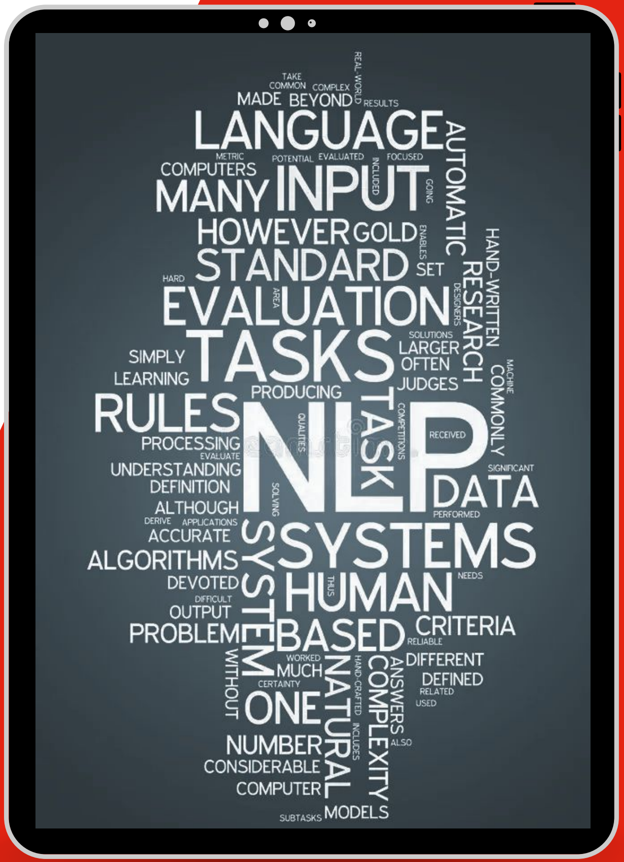
Basic Text Representation

Muhajir Akbar Hasibuan

Class Meeting 3

2023



Muhajir Akbar Hasibuan

Data Scientist

muhajirakbarhsb@gmail.com

+62 85297023234

Jakarta, Indonesia

linkedin.com/in/muhajirakbarhsb

github.com/muhajirakbarhsb

medium.com/@muhajirakbarhsb

Experience

Telkom Indonesia *Sept 2020 - present*

- Develop internal and external use cases
- Provide data understanding in making a model
- Provide Preparation and data engineering according to the use case implemented
- Provide data validation so that the analysis results are as expected
- Building modeling for the development or improvement of internal and external programs
- Provide descriptive and diagnostic insight into data processing
- Recommend and define new growth hacking strategy for digital marketing team

Project

- Pioneered a Robust Big Data Solution for MyIndihome, Revolutionizing Customer Experiences
- Orchestrated a High-Performing ML Team, Elevating myIndihome TV's Personalized Content Impact by 25%
- Envisioned and Executed a Cutting-Edge Big Data Solution for Elevated Customer Engagement on myIndihomeTV
- Fueled Business Insights via Dynamic Data Profiling, Performance Dashboards, and Insights for Langit Musik, RBT, and Upoint
- Engineered Innovative Big Data Solutions that Propelled Growth for PadiUmkm (E-Commerce)
- Powered Success for GameQoo through Strategic Big Data Solutions
- Architected and Established MLOps Framework, Elevating Telkom Indonesia's Digital Business Products
- Crafted Visionary Video Analytics Solutions for Telkom Indonesia's Revolutionary Digital Business IoT Product

MEMBER OF DATA SCIENTIST TASK FORCE | NOVEMBER 2021 - PRESENT
MEMBER OF AI TASK FORCE | NOVEMBER 2022- PRESENT

A Pool of data scientists and AI Engineer Expert in Telkom Indonesia. It was established to leverage data-driven culture for decision-making within the organization. (Applied Research, Standardization, Consultation)

Achievement

BEST Talent of the Year at Telkom Indonesia, Digital Business and Technology Division - Digital Technology and Platform -2022

Codex by Telkom Indonesia *Apr 2020- Sept 2020*

- Building Data Pipeline for Langit Musik Recommender System
- Business Analytic for Langit Musik
- Build Recommender System for Langit Musik

