Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2021 р

# Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**МОДЕЛЬ КОМП'ЮТЕРНОЇ СИСТЕМИ ЕЛЕКТРОПОСТАЧАННЯ ЕЛЕКТРОМОБІЛЯ»**

Захищено на засіданні
Атестаційної комісії № 38
протокол № __ від __.12.2021 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **МІНУХІН**
**Сергій Володимирович**

**Виконав:**
студент 6 курсу, групи КІ– 61
Галузь знань: 12 – Інформаційні технології
за спеціальністю 123 – Комп'ютерна інженерія.
**БЕНХАДДУ Мохаммед Амін** _____

**Керівник:**
к.ф.-м.н., с.н.с., доцент кафедри електроніки і управляючих систем
**ХРУСЛОВ Максим Михайлович**

**Рецензент:**
к.т.н., доцент, теоретичної та прикладної системотехніки
**БУЛАВІН Дмитро Олексійович** __

Харків – 2021

# АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи магістра складається зі вступу, трьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 78 сторінки, із яких 64 сторінок основної частини. Робота містить 19 рисунків та 51 найменуваня списку використаних джерел на 6 сторінках.

***Мета дослідження:*** На основі аналізу даних комп'ютерної моделі системи живлення електромобіля дати рекомендації щодо зниження енергоспоживання.

***Об'єкт дослідження:*** Комп'ютерна модель системи енергоспоживання електромобіля. Об'єкт дослідження новий, оскільки пов'язаний з наданням рекомендацій щодо поведінки водіїв.

***Предмет розробки:*** Нейронні мережі як спосіб побудови інтелектуальних програм і машин, здатних творчо аналізувати та вирішувати проблеми. Штучний інтелект як спосіб створення статистичних моделей та аналізу споживання акумулятора автомобіля.

Електромобілі стають все більш популярними на сучасних автомобільних ринках. Багато сервісів для електромобілів, таких як інтелектуальні системи зарядки акумулятора та мобільні додатки, розроблено для моніторингу інформації про енергію акумулятора (споживання акумулятора). Однак нехтують тим, що дані споживання енергії електромобілем можуть призвести до проблем, пов'язаних із конфіденційністю водія, наприклад, таких як розкриття шляху руху. Технології батареї та передові системи керування батареєю є одними з найпопулярніших досліджень в автомобільній галузі в результаті безпрецедентного поштовху до електромобілів. Акумулятори, що використовуються в електромобілях, досліджують різні методи прямого вимірювання на основі штучного інтелекту. Дефіцит енергії сьогодні є серйозним викликом, він вплинув на розвиток нових джерел енергії. Штучний інтелект, такий як навчання та аналіз, широко використовується для різних переваг. Його успішно застосовують для

прогнозування матеріалів, особливо матеріалів для зберігання енергії. У цьому проекті ми представляємо огляд сучасного стану штучного інтелекту в матеріалах для зберігання енергії через конденсатори та літій-іонні батареї.

У цій дипломній роботі зроблено аналіз даних (споживання енергії автомобілем), в залежності від того як водій керує автомобілем шляхом створення моделі. На основі результатів, отриманих від комп'ютерної моделі за допомогою нейронної мережі, ми побачимо, як працює кінцева комп'ютерна модель.

***Ключові слова:*** *комп'ютерна модель, штучний інтелект, машинне навчання, аналіз даних, апроксимація, програмний продукт*.

## Abstract

The explanatory note to the master's qualification work consists of an introduction, three chapters, conclusions, a list of sources used and four appendices. The total volume of the work is 78 pages, of which 64 pages are the main part. The work contains 19 figures and 51 names of the list of used sources on 6 pages.

***The aim of the study****:* based on the analysis of the data of the computer model of the electric vehicle power supply system, give recommendations for reducing energy consumption.

***Object of research:*** Computer model of the power consumption system of an electric car. The object of study is new, as it is associated with the provision of recommendations for driver behavior.

***Subject of development:*** Neural networks as a way to build intelligent programs and machines that can creatively analyze and solve problems. Artificial intelligence as a way to create statistical models and analyze vehicle battery consumption.

Electric cars are becoming increasingly popular in today's automotive markets. Many services for electric vehicles, such as intelligent battery charging systems and mobile applications, are designed to monitor information about battery energy (battery consumption). However, it is overlooked that power consumption data by electric vehicles can lead to driver privacy issues, such as traffic jams. Battery technology and advanced battery management systems are among the most popular studies in the automotive industry as a result of the unprecedented push for electric vehicles. Batteries used in electric vehicles are exploring various methods of direct measurement based on artificial intelligence. Energy shortage today is a serious challenge, it has affected the development of new energy sources. Artificial intelligence, such as learning and analysis, is widely used for various benefits. It is successfully used for forecasting materials, especially materials for energy storage. In this project, we present an overview of the current state of artificial intelligence in materials for energy storage through capacitors and lithium-ion batteries.

In this thesis, an analysis of data (energy consumption of the car), depending on how the driver drives the car by creating a model. Based on the results obtained from the computer model using the neural network, we will see how the final computer model works.

***Keywords:*** *computer model, artificial intelligence, machine learning, data analysis, approximation, software product.*

**Plan**

# Introduction

There is no doubt that neural network are developed these years, used in all fields. In this research I will emphasize on consumption of electrical energy cars (New Technology) by the use of model computer system and neural networks (use of Artificial intelligence) [1].

The problem related to reduce the consumption of electrical energy cars (New Technology) [2] by the use of model computer system and neural networks. The question is: What are the models of computer system and programs (neural networks) that are related to reduce consumption of electrical energy cars and what's the new model that can help to give some recommendations to drivers [3, 4]?

For the analysis of this problem, I will start by summary about different models that were used for computer systems in electric cars, then definition AI/Machine Learning and types, different algorithms related to this one[6] [7] [9]. After I will make an analyze of data (consumption energy cars/ behavior driver's aggressive or calm) how he drives the car and the consumption of energy related to this behavior (by making model). Based on results obtained from data/model, we will see the neural network how it works (test). [10] [11]

A neural network is a simplified model of the way the human brain processes information. It works by simulating a large number of interconnected processing units that ressemble abstract versions of neurons. The means of doing machine learning, in which a computer learns to perform some task by analyzing training examples. Usually, the examples have been hand-labeled in advance. An object recognition system, for instance, might be fed thousands of labeled images of cars, houses, coffee cups, and so on, and it would find visual patterns in the images that consistently correlate with particular labels [12].

A deep learning method is used to construct an electric vehicle travel energy consumption prediction model based on traffic conditions and vehicle states in this paper, which aims to make a robust and accurate prediction of the vehicle energy consumption during the journey. In the proposed model, we utilize traffic information

and vehicle status information, which are two different types of information. Since traffic conditions of the road network are continuously changing during the trip, traffic conditions during the trip as a sequential input were considered. And the recurrent neural network is used, which is suitable for dealing with sequential input, to extract the features of the traffic status of the whole trip. Then, the deep neural network was applied to fuse the vehicle state features with the traffic state features and output the prediction results of energy consumption of the trip. For the traffic feature extraction, the traffic information of the road network is used as the input of the model. The traffic conditions of the road network can help provide information on the traffic state changes during the driving of the vehicle in the task of predicting travel energy consumption. More specifically, the congestion, the morning peak hour, and the evening peak hour are major scenarios that would be tested in the paper. Besides, an attention mechanism was employed to help the model extract the information that has a great influence on the current task from the traffic state of road network. Therefore, we extract the feature of these two kinds of information separately and fuse the extracted feature information in the hidden space. Finally, we use the fused feature to predict the energy consumption of the trip. Furthermore, we show how attention mechanism can help the energy consumption forecasting model extract traffic state information and how its attention degree changes in different sections of the road network during the forecasting process [13].

Modeled loosely on the human brain, a neural net consists of thousands or even millions of simple processing nodes that are densely interconnected. Most of today's neural nets are organized into layers of nodes, and they're feed-forward, meaning that data moves through them in only one direction. An individual node might be connected to several nodes in the layer beneath it, from which it receives data, and several nodes in the layer above it, to which it sends data. To each of its incoming connections, a node will assign a number known as a weight. When the network is active, the node receives a different data item a different number over each of its connections and multiplies it by the associated weight. It then adds the resulting products together, yielding a single number [14]. If that number is below a threshold

value, the node passes no data to the next layer. If the number exceeds the threshold value, the node “fires,” which in today’s neural nets generally means sending the number  the sum of the weighted inputs along all its outgoing connections.When a neural net is being trained, all of its weights and thresholds are initially set to random values. Training data is fed to the bottom layer the input layer and it passes through the succeeding layers, getting multiplied and added together in complex ways, until it finally arrives, radically transformed, at the output layer. During training, the weights and thresholds are continually adjusted until training data with the same labels consistently yield similar outputs.

Deep learning networks are distinguished from the more commonplace single-hidden-layer neural networks by their depth; that is, the number of node layers through which data must pass in a multistep process of pattern recognition.

Earlier versions of neural networks such as the first perceptrons were shallow, composed of one input and one output layer, and at most one hidden layer in between. More than three layers (including input and output) qualifies as “deep” learning. So deep is not just a buzzword to make algorithms seem like they read Sartre and listen to bands you haven’t heard of yet. It is a strictly defined term that means more than one hidden layer.

In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer’s output. The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer [15].

This is known as feature hierarchy, and it is a hierarchy of increasing complexity and abstraction. It makes deep-learning networks capable of handling very large, high-dimensional data sets with billions of parameters that pass through nonlinear functions [16].

Above all, these neural nets are capable of discovering latent structures within unlabeled**,** unstructured data, which is the vast majority of data in the world. Another word for unstructured data is raw media; i.e. pictures, texts, video and audio recordings. Therefore, one of the problems deep learning solves best is in processing

and clustering the world's raw, unlabeled media, discerning similarities and anomalies in data that no human has organized in a relational database or ever put a name to [16].

For example, deep learning can take a million images, and cluster them according to their similarities: cats in one corner, ice breakers in another, and in a third all the photos of your grandmother. This is the basis of so-called smart photo albums.

Now apply that same idea to other data types: Deep learning might cluster raw text such as emails or news articles. Emails full of angry complaints might cluster in one corner of the vector space, while satisfied customers, or spambot messages, might cluster in others. This is the basis of various messaging filters, and can be used in customer-relationship management (CRM). The same applies to voice messages [16].

With time series, data might cluster around normal/healthy behavior and anomalous/dangerous behavior. If the time series data is being generated by a smart phone, it will provide insight into users' health and habits; if it is being generated by an autopart, it might be used to prevent catastrophic breakdowns [16].

Deep-learning networks perform automatic feature extraction without human intervention, unlike most traditional machine-learning algorithms. Given that feature extraction is a task that can take teams of data scientists years to accomplish, deep learning is a way to circumvent the chokepoint of limited experts. It augments the powers of small data science teams, which by their nature do not scale [16].

When training on unlabeled data, each node layer in a deep network learns features automatically by repeatedly trying to reconstruct the input from which it draws its samples, attempting to minimize the difference between the network's guesses and the probability distribution of the input data itself. Restricted Boltzmann machines, for examples, create so-called reconstructions in this manner [16].

In the process, these neural networks learn to recognize correlations between certain relevant features and optimal results – they draw connections between feature signals and what those features represent, whether it be a full reconstruction, or with labeled data.

A deep-learning network trained on labeled data can then be applied to unstructured data, giving it access to much more input than machine-learning nets.

This is a recipe for higher performance: the more data a net can train on, the more accurate it is likely to be. (Bad algorithms trained on lots of data can outperform good algorithms trained on very little.) Deep learning's ability to process and learn from huge quantities of unlabeled data give it a distinct advantage over previous algorithms.

Deep-learning networks end in an output layer: a logistic, or softmax, classifier that assigns a likelihood to a particular outcome or label. We call that predictive, but it is predictive in a broad sense. Given raw data in the form of an image, a deep-learning network may decide, for example, that the input data is 90 percent likely to represent a person[16].

The processing units are arranged in layers [5] [8]. There are typically three parts in a neural network: an input layer, with units representing the input fields; one or more hidden layers; and an output layer, with a unit or units representing the target field. The units are connected with varying connection strengths. Input data are presented to the first layer, and values are propagated from each neuron to every neuron in the next layer. Eventually, a result is delivered from the output layer.

The network learns by examining individual records, generating a prediction for each record, and making adjustments to the weights whenever it makes an incorrect prediction. This process is repeated many times, and the network continues to improve its predictions until one or more of the stopping criteria have been met.

Initially, all weights are random, and the answers that come out of the net are probably nonsensical. The network learns through training. Examples for which the output is known are repeatedly presented to the network, and the answers it gives are compared to the known outcomes. Information from this comparison is passed back through the network, gradually changing the weights. As training progresses, the network becomes increasingly accurate in replicating the known outcomes. Once trained, the network can be applied to future cases where the outcome is unknown [17] [13] [14].

Neural networks grown to have a significant impact on the world with large amounts of data being generated by different applications and sources, machine learning systems can learn from the test data and perform intelligent tasks.

Artificial Intelligence is the field of computer science that deals with imparting the decisive ability and thinking the ability to machines. Artificial Intelligence is thus a blend of computer science, data analytics, and pure mathematics [12].

Artificial intelligence and machine learning (ML) are the part of computer science that are correlated with each other. These two technologies are the most trending technologies which are used for creating intelligent systems.

Although these are two related technologies and sometimes people use them as a synonym for each other, but still both are the two different terms in various cases [15].

# Chapter I: Artificial Intelligence and Machine Learning

The term artificial intelligence was first used in 1956, at a computer science conference in Dartmouth. AI described an attempt to model how the human brain works and, based on this knowledge, create more advanced computers. The scientists expected that to understand how the human mind works and digitalize it shouldn't take too long. After all, the conference collected some of the brightest minds of that time for an intensive 2 months brainstorming session (see Fig.1.)

*Fig.1. Description of AI*

Over the past few years AI has exploded, and especially since 2015. Much of that has to do with the wide availability of GPUs that make parallel processing ever faster, cheaper, and more powerful. It also has to do with the simultaneous one-two punch of practically infinite storage and a flood of data of every stripe (that whole Big Data movement) – images, text, transactions, mapping data, you name it.

Artificial intelligence is based on the principle that human intelligence can be defined in a way that a machine can easily mimic it and execute tasks, from the most simple to those that are even more complex. The goals of artificial intelligence include learning, reasoning, and perception.

As technology advances, previous benchmarks that defined artificial intelligence become outdated. For example, machines that calculate basic functions or recognize text through optimal character recognition are no longer considered to embody artificial intelligence, since this function is now taken for granted as an inherent computer function.

