# DL Assignment 3

**Shah Kaushal Kamleshkumar**

**Enrollment no - 211370680003**

**DEEP LEARNING - 1628006**

**Post Graduate Diploma in Data Science**

**Graduate School of Engineering and Technology**

**Semester 2**

## 1 Count the number of objects in given image using opencv

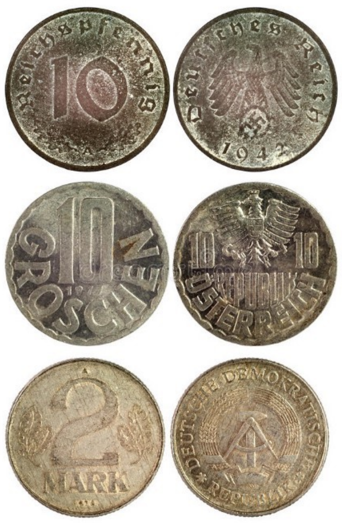
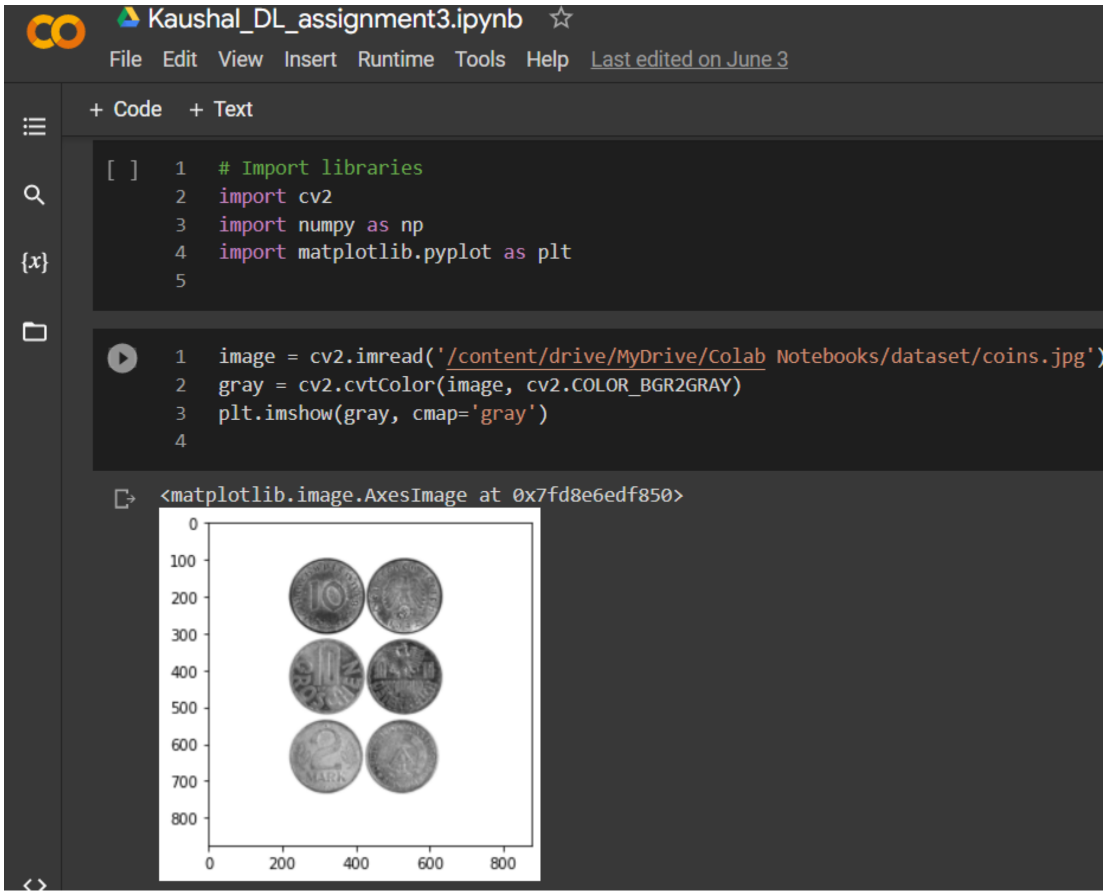


Figure 1: Input coins image for counting operation

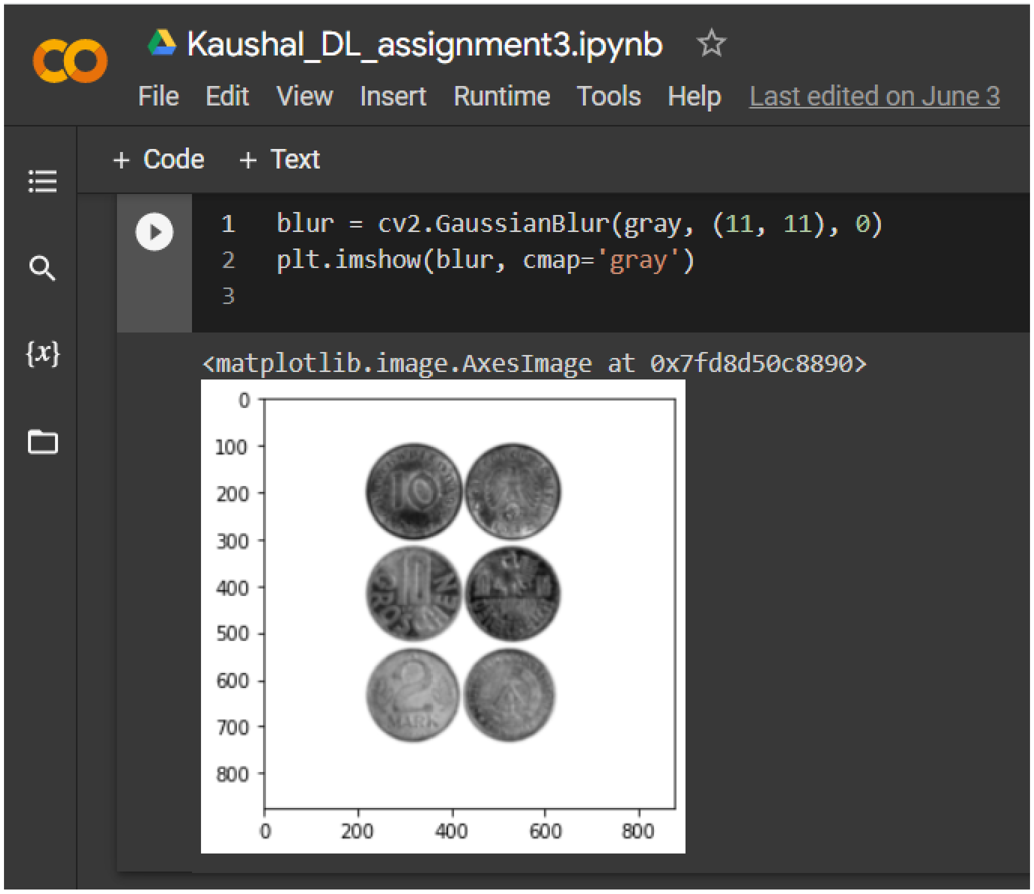

Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 3
+ Code + Text

```
# Import libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
image = cv2.imread('/content/drive/MyDrive/Colab Notebooks/dataset/coins.jpg'
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7fd8e6edf850>
```

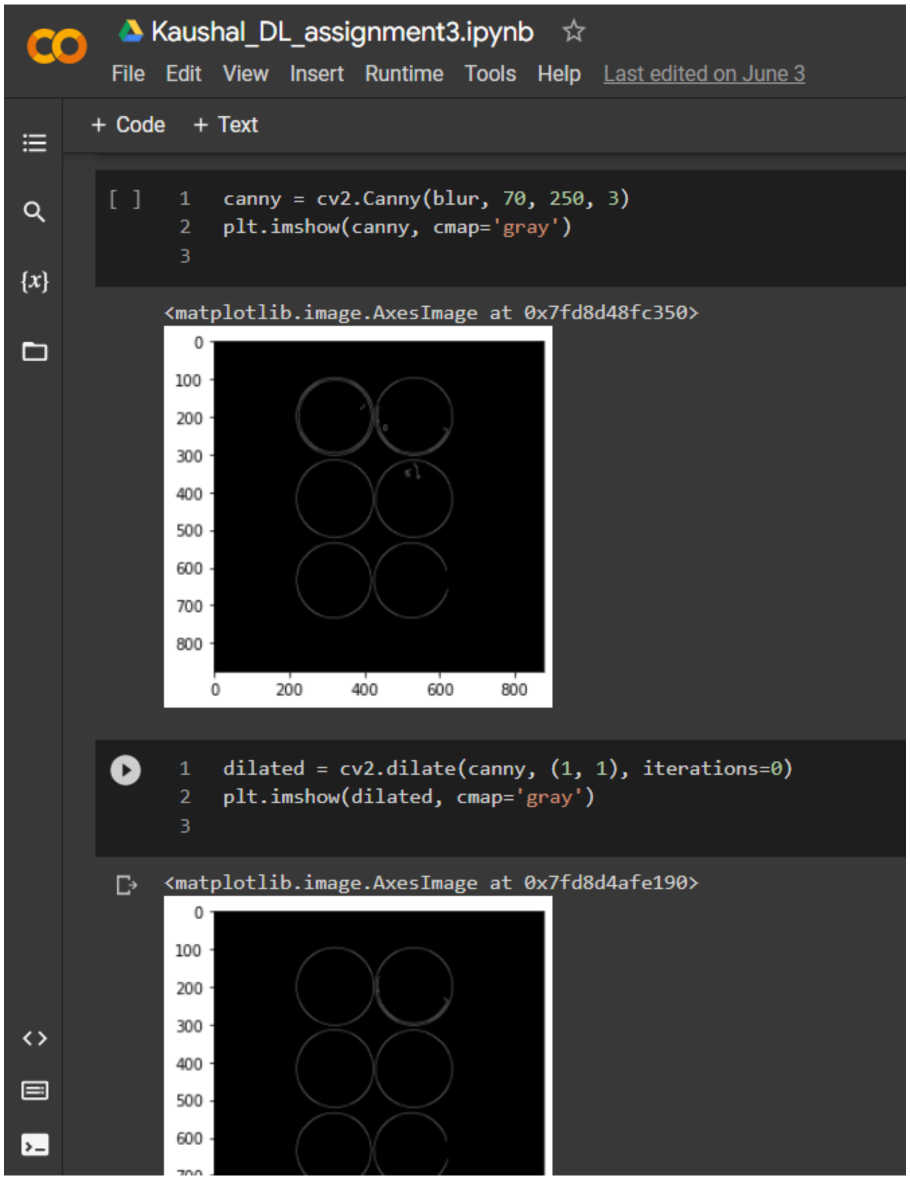
Figure 2: Importing OpenCV library and converting image into gray scale


Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 3
+ Code + Text

```python
blur = cv2.GaussianBlur(gray, (11, 11), 0)
plt.imshow(blur, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7fd8d50c8890>
```

Figure 3: Implementation of gausian filter operation using OpenCV


Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 3
+ Code + Text

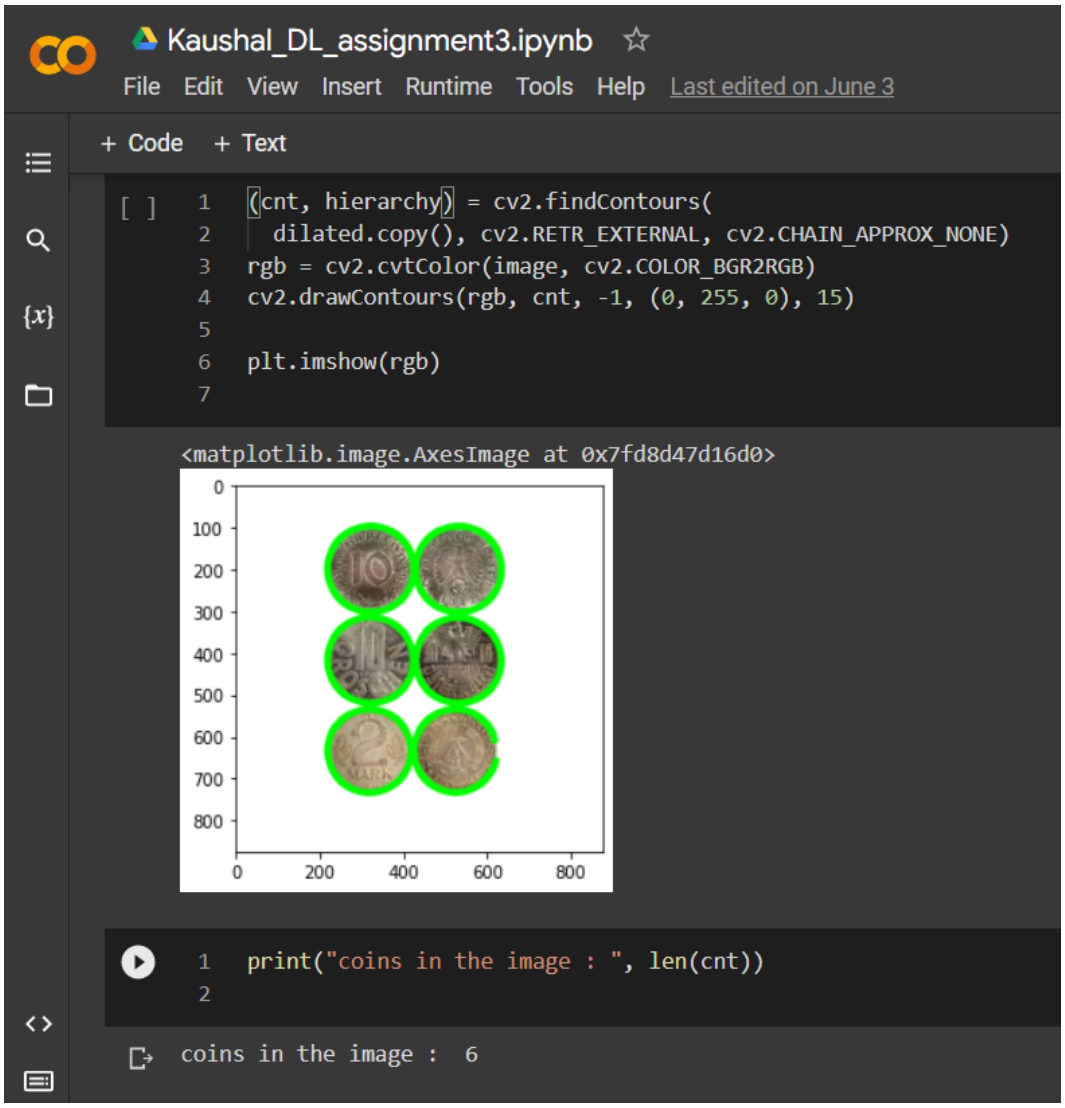
```
canny = cv2.Canny(blur, 70, 250, 3)
plt.imshow(canny, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7fd8d48fc350>
```

```
dilated = cv2.dilate(canny, (1, 1), iterations=0)
plt.imshow(dilated, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7fd8d4afe190>
```

Figure 4: Implementation of canny edge detection and morphological operation using OpenCV


Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 3
+ Code + Text

```
(cnt, hierarchy) = cv2.findContours(
  dilated.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
cv2.drawContours(rgb, cnt, -1, (0, 255, 0), 15)

plt.imshow(rgb)

```

```
<matplotlib.image.AxesImage at 0x7fd8d47d16d0>
```

```
print("coins in the image : ", len(cnt))

```

```
coins in the image :  6
```

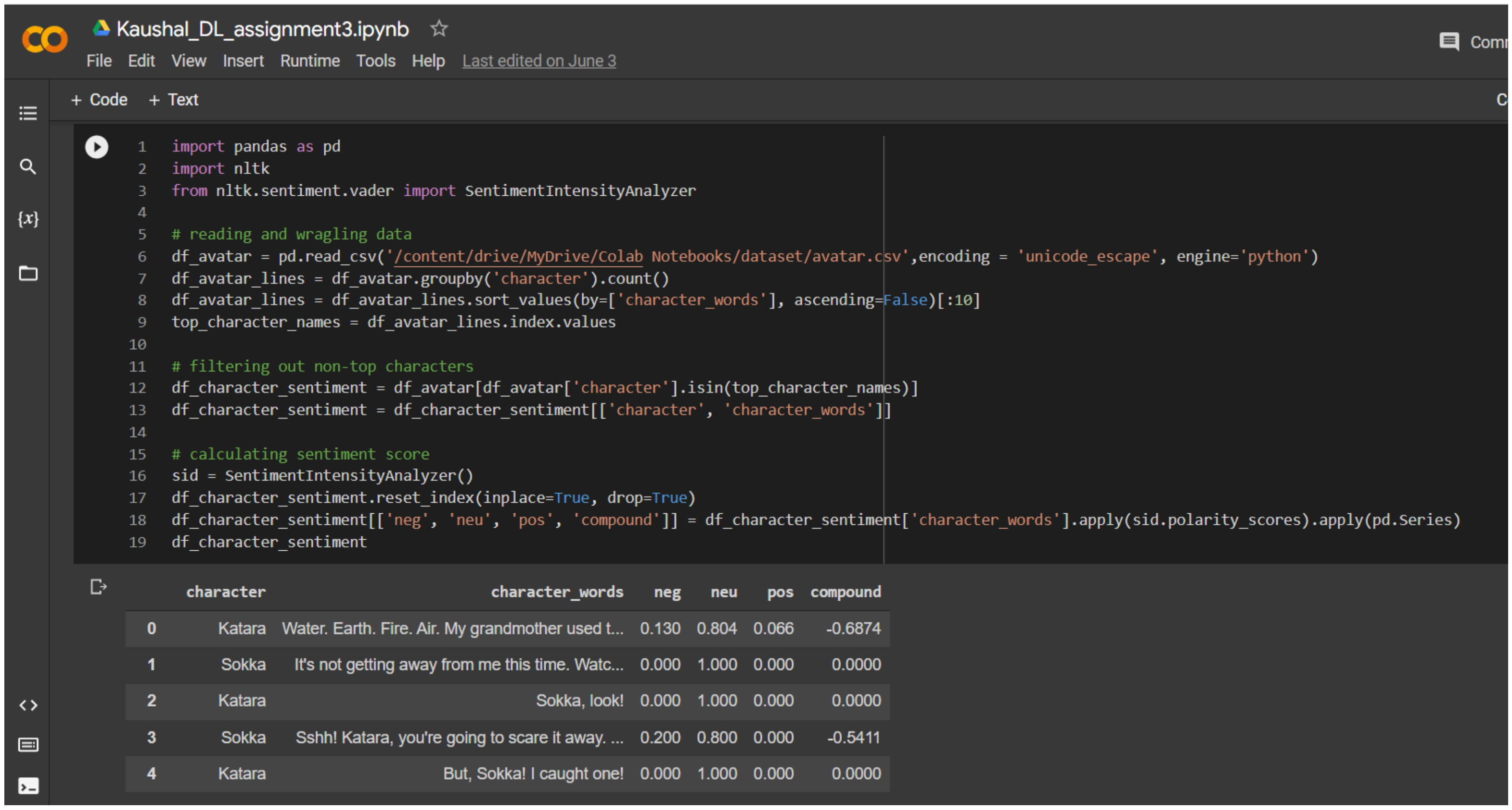
Figure 5: Implementation of contour detection operation using OpenCV for counting number of coins

**Output - As we can see in the above image Number of objects in the input image are 6**

## 2 Perform the below mentioned NLP techniques

- **Sentiment analysis**
- **NER**
- **Stemming**
- **Lemmatization**
- **BoW**
- **TF-IDF**
- **Word cloud**

