



```
!pip install google-play-scraper
```

 Requirement already satisfied: google-play-scraper in /usr/local/lib/python3.10/dist-packages (1.2.6)

```
from google_play_scraper import Sort, reviews
from google_play_scraper import app
import pandas as pd
import numpy as np

result, continuation_token = reviews(
    'com.shopee.id',
    lang='id', #bahasa
    country='id', #negara
    sort=Sort.MOST_RELEVANT, #sorting yg paling relevan
    count=2000, #jumlah dataset
    filter_score_with= None #Isi dengan 1, 2, 3, 4, 5 None jika ingin bercampur
)
```

```
# Dataframe dengan nama
data = pd.DataFrame(np.array(result),columns=['review'])
data = data.join(pd.DataFrame(data.pop('review').tolist()))
data.head()
```




	reviewId	userName	userImage	content	score	th
0	3ad964ff-2ba3-42b8-ba51-2fe14f7437d0	J E GUNARSO PASARIBU	lh.googleusercontent.com/a-/ALV-U...	sudah dua kali lho kurirnya menggagalkan. Kasi...	1	
1	11594b19-8686-4f76-8ea8-fab60d1a33e2	nico rafael	lh.googleusercontent.com/a/ACg8oc...	Aplikasi yg sangat membantu saat membutuhkan b...	5	
2	72b85e99-6607-4018-a639-1c664f0601b1	Resma Fatika	lh.googleusercontent.com/a-/ALV-U...	Jujur dari banyaknya aplikasi belanja, shopee ...	4	
3	4585d7db-41f9-4398-9969-	Suharyanto	lh.googleusercontent.com/a-/ALV-U...	System otomatis "cod" bisa	1	



Langkah berikutnya: [Lihat plot yang direkomendasikan](#)

```
len(data)
```

 199

```
data = data[['content', 'score']]
data.head()
```



	content	score	
0	sudah dua kali lho kurirnya menggagalkan. Kasi...	1	
1	Aplikasi yg sangat membantu saat membutuhkan b...	5	
2	Jujur dari banyaknya aplikasi belanja, shopee ...	4	
3	System otomatis "cod" bisa menjebak pembeli k...	1	
4	Shopee performanya menurun sih kalau menurutku...	4	

Langkah berikutnya: [Lihat plot yang direkomendasikan](#)

Klik dua kali (atau tekan Enter) untuk mengedit

```
data = data.rename(columns={'content': 'ulasan', 'score': 'label'}) #mengganti nama fitur
data.head()
```



	ulasan	label
0	sudah dua kali lho kurirnya menggagalkan. Kasi...	1
1	Aplikasi yg sangat membantu saat membutuhkan b...	5
2	Jujur dari banyaknya aplikasi belanja, shopee ...	4
3	System otomatis "cod" bisa menjebak pembeli k...	1
4	Shopee performanya menurun sih kalau menurutku...	4



0	sudah dua kali lho kurirnya menggagalkan. Kasi...	1
1	Aplikasi yg sangat membantu saat membutuhkan b...	5
2	Jujur dari banyaknya aplikasi belanja, shopee ...	4
3	System otomatis "cod" bisa menjebak pembeli k...	1
4	Shopee performanya menurun sih kalau menurutku...	4



Langkah berikutnya: [Lihat plot yang direkomendasikan](#)

```
data.to_csv("Ulasan Shopee 2000 Data.csv", index = False , encoding='utf-8')
```

```
dataShopee = pd.read_csv('/content/Ulasan Shopee 2000 Data.csv')
dataShopee.head()
```



	ulasan	label
0	sudah dua kali lho kurirnya menggagalkan. Kasi...	1
1	Aplikasi yg sangat membantu saat membutuhkan b...	5
2	Jujur dari banyaknya aplikasi belanja, shopee ...	4
3	System otomatis "cod" bisa menjebak pembeli k...	1
4	Shopee performanya menurun sih kalau menurutku...	4



0	sudah dua kali lho kurirnya menggagalkan. Kasi...	1
1	Aplikasi yg sangat membantu saat membutuhkan b...	5
2	Jujur dari banyaknya aplikasi belanja, shopee ...	4
3	System otomatis "cod" bisa menjebak pembeli k...	1
4	Shopee performanya menurun sih kalau menurutku...	4



Langkah berikutnya: [Lihat plot yang direkomendasikan](#)

```
# Import libraries yang diperlukan
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Contoh data sederhana
data = {
    'login_count': [5, 50, 15, 3, 100, 25],
    'login_from_foreign_location': [0, 1, 0, 0, 1, 1],
    'is_fake': [0, 1, 0, 0, 1, 1]
}

# Buat DataFrame
df = pd.DataFrame(data)

# Tentukan variabel independen dan dependen
X = df[['login_count', 'login_from_foreign_location']]
y = df['is_fake']

# Pisahkan data menjadi data latihan dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Buat model logistic regression
model = LogisticRegression()

# Latih model
model.fit(X_train, y_train)

# Lakukan prediksi
predictions = model.predict(X_test)

# Hitung akurasi
accuracy = accuracy_score(y_test, predictions)
print(f'Akurasi Model: {accuracy}')
```



