LAPORAN
PEMBELAJARAN MESIN
TUGAS BESAR TAHAP 2: CLASSIFICATION

Disusun untuk Memenuhi Tugas Besar Mata Kuliah Pembelajaran Mesin

Disusun Oleh:

Fadlan Akmal Ramadhan (1301190351)
Musthafa Zaki Bahar (1301190335)

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
2021

DAFTAR ISI

Kata Pengantar

Puji dan syukur penyusun panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya kepada kita semua. Shalawat dan salam semoga tetap terlimpah curah kepada junjungan kita Nabi Muhammad SAW. Kepada keluarganya, para sahabatnya dan kita semua sebagai umatnya yang setia sampai akhir zaman.

Alhamdulillah dengan pertolongan-Nya penyusun dapat menyelesaikan tugas besar tahap 02 yaitu classification untuk memenuhi tugas mata kuliah Pembelajaran Mesin.

Penulis sadar dalam penulisan laporan ini, penulis mendapatkan banyak bantuan, bimbingan dan saran dari berbagai pihak. Dalam kesempatan ini penulis mengucapkan terima kasih kepada Bapak Donni Richasdy, S.T.,M.T., selaku dosen mata kuliah Pembelajaran Mesin yang telah memberikan bimbingannya.

Penulis menyadari bahwa dalam penulisan laporan ini masih terdapat kekurangan dan jauh dari kata sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran agar laporan selanjutnya akan jauh lebih baik dari laporan ini.

Bekasi, 15 Desember 2021

Penulis

1. Formulasi Masalah

Classification adalah suatu teknik pemodelan dalam data mining yang digunakan untuk menemukan model atau kelompok hasil yang ada pada data untuk kepentingan tertentu. Teknik Classification masuk kedalam kategori Supervised Learning, yaitu jenis algoritma yang tidak dapat belajar sendiri, tetapi harus diberi contoh terlebih dahulu dengan cara memberi label pada dataset yang akan diolah. Label yang dimaksud adalah pada dataset telah diberikan nilai kebenaran yang akan dijadikan sebagai nilai target atau nilai acuan.

Pada tugas tahap 2 pembelajaran mesin, diminta untuk melakukan Classification (Supervised learning) dari data yang telah diberikan yaitu mengelompokkan pelanggan berdasarkan data pelanggan di dealer dengan memperhatikan feature tertarik. Dan dari hal tersebut mahasiswa dapat melakukan Classification diawali dengan proses eksplorasi dan persiapan data, pemodelan, evaluasi, eksperimen, dan diakhiri dengan memberikan kesimpulan dari observasi yang dilakukan.

2. Eksplorasi dan Persiapan Data

- Melakukan read data dari data yang telah diberikan.

```
[144] 1 data = pd.read_csv("https://raw.githubusercontent.com/mzakibhr/Tubes2_ML_Classification/main/kendaraan_train.csv")
      2 data_test = pd.read_csv("https://raw.githubusercontent.com/mzakibhr/Tubes2_ML_Classification/main/kendaraan_test.csv")
      3 data.head(5)
```

	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	1	Wanita	30.0	1.0	33.0	1.0	< 1 Tahun	Tidak	28029.0	152.0	97.0	0
1	2	Pria	48.0	1.0	39.0	0.0	> 2 Tahun	Pernah	25800.0	29.0	158.0	0
2	3	NaN	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	32733.0	160.0	119.0	0
3	4	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.0	0
4	5	Pria	50.0	1.0	35.0	0.0	> 2 Tahun	NaN	34857.0	88.0	194.0	0

- Mengecek null value yang pada data_train dan data_test
  Dari hasil pengecekan data_train, masih terdapat null value yang cukup banyak.

```
[145]  1 #Jumlah Missing Value
       2 print(data.isnull().sum())
       3 data.shape
```

```
id                      0
Jenis_Kelamin       14440
Umur                14214
SIM                 14404
Kode_Daerah         14306
Sudah_Asuransi      14229
Umur_Kendaraan      14275
Kendaraan_Rusak     14188
Premi               14569
Kanal_Penjualan     14299
Lama_Berlangganan   13992
Tertarik                0
dtype: int64
(285831, 12)
```