Universitas Syiah Kuala *2015-2020*

- Bachelor of Science, Statistics

Skills

Hard Skills

- Data Analytics
- Statistics
- Machine Learning
- Deep Learning
- MlOps
- Business Intelligence
- Data Engineering
- Cloud Computing
- Time Series & Demand Forecasting
- Natural Language Processing
- Fraud & Anomaly Detection
- Computer Vision

Tools Skills

- Python
- Pyspark
- Sql
- Apache Airflow
- GCP
- Docker
- MLflow
- Prometheus
- Evidently
- Grafana
- Redash, Metabase, Superset, Looker Studio
- Pytorch

Class Meeting 1: Introduction to NLP (1 session)
Introduction to NLP
Basics of text data and its characteristics
The importance of NLP in today's world
Overview of the course structure and objectives

Class Meeting 2: Text Preprocessing (1 session)
Understanding the text preprocessing pipeline
Tokenization, stemming, and lemmatization
Stop words removal
Hands-on exercises with Python for text preprocessing

Class Meeting 3: Text Representation (1 session)
Introduction to text representation techniques
Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF)
n-gram
Word embeddings (Word2Vec, GloVe)**just an introduction
Practical exercises on text representation

Class Meeting 4: Introduction to ML, DL Supervised and Unsupervised (1 session)
Introduction to deep learning (ML)
Introduction to deep learning (DL)
DL vs. traditional machine learning (ML)
Machine Learning and their applications
Neural networks and their applications
Basic ML DL concepts and terminology

Class Meeting 5: Sentiment Analysis (1 session)
What is sentiment analysis?
Data collection and labeling for sentiment analysis
Building a sentiment analysis model
Practical sentiment analysis examples

Class Meeting 6: Text Classification (1 session)
Introduction to text classification
Binary and multi-class classification
Building a text classification model
Real-world text classification examples

Class Meeting 7: Introduction to Topic Modeling(1 session)
Introduction to topic Modelingg
Topic Modeling Implementation
LDA(Latent dirichlet allocation)
Hands-on exercises with Topic Modeling

Class Meeting 8: Mid-term Exam

Class Meeting 9: Introduction to Word Embeddings (1 session)
Fundamentals of word embeddings
Word2Vec, GloVe, and FastText
Word embedding applications
Hands-on exercises with word embedding

Class Meeting 10: Attention Mechanisms (1 session)
Introduction to attention mechanisms
Self-attention and multi-head attention
Transformers architecture
Practical examples of attention in NLP

Class Meeting 11: Transformer Models (1 session)
In-depth study of the Transformer model
Pre-trained transformer models (BERT, GPT-2)
Fine-tuning transformers for NLP tasks
Transformer-based applications

Class Meeting 12: Advanced NLP Topics (1 session)
Advanced NLP topics such as BERT, XLNet, and RoBERTa
Transfer learning in NLP
Ethics in NLP
Recent developments and trends in NLP

Class Meeting 13: Advanced NLP Topics and Deployment Process in Industry (1 session)
Introduction to LLM
Introduction how industry utilize NLP to generate revenue
Introduction MLOps for NLP(bonuses from practitioners)

Class Meeting 14-15: Project Work (2 sessions)
Dedicated sessions for students to work on NLP projects with guidance and assistance.

Class Meeting 16: Project Presentations and Conclusion (1 session)
Students present their NLP projects
Recap of key takeaways from the course
Discuss further resources for NLP enthusiasts
Course conclusion and feedback

