1. Collect Data

1.1 Install newspaper3k

```
[1]  pip install newspaper3k
```

Digunakan untuk mengekstrak dan menguraikan informasi dari artikel dan berita yang ditemukan di situs web berita dan sumber berita lainnya.

1.2 Collect Data Article

```python
from newspaper import Article

# URL berita yang ingin Anda ambil
url = 'https://www.cnnindonesia.com/gaya-hidup/20231103082041-255-1019437/hati-hati-cacar-monyet-disebut-bisa-menular-le

# Inisialisasi objek Artikel
article = Article(url, 'id')  # Ganti 'id' dengan kode bahasa yang sesuai jika tidak bahasa Indonesia

# Mengunduh dan mengurai artikel
article.download()
article.parse()

# Menghilangkan teks iklan (misalnya dengan regex)
cleaned_text = re.sub(r'\biklan\b', '', article.text, flags=re.IGNORECASE)

# Menambahkan informasi media dan kategori berita
media = "CNN Indonesia"  # Ganti dengan sumber media yang sesuai
kategori = "Kesehatan"  # Ganti dengan kategori berita yang sesuai

# Membuat DataFrame Pandas untuk data baru
data = {
    'Teks': [cleaned_text],
    'Media': [media],
    'Label': [kategori]
}

# Membaca data yang sudah ada dari file Excel
existing_data = pd.read_excel('Data_Berita.xlsx')

# Menggabungkan data baru dengan data yang sudah ada
combined_data = pd.concat([existing_data, pd.DataFrame(data)], ignore_index=True)

# Simpan DataFrame yang telah digabung ke dalam file Excel
combined_data.to_excel('Data_Berita.xlsx', index=False)
```

Data diambil menggunakan URL berita tersebut. Data berita yang diambil merupakan berita dari Indonesia sehingga harus diinisiasikan terlebih dahulu. Menambahkan informasi sumber media yang diambil dan ketegori berita yang diambil untuk mempermudah labeling saat data sudah dimasukan ke dalam excel. Kemudian agar data yang baru bisa digabungkan dengan data yang sebelumnya menggunakan pd.concat(). Excel yang ingin digunakan untuk menampung data berita yang di-scraping harus disiapkan terlebih dahulu.

2. Cleaning Data

2.1 Cleaning Data from Advertisement

```python
def text_Clean(df):
    # Salin DataFrame yang masuk agar tidak mengubah DataFrame asli
    df_cleaned = df.copy()

    # Membersihkan teks dalam kolom "Teks"
    df_cleaned['Teks'] = df_cleaned['Teks'].str.replace('\n', '')
    df_cleaned['Teks'] = df_cleaned['Teks'].str.replace('ADVERTISEMENT', '')
    df_cleaned['Teks'] = df_cleaned['Teks'].str.replace('SCROLL TO RESUME CONTENT', '')
    df_cleaned['Teks'] = df_cleaned['Teks'].str.replace('SCROLL TO CONTINUE WITH CONTENT', '')
    df_cleaned['Teks'] = df_cleaned['Teks'].str.replace('Jakarta, CNBC Indonesia -', '')
    df_cleaned['Teks'] = df_cleaned['Teks'].str.replace('KOMPAS.com -', '')

    return df_cleaned

# Contoh penggunaan fungsi
df = pd.read_excel('Data_Berita.xlsx')
df_cleaned = text_Clean(df)

# Tampilkan DataFrame yang telah dibersihkan
print(df_cleaned)
```

Pada setiap berita di internet biasanya terdapat iklan di tengah-tengah artikel. Iklan tidak ada hubungannya dengan isi dari berita sehingga harus dihapus. Beberapa teks yang sudah diambil terdapat kalimat yang menunjukkan iklan yang memiliki pola yang serupa. Selain itu, terdapat enter yang harus dihapus agar semuanya tergabung dalam satu paragraf.

3. Case Folding

3.1 Lowercase

```python
df_lowercase=df_cleaned['Teks'].str.lower()
df_lowercase.head()
```

```
0    arsenal tampak lebih solid musim ini dengan da...
1    kevin sanjaya sukamuljo/rahmat hidayat lolos d...
2    marc marquez akan menunggangi ducati di motogp...
3    perhelatan ajang bulutangkis bni sirnas a dki ...
4    suraaya - jelang laga melawan ekuador pada 10 ...
Name: Teks, dtype: object
```

Seluruh teks di dalam data diubah menjadi lowercase.

