Machine Learning Workshop 2023

Session 1: Machine Learning Fundamentals

By

Engr. Sumayyea Salahuddin

Overview

- Artificial Intelligence.
- Machine Learning.
- Machine Learning Models.
    a) Supervised Learning.
    b) Unsupervised Learning.
    c) Reinforcement Learning.
- Machine Learning Algorithms.
- Summary.
- References.

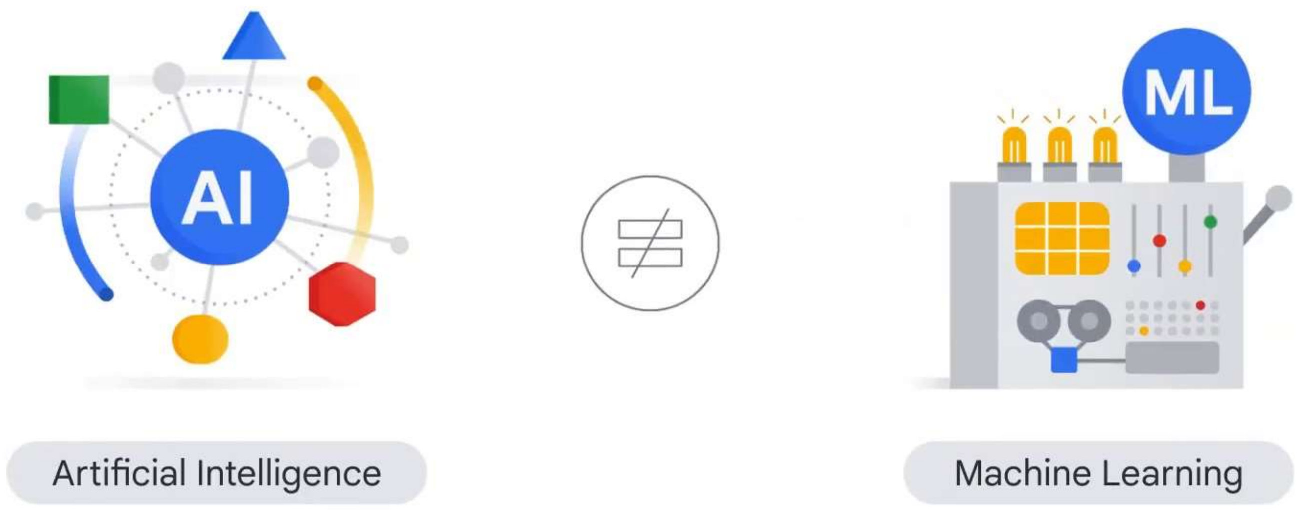
Artificial Intelligence (AI)

AI is the theory and development of computer systems able to perform tasks normally requiring human intelligence.

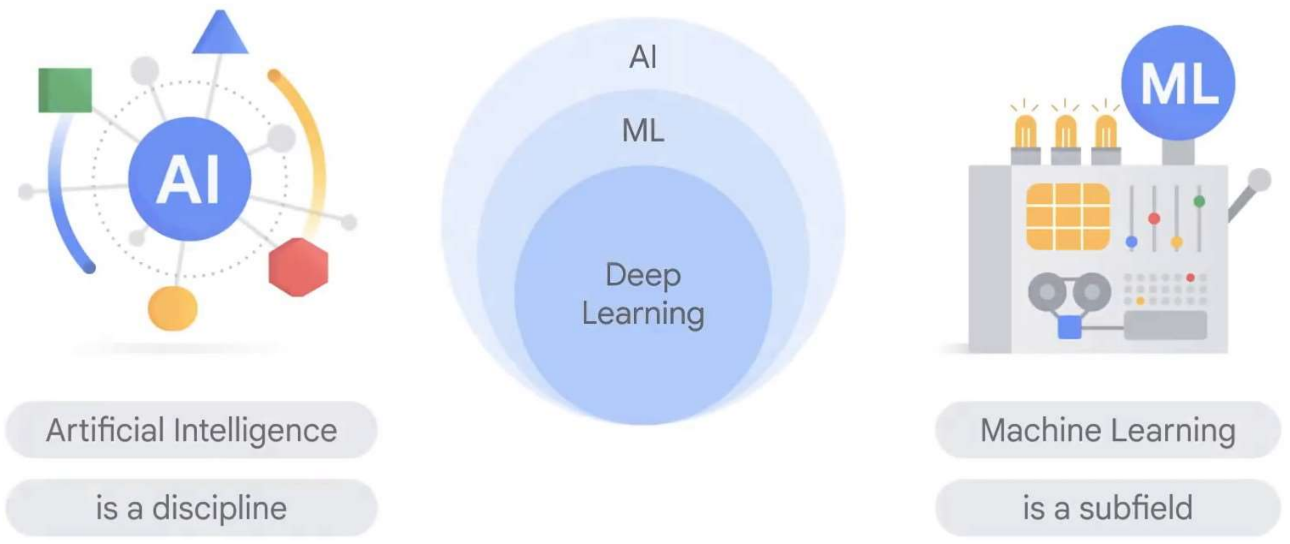
Machine Learning (ML)

ML gives computers the ability to learn without explicit programming.

Artificial Intelligence vs. Machine Learning



Artificial Intelligence vs. Machine Learning (Cont.)



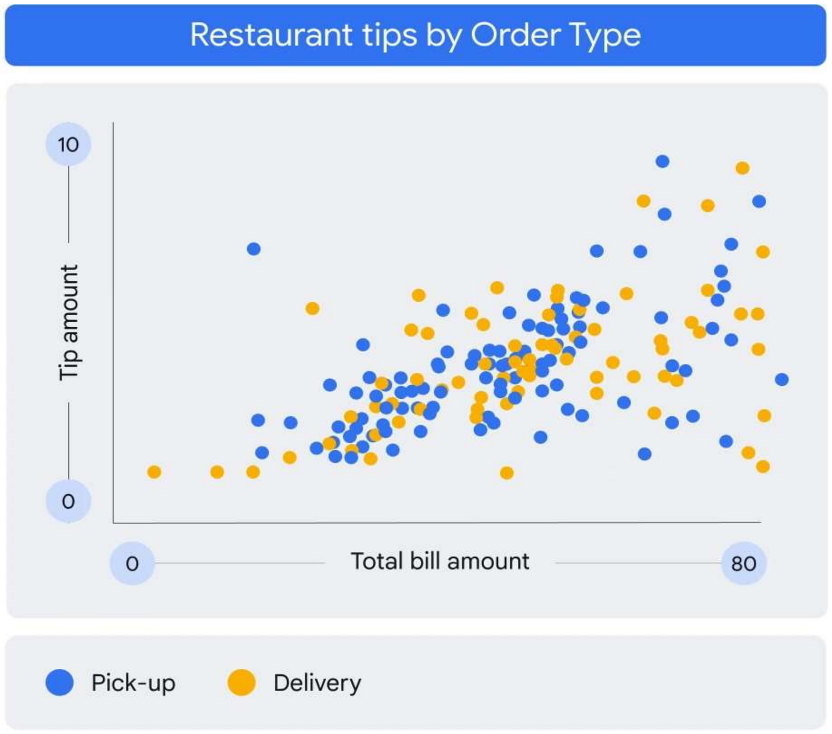
ML Models

- Unsupervised ML models
- Supervised ML models
- Reinforcement ML models

Supervised Learning

Supervised learning implies the data is already labeled

In supervised learning we are learning from past examples to predict future values.

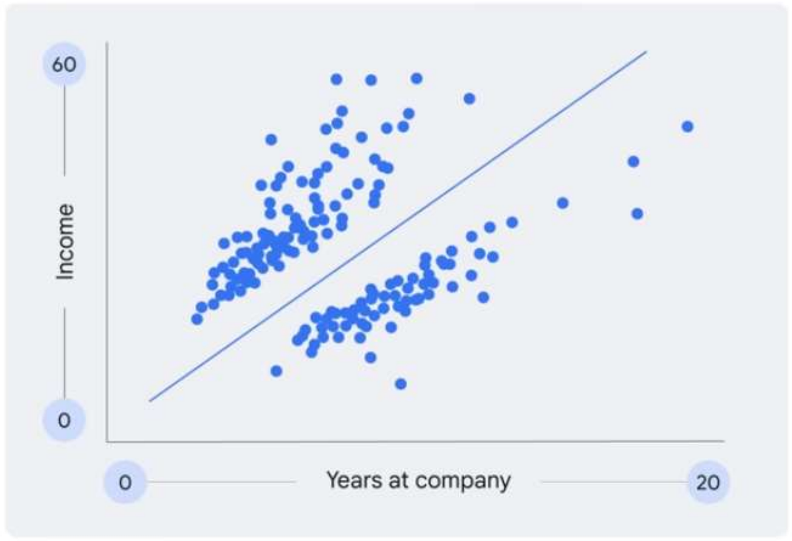


Unsupervised Learning

Unsupervised learning implies the data is not labeled

Unsupervised problems are all about looking at the raw data, and seeing if it naturally falls into groups

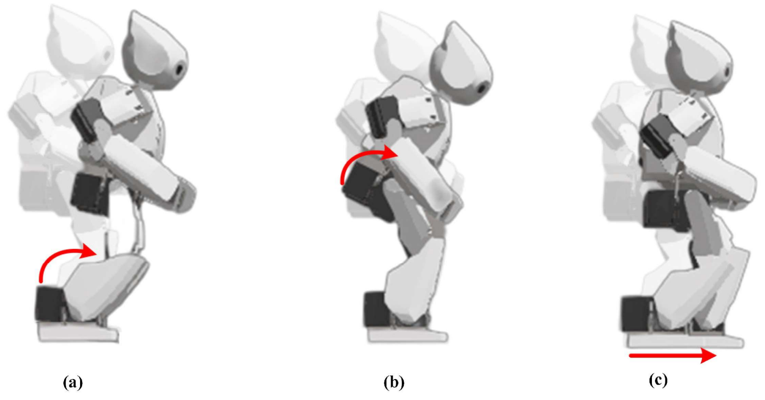
Income vs Job tenure



Example Model: Clustering
Is this employee on the "fast-track" or not?

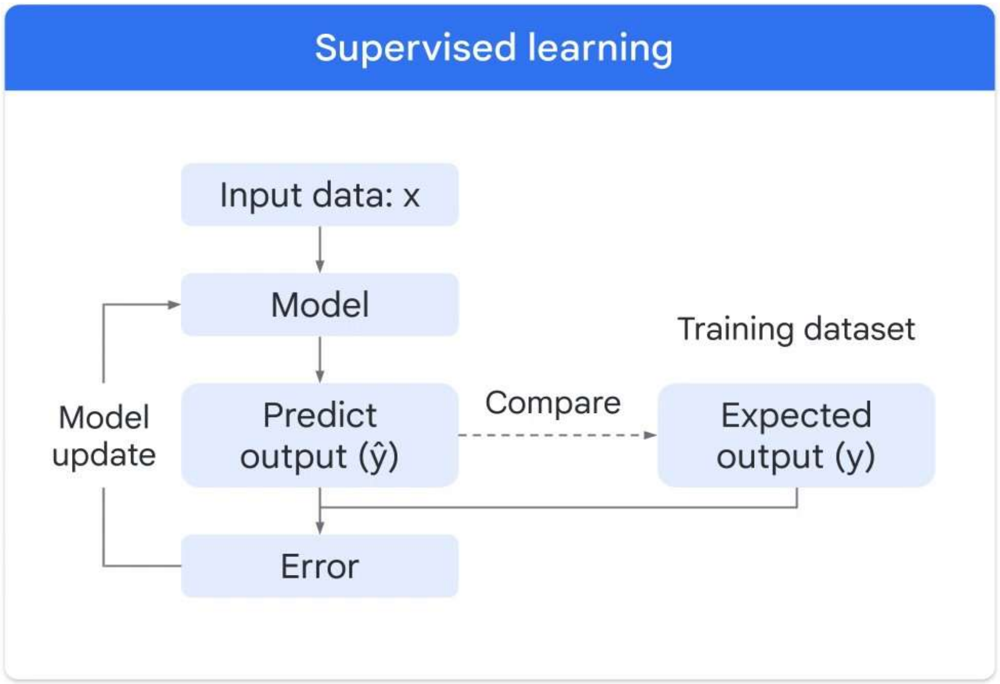
Reinforcement Learning

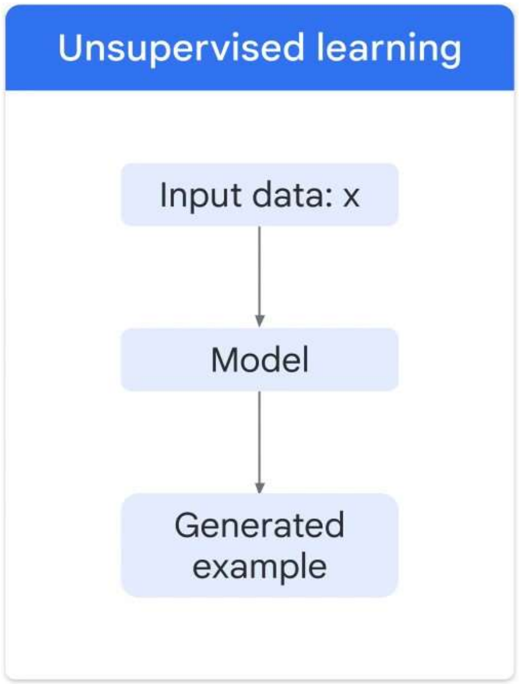
Reinforcement learning (RL) is a type of machine learning that allows an agent to learn how to behave in an environment by trial and error. The agent is rewarded for taking actions that lead to desired outcomes, and penalized for taking actions that lead to undesired outcomes. Over time, the agent learns to take the actions that maximize its rewards.



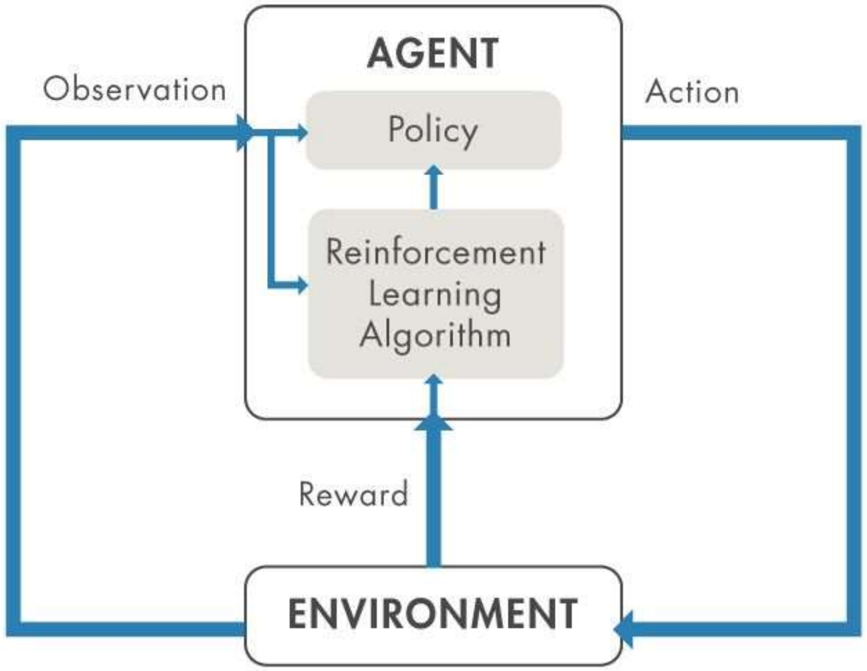
Example: Imagine a robot that is learning to walk.
The robot is rewarded for moving forward, and penalized for falling down. Over time, the robot learns to take the steps necessary to move forward without falling.

Supervised Learning vs. Unsupervised Learning

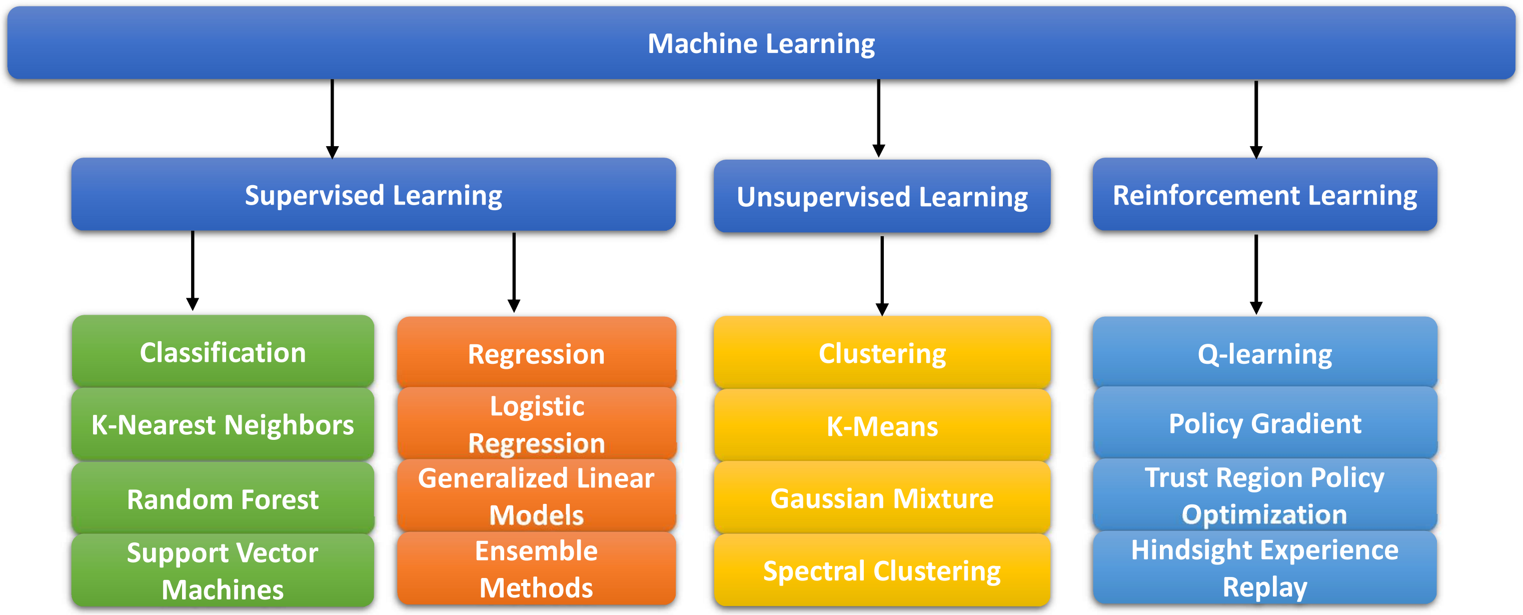




Reinforcement Learning



Machine Learning Algorithms



Summary

- **Artificial Intelligence (AI):**
  - AI refers to the development of computer systems that can perform tasks that typically require human intelligence, such as understanding natural language, recognizing patterns, solving problems, and making decisions.
- **Machine Learning (ML):**
  - ML is a subset of AI that focuses on teaching computers to learn from data. It involves creating algorithms and models that can improve their performance over time through experience.
- **Machine Learning Models:**
  - Machine learning models are mathematical representations of real-world processes. They are designed to make predictions or decisions based on input data. Examples include logistic regression, support vector machines, and random forest.

