BAB IV
PEMBAHASAN

4.1 Data

Data yang digunakan berasal dari abstrak Tugas Akhir Fakultas Teknik Universitas Muhammadiyah Jember yang ada *repository* Universitas Muhammadiyah Jember. Teknik pengumpulan data yang digunakan adalah studi literatur dengan cara mengambil abstrak Tugas Akhir mahasiswa Fakultas Teknik Universitas Muhammadiyah Jember. Dari hasil studi literatur didapatkan data abstrak Tugas Akhir mahasiswa Fakultas Teknik Universitas Muhammadiyah Jember berjumlah 100 data abstrak Tugas Akhir, dengan masing-masing kelas atau prodi memiliki data abstrak berjumlah 20 abstrak. Data abstrak Tugas Akhir selengkapnya dapat dilihat pada Lampian 1 (halaman 67). Berikut beberapa data hasil studi literatur data yang ditunjukkan pada Tabel 4.1.

Tabel 4.1 Contoh Abstrak Hasil Studi Literatur

Abstrak	Program Studi
A-RAYA Tour & Travel Jember adalah perusahaan yang bergerak dibidang biro perjalanan. Dalam mempromosikan layanannya A-RAYA Tour & Travel Jember masih menggunakan media cetak dalam bentuk brosur dan membuatpelanggan harus datang ke kantor untuk melakukan reservasi. Informasi merupakan salah satu kebutuhan masyarakat saat ini. Masyarakat pada umumnya ingin mendapatkan informasi yang diinginkan melalui media yang bervariasi, salah satunya melalui website. Pada umumnya orang menggunakan website sebagai pilihan untuk mengakses informasi yang lebih cepat, efektif dan terkini. Website ini dibangun dengan menggunakan bahasa PHP menggunakan database MySQL untuk memudahkan dalam mendesain website sebagai media informasi.	Manajemen Informatika
Kontribusi emisi gas buang pada kendaraan bermotor merupakan penyumbang emisi gas terbesar di dunia. Setelah emisi dari industri seperti pabrik baik pabrik dalam sekala besar maupun kecil dan juga emisi rumah tangga. Dampak emisi dari kendaraan bermotor	Teknik Mesin

semakin besar dengan bertambahnya jumlah kendaraan bermotor khususnya mobil. Setiap mobil yang telah dignakan tentunya memerlukan perawatan pada mesinya. Pada karya tulis ini penulis akan membandingkan pengaruh dari variasi celah busi terhadap emisi gas buang yang dihasilkan oleh pembakaran mesin 4 langka pada kendaraan roda 4 (empat). Pada celah busi 0,9 mm menghasilkan rata rata 0,06% CO, 14,3% CO2, 2,5 ppm HC, 0,12% O2, 1,003 λ. Pada celah busi 1,1 mm menghasilkan 0,05% CO, 14,6% CO2, 7 ppm HC, 0,08% O2, 1,004 λ. Pada celah busi 1,3 mm menghaasilkan 0,06% CO, 14,5% CO2, 17,5 ppm HC, 1,003 λ, semua pengujian dilakukan pada RPM 20005000 dengan kondisi mesin pada temperatur kerja	

4.2 *Pre-processing* Data

Pre-processing data abstrak dilakukan teknik *case folding, tokenizing, filtering,* dan *stemming* pada setiap abstrak Tugas Akhir tersebut. Berikut adalah *source code* proses *pre-processing* dalam *Jupyter Notebook* yang di tunjukan pada gambar 4.1.

```
def preprocessing(input_text):

    #CaseFolding
    Abstrak_cf = input_text.lower()

    #Tokenizing
    list_removeRegex = "(@[A-Za-z0-9]+)|(\w+.co\S+)|(\w+:\/\/\S+)|(\w+@\S+)|(#[A-Za-z0-9]+)|(\w+[0-9]+[a-z]+)|([0-9])|(\w+[0-9]+'
    removeRegex = ' '.join(re.sub(list_removeRegex, " ",Abstrak_cf).split())
    tokenizer = RegexpTokenizer(r'\w+')
    Abstrak_tk = tokenizer.tokenize(removeRegex)

    #Filtering
    stoplist = stopwords.words('indonesian')
    addStopwords1 = 'teknik'
    stoplist += addStopwords1.split()

    filtered = [w for w in Abstrak_tk if not w in stoplist]
    Abstrak_fl = " ".join(filtered)

    #Stemming
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    Abstrak_st = stemmer.stem(Abstrak_fl)

    return Abstrak_st
```


