LAPORAN TUGAS PEMROGRAMAN
PENGANTAR KECERDASAN BUATAN
LEARNING

Disusun Oleh :
Kelompok 1
Anyelir Belia Azzahra (1301200048)
Risma Amaliyah Mahmudah (1301204087)
Muhammad Rafi Irfansyah (1301204500)

Program Studi Informasi
Fakultas Informatika
Telkom University
2021/2022

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pada Tugas Besar kali ini, kami mengimplementasikan model dengan metode KNN. K-nearest neighbors atau KNN adalah algoritma yang berfungsi untuk melakukan klasifikasi suatu data berdasarkan data pembelajaran (*train data sets*), yang diambil dari k tetangga terdekatnya (*nearest neighbors*). Dengan k merupakan banyaknya tetangga terdekat.

KNN memiliki beberapa kelebihan yaitu bahwa dia tangguh terhadap training data yang noisy dan efektif apabila data latih nya besar. Sedangkan kelemahan dari KNN adalah KNN perlu menentukan nilai dari parameter K (jumlah dari tetangga terdekat), pembelajaran berdasarkan jarak tidak jelas mengenai jenis jarak apa yang harus digunakan dan atribut mana yang harus digunakan untuk mendapatkan hasil yang terbaik, dan biaya komputasi cukup tinggi karena diperlukan perhitungan jarak dari tiap sample uji pada keseluruhan sample latih.

Secara umum, cara kerja algoritma KNN adalah sebagai berikut.

1. Tentukan jumlah tetangga (K) yang akan digunakan untuk pertimbangan penentuan kelas.
2. Hitung jarak dari data baru ke masing-masing data point di dataset.
3. Ambil sejumlah K data dengan jarak terdekat, kemudian tentukan kelas dari data baru tersebut.

1.2. Deskripsi Tugas Besar

Deskripsi tugas besar kami adalah sebagai berikut.

Diberikan file traintest.xlsx yang terdiri dari dua sheet: train dan test, yang berisi dataset untuk problem klasifikasi biner (binary classification). Setiap record atau baris data dalam dataset tersebut secara umum terdiri dari nomor baris data (*id*), fitur input (*x*1 sampai *x*3), dan output kelas (*y*). Fitur input terdiri dari nilai-nilai integer dalam range tertentu untuk setiap fitur. Sedangkan output kelas bernilai biner (0 atau 1).

id	x1	x2	x3	y
1	60	64	0	1
2	54	60	11	0
3	65	62	22	0
4	34	60	0	1
5	38	69	21	0

Sheet train berisi 296 baris data, lengkap dengan target output kelas (*y*). Gunakan sheet ini untuk tahap pemodelan atau pelatihan (training) model sesuai metode yang Anda gunakan. Adapun sheet test berisi 10 baris data, dengan output kelas (*y*) yang disembunyikan. Gunakan sheet ini untuk tahap pengujian (testing) model yang sudah dilatih. Nantinya output program Anda untuk data uji ini akan dicocokkan dengan target atau kelas sesungguhnya.

BAB II
IMPLEMENTASI

2.1. Membaca Data Latih atau Uji

Pada tugas kali ini kami mengimport file traintest.xlsx yang telah diberikan dalam bentuk drive. Di sini juga dilakukan pembacaan data file traintest tadi dengan membaca file excel untuk menyimpan di df dan drop ‘id’ karena tidak dipakai. Sehingga akan muncul tabel seperti gambar berikut. Tabel tersebut sudah sesuai dengan yang ada pada data traintest.xlsx.

```
!gdown --id 1oHwMSB3NeKYMpQCjwuSd7dbb6zpAYhyj
```

```
/usr/local/lib/python3.7/dist-packages/gdown/cli.py:131: FutureWarning: Option `--id` was deprecated in version 4.3.1 and will be removed in 5.0. You don't need to pass it anymore to use a file ID.
  category=FutureWarning,
Downloading...
From: https://drive.google.com/uc?id=1oHwMSB3NeKYMpQCjwuSd7dbb6zpAYhyj
To: /content/traintest.xlsx
100% 17.7k/17.7k [00:00<00:00, 20.0MB/s]
```

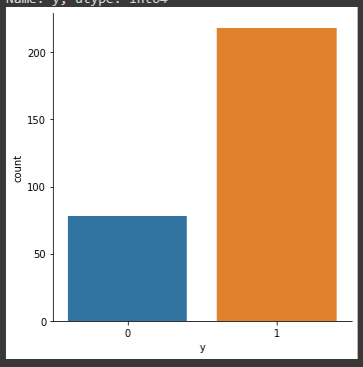
```
df = pd.read_excel('./traintest.xlsx')
df = df.drop('id', axis=1)

df.head()
```

	x1	x2	x3	y
0	60	64	0	1
1	54	60	11	0
2	65	62	22	0
3	34	60	0	1
4	38	69	21	0

Membaca output kelas (y) dalam df, kemudian dibuat plot seperti dibawah ini, diperoleh biner 1 sebanyak 218 dan biner 0 sebanyak 78. Kemudian mendeskripsikan x_data yang berisi data fitur input (x) saja. Kemudian mendeskripsikan y_data yang berisi output kelas (y) saja.

```
[ ] print(df['y'].value_counts())
    sns.catplot(x='y', data=df, kind='count')
    plt.show()
```

```
1    218
0     78
Name: y, dtype: int64
```




```
[ ] x_data = df.drop("y", axis=1).to_numpy()
    y_data = df["y"].to_numpy()
```