Summary (Cont.)

- **Supervised Learning:**
  - Supervised learning is a type of machine learning where the algorithm learns from labeled data, which means it is provided with input-output pairs during training. The goal is to learn a mapping from inputs to outputs so that it can make predictions on new, unseen data.
- **Unsupervised Learning:**
  - Unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data. It tries to find patterns, structures, or groupings within the data without any predefined output. Clustering and dimensionality reduction are common tasks in unsupervised learning.
- **Reinforcement Learning:**
  - Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with an environment. The agent receives rewards or penalties based on its actions, and its goal is to learn a policy that maximizes cumulative rewards over time.

Summary (Cont.)

- **Machine Learning Algorithms:**
  - Machine learning algorithms are the specific techniques used to train machine learning models. They include methods like linear regression, random forests, k-nearest neighbors, k-means clustering, deep neural networks, and many others. Each algorithm has its own strengths and weaknesses, making them suitable for different types of problems.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson Higher Ed, 2021.

[2] S. Raschka and V. Mirjalili, *Python Machine Learning*. 2019.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2016.

[4] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python*. "O'Reilly Media, Inc.," 2016.

[5] A. A. Patel, *Hands-On Unsupervised Learning Using Python*. "O'Reilly Media, Inc.," 2019.

[6] R. S. Sutton, A. G. Barto, and C.-D. A. L. L. A. G. Barto, *Reinforcement Learning*. MIT Press, 1998.

[7] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2022.

Machine Learning Workshop 2023

Session 2: About Git and GitHub

By

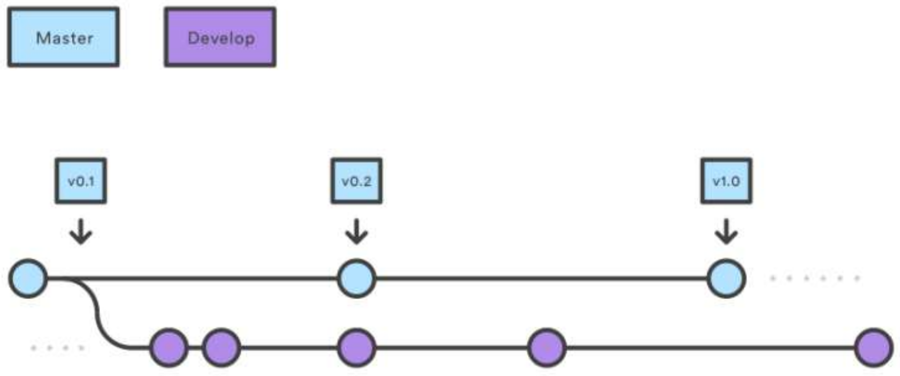
Engr. Sumayyea Salahuddin

Overview

- What is Git?
- Git Installation.
- Some Useful Git Commands.
- How to use Git?
- What is GitHub?
- Demo.
- Summary.
- References.

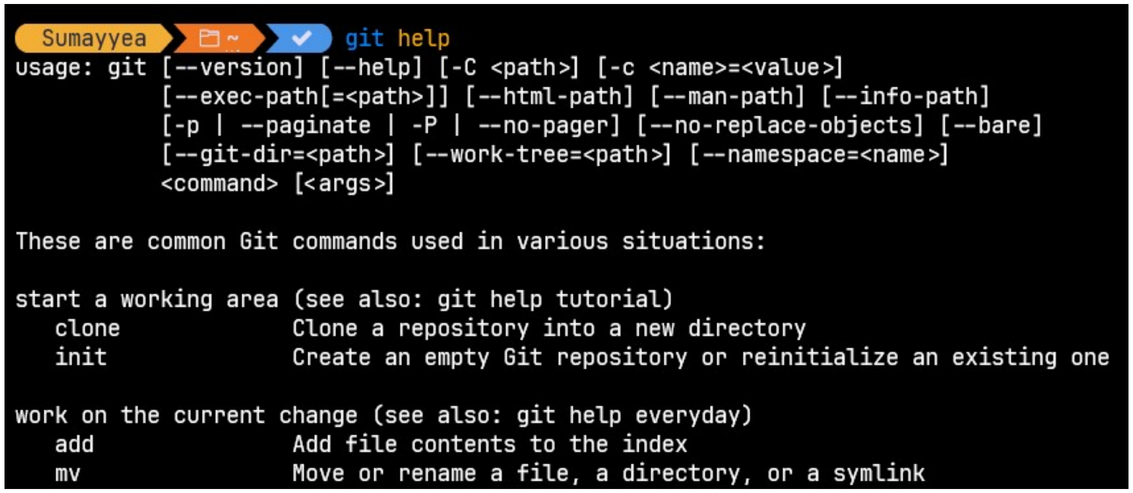
What it Git?

- Created by Linux Torvalds in 2005.
- It is a Distributed Version Control system.
    - To track every change.
    - Maintain different versioning for Dev, QA, Prod.
    - Easy for collaboration.
- It is written in collection of Perl, C, and shell scripts.
- It is used by Google, Quora, Facebook, Netflix, reddit, Lyft etc.
- Latest version (2.42.0): August, 2023.
- Visit Git at: https://git-scm.com/.



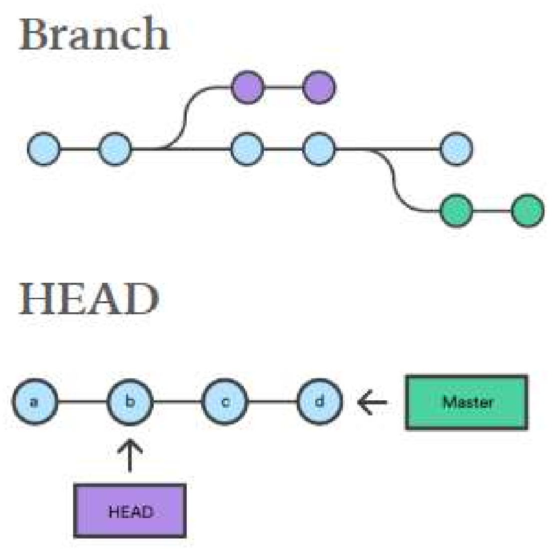
Git Installation

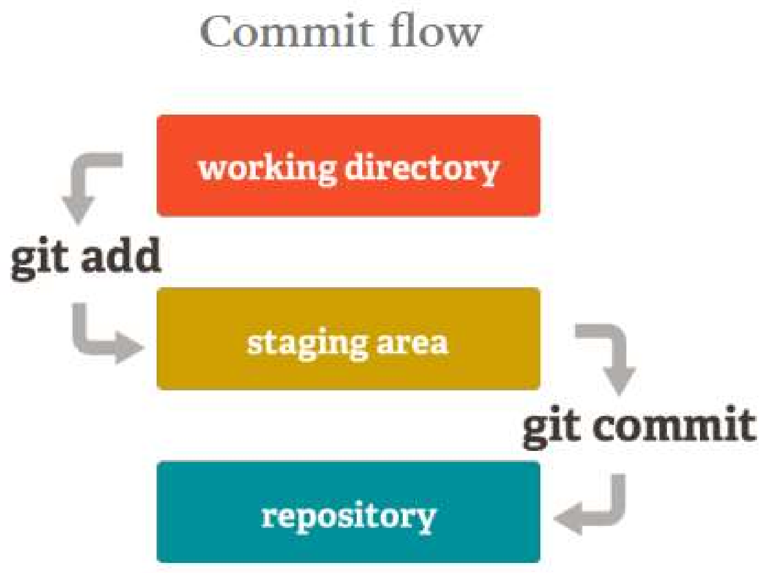
- **Windows**.
  - Download windows git installer from https://git-scm.com/download/win.
  - Install it with default options.
- **Linux**.
  - Follow instructions given on https://git-scm.com/download/linux.
- **MacBook**.
  - Follow instructions given on https://git-scm.com/download/mac.



```
Sumayyea  ~    git help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
```

About Git Branch, Head, Master, & Commit





Some Useful Git Commands

1. **git clone**
   - ✓ Let's say your friend has a folder with some cool stuff in it, and you want a copy of everything they have. Using **git clone**, you can make an exact copy of your friend's folder on your computer. It's like making a duplicate of their stuff so you can work on it or see what they're up to.
2. **git init**
   - ✓ Imagine you have a folder on your computer, and you want to start keeping track of changes you make to the files in that folder. Running **git init** in that folder is like telling your computer, *"Hey, this folder is now a place where I'll use Git to track changes.*" It sets up a special system for recording those changes.
3. **git add**
   - ✓ When you make changes to the files in your Git-tracked folder, Git doesn't automatically pay attention to those changes. You have to tell Git which changes you want to keep track of. **git add** is like telling Git, "*Hey, I've made some changes here, and I want you to remember them.*" It's like putting those changes in a box to show Git later.

Some Useful Git Commands (Cont.)

4. **git rm**

   ✓ Sometimes, you want to remove a file from your Git-tracked folder. **git rm** is like telling Git, "*I want to get rid of this file, so don't pay attention to it anymore.*" It's like throwing something out of the box you showed Git earlier.

5. **git commit**

   ✓ After you've added your changes with git add, you need to tell Git to save those changes as a snapshot. **git commit** is like taking a picture of your changes and giving it a description, so you can remember what you did. This way, you can always go back to that point in time if you need to.

6. **git log**

   ✓ Git keeps a record of all the snapshots you've taken with git commit. **git log** is like looking at a history book of all the changes you've made. It shows you a list of snapshots, who made them, and when they were made. It's like a timeline of all your work.

Some Useful Git Commands (Cont.)

7. **git branch**
   - ✓ Think of your project as a tree with different branches. Each branch represents a different line of work or a feature you're developing. **git branch** is like creating a new branch on that tree. It allows you to work on separate ideas or tasks without affecting the main project until you're ready to merge your changes.

8. **git push**
   - ✓ Imagine you and your friend are working on a shared document, and you've made some changes. When you're ready to share your changes with your friend, you "push" those changes to a common place, like a shared online folder. In Git, **git push** is like sending your changes to a central location, so others can see and use them.

9. **git pull**
   - ✓ Now, imagine your friend has made some updates to the shared document, and you want to get those updates to work on the latest version. **git pull** is like grabbing the latest changes from that central location (like the shared online folder) and bringing them to your own copy. It ensures you're working with the most up-to-date information.

Some Useful Git Commands (Cont.)

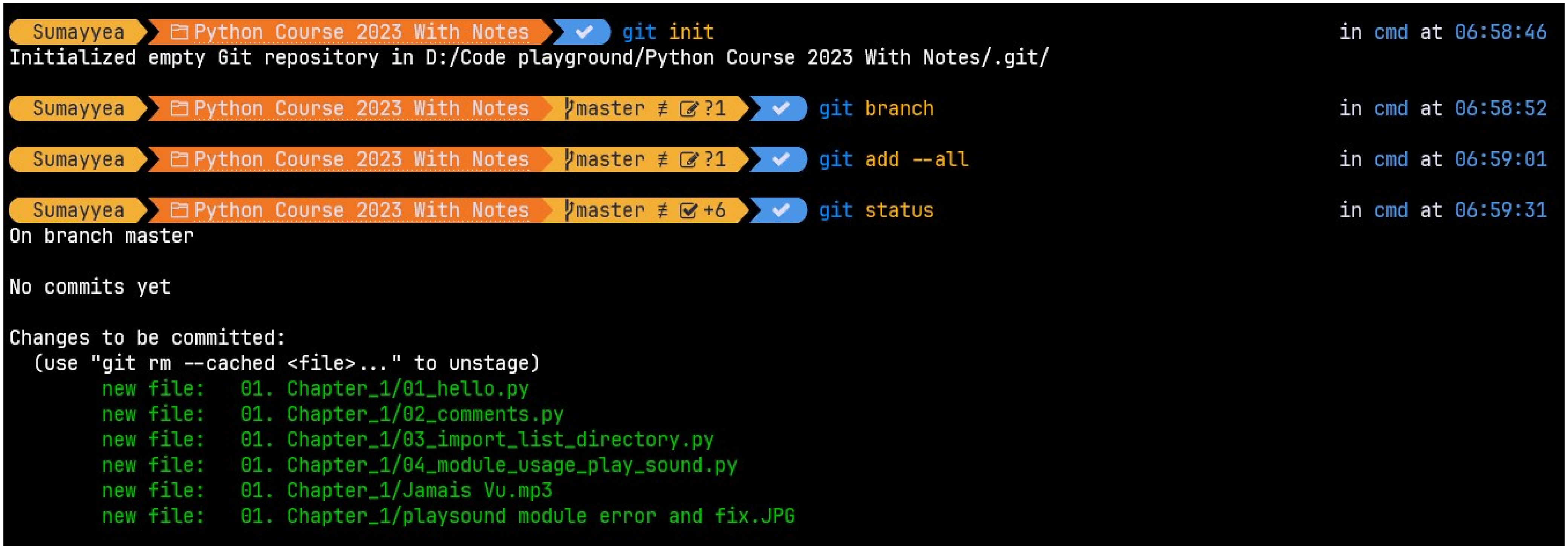
10. git merge

- ✓ When you've finished working on a branch and want to combine your changes with the main project, you "merge" your branch back into the main branch. It's like taking the changes you made on a side path and smoothly integrating them into the main road. **git merge** is the command that does this in Git, bringing together the work from different branches into one cohesive whole.

How to use Git?

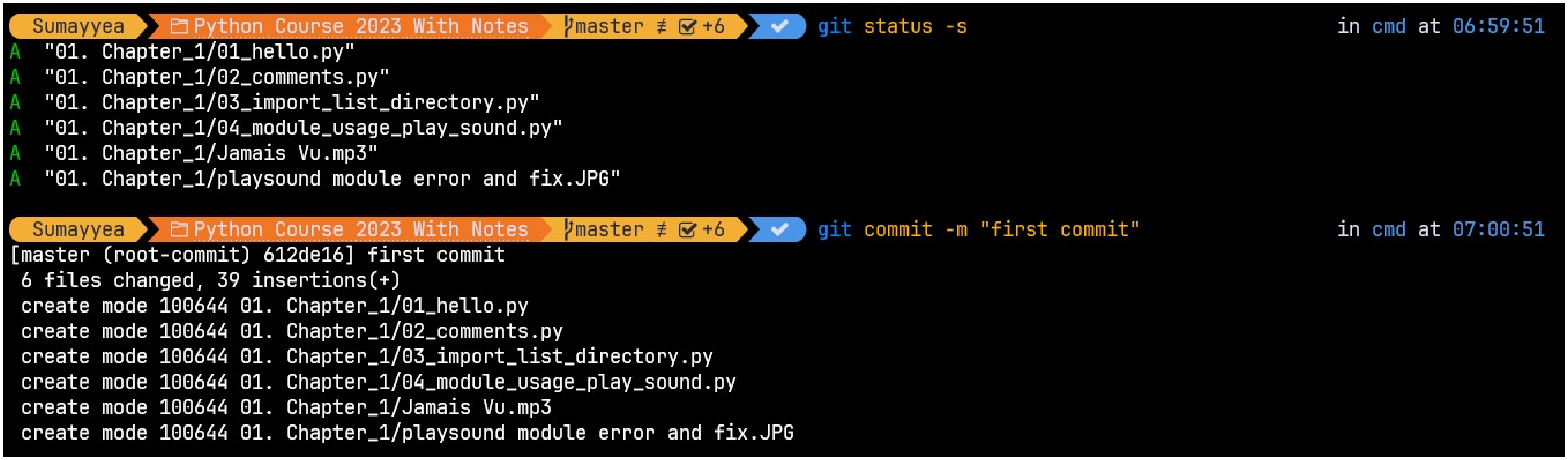
- So, **Git** is like a **tool** that keeps track of changes in your digital files.
- You can use it on your own computer to track changes in your files. Imagine you're writing a story, and you want to save every version you write, like drafts. Git helps you do that.
- Now, let's say you want to share your story with others, maybe to get feedback or work together. You can take your Git-tracked story and put it on **GitHub**. This is like putting your magic notebook on a big public shelf in the library (GitHub).
- GitHub is like a website that helps you share and collaborate on those tracked changes with other people.

Practical Demo 1 – Using Git Locally



```
 Sumayyea  Python Course 2023 With Notes  ✔ git init                              in cmd at 06:58:46
Initialized empty Git repository in D:/Code playground/Python Course 2023 With Notes/.git/

 Sumayyea  Python Course 2023 With Notes  master ≢ ?1  ✔ git branch               in cmd at 06:58:52

 Sumayyea  Python Course 2023 With Notes  master ≢ ?1  ✔ git add --all            in cmd at 06:59:01

 Sumayyea  Python Course 2023 With Notes  master ≢ +6  ✔ git status               in cmd at 06:59:31
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   01. Chapter_1/01_hello.py
        new file:   01. Chapter_1/02_comments.py
        new file:   01. Chapter_1/03_import_list_directory.py
        new file:   01. Chapter_1/04_module_usage_play_sound.py
        new file:   01. Chapter_1/Jamais Vu.mp3
        new file:   01. Chapter_1/playsound module error and fix.JPG
```

Practical Demo 1 – Using Git Locally (Cont.)