Untuk data_test tidak terdapat null value.

```
[146]  1 #Jumlah Missing Value
       2 print(data_test.isnull().sum())
       3 data_test.shape
```

```
Jenis_Kelamin       0
Umur                0
SIM                 0
Kode_Daerah         0
Sudah_Asuransi      0
Umur_Kendaraan      0
Kendaraan_Rusak     0
Premi               0
Kanal_Penjualan     0
Lama_Berlangganan   0
Tertarik            0
dtype: int64
(47639, 11)
```

- Melakukan drop feature id

```
[148]  1 #Drop Kolom id
       2 data = data.drop(axis=1, columns=["id"])
```

- Mengecek tipe data pada tiap kolom, dan didapati terdapat 3 kolom yang bukan merupakan data numerik sehingga diatasi dengan melakukan label encoding

```
[169]  1 #Mengecek tipe data tiap kolom
       2 data.dtypes
```

```
Jenis_Kelamin          object
Umur                  float64
SIM                   float64
Kode_Daerah           float64
Sudah_Asuransi        float64
Umur_Kendaraan         object
Kendaraan_Rusak        object
Premi                 float64
Kanal_Penjualan       float64
Lama_Berlangganan     float64
Tertarik                int64
dtype: object
```

- Melakukan label encoding untuk mengubah fitur data yang bukan numerik seperti Jenis_Kelamin, Umur_Kendaraan, dan Kendaraan_Rusak menjadi data numerik

```
[171]  1 #Kendaraan Rusak
       2 scale_Kendaraan_Rusak = {"Tidak":0,"Pernah":1}
       3 data["Kendaraan_Rusak"] = data["Kendaraan_Rusak"].replace(scale_Kendaraan_Rusak)
       4 data_test["Kendaraan_Rusak"] = data_test["Kendaraan_Rusak"].replace(scale_Kendaraan_Rusak)
       5
       6 #Jenis Kelamin
       7 scale_Jenis_Kelamin = {"Wanita":0,"Pria":1}
       8 data["Jenis_Kelamin"] = data["Jenis_Kelamin"].replace(scale_Jenis_Kelamin)
       9 data_test["Jenis_Kelamin"] = data_test["Jenis_Kelamin"].replace(scale_Jenis_Kelamin)
      10
      11 #Umur Kendaraan
      12 scale_Umur_Kendaraan = {"< 1 Tahun":0,"1-2 Tahun":1 ,"> 2 Tahun":2}
      13 data["Umur_Kendaraan"] = data["Umur_Kendaraan"].replace(scale_Umur_Kendaraan)
      14 data_test["Umur_Kendaraan"] = data_test["Umur_Kendaraan"].replace(scale_Umur_Kendaraan)
      15
      16 data.head(10)
```

Hasil setelah dilakukannya label encoding :

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	0.0	30.0	1.0	33.0	1.0	0.0	0.0	28029.0	152.0	97.0	0
1	1.0	48.0	1.0	39.0	0.0	2.0	1.0	25800.0	29.0	158.0	0
2	NaN	21.0	1.0	46.0	1.0	0.0	0.0	32733.0	160.0	119.0	0
3	0.0	58.0	1.0	48.0	0.0	1.0	0.0	2630.0	124.0	63.0	0
4	1.0	50.0	1.0	35.0	0.0	2.0	NaN	34857.0	88.0	194.0	0
5	1.0	21.0	1.0	35.0	1.0	0.0	0.0	22735.0	152.0	171.0	0
6	0.0	33.0	1.0	8.0	0.0	NaN	1.0	32435.0	124.0	215.0	1
7	1.0	23.0	NaN	28.0	1.0	0.0	0.0	26869.0	152.0	222.0	0
8	0.0	20.0	1.0	8.0	1.0	0.0	0.0	30786.0	160.0	31.0	0
9	NaN	54.0	1.0	29.0	0.0	2.0	1.0	88883.0	124.0	28.0	1

