By Khadijah Sa'ad Mohammed

Python Programming Tutorial for Machine Learning Beginners Using Google Colab

Introduction to Google Colab

Google Colab is an online platform that lets you write and run Python code through the browser. It is particularly useful for machine learning projects because it provides free access to GPUs (Graphics Processing Units), which can speed up the computation needed for machine learning.

Getting Started with Google Colab:

1. Go to Google Colab. https://colab.research.google.com/
2. Sign in with your Google account.
3. Click `File` > `New notebook` to start a new project, creating a notebook where you can enter Python code.

Session 1: Python Basics in Google Colab

1. Writing Your First Program:

- Task: Print "Hello, Machine Learning!" to the screen.
- How: Type the following Python code into a new cell in your Google Colab notebook:
  ```
  print("Hello, Machine Learning!")
  ```
- Execute: Press `Shift + Enter` to run the code in the cell.

2. Understanding Comments

In Python, the hash symbol # is used to start a comment in the code. A comment is a line of text in your program that is not executed as part of the program. Its primary purpose is to annotate the code to help programmers understand the code's functionality or intent more easily. Comments can also be used to temporarily disable parts of your code during testing without deleting it.

Here's how you use comments in Python:

Examples of Using Comments

1. Single-Line CommentsYou can write a comment on its own line or at the end of a line of code:

```
# This is a single-line comment

print("Hello, world!") # This comment follows a line of code
```

3. Understanding Variables and Basic Data Types

- Variables: Think of variables as containers that store data values. You can name them whatever you like.
- Data Types: Python has several data types including integers, floats (decimal numbers), and strings (text).

```
age = 25 # Integer: Whole number
height = 5.9 # Float: Decimal number
name = "Alice" # String: Text
print(age, height, name)
```

Session 2: Lists, Dictionaries, Arrays and Import

1. Working with Lists:

- Purpose: Lists store multiple items in a single variable.
- Example:

```
fruits = ["apple", "banana", "cherry"]
print(fruits)
```

2. Exploring Dictionaries:

- Purpose: Dictionaries hold data as key-value pairs, which is similar to how a real dictionary works with word definitions.
- Example:

```
student = {"name": "John", "age": 22}
print(student)
```

What is an Array?

An array is a data structure that stores a collection of items. In programming, arrays are used to organize data so that a related set of values can be easily sorted or searched. Unlike Python's built-in `list` type, which can store items of different data types, arrays typically require all elements to be of the same type, making them more efficient for certain operations.

Arrays in Python can be created and manipulated using the NumPy library, which provides a high-performance array object that is central to doing numerical computations.

What is import?

In Python, `import` is a keyword used to include the code contained in another Python source file. In other words, import allows you to access and use functions, classes, and variables defined

in other Python files. This is particularly useful for accessing Python libraries, which are collections of modules that include pre-written code you can include in your projects.

For example:

```
import numpy
```

This line tells Python to load the NumPy library, making all of NumPy's functions and features available in your script.

What does `as` do in Python?

The `as` keyword in Python is used in conjunction with the `import` statement to create an alias for the imported module. This lets you refer to the module with a different name, usually a shorter one, which is handy if you need to call it frequently in your code.

For example:

```
import numpy as np
```

Here, `np` is an alias for `numpy`. This means that instead of typing `numpy.array()` to create a new array, you can use the shorter `np.array()`.

What is NumPy?

NumPy, which stands for Numerical Python, is an open-source Python library that is widely used in data science and scientific computing. It's known for its powerful array object, but it also provides:

- Functions for performing complex mathematical and logical operations on arrays.
- Tools for integrating C, C++, and Fortran code.
- Useful features for linear algebra, Fourier transforms, and random number generation.

NumPy arrays are faster and more compact than Python lists. An array consumes less memory and is convenient to use for mathematical operations, particularly if you have to perform operations on large data sets, which is typical in machine learning and data analysis tasks.

Example of Using NumPy

Here's a simple example that demonstrates creating an array with NumPy and performing a mathematical operation:

```
# Importing the NumPy library and giving it an alias 'np'
import numpy as np

# Creating a NumPy array
arr = np.array([1, 2, 3, 4, 5])

# Performing an element-wise addition
new_arr = arr + 5

print(new_arr) # Output: [6 7 8 9 10]
```

This example shows how to create a NumPy array and then add a number to every element in the array using NumPy's ability to handle vectorized operations efficiently. Such capabilities make NumPy an invaluable tool for data processing in Python.

Session 3: Data Handling with Pandas

1. Introduction to Pandas

- Purpose: Pandas is a library that simplifies data manipulation and analysis.
- Loading Data:

```
import pandas as pd # Import the pandas library
url =
'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
data = pd.read_csv(url)
print(data.head()) # Displays the first 5 rows of the dataset
```

2. Basic Data Operations:

- View Data: See what your data looks like.

```
print(data.head()) # First 5 rows
```

- Statistics: Get a summary of the statistics pertaining to the DataFrame.

```
print(data.describe()) # Summary statistics
```

Further Explanation: Loading and Understanding the Titanic Survival Dataset

This dataset includes various passenger attributes such as age, sex, passenger class, and whether the passenger survived the sinking of the Titanic.