Gambar 4.1. *Pre-processing* pada *Jupyter Notebook*

Kemudian dilakukan pelabelan ulang pada data abstrak. Berikut adalah *source code* pelabelan data dalam *Jupyter Notebook* ditunjukan pada Gambar 4.2.

```
def pelabelan(input_label):
    global id
    if input_label == "Teknik Elektro":
        id = 1
    elif input_label == "Teknik Sipil":
        id = 2
    elif input_label == "Teknik Informatika":
        id = 3
    elif input_label == "Teknik Mesin":
        id = 4
    elif input_label == "Manajemen Informatika":
        id = 5
    return id
```

Gambar 4.2. *Source code* pelabelan data

4.3 Implementasi Pembobotan

Pembobotan pada data abstrak Tugas Akhir dilakukan menggunakan algoritma TF-IDF. Pembobotan ini dilakukan untuk mencari nilai dari tiap-tiap kata yang ada pada sekumpulan data yang akan dibentuk menjadi sebuah vektor pada proses klasifikasi. Dengan teknik pembobotan ini diimplementasikan ke dalam *Jupyter Notebook,* jumlah fitur atau kata pada semua abstrak Tugas Akhir didapatkan 2187 kata yang digunakan. Berikut adalah proses pembobotan dan hasil pembobotan dari dokumen pertama kata menggunakan TF-IDF pada *Jupyter Notebook* ditunjukan pada gambar 4.3.

```
Tfidf_vect = TfidfVectorizer()
Tfidf_vect.fit(data['hasil_preprocessing'])
Train_X_Tfidf = Tfidf_vect.transform(x)
print(Train_X_Tfidf.shape)
print(Train_X_Tfidf[0].data)
```

```
(100, 2187)
[0.09035563 0.0464055  0.05496883 0.07296837 0.04521886 0.04769544
 0.05439613 0.06269245 0.05241653 0.07296837 0.05067039 0.05067039
 0.07296837 0.05938434 0.06695734 0.056381   0.08824042 0.37615469
 0.29498421 0.10483305 0.07296837 0.09043772 0.07296837 0.07296837
 0.06269245 0.04039447 0.06269245 0.36484186 0.02971235 0.04769544
 0.07296837 0.0464055  0.07937047 0.03438344 0.04309739 0.07296837
 0.11876868 0.03959292 0.66957342 0.06695734 0.03959292 0.03676052
 0.04309739 0.03741952 0.04769544 0.06695734 0.05439613 0.08824042
 0.03612957 0.07296837 0.04769544]
```

Gambar 4.3 Proses dan hasil dari pembobotan pada *Jupyter Notebook*

4.4 Hasil Klasifikasi

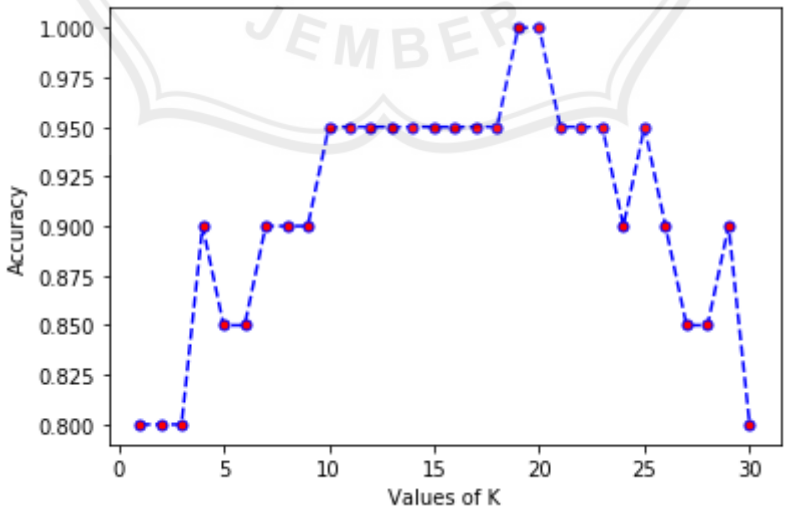
Data abstrak yang sebelumnya telah melalui proses *pre-processing* dan telah ditentukan hasil pembobotan setiap kata pada Gamabr 4.1, selanjutnya dilakukan klasifikasi menggunakan metode *Multinomial Naïve Bayes* dan *K-Nearest Neighbour*. Klasifikasi menggunakan metode *Multinomial Naïve Bayes* (MNB) dan *K-Nearest Neighbour* (KNN) dilakukan untuk mendapatkan hasil akurasi yang paling optimal dari kedua metode tersebut. Berikut adalah contoh perhitungan klasifikasi algoritma *Multinomial Naïve Bayes* pada *Jupyter Notebook* yang ditunjukan pada Gambar 4.4.

