**YOUTUBE SPAM DETECTION: LEVERAGING ENSEMBLE ALGORITHMS FOR ROBUST FILTERING**

**ABSTRACT:**

The ever-growing popularity of YouTube has brought with it a deluge of unwelcome guests: spam comments. These disruptive messages not only detract from user experience but also stifle genuine conversation. While YouTube employs its own filtering system, it often falls short in completely eradicating the problem.

This research proposes a novel approach to combating YouTube spam – leveraging the power of ensemble learning. We delve into existing studies on YouTube spam detection and conduct a series of classification experiments. Six individual machine learning algorithms are put to the test: Decision Tree, Logistic Regression, Random Forest, Support Vector Machine, Extra Tree Classifier. Additionally, we explore the potential of two ensemble models: Ensemble with Hard Voting and Ensemble with Soft Voting. These models combine the strengths of individual algorithms, aiming to achieve a superior level of spam detection accuracy.

To comprehensively evaluate these techniques, we train them on a dataset of comments from popular music videos by renowned artists. By harnessing the capabilities of machine learning and ensemble approaches, this research aims to develop a robust system for filtering YouTube spam comments, fostering a more positive and engaging online platform for users.

**CHAPTER 1**

**INTRODUCTION**

* 1. **OBJECTIVE:**

YouTube, the world’s largest video sharing site, was founded in 2005 and acquired by Google in 2006. YouTube has grown tremendously as a video content platform, with the recent shift in online content to video. At present, more than 400 hours of video are uploaded and 4.5 million videos are watched every minute on YouTube. It is easy for users to watch and upload videos without any restrictions. This great accessibility has increased the number of personal media, and some of them have become online influencers. YouTube creators can monetize if they have more than 1,000 subscribers and 4,000 hours of watch time for the last 12 months. Accordingly, spam comments are being created to promote their channels or videos in popular videos. Some creators closed the comment function due to aggression such as political comments, abusive speech, or derogatory comments not related to their videos.

Develop a web application that leverages machine learning to create a robust and user-friendly solution for identifying and filtering spam comments on YouTube videos. This application aims to achieve the following:

* Reduce the Prevalence of Spam: By utilizing a pre-trained machine learning model to classify comments, the application will significantly decrease the number of spam comments displayed on videos. This will help to:
  + Protect users from malicious links and phishing attempts often embedded in spam.
  + Safeguard content creators from comment sections flooded with irrelevant promotional content.
* Enhance User Experience: By filtering out spam, the application will foster a more engaging and productive comment environment for YouTube users. This will be achieved by:
  + Encouraging genuine conversation and fostering a sense of community around videos.
  + Empowering content creators to maintain active comment sections without the burden of managing spam.
  + Increasing user trust and satisfaction with the overall YouTube experience.
* We will measure the success of this objective through a combination of quantitative and qualitative metrics:
* Reduction in Spam Rate: The primary metric will be the percentage decrease in comments identified as spam by the application compared to the total number of comments.
* Improved User Engagement: We will track changes in user interaction within the comment sections, such as an increase in the number of comments posted and discussions initiated.
* Positive User Feedback: Conduct surveys or gather user reviews to gauge their satisfaction with the quality and relevance of comments displayed after spam filtering is implemented.
  1. **SCOPE OF PROJECT:**

This project focuses on developing a web application specifically designed to address spam comments on YouTube videos. While existing research explores spam detection across various online platforms, our project will be limited to the YouTube ecosystem. Here's a breakdown of the scope:

* **Target Platform:** YouTube video comments
* **Functionality:**
  + **User Input:** Provide YouTube video URL
  + **Comment Extraction:** Utilize Selenium to extract a designated number of comments from the specified video.
  + **Spam Detection:** Employ a pre-trained machine learning model trained on a curated dataset of labeled YouTube comments (spam vs. legitimate) to analyze and classify each extracted comment.
  + Spam Reporting Integration: After classification, users can select comments identified as spam and flag them for reporting.
  + PDF Report Generation: Upon selecting comments for reporting, the application can generate a PDF report containing:
    - YouTube video information (URL, title, channel name)
    - Extracted comments flagged as spam
    - Corresponding classifications (spam) with accuracy scores
    - Timestamp of report generation
  + **Output:** Present extracted comments alongside the model's classification (spam or legitimate) with an optional accuracy score for each prediction.
* **Focus:** Reducing the prevalence of spam comments and enhancing user experience within YouTube comment sections.
  1. **Problem Statement**

The phenomenal growth of YouTube has unfortunately attracted a surge of malicious actors. Spam comments disrupt user experience by flooding comment sections with irrelevant promotional content, phishing attempts, and other security risks. While YouTube employs a filtering system to mitigate this issue, it faces significant limitations:

* Static Rule Vulnerability: The current filter relies on pre-defined rules, making it susceptible to evolving spam tactics. As spammers employ more sophisticated techniques and obfuscate messages, the filter's effectiveness diminishes.
* Inaccurate Detection: Strict filtering rules can lead to false positives, inadvertently removing legitimate comments that contain keywords or phrases commonly associated with spam. This frustrates users attempting genuine conversation.
* Lack of Transparency: The filtering process remains opaque to users. When comments are flagged as spam, there's minimal explanation provided, hindering trust and user understanding.

These limitations highlight the critical need for a more robust and adaptable approach to tackling YouTube spam comments. This project aims to address this challenge by proposing a novel solution.

* 1. **ACHIEVEMENTS**

**Successful Implementation**

Successfully designed and developed a web application leveraging machine learning algorithms to identify and filter spam comments within YouTube video comment sections. This application empowers users to analyze comment landscapes and fosters a more positive online environment for YouTube users.

**Challenges Overcome**

A key hurdle was fetching YouTube comments, which load as users scroll. We tackled this by implementing Selenium, a web automation tool. Selenium simulates user interaction, allowing the application to scroll and trigger comment loading, resulting in comprehensive comment extraction for analysis.

**CHAPTER 2**

**LITERATURE SURVEY**

* 1. **Existing System**

While YouTube offers a built-in spam filter as a first line of defense, it has limitations. This filtering system relies on pre-defined rules and algorithms to identify and remove spam comments. However, these rules may not be exhaustive, and spammers are constantly innovating their tactics. As a result, a significant number of spam comments can bypass the filter, negatively impacting user experience and hindering genuine conversation within comment sections.

* 1. **Related Work**

Researchers have actively explored the potential of machine learning in tackling YouTube spam comments. These studies have investigated the application of various individual machine learning algorithms, such as Decision Trees, Naive Bayes, Support Vector Machines, and Random Forests, for spam detection. Each approach has its strengths and weaknesses. For instance, Decision Trees offer clear decision-making logic but can be susceptible to overfitting the training data. Naive Bayes is efficient but may struggle with complex data patterns, such as the nuanced language used in some spam comments.

