18.01.2024

BITS F464: Machine Learning

MACHINE LEARNING FRAMEWORKS

Chittaranjan Hota, Sr. Professor
Dept. of Computer Sc. and Information Systems
hota@hyderabad.bits-pilani.ac.in

Recap

- Do you need ML to calculate Payroll?

ISRO's Chandrayaan-3 mission

23rd Aug, 2023

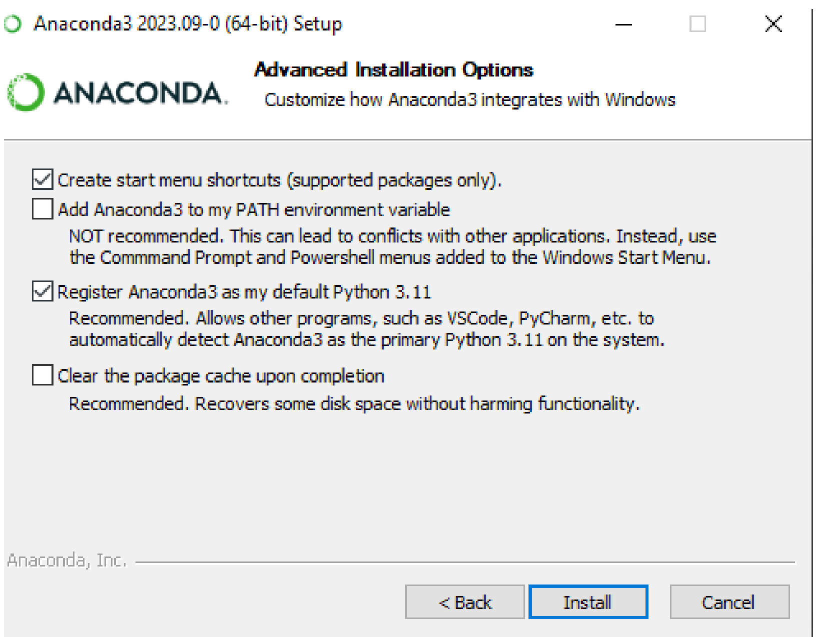
- Machine learning: Any change in a system that allows it to perform better the second time on repetition of the same task or on another task drawn from the same population.

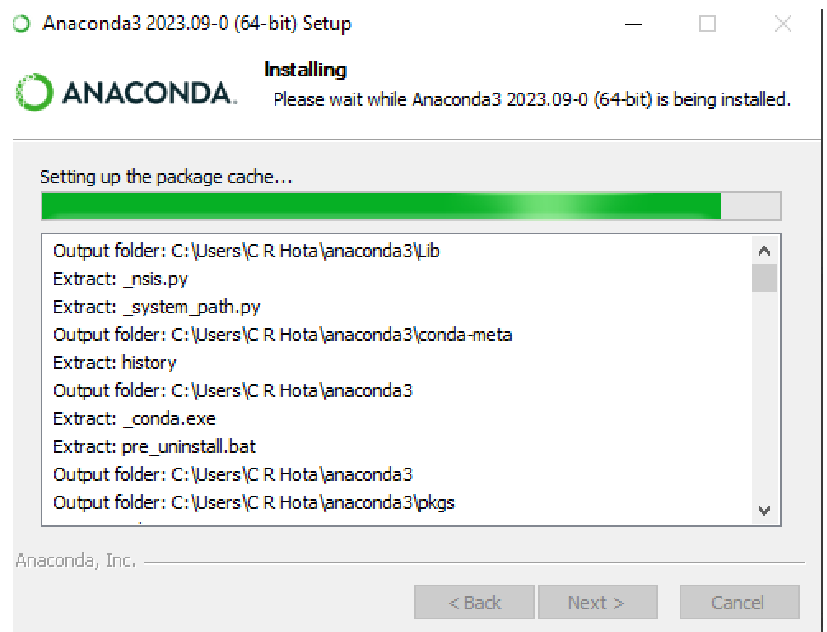
  -- Herbert Simon, 1983

- Clearly, being able to adapt & generalize are key to intelligence.