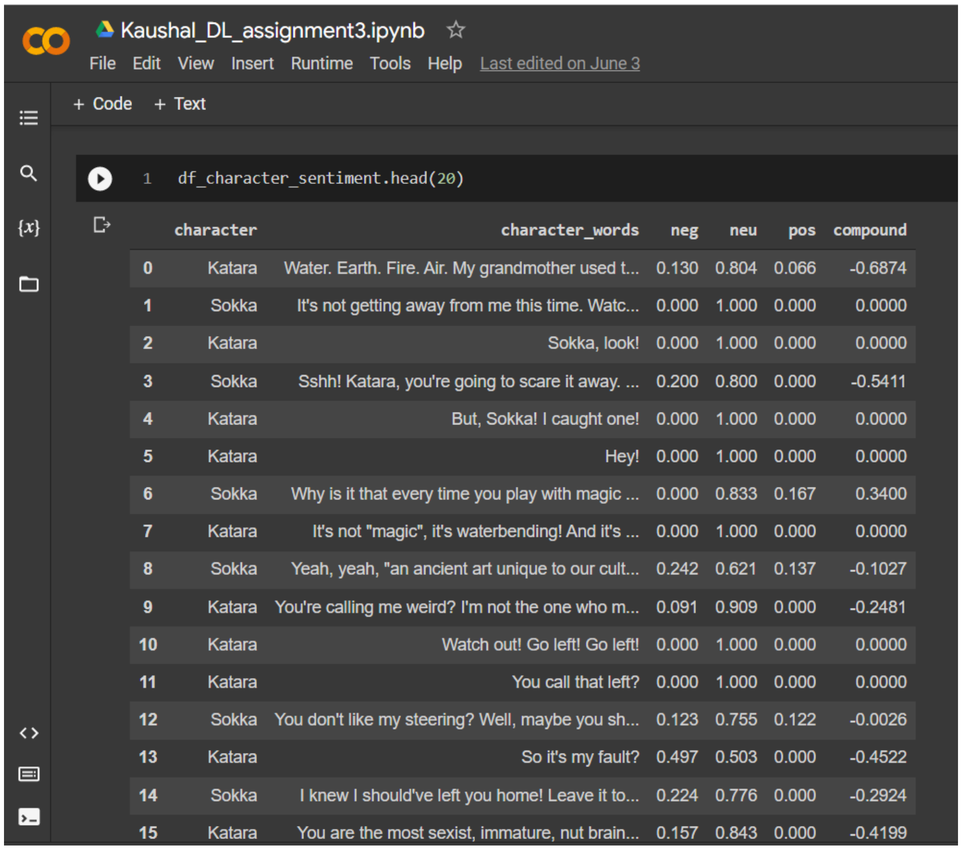
### 2.1 Sentiment analysis


Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 3
+ Code + Text

```python
import pandas as pd
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# reading and wragling data
df_avatar = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/dataset/avatar.csv',encoding = 'unicode_escape', engine='python')
df_avatar_lines = df_avatar.groupby('character').count()
df_avatar_lines = df_avatar_lines.sort_values(by=['character_words'], ascending=False)[:10]
top_character_names = df_avatar_lines.index.values

# filtering out non-top characters
df_character_sentiment = df_avatar[df_avatar['character'].isin(top_character_names)]
df_character_sentiment = df_character_sentiment[['character', 'character_words']]

# calculating sentiment score
sid = SentimentIntensityAnalyzer()
df_character_sentiment.reset_index(inplace=True, drop=True)
df_character_sentiment[['neg', 'neu', 'pos', 'compound']] = df_character_sentiment['character_words'].apply(sid.polarity_scores).apply(pd.Series)
df_character_sentiment
```

| | character | character_words | neg | neu | pos | compound |
|---|---|---|---|---|---|---|
| 0 | Katara | Water. Earth. Fire. Air. My grandmother used t... | 0.130 | 0.804 | 0.066 | -0.6874 |
| 1 | Sokka | It's not getting away from me this time. Watc... | 0.000 | 1.000 | 0.000 | 0.0000 |
| 2 | Katara | Sokka, look! | 0.000 | 1.000 | 0.000 | 0.0000 |
| 3 | Sokka | Sshh! Katara, you're going to scare it away. ... | 0.200 | 0.800 | 0.000 | -0.5411 |
| 4 | Katara | But, Sokka! I caught one! | 0.000 | 1.000 | 0.000 | 0.0000 |

Figure 6: Sentiment analysis implementation code

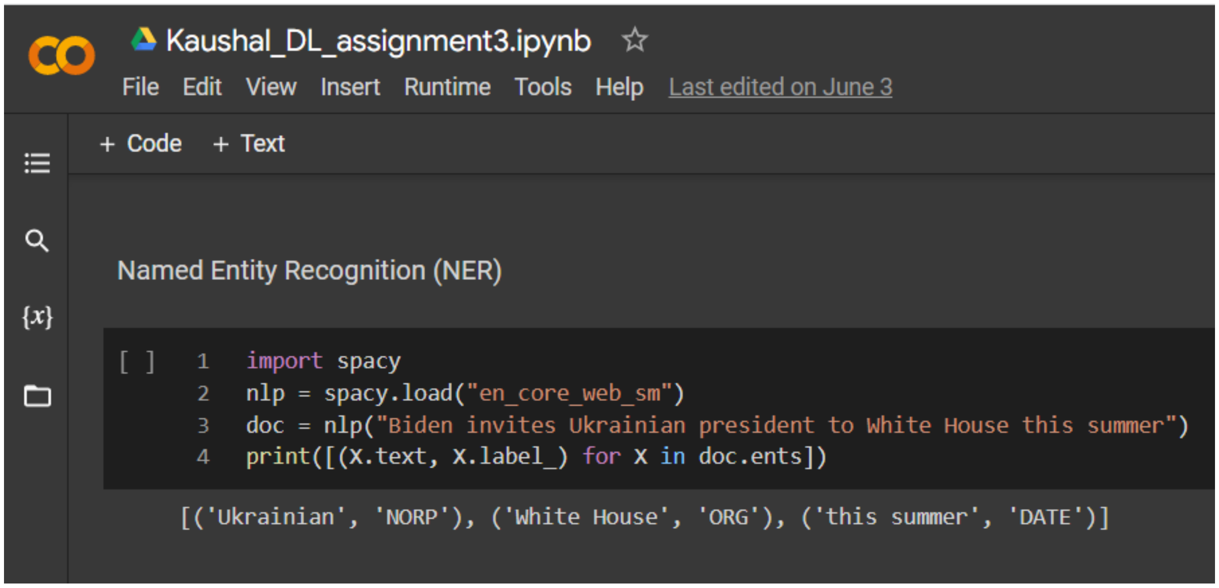

Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 3
+ Code + Text

```
df_character_sentiment.head(20)
```

| | character | character_words | neg | neu | pos | compound |
|---|---|---|---|---|---|---|
| 0 | Katara | Water. Earth. Fire. Air. My grandmother used t... | 0.130 | 0.804 | 0.066 | -0.6874 |
| 1 | Sokka | It's not getting away from me this time. Watc... | 0.000 | 1.000 | 0.000 | 0.0000 |
| 2 | Katara | Sokka, look! | 0.000 | 1.000 | 0.000 | 0.0000 |
| 3 | Sokka | Sshh! Katara, you're going to scare it away. ... | 0.200 | 0.800 | 0.000 | -0.5411 |
| 4 | Katara | But, Sokka! I caught one! | 0.000 | 1.000 | 0.000 | 0.0000 |
| 5 | Katara | Hey! | 0.000 | 1.000 | 0.000 | 0.0000 |
| 6 | Sokka | Why is it that every time you play with magic ... | 0.000 | 0.833 | 0.167 | 0.3400 |
| 7 | Katara | It's not "magic", it's waterbending! And it's ... | 0.000 | 1.000 | 0.000 | 0.0000 |
| 8 | Sokka | Yeah, yeah, "an ancient art unique to our cult... | 0.242 | 0.621 | 0.137 | -0.1027 |
| 9 | Katara | You're calling me weird? I'm not the one who m... | 0.091 | 0.909 | 0.000 | -0.2481 |
| 10 | Katara | Watch out! Go left! Go left! | 0.000 | 1.000 | 0.000 | 0.0000 |
| 11 | Katara | You call that left? | 0.000 | 1.000 | 0.000 | 0.0000 |
| 12 | Sokka | You don't like my steering? Well, maybe you sh... | 0.123 | 0.755 | 0.122 | -0.0026 |
| 13 | Katara | So it's my fault? | 0.497 | 0.503 | 0.000 | -0.4522 |
| 14 | Sokka | I knew I should've left you home! Leave it to... | 0.224 | 0.776 | 0.000 | -0.2924 |
| 15 | Katara | You are the most sexist, immature, nut brain... | 0.157 | 0.843 | 0.000 | -0.4199 |

Figure 7: Sentiment analysis results

## 2.2 NER


Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 3
+ Code + Text

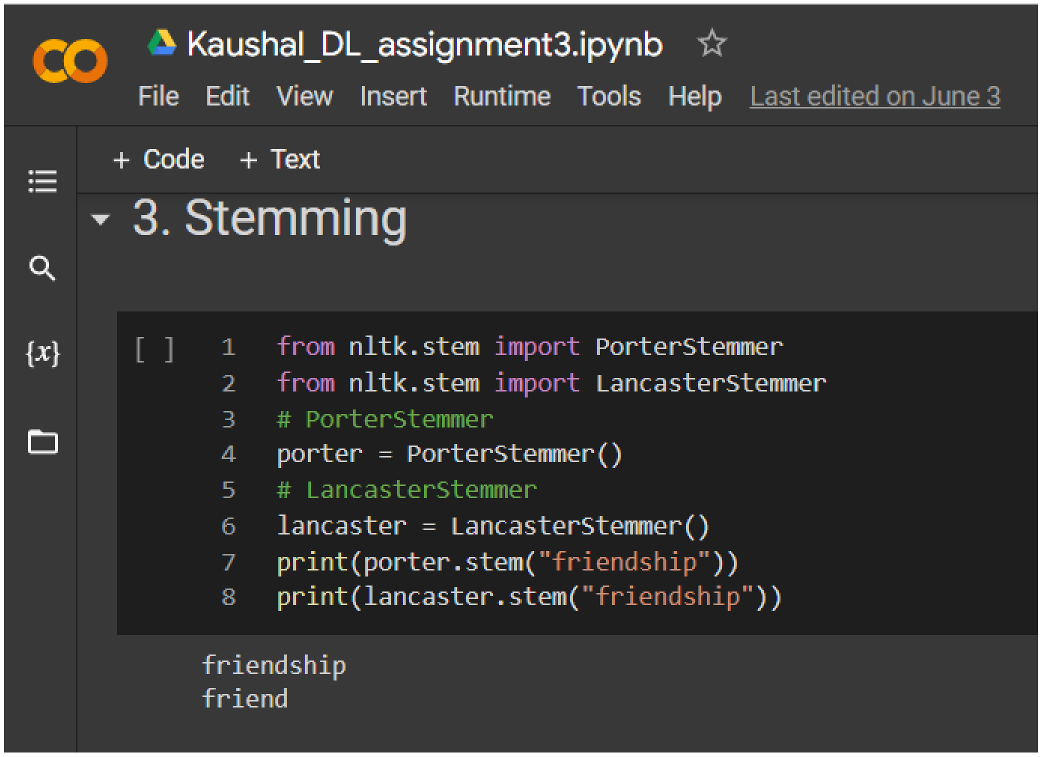
Named Entity Recognition (NER)

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("Biden invites Ukrainian president to White House this summer")
print([(X.text, X.label_) for X in doc.ents])
```