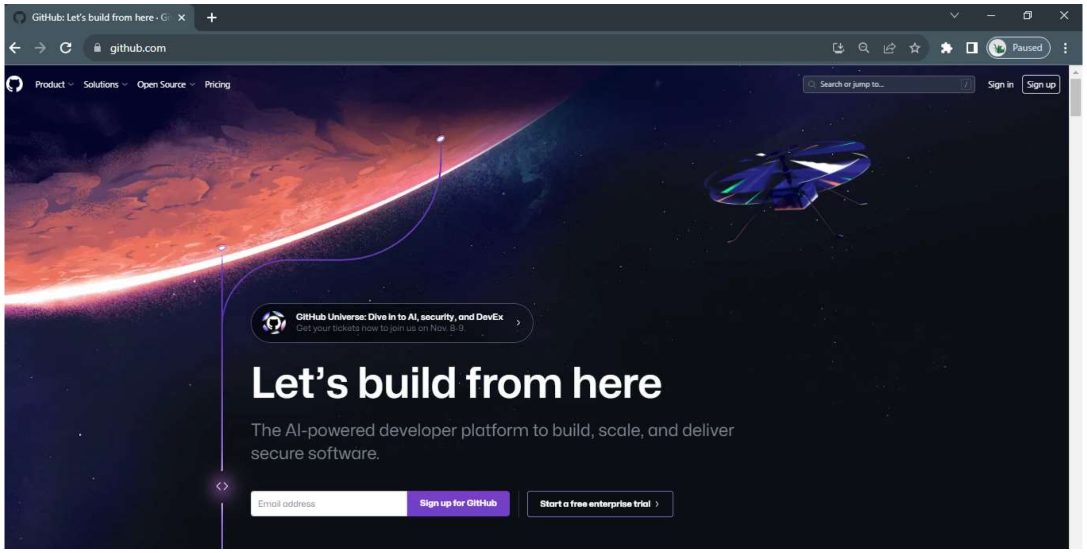
```
 Sumayyea   Python Course 2023 With Notes   master ≢ ☑ +6   ✔  git status -s                        in cmd at 06:59:51
A  "01. Chapter_1/01_hello.py"
A  "01. Chapter_1/02_comments.py"
A  "01. Chapter_1/03_import_list_directory.py"
A  "01. Chapter_1/04_module_usage_play_sound.py"
A  "01. Chapter_1/Jamais Vu.mp3"
A  "01. Chapter_1/playsound module error and fix.JPG"

 Sumayyea   Python Course 2023 With Notes   master ≢ ☑ +6   ✔  git commit -m "first commit"         in cmd at 07:00:51
[master (root-commit) 612de16] first commit
 6 files changed, 39 insertions(+)
 create mode 100644 01. Chapter_1/01_hello.py
 create mode 100644 01. Chapter_1/02_comments.py
 create mode 100644 01. Chapter_1/03_import_list_directory.py
 create mode 100644 01. Chapter_1/04_module_usage_play_sound.py
 create mode 100644 01. Chapter_1/Jamais Vu.mp3
 create mode 100644 01. Chapter_1/playsound module error and fix.JPG
```

What it GitHub?

- Online Code repository.
- Project Management.
- Team Management.
- Helps in secure development.
- Better Code Review.
- Branches.
- Bug Tracking.
- Read More here: https://github.com/features.

How to Use GitHub?

- Visit https://github.com/ to create your account.



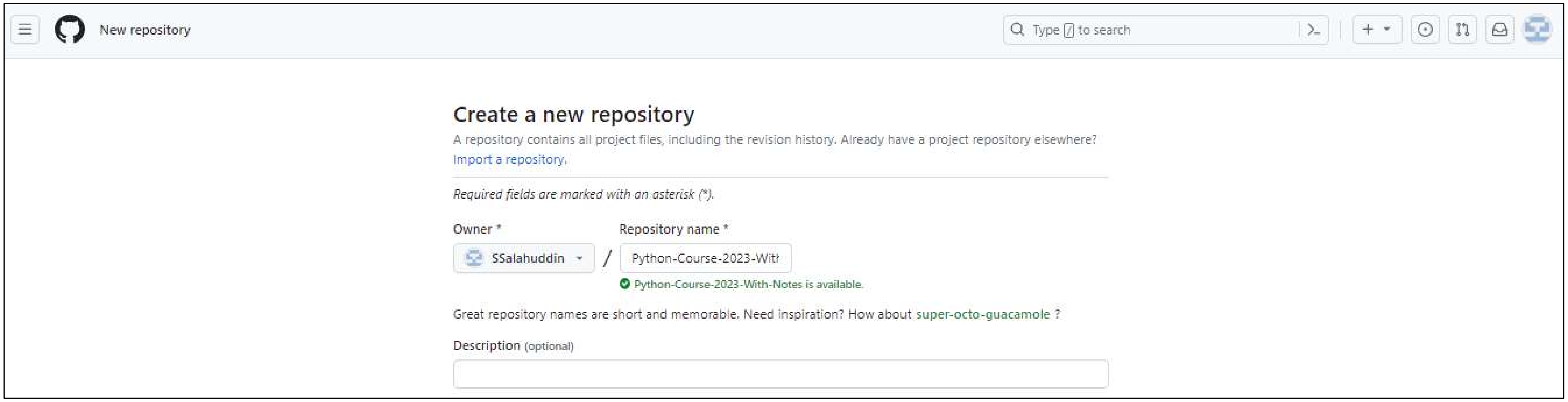
GitHub: Let's build from here · G
github.com
Product
Solutions
Open Source
Pricing
Search or jump to...
Sign in
Sign up
GitHub Universe: Dive in to AI, security, and DevEx
Get your tickets now to join us on Nov. 8-9.
Let's build from here
The AI-powered developer platform to build, scale, and deliver secure software.
Email address
Sign up for GitHub
Start a free enterprise trial

- Once your account is created, **Sign in to GitHub**.

Practical Demo 2 – Using Git via GitHub

1. Setting Up Repository.
2. Uploading Files.
3. Optional:
    a) Add a README file
    b) Add a .gitignore file

Practical Demo 2 – Step 1: Setting Up Repository.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

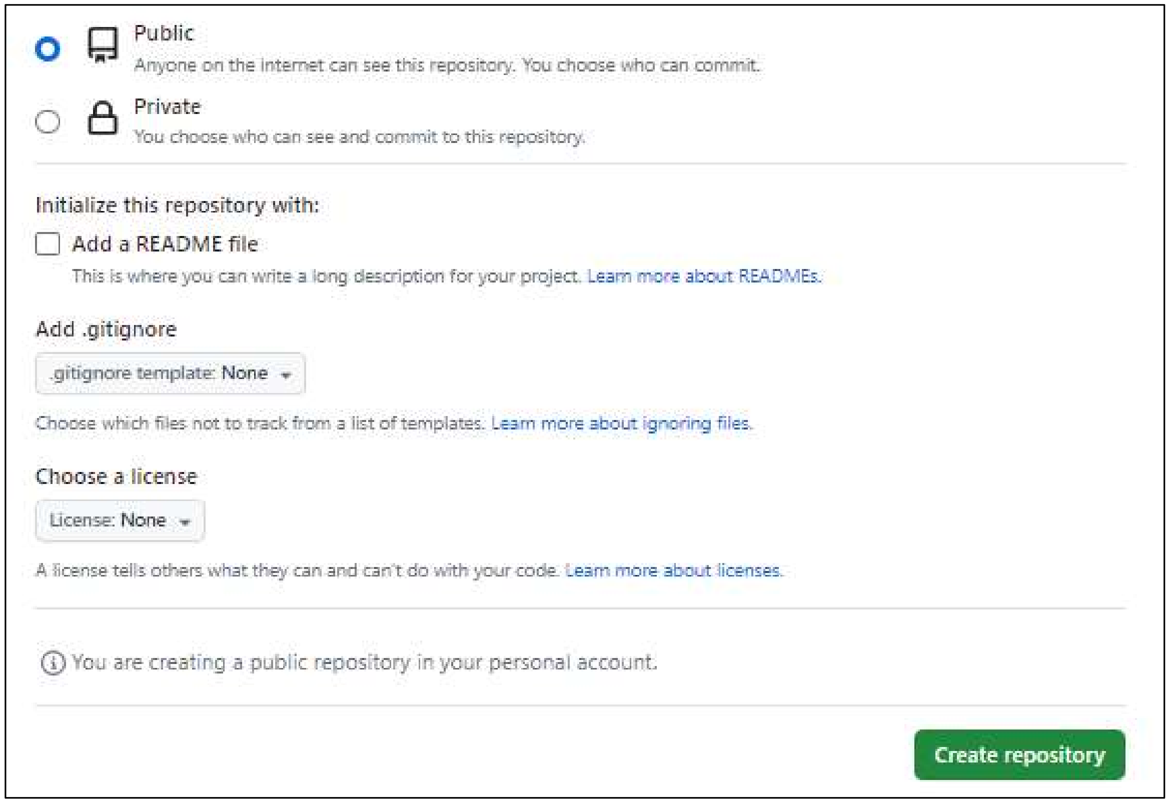
Owner * SSalahuddin / Repository name * Python-Course-2023-With

Python-Course-2023-With-Notes is available.

Great repository names are short and memorable. Need inspiration? How about super-octo-guacamole ?

Description (optional)

Practical Demo 2 – Step 1: Setting Up Repository (Cont.)



[✓ Public]
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

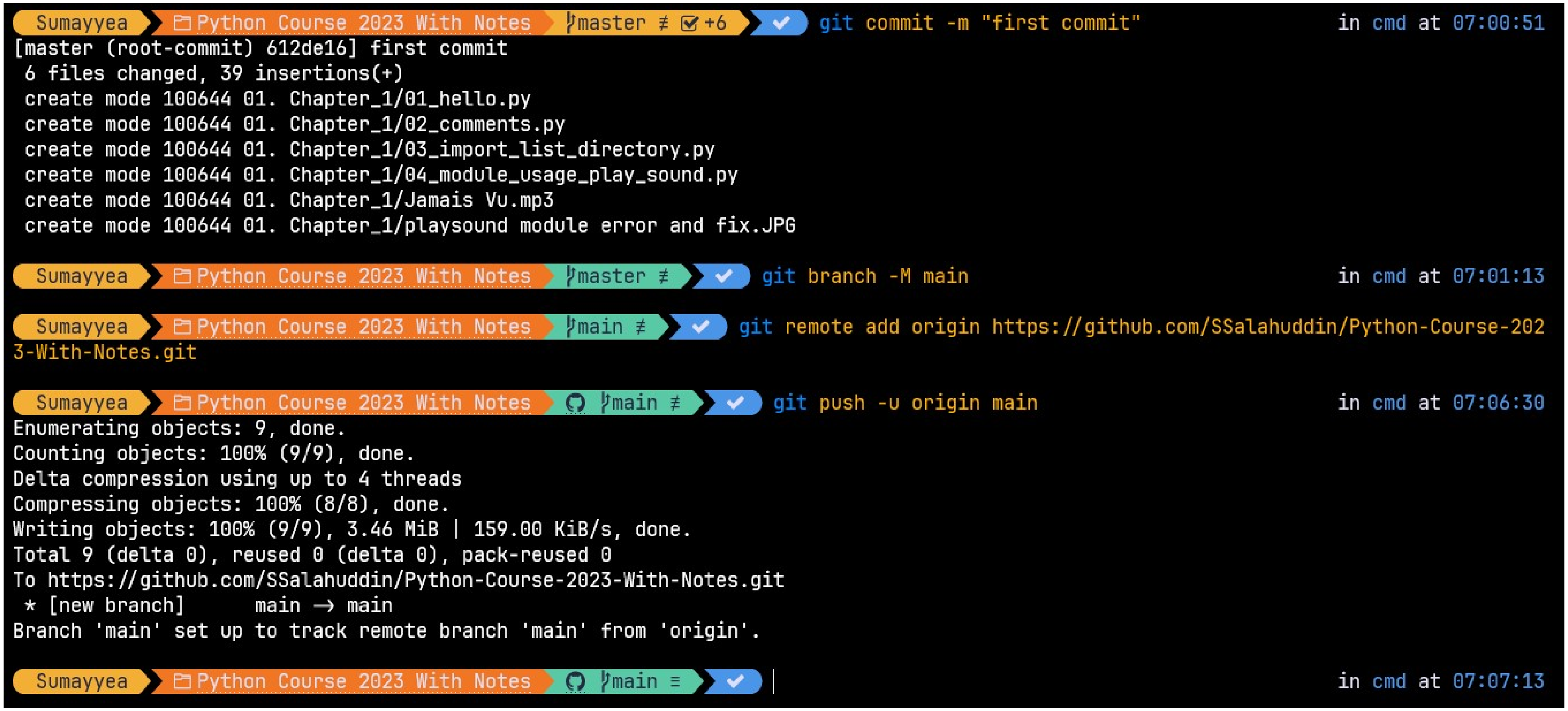
Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

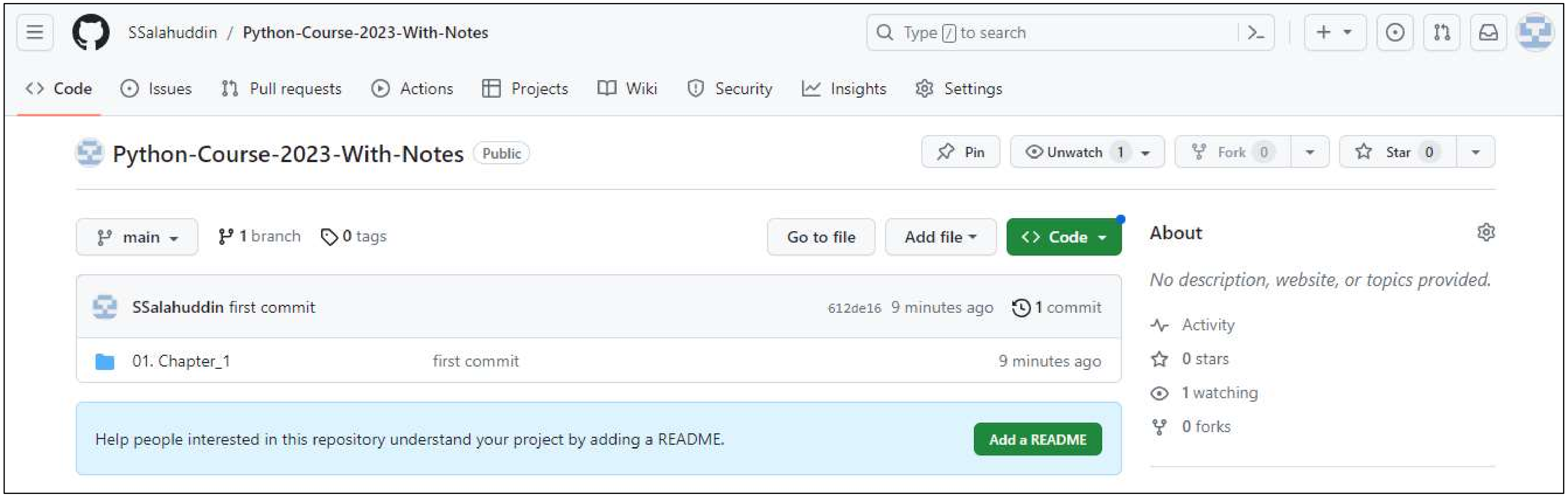
ⓘ You are creating a public repository in your personal account.

Practical Demo 2 – Step 2: Uploading Files.



```
Sumayyea  Python Course 2023 With Notes  master ≠ ☑ +6  ✔ git commit -m "first commit"                        in cmd at 07:00:51
[master (root-commit) 612de16] first commit
 6 files changed, 39 insertions(+)
 create mode 100644 01. Chapter_1/01_hello.py
 create mode 100644 01. Chapter_1/02_comments.py
 create mode 100644 01. Chapter_1/03_import_list_directory.py
 create mode 100644 01. Chapter_1/04_module_usage_play_sound.py
 create mode 100644 01. Chapter_1/Jamais Vu.mp3
 create mode 100644 01. Chapter_1/playsound module error and fix.JPG

Sumayyea  Python Course 2023 With Notes  master ≠  ✔ git branch -M main                                       in cmd at 07:01:13

Sumayyea  Python Course 2023 With Notes  main ≠  ✔ git remote add origin https://github.com/SSalahuddin/Python-Course-202
3-With-Notes.git

Sumayyea  Python Course 2023 With Notes  main ≠  ✔ git push -u origin main                                    in cmd at 07:06:30
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 3.46 MiB | 159.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/SSalahuddin/Python-Course-2023-With-Notes.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

Sumayyea  Python Course 2023 With Notes  main ≡  ✔ |                                                          in cmd at 07:07:13
```

Practical Demo 2 – Output



Python-Course-2023-With-Notes Public

1 branch 0 tags

SSalahuddin first commit 612de16 9 minutes ago 1 commit

01. Chapter_1 first commit 9 minutes ago

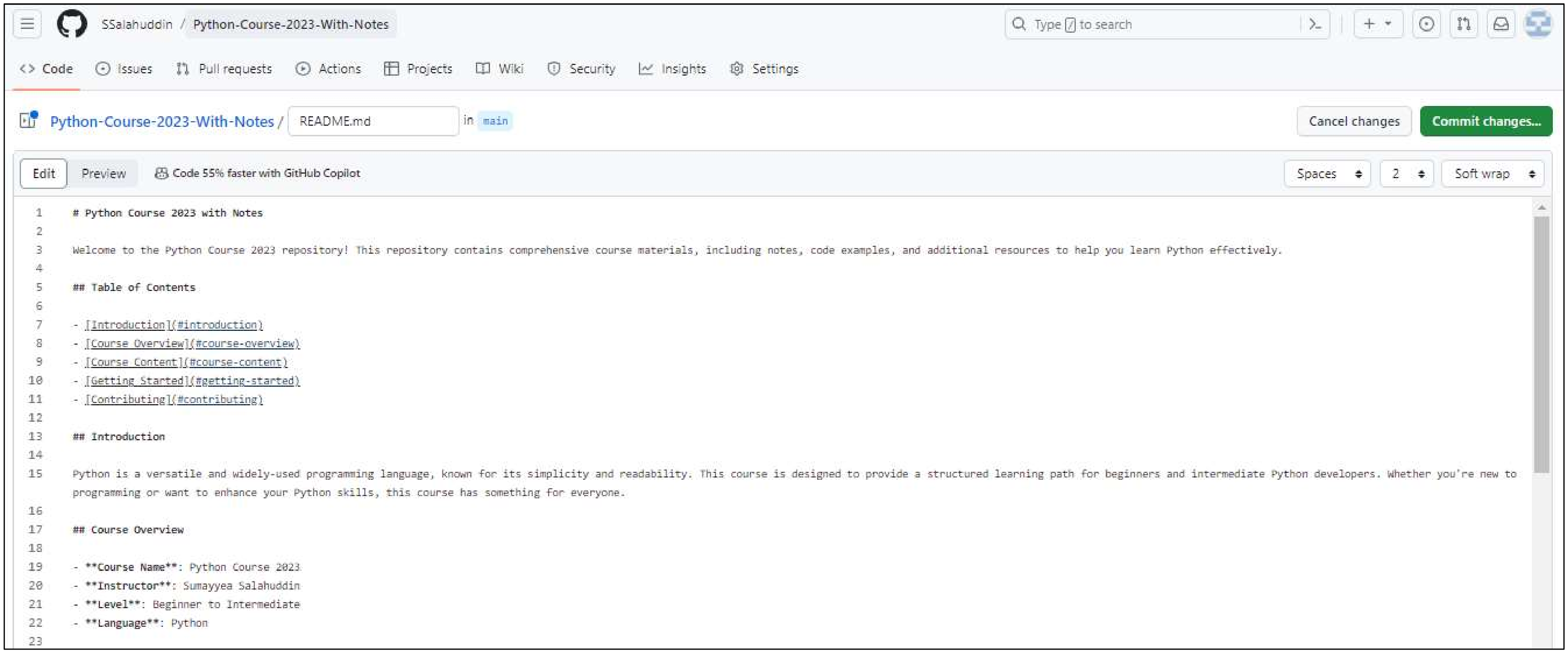
Help people interested in this repository understand your project by adding a README.