ML Framework: Anaconda

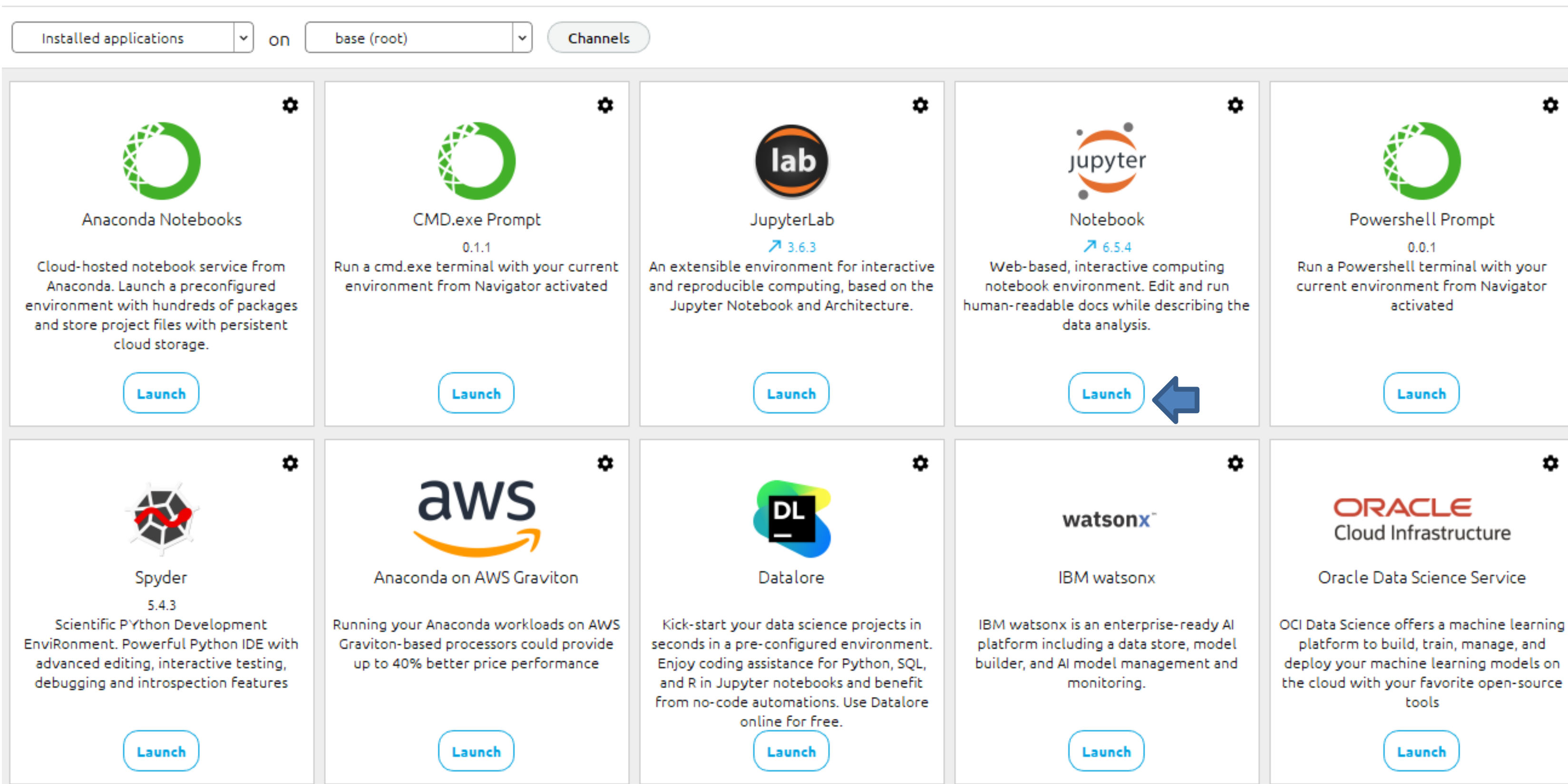
- An ML framework is any tool, interface, or library that lets you develop ML models easily, without understanding the underlying algorithms.
- Anaconda is a distribution of the Python and R programming language for scientific computing suitable for ML.



