Project Report

**On**

Machine Learning For Weather Prediction

### By

**Vinayak B Kamath (RA1931005020184)**

**Submitted to the**

**DEPARTMENT OF COMPUTER SCIENCE**

***Under the guidance of***

## DR. S UMARANI

**Dr , Department of Computer Science**

***Submitted in partial fulfillment of the requirement for the award of the degree of***

# BACHELOR OF COMPUTER SCIENCE



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

### Ramapuram, Chennai.

**April 2021**

**COLLEGE OF SCIENCE & HUMANITIES**

**Ramapuram, Chennai.**

**Department of Computer Science**

**BONAFIDE CERTIFICATE**

Certified that this project report titled **“Machine Learning for Weather Prediction ”** is the bonafide work of **Vinayak B Kamath (RA1931005020184)** who carried out the Mini project work done under my supervision

Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**Signature of Internal Guide Signature of Head of the Department**

**Signature of External Examiner**

Machine Learning For Weather Prediction

**ABSTRACT**

Weather is an important aspect of our society. It forms the backbone of society, and many of society’s functions depend on the weather. Weather decides our food yield, it decides whether societal operations continue or not, it also alters our mood. However, weather is very fickle and can easily change from one moment to another. The following project is an attempt at trying to predict the weather through computerized models. A database of weather events and related variables are taken and we use computerized machine learning models to calculate the probability of occurrence of certain events.

**ACKNOWLEDGEMENT**

I extend my sincere gratitude to the Chancellor **Dr. T.R. PACHAMUTHU** and to Chairman **Dr. R. SHIVAKUMAR** of SRM Institute of Science and Technology, Ramapuram, Chennai for providing me the opportunity to pursue the BSC (CS) degree at this University.

I express my sincere gratitude to **Dr. C. SUNDAR**, Principal**(S&H)**, SRM IST, Ramapuram for his support and encouragement for the successful completion of the project.

I record my sincere thanks to **Dr. J. DHILIPAN M.Sc., MBA., M.Phil., Ph.D., Head & Vice Principal (Admin, S&H)**, SRM IST, Ramapuram for his continuous support and keen interest to make this project a successful one.

I record my sincere thanks to **Dr. N. ASOKAN Ph.D., Vice Principal (Academic, S&H)**, SRM IST, Ramapuram for his encouragement and keen interest to make this project a successful one.

I would further like to express our gratitude to **Dr. S. UMARANI MCA., Ph.D., Professor & Head of the Department**, SRM IST, Ramapuram for her commendable support and encouragement for the completion of project.

I find no word to express profound gratitude to my guide **DR. S Umarani** **Ph.D., Head Of Department** Department of Computer Science, SRM IST, Ramapuramfor her kind co-operation and encouragement which helped us for the completion of this project.

I thank the almighty who has made this possible. Finally, I thank my beloved family member and friend for their motivation, encouragement and cooperation in all aspect which led me to the completion of this project.

**Vinayak B Kamath (RA1931005020184)**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT ACKNOWLEDGEMENT LIST OF TABLES** | **3**  **4**  **5** |
| **CHAPTERS** | **TITLE** | **PAGE NO.** |
| **1** | **INTRODUCTION**  1.1 PROJECT INTRODUCTION | 7 |
| **2** | **WORKING ENVIRONMENT**   * 1. HARDWAREREQUIREMENT   2. SOFTWAREREQUIREMENT   3. SYSTEMSOFTWARE | 8  8  9 |
| **3** | **SYSTEM ANALYSIS**   * 1. FEASIBILITY STUDY   2. EXISTINGSYSTEM   3. DRAWBACKS OF EXISTINGSYSTEM   4. PROPOSED SYSTEM   5. BENEFITS OF PROPOSEDSYSTEM   6. SCOPE OF THEPROJECT | 31  33  33  34  34  35 |
| 4 | **SYSTEM DESIGN**   * 1. DATA FLOWDIAGRAM | 37 |
| **5** | **PROJECT DESCRIPTION**   * 1. OBJECTIVE   2. MODULEDESCRIPTION | 38  38 |
| **6** | **SYSTEM TESTING**   * 1. TESTINGDEFINITON   2. TESTINGOBJECTIVE   3. TYPES OFTESTING | 39  40  41 |
| **7** | **CONCLUSION**   * 1. SUMMARY   2. FUTURE ENHANCEMENTS | 42  42 |
| **8** | **APPENDIX**   * 1. SCREENSHOTS   2. CODING | 43  46 |
| **09** | **BIBLIOGRAPHY AND REFERENCES** | 51 |

**CHAPTER-1**

**INTRODUCTION**

**1.1 PROJECT INTRODUCTION**

To be able to predict weather events, we first need a dataset. The dataset would

contain various parameters such as Temperature, Humidity, Visibility, Atmospheric

Pressure, Dew Point, Wind Speed, etc. We would then use various classification algorithms

like K Nearest Neighbors, Support Vector Machines, etc

The time it takes to train the model is calculated, along with it’s accuracy. The model with the best accuracy is chosen to be used as the backend of our flask – based app.

Then, an app using flask is created, that takes a csv file containing the parameters for prediction, after which the prediction for the weather event is given.

## CHAPTER-2

**WORKING ENVIRONMENT**

* 1. **HARDWAREREQUIREMENT**
* SYSTEM : Pentium IV 2.4GHz, Google Colab
* HARDDISK : 500GB
* MONITOR : 15 VGA color
* MOUSE :Logitech
* RAM :2GB

## SOFTWAREREQUIREMENT

|  |  |  |
| --- | --- | --- |
| • | Operating system : | Windows 7 |
| • | Frontend : | HTML, CSS |
| • | Coding Language : | python |
| • | Back-End : | Flask, sklearn |
| **.** | Web Server : | Apache Tomcat, Cloud Server |

* 1. **SYSTEM SOFTWARE**

**2.3.1 PYTHON**

**Python** is a [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Its design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with the use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) and [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small- and large-scale projects.[[30]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-7-30)

Python is [dynamically-typed](https://en.wikipedia.org/wiki/Type_system#DYNAMIC) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [structured](https://en.wikipedia.org/wiki/Structured_programming) (particularly [procedural](https://en.wikipedia.org/wiki/Procedural_programming)), object-oriented and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). It is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).[[31]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-About-31)[[32]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-32)

[Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) began working on Python in the late 1980s as a successor to the [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)) and first released it in 1991 as Python 0.9.0.[[33]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-33) Python 2.0 was released in 2000 and introduced new features such as [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension), [cycle-detecting](https://en.wikipedia.org/wiki/Cycle_detection) garbage collection, [reference counting](https://en.wikipedia.org/wiki/Reference_counting), and [Unicode](https://en.wikipedia.org/wiki/Unicode) support. Python 3.0, released in 2008, was a major revision that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility) with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.[[34]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-34)

Python consistently ranks as one of the most popular programming languages.

Python was conceived in the late 1980s[[39]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-venners-interview-pt-1-39) by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) at [Centrum Wiskunde & Informatica](https://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica) (CWI) in the [Netherlands](https://en.wikipedia.org/wiki/Netherlands) as a successor to the [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)), which was inspired by [SETL](https://en.wikipedia.org/wiki/SETL),[[40]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-12-40) capable of [exception handling](https://en.wikipedia.org/wiki/Exception_handling) and interfacing with the [Amoeba](https://en.wikipedia.org/wiki/Amoeba_(operating_system)) operating system.[[11]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-faq-created-11) Its implementation began in December 1989.[[41]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-timeline-of-python-41) Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "[benevolent dictator for life](https://en.wikipedia.org/wiki/Benevolent_dictator_for_life)", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker.[[42]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-lj-bdfl-resignation-42) In January 2019, active Python core developers elected a five-member Steering Council to lead the project.[[43]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-43)[[44]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-44)

Python 2.0 was released on 16 October 2000, with many major new features.[[45]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-newin-2.0-45) Python 3.0, released on 3 December 2008, with many of its major features [backported](https://en.wikipedia.org/wiki/Backporting) to Python 2.6.x[[46]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-pep-3000-46) and 2.7.x. Releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.[[47]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-47)

Python 2.7's [end-of-life](https://en.wikipedia.org/wiki/End-of-life_(product)) was initially set for 2015, then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.[[48]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-48)[[49]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-49) No further security patches or other improvements will be released for it.[[50]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-50)[[51]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-51) With Python 2's [end-of-life](https://en.wikipedia.org/wiki/End-of-life_(product)), only Python 3.6.x[[52]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-52) and later where supported, and later support for 3.6 also dropped.

In 2022, Python 3.10.4 and 3.9.12 were expedited[[53]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-53) and so were older releases including 3.8.13, and 3.7.13 because of many security issues in 2022.[[54]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-54) In 2021, Python 3.9.2 and 3.8.8 were also expedited[[55]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-55) as all versions of Python (including 2.7[[56]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-56)) had security issues leading to possible [remote code execution](https://en.wikipedia.org/wiki/Remote_code_execution)[[57]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-57) and [web cache poisoning](https://en.wikipedia.org/wiki/Cache_poisoning).[[58]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-58)

Python 3.6 (and every older release), is no longer supported as of 2021.

Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support functional programming and [aspect-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming) (including by [metaprogramming](https://en.wikipedia.org/wiki/Metaprogramming)[[59]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-13-59) and [metaobjects](https://en.wikipedia.org/wiki/Metaobject) [magic methods] ).[[60]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-14-60) Many other paradigms are supported via extensions, including [design by contract](https://en.wikipedia.org/wiki/Design_by_contract)[[61]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-15-61)[[62]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-16-62) and [logic programming](https://en.wikipedia.org/wiki/Logic_programming).[[63]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-17-63)

Python uses [dynamic typing](https://en.wikipedia.org/wiki/Dynamic_typing), and a combination of [reference counting](https://en.wikipedia.org/wiki/Reference_counting) and a cycle-detecting garbage collector for [memory management](https://en.wikipedia.org/wiki/Memory_management).[[64]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-Reference_counting-64) It uses dynamic [name resolution](https://en.wikipedia.org/wiki/Name_resolution_(programming_languages)) ([late binding](https://en.wikipedia.org/wiki/Late_binding)), which binds method and variable names during program execution.