About

No description, website, or topics provided.

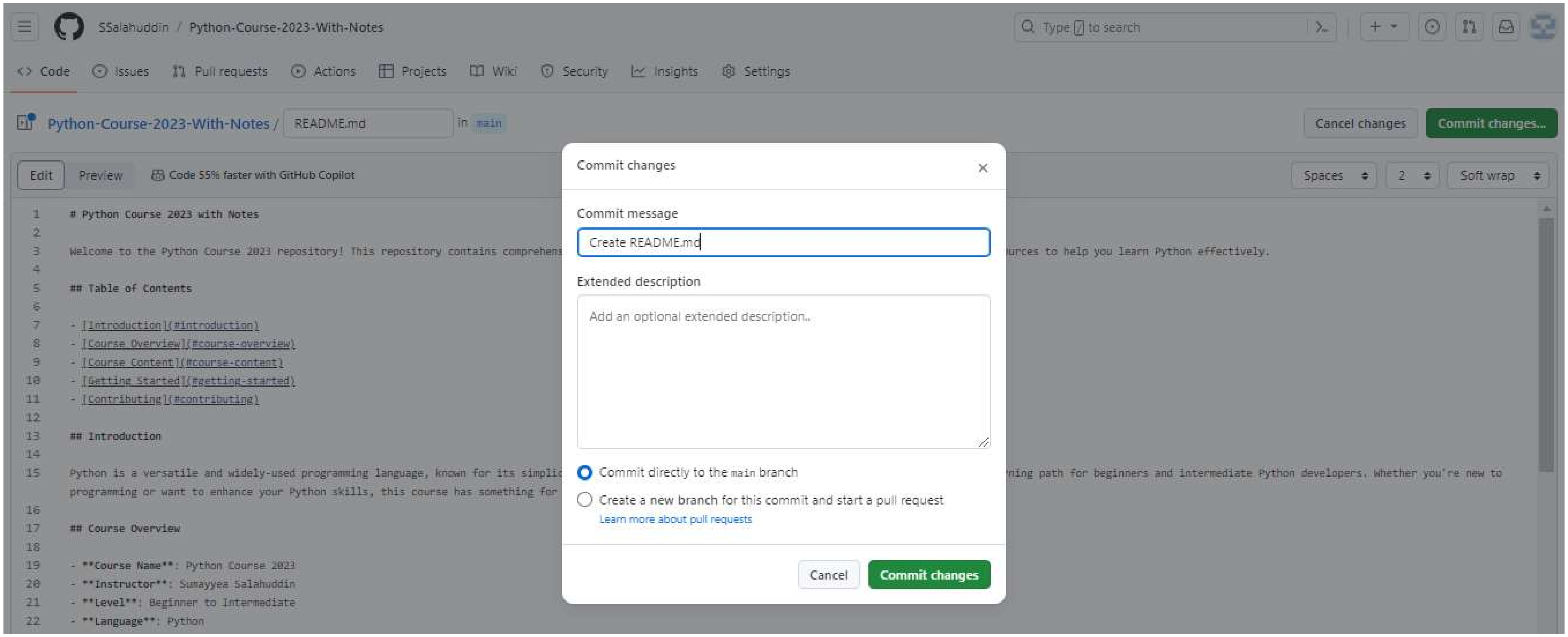
Activity
0 stars
1 watching
0 forks

Practical Demo 2 – Step 3: Add a README file.



```
# Python Course 2023 with Notes

Welcome to the Python Course 2023 repository! This repository contains comprehensive course materials, including notes, code examples, and additional resources to help you learn Python effectively.

## Table of Contents

- [Introduction](#introduction)
- [Course Overview](#course-overview)
- [Course Content](#course-content)
- [Getting Started](#getting-started)
- [Contributing](#contributing)

## Introduction

Python is a versatile and widely-used programming language, known for its simplicity and readability. This course is designed to provide a structured learning path for beginners and intermediate Python developers. Whether you're new to programming or want to enhance your Python skills, this course has something for everyone.

## Course Overview

- **Course Name**: Python Course 2023
- **Instructor**: Sumayyea Salahuddin
- **Level**: Beginner to Intermediate
- **Language**: Python

```

Practical Demo 2 – Step 3: Add a README file (Cont.)



```
# Python Course 2023 with Notes

Welcome to the Python Course 2023 repository! This repository contains comprehens                urces to help you learn Python effectively.

## Table of Contents

- [Introduction](#introduction)
- [Course Overview](#course-overview)
- [Course Content](#course-content)
- [Getting Started](#getting-started)
- [Contributing](#contributing)

## Introduction

Python is a versatile and widely-used programming language, known for its simplic                ning path for beginners and intermediate Python developers. Whether you're new to
programming or want to enhance your Python skills, this course has something for

## Course Overview

- **Course Name**: Python Course 2023
- **Instructor**: Sumayyea Salahuddin
- **Level**: Beginner to Intermediate
- **Language**: Python
```

Commit changes

Commit message: Create README.md

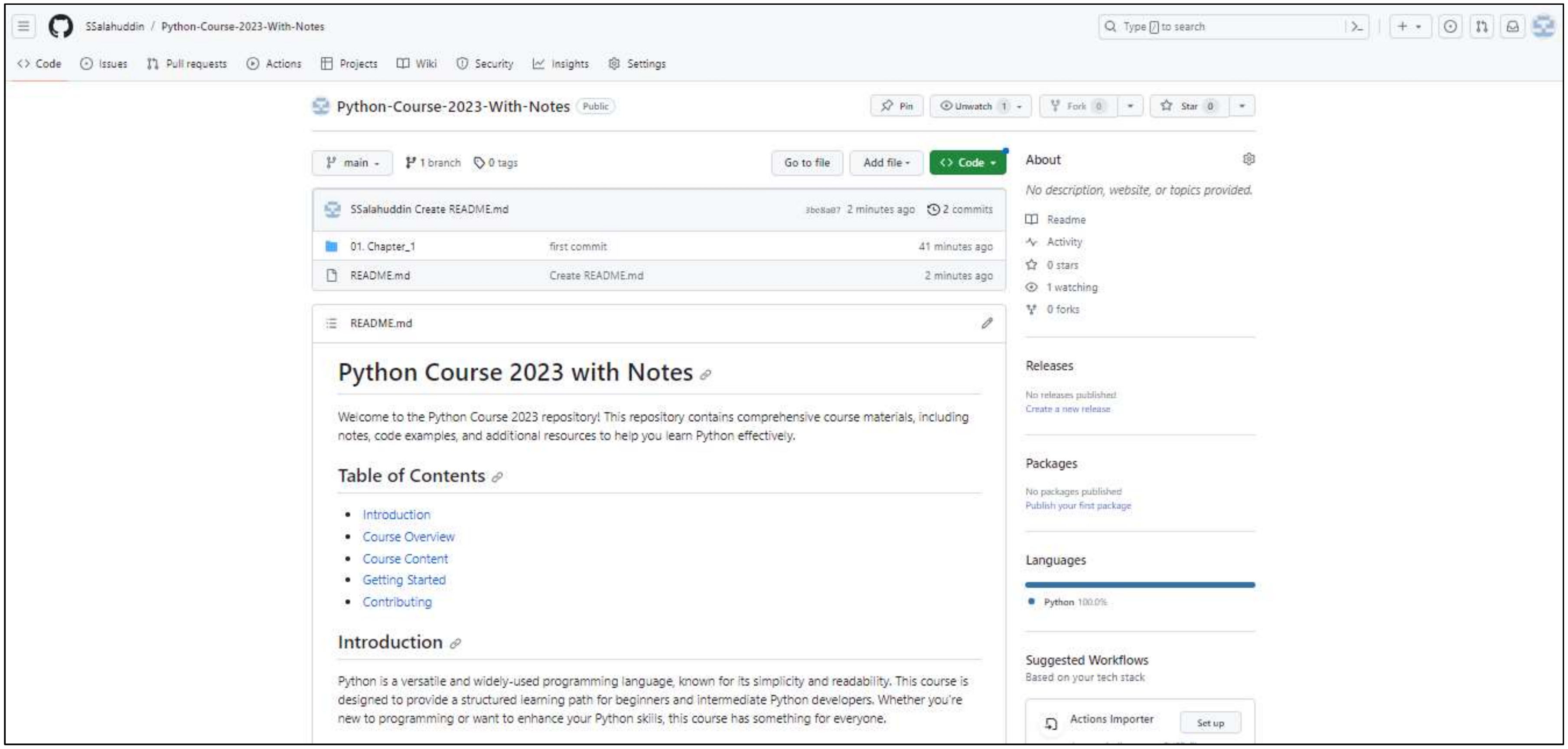
Extended description: Add an optional extended description..

[✓ Commit directly to the main branch]

Create a new branch for this commit and start a pull request

Learn more about pull requests

Practical Demo 2 – Output



Python-Course-2023-With-Notes Public

SSalahuddin Create README.md 3be8a07 2 minutes ago 2 commits

01. Chapter_1	first commit	41 minutes ago
README.md	Create README.md	2 minutes ago

README.md

Python Course 2023 with Notes

Welcome to the Python Course 2023 repository! This repository contains comprehensive course materials, including notes, code examples, and additional resources to help you learn Python effectively.

Table of Contents

- Introduction
- Course Overview
- Course Content
- Getting Started
- Contributing

Introduction

Python is a versatile and widely-used programming language, known for its simplicity and readability. This course is designed to provide a structured learning path for beginners and intermediate Python developers. Whether you're new to programming or want to enhance your Python skills, this course has something for everyone.

About

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● Python 100.0%

Suggested Workflows
Based on your tech stack

Actions Importer Set up

About .gitignore File

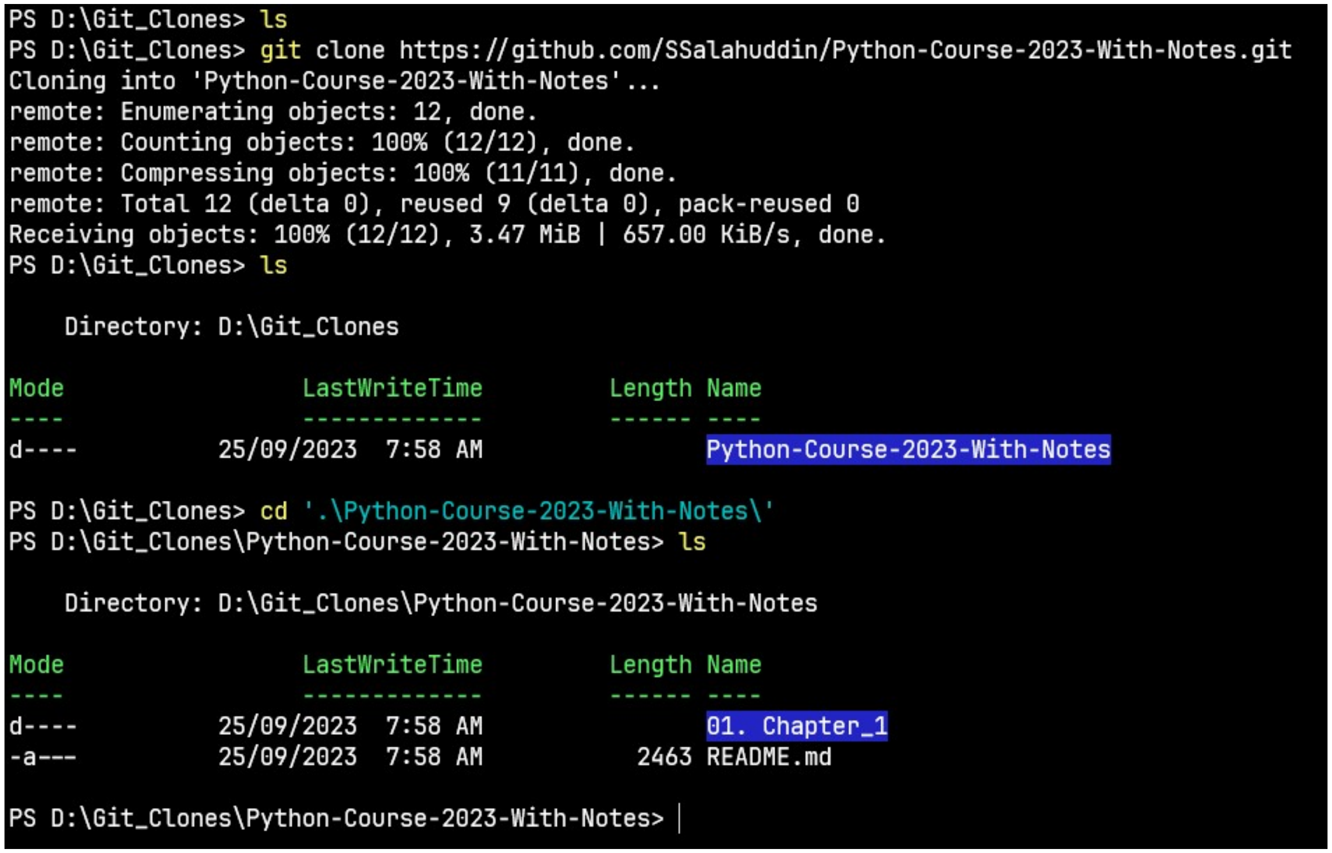
1. Ignore files/folders that you don’t want to commit.
2. It can be local and global as well.
3. Create .gitignore file inside a root directory of the repo.
4. .gitignore templates from GitHub: https://github.com/github/gitignore.
5. In Demo 2, we are not adding it.

Practical Demo 3 – Cloning Existing Repo

1. Clone the repository from GitHub using **git clone** command.
   - **Syntax**: **git clone** *<remote-repo-url> <local-repo-name>*
2. Lets clone the following repository:
   - **git clone https://github.com/SSalahuddin/Python-Course-2023-With-Notes.git**

     Or
   - **git clone https://github.com/SSalahuddin/Machine_Learning_Workshop2023.git**

Practical Demo 3 – Output



```
PS D:\Git_Clones> ls
PS D:\Git_Clones> git clone https://github.com/SSalahuddin/Python-Course-2023-With-Notes.git
Cloning into 'Python-Course-2023-With-Notes'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 12 (delta 0), reused 9 (delta 0), pack-reused 0
Receiving objects: 100% (12/12), 3.47 MiB | 657.00 KiB/s, done.
PS D:\Git_Clones> ls

    Directory: D:\Git_Clones

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          25/09/2023   7:58 AM                Python-Course-2023-With-Notes

PS D:\Git_Clones> cd '.\Python-Course-2023-With-Notes\'
PS D:\Git_Clones\Python-Course-2023-With-Notes> ls

    Directory: D:\Git_Clones\Python-Course-2023-With-Notes

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          25/09/2023   7:58 AM                01. Chapter_1
-a---          25/09/2023   7:58 AM           2463 README.md

PS D:\Git_Clones\Python-Course-2023-With-Notes> |
```

Summary

- **Git:**
  - Git is a distributed version control system that allows individuals and teams to efficiently track and manage changes in their code and documents. It records each modification, making it easy to collaborate, revert to previous versions, and maintain project history.

- **GitHub:**
  - GitHub is a web-based platform built around Git that enhances collaboration and provides a central hub for developers to host, share, and work on their Git repositories. It offers features like pull requests, issues, and project management tools.

References

[1] S. Chacon and B. Straub, *Pro Git*. Apress, 2023. Available at: https://git-scm.com/book/en/v2.

[2] D. Demaree, *Git for Humans*. 2016.

