Artificial Intelligence and Machine Learning Internship

INTERNSHIP REPORT

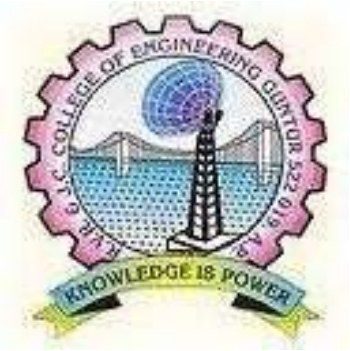
Submitted in the partial fulfillment of requirements to

Summer Internship (CS- 353)

III/IV B.Tech (V Semester)

Submitted By

DIAREDDY PHANEDRA REDDY (Y22CS040)



October, 2024

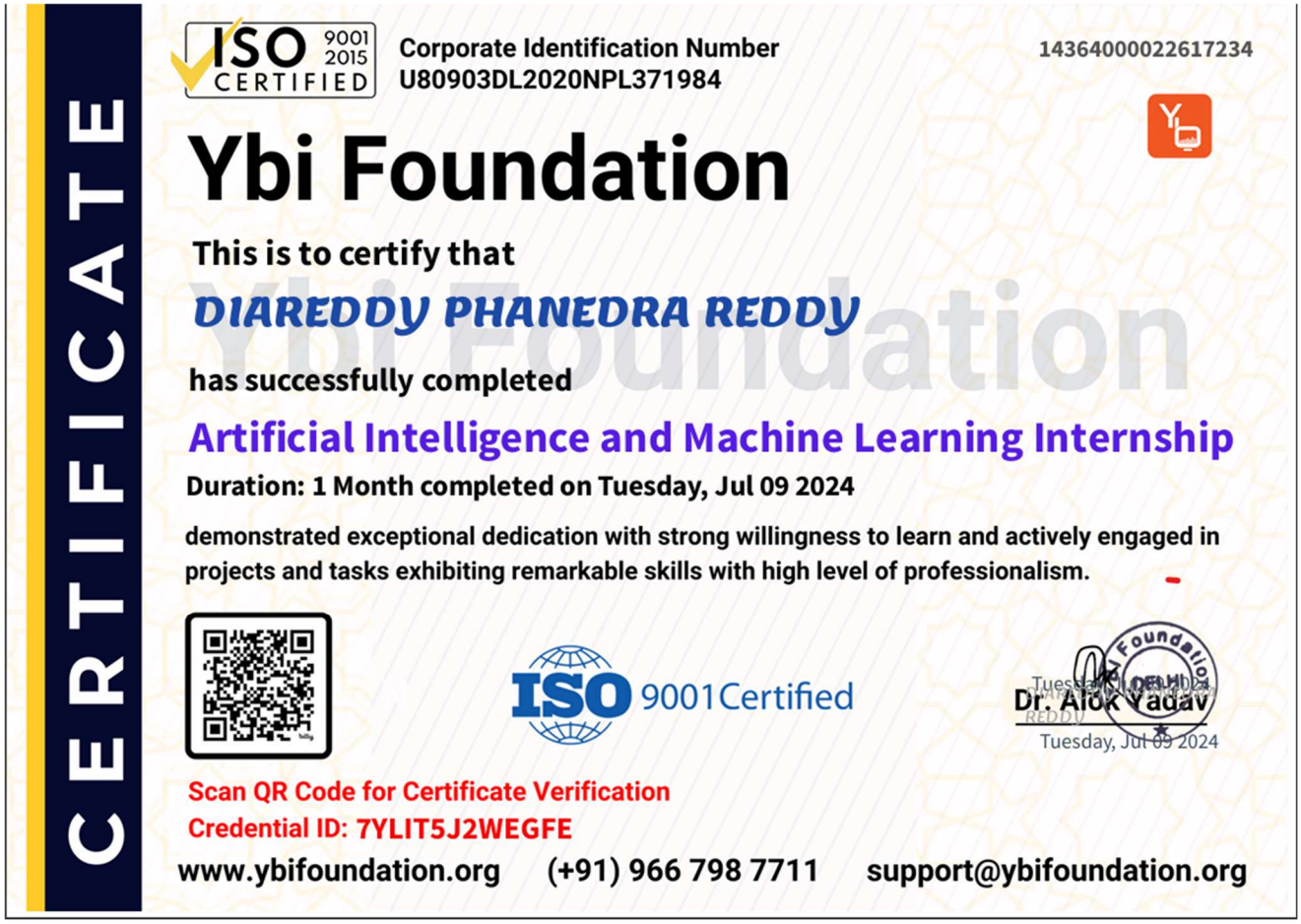
R.V.R.&J.C.COLLEGEOFENGINEERING(Autonomous)

(Affiliated to Acharya Nagarjuna University)

Chandramoulipuram : Chowdavaram

GUNTUR – 522 019

Internship Certificate



ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without proper suggestions, guidance and environment. Combination of these three factors acts like backbone to our Internship **"**Artificial Intelligence and Machine Learning Internship**".**

We would like to express our profound gratitude to **Dr.M.Sreelatha**, Head of the Department of Computer Science and Engineering and **Dr. Kolla Srinivas,** Principal, for their encouragement and support to carry out this Internship successfully.

We are very much thankful to YBI Foundation, for having allowed us to conduct the study needed for the Internship.

Finally we submit our heartfelt thanks to all the staff in the Department of Computer Science and Engineering and to all our friends for their cooperation during the Internship work.