MNB, FOLD K = 2

```
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix

KFold2a = KFold(n_splits=2, shuffle=False)
for train_index, test_index in KFold2a.split(Train_X_Tfidf):

    print("TRAIN:", train_index, "TEST:", test_index)
    x_train, x_test = Train_X_Tfidf[train_index], Train_X_Tfidf[test_index]
    y_train, y_test = y[train_index], y[test_index]

    naive.fit(x_train, y_train)
    y_pred2a = naive.predict(x_test)
    print ()
    print ('Confusion Matrix')
    print (confusion_matrix(y_test, y_pred2a))
    print ()
    print (classification_report(y_test, y_pred2a))
```

Gambar 4.4. Klasifikasi algoritma *Multinomial Naïve Bayes*

Untuk KNN, terlebih dulu dilakukan pencarian *K* paling optimal. Dari 30 percobaan menggunakan pembagian data 80% data *training* dan 20% data *testing* didapatkan 2 nilai *K* dengan tingkat akurasi tertinggi yaitu *K* ke 19 dan 20. Namun pada penelitian ini hanya digunakan 1 nilai *K* yaitu *K* ke 20. Setelah ditemukan *K* paling optimal lalu dilanjutkan dengan proses klasifikasi yang ditunjukan pada Grafik 4.1, Tabel 4.3 dan Gambar 4.5.



Grafik 4.1 Penentuan K terbaik dari KNN

Tabel 4.2 Akurasi setiap *K* dalam penentuan *K* pada KNN

Nilai K	Akurasi
1	80%
2	80%
3	80%
4	90%
5	85%
6	85%
7	90%
8	90%
9	90%
10	95%
11	95%
12	95%
13	95%
14	95%
15	95%
16	95%
17	95%
18	95%
19	100%
20	100%
21	95%
22	95%
23	95%
24	90%
25	95%
26	90%
27	85%
28	85%
29	90%
30	80%

KNN, K-FOLD = 2

```
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import DistanceMetric

KFold20b = KFold(n_splits=2, shuffle=False)
for train_index, test_index in KFold20b.split(Train_X_Tfidf):

    print("TRAIN:", train_index, "TEST:", test_index)
    x_train, x_test = Train_X_Tfidf[train_index], Train_X_Tfidf[test_index]
    y_train, y_test = y[train_index], y[test_index]

    knn.fit(x_train, y_train)
    y_pred20b = knn.predict(x_test)
    print ()
    print ('Confusion Matrix')
    print (confusion_matrix(y_test, y_pred20b))
    print ()
    print (classification_report(y_test, y_pred20b))
```

Gambar 4.5. Klasifikasi algoritma *K-Nearest Neighbour*

Untuk mengevaluasi hasil pengklasifikasian digunakan sebuah teknik evaluasi yaitu *Confusion Matrix*. *Confusion Matrix* akan menganalisis kualitas *classifier* dalam mengenali tuple-tuple dari kelas yang ada. Untuk menilai atau mengevaluasi suatu model klasifikasi dapat menggunakan akurasi. Akurasi atau t menyatakan presentase dari jumlah tuple dalam data uji yang diklasifikasi dengan benar oleh *classifier*. Terdapat 4 istilah pada *Confusion Matrix* yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), *False Negative* (). TP, TN, FP dan FN dapat dijelaskan sebagai berikut :

- TP adalah *True Positive*, merupakan tuple positif yang dilabeli dengan benar oleh model klasifikasi/algoritma.
- TN adalah True Negative, merupakan tuple negatif yang dilabeli dengan benar oleh model klasifikasi/algoritma.
- FP adalah *False Positive*, merupakan tuple negatif yang salah dilabeli oleh model klasifikasi/algortima.
- FN adalah *False Negative*, merupakan tuple positif yang salah dilabeli oleh model klasifikasi/ algoritma.

4.4.1 Hasil Klasifikasi *Multinomial Naive Bayes* dan *K-Nearest Neighbour Fold K* = 2

Uji akurasi klasifikasi menggunakan *fold K* = 2 didapatkan 2 himpunan data yang terdiri dari 100 data. Masing-masing himpunan terdiri dari 50 data *testing* dan 50 data *training*.