```
[('Ukrainian', 'NORP'), ('White House', 'ORG'), ('this summer', 'DATE')]
```

Figure 8: NER implementation code and results

## 2.3 Stemming

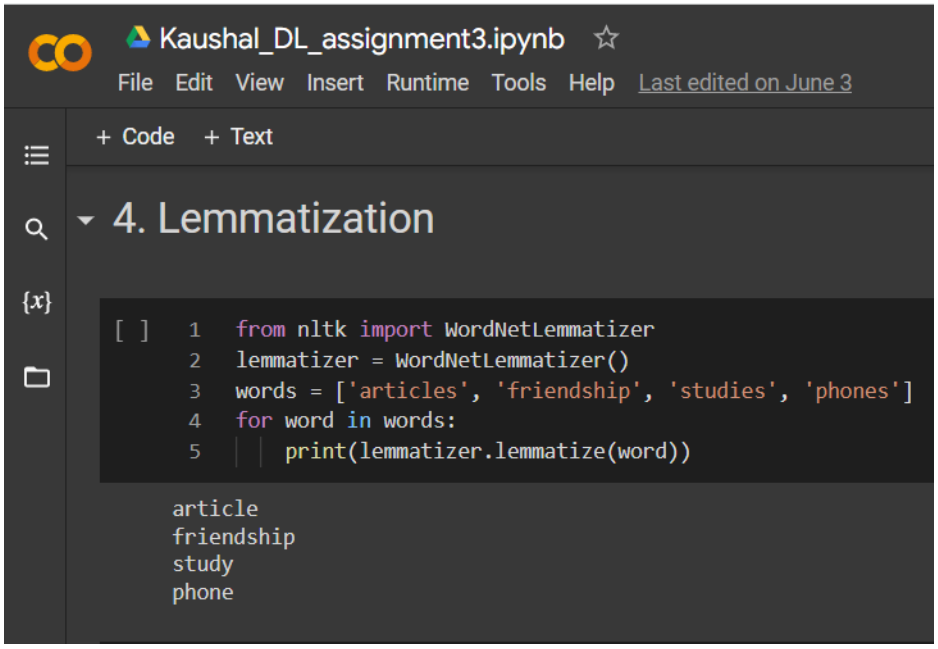

Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 3
+ Code + Text
3. Stemming

```python
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
# PorterStemmer
porter = PorterStemmer()
# LancasterStemmer
lancaster = LancasterStemmer()
print(porter.stem("friendship"))
print(lancaster.stem("friendship"))
```

```
friendship
friend
```

Figure 9: Stemming implementation code and results

## 2.4 Lemmatization

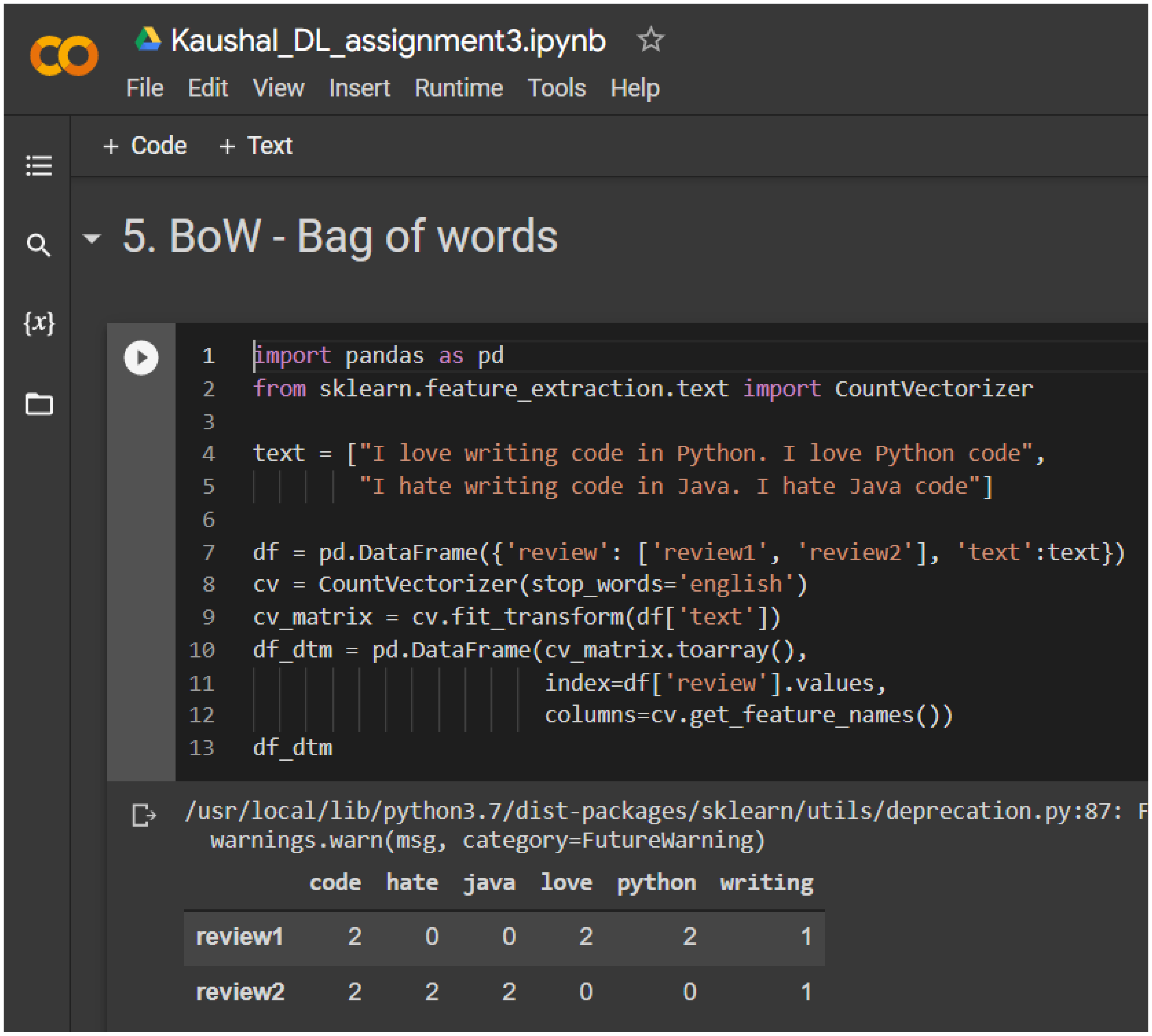

Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 3
+ Code + Text
4. Lemmatization

```python
from nltk import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
words = ['articles', 'friendship', 'studies', 'phones']
for word in words:
    print(lemmatizer.lemmatize(word))
```

```
article
friendship
study
phone
```

Figure 10: Lemmatization implementation code and results

## 2.5 BoW


Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
5. BoW - Bag of words

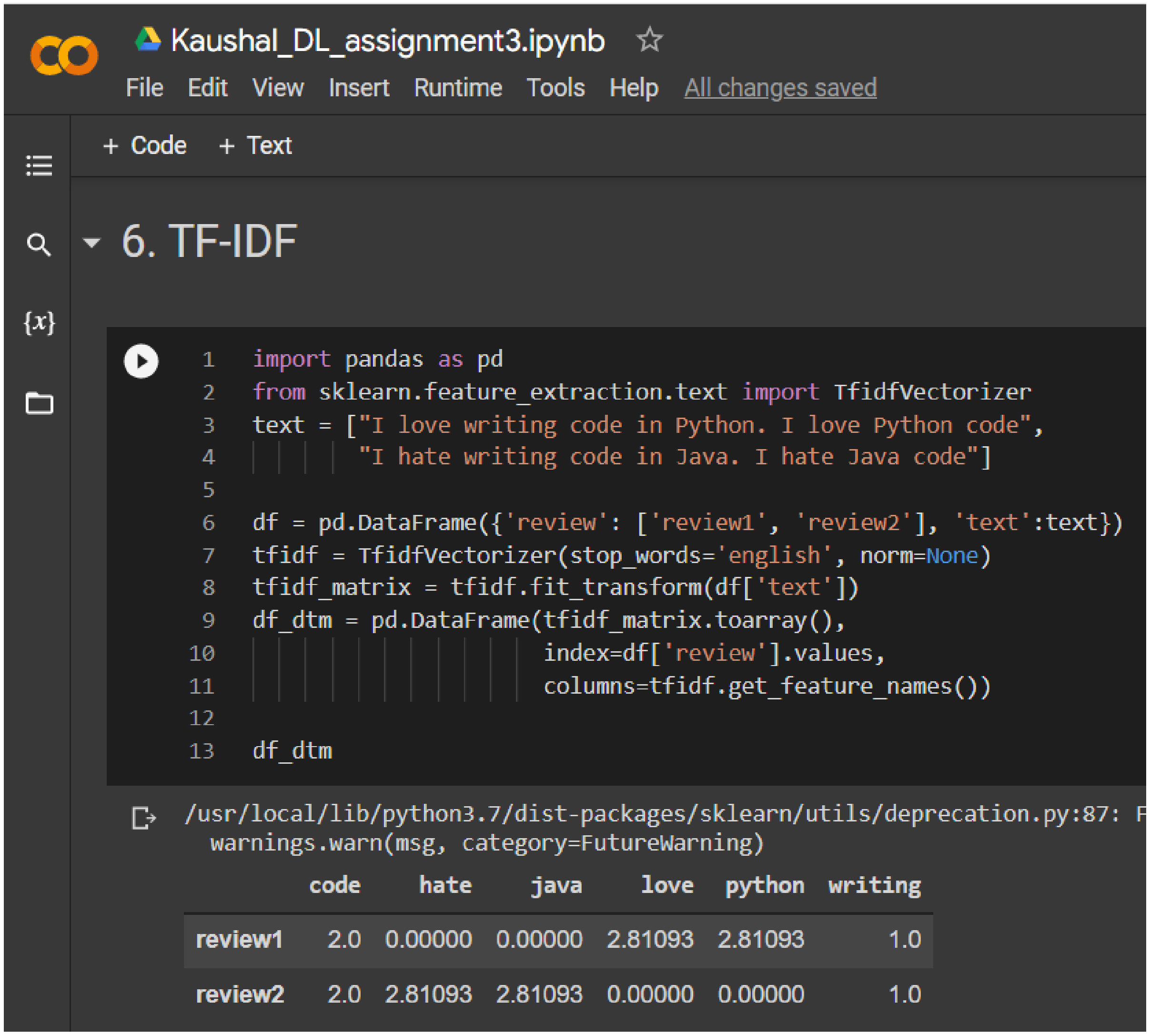
```python
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

text = ["I love writing code in Python. I love Python code",
        "I hate writing code in Java. I hate Java code"]