DIAREDDDY PHANEDRA REDDY (Y22CS042)

\

ABSTRACT

This report outlines the contributions and experiences of an Artificial Intelligence

and Machine Learning (AI&ML) intern at YBI Foundation. During the internship, I engaged in various projects aimed at leveraging AI technologies to drive social impact and enhance operational efficiencies. Key responsibilities included data analysis, model development, and the application of machine learning algorithms to solve real-world problems.

I collaborated with a multidisciplinary team to design and implement predictive models that informed decision-making processes within the organization. Additionally, I participated in workshops and training sessions, enhancing my understanding of AI ethics, data privacy, and the implications of AI in societal contexts.

The internship not only deepened my technical skills in programming languages such as Python and R but also provided valuable insights into the intersection of technology and social entrepreneurship. This experience has equipped me with the necessary tools to contribute effectively to future AI&ML projects, emphasizing responsible innovation and sustainable practices in technology.

Overall, my time at YBI Foundation has been instrumental in shaping my career aspirations in the field of AI&ML, reinforcing my commitment to harnessing technology for positive change.

CONTENTS

1. Introduction to Python

→*Installation and Setup*

Step-1:

Install Python: Jupyter requires Python to run. Download and install Python from python.org.

Step-2:

Install Jupyter using pip: Open a command prompt or terminal and run:

pip install jupyter

Step-3:

Launch Jupyter Notebook: After installation, run the following command to start Jupyter run:

jupyter notebook

Step-4:

Create a Notebook: In the Jupyter interface, click on "New" and select "Python 3" to create a new notebook for your projects.

Step-5:

Save and Manage Notebooks: You can save your notebooks with the *".ipynb"* extension and organize them using the web-based interface.

→*Variables*

- Definition: Variables are containers used to store data values in a program.
- Dynamic Typing: In Python, variables do not need explicit declaration and can change types dynamically.
- Naming Rules: Variable names must start with a letter or underscore (_) and cannot begin with a number. They should not include special characters (e.g., @, #, $).
- Type Checking: Use type(variable_name) to check the type of a variable.
- Reassignment: Variables can be reassigned to different data types as needed.

→ ***Data Types***

Python has several built-in data types to handle different kinds of data:

- **Numeric Types**:
  - **int**: Integer values (e.g., `10`, `-5`).
  - **float**: Floating-point numbers (e.g., `3.14`, `-7.2`).
  - **complex**: Complex numbers (e.g., `3 + 4j`).
- **Text Type**:
  - **str**: String values (e.g., `"Hello"`, `'World'`).
- **Sequence Types**:
  - **list**: Ordered, mutable collection (e.g., `[1, 2, 3]`).
  - **tuple**: Ordered, immutable collection (e.g., `(1, 2, 3)`).
  - **range**: Sequence of numbers (e.g., `range(5)`).
- **Mapping Type**:
  - **dict**: Key-value pairs (e.g., `{"name": "John", "age": 25}`).
- **Set Types**:
  - **set**: Unordered collection of unique elements (e.g., `{1, 2, 3}`).
  - **frozenset**: Immutable set.
- **Boolean Type**:
  - **bool**: Represents `True` or `False`.
- **None Type**:
  - **None**: Represents the absence of a value (`None`).

→ ***Conditional Statements***

In Python, conditional statements are used to execute certain blocks of code based on specific conditions. The most commonly used conditional statements include if, elif (short for "else if"), and else. Let's look at their syntax and usage.

1. *if Statement*:The if statement is used to test a condition. If the condition evaluates to True, the code block under it is executed.

   Syntax:

   ```
   if condition:
       # Code block to execute if the condition is True
   ```

2. *if-else*:The else statement is used to execute a block of code if the condition in the if statement is False.

Syntax:

```
if condition:
        # Code block to execute if the condition is True
else:
        # Code block to execute if the condition is False
```

3. *if-elif-else:*The elif statement allows you to check multiple expressions for True. If a previous if or elif condition is False, the next elif statement is checked.

Syntax:

```
if condition1:
        # Code block to execute if condition1 is True
elif condition2:
        # Code block to execute if condition2 is True
else:
        # Code block to execute if none of the conditions are True
```

4. *Nested if statements:*You can nest if statements inside other if statements. This allows you to check multiple conditions in a hierarchical manner.

Syntax:

```
if condition1:
if condition2:
        # Code block to execute if condition1 and condition2 are True
```

5. *Short-Hand if:*Python allows a short form of the if statement, especially for simple assignments.

Syntax:

```
value_if_true if condition else value_if_false
```

→*Loops:*

There are Two types of loops. They are :

1.for Loop
2.while Loop

-

Syntax:

```
for variable in sequence:
        # Code block to execute
```

1. for Loop:

The for loop in Python is used to iterate over a sequence (such as a list, tuple, dictionary, set, or string). It is commonly used when you know the number of iterations in advance

2. ***while Loop:***

The while loop executes a block of code as long as a specified condition is True. It is useful when the number of iterations is not known beforehand.

Syntax:

```
while condition:
    # Code block to execute
```

→*Functions:*

In Python, functions are blocks of reusable code that perform a specific task. They help organize and modularize code, making it easier to manage, debug, and reuse. Python functions can take inputs, execute code, and return outputs.

Defining a Function:A function is defined using the def keyword, followed by the function name and parentheses ().

Syntax:

```
def function_name(parameters):

    # Code block return value

    # Optional: Returns a value
```

→OOP's Concepts:

Object-Oriented Programming (OOP) in Python is a programming paradigm that uses objects and classes to design and structure the code. It enables developers to build programs using concepts like inheritance, encapsulation, abstraction, and polymorphism, making the code more modular, reusable, and organized.