AI is continuously evolving to benefit many different industries. Machines are wired using a cross-disciplinary approach based in mathematics, computer science, linguistics, psychology, and more [21].

Artificial intelligence is a field of computer science which makes a computer system that can mimic human intelligence. It is comprised of two words "Artificial" and "intelligence", which means a human-made thinking power.

It is a science like mathematics or biology, it studies ways to build intelligent programs and machines that can creatively solve problems, which has always been considered a human prerogative.

The Artificial intelligence system does not require to be pre-programmed, instead of that, they use such algorithms which can work with their own intelligence. It involves machine learning algorithms such as Reinforcement learning algorithm and deep learning neural networks. AI is being used in multiple places such as Siri, Googles AlphaGo, AI in Chess playing, etc.

### 1.1 Classification of AI:

Based on capabilities, AI can be classified into three types:

- **Weak AI**: is the most limited and the most common of the three types of AI. It's also known as narrow AI or artificial narrow intelligence (ANI).

Weak AI refers to any AI tool that focuses on doing one task really well. That is, it has a narrow scope in terms of what it can do. The idea behind weak AI isn't to mimic or replicate human intelligence. Rather, it's to simulate human behavior. So, it's nowhere near matching human intelligence, and it isn't trying to.

A common misconception about weak AI is that it’s barely intelligent at all more like artificial stupidity than AI. But even the smartest seeming AI of today are only weak AI.

In reality, then, narrow or weak AI is more like an intelligent specialist. It’s very intelligent at completing the specific tasks it’s programmed to do.

**- Narrow AI:** machine intelligence comes from the use of natural language processing (NLP) to perform tasks. NLP is evident in chat bots and similar AI technologies. By understanding speech and text in natural language, AI is programmed to interact with humans in a natural, personalized manner.

Narrow AI can either be reactive, or have a limited memory. Reactive AI is incredibly basic; it has no memory or data storage capabilities, emulating the human mind’s ability to respond to different kinds of stimuli without prior experience. Limited memory AI is more advanced, equipped with data storage and learning capabilities that enable machines to use historical data to inform decisions.

Most AI is limited memory AI, where machines use large volumes of data for deep learning. Deep learning enables personalized AI experiences, for example, virtual assistants or search engines that store your data and personalize your future experiences [22].

- **General AI**: Artificial general intelligence (AGI), is the concept of a machine with general intelligence that make human intelligence and behaviors, with the ability to learn and apply its intelligence to solve any problem. AGI can think, understand, and act in a way that is indistinguishable from a human in any given situation. it's the representation of generalized human cognitive abilities in software so that, faced with an unfamiliar task, the AI system could find a solution. An AGI system could perform any task that a human is capable of.

AGI, sometimes referred to as strong AI, involves a system with comprehensive knowledge and cognitive computing capabilities such that its performance is indistinguishable from that of a human, at least in those terms. However, the broad intellectual capacities of AGI would be boosted far beyond

human capacities by its ability to access and process huge amounts of data at incredible speeds.

- **Super AI**: Super AI is AI that surpasses human intelligence and ability. It's also known as artificial super intelligence (ASI) or super intelligence. It's the best at everything math's, science, medicine, hobbies, you name it. Even the brightest human minds cannot come close to the abilities of super AI.

Types of AI, super AI is the one most people mean when they talk about robots taking over the world. Or about AI overthrowing or enslaving humans. (Or most other science fiction AI tropes.)

But rest assured, super AI is purely speculative at this point. That is, it's not likely to exist for an exceedingly long time

Currently, we are working with weak AI and general AI. The future of AI is Strong AI for which will be intelligent than humans.

There are two categories of A.I. tasks:

- Abstract and formal: easy for computers but difficult for humans, e.g. play chess (IBM's Deep Blue 1997). → Knowledge-based approach to artificial intelligence.

- Intuitive for humans but hard to describe formally: e.g. recognizing faces in images or spoken words. → Concept capture and generalization

**Machine learning:**

Machine learning is about extracting knowledge from the data. It can be defined as, a subset of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. In ML, there are different algorithms (e.g. neural networks) that help to solve problems.

In the context of Big Data, Machine Learning is used to keep up or improvising by itself with the ever-growing and ever-changing stream of data and deliver continuously evolving and valuable insights.

Machine learning algorithms define the incoming data and identify patterns involved with it, which are subsequently translated into valuable insights that can be further implemented into the business operation. After that, the algorithms also used to automate certain aspects of the decision-making process [23].

For these purposes, machine learning algorithms are used with a variety of techniques like decision trees or neural networks.

Machine learning enables a computer system to make predictions or take some decisions using historical data without being explicitly programmed. Machine learning uses a massive amount of structured and semi-structured data so that a machine learning model can generate accurate result or give predictions based on that data.

Machine learning is a technology that allows computers to learn directly from examples and experience in the form of data. Traditional approaches to programming rely on hardcoded rules, which set out how to solve a problem, step-by-step. In contrast, machine learning systems are set a task, and given a large amount of data to use as examples of how this task can be achieved or from which to detect patterns. The system then learns how best to achieve the desired output. It can be thought of as narrow AI: machine learning supports intelligent systems, which are able to learn a particular function, given a specific set of data to learn from. AI is the all-encompassing umbrella that covers everything from Good Old Fashion AI (GOFAI) all the way to connectionist architectures like Deep Learning. ML is a sub-field of AI that covers anything that has to do with the study of learning algorithms by training with data.

Machine learning works on algorithms which learn by its own using historical data. It works only for specific domains, such as if we are creating a machine learning model to detect pictures of dogs, it will only give result for dog images, but if we provide a new data like cat image then it will become unresponsive. Machine learning is being used in various places such as for online recommender system, for Google

search algorithms, Email spam filter, Facebook Auto friend tagging suggestion, etc [https://thetaxtalk.com/2021/05/21/artificial-learning-vs-machine/].

## 1.2 Types of Machine Learning

It can be divided into three types:

- **Supervised learning**: supervised learning, the computer is provided with example inputs that are labeled with their desired outputs (see Fig.2.). The purpose of this method is for the algorithm to be able to learn by comparing its actual output with the taught outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data. In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function h(x) (sometimes called the hypothesis). Learning consists of using sophisticated mathematical algorithms to optimize this function so that, given input data x about a certain domain (say, square footage of a house), it will accurately predict some interesting value h(x) say, price for house

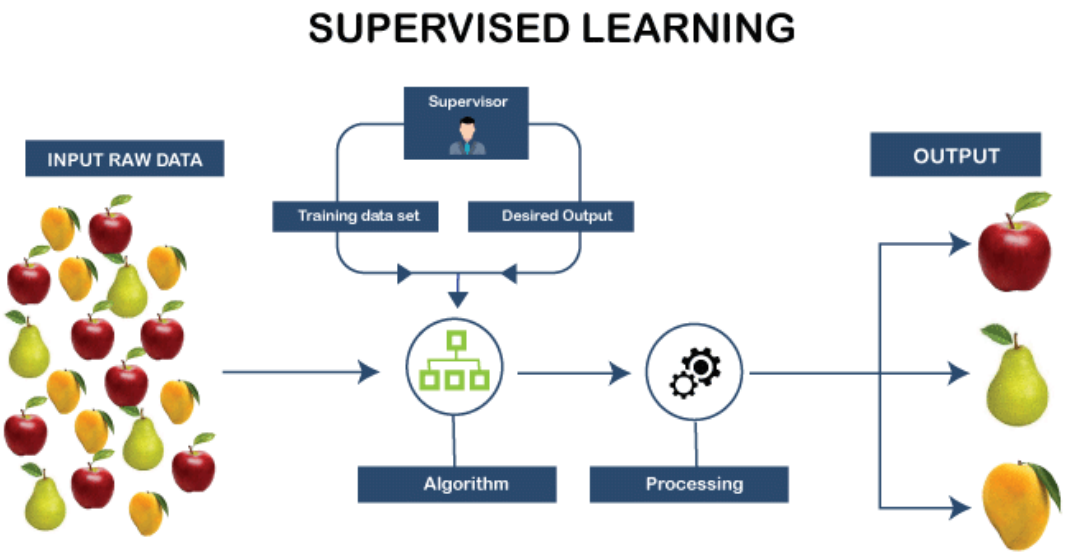



*Fig.2. Process of working for supervised learning*

- **Reinforcement learning**: it's concerned with how software agents should take actions in an environment. Reinforcement Learning is a part of the deep learning

method that helps you to maximize some portion of the cumulative reward. Here are some important terms used in Reinforcement AI [24]:

**Agent**: It is an assumed entity which performs actions in an environment to gain some reward.

**Environment** (e): A scenario that an agent has to face.

**Reward** (R): An immediate return given to an agent when he or she performs specific action or task.

**State** (s): State refers to the current situation returned by the environment.

**Policy** ($\pi$): It is a strategy which applies by the agent to decide the next action based on the current state.

**Value**(V): It is expected long-term return with discount, as compared to the short-term reward.

**Value Function**: It specifies the value of a state that is the total amount of reward. It is an agent which should be expected beginning from that state.

**Model of the environment:** This mimics the behavior of the environment. It helps you to make inferences to be made and also determine how the environment will behave.

**Model based methods:** It is a method for solving reinforcement learning problems which use model-based methods.

**Q value or action value (Q):** Q value is quite similar to value. The only difference between the two is that it takes an additional parameter as a current action [24].

There is two important learning models in reinforcement learning Markov Decision Process and Q learning

- **Unsupervised learning:** data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable (see Fig.3.). The goal of unsupervised learning may be as straight forward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover

the representations that are needed to classify raw data. Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases. Without being told a correct answer, unsupervised learning methods can look at complex data that is more expensive and seemingly unrelated in order to organize it in potentially meaningful ways [25].

Algorithms are used for calculation, data processing, and automated reasoning, algorithms are becoming a ubiquitous part of our lives. Some pundits see danger in this trend. “The NSA revelations highlight the role sophisticated algorithms play in sifting through masses of data. But more surprising is their widespread use in our everyday lives. So should we be more wary of their power [“How algorithms rule the world,” The Guardian, 1 July 2013] It’s a bit hyperbolic to declare that algorithms rule the world but, I agree that their use is becoming more widespread. That’s because computers are playing increasingly important roles in so many aspects of our lives.

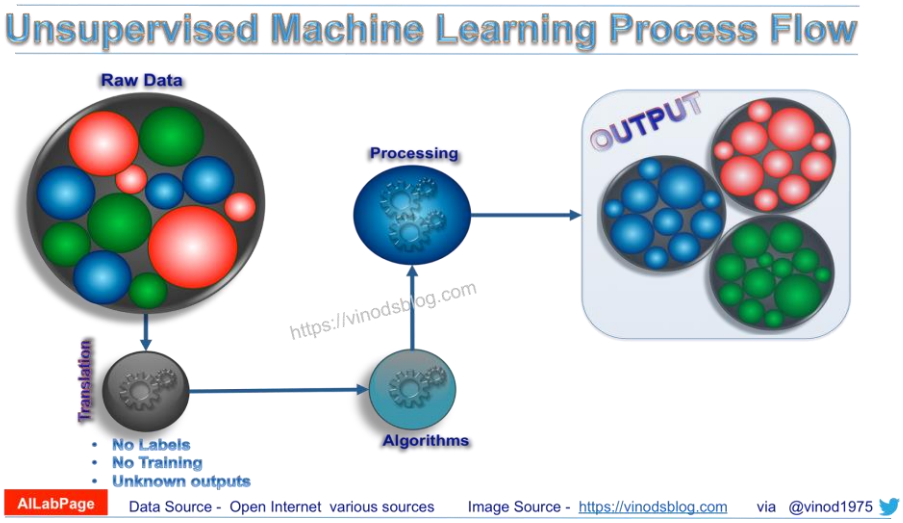



*Fig.3. Process of working unsupervised Learning*

## 1.3 Artificial Intelligence Algorithms

As mentioned above, different Artificial Intelligence algorithms can be used to solve a category of problems. In the below section we'll see the different types of algorithms that fall under Classification, Regression and Clustering problems.

**Models:**

Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.

### 1.3.1 Classification Algorithms and regression/Supervised Learning

Classification, as the name suggests is the act of dividing the dependent variable (the one we try to predict) into classes and then predict a class for a given input. It falls into the category of Supervised Machine Learning, where the data set needs to have the classes, to begin with.
Thus, classification comes into play at any place where we need to predict an outcome, from a set number of fixed, predefined outcomes.

Classification uses an array of algorithms, a few of them listed below

- Naive Bayes
- Decision Tree
- Random Forest
- Logistic Regression
- Support Vector Machines
- K Nearest Neighbours

**a) Naïve Bayes algorithm:**

It is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast

machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

- Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

- Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Naive Bayes algorithm follows the Bayes theorem, which unlike all the other algorithms in this list follows a probabilistic approach. This essentially means that instead of jumping straight into the data, the algorithm has a set of prior probabilities set for each of the classes for your target.

We use the Bayesian formula, here an example for it:

P(Yes | Sunny) = P( Sunny | Yes) * P(Yes) / P (Sunny) Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P( Yes)= 9/14 = 0.64

Advantages of Naïve Bayes Classifier [26]:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.

- It can be used for Binary as well as Multi-class Classifications.

- It performs well in Multi-class predictions as compared to the other Algorithms.

- It is the most popular choice for text classification problems.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

**Applications of Naïve Bayes Classifier:**

- It is used for Credit Scoring.

- It is used in medical data classification.

- It can be used in real-time predictions because Naïve Bayes Classifier is an eager learner.

- It is used in Text classification such as Spam filtering and Sentiment analysis.

**Types of Naïve Bayes Model:**

There are three types of Naive Bayes Model, which are given below [26]:

- Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.

- Multinomial: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems; it means a particular document belongs to which category such as Sports, Politics, education, etc.
The classifier uses the frequency of words for the predictors.

- Bernoulli: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks [26].

**b) Decision Tree**

The Decision Tree can essentially be summarized as a flowchart-like tree structure where each external node denotes a test on an attribute and each branch represents the outcome of that test. The leaf nodes contain the actual predicted labels. We start from the root of the tree and keep comparing attribute values until we reach a leaf node.

We use this classifier when handling high dimensional data and when little time has been spent behind data preparation. However, a word of caution – they tend

to overfit and are prone to change drastically even with slight nuances in the training dat

**Algorithms for Decision Trees:**

Decision trees operate on an algorithmic approach which splits the dataset up into individual data points based on different criteria. These splits are done with different variables, or the different features of the dataset. For example, if the goal is to determine whether or not a dog or cat is being described by the input features, variables the data is split on might be things like “claws” and “barks” [27].