df = pd.DataFrame({'review': ['review1', 'review2'], 'text':text})
cv = CountVectorizer(stop_words='english')
cv_matrix = cv.fit_transform(df['text'])
df_dtm = pd.DataFrame(cv_matrix.toarray(),
                      index=df['review'].values,
                      columns=cv.get_feature_names())
df_dtm
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: F
  warnings.warn(msg, category=FutureWarning)
```

| | code | hate | java | love | python | writing |
|---|---|---|---|---|---|---|
| review1 | 2 | 0 | 0 | 2 | 2 | 1 |
| review2 | 2 | 2 | 2 | 0 | 0 | 1 |

Figure 11: Bag of words(BoW) implementation code and results

## 2.6 TF-IDF


Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
6. TF-IDF

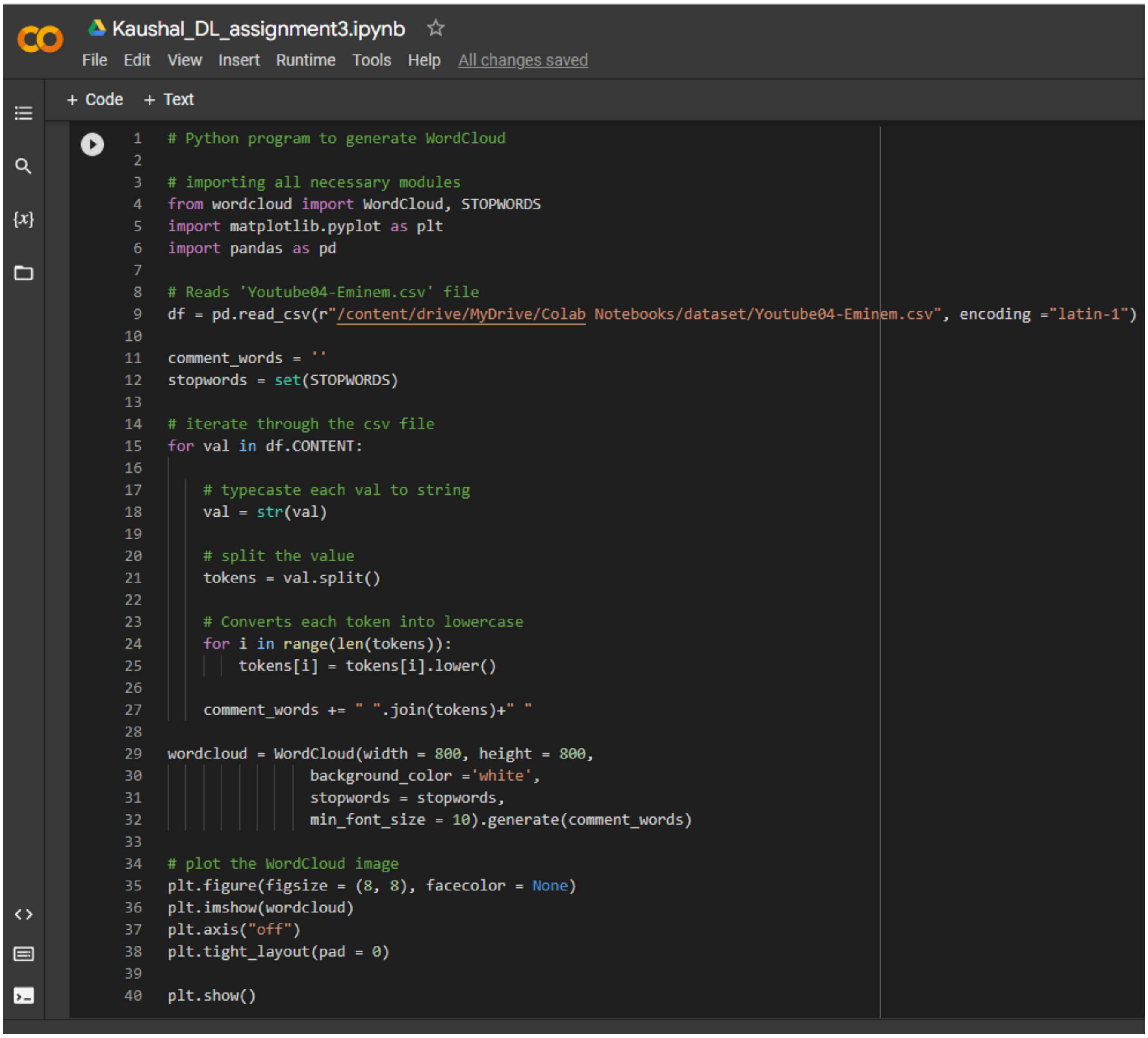
```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
text = ["I love writing code in Python. I love Python code",
        "I hate writing code in Java. I hate Java code"]

df = pd.DataFrame({'review': ['review1', 'review2'], 'text':text})
tfidf = TfidfVectorizer(stop_words='english', norm=None)
tfidf_matrix = tfidf.fit_transform(df['text'])
df_dtm = pd.DataFrame(tfidf_matrix.toarray(),
                      index=df['review'].values,
                      columns=tfidf.get_feature_names())

df_dtm
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: F
  warnings.warn(msg, category=FutureWarning)
```

| | code | hate | java | love | python | writing |
|---|---|---|---|---|---|---|
| review1 | 2.0 | 0.00000 | 0.00000 | 2.81093 | 2.81093 | 1.0 |
| review2 | 2.0 | 2.81093 | 2.81093 | 0.00000 | 0.00000 | 1.0 |

Figure 12: TF-IDF implementation code and results

## 2.7 Word cloud



Kaushal_DL_assignment3.ipynb

```python
# Python program to generate WordCloud

# importing all necessary modules
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

# Reads 'Youtube04-Eminem.csv' file
df = pd.read_csv(r"/content/drive/MyDrive/Colab Notebooks/dataset/Youtube04-Eminem.csv", encoding ="latin-1")

comment_words = ''
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in df.CONTENT:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    comment_words += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

Figure 13: Word cloud implementation code

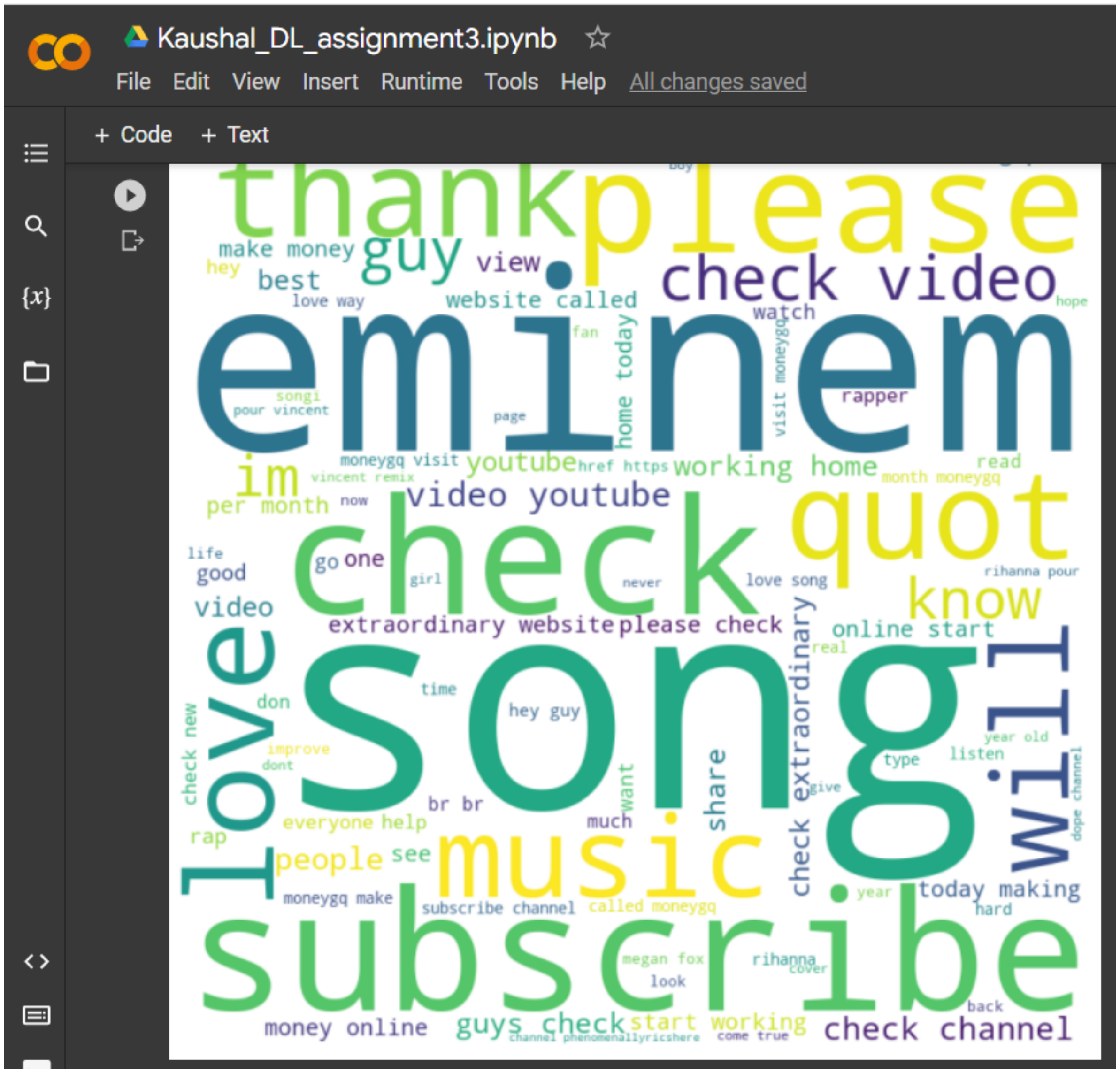
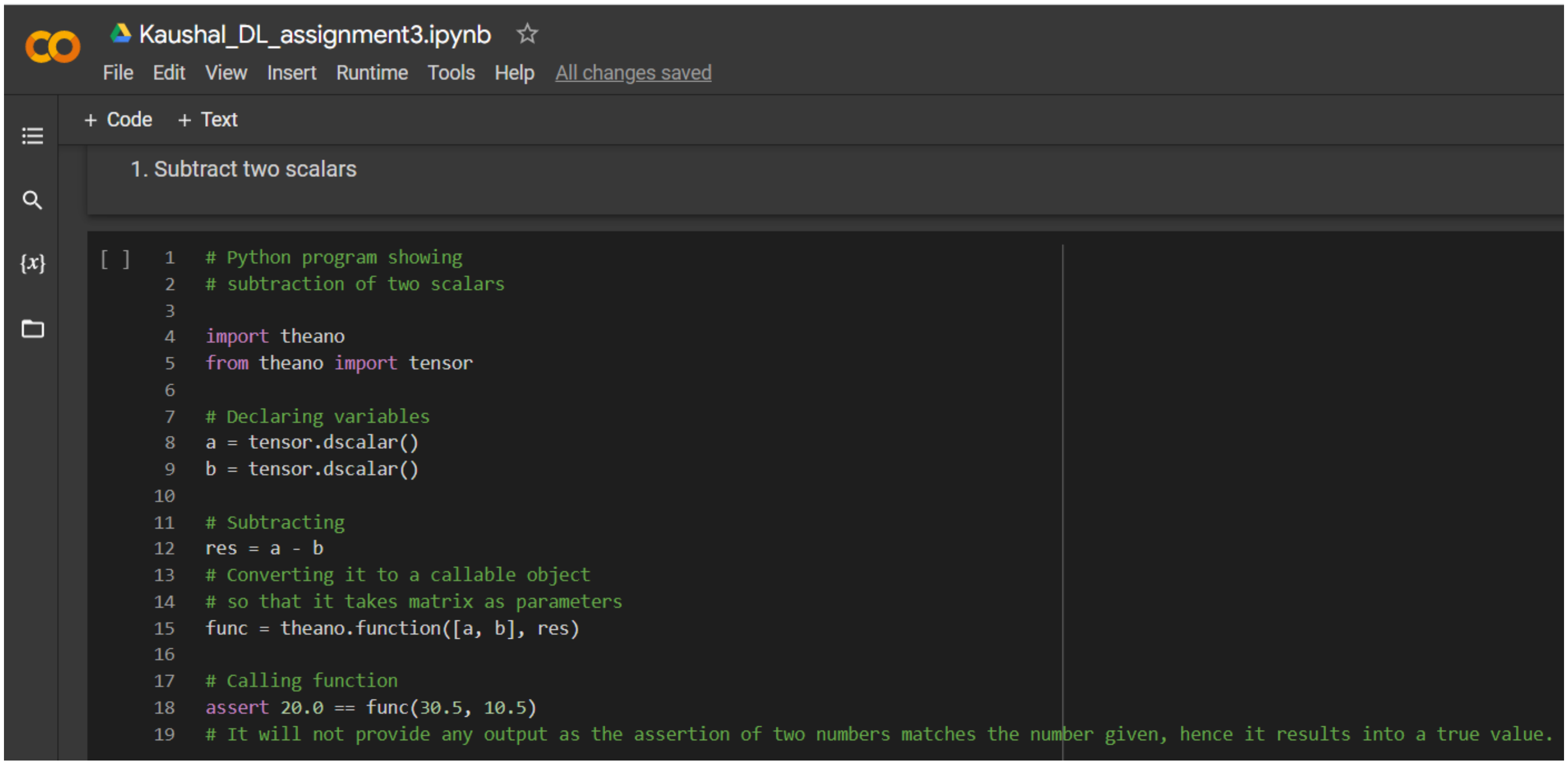

Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
thank please check video eminem quot check song love will music subscribe

Figure 14: Word cloud implementation results

## 3 Perform following operations using theano libraray

- **Subtract two scalars**
- **Add two matrices**
- **Logistic function**

### 3.1 Subtract two scalars

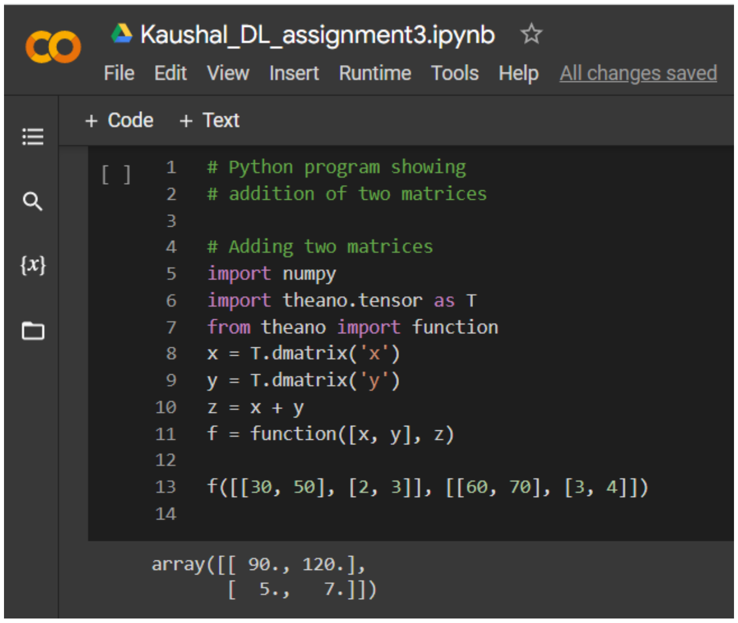


Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
1. Subtract two scalars

```
# Python program showing
# subtraction of two scalars

import theano
from theano import tensor

# Declaring variables
a = tensor.dscalar()
b = tensor.dscalar()

# Subtracting
res = a - b
# Converting it to a callable object
# so that it takes matrix as parameters
func = theano.function([a, b], res)

# Calling function
assert 20.0 == func(30.5, 10.5)
# It will not provide any output as the assertion of two numbers matches the number given, hence it results into a true value.
```

Figure 15: Substraction of two scalars using theano

## 3.2 Add two matrices

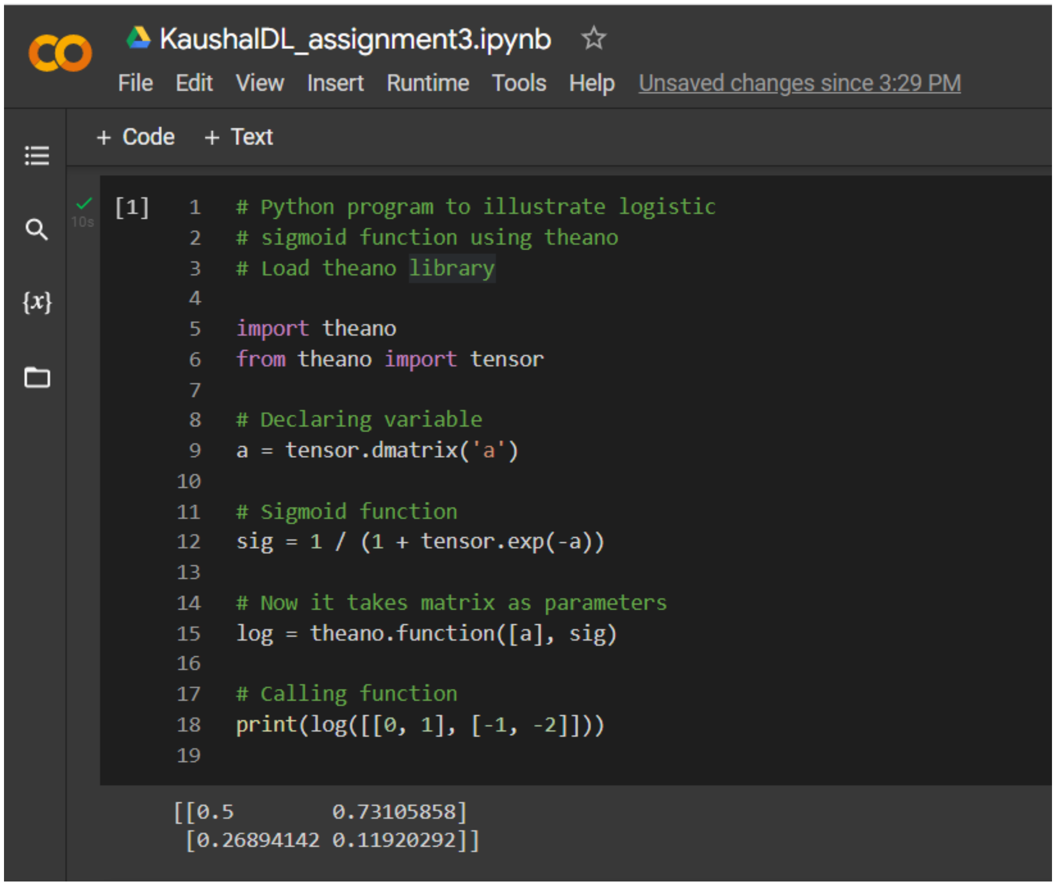

Kaushal_DL_assignment3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
# Python program showing
# addition of two matrices

# Adding two matrices
import numpy
import theano.tensor as T
from theano import function
x = T.dmatrix('x')
y = T.dmatrix('y')
z = x + y
f = function([x, y], z)

f([[30, 50], [2, 3]], [[60, 70], [3, 4]])

```

```
array([[ 90., 120.],
       [  5.,   7.]])
```

Figure 16: Addition of two matrices using theano

## 3.3 Logistic function


KaushalDL_assignment3.ipynb
File Edit View Insert Runtime Tools Help Unsaved changes since 3:29 PM
+ Code + Text
10s
[1]

```
# Python program to illustrate logistic
# sigmoid function using theano
# Load theano library