Step-by-Step Instructions to Load and View the Titanic Dataset:

1. Importing Libraries:

```
import pandas as pd # Pandas library for data manipulation
```

2. Loading the Dataset from a URL:
   The Titanic dataset is available on many platforms, but we'll use a version from GitHub that is clean and ready to use.

```
url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'

data = pd.read_csv(url)
```

3. Displaying the First Few Rows:

```
print(data.head()) # Displays the first 5 rows of the dataset
```

Here's how the output might look, displaying key data for the first few passengers:

```
PassengerId Survived Pclass Name Sex Age SibSp Parch Ticket Fare Cabin Embarked

0 1 0 3 Braund, Mr. Owen Harris male 22.0 1 0 A/5 21171 7.2500 NaN S

1 2 1 1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1 0 PC 17599 71.2833 C85 C

2 3 1 3 Heikkinen, Miss. Laina female 26.0 0 0 STON/O2. 3101282 7.9250 NaN S

3 4 1 1 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1 0 113803 53.1000 C123 S

4 5 0 3 Allen, Mr. William Henry male 35.0 0 0 373450 8.0500 NaN S
```

Key Columns in the Dataset:

- PassengerId: An identifier for the passenger.
- Survived: Whether the passenger survived (1) or not (0).
- Pclass: The passenger's class (1st, 2nd, or 3rd) which is a proxy for socio-economic status.
- Name: The name of the passenger.
- Sex: The passenger's gender.
- Age: The passenger's age.
- SibSp: The number of siblings or spouses the passenger had aboard.
- Parch: The number of parents or children the passenger had aboard.
- Ticket: The passenger's ticket number.
- Fare: How much the passenger paid for the ticket.
- Cabin: The passenger's cabin number.
- Embarked: The port where the passenger embarked (C = Cherbourg; Q = Queenstown; S = Southampton).

Basic Data Examination

It's useful to get a quick overview of the dataset, including checking for missing values and understanding the distribution of numerical values.

```
# Summary statistics for numerical features
print(data.describe())


# Count of missing values per column
print(data.isnull().sum())
```

Simple Visualization

Visualizing data can provide insights that are not obvious from raw numbers alone.

```
import seaborn as sns
import matplotlib.pyplot as plt
# Visualizing survival rates based on passenger class
sns.barplot(x='Pclass', y='Survived', data=data)
plt.title('Survival Rates by Passenger Class')
plt.show()
```

This approach with the Titanic dataset makes it easier for beginners to grasp basic data handling and analysis operations in Python, providing a foundation to build on for more complex machine learning tasks.

Session 4: Introduction to Machine Learning with Scikit-learn

1. Supervised Learning Example: Logistic Regression

- Purpose: Predict a category based on input variables.
- Preparing Data:

```
from sklearn.model_selection import train_test_split
X = data.drop(columns=['Outcome']) # Features
y = data['Outcome'] # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
```

- Building and Training the Model:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

- Making Predictions and Evaluating the Model:

```
predictions = model.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy:", accuracy_score(y_test, predictions))
```

2. Unsupervised Learning Example: K-means Clustering

- Purpose: Group data points into clusters based on feature similarity.
- Example:

```
from sklearn.cluster import KMeans
# Example dataset
X = pd.DataFrame({
 "x": [1, 2, 3, 6, 7, 8],
 "y": [1, 1, 2, 6, 7, 8]
})
kmeans = KMeans(n_clusters=2)
```

Further explanation: K-means Clustering Explained

K-means clustering is an unsupervised learning algorithm that seeks to partition a set of data points into a specified number of clusters K. The goal is to divide the data such that the sum of the squared distance between the data points and the centroid (mean) of their respective clusters is minimized

1. Importing Libraries

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

   - `pandas` is used for data manipulation and analysis.
   - `sklearn.cluster` contains the K-means clustering algorithm.
   - `matplotlib.pyplot` is used for creating visualizations to see the results.

2. Creating an Example Dataset

```
X = pd.DataFrame({
 "x": [1, 2, 3, 6, 7, 8],
 "y": [1, 1, 2, 6, 7, 8]
```


```
})
```

- This step initializes a DataFrame `X` with two features: `x` and `y`. These features represent coordinates in a 2D space where each point is a data point to be clustered.

3. Applying K-means Clustering

```
kmeans = KMeans(n_clusters=2)

kmeans.fit(X)
```

- `KMeans(n_clusters=2)` initializes the K-means algorithm to partition the data into 2 clusters.
- `.fit(X)` fits the model to the dataset `X`. This method runs the K-means clustering algorithm, which involves assigning initial centroids randomly, then iteratively relocating them to minimize the within-cluster sum of squares.

4. Storing Cluster Labels

```
labels = kmeans.labels_

X['Cluster'] = labels
```

- `kmeans.labels_` retrieves the cluster labels for each data point in `X`. These labels indicate the cluster to which each data point belongs.
- Adding these labels to the DataFrame `X` as a new column named `Cluster` helps in tracking and visualizing which point belongs to which cluster.

5. Visualizing the Clusters

```
plt.scatter(X['x'], X['y'], c=X['Cluster'], cmap='viridis')
plt.title('K-means Clustering')
plt.xlabel('X coordinate')
plt.ylabel('Y coordinate')
plt.show()
```

- `plt.scatter()` creates a scatter plot of the `x` and `y` coordinates. The color of each point (`c=X['Cluster']`) varies according to the cluster assignment, which helps in visually distinguishing the clusters.
- `cmap='viridis'` specifies the color map used to color the data points. Different color maps can be used depending on preference or visibility.
- `plt.show()` displays the plot. This visualization shows the final clustering result, where the goal of minimizing the intra-cluster distances and maximizing the inter-cluster distances is visually evident.

Thank you for following this Python Programming Tutorial for Machine Learning Beginners. If you have any questions or feedback, please feel free to reach out!

Khadijah Sa'ad Mohammed
admin@culminatech.com
www.culminatech.com

Happy coding!