LAPORAN PRAKTIKUM
PENERAPAN MACHINE LEARNING
MENGGUNAKAN ALGORITMA KLASIFIKASI
MATA KULIAH MACHINE LEARNING
KELAS C

DISUSUN OLEH:

SANDRIA AMELIA PUTRI

21083010005

DOSEN PENGAMPU:

AVIOLLA TERZA DAMALIANA, S.SI., M.STAT

DR. ENG. IR. ANGGRAINI PUSPITA SARI., ST., MT.

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

JAWA TIMUR

2023

DAFTAR ISI

BAB 1: PENDAHULUAN

1.1. Latar Belakang

Statistika dan machine learning membentuk satu konsep yang disebut statistical machine learning dengan menggunakan model regresi logistik. Statistika dan machine learning menggunakan teori serta rumus yang sama dalam menyelesaikan suatu permasalahan. Keduanya sama-sama membahas konsep random variable, distribusi statistik, expected value, variansi, sampai pada konsep distribusi prior dan posterior (R., 2018).

Klasifikasi dalam machine learning merupakan suatu pengelompokan data di mana data yang digunakan memiliki kelas label atau target. Sehingga algoritma untuk menyelesaikan masalah klasifikasi dikategorisasikan ke dalam supervised learning (Ramadhan, 2018). Algoritma klasifikasi dapat digunakan untuk klasifikasi teks, analisis sentiment, deteksi spam, deteksi penipuan, segmentasi pelanggan, dan klasifikasi gambar. Algoritma yang dipilih bergantung pada kumpulan data dan tujuan yang ingin dicapai. Beberapa algoritma klasifikasi di antaranya yaitu Naïve Bayes Classifier, decision tree, Support Vector Machine, dan Random Forest Classifier (K, 2021). Pada laporan ini akan dilakukan algoritma klasifikasi dengan algoritma Naïve Bayes dan decision tree pada suatu permasalahan.

1.2. Permasalahan

Untuk penerapan machine learning menggunakan algoritma klasifikasi digunakan dataset “Diabetes.” Kumpulan data ini berasal dari National Institute of Diabetes and Digestive and Kidney Diseases. Tujuan dari kumpulan data ini adalah untuk memprediksi secara diagnostik apakah seorang pasien menderita diabetes, berdasarkan pengukuran diagnostik tertentu yang termasuk dalam kumpulan data. Secara khusus, semua pasien di sini adalah perempuan berusia minimal 21 tahun keturunan Indian Pima. Terdapat beberapa variabel independen (beberapa variabel prediktor medis) dan satu variabel dependen target (Outcome). Berikut merupakan bentuk data dalam excel:

	A	B	C	D	E	F	G	H	I	J
1	Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age,Outcome									
2	6,148,72,35,0,33.6,0.627,50,1									
3	1,85,66,29,0,26.6,0.351,31,0									
4	8,183,64,0,0,23.3,0.672,32,1									
5	1,89,66,23,94,28.1,0.167,21,0									
6	0,137,40,35,168,43.1,2.288,33,1									
7	5,116,74,0,0,25.6,0.201,30,0									
8	3,78,50,32,88,31,0.248,26,1									
9	10,115,0,0,0,35.3,0.134,29,0									
10	2,197,70,45,543,30.5,0.158,53,1									
11	8,125,96,0,0,0,0.232,54,1									
12	4,110,92,0,0,37.6,0.191,30,0									
13	10,168,74,0,0,38,0.537,34,1									
14	10,139,80,0,0,27.1,1.441,57,0									
15	1,189,60,23,846,30.1,0.398,59,1									
16	5,166,72,19,175,25.8,0.587,51,1									
17	7,100,0,0,0,30,0.484,32,1									
18	0,118,84,47,230,45.8,0.551,31,1									

Berikut merupakan deskripsi data:

No	Variabel	Keterangan	Jenis Data
1	Pregnancies	Jumlah kehamilan pasien	Integer

2	Glucose	Kadar glukosa dalam darah pasien	Integer
3	BloodPressure	Ukuran tekanan darah pasien	Integer
4	SkinThickness	Ketebalan kulit pasien	Integer
5	Insulin	Kadar insulin dalam darah pasien	Integer
6	BMI	Indeks massa tubuh pasien	Float
7	DiabetesPedigreeFunction	Persentase diabetes pasien	Float
8	Age	Umur pasien	Integer
9	Outcome	Hasil akhir dengan 1 adalah ya dan 0 adalah tidak	Kategorik

Variabel Outcome merupakan variabel target yang merupakan inti permasalahannya dan ingin diprediksi nilainya.

1.3. Tujuan

1. Memahami teori regresi logistik dengan tepat.
2. Memahami teori Naïve Bayes dengan tepat.
3. Memahami decision tree dengan tepat.
4. Memahami evaluasi performance pada klasifikasi.

1.4. Manfaat

1. Dapat memahami teori regresi logistik dengan tepat.
2. Dapat memahami teori Naïve Bayes dengan tepat.
3. Dapat memahami decision tree dengan tepat.
4. Dapat memahami evaluasi performance pada klasifikasi.

BAB 2: TINJAUAN PUSTAKA

2.1. Data Eksplorasi

Analisis Data Eksplorasi (ADE) digunakan saat menganalisis dan menyelidiki kumpulan data. Hal ini membantu para ilmuwan data untuk menemukan pola data, anomali spot, pengujian hipotesis, dan asumsi. Tujuan utama dari ADE yaitu untuk membantu melihat lebih dalam pada kumpulan data sebelum membuat asumsi apa pun, mengidentifikasi kesalahan yang jelas, mendapatkan pemahaman yang lebih baik tentang pola di dalam kumpulan data, mencari tahu outlier dan kejadian anomali, serta menemukan hubungan yang menarik antar variabel (Admin, 2022).

2.2. Regresi Logistik

Regresi logistik merupakan algoritma untuk mengklasifikasi data biner dengan output data biner juga. Cara kerja regresi logistik yaitu dengan mengombinasi fitur linier dan fungsi non linier (sigmoid). Algoritma ini menghasilkan banyak cara untuk mengatur model dan menghasilkan interpretasi probabilitas yang bagus. Contoh penggunaan regresi logistik adalah untuk memprediksi pelanggan, penilaian kredit dan mendeteksi penipuan, mengukur efektivitas iklan pemasaran, dan masih banyak lagi (Yovita, 2020). Berikut merupakan persamaan regresi logistik (Hidayat, 2015):

$$ln\left(\frac{\hat{p}}{1-\hat{p}}\right) = B_0 + B_1X$$

dengan,

ln : logaritma natural (2,72)

$B_0 + B_1X$: persamaan yang biasa dikenal dalam OLS

$\hat{p}$: probabilitas logistik yang didapat dari rumus berikut,

$$\hat{p} = \frac{exp(B_0 + B_1X)}{1 + exp(B_0 + B_1X)} = \frac{e^{B_0+B_1X}}{1 + e^{B_0+B_1X}}$$

dengan,

exp atau e : fungsi eksponen (kebalikan dari ln)

2.3. Naïve Bayes

Jika kita memiliki data yang sangat besar, memiliki kendala spesifikasi CPU dan memori, teknik klasifikasi berdasarkan teorema Bayes (Naïve Bayes) merupakan pilihan terbaik. Algoritma ini memiliki beberapa asumsi. Jika asumsi independensi terpenuhi, maka proses pengklasifikasian akan berjalan lebih cepat jika dibandingkan dengan algoritma regresi logistik sehingga kita tidak membutuhkan banyak data latihan (data training). Namun terkadang, jika asumsi tidak terpenuhi pun, algoritma ini tetap bisa digunakan. Contoh penggunaan algoritma Naïve Bayes adalah untuk analisis sentimen dan klasifikasi teks, sistem rekomendasi seperti pada Netflix dan Amazon, klasifikasi email sebagai spam atau bukan spam, pengenalan wajah, dan masih banyak lagi (Yovita, 2020). Berikut merupakan persamaan Naïve Bayes Classifier berdasarkan probabilitas yaitu (ITATS, 2015):