- Karena pada data_train masih terdapat cukup banyak null value, diatasi dengan mengisi nilai tersebut dengan mean/modus dari tiap kolom.

```
[173]  1 #Mengatasi Missing Value data train dengan mean/modus
       2 #Dengan Mean
       3 data['Umur'] = data['Umur'].fillna(data['Umur'].mean())
       4 data['Premi'] = data['Premi'].fillna(data['Premi'].mean())
       5 data['Lama_Berlangganan'] = data['Lama_Berlangganan'].fillna(data['Lama_Berlangganan'].mean())
       6
       7 #Dengan Modus
       8 data['Jenis_Kelamin'] = data['Jenis_Kelamin'].fillna(data['Jenis_Kelamin'].mode()[0])
       9 data['SIM'] = data['SIM'].fillna(data['SIM'].mode()[0])
      10 data['Kode_Daerah'] = data['Kode_Daerah'].fillna(data['Kode_Daerah'].mode()[0])
      11 data['Sudah_Asuransi'] = data['Sudah_Asuransi'].fillna(data['Sudah_Asuransi'].mode()[0])
      12 data['Umur_Kendaraan'] = data['Umur_Kendaraan'].fillna(data['Umur_Kendaraan'].mode()[0])
      13 data['Kendaraan_Rusak'] = data['Kendaraan_Rusak'].fillna(data['Kendaraan_Rusak'].mode()[0])
      14 data['Kanal_Penjualan'] = data['Kanal_Penjualan'].fillna(data['Kanal_Penjualan'].mode()[0])
```

3. Pemodelan

Pemodelan dilakukan menggunakan algoritma Naive Bayes dan Logistic Regression yang termasuk metode Classification, yaitu untuk menemukan model atau kelompok hasil yang ada pada data. Kedua algoritma tersebut mempunyai kemampuan menemukan model mengelompokkan data dalam jumlah yang cukup besar dengan waktu komputasi yang cukup efisien, sehingga dipilih untuk pengerjaan classification pada tugas ini.

- Algoritma **Naive Bayes** merupakan algoritma yang melakukan perhitungan peluang klasifikasi. Perhitungan pada algoritma ini diawali dengan menghitung probabilitas distribusi data menggunakan fungsi Gaussian Density. Kemudian dilakukan perhitungan mean dan varian untuk setiap kolom dan diubah menjadi array untuk tahap perhitungan selanjutnya.

```
 1 # Classicfication menggunakan algoritma Naive Bayes
 2 class NaiveBayesClassifier():
 3     def density(self, class_idx, x):
 4         # Implementasi Fungsi Gaussian Density untuk
 5         # Menghitung probabilitas distribusi data
 6         mean = self.mean[class_idx]
 7         var = self.var[class_idx]
 8         numerator = np.exp((-1/2)*((x-mean)**2)/(2 * var))
 9         denominator = np.sqrt(2 * np.pi * var)
10         probability = numerator / denominator
11         return probability
12
13     def statisticsCalculation(self, features, target):
14         # Menghitung mean dan varian untuk setiap kolom
15         # dan mengubah menjadi array numPy untuk perhitungan selanjutnya
16         self.mean = features.groupby(target).apply(np.mean).to_numpy()
17         self.var = features.groupby(target).apply(np.var).to_numpy()
18         return self.mean, self.var
```

