```
pip install PyPDF2
```

```
Collecting PyPDF2
  Downloading pypdf2-3.0.1-py3-none-any.whl (232 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 4.4 MB/s eta 0:00:00
Installing collected packages: PyPDF2
Successfully installed PyPDF2-3.0.1
```

```
import PyPDF2

file_path = 'MALIN_KUNDANG.pdf'

# Buka file PDF dalam mode binary ('rb')
with open(file_path, 'rb') as pdf_file:
    pdf_reader = PyPDF2.PdfReader(pdf_file)

    # Inisialisasi variabel untuk menyimpan teks dari PDF
    text = ''

    # Loop melalui setiap halaman dan ekstrak teks
    for page_num in range(len(pdf_reader.pages)):
        page = pdf_reader.pages[page_num]
        text += page.extract_text()

# Tampilkan teks dari PDF
print(text)
```

```
diadu dengan ayam Raden Putra dengan satu syarat, jika ayam Cindelaras kalah maka ia
bersedia kepalanya dipancung, tetapi jika ayamnya menan g maka setengah kekayaan
Raden Putra menjadi milik Cindelaras.
Dua ekor ayam itu bertarung dengan gagah berani. Tetapi da lam waktu singkat, ayam
Cindelaras berhasil menaklukkan ayam sang Raja. Para pen onton bersorak sorai mengelu-
elukan Cindelaras dan ayamnya. "Baiklah aku mengaku kal ah. Aku akan menepati janjiku.
Tapi, siapakah kau sebenarnya, anak muda?" Tanya Baginda Ra den Putra. Cindelaras
segera membungkuk seperti membisikkan sesuatu pada ayamnya.  Tidak berapa lama
ayamnya segera berbunyi. "Kukuruyuk... Tuanku Cindelaras, ruma hnya di tengah rimba,
atapnya daun kelapa, ayahnya Raden Putra...," ayam ja ntan itu berkokok berulang-ulang.
Raden Putra terperanjat mendengar kokok ayam Cindelaras.  "Benarkah itu?" Tanya
baginda keheranan. "Benar Baginda, nama hamba Cindelaras, ibu hamba adalah permaisuri
Baginda."
Bersamaan dengan itu, sang patih segera menghadap dan m enceritakan semua peristiwa
yang sebenarnya telah terjadi pada permaisuri. "Aku te lah melakukan kesalahan," kata
Baginda Raden Putra. "Aku akan memberikan hukuman yang setimpal pada selirku," lanjut
Baginda dengan murka. Kemudian, selir Raden Putra pun di buang
ke hutan. Raden Putra segera memeluk anaknya dan
meminta maaf atas kesalahannya. Setelah itu, Raden Putra
dan hulubalang segera menjemput permaisuri ke hutan.
Akhirnya Raden Putra, permaisuri dan Cindelaras dapat
berkumpul kembali. Setelah Raden Putra meninggal dunia,
Cindelaras menggantikan kedudukan ayahnya. Ia
memerintah negerinya dengan adil dan bijaksana.
AJI SAKA
Dahulu kala, ada sebuah kerajaan bernama Medang Kamulan  yang diperintah oleh raja
bernama Prabu Dewata Cengkar yang buas dan suka makan  manusia. Setiap hari sang raja
memakan seorang manusia yang dibawa oleh Patih Jugul Muda.  Sebagian kecil dari rakyat
yang resah dan ketakutan mengungsi secara diam-diam ke da erah lain.
Di dusun Medang Kawit ada seorang pemuda bernama Aji
Saka yang sakti, rajin dan baik hati. Suatu hari, Aji Saka
berhasil menolong seorang bapak tua yang sedang dipukuli
oleh dua orang penyamun. Bapak tua yang akhirnya
diangkat ayah oleh Aji Saka itu ternyata pengungsi dari
Medang Kamulan. Mendengar cerita tentang kebuasan
Prabu Dewata Cengkar, Aji Saka berniat
menolong rakyat Medang Kamulan. Dengan mengenakan se rban di kepala Aji Saka berangkat ke Medang Kamulan.
Perjalanan menuju Medang Kamulan tidaklah mulus, Aji Sak a sempat bertempur selama
tujuh hari tujuh malam dengan setan penunggu hutan, karena  Aji Saka menolak dijadikan
budak oleh setan penunggu selama sepuluh tahun sebelum dip erbolehkan melewati hutan
itu. Tapi berkat kesaktiannya, Aji Saka berhasil mengelak dari semburan api si setan.
Sesaat setelah Aji Saka berdoa, seberkas sinar kuning me nyorot dari langit menghantam
setan penghuni hutan sekaligus melenyapkannya.
Aji Saka tiba di Medang Kamulan yang sepi. Di istana, Prab u Dewata Cengkar sedang
murka karena Patih Jugul Muda tidak membawa korban untuk sang Prabu.
Dengan berani, Aji Saka menghadap Prabu Dewata Cengka r dan menyerahkan diri untuk
disantap oleh sang Prabu dengan imbalan tanah seluas serban yang digunakannya.
Saat mereka sedang mengukur tanah sesuai permintaan Aji S aka, serban terus
memanjang sehingga luasnya melebihi luas kerajaan Prabu D ewata Cengkar. Prabu marah
setelah mengetahui niat Aji Saka sesungguhnya adalah untuk mengakhiri kelalimannya.
Ketika Prabu Dewata Cengkar sedang marah, serban Aji S aka melilit kuat di tubuh sang
Prabu. Tubuh Prabu Dewata Cengkar dilempar Aji Saka dan jat uh ke laut selatan
kemudian hilang ditelan ombak.
Aji Saka kemudian dinobatkan menjadi raja Medang Kamulan. I a memboyong ayahnya ke
istana. Berkat pemerintahan yang adil dan bijaksana, Aji Saka menghantarkan Kerajaan
Medang Kamulan ke jaman keemasan, jaman dimana rakyat hid up tenang, damai, makmur
dan sejahtera.
```

```
pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