Its design offers some support for functional programming in the [Lisp](https://en.wikipedia.org/wiki/Lisp_(programming_language)) tradition. It has filter,mapandreduce functions; [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension), [dictionaries](https://en.wikipedia.org/wiki/Associative_array), sets, and [generator](https://en.wikipedia.org/wiki/Generator_(computer_programming)) expressions.[[65]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-59-65) The standard library has two modules (itertools and functools) that implement functional tools borrowed from [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)) and [Standard ML](https://en.wikipedia.org/wiki/Standard_ML).[[66]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-18-66)

Its core philosophy is summarized in the document *The*[*Zen of Python*](https://en.wikipedia.org/wiki/Zen_of_Python) (*PEP 20*), which includes [aphorisms](https://en.wikipedia.org/wiki/Aphorism) such as:[[67]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-PEP20-67)

* Beautiful is better than ugly.
* Explicit is better than implicit.
* Simple is better than complex.
* Complex is better than complicated.
* Readability counts.

Rather than building all of its functionality into its core, Python was designed to be highly [extensible](https://en.wikipedia.org/wiki/Extensibility) via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with [ABC](https://en.wikipedia.org/wiki/ABC_(programming_language)), which espoused the opposite approach.[[39]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-venners-interview-pt-1-39)

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to [Perl](https://en.wikipedia.org/wiki/Perl)'s "[there is more than one way to do it](https://en.wikipedia.org/wiki/There_is_more_than_one_way_to_do_it)" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" philosophy.[[67]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-PEP20-67) [Alex Martelli](https://en.wikipedia.org/wiki/Alex_Martelli), a [Fellow](https://en.wikipedia.org/wiki/Fellow) at the [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation) and Python book author, wrote: "To describe something as 'clever' is *not* considered a compliment in the Python culture."[[68]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-19-68)

Python's developers strive to avoid [premature optimization](https://en.wikipedia.org/wiki/Premature_optimization), and reject patches to non-critical parts of the [CPython](https://en.wikipedia.org/wiki/CPython) reference implementation that would offer marginal increases in speed at the cost of clarity.[[69]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-20-69) When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C; or use [PyPy](https://en.wikipedia.org/wiki/PyPy), a [just-in-time compiler](https://en.wikipedia.org/wiki/Just-in-time_compilation). [Cython](https://en.wikipedia.org/wiki/Cython) is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim for it to be fun to use. This is reflected in its name—a tribute to the British comedy group [Monty Python](https://en.wikipedia.org/wiki/Monty_Python)[[70]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-24-70)—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a [Monty Python sketch](https://en.wikipedia.org/wiki/Spam_(Monty_Python))) instead of the standard [foo and bar](https://en.wikipedia.org/wiki/Foobar).[[71]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-71)[[72]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-72)

A common [neologism](https://en.wikipedia.org/wiki/Neologism) in the Python community is *pythonic*, which has a wide range of meanings related to program style. "Pythonic" code may use Python idioms well, be natural or show fluency in the language, or conform with Python's minimalist philosophy and emphasis on readability. Code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.[[73]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-73)[[74]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-74)

**Limitations of Python**

#### **1. Slow Speed**

We discussed above that Python is an **interpreted** language and **dynamically-typed** language. The line by line execution of code often leads to **slow execution**.

The dynamic nature of Python is also responsible for the **slow speed**of Python because it has to do the extra work while executing code. So, Python is not used for purposes where speed is an important aspect of the project.

#### **2. Not Memory Efficient**

To provide simplicity to the developer, Python has to do a little tradeoff. The Python programming language uses a **large amount of memory**. This can be a disadvantage while building applications when we prefer memory optimization.

#### **3. Weak in Mobile Computing**

Python is generally used in **server-side programming**. We don’t get to see Python on the client-side or mobile applications because of the following reasons. Python is **not memory efficient** and it has **slow processing power** as compared to other languages.

#### **4. Database Access**

Programming in Python is **easy** and **stress-free**. But when we are interacting with the database, it lacks behind.

The Python’s database access layer is primitive and underdeveloped in comparison to the popular technologies like **JDBC** and **ODBC**.

Huge enterprises need smooth **interaction** of complex legacy data and Python is thus rarely used in enterprises.

#### **5. Runtime Errors**

As we know Python is a dynamically typed language so the data type of a variable can change anytime. A variable containing integer number may hold a string in the future, which can lead to **Runtime Errors**.

Therefore Python programmers need to perform thorough testing of the applications.

**2.3.2 Machine Learning**

**Machine learning** (**ML**) is the study of computer [algorithms](https://en.wikipedia.org/wiki/Algorithm) that can improve automatically through experience and by the use of data.[[1]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-1) It is seen as a part of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence). Machine learning algorithms build a model based on sample data, known as [training data](https://en.wikipedia.org/wiki/Training_data), in order to make predictions or decisions without being explicitly programmed to do so.[[2]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-2) Machine learning algorithms are used in a wide variety of applications, such as in medicine, [email filtering](https://en.wikipedia.org/wiki/Email_filtering), [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition), and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.[[3]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-tvt-3)

A subset of machine learning is closely related to [computational statistics](https://en.wikipedia.org/wiki/Computational_statistics), which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of [mathematical optimization](https://en.wikipedia.org/wiki/Mathematical_optimization) delivers methods, theory and application domains to the field of machine learning. [Data mining](https://en.wikipedia.org/wiki/Data_mining) is a related field of study, focusing on [exploratory data analysis](https://en.wikipedia.org/wiki/Exploratory_data_analysis) through [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning).[[5]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-5)[[6]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-6) Some implementations of machine learning use data and [neural networks](https://en.wikipedia.org/wiki/Neural_networks) in a way that mimics the working of a biological brain.[[7]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-7)[[8]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-8) In its application across business problems, machine learning is also referred to as [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics).

**Overview**

**Machine learning** (**ML**) is the study of computer [algorithms](https://en.wikipedia.org/wiki/Algorithm) that can improve automatically through experience and by the use of data.[[1]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-1) It is seen as a part of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence). Machine learning algorithms build a model based on sample data, known as [training data](https://en.wikipedia.org/wiki/Training_data), in order to make predictions or decisions without being explicitly programmed to do so.[[2]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-2) Machine learning algorithms are used in a wide variety of applications, such as in medicine, [email filtering](https://en.wikipedia.org/wiki/Email_filtering), [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition), and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.[[3]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-tvt-3)

A subset of machine learning is closely related to [computational statistics](https://en.wikipedia.org/wiki/Computational_statistics), which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of [mathematical optimization](https://en.wikipedia.org/wiki/Mathematical_optimization) delivers methods, theory and application domains to the field of machine learning. [Data mining](https://en.wikipedia.org/wiki/Data_mining) is a related field of study, focusing on [exploratory data analysis](https://en.wikipedia.org/wiki/Exploratory_data_analysis) through [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning).[[5]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-5)[[6]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-6) Some implementations of machine learning use data and [neural networks](https://en.wikipedia.org/wiki/Neural_networks) in a way that mimics the working of a biological brain.[[7]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-7)[[8]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-8) In its application across business problems, machine learning is also referred to as [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics).

**History**

The term *machine learning* was coined in 1959 by [Arthur Samuel](https://en.wikipedia.org/wiki/Arthur_Samuel), an American [IBMer](https://en.wikipedia.org/wiki/IBMer) and pioneer in the field of [computer gaming](https://en.wikipedia.org/wiki/Computer_gaming) and [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence).[[11]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-Samuel-11)[[12]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-12) Also the synonym *self-teaching computers* was used in this time period.[[13]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-cyberthreat-13)[[14]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-14) A representative book of the machine learning research during the 1960s was the Nilsson's book on Learning Machines, dealing mostly with machine learning for pattern classification.[[15]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-15) Interest related to pattern recognition continued into the 1970s, as described by Duda and Hart in 1973.[[16]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-16) In 1981 a report was given on using teaching strategies so that a [neural network](https://en.wikipedia.org/wiki/Neural_network) learns to recognize 40 characters (26 letters, 10 digits, and 4 special symbols) from a computer terminal.[[17]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-17)

[Tom M. Mitchell](https://en.wikipedia.org/wiki/Tom_M._Mitchell) provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."[[18]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-Mitchell-1997-18) This definition of the tasks in which machine learning is concerned offers a fundamentally [operational definition](https://en.wikipedia.org/wiki/Operational_definition) rather than defining the field in cognitive terms. This follows [Alan Turing](https://en.wikipedia.org/wiki/Alan_Turing)'s proposal in his paper "[Computing Machinery and Intelligence](https://en.wikipedia.org/wiki/Computing_Machinery_and_Intelligence)", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?".[[19]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-19)

Modern day machine learning has two objectives, one is to classify data based on models which have been developed, the other purpose is to make predictions for future outcomes based on these models. A hypothetical algorithm specific to classifying data may use computer vision of moles coupled with supervised learning in order to train it to classify the cancerous moles. A machine learning algorithm for stock trading may inform the trader of future potential predictions

**AI**

As a scientific endeavor, machine learning grew out of the quest for artificial intelligence. In the early days of AI as an [academic discipline](https://en.wikipedia.org/wiki/Discipline_(academia)), some researchers were interested in having machines learn from data. They attempted to approach the problem with various symbolic methods, as well as what was then termed "[neural networks](https://en.wikipedia.org/wiki/Neural_network)"; these were mostly [perceptrons](https://en.wikipedia.org/wiki/Perceptron) and [other models](https://en.wikipedia.org/wiki/ADALINE) that were later found to be reinventions of the [generalized linear models](https://en.wikipedia.org/wiki/Generalized_linear_model) of statistics.[[23]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-23) [Probabilistic reasoning](https://en.wikipedia.org/wiki/Probabilistic_reasoning) was also employed, especially in [automated medical diagnosis](https://en.wikipedia.org/wiki/Automated_medical_diagnosis).[[24]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-aima-24): 488

