model in a fair and unbiased way, which can help you make better decisions about which model to use in a production setting. The train-test split is also useful when evaluating the model performance when the data is limited, since if the data is not splitted, the model might perform well on the training dataset, but poorly on real-world unseen data.

In Julia, there are several packages available for data splitting, such as MLDataPatterns.jl and DataDeps.jl. These packages provide functions for splitting data into training, validation, and test sets. Additionally, the Julia Statistics library provides a function called "splitobs" which can be used to split a dataset into training and test sets. The function takes the dataset as an argument and a ratio indicating the proportion of the data that should be used for training, and returns a tuple containing the training and test sets.

In Julia the TrainTestSplit function from the Lathe.preprocess package was used to split the Tesla dataset df into a training set (75%) and a test set (25%). The train\_test\_split function from the ScikitLearn package was used to split the german cars dataset into training and test sets. The function takes in three arguments: the predictor variables (df\_x), the target variable (df\_y), and the proportion of the data that should be used for testing (test\_size). It splits the dataset df\_x and df\_y into training sets x\_train and y\_train and test sets x\_test and y\_test. The test\_size argument of 0.75, which means that 75% of the data was used for testing and 25% was used for training.The train\_test\_split function returns a tuple containing the training and test sets, which can be used to train and evaluate a machine learning model.

In python the sklearn train\_test\_split was used to split a dataset into training and test sets, which can be used to train and evaluate a machine learning model. The function takes in several arguments, including the predictor variables (X), the target variable (y), the proportion of the data that should be used for testing (test\_size), and a random seed (random\_state) to ensure reproducibility.

* 1. **Data Pre-processing**

Data pre-processing in julia can be a daunting task. There are a variety of ways to choose from and many different boundary cases to consider. The most important and fundamental step in data pre-processing is choosing your data type. This is a critical decision that will affect everything else about your data and it should be made early on in the process so that you have time to learn and consider all of the implications of your choice. (Witold Pedrycz, 2017)

For a long time, Julia was defined by its simplicity. It's just one language. It's designed to do exactly one thing: provide high-level mathematical tools that are very easy to use. With this focus on ease of use, the functionality has been largely limited—Julia provides powerful optimization algorithms and array operations but lacks the non-trivial data structures and libraries needed for working with large amounts of data in a concurrent manner. However, Julia 0.5 is coming at a time when tools like MapReduce and Apache Hadoop are gaining widespread use in large distributed data processing projects. Julia developers are often asked for an ability to process large amounts of data, and recently the need for concurrent programming libraries has come to the fore.

Data preprocessing is an important step in the machine learning pipeline and it involves cleaning, transforming and organizing the data so that it can be used to train a model. There are several packages in Julia that can be used for data preprocessing, some popular options include:

DataFrames.jl: DataFrames.jl is a package for working with data in Julia. It provides a DataFrame type that is similar to a data frame in R or a data table in Python. It includes functions for cleaning and manipulating data, such as missing value imputation, outlier detection, and data normalization.

Missings.jl: Missings.jl is a package for handling missing data in Julia. It provides a variety of techniques for imputing missing values, such as mean imputation and multiple imputation.

MLDataUtils.jl: MLDataUtils.jl is a package for machine learning data preprocessing in Julia. It provides a number of functions for handling and normalizing data, such as splitting data into train and test sets, handling categorical variables and one-hot encoding.

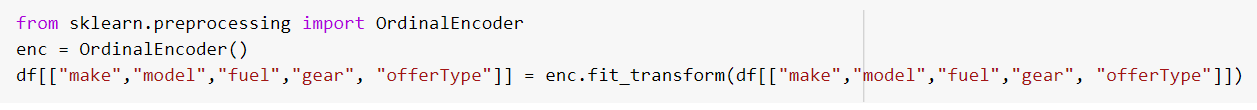


Figure 8 One hot encoding in python

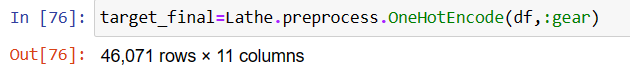


Figure 9 One hot encoding the dataset

The above two figures shows encoding in Julia and Python which converted categorical values to integers.