So what algorithms are used to actually split the data into branches and leaves? There are various methods that can be used to split a tree up, but the most common method of splitting is probably a technique referred to as “recursive binary split”. When carrying out this method of splitting, the process starts at the root and the number of features in the dataset represents the possible number of possible splits. A function is used to determine how much accuracy every possible split will cost, and the split is made using the criteria that sacrifices the least accuracy. This process is carried out recursively and sub-groups are formed using the same general strategy [27].

In order to determine the cost of the split, a cost function is used. A different cost function is used for regression tasks and classification tasks. The goal of both cost functions is to determine which branches have the most similar response values, or the most homogenous branches. Consider that you want test data of a certain class to follow certain paths and this makes intuitive sense [27].

In terms of the regression cost function for recursive binary split, the algorithm used to calculate the cost is as follows:

sum(y – prediction)^2

The prediction for a particular group of data points is the mean of the responses of the training data for that group. All the data points are run through the cost function to determine the cost for all the possible splits and the split with the lowest cost is selected.

Regarding the cost function for classification, the function is as follows:

G = sum(pk * (1 – pk))

This is the Gini score, and it is a measurement of the effectiveness of a split, based on how many instances of different classes are in the groups resulting from the split. In other words, it quantifies how mixed the groups are after the split. An optimal split is when all the groups resulting from the split consist only of inputs from one class. If an optimal split has been created the "pk" value will be either 0 or 1 and G will be equal to zero. You might be able to guess that the worst-case split is one where there is a 50-50 representation of the classes in the split, in the case of binary classification. In this case, the "pk" value would be 0.5 and G would also be 0.5 [27].

The splitting process is terminated when all the data points have been turned into leaves and classified. However, you may want to stop the growth of the tree early. Large complex trees are prone to overfitting, but several different methods can be used to combat this. One method of reducing overfitting is to specify a minimum number of data points that will be used to create a leaf. Another method of controlling for over fitting is restricting the tree to a certain maximum depth, which controls how long a path can stretch from the root to a leaf [27].

Another process involved in the creation of decision trees is pruning. Pruning can help increase the performance of a decision tree by stripping out branches containing features that have little predictive power/little importance for the model. In this way, the complexity of the tree is reduced, it becomes less likely to overfit, and the predictive utility of the model is increased [27].

When conducting pruning, the process can start at either the top of the tree or the bottom of the tree. However, the easiest method of pruning is to start with the leaves and attempt to drop the node that contains the most common class within that leaf. If the accuracy of the model doesn't deteriorate when this is done, then the change is preserved. There are other techniques used to carry out pruning, but the method described above – reduced error pruning – is probably the most common method of decision tree pruning [27].

**Random Forest**

Think of this as a committee of Decision Trees, where each decision tree has been fed a subset of the attributes of data and predicts on the basis of that subset. The average of the votes of all decision trees are taken into account and the answer is given.

An advantage of using Random Forest is that it alleviates the problem of overfitting which was present in a standalone decision tree, leading to a much more robust and accurate classifier.

**c) Logistic Regression**

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable (see Fig.4.). The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc

It’s a go-to method mainly for binary classification tasks. The term ‘logistic’ comes from the logit function that is used in this method of classification. The logistic function, also called as the sigmoid function is an S-shaped curve that can take any real-valued number and map it between 0 and 1 but never exactly at those limits.

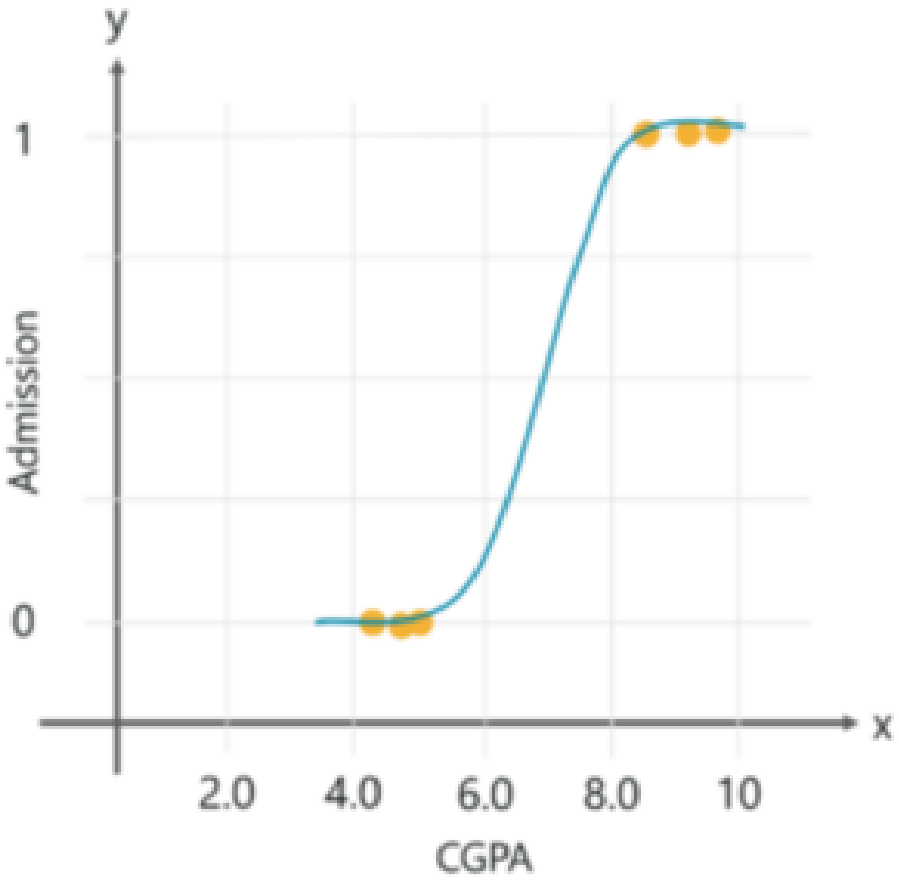



*Fig.4. demonstrate an example of logistic regression*

**Types of Logistic Regression:**

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on that number of categories, Logistic regression can be divided into following types [28]:

Binary or Binomial: In such a kind of classification, a dependent variable will have only two possible types both 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc [28].

Multinomial: In such a kind of classification, dependent variable can have 3 or more possible unordered types or the types having no quantitative significance. For example, these variables may represent “Type A” or “Type B” or “Type C”.

Ordinal: In such a kind of classification, dependent variable can have 3 or more possible ordered types or the types having a quantitative significance. For

example, these variables may represent “poor” or “good”, “very good”, “Excellent” and each category can have the scores like 0,1,2,3 [28].

**Logistic Regression Assumptions** [28]**:**

Before diving into the implementation of logistic regression, we must be aware of the following assumptions:

- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.

- There should not be any multi-collinearity in the model, which means the independent variables must be independent of each other.

- We must include meaningful variables in our model.

- We should choose a large sample size for logistic regression [28].

- These methods listed below are often used to help improve logistic regression models:

- Include interaction terms

- Eliminate features

- Regularize techniques

- Use a non-linear model

**d) Support Vector machine (SVM):**

An SVM is unique, in the sense that it tries to sort the data with the margins between two classes as far apart as possible. This is called maximum margin separation (see Fig.5.).

Another thing to take note of here is the fact that SVM’s take into account only the support vectors while plotting the hyperplane, unlike linear regression which uses the entire dataset for that purpose. This makes SVM’s quite useful in situations when data is in high dimensions.

SVM is binary classification algorithm. Given a set of points of 2 types in N dimensional place, SVM generates a (N—1) dimensional hyperplane to separate those points into 2 groups. Say you have some points of 2 types in a paper which are

linearly separable. SVM will find a straight line which separates those points into 2 types and situated as far as possible from all those points.

SVM distinguishes classes by drawing a decision boundary. How to draw or determine the decision boundary is the most critical part in SVM algorithms. Before creating the decision boundary, each observation (or data point) is plotted in n-dimensional space. “n” is the number of features used. For instance, if we use “length” and “width” to classify different “cells”, observations are plotted in a 2-dimensional space and decision boundary is a line. If we use 3 features, decision boundary is a plane in 3-dimensional space. If we use more than 3 features, decision boundary becomes a hyperplane which is really hard to visualize.

Decision boundary is drawn in a way that the distance to support vectors are maximized. If the decision boundary is too close to a support vector, it will be highly sensitive to noises and not generalize well. Even very small changes in independent variables may cause a misclassification [29].

The data points are not always linearly separable like in the figure above. In these cases, SVM uses kernel trick which measures the similarity (or closeness) of data points in a higher dimensional space in order to make them linearly separable [29].

Kernel function is kind of a similarity measure. The inputs are original features and the output is a similarity measure in the new feature space. Similarity here means a degree of closeness. It is a costly operation to actually transform data points to a high-dimensional feature space. The algorithm does not actually transform the data points to a new, high dimensional feature space. Kernelized SVM compute decision boundaries in terms of similarity measures in a high-dimensional feature space without actually doing a transformation. I think this is why it is also called kernel trick [29].

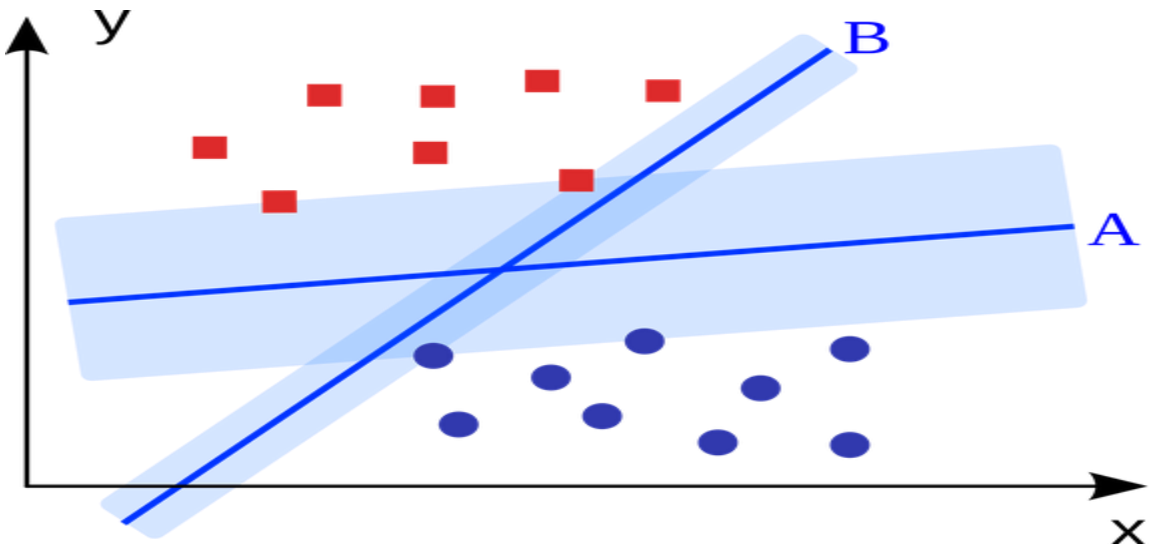



*Fig.5.Description of SVM*

**e) Regression Model K-Nearest Neighbors (KNN)**

KNN is a non-parametric (here non-parametric is just a fancy term which essentially means that KNN does not make any assumptions on the underlying data distribution), lazy learning algorithm (here lazy means that the "training" phase is fairly short) [30].

Its purpose is to use a whole bunch of data points separated into several classes to predict the classification of a new sample point.

The following points serve as an overview of the general working of the algorithm [30].:

- A positive integer N is specified, along with a new sample
- We select the N entries in our database which are closest to the new sample
- We find the most common classification of these entries
- This is the classification we give to the new sample

However, there are some downsides to using KNN. These downsides mainly revolve around the fact that KNN works on storing the entire dataset and comparing new points to existing ones. This means that the storage space increases as our training set increases. This also means that the estimation time increases in proportion to the number of training points [30].

In the case of regression problems, the output is a continuous quantity. Meaning that we can use regression algorithms in cases where the target variable is a continuous variable. It falls into the category of Supervised Machine Learning, where the data set needs to have the labels, to begin with.

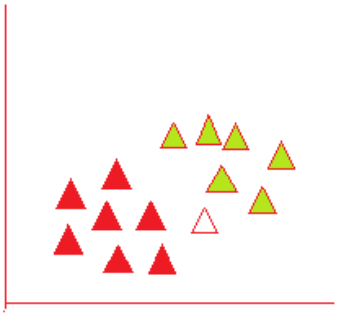


*Fig.6. Example about KNN*

**The Importance of K in a K-Nearest Neighbors Algorithm**

Although it might not be obvious from the start, changing the value of K in a K-nearest neighbor's algorithm will change which category a new point is assigned to.

More specifically, having a very low K value will cause your model to perfectly predict your training data and poorly predict your test data. Similarly, having too high of a K value will make your model unnecessarily complex.

**f) Linear Regression**

Linear Regression is the most simple and effective regression algorithm. It is utilized to gauge genuine qualities (cost of houses, number of calls, all out deals and so forth.) in view of the consistent variable(s). Here, we build up a connection between free and ward factors by fitting the best line. This best fit line is known as regression line and spoken to by a direct condition Y= a *X + b (see Fig.7.).

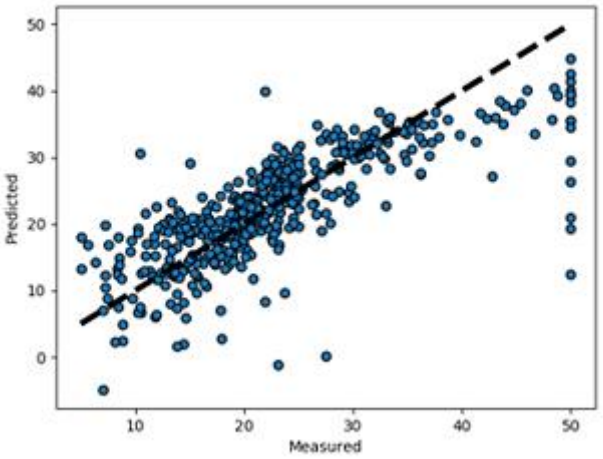



*Fig.7. Example Graphic for linear regression*

Let us take a simple example here to understand linear regression:

Consider that you are given the challenge to estimate an unknown person's weight by just looking at them. With no other values in hand, this might look like a fairly difficult task, however using your past experience you know that generally speaking the taller someone is, the heavier they are compared to a shorter person of the same build. This is linear regression, in actuality! However, linear regression is best used in approaches involving a low number of dimensions. Also, not every problem is linearly separable [30].