These prior studies provide valuable insights into the effectiveness of individual machine learning algorithms for YouTube spam detection. However, they typically evaluate the performance of algorithms on specific datasets. This approach may not adequately address the ever-evolving nature of spam tactics. Spammers continuously adapt their strategies, and a model trained on a static dataset may not be effective in identifying new forms of spam

* 1. **SIGNIFICANCE**

The ever-growing presence of spam comments on YouTube disrupts user experience and undermines legitimate discourse within comment sections. Existing filtering systems, while offering a first line of defense, often struggle with adaptability and accuracy. This research proposes a groundbreaking solution – ensemble machine learning – to address these limitations. Ensemble learning harnesses the collective power of multiple machine learning algorithms, each with its strengths in identifying different spam characteristics. By combining their diverse perspectives, the proposed system aims to achieve:

* Enhanced Spam Detection Accuracy: Achieve a significantly higher level of accuracy compared to traditional rule-based filters. This translates to a cleaner comment environment, fostering more meaningful interactions.
* Superior Adaptability to Evolving Tactics: Unlike static filters, the proposed system has the inherent ability to learn and adapt to new spam strategies as they emerge. This ensures long-term effectiveness in the face of continuous spammer innovation.
* Reduced False Positives: Machine learning models can be trained to distinguish between legitimate comments and spam with greater nuance, minimizing the accidental removal of genuine user contributions.
* Scalability for Massive Data Volumes: The proposed system is designed to efficiently handle the immense number of comments generated on YouTube, ensuring its effectiveness across the platform.

By implementing this innovative ensemble learning approach, this project has the potential to revolutionize YouTube spam detection. This will create a more positive and engaging user experience for both content creators and viewers, promoting genuine discussion and a healthier online environment on the platform.

* 1. **COMPARISON OF EXISTING AND PROPOSED SYSTEM**

|  |  |  |
| --- | --- | --- |
| Feature | Existing System (YouTube Filter) | Proposed System (Ensemble Learning) |
| Functionality | Automated spam filtering based on pre-defined rules | Leverages machine learning for spam detection and classification |
| Strengths | Easy to implement, reduces some spam | Potentially higher accuracy, adaptable to evolving spam |
| Weaknesses | Limited adaptability, may miss complex spam | More complex to implement, requires training data |
| Overall Effectiveness | Moderate, may struggle with evolving tactics | Potentially superior spam detection, ability to learn |

**CHAPTER 3**

**SYSTEM SPECIFICATION**

* 1. **HARDWARE REQUIREMENTS**

The application has been developed with the system having the following requirements:

* Processor: AMD Ryzen 5 5600H with Radeon Graphics, 3301 Mhz
* RAM: 8 Gb
* Hard Disk: 500 GB (SSD)
  1. **SOFTWARE REQUIREMENTS**
* Operating System: Windows 11
* Front End: Html, CSS, JavaScript
* Scripts: Python Language (Version 3.10.8)
* Software: VS Code with Jupiter Extension
  1. **DOMAIN KNOWLEDGE**

This project delves into the realm of YouTube spam comments, requiring a multifaceted understanding of the technical and strategic aspects involved. Here's a breakdown of the crucial domain knowledge areas:

**Understanding the YouTube Spam Landscape:**

* Spammer Tactics: Familiarity with common strategies employed by spammers on YouTube, including keyword stuffing, promotional links, phishing attempts, and comment manipulation techniques. This knowledge is vital for effectively training the machine learning model to identify these malicious content patterns.
* Evolution of Spam: An awareness of how spam tactics adapt and evolve over time. Spammers constantly seek new ways to bypass detection. Understanding this dynamic is essential for developing a system that can maintain its effectiveness in the long run.
* Limitations of Existing Filters: Knowing the weaknesses of current YouTube spam filters, such as their reliance on static rules and susceptibility to new spam tactics. This knowledge helps identify areas where the proposed ensemble machine learning approach can offer significant improvements.

**Machine Learning for Text Classification:**

* Text Classification Algorithms: A solid grasp of machine learning algorithms adept at text classification tasks, specifically those suited for spam detection. This knowledge base informs the selection of the most appropriate pre-trained model for the project.
* Model Selection and Training: The ability to evaluate different pre-trained models based on factors like accuracy, efficiency, and suitability for the specific task of YouTube comment classification. Additionally, understanding how to fine-tune or retrain the chosen model if necessary.
* Evaluation Metrics: Knowledge of relevant metrics for assessing the performance of the machine learning model in spam detection. This could include metrics like precision, recall, F1-score, and accuracy.

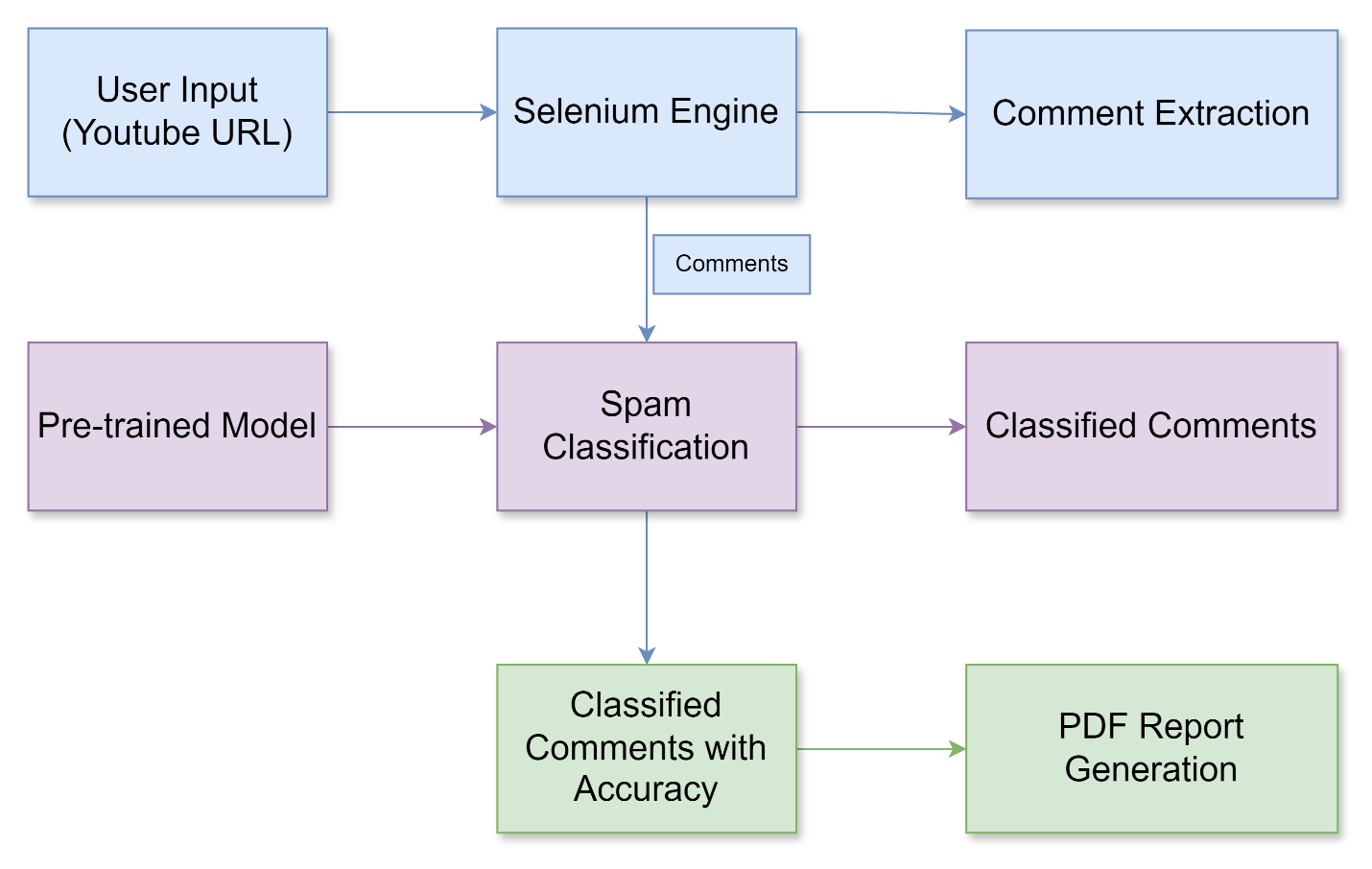
**Web Scraping with Ethical Considerations:**

* Selenium for Comment Extraction: Proficiency in using Selenium, a web automation tool, to ethically extract comments from YouTube videos while adhering to YouTube's Terms of Service and respecting user privacy. This involves understanding best practices for responsible scraping and avoiding overloading YouTube's servers with excessive requests.
* Respecting User Privacy: Awareness of the importance of user privacy when scraping comments. This might involve anonymizing or not storing any personally identifiable information extracted from the comments.