Berikut hasil *confusion matrix* dari algoritma *Multinomial Naïve Bayes* dengan menggunakan *fold K* = 2 pada skenario pengujian terbaik yaitu skenario pengujian kedua yang ditunjukan pada Gambar 4.6.

```
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49] TEST: [50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97
 98 99]

Confusion Matrix
[[ 8  1  0  0  0]
 [ 1  8  0  0  0]
 [ 1  0  7  0  2]
 [ 3  0  0  8  0]
 [ 0  0  1  0 10]]

              precision    recall  f1-score   support

           1       0.62      0.89      0.73         9
           2       0.89      0.89      0.89         9
           3       0.88      0.70      0.78        10
           4       1.00      0.73      0.84        11
           5       0.83      0.91      0.87        11

    accuracy                           0.82        50
   macro avg       0.84      0.82      0.82        50
weighted avg       0.85      0.82      0.82        50
```

Gambar 4.6 *Confusion Matrix* pada algoritma MNB, *fold K* = 2

Berdasarkan Gambar 4.6, didapatkan hasil perhitungan akurasi pada algoritma *Multinomial Naïve Bayes* dengan menggunakan *fold K* = 2 sebesar 82%, juga di dapatkan presisi serta *recall* sebesar 84% dan 82%.

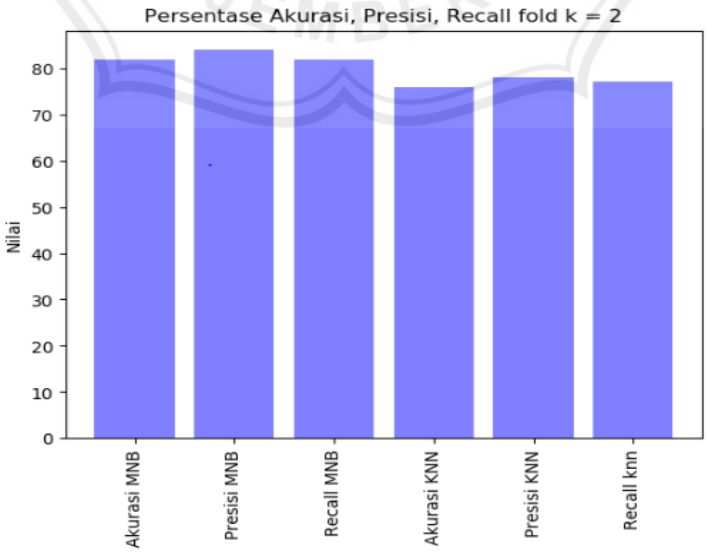
Berikut hasil *confusion matrix* algoritma *K-Nearest Neighbour* dengan menggunakan *fold K* = 2 pada skenario pengujian terbaik yaitu skenario pengujian kedua yang ditunjukan pada Gambar 4.7.

```
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49] TEST: [50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97
 98 99]

Confusion Matrix
[[8 1 0 0 0]
 [0 9 0 0 0]
 [1 0 7 0 2]
 [3 0 1 7 0]
 [0 1 3 0 7]]

              precision    recall  f1-score   support

           1       0.67      0.89      0.76         9
           2       0.82      1.00      0.90         9
           3       0.64      0.70      0.67        10
           4       1.00      0.64      0.78        11
           5       0.78      0.64      0.70        11

    accuracy                           0.76        50
   macro avg       0.78      0.77      0.76        50
weighted avg       0.79      0.76      0.76        50
```

Gambar 4.7 *Confusion Matrix* pada algoritma KNN, *fold K* = 2

Berdasarkan Gambar 4.7, didapatkan hasil perhitungan akurasi pada algoritma *K-Nearest Neighbour* dengan menggunakan *fold K* = 2 sebesar 76%, juga di dapatkan presisi serta *recall* sebesar 78% dan 77%.

Hasil akurasi klasifikasi MNB dan KNN menggunakan *fold K* = 2 ditampilkan selengkapnya pada Grafik 4.2.



Grafik 4.2 Hasil Perhitungan Akurasi Pada *Fold K* = 2

Berdasarkan Grafik 4.2, pada algoritma *Multinomial Naïve Bayes* mendapatkan hasil akurasi = 82%, presisi = 84% dan *recall* = 82%. Pada algoritma *K-Nearest Neighbour* didapatkan hasil akurasi = 76%, presisi = 78% dan *recall* = 77%.