Machine Learning Workshop 2023

Session 3: Mastering Machine Learning: Techniques, Metrics, Classification Report, and Grid Search

By

Engr. Sumayyea Salahuddin, Engr. M. Arsalan Kamran, and Engr. Noor-ul-Haq

Overview

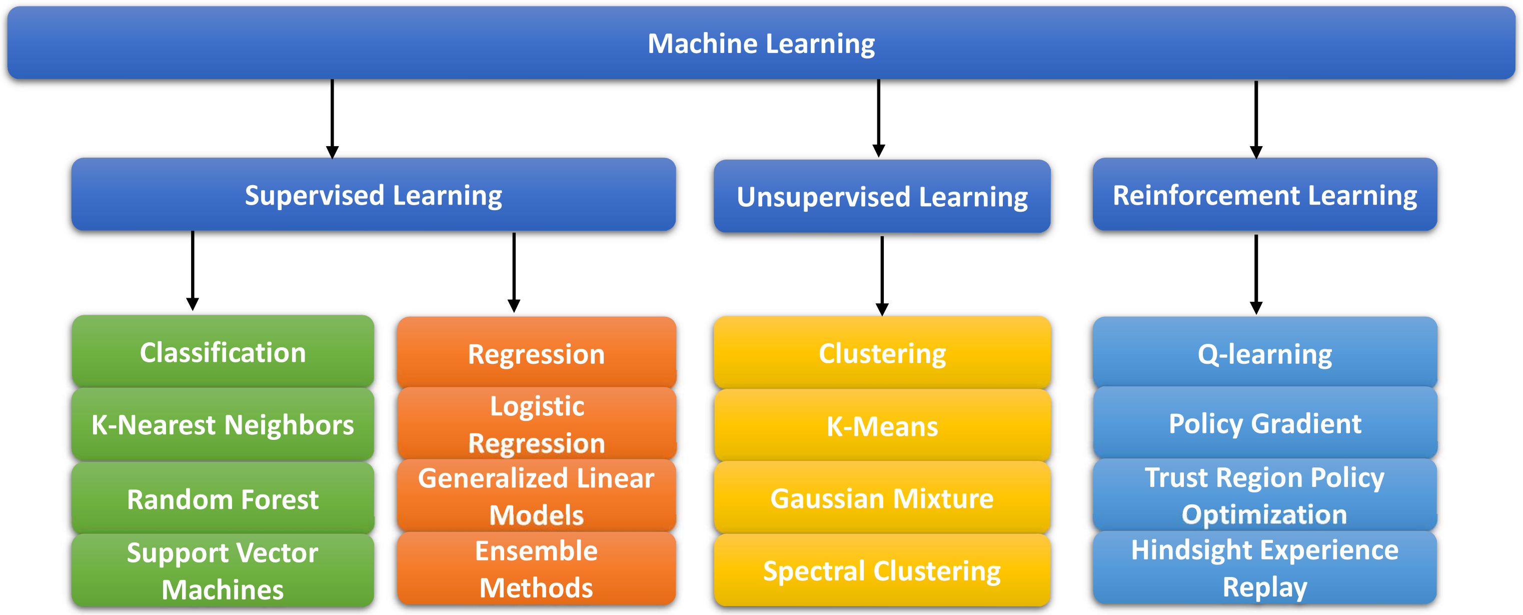
- Machine Learning Algorithms.
    1. Logistic Regression.
    2. Support Vector Machine.
    3. Random Forest.
- Metrics.
    1. Accuracy.
    2. Confusion Matrix.
        - True Positive (TP).
        - False Positive (FP).
        - True Negative (TN).
        - False Negative (FN).
- Classification Report.
    1. Precision.
    2. Recall.
    3. F1-Score.
- Grid Search.

Machine Learning Algorithms

Regressor vs. Classifier

- A Model may consist of a regressor (real value) or a classifier.
- A regressor outputs a continuous value.
  - Infinite set of values.
  - Maybe a probability (i.e., between 0 and 1).
  - Maybe an unbounded value (e.g., income).
- A classifier outputs a discrete value.
  - Finite set of values.
  - Maybe an enumeration (e.g., types of fruit).
  - Maybe a fixed set of numerical ranges (e.g., 10, 20, 30).

Algorithms


Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Classification
K-Nearest Neighbors
Random Forest
Support Vector Machines
Regression
Logistic Regression
Generalized Linear Models
Ensemble Methods
Clustering
K-Means
Gaussian Mixture
Spectral Clustering
Q-learning
Policy Gradient
Trust Region Policy Optimization
Hindsight Experience Replay

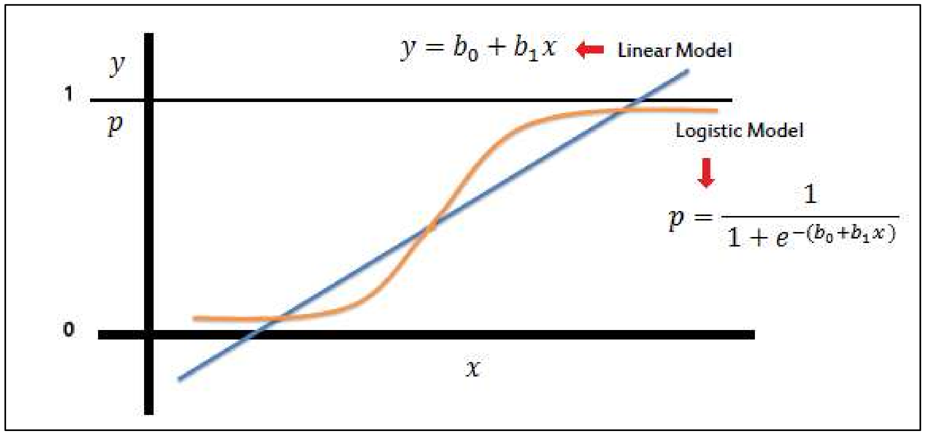
Logistic Regression

Overview

- Introduction.
- Types of classifications.
- How Logistic Regression Works?
    - Sigmoid Function.
    - Hypothesis.
    - Decision Boundary.
- Syntax of Logistic Regression.
    - Main parameters and their default values.
- Initializing Logistic Regression.
- References.

Introduction

- Logistic Regression is a “Supervised machine learning” algorithm that can be used to model the probability of a certain class or event.
- It takes input features and transforms them using a logistic function (also known as the sigmoid function), which maps the output to a range between 0 and 1
- This output can be interpreted as the probability of the given input belongs to the positive class.



Types of Classifications

- Binary classification
  - Tumor Malignant or Benign.
  - Male or Female.
  - Car or Bike.
  - Cat or Dog.
- Multi-Class Classification
  - Iris Flower set (setosa, versicolor or virginica).
  - Car, bike or airplane.
  - Cat, Dog or sheep.
  - Digits.

How Logistic Regression Works?

Sigmoid Function

- The logistic regression model uses a sigmoid function (also known as the logistic function) to map the linear combination of input features and their associated weights to a value between 0 and 1.
- The sigmoid function has an S-shaped curve that transforms any input into a probability value.

$$S(z) = \frac{1}{1 + e^{-z}}$$

Where (z) is the linear combination of the feature values (x)) and their corresponding weights (Θ) plus a bias term (b):

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

How Logistic Regression Works?

Hypothesis

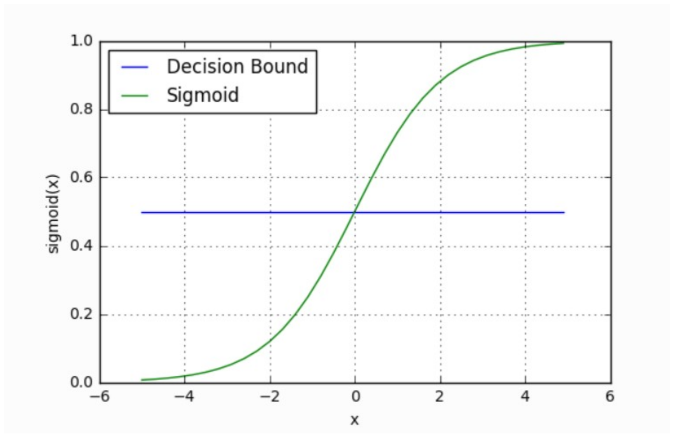
- The output of the sigmoid function is the hypothesis of the logistic regression model.
- It represents the estimated probability that an instance belongs to the positive class (class 1).
- The hypothesis is given by:

$$h_\theta(x) = S(z) = \frac{1}{1 + e^{-z}}$$

How Logistic Regression Works?

Decision Boundary

- To make a classification decision, you need to set a threshold (usually 0.5) on the predicted probability.
- If $h_\theta(x)$ is greater than or equal to the threshold, the instance is classified as the positive class; otherwise, it's classified as the negative class.



Syntax

#Initialization with default values

Model_name = LogisticRegression()

Default Values

Parameter Name	Default Value	Brief Explanation
penalty	**l2**	L2 regularization. It adds the sum of squared coefficients to the loss function to prevent overfitting.
dual	**False**	When the number of samples (n_samples) is larger than the number of features (n_features).
tol	**0.0001**	This sets the tolerance for stopping criteria.
C	**1.0**	The inverse of regularization strength.
fit_intercept	**True**	Boolean parameter that specifies whether or not to include an intercept term (bias) in the logistic regression model.
intercept_scaling	**1**	It scales the intercept term.
class_weight	**None**	It allows you to assign weights to classes to handle imbalanced datasets.
random_state	**None**	This is a seed for the random number generator.
Solver	**lbfgs**	This specifies the optimization algorithm to use.
max_iter	**100**	The maximum number of iterations for the solver to converge.
multi_class	**auto**	Specifies how the algorithm should handle multiclass classification problems.
Verbose	**0**	Controls the verbosity of the model's output during training.
warm_start	**False**	The model can be trained incrementally.
n_jobs	**None**	This parameter allows you to specify the number of CPU cores to use for parallelism during training.
l1_ratio	**None**	This parameter controls blend of L1 and L2 regularization but is active only when using 'elasticnet' penalty. If set to None, it defaults to L2 regularization.

Initializing Logistic Regression

```
from sklearn.linear_model import LogisticRegression
# Create a LogisticRegression model with default parameters
model = LogisticRegression()
# Fit the model to your training data
model.fit(X_train, y_train)
# Make predictions on new data
predictions = model.predict(X_test)
```

Summary

- **Logistic Regression** is a statistical method used for binary and multi-class classification problems. Despite its name, it is a classification algorithm, not a regression one.
- It models the relationship between a binary dependent variable (yes/no, 1/0) and one or more independent variables by estimating probabilities.
- Logistic Regression is simple yet powerful, providing interpretable results and serving as a baseline algorithm for many classification tasks.
- Its output is a probability score, and it uses the logistic function (sigmoid) to transform linear combinations of input features into values between 0 and 1.
- It is widely used in various fields, including healthcare (disease prediction), finance (credit scoring), and natural language processing (text classification).

References

[1] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression.* Hoboken, NJ: Wiley, 2013.

[2] “Sklearn.linear_model.logisticregression,” scikit, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (Accessed Sep. 25, 2023).

[3] “Machine learning - logistic regression,” Python Machine Learning - Logistic Regression, https://www.w3schools.com/python/python_ml_logistic_regression.asp (Accessed Sep. 25, 2023).

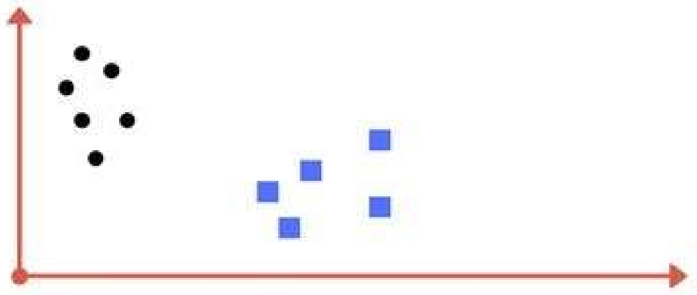
[4] “Scikit learn - logistic regression,” Online Courses and eBooks Library, https://www.tutorialspoint.com/scikit_learn/scikit_learn_logistic_regression.htm (Accessed Sep. 25, 2023).

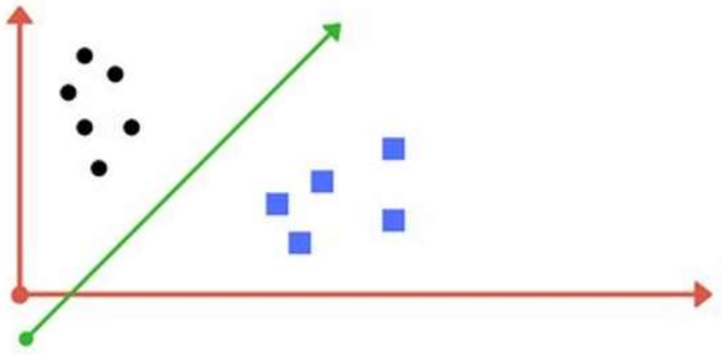
Support Vector Machine

Introduction

- A Support Vector Machine (SVM) is a discriminative classifier, which intakes the training data (supervised learning), the algorithm outputs an optimal hyperplane that categorizes new examples.

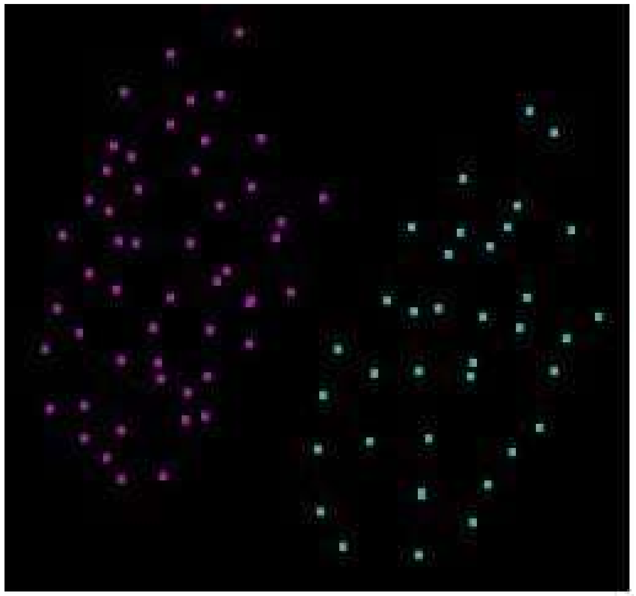
What could be drawn to classify the black dots from blue squares?

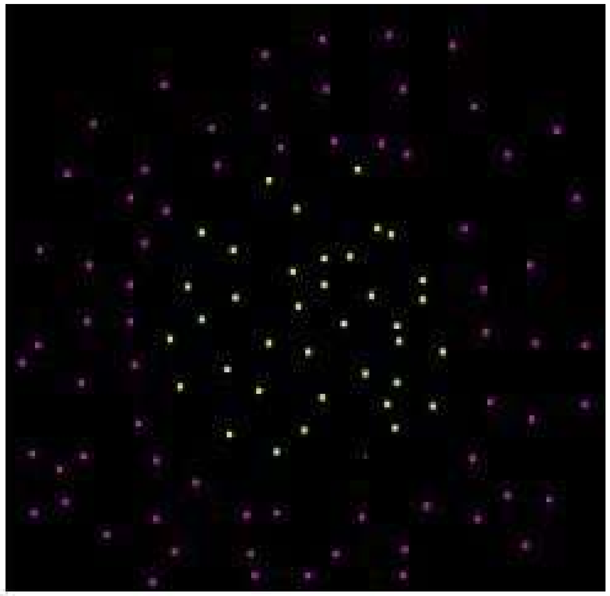




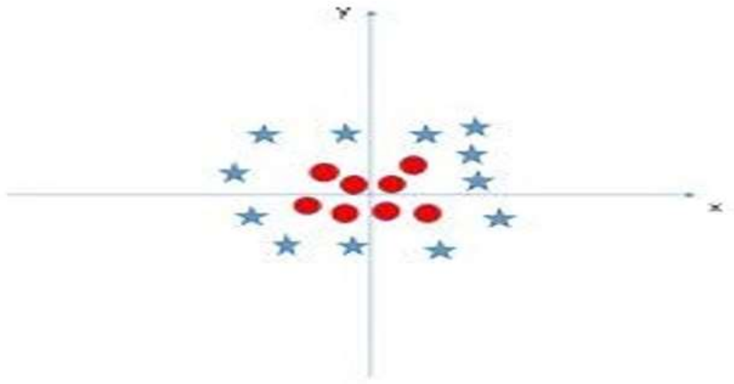
A line drawn between these data points classify the black dots and blue squares.

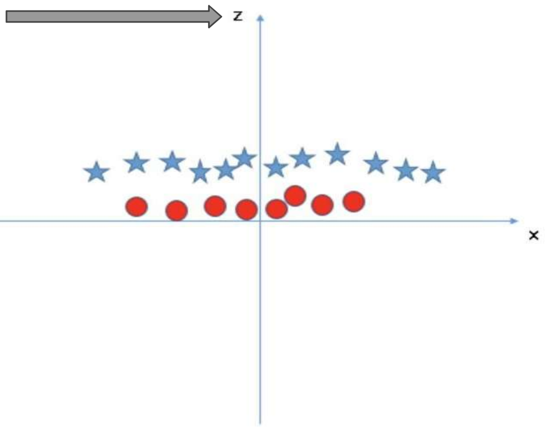
Linear vs. Nonlinear separable data


Linearly separable data


Non linearly separable data

Nonlinearly Separable Data

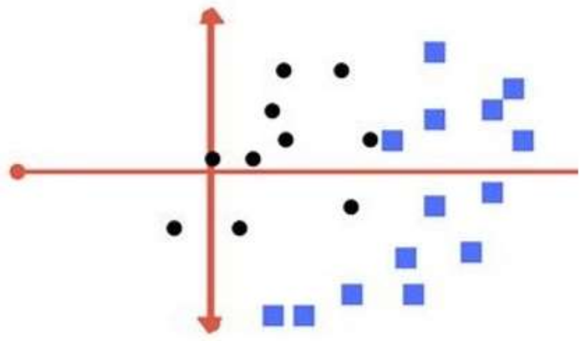


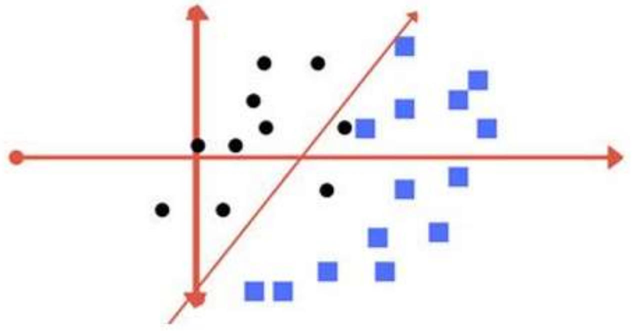


What could be drawn to classify these data points i.e. the red dots from blue stars?

