PRML05

April 15, 2022

1 Report

We were given mean of A and B.

1.1 Part-A

We were given covariance matrix as well which denotes that A and B are not co-related because covariance between them are zero. To calculate the decision boundry, we'll use the following formulas -

x0 = 1/2*(i + j) − (2/|| i − j||2)* (ln P(i)/P(j)) * (i − j)

w = i − j

G1(x) - G2(x) = wt*(x − x0) = 0

where x in our case is consisted of variables 'x' and 'y'.

We'll use CASE-I because features are statistically independent and each feature has the same variance. Decision boundry is coming out to be overlapping on Y-axis. features are independent and spread is also equally distributed.

1.2 Part-B

We were given covariance matrix which denotes that A and B are not co-related because covariance between them are zero but spread of all the variables are different.

We'll use CASE-II because the covariance matrices for all of the classes are identical but otherwise arbitrary. Decision boundry is coming out to be overlapping on Y-axis. features are independent but the spread is not equally distributed.

1.3 Part-C

We'll use CASE-III here because the covariance matrices are arbitrary. Decision boundry is not overlapping with Y-axis in this case instead it's a line with positive slope with respect to x-axis.

2 Importing libraries and storing common data

[ ]:
```
!pip3 install sympy
```

```
Collecting sympy
  Downloading sympy-1.10.1-py3-none-any.whl (6.4 MB)
     |                    | 6.4 MB 3.5 MB/s eta 0:00:01
|                      | 4.8 MB 743 kB/s eta 0:00:03
Collecting mpmath>=0.19
  Downloading mpmath-1.2.1-py3-none-any.whl (532 kB)
     |                    | 532 kB 5.6 MB/s eta 0:00:01
Installing collected packages: mpmath, sympy
Successfully installed mpmath-1.2.1 sympy-1.10.1
```

[ ]:
```
import numpy as np
import math
```

[ ]:
```
mean1 = np.array([-1,1])
mean2 = np.array([1,1])
```

3 Part-A

[ ]:
```
sigma1 = np.array([[0.6, 0], [0, 0.6]])
sigma2 = np.array([[0.6, 0], [0, 0.6]])
```

3.0.1 Generating Normally distributed dataset based and mean and sigma

[ ]:
```
x1, y1 = np.random.multivariate_normal(mean1, sigma1, 2000).T
```

[ ]:
```
x2, y2 = np.random.multivariate_normal(mean2, sigma2, 2000).T
```

[ ]:
```
x0 = (mean1+mean2)/2
x0
```

[ ]:
```
array([0., 1.])
```

[ ]:
```
w = (mean1-mean2)
w
```

[ ]:
```
array([-2,  0])
```

3.0.2 calculating the equation of G1(x) - G2(x)

[ ]:
```
import sympy as sym

arr = list()
x = sym.Symbol('x')
y = sym.Symbol('y')
arr.append(x)
arr.append(y)

arr = np.array(arr)

equation = w.T @ (arr-x0)
equation
```