4.4.2 Hasil Klasifikasi *Multinomial Naive Bayes* dan *K-Nearest Neighbour Fold K* = 4

Uji akurasi klasifikasi menggunakan *fold K* = 4 didapatkan 4 himpunan data yang terdiri dari 100 data. Masing-masing himpunan terdiri dari 25 data *testing* dan 75 data.

Berikut hasil *confusion matrix* algoritma *Multinomial Naïve Bayes* dengan menggunakan *fold K* = 4 pada skenario pengujian terbaik yaitu skenario pengujian ketiga yang ditunjukan pada Gambar 4.8

```
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96
 97 98 99] TEST: [50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
 74]

Confusion Matrix
[[4 0 0 0 0]
 [0 5 0 0 0]
 [1 0 3 0 3]
 [1 0 0 5 0]
 [0 0 0 0 3]]

              precision    recall  f1-score   support

           1       0.67      1.00      0.80         4
           2       1.00      1.00      1.00         5
           3       1.00      0.43      0.60         7
           4       1.00      0.83      0.91         6
           5       0.50      1.00      0.67         3

    accuracy                           0.80        25
   macro avg       0.83      0.85      0.80        25
weighted avg       0.89      0.80      0.79        25
```

Gambar 4.8 *Confusion Matrix* pada algoritma MNB, *fold K* = 4

Berdasarkan Gambar 4.8, didapatkan hasil perhitungan akurasi pada algoritma *Multinomial Naïve Bayes* dengan menggunakan *fold K* = 4 sebesar 80%, juga di dapatkan presisi serta *recall* sebesar 83% dan 85%.

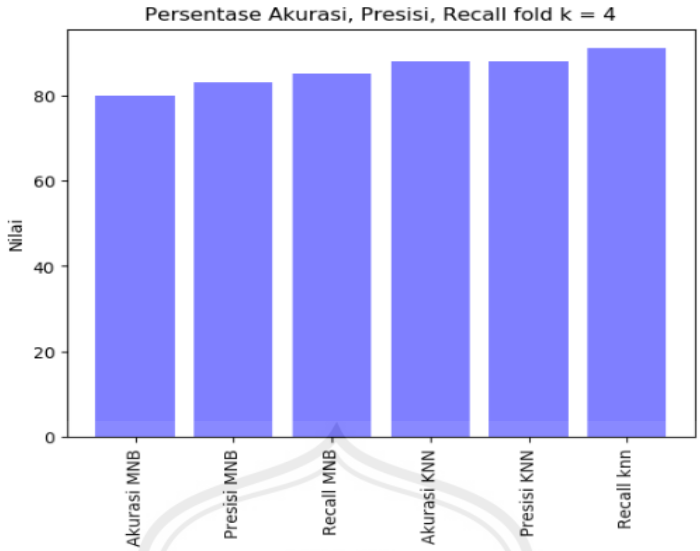
Berikut hasil *confusion matrix* algoritma *K-Nearest Negihbour* dengan menggunakan *fold K* = 4 pada skenario pengujian terbaik yaitu skenario pengujian ketiga yang ditunjukan pada Gambar 4.9.

```
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96
 97 98 99] TEST: [50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
 74]

Confusion Matrix
[[4 0 0 0 0]
 [0 5 0 0 0]
 [1 0 4 0 2]
 [0 0 0 6 0]
 [0 0 0 0 3]]

              precision    recall  f1-score   support

           1       0.80      1.00      0.89         4
           2       1.00      1.00      1.00         5
           3       1.00      0.57      0.73         7
           4       1.00      1.00      1.00         6
           5       0.60      1.00      0.75         3

    accuracy                           0.88        25
   macro avg       0.88      0.91      0.87        25
weighted avg       0.92      0.88      0.88        25
```

Gambar 4.9 *Confusion Matrix* pada algoritma KNN, *fold K* = 4

Berdasarkan Gambar 4.9, didapatkan hasil perhitungan akurasi pada algoritma *K-Nearest Neighbour* dengan menggunakan *fold K* = 4 sebesar 88%, juga di dapatkan presisi serta *recall* sebesar 88% dan 91%.

Hasil akurasi klasifikasi MNB dan KNN menggunakan *fold K* = 4 ditampilkan selengkapnya pada Grafik 4.3.