1. **Model Selection and Training**
   1. **Overview**

The crucial next step is the model training after the data has been loaded and formatted correctly. There are numerous models accessible, and new models are developed and used every day. Correct model evaluation, model selection, and algorithm selection procedures are essential in both academic and industrial machine learning research. (Raschka, 2018). The right model to use while solving an ML challenge is never easy to choose. We can choose models more methodically by carefully analysing the issue at hand.

When comparing Julia with Python, it would be useful to compare the performance and ease of use of a variety of models across the two languages. Some models that you could consider comparing include: Linear and logistic regression: These are basic models that are commonly used in machine learning, and will allow you to compare the performance and ease of use of Julia and Python for basic statistical modeling tasks. Decision trees and Random Forest: These models are commonly used for classification tasks and will allow you to compare the performance and ease of use of Julia and Python for more complex models.

But I didn’t use unsupervised models in this project.

* 1. **Linear Regression**

Regression models in Python are an analysis of a dataset, such as income and education levels. The data is fitted to a regression function with the goal of coming up with some relationship between the two variables. Regression models are useful for developing hypotheses about relationships among variables, but can also be used for prediction purposes.

Linear Regression is an algorithm that takes in data points and spits out an estimated value for a fixed point. Linear regression can be used for many types of data: time series, categorical data, spatial data, and so on. For example one can fit linear regression models for time series data to get predictions for future values for time-series objects. In multiple regression we can use linear models to predict various dependent variables. The LinearRegression fits a linear model of the form Y = l + b1·X1 + ... + b"m·XM". If any of the X variables is more than one column wide, then it is converted to multiple columns.

Linear regression in python is a relatively easy to understand and implement algorithm for solving a number of real-world problems, in python linear regression is very easy to implement using the numpy library.

Linear regression in julia is a simple statistical technique for predicting continuous variables from observations of independent, normally distributed data. It is used all over statistics to assess causality and to model changes, giving us the power to predict trends in individual data sets. Furthermore, it is particularly useful for dealing with large datasets. Linear regression analysis is based on fitting a line to the data between two sets of points.

The advantage of linear regression is that it can be used to model relationships between continuous or categorical predictor variables and a quantitative response variable.

We can use LinearRegression function from the GLM package and pass it some data points and a coefficient vector (or multiple coefficient vectors) to obtain an equation for our model. The GLM package contains all the tools necessary to fit statistical models based on General Linear Models. It provides a number of features such as automatic differentiation, support for various probability distributions, and model fitting methods. It can be used to create linear regression models in Julia by specifying the appropriate family and link function for the model. The linear regression was used to predict the Adj\_close using the other features.

* 1. **Logistic Regression**

Logistic regression is a regression model that uses a logistic of the odds of binary response. It was invented by statistician George W. Brown in 1954 and is used in statistics, machine learning, data mining and pattern recognition to predict categorical response based on one or more explanatory variables. Logistic regression is intended for use with binary outcomes when the underlying distribution of the outcome variable may be either (multinomial) logistic or other distributions related to the normal distribution. These cases do not occur often in science and industry, but can arise in many contexts.

There are three types of logistic regression models. The first is an additive model which applies the logistic model to a combination of explanatory variables. The second is an interaction model which is a special case of the additive model with only one explanatory variable. The third is an ordinal model which is a special case of the additive model with one or more explanatory variables.

The term "logistic" reflects the fact that this method applies the function "logit" to different variables. The name "logistic regression" was originally proposed by Arthur Goldberger in 1969, since it was a generalization of proportional-odds (probit) regression; later authors have acknowledged that Brown's 1954 paper should have been given that credit.