**User Interface (UI) Design:**

* User-friendliness: Understanding the principles of user-friendly interface design to create an application that is easy to navigate and understand for users of varying technical backgrounds. This involves clear instructions, intuitive layouts, and informative feedback mechanisms.
* Data Visualization (Optional): If you plan to incorporate visualizations of the classified comments (e.g., bar charts displaying the distribution of spam vs. legitimate comments), some knowledge of data visualization best practices would be beneficial. This ensures the visualizations are clear, informative, and don't overwhelm users with excessive information.
* Accessibility: Awareness of accessibility guidelines to ensure the UI is usable for people with disabilities. This might involve considerations for color contrast, keyboard navigation, and screen reader compatibility.

**BLOCK DIAGRAM:**



**CHAPTER -IV**

**MODULES DESCRIPTION**

**MODULES LIST:**

* Youtube dataset.
* Preprocessing
* Feature selection
* Feature extraction and feature engineering
* Extra-Tree Classifier classifier

**YOUTUBE DATASET:**

The benefit of using these words based on their entropy score in the characteristic-set is that we have been capable of lessen uncertainty in the prediction final results as those phrases have a exceptional effect of frequency count in spam and non-spam YouTube.

**PREPROCESSING:**

Before starting with preparation preprocessing of the messages must be done. First all the characters must be in lowercase. The word which is both in uppercase and lowercase must be considered as same words and not as two different words. Then tokenization must be done for each message in the data set.

**FEATURE SELECTION:**

The main advantage of using the words present in the dataset is that it is capable of reducing uncertainty in the prediction of the final results as those phrases have a remarkable effect of frequency count in spam and ham comments in YouTube.

**FEATURE EXTRACTION AND FEATURE ENGINEERING:**

Attribute significance is a supervised characteristic that ranks attributes in a step by step manner with their significance in predicting an aim. Here Count Vectorizer is used which convert a “collection of text documents to a matrix of token counts . This undergoes the following technique:

N-grams: N-grams is used to improve the accuracy. It is dealt with single word but when there are two mutual words the complete meaning will be changed. So, the variation of accuracy is better occurred when text is split into token of two or more words rather than being a single word.

Analyzer: “Whether the feature should be made of word or character n-grams. Option ‘char\_wb’ creates character n-grams only from text inside word boundaries; n-grams at the edges of words are padded with space.”

**EXTRA TREE CLASSIFIER:**

The Extra Trees classifier algorithm, a member of the ensemble learning family, shares similarities with Random Forests but introduces distinctive features in its tree-building process. Notably, Extra Trees employs a higher degree of randomization during the creation of individual decision trees. This involves selecting random subsets of features and utilizing random thresholds for node splitting, enhancing the model's robustness to noisy data. This algorithm excels in parallelization, allowing for efficient construction of each tree independently. By embracing randomness, Extra Trees mitigates overfitting concerns, making it less sensitive to the intricacies of the training data. As a result, Extra Trees stands as a powerful and efficient tool for predictive modeling tasks, offering reliable predictions across diverse datasets.

**PROPOSED SYSTEM ALGORITHM:**

**EXTRA TREE ALGORITHM:**

**Prerequisites:** [Decision Tree Classifier](https://www.geeksforgeeks.org/decision-tree-introduction-example/) **Extremely Randomized Trees Classifier(Extra Trees Classifier)** is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a “forest” to output it’s classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest. Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, Each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple de-correlated decision trees. To perform feature selection using the above forest structure, during the construction of the forest, for each feature, the normalized total reduction in the mathematical criteria used in the decision of feature of split (Gini Index if the Gini Index is used in the construction of the forest) is computed. This value is called the Gini Importance of the feature. To perform feature selection, each feature is ordered in descending order according to the Gini Importance of each feature and the user selects the top k features according to his/her choice.

Let us build a hypothetical Extra Trees Forest for the above data with **five decision trees** and the value of k which decides the number of features in a random sample of features be **two**. Here the decision criteria used will be Information Gain. First, we calculate the entropy of the data. Note the formula for calculating the entropy is:-where c is the number of unique class labels and is the proportion of rows with output label is i. Therefore for the given data, the **entropy** is:- [Tex]\Rightarrow Entropy(S) = 0.940  [/Tex]Let the decision trees be constructed such that:-

* **1st Decision Tree gets data with the features Outlook and Temperature:** Note that the formula for Information Gain is:-
* Thus the most important variable to determine the output label according to the above constructed Extra Trees Forest is the feature “Outlook”.

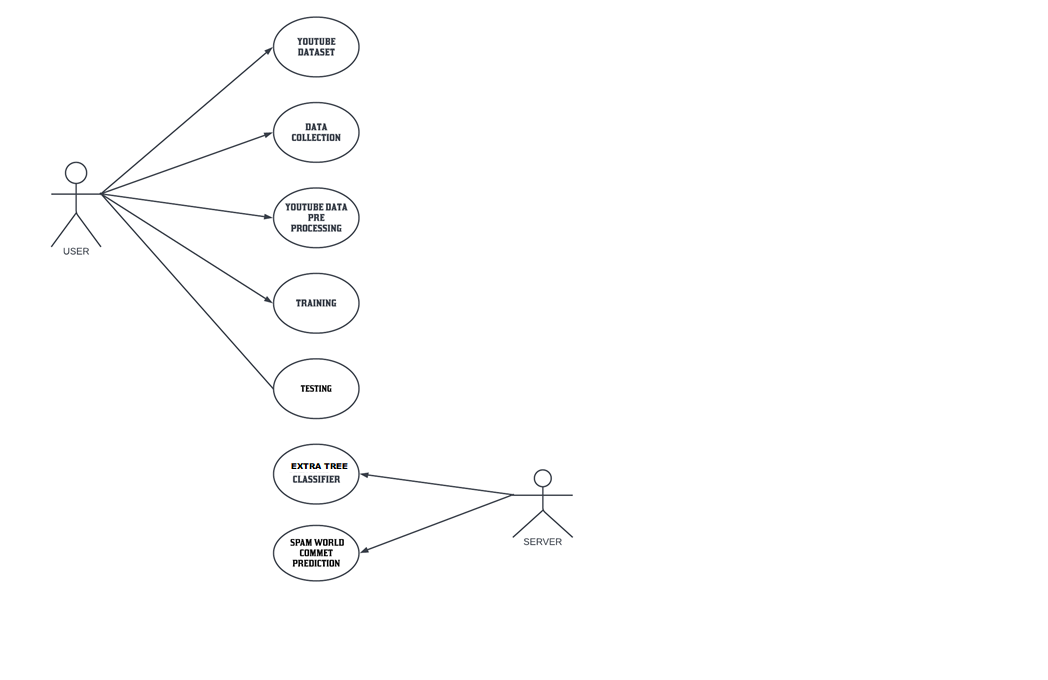
### The Extra Trees Classifier for feature selection offers several advantages:

1. Robust to noise and irrelevant features: Extra Trees Classifier utilizes multiple decision trees and selects features based on their importance scores, making it less sensitive to noise and irrelevant features. It can effectively handle datasets with a large number of features and noisy data.
2. Computational efficiency: Extra Trees Classifier constructs decision trees in parallel, which can significantly speed up the training process compared to other feature selection techniques. It is particularly useful for high-dimensional datasets where efficiency is crucial.
3. Bias reduction: The random selection of subsets and random splitting points in Extra Trees Classifier helps to reduce the bias that can arise from using a single decision tree. By considering multiple decision trees, it provides a more comprehensive evaluation of feature importance.
4. Feature ranking: Extra Trees Classifier assigns importance scores to each feature, allowing you to rank them based on their relative importance. This ranking can provide insights into the relevance and contribution of each feature to the target variable.
5. Handling multicollinearity: The Extra Trees Classifier can handle correlated features effectively. By randomly selecting subsets of features and utilizing random splits, it reduces the impact of multicollinearity, unlike methods that rely on explicit feature correlations.
6. Feature selection flexibility: The feature selection process in Extra Trees Classifier is based on feature importances, allowing you to adapt the threshold for feature inclusion according to your specific needs. You can choose to include only the most important features or a larger subset, depending on the desired balance between feature reduction and model performance.
7. Generalization and interpretability: By selecting a subset of relevant features, Extra Trees Classifier can improve model generalization by reducing overfitting. Additionally, the selected features can provide interpretable insights into the factors that drive predictions and influence the target variable.

These advantages make the Extra Trees Classifier a valuable tool for feature selection, especially when dealing with high-dimensional datasets, noisy data, and situations where computational efficiency is essential.

**UML DIAGRAM:**

**USE CASE DIAGRAM:**

****

**ACTIVITY DIAGRAM:**

****

**SOFTWARE SPECIFICATION:**

## INTRODUCTION TO PYTHON

Python is a high-level object-oriented programming language that was created by Guido van Rossum. It is also called general-purpose programming language as it is used in almost every domain we can think of as mentioned below:

* Web Development
* Software Development
* Game Development
* AI & ML
* Data Analytics

This list can go on as we go but why python is so much popular let’s see it in the next topic.

## WHY PYTHON PROGRAMMING?

You guys might have a question in mind that, why python? why not another programming language?

So let me explain:

Every Programming language serves some purpose or use-case according to a domain. for eg, Javascript is the most popular language amongst web developers as it gives the developer the power to handle applications via different frameworks like react, vue, angular which are used to build beautiful User Interfaces. Similarly, they have pros and cons at the same time. so if we consider python it is general-purpose which means it is widely used in every domain the reason is it’s very simple to understand, scalable because of which the speed of development is so fast. Now you get the idea why besides learning python it doesn’t require any programming background so that’s why it’s popular amongst developers as well. Python has simpler syntax similar to the English language and also the syntax allows developers to write programs with fewer lines of code. Since it is open-source there are many libraries available that make developers’ jobs easy ultimately results in high productivity. They can easily focus on business logic and Its demanding skills in the digital era where information is available in large data sets.

### HOW DO WE GET STARTED?

Now in the era of the digital world, there is a lot of information available on the internet that might confuse us believe me. what we can do is follow the documentation which is a good start point. Once we are familiar with concepts or terminology we can dive deeper into this.

Following are references where we can start our journey:

Official Website: https://www.python.org/

Udemy Course: https://www.udemy.com/course/python-the-complete-python-developer-course/

YouTube: https://www.youtube.com/watch?v=\_uQrJ0TkZlc

CodeAcademy: https://www.codecademy.com/catalog/language/python

I hope now you guys are excited to get started right so you might be wondering where we can start coding right so there are a lot of options available in markets. we can use any IDE we are comfortable with but for those who are new to the programming world I am listing some of IDE’s below for python:

1) Visual Studio: https://visualstudio.microsoft.com/

2) PyCharm: https://www.jetbrains.com/pycharm/

3) Spyder: https://www.spyder-ide.org/

4) Atom: https://atom.io/

5) Google Colab: https://research.google.com/colaboratory/

### Real-World Examples:

1) **NASA (National Aeronautics and Space Agency):** One of Nasa’s Shuttle Support Contractors, United Space Alliance developed a Workflow Automation System (WAS) which is fast. Internal Resources Within critical project stated that:

“Python allows us to tackle the complexity of programs like the WAS without getting bogged down in the language”.

Nasa also published a website (https://code.nasa.gov/) where there are 400 open source projects which use python.

2) **Netflix**: There are various projects in Netflix which use python as follow:

* Central Alert Gateway
* Chaos Gorilla
* Security Monkey
* Chronos

Amongst all projects, Regional failover is the project they have as the system decreases outage time from 45 minutes to 7 minutes with no additional cost.

3) **Instagram:**Instagram also uses python extensively. They have built a photo-sharing social platform using Django which is a web framework for python. Also, they are able to successfully upgrade their framework without any technical challenges.

### Applications of Python Programming:

1) **Web Development**: Python offers different frameworks for web development like Django, Pyramid, Flask. This framework is known for security, flexibility, scalability.

2) **Game Development:** PySoy and PyGame are two python libraries that are used for game development

3) **Artificial Intelligence and Machine Learning:** There is a large number of open-source libraries which can be used while developing AI/ML applications.

4) **Desktop GUI:** Desktop GUI offers many toolkits and frameworks using which we can build desktop applications.PyQt, PyGtk, PyGUI are some of the GUI frameworks.

### How to Become Better Programmer:

The last but most important thing is how you get better at what programming you choose is practice practice practice. Practical knowledge only acquired by playing with things so you will get more exposure to real-world scenarios. Consistency is more important than anything because if you practice it for some days and then you did nothing then when you start again it will be difficult to practice consistently. So I request you guys to learn by doing projects so it will help you understand how things get done and important thing is to have fun at the same time.

### Approach to be followed to master Python:

**“Beginning is the end and end is the beginning”.**I know what you are thinking about. It is basically a famous quote from a web series named “Dark”. Now how it relates to Python programming?

If you researched on google, youtube, or any development communities out there, you will find that people explained how you can master programming in let’s say some “x” number of days and like that.

Well, the reality is like the logo of infinity which we can see above. In the programming realm, there is no such thing as mastery. It’s simply a trial and error process. For example. Yesterday I was writing some code where I was trying to print a value of a variable before declaring it inside a function. There I had seen a new error named “**UnboundLocalErrorException**“.

So the important thing to keep in mind is that programming is a surprising realm. Throughout your entire career, you will be seeing new errors and exceptions. Just remember the quote – **“Practise makes a man perfect”.**

Now here is the main part. What approach to follow in order to master Python Programming?

Well here it is:

### Step-1: Start with a “Hello World” Program

If you happened to learn some programming languages, then I am sure you are aware of what I am talking about. The “Hello World” program is like a tradition in the developer community. If you want to master any programming language, this should be the very first line of code we should be seeking for.

**Simple Hello World Program in Python:**

print("Hello World")

### Step-2: Start learning about variables

Now once we have mastered the “Hello World” program in Python, the next step is to master variables in python. Variables are like containers that are used to store values.