However, an increasing emphasis on the [logical, knowledge-based approach](https://en.wikipedia.org/wiki/GOFAI) caused a rift between AI and machine learning. Probabilistic systems were plagued by theoretical and practical problems of data acquisition and representation.[[24]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-aima-24): 488 By 1980, [expert systems](https://en.wikipedia.org/wiki/Expert_system) had come to dominate AI, and statistics was out of favor.[[25]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-changing-25) Work on symbolic/knowledge-based learning did continue within AI, leading to [inductive logic programming](https://en.wikipedia.org/wiki/Inductive_logic_programming), but the more statistical line of research was now outside the field of AI proper, in [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition) and [information retrieval](https://en.wikipedia.org/wiki/Information_retrieval).[[24]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-aima-24): 708–710, 755 Neural networks research had been abandoned by AI and [computer science](https://en.wikipedia.org/wiki/Computer_science) around the same time. This line, too, was continued outside the AI/CS field, as "[connectionism](https://en.wikipedia.org/wiki/Connectionism)", by researchers from other disciplines including [Hopfield](https://en.wikipedia.org/wiki/John_Hopfield), [Rumelhart](https://en.wikipedia.org/wiki/David_Rumelhart) and [Hinton](https://en.wikipedia.org/wiki/Geoff_Hinton). Their main success came in the mid-1980s with the reinvention of [backpropagation](https://en.wikipedia.org/wiki/Backpropagation).[[24]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-aima-24): 25

Machine learning (ML), reorganized as a separate field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to tackling solvable problems of a practical nature. It shifted focus away from the [symbolic approaches](https://en.wikipedia.org/wiki/Symbolic_artificial_intelligence) it had inherited from AI, and toward methods and models borrowed from statistics, [fuzzy logic](https://en.wikipedia.org/wiki/Fuzzy_logic), and [probability theory](https://en.wikipedia.org/wiki/Probability_theory).[[25]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-changing-25)

The difference between ML and AI is frequently misunderstood. ML learns and predicts based on passive observations, whereas AI implies an agent interacting with the environment to learn and take actions that maximize its chance of successfully achieving its goals.[[26]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-:3-26)

As of 2020, many sources continue to assert that ML remains a subfield of AI.[[27]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-:4-27)[[28]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-islr-28)[[25]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-changing-25) Others have the view that not all ML is part of AI, but only an 'intelligent subset' of ML should be considered AI.

**Data Mining**

Machine learning and [data mining](https://en.wikipedia.org/wiki/Data_mining) often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on *known* properties learned from the training data, [data mining](https://en.wikipedia.org/wiki/Data_mining) focuses on the [discovery](https://en.wikipedia.org/wiki/Discovery_(observation)) of (previously) *unknown* properties in the data (this is the analysis step of [knowledge discovery](https://en.wikipedia.org/wiki/Knowledge_discovery) in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, [ECML PKDD](https://en.wikipedia.org/wiki/ECML_PKDD) being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to *reproduce known* knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously *unknown* knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

**Optimization**

Machine learning also has intimate ties to [optimization](https://en.wikipedia.org/wiki/Mathematical_optimization): many learning problems are formulated as minimization of some [loss function](https://en.wikipedia.org/wiki/Loss_function) on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set of examples).

**Generalization**

The difference between optimization and machine learning arises from the goal of generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples. Characterizing the generalization of various learning algorithms is an active topic of current research, especially for [deep learning](https://en.wikipedia.org/wiki/Deep_learning) algorithms.

**Statistics**

Machine learning and [statistics](https://en.wikipedia.org/wiki/Statistics) are closely related fields in terms of methods, but distinct in their principal goal: statistics draws population [inferences](https://en.wikipedia.org/wiki/Statistical_inference) from a [sample](https://en.wikipedia.org/wiki/Sample_(statistics)), while machine learning finds generalizable predictive patterns.[[32]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-32) According to [Michael I. Jordan](https://en.wikipedia.org/wiki/Michael_I._Jordan), the ideas of machine learning, from methodological principles to theoretical tools, have had a long pre-history in statistics.[[33]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-mi_jordan_ama-33) He also suggested the term [data science](https://en.wikipedia.org/wiki/Data_science) as a placeholder to call the overall field.[[33]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-mi_jordan_ama-33)

[Leo Breiman](https://en.wikipedia.org/wiki/Leo_Breiman) distinguished two statistical modeling paradigms: data model and algorithmic model,[[27]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-:4-27) wherein "algorithmic model" means more or less the machine learning algorithms like [Random forest](https://en.wikipedia.org/wiki/Random_forest).

Some statisticians have adopted methods from machine learning, leading to a combined field that they call *statistical learning*

**Theory**

A core objective of a learner is to generalize from its experience.[[4]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-bishop2006-4)[[29]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-:5-29) Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of [theoretical computer science](https://en.wikipedia.org/wiki/Theoretical_computer_science) known as [computational learning theory](https://en.wikipedia.org/wiki/Computational_learning_theory). Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The [bias–variance decomposition](https://en.wikipedia.org/wiki/Bias%E2%80%93variance_decomposition) is one way to quantify generalization [error](https://en.wikipedia.org/wiki/Errors_and_residuals).

For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has under fitted the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to [overfitting](https://en.wikipedia.org/wiki/Overfitting) and generalization will be poorer.[[30]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-alpaydin-30)

In addition to performance bounds, learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in [polynomial time](https://en.wikipedia.org/wiki/Time_complexity#Polynomial_time). There are two kinds of [time complexity](https://en.wikipedia.org/wiki/Time_complexity) results: Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

**Approaches**

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

* [Supervised learning](https://en.wikipedia.org/wiki/Supervised_learning): The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that [maps](https://en.wikipedia.org/wiki/Map_(mathematics)) inputs to outputs.
* [Unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning): No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end ([feature learning](https://en.wikipedia.org/wiki/Feature_learning)).
* [Reinforcement learning](https://en.wikipedia.org/wiki/Reinforcement_learning): A computer program interacts with a dynamic environment in which it must perform a certain goal (such as [driving a vehicle](https://en.wikipedia.org/wiki/Autonomous_car) or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize

**Supervised Learning**

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs.[[34]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-34) The data is known as [training data](https://en.wikipedia.org/wiki/Training_data), and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an [array](https://en.wikipedia.org/wiki/Array_data_structure) or vector, sometimes called a feature vector, and the training data is represented by a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)). Through [iterative optimization](https://en.wikipedia.org/wiki/Mathematical_optimization#Computational_optimization_techniques) of an [objective function](https://en.wikipedia.org/wiki/Loss_function), supervised learning algorithms learn a function that can be used to predict the output associated with new inputs.[[35]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-35) An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.[[18]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-Mitchell-1997-18)

Types of supervised learning algorithms include [active learning](https://en.wikipedia.org/wiki/Active_learning_(machine_learning)), [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis).[[26]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-:3-26) Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email.

[Similarity learning](https://en.wikipedia.org/wiki/Similarity_learning) is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in [ranking](https://en.wikipedia.org/wiki/Ranking), [recommendation systems](https://en.wikipedia.org/wiki/Recommendation_systems), visual identity tracking, face verification, and speaker verification.

**Unsupervised Learning**

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of [density estimation](https://en.wikipedia.org/wiki/Density_estimation) in [statistics](https://en.wikipedia.org/wiki/Statistics), such as finding the [probability density function](https://en.wikipedia.org/wiki/Probability_density_function).[[36]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-JordanBishop2004-36) Though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Cluster analysis is the assignment of a set of observations into subsets (called *clusters*) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some *similarity metric* and evaluated, for example, by *internal compactness*, or the similarity between members of the same cluster, and *separation*, the difference between clusters. Other methods are based on *estimated density* and *graph connectivity*.

**Semi-Supervised Learning**

Semi-supervised learning falls between [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) (without any labeled training data) and [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) (with completely labeled training data). Some of the training examples are missing training labels, yet many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy.

In [weakly supervised learning](https://en.wikipedia.org/wiki/Weak_supervision), the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets

**Reinforcement Learning**

Reinforcement learning is an area of machine learning concerned with how [software agents](https://en.wikipedia.org/wiki/Software_agent) ought to take [actions](https://en.wikipedia.org/wiki/Action_selection) in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as [game theory](https://en.wikipedia.org/wiki/Game_theory), [control theory](https://en.wikipedia.org/wiki/Control_theory), [operations research](https://en.wikipedia.org/wiki/Operations_research), [information theory](https://en.wikipedia.org/wiki/Information_theory), [simulation-based optimization](https://en.wikipedia.org/wiki/Simulation-based_optimization), [multi-agent systems](https://en.wikipedia.org/wiki/Multi-agent_system), [swarm intelligence](https://en.wikipedia.org/wiki/Swarm_intelligence), [statistics](https://en.wikipedia.org/wiki/Statistics) and [genetic algorithms](https://en.wikipedia.org/wiki/Genetic_algorithm). In machine learning, the environment is typically represented as a [Markov decision process](https://en.wikipedia.org/wiki/Markov_decision_process) (MDP). Many reinforcement learning algorithms use [dynamic programming](https://en.wikipedia.org/wiki/Dynamic_programming) techniques.[[38]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-38) Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

**Dimensionality Reduction**

Dimensionality reduction is a process of reducing the number of random variables under consideration by obtaining a set of principal variables.[[39]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-39) In other words, it is a process of reducing the dimension of the feature set, also called "number of features". Most of the dimensionality reduction techniques can be considered as either feature elimination or extraction. One of the popular methods of dimensionality reduction is [principal component analysis](https://en.wikipedia.org/wiki/Principal_component_analysis) (PCA). PCA involves changing higher-dimensional data (e.g., 3D) to a smaller space (e.g., 2D). This results in a smaller dimension of data (2D instead of 3D), while keeping all original variables in the model without changing the data.[[40]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-40) The [manifold hypothesis](https://en.wikipedia.org/wiki/Manifold_hypothesis) proposes that high-dimensional data sets lie along low-dimensional [manifolds](https://en.wikipedia.org/wiki/Manifold), and many dimensionality reduction techniques make this assumption, leading to the area of [manifold learning](https://en.wikipedia.org/wiki/Manifold_learning) and [manifold regularization](https://en.wikipedia.org/wiki/Manifold_regularization).

**Self Learning**

Self-learning as a machine learning paradigm was introduced in 1982 along with a neural network capable of self-learning named *crossbar adaptive array* (CAA).[[42]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-42) It is a learning with no external rewards and no external teacher advice. The CAA self-learning algorithm computes, in a crossbar fashion, both decisions about actions and emotions (feelings) about consequence situations. The system is driven by the interaction between cognition and emotion.[[43]](https://en.wikipedia.org/wiki/Machine_learning#cite_note-43) The self-learning algorithm updates a memory matrix W =||w(a,s)|| such that in each iteration executes the following machine learning routine:

In situation s perform an action a;

Receive consequence situation s’;

Compute emotion of being in consequence situation v(s’);

Update crossbar memory w’(a,s) = w(a,s) + v(s’).

