

**LAPORAN TUGAS
ANALISIS BIG DATA**



Dosen Pengampu:
Ultach Enri, M.Kom.

Disusun Oleh:

Nabila Khairunisa	(2010631170101)
Naufal Ammar Hidayatulloh	(2010631170104)
Nabila Aulia Rahmah	(2010631170153)

**PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG
2023**

Studi Kasus

Adil ingin membuat sebuah program yaitu program pengelolaan kesehatan sehingga dapat menghindarkan pelanggan dari kemungkinan terkena serangan jantung. Untuk bisa sukses melakukan program ini, Adil harus menemukan pola, pelanggan asuransi dengan profile seperti apa yang kemungkinan terkena serangan jantung.

1. Untuk keperluan di atas, Adil mengambil data pelanggan asuransi sebanyak 138 orang (SeranganJantung.csv). Data pelanggan asuransi yang didapatkan Adil memiliki masalah karena beberapa ada yang kosong (hilang) dan juga ada yang noisy atau nilainya tidak wajar.
2. Analisis masalah di atas dan mari kita bantu Adil menyelesaikan masalahnya dimulai dari data preprocessing, pemodelan sampai evaluasi.
3. Perlu dicatat bahwa karena terbatasnya jumlah data, maka data yang kosong tidak diperbolehkan untuk dihapus, tapi harus diganti dengan nilai rata-rata dari data yang lain. Data noisy boleh dihapus.
4. Pada fase pemodelan lakukan berbagai hal sebagai berikut: Lakukan komparasi minimal algoritma data science, gunakan 10-fold *X Validation*. Berikan penjelasan yang mendukung anda memilih algoritma-algoritma tersebut!
5. Gunakan model yang dihasilkan untuk diterapkan di 10 data pelanggan baru yang masuk ke Adil (SeranganJantung-PelangganBaru.csv), analisis hasilnya.
6. Rangkumkan semua proses yang anda lakukan dalam bentuk docx. Ambil *printscreen* dari coding yang diperlukan untuk mempermudah penjelasan.
7. Keterangan dataset:

SeranganJantung.csv

Umur	0-150			
Status Pernikahan	0 = belum menikah	1 = menikah	2 = duda	3 = janda
Jenis Kelamin	0 = perempuan	1 = laki-laki		
Kategori Berat Badan	0 = normal	1 = kelebihan berat badan	2 = obesitas	
Kolesterol	0-400			
Pelatihan Pengelolaan Stress	0 = tidak ikut	1 = ikut		
Tingkat Stres	0-100			
Serangan Jantung	0 = No 1 = Yes			

Penyelesaian

Penyakit jantung adalah istilah umum untuk semua jenis gangguan yang mempengaruhi jantung. Penyakit jantung berarti sama dengan penyakit jantung tetapi tidak penyakit kardiovaskular. Penyakit kardiovaskular mengacu pada gangguan pembuluh darah dan jantung, sedangkan penyakit jantung mengacu hanya hati. Adapun beberapa tahapan penyelesaian diantaranya:

1. Data Preprocessing

Data preprocessing atau Data Pra Pemrosesan Data merupakan langkah persiapan data yang perlu dilakukan pada data mentah agar menjadi dataset dengan format yang lebih teratur, lebih mudah dipahami dan siap untuk proses data mining. Data mentah seringkali tidak lengkap dan memiliki format yang tidak konsisten. Tahap ini bertujuan untuk meminimalkan kesalahpahaman saat input dataset. Terdapat beberapa metode yang digunakan untuk pra-pemrosesan data, yaitu:

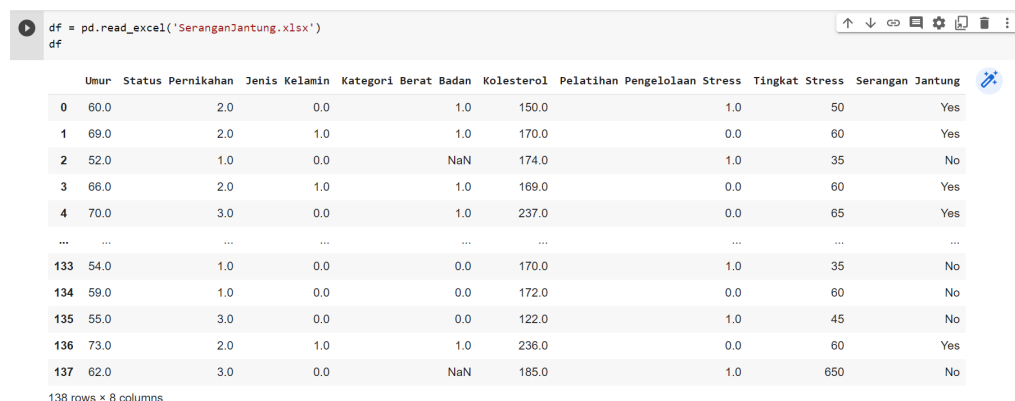
- Denoising, yang menghilangkan noise dari data.
- Imputasi, yang mensintesis data yang relevan secara statistik untuk nilai yang hilang (disini kita menggunakan nilai rata-rata untuk mengisi nilai yang kosong).

Berikut merupakan langkah-langkah dalam preprocessing:

- Menuliskan library yang dibutuhkan

```
[1] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

- Memanggil path dataset dan menampilkan deskripsi dataset



	Umur	Status	Pernikahan	Jenis Kelamin	Kategori Berat Badan	Kolesterol	Pelatihan	Pengelolaan Stress	Tingkat Stress	Serangan Jantung
0	60.0		2.0	0.0	1.0	150.0		1.0	50	Yes
1	69.0		2.0	1.0	1.0	170.0		0.0	60	Yes
2	52.0		1.0	0.0	NaN	174.0		1.0	35	No
3	66.0		2.0	1.0	1.0	169.0		0.0	60	Yes
4	70.0		3.0	0.0	1.0	237.0		0.0	65	Yes
...
133	54.0		1.0	0.0	0.0	170.0		1.0	35	No
134	59.0		1.0	0.0	0.0	172.0		0.0	60	No
135	55.0		3.0	0.0	0.0	122.0		1.0	45	No
136	73.0		2.0	1.0	1.0	236.0		0.0	60	Yes
137	62.0		3.0	0.0	NaN	185.0		1.0	650	No

138 rows × 8 columns

- Mengecek berapa missing value perkolom/atribut tabel dataset

```
df.isna().sum()
```

Umur	3
Status Pernikahan	2
Jenis Kelamin	1
Kategori Berat Badan	2
Kolesterol	3
Pelatihan Pengelolaan Stress	1
Tingkat Stress	0
Serangan Jantung	0
dtype: int64	

- Mengisi nilai kosong (*missing value*) yang ada pada sebuah dataset

```
[4] df = df.fillna(df.mean())
```

```
df.isna().sum()
```

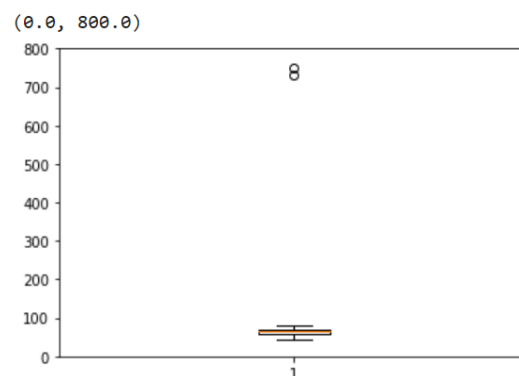
Umur	0
Status Pernikahan	0
Jenis Kelamin	0
Kategori Berat Badan	0
Kolesterol	0
Pelatihan Pengelolaan Stress	0
Tingkat Stress	0
Serangan Jantung	0
dtype: int64	

- Proses pencarian dan penghapusan data noisy

```
[6] mean = df['Umur'].mean()
std = df['Umur'].std()
threshold = 3 # Ambil nilai 3 deviasi standar sebagai threshold
noise = df[df['Umur'].abs() > (mean + threshold * std)]
noise
```

	Umur	Status Pernikahan	Jenis Kelamin	Kategori Berat Badan	Kolesterol	Pelatihan Pengelolaan Stress	Tingkat Stress	Serangan Jantung
112	750.0	2.000000	1.0	2.0	143.0	1.0	80	Yes
122	730.0	1.698529	1.0	2.0	143.0	1.0	80	Yes

```
[7] plt.boxplot(df["Umur"])
plt.ylim([0,800])
```

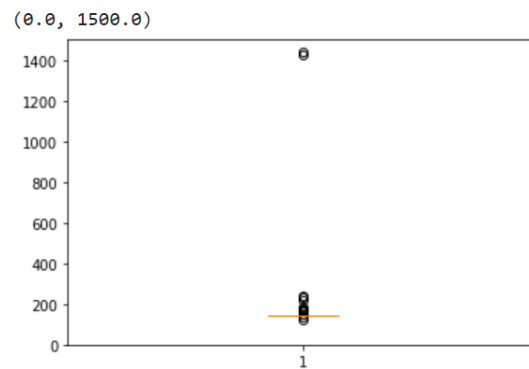


```
remove_noise1 = df.drop(df[df['Umur'].abs() > (mean + threshold * std)].index)
remove_noise1
```

	Umur	Status Pernikahan	Jenis Kelamin	Kategori Berat Badan	Kolesterol	Pelatihan Pengelolaan Stress	Tingkat Stress	Serangan Jantung
0	60.0	2.0	0.0	1.000000	150.0	1.0	50	Yes
1	69.0	2.0	1.0	1.000000	170.0	0.0	60	Yes
2	52.0	1.0	0.0	0.933824	174.0	1.0	35	No
3	66.0	2.0	1.0	1.000000	169.0	0.0	60	Yes
4	70.0	3.0	0.0	1.000000	237.0	0.0	65	Yes
...
133	54.0	1.0	0.0	0.000000	170.0	1.0	35	No
134	59.0	1.0	0.0	0.000000	172.0	0.0	60	No
135	55.0	3.0	0.0	0.000000	122.0	1.0	45	No
136	73.0	2.0	1.0	1.000000	236.0	0.0	60	Yes
137	62.0	3.0	0.0	0.933824	185.0	1.0	650	No

136 rows × 8 columns

```
[13] plt.boxplot(remove_noise1["Kolesterol"])
plt.ylim([0,1500])
```

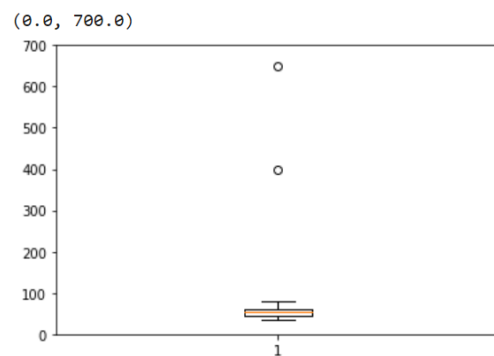


```
remove_noise2 = remove_noise1.drop(remove_noise1[remove_noise1['Kolesterol'].abs() > (mean + threshold * std)].index)
remove_noise2
```

	Umur	Status Pernikahan	Jenis Kelamin	Kategori Berat Badan	Kolesterol	Pelatihan Pengelolaan Stress	Tingkat Stress	Serangan Jantung
0	60.0	2.0	0.0	1.000000	150.0	1.0	50	Yes
1	69.0	2.0	1.0	1.000000	170.0	0.0	60	Yes
2	52.0	1.0	0.0	0.933824	174.0	1.0	35	No
3	66.0	2.0	1.0	1.000000	169.0	0.0	60	Yes
4	70.0	3.0	0.0	1.000000	237.0	0.0	65	Yes
...
133	54.0	1.0	0.0	0.000000	170.0	1.0	35	No
134	59.0	1.0	0.0	0.000000	172.0	0.0	60	No
135	55.0	3.0	0.0	0.000000	122.0	1.0	45	No
136	73.0	2.0	1.0	1.000000	236.0	0.0	60	Yes
137	62.0	3.0	0.0	0.933824	185.0	1.0	650	No

134 rows × 8 columns

```
[17] plt.boxplot(remove_noise2["Tingkat Stress"])
plt.ylim([0,700])
```



```
remove_noise3 = remove_noise2.drop(remove_noise2[remove_noise2['Tingkat Stress']].abs() > (mean + threshold * std)).index
remove_noise3
```

	Umur	Status Pernikahan	Jenis Kelamin	Kategori Berat Badan	Kolesterol	Pelatihan Pengelolaan Stress	Tingkat Stress	Serangan Jantung
0	60.0	2.0	0.0	1.000000	150.0	1.0	50	Yes
1	69.0	2.0	1.0	1.000000	170.0	0.0	60	Yes
2	52.0	1.0	0.0	0.933824	174.0	1.0	35	No
3	66.0	2.0	1.0	1.000000	169.0	0.0	60	Yes
4	70.0	3.0	0.0	1.000000	237.0	0.0	65	Yes
...
132	80.0	3.0	1.0	1.000000	222.0	0.0	80	Yes
133	54.0	1.0	0.0	0.000000	170.0	1.0	35	No
134	59.0	1.0	0.0	0.000000	172.0	0.0	60	No
135	55.0	3.0	0.0	0.000000	122.0	1.0	45	No
136	73.0	2.0	1.0	1.000000	236.0	0.0	60	Yes

132 rows x 8 columns

- Menampilkan 10 baris pertama pada dataset remove_noise3

```
remove_noise3.head(10)
```

	Umur	Status Pernikahan	Jenis Kelamin	Kategori Berat Badan	Kolesterol	Pelatihan Pengelolaan Stress	Tingkat Stress	Serangan Jantung
0	60.000000	2.0	0.0	1.000000	150.0	1.0	50	Yes
1	69.000000	2.0	1.0	1.000000	170.0	0.0	60	Yes
2	52.000000	1.0	0.0	0.933824	174.0	1.0	35	No
3	66.000000	2.0	1.0	1.000000	169.0	0.0	60	Yes
4	70.000000	3.0	0.0	1.000000	237.0	0.0	65	Yes
5	72.940741	1.0	0.0	0.000000	174.0	1.0	35	No
6	58.000000	2.0	1.0	0.000000	140.0	0.0	45	No
7	59.000000	2.0	1.0	0.000000	143.0	0.0	45	Yes
10	52.000000	1.0	0.0	0.000000	143.0	1.0	65	No
11	70.000000	2.0	1.0	1.000000	143.0	1.0	50	Yes

2. Data Modelling

Adapun pada tahapan modelling kami menggunakan 3 algoritma yaitu antara lain sebagai berikut:

1. Random Forest

Algoritma Random Forest merupakan sebuah metode pengklasifikasian atau regresi yang memanfaatkan sekumpulan pohon keputusan (*decision trees*) yang dibangun secara acak pada subset acak dari data pelatihan. Random Forest menggunakan teknik ensemble learning, yaitu menggabungkan beberapa model atau algoritma sederhana menjadi satu model yang lebih kompleks dan akurat.

Ada beberapa alasan mengapa kami menggunakan Algoritma Random Forest adalah sebagai berikut:

- Akurasi yang tinggi: Random forest dapat menghasilkan model yang akurat dalam memprediksi penyakit jantung dengan menggunakan kombinasi dari beberapa pohon keputusan (*decision tree*).
- Mengatasi overfitting: Random forest dapat mengatasi *overfitting* atau model yang terlalu kompleks dengan menggunakan teknik *bootstrapping* dan *subsampling* pada data latih dan variabel input.

- c. Stabilitas: Random forest dapat menghasilkan hasil prediksi yang stabil meskipun terdapat perubahan pada data latih atau input yang digunakan.
- d. Kemampuan memproses data yang kompleks: Random forest dapat memproses data yang kompleks dan mengatasi masalah multikolinearitas pada variabel input.

```
[23] from sklearn.ensemble import RandomForestClassifier
      from sklearn.model_selection import cross_val_score

      X = remove_noise3.iloc[:, :-1].values
      y = remove_noise3.iloc[:, -1].values
```

```
▶ model = RandomForestClassifier()
  scores = cross_val_score(model, X, y, cv=10)

  # Print results
  print("Mean Accuracy: %.2f%%" % (scores.mean()*100.0))
```

```
☞ Mean Accuracy: 97.03%
```

```
[25] model.fit(X, y)
```

```
▼ RandomForestClassifier
RandomForestClassifier()
```

```
[26] import joblib
      joblib.dump(model, 'rf_model.pkl')

      ['rf_model.pkl']
```

```

▶ new_data = pd.read_excel('SeranganJantung-PelangganBaru.xlsx')

rf_model = joblib.load('rf_model.pkl')
predictions = rf_model.predict(new_data)

for i in range(len(predictions)):
    if predictions[i] == "No":
        print(f>Data ke-{i+1}: Tidak Beresiko Sakit Jantung")
    else:
        print(f>Data ke-{i+1}: Beresiko Sakit Jantung")

```

□ Data ke-1: Tidak Beresiko Sakit Jantung
 Data ke-2: Beresiko Sakit Jantung
 Data ke-3: Tidak Beresiko Sakit Jantung
 Data ke-4: Beresiko Sakit Jantung
 Data ke-5: Beresiko Sakit Jantung
 Data ke-6: Tidak Beresiko Sakit Jantung
 Data ke-7: Tidak Beresiko Sakit Jantung
 Data ke-8: Tidak Beresiko Sakit Jantung
 Data ke-9: Beresiko Sakit Jantung
 Data ke-10: Tidak Beresiko Sakit Jantung

Adapun hasil yang diperoleh dengan menggunakan algoritma Random Forest yaitu sebanyak 60% pelanggan baru tidak beresiko mengidap penyakit jantung dan 40 % pelanggan baru beresiko mengidap penyakit jantung dengan akurasi sebesar 97,03%

2. K-Nearest Neighbour (KNN)

Algoritma K-Nearest Neighbors (KNN) merupakan salah satu algoritma pembelajaran mesin (machine learning) yang paling sederhana dan populer dalam klasifikasi dan regresi. Algoritma ini bekerja dengan mencari K tetangga terdekat dari sebuah data uji (query data) pada dataset latihan (training data), kemudian menentukan label atau nilai prediksi untuk data uji tersebut berdasarkan mayoritas label atau nilai dari tetangga terdekat.

Ada beberapa alasan mengapa kami menggunakan Algoritma KNN adalah sebagai berikut:

- Kemudahan dalam implementasi: K-NN merupakan salah satu algoritma pembelajaran mesin yang relatif mudah diimplementasikan, terutama pada data yang relatif kecil.
- Tidak memerlukan asumsi terhadap distribusi data: K-NN tidak memerlukan asumsi tertentu terhadap distribusi data. Hal ini menjadikannya cocok untuk digunakan pada data yang tidak berdistribusi normal atau terdapat outlier.
- Kemampuan menangani data yang kompleks: K-NN dapat menangani data yang kompleks dan beragam, serta dapat mengatasi masalah multikolinearitas pada variabel input.

- d. Kemampuan adaptasi dengan cepat: K-NN dapat dengan cepat mengadaptasi perubahan pada data latih, sehingga dapat menghasilkan hasil prediksi yang akurat dan *up-to-date*.

```
from sklearn.neighbors import KNeighborsClassifier

X = remove_noise3.iloc[:, :-1].values
y = remove_noise3.iloc[:, -1].values
```

```
[29] model = KNeighborsClassifier()
      scores = cross_val_score(model, X, y, cv=10)

      # Print results
      print("Mean Accuracy: %.2f%%" % (scores.mean()*100.0))
```

Mean Accuracy: 73.63%

```
[30] model.fit(X, y)
```

```
▼ KNeighborsClassifier
KNeighborsClassifier()
```

```
[31] joblib.dump(model, 'knn_model.pkl')
```

['knn_model.pkl']

```
▶ knn_model = joblib.load('knn_model.pkl')
  predictions = knn_model.predict(new_data)

  for i in range(len(predictions)):
      if predictions[i] == "No":
          print(f>Data ke-{i+1}: Tidak Beresiko Sakit Jantung")
      else:
          print(f>Data ke-{i+1}: Beresiko Sakit Jantung")
```

```
↳ Data ke-1: Tidak Beresiko Sakit Jantung
   Data ke-2: Tidak Beresiko Sakit Jantung
   Data ke-3: Tidak Beresiko Sakit Jantung
   Data ke-4: Beresiko Sakit Jantung
   Data ke-5: Tidak Beresiko Sakit Jantung
   Data ke-6: Tidak Beresiko Sakit Jantung
   Data ke-7: Tidak Beresiko Sakit Jantung
   Data ke-8: Tidak Beresiko Sakit Jantung
   Data ke-9: Tidak Beresiko Sakit Jantung
   Data ke-10: Tidak Beresiko Sakit Jantung
```

Adapun hasil yang diperoleh dengan menggunakan algoritma K-Nearest Neighbors (KNN) yaitu sebanyak 90% pelanggan baru tidak beresiko mengidap penyakit jantung dan 10% pelanggan baru beresiko mengidap penyakit jantung dengan akurasi sebesar 73,63%.

3. Support Vector Machines (SVM)

Support Vector Machine (SVM) merupakan salah satu algoritma pembelajaran mesin yang digunakan untuk masalah klasifikasi dan regresi. SVM bekerja dengan mencari hyperplane terbaik yang dapat memisahkan antara dua kelas data. Hyperplane adalah garis atau bidang yang memisahkan antara data dengan label yang berbeda.

Ada beberapa alasan mengapa kami menggunakan Algoritma SVM adalah sebagai berikut:

- a. Kemampuan mengatasi data yang tidak linier: SVM dapat mengatasi data yang tidak linier dan tidak memiliki pemisah yang jelas, sehingga cocok untuk digunakan pada data medis yang kompleks dan beragam.
- b. Kemampuan menangani data yang berdimensi tinggi: SVM dapat menangani data dengan dimensi yang tinggi, sehingga cocok untuk digunakan pada data medis yang memiliki banyak variabel input.
- c. Kemampuan menangani kasus dengan jumlah sampel yang lebih sedikit: SVM dapat digunakan pada kasus dengan jumlah sampel yang lebih sedikit, sehingga cocok untuk digunakan pada data medis yang relatif sulit didapatkan atau terbatas.
- d. Toleransi terhadap outlier: SVM dapat menangani outlier dan noise pada data, sehingga dapat menghasilkan model yang lebih robust dan akurat.

```
[33] from sklearn.svm import SVC

X = remove_noise3.iloc[:, :-1].values
y = remove_noise3.iloc[:, -1].values
```

```
[34] model = SVC(kernel='linear')
      scores = cross_val_score(model, X, y, cv=10)

      # Print results
      print("Mean Accuracy: %.2f%%" % (scores.mean()*100.0))
```

Mean Accuracy: 90.22%

```
[35] model.fit(X, y)
```

▼ SVC
SVC(kernel='linear')

```
[36] joblib.dump(model, 'svm_model.pkl')
```

['svm_model.pkl']

```
▶ svm_model = joblib.load('svm_model.pkl')
  predictions = svm_model.predict(new_data)

  for i in range(len(predictions)):
      if predictions[i] == "No":
          print(f>Data ke-{i+1}: Tidak Beresiko Sakit Jantung")
      else:
          print(f>Data ke-{i+1}: Beresiko Sakit Jantung")
```

```
↳ Data ke-1: Tidak Beresiko Sakit Jantung
  Data ke-2: Beresiko Sakit Jantung
  Data ke-3: Tidak Beresiko Sakit Jantung
  Data ke-4: Beresiko Sakit Jantung
  Data ke-5: Beresiko Sakit Jantung
  Data ke-6: Tidak Beresiko Sakit Jantung
  Data ke-7: Tidak Beresiko Sakit Jantung
  Data ke-8: Tidak Beresiko Sakit Jantung
  Data ke-9: Beresiko Sakit Jantung
  Data ke-10: Tidak Beresiko Sakit Jantung
```

Adapun hasil yang diperoleh dengan menggunakan algoritma Support Vector Machine (SVM) yaitu sebanyak 60% pelanggan baru tidak beresiko mengidap penyakit jantung dan 40% pelanggan baru beresiko mengidap penyakit jantung dengan akurasi sebesar 90.22%.

Link Github: <https://github.com/naufalamr17/olehOlehPert4BD>

Daftar Pustaka

- Purnomo, W. G., & Purnomo, P. P. (2017). Akurasi Text Mining Menggunakan Algoritma K-Nearest Neighbour pada Data Content Berita SMS. *Format*, 6(1), 1-13.
- Agustina, D. A., Subanti, S., & Zukhronah, E. (2021). Implementasi Text Mining Pada Analisis Sentimen Pengguna Twitter Terhadap Marketplace di Indonesia Menggunakan Algoritma Support Vector Machine. *Indonesian Journal of Applied Statistics*, 3(2), 109-122.
- Annisa, R. (2019). Analisis Komparasi Algoritma Klasifikasi Data Mining Untuk Prediksi Penderita Penyakit Jantung. *JTIK (Jurnal Teknik Informatika Kaputama)*, 3(1), 22-28.