Some of the most popular applications of Linear regression are in financial portfolio prediction, salary forecasting, real estate predictions and in traffic in arriving at ETAs

Now let's discuss how clustering problems can be solved by using the K-means algorithm. Before that, let's understand what clustering is [30].

### 1.3.2 Clustering Algorithms/ Unsupervised learning

The basic idea behind clustering is to assign the input into two or more clusters based on feature similarity. It falls into the category of Unsupervised Machine Learning, where the algorithm learns the patterns and useful insights from data without any guidance (labeled data set).

For example, clustering viewers into similar groups based on their interests, age, geography, etc can be done by using Unsupervised Learning algorithms like K-Means Clustering.

**K-Means :**

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are:

- The centroids of the K clusters, which can be used to label new data
- Labels for the training data (each data point is assigned to a single cluster)

K-means is probably the simplest unsupervised learning approach. The idea here is to gather similar data points together and bind them together in the form of a cluster. It does this by calculating the centroid of the group of data points.

To carry out effective clustering, k-means evaluates the distance between each point from the centroid of the cluster. Depending on the distance between the data point and the centroid, the data is assigned to the closest cluster. The goal of clustering is to determine the intrinsic grouping in a set of unlabelled data.

The ‘K’ in K-means stands for the number of clusters formed. The number of clusters (basically the number of classes in which your new instances of data can fall into) is determined by the user.

K-means is used majorly in cases where the data set has points which are distinct and well separated from each other, otherwise, the clusters won’t be far apart, rendering them inaccurate. Also, K-means should be avoided in cases where the data set contains a high amount of outliers or the data set is non-linear.

**Algorithm**

The K-means clustering algorithm uses iterative refinement to produce a final result. The algorithm inputs are the number of clusters K and the data set. The data set is a collection of features for each data point. The algorithms starts with initial estimates for the K centroids, which can either be randomly generated or randomly selected from the data set. The algorithm then iterates between two steps:

- Data assigment step:

Each centroid defines one of the clusters. In this step, each data point is assigned to its nearest centroid, based on the squared Euclidean distance. More formally, if ci is the collection of centroids in set C, then each data point x is assigned to a cluster based on (see Eq.1.):

$$\underset{c_i \in C}{\arg\min}\ dist(c_i, x)^2 \qquad \textit{(Eq.1.)}$$

where dist(ci, x) is the standard (L2) Euclidean distance. Let the set of data point assignments for each ith cluster centroid be Si.

- Centroid update step:

In this step, the centroids are recomputed. This is done by taking the mean of all data points assigned to that centroid's cluster.

The algorithm iterates between steps one and two until a stopping criteria is met (i.e., no data points change clusters, the sum of the distances is minimized, or some maximum number of iterations is reached).

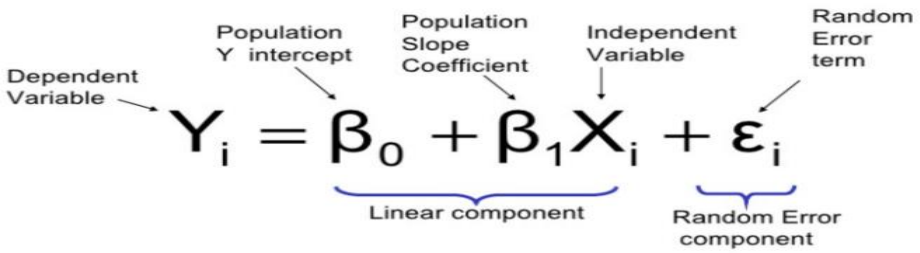
This algorithm is guaranteed to converge to a result. The result may be a local optimum (i.e. not necessarily the best possible outcome), meaning that assessing more than one run of the algorithm with randomized starting centroids may give a better outcome.

**Choosing K** [31]**:**

The algorithm described above finds the clusters and data set labels for a particular pre-chosen K. To find the number of clusters in the data, the user needs to run the K-means clustering algorithm for a range of K values and compare the results. In general, there is no method for determining exact value of K, but an accurate estimate can be obtained using the following techniques [31].

One of the metrics that is commonly used to compare results across different values of K is the mean distance between data points and their cluster centroid. Since increasing the number of clusters will always reduce the distance to data points, increasing K will always decrease this metric, to the extreme of reaching zero when K is the same as the number of data points. Thus, this metric cannot be used as the sole target. Instead, mean distance to the centroid as a function of K is plotted and the "elbow point," where the rate of decrease sharply shifts, can be used to roughly determine K (see Eq.2.) [31].

A number of other techniques exist for validating K, including cross-validation, information criteria, the information theoretic jump method, the silhouette method, and the G-means algorithm. In addition, monitoring the distribution of data points across groups provides insight into how the algorithm is splitting the data for each K [31].




*Eq.2.*

# Chapter II: Modeling of computer system

Anticipating a world dominated by electric vehicles, materials scientists are working on two big challenges. One is how to cut down on the metals in batteries that are scarce, expensive, or problematic because their mining carries harsh environmental and social costs. Another is to improve battery recycling, so that the valuable metals in spent car batteries can be efficiently reused. “Recycling will play a key role in the mix,” says Kwasi Ampofo, a mining engineer who is the lead analyst on metals and mining at BNEF.

## 2.1 Process of computer system in electric cars

Battery- and carmakers are already spending billions of dollars on reducing the costs of manufacturing and recycling electric-vehicle (EV) batteries — spurred in part by government incentives and the expectation of forthcoming regulations. National research funders have also founded centres to study better ways to make and recycle batteries. Because it is still less expensive, in most instances, to mine metals than to recycle them, a key goal is to develop processes to recover valuable metals cheaply enough to compete with freshly mined ones. “The biggest talker is money,” says Jeffrey Spangenberger, a chemical engineer at Argonne National Laboratory in Lemont, Illinois, who manages a US federally funded lithium-ion battery-recycling initiative, called ReCell (see Fig.8.) [32].

The lithium-ion battery packs used in electric cars are similar to those used in cell phones and laptop computers, only they’re much larger. They’re far different than the heavy lead-acid batteries used in conventional cars and have a higher energy density than rechargeable nickel-metal hydride batteries. They’re also less prone than other battery types to lose their charge when not being used. EV battery packs generally contain a series of connected individual cells, perhaps several hundred of them depending on the model, instead of a single massive unit [33].

An electric car's battery capacity is expressed in terms of kilowatt-hours, which is abbreviated as kWh. More is better here. Choosing an EV with a higher kWh rating is like buying a car that comes with a larger gas tank in that you'll be able to drive for more miles before needing a "fill up." At that, be aware that an electric car's management system prevents the battery from either becoming 100 percent fully charged or 100 percent discharged to preserve its efficiency and extend its usable life [33].

The Environmental Protection Agency rates electric cars according to their energy efficiency and estimates each model's average operating range on a full charge. However, as they say, your mileage may vary. If you own an electric car and find that you're not getting anywhere near the estimated range, that doesn't necessarily mean the car's battery pack is becoming seriously depleted [33].

On the downside, electric cars kept in the hottest climates can be expected to lose battery capacity a bit quicker than those living in more temperate areas. Extreme heat is the enemy of lithium-ion chemistry, which is why many electric cars come with liquid-cooled battery packs. Also, older electric cars having relatively short operating ranges may suffer quicker deterioration. That's because draining most or all of a battery's charge on a regular basis tends to cut into its capacity more quickly over time. That's far less of an issue with today's longer-range models that are typically driven for a fraction of their available capacity on a daily basis and are merely "topped off" at night [33].

Excessive use of public Level 3 DC Fast Charging stations (they can bring an EV up to 80 percent of its capacity in as little as 30 minutes) can also take a toll on a battery's long-term performance. That's because the faster an electric car is charged, the hotter it becomes and, again, that's not battery friendly. However, a study conducted by the Idaho National Laboratory concluded that the effect isn't particularly pronounced. The INL tested two pairs of identical 2012 Nissan Leaf models, with one set using 240-volt Level 2 home charging and the other exclusively relying on DC Fast Charging public units. After each was driven 50,000 miles, the difference between the Level 2 and Level 3 vehicles' diminished battery capacities

amounted to just four percent. In Fig.8. there is a description of computer system model [33]:

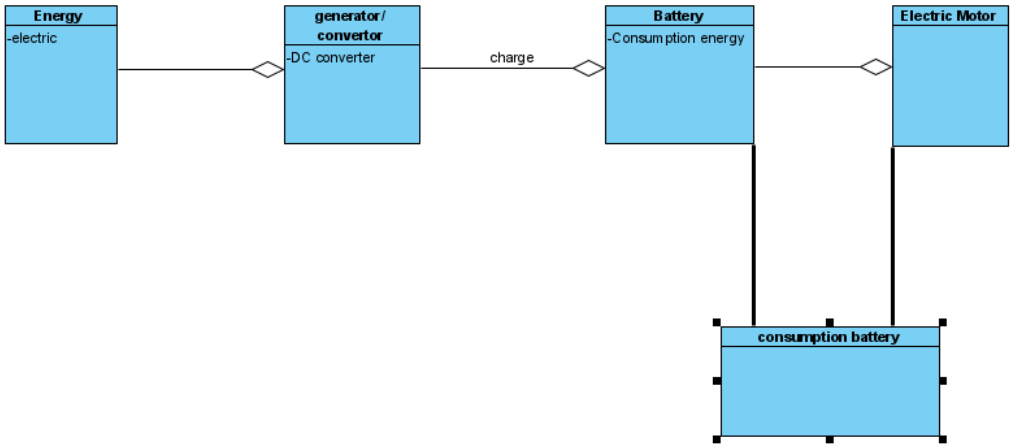



*Fig.8. Model of computer system for consumption of battery electric cars*

## 2.2 Different elements for computer system in electric cars

The power system of an electric vehicle consists of just two components: the motor that provides the power and the controller that controls the application of this power. In comparison, the power system of gasoline-powered vehicles consists of a number of components, such as the engine, carburetor, oil pump, water pump, cooling system, starter, exhaust system, etc. Motors Electric motors convert electrical energy into mechanical energy. Two types of electric motors are used in electric vehicles to provide power to the wheels: the direct current (DC) motor and the alternating current (AC) motor. DC electric motors have three main components [34]:

- A set of coils (field) that creates the magnetic forces which provide torque
- A rotor or armature mounted on bearings that turns inside the field
- Commutating device that reverses the magnetic forces and makes the armature turn, there by providing horsepower.

As in the DC motor, an AC motor also has a set of coils (field) and a rotor or armature, however, since there is a continuous current reversal, a commutating device is not needed. Both types of electric motors are used in electric vehicles and have

advantages and disadvantages, as shown here. While the AC motor is less expensive and lighter weight, the DC motor has a simpler controller, making the DC motor/controller combination less expensive. The main disadvantage of the AC motor is the cost of the electronics package needed to convert (invert) the battery‘s direct current to alternating current for the motor. Past generations of electric vehicles used the DC motor/controller system because they operate off the battery current without complex electronics. The DC motor/controller system is still used today on some electric vehicles to keep the cost down. However, with the advent of better and less expensive electronics, a large number of today’s electric vehicles are using AC motor/controller systems because of their improved motor efficiency and lighter weight [34].

## 2.3 Description of software used for modeling computer system (python):

Software python is an interpreter object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interprete  and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed (see Fig.9.) [35].

One of the most common uses for Python is in its ability to create and manage data structures quickly Pandas, for instance, offers a plethora of tools to manipulate, analyze, and even represent data structures and complex datasets.

This includes time series and more complex data structures such as merging, pivoting, and slicing tables to create new views and perspectives on existing sets.

Elsewhere, tools like Scikit-Learn (also known as Sklearn) provides advanced analytics tools combined with complex machine learning capabilities.

This allows you to build more sophisticated models, performing more complex and multivariate regressions, as well as data preprocessing.

Combined with libraries such as iPython and NumPy itself, these tools can form the foundation of a powerful data analytics suite.

There are a ton of evaluations for the time it takes to learn Python. For data science explicitly, estimates a range from 3 months to a time of predictable practice. We've watched individuals travel through our courses at lightning velocity and other people who have taken it much slower. Truly, everything relies upon your ideal timeline and free time that you can commit to learning Python programming and the pace at which you learn. Additionally, there is also:

- Data visualization using Matplotlib
- SQL with Python
- Statistics with Python

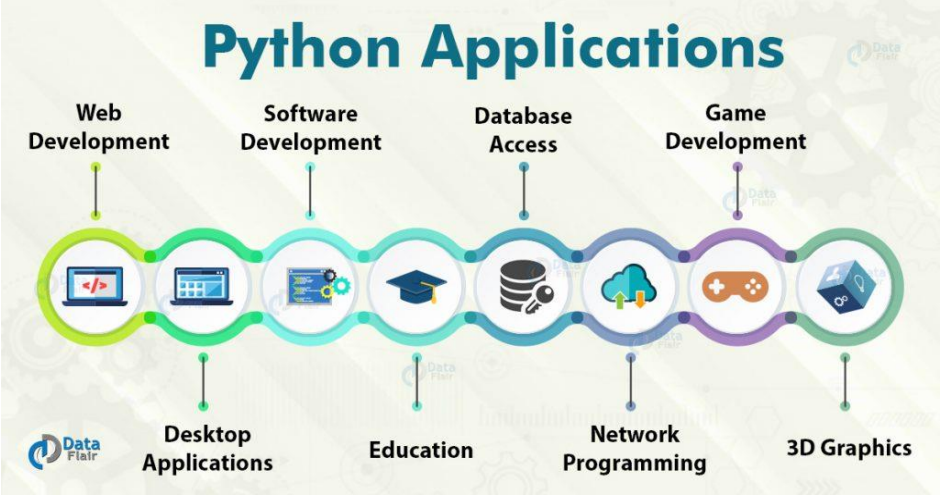



*Fig.9. Processing & steps of software Python*

Use of data in python need some librairies :

- NumPy stands for Numerical Python. The most powerful feature of NumPy is n-dimensional array. This library also contains basic linear algebra functions, Fourier transforms, advanced random number capabilities and tools for integration with other low level languages like Fortran, C and C++.

- SciPy stands for Scientific Python. It is built on NumPy. Scipy is one of the most useful library for variety of high level science and engineering modules like discrete Fourier transform, Linear Algebra, Optimization and Sparse matrices.

- Pandas for structured data operations and manipulations. It is extensively used for data munging and preparation. Pandas were added relatively recently to Python and have been instrumental in boosting Python's usage in data scientist community.