It is a system with only one input, situation s, and only one output, action (or behavior) a. There is neither a separate reinforcement input nor an advice input from the environment. The backpropagated value (secondary reinforcement) is the emotion toward the consequence situation. The CAA exists in two environments, one is the behavioral environment where it behaves, and the other is the genetic environment, wherefrom it initially and only once receives initial emotions about situations to be encountered in the behavioral environment. After receiving the genome (species) vector from the genetic environment, the CAA learns a goal-seeking behavior, in an environment that contains both desirable and undesirable situations

**2.3.3 SK-Learn**

**Scikit-learn** (formerly **scikits.learn** and also known as **sklearn**) is a [free software](https://en.wikipedia.org/wiki/Free_software) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) [programming language](https://en.wikipedia.org/wiki/Programming_language).[[3]](https://en.wikipedia.org/wiki/Scikit-learn#cite_note-jmlr-3) It features various [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) algorithms including [support-vector machines](https://en.wikipedia.org/wiki/Support_vector_machine), [random forests](https://en.wikipedia.org/wiki/Random_forests), [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting), [*k*-means](https://en.wikipedia.org/wiki/K-means_clustering) and [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN), and is designed to interoperate with the Python numerical and scientific libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and [SciPy](https://en.wikipedia.org/wiki/SciPy). Scikit-learn is a [NumFOCUS](https://en.wikipedia.org/w/index.php?title=NumFOCUS&action=edit&redlink=1) fiscally sponsored project.

**Overview**

The scikit-learn project started as scikits.learn, a [Google Summer of Code](https://en.wikipedia.org/wiki/Google_Summer_of_Code) project by French [data scientist](https://en.wikipedia.org/wiki/Data_scientist) [David Cournapeau](https://en.wikipedia.org/wiki/David_Cournapeau). Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to [SciPy](https://en.wikipedia.org/wiki/SciPy).[[5]](https://en.wikipedia.org/wiki/Scikit-learn#cite_note-5) The original [codebase](https://en.wikipedia.org/wiki/Codebase) was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from the [French Institute for Research in Computer Science and Automation](https://en.wikipedia.org/wiki/French_Institute_for_Research_in_Computer_Science_and_Automation) in [Rocquencourt](https://en.wikipedia.org/wiki/Rocquencourt,_Yvelines), [France](https://en.wikipedia.org/wiki/France), took leadership of the project and made the first public release on February the 1st 2010.[[6]](https://en.wikipedia.org/wiki/Scikit-learn#cite_note-6) Of the various scikits, scikit-learn as well as [scikit-image](https://en.wikipedia.org/wiki/Scikit-image) were described as "well-maintained and popular" in November 2012.[[7]](https://en.wikipedia.org/wiki/Scikit-learn#cite_note-7) Scikit-learn is one of the most popular machine learning libraries on [GitHub](https://en.wikipedia.org/wiki/GitHub).[[8]](https://en.wikipedia.org/wiki/Scikit-learn#cite_note-8)

**Implementation**

Scikit-learn is largely written in Python, and uses [NumPy](https://en.wikipedia.org/wiki/NumPy) extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in [Cython](https://en.wikipedia.org/wiki/Cython) to improve performance. Support vector machines are implemented by a Cython wrapper around [LIBSVM](https://en.wikipedia.org/wiki/LIBSVM); logistic regression and linear support vector machines by a similar wrapper around [LIBLINEAR](https://en.wikipedia.org/wiki/LIBLINEAR). In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as [Matplotlib](https://en.wikipedia.org/wiki/Matplotlib) and [plotly](https://en.wikipedia.org/wiki/Plotly) for plotting, [NumPy](https://en.wikipedia.org/wiki/NumPy) for array vectorization, [Pandas](https://en.wikipedia.org/wiki/Pandas_(software)) dataframes, [SciPy](https://en.wikipedia.org/wiki/SciPy), and many more.

**Essential Features**

The [Machine Learning library](https://www.upgrad.com/blog/machine-learning-libraries/) scikit-learn in Python comes with a load of features to simplify Machine Learning. Here we will discuss some of them:

* **Supervised learning algorithms:** Any supervised Machine Learning algorithm that you may have heard of has a very high possibility of belonging to the scikit-learn library. The scikit-learn toolkit has a repertoire of such supervised learning algorithms, which includes – Generalized linear models such as [Linear regression](https://www.upgrad.com/blog/linear-regression-in-machine-learning/), Decision Trees, Support Vector Machines, and Bayesian methods.
* **Unsupervised learning algorithms:** This algorithm collection includes factoring, cluster analysis, principal component analysis, and unsupervised neural networks.
* **Feature extraction:** Using scikit-learn, you can extract features from text and images.
* **Cross-validation:** The accuracy and validity of supervised models on unseen data can be checked with the help of scikit-learn.
* **Dimensionality Reduction:** With this feature, the number of attributes in data can be reduced for subsequent visualization, summarization, and feature selection.
* **Clustering:** This feature allows the grouping of unlabeled data.
* **Ensemble methods:** The predictions of several supervised models can be combined by using this feature.

**Advantages**

The [Machine Learning library](https://www.upgrad.com/blog/machine-learning-libraries/) scikit-learn in Python comes with a load of features to simplify Machine Learning. Here we will discuss some of them:

* **Supervised learning algorithms:** Any supervised Machine Learning algorithm that you may have heard of has a very high possibility of belonging to the scikit-learn library. The scikit-learn toolkit has a repertoire of such supervised learning algorithms, which includes – Generalized linear models such as [Linear regression](https://www.upgrad.com/blog/linear-regression-in-machine-learning/), Decision Trees, Support Vector Machines, and Bayesian methods.
* **Unsupervised learning algorithms:** This algorithm collection includes factoring, cluster analysis, principal component analysis, and unsupervised neural networks.
* **Feature extraction:** Using scikit-learn, you can extract features from text and images.
* **Cross-validation:** The accuracy and validity of supervised models on unseen data can be checked with the help of scikit-learn.
* **Dimensionality Reduction:** With this feature, the number of attributes in data can be reduced for subsequent visualization, summarization, and feature selection.
* **Clustering:** This feature allows the grouping of unlabeled data.
* **Ensemble methods:** The predictions of several supervised models can be combined by using this feature.

**Disadvantages**

* Not the best choice for deep learning

**2.3.4 KNN(K Nearest Neighbors)**

In [statistics](https://en.wikipedia.org/wiki/Statistics), the ***k*-nearest neighbors algorithm** (***k*-NN**) is a [non-parametric](https://en.wikipedia.org/wiki/Non-parametric_statistics) [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) method first developed by [Evelyn Fix](https://en.wikipedia.org/wiki/Evelyn_Fix) and [Joseph Hodges](https://en.wikipedia.org/wiki/Joseph_Lawson_Hodges_Jr.) in 1951,[[1]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-1) and later expanded by [Thomas Cover](https://en.wikipedia.org/wiki/Thomas_M._Cover).[[2]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-2) It is used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis). In both cases, the input consists of the *k* closest training examples in a [data set](https://en.wikipedia.org/wiki/Data_set). The output depends on whether *k*-NN is used for classification or regression:

* In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive [integer](https://en.wikipedia.org/wiki/Integer), typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.
* In *k-NN regression*, the output is the property value for the object. This value is the average of the values of *k* nearest neighbors.

*k*-NN is a type of [classification](https://en.wikipedia.org/wiki/Classification) where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then [normalizing](https://en.wikipedia.org/wiki/Normalization_(statistics)) the training data can improve its accuracy dramatically.[[3]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-:0-3)[[4]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-4)

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/*d*, where *d* is the distance to the neighbor.[[5]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-5)

The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the *k*-NN algorithm is that it is sensitive to the local structure of the data.

**Algorithm**

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the [feature vectors](https://en.wikipedia.org/wiki/Feature_vector) and class labels of the training samples.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.

A commonly used distance metric for [continuous variables](https://en.wikipedia.org/wiki/Continuous_variable) is [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance). For discrete variables, such as for text classification, another metric can be used, such as the **overlap metric** (or [Hamming distance](https://en.wikipedia.org/wiki/Hamming_distance)). In the context of gene expression microarray data, for example, *k*-NN has been employed with correlation coefficients, such as Pearson and Spearman, as a metric.[[6]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-6) Often, the classification accuracy of *k*-NN can be improved significantly if the distance metric is learned with specialized algorithms such as [Large Margin Nearest Neighbor](https://en.wikipedia.org/wiki/Large_Margin_Nearest_Neighbor) or [Neighbourhood components analysis](https://en.wikipedia.org/wiki/Neighbourhood_components_analysis).

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the *k* nearest neighbors due to their large number.[[7]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-Coomans_Massart1982-7) One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its *k* nearest neighbors. The class (or value, in regression problems) of each of the *k* nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example, in a [self-organizing map](https://en.wikipedia.org/wiki/Self-organizing_map) (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. *K*-NN can then be applied to the SOM.

**Parameter Selection**

The best choice of *k* depends upon the data; generally, larger values of *k* reduces effect of the noise on the classification,[[8]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-8) but make boundaries between classes less distinct. A good *k* can be selected by various [heuristic](https://en.wikipedia.org/wiki/Heuristic_(computer_science)) techniques (see [hyperparameter optimization](https://en.wikipedia.org/wiki/Hyperparameter_optimization)). The special case where the class is predicted to be the class of the closest training sample (i.e. when *k* = 1) is called the nearest neighbor algorithm.

The accuracy of the *k*-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into [selecting](https://en.wikipedia.org/wiki/Feature_selection) or [scaling](https://en.wikipedia.org/wiki/Feature_scaling) features to improve classification. A particularly popular[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] approach is the use of [evolutionary algorithms](https://en.wikipedia.org/wiki/Evolutionary_algorithm) to optimize feature scaling.[[9]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-9) Another popular approach is to scale features by the [mutual information](https://en.wikipedia.org/wiki/Mutual_information) of the training data with the training classes.[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]

In binary (two class) classification problems, it is helpful to choose *k* to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal *k* in this setting is via bootstrap method.[[10]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-HPS2008-10)

**Properties**

*k*-NN is a special case of a [variable-bandwidth, kernel density "balloon" estimator](https://en.wikipedia.org/wiki/Variable_kernel_density_estimation) with a uniform [kernel](https://en.wikipedia.org/wiki/Kernel_(statistics)).[[13]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-Terrell_Scott1992-13) [[14]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-Mills2010-14)

The naive version of the algorithm is easy to implement by computing the distances from the test example to all stored examples, but it is computationally intensive for large training sets. Using an approximate [nearest neighbor search](https://en.wikipedia.org/wiki/Nearest_neighbor_search) algorithm makes *k-*NN computationally tractable even for large data sets. Many nearest neighbor search algorithms have been proposed over the years; these generally seek to reduce the number of distance evaluations actually performed.