Here, the hyperplane is a 2d plane drawn parallel to x-axis that is a separator.

Nonlinearly Separable Data (Cont.)

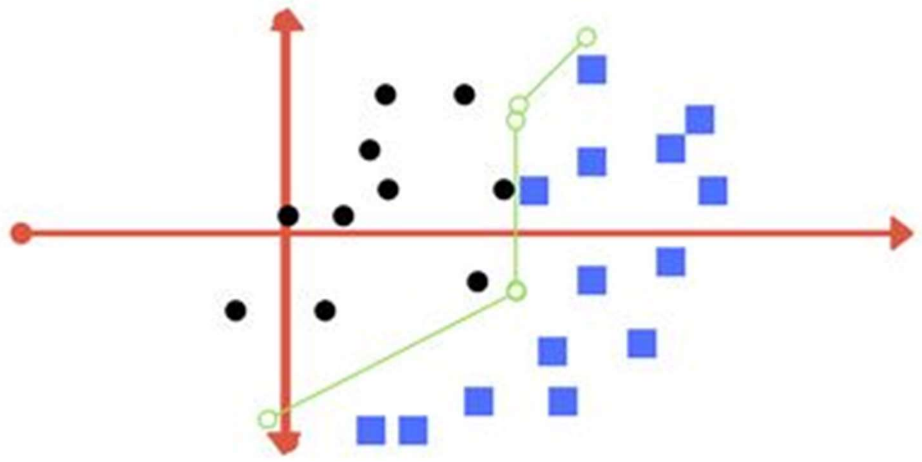

Raw Data


Line as Hyperplane

If the **line** is used as a **Hyperplane:**

- Two black dots also fall in category of blue squares
- Data separation is not perfect
- It tolerates some **outliers** in the classification

Nonlinearly Separable Data (Cont.)



This type of separator best provides the classification.

But

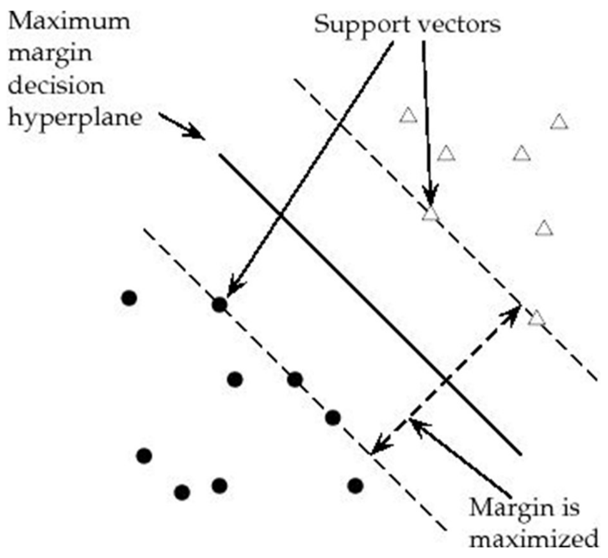
- It is quite difficult to train a model like this.
- This is termed as **Regularization parameter**.

Parameter Tuning

- Kernel
- Regularization
- Gamma
- Margin

Kernel

- Margin is the perpendicular distance between the closest data points and the Hyperplane (on both sides)
- The best optimized line (hyperplane) with maximum margins termed as Margin Maximal Hyperplane.
- The closest points where the margin distance is calculated are considered as the support vectors.



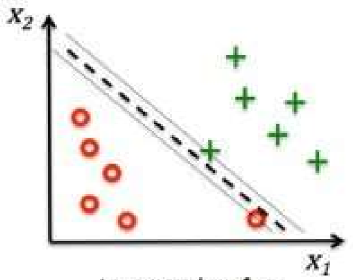
Regularization

- Also the ‘ C ‘ parameter in Python’s Sklearn Library
- Optimizes SVM classifier to avoid misclassifying the data.

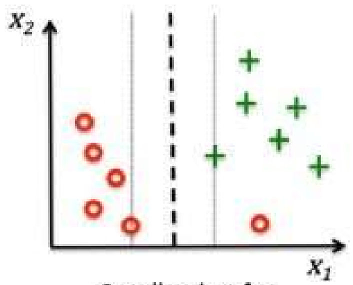
- C → large
- C → small

Margin of hyperplane → small
Margin of hyperplane → large
(misclassification possible)

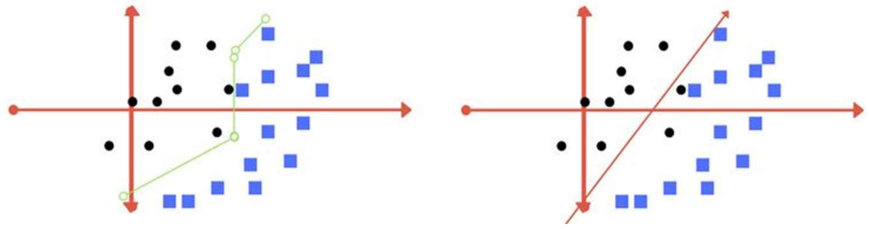
- C → large, chance of overfitting
- C → small, chance of underfitting



Large value for parameter *C*

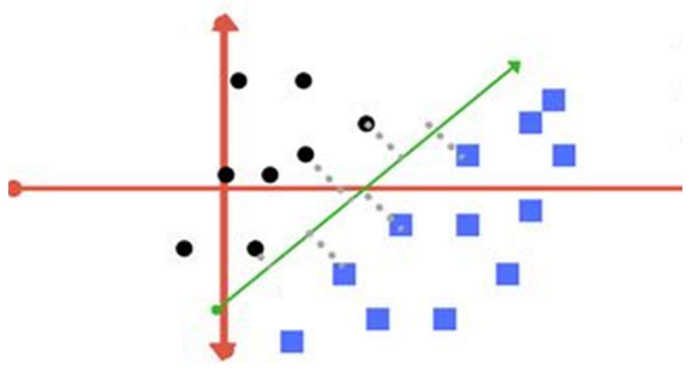


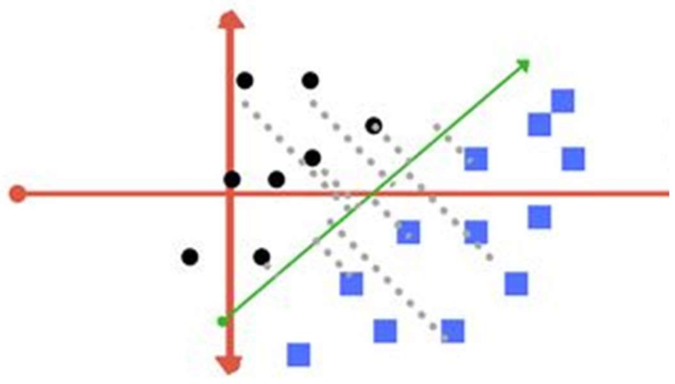
Small value for parameter *C*



Gamma

- Defines how far influences the calculation of plausible line of separation.
- Low gamma → points far from plausible line are considered for calculation
- High gamma → points close to plausible line are considered for calculation


High Gamma Value


Low Gamma Value

Kernels

- Mathematical functions for transforming data using linear algebra
- Different SVM algorithms use different types of kernel functions
- SVM Kernels:
  1. **Linear kernel**
  2. **Non - linear kernel**
  3. **Radial basis function (RBF)**
  4. **Sigmoid**
  5. **Polynomial**
  6. **Exponential**
- Example: $K(x, y) = \langle f(x), f(y) \rangle$

$$\underbrace{K(x, y)}_{\text{Kernel function}} = \underbrace{\langle f(x), f(y)\rangle}_{\text{dot product of n- dimensional inputs}}$$

Kernel Numerical Example

x = (x1, x2, x3); y = (y1, y2, y3)

f(x) = (x1x1, x1x2, x1x3, x2x1, x2x2, x2x3, x3x1, x3x2, x3x3)

f(y) = (y1y1, y1y2, y1y3, y2y1, y2y2, y2y3, y3y1, y3y2, y3y3)

$K(x, y) = (\langle x, y \rangle)^2$

x = (1, 2, 3)

y = (4, 5, 6)

f(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)

f(y) = (16, 20, 24, 20, 25, 30, 24, 30, 36)

$\langle f(x), f(y) \rangle = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$

$K(x, y) = (4 + 10 + 18)^2 = 1024 \rightarrow$ Kernel function

Syntax

#Initialization with default values

Model_name = LinearSVC()

Default Values

Parameter Name	Default Value	Brief Explanation
C	**1.0**	Regularization parameter. It controls the trade-off between maximizing the margin and minimizing the classification error.
loss	**squared_hinge**	It specifies the loss function to be used. 'squared_hinge' is typically used for linear SVMs.
penalty	**l2**	Type of regularization term. L2 is default.
dual	**True**	This determines whether to solve the primal or dual optimization problem. The default is dual.
tol	**0.0001**	This specifies the tolerance for stopping criterion.
multi_class	**ovr**	This determines the strategy for multiclass classification. The default is **ovr**, which stands for "one-vs-rest."
fit_intercept	**True**	This specifies whether to calculate the intercept for this model.
intercept_scaling	**1**	This specifies the scaling factor for the intercept.
class_weight	**None**	Weights associated with classes. Can be used to address class imbalance.
verbose	**0**	This controls the verbosity of the output.
random_state	**None**	This is used as a seed for the random number generator.
max_iter	**1000**	This controls the maximum number of iterations for optimization.

Initializing Support Vector Machine

```
from sklearn.linear_model import LinearSVC
# Create a LogisticRegression model with default parameters
model = LinearSVC()
# Fit the model to your training data
model.fit(X_train, y_train)
# Make predictions on new data
predictions = model.predict(X_test)
```

Pros & Cons

Pros

- It works really well with clear margin of separation
- It is effective in high dimensional spaces.
- It is effective in cases where number of dimensions is greater than the number of samples.
- It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Cons

- It doesn’t perform well, when we have large data set because the required training time is higher
- It also doesn’t perform very well, when the data set has more noise i.e. target classes are overlapping
- SVM doesn’t directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Applications

- Face detection
- Text and hypertext categorization
- Classification of images
- Bioinformatics
- Handwriting recognition
- Protein fold and remote homology detection
- Generalized predictive control(GPC)

Summary

- **Support Vector Machines (SVM)** are a powerful and versatile class of supervised machine learning algorithms used for both classification and regression tasks.
- SVM aims to find a hyperplane (decision boundary) that best separates data points into different classes while maximizing the margin between classes.
- SVM is particularly effective in high-dimensional spaces and can handle complex datasets with non-linear boundaries through techniques like the kernel trick.
- SVMs are known for their ability to handle outliers effectively and provide robust generalization to new, unseen data.
- While SVMs are powerful, they may require parameter tuning and can be computationally intensive for large datasets.
- They have applications in various domains, including image classification, text classification, bioinformatics, and finance.

References

[1] B. Schölkopf, A. J. Smola, and F. Bach, *Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2018.

[2] “Sklearn.svm.LinearSVC,” scikit, https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html (Accessed Sep. 25, 2023).

[3] “ML - Support Vector Machine(SVM),” Online Courses and eBooks Library, https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_support_vector_machine.htm (Accessed Sep. 25, 2023).

[4] “Scikit Learn - Support Vector Machines,” Online Courses and eBooks Library, https://www.tutorialspoint.com/scikit_learn/scikit_learn_support_vector_machines.htm (Accessed Sep. 25, 2023).

Random Forest

Overview

- Introduction.
    - Decision Trees vs. Random Forest
- How Random Forest Works?
- Syntax of Random Forest.
    - Main parameters and their default values.
- Initializing Random Forest.
- References.

Introduction

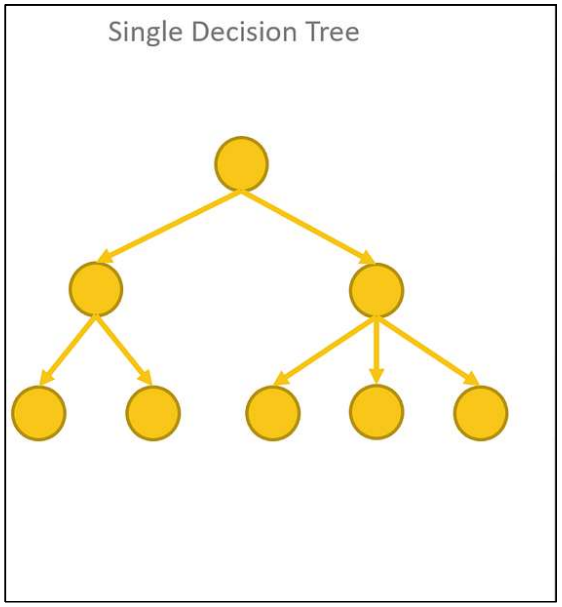
Decision Tree vs. Random Forest

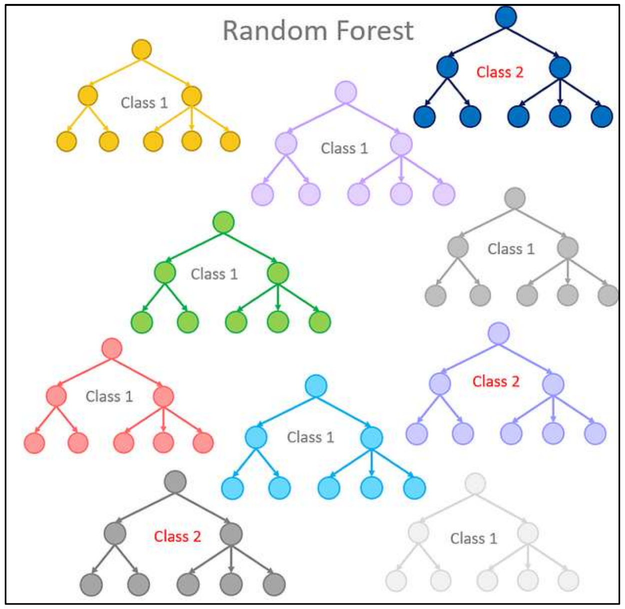
- Decision Tree is a versatile and intuitive machine learning algorithm
- They work by recursively splitting the dataset into subsets based on the most significant attributes, creating a tree-like structure.
- Each internal node represents a decision based on a specific feature, while each leaf node corresponds to a predicted outcome.
- Decision Trees are valued for their interpretability and ability to handle both numerical and categorical data.

- Random Forest is a popular machine learning algorithm that excels at both classification and regression tasks.
- It is made up of several decision trees to create a more accurate and robust final prediction.
- Random Forest combines the output of multiple decision trees and reach to a single final result.

Introduction (Cont.)

Decision Tree vs. Random Forest

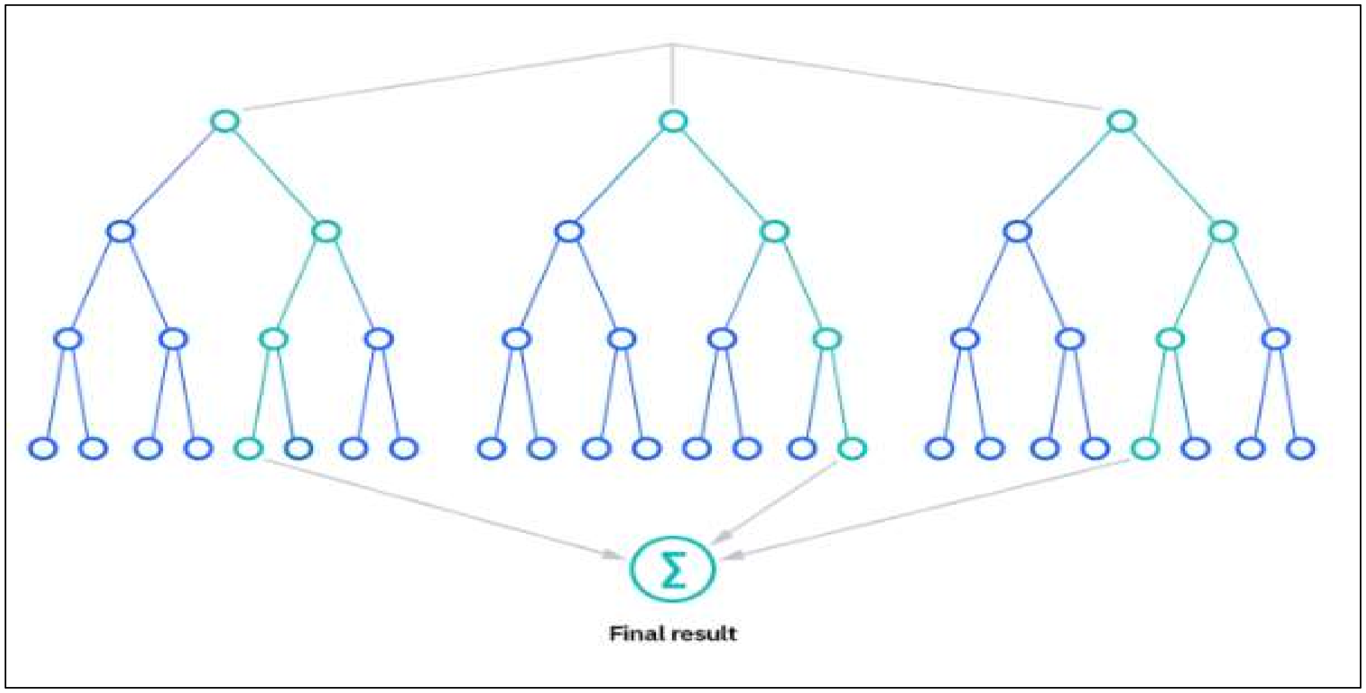




How Random Forest Works?

