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

**ABSTRACT:**

This paper proposes a technique to detect spam comments on YouTube, which have recently seen tremendous growth. YouTube is running its own spam blocking system but continues to fail to block them properly. Therefore, we 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 from popular music videos Psy, Katy Perry, LMFAO, Eminem and Shakira.

**SCOPE OF PROJECT:**

Research on detecting spam content and users focus on various fields. Many studies focused on spam on websites (e.g., portal sites and blogs).As YouTube gains popularity as a video sharing platform, spammers target it with low quality content or promotions. Since spammers that harm the YouTube community are increasing, detecting them becomes an interesting source to research. So, we divide the literature of detecting spam into two sections, spam on websites and spam on YouTube.

**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.

**INTRODUCTION:**

In the previous years, informal online communities like Face book and YouTube have become progressively common platform in an individual person’s day to day life. People use social media as a virtual community platform to stay in touch with friends and family and to also share thoughts and ideas in blogs. Due to this developing pattern, these platforms pull in an enormous number of clients and are easy targets for spammers. YouTube has become the most well-known informal community among youngsters. For example, many makeup tutorials have been started by bloggers who are referred to as “beauty guru” or “beauty influencers” in which majority of the audiences are teenage girls. These days, 200 million clients produce 400 million new YouTube content (videos) every day. This extensive environment provided by YouTube also creates an opportunity for spammers to create irrelevant content directed to users. These irrelevant or unsolicited messages are aimed to attack users by luring them into clicking links to view malicious sites containing malware, phising and scams. One of the most highlighted features of YouTube is the comments section below every video posted by a user. This feature allows users to share opinions and ideas. In this project, the prediction of the spam comments present in the comments section of Youtube videos using the concept called machine learning, it is also known as subset of artificial intelligence, is done. Supervised learning approach depends on a very large number of labeled datasets.The proposed classification algorithm (Logistic Regression) is used in order to predict the spam comment. The purpose of project is to introduce briefly the techniques of machine learning and to outline the prediction technique. Being much more superior to the conventional data analysis techniques, machine learning can open a new opportunity to explore and increase the prediction accuracy. Spam remarks are regularly completely immaterial to the given video and are normally created via mechanized bots camouflaged as a client. The comments section is target by spammers to post completely irrelevant messages, comments, links and ideas. AI is the strategy for extraction, changing, stacking and anticipating the significant data from enormous information to remove a few examples and furthermore change it into justifiable structure for additional utilization. Grouping and expectation are two sorts of dissecting information which portray principal classes of information and forecast of patterns in future information. The noxious spam remarks will ruin the positive perspective of the contents present in the videos posted. The contingency for anticipating the spam remarks has started but has yet not been concluded and built up for an exact forecast of spam remarks.

**MOTIVATION OF PROJECT:**

Spam, poses a huge challenge for researchers aiming to find ways to detect or exclude these unwanted messages from nearly every online media. Many studies focused on traditional methods, such as content-based analysis or extraction of features from the content or their information. Analysis of users’ behavior is often, used to improve the accuracy and performance of spam detection. In an experiment based on content analysis, Rathod and Pattewar analyzed a body of the Gmail dataset to classify legitimate and spam email using the Bayesian classifier model. The proposed bayesian classifier was able to achieve as high as 96.46% accuracy. Moreover, similarity and relevance are important features used to distinguish spams from regular messages. Liu et al. developed two algorithms to identify false reviews on Amazon.com based on the similarity of the reviews and how much the review content is related to the product and to describe some common behavior features of spammers in the spam review. According to the observation, it was found that if the similarity of the two reviews is greater than 70%, then the second review was identified as a copied review. The results showed that 54% of mobile phone reviews on Amazon.com are copied from existing reviews. Jindal and Liu categorized customer reviews of Amazon.com into three types: false opinion, brand review, and non-reviews.

