```
pip install jcopml
```

```
Requirement already satisfied: pyzmq>=17 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets->jcopml) (24.0
Requirement already satisfied: tornado>=6.1 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets->jcopml) (6
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets->jcopml)
Collecting jedi>=0.16 (from ipython>=4.0.0->ipywidgets->jcopml)
  Downloading jedi-0.19.2-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets->jcopml) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets->jcopml) (0.7.
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from ipython>
Requirement already satisfied: pygments in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets->jcopml) (2.19.1)
Requirement already satisfied: backcall in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets->jcopml) (0.2.0)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets->jcopml) (4.9.
Requirement already satisfied: PyYAML in /usr/local/lib/python3.11/dist-packages (from pyaml>=16.9->scikit-optimize->jcopml) (6.0.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib->jcopml) (1
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.11/dist-packages (from widgetsnbextension~=3.6.0->ipywidgets
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.11/dist-packages (from jedi>=0.16->ipython>=4.0.0->ipywi
Requirement already satisfied: jupyter-core>=4.6.0 in /usr/local/lib/python3.11/dist-packages (from jupyter-client>=6.1.12->ipykernel
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ip
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.
Requirement already satisfied: nbformat in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.11/dist-packages (from pexpect>4.3->ipython>=4.0.0->ipywidge
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0-
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.11/dist-packages (from jupyter-core>=4.6.0->jupyter-client
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.11/dist-packages (from nbclassic>=0.4.7->notebook>=4.4.
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgets
Requirement already satisfied: bleach!=5.0.0 in /usr/local/lib/python3.11/dist-packages (from bleach[css]!=5.0.0->nbconvert>=5->noteb
Requirement already satisfied: defusedxml in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbex
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->wi
Requirement already satisfied: markupsafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widget
Requirement already satisfied: mistune<4,>=2.0.3 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widg
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widget
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->w
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.11/dist-packages (from nbformat->notebook>=4.4.1->widge
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.11/dist-packages (from nbformat->notebook>=4.4.1->widgetsnbe
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.11/dist-packages (from argon2-cffi->notebook>=4.4.1->wi
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from bleach!=5.0.0->bleach[css]!=5.0.0->nbcon
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from bleach[css]!=5.0.0->nbconvert>=5
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbformat->notebook>=4.
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6-
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbformat->notebo
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbformat->notebook>=4
Requirement already satisfied: jupyter-server<3,>=1.8 in /usr/local/lib/python3.11/dist-packages (from notebook-shim>=0.2.3->nbclassi
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from argon2-cffi-bindings->argon2-cffi->notebo
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4->nbconvert>=5->notebook>
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4->nbconvert>=5
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-c
Requirement already satisfied: anyio>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from jupyter-server<3,>=1.8->notebook-shim>=0
Requirement already satisfied: websocket-client in /usr/local/lib/python3.11/dist-packages (from jupyter-server<3,>=1.8->notebook-shi
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio>=3.1.0->jupyter-server<3,>=1.8->noteb
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio>=3.1.0->jupyter-server<3,>=1.8->no
Downloading scikit_optimize-0.10.2-py2.py3-none-any.whl (107 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 107.8/107.8 kB 1.8 MB/s eta 0:00:00
Downloading pyaml-25.1.0-py3-none-any.whl (26 kB)
Downloading jedi-0.19.2-py2.py3-none-any.whl (1.6 MB)
```

```
pip install sastrawi
```

```
Collecting sastrawi
  Downloading Sastrawi-1.0.1-py2.py3-none-any.whl.metadata (909 bytes)
Downloading Sastrawi-1.0.1-py2.py3-none-any.whl (209 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 209.7/209.7 kB 4.0 MB/s eta 0:00:00
Installing collected packages: sastrawi
Successfully installed sastrawi-1.0.1
```

```
pip install emoji
```

```
Collecting emoji
  Downloading emoji-2.14.1-py3-none-any.whl.metadata (5.7 kB)
Downloading emoji-2.14.1-py3-none-any.whl (590 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 590.6/590.6 kB 7.7 MB/s eta 0:00:00
Installing collected packages: emoji
Successfully installed emoji-2.14.1
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from warnings import filterwarnings
filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline

from jcopml.utils import save_model, load_model


from google.colab import drive
drive.mount('/content/drive/')
home_contents = !ls 'drive/My Drive/'
```

```
Mounted at /content/drive/
```

```
fileku = 'drive/My Drive/datasetku.csv'


df = pd.read_csv(fileku, delimiter=";", quotechar='"')
df.head(1189)
```

| | Komen | Rating |
|---|---|---|
| **0** | Buat owner tolong diperbaiki lapangan jaringny... | 2.0 |
| **1** | Lapangan Bagus, lorong dan bench pemaian bersi... | 5.0 |
| **2** | futsalnya enak nyaman | 5.0 |
| **3** | Lapangan bersih. Kamar mandi lumayan. Harga st... | 3.0 |
| **4** | Sayang kipasnya ada yg konslet . Sumuk | 4.0 |
| **...** | ... | ... |
| **1184** | Toilet kurang bersih, parkir roda4 lumayan sul... | 3.0 |
| **1185** | Bersih,nyaman,luas | 5.0 |
| **1186** | Masjidnya bersih dan sangat menyenangkan | 5.0 |
| **1187** | Parkir luas, pelayanan ramah, tersedia foodcur... | 4.0 |
| **1188** | Lumayan | 5.0 |

1189 rows × 2 columns

```
import nltk
nltk.download("stopwords")
nltk.download("punkt")
nltk.download('punkt_tab')
import re, emoji
from string import punctuation
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report
from sklearn.metrics import multilabel_confusion_matrix
```

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

```
sw_indo = stopwords.words('indonesian')
kata_sentimen_penting = [
    'baik', 'buruk', 'bagus', 'jelek', 'mantap', 'parah',
    'puas', 'kecewa', 'senang', 'marah', 'tempat', 'harga',
    'murah', 'mahal', 'pelayanan', 'fasilitas', 'ramah', 'cepat'
]

sw_indo_custom = [word for word in stopwords.words('indonesian') if word not in kata_sentimen_penting]


emoji_pattern = re.compile(
    "["u"\U0001F600-\U0001F64F"  # emoticon
    u"\U0001F300-\U0001F5FF"  # simbol & pictograph
    u"\U0001F680-\U0001F6FF"  # transportasi & simbol
    u"\U0001F1E0-\U0001F1FF"  # bendera
    u"\U00002700-\U000027BF"  # simbol tambahan
    u"\U000024C2-\U0001F251"  # enclosed characters
    "]+", flags=re.UNICODE
)

def is_emoji_only(text):
    text = str(text).strip()
    return bool(text) and not re.sub(emoji_pattern, '', text)


factory = StemmerFactory()
stemmer = factory.create_stemmer()
custom_dict = {
    "mantaf": "mantap",
    "jos": "keren",
    "mantab": "mantap",
    "rekomended": "rekomendasi",
    "bokin": "pesan",
    "booking": "pesan",
    "nyesel": "kecewa",
    "footsal": "futsal",
    "boking": "pesan",
    "plester": "semen",
    "free": "gratis",
    "rosak": "rusak",
    "koplak": "kocak",
    "hareudang": "gerah",
    "sumuk": "gerah",
    "sumpek": "pengap",
    "guyon": "bercanda",
    "betol": "betul",
    "jutek": "ketus",
    "lap.": "lapangan",
    "ujan": "hujan",
    "yg": "yang",
    "dr": "dari",
    "tlg": "tolong",
    "jg": "juga",
    "trs": "terus",
    "krg": "kurang",
    "tp": "tapi",
    "tdk": "tidak",
    "kmr": "kamar",
    "fisilitas": "fasilitas"
    # Tambahkan lebih banyak kata slang sesuai kebutuhan
}

def cleansing(text):
    original_text = text  # simpan salinan awal untuk fallback

    # Normalisasi huruf berulang
    text = re.sub(r'(.)\1{2,}', r'\1', text)

    # Ubah emoji jadi teks
    text = emoji.demojize(str(text))

    # Lowercase
    text = text.lower()

    # Hilangkan URL
    text = re.sub(r'^https?://.*[\r\n]*', '', text, flags=re.MULTILINE)

    # Ganti slang/diksi informal lebih awal
```

```
        for slang, word in custom_dict.items():
            text = text.replace(slang, word)

        # Tokenisasi
        word_list = word_tokenize(text)

        # Stemming
        word_list = [stemmer.stem(word) for word in word_list]

        excluded_words = ["badminton", "voli", "basket", 'nya']

        word_list = [word for word in word_list if word not in excluded_words]

        # Stopword removal
        word_list = [word for word in word_list if word not in sw_indo_custom]

        cleaned_text = ' '.join(word_list)

        # Jika hasil kosong, fallback ke teks awal
        if cleaned_text.strip() == "":
            fallback = re.sub(r'[^\w\s]', '', original_text.lower())  # hilangkan tanda baca
            fallback = re.sub(r'\s+', ' ', fallback).strip()  # rapikan spasi
            return fallback

        return cleaned_text

    df['Komen'] = df.Komen.apply(cleansing)


    df1 = df[['Komen']].copy()


    print(df.head(254))
```

```
                                                 Komen  Rating
    0    owner tolong baik lapang jaring bolong bola la...     2.0
    1    lapang bagus  lorong bench mai bersih  jaring ...     5.0
    2                                futsalnya enak nyaman     5.0
    3    lapang bersih  kamar mandi lumayan  harga stan...     3.0
    4                          sayang kipas konslet  gerah     4.0
    ..                                                 ...     ...
    249  lapang bagus  rumput sintetis  cuman blower ve...     5.0
    250  udah pesan 2 jam wa tulis buku 1 jam  kamar ma...     1.0
    251                                             muas     5.0
    252  harga sewa murah  kamar mandi bersih  ntah krn...     3.0
    253                                      nomor wa gk      5.0

    [254 rows x 2 columns]
```

```
    print(df1.head(254))
```

```
                                                 Komen
    0    owner tolong baik lapang jaring bolong bola la...
    1    lapang bagus  lorong bench mai bersih  jaring ...
    2                                futsalnya enak nyaman
    3    lapang bersih  kamar mandi lumayan  harga stan...
    4                          sayang kipas konslet  gerah
    ..                                                 ...
    249  lapang bagus  rumput sintetis  cuman blower ve...
    250  udah pesan 2 jam wa tulis buku 1 jam  kamar ma...
    251                                             muas
    252  harga sewa murah  kamar mandi bersih  ntah krn...
    253                                      nomor wa gk

    [254 rows x 1 columns]
```

```
    df.head(1189)
```

| | Komen | Rating |
|---|---|---|
| 0 | owner tolong baik lapang jaring bolong bola la... | 2.0 |
| 1 | lapang bagus lorong bench mai bersih jaring ... | 5.0 |
| 2 | futsalnya enak nyaman | 5.0 |
| 3 | lapang bersih kamar mandi lumayan harga stan... | 3.0 |
| 4 | sayang kipas konslet gerah | 4.0 |
| ... | ... | ... |
| 1184 | toilet bersih parkir roda4 lumayan sulit krn ... | 3.0 |
| 1185 | bersih nyaman luas | 5.0 |
| 1186 | masjid bersih senang | 5.0 |
| 1187 | parkir luas layan ramah sedia foodcurt makan... | 4.0 |
| 1188 | lumayan | 5.0 |

1189 rows × 2 columns

Labeling

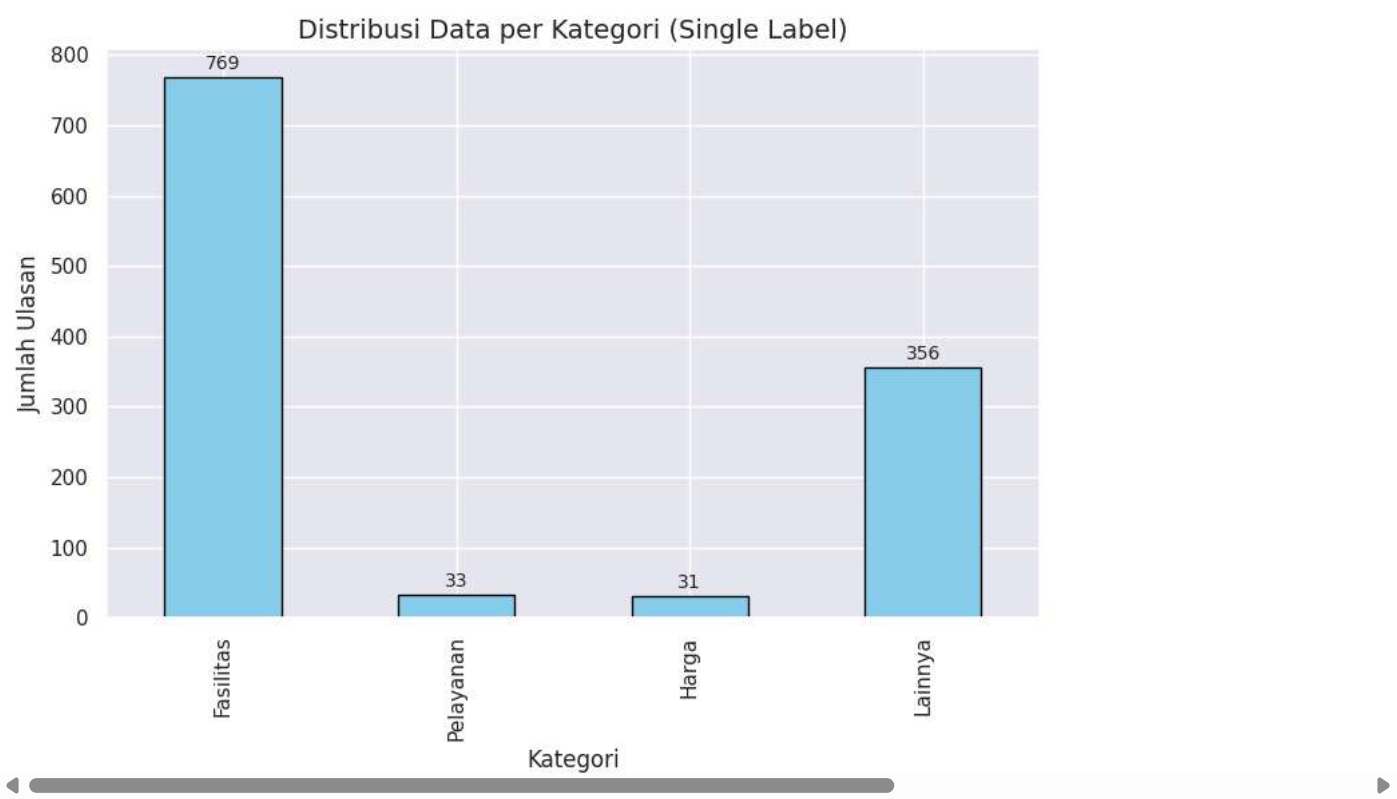
single labeling

```
def single_label_labeling(text):
    counts = {'Fasilitas': 0, 'Pelayanan': 0, 'Harga': 0}

    fasilitas_keywords = ['toilet', 'ventilasi', 'lapang', 'parkir', 'lampu', 'kamar mandi', 'galon',
                          'kipas', 'air', 'bola', 'jaring', 'lantai', 'bersih', 'bagus', 'strategis',
                          'nyaman', 'wangi','tempat','fasilitas','rumput', 'mushola', 'musholla', 'lampu', 'cahaya','listrik']
    pelayanan_keywords = ['staf', 'responsif', 'layanan', 'admin', 'pengurus', 'abang', 'jaga', 'pesan', 'dibook',
                          'pelayan', 'pegawai', 'ramah', 'lucu','pelayanan','manajemen']
    harga_keywords = ['mahal', 'murah', 'jangkau', 'biaya', 'tarif', 'harga', 'sewa']

    for word in fasilitas_keywords:
        counts['Fasilitas'] += text.lower().count(word)

    for word in pelayanan_keywords:
        counts['Pelayanan'] += text.lower().count(word)

    for word in harga_keywords:
        counts['Harga'] += text.lower().count(word)

    if all(value == 0 for value in counts.values()):
        return 'Lainnya'
    else:
        return max(counts, key=counts.get)

df['Single_Label'] = df['Komen'].apply(single_label_labeling)


df[['Komen', 'Single_Label']].head(1189)
```

| | Komen | Single_Label |
|---|---|---|
| 0 | owner tolong baik lapang jaring bolong bola la... | Fasilitas |
| 1 | lapang bagus lorong bench mai bersih jaring ... | Fasilitas |
| 2 | futsalnya enak nyaman | Fasilitas |
| 3 | lapang bersih kamar mandi lumayan harga stan... | Fasilitas |
| 4 | sayang kipas konslet gerah | Fasilitas |
| ... | ... | ... |
| 1184 | toilet bersih parkir roda4 lumayan sulit krn ... | Fasilitas |
| 1185 | bersih nyaman luas | Fasilitas |
| 1186 | masjid bersih senang | Fasilitas |
| 1187 | parkir luas layan ramah sedia foodcurt makan... | Fasilitas |
| 1188 | lumayan | Lainnya |

1189 rows × 2 columns

```
sl = pd.read_csv('drive/My Drive/single_label.csv')


import pandas as pd
import matplotlib.pyplot as plt

# Menghitung jumlah data per kategori
category_counts = sl['Single_Label'].value_counts()

# Menentukan urutan label yang diinginkan
desired_order = ['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya']

# Mengatur ulang indeks sesuai dengan urutan yang diinginkan
category_counts = category_counts.reindex(desired_order)

# Membuat bar chart
plt.figure(figsize=(8, 6))
category_counts.plot(kind='bar', color='skyblue', edgecolor='black')

# Menambahkan judul dan label sumbu
plt.title('Distribusi Data per Kategori (Single Label)', fontsize=14)
plt.xlabel('Kategori', fontsize=12)
plt.ylabel('Jumlah Ulasan', fontsize=12)

# Menampilkan nilai di atas setiap bar
for index, value in enumerate(category_counts):
    plt.text(index, value + 5, str(value), ha='center', va='bottom', fontsize=10)

# Menampilkan grafik
plt.tight_layout()
plt.show()
```




```
print(sl['Komen'].apply(type).value_counts())
# Lihat baris yang bertipe float
print(sl[sl['Komen'].apply(type) == float]['Komen'])
```

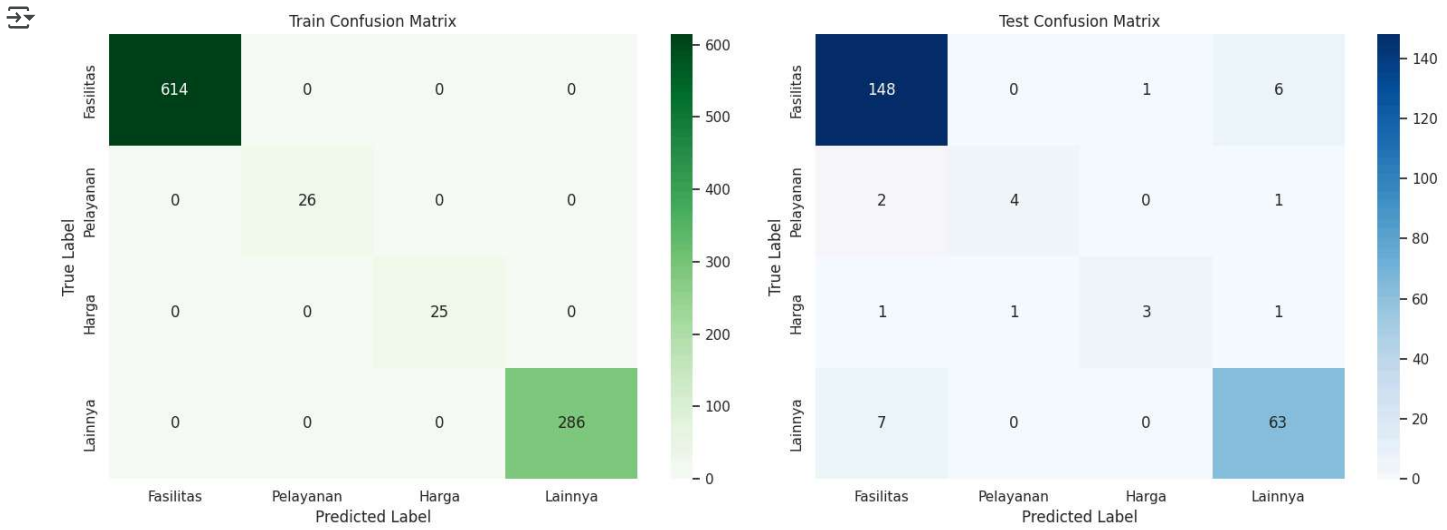
```
Komen
<class 'str'>    1189
Name: count, dtype: int64
Series([], Name: Komen, dtype: object)
```

```
from sklearn.pipeline import Pipeline

from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report, confusion_matrix

# Data

X = sl['Komen']
y = sl['Single_Label']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Pipeline + Grid Search
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('clf', SVC())
])

param_grid = {
    'tfidf__max_df': [0.9, 1.0],
    'tfidf__ngram_range': [(1, 1), (1, 2)],
    'clf__C': [0.1, 1, 10],
    'clf__kernel': ['linear']
}

grid = GridSearchCV(pipeline, param_grid, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)

# Evaluation
y_pred = grid.predict(X_test)
```

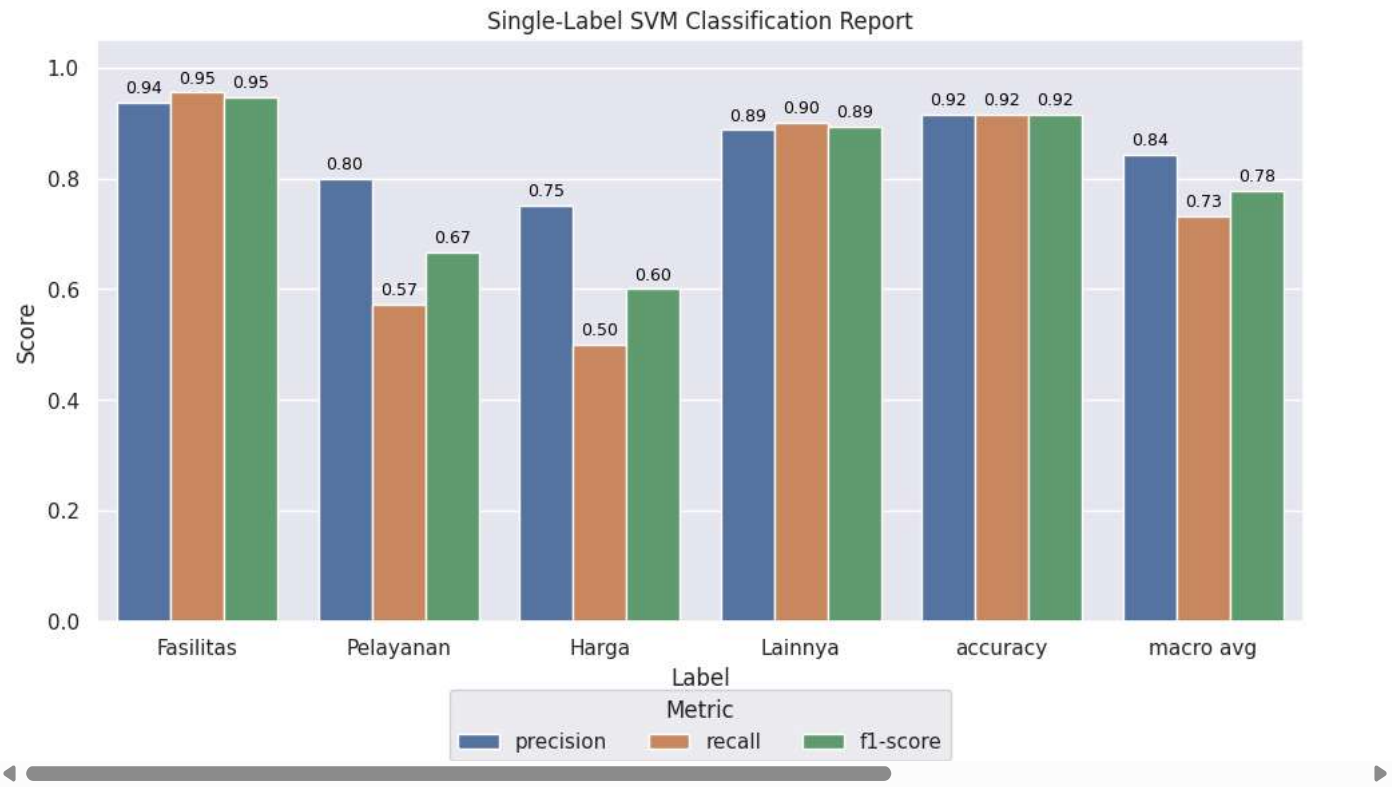
```
# Prediksi untuk training dan testing
y_train_pred = grid.predict(X_train)
y_test_pred = grid.predict(X_test)

# Evaluasi
print("=== SINGLE LABEL CLASSIFICATION REPORT ===")
print("TRAIN REPORT")
print(classification_report(y_train, y_train_pred))
print("TEST REPORT")
print(classification_report(y_test, y_test_pred))

print("Confusion Matrix (Test):")
print(confusion_matrix(y_test, y_test_pred, labels=['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya']))
```

```
=== SINGLE LABEL CLASSIFICATION REPORT ===
TRAIN REPORT
              precision    recall  f1-score   support

   Fasilitas       1.00      1.00      1.00       614
       Harga       1.00      1.00      1.00        25
     Lainnya       1.00      1.00      1.00       286
   Pelayanan       1.00      1.00      1.00        26

    accuracy                           1.00       951
   macro avg       1.00      1.00      1.00       951
weighted avg       1.00      1.00      1.00       951

TEST REPORT
              precision    recall  f1-score   support

   Fasilitas       0.94      0.95      0.95       155
       Harga       0.75      0.50      0.60         6
     Lainnya       0.89      0.90      0.89        70
   Pelayanan       0.80      0.57      0.67         7

    accuracy                           0.92       238
   macro avg       0.84      0.73      0.78       238
weighted avg       0.91      0.92      0.91       238

Confusion Matrix (Test):
[[148   0   1   6]
 [  2   4   0   1]
 [  1   1   3   1]
 [  7   0   0  63]]
```

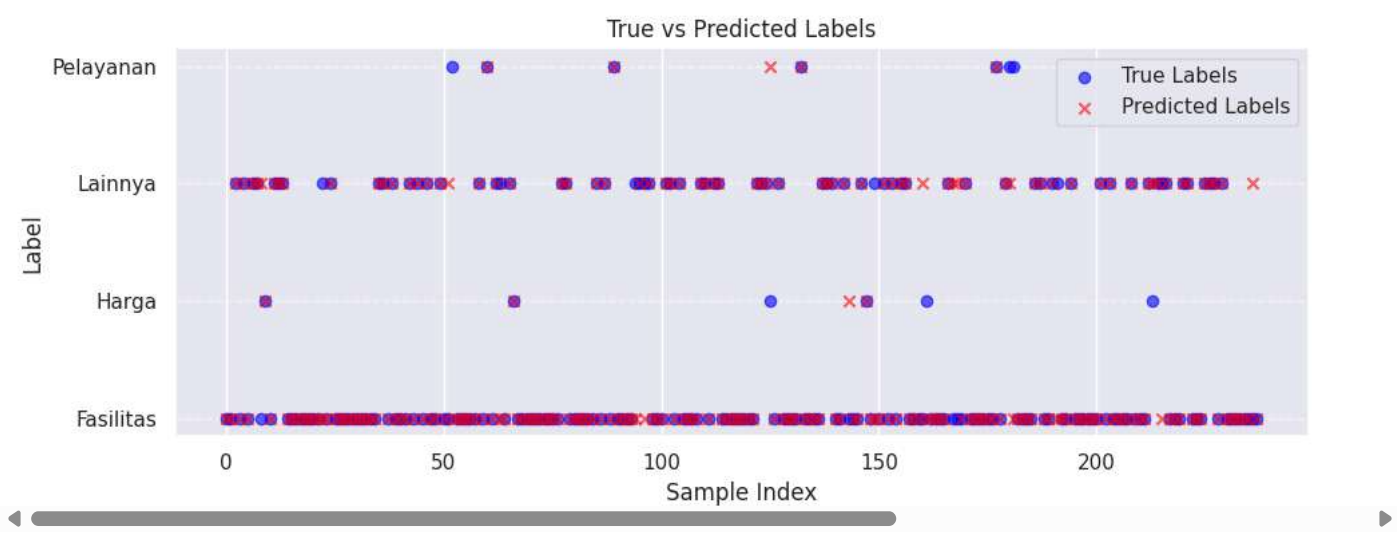
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

labels = ['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya']

# Hitung confusion matrices
cm_train = confusion_matrix(y_train, y_train_pred, labels=labels)
cm_test = confusion_matrix(y_test, y_test_pred, labels=labels)

# Plot keduanya
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

sns.heatmap(cm_train, annot=True, fmt='d', cmap='Greens', xticklabels=labels, yticklabels=labels, ax=axes[0])
axes[0].set_title("Train Confusion Matrix")
axes[0].set_xlabel("Predicted Label")
axes[0].set_ylabel("True Label")

sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels, ax=axes[1])
axes[1].set_title("Test Confusion Matrix")
axes[1].set_xlabel("Predicted Label")
axes[1].set_ylabel("True Label")

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




```
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Generate classification report as dictionary
report_dict = classification_report(y_test, y_test_pred, output_dict=True)

# Konversi ke DataFrame (ambil hanya label yang kamu butuhkan)
df_report = pd.DataFrame(report_dict).transpose()
df_report = df_report.loc[['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya','accuracy','macro avg'], ['precision', 'recall', 'f1-score']]

# Reset index biar bisa dipakai seaborn
df_report = df_report.reset_index().rename(columns={'index': 'Label'})
df_melted = df_report.melt(id_vars='Label', var_name='Metric', value_name='Score')

# Plot chart
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=df_melted, x='Label', y='Score', hue='Metric')

# Tambahkan nilai di atas setiap batang
for p in ax.patches:
    height = p.get_height()
    if height > 0:
        ax.annotate(f'{height:.2f}',
                    (p.get_x() + p.get_width() / 2., height),
                    ha='center', va='bottom',
                    fontsize=9, color='black', xytext=(0, 3),
                    textcoords='offset points')

plt.ylim(0, 1.05)
plt.title('Single-Label SVM Classification Report')
plt.ylabel('Score')
plt.xlabel('Label')
plt.legend(title='Metric')
plt.legend(title='Metric', loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=3)
plt.tight_layout()
plt.show()
```




```
import matplotlib.pyplot as plt
import numpy as np

# Misal label kamu seperti ini (ganti sesuai datamu)
true_labels = y_test  # atau array label asli
pred_labels = y_pred  # hasil prediksi dari model

# Konversi label ke angka jika masih string
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
true_encoded = le.fit_transform(true_labels)
pred_encoded = le.transform(pred_labels)

# Buat scatter plot
plt.figure(figsize=(10, 4))
plt.scatter(range(len(true_encoded)), true_encoded, color='blue', marker='o', label='True Labels', alpha=0.6)
plt.scatter(range(len(pred_encoded)), pred_encoded, color='red', marker='x', label='Predicted Labels', alpha=0.6)

# Set label axis y
plt.yticks(ticks=np.unique(true_encoded), labels=le.inverse_transform(np.unique(true_encoded)))
plt.xlabel("Sample Index")
plt.ylabel("Label")
plt.title("True vs Predicted Labels")
plt.legend(loc='upper right')
plt.grid(axis='y', linestyle='--', alpha=0.5)

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




multi labeling

```
def multi_label_labeling(row):
    text = row['Komen']  # ambil isi komen

    labels = {'Fasilitas': 0, 'Pelayanan': 0, 'Harga': 0}

    fasilitas_keywords = ['toilet', 'ventilasi', 'lapang', 'kondisi', 'parkir', 'lampu', 'kamar mandi', 'galon',
                          'kipas', 'air', 'kualitas', 'bola', 'jaring', 'lantai', 'bersih', 'bagus', 'strategis',
                          'nyaman', 'enak', 'wangi','tempat','fasilitas','rumput', 'mushola', 'musholla', 'lampu', 'cahaya']
    pelayanan_keywords = ['staf', 'responsif', 'layanan', 'admin', 'pengurus', 'abang', 'jaga', 'pesan', 'dibook',
                          'pelayan', 'pegawai', 'ramah', 'lucu','pelayanan','manajemen']
    harga_keywords = ['mahal', 'murah', 'jangkau', 'biaya', 'tarif', 'harga', 'sewa']

    for keyword in fasilitas_keywords:
        if keyword in text.lower():
            labels['Fasilitas'] = 1

    for keyword in pelayanan_keywords:
        if keyword in text.lower():
            labels['Pelayanan'] = 1

    for keyword in harga_keywords:
        if keyword in text.lower():
            labels['Harga'] = 1

    if sum(labels.values()) == 0:
        labels['Lainnya'] = 1
    else:
        labels['Lainnya'] = 0

    return labels
# Apply ke setiap baris
multi_labels = df1.apply(multi_label_labeling, axis=1)

# Convert ke DataFrame
multi_df = pd.DataFrame(multi_labels.tolist())

# Gabungkan
df1 = df1.join(multi_df)

df1.head()
```

| | Komen | Fasilitas | Pelayanan | Harga | Lainnya |
|---|---|---|---|---|---|
| 0 | owner tolong baik lapang jaring bolong bola la... | 1 | 0 | 0 | 0 |
| 1 | lapang bagus lorong bench mai bersih jaring ... | 1 | 0 | 0 | 0 |
| 2 | futsalnya enak nyaman | 1 | 0 | 0 | 0 |
| 3 | lapang bersih kamar mandi lumayan harga stan... | 1 | 0 | 1 | 0 |
| 4 | sayang kipas konslet gerah | 1 | 0 | 0 | 0 |

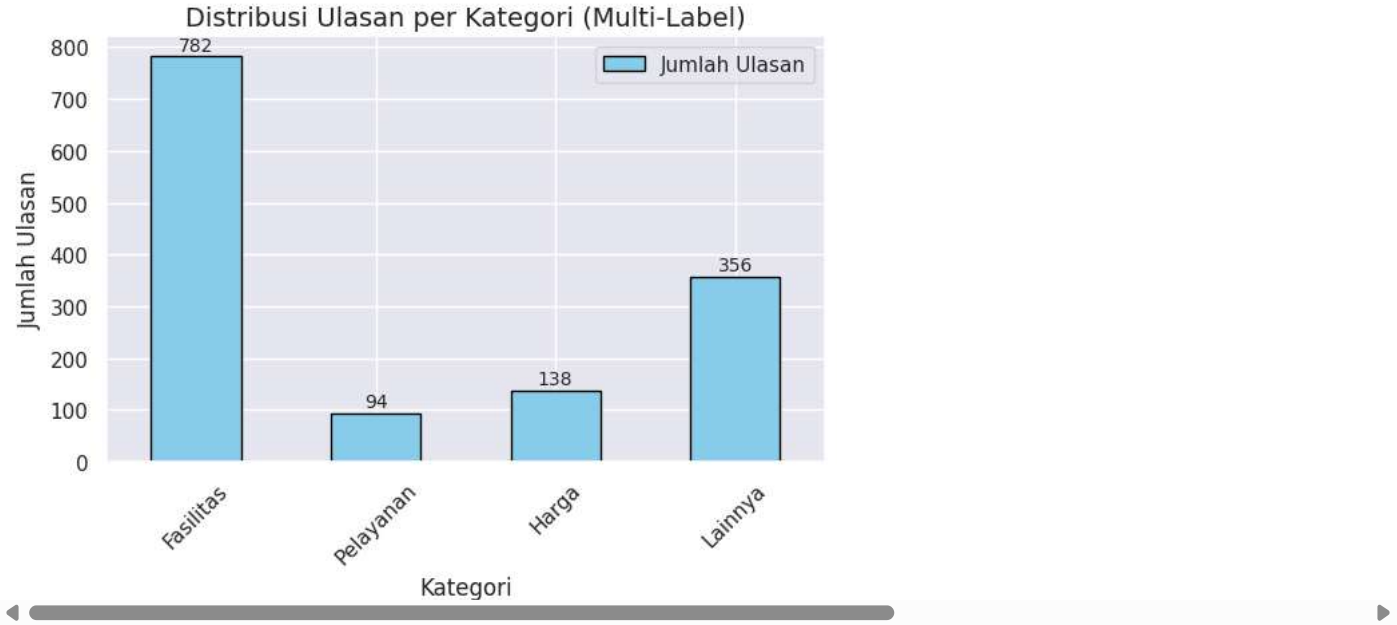
```
df1.head(1189)
```

| | Komen | Fasilitas | Pelayanan | Harga | Lainnya |
|---|---|---|---|---|---|
| 0 | owner tolong baik lapang jaring bolong bola la... | 1 | 0 | 0 | 0 |
| 1 | lapang bagus lorong bench mai bersih jaring ... | 1 | 0 | 0 | 0 |
| 2 | futsalnya enak nyaman | 1 | 0 | 0 | 0 |
| 3 | lapang bersih kamar mandi lumayan harga stan... | 1 | 0 | 1 | 0 |
| 4 | sayang kipas konslet gerah | 1 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... |
| 1184 | toilet bersih parkir roda4 lumayan sulit krn ... | 1 | 0 | 0 | 0 |
| 1185 | bersih nyaman luas | 1 | 0 | 0 | 0 |
| 1186 | masjid bersih senang | 1 | 0 | 0 | 0 |
| 1187 | parkir luas layan ramah sedia foodcurt makan... | 1 | 1 | 0 | 0 |
| 1188 | lumayan | 0 | 0 | 0 | 1 |

1189 rows × 5 columns

```
import pandas as pd
import matplotlib.pyplot as plt

# Hitung jumlah ulasan per kategori
category_counts = df1[['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya']].sum()

# Buat DataFrame untuk plotting
df_plot = pd.DataFrame(category_counts, columns=['Jumlah Ulasan'])

# Buat diagram batang
plt.figure(figsize=(8, 6))
df_plot.plot(kind='bar', stacked=True, color=['skyblue'], edgecolor='black')

# Tambahkan judul dan label
plt.title('Distribusi Ulasan per Kategori (Multi-Label)', fontsize=14)
plt.xlabel('Kategori', fontsize=12)
plt.ylabel('Jumlah Ulasan', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()

for index, value in enumerate(category_counts):
    plt.text(index, value + 5, str(value), ha='center', va='bottom', fontsize=10)
# Tampilkan grafik
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```

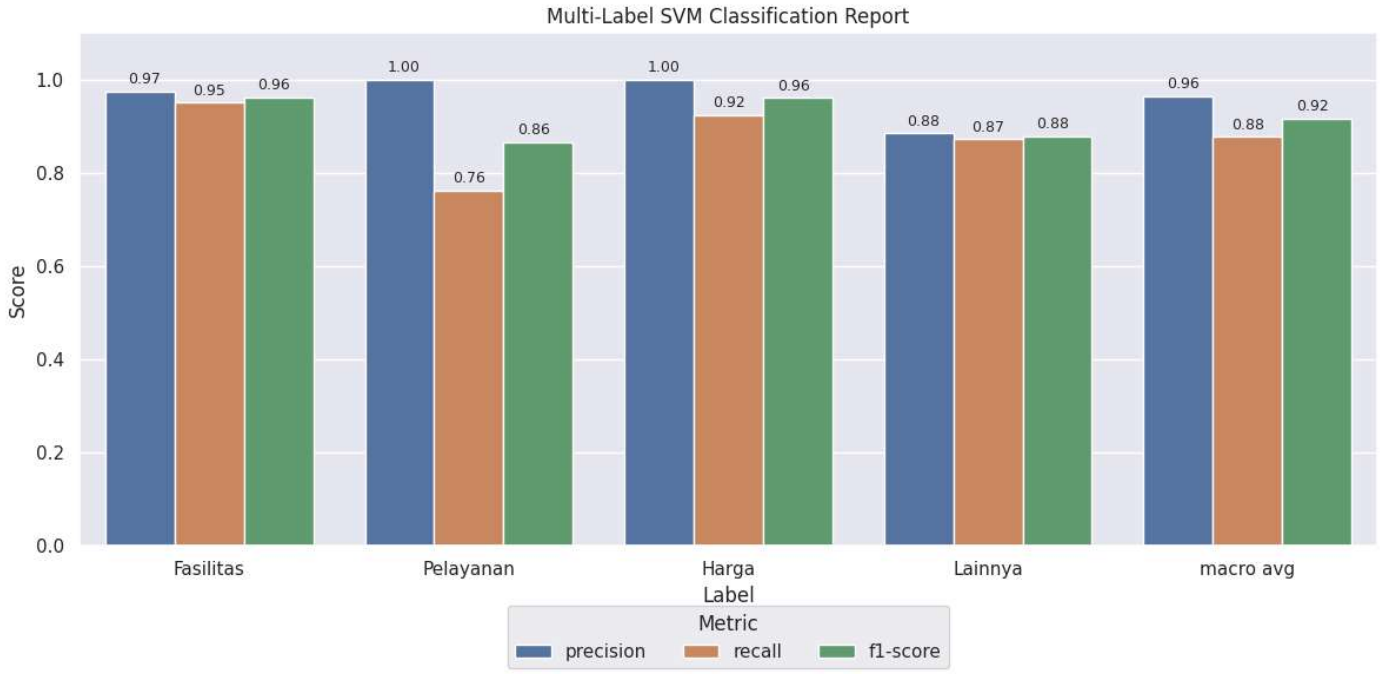



```
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.multiclass import OneVsRestClassifier
from sklearn.feature_extraction.text import TfidfVectorizer

# Pisahkan fitur dan label
X1 = df1['Komen']
y_multi = df1[['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya']]

# Split data ke train dan test
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y_multi, test_size=0.2, random_state=42)

# Pipeline untuk preprocessing dan modeling
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),  # TF-IDF untuk fitur teks
    ('clf', OneVsRestClassifier(SVC()))  # One-vs-Rest Classifier dengan SVM
])

# Grid Search untuk hyperparameter tuning
param_grid = {
    'tfidf__max_df': [0.9, 1.0],  # Filter kata yang terlalu umum
    'tfidf__ngram_range': [(1, 1), (1, 2)],  # N-gram range
    'clf__estimator__C': [0.1, 1, 10],  # Regularisasi SVM
    'clf__estimator__kernel': ['linear']  # Gunakan kernel linear
}

grid = GridSearchCV(pipeline, param_grid, cv=3, n_jobs=-1)
grid.fit(X_train1, y_train1)

# Evaluasi model
y_pred = grid.predict(X_test1)

# Menampilkan classification report
print("\n=== MULTI LABEL CLASSIFICATION REPORT ===")
print("TRAIN REPORT")
print(classification_report(y_train1, grid.predict(X_train1), target_names=['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya']))
print("TEST REPORT")
print(classification_report(y_test1, y_pred, target_names=['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya']))

# Menampilkan confusion matrix per kategori
for i, label in enumerate(['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya']):
    print(f"\nConfusion Matrix for {label}:")
    print(confusion_matrix(y_test1.iloc[:, i], y_pred[:, i]))
```

```
=== MULTI LABEL CLASSIFICATION REPORT ===
```

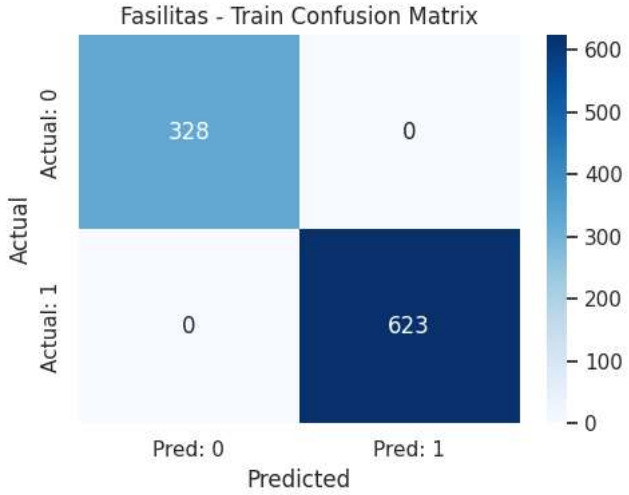
```
TRAIN REPORT
              precision    recall  f1-score   support

   Fasilitas       1.00      1.00      1.00       623
   Pelayanan       1.00      1.00      1.00        73
       Harga       1.00      1.00      1.00       112
     Lainnya       1.00      1.00      1.00       286

   micro avg       1.00      1.00      1.00      1094
   macro avg       1.00      1.00      1.00      1094
weighted avg       1.00      1.00      1.00      1094
 samples avg       1.00      1.00      1.00      1094

TEST REPORT
              precision    recall  f1-score   support

   Fasilitas       0.97      0.95      0.96       159
   Pelayanan       1.00      0.76      0.86        21
       Harga       1.00      0.92      0.96        26
     Lainnya       0.88      0.87      0.88        70

   micro avg       0.95      0.91      0.93       276
   macro avg       0.96      0.88      0.92       276
weighted avg       0.96      0.91      0.93       276
 samples avg       0.93      0.92      0.92       276


Confusion Matrix for Fasilitas:
[[ 75   4]
 [  8 151]]

Confusion Matrix for Pelayanan:
[[217   0]
 [  5  16]]

Confusion Matrix for Harga:
[[212   0]
 [  2  24]]

Confusion Matrix for Lainnya:
[[160   8]
 [  9  61]]
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.metrics import classification_report

# Hasil evaluasi
report = classification_report(
    y_test1, y_pred,
    target_names=['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya'],
    output_dict=True
)

# Buat DataFrame dan filter
df_report = pd.DataFrame(report).T[['precision', 'recall', 'f1-score']].reset_index()
df_report = df_report.rename(columns={'index': 'Label'})

# Hanya tampilkan label kelas dan macro avg
kelas_asli = ['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya', 'macro avg']
df_report = df_report[df_report['Label'].isin(kelas_asli)].reset_index(drop=True)  # reset index!

# Plot
plt.figure(figsize=(12, 6))
df_melted = df_report.melt(id_vars='Label', var_name='Metric', value_name='Score')
sns.barplot(data=df_melted, x='Label', y='Score', hue='Metric')

# Tambahkan label nilai
for i in range(df_report.shape[0]):
    for j, metric in enumerate(['precision', 'recall', 'f1-score']):
        value = df_report.loc[i, metric]
        plt.text(i - 0.25 + j * 0.25, value + 0.01, f"{value:.2f}", ha='center', va='bottom', fontsize=9)

plt.ylim(0, 1.1)
plt.title("Multi-Label SVM Classification Report")
plt.legend(title='Metric', loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=3)
plt.tight_layout()
plt.show()
```

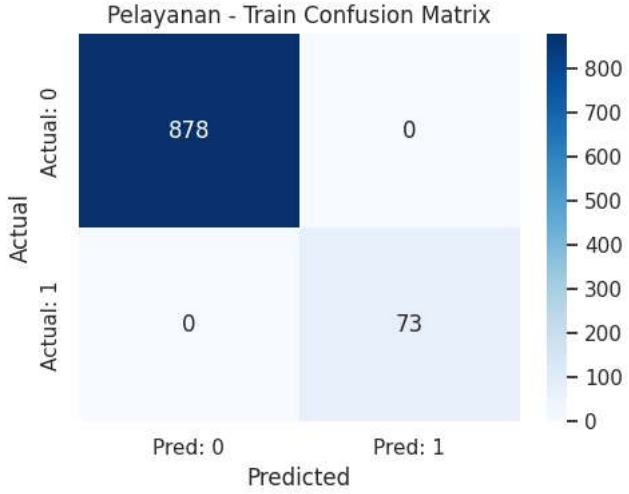



```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

labels = ['Fasilitas', 'Pelayanan', 'Harga', 'Lainnya']

# Fungsi bantu untuk plot heatmap
def plot_cm(cm, label, dataset_type):
    plt.figure(figsize=(5, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Pred: 0', 'Pred: 1'],
                yticklabels=['Actual: 0', 'Actual: 1'])
    plt.title(f'{label} - {dataset_type} Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.tight_layout()
    plt.show()

# Train confusion matrix
y_train_pred = grid.predict(X_train1)
for i, label in enumerate(labels):
    cm_train = confusion_matrix(y_train1.iloc[:, i], y_train_pred[:, i])
    print(f"\nConfusion Matrix (Train) for {label}:\n", cm_train)
    plot_cm(cm_train, label, 'Train')

# Test confusion matrix
for i, label in enumerate(labels):
    cm_test = confusion_matrix(y_test1.iloc[:, i], y_pred[:, i])
    print(f"\nConfusion Matrix (Test) for {label}:\n", cm_test)
    plot_cm(cm_test, label, 'Test')
```

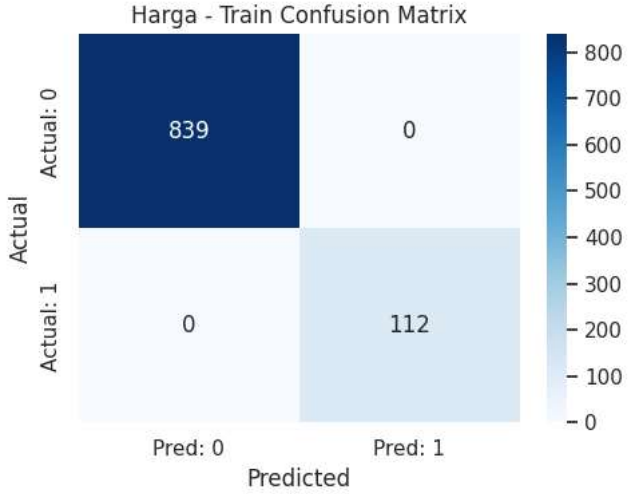
```
Confusion Matrix (Train) for Fasilitas:
 [[328   0]
 [  0 623]]
```



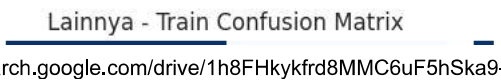

```
Confusion Matrix (Train) for Pelayanan:
 [[878   0]
 [  0  73]]
```

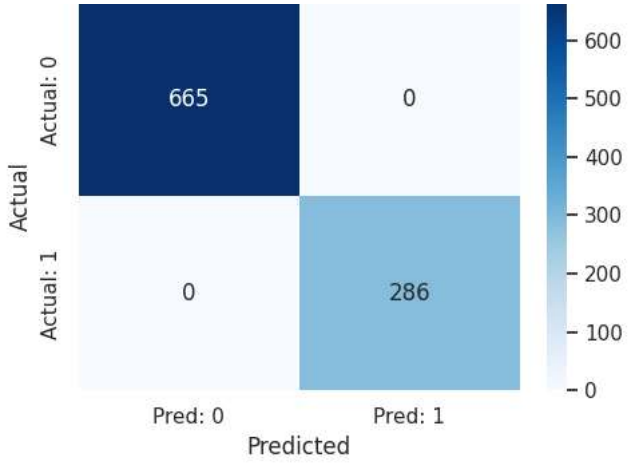



```
Confusion Matrix (Train) for Harga:
 [[839   0]
 [  0 112]]
```

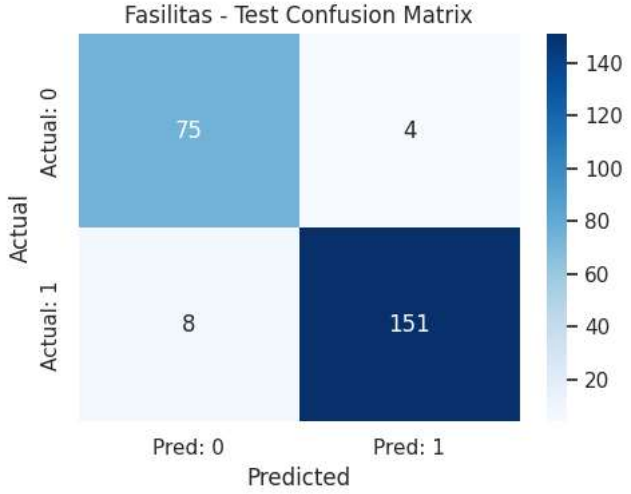



```
Confusion Matrix (Train) for Lainnya:
 [[665   0]
 [  0 286]]
```

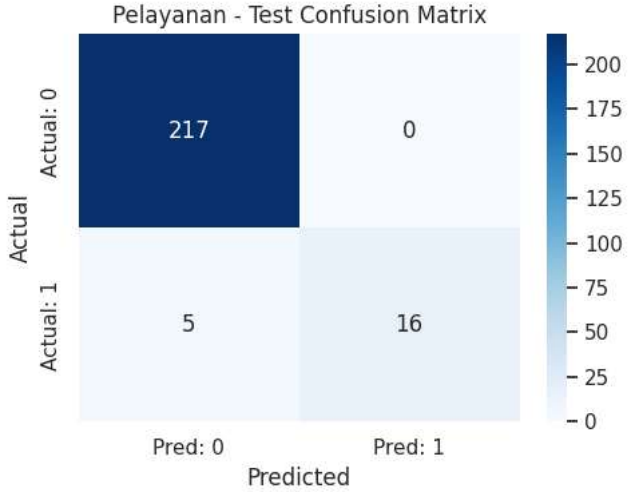




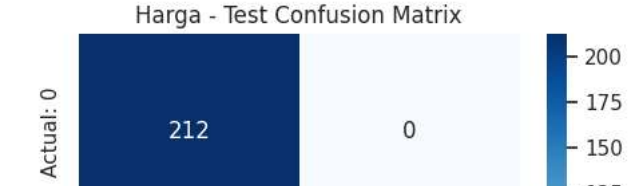

```
Confusion Matrix (Test) for Fasilitas:
 [[ 75   4]
 [  8 151]]
```

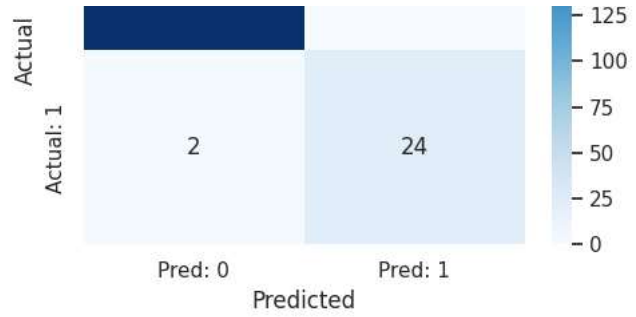



```
Confusion Matrix (Test) for Pelayanan:
 [[217   0]
 [  5  16]]
```

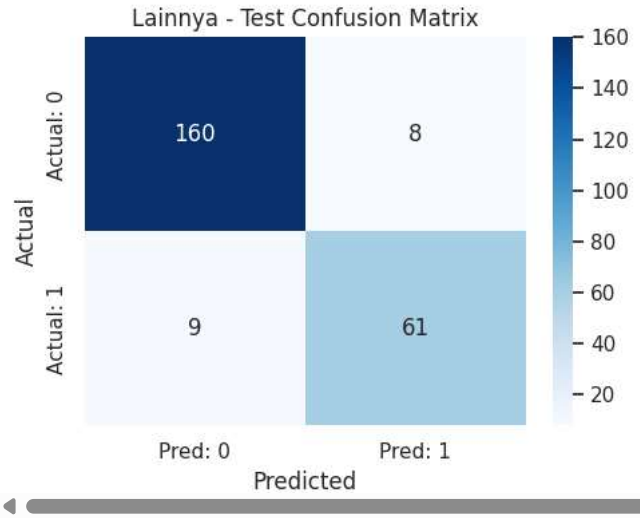



```
Confusion Matrix (Test) for Harga:
 [[212   0]
 [  2  24]]
```

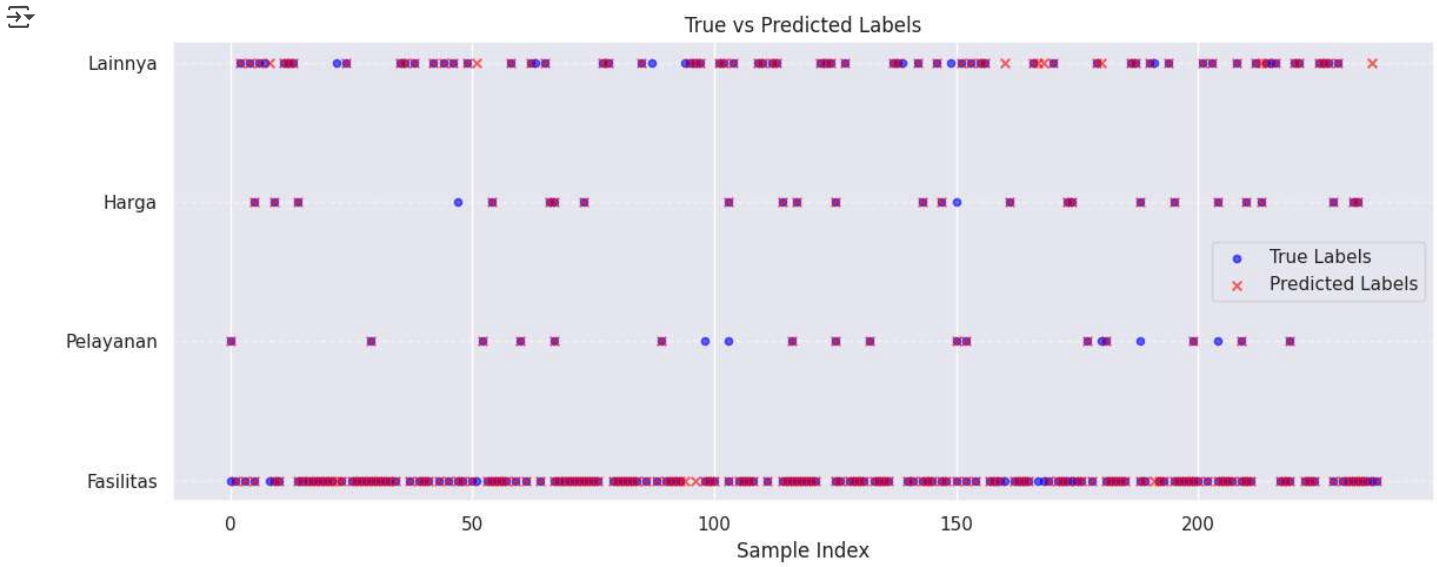





```
Confusion Matrix (Test) for Lainnya:
 [[160   8]
 [  9  61]]
```



```
import matplotlib.pyplot as plt
import numpy as np

# Konversi prediksi ke DataFrame agar sesuai dengan y_test1
y_pred_df = pd.DataFrame(y_pred, columns=y_test1.columns)

# Buat plot
plt.figure(figsize=(12, 5))
label_names = y_test1.columns
num_labels = len(label_names)

for i, label in enumerate(label_names):
    # Y = i untuk label ke-i
    # True label (bulat biru)
    true_idx = y_test1[label] == 1
    plt.scatter(np.where(true_idx)[0], [i]*true_idx.sum(), color='blue', label='True Labels' if i == 0 else "", alpha=0.6, s=20)

    # Predicted label (silang merah)
    pred_idx = y_pred_df[label] == 1
    plt.scatter(np.where(pred_idx)[0], [i]*pred_idx.sum(), color='red', marker='x', label='Predicted Labels' if i == 0 else "", alpha=0.6, s

plt.yticks(ticks=range(num_labels), labels=label_names)
plt.xlabel('Sample Index')
plt.title('True vs Predicted Labels')
plt.legend()
plt.tight_layout()
plt.grid(axis='y', linestyle='--', alpha=0.3)
plt.show()
```




dr beberapa cara yg sudah dicoba, saya akan menggunakan hasil yg paling baru/paling bawah untuk penulisan artikel. yakni single label vs multi label

word cloud

```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.11/dist-packages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.11/dist-packages (from wordcloud) (2.0.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from wordcloud) (11.2.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from wordcloud) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib->wordcloud) (2.9.0.poste
```

```
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1
```

single label

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# List kategori single-label
single_categories = sl['Single_Label'].dropna().unique()

# Loop setiap kategori
for cat in single_categories:
    text = " ".join(sl[sl['Single_Label'] == cat]['Komen'].astype(str))
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)

    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f'Word Cloud - Single Label: {cat}')
    plt.show()
```

Word Cloud - Single Label: Fasilitas