- Random forest algorithms have three main hyperparameters, which needed to be set before the training. These include **node size**, the **number of trees**, and the **number of features** sampled.
- The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample.
- Of that training sample, one-third of it is set aside as test data, known as the **out-of-bag** (oob) sample.
- Another instance of randomness is then injected through **feature bagging**, adding more diversity to the dataset and reducing the correlation among decision trees.
- For a regression task, the individual decision trees will be averaged, and for a classification task, a majority vote will yield the predicted class.
- Finally, the oob sample is then used for cross-validation, finalizing that prediction.

How Random Forest Works? (Cont.)



Syntax

#Initialization with default values

Model_name = RandomForestClassifier ()

Default Values

Parameter Name	Default Value	Brief Explanation
n_estimators	**100**	This specifies the number of decision trees to be used in the random forest.
criterion	**Gini**	This parameter specifies the function used to measure the quality of a split in each decision tree.
max_depth	**None**	It controls the maximum depth of each decision tree in the forest.
min_samples_split	**2**	This parameter defines the minimum number of samples required to split an internal node in a decision tree.
min_samples_leaf	**1**	It sets the minimum number of samples required to be in a leaf node.
min_weight_fraction_leaf	**0.0**	It sets a minimum weighted fraction of the total number of samples required to be in a leaf node.
max_features	**sqrt**	It determines the number of features to consider when looking for the best split in each tree.
max_leaf_nodes	**None**	his parameter restricts the maximum number of leaf nodes in a tree.
min_impurity_decrease	**0.0**	It sets a threshold for a node to split based on a decrease in impurity. If the impurity decrease is less than this value, the split will not be performed.
bootstrap	**True**	This Boolean parameter specifies whether or not to use bootstrapping when building the decision trees.
oob_score	**False**	OOB samples are data points that were not used in the construction of a particular decision tree and can be used for estimating the model's accuracy.
n_jobs	**None**	It specifies the number of CPU cores to use for parallelism during tree construction.

Syntax (Cont.)

#Initialization with default values

Model_name = RandomForestClassifier ()

Default Values

Parameter Name	Default Value	Brief Explanation
random_state	**None**	This is a seed for the random number generator.
verbose	**0**	Controls the verbosity of the model's output during training. It's set to 0, meaning no output during training.
warm_start	**False**	The controls if the model can be trained incrementally or not.
class_weight	**None**	This parameter allows you to assign weights to classes to balance the impact of imbalanced datasets
ccp_alpha	**0.0**	It's used for cost-complexity pruning of the decision trees within the random forest.
max_samples	**None**	The maximum number of samples to use when building each tree.

Initializing Random Forest

```python
from sklearn.ensemble import RandomForestClassifier

# Create a Random Forest Classifier with default parameters

rf_classifier = RandomForestClassifier()

# Fit the model on training data

rf_classifier.fit(X_train, y_train)

# Make predictions

predictions = rf_classifier.predict(X_test)
```

Summary

- Random Forest is a versatile and robust ensemble learning method used in supervised machine learning for classification and regression tasks.
- It's based on the idea of building multiple decision trees during training and combining their predictions to make more accurate and stable predictions.
- Random Forest mitigates overfitting, a common issue in decision trees, by introducing randomness in the tree-building process. This randomness includes bootstrapping (randomly selecting subsets of the data) and feature selection.
- It's highly effective for handling high-dimensional data, dealing with both categorical and continuous features, and capturing complex relationships in the data.
- Random Forest provides feature importance scores, helping in feature selection and understanding the most influential variables.
- It's known for its robustness to noisy data, missing values, and outliers, making it a valuable tool in real-world applications.
- Random Forest has broad applications in various domains, including healthcare, finance, and image analysis.

References

[1] R. Genuer and J.-M. Poggi, *Random Forests with R*. Cham, Switzerland: Springer, 2020.

[2] “Sklearn.ensemble.randomforestclassifier,” scikit, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (Accessed Sep. 25, 2023).

[3] A. Shafi, “Random Forest Classification with Scikit-Learn,” DataCamp, https://www.datacamp.com/tutorial/random-forests-classifier-python (Accessed Sep. 25, 2023).

[4] “Random Forest Classifier using Scikit-learn,” GeeksforGeeks, https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/ (Accessed Sep. 25, 2023).

Metrics

Accuracy Metrics in Regression Model

- After a regression model has been trained, a test data set is run against the model to determine accuracy.
  - The test data set has labels indicating the expected result (y).
  - For instance, expected spending level.
  - The model outputs predictions (ŷ).
  - For instance, Amount of spending.
- Accuracy is measured as a cost (or loss) function between the expected result and predicted result.
  - For instance, Mean Square Error (MSE)

Accuracy Metrics in Classification Model

- After a classification model has been trained (e.g., apple vs. pear), a test data set is run against the model to determine accuracy.
  - The test data set has labels indicating the expected result (y).
  - For instance, an apple or pear.
  - The model outputs predictions (ŷ).
  - For instance, Is it an apple or a pear?
- Accuracy is measured as the percentage of predicted results that match the expected results.
  - For instance, if there are 1000 results, and 850 predicted results match the expected results, then the accuracy is 85%.

Problem with Accuracy Metrics

- If the training and test data are skewed towards one classification, then the model will predict everything as being that class.
    - For instance, in Titanic training data, 68% of people died. If one trained a model to predict everybody died, it would be 68% accurate.
- The data may be fitted against a feature that is not relevant.
    - For instance, in image classification, if all images of one class have small/similar background, the model may match based on the background, not the object in the image.

Confusion Matrix

- A common method for describing the performance of a classification model consisting of **true positives**, **true negatives**, **false positives**, and **false negatives**.
- It is called a confusion matrix because it shows how confused the model is between the classes.

Confusion Metrix Example

- Confusion matrix consists of:
  1. **True Positive (TP):** This is the number of data points that were actually positive and were also predicted as positive.
  2. **False Positive (FP):** This is the number of data points that were actually negative but were predicted as positive.
  3. **True Negative (TN):** This is the number of data points that were actually negative and were also predicted as negative.
  4. **False Negative (FN):** This is the number of data points that were actually positive but were predicted as negative.
- Let us understand it using **Iris Dataset** example where three flowers i.e. **Setosa**, **Versicolor**, and **Virginica** are classified.

IRIS DATASET

True Positive (All Classes)

Predicted Values

Actual Values

	Setosa	Versicolor	Virginica
Setosa	**23 (Cell 1)**	0 (Cell 2)	0 (Cell 3)
Versicolor	0 (Cell 4)	**15 (Cell 5)**	4 (Cell 6)
Virginica	0 (Cell 7)	1 (Cell 8)	**17 (Cell 9)**

The model correctly classified 23 Setosa, 15 Versicolor, and 17 Virginica flowers.

IRIS DATASET

True Negative for Setosa

		Predicted Values		
		Setosa	**Versicolor**	**Virginica**
Actual Values	**Setosa**	23 (Cell 1)	0 (Cell 2)	0 (Cell 3)
	Versicolor	0 (Cell 4)	**15 (Cell 5)**	**4 (Cell 6)**
	Virginica	0 (Cell 7)	**1 (Cell 8)**	**17 (Cell 9)**

The model correctly classified 37 non-Setosa flowers.

IRIS DATASET

True Negative for Versicolor

		Predicted Values		
		Setosa	Versicolor	Virginica
Actual Values	Setosa	**23 (Cell 1)**	0 (Cell 2)	**0 (Cell 3)**
	Versicolor	0 (Cell 4)	15 (Cell 5)	4 (Cell 6)
	Virginica	**0 (Cell 7)**	1 (Cell 8)	**17 (Cell 9)**

The model correctly classified 40 non-Versicolor flowers.

IRIS DATASET

True Negative for Virginica

		Predicted Values		
		Setosa	Versicolor	Virginica
Actual Values	Setosa	**23 (Cell 1)**	**0 (Cell 2)**	0 (Cell 3)
	Versicolor	**0 (Cell 4)**	**15 (Cell 5)**	4 (Cell 6)
	Virginica	0 (Cell 7)	1 (Cell 8)	17 (Cell 9)

The model correctly classified 38 non-Virginica flowers.

IRIS DATASET

False Positive for Setosa

		Predicted Values		
		Setosa	**Versicolor**	**Virginica**
Actual Values	**Setosa**	23 (Cell 1)	0 (Cell 2)	0 (Cell 3)
	Versicolor	**0 (Cell 4)**	15 (Cell 5)	4 (Cell 6)
	Virginica	**0 (Cell 7)**	1 (Cell 8)	17 (Cell 9)

The model incorrectly classified 0 cases as Setosa flowers.

IRIS DATASET

False Positive for Versicolor

		Predicted Values		
		Setosa	**Versicolor**	**Virginica**
Actual Values	**Setosa**	23 (Cell 1)	**0 (Cell 2)**	0 (Cell 3)
	Versicolor	0 (Cell 4)	15 (Cell 5)	4 (Cell 6)
	Virginica	0 (Cell 7)	**1 (Cell 8)**	17 (Cell 9)

The model incorrectly classified 1 case as a Versicolor flower.

IRIS DATASET

False Positive for Virginica

Actual Values		Predicted Values		
		Setosa	Versicolor	Virginica
	Setosa	23 (Cell 1)	0 (Cell 2)	**0 (Cell 3)**
	Versicolor	0 (Cell 4)	15 (Cell 5)	**4 (Cell 6)**
	Virginica	0 (Cell 7)	1 (Cell 8)	17 (Cell 9)

The model incorrectly classified 4 cases as a Virginica flowers.

IRIS DATASET

False Negative for Setosa

Predicted Values

Actual Values		Setosa	Versicolor	Virginica
	Setosa	23 (Cell 1)	**0 (Cell 2)**	**0 (Cell 3)**
	Versicolor	0 (Cell 4)	15 (Cell 5)	4 (Cell 6)
	Virginica	0 (Cell 7)	1 (Cell 8)	17 (Cell 9)

The model incorrectly classified 0 cases as not belonging to Setosa flowers.

IRIS DATASET

False Negative for Versicolor

		Predicted Values		
		Setosa	Versicolor	Virginica
Actual Values	Setosa	23 (Cell 1)	0 (Cell 2)	0 (Cell 3)
	Versicolor	**0 (Cell 4)**	15 (Cell 5)	**4 (Cell 6)**
	Virginica	0 (Cell 7)	1 (Cell 8)	17 (Cell 9)

The model incorrectly classified 4 cases as not belonging to Versicolor flowers.

False Negative for Virginica

		Predicted Values		
		Setosa	Versicolor	Virginica
Actual Values	Setosa	23 (Cell 1)	0 (Cell 2)	0 (Cell 3)
	Versicolor	0 (Cell 4)	15 (Cell 5)	4 (Cell 6)
	Virginica	**0 (Cell 7)**	**1 (Cell 8)**	17 (Cell 9)

The model incorrectly classified 1 case as not belonging to Virginica flowers.

Confusion Matrix Summary

- Confusion matrix is a classifier evaluation tool.
- Classes need not be opposites.
- One class is labelled positive, others are labelled negative. Just labelled, does not mean anything.
- Accuracy can also be expressed in terms of true positives (**TP**), true negatives (**TN**), false positives (**FP**), and false negatives (**FN**). The formula is: **Accuracy = (TP + TN) / (TP + TN + FP + FN)**.
- The accuracy of three flower classes are:

Setosa Accuracy = (23+37)/(23+0+0+37) = 60 / 60 = **100%**

Versicolor Accuracy = (15+40)/(15+4+1+40) = 55 / 60 = **91.67%**

Virginica Accuracy = (17+38)/(17+1+4+38) = 55 / 60= **91.67%**

References

[1] A. Mishra, “Metrics to Evaluate your Machine Learning Algorithm,” Medium, https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234 (Accessed Sep. 25, 2023).

[2] “Confusion Matrix in Machine Learning,” GeeksforGeeks, https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ (Accessed Sep. 25, 2023).

[3] S. Narkhede, “Understanding Confusion Matrix,” Medium, https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 (Accessed Sep. 25, 2023).

[4] J. Brownlee, “What is a Confusion Matrix in Machine Learning,” MachineLearningMastery.com, https://machinelearningmastery.com/confusion-matrix-machine-learning/ (Accessed Sep. 25, 2023).

Classification Report

Overview

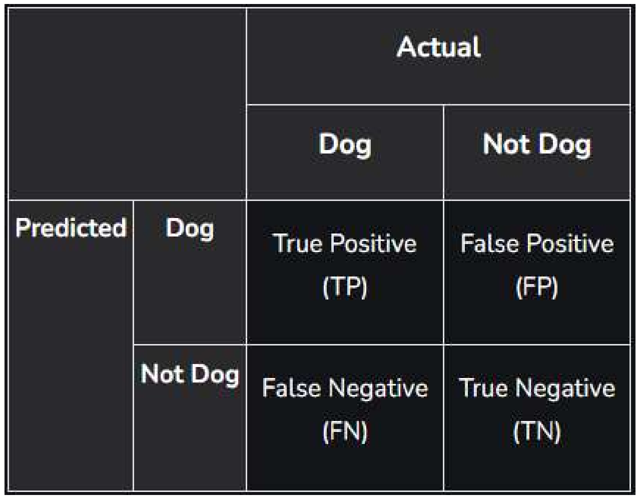
- Introduction.
- Cases.
- Metrics.
  - Precision.
  - Recall.
  - F1-Score.
  - Support.
- How to use Classification Report.
- References.

Introduction

- A Classification Report is a concise summary used to evaluate the performance of a classification model.
- It provides crucial metrics such as precision, recall, F1-score, and support for each class in the dataset.
- This report aids in assessing the model's effectiveness in correctly classifying instances, highlighting strengths and weaknesses across different classes and helping to make informed decisions about model improvements.

Cases

- **TN / True Negative**: the case was negative and predicted negative
- **TP / True Positive**: the case was positive and predicted positive
- **FN / False Negative**: the case was positive but predicted negative
- **FP / False Positive**: the case was negative but predicted positive
- **Example**:



		Actual	
		Dog	Not Dog
Predicted	Dog	True Positive (TP)	False Positive (FP)
	Not Dog	False Negative (FN)	True Negative (TN)

Metric

Precision

- Precision is the ability of a classifier not to label an instance positive that is actually negative.
- For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive.

Precision: Accuracy of positive predictions.

Precision = TP/(TP + FP)

Metric

Recall

- Recall is the ability of a classifier to find all positive instances.
- For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.

Recall: Fraction of positives that were correctly identified.

Recall = TP/(TP+FN)

Metric

F1-Score

- The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0.
- F1 scores are lower than accuracy measures as they embed precision and recall into their computation.
- As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

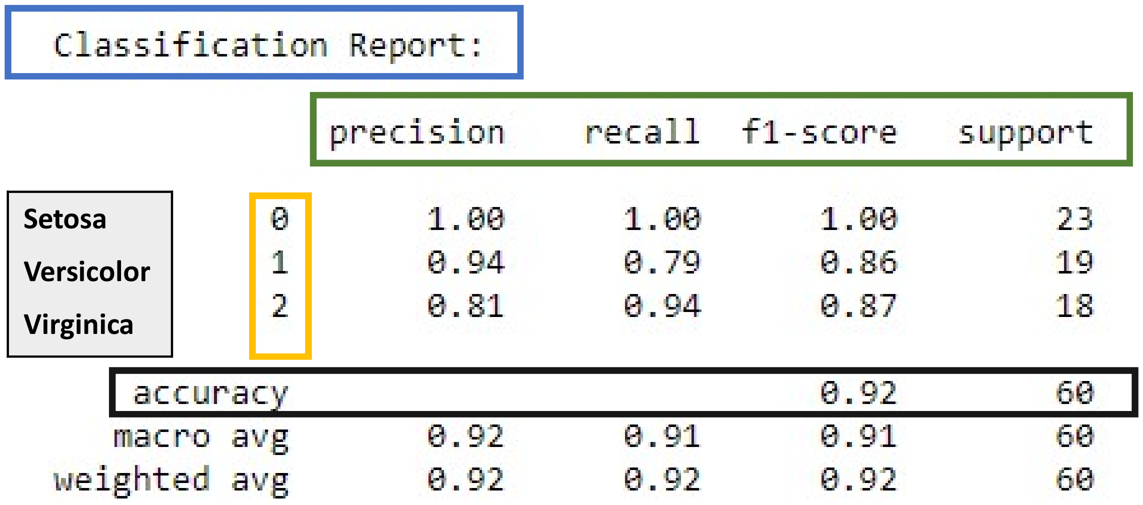
F1 Score = 2*(Recall * Precision) / (Recall + Precision).

Metric

Support

- Support is the number of actual occurrences of the class in the specified dataset.
- Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing.
- Support doesn’t change between models but instead diagnoses the evaluation process.