import theano
from theano import tensor

# Declaring variable
a = tensor.dmatrix('a')

# Sigmoid function
sig = 1 / (1 + tensor.exp(-a))

# Now it takes matrix as parameters
log = theano.function([a], sig)

# Calling function
print(log([[0, 1], [-1, -2]]))

```

```
[[0.5        0.73105858]
 [0.26894142 0.11920292]]
```

Figure 17: Logistic function implementation using theano

## 4 Case study:

## • Caffe framework

## • H2O.ai

## • GraphLab

### 4.1 Caffe framework

Caffe is a framework for machine learning in vision applications. It was created by Yangqing Jia during his PhD at UC Berkeley and was developed by the Berkeley Vision and Learning Center.

**Features:**
1. It is an open source library.

2. It has got an extensive architecture which encourages innovation and application.

3. It has extensible code which encourages development.

4. It is quite fast. It takes 1 ms/image for inference and 4 ms/image for learning.
They say "We believe that Caffe is the fastest ConvNet implementation available."

5. It has a huge community.
It is used by companies such asFlicker, Yahoo, and Adobe.

**There are 4 steps in training a CNN using Caffe:**

•Step 1 - Data preparation: In this step, we clean the images and store them in a format that can be used by Caffe. We will write a Python script that will handle both image pre-processing and storage.

• Step 2 - Model definition: In this step, we choose a CNN architecture and we define its parameters in a configuration file with extension .prototxt.

•Step 3 - Solver definition: The solver is responsible for model optimization. We define the solver parameters in a configuration file with extension .prototxt.

•Step 4 - Model training: We train the model by executing one Caffe command from the terminal. After training the model, we will get the trained model in a file with extension .caffemodel.

## 4.2 H2O.ai

H2O is a fully open source, distributed in-memory machine learning platform with linear scalability. H2O supports the most widely used statistical and machine learning algorithms including gradient boosted machines, generalized linear models, deep learning and more. H2O also has an industry leading AutoML functionality that automatically runs through all the algorithms and their hyperparameters to produce a leaderboard of the best models. The H2O platform is used by over 18,000 organizations globally and is extremely popular in both the R and Python communities.

H2O.ai is an open-source leader in AI and machine learning. It aims to democratize artificial intelligence for all. This platform is transforming the use of AI with software with its category-creating visionary open-source machine learning platform.

• It fully automates some of the most challenging and productive tasks in applied data science such as feature engineering, model tuning, and model deployment.

• H2O's core strength is its high-performing ML components, which are tightly integrated.
• H2O Sparkling water allows users to combine the fast, scalable machine learning algorithms of H2O with the capabilities of Spark.
• Data scientists of all proficiency levels can train and deploy modeling pipelines easily from the GUI with Driverless AI.
• H2O is an anti-money laundering and uses AI to stop crimes.
• Can detect frauds.
• Personalizes credit decisions with AI.
• Supports commonly used algorithms of machine learning.

**H2O Sparkling Water**
Sparkling Water allows users to combine the fast, scalable machine learning algorithms of H2O with the capabilities of Spark. Sparkling Water is ideal for H2O users who need to manage large clusters for their data processing needs and want to transfer data from Spark to H2O (or vice versa).

**H2O4GPU**
H2O4GPU is an open-source, GPU-accelerated machine learning package with APIs in Python and R that allows anyone to take advantage of GPUs to build advanced machine learning models.

**H2O Driverless AI** H2O Driverless AI is H2O.ai's flagship product for automatic machine learning. It fully automates some of the most challenging and productive tasks in applied data science such as feature engineering, model tuning, model ensembling and model deployment. With Driverless AI, data scientists of all proficiency levels can train and deploy modelling pipelines with just a few clicks from the GUI. Driverless AI is a commercially licensed product with a 21-day free trial version.

## 4.3 GraphLab

GraphLab is a new parallel framework for machine learning written in C++. It is an open source project and has been designed considering the scale, variety and complexity of real world data. It incorporates various high level algorithms such as Stochastic Gradient Descent (SGD), Gradient Descent and Locking to deliver high performance experience. It helps data scientists and developers easily create and install applications at large scale.

But, what makes it amazing? It's the presence of neat libraries for data transformation, manipulation and model visualization. In addition, it comprises of scalable machine learning toolkits which has everything (almost) required to improve machine learning models. The toolkit includes implementation for deep learning, factor machines, topic modeling, clustering, nearest neighbors and more.

Here is the complete architecture of GraphLab Create.

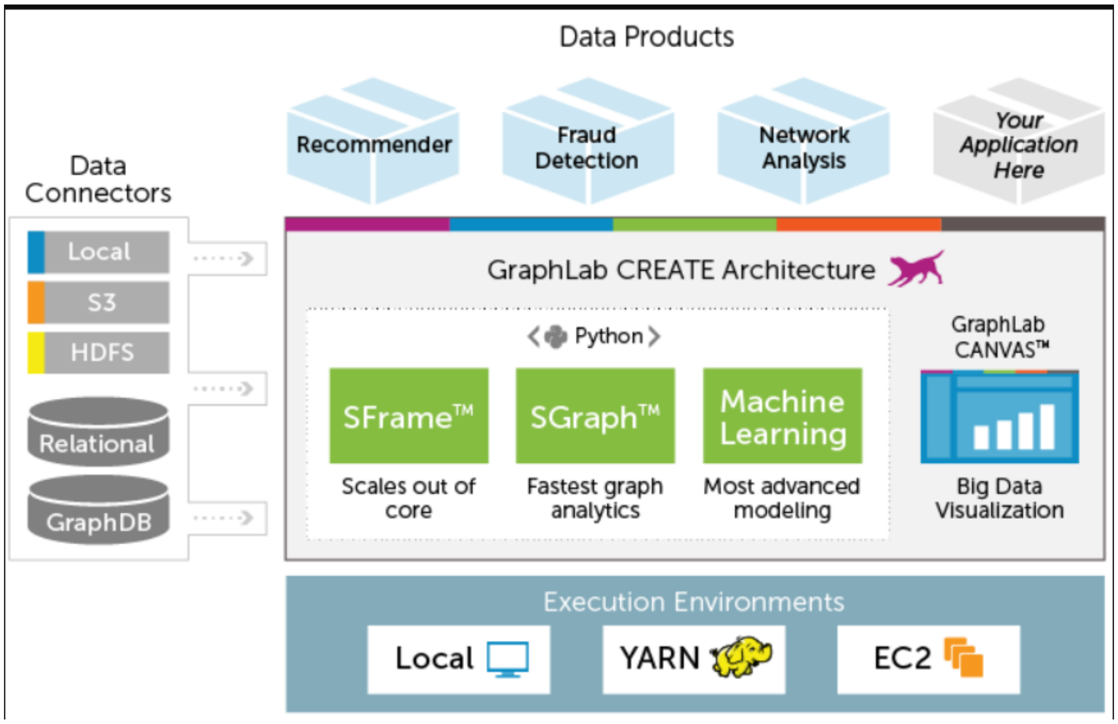


Figure 18: GraphLab Architecture

**Benefits of using GraphLab**
There are multiple benefits of using GraphLab as described below:

Handles Large Data: Data structure of GraphLab can handle large data sets which result into scalable machine learning. Let's look at the data structure of Graph Lab:

•SFrame: It is an efficient disk-based tabular data structure which is not limited by RAM. It helps to scale analysis and data processing to handle large data set (Tera byte), even on your laptop. It has similar syntax like pandas or R data frames. Each column is an SArray, which is a series of elements stored on disk. This makes SFrames disk based. I have discussed the methods to work with "SFrames" in following sections.