$$p(A|B) \cdot p(B) = p(B|A) \cdot p(A)$$

$$p(A_i|B) = \frac{p(A_i) \cdot p(B|A_i)}{\sum_{j=1}^{c} p(A_j) \cdot p(B|A_j)}$$

dengan mengubah nilai A_i dan A_j ke dalam vektor x maka didapatkan persamaan sebagai berikut:

$$p(x|i) = \frac{p(i) \cdot p(x|i)}{\sum_{j=1}^{c} p(j) \cdot p(x|j)}$$

Apabila nilai p disubstitusikan ke dalam x yang bersifat independen tidak saling terkait, maka didapatkan persamaan baru sebagai berikut:

$$p(x|i) = \prod_{k=1}^{p} p(x^{(k)}|i)$$

dengan,

$p(x|i)$: probabilitas hipotesis x jika diberikan fakta atau record i (posterior probability)

$p(i|x)$: mencari nilai parameter yang memberi kemungkinan yang paling besar (likelihood)

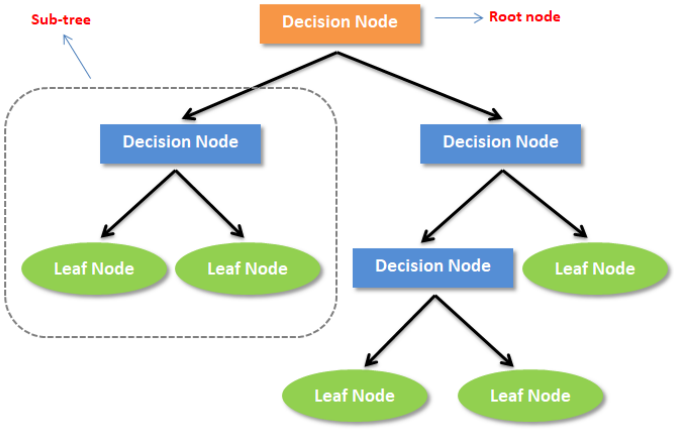
$p(x)$: prior probability dari i (prior probability)

$p(i)$: jumlah probability tuple yang muncul

2.4. Decision Tree

Algoritma decision tree dalam jumlah banyak dapat digunakan untuk membangun algoritma yang efisien seperti random forest atau gradient tree boosting. Algoritma ini termasuk ke dalam statistik non parametrik sehingga tidak perlu mengkhawatirkan outlier. Decision tree dapat digunakan untuk menganalisis keputusan investasi, menganalisis masalah bank yang tidak membayar pinjaman, analisis keputusan buy and build, kualifikasi prospek penjualan, dan masih banyak lagi. Kelemahan dari algoritma ini yaitu tidak dapat diterapkan pada data baru dan membutuhkan banyak memori. Jika ingin menggunakan decision tree untuk data baru, kita harus membuat ulang algoritmanya. Selain itu, semakin banyak cabang dari pohonnya, maka semakin besar juga memori yang dibutuhkan (Yovita, 2020).

Dalam decision tree terdapat dua jenis node, yaitu decision node dan leaf node. Decision node mewakili fitur dari dataset dan digunakan untuk membuat keputusan. Sedangkan, leaf node adalah output dari keputusan tersebut dan tidak berisi cabang lebih lanjut (Afifah, t.thn.).