4. Remove Punctuation

4.1 Remove Punctuation Mark

```
import string
clean_text=df_lowercase.str.translate(str.maketrans('', '', string.punctuation))
clean_text.sample(10)
```

```
19    majelis kehormatan mahkamah konstitusi mkmk me...
2     marc marquez akan menunggangi ducati di motogp...
30    obat antibiotik merupakan senyawa alami yang d...
37    jakarta  cnbc indonesia  pt morula indonesia m...
7      golf adalah olahraga yang identik dengan kala...
24     luhut binsar pandjaitan menteri koordinator b...
32    satgas mpox berikan sejumlah rekomendasi untuk...
16    politikus pdi perjuangan pdip deddy yevry sito...
15    majelis kehormatan mahkamah konstitusi mkmk me...
18    ketua dpp pan saleh partaonan daulay heran lan...
Name: Teks, dtype: object
```

Karena fokus kepada teks sehingga tanda baca harus dihapus. Tanda baca akan mengganggu saat permodelan, oleh karena itu pilihan yang baik jika menghapus seleuruh tanda baca di dalam teks data.

5. Remove Multiple Whitespace

5.1 Remove Multiple Space

```
[7]  df_cleaned['Teks'] = df_cleaned['Teks'].apply(lambda x: re.sub(r'\s+', ' ', x).strip())
```

Seperti tanda baca, kelebihan spasi juga akan mengganggu saat nanti permodelan sehingga harus dihapus.

6. Tokenization

6.1 Tokenize

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

df_cleaned['Teks'] = df_cleaned['Teks'].apply(lambda text: word_tokenize(text))
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Mempartisi kata demi kata menjadi sebuah token di dalam list. Hal ini bertujuan untuk mempermudah untuk proses selanjutnya seperti remove stop word dan stemming.

7. Remove Stop Word

7.1 Stop Word Removals

```
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
list_stopwords=set(stopwords.words('indonesian', 'english'))
```

```
# Menghapus stopwords dari setiap teks dalam kolom "Teks"
df_cleaned['Teks'] = df_cleaned['Teks'].apply(lambda tokens: [word for word in tokens if word not in list_stopwords])
```

Stop word adalah kata yang sering muncul tetapi tidak memiliki makna spesifik seperti kata lainnya sehingga harus dihapus karena saat di permodelan tidak akan terlalu mempengaruhi bahkan bisa menurunkan akurasi.

8. Stemming

8.1 Install Sastrawi

```
pip install Sastrawi
```

```
Requirement already satisf
```

Karena teks berita yang diambil merupakan berita Indonesia sehingga kalimat pada berita berisi bahasa Indonesia.

8.2 Sastawi for Stemming

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# Inisialisasi stemmer dari Sastrawi
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# Melakukan stemming pada setiap teks dalam kolom "Teks"
df_cleaned['Teks'] = df_cleaned['Teks'].apply(lambda tokens: [stemmer.stem(word) for word in tokens])
```

Pada Stemming akan mengubah kata-kata yang sudah di tokenize menjadi kata baku. Penggunaan Sastrawi sangat membantu untuk mengubah kata-kata bahasa Indonesia kembali ke dalam bentuk bakunya.

9. Mengembalikan Teks

9.1 Cleaned Text

```
# Menggabungkan list token dalam kolom "Teks" kembali menjadi string
df_cleaned['Teks'] = df_cleaned['Teks'].apply(lambda tokens: ' '.join(tokens))
```

Teks yang sudah dibersihkan dari stop word kemudian sudah dikembalikan dalam bentuk baku, tetapi masih dalam bentuk token. Oleh karena itu, kita harus mengembalikan teks yang dalam bentuk token menjadi teks sebagaimana mulanya dalam bentuk paragraf.

10. Label Encoding

10. 1 Encoding

```python
from sklearn.preprocessing import LabelEncoder

# Inisialisasi LabelEncoder
label_encoder = LabelEncoder()

# Melakukan encoding pada kolom "Kategori"
df_cleaned['Label'] = label_encoder.fit_transform(df_cleaned['Label'])
```

Saat nanti masuk permodelan feature target harus dalam berbentuk data numerik. Sehingga kategori “Olahraga”, “Hiburan”, dan “Politik” harus kita berikan label dalam bentuk numerik. Hiburan diberi label “0”, Olahraga diberi label “1”, dan Politik diberi label “2”.

11. Split Data

11.1 Split Train Test Data

```python
from sklearn.model_selection import train_test_split

X = df_cleaned['Teks']
y = df_cleaned['Label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fokus dari permodelan adalah Teks dan Label, sehingga media tidak ikut masuk dalam training dan testing data. Saya membagi data menjadi 80% train dan 20% test dengan random_state = 42.

12. Text Representation

12.1 TF-IDF

```python
from sklearn.feature_extraction.text import TfidfVectorizer

# Inisialisasi TF-IDF Vectorizer
tfidf = TfidfVectorizer(max_features=50, min_df=3)

# Melakukan fit dan transform pada data teks dalam kolom "Teks" pada data pelatihan
tfidf_train = tfidf.fit_transform(X_train)

# Melakukan transform pada data teks dalam kolom "Teks" pada data pengujian
tfidf_test = tfidf.transform(X_test)
```

TF-IDF berperan untuk mengambil informasi untuk menilai pentingnya sebuah kata dalam sebuah teks dengan melihat berapa banyak frekuensi kata yang muncul. semakin banyak kata yang muncul maka bobot atau skor semakin besar.