The main difference between logistic regression and linear regression is that in logistic regression the dependent variable is binary. Logistic regression adds to the usual linear regression a penalty term that converts the odds-ratio (a measure of association used in information theory) into a coefficient. This coefficient is interpreted as a measure of effect of the independent variable on the dependent variable.

In python logistic regression is done using the library Scikit-learn. Sklearn is a library for machine learning which was developed by the famous canadian professor, Yoshua Bengio. It has been widely accepted because of its extremely easy-to-use interface via high-level functions that require few lines of code. Logistic regression can be done easily with sklearn in python

Julia has a package called "GLM" and “ScikitLearn” which contains everything that you need to perform logistic regression. The " ScikitLearn " package includes everything from initial data preparation (sample labeling), model creation and checking, and hyperparameter optimization. It also contains helper functions for calculating various statistical tests of your model as well as visualizations for understanding the output of your models.

Logistic regression was used to predict if the car gear was Automatic or Manual.

* 1. **Decision Tree Classifier**

Decision tree classifier is a machine learning algorithm that learns from the training data, classified them in the same way as in the training data, and can be used to predict target value of test data. The decision tree classifier is also called decision tree induction or classification tree. It is an effective exploratory (or unsupervised) machine learning technique for supervised classification problems to provide a model with more accuracy based on fewer possible variables. Decision tree classifier (DTC) is one part of a family of supervised classification techniques. The basic idea behind the DTC classifier is to reduce the dimensionality by representing data with a set of decision trees, then forecasting target values for future data points in the same way as we do for training examples. By doing this, DTC does not require knowledge about all possible variables, thus provides more accurate results than other supervised classification algorithms.

The process behind the DTC is that it learns the relationships between the variables and data points to build decision trees from a set of examples. To build a decision tree, one must determine two things: which attributes or features to use, and which attribute splits to perform. The choice for these decisions heavily depends on which problem we are trying to solve. When building a DTC, we learn from training data with labeled target values on what is the best way for each possible value of target variable. For example, we can have attributes such as gender and age of our users and data points corresponding to each user. The DTC will then predict the gender for any unknown new users in a training set. Similarly to other machine learning algorithms, DTC iteratively learns from the training data and makes predictions on test data points. One of many advantages of DTC over other machine learning algorithms is that it builds decision trees based on an internal knowledge representation rather than a predefined form that leads to overfitting.

The advantage of decision tree classifier over linear regression models is that it can be used for problems with a multiplicative outcome and many input variables. Decision trees take into consideration interactions between variables and the tree grows towards a classification decision. This design of decision trees allows prediction of an event’s probability by factor in the type of event.

In python decision tree classifier can be done using the below steps:

1. Create a split point for the tree. This is a decision node where the input of an example is evaluated to separate it into one of two branches: one path will contain examples that have this value and the other path will have ones that don't have this value.

2. Create branches from decision nodes, each with its own linear function (in python we can use lambda).

The library used for creating a decision tree in python includes the scikit-learn library. Decisiontree.jl package is the julia package aimed at generating decision trees for classification and regression. The package implements creation of decision trees with both categorical and numerical data types, as well as prediction without a tree using bootstrap aggregating or by sampling the underlying distribution of training points. Decisiontree.jl package was developed at the Technion by Aaron Kome, who was advised by Prof. Dan Sommer and Prof. Uri Shiran. The decision tree algorithm used in the package is a combination of the C4.5 decision tree algorithm and the check-splitting decision tree algorithm.

The decision tree was used to predict the gear type of the car.

* 1. **Random Forest Classifier**

Random forest classifier is a classification model. Normally, it is used for exploring the data and identifying different categories that exist in the data set. The Random forest classifier is a probabilistic classification model that is generally used to predict the outcome of a binary classification based on a set of objects (X) with known classes. The observations are randomly sampled from the training data set and then an out-of-sample prediction is made for an instance not present in the original data set. This entire process is repeated hundreds of times, and finally, the decision value for each sample is computed as a weighted average of this prediction from all samples in order to determine which class it belongs to. The Random forest classifier is an ensemble approach to the less accurate classifiers that make up a training set. In addition to being a more accurate predictor of class membership, the Random forest classifier can be thought of as combining individual decision trees until the error between the prediction and actual outcome reaches the desired threshold. The output is an ensemble where each sample represents an individual decision tree.