[ ]: $-2x$

3.0.3 Observation

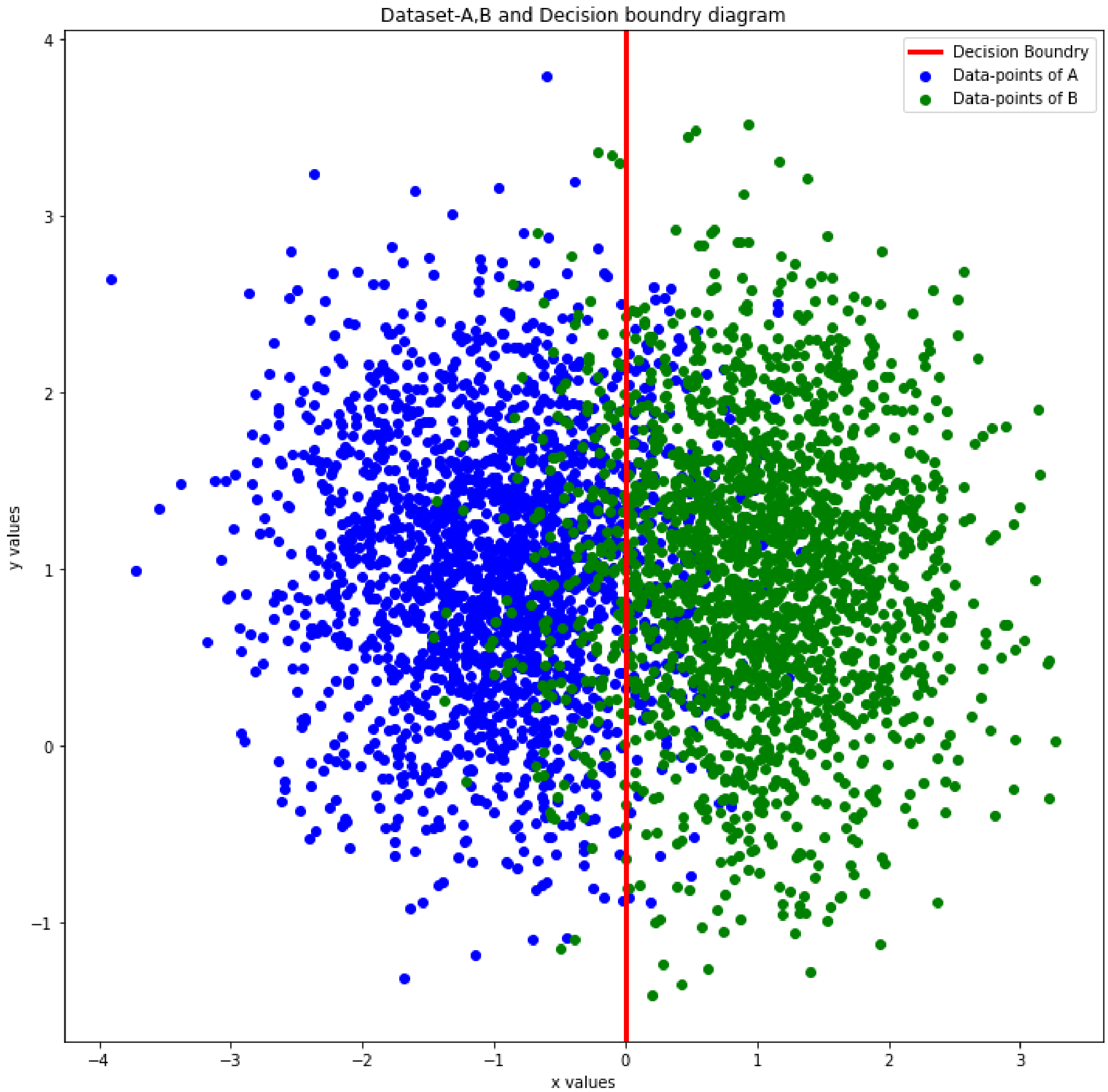
We have calculated the equation for { G1(x)-G2(x)=0 }. The final equation comes out to be overlapping with Y-axis. So, we will plot a line along Y-axis.

3.0.4 Dataset and Decision boundry diagram

[ ]:
```
####### for plotting the Decision boundry

from matplotlib import pyplot as plt

plt.figure(figsize = (12,12))
plt.scatter(x1, y1, color = 'b', label = 'Data-points of A')
plt.scatter(x2, y2, color = 'g', label = 'Data-points of B')
plt.axvline(color = 'r', lw = 3, label = 'Decision Boundry')
plt.title('Dataset-A,B and Decision boundry diagram')
plt.xlabel('x values')
plt.ylabel('y values')
plt.legend()
plt.show()
```



4 Part-B

[ ]:
```
sigma1 = np.array([[0.7, 0], [0, 0.3]])
sigma2 = np.array([[0.7, 0], [0, 0.3]])
```

4.0.1 Generating Normally distributed dataset based and mean and sigma

[ ]:
```
x1, y1 = np.random.multivariate_normal(mean1, sigma1, 2000).T
```

[ ]:
```
x2, y2 = np.random.multivariate_normal(mean2, sigma2, 2000).T
```

[ ]:
```
x0 = (mean1+mean2)/2
x0
```

[ ]:
```
array([0., 1.])
```

[ ]:
```
w = np.linalg.inv(sigma1) @ (mean1-mean2)
w
```

[ ]:
```
array([-2.85714286,  0.        ])
```

4.0.2 calculating the equation of G1(x) - G2(x)

[ ]:
```
import sympy as sym

arr = list()
x = sym.Symbol('x')
y = sym.Symbol('y')
arr.append(x)
arr.append(y)

arr = np.array(arr)

equation = w.T @ (arr-x0)
equation
```

[ ]:
$-2.85714285714286x$

4.0.3 Observation

We have calculated the equation for { G1(x)-G2(x)=0 }. The final equation comes out to be overlapping with Y-axis. So, we will plot a line along Y-axis.