Key OOP Concepts in Python:

1. *Class*
   - A class is a blueprint for creating objects. It defines a set of attributes and methods that the created objects can use.
   -
   - Syntax:

```
class ClassName:

    # Class attributes and methods go here
```

2. *Object*
   - An object is an instance of a class. It has its own state and can interact with other objects using class-defined methods.
   - Example:

```
class Dog:

    def bark(self):

        print("Woof!")

my_dog = Dog()  # Creating an object of the Dog class

my_dog.bark()   # Outputs: Woof!
```

3. *Attributes*
   - Attributes are variables that belong to a class or an object. They define the properties of the class.
4. *Methods*

- Methods are functions defined inside a class that describe the behavior of the objects. The first parameter of a method is always self, which refers to the current object.

Four Pillars of OOP:

1. *Encapsulation*
   - Bundling the data (attributes) and methods that operate on the data into a single unit (class).
   - Restricting access to certain attributes or methods using private (__), protected (_), and public attributes.
   - Example:
2. *Inheritance*
   - Creating a new class (child) that derives attributes and behaviors from an existing class (parent). It promotes code reusability.
   - Example:
3. *Polymorphism*
   - The ability to use a common interface for multiple forms (data types). It allows methods to perform different tasks based on the object calling them.
   - Example:
4. *Abstraction*
   - Hiding the internal details and showing only the essential features of an object.
   - Achieved using abstract classes (with the abc module) and interfaces.

2. NumPy & Pandas

→***NumPy:***

NumPy (Numerical Python) is a powerful library in Python used for numerical computations, particularly with arrays and matrices. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays.

Key Features of NumPy:

1. N-dimensional Arrays (ndarray):
   - The core data structure of NumPy is the ndarray, which allows efficient storage and manipulation of large datasets.
2. Mathematical Operations:
   - Provides functions for mathematical operations like addition, subtraction, multiplication, and more on arrays.
3. Broadcasting:
   - Supports broadcasting, which allows operations on arrays of different shapes.
4. Linear Algebra, Fourier Transforms, and Random Number Generation:
   - Includes modules for performing linear algebra operations, Fast Fourier Transforms (FFT), and generating random numbers.
5. Integration with Other Libraries:
   - Works well with other scientific libraries like SciPy, Pandas, and Matplotlib.

Some Basic NumPy Functions:

1.numpy.array():Creates a NumPy array from a list, tuple, or other sequence-like object.

2. numpy.zeros():Generates an array filled with zeros of a specified shape.

3. numpy.ones():Generates an array filled with ones of a specified shape.

4. numpy.arange():Returns an array with evenly spaced values within a specified range.

5. numpy.linspace():Creates an array of evenly spaced values between a start and stop point with a specified number of points.

6. numpy.reshape():Reshapes an existing array into a new shape without changing its data.

7. numpy.sum():Computes the sum of array elements over a given axis.

8. numpy.mean():Calculates the mean (average) of array elements along a specified axis.

9. numpy.max() and numpy.min():Returns the maximum and minimum values in an array, respectively.

10. numpy.concatenate():Joins two or more arrays along a specified axis.

→***Pandas:***

Pandas is a powerful and flexible open-source library in Python used for data manipulation and analysis. It provides data structures like Series (1D) and Data Frame (2D) to handle and process structured data. It is widely used for cleaning, transforming, and analysing large datasets, making it essential for data science and machine learning tasks.

Key Features of Pandas:

1. **Data Structures**:
    - **Series**: A one-dimensional labelled array capable of holding any data type (integers, strings, floating points, etc.). It is similar to a column in a table.
    - **Data Frame**: A two-dimensional labelled data structure with columns of potentially different data types. It is similar to a table or a spreadsheet.
2. **Data Cleaning and Preprocessing**:
    - Handling missing data, removing duplicates, and filling or dropping null values.
3. **Data Manipulation**:
    - Functions for merging, joining, concatenating, and reshaping data.

4. **Data Analysis**:
    - Grouping data using groupby, generating pivot tables, and performing statistical operations.
5. **File I/O Operations**:
    - Reading from and writing to various file formats like CSV, Excel, JSON, SQL databases, and more.
6. **Data Indexing and Selection**:
    - Allows label-based, position-based, and Boolean indexing for selecting specific rows or columns.
7. **Time-Series Analysis**:
    - Powerful time-series data handling, including date range generation and frequency conversion.

USE CASES:

- **Data Wrangling**:
  Cleaning and transforming raw data.
- **Exploratory Data Analysis (EDA)**:
  Summarizing and visualizing the main characteristics of data.
- **Data Visualization**:
  Integration with Matplotlib and Seaborn for visualizing datasets.
- **Data Export**:
  Saving the cleaned and processed data in different formats like CSV or Excel.

Here are 10 basic Pandas functions with brief descriptions:

1. **pd.read_csv()**:
    - Reads a CSV file and converts it into a Pandas DataFrame.
2. **DataFrame.head()**:
    - Returns the first few rows of a DataFrame (default is 5 rows). Useful for quickly viewing the top of a dataset.
3. **DataFrame.info()**:
    - Displays a concise summary of the DataFrame, including the index, column names, data types, and memory usage.
4. **DataFrame.describe()**:
    - Generates descriptive statistics for numerical columns, such as mean, median, min, max, and standard deviation.
5. **DataFrame.shape**:
    - Returns the dimensions of the DataFrame (number of rows and columns).