- Matplotlib for plotting vast variety of graphs, starting from histograms to line plots to heat plots.. You can use Pylab feature in ipython notebook (ipython notebook –pylab = inline) to use these plotting features inline. If you ignore the inline option, then pylab converts ipython environment to an environment, very similar to Matlab.

- Scikit Learn for machine learning. Built on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensional reduction.

- Statsmodels for statistical modeling. It is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator.

- Seaborn for statistical data visualization. It is a library for making attractive and informative statistical graphics in Python. It is based on matplotlib. Seaborn aims to make visualization a central part of exploring and understanding data.

As conclusion the possibilities of Python language, it takes a lot of practice to master all the subtitles. Just as a language has a vocabulary, syntax, that everyone can make their own, so does Python. With the basics it is possible to deal with many simple or less simple projects, and the fact that Python is free and open-source and that there is a huge community makes it possible to find solutions.

Python supports both function-oriented and structure-oriented programming. It has features of dynamic memory management which can make use of computational resources efficiently. It is also compatible with all popular operating systems and platforms. Hence this language can be universally accepted by all programmers.

# Chapter III: Implementation of algorithms used in electric battery cars

## 3.1 Different types of Deep learning:

Types of Neural Networks in Deep Learning :

There is three important types of neural networks that form the basis for most pre-trained models in deep learning:

- Artificial Neural Networks (ANN)
- Convolution Neural Networks (CNN)
- Recurrent Neural Networks (RNN)

Artificial Neural Network (ANN), What is a ANN and why should you use it?

A single perceptron (or neuron) can be imagined as a Logistic Regression. Artificial Neural Network, or ANN, is a group of multiple perceptrons/ neurons at each layer. ANN is also known as a Feed-Forward Neural network because inputs are processed only in the forward direction:

As you can see here, ANN consists of 3 layers – Input, Hidden and Output. The input layer accepts the inputs, the hidden layer processes the inputs, and the output layer produces the result. Essentially, each layer tries to learn certain weights.

An artificial neural network is an application, non linear with respect to its parameters "teta" that associates to an entry x an out put. For the sake of simplicity, we assume that y is one-dimensional, but it could also be multidimensional. This application f has a particular form that we will precise. The neural networks can be use for regression or classification. As usual in statistical learning, the parameters _are estimated from a learning sample. The function to minimize is not convex, leading to local minimizers. The success of the method came from a universal approximation theorem due to Cybenko (1989) and Hornik (1991). Moreover, Le Cun (1986) proposed an efficient way to compute the gradient of a neural network, called back propagation of the gradient, that allows to obtain a local minimize of the quadratic criterion easily.

**Advantages of Artificial Neural Network (ANN)** [36]**:**

Artificial Neural Network is capable of learning any nonlinear function. Hence, these networks are popularly known as Universal Function Approximators. ANNs have the capacity to learn weights that map any input to the output [36].

One of the main reasons behind universal approximation is the activation function. Activation functions introduce nonlinear properties to the network. This helps the network learn any complex relationship between input and output [36].

Perceptron :

As you can see here, the output at each neuron is the activation of a weighted sum of inputs. But wait  what happens if there is no activation function? The network only  learns the linear function and can never learn complex relationships [36].

**Challenges with Artificial Neural Network (ANN):**

- While solving an image classification problem using ANN, the first step is to convert a 2-dimensional image into a 1-dimensional vector prior to training the model. This has two drawbacks [36]:

The number of trainable parameters increases drastically with an increase in the size of the image

**ANN: Image classification** [36]

ANN loses the spatial features of an image. Spatial features refer to the arrangement of the pixels in an image. I will touch upon this in detail in the following sections One common problem in all these neural networks is the Vanishing and Exploding Gradient. This problem is associated with the back propagation algorithm. The weights of a neural network are updated through this back propagation algorithm by finding the gradients [36]:

**Backward Propagation**

So, in the case of a very deep neural network (network with a large number of hidden layers), the gradient vanishes or explodes as it propagates backward which leads to vanishing and exploding gradient.

ANN cannot capture sequential information in the input data which is required for dealing with sequence data

**Recurrent Neural Networks**

can be thought of as a series of networks linked together. They often have a chain-like architecture, making them applicable for tasks such as speech recognition, language translation, etc. An RNN (see Fig.10.) can be designed to operate across sequences of vectors in the input, output, or both. For example, a sequenced input may take a sentence as an input and output a positive or negative sentiment value. Alternatively, a sequenced output may take an image as an input, and produce a sentence as an output.

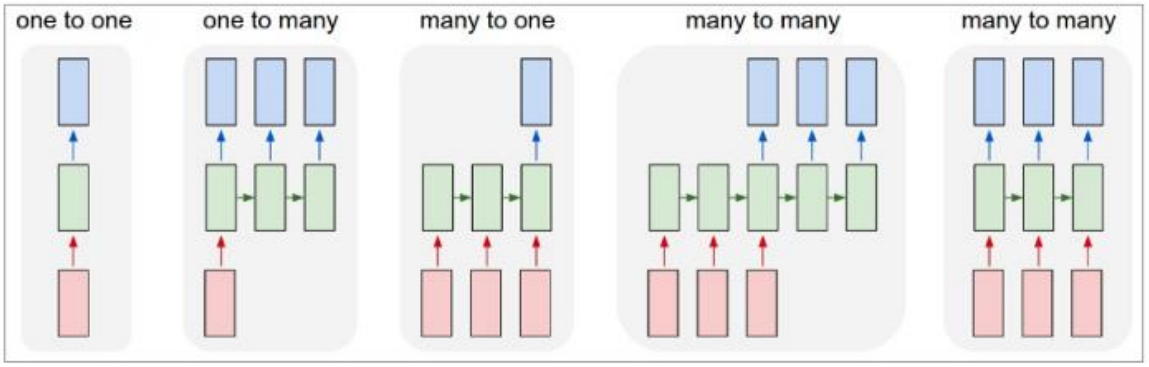



*Fig.10. Different types of RNN*

An artificial neuron is a function fj of the input x = (x1;.....;xd) weighted by a vector of connection weights wj= (wj;1;.....;wj;d), (see Fig.11.) completed by a neuron bias bj, and associated to an activation function , namely (see Eq.3.):

$$y_j = f_j(x) = \emptyset((w_j, x) \quad + b_j) \quad \textit{Eq.3}$$

.

Several activation functions can be considered. The identity function (see Eq.4.):

$$\phi(x) = x \qquad \textit{Eq.4.}$$

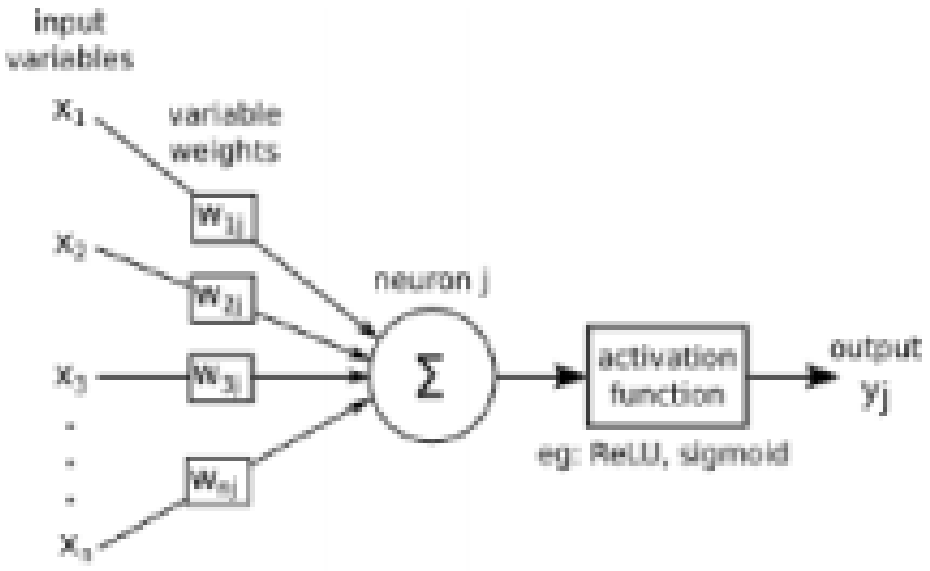



*Fig.11. Steps of Artificial Neuron*

Recurrent Neural Networks, or RNNs, were designed to work with sequence prediction problems. Sequence prediction problems come in many forms and are best described by the types of inputs and outputs supported. Some examples of sequence prediction problems include [37]:

- One-to-Many: An observation as input mapped to a sequence with multiple steps as an output.

- Many-to-One: A sequence of multiple steps as input mapped to class or quantity prediction.

- Many-to-Many: A sequence of multiple steps as input mapped to a sequence with multiple steps as output.

The Many-to-Many problem is often referred to as sequence-to-

sequence, or seq to seq for short.

Recurrent neural networks were traditionally difficult to train:

- The Long Short-Term Memory, or LSTM, network is perhaps the most successful RNN because it overcomes the problems of training a recurrent network and in turn has been used on a wide range of applications.

- RNNs in general and LSTMs in particular have received the most success when working with sequences of words and paragraphs, generally called natural language processing [37].

This includes both sequences of text and sequences of spoken language represented as a time series. They are also used as generative models that require a sequence output, not only with text, but on applications such as generating handwriting [37].

Use RNNs For:

- Text data

- Speech data

- Classification prediction problems

- Regression prediction problems

- Generative models

**LSTM (Long Short term Memory)**

It is special kind of recurrent neural network that is capable of learning long term dependencies in data. This is achieved because the recurring module of the model has a combination of four layers interacting with each other.

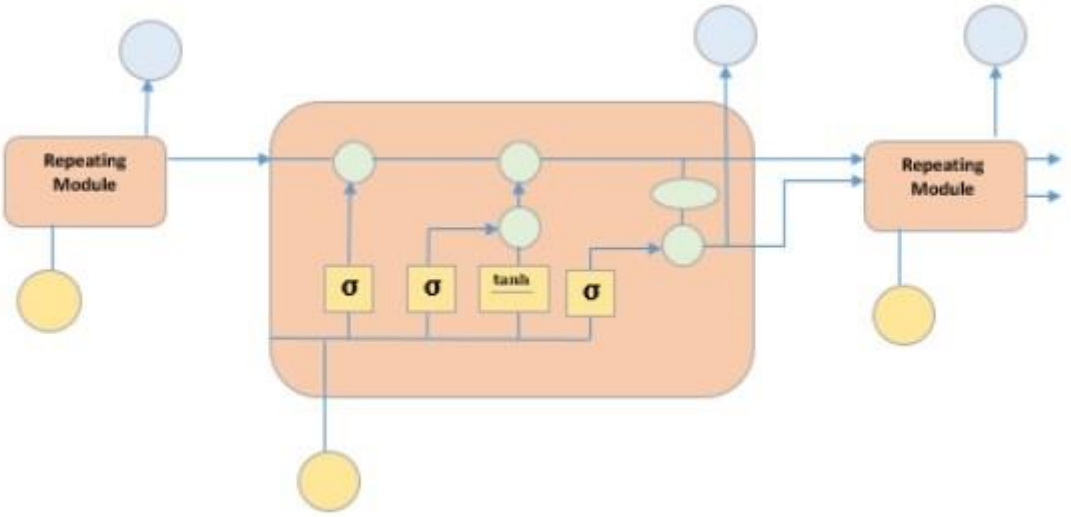



*Fig.12. Processing system of LSTM*

This Fig.12. Above depicts four neural network layers in yellow boxes, point wise operators in green circles, input in yellow circles and cell state in blue circles. An LSTM module has a cell state and three gates which provides them with the power to selectively learn, unlearn or retain information from each of the units. The cell state in LSTM helps the information to flow through the units without being altered by allowing only a few linear interactions. Each unit has an input, output and a forget gate which can add or remove the information to the cell state. The forget gate decides which information from the previous cell state should be forgotten for which it uses a sigmoid function. The input gate controls the information flow to the current cell state using a point-wise multiplication operation of 'sigmoid' and 'tanh' respectively. Finally, the output gate decides which information should be passed on to the next hidden state [38].

With conventional Back-Propagation Through Time (BPTT) or Real Time Recurrent Learning (RTTL), error signals flowing backward in time tend to either explode or vanish [39].

The temporal evolution of the back-propagated error exponentially depends on the size of the weights. Weight explosion may lead to oscillating

weights, while in vanishing causes learning to bridge long time lags and takes a prohibitive amount of time, or does not work at all [39].

- LSTM is a novel recurrent network architecture training with an appropriate gradient-based learning algorithm.

- LSTM is designed to overcome error back-flow problems. It can learn to bridge time intervals in excess of 1000 steps.

- This true in presence of noisy, incompressible input sequences, without loss of short time lag capabilities.

Error back-flow problems are overcome by an efficient, gradient-based algorithm for an architecture enforcing constant (thus neither exploding nor vanishing) error flow through internal states of special units. These units reduce the effects of the "Input Weight Conflict" and the "Output Weight Conflict" [39].

The Input Weight Conflict: Provided the input is non-zero, the same incoming weight has to be used for both storing certain inputs and ignoring others, then will often receive conflicting weight update signals.

These signals will attempt to make the weight participate in storing the input and protecting the input. This conflict makes learning difficult and calls for a more context-sensitive mechanism for controlling "write operations" through input weights [39].

**- The Output Weight Conflict**: As long as the output of a unit is non-zero, the weight on the output connection from this unit will attract conflicting weight update signals generated during sequence processing.

These signals will attempt to make the outgoing weight participate in accessing the information stored in the processing unit and, at different times, protect the subsequent unit from being perturbed by the output of the unit being fed forward [39].

These conflicts are not specific to long-term lags and can equally impinge on short term lags. Of note though is that as lag increases, stored information must be protected from perturbation, especially in the advanced stages of learning.

**- Network Architecture**: Different types of units may convey useful information about the current state of the network. For instance, an input gate (output gate) may use inputs from other memory cells to decide whether to store (access) certain information in its memory cell.

Memory cells contain gates. Gates are specific to the connection they mediate. Input gates work to remedy the Input Weight Conflict while Output Gates work to eliminate the Output Weight Conflict [39].

**- Gates**: Specifically, to alleviate the input and output weight conflicts and perturbations, a multiplicative input gate unit is introduced to protect the memory contents stored from perturbation by irrelevant inputs and a multiplicative output gate unit protects other units from perturbation by currently irrelevant memory contents stored. (See Fig.13.) [39].