*k-*NN has some strong [consistency](https://en.wikipedia.org/wiki/Consistency_(statistics)) results. As the amount of data approaches infinity, the two-class *k-*NN algorithm is guaranteed to yield an error rate no worse than twice the [Bayes error rate](https://en.wikipedia.org/wiki/Bayes_error_rate) (the minimum achievable error rate given the distribution of the data).[[15]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-15) Various improvements to the *k*-NN speed are possible by using proximity graphs.[[16]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-16)

**Metric Learning**

The K-nearest neighbor classification performance can often be significantly improved through ([supervised](https://en.wikipedia.org/wiki/Supervised_learning)) metric learning. Popular algorithms are [neighbourhood components analysis](https://en.wikipedia.org/wiki/Neighbourhood_components_analysis) and [large margin nearest neighbor](https://en.wikipedia.org/wiki/Large_margin_nearest_neighbor). Supervised metric learning algorithms use the label information to learn a new [metric](https://en.wikipedia.org/wiki/Metric_(mathematics)) or [pseudo-metric](https://en.wikipedia.org/wiki/Pseudometric_space).

**Feature Extraction**

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called [feature extraction](https://en.wikipedia.org/wiki/Feature_extraction). If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. Feature extraction is performed on raw data prior to applying *k*-NN algorithm on the transformed data in [feature space](https://en.wikipedia.org/wiki/Feature_space).

An example of a typical [computer vision](https://en.wikipedia.org/wiki/Computer_vision) computation pipeline for [face recognition](https://en.wikipedia.org/wiki/Facial_recognition_system) using *k*-NN including feature extraction and dimension reduction pre-processing steps (usually implemented with [OpenCV](https://en.wikipedia.org/wiki/OpenCV)):

1. [Haar](https://en.wikipedia.org/wiki/Haar_wavelet) face detection
2. [Mean-shift](https://en.wikipedia.org/wiki/Mean-shift) tracking analysis
3. [PCA](https://en.wikipedia.org/wiki/Principal_Component_Analysis) or [Fisher LDA](https://en.wikipedia.org/wiki/Linear_discriminant_analysis) projection into feature space, followed by *k*-NN classification

**Dimension Reduction**

For high-dimensional data (e.g., with number of dimensions more than 10) [dimension reduction](https://en.wikipedia.org/wiki/Dimension_reduction) is usually performed prior to applying the *k*-NN algorithm in order to avoid the effects of the [curse of dimensionality](https://en.wikipedia.org/wiki/Curse_of_Dimensionality). [[18]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-18)

The [curse of dimensionality](https://en.wikipedia.org/wiki/Curse_of_dimensionality) in the *k*-NN context basically means that [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance) is unhelpful in high dimensions because all vectors are almost equidistant to the search query vector (imagine multiple points lying more or less on a circle with the query point at the center; the distance from the query to all data points in the search space is almost the same).

[Feature extraction](https://en.wikipedia.org/wiki/Feature_extraction) and dimension reduction can be combined in one step using [principal component analysis](https://en.wikipedia.org/wiki/Principal_Component_Analysis) (PCA), [linear discriminant analysis](https://en.wikipedia.org/wiki/Linear_discriminant_analysis) (LDA), or [canonical correlation analysis](https://en.wikipedia.org/wiki/Canonical_correlation) (CCA) techniques as a pre-processing step, followed by clustering by *k*-NN on [feature vectors](https://en.wikipedia.org/wiki/Feature_(machine_learning)) in reduced-dimension space. This process is also called low-dimensional [embedding](https://en.wikipedia.org/wiki/Embedding).[[19]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-19)

For very-high-dimensional datasets (e.g. when performing a similarity search on live video streams, DNA data or high-dimensional [time series](https://en.wikipedia.org/wiki/Time_series)) running a fast **approximate** *k*-NN search using [locality sensitive hashing](https://en.wikipedia.org/wiki/Locality_Sensitive_Hashing), "random projections",[[20]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-20) "sketches" [[21]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-21) or other high-dimensional similarity search techniques from the [VLDB](https://en.wikipedia.org/wiki/VLDB_conference) toolbox might be the only feasible option.

**Decision Boundary**

Nearest neighbor rules in effect implicitly compute the [decision boundary](https://en.wikipedia.org/wiki/Decision_boundary). It is also possible to compute the decision boundary explicitly, and to do so efficiently, so that the computational complexity is a function of the boundary complexity.

**Data Reduction**

[Data reduction](https://en.wikipedia.org/wiki/Data_reduction) is one of the most important problems for work with huge data sets. Usually, only some of the data points are needed for accurate classification. Those data are called the *prototypes* and can be found as follows:

1. Select the *class-outliers*, that is, training data that are classified incorrectly by *k*-NN (for a given *k*)
2. Separate the rest of the data into two sets: (i) the prototypes that are used for the classification decisions and (ii) the *absorbed points* that can be correctly classified by *k*-NN using prototypes. The absorbed points can then be removed from the training set.

### Selection of class-outliers

A training example surrounded by examples of other classes is called a class outlier. Causes of class outliers include:

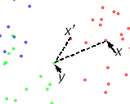
* random error
* insufficient training examples of this class (an isolated example appears instead of a cluster)
* missing important features (the classes are separated in other dimensions which we don't know)
* too many training examples of other classes (unbalanced classes) that create a "hostile" background for the given small class

Class outliers with *k*-NN produce noise. They can be detected and separated for future analysis. Given two natural numbers, *k>r>0*, a training example is called a *(k,r)*NN class-outlier if its *k* nearest neighbors include more than *r* examples of other classes

.

### Condensed Nearest Neighbor for data reduction

Condensed nearest neighbor (CNN, the [*Hart*](https://en.wikipedia.org/wiki/Peter_E._Hart)*algorithm*) is an algorithm designed to reduce the data set for *k*-NN classification.[[23]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-23) It selects the set of prototypes *U* from the training data, such that 1NN with *U* can classify the examples almost as accurately as 1NN does with the whole data set.

[](https://en.wikipedia.org/wiki/File:BorderRAtio.PNG)

Calculation of the border ratio.

[](https://en.wikipedia.org/wiki/File:PointsTypes.png)

Three types of points: prototypes, class-outliers, and absorbed points.

Given a training set *X*, CNN works iteratively:

1. Scan all elements of *X*, looking for an element *x* whose nearest prototype from *U* has a different label than *x*.
2. Remove *x* from *X* and add it to *U*
3. Repeat the scan until no more prototypes are added to *U*.

Use *U* instead of *X* for classification. The examples that are not prototypes are called "absorbed" points.

It is efficient to scan the training examples in order of decreasing border ratio.[[24]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-MirkesKnn-24) The border ratio of a training example *x* is defined as

*a*(*x*) = ||*x'-y*||/||*x-y*||

where ||*x-y*|| is the distance to the closest example *y* having a different color than *x*, and ||*x'-y*|| is the distance from *y* to its closest example *x'*with the same label as *x*.

The border ratio is in the interval [0,1] because ||*x'-y*||never exceeds ||*x-y*||. This ordering gives preference to the borders of the classes for inclusion in the set of prototypes *U*. A point of a different label than *x* is called external to *x*. The calculation of the border ratio is illustrated by the figure on the right. The data points are labeled by colors: the initial point is *x* and its label is red. External points are blue and green. The closest to *x* external point is *y*. The closest to *y* red point is *x'*. The border ratio *a*(*x*) = ||*x'-y*|| / ||*x-y*||is the attribute of the initial point *x*.

Below is an illustration of CNN in a series of figures. There are three classes (red, green and blue). Fig. 1: initially there are 60 points in each class. Fig. 2 shows the 1NN classification map: each pixel is classified by 1NN using all the data. Fig. 3 shows the 5NN classification map. White areas correspond to the unclassified regions, where 5NN voting is tied (for example, if there are two green, two red and one blue points among 5 nearest neighbors). Fig. 4 shows the reduced data set. The crosses are the class-outliers selected by the (3,2)NN rule (all the three nearest neighbors of these instances belong to other classes); the squares are the prototypes, and the empty circles are the absorbed points. The left bottom corner shows the numbers of the class-outliers, prototypes and absorbed points for all three classes. The number of prototypes varies from 15% to 20% for different classes in this example. Fig. 5 shows that the 1NN classification map with the prototypes is very similar to that with the initial data set. The figures were produced using the Mirkes applet.

**K-NN Regression**

In *k*-NN regression, the *k*-NN algorithm[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] is used for estimating continuous variables. One such algorithm uses a weighted average of the *k* nearest neighbors, weighted by the inverse of their distance. This algorithm works as follows:

1. Compute the Euclidean or [Mahalanobis distance](https://en.wikipedia.org/wiki/Mahalanobis_distance) from the query example to the labeled examples.
2. Order the labeled examples by increasing distance.
3. Find a heuristically optimal number *k* of nearest neighbors, based on [RMSE](https://en.wikipedia.org/wiki/RMSE). This is done using cross validation.
4. Calculate an inverse distance weighted average with the *k*-nearest multivariate neighbors.

**K-NN Outlier**

The distance to the *k*th nearest neighbor can also be seen as a local density estimate and thus is also a popular outlier score in [anomaly detection](https://en.wikipedia.org/wiki/Anomaly_detection). The larger the distance to the *k*-NN, the lower the local density, the more likely the query point is an outlier.[[25]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-25) Although quite simple, this outlier model, along with another classic data mining method, [local outlier factor](https://en.wikipedia.org/wiki/Local_outlier_factor), works quite well also in comparison to more recent and more complex approaches, according to a large scale experimental analysis.