6. **DataFrame.dropna()**:
    - Removes missing values (NaN) from the DataFrame, either row-wise or column-wise.
7. **DataFrame.fillna()**:
    - Replaces missing values with a specified value or method (e.g., filling NaN values with 0 or the mean of the column).
8. **DataFrame.groupby()**:
    - Groups the DataFrame by one or more columns and allows aggregation and computation on grouped data.
9. **DataFrame.sort_values()**:
    - Sorts the DataFrame based on the values of one or more columns, either in ascending or descending order.
10. **DataFrame.merge()**:
    - Merges two DataFrames based on a common column or index, similar to SQL joins (e.g., inner, outer, left, right joins).

These functions are fundamental for data manipulation and analysis in Pandas

3. Introduction to Google Colab

Google Colab

Google Colab is a cloud-based Jupyter notebook environment from Google

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Introducing Colaboratory

Introducing Colaboratory

This 3-minute video gives an overview of the key features of Colaboratory:

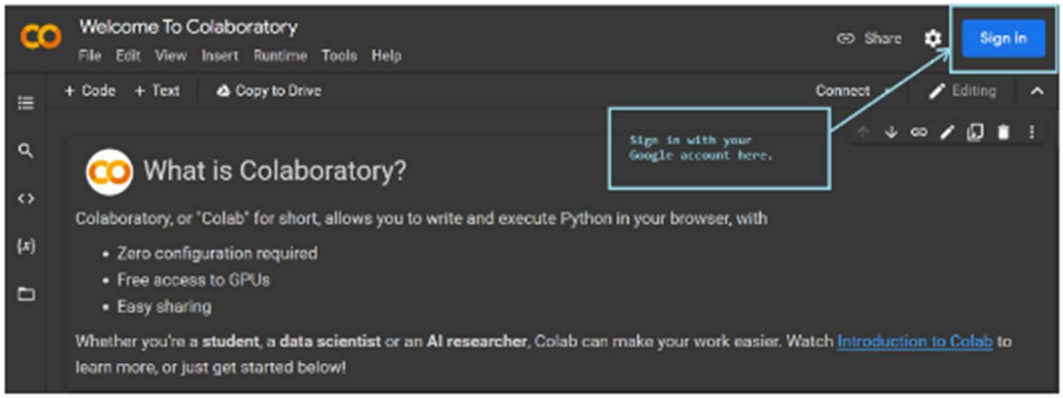
Show code

Video URL: https://youtu.be/inN8seMm7UI

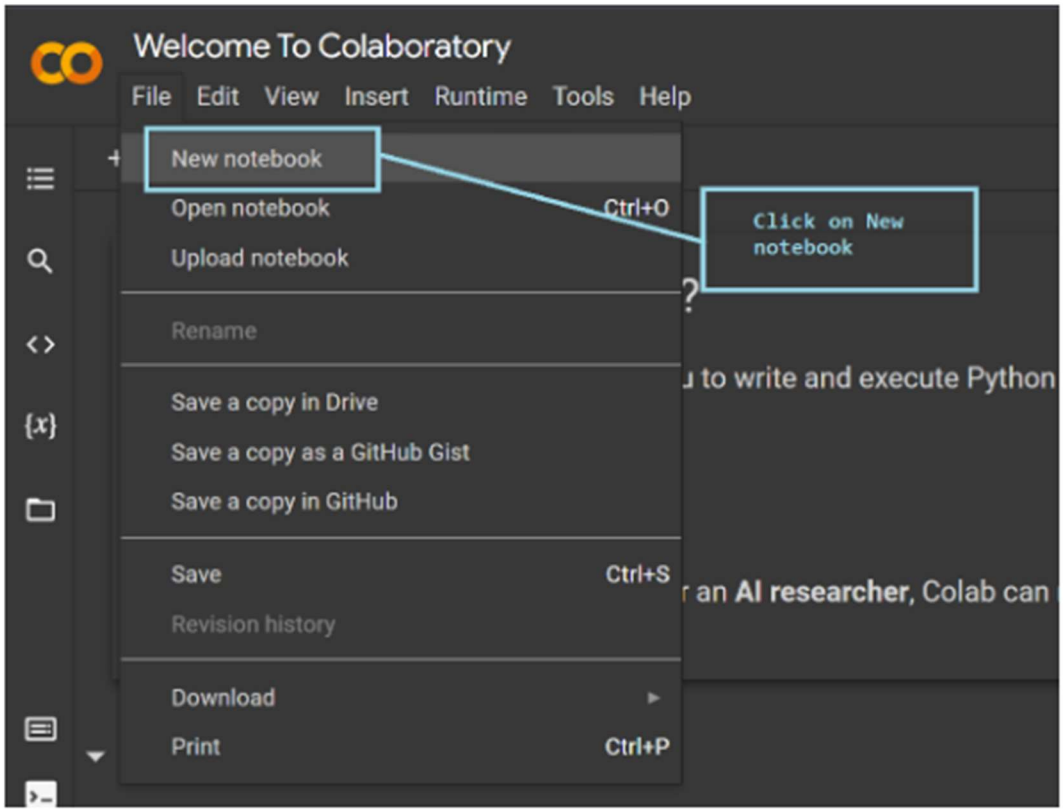
Creating your first Google Colab notebook

Go to colab.research.google.com

Sign in with your Google credentials



Once you've signed in to Colab, you can create a new notebook by clicking on 'File' → 'New notebook', as shown below:



Import Library

In [1]:
```
import pandas as pd
```

Import Dataset

In [2]:
```
df=pd.read_csv(r'https://raw.githubusercontent.com/YBI-Foundation/Dataset/main/Big%20Sales%20Data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishı
0	FDT36	12.3	Low Fat	0.111448	Baking Goods	33.4874	OUT049	1999
1	FDT36	12.3	Low Fat	0.111904	Baking Goods	33.9874	OUT017	2007
2	FDT36	12.3	LF	0.111728	Baking Goods	33.9874	OUT018	2009
3	FDT36	12.3	Low Fat	0.000000	Baking Goods	34.3874	OUT019	1985
4	FDP12	9.8	Regular	0.045523	Baking Goods	35.0874	OUT017	2007

```
# This is code cell
```

```
a = 10
a
```

```
10
```

Text cells

This is a **text cell**. You can **double-click** to edit this cell. Text cells use markdown syntax. To learn more, see our markdown guide.

You can also add math to text cells using LaTeX to be rendered by MathJax. Just place the statement within a pair of $ signs. For example `$\sqrt{3x-1}+(1+x)^2$` becomes $\sqrt{3x-1}+(1+x)^2$.

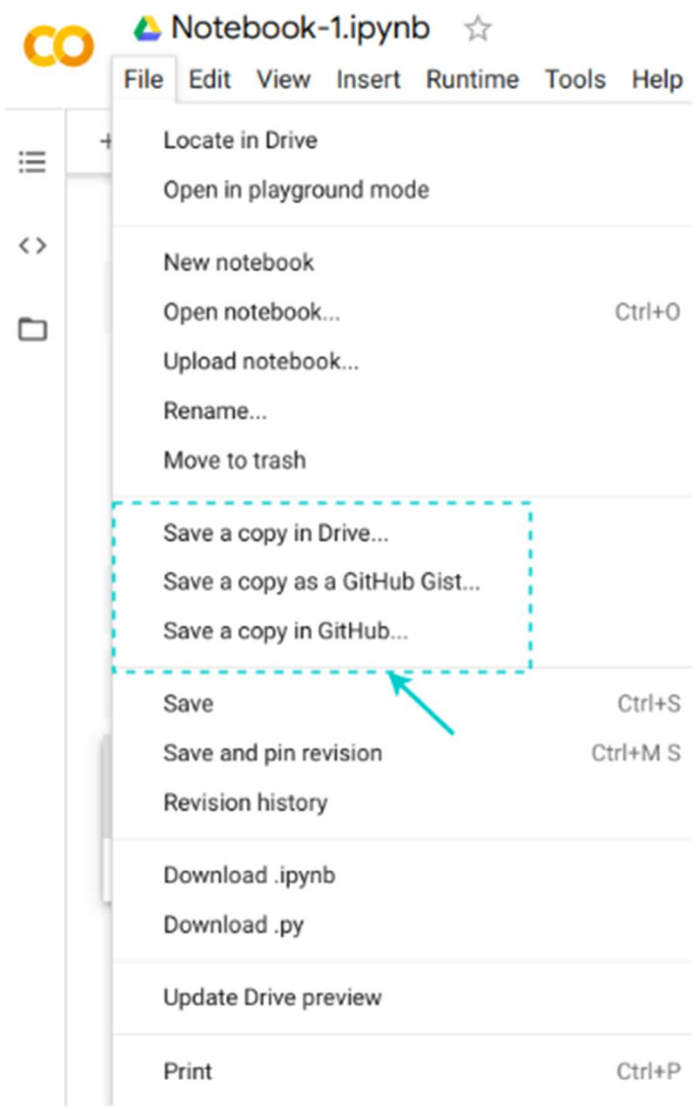
Adding and moving cells

You can add new cells by using the **+ CODE** and **+ TEXT** buttons that show when you hover between cells. These buttons are also in the toolbar above the notebook where they can be used to add a cell below the currently selected cell.

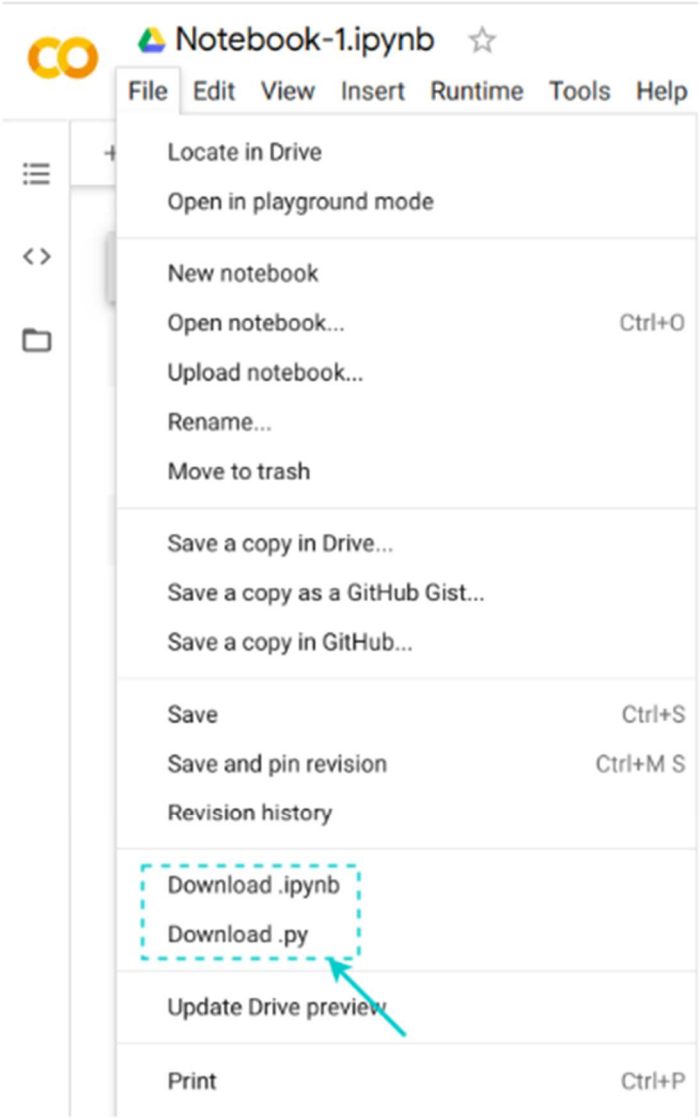
You can move a cell by selecting it and clicking **Cell Up** or **Cell Down** in the top toolbar.

Consecutive cells can be selected by "lasso selection" by dragging from outside one cell and through the group. Non-adjacent cells can be selected concurrently by clicking one and then holding down Ctrl while clicking another. Similarly, using Shift instead of Ctrl will select all intermediate cells.

Save


Notebook-1.ipynb
File Edit View Insert Runtime Tools Help
Locate in Drive
Open in playground mode
New notebook
Open notebook... Ctrl+O
Upload notebook...
Rename...
Move to trash
Save a copy in Drive...
Save a copy as a GitHub Gist...
Save a copy in GitHub...
Save Ctrl+S
Save and pin revision Ctrl+M S
Revision history
Download .ipynb
Download .py
Update Drive preview
Print Ctrl+P

Download



Easy Sharing

What is Markdown?

Colab has two types of cells: text and code. Text cells are formatted using a simple markup language called Markdown.

To see the Markdown source, double-click a text cell, showing both the Markdown source and the rendered version. Above the Markdown source there is a toolbar to assist editing.

Reference

Markdown	Preview
`**bold text**`	**bold text**
`*italicized text*` or `_italicized text_`	*italicized text*
`` `Monospace` ``	`Monospace`
`~~strikethrough~~`	~~strikethrough~~
`[A link](https://www.google.com)`	A link

Markdown	Preview
`![An image](https://www.google.com/images/rss.png)`	

Headings are rendered as titles.