**PROBLEM OF STATEMENT:**

Research on detecting spam content and users focus on various fields. Many studies focused on spam on websites (e.g., portal sites and blogs). As YouTube gains popularity as a video sharing platform, spammers target it with low quality content or promotions. Since spammers that harm the YouTube community are increasing, detecting them becomes an interesting source to research. So, we divide the literature of detecting spam into two sections, spam on websites and spam on YouTube. To detect untruthful reviews of the specific product on the Internet, used an n-gram language model. They focused on untruthful reviews that could be duplicated from different ids. Each review was compared with all the others to identify duplicate reviews. N-gram was used to estimate word sequence like which words would be next. In other words, n-grams decompose sentences automatically, breaking them into several small pieces.

**CHAPTER –II**

**LITERATURE SURVEY**

**PAPER 1**

**TITLE NAME:**CASCARO: Cascade Of Classifiers For Minimizing The Cost Of Prediction

**AUTHOR:** Blaise Hanczar, Avner Bar-Hen

**YEAR:** 2021

**ABSTRACT:**

Although the prediction performance is crucial for a classifier, its cost of use is also an essential issue forpracticalapplication, however, this question is rarely addressed in the literature. The aim of this article is topropose a prediction method that controls not only the error rate but also the cost of the construction of theclassifier. The main idea is that some examples are easier to predict than others and can be predicted using fewervariables i.e. with a lower prediction cost. Our method, called CASCARO, is based on a cascade of rejectclassifiers of increasing cost. The first classifier of the cascade required only one variable, if the prediction is notreliable the second classifier requiring one more variable is used. The principle is repeated until the last classifierusing all variables. We solve the two main problems for the construction of this type of cascade: its architecture(the order of the classifier) and the simultaneous computation of the rejection regions of the classifiers. Theexperiments show that CASCARO produces significant improvements in the use cost without decreasingprediction performance

**PAPER 2**

**TITLE NAME:** Twitter Spam Detection Based on Deep Learning

**AUTHOR:** Tingmin Wu, Shigang Liu, Jun Zhang, Yang Xiang

**YEAR:** 2021

**ABSTRACT:**

Twitter spam has long been a critical but difficult problem to be addressed. So far, researchers havedeveloped a series of machine learning-based methods and blacklisting techniques to detect spamming activities onTwitter. According to our investigation, current methods and techniques have achieved the accuracy of around 80%.However, due to the problems of spam drift and information fabrication, these machine-learning based methodscannot efficiently detect spam activities in real-life scenarios. Moreover, the blacklisting method cannot catch upwith the variations of spamming activities as manually inspecting suspicious URLs is extremely time-consuming. Inthis paper, we proposed a novel technique based on deep learning techniques to address the above challenges. Thesyntax of each tweet will be learned through WordVector Training Mode. We then constructed a binary classifierbased on the preceding representation dataset. In experiments, we collected and implemented a 10-day real Tweetdatasets in order to evaluate our proposed method. We first studied the performance of different classifiers, and thencompared our method to other existing text-based methods.

**PAPER 3**

**TITLE NAME:** Two-Stage Unsupervised Approach For Combating Social Spammers

**AUTHOR:** Darshika Koggalahewa, Yue Xu, Emest Foo

**YEAR: 2**0**20**

**ABSTRACT:**

Spammers use Online Social Networks (OSNs) as a popular platform for spreading malicious content andlinks. The nature of OSNs allows the spammers to bypass the combating techniques by changing their behaviours.Classification based approaches are the most common technique for spam detection. “Data labelling”, “spam drift”,“imbalanced datasets” and “data fabrication” are the most common limitations of classification techniques thathinder the accuracy of spam detection. The paper presents a two-stage fully unsupervised approach using a user’speer acceptance within OSN to distinguish spammers from genuine users. User’s common shared interest overmultiple topics and the mentioning behaviour are used to derive the peer acceptance. The contribution of the paperis a pure unsupervised method to detect spammers based on users’ peer acceptance without labelled datasets. Ourunsupervised approach is able to achieve 95.9% accuracy without the need for labelling