```
import nltk
nltk.download('punkt')
from nltk.tokenize import sent_tokenize

text_sent = sent_tokenize(text)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
# mengambil baris pertama untuk di one hot
from nltk.tokenize import word_tokenize
teks = ''.join(text_sent)
teks = word_tokenize(teks)
```

```
import pandas as pd

# Simpan teks ke dalam dataframe
df = pd.DataFrame({'text': [teks]})

# Simpan dataframe ke dalam file CSV
df.to_csv('text.csv', index=False)
```

Buatlah representasi data dari teks cerita malin kundang dalam bentuk PDF berikut. Metode Representasi yang harus Anda tampilkan adalah:

1. One hot encoding
2. Hash
3. Co-occurance matrix
4. Word2Vec
5. Fast text

```
pip install fasttext
```

```
Collecting fasttext
  Downloading fasttext-0.9.2.tar.gz (68 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 68.8/68.8 kB 1.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting pybind11>=2.2 (from fasttext)
  Using cached pybind11-2.11.1-py3-none-any.whl (227 kB)
Requirement already satisfied: setuptools>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from fasttext) (67.7.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from fasttext) (1.23.5)
Building wheels for collected packages: fasttext
  Building wheel for fasttext (setup.py) ... done
  Created wheel for fasttext: filename=fasttext-0.9.2-cp310-cp310-linux_x86_64.whl size=4199774 sha256=11c6e69d7567d1eb9e56ca489d548
  Stored in directory: /root/.cache/pip/wheels/a5/13/75/f811c84a8ab36eedbaef977a6a58a98990e8e0f1967f98f394
Successfully built fasttext
Installing collected packages: pybind11, fasttext
Successfully installed fasttext-0.9.2 pybind11-2.11.1
```

```
pip install gensim
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages (4.3.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from gensim) (1.23.5)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from gensim) (1.11.3)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.10/dist-packages (from gensim) (6.4.0)
```

```
import re
import nltk
import gensim
import itertools
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
from nltk import bigrams
from tensorflow import keras
import matplotlib.pyplot as plt
from nltk.tokenize import word_tokenize
from tensorflow.keras.models import Sequential
from gensim.models import Word2Vec, KeyedVectors
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, Bidirectional, LSTM, Dense
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, HashingVectorizer
```

One Hot Encoding

```
one_hot = pd.get_dummies(teks)
one_hot
```

	!	'	''	,	-buahan	-manggil	-masing	-mutar	.	..	...	yata	yatim	yelamatkan	yet.Mereka	yik	yikan	yir	yuk	yut	z
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
39398	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
39399	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
39400	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
39401	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
39402	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	

39403 rows × 7629 columns

Hash