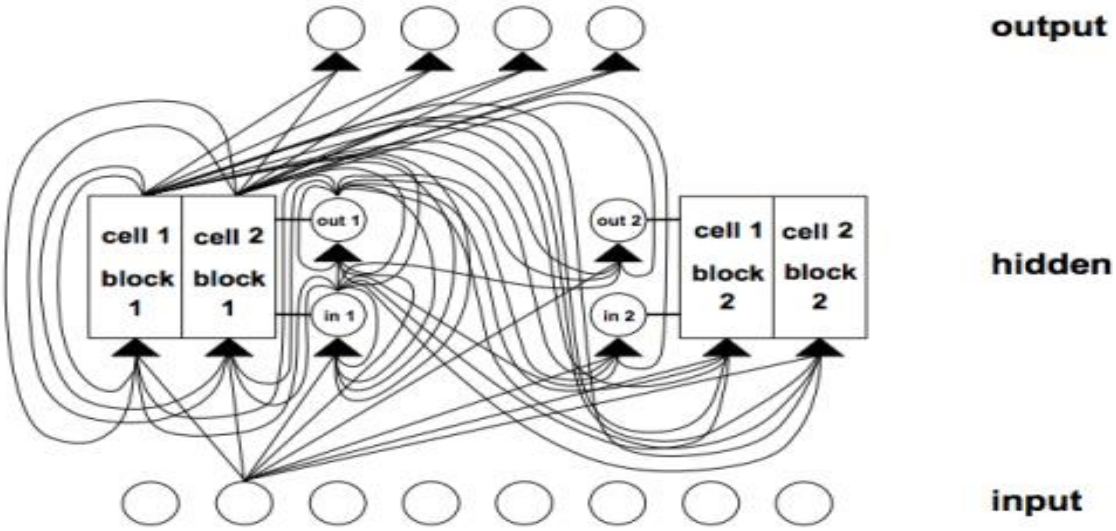



*Fig.13. steps of LSTM*

Connectivity in LSTM is complicated compared to the multilayer

Perceptron because of the diversity of processing elements and the inclusion of feedback connections [39].

Memory cell blocks: Memory cells sharing the same input gate and the same output gate form a structure called a “memory cell block”.

Memory cell blocks facilitate information storage; as with conventional neural nets, it is not so easy to code a distributed input within a single cell. A memory cell block of size 1 is just a simple memory cell [39].

Learning: A variant of Real Time Recurrent Learning (RTRL) that takes into account the altered, multiplicative dynamics caused by input and output gates is used to ensure non-decaying error back propagated through internal states of memory cells errors arriving at “memory cell net inputs” do not get propagated back further in time [39].

Guessing: This stochastic approach can outperform many term lag algorithms. It has been established that many long-time lag tasks used in previous work can be solved more quickly by simple random weight guessing than by the proposed algorithms [39].

**- LSTM Limitations:**

The efficient, truncated version of LSTM will not easily solve problems similar to “strongly delayed XOR.”

Each memory cell block needs an input gate and an output gate. Not necessary in other recurrent approaches.

Constant error flow through “Constant Error Carrousels” inside memory cells produces the same effect as a conventional feed-forward architecture being presented with the entire input string at once [39].

LSTM is as flawed with the concept of “regency” as other feed-forward approaches. Additional counting mechanisms may be required if fine-precision counting time steps is needed [39].

**- LSTM Advantages:**

The algorithms ability to bridge long time lags is the result of constant error Back propagation in the architecture’s memory cells.

LSTM can approximate noisy problem domains, distributed representations, and continuous values.

LSTM generalizes well over problem domains considered. This is important given some tasks are intractable for already established recurrent networks.

Fine tuning of network parameters over the problem domains appears to be unnecessary.

In terms of update complexity per weight and time steps, LSTM is essentially equivalent to BPTT.

LSTMs are showing to be powerful, achieving state-of-the-art results in domains like machine translation.

## 3.2 Method of Recurrent Neural Networks:

I chose the method of RNN about neural network (deep learning), because it's the best method for time series and for my data, the description of method for example we provide the first 4 letters i.e. h,e,l,l and ask the network to predict the last letter i.e.’o’. So here the vocabulary of the task is

just 4 letters {h,e,l,o}. In real case

Scenarios involving natural language processing, the vocabularies include the words in entire wikipedia database, or all the words in a language. Here for simplicity we have taken a very small set of vocabulary.

Recurrence (RNN) is by formula to the input vector and also its previous state. In this case, the letter “h” has nothing preceding it, let’s take the letter “e”. So at the time the letter “e” is supplied to the network, a recurrence formula is applied to the letter “e” and the previous state which is the letter “h”. These are known as various time steps of the input. So if at time t, the input is “e”, at time t-1, the input was “h”. The recurrence formula is applied to e and h both and we get a new state. The formula for the current state can be written as (see Eq.5.) [40]:

$$h_t = f(h_{t-1}, x_t) \qquad \textit{Eq.5.}$$

Here, Ht is the new state; ht-1 is the previous state while xt is the current input. We have now a state of the previous input instead of the input itself, because the input neuron would have applied the transformations on our previous input. So each successive input is called as a time step [40].

In this case we have four inputs to be given to the network, during a recurrence formula, the same function and the same weights are applied to the network at each time step.

Taking the simplest form of a recurrent neural network, let's say that the activation function is tanh, the weight at the recurrent neuron is Whh and the weight at the input neuron is W x h, we can write the equation for the state at time t as (see Eq.6.):

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t) \quad \textit{Eq.6.}$$

The Recurrent neuron in this case is just taking the immediate previous state into consideration. For longer sequences the equation can involve multiple such states. Once the final state is calculated we can go on to produce the output. Now, once the current state is calculated we can calculate the output state as (see Eq.7.) [[40]]:

$$y_t = W_{hy}h_t \quad \textit{Eq.7.}$$

The steps in a recurrent neuron:

- A single time step of the input is supplied to the network i.e. xt is supplied to the network

- We then calculate its current state using a combination of the current input and the previous state i.e. we calculate ht

- The current ht becomes ht-1 for the next time step

- We can go as many time steps as the problem demands and combine the information from all the previous states

- Once all the time steps are completed the final current state is used to calculate the output yt

- The output is then compared to the actual output and the error is generated

- The error is then back propagated to the network to update the weights and the network is trained

**-Tips for Training Recurrent Neural Networks:**

- **Adaptive learning rate.** We usually use adaptive optimizers because they can better handle the complex training dynamics of recurrent networks that plain gradient descent.

- **Gradient clipping.** Print or plot the gradient norm to see its usual range, then scale down gradients that exceeds this range. This prevents spikes in the gradients to mess up the parameters during training.

- **Normalizing the loss.** To get losses of similar magnitude across datasets, you can sum the loss terms along the sequence and divide them by the maximum sequence length. This makes it easier to reuse hyper parameters between experiments. The loss should be averaged across the batch [41].

-**Truncated back propagation.** Recurrent networks can have a hard time learning long sequences because of vanishing and noisy gradients. Train on overlapping chunks. You can also gradually increase the chunk length during training. Preserve the hidden state between chunk boundaries.

-**Long training time.** Especially in language modeling, small improvements in loss can make a big difference in the perceived quality of the model. Stop training

when the training loss does not improve for multiple epochs or the evaluation loss starts increasing [41].

-**Multi-step loss.** When training generative sequence models, there is a trade-off between 1-step losses (teacher forcing) and training longer imagined sequences towards matching the target. Professor forcing combines the two but is more involved.

**Network Structure:**

- **Gated Recurrent Unit**: (GRU) alternative memory cell design to LSTM. I found it to often reach the equal or better performance while using fewer parameters and being faster to compute.

- **Layer normalization.** Adding layer normalization to all linear mappings of the recurrent network speeds up learning and often improves final performance. Multiple inputs to the same mapping should be normalized separately as done in the paper.

- **Feed-forward layers first.** Preprocessing the input with feed-forward layers allows your model to project the data into a space with easier temporal dynamics. This can improve performance on the task.

- **Stacked recurrent networks.** Recurrent networks need a quadratic number of weights in their layer size. It can be more efficient to stack two or three smaller layers instead of one big one. Sum the outputs of all layers instead of using only the last one, similar to a ResNet or DenseNet [41].

**Model Parameters:**

- **Learned initial state:** Initializing the hidden state as zeros can cause large loss terms for the first few time steps, so that the model focuses less on the actual sequence [41].

- **Forget gate bias:** It can take a while for a recurrent network to learn to

remember information from the last time step. Initialize biases for LSTM's forget gate to 1 to remember more by default. Similarly, initialize biases for GRU's reset gate to -1.

- **Regularization:** If your model is over fitting, use specific regularization methods for recurrent networks.

### 3.3 Analysis of data and results

I will start by making architecture of steps from analysis to results (see Fig.14.)

:

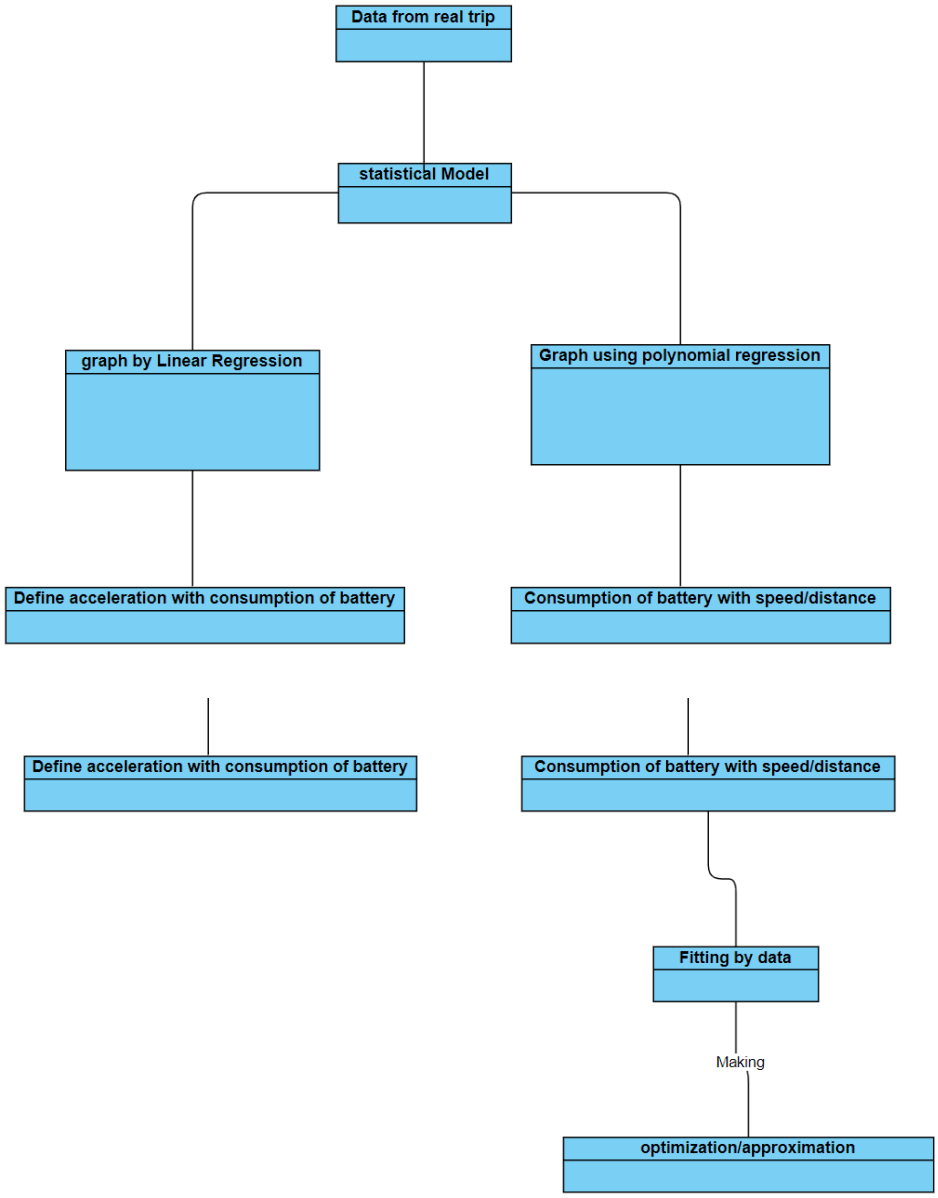



*Fig.14. Architecture of analysis data*

In this part I used a data of real trip (TESLA), which I studied the consumption of battery and speed (acceleration), as first step there is some equations and graphs related to linear regression (see Fig.15.)

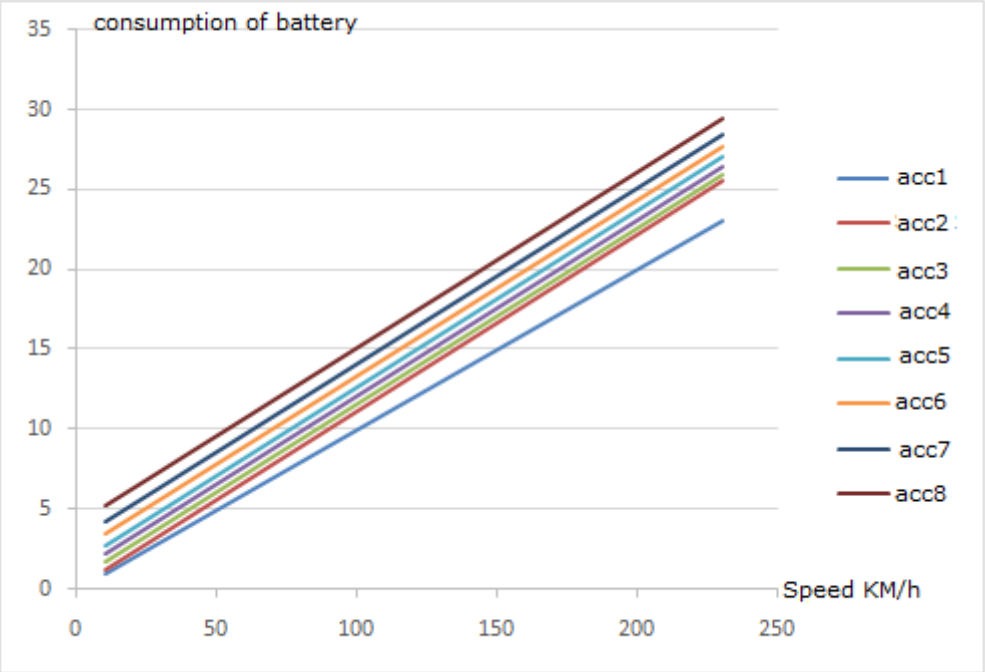



*Fig.15. Different accelerations with consumption*

**Equation for each acceleration:**

y = 1.016X-2.782

Y= 2.117X+2.098

Y= 1.220X+1.259

Y= 2.365X+3.876

Y= 3.489X+2.0645

Y= 1.100X+5.009

Y= 4.900X+ 2.754

Y= 2.765X+7.01

With the help of some functions and informations of data, an analysis of how is the consumption of battery (and charge of battery) related to speed/distance, using also the data from a real trip to describe consumption (see Fig.16 &17):