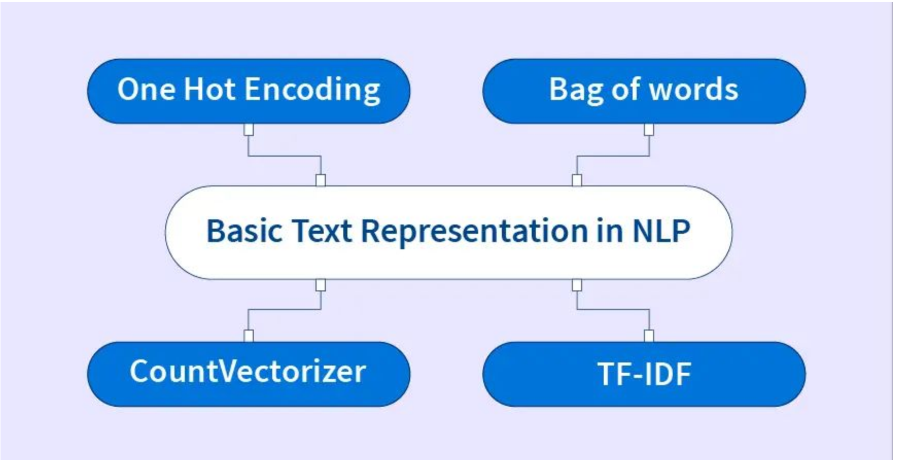
Agenda

- Introduction to Text Representation Techniques
- Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF)
- n-gram
- Hands-on Python Exercises

Introduction to Text Representation

Text Representation

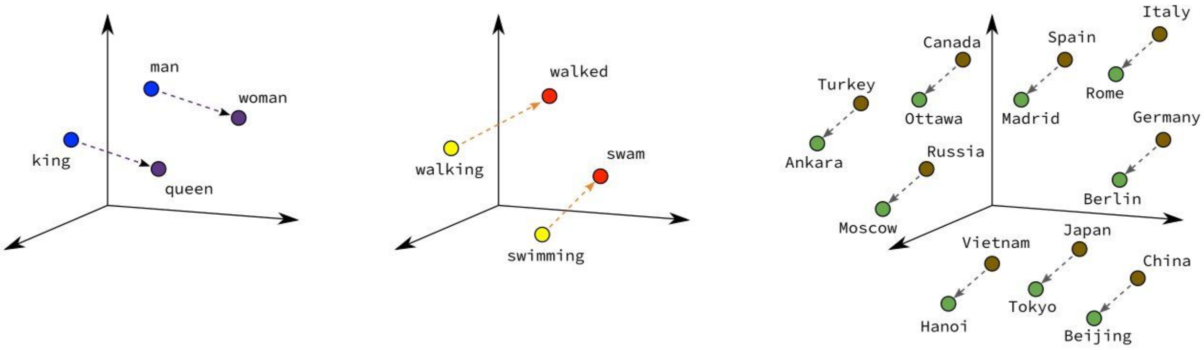
Computers are brilliant when dealing with **numbers**. They are faster than humans in calculations & decoding patterns by many orders of magnitude. But what if the data is **not numerical**? What if it's **language**? What happens when the data is in **characters, words & sentences?** How do we make **computers process our language?**



Text Representation

Feature Extraction is a general term that is also known as a **text representation of text vectorization** which is a **process of converting text into numbers**. we call vectorization because when text is **converted in numbers it is in vector form**.

Now the second question would be Why do we need feature extraction? So we know that machines can only understand numbers and to make machines able to identify language we need to convert it into numeric form.



Male-Female   Verb Tense   Country-Capital

Common term used

- **Corpus(C)** ~ The total number of combinations of words in the whole dataset is known as Corpus. In simple words concatenating all the text records of the dataset forms a corpus.

- **Vocabulary(V)** ~ a total number of distinct words which form your corpus is known as Vocabulary.

- **Document(D)** ~ There are multiple records in a dataset so a single record or review is referred to as a document.

- **Word(W)** ~ Words that are used in a document are known as Word.

Discrete Text Representation

Discrete Text Representation

These are representations where words are represented by their corresponding indexes to their position in a dictionary from a larger corpus or corpora.

Famous representations that fall within this category are:

- One-Hot encoding
- Bag-of-words representation (BOW)
- n-gram
- Basic BOW — CountVectorizer
- Advanced BOW — TF-IDF

One-hot Encoding

It is a type of representation that assigns 0 to all elements in a vector except for one, which has a value of 1. This value represents a category of an element.

For example:

If i had a sentence, “**I love my dog**”, each word in the sentence would be represented as below:

```
I → [1 0 0 0], love → [0 1 0 0], my → [0 0 1 0], dog → [0 0 0 1]
```

The entire sentence is then represented as:

```
sentence = [ [1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1] ]
```

One-hot Encoding

The intuition behind one-hot encoding is that each bit **represents a possible category** & if a particular variable cannot fall into multiple categories, then a **single bit is enough** to represent it

As you may have grasped, **the length of an array of word depends on the vocabulary size**. This is **not scalable** for a very large corpus which could contain up to 100,000 unique words or even more.