**Variables in Python:**

my\_var = 100

As you can see here, we have created a variable named “my\_var” to assign a value 100 to the same.

### Step-3: Start learning about Data Types and Data Structures

The next outpost is to learn about data types. Here I have seen that there is a lot of confusion between data types and data structures. The important thing to keep in mind here is that data types represent the type of data. For example. in Python, we have something like int, string, float, etc. Those are called data types as they indicate the type of data we are dealing with.

While data structures are responsible for deciding how to store this data in a computer’s memory.

**String data type in Python:**

my\_str = "ABCD"

As you can see here, we have assigned a value “ABCD” to a variable my\_str. This is basically a string data type in Python.

**Data Structure in Python:**

my\_dict={1:100,2:200,3:300}

This is known as a dictionary data structure in Python.

Again this is just the tip of the iceberg. There are lots of data types and data structures in Python. To give a basic idea about data structures in Python, here is the complete list:

1.Lists

2.Dictionary

3.Sets

4.Tuples

5.Frozenset

### Step-4: Start learning about conditionals and loops

In any programming language, conditionals and loops are considered one of the backbone.

Python is no exception for that as well. This is one of the most important concepts that we need to master.

**IF-ELIF-ELSE conditionals:**

if(x < 10):

  print("x is less than 10")

elif(x > 10):

   print("x is greater than 10")

else:

   print("Do nothing")

As you can see in the above example, we have created what is known as the if-elif-else ladder

**For loop:**

for i in "Python":

  print(i)

The above code is basically an example of for loop in python.

### PRO Tip:

Once you start programming with Python, you will be seeing that if we missed any white spacing in python then python will start giving some errors. This is known as Indentation in python. Python is very strict with indentation. Python is created with a mindset to help everyone become a neat programmer. This indentation scheme in python is introduced in one of python’s early PEP(Python Enhancement Proposal).

# THE PYTHON STANDARD LIBRARY

While [The Python Language Reference](https://docs.python.org/3/reference/index.html#reference-index) describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the [Python Package Index](https://pypi.org/).

## **What Is a Python Package?**

To understand Python packages, we’ll briefly look at scripts and modules. A “script” is something you execute in the shell to accomplish a defined task. To write a script, you’d type your code into your favorite [text editor](https://hackr.io/blog/best-python-ide) and save it with the .py extension. You can then use the python command in a terminal to execute your script.

A module on the other hand is a Python program that you import, either in [interactive mode](https://docs.python.org/3/tutorial/interpreter.html#interactive-mode) or into your other programs. “Module” is really an umbrella term for reusable code.

A Python package usually consists of several modules. Physically, a package is a folder containing modules and maybe other folders that themselves may contain more folders and modules. Conceptually, it’s a namespace. This simply means that a package’s modules are bound together by a package name, by which they may be referenced.

Circling back to our earlier definition of a module as reusable, importable code, we note that every package is a module — but not every module is a package. A package folder usually contains one file named \_\_init\_\_.py that basically tells Python: “Hey, this directory is a package!” The init file may be empty, or it may contain code to be executed upon package initialization.

You’ve probably come across the term “library” as well. For Python, a library isn’t as clearly defined as a package or a module, but a good rule of thumb is that whenever a package has been published, it may be referred to as a library.

## **HOW TO USE A PYTHON PACKAGE**

We’ve mentioned namespaces, publishing packages and importing modules. If any of these terms or concepts aren’t entirely clear to you, we’ve got you! In this section, we’ll cover everything you’ll need to really grasp the pipeline of using Python packages in your code.

### **Importing a Python Package**

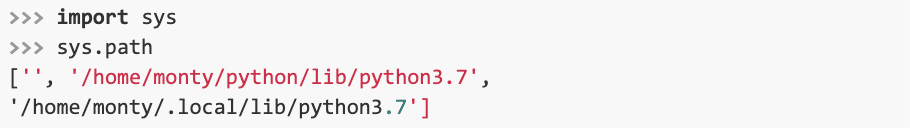
We’ll import a package using the **import** statement:

https://www.udacity.com/blog/wp-content/uploads/2021/01/Screen-Shot-2021-01-04-at-2.25.10-PM.png

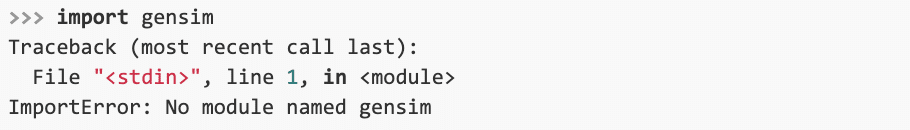
Let’s assume that we haven’t yet installed any packages. Python comes with a big collection of pre-installed packages known as the Python Standard Library. It includes tools for a range of use cases, such as text processing and doing math. Let’s import the latter:

https://www.udacity.com/blog/wp-content/uploads/2021/01/Screen-Shot-2021-01-04-at-2.25.46-PM.png

You might think of an import statement as a search trigger for a module. Searches are strictly organized: At first, Python looks for a module in the cache, then in the standard library and finally in a list of paths. This list may be accessed after importing sys (another standard library module).



The sys.path command returns all the directories in which Python will try to find a package. It may happen that you’ve downloaded a package but when you try importing it, you get an error:



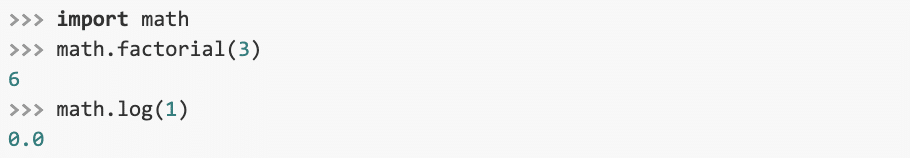
In such cases, check whether your imported package has been placed in one of Python’s search paths. If it hasn’t, you can always expand your list of search paths:

https://www.udacity.com/blog/wp-content/uploads/2021/01/Screen-Shot-2021-01-04-at-2.27.36-PM.png

At that point, the interpreter will have more than one more location to look for packages after receiving an **import** statement.

### **Namespaces and Aliasing**

When we had imported the math module, we initialized the math namespace. This means that we can now refer to functions and classes from the math module by way of “dot notation”:



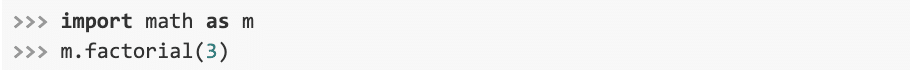
Assume that we were only interested in our math module’s factorial function, and that we’re also tired of using dot notation. In that case, we can proceed as follows:

https://www.udacity.com/blog/wp-content/uploads/2021/01/Screen-Shot-2021-01-04-at-2.28.55-PM.png

If you’d like to import multiple resources from the same source, you can simply comma-separate them in the import statement:

https://www.udacity.com/blog/wp-content/uploads/2021/01/Screen-Shot-2021-01-04-at-2.29.24-PM.png

There is, however, always a small risk that your variables will clash with other variables in your namespace. What if one of the variables in your code was named log, too? It would overwrite the log function, causing bugs. To avoid that, it’s better to import the package as we did before. If you want to save typing time, you can alias your package to give it a shorter name:



Aliasing is a pretty common technique. Some packages have commonly used aliases: For instance, the numerical computation library NumPy is almost always imported as “np.”