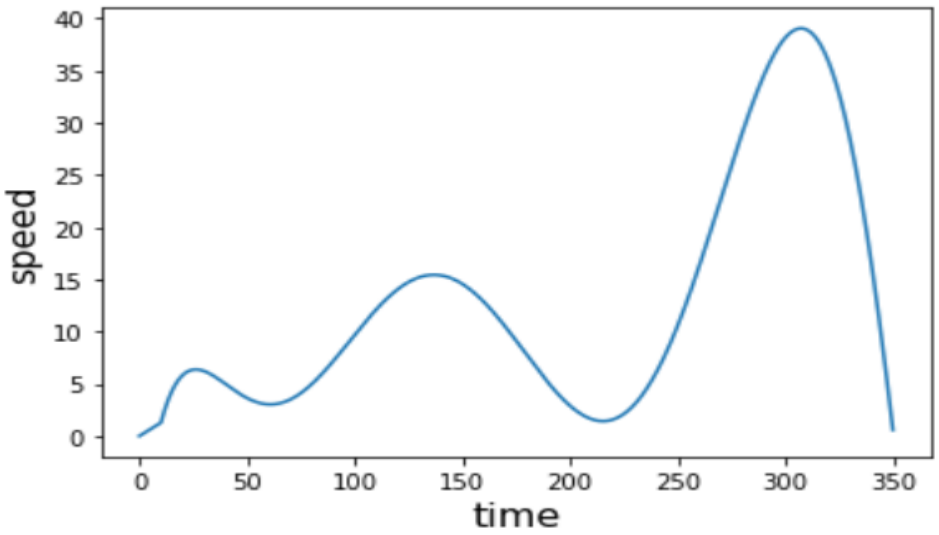



*Fig.16 Description of consumption battery related to speed and time*

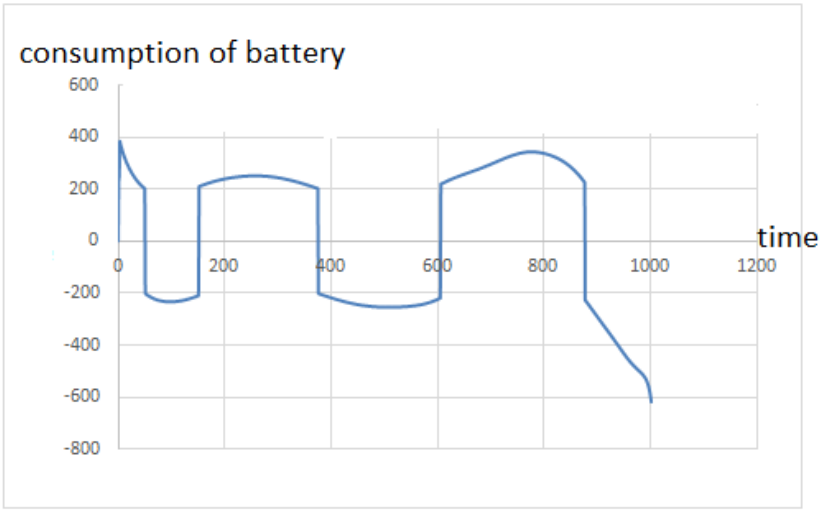



*Fig.17. Consumption of battery (Real trip)*

- **Approximation of linear regression :**

Regression analysis is the most commonly applied statistical method of all scientific fields. The extensive utility incurs continuous investigations to give various interpretations, extensions, and computing algorithms for the development and formulation of empirical models. General guidelines and fundamental principles on regression analysis have been well documented in the standard texts of Cohen et al., Kutner et al., and Montgomery, Peck, and Vining, among others. Among the methodological issues and statistical implications of regression analysis, model

adequacy and validity represent two vital aspects for justifying the usefulness of the underlying regression model. In the process of model selection, residual analysis and diagnostic checking are employed to identify influential observations, leverage, outliers, multicollinearity, and other lack of fit problems. Alternatively, model validation refers to the plausibility and generalizability of the regression function in terms of the stability and suitability of the regression coefficients [42].

The statistical inferences for the regression coefficients are based on the conditional distribution of the continuous predictors. However, unlike the fixed factor configurations and treatment levels in analysis of variance (ANOVA) and other experimental designs, the continuous measurements of the predictor variables in regression studies are typically available only after the data has been collected. For advance planning research design, the distribution and power functions of the test procedure need to be appraised over possible values of the predictors. Thus, it is important to recognize the stochastic nature of the predictor variables. The fundamental differences between fixed and random models have been explicated in Binkley and Abbot, Cramer and Appelbaum, Sampson, and Shaffer. Despite the complexity associated with the unconditional properties of the test procedure, the inferential procedures are the same under both fixed and random formulations. Hence, the usual rejection rule and critical value remain unchanged. The distinction between the two modeling approaches becomes critical for power analysis and sample size planning [42].

This is related to real trip data and linear regression (Fig.18.), to find the approximation linear about it in Fig.19.

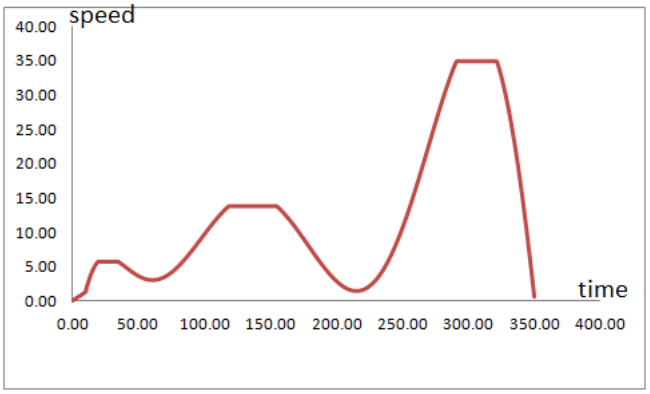



*Fig.18. Consumption of battery using linear approximation (Real trip)*

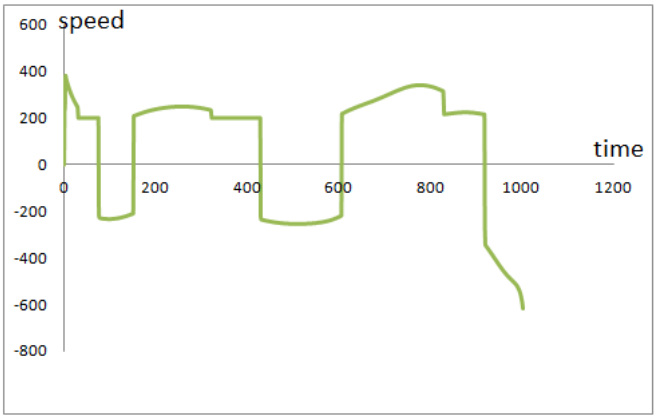



*Fig.19. Approximation.optimization for linear regression*

Polynomial functions are the most easiest and commonly used mathematical equation. It can be expressed in terms of a polynomial. The polynomial equation is used to represent the polynomial function. Generally, a polynomial is denoted as P(x). The greatest exponent of the variable P(x) is known as the degree of a polynomial. It is important to understand the degree of a polynomial as it describes the behavior of function P(x) when the value of x gets enlarged. The domain of polynomial functions is entirely real numbers (R).

Polynomial function is usually represented in the following way:

$a_n k^n + a_{n-1} k^{n-1} + \ldots + a_2 k^2 + a_1 k + a_0$, then for $k \gg 0$ or $k \ll 0$, $P(k) \approx a_n k^n$.

Hence, the polynomial functions reach power functions for the largest values of their variables.

**Types of Polynomial Function**

Some of the different types of polynomial functions on the basis of its degrees are given below:

- Constant Polynomial Function: A constant polynomial function is a function whose value does not change. It remains the same and also it does not include any variables.
- Zero Polynomial Function: Polynomial functions with a degree of 1 are known as Linear Polynomial functions.
- Linear Polynomial Function: Polynomial functions with a degree of 1 are known as Linear Polynomial functions.
- Quadratic Polynomial Function: Polynomial functions with a degree of 2 are known as Quadratic Polynomial functions.
- Cubic Polynomial Function: Polynomial functions with a degree of 3 are known as Cubic Polynomial functions.
- Quartic Polynomial Function: Polynomial functions with a degree of 4 are known as Quartic Polynomial functions.

To conclude in this chapter the optimization/approximation of real data concerning consumption of battery, I recommand the part [400-600] where is the stability and optimization of speed in consumption of battery. From [600-800] there is an approximation/fitting data.

# Conclusion

During the thesis, a computer model of the power system of an electric car was developed using the Python programming language.

Modern methods of data analysis are used to solve the problem of analysis and storage of input data.

In this project  presents a dynamic recurrent artificial neural network, based in the estimation of battery pack of a electric vehicle.

I can say that with the help of these methods and the obtained results we can proposed to the user change the energy consumption of cars  with the help of recommendations provided by the neural network.

Successful testing of the developed computer model of the power consumption system of the electric car was performed.

During the development of the project, a lot of experience was gained in designing computer models, as well as in development and programming in Python.

I can conclude that this simple demonstration of consumption battery energy cars by using statistical model represent for me good way to make more research in this topic using the Artificial intelligence and machine learning.

# References

1- Consumption Estimation Models for Electric Vehicles/Wang, J.B., Liu, K.,Yamamoto, T. and Morikawa, T. Improving estimation accuracy for electric vehicle energy consumption considering the effects of ambient temperature. Energy Procedia, 2017, 105, pp.2904-2909.

2- Quoc V Le et al. A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. Google Brain, pages 1{20, 2015

3- Chaochun Liu, Yaliang Li, Hongliang Fei, and Ping Li. Deep skip-gram networks for text classi_cation. In Proceedings of the 2019 SIAM International Conference on Data Mining, pages 145{153. SIAM, 2019.

4- Jonathan Lorraine and David Duvenaud. Stochastic hyper parameter optimization through hypernetworks. arXiv preprint arXiv:1802.09419, 2018.

5- Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Du_y, et al. Evolving deep neural networks. In Arti_cial Intelligence in the Age of Neural Networks and Brain Computing, pages 293{312. Elsevier, 2019. doi: 10.1016/B978-0-12-815480-9.00015-3.

6- Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timo-thy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pages 1928{1937, 2016.

7- Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345{1359, 2009. doi:10.1109/TKDE.2009.191.

8- Abhishek Panigrahi, Yueru Chen, and C-C Jay Kuo. Analysis on gradient propagation in batch normalized residual networks. arXiv preprint arXiv:1812.00342, 2018.

9- Ruslan Salakhutdinov and Geo_rey Hinton. Semantic hashing. International Journal of Approximate Reasoning, 50(7):969{978, 2009. doi: 10.1016/j.ijar.2008.11.006.

10- Bernhard Scholkopf and Alexander J Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2001.

12- Badin, F.; le Berr, F.; Briki, H.; Petit, M.; Magand, S.; Condemine, E. Evaluation of EVs energy consumption influencing factors. In Proceedings of the 2013 World Electric Vehicle Symposium and Exhibition (EVS27), Barcelona, Spain, 17–20 November 2013; pp. 1–12.

12- https://scitechdaily.com/deep-learning-ai-explained-neural-networks/

13- https://www.hindawi.com/journals/ace/2021/5571271/

14- https://indatalabs.com/blog/face-recognition-technology-industries

15- https:/skillrary.us/blogs/read/neural-networks-an-overview

16- https://mgubaidullin.github.io/deeplearning4j-docs/neuralnet-overview

17- https://themainstreamseer.blogspot.com/2011/10/why-neural-net-models-are-great-at.html

18- de Cauwer, C.; Maarten, M.; Coosemans, T.; van Mierlo, J.; Heyvaert, S. Electric vehicle use and energy consumption based on real-world electric vehicle fleet trip and charge data and its impact on existing EV research models. In Proceedings of the International Electric Vehicle Symposium and Exhibition, Kintex, Korea, 3–6 May 2015; pp. 1–11.

19- Lv, C.; Zhang, J.; Li, Y.; Yuan, Y. Mechanism analysis and evaluation methodology of regenerative braking contribution to energy efficiency improvement of electrified vehicles. Energy Convers. Manag. 2015, 92, 469–482.

20- Smuts M, Scholtz B, Wesson J. A critical review of factors influencing the remaining driving range of electric vehicles. IEEE 2017 1st International Conference on Next Generation Computing Applications (NextComp) 2017;196-201.

21- https://techflaunt.com/how-does-artificial-intelligence-work/

22- https://codinghero.ai/6-major-branches-of-artificial-intelligence-explained-to-kids/

23- https://www.edureka.co/blog/machine-learning-and-big-data/

24- https://www.guru99.com/reinforcement-learning-tutorial.html

25- https://faisal-kushha.github.io/reading-notes/401class16.html

26- https://www.javatpoint.com/machine-learning-naive-bayes-classifier

27- https://www.unite.ai/what-is-a-decision-tree/

28- https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_quick_guide.htm

29- https://towardsdatascience.com/11-most-common-machine-learning-algorithms-explained-in-a-nutshell-cc6e98df93be

30- https://www.edureka.co/blog/artificial-intelligence-algorithms/

31- https://www.linkedin.com/pulse/k-means-clustering-its-real-use-case-security-domain-duble-gurjar-

32- https://evpedia.net.au/2021/10/03/electric-cars-and-batteries-how-will-the-world-produce-enough/

33- https://www.myev.com/research/ev-101/how-long-should-an-electric-cars-battery-last

34- https://ave.dee.isep.ipp.pt/~mjf/act_lect/SIAUT/Trabalhos%202011-12/SIAUT_2011-12_ElectricVehicleCharging.pdf

35- https://legacy.python.org/doc/essays/blurb/

36- https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/

37- https://lmsapi.plagiat.pl/report/?shareId=e2c22c79-a19b-4af7-816c-dbf2a89f2d9d&language=uk

38- https://blog.csdn.net/cunzai1985/article/details/108751517

39- https://machinelearningmastery.com/recurrent-neural-network-algorithms-for-deep-learning/

40- https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

41- https://danijar.com/tips-for-training-recurrent-neural-networks/

42- https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/s12874-019-0697-9

43- Kernel Regression/Advanced Methods for Data Analysis (36-402/36-608)/ Spring 2014

44- IEA, 2019. Global Electric Vehicle Outlook 2019. International Energy Agency, Paris, France; 2019.