The difference between a decision tree and a random forest is that a decision tree builds a single tree that is used to make predictions for the input whereas a random forest builds many small trees (called decision trees) and votes for the class. Finally, a trained random forest model is stored in an object called a forest. In other words, the forest stores information about each individual tree and how they are combined to make prediction. The main advantage of Random Forests over other modeling techniques is that it performs very well on both small and large data sets when the number of variables exceeds the number of observations.

The advantage of random forest classifier is that it is a very flexible and robust classifier that is ideal for both classification and regression problems. Generally, this classifier is not suitable for small data sets because the decision trees tend to become too large. Also, forest ensemble learning can be used to build models that are more accurate than a single classifier.

Scikit-learn is a widely-used machine learning library for Python. It includes a RandomForestClassifier class for creating and fitting random forest classifiers. There are several packages in Julia that can be used to create random forest classification models. Some popular options include:

DecisionTree.jl: DecisionTree.jl is a package for creating decision trees in Julia. It includes functions for creating random forests by training multiple decision trees on different subsets of the data.

RandomForest.jl: RandomForest.jl is a package for creating random forests in Julia. It includes functions for fitting and predicting with random forests, as well as methods for tuning the model's hyperparameters.

The random forest classification was used to predict the gear as well.

* 1. **Model Training**

Model training is an essential part of machine learning. After the raw data is transformed into a more meaningful format, Model training is done where the model itself is computed. This includes both supervised and unsupervised learning stages. The exact preprocessing and algorithm used will depend on the type of model being trained. For example, linear models like logistic regression or support vector machines require very little pre-processing. On the other hand, deep neural networks often require a lot of pre-processing . All the models were trained on the Jupiter notebook and its execution time for running the training process was calculated.

On python the execution time was calculated using the "time" module, specifically the time.time() function. This function returns the current time in seconds since the epoch (the epoch is a predefined point in time, usually the beginning of the year 1970). To measure the execution time of a specific block of code, you can take the difference between the time before and after the code block is executed.

In Julia the execution time is calculated using the @time macro or the tic() and toc() functions. The @time macro can be used to time a single expression, while tic() and toc() can be used to time a block of code. Additionally, the BenchmarkTools.jl package can be use for more advanced timing and benchmarking.

The execution time and the accuracy of Linear Regression model, Logistic regression model, Decision Tree classifier and Random Forest classifier models were calculated and noted in Julia and Python and both the languages were compared based on the results obtained.

1. **Model Evaluation**
   1. **Overview**

A machine learning model evaluation is the process of assessing the performance of a model on a specific task using a set of metrics and data. This can include metrics such as accuracy, precision, recall, and F1 score, and may use techniques such as cross-validation and holdout testing. The process of evaluating a machine learning model is similar to that of evaluating any mathematical model, except that the "data" used by the model plays a more central role. In other words, to evaluate a model, one needs to first create training data for it, use the training data to build mathematical functions through which the model predicts values for new values of the same variables, and then test these predictions on new (unseen) data. These predictions are then used as metrics to evaluate how well the model performed on these unseen data points.

Machine learning model evaluation is important as it helps to determine the model's ability to generalize to new, unseen data. This is crucial as a model that performs well on the training data may not perform well on new data. It helps to identify overfitting and underfitting. Overfitting occurs when a model is too complex and has learned the noise in the data. Underfitting occurs when a model is too simple and has not learned the underlying trend of the data. It helps to compare different models and select the best one for a given task. It helps to identify areas for improvement. By evaluating a model, one can identify the strengths and weaknesses of the model and make adjustments to improve performance. It helps to ensure that the model is ready for deployment. A model should be evaluated and deemed suitable before it is used in production.