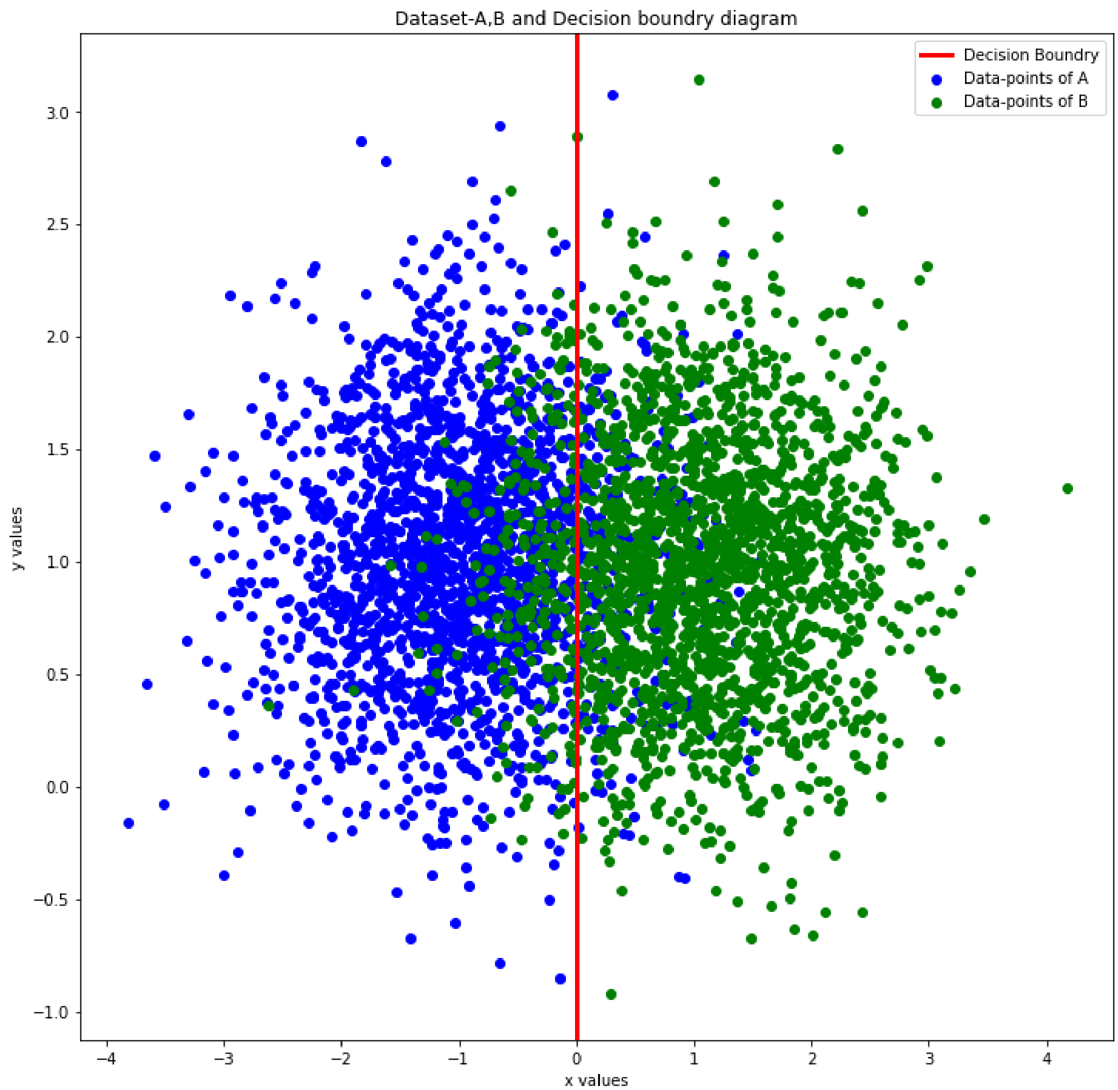
4.0.4 Dataset and Decision boundry diagram