Sub-tree
Decision Node
Root node
Decision Node
Decision Node
Leaf Node
Leaf Node
Decision Node
Leaf Node
Leaf Node
Leaf Node

BAB 3: ANALISIS DAN PEMBAHASAN

3.1. Menampilkan dataset Diabetes ke dalam dataFrame

- Import library yang diperlukan yaitu pandas dan numpy. Library pandas digunakan untuk analisis data dan manipulasi data. Library numpy digunakan untuk melakukan komputasi numerik.

```
[1]: # import library
     import pandas as pd
     import numpy as np
```

- Tampilkan dataset Diabetes ke dalam dataFrame dengan menggunakan code pd.read_csv(). Simpan pada variabel baru yaitu df_diabetes. Diketahui bahwa data tersebut memiliki 768 baris dan 9 kolom.

```
[2]: df_diabetes = pd.read_csv("diabetes.csv")
     df_diabetes
```

[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

- Tampilkan 5 baris pertama dataset Diabetes dengan menggunakan code head().

```
[3]: df_diabetes.head()
```

[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

3.2. Melakukan tahapan pre-processing data

- Periksa jenis data setiap variabel/kolom dengan menggunakan code dtypes. Dihasilkan output bahwa data tersebut memiliki variabel yang berjenis float dan integer.

```
[4]: # memeriksa jenis data setiap variabel
     df_diabetes.dtypes
```

```
[4]: Pregnancies                 int64
     Glucose                     int64
     BloodPressure               int64
     SkinThickness               int64
     Insulin                     int64
     BMI                       float64
     DiabetesPedigreeFunction  float64
     Age                         int64
     Outcome                     int64
     dtype: object
```

- Pada poin 1.2 diketahui bahwa data variabel Outcome berjenis kategori, namun dataset tersebut masih bertipe integer. Sehingga kita perlu mengubah jenis data tersebut menjadi kategorik dengan menggunakan code astype().

```
[5]: # mengubah jenis data menjadi kategorikal pada variabel Outcome
     df_diabetes["Outcome"] = df_diabetes["Outcome"].astype('category')
```

- Periksa apakah jenis data variabel Outcome telah berubah menjadi kategorik dengan menggunakan code dtypes.

```
[6]: # periksa apakah jenis data variabel Outcome sudah berubah
     df_diabetes["Outcome"].dtypes
```

```
[6]: CategoricalDtype(categories=[0, 1], ordered=False)
```

- Periksa dimensi data dengan menggunakan code np.shape(). Dihasilkan output bahwa dataset tersebut memiliki 768 baris dan 9 kolom.

```
[7]: # memeriksa dimensi data
     np.shape(df_diabetes)
```

```
[7]: (768, 9)
```

- Memeriksa missing value setiap variabel/kolom dengan menggunakan code isnull().any(). Code tersebut akan mengembalikan nilai Boolean (True atau False) untuk setiap kolom. Jika hasilnya True, maka kolom tersebut mengandung setidaknya satu missing value. Sebaliknya jika hasilnya False, maka kolom tersebut tidak mengandung missing value. Didapatkan bahwa dataset Diabetes tidak mengandung missing value.

```
[8]: # memeriksa missing value setiap variabel
     df_diabetes.isnull().any()
```

```
[8]: Pregnancies                 False
     Glucose                     False
     BloodPressure               False
     SkinThickness               False
     Insulin                     False
     BMI                         False
     DiabetesPedigreeFunction    False
     Age                         False
     Outcome                     False
     dtype: bool
```

- Melakukan transformasi data (variabel prediktor) dengan metode MinMaxScaler. import terlebih dahulu modul preprocessing dari library sklearn, yang banyak digunakan untuk machine learning. Buat instance kelas MinMaxScaler() dari preprocessing dan tugaskan ke variabel transformasi_min_max. Scaler ini digunakan untuk mengubah fitur dengan menykalakannya ke rentang tertentu.

```
[9]: # transforming variabel prediktor dengan metode MinMaxScaler
     from sklearn import preprocessing
     transformasi_min_max = preprocessing.MinMaxScaler()
```