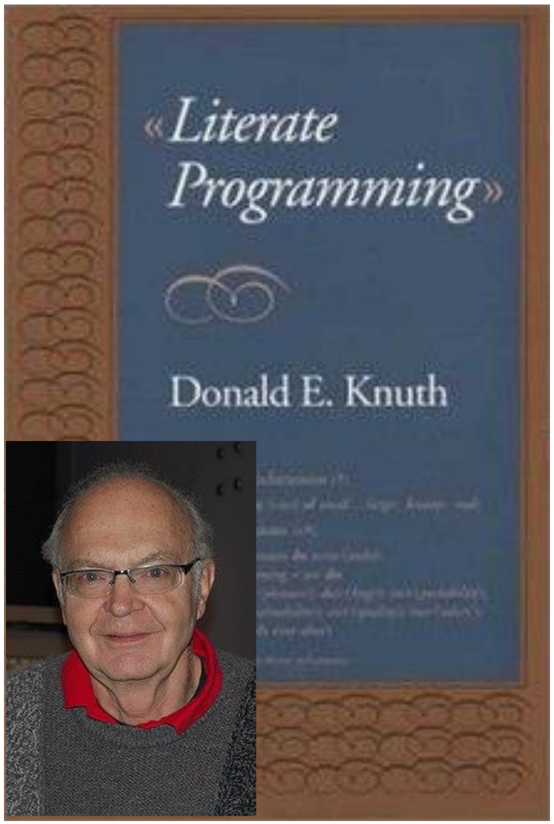


Source: https://www.anaconda.com/definitive-guide-to-ai-platforms

Anaconda Navigator



Notebook: Good Practices of Code Writing



Source: Wiki

- Linear flow of execution.
- Little amount of code.
- Extract reusable code into a package.
- Clean it before storing it in a repository or sharing it with others.
- Develop your code as a story with text, small code fragments and images.

Jupyter Notebook

- An interactive web application for creating and sharing computational documents.

jupyter Untitled Last Checkpoint: 2 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run | Code

In [1]:

```python
import pandas as pd
from sklearn.datasets import load_wine

wine_data = load_wine()

wine_df=pd.DataFrame(wine_data.data,columns=wine_data.feature_names)
wine_df["target"] = wine_data.target

wine_df.head()
```

iris, house prices, diabetes, digits...

Out[1]:

:alinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	target
15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0	0
11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0	0
18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0	0
16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0	0
21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0	0

In [ ]:

Scikit-Learn Example Continued…

```
from sklearn.preprocessing import StandardScaler
x = wine_df[wine_data.feature_names].copy()
y = wine_df["target"].copy()
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X.values)
print(X_scaled[0])
```

```
[ 1.51861254 -0.5622498   0.23205254 -1.16959318  1.91390522  0.80899739
  1.03481896 -0.65956311  1.22488398  0.25171685  0.36217728  1.84791957
  1.01300893]
```

//MinMaxScaler : (0,1)

//Other things possible: Missing val., Redundant val.

Pre-processing

```
from sklearn.model_selection import train_test_split
X_train_scaled, X_test_scaled, y_train, y_test = train_test_split(X_scaled, y, train_size=.7, random_state=25)
from sklearn.linear_model import LogisticRegression
logistic_regression = LogisticRegression()
logistic_regression.fit(X_train_scaled, y_train)
log_reg_preds = logistic_regression.predict(X_test_scaled)
```

Building the model

Source: https://www.datacamp.com/tutorial/machine-learning-python

Classification Reports

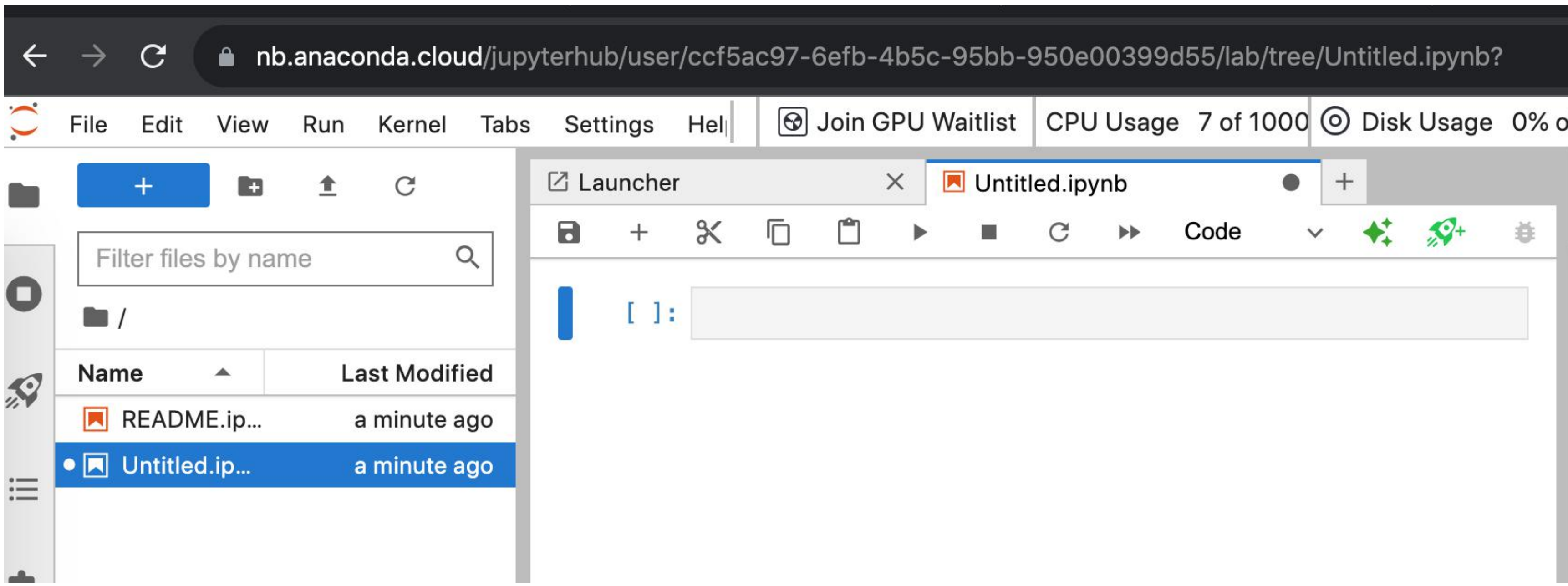
from sklearn.metrics import classification_report