[ ]:
```
####### for plotting the Decision boundry

from matplotlib import pyplot as plt

plt.figure(figsize = (12,12))
plt.scatter(x1, y1, color = 'b', label = 'Data-points of A')
plt.scatter(x2, y2, color = 'g', label = 'Data-points of B')
plt.axvline(color = 'r', lw = 3, label = 'Decision Boundry')
plt.title('Dataset-A,B and Decision boundry diagram')
plt.xlabel('x values')
```

```
plt.ylabel('y values')
plt.legend()
plt.show()
```



5 Part-C

[ ]:
```
sigma1 = np.array([[0.6, 0.25], [0.25, 0.4]])
sigma2 = np.array([[0.6, 0.25], [0.25, 0.4]])
```

5.0.1 Generating Normally distributed dataset based and mean and sigma

[ ]:
```
x1, y1 = np.random.multivariate_normal(mean1, sigma1, 2000).T
```

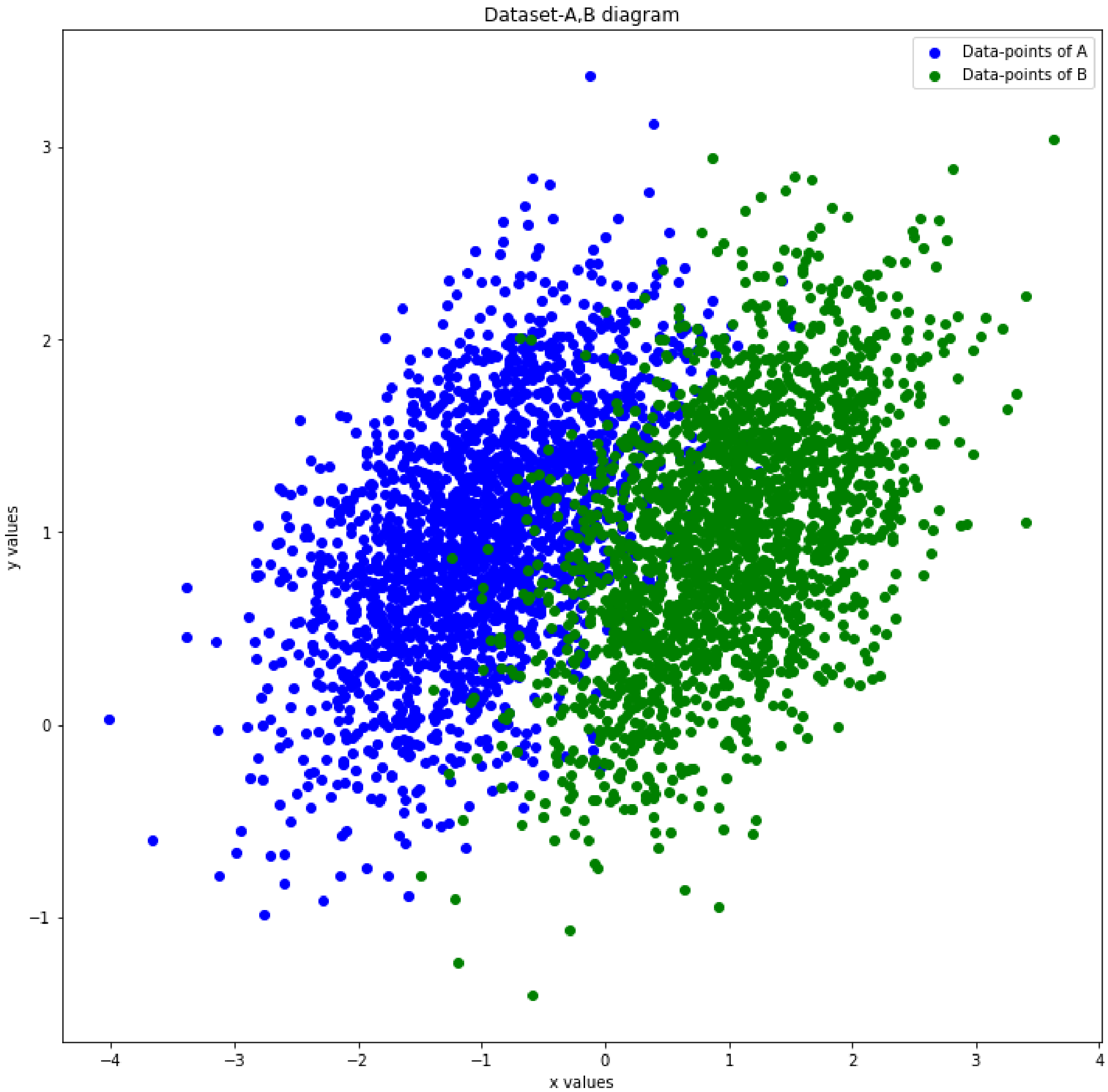
[ ]:
```
x2, y2 = np.random.multivariate_normal(mean2, sigma2, 2000).T
```

