Python for machine learning

Friday, 16 February 2024 8:09 PM

Why Python is prefered for Machine learning?

- Its data handling capacity is good
- Python is an extendable language and has support to interact with almost all the other languages and platform.
- Python is open source and hence, accessible to more people across the globe
- As a language, Python is quite easy to learn and user friendly

Python libraries for ML

Pandas : libraries used to work with datasets in python

Installation command

1. pip install pandas
2. pip3 install pandas

You Import can import pandas in python

```
import pandas as pd
```

Pandas DataFrames

Data frames let you store the observations in the form of rows and columns.

```
df_dict = { "fruits" = ["Apple", "mango", "Banana", "Pome"],
  "Vegetables": ["Carrot", "Zucchini", "Onion", "Tomato"],
 "Cereals": [ "Corn", "Wheat", "Barley", "Rice"],
 " price": [900, 100, 60, 75]}
```

```
import pandas as pd
groceries = pd. DataFrame (dF_dict)
 print (groceries)
```

	Fruits	Vegetables	Cereals	Price
0	Apple	Carrot	Corn	900
1	Mango	Zucchi	Wheat	100
2	Banana	Onion	Barley	60
3	Pome	Tomato	Rice	75

Default index

Assign your index to dataframe

```
df_dict = { "fruits" = ["Apple", "mango", "Banana", "Pome"],
  "Vegetables": ["Carrot", "Zucchini", "Onion", "Tomato"],
 "Cereals": [ "Corn", "Wheat", "Barley", "Rice"],
 " price": [900, 100, 60, 75]}
```

```
import pandas as pd
```

```
groceries = pd. DataFrame (df_dict)
 groceries. index = ["F1", "F2", "F3", "F4"]
 print (groceries)
```

	Fruits	Vegetables	Cereals	price
0	Apple	Carrot	Corn	400
1	Mango	Zucchi	Wheat	100
2	Banana	Onion	Barley	60
3	Pome	Tomato	Rice	75

Convert the Dataframe to CSV File " df. to_csv ('filename.csv')

CSV file

```
import pandas as pd
groceries = pd. read_csv ('dict.csv')
print (groceries)
```

Indexing data frame : You can use [ ] or dot (.) operator to index the columns in a dataframe.

```
                      print (df. vegetables)
pd. read_csv ('dict.csv')
```

print (df['Vegetables'])	Print vegetable Column as Pandas series
print (df [['Vegetables']])	Print vegetable Column as Pandas Dataframe
print (df [['Vegetables', 'Cereals']])	Index both Cereals and Vegetable Column as Dataframe
print (df ["Cereals"][0]	Index the First observation of Column.

Pandas loc and iloc

loc and iloc are used to index the columns and the individual observations.

To use loc , we need to use the labels , whereas iloc uses integers to index

```
import pandas as pd
df = pd. read_csv ('dict.csv')
df. index = ['0', '1', '2']
# Indexing the first row
print(df. loc [['0']])
# Indexing the first two row elements using iloc
   print (df. iloc [0][0:2])
```

x Numpy - x NumPy stands for Numerical Python, which can be used for performing all the mathematical and logical operations on multi-dimensional arrays in Python.

Command to install

```
pip install numpy
pip3 install numpy
```

numpy is Imported as

```
import numpy as np
```

Numpy array syntax and parameters

Syntax to declare a numpy array

```
numpy.array ( object , dtype=None, copy=True , order='K' , subok=False,
              ndmin=0)
```

Imp Parameters

- dtype: dtype is used to mention the type of array that you want to declare. It can be an integer, complex, a float and so on.
- ndmin: Specifies the minimum number of dimensions that the resulting array should have.

Other parameter are not often used.