12.2 CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer

# Inisialisasi CountVectorizer
cv = CountVectorizer(max_features=50, min_df=3)

# Melakukan fit dan transform pada data teks dalam kolom "Teks" pada data pelatihan
cv_train = cv.fit_transform(X_train)

# Melakukan transform pada data teks dalam kolom "Teks" pada data pengujian
cv_test = cv.transform(X_test)
```

Sama seperti TF-IDF, CountVectorizer menghitung seberapa banyak frekuensi kata yang muncul. Pada CountVerctorizer akan menghasilkan matriks di mana setiap kolom akan mewakili kata pada setiap teks. Nilai pada matriks ada jumlah kemunculan kata.

Perbedaan signifikan dari kedua text representation tersebut adalah CountVectorizer hanya mengukur frekuensi kemunculan kata-katadalam teks, sedangkan TF-IDF memberikan bobot yang memperhitungkan pentingnya kata-kata dalam teks.

13. Modeling

13.1 SVC with TF-IDF

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
svm_model = SVC()
svm_model.fit(tfidf_train, y_train)

# Melakukan prediksi pada data pengujian
y_pred = svm_model.predict(tfidf_test)

# Menampilkan akurasi dan laporan klasifikasi
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi SVM:", accuracy)
print(classification_report(y_test, y_pred))
```

```
Akurasi SVM: 0.7777777777777778
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         4
           1       0.60      1.00      0.75         3
           2       1.00      1.00      1.00         2

    accuracy                           0.78         9
   macro avg       0.87      0.83      0.81         9
weighted avg       0.87      0.78      0.77         9
```

13.2 RF with TF-IDF

```
from sklearn.ensemble import RandomForestClassifier

# Inisialisasi dan melatih model Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(tfidf_train, y_train)

# Melakukan prediksi pada data pengujian
y_pred_rf = rf_model.predict(tfidf_test)

# Menampilkan akurasi dan laporan klasifikasi
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Akurasi Random Forest:", accuracy_rf)
print(classification_report(y_test, y_pred_rf))
```

```
Akurasi Random Forest: 0.7777777777777778
              precision    recall  f1-score   support

           0       0.67      1.00      0.80         4
           1       1.00      0.67      0.80         3
           2       1.00      0.50      0.67         2

    accuracy                           0.78         9
   macro avg       0.89      0.72      0.76         9
weighted avg       0.85      0.78      0.77         9
```

13.3 SVC with CountVectorizer

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
svm_model = SVC()
svm_model.fit(cv_train, y_train)

# Melakukan prediksi pada data pengujian
y_pred = svm_model.predict(cv_test)

# Menampilkan akurasi dan laporan klasifikasi
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi SVM:", accuracy)
print(classification_report(y_test, y_pred))
```

```
Akurasi SVM: 0.3333333333333333
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         4
           1       0.33      1.00      0.50         3
           2       0.00      0.00      0.00         2

    accuracy                           0.33         9
   macro avg       0.11      0.33      0.17         9
weighted avg       0.11      0.33      0.17         9
```

13.4 RF with CountVectorizer

```python
from sklearn.ensemble import RandomForestClassifier

# Inisialisasi dan melatih model Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(cv_train, y_train)

# Melakukan prediksi pada data pengujian
y_pred_rf = rf_model.predict(cv_test)

# Menampilkan akurasi dan laporan klasifikasi
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Akurasi Random Forest:", accuracy_rf)
print(classification_report(y_test, y_pred_rf))
```

```
Akurasi Random Forest: 0.6666666666666666
              precision    recall  f1-score   support

           0       0.57      1.00      0.73         4
           1       1.00      0.33      0.50         3
           2       1.00      0.50      0.67         2

    accuracy                           0.67         9
   macro avg       0.86      0.61      0.63         9
weighted avg       0.81      0.67      0.64         9
```

14. Evaluation

Hasil permodelan akan berubah jika komposisi dari train test data berubah. jika komposisi train 70% dan test 30% maka hasil akurasi yang dikeluarkan akan lebih rendah. Hal ini karena komposisi dari train berkurang sehingga permodelan kurang belajar. Selain itu perobahan random_state juga mempengaruhi akurasi permodelan. Alasan kenapa akurasi ada yang tinggi dan rendah adalah karena data yang digunakan tidak beragam sehingga permodelan akan sangat mudah untuk mempelajari dan menebak label dari teks tersebut dan sebaliknya.