3.3. Membagi data menjadi data training dan testing

- Membagi dataFrame menjadi variabel target (Y) dan atribut (X). variabel target yaitu Outcome dan sisanya merupakan atribut.

```
[10]: # membagi data menjadi variabel target dan atribut
      target = 'Outcome'
      x = df_diabetes.drop('Outcome', axis=1)
      y = df_diabetes[target]
      nama_var_x = ['Pregnancies','Glucose','BloodPressure','SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
      x = pd.DataFrame(data=transformasi_min_max.fit_transform(x), columns=nama_var_x)
```

- Berikut merupakan dataFrame atribut (X) yang sudah dihilangkan variabel targetnya.

```
[11]: x
```

[11]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000
...	...	...	...	...	...	...	...	...
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	0.039710	0.700000
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	0.111870	0.100000
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	0.071307	0.150000
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	0.115713	0.433333
767	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	0.101196	0.033333

768 rows × 8 columns

- Membagi data menjadi data training 75% dan data testing 25% dengan mengimpor modul train_tes_split dari library skleaern. x dan y masing-masing adalah data input dan target yang akan dibagi menjadi data training dan data testing. test_size=0.25 menentukan bahwa 25% data digunakan untuk testing dan 75% data digunakan untuk training. random_state=1 menentukan random seed yang digunakan oleh fungsi untuk membagi data agar memastikan bahwa pemisahan yang sama diperoleh setiap kali code dijalankan. Hasilnya akan disimpan pada variabel x_train, x_test, y_train, dan y_test. x_train dan y_train masing-masing berisi data input dan target untuk training set. Sedangkan, x_test dan y_test masing-masing berisi data input dan target untuk testing set.

```
[12]: # membagi data menjadi data training dan data testing (75% Training dan 25% Testing)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25,random_state=1)
```

3.4. Data eksplorasi

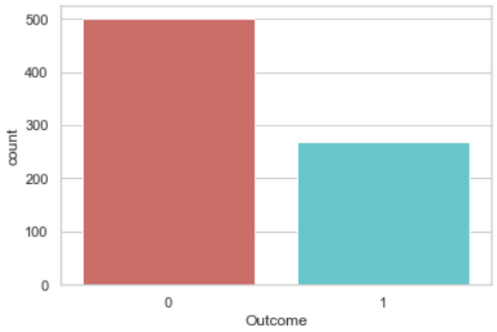
- Melihat seberapa banyak nilai 1 dan 0 pada variabel Outcome dengan menggunakan code value_counts(). Didapatkan output yaitu terdapat 268 nilai 1 dan 500 nilai 0, artinya terdapat lebih banyak pasien yang tidak terindikasi penyakit diabetes (0).

```
[13]: # melihat seberapa banyak nilai 1 dan 0 pada variabel Outcome
df_diabetes['Outcome'].value_counts()
```

```
[13]: 0    500
      1    268
      Name: Outcome, dtype: int64
```

- Membuat plot nilai 1 dan 0 pada variabel Outcome. Import terlebih dahulu modul pyplot dari library matplotlib dan import library seaborn. Fungsi set() digunakan untuk mengatur style menjadi “white” dan mengatur parameter style menjadi “whitegrid” serta parameter color_codes menjadi True guna menambahkan garis kisi ke plot dan memungkinkan kode warna digunakan. Fungsi countplot() digunakan untuk membuat count plot dengan x=’Outcome’ yaitu menentukan kolom dari dataFrame yang harus dihitung dan diplot, data=df_diabetes yaitu menentukan dataFrame yang menjadi dasar perhitungan plot, dan palette=’hls’ yaitu menentukan palet warna yang digunakan plot. Simpan plot tersebut dengan fungsi plt.savefig(). Dan tampilkan plot dengan fungsi plt.show().

```
[14]: import matplotlib.pyplot as plt
      import seaborn as sns
      sns.set(style="white")
      sns.set(style="whitegrid", color_codes=True)
      sns.countplot(x='Outcome',data=df_diabetes,palette='hls')
      plt.savefig('count_plot')
      plt.show()
```