Store model predictions in a dictionary which makes it's easier to iterate through the model and print the results.

```
Logistic Regression Results:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      0.92      0.96        25
           2       0.86      1.00      0.92        12

    accuracy                           0.96        54
   macro avg       0.95      0.97      0.96        54
weighted avg       0.97      0.96      0.96        54
```

Source: https://www.datacamp.com/tutorial/machine-learning-python

Anaconda Cloud Option



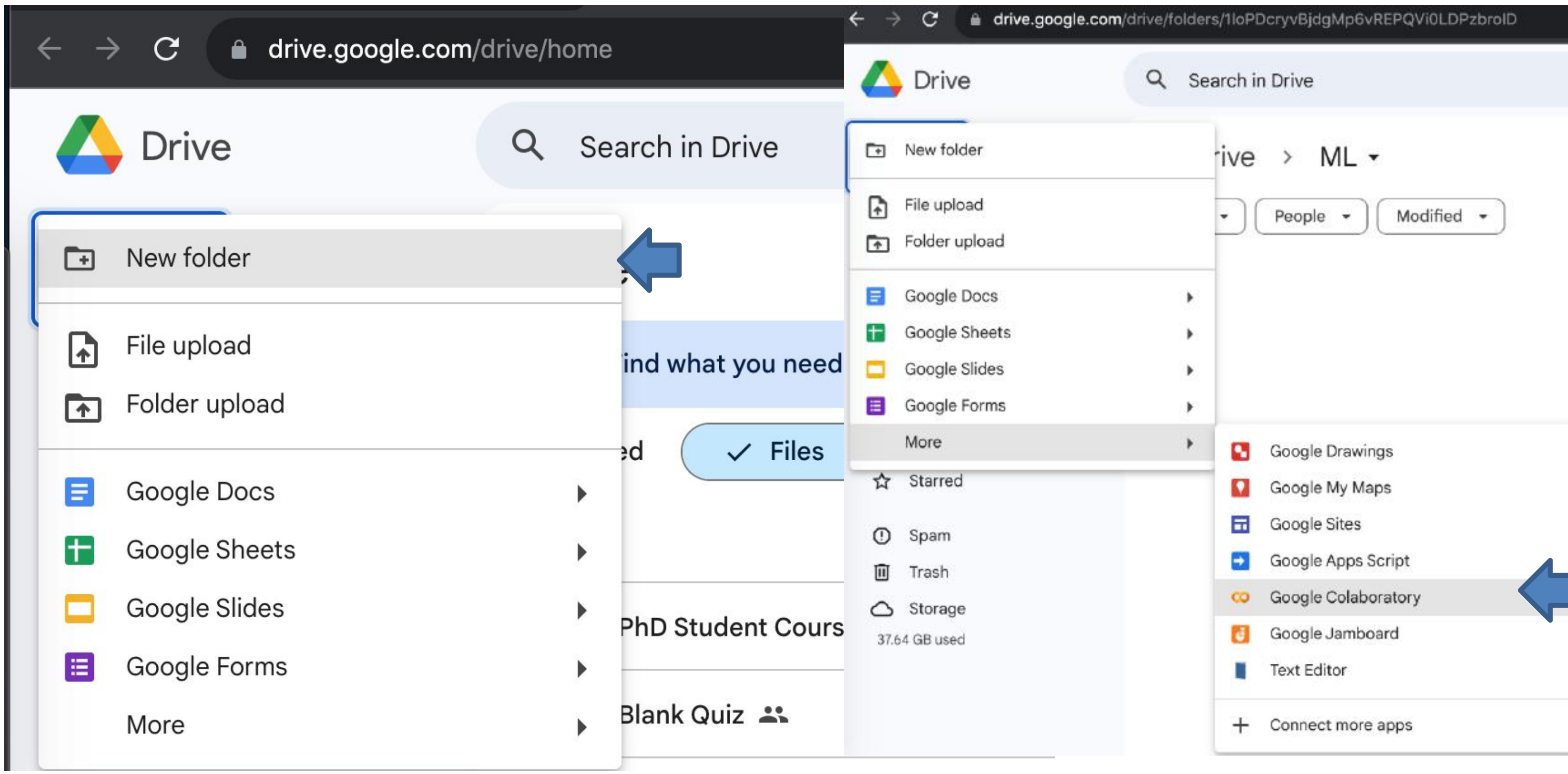
You can install in both the standalone and cloud options, many other packages/ libraries like:

An alternative to Anaconda

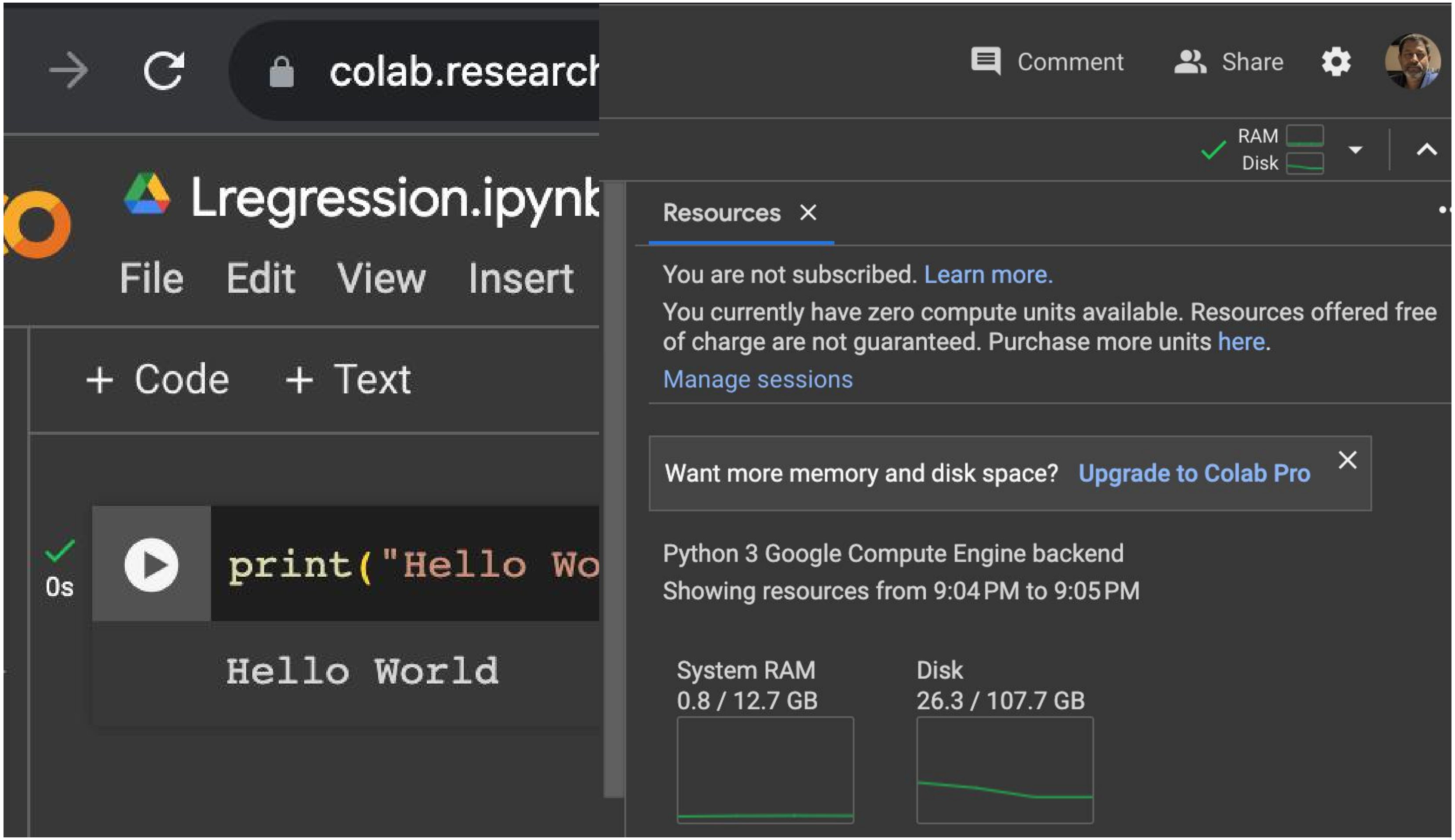
Google Colaboratory

- Sharing is allowed in which one?
- More powerful hardware (TPU/ GPU etc. are available in which one?

Google Colab Continued…



Continued...



colab.research
Comment
Share
RAM
Disk
Lregression.ipynb
File
Edit
View
Insert
+ Code
+ Text
0s
print("Hello Wo
Hello World
Resources
You are not subscribed. Learn more.
You currently have zero compute units available. Resources offered free of charge are not guaranteed. Purchase more units here.
Manage sessions
Want more memory and disk space? Upgrade to Colab Pro
Python 3 Google Compute Engine backend
Showing resources from 9:04 PM to 9:05 PM
System RAM
0.8 / 12.7 GB
Disk
26.3 / 107.7 GB

Continued...

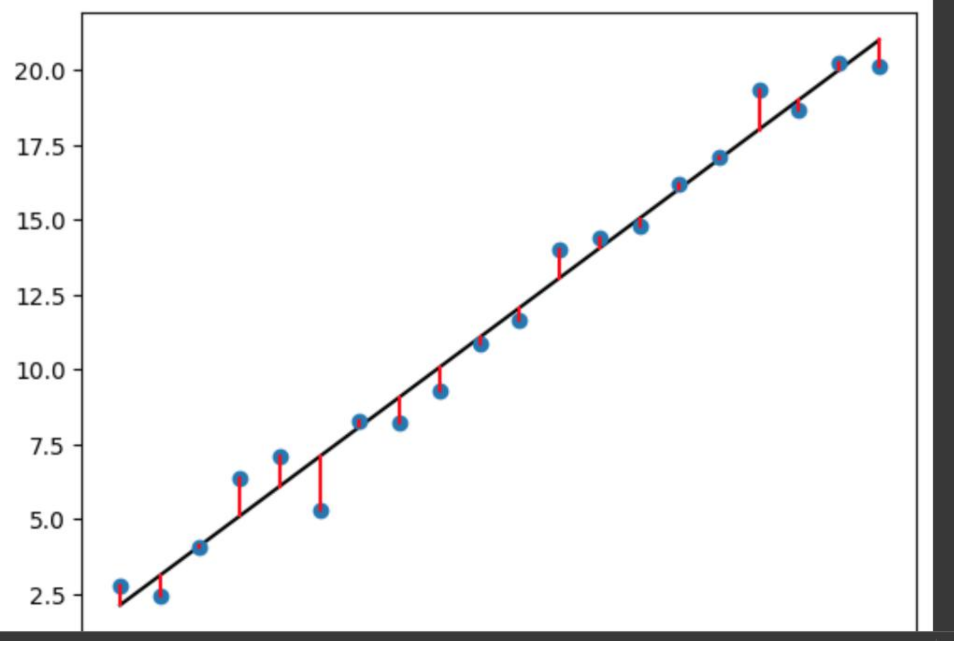
This is the first statement in Co

```
import numpy as np
import matplotlib.pypl
import sklearn

np.random.seed(0)
n=20    # Number of dat
x=np.linspace(0, 10, n
y=x*2 + 1 + 1*np.rando
print(x)
print(y)
```

```
[ 0.          0.526315
  3.15789474  3.684210
  6.31578947  6.842105
  9.47368421 10.
[ 2.76405235  2.452788
  8.26587789  8.217063
 14.39261667 14.805885
 20.26043612 20.145904
```

```
from sklearn.linear_model import LinearRegression

model=LinearRegression(fit_intercept=True)
model.fit(x[:,np.newaxis], y)
xfit=np.linspace(0,10,100)
yfit=model.predict(xfit[:, np.newaxis])
plt.plot(xfit,yfit, color="black")
plt.plot(x,y, 'o')
# The following will draw as many line segments as there are columns in matrices x and y
plt.plot(np.vstack([x,x]), np.vstack([y, model.predict(x[:, np.newaxis])]), color="red");
```

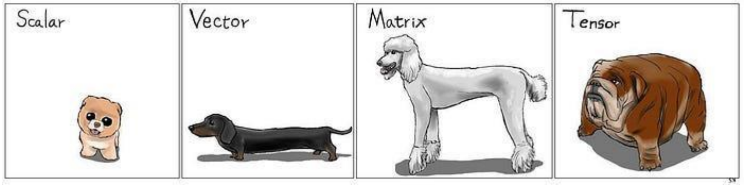
```
[ 0.          0.52631579  1.05263158  1.57894737  2.10526316  2.63157895
  3.15789474  3.68421053  4.21052632  4.73684211  5.26315789  5.78947368
  6.31578947  6.84210526  7.36842105  7.89473684  8.42105263  8.94736842
  9.47368421 10.        ]
[ 2.76405235  2.45278879  4.08400114  6.39878794  7.07808431  5.28588001
  8.26587789  8.21706384  9.31783378 10.88428271 11.67035936 14.03322088
 14.39261667 14.80588554 16.18070534 17.12314801 19.33618434 18.68957858
 20.26043612 20.14590426]
```



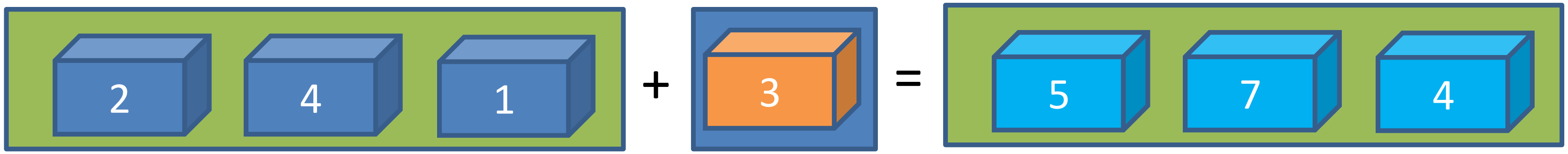
TensorFlow

Google DeepMind

- Supports distributed ML
- Large-scale ML models in real-world applications (Production environment)
- What is a Tensor?



  - A multi-dimensional array on which mathematical operations can be performed. (Ex: Addition of two tensors)



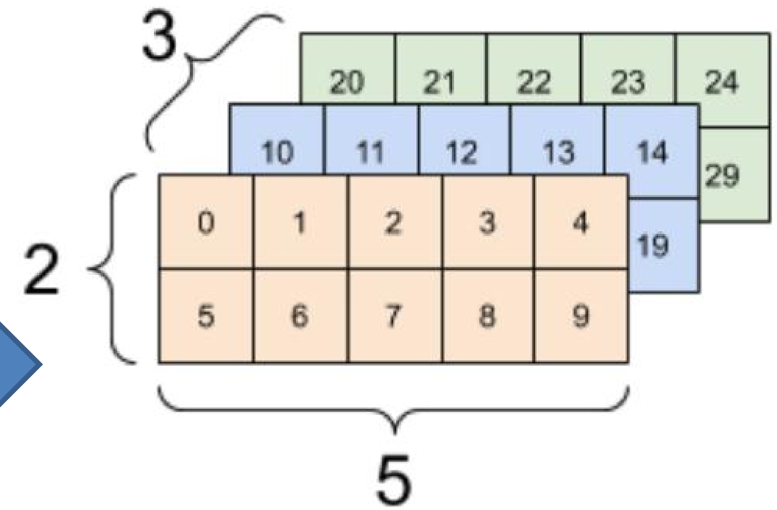
(Rank 1) (Rank 0) (Rank 1)

- GPU acceleration for Tensors

Tensors Continued…

```python
import tensorflow as tf
import numpy as np
rank_0_tensor = tf.constant(4)
print(rank_0_tensor)
rank_1_tensor = tf.constant([2.0, 3.0, 4.0])
print(rank_1_tensor)
rank_2_tensor = tf.constant([[1, 2],
                             [3, 4],
                             [5, 6]], dtype=tf.float16)
print(rank_2_tensor)

rank_3_tensor = tf.constant([
  [[0, 1, 2, 3, 4],
   [5, 6, 7, 8, 9]],
  [[10, 11, 12, 13, 14],
   [15, 16, 17, 18, 19]],
  [[20, 21, 22, 23, 24],
   [25, 26, 27, 28, 29]],])

print(rank_3_tensor)
```

```
tf.Tensor(4, shape=(), dtype=int32)
tf.Tensor([2. 3. 4.], shape=(3,), dtype=float32)
tf.Tensor(
[[1. 2.]
 [3. 4.]
 [5. 6.]], shape=(3, 2), dtype=float16)
tf.Tensor(
[[[ 0  1  2  3  4]
  [ 5  6  7  8  9]]

 [[10 11 12 13 14]
  [15 16 17 18 19]]

 [[20 21 22 23 24]
  [25 26 27 28 29]]], shape=(3, 2, 5), dtype=int32)
```