```
import hashlib
import pandas as pd

# Fungsi untuk melakukan hash vectoring pada teks
def hash_vectoring(text, vector_size):
    vector = [0] * vector_size

    # Konversi teks menjadi hash
    hashed_text = hashlib.sha256(text.encode()).hexdigest()

    # Ambil sebagian dari hash (sesuai dengan panjang vektor)
    hash_subset = hashed_text[:vector_size]

    # Konversi hash menjadi bilangan bulat (integer)
    hash_integer = int(hash_subset, 16)

    # Modulus hash dengan ukuran vektor untuk mendapatkan indeks
    index = hash_integer % vector_size

    # Set nilai indeks vektor menjadi 1
    vector[index] = 1

    return vector

# Membaca data dari file CSV
data_hash = pd.read_csv("text.csv")

# Ukuran vektor
vector_size = 10

# Melakukan hash vectoring untuk setiap teks dalam data CSV
vectors = []
for text in data_hash["text"]:  # Ganti "text" dengan nama kolom teks yang sesuai dalam file CSV
    vector = hash_vectoring(text, vector_size)
    vectors.append(vector)

# Menambahkan vektor ke dalam DataFrame
data_hash["vector"] = vectors

# Menyimpan data hasilnya ke dalam file CSV (jika diperlukan)
data_hash.to_csv("output.csv", index=False)
```

```
# Menampilkan DataFrame dengan vektor
print(data_hash)
```

```
                                                text  \
0  ['MALIN', 'KUNDANG', 'Pada', 'suatu', 'waktu',...

                            vector
0  [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Co-occurance matrix

```
import numpy as np
import nltk
from nltk import bigrams
import itertools
import pandas as pd

# Step 4-2 Create function for co-occurrence matrix
def co_occurrence_matrix(corpus):
    vocab = set(corpus)
    vocab = list(vocab)
    vocab_to_index = {word: i for i, word in enumerate(vocab)}

    # Create bigrams from all words in corpus
    bi_grams = list(bigrams(corpus))

    # Frequency distribution of bigrams ((word1, word2), num_occurrences)
    bigram_freq = nltk.FreqDist(bi_grams).most_common(len(bi_grams))

    # Initialise co-occurrence matrix
    co_occurrence_matrix = np.zeros((len(vocab), len(vocab)))

    # Loop through the bigrams taking the current and previous word,
    # and the number of occurrences of the bigram.
    for bigram in bigram_freq:
        current = bigram[0][1]
        previous = bigram[0][0]
        count = bigram[1]
        pos_current = vocab_to_index[current]
        pos_previous = vocab_to_index[previous]
        co_occurrence_matrix[pos_current][pos_previous] = count

    co_occurrence_matrix = np.matrix(co_occurrence_matrix)

    # Return the matrix and the index
    return co_occurrence_matrix, vocab_to_index

merged = list(itertools.chain.from_iterable(teks))
matrix, vocab_to_index = co_occurrence_matrix(merged)

CoMatrixFinal = pd.DataFrame(matrix, index=vocab_to_index, columns=vocab_to_index)
print(CoMatrixFinal)
```

```
         H       b     z       m       j      I    !      B       g    0  ...  \
H     0.0     0.0   0.0     0.0     0.0    1.0  6.0    0.0     0.0  0.0  ...
b     0.0     1.0   0.0   955.0     0.0   36.0  2.0    0.0   317.0  0.0  ...
z     0.0     0.0   0.0     0.0     0.0    1.0  0.0    0.0     0.0  0.0  ...
m    15.0    12.0   0.0    55.0     0.0    0.0  2.0    0.0   332.0  1.0  ...
j     0.0     0.0   0.0    14.0     0.0    0.0  0.0    0.0    27.0  0.0  ...
..    ...     ...   ...     ...     ...    ...  ...    ...     ...  ...  ...
c     0.0     0.0   0.0     0.0     0.0    0.0  2.0    0.0    57.0  1.0  ...
W     0.0     0.0   0.0     0.0     0.0    1.0  2.0    0.0     0.0  0.0  ...
3     0.0     0.0   0.0     0.0     0.0    0.0  0.0    0.0     1.0  0.0  ...
a   123.0  1556.0  13.0  2099.0  1534.0  228.0  1.0  165.0  2303.0  0.0  ...
y     1.0     2.0   0.0    12.0     0.0    2.0  0.0    3.0    94.0  0.0  ...

         S     G    4     Y    V      c     W    3       a       y
H     0.0   0.0  0.0   0.0  0.0    0.0   0.0  0.0    13.0     0.0
b     0.0   0.0  5.0   0.0  0.0    0.0   0.0  1.0   738.0     2.0
z     0.0   0.0  0.0   0.0  0.0    0.0   0.0  0.0     4.0     0.0
m     0.0   0.0  0.0   0.0  0.0    0.0   0.0  2.0  2727.0     7.0
j     0.0   0.0  0.0   0.0  0.0    0.0   0.0  0.0   659.0     0.0
..    ...   ...  ...   ...  ...    ...   ...  ...     ...     ...
c     0.0   0.0  0.0   0.0  0.0    0.0   0.0  0.0   117.0     0.0
W     0.0   0.0  0.0   0.0  0.0    0.0   0.0  0.0     0.0     0.0
3     0.0   0.0  0.0   0.0  0.0    0.0   0.0  0.0     1.0     0.0
a   221.0  22.0  0.0  31.0  0.0  373.0  96.0  0.0   600.0  3684.0
y     5.0   0.0  0.0   0.0  0.0    0.0   0.0  0.0   700.0     1.0

[66 rows x 66 columns]
```