**PAPER 4**

**TITLE NAME:** BotNet Detection on Social media

**AUTHOR:** Aniket Chandrakant Devle, Julia Ann Jose, Abhay Shrinivas Saraswathula, Shubham Mehta,

Siddhant Srivastava, Sirisha Kona, Sudheera Daggumalli

**YEAR:** 2020

**ABSTRACT:**

As our reliance on social media platforms and web services increase day by day, exploiters view theseplatforms as an opportunity to manipulate our thoughts and actions. These platforms have become an openplayground for social bot accounts. Social bots not only learn human conversations, manners, and presence butalso manipulate public opinion, act as scammers, manipulate stock markets, and so on. There has been evidenceof bots manipulating people’s opinions and thoughts which can be a great threat to democracy. Identification andprevention of such campaigns that release or create these bots have become critical. Our goal in this paper is toleverage web mining techniques to help detect fake bots on social media platforms such asTwitter, therebymitigating the spread of fake news.

**PAPER 5**

**TITLE NAME:** A Novel Recruitment Policy to Defend against Sybils in Vehicular Crowdsourcing

**AUTHOR:** Federico Concone, Fabrizio De Vita, Ajay Pratap, Dario Bruneo, Giuseppe Lo Re, Sajal K.Da

**YEAR:** 2021

**ABSTRACT:**

Vehicular Social Networks (VSNs) is an emerging communication paradigm, derived by merging the concepts of Online Social Networks (OSNs) and Vehicular Ad-hoc Networks (VANETs). Due to the lack ofrobust authentication mechanisms, social-based vehicular applications are vulnerable to numerous attacksincluding the generation of sybil entities in the networks. We address this important issue in vehicularcrowdsourcing campaigns where sybils are usually employed to increase their influence and worsen thefunctioning of the system. In particular, we propose a novel User Recruitment Policy (URP) that, after extractingthe participants within the event radius of a crowdsourcing campaign, detects and filters out the sybil vehicles byusing a novel sybil detection approach, called SybilDriver. This technique combines the advantages of VANETsand OSNs by means of an innovative concept of proximity graph obtained from the physical vehicular network, inconjunction with a community detection and Random Forest techniques adopted in the OSN domain.

**CHAPTER-III**

**METHEDOLOGY**

**EXISTING SYSTEM:**

YouTube has its own spam filtering system, though there are still spam comments that are not being caught. In this paper, we review related studies on YouTube spam comments and propose the Cascaded Ensemble Machine Learning Model aware YouTube Spam Comments Detection Scheme to improve the performance of the model. In previous studies, various machine learning techniques were applied to each dataset to detect spam comments and compare their performance. Therefore, in this paper, we propose an ensemble machine learning method that combines the results of several models to produce the final result.

**DISADVANTAGE:**

* The user evaluation can collect desired information such as usage problems or fitness of recommendation to users who have been invited for evaluation.
* This is a method for determining whether the prediction result is reliable in the case of the data scarcity problem

**PROPOSED SYSTEM:**

In this paper on spam or normal labeled datasets. The dataset consists of 6,431,471 crawled comments of which 481,334 comments were spam in the 6,407 videos that were most viewed between October 31, 2011 and January 17, 2012 in the United States. This dataset was mixed with English and non-English comments, so we extracted only English comments for the experiment. In addition, to make it similar to the data size used in the experiment of 3, we extracted 1,000 spam comments and normal comments, and compared them with 5,000 samples. In the experiment, we used an Extra-Tree Classifier with the techniques used in 3. Finally, we plotted the Precision, Recall, F1-score, and ROC curves by adding 1,000 data points from 1,000 to 5,000.

**ADVANTAGE:**

* The advantages are that it is very efficient in terms of storage space and computation time and handles noise and missing data well.
* Offline evaluation has the advantage of standardizing evaluation methods and evaluation items, and there are various evaluation items such as accuracy, coverage, confidence, and novelty.