- Membagi data menjadi data training 80% dan data testing 20% dengan mengimpor modul train_tes_split dari library skleaern. x dan y masing-masing adalah data input dan target yang akan dibagi menjadi data training dan data testing. test_size=0.2 menentukan bahwa 20% data digunakan untuk testing dan 80% data digunakan untuk training. random_state=1 menentukan random seed yang digunakan oleh fungsi untuk membagi data agar memastikan bahwa pemisahan yang sama diperoleh setiap kali code dijalankan. Hasilnya akan disimpan pada variabel x_train, x_test, y_train, dan y_test. x_train dan y_train masing-masing berisi data input dan target untuk training set. Sedangkan, x_test dan y_test masing-masing berisi data input dan target untuk testing set.

```
[15]: # membagi data menjadi data training dan data testing (80% Training dan 20% Testing)
      from sklearn.model_selection import train_test_split
      x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=1)
```

3.5. Regresi logistik

- Import class LogisticRegression dari modul linear_model pada library sklearn yang digunakan untuk memecahkan masalah klasifikasi biner. Import modul metrics dari library sklearn untuk mengevaluasi kinerja model. Inisialisasikan variabel logreg untuk menyimpan class LogisticRegression yang menyesuaikan model menggunakan fungsi fit() dengan x_train dan y_train sebagai input. x_train adalah input data training, y_train adalah output yang sesuai atau nilai target untuk data training.

```
[16]: from sklearn.linear_model import LogisticRegression
      from sklearn import metrics
      logreg = LogisticRegression()
      logreg.fit(x_train, y_train)
```

- Fungsi predict() digunakan untuk menghasilkan prediksi data input pengujian x_test. Nilai prediksi tersebut disimpan pada variabel y_pred. fungsi score() digunakan untuk menghitung akurasi model pada data testing dan mengembalikan akurasi rata-rata pada data testing dan label yang diberikan. Didapatkan keakuratan pengklasifikasian regresi logistik pada data testing yaitu 0.77.

```
[17]: y_pred = logreg.predict(x_test)
      print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(x_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.77
```

- Import fungsi confusion_matrix dari modul metrics pada library sklearn. Fungsi tersebut mengambil dua argumen yaitu y_test (nilai target sebenarnya) dan y_pred (prediksi nilai target). Nilai confusion matrix yang disimpan pada variabel confusion_matrix akan digunakan untuk mengevaluasi kinerja model klasifikasi. Outputnya menghasilkan matriks 2x2, di mana baris sesuai dengan label kelas sebenarnya dan kolom sesuai dengan label kelas yang diprediksi.

```
[18]: from sklearn.metrics import confusion_matrix
      confusion_matrix = confusion_matrix(y_test, y_pred)
      print(confusion_matrix)

      [[91  8]
       [27 28]]
```

- Import fungsi classification_report dari modul metrics pada library sklearn. Fungsi tersebut mengambil dua argumen yaitu y_test (nilai target sebenarnya) dan y_pred (prediksi nilai target). Outputnya akan menghasilkan classification report, di antaranya yaitu precision, recall, f1-score, dan support untuk setiap kelas.

```
[19]: from sklearn.metrics import classification_report
      print(classification_report(y_test, y_pred))

                    precision    recall  f1-score   support

                 0       0.77      0.92      0.84        99
                 1       0.78      0.51      0.62        55

          accuracy                           0.77       154
         macro avg       0.77      0.71      0.73       154
      weighted avg       0.77      0.77      0.76       154
```