```
# Section 1
# Section 2
## Sub-section under Section 2
### Sub-section under the sub-section under Section 2
# Section 3
```

Section 1

Section 2

Sub-section under Section 2

Sub-section under the sub-section under Section 2

Section 3

The table of contents, available on the left side of Colab, is populated using at most one section title from each text cell.

```
>One level of indentation
```

Code blocks

```python
print("a")
```

```
print("a")
```

Ordered lists:

```
1. One
1. Two
1. Three
```

1. One
2. Two
3. Three

Unordered lists:

```
* One
* Two
* Three
```

- One
- Two
- Three

Equations:

```
$y=x^2$

$e^{i\pi} + 1 = 0$

$e^x=\sum_{i=0}^\infty \frac{1}{i!}x^i$

$\frac{n!}{k!(n-k)!} = {n \choose k}$

$A_{m,n} =
 \begin{pmatrix}
  a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
  a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
  \vdots  & \vdots  & \ddots & \vdots  \\
  a_{m,1} & a_{m,2} & \cdots & a_{m,n}
```

4. Introduction To Machine Learning

→Overview:

Machine Learning (ML) is a subfield of artificial intelligence (AI) that enables computers to learn from data and improve their performance on tasks without being explicitly programmed. It involves building models that can make predictions or decisions based on data. ML has become a core part of technology and is used in various fields, including finance, healthcare, marketing, and many others.

How Machine Learning Works???

Machine learning models are trained using historical data, identifying patterns and relationships within that data. Once trained, these models can be used to make predictions on new, unseen data. The typical workflow involves:

1. **Data Collection**: Gathering data from various sources (databases, sensors, or files).
2. **Data Preprocessing**: Cleaning, formatting, and transforming the data to make it suitable for training.
3. **Model Training**: Feeding the prepared data into a learning algorithm that creates a model by adjusting its parameters based on patterns in the data.
4. **Model Evaluation**: Assessing the model's performance using metrics such as accuracy, precision, and recall.
5. **Model Deployment**: Using the trained model to make predictions on new data.

→Types:

Supervised Learning:

- The model is trained on labelled data, meaning each training example has a corresponding label or output.
- Examples: Classification (spam detection), Regression (predicting house prices).

Unsupervised Learning:

- The model is trained on unlabelled data, meaning the system tries to learn the patterns and structure from the input data without predefined labels.
- Examples: Clustering (customer segmentation), Association (market basket analysis).

Reinforcement Learning:

- Involves training models to make sequences of decisions by rewarding or penalizing them based on their actions. The model learns through trial and error.
- Examples: Robotics, game playing, and autonomous driving.

→**Applications of Machine Learning:**

Machine learning is used in a wide variety of real-world applications, such as:

- Image and Speech Recognition: Face detection, voice assistants (e.g., Siri, Alexa).
- Natural Language Processing (NLP): Sentiment analysis, language translation.
- Recommender Systems: Personalized content on platforms like Netflix, YouTube, and Amazon.
- Predictive Analytics: Stock market predictions, risk assessment.
- Healthcare: Disease detection, personalized treatment.

Benefits of Machine Learning

- Automation of Tasks: Enables automation of repetitive and time-consuming tasks.
- Data-Driven Decisions: Helps in making better decisions based on data insights.
- Scalability: Can handle large volumes of data more efficiently than traditional methods.

Adaptability: Models can learn from new data and improve over time.

5. Linear Regression

Linear Regression is a fundamental statistical and machine learning algorithm used to model the relationship between a dependent (target) variable and one or more independent (predictor) variables. It attempts to establish a linear relationship by fitting a straight line (y = mx + c) through the data points that minimizes the difference between the actual and predicted values.

Key Concepts:

1. Dependent Variable (Y): The outcome or target variable we want to predict.
2. Independent Variable (X): The input features or variables used to predict the dependent variable.
3. Line of Best Fit: A straight line ($Y = b0 + b1X$) that best represents the relationship between the variables.
    - *b0* is the intercept (the value of Y when X = 0).
    - *b1* is the slope (how much Y changes for a unit change in X).

→***Types of Linear Regression:***

We have triggered of mainly 3 types of Linear Regression.

The are:

1. Simple Linear Regression.

2. Multiple Linear Regression

1.Simple Linear Regression: It is a statistical technique used to model the relationship between two variables: one independent variable (predictor) and one dependent variable (outcome). It aims to find the best-fitting straight line that describes how changes in the independent variable affect the dependent variable.

Key Components:

- Equation: The relationship is represented by the equation:

$$Y=b_0+b_1X$$

where:

- Y = Dependent variable (what you are trying to predict)
- X = Independent variable (the predictor)
- *b0* = Intercept (the value of Y when X is 0)
- b_1 = Slope (the change in Y for a one-unit change in X)