•SGraph: Graph helps us to understand networks by analyzing relationships between pair of items. Each item is represented by a vertex in the graph. Relationship between items is represented by edges. In GraphLab, to perform a graph-oriented data analysis, it uses SGraph object. It is a scalable graph data structure which store vertices and edges in SFrames. To know more about this, please refer this link. Below is graph representation of James Bond characters.

•Integration with various data sources: GraphLab supports various data sources like S3, ODBC, JSON, CSV, HDFS and many more.

•Data exploration and visualization with GraphLab Canvas. GraphLab Canvas is a browser-based interactive GUI which allows you to explore tabular data, summary statistics and bi-variate plots. Using this feature, you spend less time coding for data exploration. This will help you to focus more on understanding the relationship and distribution of variables. I have discussed this part following sections.

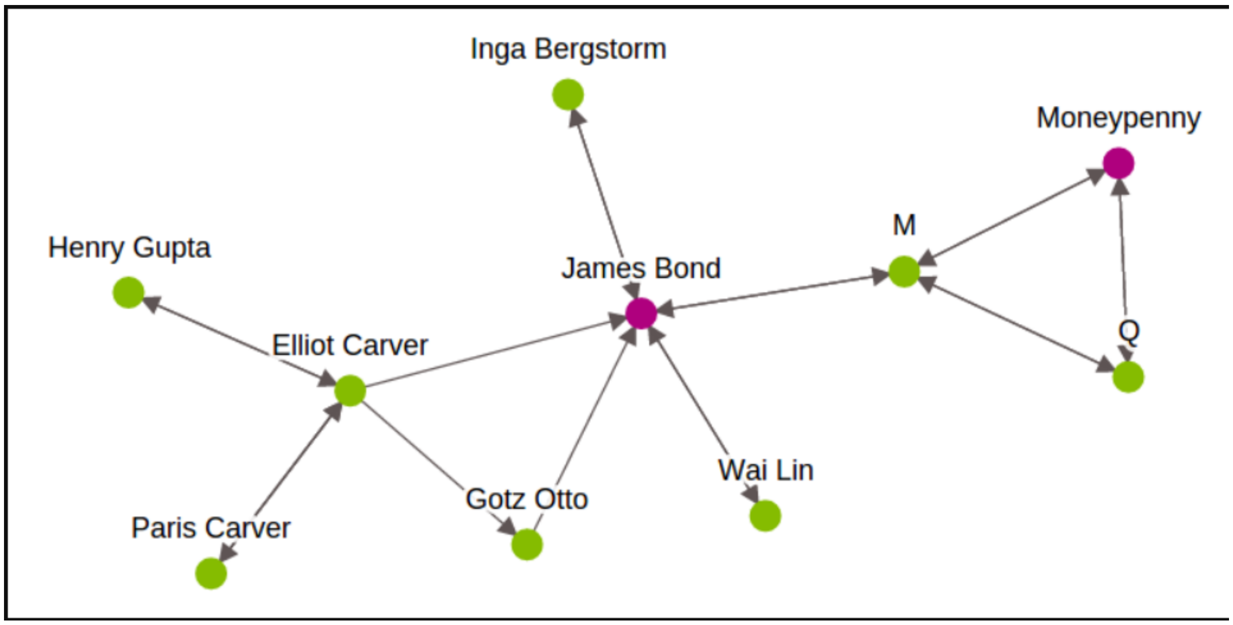


Figure 19: SGraph

•Feature Engineering: GraphLab has an inbuilt option to create new useful features to enhance model performance. It comprises of various options like transformation, binning, imputation, One hot encoding, tf-idf etc.

•Modeling: GraphLab has various toolkits to deliver easy and fast solution for ML problems. It allows you to perform various modeling exercise (regression, classification, clustering) in fewer lines of code. You can work on problems like recommendation system, churn prediction, sentiment analysis, image analysis and many more.

•Production automation: Data pipelines allow you to assemble reusable code task into jobs. Then, automatically run them on common execution environments (e.g. Amazon Web Services, Hadoop).

•GraphLab Create SDK: Advance users can extend the capabilities of GraphLab Create using GraphLab Creat SDK. You can define new machine learning models/ programs and integrate them with the rest of the package. See the GitHub repository here.

•License: It has limitation to use. You can go for 30 days free trial period or one year license for academic edition.

## 5 Why are Convolutions Important in CNNs? What are the steps to perform it?

A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.

A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

**Convolution Layer**

The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load.

This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

The main building block of CNN is the convolutional layer. Convolution is a mathematical operation to merge two sets of information. In our case the convolution is applied on the input data using a convolution filter to produce a feature map. There are a lot of terms being used so let's visualize them one by one.

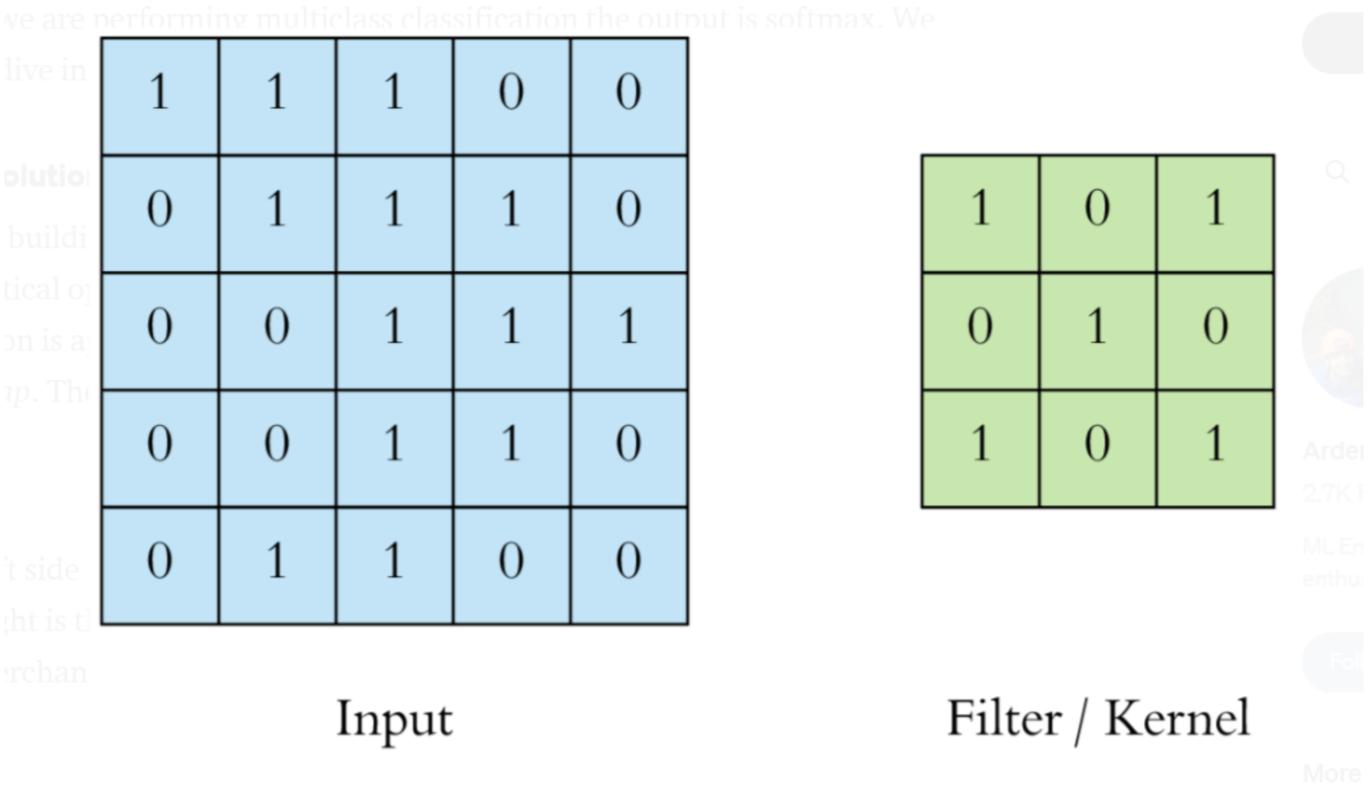


Figure 20: Input image with kernal

On the left side is the input to the convolution layer, for example the input image. On the right is the convolution filter, also called the kernel, we will use these terms interchangeably. This is called a 3x3 convolution due to the shape of the filter.

We perform the convolution operation by sliding this filter over the input. At every location, we do element-wise matrix multiplication and sum the result. This sum goes into the feature map. The green area where the convolution operation takes place is called the receptive field. Due to the size of the filter the receptive field is also 3x3.

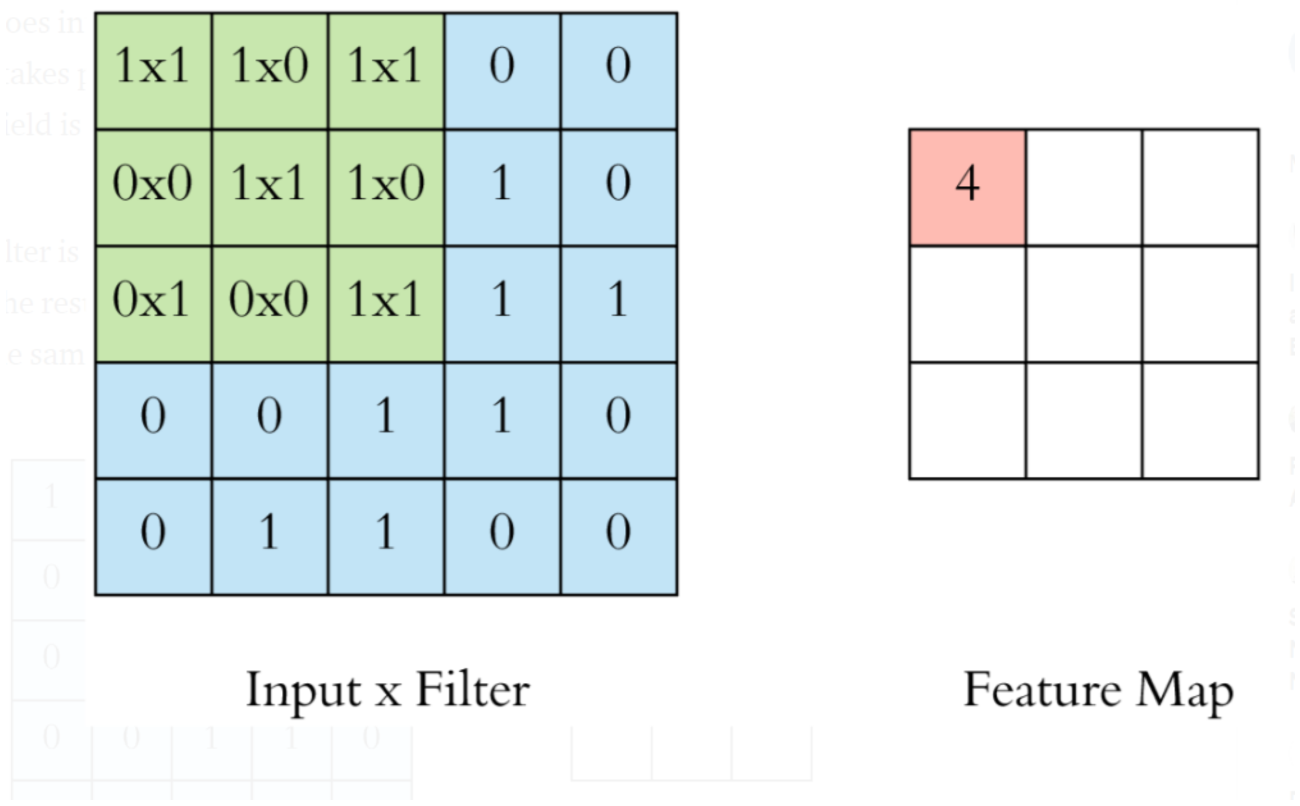


Figure 21: Convolution operation implementation

Here the filter is at the top left, the output of the convolution operation "4" is shown in the resulting feature map. We then slide the filter to the right and perform the same operation, adding that result to the feature map as well.

We continue like this and aggregate the convolution results in the feature map.

**Pooling Layer**
The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

After a convolution operation we usually perform pooling to reduce the dimensionality. This enables us to reduce the number of parameters, which both shortens the training time and combats overfitting. Pooling layers downsample each feature map independently, reducing the height and width, keeping the depth intact.

The most common type of pooling is max pooling which just takes the max value in the pooling window. Contrary to the convolution operation, pooling has no parameters. It slides a window over its input, and simply takes the max value in the window. Similar to a convolution, we specify the window size and stride.

Here is the result of max pooling using a 2x2 window and stride 2. Each color denotes a different window. Since both the window size and stride are 2, the windows are not overlapping.

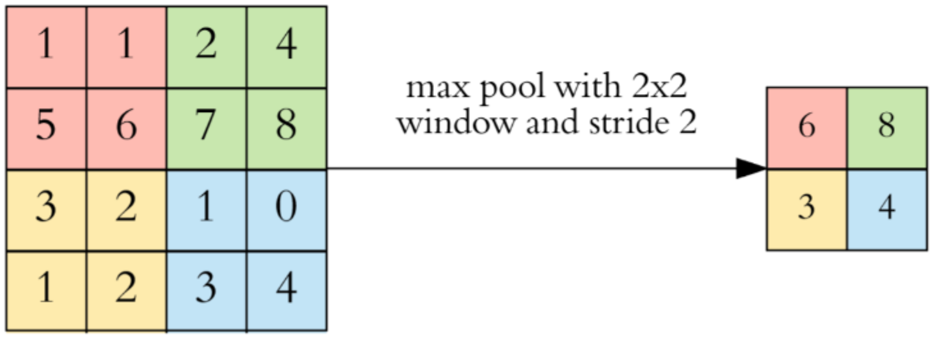


Figure 22: Pooling operation using 2x2 window

Note that this window and stride configuration halves the size of the feature map. This is the main use case of pooling, downsampling the feature map while keeping the important information.

Now let's work out the feature map dimensions before and after pooling. If the input to the pooling layer has the dimensionality 32x32x10, using the same pooling parameters described above, the result will be a 16x16x10 feature map. Both the height and width of the feature map are halved, but the depth doesn't change because pooling works independently on each depth slice the input.

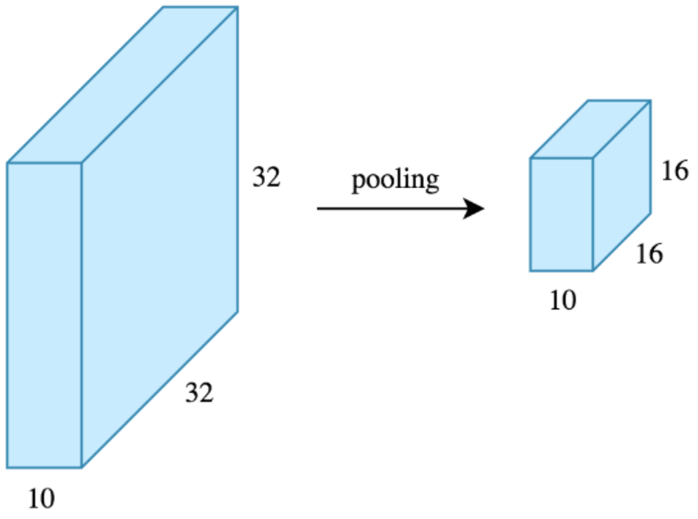


Figure 23: Dimentionality reduction using pooling operation

By halving the height and the width, we reduced the number of weights to 1/4 of the input. Considering that we typically deal with millions of weights in CNN architectures, this reduction is a pretty big deal.

In CNN architectures, pooling is typically performed with 2x2 windows, stride 2 and no padding. While convolution is done with 3x3 windows, stride 1 and with padding.

**Fully Connected Layer**

Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. This is why it can be computed as usual by a matrix multiplication followed by a bias effect.

The FC layer helps to map the representation between the input and the output.