One-hot Encoding

Advantages of one-hot encoding:

- Easy to understand & implement

Disadvantages of one-hot encoding:

- Explosion in feature space if number of categories are very high
- The vector representation of words is orthogonal and cannot determine or measure relationship between different words
- Cannot measure importance of a word in a sentence but understand mere presence/absence of a word in a sentence
- High dimensional s**parse matrix representation can be memory & computationally** expensive

Dense Matrix

1	2	31	2	9	7	34	22	11	5
11	92	4	3	2	2	3	3	2	1
3	9	13	8	21	17	4	2	1	4
8	32	1	2	34	18	7	78	10	7
9	22	3	9	8	71	12	22	17	3
13	21	21	9	2	47	1	81	21	9
21	12	53	12	91	24	81	8	91	2
61	8	33	82	19	87	16	3	1	55
54	4	78	24	18	11	4	2	99	5
13	22	32	42	9	15	9	22	1	21

Sparse Matrix

1	.	3	.	9	.	3	.	.	.
11	.	4	.	.	.	.	.	2	1
.	.	1	.	.	.	4	.	1	.
8	.	.	.	3	1	.	.	.	.
.	.	.	9	.	.	1	.	17	.
13	21	.	9	2	47	1	81	21	9
.	.	.	.	.	.	.	.	.	.
.	.	.	.	19	8	16	.	.	55
54	4	.	.	.	11	.	.	.	.
.	.	2	.	.	.	.	22	.	21

Bag of Words Representation

Bag-of-words representation as the name suggests intutively, **puts words in a "bag" & computes frequency of occurrence of each word**. It does not take into account the word order or lexical information for text representation

The intuition behind BOW representation is that **document having similar words are similar irrespective of the word positioning**

Bag of Words - CountVectorizer

The CountVectorizer computes the **frequency of occurrence of a word in a document**. It converts the corpus of multiple sentences (say product reviews) into a matrix of reviews & words & fills it with frequency of each word in a sentence


```
from sklearn.feature_extraction.text import CountVectorizer
text = ["i love nlp. nlp is so cool"]
vectorizer = CountVectorizer()
# tokenize and build vocab
vectorizer.fit(text)
print(vectorizer.vocabulary_)
# Output: {'love': 2, 'nlp': 3, 'is': 1, 'so': 4, 'cool': 0}
# encode document
vector = vectorizer.transform(text)
# summarize encoded vector
print(vector.shape) # Output: (1, 5)
print(vector.toarray())
```

```
{'love': 2, 'nlp': 3, 'is': 1, 'so': 4, 'cool': 0}
(1, 5)
[[1 1 1 2 1]]
```

As you see the word “nlp” occurs twice in the sentence & also falls in index 3. Which we can see as the output of the final print statement

The “weight” of a word in a sentence is its frequency

Bag of Words - CountVectorizer

Advantage of CountVectorizer:

- CountVectorizer also gives us **frequency** of words in a text document/sentence which One-hot encoding fails to provide

- Length of the encoded vector is the length of the dictionary

Disadvantages of CountVectorizer:

- This method ignores the **location information of the word**. It is not possible to grasp the **meaning of a word** from this representation

- The intuition tha**t high-frequency words are more important or give more information about the sentence fails** when it comes to stop words like **"is, the, an, I" & when the corpus is context-specific.** For example, in a corpus about **covid-19, the word coronavirus may not add a lot of value**

Bag of Words - TF-IDF

To suppress the **very high-frequency words & ignore the low-frequency words**, there is a need to **normalize the “weights”** of the words accordingly

TF-IDF representation: The full form of TF-IDF is term frequency-inverse document frequency is a product of 2 factors

$$TFIDF = TF(w, d) * IDF(w)$$

Where, TF(w, d) is frequency of word ‘w’ in document ‘d’

IDF(w) can be further broken down as:

$$IDF(w) = log(\frac{N}{df(w)})$$

Where, N is total number of documents, & df(w) is the frequency of documents containing the word ‘w’

$$\boxed{\textbf{TF-IDF} = \text{Term frequency in document} \times \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing the term}}\right)}$$

Bag of Words - TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency. This is better than BoW since it **interprets the importance of a word in a document**. The idea behind TF-IDF is to weight words based on **how often they appear in a document** (the term frequency) and **how common they are across all documents** (the inverse document frequency).