Word2Vec

```
import pandas as pd
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize

# Baca data dari file CSV
data = pd.read_csv('text.csv')

# Ubah data yang telah ditokenisasi menjadi list kata-kata
tokenized_corpus = [word_tokenize(sentence) for sentence in data['text']]

# Membangun model Word2Vec
model = Word2Vec(tokenized_corpus, vector_size=150, window=5, min_count=1, sg=1)

# Simpan model Word2Vec
model.save("word2vec_model_data1.model")
```

```
model_w2v = Word2Vec.load("word2vec_model_data1.model")
vector = model_w2v.wv["a"]
vector
```

```
array([ 0.06541835, -0.15488496,  0.08822116,  0.19352631, -0.19494219,
        0.05727352,  0.16806684,  0.18550295, -0.27462822, -0.08434363,
        0.08766679,  0.01461103, -0.10631087, -0.03213162, -0.1536769 ,
        0.05065128,  0.1813673 , -0.03502091, -0.0076098 , -0.10072668,
        0.22985263,  0.01443392,  0.05595813,  0.03845434,  0.13520008,
        0.01460027, -0.14815022,  0.1768467 , -0.11273956, -0.23869015,
       -0.15644172,  0.09282051,  0.4366842 , -0.24157026, -0.12444856,
       -0.12189184,  0.18715294, -0.34063217, -0.08438221, -0.40885478,
        0.10753236,  0.12377105, -0.06357777, -0.23069043, -0.08659773,
       -0.04667555, -0.11809205,  0.1251267 , -0.15558544,  0.16864927,
       -0.05158985,  0.0915096 , -0.23363322,  0.01504296,  0.14670219,
       -0.16142713,  0.02071878, -0.20380287, -0.21745738,  0.12084766,
        0.04624774, -0.00348052, -0.05025448, -0.22667857, -0.07381582,
       -0.01871273,  0.19861656,  0.24960053, -0.15663633,  0.03415573,
        0.02499186,  0.07776707, -0.15482832, -0.26916412,  0.08269785,
       -0.07845321,  0.26007557,  0.16349815, -0.13487735, -0.00429137,
       -0.09512714,  0.02587758, -0.00478538,  0.15817565, -0.3100529 ,
        0.0802611 ,  0.2368676 , -0.13675193, -0.19805929,  0.14282425,
       -0.13492149,  0.01255301,  0.07524248,  0.0041525 ,  0.1930431 ,
        0.26915768, -0.08903449, -0.14972055, -0.03566588,  0.21326822,
       -0.18423703, -0.07172637,  0.16111737,  0.07136426,  0.02745449,
       -0.17566799, -0.04134006, -0.13430193, -0.12210789, -0.16063485,
        0.05663039, -0.27130663,  0.1049847 ,  0.1699623 ,  0.03219339,
        0.31932184,  0.23229744,  0.13719311, -0.28516394, -0.166813  ,
        0.12328844,  0.13860749,  0.2578085 , -0.1331749 ,  0.07486887,
        0.19789982, -0.02351444,  0.24322028, -0.10399351,  0.12796938,
        0.3183002 , -0.19557045, -0.05929246, -0.16078842,  0.03355652,
       -0.10598187,  0.07762272,  0.01347402,  0.33051273,  0.16420212,
        0.12199347,  0.01077343, -0.15377568, -0.1264059 ,  0.10917049,
        0.01479815,  0.08791146, -0.02479964,  0.21831124, -0.18538159],
      dtype=float32)
```