5.0.2 Plotting only DataPoints

[ ]:
```
####### for plotting the data-points only

from matplotlib import pyplot as plt

plt.figure(figsize = (12,12))
plt.scatter(x1, y1, color = 'b', label = 'Data-points of A')
plt.scatter(x2, y2, color = 'g', label = 'Data-points of B')
plt.title('Dataset-A,B diagram')
plt.xlabel('x values')
plt.ylabel('y values')
plt.legend()
plt.show()
```



5.0.3 Calculating all the variables involved for finding the equation for G1(x)-G2(x)

[ ]:
```
w10 = -0.5 * ((mean1.T)@(np.linalg.inv(sigma1))@mean1)
w10
```

[ ]: -4.225352112676056

[ ]:
```
w20 = -0.5 * ((mean2.T)@(np.linalg.inv(sigma2))@mean2)
w20
```

[ ]: -1.408450704225352

```
[ ]: w1 = np.linalg.inv(sigma1)@mean1
     w1
```

```
[ ]: array([-3.66197183,  4.78873239])
```

```
[ ]: w2 = np.linalg.inv(sigma2)@mean2
     w2
```

```
[ ]: array([0.84507042, 1.97183099])
```

```
[ ]: W1 = -0.5 * np.linalg.inv(sigma1)
     W1
```

```
[ ]: array([[-1.12676056,  0.70422535],
            [ 0.70422535, -1.69014085]])
```

```
[ ]: W2 = -0.5 * np.linalg.inv(sigma2)
     W2
```

```
[ ]: array([[-1.12676056,  0.70422535],
            [ 0.70422535, -1.69014085]])
```

5.0.4 calculating the equation of G1(x) - G2(x)

```
[ ]: import sympy as sym

     arr = list()
     x = sym.Symbol('x')
     y = sym.Symbol('y')
     arr.append(x)
     arr.append(y)

     arr = np.array(arr)
     arr
```

```
[ ]: array([x, y], dtype=object)
```

```
[ ]: equation = arr.T@(W1 - W2)@arr + (w1.T - w2.T)@arr + (w10 - w20)
     equation
```

[ ]:

$$-4.50704225352113x + 2.8169014084507y - 2.8169014084507$$

5.0.5 Observation

Because of same sigma values for both the cases, decision boundry will be linear. If sigma for both A and B were different then the resultant decision boundry would have been non-linear.

