Bab 6. Pertemuan 6 - Organisasi dan Analisis Dataset

Hal-hal yang dibahas pada Bab ini:

1. Pengenalan feature engineering.
2. Teknik data augmentation.
3. Metode normalisasi data.
4. Penggunaan ensemble methods.
5. Integrasi dataset dari berbagai sumber.

6.1. Feature Engineering: definisi, teknik, dan implementasi

Feature engineering adalah proses di mana kita menggunakan pengetahuan domain untuk membuat fitur-fitur baru yang lebih informatif dari data mentah (raw data) yang ada. Tujuannya adalah untuk meningkatkan kualitas data yang digunakan dalam model machine learning, sehingga model dapat membuat prediksi yang lebih akurat.

Teknik Feature Engineering:

1. Penggabungan Fitur (Feature Combination): Menggabungkan beberapa fitur untuk menciptakan fitur baru yang lebih informatif. Contohnya, dalam data geospasial, bisa digabungkan koordinat latitude dan longitude menjadi fitur jarak atau fitur lain yang lebih bermakna.
2. Pengurangan Dimensi (Dimensionality Reduction): Mengurangi jumlah fitur dalam dataset dengan teknik seperti Principal Component Analysis (PCA) atau t-SNE untuk menghilangkan fitur yang kurang relevan atau redundan.
3. Penambahan Informasi (Feature Augmentation): Menambahkan informasi tambahan ke dalam dataset, misalnya menambahkan fitur tanggal menjadi hari kerja atau hari libur, yang dapat memberikan wawasan tambahan pada model.
4. Transformasi Fitur (Feature Transformation): Mengubah skala atau distribusi dari fitur untuk memperbaiki kinerja model. Contoh umumnya adalah normalisasi atau transformasi logaritmik.
5. Seleksi Fitur (Feature Selection): Memilih subset fitur yang paling penting atau relevan untuk meningkatkan performa model. Teknik seperti chi-square, information gain, atau recursive feature elimination (RFE) digunakan untuk seleksi fitur.
6. Encoding Kategori (Category Encoding): Mengubah variabel kategori menjadi bentuk yang dapat dimengerti oleh model, seperti menggunakan teknik one-hot encoding atau label encoding.

```
# Implementasi:
# Berikut adalah contoh implementasi sederhana beberapa teknik feature engineering
menggunakan Python dan scikit-learn:
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Contoh dataset
data = {
    'City': ['New York', 'Los Angeles', 'Chicago', 'Boston', 'Miami'],
    'Temperature': [25, 30, 22, 18, 28],
    'Category': ['A', 'B', 'A', 'C', 'B']
}
df = pd.DataFrame(data)

# Contoh 1: Encoding kategori
encoder = OneHotEncoder()
encoded_category = encoder.fit_transform(df[['Category']])

# Contoh 2: Transformasi fitur numerik
scaler = StandardScaler()
scaled_temperature = scaler.fit_transform(df[['Temperature']])

# Contoh 3: Penggabungan fitur
df['City_Temperature'] = df['City'] + '_' + df['Temperature'].astype(str)

# Contoh 4: Pengurangan dimensi dengan PCA
pca = PCA(n_components=1)
pca_result = pca.fit_transform(encoded_category.toarray())

# Contoh 5: Ekstraksi fitur teks dengan TF-IDF
# NOTE: corpus has been modified to have the same number of samples (5) as
df['Category']
corpus = [
    'This is the first document.',
    'This document is the second document.',
    'And this is the third one.',
    'Is this the first document?',
    'This is the fifth document.' # Added a fifth document to match the number
of samples in df['Category']
]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)

# Memisahkan fitur dan label, dan membagi dataset
X_train, X_test, y_train, y_test = train_test_split(X, df['Category'],
test_size=0.2, random_state=42)

# Contoh menggunakan model machine learning setelah feature engineering
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
Accuracy: 0.0
```

Dalam contoh di atas:

a. Kita mulai dengan dataset sederhana yang berisi fitur kategori, numerik, dan teks.

b. Dilakukan encoding kategori menggunakan OneHotEncoder, transformasi skala fitur numerik menggunakan StandardScaler, penggabungan fitur dengan menambahkan fitur baru City_Temperature, dan pengurangan dimensi dengan menggunakan PCA.

c. Fitur teks diekstraksi menggunakan TfidfVectorizer.

d. Setelah melakukan feature engineering, dataset dibagi menjadi data pelatihan dan data pengujian, dan digunakan model RandomForestClassifier untuk memprediksi kategori dari contoh data pengujian.

e. Akurasi dari model diukur menggunakan metrik accuracy.

Feature engineering adalah langkah penting dalam pengembangan model machine learning yang dapat meningkatkan performa model dengan memanfaatkan informasi tambahan atau lebih relevan dari dataset yang tersedia.

6.2. Data Augmentation

Data augmentation adalah teknik yang digunakan untuk meningkatkan variasi dataset dengan membuat salinan data yang dimodifikasi dari data asli. Tujuan utamanya adalah untuk memperluas dataset yang tersedia tanpa mengumpulkan lebih banyak data, yang dapat membantu mengurangi overfitting dan meningkatkan generalisasi model.

Tujuan Data Augmentation

1. Meningkatkan Keanekaragaman Data: Dengan membuat variasi data yang lebih besar, model dapat mempelajari pola yang lebih umum dan lebih baik dalam data yang belum pernah dilihat sebelumnya.

2. Mengurangi Overfitting: Dengan menggunakan augmentasi, kita dapat mengurangi risiko model mempelajari detail yang spesifik dari data pelatihan yang mungkin tidak relevan untuk data baru.

3. Memperbaiki Kinerja Model: Dengan dataset yang lebih besar dan lebih bervariasi, kita dapat meningkatkan performa model dalam tugas-tugas seperti klasifikasi gambar atau teks.

Data Augmentation Gambar

- Rotasi: Memutar gambar dalam berbagai sudut.
- Flip: Membalikkan gambar secara horizontal atau vertikal.
- Zoom: Memperbesar atau memperkecil bagian-bagian gambar.
- Pergeseran: Memindahkan gambar ke arah horizontal atau vertikal.
- Cropping: Memotong bagian dari gambar.

Implementasi menggunakan imgaug (Python):

imgaug adalah pustaka Python yang kuat untuk augmentasi gambar dengan berbagai teknik. Berikut adalah contoh penggunaan imgaug untuk melakukan beberapa teknik augmentasi gambar:

```
import numpy as np
import imgaug.augmenters as iaa
import matplotlib.pyplot as plt
from PIL import Image

# Contoh gambar
image = np.array(Image.open('example_image.jpg'))

# Definisi augmentor
seq = iaa.Sequential([
    iaa.Fliplr(0.5),  # Flip horizontal dengan peluang 50%
    iaa.Affine(rotate=(-10, 10)),  # Rotasi gambar dalam rentang -10 sampai 10
derajat
    iaa.GaussianBlur(sigma=(0, 1.0))  # Blur Gaussian dengan sigma antara 0 dan
1.0
])

# Augmentasi gambar
augmented_image = seq(image=image)

# Tampilkan hasil augmentasi
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(image)
plt.title('Gambar Asli')

plt.subplot(1, 2, 2)
plt.imshow(augmented_image)
plt.title('Gambar Setelah Augmentasi')

plt.tight_layout()
plt.show()
```

Data Augmentation Teks

a. Pertambahan Kata (Word Insertion): Menambahkan kata-kata baru ke dalam teks.

b. Penggantian Kata (Word Replacement): Mengganti kata-kata dengan sinonim atau kata-kata yang serupa.

c. Pemotongan Kata (Word Truncation): Memotong atau menghapus kata-kata dari teks.

d. Pengacakan Urutan Kata (Word Reordering): Mengacak urutan kata dalam teks.

Implementasi menggunakan TextBlob (Python):

TextBlob adalah pustaka Python yang memudahkan analisis teks dengan menyediakan akses mudah ke operasi linguistik. Berikut adalah contoh penggunaan TextBlob untuk beberapa teknik augmentasi teks:

```
from textblob import TextBlob
import random
import nltk
nltk.download('punkt')

# Contoh teks
text = "Ini adalah contoh kalimat untuk augmentasi teks."

# Objek TextBlob untuk teks
blob = TextBlob(text)

# Pertambahan kata
augmented_text = blob.words + ['baru', 'kata']

# Penggantian kata acak
for i in range(len(blob.words)):
    if random.random() < 0.3:  # Probabilitas 30% untuk penggantian kata
        augmented_text[i] = 'kata_baru'

# Konversi kembali ke teks
augmented_text = ' '.join(augmented_text)

print("Teks Asli:", text)
print("Teks Setelah Augmentasi:", augmented_text)
```

Output:

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
Teks Asli: Ini adalah contoh kalimat untuk augmentasi teks.
Teks Setelah Augmentasi: Ini kata_baru contoh kalimat untuk augmentasi teks baru
kata
```

6.3. Normalisasi Data

Normalisasi data adalah proses mengubah nilai-nilai dalam dataset sehingga memiliki skala yang seragam. Tujuannya adalah untuk menghindari dominasi oleh fitur-fitur dengan skala besar dan memastikan setiap fitur memiliki kontribusi yang seimbang terhadap hasil akhir model.

Implementasi menggunakan Scikit-Learn

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Contoh dataset
data = np.array([[1.0, 2.0, 3.0],
                 [4.0, 5.0, 6.0],
                 [7.0, 8.0, 9.0]])

# 1. Min-Max Scaling (Normalization)
scaler_minmax = MinMaxScaler()
data_minmax_scaled = scaler_minmax.fit_transform(data)

print("Min-Max Scaled Data:")
print(data_minmax_scaled)
print()

# 2. Z-Score Normalization (Standardization)
scaler_standard = StandardScaler()
data_standard_scaled = scaler_standard.fit_transform(data)

print("Z-Score Standardized Data:")
print(data_standard_scaled)
```

Output:

```
Min-Max Scaled Data:
[[0.  0.  0. ]
 [0.5 0.5 0.5]
 [1.  1.  1. ]]

Z-Score Standardized Data:
[[-1.22474487 -1.22474487 -1.22474487]
 [ 0.          0.          0.        ]
 [ 1.22474487  1.22474487  1.22474487]]
```

Dalam contoh di atas, data_minmax_scaled dan data_standard_scaled adalah hasil dari normalisasi data menggunakan Min-Max Scaling dan Z-Score Normalization, masing-masing. Hasilnya dapat digunakan sebagai input untuk model machine learning untuk memastikan setiap fitur memiliki pengaruh yang setara terhadap hasil akhir model. Dengan menggunakan teknik normalisasi, kita dapat mempersiapkan data dengan cara yang optimal untuk berbagai jenis model machine learning, meningkatkan interpretabilitas, konvergensi, dan performa model secara keseluruhan.

6.4. Ensemble Methods

Ensemble Methods adalah teknik dalam machine learning di mana beberapa model (yang disebut "learners" atau "base models") digabungkan bersama untuk meningkatkan kinerja prediksi secara keseluruhan. Ide dasarnya adalah bahwa gabungan dari beberapa model yang berbeda sering kali lebih baik daripada model individu yang digunakan secara terpisah.

Jenis Ensemble Methods

1. Bagging (Bootstrap Aggregating): Menggunakan beberapa dataset bootstrap (sampel acak dengan penggantian dari dataset pelatihan) untuk melatih beberapa model serentak. Contoh: Random Forest.
2. Boosting: Mengurangi bias model dengan memusatkan perhatian pada data yang salah diperkirakan oleh model sebelumnya. Contoh: AdaBoost, Gradient Boosting Machines (GBM), XGBoost, LightGBM
3. Stacking (Stacked Generalization): Menggabungkan output dari beberapa model berbeda sebagai input untuk model meta-learner (model yang lebih tinggi). Contoh: Menggunakan hasil prediksi dari SVM, Random Forest, dan Neural Network sebagai input untuk model klasifikasi logistic regression.

Implementasi Ensemble Methods

Berikut adalah contoh implementasi ensemble methods menggunakan Python dengan scikit-learn:

```
# Contoh dengan Random Forest (Bagging):
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh dataset
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)

# Bagi dataset menjadi data pelatihan dan data pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inisialisasi dan latih model Random Forest
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Prediksi menggunakan model yang sudah dilatih
y_pred = clf.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
Accuracy: 0.9
```

```
# Contoh dengan AdaBoost (Boosting)
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh dataset
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)

# Bagi dataset menjadi data pelatihan dan data pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Inisialisasi dan latih model AdaBoost
clf = AdaBoostClassifier(n_estimators=50, random_state=42)
clf.fit(X_train, y_train)

# Prediksi menggunakan model yang sudah dilatih
y_pred = clf.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
/usr/local/lib/python3.10/dist-
packages/sklearn/ensemble/_weight_boosting.py:527: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
Accuracy: 0.87
```

Keunggulan Ensemble Methods:

1. Meningkatkan Kinerja: Ensemble methods sering kali menghasilkan performa yang lebih baik daripada model tunggal karena mampu mengurangi varians (overfitting) dan meningkatkan akurasi prediksi.
2. Stabilitas: Mengurangi risiko kesalahan yang disebabkan oleh variasi dalam dataset pelatihan atau noise.
3. Fleksibilitas: Dapat digunakan dengan berbagai jenis model dasar, memungkinkan untuk mengkombinasikan kekuatan dari berbagai pendekatan pemodelan.

6.5. Identifikasi Pola dan Tren

Identifikasi pola dan tren merupakan bagian penting dalam analisis data, terutama dalam konteks analisis time series atau dataset yang mengandung data kronologis. Pola dan tren mencerminkan perilaku atau perubahan dalam data dari waktu ke waktu, yang dapat memberikan wawasan berharga untuk pengambilan keputusan atau prediksi masa depan. Berikut adalah cara umum untuk mengidentifikasi pola dan tren dalam data.

Identifikasi Pola

1. Visualisasi Data: Gunakan grafik seperti line plot, scatter plot, atau histogram untuk memvisualisasikan data. Pola dapat terlihat sebagai pengelompokan data, pola siklus, atau pola lain yang menunjukkan hubungan antara variabel.
2. Analisis Deskriptif: Hitung statistik deskriptif seperti mean, median, dan mode untuk memahami distribusi data. Pola bisa terlihat sebagai pola konsentrasi nilai di sekitar nilai tengah atau pola distribusi yang asimetris.
3. Analisis Cluster: Gunakan teknik clustering seperti k-means untuk mengidentifikasi kelompok data yang serupa. Pola mungkin muncul sebagai kelompok data yang terpisah dengan karakteristik yang berbeda-beda.

Identifikasi Tren

1. Trendline: Buat trendline atau garis tren menggunakan metode seperti regresi linier. Tren dapat terlihat sebagai pola umum dari data yang menunjukkan arah pergerakan yang jelas ke atas (tren naik), ke bawah (tren turun), atau datar (tren stabil).
2. Moving Average: Gunakan moving average untuk meratakan fluktuasi jangka pendek dan menyoroti tren jangka panjang. Tren bisa terlihat sebagai kecenderungan nilai rata-rata yang berubah dari waktu ke waktu.
3. Decomposition: Lakukan dekomposisi time series untuk memisahkan data menjadi komponen tren, musiman, dan residual. Tren dapat terlihat sebagai komponen yang menunjukkan perubahan sistematis dari waktu ke waktu.

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose

# Load data from CSV
data = pd.read_csv('contoh-data.csv', parse_dates=['date'], index_col='date')

# Visualize data with line plot
plt.figure(figsize=(10, 6))
plt.plot(data.index, data['value'], marker='o', linestyle='-')
plt.title('Line Plot of Time Series Data')
plt.xlabel('Date')
```

```
plt.ylabel('Value')
plt.grid(True)
plt.show()

# Decompose time series data
result = seasonal_decompose(data['value'], model='additive', period=1)

# Visualize trend component
plt.figure(figsize=(10, 6))
plt.plot(result.trend)
plt.title('Trend Component')
plt.xlabel('Date')
plt.ylabel('Trend Value')
plt.grid(True)
plt.show()
```

Dengan menjalankan script ini, Anda akan melihat visualisasi pola dan tren dari data yang ada dalam data.csv. Pastikan untuk menginstal library yang diperlukan seperti Pandas, Matplotlib, dan statsmodels jika belum terinstal di lingkungan Python Anda (pip install pandas matplotlib statsmodels).

Contoh Kasus NLP: Analisis Sentimen Sederhana

Kita akan membuat sebuah contoh kasus di mana kita akan melakukan analisis sentimen terhadap ulasan film. Kita akan menggunakan dataset yang sudah disediakan oleh NLTK dan menghitung nilai positif atau negatif dari setiap ulasan.

```
!pip install nltk
```

Output:

```
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) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-
packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk) (4.66.6)
```

```
import nltk
from nltk.corpus import movie_reviews
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Download dataset movie_reviews jika belum ada
```

```
nltk.download('movie_reviews')
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('vader_lexicon')

# Ambil dataset ulasan film dari NLTK
reviews = []
for fileid in movie_reviews.fileids():
    review = movie_reviews.raw(fileid)
    reviews.append(review)

# Ambil contoh ulasan
sample_review = reviews[0]

# Tokenisasi kata-kata dalam ulasan
tokens = word_tokenize(sample_review)

# Hilangkan stop words (kata-kata umum yang tidak memiliki makna penting)
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]

# Analisis sentimen menggunakan VADER (Valence Aware Dictionary and sEntiment
Reasoner)
sid = SentimentIntensityAnalyzer()
sentiment_score = sid.polarity_scores(sample_review)

# Tampilkan hasil
print("Contoh Ulasan Film:")
print(sample_review)
print("\nTokenisasi Kata-kata:")
print(tokens[:20])  # Tampilkan 20 token pertama
print("\nTokenisasi Kata-kata setelah filtering Stop Words:")
print(filtered_tokens[:20])  # Tampilkan 20 token pertama setelah filtering stop
words
print("\nAnalisis Sentimen:")
print(sentiment_score)
```

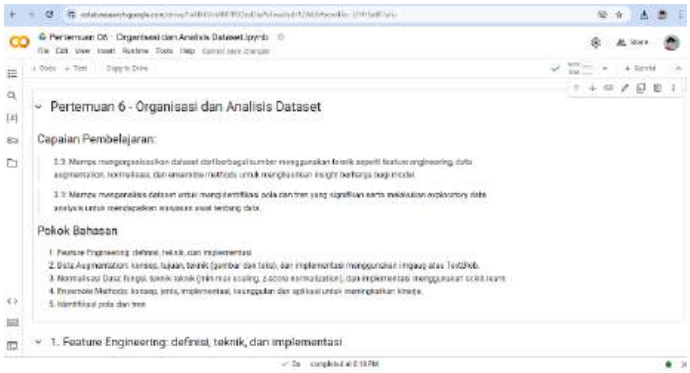
Output:

```
Contoh Ulasan Film:
plot : two teen couples go to a church party , drink and then drive .
they get into an accident .
one of the guys dies , but his girlfriend continues to see him in her life , and
has nightmares .
what's the deal ?
watch the movie and " sorta " find out . . .
critique : a mind-fuck movie for the teen generation that touches on a very cool
idea , but presents it in a very bad package .
which is what makes this review an even harder one to write , since i generally
applaud films which attempt to break the mold , mess with your head and such (
lost highway & memento ) , but there are good and bad ways of making all types
of films , and these folks just didn't snag this one correctly .
they seem to have taken this pretty neat concept , but executed it terribly .
so what are the problems with the movie ?
well , its main problem is that it's simply too jumbled .
it starts off " normal " but then downshifts into this " fantasy " world in which
you , as an audience member , have no idea what's going on .
there are dreams , there are characters coming back from the dead , there are
others who look like the dead , there are strange apparitions , there are
```

```
disappearances , there are a looooot of chase scenes , there are tons of weird
things that happen , and most of it is simply not explained .
now i personally don't mind trying to unravel a film every now and then , but
when all it does is give me the same clue over and over again , i get kind of fed
up after a while , which is this film's biggest problem .
it's obviously got this big secret to hide , but it seems to want to hide it
completely until its final five minutes .
and do they make things entertaining , thrilling or even engaging , in the
meantime ?
not really .
the sad part is that the arrow and i both dig on flicks like this , so we actually
figured most of it out by the half-way point , so all of the strangeness after
that did start to make a little bit of sense , but it still didn't the make the
film all that more entertaining .
i guess the bottom line with movies like this is that you should always make sure
that the audience is " into it " even before they are given the secret password
to enter your world of understanding .
i mean , showing melissa sagemiller running away from visions for about 20 minutes
throughout the movie is just plain lazy ! !
okay , we get it . . . there
are people chasing her and we don't know who they are .
do we really need to see it over and over again ?
how about giving us different scenes offering further insight into all of the
strangeness going down in the movie ?
apparently , the studio took this film away from its director and chopped it up
themselves , and it shows .
there might've been a pretty decent teen mind-fuck movie in here somewhere , but
i guess " the suits " decided that turning it into a music video with little edge
, would make more sense .
the actors are pretty good for the most part , although wes bentley just seemed
to be playing the exact same character that he did in american beauty , only in
a new neighborhood .
but my biggest kudos go out to sagemiller , who holds her own throughout the
entire film , and actually has you feeling her character's unraveling .
overall , the film doesn't stick because it doesn't entertain , it's confusing ,
it rarely excites and it feels pretty redundant for most of its runtime , despite
a pretty cool ending and explanation to all of the craziness that came before it
.
oh , and by the way , this is not a horror or teen slasher flick . . . it's
just packaged to look that way because someone is apparently assuming that the
genre is still hot with the kids .
it also wrapped production two years ago and has been sitting on the shelves ever
since .
whatever . . . skip
it !
where's joblo coming from ?
a nightmare of elm street 3 ( 7/10 ) - blair witch 2 ( 7/10 ) - the crow ( 9/10
) - the crow : salvation ( 4/10 ) - lost highway ( 10/10 ) - memento ( 10/10 ) -
the others ( 9/10 ) - stir of echoes ( 8/10 )


Tokenisasi Kata-kata:
['plot', ':', 'two', 'teen', 'couples', 'go', 'to', 'a', 'church', 'party', ',',
'drink', 'and', 'then', 'drive', '.', 'they', 'get', 'into', 'an']

Tokenisasi Kata-kata setelah filtering Stop Words:
['plot', ':', 'two', 'teen', 'couples', 'go', 'church', 'party', ',', 'drink',
'drive', '.', 'get', 'accident', '.', 'one', 'guys', 'dies', ',', 'girlfriend']

Analisis Sentimen:
{'neg': 0.093, 'neu': 0.762, 'pos': 0.145, 'compound': 0.9924}
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
```

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

Scan disi atau klik gambar di samping untuk mengakses Google Collab pembelajaran



6.6. Perbedaan Overfitting dan Underfitting

Pada kesempatan ini, kita akan membahas konsep penting dalam machine learning, yaitu perbedaan antara overfitting dan underfitting. Kedua istilah ini merujuk pada masalah yang sering dihadapi saat membangun model prediktif. Overfitting terjadi ketika model terlalu kompleks dan terlalu sesuai dengan data pelatihan, sehingga tidak dapat menggeneralisasi dengan baik pada data baru. Sebaliknya, underfitting terjadi ketika model terlalu sederhana untuk menangkap pola yang ada dalam data, sehingga kinerjanya buruk baik pada data pelatihan maupun data baru. Memahami perbedaan antara kedua kondisi ini sangat penting untuk membangun model yang efektif dan akurat. Selanjutnya, kita akan melihat ilustrasi yang menggambarkan perbedaan antara overfitting dan underfitting.

6.7. Ringkasan Natural Language Processing

Pengantar berikut ini akan memberikan gambaran singkat tentang Natural Language Processing (NLP). NLP adalah bidang ilmu komputer dan kecerdasan buatan yang berfokus pada interaksi antara komputer dan bahasa manusia. Dengan menggunakan algoritma dan model matematis, NLP memungkinkan mesin untuk memahami, menganalisis, dan menghasilkan teks dalam bahasa alami. Teknologi ini telah banyak diterapkan dalam berbagai aplikasi, seperti penerjemahan bahasa, analisis sentimen, dan chatbot, yang semakin memudahkan komunikasi antara manusia dan mesin. Mari kita simak ringkasan lebih lanjut tentang NLP.

6.8. Natural Language Processing

Dalam video ini, kita akan menjelaskan konsep Natural Language Processing (NLP) dan menjadikannya salah satu bidang yang paling menarik dalam kecerdasan buatan. NLP merupakan cabang dari machine learning yang berfokus pada interaksi antara komputer dan bahasa manusia. Berbagai tugas yang dapat dilakukan dengan NLP, seperti analisis sentimen, terjemahan otomatis, pengenalan suara, dan pembuatan teks, akan dibahas secara mendetail. Kita juga akan mengulas beberapa teknik dasar dalam NLP, seperti tokenisasi, stemming, dan lemmatization. Selain itu, penggunaan model bahasa seperti Word2Vec dan Transformer akan menjadi sorotan utama dalam video ini. Dengan pemahaman ini, Anda akan lebih siap untuk mengeksplorasi aplikasi NLP dalam berbagai konteks.