Example of how a numpy array can be declared using the dtype

```
import numpy as np
arr = np.array([1,2,3], dtype=float)
print(arr)                                  → [1. 2. 3.]
```

Numpy ndarray creation

```
import numpy as np

arr = np.array([[-1, 2, 0, 4],
                [4, -0.5, 6, 0],
                [2.6, 0, 7, 8],
                [3, -7, 4, 2.0]])
print(arr)
fruit_cost = [20, 60, 90, 40, 50]
vegetable_cost = [10, 20, 30, 40, 50]
combine_np = np. array([fruit_cost, vegetable_cost])
print(combine_np)
print(type(fruit_np))              ← {class 'numpy.ndarray'}
```

Subsetting numpy array

Numpy array can be subsetted using square brackets ([]). You can use the higher than(>) or less than (<) symbol to subset the arrays

```
import numpy as np
arr = np. array([1,2,3,4,5,6,7,8,9,10])
```

Subset the Elements greater then 5

```
print (arr[arr>5])              ← [6, 7, 8, 9, 10]
```

Scipy: SciPy stands for Scientific Python
- helps in Scientific computation
- Build on top of NumPy, Operates on NumPy array

Scipy has added data science functionalities.

You can install scipy library with the following command

```
① pip install scipy
② pip3 install scipy
```

Functions of SciPy

Scipy has Package named special, which has numerous function like Exponential, cubic root, log, permutations and combinations and many more.

Exponential Function

```
  from scipy.special import exp10
  val = exp10([1,10])
  print(val)          ← [1.e+01  1.e+10]
```

Cubic root function

```
  from scipy.special import cbrt
 num = cbrt(27)
  print(num)       ← 3.0
```

Permutations and combinations

```
 from scipy.special import perm
 premutation = prem(4,2)
 print(permutation)      ← 12
```

```
 from scipy.special import comb
 combination = comb(4,2)
 print(combination)      ← 6
```

linear algebra with SciPy

package name: linalg (linear algebra)

```
 from scipy import linalg
 import numpy as np
 arr = np.array([[1,2],[3,4]])
 determinant = linalg.det(arr)
 print(determinant)      ← -2.0
```

```
 from scipy import linalg
  import numpy as np
  arr = np.array([[1,2],[3,4]])
  inverse = linalg.inv(arr)
   print(inverse)   ←        [[ 2.5  -3.5]
                               [-2.   3. ]]
```

Matplotlib

Used for plotting graphs to gain insights into your data.

Matplotlib.pyplot is a package used to plot 2-D graphs in python.

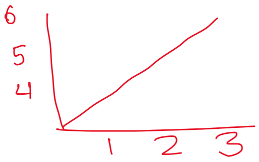
```
 pip install matplotlib
 pip3 install matplotlib
```

Imported as:

```
   from matplotlib import pyplot as plt
```

1. Simple linear plot.

```
from matplotlib import pyplot as plt
  plt.plot([1,2,3], [4,5,6])
 plt.show()
```



2. Bar Graph

Bar graph uses bars to compare data between different categories

Code:

```
from matplotlib import pyplot as plt
 plt.bar([0.5, 1.0, 1.5, 2.0], [10, 30, 60, 90], label="Mango", width=0.5)
  plt.bar [0.5, 1.0, 1.5, 2.0], [60, 30, 10, 40], label="Apple", color='r', width=0.5)
  plt.legend()
  plt.xlabel('season')
  plt.y label('Price')
   plt.tile('Fruit season')
  plt.show()
```

Histogram: A histogram focus on a single entity and show its distribution. The values in the histogram are generally split into intervals to plot the distribution.

Code:

```
  from matplotlib import pyplot as plt
    price_season =[1,2,11,12, 13, 24, 25, 35, 49, 47]
    bins = [0, 10, 20, 30, 40, 50]
     plt.hist(price-season, bins, histtype='bar', rwidth=0.7)
     plt.xlabel('price groups')
     plt.ylabel('price of fruits)
      plt.title('Histogram')
      plt.show()
```