- Algoritma Naive Bayes membutuhkan perhitungan posterior maksimal yang didapatkan dari teorema perhitungan Bayes. Perhitungan distribusi yang dilakukan dapat mengurangi perhitungan komputasi dengan cukup signifikan.

```
20    def calculatePrior(self, features, target):
21        # Menghitung peluang sebelumny
22        self.prior = (features.groupby(target).apply(lambda x: len(x)) / self.rows).to_numpy()
23        return self.prior
24
25    def calculatePosterior(self, x):
26        posteriors = []
27        # Menghitung probabilitas posterior pada setiap kelas
28        for i in range(self.count):
29            prior = np.log(self.prior[i])
30            conditional = np.sum(np.log(self.density(i, x)))
31            posterior = prior + conditional
32            posteriors.append(posterior)
33
34        # Mengembalikan kelas dengan probabilitas posterior tertinggi
35        return self.classes[np.argmax(posteriors)]
```

- Algoritma **Logistic Regression** merupakan sebuah teknik klasifikasi yang menggunakan fungsi logistic untuk membuat model variabel yang dependent. Perhitungan pada algoritma ini menggunakan perhitungan weights yang merupakan fungsi sigmoid. Dengan Logistic Regression dapat membuat fungsi hipotesis yang akan dimasukkan ke dataset biner. Fungsi yang digunakan adalah sigmoid function untuk menyatukan kurva dengan bentuk 'S' ketika dilakukan plot pada graf. Formula fungsi sigmoid:

$$y = 1/(1 + e^{-x})$$

- Implementasi dari fungsi sigmoid pada logistic regression adalah sebagai berikut:

```python
# Classicfication menggunakan Logistic Regression

class LogisticRegression() :
    def __init__(self, learning_rate, iterations) :
        self.learning_rate = learning_rate
        self.iterations = iterations

    # Fungsi untuk model training
    def fit(self, X, Y) :
        self.m, self.n = X.shape
        # Inisialisasi weight
        self.W = np.zeros(self.n)
        self.b = 0
        self.X = X
        self.Y = Y

        for i in range(self.iterations) :
            self.update_weights()
        return self

    # Fungsi untuk melakukan update weights
    def update_weights( self ) :
        A = 1 / (1 + np.exp( - (self.X.dot(self.W) + self.b)))

        # Menghitung Gradients
        tmp = (A - self.Y.T)
        tmp = np.reshape(tmp, self.m)
        dW = np.dot(self.X.T, tmp) / self.m
        db = np.sum(tmp) / self.m

        # update weights
        self.W = self.W - self.learning_rate * dW
        self.b = self.b - self.learning_rate * db
        return self
```

- Untuk melakukan perhitungan pada kedua algoritma dibutuhkan fungsi fit dan predict. Fungsi fit dibutuhkan untuk mendapatkan hasil dari perhitungan fungsi-fungsi yang ada dalam kelas kedua algoritma. Fungsi predict dibutuhkan untuk mengembalikan hasil perhitungan dalam bentuk array.

```python
    def fit(self, features, target):
        self.classes = np.unique(target)
        self.count = len(self.classes)
        self.feature_nums = features.shape[1]
        self.rows = features.shape[0]

        self.statisticsCalculation(features, target)
        self.calculatePrior(features, target)

    def predict(self, features):
        predicted = [self.calculatePosterior(f) for f in features.to_numpy()]
        return predicted
```