Aa Sentence
news mentioned fake
audience encourage fake news
fake news false misleading

# Vector (Number)	Aa Words	# Frequency
0	fake	3
1	news	3
2	audience	1
3	encourage	1
4	false	1
5	mentioned	1
6	misleading	1

Bag of Words - TF-IDF

Let us calculate TF for sentence — 1

- news — 1 / 3 = 0.33 {News is repeated once in the sentence, and total words are 3 — giving 1/3}
- mentioned — 1 / 3 = 0.33
- fake — 1 / 3 = 0.33
- audience , encourage, false, mentioned, misleading — 0 / 3 = 0 {These words did not occur in the sentence — there is no repetition, hence zero}

Let us calculate TF for sentence — 2

- audience — 1 / 4 = 0.25 (Audience word is repeated once in the sentence, and total words in the sentence are 4 — giving 1/4)
- encourage — 1 / 4 = 0.25
- fake — 1 / 4 = 0.25
- news — 1 / 4 = 0.25
- false, mentioned, misleading — 0 / 4 = 0

Sentence
news mentioned fake
audience encourage fake news
fake news false misleading

Vector (Number)	Words	Frequency
0	fake	3
1	news	3
2	audience	1
3	encourage	1
4	false	1
5	mentioned	1
6	misleading	1

Sentence	0 (fake)	1 (news)	2 (audie...	3 (enco...	4 (false)	5 (menti...	6 (misle...
news mentioned fake	0.33	0.33	0	0	0	0.33	0
audience encourage fake news	0.25	0.25	0.25	0.25	0	0	0

Bag of Words - TF-IDF

Let us calculate IDF for all the words:

- news — log_e(3/3) = 0 {we have 3 sentences, and news word is present in all three sentences, hence log(3/3)}
- mentioned — log_e(3/1) = 1.0986
- fake — log_e(3/3) = 0
- audience — log_e(3/1) = 1.0986
- encourage — log_e(3/1) = 1.0986
- false — log_e(3/1) = 1.0986
- misleading — log_e(3/1) = 1.0986

```
            audience  encourage      fake     false  mentioned  misleading  \
Document 1  0.000000   0.000000  0.453295  0.000000   0.767495    0.000000
Document 2  0.608845   0.608845  0.359594  0.000000   0.000000    0.000000
Document 3  0.000000   0.000000  0.359594  0.608845   0.000000    0.608845

                news
Document 1  0.453295
Document 2  0.359594
Document 3  0.359594
```

Sentence
news mentioned fake
audience encourage fake news
fake news false misleading

Vector (Number)	Words	Frequency
0	fake	3
1	news	3
2	audience	1
3	encourage	1
4	false	1
5	mentioned	1
6	misleading	1

First, it smooths document count (so there is no `0`, ever):

```
df += int(self.smooth_idf)
n_samples += int(self.smooth_idf)
```

and it uses natural logarithm (`np.log(np.e)==1`)

```
idf = np.log(float(n_samples) / df) + 1.0
```

There is also default `l2` normalization applied. In short, scikit-learn does much more "nice, little things" while computing tfidf. None of these approaches (their or yours) is bad. Their is simply more advanced.