Another option is to import all a module’s resources into your namespace:

https://www.udacity.com/blog/wp-content/uploads/2021/01/Screen-Shot-2021-01-04-at-2.30.32-PM.png

However, this method poses serious risk since you usually don’t know all the names contained in a package, increasing the likelihood of your variables being overwritten. It’s for this reason that most seasoned Python programmers will discourage use of the wildcard \* in imports. Also, as the [Zen of Python](https://www.python.org/dev/peps/pep-0020/) states, “namespaces are one honking great idea!”

### **How to Install a Python Package**

How about packages that are not part of the standard library? The official repository for finding and downloading such third-party packages is the Python Package Index, usually referred to simply as [PyPI](https://pypi.org/). To install packages from PyPI, use the package installer [pip](https://packaging.python.org/tutorials/installing-packages/):

https://www.udacity.com/blog/wp-content/uploads/2021/01/Screen-Shot-2021-01-04-at-2.31.11-PM.png

pip can install Python packages from any source, not just PyPI. If you installed Python using [Anaconda](https://www.anaconda.com/) or [Miniconda](https://docs.conda.io/projects/conda/en/latest/user-guide/install/download.html" \l "anaconda-or-miniconda), you can also use the conda command to install Python packages.

https://www.udacity.com/blog/wp-content/uploads/2021/01/Screen-Shot-2021-01-04-at-2.31.47-PM.png

While conda is very easy to use, it’s not as versatile as pip. So if you cannot install a package using conda, you can always try pip instead.

### **Reloading a Module**

If you’re programming in interactive mode, and you change a module’s script, these changes won’t be imported, even if you issue another import statement. In such case, you’ll want to use the reload() function from the importlib library:



## **How to Create Your Own Python Package**

Packaging your code for further use doesn’t necessarily mean you’ll want it published to PyPI. Maybe you just want to share it with a friend, or reuse it yourself. Whatever your aim, there are several files that you should include in your project. We’ve already mentioned the \_\_init\_\_.py file.

Another important file is setup.py. Using the setuptools package, this file provides detailed information about your project and lists all dependencies — packages required by your code to run properly.

Publishing to PyPI is beyond the scope of this introductory tutorial. But if you do have a package for distribution, your project should include two more files: a README.md written in Markdown, and a license. Check out the official Python Packaging User Guide ([PyPUG](https://packaging.python.org/tutorials/packaging-projects/)) if you want to know more.

# INSTALLING PACKAGES

This section covers the basics of how to install Python [packages](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package).

It’s important to note that the term “package” in this context is being used to describe a bundle of software to be installed (i.e. as a synonym for a [distribution](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package)). It does not to refer to the kind of [package](https://packaging.python.org/en/latest/glossary/#term-Import-Package) that you import in your Python source code (i.e. a container of modules). It is common in the Python community to refer to a [distribution](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) using the term “package”. Using the term “distribution” is often not preferred, because it can easily be confused with a Linux distribution, or another larger software distribution like Python itself.

## [Requirements for Installing Packages](https://packaging.python.org/en/latest/tutorials/installing-packages/#id11)

This section describes the steps to follow before installing other Python packages.

### [Ensure you can run Python from the command line](https://packaging.python.org/en/latest/tutorials/installing-packages/#id12)

Before you go any further, make sure you have Python and that the expected version is available from your command line. You can check this by running:

Unix/macOS

python3 --version

Windows

You should get some output like Python 3.6.3. If you do not have Python, please install the latest 3.x version from [python.org](https://www.python.org/) or refer to the [Installing Python](https://docs.python-guide.org/starting/installation/#installation) section of the Hitchhiker’s Guide to Python.

**Note**

If you’re a newcomer and you get an error like this:

**>>>** python --version

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'python' is not defined

It’s because this command and other suggested commands in this tutorial are intended to be run in a shell (also called a terminal or console). See the Python for Beginners [getting started tutorial](https://opentechschool.github.io/python-beginners/en/getting_started.html#what-is-python-exactly) for an introduction to using your operating system’s shell and interacting with Python.

**Note**

If you’re using an enhanced shell like IPython or the Jupyter notebook, you can run system commands like those in this tutorial by prefacing them with a ! character:

In [1]: import sys

!{sys.executable} --version

Python 3.6.3

It’s recommended to write {sys.executable} rather than plain python in order to ensure that commands are run in the Python installation matching the currently running notebook (which may not be the same Python installation that the python command refers to).

**Note**

Due to the way most Linux distributions are handling the Python 3 migration, Linux users using the system Python without creating a virtual environment first should replace the python command in this tutorial with python3 and the python -m pip command with python3 -m pip --user. Do not run any of the commands in this tutorial with sudo: if you get a permissions error, come back to the section on creating virtual environments, set one up, and then continue with the tutorial as written.

### [Ensure you can run pip from the command line](https://packaging.python.org/en/latest/tutorials/installing-packages/#id13)

Additionally, you’ll need to make sure you have [pip](https://packaging.python.org/en/latest/key_projects/#pip) available. You can check this by running:

Unix/macOS

python3 -m pip --version

Windows

If you installed Python from source, with an installer from [python.org](https://www.python.org/), or via [Homebrew](https://brew.sh/) you should already have pip. If you’re on Linux and installed using your OS package manager, you may have to install pip separately, see [Installing pip/setuptools/wheel with Linux Package Managers](https://packaging.python.org/en/latest/guides/installing-using-linux-tools/).

If pip isn’t already installed, then first try to bootstrap it from the standard library:

Unix/macOS

python3 -m ensurepip --default-pip

Windows

If that still doesn’t allow you to run python -m pip:

* Securely Download [get-pip.py](https://bootstrap.pypa.io/get-pip.py) [1](https://packaging.python.org/en/latest/tutorials/installing-packages/#id7)
* Run python get-pip.py. [2](https://packaging.python.org/en/latest/tutorials/installing-packages/#id8) This will install or upgrade pip. Additionally, it will install [setuptools](https://packaging.python.org/en/latest/key_projects/" \l "setuptools) and [wheel](https://packaging.python.org/en/latest/key_projects/#wheel) if they’re not installed already.

**Warning**

Be cautious if you’re using a Python install that’s managed by your operating system or another package manager. get-pip.py does not coordinate with those tools, and may leave your system in an inconsistent state. You can use python get-pip.py --prefix=/usr/local/ to install in /usr/local which is designed for locally-installed software.

### [Ensure pip, setuptools, and wheel are up to date](https://packaging.python.org/en/latest/tutorials/installing-packages/#id14)

While pip alone is sufficient to install from pre-built binary archives, up to date copies of the setuptools and wheel projects are useful to ensure you can also install from source archives:

Unix/macOS

python3 -m pip install --upgrade pip setuptools wheel

Windows

### [Optionally, create a virtual environment](https://packaging.python.org/en/latest/tutorials/installing-packages/#id15)

See [section below](https://packaging.python.org/en/latest/tutorials/installing-packages/#creating-and-using-virtual-environments) for details, but here’s the basic [venv](https://docs.python.org/3/library/venv.html" \o "(in Python v3.10)) [3](https://packaging.python.org/en/latest/tutorials/installing-packages/#id9) command to use on a typical Linux system:

Unix/macOS

python3 -m venv tutorial\_env

source tutorial\_env/bin/activate

Windows

This will create a new virtual environment in the tutorial\_env subdirectory, and configure the current shell to use it as the default python environment.