* 1. **Evaluation Matrix**

There are several evaluation metrics that can be used to assess the performance of a machine learning model. Some common evaluation metrics include-Accuracy: This measures the proportion of correct predictions made by the model. Precision: This measures the proportion of true positive predictions among all positive predictions made by the model. Recall: This measures the proportion of true positive predictions among all actual positive instances. F1 Score: This is the harmonic mean of precision and recall. It balances both metrics and is commonly used when there is an imbalance in the dataset.

Confusion Matrix: This is a table that is used to define the performance of a classification algorithm. It is a summary of prediction results on a classification problem. Root Mean Squared Error (RMSE): This measures the difference between the predicted values and the actual values. It is commonly used for regression problems.

These are just a few examples of the many evaluation metrics that can be used to assess the performance of a machine learning model. The choice of evaluation metric will depend on the specific task and the type of model being used. As the models involves classification and regression models the F1 score and accuracy was used to evaluate the applied models.

* 1. **Evaluation**

In terms of speed of loading the data it was clear that the Julia was waster than Python.

For tesla dataset

Graphical user interface, text, application, Word, website

Description automatically generated

Figure 10 Julia load tesla

Text

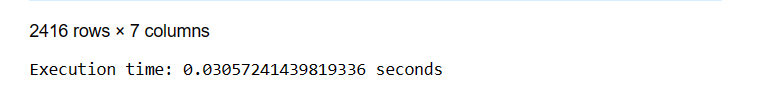
Description automatically generated

Figure 11 Python load tesla

A picture containing graphical user interface

Description automatically generated

Figure 12 Julia load time cars

Text

Description automatically generated

Figure 13 Python load time cars

Even for a small dataset like the tesla we can see Julia outperforms python in terms of load speed. Julia is 2-4 times faster than Python in loading data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Linear regression | Python | | Julia | |
| Time | Accuracy | Time | Acuuracy |
| 0.737 | 96.7 | 0.274 | 99.9 |

Table 3: Tesla output

We can see that there is a big difference in the execution time between the two languages for a small dataset.

The outputs obtained for the German cars dataset is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Python | | Julia | |
| Execution Time(seconds) | Accuracy | Execution Time(seconds) | Accuracy |
| Linear Regression | 0.037 | 42% | 0.0161 | 63% |
| Logistic Regression | 6.44 | 81% | 0.911 | 82% |
| Decision Tree Classifier | 0.5097 | 83% | 0.265 | 79% |
| Random Forest Classifier | 2.948 | 83% | 0.7977 | 83% |

Table 4: German cars outputs

As per the outputs obtained it was seen that the accuracy were almost the same for all the models but there were some notable difference in the execution time.In case of linear regression there is almost no difference in the execution time but in the case of the logistic regression the accuracy is almost same but in terms of execution time of the model Julia is observed to be over 6 times faster. In terms of Decision tree and Random Forest classification the accuracy of the models is same but we can see that the Julia model was 2-3 times faster in execution time.

There could be several reasons for the faster execution time of Julia models compared to other languages. Julia is designed to be fast and efficient, with a low-level, high-performance language that is easy to use. Julia's JIT (Just-In-Time) compilation allows for efficient code execution, which can result in faster performance. The codes in Julia was more readable and closer to the mathematical notation, this makes it more suited for data science and scientific computing. Julia also provided a lot of built-in functionality for scientific computing and data science, which made it easier to implement machine learning models and perform other data analysis tasks. Julia has a number of libraries and packages that are optimized for specific tasks, such as machine learning and data science, which made it easier to implement complex algorithms and models in Julia than in python.

1. **Conclusion**
   1. **Overview**

The aim of the project was to see if Julia has some advantages over python in Data Science. We have taken note of the time each language takes to perform some common tasks and make a comparison. The first set of tests we ran was the loading the dataset. In this test, Python took about 2x-4x longer than Julia. This was not really surprising since it is a programming language that is more textual than mathematical. In the second set of tests the time taken for doing different machine learning algorithms were noted. In this test I have tested the time taken using both languages for different machine learning algorithms. The implementation of all the tests were made in the same manner. The machine used is an i7 9th generation laptop with 16GB RAM, and 2.4 GHz clock speed processor. The OS used is Windows 10 professional 64-bit operating system and the most updated version of python and julia were used.