45- Smuts M, Scholtz B, Wesson J. A critical review of factors influencing the remaining driving range of electric vehicles. IEEE 2017 1st International Conference on Next Generation Computing Applications (NextComp) 2017;196- 201.

46- Zhang R, Yao E. Eco-driving at signalised intersections for electric

vehicles. IET Intelligent Transport Systems 2015;9(5):488-97.

47- Consumption Estimation Models for Electric Vehicles/Wang, J.B., Liu, K., Yamamoto, T. and Morikawa, T. Improving estimation accuracy for electric vehicle energy consumption considering the effects of ambient temperature. Energy Procedia, 2017, 105, pp.2904-2909.

48- Quoc V Le et al. A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. Google Brain, pages 1{20, 2015 - Chaochun Liu, Yaliang Li, Hongliang Fei, and Ping Li. Deep skip-gram networks for text classi_cation. In Proceedings of the 2019 SIAM International Conference on Data Mining, pages 145{153. SIAM, 2019.

49- Jonathan Lorraine and David Duvenaud. Stochastic hyperparameter optimization through hypernetworks. arXiv preprint arXiv:1802.09419, 2018.

50- Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Du_y, et al. Evolving deep neural networks. In Arti_cial Intelligence in the Age of Neural Networks and Brain Computing, pages 293{312. Elsevier, 2019. doi: 10.1016/B978-0-12-815480-9.00015-3.

51- Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timo thy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pages 1928{1937, 2016.

52- Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345{1359, 2009. doi: 10.1109/TKDE.2009.191.

53- Abhishek Panigrahi, Yueru Chen, and C-C Jay Kuo. Analysis on gradient propagation in batch normalized residual networks. arXiv preprint arXiv:1812.00342, 2018.

54- Ruslan Salakhutdinov and Geo_rey Hinton. Semantic hashing. International Journal of Approximate Reasoning, 50(7):969{978, 2009. doi: 10.1016/j.ijar.2008.11.006.

55- Bernhard Scholkopf and Alexander J Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2001.

56- George AF Seber and Alan J Lee. Linear regression analysis, volume 329. John Wiley & Sons, 2012.

57- Pulkit Sharma. Top 5 deep learning frameworks, their applications, and compar isons!, May 2019.

58- Toshihiro Takahashi. Statistical max pooling with deep learning, July 3 2018. US Patent 10,013,644.

59- Bhiksha Wang, HaohanandRaj. On the origin of deep learning. arXiv preprint arXiv:1702.07800, 2017.

60- Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. Insights into imaging, 9(4):611{629, 2018. doi: 10.1007/s13244-018-0639-9.

61- Alma, Ö. G. (2011). Comparison of Robust Regression Methods in Linear Regression. International Journal of Contemporary Mathematical Sciences, 6(9), 409-421.

62- Baltagi, Badi H. (2013) Econometric Analysis of Panel Data, 5th ed., John Wiley and Sons.

63- Dodge, Y. (2008). The Concise Encyclopedia of Statistics. Springer Science & Business Media.

64- Doornik, J. A. (2011). Robust Estimation Using Least Trimmed Squares, Institute for Economic Modelling, Oxford Martin School, and Economics Department, University of Oxford, UK.

# Annex-A

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) Магістр
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 –Комп'ютерна інженерія

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри теоретичної та прикладної системотехніки**
**_____д.т.н., проф. Шматков С. І.**

«____» _____2020року

**З А В Д А Н Н Я**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

_____ БЕНХАДДУ Мохаммед Амін _____
(прізвище, ім'я, по батькові студента)

1. Тема роботи « Модель комп'ютерної системи електропостачання електромобіля»
_____

керівник роботи ХРУСЛОВ Максим Михайлович, кандидат фіз.-мат. наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "___" _____20__ року №_____

2. Строк подання студентом роботи_____

3. Перелік питань, які потрібно розробити)
1. Пошук та вивчення існуючих рішень, аналіз їх переваг та недоліків.
2. Ознайомлення та вивчення відповідної літератури.
3. Пошук або створення відповідних методів для вирішення задачі щодо створення компютерної моделі системи енергоспоживання електромобіля
4. Проектування архітектури та структури компютерної моделі системи енергоспоживання електромобіля
5. Розробка системи аналізу та зберігання вхідних даних.
6. Тестування.
7. Застосування апроксимаційних алгоритмів для аналізу данних реальної подорожі.
8. Надати спостереження та поради щодо споживання батареї
9. Написання технічної документації.

4. План роботи

| № з/п | Назви етапів роботи | Термін виконання етапів роботи |
|---|---|---|
| 1 | Пошук та вивчення існуючих рішень, аналіз їх переваг та недоліків. | |
| 2 | Ознайомлення та вивчення відповідної літератури. | |
| 3 | Пошук або створення відповідних методів для вирішення задачі щодо створення компютерної моделі системи енергоспоживання електромобіля | |
| 4 | Проектування архітектури та структури компютерної моделі системи енергоспоживання електромобіля | |
| 5 | Розробка системи аналізу та зберігання вхідних даних | |
| 6 | Тестування. | |
| 7 | Застосування апроксимаційних алгоритмів для аналізу данних реальної подорожі. | |
| 8 | Надати спостереження та поради щодо споживання батареї | |
| 9 | Написання технічної документації. | |
| | | |

5. Дата видачі завдання_____

Студент _БЕНХАДДУ М. А. ___ _____
ініціали, прізвище підпис

Керівник роботи ___ХРУСЛОВ М.М.____ _____
ініціали, прізвище підпис

# Annex-B

# Code listing of the developed computer model.

# (Fragments)

Entrée [19]:
```
X=data["speed"]

Y=data["accel=1, 2, 3, 4, 5, 6, 7, 8"]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)

print('Training Data Count: {}'.format(X_train.shape[0]))
print('Testing Data Count: {}'.format(X_test.shape[0]))
```

Entrée [ ]:
```
plt.scatter(X, y, )
plt.plot(X, regressor.predict(X), )
plt.title('accelerations vs speed')
plt.xlabel('speed')
plt.ylabel('accelerations')
plt.show()
```

Entrée [8]:
```
# Model initialization
model = LinearRegression(normalize=True)
print(model.normalize)
model.fit(X,y)
model = LinearRegression().fit(x, y)
print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)
```

```
True
```

Entrée [ ]:
```
# with statistics models
X = sm.add_constant(X) # adding a constant

model = sm.OLS(Y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
print(print_model)
```

Entrée [ ]:
```
# model evaluation
rmse = mean_squared_error(y, y_predicted)
r2 = r2_score(y, y_predicted)
```

Entrée [ ]:
```
# printing values
print('Slope:' ,regression_model.coef_)
print('Intercept:', regression_model.intercept_)
print('Root mean squared error: ', rmse)
print('R2 score: ', r2)
```

Entrée [2]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import pylab as pl

sns.set()
```

Entrée [4]:
```
import pandas as pd
data = pd.read_csv ('C:\\Users\\HP\\Desktop\\data project\\data.csv')

print (data)
```

Entrée [19]:
```
X=data["speed"]

Y=data["accel=1, 2, 3, 4, 5, 6, 7, 8"]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)

print('Training Data Count: {}'.format(X_train.shape[0]))
print('Testing Data Count: {}'.format(X_test.shape[0]))
```

Entrée [8]:
```
# Model initialization
model = LinearRegression(normalize=True)
print(model.normalize)
model.fit(X,y)
model = LinearRegression().fit(x, y)
print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)
```

Entrée [10]:
```
data = load_data() #Loading boston datasets
x = data.data # Creating Regression Design Matrix
y = data.target # Creating target dataset
model = LinearRegression(normalize=True)
print(model.normalize)
model.fit(X,y)
model = LinearRegression().fit(x, y)
```

Entrée [ ]:
```
y_pred = regressor.predict(X)
```

Entrée [ ]:
```
plt.scatter(X, y, )
plt.plot(X, regressor.predict(X), )
plt.title('accelerations vs speed')
plt.xlabel('speed')
plt.ylabel('accelerations')
plt.show()
```

Entrée [19]:
```
X=data["speed"]

Y=data["accel=1, 2, 3, 4, 5, 6, 7, 8"]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)

print('Training Data Count: {}'.format(X_train.shape[0]))
print('Testing Data Count: {}'.format(X_test.shape[0]))
```

## Screen about algorithms for RNN

Entrée [ ]:
```
from sklearn.neural_network import MLPClassifier
 X = [[0., 0.], [1., 1.]]
 y = [0, 1]
 clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
                     hidden_layer_sizes=(5, 2), random_state=1)

 clf.fit(X, y)
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(5, 2), random_state=1, solver='lbfgs')

 clf.predict([[2., 2.], [-1., -2.]])
array([1, 0])

[coef.shape for coef in clf.coefs_]
[(2, 5), (5, 2), (2, 1)]
```

Entrée [ ]:
```
from sklearn.neural_network import MLPRegressor
 from sklearn.datasets import make_regression
 from sklearn.model_selection import train_test_split
 X, y = make_regression(n_samples=200, random_state=1)
 X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=1)
 regr = MLPRegressor(random_state=1, max_iter=500).fit(X_train, y_train)
 regr.predict(X_test[:2])
array([-0.9..., -7.1...])
 regr.score(X_test, y_test)
X = [[0., 0.], [1., 1.]]
 y = [0, 1]
 clf = MLPClassifier(hidden_layer_sizes=(15,), random_state=1, max_iter=1, warm_start=True)
 for i in range(10):
clf.fit(X, y)
     # additional monitoring / inspection
MLPClassifier(...
```

```
for epoch in range(nepoch):
    # check loss on train
    loss = 0.0

    # do a forward pass to get prediction
    for i in range(Y.shape[0]):
        x, y = X[i], Y[i]                    # get input, output values of each record
        prev_s = np.zeros((hidden_dim, 1))   # here, prev-s is the value of the previous activation of hidden layer;

        for t in range(T):
            new_input = np.zeros(x.shape)    # we then do a forward pass for every timestep in the sequence
            new_input[t] = x[t]              # for this, we define a single input for that timestep
            mulu = np.dot(U, new_input)
            mulw = np.dot(W, prev_s)
            add = mulw + mulu
            s = sigmoid(add)
            mulv = np.dot(V, s)
            prev_s = s

    # calculate error
        loss_per_record = (y - mulv)**2 / 2
        loss += loss_per_record
    loss = loss / float(y.shape[0])
```

```
                # check loss on val
                    val_loss = 0.0
                    for i in range(Y_val.shape[0]):
                        x, y = X_val[i], Y_val[i]
                        prev_s = np.zeros((hidden_dim, 1))
                        for t in range(T):
                            new_input = np.zeros(x.shape)
                            new_input[t] = x[t]
                            mulu = np.dot(U, new_input)
                            mulw = np.dot(W, prev_s)
                            add = mulw + mulu
                            s = sigmoid(add)
                            mulv = np.dot(V, s)
                            prev_s = s

                        loss_per_record = (y - mulv)**2 / 2
                        val_loss += loss_per_record
                    val_loss = val_loss / float(y.shape[0])

                    print('Epoch: ', epoch + 1, ', Loss: ', loss, ', Val Loss: ', val_loss)
```

```
# train model
   for i in range(Y.shape[0]):
       x, y = X[i], Y[i]

       layers = []
       prev_s = np.zeros((hidden_dim, 1))
       dU = np.zeros(U.shape)
       dV = np.zeros(V.shape)
       dW = np.zeros(W.shape)

       dU_t = np.zeros(U.shape)
       dV_t = np.zeros(V.shape)
       dW_t = np.zeros(W.shape)

       dU_i = np.zeros(U.shape)
       dW_i = np.zeros(W.shape)

       # forward pass
       for t in range(T):
           new_input = np.zeros(x.shape)
           new_input[t] = x[t]
           mulu = np.dot(U, new_input)
           mulw = np.dot(W, prev_s)
           add = mulw + mulu
           s = sigmoid(add)
           mulv = np.dot(V, s)
           layers.append({'s':s, 'prev_s':prev_s})
           prev_s = s
```

```
# derivative of pred
       dmulv = (mulv - y)

       # backward pass
       for t in range(T):
           dV_t = np.dot(dmulv, np.transpose(layers[t]['s']))
           dsv = np.dot(np.transpose(V), dmulv)

           ds = dsv
           dadd = add * (1 - add) * ds

           dmulw = dadd * np.ones_like(mulw)

           dprev_s = np.dot(np.transpose(W), dmulw)


           for i in range(t-1, max(-1, t-bptt_truncate-1), -1):
               ds = dsv + dprev_s
               dadd = add * (1 - add) * ds

               dmulw = dadd * np.ones_like(mulw)
               dmulu = dadd * np.ones_like(mulu)

               dW_i = np.dot(W, layers[t]['prev_s'])
               dprev_s = np.dot(np.transpose(W), dmulw)
```

```
                new_input = np.zeros(x.shape)
                new_input[t] = x[t]
                dU_i = np.dot(U, new_input)
                dx = np.dot(np.transpose(U), dmulu)

                dU_t += dU_i
                dW_t += dW_i

            dV += dV_t
            dU += dU_t
            dW += dW_t
```

```
if dU.max() > max_clip_value:
                dU[dU > max_clip_value] = max_clip_value
            if dV.max() > max_clip_value:
                dV[dV > max_clip_value] = max_clip_value
            if dW.max() > max_clip_value:
                dW[dW > max_clip_value] = max_clip_value


            if dU.min() < min_clip_value:
                dU[dU < min_clip_value] = min_clip_value
            if dV.min() < min_clip_value:
                dV[dV < min_clip_value] = min_clip_value
            if dW.min() < min_clip_value:
                dW[dW < min_clip_value] = min_clip_value
```

```
        # update
        U -= learning_rate * dU
        V -= learning_rate * dV
        W -= learning_rate * dW
```

Entrée [ ]:

```
preds = []
for i in range(Y.shape[0]):
    x, y = X[i], Y[i]
    prev_s = np.zeros((hidden_dim, 1))
    # Forward pass
    for t in range(T):
        mulu = np.dot(U, x)
        mulw = np.dot(W, prev_s)
        add = mulw + mulu
        s = sigmoid(add)
        mulv = np.dot(V, s)
        prev_s = s

    preds.append(mulv)

preds = np.array(preds)

plt.plot(preds[:, 0, 0], 'g')
plt.plot(Y[:, 0], 'r')
plt.show()
from sklearn.metrics import mean_squared_error

math.sqrt(mean_squared_error(Y_val[:, 0] * max_val, preds[:, 0, 0] * max_val))
```