N-gram

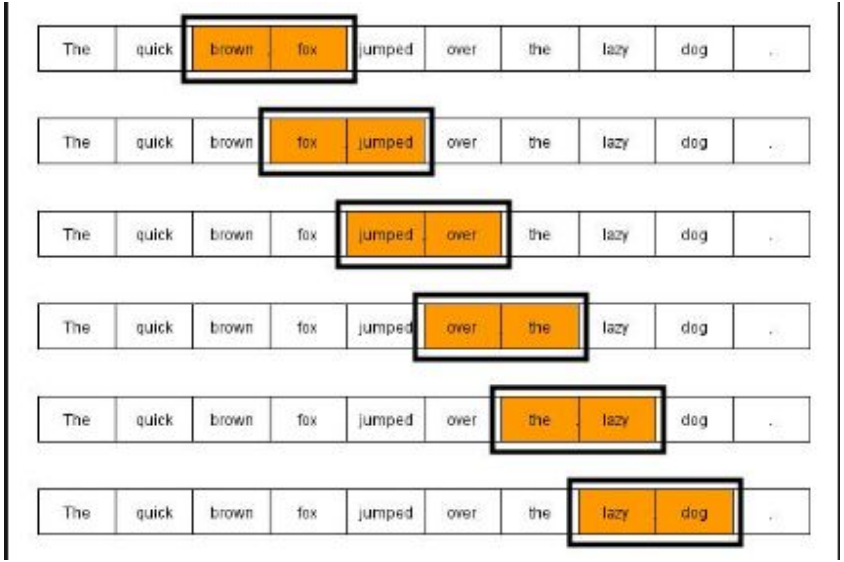
An N-gram is a traditional text representation technique that involves breaking down the text into contiguous sequences of n-words. A uni-gram gives all the words in a sentence. A Bi-gram gives sets of two consecutive words and similarly, a Tri-gram gives sets of consecutive 3 words, and so on.

Example: *The dog in the house*

Uni-gram: "The", "dog", "in", "the", "house"

Bi-gram: "The dog", "dog in", "in the", "the house"

Tri-gram: "The dog in", "dog in the", "in the house"



Class Meeting 1: Introduction to NLP (1 session)
Introduction to NLP
Basics of text data and its characteristics
The importance of NLP in today's world
Overview of the course structure and objectives

Class Meeting 2: Text Preprocessing (1 session)
Understanding the text preprocessing pipeline
Tokenization, stemming, and lemmatization
Stop words removal
Hands-on exercises with Python for text preprocessing

Class Meeting 3: Text Representation (1 session)
Introduction to text representation techniques
Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF)
n-gram
Word embeddings (Word2Vec, GloVe)**just an introduction
Practical exercises on text representation

Class Meeting 4: Introduction to ML, DL Supervised and Unsupervised (1 session)
Introduction to deep learning (ML)
Introduction to deep learning (DL)
DL vs. traditional machine learning (ML)
Machine Learning and their applications
Neural networks and their applications
Basic ML DL concepts and terminology

Class Meeting 5: Sentiment Analysis (1 session)
What is sentiment analysis?
Data collection and labeling for sentiment analysis
Building a sentiment analysis model
Practical sentiment analysis examples

Class Meeting 6: Text Classification (1 session)
Introduction to text classification
Binary and multi-class classification
Building a text classification model
Real-world text classification examples

Class Meeting 7: Introduction to Topic Modeling(1 session)
Introduction to topic Modelingg
Topic Modeling Implementation
LDA(Latent dirichlet allocation)
Hands-on exercises with Topic Modeling

Class Meeting 8: Mid-term Exam

Class Meeting 9: Introduction to Word Embeddings (1 session)
Fundamentals of word embeddings
Word2Vec, GloVe, and FastText
Word embedding applications
Hands-on exercises with word embedding

Class Meeting 10: Attention Mechanisms (1 session)
Introduction to attention mechanisms
Self-attention and multi-head attention
Transformers architecture
Practical examples of attention in NLP

Class Meeting 11: Transformer Models (1 session)
In-depth study of the Transformer model
Pre-trained transformer models (BERT, GPT-2)
Fine-tuning transformers for NLP tasks
Transformer-based applications

Class Meeting 12: Advanced NLP Topics (1 session)
Advanced NLP topics such as BERT, XLNet, and RoBERTa
Transfer learning in NLP
Ethics in NLP
Recent developments and trends in NLP

Class Meeting 13: Advanced NLP Topics and Deployment Process in Industry (1 session)
Introduction to LLM
Introduction how industry utilize NLP to generate revenue
Introduction MLOps for NLP(bonuses from practitioners)

Class Meeting 14-15: Project Work (2 sessions)
Dedicated sessions for students to work on NLP projects with guidance and assistance.

Class Meeting 16: Project Presentations and Conclusion (1 session)
Students present their NLP projects
Recap of key takeaways from the course
Discuss further resources for NLP enthusiasts
Course conclusion and feedback

Python Time

https://github.com/muhajirakbarhsb/NLP_class_2023/blob/main/Class_Meeting_3.ipynb