**SYSTEM CONFIGURATION:**

# H/W SYSTEM CONFIGURATION:

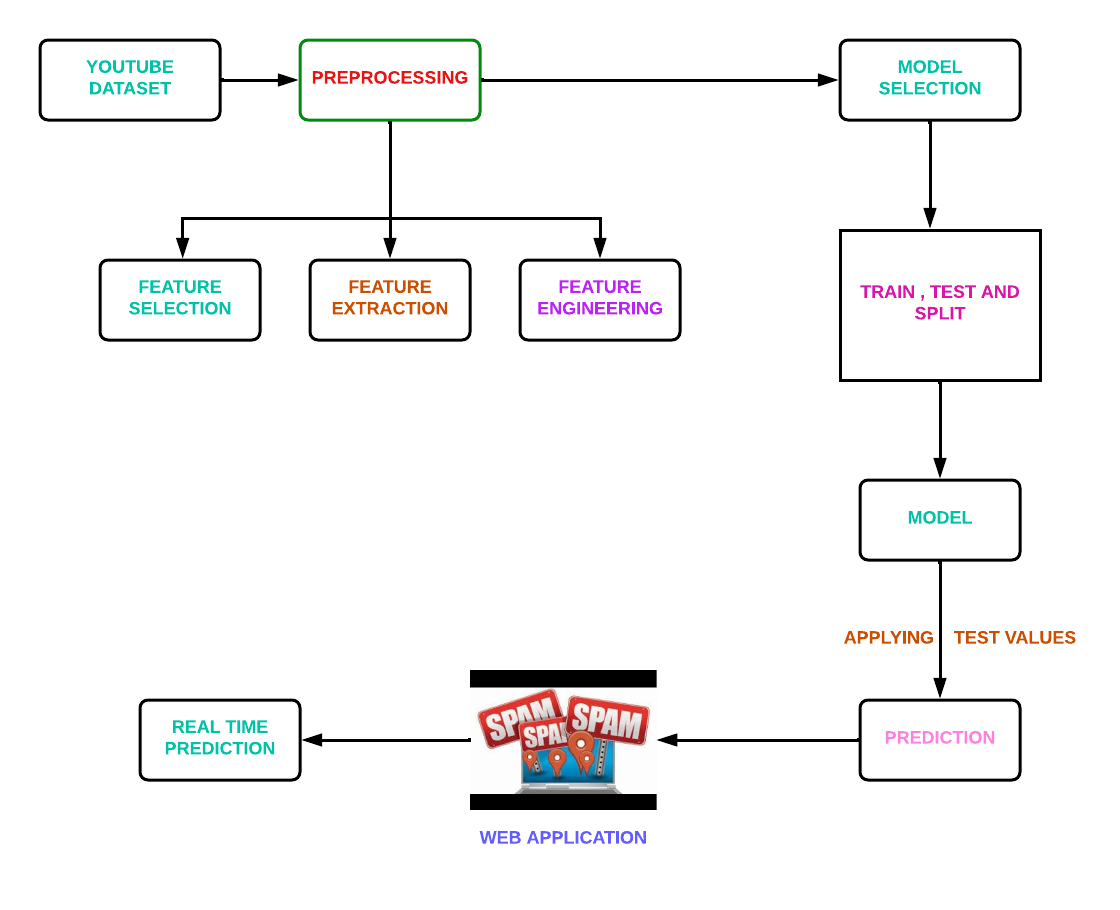
# Processor - I3, I5,I7

* RAM - 8 Gb
* Hard Disk - 500 GB

# S/W SYSTEM CONFIGURATION:

* Operating System - Windows 7/8/10
* Front End - Html,Css
* Scripts - python language