**Validation of Results**

A [confusion matrix](https://en.wikipedia.org/wiki/Confusion_matrix) or "matching matrix" is often used as a tool to validate the accuracy of *k*-NN classification. More robust statistical methods such as [likelihood-ratio test](https://en.wikipedia.org/wiki/Likelihood-ratio_test) can also be applied

**2.3.5 SVM**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), **support-vector machines** (**SVMs**, also **support-vector networks**[[1]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-CorinnaCortes-1)) are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis). Developed at [AT&T Bell Laboratories](https://en.wikipedia.org/wiki/AT%26T_Bell_Laboratories) by [Vladimir Vapnik](https://en.wikipedia.org/wiki/Vladimir_Vapnik) with colleagues (Boser et al., 1992, [Guyon](https://en.wikipedia.org/wiki/Isabelle_Guyon) et al., 1993, [Cortes](https://en.wikipedia.org/wiki/Corinna_Cortes) and Vapnik, 1995,[[2]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-article1995-2) Vapnik et al., 1997[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]) SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or [VC theory](https://en.wikipedia.org/wiki/VC_theory) proposed by Vapnik (1982, 1995) and Chervonenkis (1974). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-[probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classification) [binary](https://en.wikipedia.org/wiki/Binary_classifier) [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier) (although methods such as [Platt scaling](https://en.wikipedia.org/wiki/Platt_scaling) exist to use SVM in a probabilistic classification setting). SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing [linear classification](https://en.wikipedia.org/wiki/Linear_classifier), SVMs can efficiently perform a non-linear classification using what is called the [kernel trick](https://en.wikipedia.org/wiki/Kernel_method#Mathematics:_the_kernel_trick), implicitly mapping their inputs into high-dimensional feature spaces.

When data are unlabelled, supervised learning is not possible, and an [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) approach is required, which attempts to find natural [clustering of the data](https://en.wikipedia.org/wiki/Cluster_analysis) to groups, and then map new data to these formed groups. The **support-vector clustering**[[3]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-HavaSiegelmann-3) algorithm, created by [Hava Siegelmann](https://en.wikipedia.org/wiki/Hava_Siegelmann) and [Vladimir Vapnik](https://en.wikipedia.org/wiki/Vladimir_Vapnik), applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data.

**Applications**

SVMs can be used to solve various real-world problems:

* SVMs are helpful in [text and hypertext categorization](https://en.wikipedia.org/wiki/Text_categorization), as their application can significantly reduce the need for labeled training instances in both the standard inductive and [transductive](https://en.wikipedia.org/wiki/Transduction_(machine_learning)) settings.[[8]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-8) Some methods for [shallow semantic parsing](https://en.wikipedia.org/wiki/Shallow_semantic_parsing) are based on support vector machines.[[9]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-9)
* [Classification of images](https://en.wikipedia.org/wiki/Image_classification) can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for [image segmentation](https://en.wikipedia.org/wiki/Image_segmentation) systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.[[10]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-10)[[11]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-11)
* Classification of satellite data like [SAR](https://en.wikipedia.org/wiki/Synthetic-aperture_radar) data using supervised SVM.[[12]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-12)
* Hand-written characters can be [recognized](https://en.wikipedia.org/wiki/Handwriting_recognition) using SVM.[[13]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-13)[[14]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-14)
* The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly. [Permutation tests](https://en.wikipedia.org/wiki/Permutation_test) based on SVM weights have been suggested as a mechanism for interpretation of SVM models.[[15]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-15)[[16]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-16) Support-vector machine weights have also been used to interpret SVM models in the past.[[17]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-17) Posthoc interpretation of support-vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.

**Multiclass SVM**

Multiclass SVM aims to assign labels to instances by using support-vector machines, where the labels are drawn from a finite set of several elements.

The dominant approach for doing so is to reduce the single [multiclass problem](https://en.wikipedia.org/wiki/Multiclass_problem) into multiple [binary classification](https://en.wikipedia.org/wiki/Binary_classification) problems.[[26]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-duan2005-26) Common methods for such reduction include:[[26]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-duan2005-26)[[27]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-hsu2002-27)

* Building binary classifiers that distinguish between one of the labels and the rest (*one-versus-all*) or between every pair of classes (*one-versus-one*). Classification of new instances for the one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest-output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). For the one-versus-one approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the most votes determines the instance classification.
* [Directed acyclic graph](https://en.wikipedia.org/wiki/Directed_acyclic_graph) SVM (DAGSVM)[[28]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-28)
* [Error-correcting output codes](https://en.wikipedia.org/wiki/Error_correcting_code)[[29]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-29)

Crammer and Singer proposed a multiclass SVM method which casts the [multiclass classification](https://en.wikipedia.org/wiki/Multiclass_classification) problem into a single optimization problem, rather than decomposing it into multiple binary classification problems.[[30]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-30) See also Lee, Lin and Wahba[[31]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-31)[[32]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-32) and Van den Burg and Groenen.

**Implementation**

The parameters of the maximum-margin hyperplane are derived by solving the optimization. There exist several specialized algorithms for quickly solving the [quadratic programming](https://en.wikipedia.org/wiki/Quadratic_programming) (QP) problem that arises from SVMs, mostly relying on heuristics for breaking the problem down into smaller, more manageable chunks.

Another approach is to use an [interior-point method](https://en.wikipedia.org/wiki/Interior-point_method) that uses [Newton](https://en.wikipedia.org/wiki/Newton%27s_method)-like iterations to find a solution of the [Karush–Kuhn–Tucker conditions](https://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions) of the primal and dual problems.[[41]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-41) Instead of solving a sequence of broken-down problems, this approach directly solves the problem altogether. To avoid solving a linear system involving the large kernel matrix, a low-rank approximation to the matrix is often used in the kernel trick.

Another common method is Platt's [sequential minimal optimization](https://en.wikipedia.org/wiki/Sequential_minimal_optimization) (SMO) algorithm, which breaks the problem down into 2-dimensional sub-problems that are solved analytically, eliminating the need for a numerical optimization algorithm and matrix storage. This algorithm is conceptually simple, easy to implement, generally faster, and has better scaling properties for difficult SVM problems.[[42]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-42)

The special case of linear support-vector machines can be solved more efficiently by the same kind of algorithms used to optimize its close cousin, [logistic regression](https://en.wikipedia.org/wiki/Logistic_regression); this class of algorithms includes [sub-gradient descent](https://en.wikipedia.org/wiki/Stochastic_gradient_descent) (e.g., PEGASOS[[43]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-43)) and [coordinate descent](https://en.wikipedia.org/wiki/Coordinate_descent) (e.g., LIBLINEAR[[44]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-44)). LIBLINEAR has some attractive training-time properties. Each convergence iteration takes time linear in the time taken to read the train data, and the iterations also have a [Q-linear convergence](https://en.wikipedia.org/wiki/Rate_of_convergence) property, making the algorithm extremely fast.

The general kernel SVMs can also be solved more efficiently using [sub-gradient descent](https://en.wikipedia.org/wiki/Stochastic_gradient_descent) (e.g. P-packSVM[[45]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-45)), especially when [parallelization](https://en.wikipedia.org/wiki/Parallelization) is allowed.

Kernel SVMs are available in many machine-learning toolkits, including [LIBSVM](https://en.wikipedia.org/wiki/LIBSVM), [MATLAB](https://en.wikipedia.org/wiki/MATLAB), [SAS](http://support.sas.com/documentation/cdl/en/whatsnew/64209/HTML/default/viewer.htm#emdocwhatsnew71.htm), SVMlight, [kernlab](https://cran.r-project.org/package=kernlab), [scikit-learn](https://en.wikipedia.org/wiki/Scikit-learn), [Shogun](https://en.wikipedia.org/wiki/Shogun_(toolbox)), [Weka](https://en.wikipedia.org/wiki/Weka_(machine_learning)), [Shark](http://image.diku.dk/shark/), [JKernelMachines](https://mloss.org/software/view/409/), [OpenCV](https://en.wikipedia.org/wiki/OpenCV) and others.

Preprocessing of data (standardization) is highly recommended to enhance accuracy of classification.[[46]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-46) There are a few methods of standardization, such as min-max, normalization by decimal scaling, Z-score.[[47]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-47) Subtraction of mean and division by variance of each feature is usually used for SVM.

**2.3.5** **Flask**

**Flask** is a micro [web framework](https://en.wikipedia.org/wiki/Web_framework) written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). It is classified as a [microframework](https://en.wikipedia.org/wiki/Microframework) because it does not require particular tools or libraries.[[2]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-2) It has no [database](https://en.wikipedia.org/wiki/Database) abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.[[3]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-3)

Applications that use the Flask framework include [Pinterest](https://en.wikipedia.org/wiki/Pinterest) and [LinkedIn](https://en.wikipedia.org/wiki/LinkedIn).

Flask was created by [Armin Ronacher](https://en.wikipedia.org/wiki/Armin_Ronacher) of Pocoo, an international group of Python enthusiasts formed in 2004.[[6]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-6) According to Ronacher, the idea was originally an [April Fool's](https://en.wikipedia.org/wiki/April_Fool%27s) joke that was popular enough to make into a serious application.[[7]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-openingflask-7)[[8]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-8)[[9]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-aprilfoolspage-9) The name is a play on the earlier Bottle framework.[[7]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-openingflask-7)

When Ronacher and Georg Brandl created a bulletin board system written in Python in 2004, the Pocoo projects Werkzeug and [Jinja](https://en.wikipedia.org/wiki/Jinja_(template_engine)) were developed.[[10]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-10)

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.[[11]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-11)[[12]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-12)

Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on [GitHub](https://en.wikipedia.org/wiki/GitHub) among Python web-development frameworks, only slightly behind [Django](https://en.wikipedia.org/wiki/Django_(web_framework)),[[13]](https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-13) and was voted the most popular web framework in the Python Developers Survey 2018.

{\displaystyle \|X\_{(1)}-x\|\leq \dots \leq \|X\_{(n)}-x\|}.

**CHAPTER-3**

**SYSTEM ANALYSIS**

* 1. **FEASIBILITYSTUDY**

The Feasibility study is a test of system proposal according to its workability, impact on the organization, ability to meet user needs and effective use of resources. The object of Feasibility study is not to solve the problem, but to acquire the sense of its scope. During the study the problem definition is crystallized and aspects of the problem to be included in the system are determined, consequently costs and benefits are estimated with greater detail at this stage. The result of Feasibility study is the system formal proposal. This is simply a form of documenting or detailing the nature and scope of proposed solutions. The proposal summarizes what is known and what is going to be done. Three key considerations involved in Feasibility analyses,

* + 1. Economic Feasibility
    2. Technical Feasibility
    3. Behavioral Feasibility

## Economic Feasibility

## As observed, the machine learning model can compute weather events relatively quickly using a mid-range consumer CPU. While using the model at scale might require more computing resources, the cost for the devices will be paid off quickly as we can reduce economic loss by quicker prediction of weather.