In these tests we implemented the following machine learning algorithms: Linear Regression, Logistic Regression, Decision tree classification and Random forest classification.

* 1. **Findings and Recommendations**

As two datasets were used in the project one having only 2400 rows and the other having over 46000 rows. It was found that for the smaller dataset the time taken to load the dataset was almost same with a slight edge to Julia while for the larger dataset there was significant difference in the load time

In Linear regression the Julia implementation took almost the same time as the python implementation. In Logistic Regression and Random forest classification, Julia was way faster than python models in the performance and the implementation. The most surprising thing to me was that in Decision tree classification, Julia outperformed python while doing it using less time. In conclusion I feel that Julia has it's own advantages over python for doing Data Science. Both the languages are good and have their own advantages. The good thing about python is that it can do pretty much anything and since it is the most popular language in Data Science, there are a lot of resources available for it. On the other hand, Julia being a very new language is trying to make it self more popular with every release. Even in terms of the obtained accuracy we can see that Julia had a slight edge over python.

The fact that Julia has dedicated packages for these machine learning algorithms is a big plus for Julia. Most machine learning algorithms are hard coded in python so it can be hard to update the libraries with new or better models. It is even harder for building new models because you have to start from the scratch. The other good thing about Julia is that it's interfaces for Machine learning algorithms are very good and efficient. Using Julia as a first language can make Python no longer required when doing Machine learning processes of any complexity.

Julia was especially faster than python in the tree models. This shows that Julia is good at factorizing problems into smaller subproblems and solving them. I think that this comes from it's mathematical base. The downside of Julia is that the language is changing all the time, so it can be a little hard to get all the libraries installed properly with every new update, which will help you when you are trying to learn any new libraries or packages.

* 1. **Areas of Future Work**

Despite the fact that the project was successfully finished and the desired outcome was attained, there is always room for improvement and more effort can be done in the future to enhance the final product and the procedure used.

Currently the project only compares Julia with python in terms of speed and ease of use. The project can be made better by comparing Julia with some more top programming languages like R. The machine learning models can be implemented in the other languages and the outcome of this can be used to analyse the advantages of Julia.

The models trained in this project is also limited to for and all of them are supervised learning models. Some Unsupervised learning models can also be made using both the languages and this can be done in future.

* 1. **Personal Evaluation**

A project's successful completion does not always imply that every step of the process went without any problems. There were times during the project when the anticipated results were not being attained, but these problems were eventually handled by careful analysis and ongoing learning as the project progressed through its many stages. As a Data Science student, the knowledge obtained during project implementation also gave me a solid foundation to expand on as I learned more about the practical application and the elements to take into account while designing a Machine Learning project in Julia.

With a few minor changes like using additional dataset, the project's initial objectives were successfully attained. The comparison of Julia with python was also considered during the initial stages of the project but was later removed due to lack of time.

The evaluations at predetermined intervals ensured that the project was proceeding according to the previously outlined timeframe and that there were no deviations from the ultimate objective. This was helpful in achieving the intended project end objective and is necessary for every project to be completed successfully.

* 1. **Future of Julia**

It is difficult to predict exactly where Julia will be in the next 10 years, as it depends on various factors such as the development of the language and its community, as well as the evolution of the field of programming and technology as a whole. However, based on the current trajectory of the language and its growing popularity in scientific computing and data science, it is likely that Julia will continue to be a widely-used and respected programming language in these fields. Julia's ease of use and high performance make it well-suited for a wide range of applications, and it could continue to gain popularity among researchers, engineers, and data scientists. Additionally, Julia could continue to grow and expand its ecosystem, increasing the availability of packages and libraries for various tasks which intern will make it the best programming language for Data Science.