**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”.

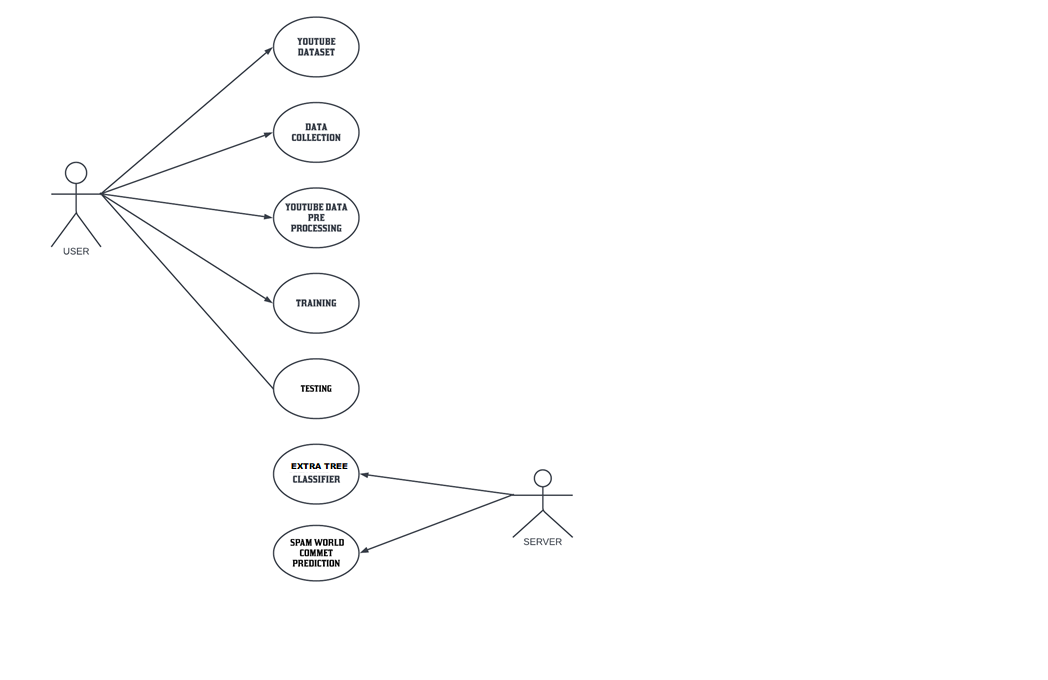
### The Extra Trees Classifier for feature selection offers several advantages:

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**

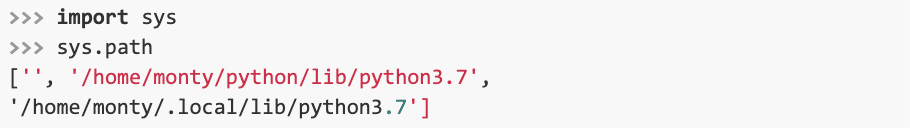
We’ll import a package using the **import** statement:

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

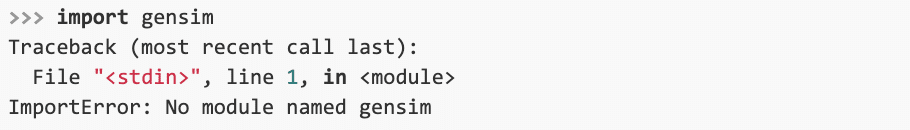
Let’s assume that we haven’t yet installed any packages. Python comes with a big collection of pre-installed packages known as the Python Standard Library. It includes tools for a range of use cases, such as text processing and doing math. Let’s import the latter:

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:



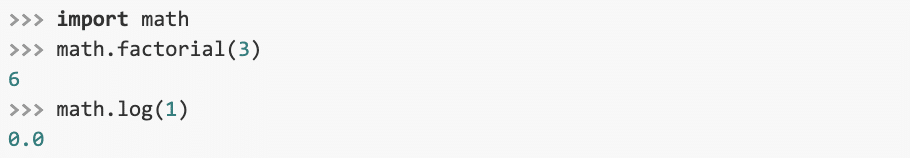
In such cases, check whether your imported package has been placed in one of Python’s search paths. If it hasn’t, you can always expand your list of search paths:

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”:



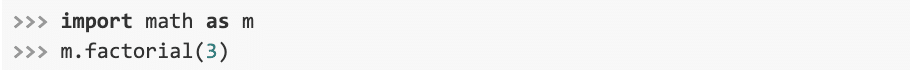
Assume that we were only interested in our math module’s factorial function, and that we’re also tired of using dot notation. In that case, we can proceed as follows:

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.