4. Evaluasi

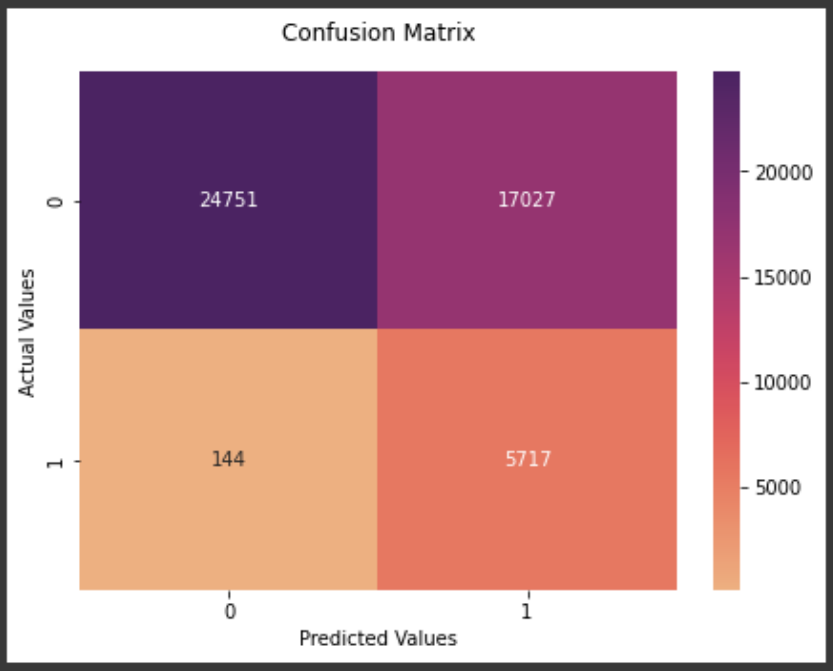
Evaluasi dilakukan menggunakan confusion matrix dan classification report yang dilakukan untuk mengetahui pemodelan mana yang lebih baik dari kedua pemodelan yang diimplementasikan.

- Terdapat beberapa perhitungan dalam confusion matrix, diantaranya adalah recall, precision, F1 Score, dan accuracy.
- Recall adalah perhitungan yang menjelaskan berapa banyak prediksi yang benar dalam semua hasil positif (formula: Recall = TP/(TP+FN)).
- Precision adalah perhitungan yang menjelaskan berapa banyak prediksi yang benar positif dari semua hasil (formula : Precision = TP/(TP+FP)).
- F1 score adalah perhitungan nilai rata-rata (mean) antara nilai recall dan precision (formula: F1 Score = (2 * Recall * Precision) / (Recall + Precision)).
- Accuracy adalah perhitungan jumlah prediksi yang benar dibandingkan dengan prediksi yang salah
  (formula: Accuracy = ((TP+TN)/(TP+TN+FP+FN)*100).
- Implementasi fungsi untuk memvisualisasikan confusion matrix

```
# Confusion Matrix
def confMatrix(y_test, prediction) :
    plt.figure(figsize=(7, 5))
    conf_matrix = confusion_matrix(y_test, prediction)
    sns.heatmap(conf_matrix, annot=True, xticklabels=[0, 1], yticklabels=[0, 1], cmap='flare', fmt='d')
    plt.ylabel('Actual Values')
    plt.xlabel('Predicted Values')
    plt.title(f'Confusion Matrix', pad=16)
    plt.show()
```

- Hasil evaluasi dari pemodelan menggunakan Naive Bayes

```
[176]  1 # Evaluasi Percobaan Naive Bayes menggunakan confusion matrix
       2 confMatrix(y_test, predictions)
       3 print("\n", classification_report(y_test, predictions), end="\n\n")
```



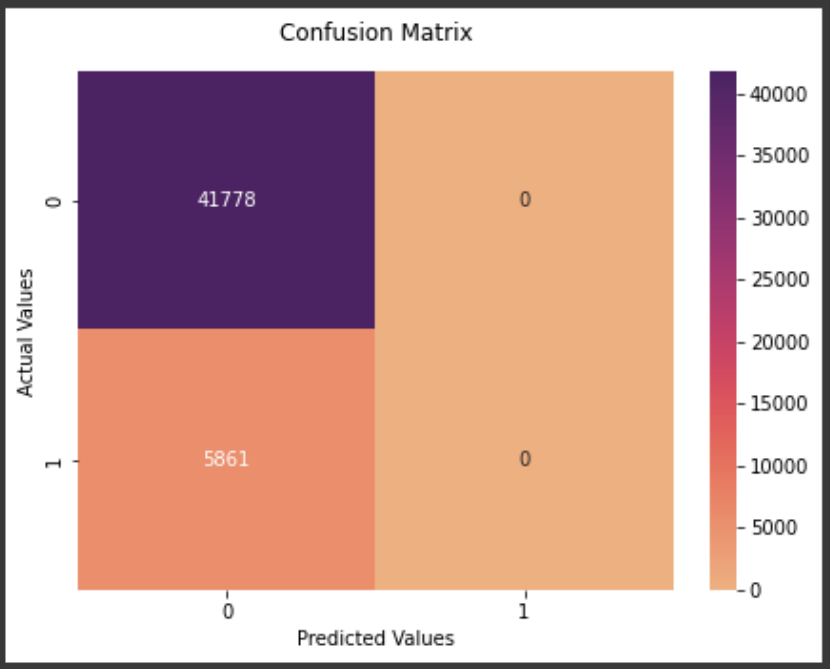

```
              precision    recall  f1-score   support

           0       0.99      0.59      0.74     41778
           1       0.25      0.98      0.40      5861

    accuracy                           0.64     47639
   macro avg       0.62      0.78      0.57     47639
weighted avg       0.90      0.64      0.70     47639
```

Untuk confusion matrix dari 47639 didapatkan,

a. True Positif = 5717 data atau 12%
b. True Negative = 24751 data atau 51%
c. False Negative = 144 data atau 0.003%
d. False Positif = 17027 data atau 35%

- Hasil evaluasi dari pemodelan menggunakan Logistic Regression

```
[178]  1 # Evaluasi Percobaan Logistic Regression menggunakan Confusion Matrix
       2 confMatrix(y_test, predictions)
       3 print("\n", classification_report(y_test, predictions), end="\n\n")
```




```
              precision    recall  f1-score   support

           0       0.88      1.00      0.93     41778
           1       0.00      0.00      0.00      5861

    accuracy                           0.88     47639
   macro avg       0.44      0.50      0.47     47639
weighted avg       0.77      0.88      0.82     47639
```

Untuk confusion matrix dari 47639 data didapatkan,

a. True Positif = 0 data atau 0%
b. True Negative = 41778 data atau 87%
c. False Negative = 5861 data atau 11%
d. False Positif = 0 data atau 0%

5. Eksperimen

Setelah semua persiapan sudah dilakukan yang dimulai dari persiapan dan eksplorasi data, pemodelan algoritma menggunakan dua algoritma yaitu Naive Bayes dan Logistic Regression. Pertama dilakukan eksperimen dari pemodelan menggunakan algoritma Naive Bayes,

```
[156]  1 # Percobaan Classification menggunakan Naive Bayes
       2
       3 # X_train dan y_train adalah data
       4 X_train = data.drop("Tertarik", axis=1)
       5 y_train = data["Tertarik"]
       6
       7 # X_test dan y_test adalah data test
       8 X_test = data_test.drop("Tertarik", axis=1)
       9 y_test = data_test["Tertarik"]
      10
      11 start = time.perf_counter()
      12 x = NaiveBayesClassifier()
      13 x.fit(X_train, y_train)
      14 predictions = x.predict(X_test)
      15 end = time.perf_counter()
      16
      17 # Hasil akurasi Naive Bayes
      18 print(f"Naive Bayes accuracy: {accuracy_score(y_test, predictions)}")
      19 print(f'Finished in {round(end-start, 3)} second(s)\n')
```

Untuk data yang digunakan dilakukan split secara manual yaitu X_train dan y_train menggunakan data_train, X_test dan y_test menggunakan data_test. Dari hasil percobaan menggunakan naive bayes didapatkan hasil accuracy 63% dengan waktu yang dibutuhkan 3.068 detik.

```
Naive Bayes accuracy: 0.6395600243497974
Finished in 3.068 second(s)
```