* 1. **Julia vs Python**

Both Julia and Python are popular programming languages for data science, and each has its own strengths and weaknesses. Python is a well-established language with a large and active community, which has led to the development of a vast ecosystem of libraries and frameworks for data science, machine learning, and visualization. Python is also relatively easy to learn, with a simple and intuitive syntax, making it a popular choice for beginners and non-experts.

Julia, on the other hand, is a relatively new language, specifically designed for high-performance numerical computing and data science. Julia is known for its speed, which is comparable to that of low-level languages like C and Fortran, but with the ease of use and expressiveness of high-level languages like Python. Julia also has a growing ecosystem of packages and libraries, which makes it easy to perform a wide range of tasks, from data visualization to machine learning.

So, depending on what you are trying to do, either Julia or Python may be a better choice. If you are looking for a language that is easy to learn, has a large community, and a vast ecosystem of libraries and frameworks, Python is probably the best choice. If you are looking for a language that is fast, and can handle large data sets quickly, then Julia is probably the better choice. Additionally, Julia's growing popularity in scientific computing, optimization, and machine learning, Julia is a great fit for data science tasks that require high performance and low-level control.

# References

* A. Edelman, (2015) "*Julia: A fresh approach to parallel programming,"* IEEE International Parallel and Distributed Processing Symposium.

Available online at <https://ieeexplore.ieee.org/document/7161539>

* Chantal D. Larose, Daniel T. Larose, (2019) “*Data Science Using Python and R*”, John Wiley & Sons, Inc.
* Edelman, (2015) "*Julia Introduction*" IEEE International Parallel and Distributed Processing Symposium Workshop, 2015, pp. 1271-1271, doi: 10.1109/IPDPSW.2015.18.

Available online at <https://ieeexplore.ieee.org/document/7284457>

* Gao, Kaifeng & Mei, Gang & Piccialli, Francesco & Cuomo, Salvatore & Tu, Jingzhi & Huo, Zenan. (2020). “*Julia Language in Machine Learning: Algorithms, Applications, and Open Issues*”. Available online at <https://www.researchgate.net/publication/341359771_Julia_Language_in_Machine_Learning_Algorithms_Applications_and_Open_Issues>
* Goutte, C. & Gaussier, E., 2005. A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. pp. 345-359.
* Jake VanderPlas(2016), “*Python Data Science Handbook: Essential Tools for Working with Data*”, O'Reilly Media, Inc.
* Jose Storopoli, Rik Huijzer, Lazaro Alonso(2021), “*Julia Data Science*”

Available online at <https://juliadatascience.io/> (Accessed on 1/6/2022)

* Heidelberg (2005). Applied Linear Regression. In: Classical Methods of Statistics.. Springer, Berlin, <https://doi.org/10.1007/3-540-29288-8_3>. pp 113–160
* Nazarathy, Y., Klok, H. (2021). “*Linear Regression and Extensions. In: Statistics with Julia*”. Springer Series in the Data Sciences.
* Raschka, S., 2018. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. 11.
* R. Gonzalez, S. Ahmed and M. Alamaniotis, "Deep Neural Network Based Methodology for Very-Short-Term Residential Load Forecasting," 2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA), Corfu, Greece, 2022, pp. 1-6, doi: 10.1109/IISA56318.2022.9904338.
* R. Karim, M. K. Alam and M. R. Hossain, (2021) "*Stock Market Analysis Using Linear Regression and Decision Tree Regression*," 1st International Conference on Emerging Smart Technologies and Applications (eSmarTA), 2021, pp. 1-6, doi: 10.1109/eSmarTA52612.2021.9515762.

Available online at <https://ieeexplore.ieee.org/document/9515762>

* Sebastian Raschka,(2015), “Python Machine Learning“ Packt Publishing Ltd
* Voulgaris, Z. 2020, “*Julia for Machine Learning*” Technics Publications.
* Witold Pedrycz, Shyi-Ming Chen. 2017. Data Science and Big Data: An Environment of Computational Intelligence. https://doi.org/10.1007/978-3-319-53474-9.

**Bibliography**

* Boris Mirkin (2013) ‘*Clustering A Data Recovery Approach Second Edition’*, Chapman & Hall.
* Charu C Aggarwal (2018) ‘*Neural Networks and Deep Learning: Textbook’*, Springer International Publishing.
* Géron, , A., 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* 2nd ed. Canada: O'Reilly Media
* Jim Frost,MS (2020) ‘*Regression Analysis: An Intuitive Guide for Using and Interpreting Linear* Models’, Statistics By Jim Publishing.
* . McKinney, W., 2011. pandas: a foundational Python library for data analysis and statistics. *Python for high performance and scientific computing,* pp. 1-9.
* Pal, M., 2005. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing,* 1, 26(1), pp. 217-222.
* S. V. Mahadevkar et al, "*A Review on Machine Learning Styles in Computer Vision—Techniques and Future Directions*," in IEEE Access, vol. 10, pp. 107293-107329, 2022, doi: 10.1109/ACCESS.2022.3209825.
* Ward, Matthew (2015), “Interactive data visualization : foundations, techniques, and applications”, Boca Raton, Florida ; London, England ; New York, New York : CRC Press
* Xiaojin Zhu, Andrew B. Goldberg. (2009) ‘*Introduction to Semi-Supervised Learning*’ (9-19), Morgan & Claypool Publishers.

# Bibliography

* Chen, T. & Guestrin, C., 2016. XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 8.pp. 785-794.
* Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K., 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 10.
* Géron, , A., 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* 2nd ed. Canada: O'Reilly Media.
* Kanakaraj, M. & Guddeti, R., 2015. *NLP based sentiment analysis on Twitter data using ensemble classifiers.* Chennai, India, IEEE.
* Lau, J. H. & Baldwin, T., 2016. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. 7.
* Marres, N. & Weltevrede, E., 2013. SCRAPING THE SOCIAL?. *Journal of Cultural Economy,* 8, 6(3), pp. 313-335.
* McKinney, W., 2011. pandas: a foundational Python library for data analysis and statistics. *Python for high performance and scientific computing,* pp. 1-9.
* Moreno, M., Gower, A., Brittain, H. & Vaillancourt, T., 2019. Applying natural language processing to evaluate news media coverage of bullying and cyberbullying. *Prevention Science,* 20(8), pp. 1274-1283.
* Muneer, A. & Fati, S., 2020. A comparative analysis of machine learning techniques for cyberbullying detection on twitter. *Future Internet,* 12(11), p. 187.
* Nelli, F., 2015. Data Visualization with matplotlib. In: *Python data analytics.* Berkeley, CA: Apress, pp. 167-235.
* Pal, M., 2005. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing,* 1, 26(1), pp. 217-222.
* Relan, K., 2019. Deploying Flask Applications. *Building REST APIs with Flask,* pp. 159-182.
* Sarlan, A., Nadam, C. & Basri, S., 2014. Twitter sentiment analysis. *Proceedings of the 6th International Conference on Information Technology and Multimedia,* 11.pp. 212-216.
* Tenney, I., Das, D. & Pavlick, E., 2019. BERT Rediscovers the Classical NLP Pipeline. 5.
* Wang, J., Fu, K. & Lu, C.-T., 2020. SOSNet: A Graph Convolutional Network Approach to Fine-Grained Cyberbullying Detection. *2020 IEEE International Conference on Big Data (Big Data),* 12.pp. 1699-1708.
* Yao, M., Chelmis, C. & Zois, D., 2109. *Cyberbullying ends here: Towards robust detection of cyberbullying in social media.* Online, The World Wide Web Conference.
* Zhang, L., Wang, S. & Liu, B., 2018. Deep learning for sentiment analysis: A survey. *WIREs Data Mining and Knowledge Discovery,* 7.8(4).