5.0.6 Dataset and Decision boundry diagram

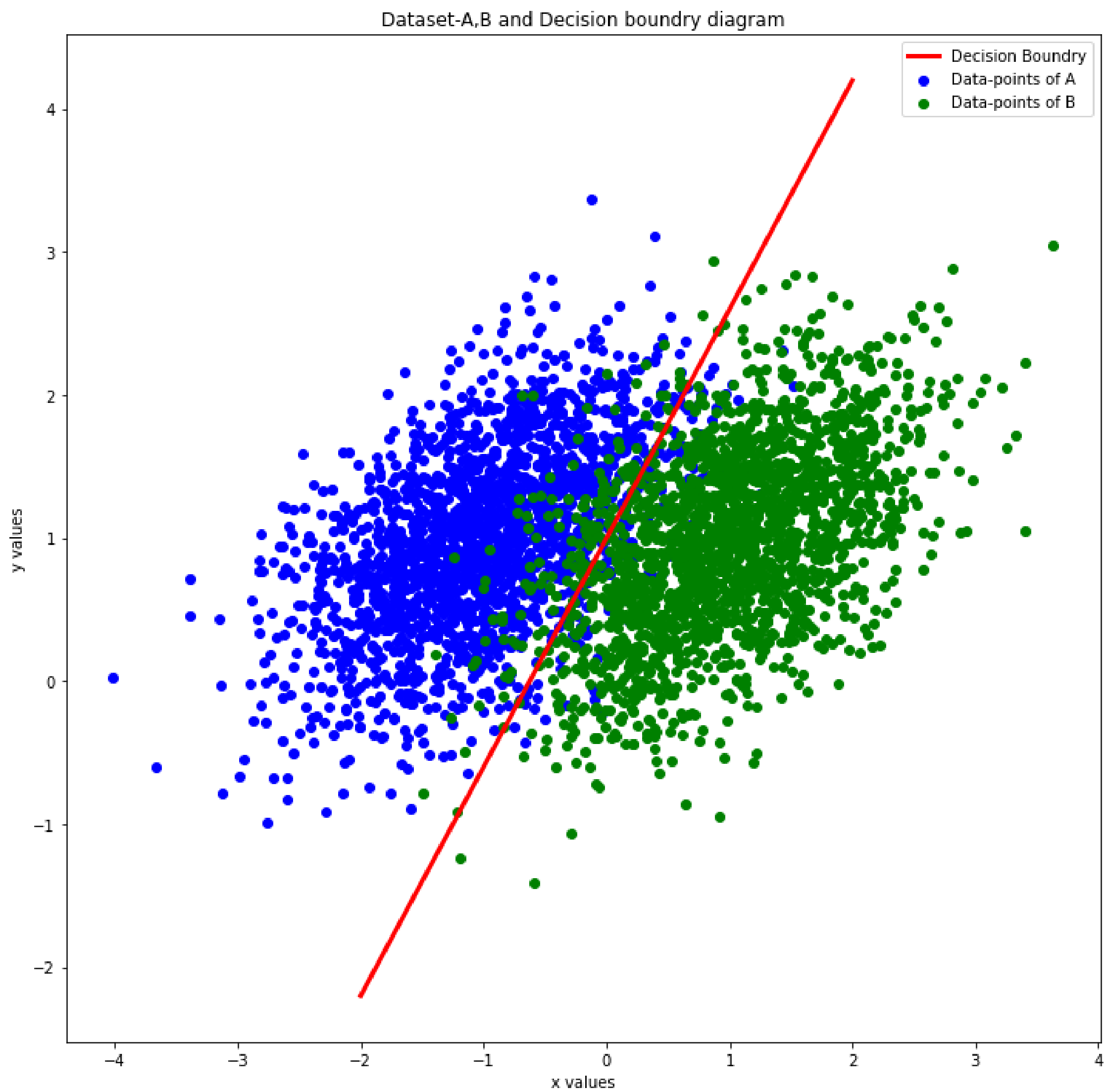
[ ]:
```
def evaluate(x):
    y = (4.50704225352113 * plotting_x + 2.8169014084507)/2.8169014084507
    return y
```

[ ]:
```
####### for plotting the Decision boundry

from matplotlib import pyplot as plt

plotting_x = np.linspace(-2, 2, num=3000)
plotting_y = evaluate(plotting_x)

plt.figure(figsize = (12,12))
plt.scatter(x1, y1, color = 'b', label = 'Data-points of A')
plt.scatter(x2, y2, color = 'g', label = 'Data-points of B')
plt.plot(plotting_x, plotting_y, color = 'r', lw = 3, label = 'Decision␣
 ↪Boundry')
plt.title('Dataset-A,B and Decision boundry diagram')
plt.xlabel('x values')
plt.ylabel('y values')
plt.legend()
plt.show()
```



6 Thank You!

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('PRML05.ipynb')
```

```
--2022-04-15 17:01:16--  https://raw.githubusercontent.com/brpy/colab-
```

```
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: ‘colab_pdf.py’

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2022-04-15 17:01:17 (28.8 MB/s) - ‘colab_pdf.py’ saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```

[ ]: