ANALYSIS BIG DATA

NATURAL LANGUAGE PROCESSING

DISUSUN OLEH:

FIQQI AHLUDZIKRI (2111010034)

DOSEN:

Dr. M. SAID HASIBUAN, S.Kom.,M.Kom

FAKULTAS ILMU KOMPUTER

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT INFORMATIKA DAN BISNIS DARMAJAYA

BANDAR LAMPUNG

TAHUN AJARAN 2023/2024

Natural Language Processing Python

1. **Mengimpor Pustaka:**

pythonCopy code

import numpy as np import pandas as pd

Mengimpor pustaka yang diperlukan, NumPy untuk operasi numerik, dan Pandas untuk penanganan data.

2. **Membaca Data:**

pythonCopy code

df = pd.read_csv('sample_data/1_Restaurant_Reviews.tsv', sep='\t', quoting=3)

Membaca file TSV (Tab-Separated Values) yang berisi ulasan restoran ke dalam DataFrame Pandas.

3. **Pra-Pemrosesan Data:**

pythonCopy code

from sklearn.model_selection import train_test_split X = df['Review'] y = df['Liked'] X_train, X_test, y_train, y_test = train_test_split(X, y)

Membagi data menjadi set pelatihan dan pengujian.

4. **Pemrosesan Teks:**

pythonCopy code

import re import nltk nltk.download('stopwords') from nltk.corpus import stopwords from nltk.stem.porter import PorterStemmer

Mengimpor NLTK untuk pemrosesan bahasa alami. Mendefinisikan fungsi **text_process** untuk membersihkan teks, termasuk menghapus karakter non-alfabet, mengonversi ke huruf kecil, menghapus kata-kata penghenti, dan stemming.

5. **Parameter Pencarian Grid untuk Model:**

pythonCopy code

rf_param_grid = { ... } # Parameter untuk model Random Forest nb_param_grid = { ... } # Parameter untuk model Naive Bayes

Menyiapkan grid parameter untuk pencarian grid.

6. **Membuat Pipelines:**

pythonCopy code

rf_pipe = Pipeline([...]) # Pipeline untuk model Random Forest nb_pipe = Pipeline([...]) # Pipeline untuk model Naive Bayes

Membuat pipa untuk pemrosesan teks, termasuk Count Vectorization, transformasi TF-IDF, dan klasifier yang ditentukan (Random Forest atau Naive Bayes).

7. **Pencarian Grid Cross-Validation:**

pythonCopy code

```
rf_grid = GridSearchCV(rf_pipe, rf_param_grid, verbose=2, cv=2) rf_grid.fit(X_train, y_train)
nb_grid = GridSearchCV(nb_pipe, nb_param_grid, verbose=2, cv=2) nb_grid.fit(X_train, y_train)
```

Melakukan pencarian grid cross-validation untuk menemukan parameter terbaik untuk kedua model.

8. **Evaluasi Model:**

pythonCopy code

```
rf_y_pred = rf_grid.predict(X_test) nb_y_pred = nb_grid.predict(X_test)
```

Membuat prediksi pada set pengujian untuk kedua model.

9. **Confusion Matrix dan Metrik:**

pythonCopy code

```
from sklearn import metrics tn, fp, fn, tp = metrics.confusion_matrix(y_test, rf_y_pred).ravel()
print('Confusion matrix:\n', metrics.confusion_matrix(y_test, rf_y_pred)) print('Akurasi:',
metrics.accuracy_score(y_test, rf_y_pred)) # ... (metrik serupa untuk Random Forest)
```

Menghitung dan mencetak berbagai metrik seperti matriks kebingungan, akurasi, presisi, recall, dan nilai F1 untuk kedua model.

10. **Parameter Terbaik:**

pythonCopy code

```
rf_grid.best_params_ nb_grid.best_params_
```

Menampilkan parameter terbaik yang ditemukan selama pencarian grid untuk kedua model.

11. **Importansi Fitur untuk Random Forest:**

pythonCopy code

```
feature_importance = pd.DataFrame(rf_pipe.steps[2][1].feature_importances_,
rf_pipe.steps[0][1].get_feature_names_out(), columns=['importance'])
feature_importance.sort_values('importance', ascending=False).head(20)
```

Melatih model Random Forest dengan parameter terbaik dan mengekstrak importansi fitur.

Skrip ini pada dasarnya melakukan analisis sentimen pada ulasan restoran menggunakan dua model yang berbeda (Random Forest dan Naive Bayes), menyesuaikan hiperparameter menggunakan pencarian grid, mengevaluasi kinerja model, dan memberikan wawasan tentang importansi fitur untuk model Random Forest.

NLP_FIQQI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Importing the libraries

```
[ ] import numpy as np
    import pandas as pd
```

Importing the dataset

```
[ ] df = pd.read_csv('sample_data/1_Restaurant_Reviews.tsv', sep='\t', quoting=3)
```

Splitting the dataset

```
[ ] from sklearn.model_selection import train_test_split
    X = df['Review']
    y = df['Liked']
    X_train, X_test, y_train, y_test = train_test_split(X, y)
```

NLP_FIQQI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Cleaning the text

```
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

def text_process(document):
    document = re.sub('[^a-zA-Z]', ' ', document)
    document = document.lower()
    document = document.split()
    all_stopwords = stopwords.words('english')
    all_stopwords.remove('not')
    document = [word for word in document if not word in set(all_stopwords)]
    ps = PorterStemmer()
    document = [ps.stem(word) for word in document]
    return document
```

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

NLP_FIQQI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Hyperparameter tunning

```
[ ] rf_param_grid = {
        'bag_of_words__ngram_range': [(1, 1)],
        'bag_of_words__max_df': [0.85, 1.0],
        'bag_of_words__min_df': [0.01, 0.05],
        'estimator__criterion': ['gini'],
        'estimator__n_estimators': [100, 300]
    }

    nb_param_grid = {
        'bag_of_words__ngram_range': [(1, 1), (1, 2)],
        'bag_of_words__max_df': [0.85, 1.0],
        'bag_of_words__min_df': [0.01, 0.05],
        'estimator__alpha': [0.01, 1.0]
    }
```

NLP_FIQQI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Creating the pipeline

```
[ ]  from sklearn.pipeline import Pipeline
     from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.feature_extraction.text import TfidfTransformer
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.naive_bayes import MultinomialNB

     rf_pipe = Pipeline([
         ('bag_of_words', CountVectorizer(analyzer=text_process)),
         ('tf_idf', TfidfTransformer()),
         ('estimator', RandomForestClassifier())
     ])

     nb_pipe = Pipeline([
         ('bag_of_words', CountVectorizer(analyzer=text_process)),
         ('tf_idf', TfidfTransformer()),
         ('estimator', MultinomialNB())
     ])
```

NLP_FIQQI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment

+ Code + Text

Fitting

```
from sklearn.model_selection import GridSearchCV

rf_grid = GridSearchCV(rf_pipe, rf_param_grid, verbose=2, cv=2)
rf_grid.fit(X_train, y_train)

nb_grid = GridSearchCV(nb_pipe, nb_param_grid, verbose=2, cv=2)
nb_grid.fit(X_train, y_train)
```

```
Fitting 2 folds for each of 8 candidates, totalling 16 fits
[CV] END bag_of_words__max_df=0.85, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=100; total time=   1.3s
[CV] END bag_of_words__max_df=0.85, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=100; total time=   0.7s
[CV] END bag_of_words__max_df=0.85, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=300; total time=   0.9s
[CV] END bag_of_words__max_df=0.85, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=300; total time=   0.6s
[CV] END bag_of_words__max_df=0.85, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=100; total time=   0.3s
[CV] END bag_of_words__max_df=0.85, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=100; total time=   0.3s
[CV] END bag_of_words__max_df=0.85, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=300; total time=   0.5s
[CV] END bag_of_words__max_df=0.85, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=300; total time=   0.5s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=100; total time=   0.3s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=100; total time=   0.3s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=300; total time=   0.5s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=300; total time=   0.6s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=100; total time=   0.3s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=100; total time=   0.3s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=300; total time=   0.5s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__criterion=gini, estimator__n_estimators=300; total time=   0.5s
Fitting 2 folds for each of 16 candidates, totalling 32 fits
[CV] END bag_of_words__max_df=0.85, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 1), estimator__alpha=0.01; total time=   0.2s
```

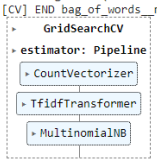
NLP_FIQQI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comme

+ Code + Text

```
  warnings.warn(
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 2), estimator__alpha=0.01; total time=   0.2s
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:544: UserWarning: The parameter 'ngram_range' will not be used since 'analyzer' is callable'
  warnings.warn(
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 2), estimator__alpha=1.0; total time=   0.2s
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:544: UserWarning: The parameter 'ngram_range' will not be used since 'analyzer' is callable'
  warnings.warn(
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.01, bag_of_words__ngram_range=(1, 2), estimator__alpha=1.0; total time=   0.2s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__alpha=0.01; total time=   0.2s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__alpha=0.01; total time=   0.2s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__alpha=1.0; total time=   0.2s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 1), estimator__alpha=1.0; total time=   0.2s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 2), estimator__alpha=0.01; total time=   0.2s
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:544: UserWarning: The parameter 'ngram_range' will not be used since 'analyzer' is callable'
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:544: UserWarning: The parameter 'ngram_range' will not be used since 'analyzer' is callable'
  warnings.warn(
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 2), estimator__alpha=0.01; total time=   0.2s
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 2), estimator__alpha=1.0; total time=   0.2s
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:544: UserWarning: The parameter 'ngram_range' will not be used since 'analyzer' is callable'
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:544: UserWarning: The parameter 'ngram_range' will not be used since 'analyzer' is callable'
  warnings.warn(
[CV] END bag_of_words__max_df=1.0, bag_of_words__min_df=0.05, bag_of_words__ngram_range=(1, 2), estimator__alpha=1.0; total time=   0.2s
```



NLP_FIQQI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Best parameters found

```
[ ] rf_grid.best_params_
```

```
{'bag_of_words__max_df': 0.85,
 'bag_of_words__min_df': 0.01,
 'bag_of_words__ngram_range': (1, 1),
 'estimator__criterion': 'gini',
 'estimator__n_estimators': 100}
```

```
nb_grid.best_params_
```

```
{'bag_of_words__max_df': 0.85,
 'bag_of_words__min_df': 0.01,
 'bag_of_words__ngram_range': (1, 1),
 'estimator__alpha': 1.0}
```

Predicting using the test set

```
[ ] rf_y_pred = rf_grid.predict(X_test)
    nb_y_pred = nb_grid.predict(X_test)
```

NLP_FIQQI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Evaluating

```
from sklearn import metrics
tn, fp, fn, tp = metrics.confusion_matrix(y_test, rf_y_pred).ravel()
print('Confusion matrix:\n', metrics.confusion_matrix(y_test, rf_y_pred))
print('Accuracy:', metrics.accuracy_score(y_test, rf_y_pred))
print('Precision:', metrics.precision_score(y_test, rf_y_pred))
print('Recall:', metrics.recall_score(y_test, rf_y_pred))
print('F1-Score:', metrics.f1_score(y_test, rf_y_pred))
print(metrics.classification_report(y_test, rf_y_pred)) # Better for multiclass problem
```

```
Confusion matrix:
 [[103  15]
 [ 52  80]]
Accuracy: 0.732
Precision: 0.8421052631578947
Recall: 0.6060606060606061
F1-Score: 0.7048458149779736
              precision    recall  f1-score   support

           0       0.66      0.87      0.75       118
           1       0.84      0.61      0.70       132

    accuracy                           0.73       250
   macro avg       0.75      0.74      0.73       250
weighted avg       0.76      0.73      0.73       250
```

NLP_FIQQI.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
from sklearn import metrics
tn, fp, fn, tp = metrics.confusion_matrix(y_test, nb_y_pred).ravel()
print('Confusion matrix:\n', metrics.confusion_matrix(y_test, nb_y_pred))
print('Accuracy:', metrics.accuracy_score(y_test, nb_y_pred))
print('Precision:', metrics.precision_score(y_test, nb_y_pred))
print('Recall:', metrics.recall_score(y_test, nb_y_pred))
print('F1-Score:', metrics.f1_score(y_test, nb_y_pred))
print(metrics.classification_report(y_test, nb_y_pred)) # Better for multiclass problem
```

```
Confusion matrix:
 [[103  15]
 [ 51  81]]
Accuracy: 0.736
Precision: 0.84375
Recall: 0.6136363636363636
F1-Score: 0.7105263157894737
              precision    recall  f1-score   support

           0       0.67      0.87      0.76       118
           1       0.84      0.61      0.71       132

    accuracy                           0.74       250
   macro avg       0.76      0.74      0.73       250
weighted avg       0.76      0.74      0.73       250
```

NLP_FIQQI.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[ ]

```
from sklearn import metrics
tn, fp, fn, tp = metrics.confusion_matrix(y_test, nb_y_pred).ravel()
print('Confusion matrix:\n', metrics.confusion_matrix(y_test, nb_y_pred))
print('Accuracy:', metrics.accuracy_score(y_test, nb_y_pred))
print('Precision:', metrics.precision_score(y_test, nb_y_pred))
print('Recall:', metrics.recall_score(y_test, nb_y_pred))
print('F1-Score:', metrics.f1_score(y_test, nb_y_pred))
print(metrics.classification_report(y_test, nb_y_pred)) # Better for multiclass problem
```

```
Confusion matrix:
 [[103  15]
 [ 51  81]]
Accuracy: 0.736
Precision: 0.84375
Recall: 0.6136363636363636
F1-Score: 0.7105263157894737
              precision    recall  f1-score   support

           0       0.67      0.87      0.76       118
           1       0.84      0.61      0.71       132

    accuracy                           0.74       250
   macro avg       0.76      0.74      0.73       250
weighted avg       0.76      0.74      0.73       250
```

NLP_FIQQI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Getting attributes (if using Random Forest)

```
[ ] rf_grid.best_params_
```

```
{'bag_of_words__max_df': 0.85,
 'bag_of_words__min_df': 0.01,
 'bag_of_words__ngram_range': (1, 1),
 'estimator__criterion': 'gini',
 'estimator__n_estimators': 100}
```

```
rf_pipe = Pipeline([
    ('bag_of_words', CountVectorizer(analyzer=text_process, max_df=1.0, min_df=0.01, ngram_range=(1,1))),
    ('tf_idf', TfidfTransformer()),
    ('estimator', RandomForestClassifier(n_estimators=300, criterion='gini'))
])

rf_pipe.fit(X_train, y_train)
```

```
Pipeline(steps=[('bag_of_words',
                 CountVectorizer(analyzer=<function text_process at 0x7c2a9260e7a0>,
                                 min_df=0.01)),
                ('tf_idf', TfidfTransformer()),
                ('estimator', RandomForestClassifier(n_estimators=300))])
```

Pipeline

CountVectorizer

TfidfTransformer

TfidfTransformer()

RandomForestClassifier

NLP_FIQQI.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
feature_importance = pd.DataFrame(rf_pipe.steps[2][1].feature_importances_,
                                  rf_pipe.steps[0][1].get_feature_names_out(),
                                  columns=['importance'])
feature_importance.sort_values('importance', ascending=False).head(20)
```

	importance
not	0.075268
great	0.073190
good	0.057067
love	0.032349
delici	0.030212
amaz	0.026094
friendli	0.021998
food	0.018385
place	0.018384
fantast	0.018215
disappoint	0.018021
best	0.017465
like	0.016419
also	0.016246
awesom	0.016210
go	0.015919
servic	0.015648
nice	0.015230
time	0.014062
back	0.012596