Akurasi Model: 1.0

```
# Import pustaka yang diperlukan
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix

# Contoh data (gantikanlah dengan data nyata Anda)
data = {
    'review': [
        "barang bagus",
        "kualitas ok",
        "tidak bagus barang!",
        "oke juga",
        "bagus barang",
        "Purely a disappointment, seems like paid reviews."
    ],
    'is_fake': [0, 1, 0, 1, 0, 1]
}

# Buat DataFrame
df = pd.DataFrame(data)

# Pisahkan data menjadi fitur dan target
X = df['review']
y = df['is_fake']

# Bagi data menjadi training dan testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Vectorize teks
vectorizer = TfidfVectorizer(stop_words='english')
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Buat model Naive Bayes
model = MultinomialNB()

# Latih model dengan data training
model.fit(X_train_tfidf, y_train)

# Prediksi data testing
predictions = model.predict(X_test_tfidf)

# Evaluasi model
accuracy = accuracy_score(y_test, predictions)
conf_matrix = confusion_matrix(y_test, predictions)

print(f'Akurasi Model: {accuracy}')
print(f'Matriks Konfusi: \n{conf_matrix}')
```

```
➦ Akurasi Model: 0.5
Matriks Konfusi:
[[1 0]
 [1 0]]
```

```
# Import pustaka yang diperlukan
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')

# Contoh data: Anda harus mengganti ini dengan dataset nyata Anda
data = {
    'review': ["Love this product, highly recommend!", "Total scam, hate it!", "Five stars, will buy again!", "Fake product reviews eve", "Total scam, hate it!"],
    'label': [0, 1, 0, 1, 0] # 0 = Genuine, 1 = Fake
}

df = pd.DataFrame(data)

# Memisahkan data menjadi fitur dan label
X = df['review']
y = df['label']

# Pembagian dataset menjadi training dan testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Vectorization menggunakan TF-IDF
tfidf_vectorizer = TfidfVectorizer(stop_words=stopwords.words('english'))
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Pelatihan model menggunakan Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_tfidf, y_train)

# Prediksi dan evaluasi
predictions = rf_classifier.predict(X_test_tfidf)
print("Classification Report:")
print(classification_report(y_test, predictions))
print("Confusion Matrix:")
print(confusion_matrix(y_test, predictions))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
Classification Report:
              precision    recall  f1-score   support

         0       0.50      1.00      0.67         1
         1       0.00      0.00      0.00         1

 accuracy          0.25
 macro avg          0.25      0.50      0.33         2
 weighted avg          0.25      0.50      0.33         2

Confusion Matrix:
[[1 0]
 [1 0]]
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
_warn_prf(average, modifier, msg_start, len(result))
```

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# URL dari halaman yang ingin Anda scrape, ubah ini dengan halaman yang spesifik
url = 'https://www.bukalapak.com/reviews/handphone/hp-smartphone/feature-phone/4h6mnaj-jual-hape-handphone-nokia-106-hp-jadul-baru-new-d'

# Mengirim HTTP request
response = requests.get(url)

# Membuat objek BeautifulSoup
soup = BeautifulSoup(response.text, 'html.parser')

# Mendefinisikan list untuk menyimpan data
reviews = []

# Mencari elemen yang mengandung ulasan
for review in soup.find_all('div', class_='review_class'): # ubah 'div' dan 'class_' dengan nilai yang sesuai
```

```

author = review.find('p', class_='author_class').text # ubah sesuai struktur halaman
content = review.find('p', class_='content_class').text # ubah sesuai struktur halaman
rating = review.find('span', class_='rating_class').text # ubah sesuai struktur halaman
reviews.append({'author': author, 'content': content, 'rating': rating})

# Menyimpan data ke dalam DataFrame dan kemudian ke CSV
df = pd.DataFrame(reviews)
df.to_csv('reviews.csv', index=False)

# Install and load required packages
install.packages("tensorflow")
library(tensorflow)

# Load required libraries
library(keras)

# Load and preprocess data
data <- read.csv("/content/reviews.csv") # Mengganti "fake_reviews_data.csv" dengan nama file data yang sebenarnya
X <- as.matrix(data[, -ncol(data)]) # Memisahkan fitur
y <- as.numeric(data[, ncol(data)]) # Memisahkan label
X <- scale(X) # Skala fitur ke rentang 0-1

# Split data menjadi data latih dan data uji
set.seed(123) # Untuk hasil yang dapat direproduksi
index <- sample(1:nrow(data), 0.8 * nrow(data))
X_train <- X[index, ]
y_train <- y[index]
X_test <- X[-index, ]
y_test <- y[-index]

# Build GAN model
generator <- keras_model_sequential() %>%
  layer_dense(units = 128, input_shape = dim(X)[2]) %>%
  layer_activation_leaky_relu(alpha = 0.01) %>%
  layer_dense(units = 64) %>%
  layer_activation_leaky_relu(alpha = 0.01) %>%
  layer_dense(units = 1, activation = "sigmoid")
discriminator <- keras_model_sequential() %>%
  layer_dense(units = 64, input_shape = dim(X)[2]) %>%
  layer_activation_leaky_relu(alpha = 0.01) %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 32) %>%
  layer_activation_leaky_relu(alpha = 0.01) %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 1, activation = "sigmoid")

# Compile discriminator
discriminator %>% compile(
  loss = "binary_crossentropy",
  optimizer = optimizer_adam(),
  metrics = "accuracy"
)

# GAN
gan <- keras_model_sequential()
gan %>% add(generator)
gan %>% add(discriminator)
discriminator$trainable <- FALSE
gan %>% compile(
  loss = "binary_crossentropy",
  optimizer = optimizer_adam(),
  metrics = "accuracy"
)

# Training
epochs <- 100
batch_size <- 64
for (epoch in 1:epochs) {
  cat("Epoch:", epoch, "\n")
  # Training discriminator
  real_indices <- sample(1:nrow(X_train), batch_size)
  real_samples <- X_train[real_indices, ]
  fake_samples <- predict(generator, matrix(rnorm(batch_size * dim(X_train)[2]), nrow = batch_size))
  X <- rbind(real_samples, fake_samples)
  y_dis <- c(rep(0.9, batch_size), rep(0, batch_size))
  ..

```