## Technical Feasibility

The feasibility center on the existing computer system (application, hardware) and to what extent can extend it can support the proposed addition. The proposed system can easily be computed by a mid-range CPU. Therefor, the system can be considered technically feasible.

## Behavioral Feasibility

## People are inherently resistant to change and computer has known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have towards the development of computerized system. In the existing system more manpower is required and time factor is more but in the proposed system, power and time factors are reduced. So, the remaining numbers are engaged with some other important works

## Existing System

## Current systems use statical calculations to predict weather.

## These methods are simpler and less computationally intensive

## DRAWBACKS OF EXISTING SYSTEM

## They are too simple to model the complexities of the weather

## Therefore, they are more inaccurate then the Machine Learning models

## 3.4 PROPOSED SYSTEM

## The proposed system is quiet simple to use. It is not complex in its functionalities. It is easy for a new or regular user to use it.

## This is because the machine learning model is used in the backend and abstracted away by the UI.

## Benefits of Proposed System

* Better accuracy than the existing system
* Better ability to model complex weather patterns than the existing system.

## 3.5 Scope of the project

* This system allows the user to predict weather quickly and accurately.
* The user only has to upload a file containing various parameters.
* The model is abstracted away and the user inly interacts with the UI.

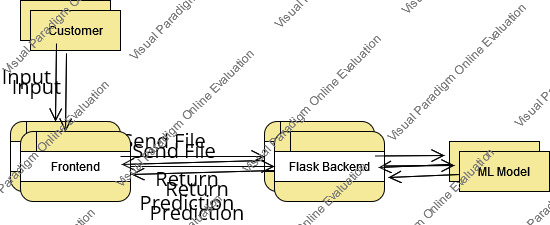
## CHAPTER-4 SYSTEM DESIGN

* 1. **DATA FLOWDIAGRAM**

The DFD takes an input-process-output view of a system i.e. data objects flow into the software, are transformed by processing elements, and resultant data objects flow out of the software. Data objects represented by labeled arrows and transformation are represented by circles also called as bubbles. DFD is presented in a hierarchical fashion i.e. the first data flow model represents the system as a whole. Subsequent DFD refine the context diagram (level 0 DFD), providing increasing details with each subsequent level.

The DFD enables the software engineer to develop models of the information domain & functional domain at the same time. As the DFD is refined into greater levels of details, the analyst performs an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of the data as it moves through the processes that embody the applications.

A context-level DFD for the system the primary external entities produce information for use by the system and consume information generated by the system. The labeled arrow represents data objects or object hierarchy.



**CHAPTER-5**

**PROJECT DESCRIPTION**

* 1. **OBJECTIVE**
     + The main objective of our project is to quick and accurate prediction of weather.
     + This is implemented by using machine learning models.
     + We first get a weather dataset
     + Then we clean it
     + Then we create multiple models and compare them
     + The model with the best accuracy is chosen for the backend
     + We create ML models of KNN and SVM
     + The SVM models use 2 kernels, one is POLY and the other is LINEAR

**5.2 MODULES**

* Submit Button
* File input field
* Flask Backend
* Machine Learning Model

**CHAPTER – 6**

**SYSTEM TESTING**

**6.1 TESTING DEFINITION**

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing 0the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is Quality assurance.

Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is done. Using detailed testing strategies, a test plan is Carrie do upon each module. The various tests performed in **“Network Backup System”** are Unit testing, Integration testing and User Acceptance testing.

**6.2 TESTING OBJECTIVE**

* + - To find errors in the developed software.
    - To check the working of the function is according to the specification. Their behavior and performance required are fulfilled.
    - To check the reliability and quality of the software.

**6.3 TYPES OF TESTING**

## Unit Testing

The software units in a system are modules and routines that are assembled and integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches to the type and size supported by java. The various controls are tested to ensure that each performs its action as required.

## Integration Testing

Data can be lost across any interface, one module can have an adverse effect on another, sub functions when combined, may not produce the desired major functions. Integration testing is a systematic testing to discover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here the server module and client module options are integrated and tested. This testing provides the assurance that the application is well integrated functional unit with smooth transition of data.

## User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the system users at time of developing and making changes whenever required.

**CHAPTER – 7**

**CONCLUSION**

**7.1 SUMMARY**

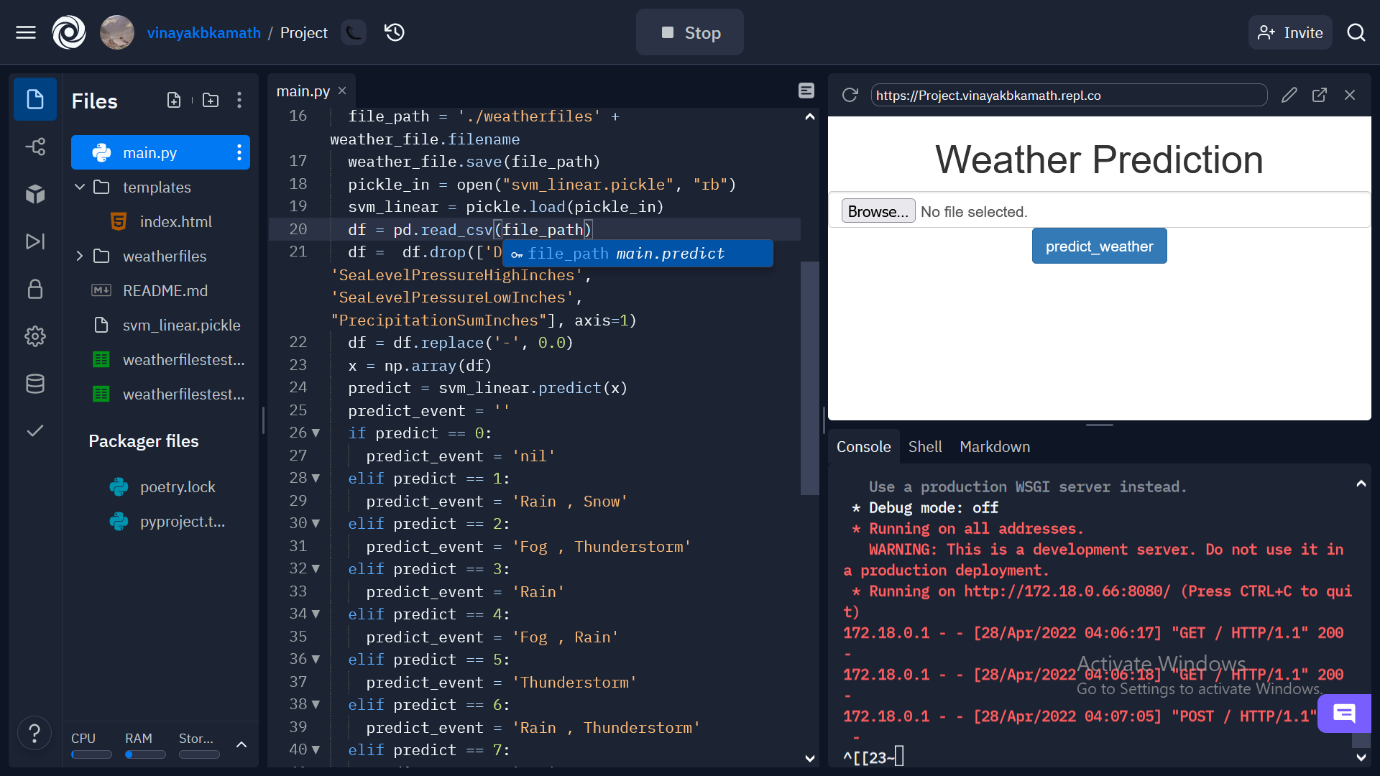
The project developed uses KNN and SVM algorithms from the SKLEARN module to mathematically predict weather events.

Since the weather is prone to frequent change, the computation does not just need to accurate, buy also fast. The algorithm that is most accurate and fast will be chosen as backend for our flask app. These machine learning algorithms are far faster and accurate than the statistical models used today.