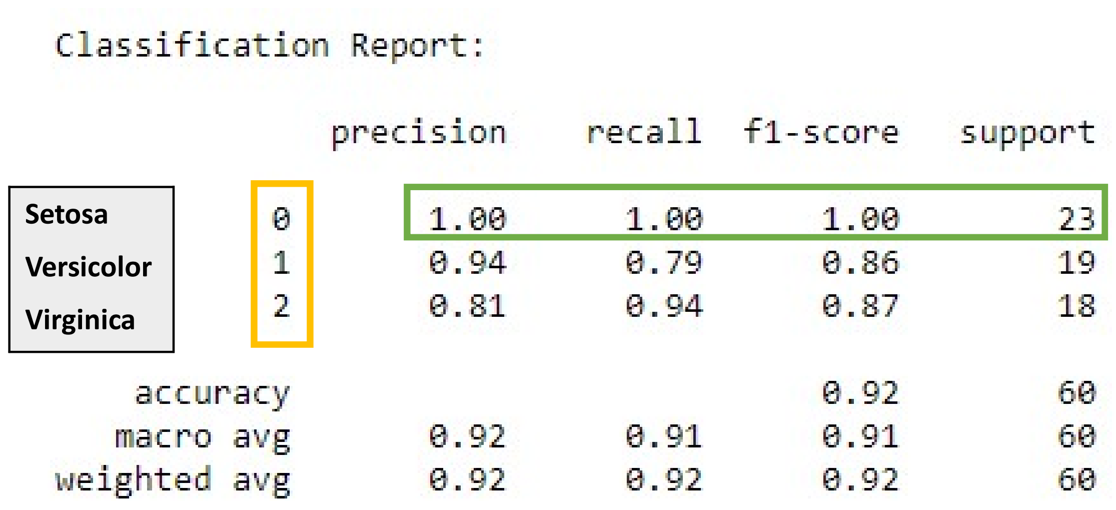
Example: Classification Report of Iris Dataset



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       0.94      0.79      0.86        19
           2       0.81      0.94      0.87        18

    accuracy                           0.92        60
   macro avg       0.92      0.91      0.91        60
weighted avg       0.92      0.92      0.92        60
```

- A classification report consisting of precision, recall, f1-score, support for all three Iris Dataset classes are shown here. Also, the overall accuracy of Iris dataset i.e. 92% is shown as well.

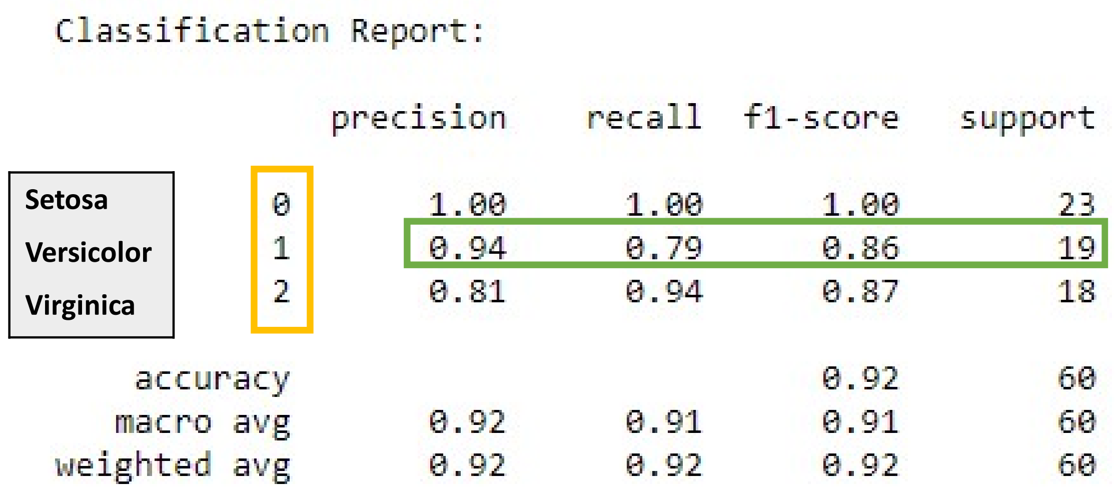
Example: Classification Report of Setosa



```
Classification Report:

             precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       0.94      0.79      0.86        19
           2       0.81      0.94      0.87        18

    accuracy                           0.92        60
   macro avg       0.92      0.91      0.91        60
weighted avg       0.92      0.92      0.92        60
```

- For Setosa class, the precision, recall, and f1-score of is 100% (1.00) while the support is 23.
- **Precision** = TP / (TP + FP) = 23 / (23+0) = 23 / 23 = **100% (1.00)**
- **Recall** = TP / (TP + FN) = 23 / (23 + 0) = 23 / 23 = **100% (1.00)**
- **F1-score** = 2 * (precision * recall) / (precision + recall) = 2 * (1*1) / (1+1) = 2/2 = **100% (1.00)**

Example: Classification Report of Versicolor

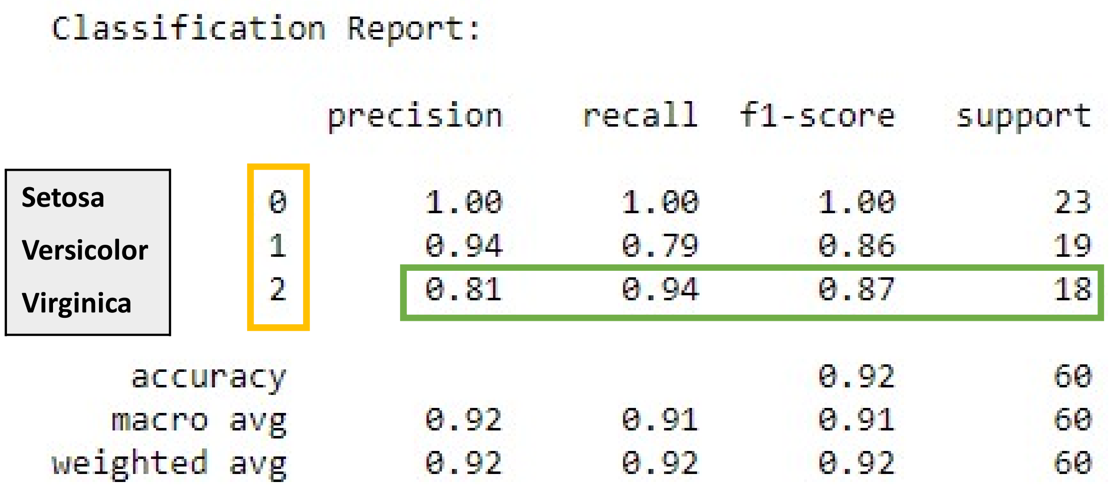


```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       0.94      0.79      0.86        19
           2       0.81      0.94      0.87        18

    accuracy                           0.92        60
   macro avg       0.92      0.91      0.91        60
weighted avg       0.92      0.92      0.92        60
```

- For Versicolor class, precision is 94%, recall is 79%, and f1-score of is 86% while support is 19.
- **Precision** = TP / (TP + FP) = 15 / (15+1) = 15 / 16 = **94% (0.94)**
- **Recall** = TP / (TP + FN) = 15 / (15 + 4) = 15 / 19 = **79% (0.79)**
- **F1-score** = 2 * (precision * recall) / (precision + recall) = 2 * (0.94*0.79) / (0.94+0.79) = **86% (0.86)**

IRIS DATASET

Example: Classification Report of Virginica



```
Classification Report:

                 precision    recall  f1-score   support

Setosa       0       1.00      1.00      1.00        23
Versicolor   1       0.94      0.79      0.86        19
Virginica    2       0.81      0.94      0.87        18

      accuracy                           0.92        60
     macro avg       0.92      0.91      0.91        60
  weighted avg       0.92      0.92      0.92        60
```

- For Virginica class, precision is 81%, recall is 94%, and f1-score of is 87% while support is 18.
- **Precision** = TP / (TP + FP) = 17 / (17+4) = 17 / 21 = **81% (0.81)**
- **Recall** = TP / (TP + FN) = 17 / (17 + 1) = 17 / 18 = **94% (0.94)**
- **F1-score** = 2 * (precision * recall) / (precision + recall) = 2 * (0.81*0.94) / (0.81+0.94) = **87% (0.87)**

References

[1] T. Kanstrén, “A Look at Precision, Recall, and F1-Score,” Medium, https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec#:~:text=F1%2DScore%20is%20a%20measure,than%20the%20traditional%20arithmetic%20mean. (Accessed Sep. 25, 2023).

[2] K. P. Shung, “Accuracy, Precision, Recall or F1?,” Medium, https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. (Accessed Sep. 25, 2023).

Grid Search

Overview

- Introduction.
    - What are Hyperparameters?
    - What is Grid Search?
- How to implement Grid Search?
- Practical Example in Python.
- Result Comparison Before and After Grid Search.
- How to choose the right hyperparameters to search over?
- Conclusion.
- References.

Introduction

Hyperparameters

- Hyperparameters are parameters that control the training process of a machine learning model.
- For example, the hyperparameters of SVM are given on Slide # 69. These are *C*, *loss*, *penalty*, *dual*, *tol*, *multi_class*, *fit_intercept*, *intercept_scaling*, *class_weight*, *verbose*, *random_state*, and *max_iter*.
- The are not learning from the data, but rather set before training begins.
- Choosing the right hyperparameters can have a significant impact on the performance of a machine learning model.

Introduction

Grid Search

- Grid Search is a hyperparameter optimization technique that exhaustively searches through a predefined grid of hyperparameter values.
- It trains a model for each combination of hyperparameter values and selects the combination that produces the best performance on a held-out validation set.

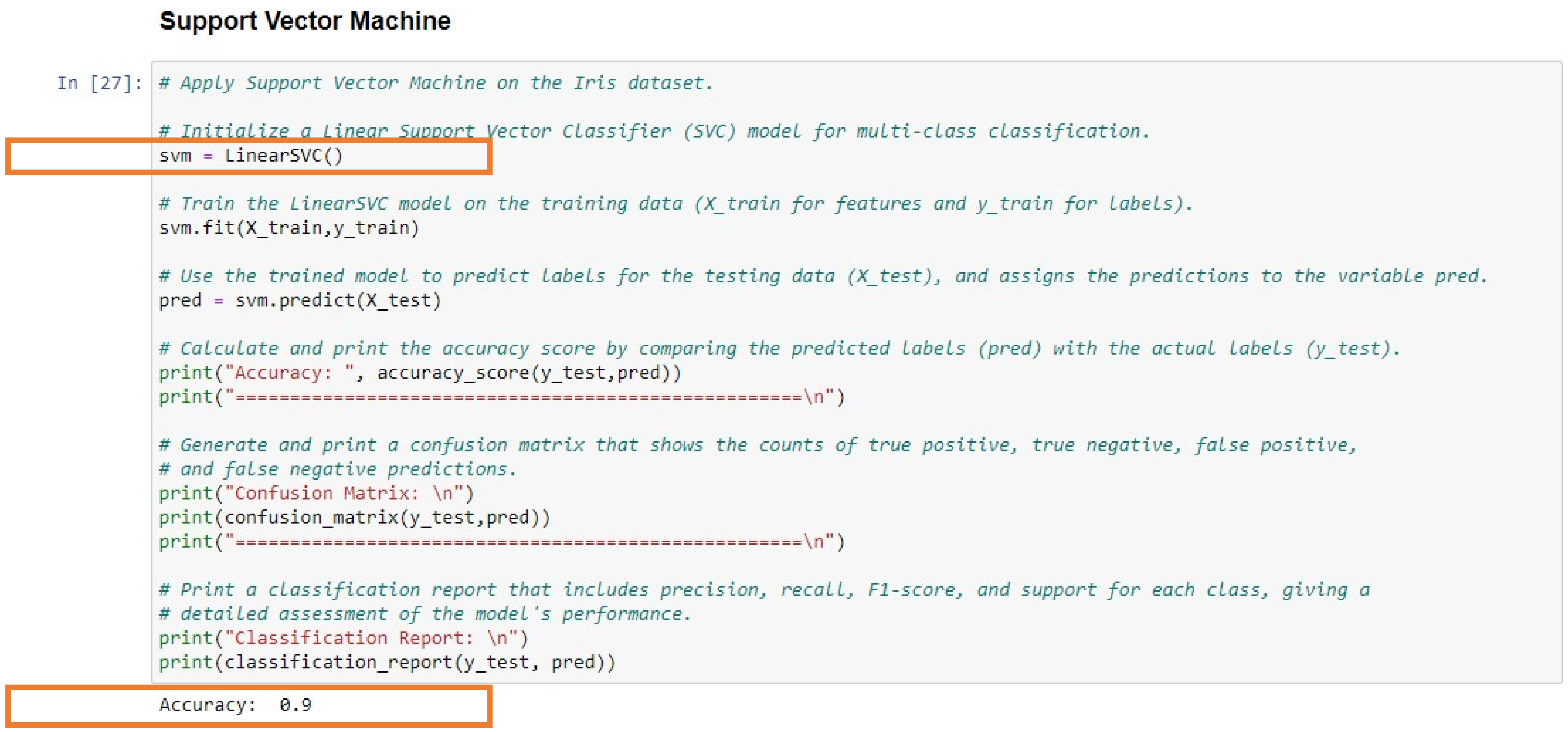
How to Implement Grid Search?

- To implement grid search, follow these steps:
    1. Define the grid of hyperparameter values to evaluate.
    2. Create a machine learning model and train it on each combination of hyperparameter values in the grid.
    3. Evaluate the performance of trained models.
    4. Select the model with the best performance.

Example of Grid Search in Python

```
# Perform Grid Search on Support Vector Machine to optimize its performance.
from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVC
# Define the grid of hyperparameter values
param_grid = {
    'C': [0.1, 1, 10, 100],
    'max_iter': [100000, 1000000],
}
# Create a GridSearchCV object
grid_search = GridSearchCV(LinearSVC(), param_grid=param_grid)
# Train the model on the training data
grid_search.fit(X_train, y_train)
# Print the best estimator found by the GridSearchCV object.
print(grid_search.best_estimator_)
```

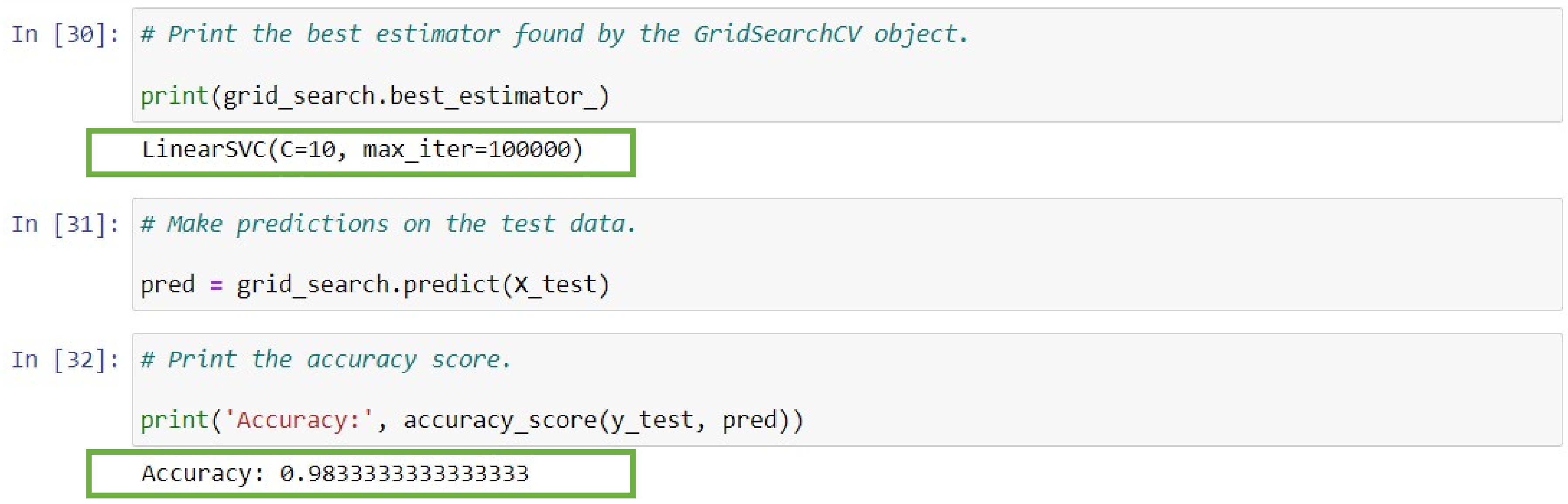
Result Comparison: Before Grid Search



Support Vector Machine

```
In [27]: # Apply Support Vector Machine on the Iris dataset.

         # Initialize a Linear Support Vector Classifier (SVC) model for multi-class classification.
         svm = LinearSVC()

         # Train the LinearSVC model on the training data (X_train for features and y_train for labels).
         svm.fit(X_train,y_train)

         # Use the trained model to predict labels for the testing data (X_test), and assigns the predictions to the variable pred.
         pred = svm.predict(X_test)

         # Calculate and print the accuracy score by comparing the predicted labels (pred) with the actual labels (y_test).
         print("Accuracy: ", accuracy_score(y_test,pred))
         print("====================================================\n")

         # Generate and print a confusion matrix that shows the counts of true positive, true negative, false positive,
         # and false negative predictions.
         print("Confusion Matrix: \n")
         print(confusion_matrix(y_test,pred))
         print("====================================================\n")

         # Print a classification report that includes precision, recall, F1-score, and support for each class, giving a
         # detailed assessment of the model's performance.
         print("Classification Report: \n")
         print(classification_report(y_test, pred))

         Accuracy:  0.9
```

Result Comparison: After Grid Search



```
In [30]: # Print the best estimator found by the GridSearchCV object.

         print(grid_search.best_estimator_)

         LinearSVC(C=10, max_iter=100000)

In [31]: # Make predictions on the test data.

         pred = grid_search.predict(X_test)

In [32]: # Print the accuracy score.

         print('Accuracy:', accuracy_score(y_test, pred))

         Accuracy: 0.9833333333333333
```

How to choose the right hyperparameters to search over?

- When choosing the right parameters to search over, it is important to consider the following factors:
  1. **The type of machine learning model**: Different machine learning models have different hyperparameters, and some hyperparameters are more important than others for a given model.
  2. **The complexity of the dataset**: More complex datasets may require more tuning of hyperparameters.
  3. **The computational resources available**: Grid search can be computationally expensive, so it is important to choose a grid of hyperparameter values that is feasible to search over.

How to choose the right hyperparameters to search over? (Cont.)

Solution

- These are some tips for choosing the right hyperparameters to search over:
  1. Start by searching over a small grid of values for the most important hyperparameters.
  2. Once you have found a good set of hyperparameters, you can refine the search by searching over a smaller grid of values around the best hyperparameters.
  3. You can also use a technique called random search to explore a wider range of hyperparameter values.

Summary

- Grid search is a powerful tool for finding optimal hyperparameters for machine learning models. It is a simple and straightforward technique to implement, and it can be used with a variety of different machine learning models. However, grid search can be computationally expensive, especially for large datasets and complex models.

References

[1] “Machine Learning - Grid Search,” Python Machine Learning - Grid Search, https://www.w3schools.com/python/python_ml_grid_search.asp (Accessed Sep. 25, 2023).

[2] J. Brownlee, “Hyperparameter Optimization With Random Search and Grid Search,” MachineLearningMastery.com, https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/ (Accessed Sep. 25, 2023).

Thank you for attending. ☺