3.6. Naïve Bayes

- Melatih pengklasifikasian Gaussian Naïve Bayes pada data training. Import terlebih dahulu class GaussianNB dari modul naïve_bayes library sklearn untuk mengimplementasikan algoritma Gaussian Naïve Bayes, yaitu algoritma probabilistic yang membuat prediksi dengan menarapkan teorema Bayes dengan asumsi independensi antar fitur. Variabel gnb digunakan untuk menyimpan class GaussianNB. Gunakan fungsi fit() untuk melatih model pada data training yaitu x_train (fitur) dan y_train (variabel target).

```
[20]: # train a Gaussian Naive Bayes classifier on the training set
      from sklearn.naive_bayes import GaussianNB

      # instantiate the model
      gnb = GaussianNB()

      # fit the model
      gnb.fit(x_train, y_train)

[20]: GaussianNB()
```

- Variabel y_pred digunakan untuk menyimpan nilai prediksi pada data testing (x_test) dengan menggunakan fungsi predict().

```
[21]: y_pred = gnb.predict(x_test)
```

- Import fungsi classification_report dari modul metrics pada library sklearn. Fungsi tersebut mengambil dua argumen yaitu y_test (nilai target sebenarnya) dan y_pred (prediksi nilai target). Outputnya akan menghasilkan classification report, di antaranya yaitu precision, recall, f1-score, dan support untuk setiap kelas.

```
[22]: from sklearn.metrics import classification_report
      print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.86      0.83        99
           1       0.71      0.62      0.66        55

    accuracy                           0.77       154
   macro avg       0.76      0.74      0.74       154
weighted avg       0.77      0.77      0.77       154
```

3.7. Decision Tree

- Membuat prediksi pada data testing dengan menggunakan pengklasifikasian decision tree pada data training. Import terlebih dahulu class DecisionTreeClassifier dari modul tree library sklearn yang disimpan pada variabel clf. Gunakan fungsi fit() untuk melatih model decision tree pada data training yaitu x_train (fitur) dan y_train (variabel target). Fungsi predict() digunakan untuk membuat prediksi pada data testing (x_test) dan disimpan pada variabel y_pred.

```
[23]: from sklearn.tree import DecisionTreeClassifier
      # create decision tree classifer object
      clf = DecisionTreeClassifier()

      # train decision tree classifer
      clf = clf.fit(x_train,y_train)

      # predict the response for test dataset
      y_pred = clf.predict(x_test)
```

- Menghitung nilai akurasi dari model prediksi dengan fungsi metrics.accuracy_score() yang mengambil dua argumen yaitu y_test (nilai target sebenarnya) dan y_pred (prediksi nilai target). Didapatkan keakuratan pengklasifikasian decision tree pada data testing yaitu 0.69.

```
[24]: # model accuracy, how often is the classifier correct?
      print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.6883116883116883
```

- Import fungsi classification_report dari modul metrics pada library sklearn. Fungsi tersebut mengambil dua argumen yaitu y_test (nilai target sebenarnya) dan y_pred (prediksi nilai target). Outputnya akan menghasilkan classification report, di antaranya yaitu precision, recall, f1-score, dan support untuk setiap kelas.

```
[25]: from sklearn.metrics import classification_report
      print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.77      0.76        99
           1       0.57      0.55      0.56        55

    accuracy                           0.69       154
   macro avg       0.66      0.66      0.66       154
weighted avg       0.69      0.69      0.69       154
```

3.8. Membandingkan performa model masing-masing algoritma

Berikut merupakan tabel perbandingan nilai weighted avg dari setiap algoritma:

Algoritma	Precision	Recall	F1-score	Support
Regresi Logistik	0.77	0.77	0.76	154
Naïve Bayes	0.77	0.77	0.77	154
Decision Tree	0.69	0.69	0.69	154

BAB 4: KESIMPULAN

Precison: Rasio true positives (TP) dengan jumlah total predicted positives (TP + FP). Ini mengukur proporsi contoh yang diberi label dengan benar oleh model sebagai positif dari semua contoh yang diberi label sebagai positif. Nilai presisi yang tinggi menunjukkan bahwa model mengidentifikasi contoh positif dengan benar dan tidak melabeli terlalu banyak contoh negative sebagai positif.