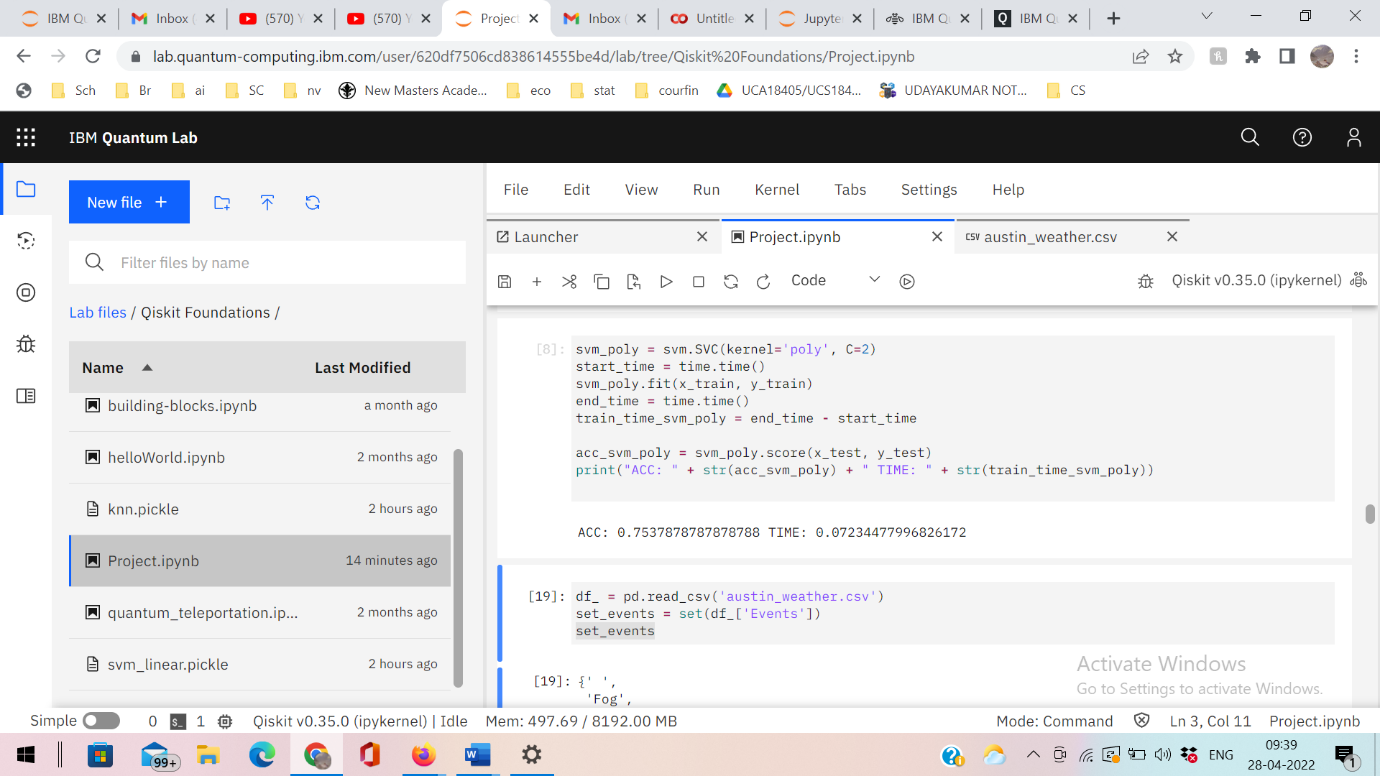
**7.2 FUTURE ENHANCEMENTS**

* + - For the future enhancement we will add quantum machine learning algorithms for the backend. They would be far more faster and accurate than current classical models for matrix calculations
    - The changes to be made will be done briefly in future.

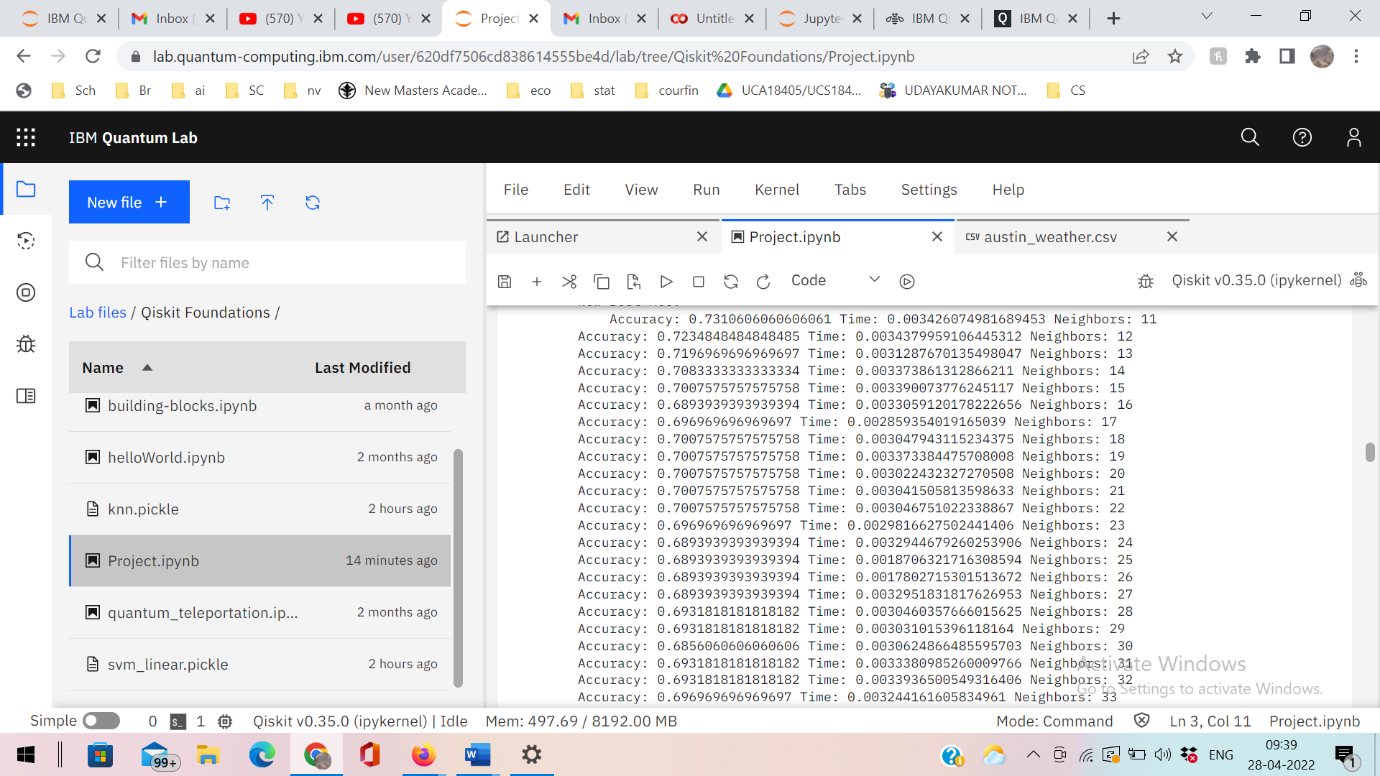
**CHAPTER 9 - IMAGES**



**9.1 WEATHER PREDICTION DEVELOPMENT**



**9.2 SVM POLY KERNEL CODE**



**9.3 KNN ACCURACY LIST**

**CHAPTER – 10 CODE**

**INDEX.HTML**

<head>

<title>Weather Prediction System</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

</head>

<body>

<h1 class='text-center'>Weather Prediction</h1>

<form class='p-3 text-center' action='/' method='post' enctype='multipart/form-data'>

<input class='form-control' type='file' name='weatherfile'>

<input class='btn btn-primary mt-3' type='submit' value='predict\_weather'>

</form>

{% if prediction %}

<p class='text-center'> The weather events is {{prediction}} </p>

{% endif %}

</body>

</html>

**MAIN.py : FLASK APP**

from flask import Flask, render\_template, request

from sklearn import svm

import pandas as pd

import pickle

import numpy as np

app = Flask(\_\_name\_\_)

@app.route('/', methods=['GET'])

def hello\_world():

return render\_template('/index.html')

@app.route('/', methods=['POST'])

def predict():

weather\_file = request.files['weatherfile']

file\_path = './weatherfiles/' + weather\_file.filename

weather\_file.save(file\_path)

pickle\_in = open("svm\_linear.pickle", "rb")

svm\_linear = pickle.load(pickle\_in)

df = pd.read\_csv(file\_path)

df = df.drop(['Date', 'SeaLevelPressureHighInches', 'SeaLevelPressureLowInches', "PrecipitationSumInches"], axis=1)

df = df.replace('-', 0.0)

x = np.array(df)

predict = svm\_linear.predict(x)

predict\_event = ''

if predict == 0:

predict\_event = 'nil'

elif predict == 1:

predict\_event = 'Rain , Snow'

elif predict == 2:

predict\_event = 'Fog , Thunderstorm'

elif predict == 3:

predict\_event = 'Rain'

elif predict == 4:

predict\_event = 'Fog , Rain'

elif predict == 5:

predict\_event = 'Thunderstorm'

elif predict == 6:

predict\_event = 'Rain , Thunderstorm'

elif predict == 7:

predict\_event = 'Fog'

else:

predict\_event = 'Fog , Rain , Thunderstorm'

return render\_template('index.html', prediction=predict\_event)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=8080)

**MACHINE LEARNING CODE**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from sklearn import svm

from sklearn.neighbors import KNeighborsClassifier

!matplotlib inline

from datetime import datetime

import time

import pickle

df = pd.read\_csv('austin\_weather.csv')

df

df = df.drop(

['Date', 'SeaLevelPressureHighInches', 'SeaLevelPressureLowInches', "PrecipitationSumInches"], axis=1)

df = df.replace('-', 0.0)

le = LabelEncoder()

x = np.array(df.drop(['Events'], 1))

y = np.array(df['Events'])

y = le.fit\_transform(list(y))

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2)

best = 0

train\_time = 0

for i in range(10, 50):

knn = KNeighborsClassifier(n\_neighbors=i)

start\_time = time.time()

knn.fit(x\_train, y\_train)

end\_time = time.time()

acc\_knn = knn.score(x\_test, y\_test)

train\_time = end\_time - start\_time

print("Accuracy: " + str(acc\_knn) + " Time: " + str(train\_time) + " Neighbors: " + str(i))

if acc\_knn > best:

best = acc\_knn

print("New Best Acc:")

print(" Accuracy: " + str(acc\_knn) + " Time: " + str(train\_time) + " Neighbors: " + str(i))

with open("knn.pickle", "wb") as f:

pickle.dump(knn, f)

svm\_linear = svm.SVC(kernel='linear', C=2)

start\_time = time.time()

svm\_linear.fit(x\_train, y\_train)

end\_time = time.time()

train\_time\_svm\_linear = end\_time - start\_time

acc\_svm\_linear = svm\_linear.score(x\_test, y\_test)

print("ACC: " + str(acc\_svm\_linear) + " TIME: " + str(train\_time\_svm\_linear))

with open("svm\_linear.pickle", "wb") as f:

pickle.dump(svm\_linear, f)

svm\_poly = svm.SVC(kernel='poly', C=2)

start\_time = time.time()

svm\_poly.fit(x\_train, y\_train)

end\_time = time.time()

train\_time\_svm\_poly = end\_time - start\_time

acc\_svm\_poly = svm\_poly.score(x\_test, y\_test)

print("ACC: " + str(acc\_svm\_poly) + " TIME: " + str(train\_time\_svm\_poly))

**CHAPTER-9 BIBILOGRAPHY AND REFERENCE**

* 1. **BIBILOGRAPHY**
     + [www.w3schools.com](http://www.w3schools.com/)
     + [www.apache.org/](http://www.apache.org/)
     + www.pythonprogramming.net

## REFERENCES

* + - MR. A Vignesh
    - Youtube ML Tutorials
    - <https://en.wikipedia.org/wiki/Machine_learning>
    - <https://en.wikipedia.org/wiki/Python_(programming_language)>
    - <https://en.wikipedia.org/wiki/Scikit-learn>
    - <https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm>
    - https://en.wikipedia.org/wiki/Support-vector\_machine