## [Creating Virtual Environments](https://packaging.python.org/en/latest/tutorials/installing-packages/#id16)

Python “Virtual Environments” allow Python [packages](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) to be installed in an isolated location for a particular application, rather than being installed globally. If you are looking to safely install global command line tools, see [Installing stand alone command line tools](https://packaging.python.org/en/latest/guides/installing-stand-alone-command-line-tools/).

Imagine you have an application that needs version 1 of LibFoo, but another application requires version 2. How can you use both these applications? If you install everything into /usr/lib/python3.6/site-packages (or whatever your platform’s standard location is), it’s easy to end up in a situation where you unintentionally upgrade an application that shouldn’t be upgraded.

Or more generally, what if you want to install an application and leave it be? If an application works, any change in its libraries or the versions of those libraries can break the application.

Also, what if you can’t install [packages](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) into the global site-packages directory? For instance, on a shared host.

In all these cases, virtual environments can help you. They have their own installation directories and they don’t share libraries with other virtual environments.

Currently, there are two common tools for creating Python virtual environments:

* [venv](https://docs.python.org/3/library/venv.html) is available by default in Python 3.3 and later, and installs [pip](https://packaging.python.org/en/latest/key_projects/#pip) and [setuptools](https://packaging.python.org/en/latest/key_projects/" \l "setuptools) into created virtual environments in Python 3.4 and later.
* [virtualenv](https://packaging.python.org/en/latest/key_projects/#virtualenv) needs to be installed separately, but supports Python 2.7+ and Python 3.3+, and [pip](https://packaging.python.org/en/latest/key_projects/#pip), [setuptools](https://packaging.python.org/en/latest/key_projects/" \l "setuptools) and [wheel](https://packaging.python.org/en/latest/key_projects/#wheel) are always installed into created virtual environments by default (regardless of Python version).

The basic usage is like so:

Using [venv](https://docs.python.org/3/library/venv.html" \o "(in Python v3.10)):

Unix/macOS

python3 -m venv <DIR>

source <DIR>/bin/activate

Windows

Using [virtualenv](https://packaging.python.org/en/latest/key_projects/" \l "virtualenv):

Unix/macOS

python3 -m virtualenv <DIR>

source <DIR>/bin/activate

Windows

For more information, see the [venv](https://docs.python.org/3/library/venv.html" \o "(in Python v3.10)) docs or the [virtualenv](https://virtualenv.pypa.io/en/stable/index.html" \o "(in virtualenv v20.14)) docs.

The use of **source** under Unix shells ensures that the virtual environment’s variables are set within the current shell, and not in a subprocess (which then disappears, having no useful effect).

In both of the above cases, Windows users should \_not\_ use the **source** command, but should rather run the **activate** script directly from the command shell like so:

<DIR>\Scripts\activate

Managing multiple virtual environments directly can become tedious, so the [dependency management tutorial](https://packaging.python.org/en/latest/tutorials/managing-dependencies/#managing-dependencies) introduces a higher level tool, [Pipenv](https://packaging.python.org/en/latest/key_projects/" \l "pipenv), that automatically manages a separate virtual environment for each project and application that you work on.

## [Use pip for Installing](https://packaging.python.org/en/latest/tutorials/installing-packages/#id17)

[pip](https://packaging.python.org/en/latest/key_projects/#pip) is the recommended installer. Below, we’ll cover the most common usage scenarios. For more detail, see the [pip docs](https://pip.pypa.io/en/latest/), which includes a complete [Reference Guide](https://pip.pypa.io/en/latest/cli/).

## [Installing from PyPI](https://packaging.python.org/en/latest/tutorials/installing-packages/#id18)

The most common usage of [pip](https://packaging.python.org/en/latest/key_projects/#pip) is to install from the [Python Package Index](https://packaging.python.org/en/latest/glossary/#term-Python-Package-Index-PyPI) using a [requirement specifier](https://packaging.python.org/en/latest/glossary/#term-Requirement-Specifier). Generally speaking, a requirement specifier is composed of a project name followed by an optional [version specifier](https://packaging.python.org/en/latest/glossary/#term-Version-Specifier). [**PEP 440**](https://www.python.org/dev/peps/pep-0440) contains a [**full specification**](https://www.python.org/dev/peps/pep-0440#version-specifiers) of the currently supported specifiers. Below are some examples.

To install the latest version of “SomeProject”:

Unix/macOS

python3 -m pip install "SomeProject"

Windows

To install a specific version:

Unix/macOS

python3 -m pip install "SomeProject==1.4"

Windows

To install greater than or equal to one version and less than another:

Unix/macOS

python3 -m pip install "SomeProject>=1,<2"

Windows

To install a version that’s [**“compatible”**](https://www.python.org/dev/peps/pep-0440#compatible-release) with a certain version: [4](https://packaging.python.org/en/latest/tutorials/installing-packages/#id10)

Unix/macOS

python3 -m pip install "SomeProject~=1.4.2"

Windows

In this case, this means to install any version “==1.4.\*” version that’s also “>=1.4.2”.

## [Source Distributions vs Wheels](https://packaging.python.org/en/latest/tutorials/installing-packages/#id19)

[pip](https://packaging.python.org/en/latest/key_projects/#pip) can install from either [Source Distributions (sdist)](https://packaging.python.org/en/latest/glossary/#term-Source-Distribution-or-sdist) or [Wheels](https://packaging.python.org/en/latest/glossary/#term-Wheel), but if both are present on PyPI, pip will prefer a compatible [wheel](https://packaging.python.org/en/latest/glossary/#term-Wheel). You can override pip`s default behavior by e.g. using its [–no-binary](https://pip.pypa.io/en/latest/cli/pip_install/#install-no-binary) option.

[Wheels](https://packaging.python.org/en/latest/glossary/#term-Wheel) are a pre-built [distribution](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) format that provides faster installation compared to [Source Distributions (sdist)](https://packaging.python.org/en/latest/glossary/#term-Source-Distribution-or-sdist), especially when a project contains compiled extensions.

If [pip](https://packaging.python.org/en/latest/key_projects/#pip) does not find a wheel to install, it will locally build a wheel and cache it for future installs, instead of rebuilding the source distribution in the future.

## [Upgrading packages](https://packaging.python.org/en/latest/tutorials/installing-packages/#id20)

Upgrade an already installed SomeProject to the latest from PyPI.

Unix/macOS

python3 -m pip install --upgrade SomeProject

Windows

## [Installing to the User Site](https://packaging.python.org/en/latest/tutorials/installing-packages/#id21)

To install [packages](https://packaging.python.org/en/latest/glossary/#term-Distribution-Package) that are isolated to the current user, use the --user flag:

Unix/macOS

python3 -m pip install --user SomeProject

Windows

For more information see the [User Installs](https://pip.pypa.io/en/latest/user_guide/#user-installs) section from the pip docs.

Note that the --user flag has no effect when inside a virtual environment - all installation commands will affect the virtual environment.