Scatter plot

A scatter us the opportunity to compare the distribution of more than one variable.

```
 import matplotlib.pyplot as plt
     X= [2, 1, 3, 4, 2, 5, 4, 3, 6, 2]
     y= [6, 5, 6.5, 3, 7, 8.5, 9, 3.5, 4.3, 8.1]
     X1 = [8, 2.5, 7, 9.5, 11, 10.2, 13]
     Y1 = [3, 2.4, 3.6, 5, 4.4, 6, 5.3]
   plt.scatter (x, y , label='example 1', color='g')
   plt.scatter (x1, y1, label='Example2', color='b' )
   plt. xlabel ('Example1')
   plt. ylabel ("example2")
   plt. title ('Scatter Plot')
   plt. legend()
   plt.show()
```

Pie chart

A Pie chart gives us a visual insight into what percentage different categories occupy when compared to the overall percentage.

Code:

```
 import matplotlib.pyplot as plt
 breakfast=[1, 2, 3, 4, 5]
 Fruit = [6, 13, 8, 12, 9]
 Cereal = [1, 4, 5, 2, 6]
 Meat = [6, 10, 8, 4, 3]
```

← Not Required

```
Eggs = [3, 7, 2, 6, 12]
slices = [6, 3, 4, 12]
breakfast_options = ['Fruits', 'cereal', 'Meat', 'Eggs']
cols = ['c', 'm', 'r', 'b']
  plt.pie(slices, labels= breakfast_options, colors= cols,
          startangle=90, shadow= True,
          explode=(0, 0.1, 0, 0),
          autopct='%.1f%%')
plt.title('Breakfast Preferences')
plt.show()
```

Seaborn: visual libraray built on the top of matplotlib, used to visualize pattern in data using graphical representation

following command to install the seaborn library

```
pip install seaborn
pip3 install seaborn
```

Generally imported as

```
import seaborn as sns
```

Seaborn - Different plots

```
 import matplotlib.pyplot as plt
 import seaborn as sns
# load the "tips" dataset
  df = sns.load_dataset('tips')
# Create violenplot using seaborn
  sns.violinplot(data=df)
# Show the plot
 plt.show()
```

Seaborn: Scatter Plot

```
import matplotlib.pyplot as plt
import seaborn as sns
# load the "tips" dataset
  df = sns.load_dataset('tips')
# Creating a scatter plot with hue (color) to one of the variables
sns.relplot(x="total_bill", y="tip", hue="smoker", data=df);
plt.show()
```

PyTorch: PyTorch is a library that offers same functionality as numpy but can leverage the GPUs in the system.

Tensors used in PyTorch are similar to the numpy ndarray

Pytorch library needs two installations

```
pip install torch
pip install torchvision
```

Imported using following syntax

```
import torch
```

code

```
  import torch
  # construct a matrix using torch
```

```
# construct a matrix using torch
  matrix = torch.empty(4,2)
  print(matrix)
# construct a random matrix using torch
  matrix_rand = torch.rand(4,2)
  print(matrix_rand)
```

To be done after Teaching basics of Machine learning

Scikit-learn Scikit-learn is a library offers several machine learning algorithm like KNN, Random forest, SVM, XGBoost, and so on and also contains some built-in dataset.

Installed by the following command

```
pip install sklearn
```

Scikit-learn - Datasets You can import existing datasets.

```
from sklearn import datasets
 digits = datasets.load_digits()
  print( digits.data)
```

Scikit-learn - Principle component analysis (PCA) helps you identify the top significant features in your feature list.

```
from sklearn import datasets
from sklearn.decomposition import PCA
digits = datasets.load_digits()
# Randomized PCA performs better when there are more number of dimension
 random-pca = PCA(n-components = 2, svd_solver='randomized')

 rpca-model = random_pca. fit_transform (digits.data)
# comparing with Regular PCA
 pca = PCA(n_components=2)
 pca-model = pca. fit_transform (digits.data)

  print (rpca_model)
  print (pca-model)
```

Scikit-learn- Pre-processing: Preprocessing module offers several functionalities like encoding the data to different formats, spliting the data into training and test sets, and many more

Tensorflow and Keras