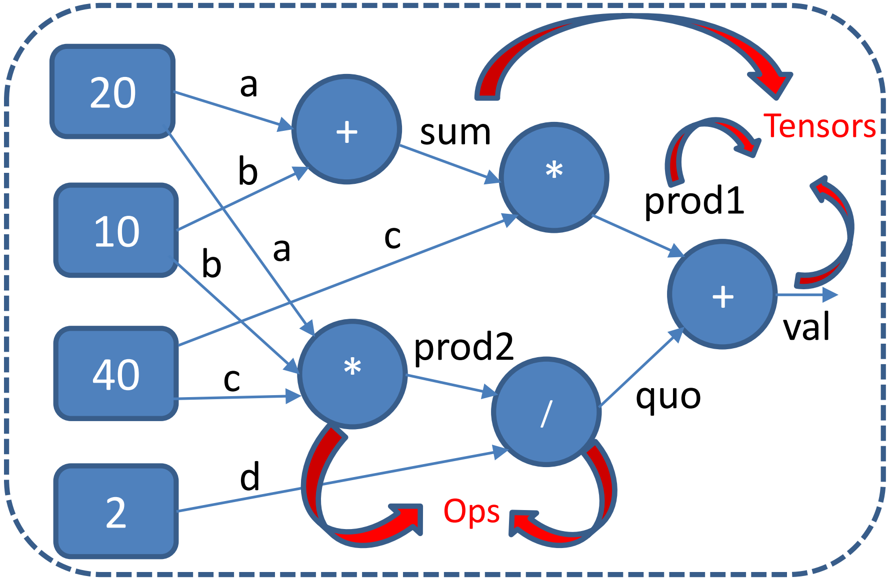
- A Tensor can hold strings too.

```python
tensor_of_strings = tf.constant(["BITS Pilani",
                                 "Hyderabad Campus",
                                 "Telangana"])
print(tensor_of_strings)
```

TensorFlow: Computational Graphs

```
import tensorflow as tf
a = 20
b = 10
c = 4
d = 2
sum = a + b
prod1 = sum * c
prod2 = a* b * c
quo = prod2/d
val = prod1 + quo
print(val)
```

```
520.0
```




```
# Load the TensorBoard notebook extension.
%load_ext tensorboard
import tensorboard
```

For you to explore…

TensorFlow with Keras

```
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
predictions = model(x_train[:1]).numpy()
predictions
tf.nn.softmax(predictions).numpy()
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
loss_fn(y_train[:1], predictions).numpy()
model.compile(optimizer='adam',loss=loss_fn,metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test,  y_test, verbose=2)
```

```
TensorFlow version: 2.15.0
Epoch 1/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.3028 - accuracy: 0.9119
Epoch 2/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1480 - accuracy: 0.9566
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1115 - accuracy: 0.9665
Epoch 4/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0908 - accuracy: 0.9724
Epoch 5/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.0776 - accuracy: 0.9768
313/313 - 1s - loss: 0.0774 - accuracy: 0.9756 - 553ms/epoch - 2ms/step
[0.07737699896097183, 0.975600004196167]
```

Source: https://www.tensorflow.org/tutorials/

```
epoch 0, loss 67.04987335205078
epoch 1, loss 30.550050735473633
epoch 2, loss 14.291292190551758
epoch 3, loss 7.043418884277344
epoch 4, loss 3.807079315185547
epoch 5, loss 2.356700897216797
epoch 6, loss 1.701522707939148
epoch 7, loss 1.400479316711425 8
epoch 8, loss 1.257222652435302 7
epoch 9, loss 1.184340834617614 7
epoch 10, loss 1.1429182291030884
```

...

```
epoch 489, loss 0.0010938026243820786
epoch 490, loss 0.0010780788725242019
epoch 491, loss 0.001062584575265646
epoch 492, loss 0.0010473171714693308
epoch 493, loss 0.0010322668822482228
epoch 494, loss 0.0010174255585297942
epoch 495, loss 0.001002818695269525
epoch 496, loss 0.0009883942548185587
epoch 497, loss 0.0009741996182128787
epoch 498, loss 0.0009601832716725767
epoch 499, loss 0.0009463919559493661
predict (after training) 4 7.964635848999023
```

```python
import torch
from torch.autograd import Variable

x_data = Variable(torch.Tensor([[1.0], [2.0], [3.0]]))
y_data = Variable(torch.Tensor([[2.0], [4.0], [6.0]]))

class LinearRegressionModel(torch.nn.Module):

  def __init__(self):
    super(LinearRegressionModel, self).__init__()
    self.linear = torch.nn.Linear(1, 1) # One in and one out
  def forward(self, x):
    y_pred = self.linear(x)
    return y_pred

our_model = LinearRegressionModel()

criterion = torch.nn.MSELoss(size_average = False)
optimizer = torch.optim.SGD(our_model.parameters(), lr = 0.01)

for epoch in range(500):
  # Forward pass: Compute predicted y by passing x to the model
  pred_y = our_model(x_data)

  # Compute and print loss
  loss = criterion(pred_y, y_data)

  # Zero gradients, perform a backward pass,and update the weights.
  optimizer.zero_grad()
  loss.backward()
  optimizer.step()
  print('epoch {}, loss {}'.format(epoch, loss.item()))

new_var = Variable(torch.Tensor([[4.0]]))
pred_y = our_model(new_var)
print("predict (after training)", 4, our_model(new_var).item())
```

Source: https://www.geeksforgeeks.org/

Thank you!