If SomeProject defines any command-line scripts or console entry points, --user will cause them to be installed inside the [user base](https://docs.python.org/3/library/site.html#site.USER_BASE)’s binary directory, which may or may not already be present in your shell’s PATH. (Starting in version 10, pip displays a warning when installing any scripts to a directory outside PATH.) If the scripts are not available in your shell after installation, you’ll need to add the directory to your PATH:

* On Linux and macOS you can find the user base binary directory by running python -m site --user-base and adding bin to the end. For example, this will typically print ~/.local (with ~ expanded to the absolute path to your home directory) so you’ll need to add ~/.local/bin to your PATH. You can set your PATH permanently by [modifying ~/.profile](https://stackoverflow.com/a/14638025).
* On Windows you can find the user base binary directory by running py -m site --user-site and replacing site-packages with Scripts. For example, this could return C:\Users\Username\AppData\Roaming\Python36\site-packages so you would need to set your PATH to include C:\Users\Username\AppData\Roaming\Python36\Scripts. You can set your user PATH permanently in the [Control Panel](https://docs.microsoft.com/en-us/windows/win32/shell/user-environment-variables?redirectedfrom=MSDN). You may need to log out for the PATH changes to take effect.

## [Requirements files](https://packaging.python.org/en/latest/tutorials/installing-packages/#id22)

Install a list of requirements specified in a [Requirements File](https://pip.pypa.io/en/latest/user_guide/#requirements-files).

Unix/macOS

python3 -m pip install -r requirements.txt

Windows

## [Installing from VCS](https://packaging.python.org/en/latest/tutorials/installing-packages/#id23)

Install a project from VCS in “editable” mode. For a full breakdown of the syntax, see pip’s section on [VCS Support](https://pip.pypa.io/en/latest/cli/pip_install/#vcs-support).

Unix/macOS

python3 -m pip install -e git+https://git.repo/some\_pkg.git#egg=SomeProject *# from git*

python3 -m pip install -e hg+https://hg.repo/some\_pkg#egg=SomeProject *# from mercurial*

python3 -m pip install -e svn+svn://svn.repo/some\_pkg/trunk/#egg=SomeProject *# from svn*

python3 -m pip install -e git+https://git.repo/some\_pkg.git@feature#egg=SomeProject *# from a branch*

Windows

## [Installing from other Indexes](https://packaging.python.org/en/latest/tutorials/installing-packages/#id24)

Install from an alternate index

Unix/macOS

python3 -m pip install --index-url http://my.package.repo/simple/ SomeProject

Windows

Search an additional index during install, in addition to [PyPI](https://packaging.python.org/en/latest/glossary/" \l "term-Python-Package-Index-PyPI)

Unix/macOS

python3 -m pip install --extra-index-url http://my.package.repo/simple SomeProject

Windows

## [Installing from a local src tree](https://packaging.python.org/en/latest/tutorials/installing-packages/#id25)

Installing from local src in [Development Mode](https://setuptools.pypa.io/en/latest/userguide/development_mode.html), i.e. in such a way that the project appears to be installed, but yet is still editable from the src tree.

Unix/macOS

python3 -m pip install -e <path>

Windows

You can also install normally from src

Unix/macOS

python3 -m pip install <path>

Windows

## [Installing from local archives](https://packaging.python.org/en/latest/tutorials/installing-packages/#id26)

Install a particular source archive file.

Unix/macOS

python3 -m pip install ./downloads/SomeProject-1.0.4.tar.gz

Windows

Install from a local directory containing archives (and don’t check [PyPI](https://packaging.python.org/en/latest/glossary/" \l "term-Python-Package-Index-PyPI))

Unix/macOS

python3 -m pip install --no-index --find-links=file:///local/dir/ SomeProject

python3 -m pip install --no-index --find-links=/local/dir/ SomeProject

python3 -m pip install --no-index --find-links=relative/dir/ SomeProject

Windows

## [Installing from other sources](https://packaging.python.org/en/latest/tutorials/installing-packages/#id27)

To install from other data sources (for example Amazon S3 storage) you can create a helper application that presents the data in a [**PEP 503**](https://www.python.org/dev/peps/pep-0503) compliant index format, and use the --extra-index-url flag to direct pip to use that index.

./s3helper --port=7777

python -m pip install --extra-index-url http://localhost:7777 SomeProject

## [Installing Prereleases](https://packaging.python.org/en/latest/tutorials/installing-packages/#id28)

Find pre-release and development versions, in addition to stable versions. By default, pip only finds stable versions.

Unix/macOS

python3 -m pip install --pre SomeProject

Windows

## [Installing Setuptools “Extras”](https://packaging.python.org/en/latest/tutorials/installing-packages/#id29)

Install [setuptools extras](https://setuptools.readthedocs.io/en/latest/userguide/dependency_management.html" \l "optional-dependencies).

Unix/macOS

python3 -m pip install SomePackage[PDF]

python3 -m pip install SomePackage[PDF]==3.0

python3 -m pip install -e .[PDF] *# editable project in current directory*

**SOFTWARE TESTING**

**6.1 UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**6.2 INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**6.3 FUNCTIONAL TEST**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**6.4 SYSTEM TEST**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**6.5 WHITE BOX TESTING**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**6.5 BLACK BOX TESTING**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**2.3 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**2.3.1 ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### 2.3.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**2.3.3 SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**CONCLUSION:**

In this paper, we proposed a technique to detect spam comments on YouTube, which have recently seen tremendous growth using a Cascaded Ensemble Machine Learning Model. It examined related studies on YouTube spam comment screening and conducted classification experiments with six different machine learning techniques (Decision tree, Logistic regression, Bernoulli Naïve Bayes, Random Forest, Support vector machine with linear kernel, Support vector machine with Gaussian kernel) and two ensemble models (Ensemble with hard voting, Ensemble with soft voting) combining these techniques in the comment data. The experimental results showed that the ESM-S model proposed in this paper had the best performance in four of five evaluation measures. We proposed a new model, combining various techniques that improved the performance results unlike previous studies that used one model for detection

**REFERENCE:**

[1]. H. Shaban. (Sep. 19, 2019). Nearly Half of Cellphone Calls Will be Scams by 2019, Report Says. The Washington Post. Accessed: Feb. 21, 2020.

[2]. O. Abayomi-Alli, S. Misra, A. Abayomi-Alli, and M. Odusami, ‘‘A review of soft techniques for SMS spam classification: Methods, approaches and applications,’’ Eng. Appl. Artif. Intell., vol. 86, pp. 197–212, Nov. 2019.

[3]. O. M. E. Ebadati and F. Ahmadzadeh, ‘‘Classification spam email with elimination of unsuitable features with hybrid of GA-naive Bayes,’’ J. Inf. Knowl. Manage., vol. 18, no. 1, Mar. 2019.

[4]. M. S. Hanif and M. Bilal, ‘‘Competitive residual neural network for image classification,’’ ICT Exp., vol. 6, no. 1, pp. 28–37, Mar. 2020.

[5]. H. Liu, M. Zhou, and Q. Liu, ‘‘An embedded feature selection method for imbalanced data classification,’’ IEEE/CAA J. Automatica Sinica, vol. 6, no. 3, pp. 703–715, May 2019.

[6]. Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations,” Feb. 2020.

[7]. S. Venkatraman, B. Surendiran, and P. A. R. Kumar, “Spam e-mail classification for the internet of things environment using semantic similarity approach,” J Supercomput, vol. 76, no. 2, pp. 756-776, 2020.

[8]. K. Sheridan. (2020). FBI: Business Email Compromise Cost Businesses 1.7B in 2019, Dark Reading. Accessed: Mar. 21, 2021.

[9]. A. Ali. (2020). Visualizing the Social Media Universe in 2020. Accessed: Jan. 14, 2021.

[10]. R. Lerman and H. Denham. (2020). 3 Charged in Massive Twitter Hack, Including Alleged Teenage ‘Mastermind’.The Washington Post. Accessed: Jan. 14, 2021.