Untuk percobaan kedua menggunakan pemodelan Logistic Regression,

```
[177]  1 # Percobaan Classification menggunakan Logistic Regression
       2
       3 # X_train dan y_train adalah data
       4 X_train = data.drop("Tertarik", axis=1)
       5 y_train = data["Tertarik"]
       6
       7 # X_test dan y_test adalah data test
       8 X_test = data_test.drop("Tertarik", axis=1)
       9 y_test = data_test["Tertarik"]
      10
      11 start = time.perf_counter()
      12 x = LogisticRegression(learning_rate = 0.01, iterations = 1000)
      13 x.fit(X_train, y_train)
      14 predictions = x.predict(X_test)
      15 end = time.perf_counter()
      16
      17 # Hasil akurasi Naive Bayes
      18 print(f"Logistic Regression accuracy: {accuracy_score(y_test, predictions)}")
      19 print(f'Finished in {round(end-start, 3)} second(s)\n')
```

sama seperti percobaan pertama data yang digunakan dilakukan split secara manual yaitu X_train dan y_train menggunakan data_train, X_test dan y_test menggunakan

data_test. Dan dari hasil percobaan menggunakan logistic regression didapatkan hasil accuracy 87% dengan waktu yang dibutuhkan 23.436 detik.

```
Logistic Regression accuracy: 0.8769705493398267
Finished in 23.436 second(s)
```

6. **Kesimpulan**

Berdasarkan hasil percobaan dapat disimpulkan bahwa algoritma Naive Bayes dan Logistic Regression dapat digunakan untuk memecahkan masalah klasifikasi data. Dalam percobaan ini digunakan bahasa pemrograman python. Data yang telah diberikan perlu dilakukan eksplorasi dan persiapan terlebih dahulu karena terdapat null value yang cukup banyak, dan untuk permasalahan tersebut pada percobaan ini diatasi dengan mengisi nilai mean pada data null tersebut. Untuk pemodelan classification digunakan metode Naive Bayes dan Logistic Regression, karena kedua metode tersebut mempunyai kemampuan klasifikasi data yang cukup besar dengan waktu yang efisien.

Dari evaluasi yang dilakukan menggunakan confusion matrix didapatkan bahwa dalam melakukan klasifikasi, teknik Logistic Regression memiliki nilai akurasi sebesar 0.877 (87,7%), sedangkan algoritma Naive Bayes memiliki nilai akurasi sebesar 0.640 (64,0%). Sehingga dapat disimpulkan bahwa Teknik Logistic Regression lebih akurat daripada algoritma Naive Bayes dalam melakukan klasifikasi. Dalam hal waktu eksekusi algoritma, Naive Bayes menghasilkan waktu eksekusi 3.068 detik, sedangkan Logistic Regression menghasilkan waktu eksekusi 23.436 detik. Dapat disimpulkan bahwa Naive Bayes memiliki waktu eksekusi lebih cepat dibandingkan dengan Logistic Regression. Sehingga kesimpulan akhir yang didapatkan adalah Logistic Regression memiliki tingkat akurasi yang lebih tinggi dibandingkan dengan Naive Bayes, tetapi memiliki waktu eksekusi lebih lama dibandingkan Naive Bayes.

7. **Link Source Code dan Dataset**

- Berikut tautan source code, Github : https://github.com/mzakibhr/Tubes2_ML_Classification/blob/main/Tubes2_Classification.ipynb

- Berikut tautan dataset hasil eksplorasi, Github : https://github.com/mzakibhr/Tubes2_ML_Classification/tree/main/Hasil%20Dataset%20Prerprocessing

- Jika link Github Error : https://github.com/mzakibhr/Tubes2_ML_Classification

8. **Daftar Pustaka**

- medium.com/@bondansatrio99/apa-itu-classification-dalam-machine-learning
- towardsdatascience.com/machine-learning-classifiers

- towardsdatascience.com/understanding-confusion-matrix
- https://kambria.io/blog/logistic-regression-for-machine-learning/