- Objective: The goal of simple linear regression is to minimize the sum of the squared differences (errors) between the observed values and the values predicted by the model. This is often done using the *Least Squares Method*.

2. Multiple Linear Regression: It is a statistical method used to model the relationship between one dependent variable (outcome) and two or more independent variables (predictors). It extends simple linear regression by allowing for multiple predictors, enabling a more comprehensive analysis of how various factors collectively influence the dependent variable.

Key Components:

- Equation: The relationship is represented by the equation:

$$Y=b_0+bX_1+bX_2+\ldots+b_nX_n$$

Where:

- Y = Dependent variable (the outcome you want to predict)
- $X_1,X_2,\ldots,X_n$= Independent variables (the predictors)
- b_0 = Intercept (the value of YYY when all XXX values are 0)
- b1,b2,…,bn= Coefficients (indicate the change in YYY for a one-unit change in each corresponding XXX)

- Objective: Similar to simple linear regression, the goal is to minimize the sum of the squared differences (errors) between the actual values and the values predicted by the model. This is also typically done using the Least Squares Method.

Assumptions:

1. Linearity: The relationship between the dependent variable and independent variables is linear.
2. Independence: The residuals (errors) are independent of each other.

3. Homoscedasticity: The residuals have constant variance at all levels of the independent variables.
4. No Multicollinearity: Independent variables should not be highly correlated with each other.
5. Normality: The residuals should be normally distributed.

→**Model Evaluation Metrics for Linear Regression:**

Evaluating a linear regression model involves assessing its accuracy and goodness-of-fit. Common metrics include:

1. Mean Absolute Error (MAE)
    - Measures the average magnitude of the errors between predicted and actual values, without considering their direction.
    - Formula: $MAE = (1/n) * \Sigma |Yi - \hat{Y}i|$
    - Interpretation: Lower values indicate better model performance.
2. Mean Squared Error (MSE)
    - Calculates the average of squared differences between predicted and actual values.
    - Formula: $MSE = (1/n) * \Sigma (Yi - \hat{Y}i)^2$
    - Interpretation: Penalizes larger errors more than MAE, making it sensitive to outliers.
3. Root Mean Squared Error (RMSE)
    - Square root of the MSE, providing an error value in the same units as the target variable.
    - Formula: $RMSE = \sqrt{((1/n) * \Sigma (Yi - \hat{Y}i)^2)}$
    - Interpretation: Lower RMSE indicates better model performance.
4. R-squared (Coefficient of Determination)
    - Represents the proportion of variance in the dependent variable explained by the independent variables.
    - Formula: $R^2 = 1 - (SSres / SStot)$
    - Where SSres is the sum of squared residuals, and SStot is the total sum of squares.
    - Interpretation: Ranges from 0 to 1, where higher values indicate a better fit.
5. Adjusted R-squared
    - Adjusts the R-squared value to account for the number of predictors in the model.

6. Logistic Regression

Logistic Regression is a statistical method used for binary classification problems, where the outcome variable is categorical and takes on two possible values, such as 0 or 1, true or false, yes or no. It predicts the probability that a given input belongs to a specific category.

Key Features:

- Prediction: Unlike linear regression, which predicts continuous outcomes, logistic regression predicts probabilities using the logistic (sigmoid) function. This function transforms any real-valued number into a value between 0 and 1, making it suitable for binary outcomes.
- Equation: The logistic regression model can be expressed as:

$$P(Y=1|X)=1 / (1+e^{-(b_0+b_2X_1+b_2X_2+\ldots+b_nX_n)})$$

where:

  - $P(Y=1|X)$ is the probability of the dependent variable YYY being 1 given the independent variables $X_1,X_2,\ldots,X_n$
  - $b_0$is the intercept, and $b_1,b_2,\ldots,b_n$ are the coefficients of the independent variables.
- Decision Boundary: A threshold (commonly 0.5) is set to classify the predicted probabilities into categories. If the predicted probability exceeds this threshold, the observation is classified as 1; otherwise, it is classified as 0.

→Binary Logistic Regression:

It is a specialized form of logistic regression used to model binary outcomes, where the dependent variable has only two possible categories (e.g., success/failure, yes/no, or 1/0). This method is particularly useful in situations where the goal is to predict the probability that a certain event occurs based on one or more independent variables.

Key Features:

- Logistic Function: Binary logistic regression uses the logistic (or sigmoid) function to transform linear combinations of the input variables into a probability that falls between 0 and 1. The function is defined as:

$$P(Y=1|X) = 1 / (1+e^{-(b_0 + b_2 X_1 + b_2 X_2 + \ldots + b_n X_n)})$$

where:

  - $P(Y=1|X)$ is the probability that the dependent variable YYY equals 1 given the independent variables $X_1, X_2, \ldots, X_n$
  - b_0 is the intercept, and $b_1, b_2, \ldots, b_n$ are the coefficients of the independent variables.
- Decision Threshold: After computing probabilities, a threshold (commonly set at 0.5) is applied to classify the observations. If the predicted probability is greater than or equal to 0.5, the outcome is classified as 1; otherwise, it is classified as 0

Assumptions:

1. Binary Outcome: The dependent variable must be binary.
2. Independence of Observations: Each observation should be independent of the others.
3. No Multicollinearity: Independent variables should not be highly correlated with each other.
4. Linearity of the Logits: The relationship between the independent variables and the log-odds of the dependent variable should be linear.

→Multinomial Logistic Regression:

It is an extension of binary logistic regression used when the dependent variable has more than two categories (more than two possible outcomes). This method is particularly useful for situations where you want to model outcomes that are categorical and do not have a natural ordering (nominal variables).

Key Features:

- Modelling Multiple Categories: Unlike binary logistic regression, which predicts the probability of two classes, multinomial logistic regression can handle three or more classes. For example, it can be used to predict outcomes such as "red," "blue," or "green" for a color choice or "cat," "dog," or "bird" for a pet selection.

- Logistic Function: The model estimates the probabilities of each class relative to a reference category. The general form of the model is expressed as:

$$P(Y=k|X) = (e^{(b_{0k} + b_{1k}X_1 + b_{2k}X_2 + \ldots\ldots\ldots + b_{nk}X_n)}) / \Sigma^{K}_{j=1}\, e^{(b_{0j} + b_{1j}X_1 + b_{2j}X_2 + \ldots\ldots\ldots + b_{nj}X_n)}$$

Where:

  - $P(Y=k|X)$ is the probability that the dependent variable Y equals category k given the independent variables $X_1, X_2, \ldots, X_n$.
  - $b_0, b_1, b_2, \ldots, b_n$ are the coefficients for the independent variables for category k.
  - K is the total number of categories.
- Reference Category: In multinomial logistic regression, one category is typically designated as the reference category. The coefficients for other categories are estimated in relation to this reference category.

Assumptions:

1. Multinomial Outcome: The dependent variable should be categorical with more than two classes.
2. Independence of Observations: Each observation must be independent of others.
3. No Multicollinearity: Independent variables should not be highly correlated with one another.
4. Linear Relationship: There should be a linear relationship between the independent variables and the log-odds of the outcome categories.

Applications:

- Marketing Research: Understanding consumer preferences among multiple product options (e.g., choosing between different brands of a product).
- Healthcare: Predicting patient treatment choices among multiple therapies or medication options.
- Social Sciences: Analysing survey data where respondents can select one of several possible responses.

→Model Evaluation Metrics:

The performance of a multinomial logistic regression model can be evaluated using several metrics, including:

1. Confusion Matrix: A table summarizing actual versus predicted classifications for all classes, including true positives, false positives, and false negatives for each class.
2. Accuracy: The proportion of correct predictions across all classes:

$$Accuracy = Total\ Correct\ Predictions / Total\ Observations$$

3. Precision: For each class, the proportion of true positive predictions to the total predicted positives.
4. Recall (Sensitivity): For each class, the proportion of true positive predictions to the actual positives.
5. F1 Score: The harmonic mean of precision and recall for each class, providing a balance between the two metrics.
6. Macro and Micro Averages: These are methods of aggregating precision, recall, and F1 scores across multiple classes.
    - Macro Average: Computes metrics for each class and then averages them, treating all classes equally.

7. Conclusion and Future Work

My artificial intelligence and machine learning internship has been a transformative and enriching experience, providing me with invaluable insights into the real-world applications of my academic knowledge. During this period, I worked on diverse projects, gaining hands-on experience with technologies such as chatbots, generative AI, object detection, visual question answering, image classification, and computer vision. This training involved a combination of instructive tasks and a practice environment that allowed me to implement what I learned, significantly solidifying my understanding of these key concepts.

The field of artificial intelligence and machine learning is rapidly evolving, and the demand for skilled professionals is increasing across various sectors. Organizations are increasingly seeking AI talent to enhance their capabilities and innovate their products and services. With the advent of technologies like Google Teachable Machine and DALL·E, the opportunities for aspiring AI professionals are vast and varied. My internship has equipped me with the essential skills and knowledge needed to thrive in this dynamic landscape, paving the way for a promising career in artificial intelligence and machine learning.

Looking to the future, the field of artificial intelligence and machine learning continues to evolve rapidly, presenting numerous opportunities for innovation and problem-solving. The skills and knowledge gained from this internship will be instrumental in developing intelligent solutions that enhance user experiences and improve automation across various industries. The journey in AI&ML promises to be dynamic and impactful, paving the way for advancements that can significantly benefit society.