Grafik 4.3 Hasil Perhitungan Akurasi Pada *Fold K*=4

Berdasarkan Grafik 4.3, pada algoritma *Multinomial Naïve Bayes* mendapatkan hasil akurasi = 80%, presisi = 83% dan *recall* = 85%. Pada algoritma *K-Nearest Neighbour* didapatkan hasil akurasi = 88%, presisi = 88% dan *recall* = 91%.

4.4.3 Hasil Klasifikasi *Multinomial Naive Bayes* dan *K-Nearest Neighbour Fold K*=5

Uji akurasi klasifikasi menggunakan *fold K* = 5 didapatkan 5 himpunan data yang terdiri dari 100 data. Masing-masing himpunan terdiri dari 20 data testing dan 80 data training.

Berikut hasil *confusion matrix* algoritma *Multinomial Naïve Bayes* dengan menggunakan *fold K* = 5 pada skenario pengujian terbaik yaitu skenario pengujian keempat yang ditunjukan pada Gambar 4.10.

```
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 80 81 82 83 84 85 86 87 88 89 90 91
 92 93 94 95 96 97 98 99] TEST: [60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79]
```

```
Confusion Matrix
[[2 0 0 0 0]
 [0 2 0 0 0]
 [0 0 5 0 1]
 [1 0 0 5 0]
 [0 0 0 0 4]]

              precision    recall  f1-score   support

           1       0.67      1.00      0.80         2
           2       1.00      1.00      1.00         2
           3       1.00      0.83      0.91         6
           4       1.00      0.83      0.91         6
           5       0.80      1.00      0.89         4

    accuracy                           0.90        20
   macro avg       0.89      0.93      0.90        20
weighted avg       0.93      0.90      0.90        20
```

Gambar 4.10. *Confusion Matrix* pada algoritma MNB, *fold K* = 5

Berdasarkan Gambar 4.10, didapatkan hasil perhitungan akurasi pada algoritma *Multinomial Naïve Bayes* dengan menggunakan *fold K* = 5 sebesar 90%, juga di dapatkan presisi serta *recall* sebesar 89% dan 93%.

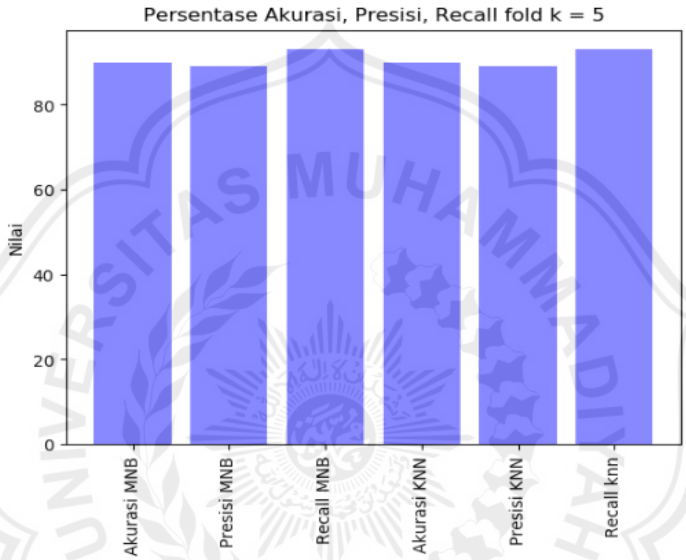
Berikut hasil *confusion matrix* algoritma *K-Nearest Negihbour* dengan menggunakan *fold K* = 5 pada skenario pengujian terbaik yaitu skenario pengujian keempat yang ditunjukan pada Gambar 4.11.

```
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 80 81 82 83 84 85 86 87 88 89 90 91
 92 93 94 95 96 97 98 99] TEST: [60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79]

Confusion Matrix
[[2 0 0 0 0]
 [0 2 0 0 0]
 [0 0 5 0 1]
 [1 0 0 5 0]
 [0 0 0 0 4]]

              precision    recall  f1-score   support

           1       0.67      1.00      0.80         2
           2       1.00      1.00      1.00         2
           3       1.00      0.83      0.91         6
           4       1.00      0.83      0.91         6
           5       0.80      1.00      0.89         4

    accuracy                           0.90        20
   macro avg       0.89      0.93      0.90        20
weighted avg       0.93      0.90      0.90        20
```

Gambar 4.11 *Confusion Matrix* pada algoritma KNN, *fold K* = 5

Berdasarkan Gambar 4.11, didapatkan hasil perhitungan akurasi pada algoritma *K-Nearest Neighbour* dengan menggunakan *fold K* = 5 sebesar 90%, juga di dapatkan presisi serta *recall* sebesar 89% dan 93%.

Hasil akurasi klasifikasi MNB dan KNN menggunakan *fold K* = 5 ditampilkan selengkapnya pada Grafik 4.4.



Grafik 4.4 Hasil Perhitungan Akurasi Pada *Fold K*=5

Berdasarkan Grafik 4.4, pada algoritma *Multinomial Naïve Bayes* dan algoritma *K-Nearest Neighbour* didapatkan hasil yang sama, yaitu akurasi sebesar 90%, persisi sebesar 89% dan *recall* sebesar 93%.

4.4.4 Hasil Klasifikasi *Multinomial Naive Bayes* dan *K-Nearest Neighbour Fold K* = 10

Uji akurasi klasifikasi menggunakan *fold K* = 10 didapatkan 10 himpunan data yang terdiri dari 100 data. Masing-masing himpunan terdiri dari 10 data testing dan 90 data training.

Berikut hasil *confusion matrix* algoritma *Multinomial Naïve Bayes* dengan menggunakan *fold K*=10 pada skenario pengujian terbaik yaitu skenario pengujian pertama yang ditunjukan pada Gambar 4.12.

```
TRAIN: [10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81
 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99] TEST: [0 1 2 3 4 5 6 7 8 9]
```

```
Confusion Matrix
[[2 0 0 0 0]
 [0 3 0 0 0]
 [0 0 1 0 0]
 [0 1 0 1 0]
 [0 0 0 0 2]]
```

	precision	recall	f1-score	support
1	1.00	1.00	1.00	2
2	0.75	1.00	0.86	3
3	1.00	1.00	1.00	1
4	1.00	0.50	0.67	2
5	1.00	1.00	1.00	2
accuracy			0.90	10
macro avg	0.95	0.90	0.90	10
weighted avg	0.93	0.90	0.89	10

Gambar 4.12. *Confusion Matrix* pada algoritma MNB, *fold K* = 10

Berdasarkan Gambar 4.12, didapatkan hasil perhitungan akurasi pada algoritma *Multinomial Naïve Bayes* dengan menggunakan *fold K* = 10 sebesar 90%, juga di dapatkan presisi serta *recall* sebesar 90% dan 95%.

Berikut hasil *confusion matrix* algoritma *K-Nearest Negihbour* dengan menggunakan *fold K* = 10 pada skenario pengujian terbaik yaitu skenario pengujian kesepuluh yang ditunjukan pada Gambar 4.13.

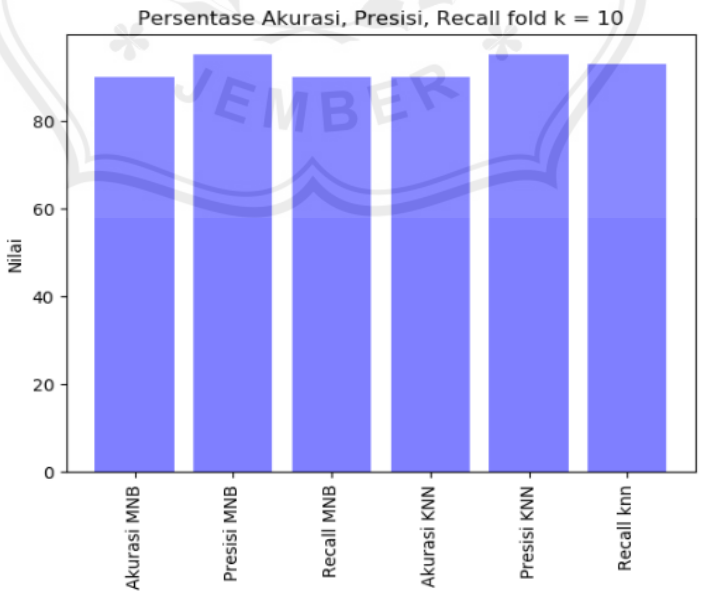
```
TRAIN: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89] TEST: [90 91 92 93 94 95 96 97 98 99]
```

```
Confusion Matrix
[[3 0 0 0 0]
 [1 2 0 0 0]
 [0 0 1 0 0]
 [0 0 0 1 0]
 [0 0 0 0 2]]

              precision    recall  f1-score   support

           1       0.75      1.00      0.86         3
           2       1.00      0.67      0.80         3
           3       1.00      1.00      1.00         1
           4       1.00      1.00      1.00         1
           5       1.00      1.00      1.00         2

    accuracy                           0.90        10
   macro avg       0.95      0.93      0.93        10
weighted avg       0.93      0.90      0.90        10
```