Fasttext

```
import fasttext

# Menyimpan contoh teks dalam file teks dengan format label dan isi
with open('text.txt', 'w') as f:
    for sentence in tokenized_corpus:
        label = '__label__text'
        sentence_text = ' '.join(sentence)
        f.write(f'{label} {sentence_text}\n')

# Membuat objek fastText dengan ukuran vektor 100
model = fasttext.train_supervised(input='text.txt', dim=100)

# Mengakses vektor kata tertentu, misalnya "artificial"
vector = model['a']

# Menampilkan vektor
print(vector)
```

```
[-0.00240832 -0.00108073  0.00192764  0.00205566  0.00030977  0.00570119
 -0.0050806  -0.00945939 -0.00838456 -0.00086132  0.00090268 -0.00906155
 -0.00083876 -0.00733031 -0.00068542 -0.00998311 -0.00687865  0.00136836
  0.004911    0.00279719 -0.00156652 -0.00420919  0.00583453  0.00100881
  0.0079287  -0.00651596 -0.00577822 -0.00902392  0.00664047 -0.00944815
```

```
 -0.00389917 -0.00723231 -0.00849627 -0.00616612 -0.00788139  0.00069372
  0.00195264 -0.00838664 -0.00514911  0.0005707  -0.00604592 -0.00359603
 -0.00998078  0.00944954 -0.00371635 -0.00866234  0.00981359 -0.0008056
  0.00282182  0.00193476  0.00744779 -0.00667745  0.00832879 -0.00170781
 -0.00800997 -0.00704154  0.00393407  0.00807225 -0.00824706 -0.00201214
 -0.00999005  0.00221943 -0.00030617  0.00615936 -0.00745515  0.00983361
 -0.00825905  0.00636589 -0.00561563  0.0047792  -0.00297074 -0.00316083
  0.00281324 -0.00666813 -0.00079759 -0.00732927  0.00704691  0.00687735
 -0.00516378  0.00773231 -0.00146494 -0.00601263 -0.00307448 -0.00968319
 -0.00633934 -0.00843898 -0.00993501  0.0041817   0.00919741 -0.00684538
  0.00592998 -0.00811254  0.00974609 -0.0059958   0.00636606 -0.00734903
 -0.00681404 -0.00917669 -0.00562744 -0.00466488]
```

```
from gensim.models import FastText
from gensim.test.utils import common_texts

# Buat model FastText dengan teks contoh
model = FastText(sentences=common_texts, vector_size=100, window=5, min_count=1, sg=1, epochs=10)

# Melakukan training model
model.train(common_texts, total_examples=len(common_texts), epochs=10)

# Mendapatkan vektor kata untuk kata tertentu
word_vector = model.wv['a']
print("Vector for 'a':")
print(word_vector)
```

```
WARNING:gensim.models.word2vec:Effective 'alpha' higher than previous training cycles
Vector for 'a':
[ 0.00172633 -0.00584458 -0.00468104  0.00237787 -0.00961328 -0.00920483
  0.00478313 -0.00023458  0.00289045  0.00763255 -0.00787646  0.0077918
  0.00764943  0.00598061 -0.00802136  0.00744899 -0.0018494   0.00900729
  0.00139307  0.00862116  0.00061596 -0.00438886  0.00555574  0.00916523
 -0.00420381 -0.00430086 -0.00610819  0.00701204 -0.00682714 -0.00296099
  0.00228778  0.0065264   0.00850059  0.00197189  0.00573468  0.00963461
  0.0061604   0.00016779  0.00157166 -0.0035138   0.00087953 -0.00766328
 -0.00768726  0.00513882 -0.00028422  0.00467048 -0.00743055  0.00475188
  0.00564932  0.00469617 -0.00389503  0.00777762 -0.00334777 -0.00096387
  0.00113582  0.00250254  0.00338     0.00239877 -0.00126538 -0.00578689
 -0.00883774 -0.00449621  0.00239474  0.00528135  0.00292266 -0.00359003
 -0.00281631 -0.00784911 -0.00642565 -0.00861917 -0.00893172 -0.00698256
 -0.00712569 -0.00436318 -0.00962574  0.00349577 -0.00654549  0.00469463
 -0.00126562  0.00830105  0.003389   -0.00232524 -0.00385465  0.00688183
 -0.00022094 -0.00808891  0.00722429 -0.00513339 -0.00109037  0.00773508
  0.004378    0.00394941 -0.00120792 -0.00494023  0.00199839 -0.0051124
 -0.008771    0.001927    0.00024278 -0.00806987]
```