Recall: Rasio true positives (TP) dengan jumlah total actual positives (TP + FN). Ini mengukur proporsi contoh yang diberi label dengan benar oleh model sebagai positif dari semua contoh yang benar-benar positif. Nilai recall yang tinggi menunjukkan bahwa model mengidentifikasi sebagian besar instance positif dengan benar dan tidak kehilangan terlalu banyak instance positif.

F1-score: Ukuran kinerja model yang menggabungkan precision (presisi) dan recall (ingatan) dari model tersebut. Nilai f1-score berkisar antara 0 dan 1, di mana nilai 1 menunjukkan model yang sempurna dan nilai 0 menunjukkan model yang sangat buruk. Nilai f1-score yang lebih tinggi menunjukkan kinerja model yang lebih baik. $f1 - score = 2 \times \frac{(precision \times recall)}{(precision + recall)}$.

Support: Menunjukkan jumlah kemunculan setiap kelas pada data yang digunakan untuk melatih model. Oleh karena itu, nilai support yang lebih tinggi menunjukkan bahwa data pelatihan memiliki representasi yang lebih baik dari setiap kelas, yang pada gilirannya dapat membantu model untuk mempelajari pola yang lebih baik dan meningkatkan kinerjanya. Namun, nilai support pada classification report tidak dapat digunakan untuk mengevaluasi kinerja model secara langsung.

Dari tabel poin 3.8 dapat disimpulkan bahwa model algoritma Naïve Bayes paling baik untuk memprediksi antara kelas 0 dan 1, dikarenakan memiliki nilai precision, recall, dan f1-score yang cukup seimbang dan lebih tinggi dibandingkan model lainnya.

DAFTAR PUSTAKA

Admin. (2022, July 18). *Analisis Data Eksplorasi (ADE) - Pengertian, Jenis, dan Pentingnya*. Diambil kembali dari LP2M: https://lp2m.uma.ac.id/2022/07/18/analisis-data-eksplorasi-ade-pengertian-jenis-dan-pentingnya/

Afifah, L. (t.thn.). *Penjelasan Algoritma Decision Tree*. Diambil kembali dari ilmudatapy: https://ilmudatapy.com/penjelasan-algoritma-decision-tree/

Hidayat, A. (2015, February). *Regresi Logistik*. Diambil kembali dari Statiskian: https://www.statistikian.com/2015/02/regresi-logistik.html

ITATS, D. (2015, September 17). *Naive Bayes Classifier*. Diambil kembali dari Oiite: https://dosen.itats.ac.id/oiite/2015/09/17/naive-bayes-classifier/

K, G. N. (2021, July 28). *Kenali Algoritma Klasifikasi Machine Learning Terpopuler di Tahun 2021*. Diambil kembali dari DQLab: https://dqlab.id/kenali-algoritma-klasifikasi-machine-learning-terpopuler-di-tahun-2021

R., A. Y. (2018, May 20). *Statistical Machine Learning*. Diambil kembali dari UGM Menara Ilmu Machine Learning: https://machinelearning.mipa.ugm.ac.id/2018/05/20/statistical-machine-learning/

Ramadhan, M. F. (2018, December). *Sentimen Analisis Menggunakan Algoritma atau Model Naive Bayes*. Diambil kembali dari bisa.ai: https://www.bisa.ai/portofolio/detail/MTI3Nw#:~:text=Klasifikasi%20dalam%20machine%20learning%20adalah,learning%20atau%20pembelajaran%20yang%20diawasi.

Yovita. (2020, November 23). *Yuk Pahami Algoritma Machine Learning yang Cocok untuk Penelitianmu!* Diambil kembali dari DQLab: https://dqlab.id/pahami-algoritma-machine-learning-untuk-penelitian#:~:text=Regresi%20logistik%20merupakan%20algoritma%20untuk,fungsi%20non%20linear%20(sigmoid).