Gambar 4.13 *Confusion Matrix* pada algoritma KNN, *fold K* = 10

Berdasarkan Gambar 4.13, didapatkan hasil perhitungan akurasi pada algoritma *K-Nearest Neighbour* dengan menggunakan *fold K*=10 sebesar 90%, juga di dapatkan presisi serta *recall* sebesar 95% dan 93%.

Hasil akurasi klasifikasi MNB dan KNN menggunakan *fold K* = 10 ditampilkan selengkapnya pada Grafik 4.5.



Grafik 4.5 Hasil Perhitungan Akurasi Pada *Fold* K = 10

Berdasarkan Grafik 4.5, pada algoritma *Multinomial Naïve Bayes* mendapatkan hasil akurasi = 90%, presisi = 95% dan *recall* = 90%. Pada algoritma *K-Nearest Neighbour* didapatkan hasil akurasi = 90%, presisi = 95% dan *recall* = 93%.

Hasil akurasi dengan *fold K* = 2, 4, 5 dan 10 pada *dataset* abstrak Tugas Akhir, untuk algoritma *Multinomial Naïve Bayes* didapatkan hasil akurasi tertinggi sebesar 90% pada nilai *fold K* = 10, skenario ke 1, presisi tertinggi sebesar 95% pada nilai *fold K* = 10, skenario ke 1 dan *recall* tertinggi sebesar 93% pada nilai *fold K* = 10, skenario ke 7, sedangkan untuk algoritma *K-Nearest Neighbour* didapatkan hasil akurasi tertinggi sebesar 90% pada nilai *fold K* = 10, skenario ke 10, presisi tertinggi sebesar 95% pada nilai *fold K* = 10, skenario ke 10 dan *recall* tertinggi sebesar 93% pada nilai *fold K* = 10, skenario ke .10 Hasil perhitungan akurasi, presisi dan *recall* selengkapnya ditampilkan dalam bentuk tabel sebagai berikut:

Tabel 4.3 Hasil akurasi, presisi, dan *recall* pada algoritma MNB dan KNN

FOLD K		**MNB**			**KNN**		
		AKURASI	**PRESISI**	***RECALL***	**AKURASI**	**PRESISI**	***RECALL***
2	SKENARIO 1	58%	74%	61%	74%	75%	75%
	SKENARIO 2	82%	84%	82%	76%	78%	77%
4	SKENARIO 1	76%	78%	79%	76%	80%	78%
	SKENARIO 2	60%	75%	65%	76%	77%	79%
	SKENARIO 3	80%	83%	85%	88%	88%	91%
	SKENARIO 4	72%	82%	78%	84%	87%	87%
5	SKENARIO 1	75%	77%	78%	80%	82%	82%
	SKENARIO 2	80%	81%	80%	75%	78%	75%
	SKENARIO 3	60%	72%	69%	75%	81%	81%
	SKENARIO 4	90%	89%	93%	90%	89%	93%
	SKENARIO 5	85%	87%	88%	90%	87%	83%
10	SKENARIO 1	90%	95%	90%	90%	93%	90%
	SKENARIO 2	70%	63%	70%	70%	63%	70%
	SKENARIO 3	70%	77%	70%	70%	77%	70%
	SKENARIO 4	80%	83%	83%	80%	90%	83%
	SKENARIO 5	80%	83%	87%	80%	83%	87%
	SKENARIO 6	80%	63%	80%	80%	63%	80%
	SKENARIO 7	90%	93%	93%	90%	95%	90%

	SKENARIO 8	90%	90%	93%	90%	90%	93%
	SKENARIO 9	90%	76%	80%	80%	76%	70%
	SKENARIO 10	80%	85%	83%	90%	95%	93%
RATA-RATA		78%	80%	80%	81%	82%	82%

Berdasarkan Tabel di atas skenario terbaik ditentukan berdasarkan akurasi tertinggi namun jika akurasi memiliki nilai yang sama pada beberapa skenario, maka skenario terbaik dipilih berdasarkan nilai presisi tertinggi. Aturan ini hanya berlaku pada klasifikasi teks menurut Jo (2018).