Berikut merupakan output dari data x1, x2, x3

```
x_data
[63, 60,  1],
[44, 61,  0],
[59, 64,  0],
[44, 63, 19],
[39, 63,  4],
[46, 69,  3],
[47, 66, 12],
[51, 59,  3],
[74, 63,  0],
[54, 62,  0],
[55, 58,  0],
[44, 67, 16],
[73, 68,  0],
[69, 65,  0],
[30, 62,  3],
[54, 60,  3],
[66, 68,  0],
[58, 58,  0],
[41, 65,  0],
[38, 67,  5],
[43, 66,  4],
[37, 58,  0],
[55, 58,  1],
[47, 66,  0],
[52, 62,  1],
[47, 61,  0],
[50, 61,  0],
[66, 58,  0],
[45, 68,  0],
[35, 63,  0],
[55, 67,  1],
[44, 61,  0],
[56, 60,  0],
[72, 67,  3],
```

Dan berikut merupakan output dari data y yang berisi array antara 1 atau 0 sesuai data pada file.

```
[ ] y_data

array([1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 1])
```

Berikut ini merupakan Resample Training set menggunakan over sampling, fungsi untuk mendapatkan hasil yang lebih bagus atau menyeimbangkan antara x_data dan y_data.

```
[ ] x_data, y_data = SMOTE().fit_resample(x_data, y_data)
```

Kemudian dibuat tabel atau frame baru untuk y_data, dan hasilnya adalah tabel y berdasarkan id sebagai berikut.

```
df_new = pd.DataFrame()
df_new['y'] = y_data
df_new
```

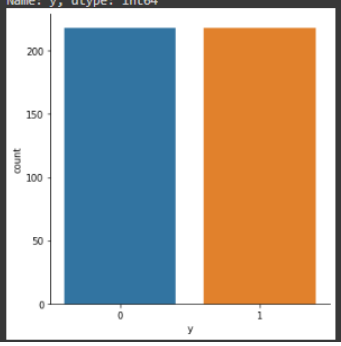
	y
0	1
1	0
2	0
3	1
4	0
...	...
431	0
432	0
433	0
434	0
435	0

436 rows × 1 columns

Berikut ini plot untuk kelas y setelah dilakukan *resample* menggunakan over sampling atau penyeimbangan data. Sehingga akan menghasilkan plot yang seimbang atau setara antara 0 dan 1 (tidak hanya condong ke 1 seperti sebelumnya) dengan biner 0 sebesar 218 dan 1 sebesar 218 pula.

```
print(df_new['y'].value_counts())
sns.catplot(x='y', data=df_new, kind='count')
plt.show()
```

```
1    218
0    218
Name: y, dtype: int64
```



Kemudian kita memisahkan data x_train, x_test, y_train, y_test dengan data train sebesar 80%, dan data test sebesar 20%. Dan akan menghasilkan hasil yang random.

```
[ ]  x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.2)
```

2.2. Pelatihan atau Training Model

Pada tugas kali ini, kami menggunakan metode KNN dengan kode program sebagai berikut yang digunakan untuk mencari jarak terdekat untuk setiap nilai x yang sudah dimapping dalam suatu kolom.

Fungsi train digunakan untuk mendeskripsikan x dan y, dimana merupakan data suatu kolom dan y merupakan label.

Fungsi prediksi dilakukan perhitungan jarak untuk setiap kolom. Perhitungannya menggunakan metode *Euclidean Distance* dengan cara sebagai berikut :

1) Hitung jarak ke semua data training
2) Urutkan berdasarkan jarak terdekat
3) Ambil label k terbaik
4) Voting yang paling banyak
5) Menggunakan fungsi jarak yang mengembalikan nilai jarak menggunakan metode *Euclidean Distance.*

```
[ ]  class KNN:
       def __init__(self, k=5):
         self.K = k

       def train(self, X, y): # x merupakan data suatu kolom, y merupakan label
         self.X_train = X
         self.y_train = y

       def predict(self, X):
         y_prediksi = []
         for i in range(len(X)):
           y_prediksi.append(self._prediksi(X[i]))

         return np.array(y_prediksi)

       def _prediksi(self, x):

         # 1. hitung jarak ke semua data training
         jarak_titik = [self.jarak(x, x_train) for x_train in self.X_train]

         # 2. urutkan berdasrkan jarak terdekat
         k_terbaik = np.argsort(jarak_titik)[:self.K]

         # 3. ambil label k terbaik
         label_k_terbaik = [self.y_train[i] for i in k_terbaik]

         # 4. voting yang paling banyak
         hasil_voting = Counter(label_k_terbaik).most_common(1)

         return hasil_voting[0][0]

       def jarak(self, x1, x2):
         # Euclidean Distance
         return np.sqrt(np.sum((x1-x2)**2))
```

2.3. Menyimpan Model Hasil Training

Pada tahap ini, kami melakukan penyimpanan model hasil training yang disimpan pada variabel model_knn dengan menggunakan k = 3. Nilai k merupakan tetangga terdekat yang digunakan untuk mencari akurasi yang paling tinggi. Kemudian melakukan training menggunakan “model_knn.train(x_train, y_train)”

```
[ ]  model_knn = KNN(k=3)
     model_knn.train(x_train, y_train)
```

Di sini kami melakukan prediksi data yang hasilnya akan disimpan pada variabel hasil_knn. Lalu hasil tersebut akan ditampilkan seperti berikut.

```
[ ]  hasil_knn = model_knn.predict(x_test)

[ ]  print(hasil_knn)

     [0 1 1 1 1 1 1 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0
      1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 0 1 0
      0 1 0 1 0 1 1 0 1 0 1 0 1 1]
```

2.4. Pengujian atau Testing Model

Pengujian atau testing model adalah untuk membandingkan hasil prediksi di atas dengan data sebenarnya (y_test).

Data y_test adalah sebagai berikut.

```
[ ]  y_test

     array([1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0,
            0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0,
            1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
            0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1])
```

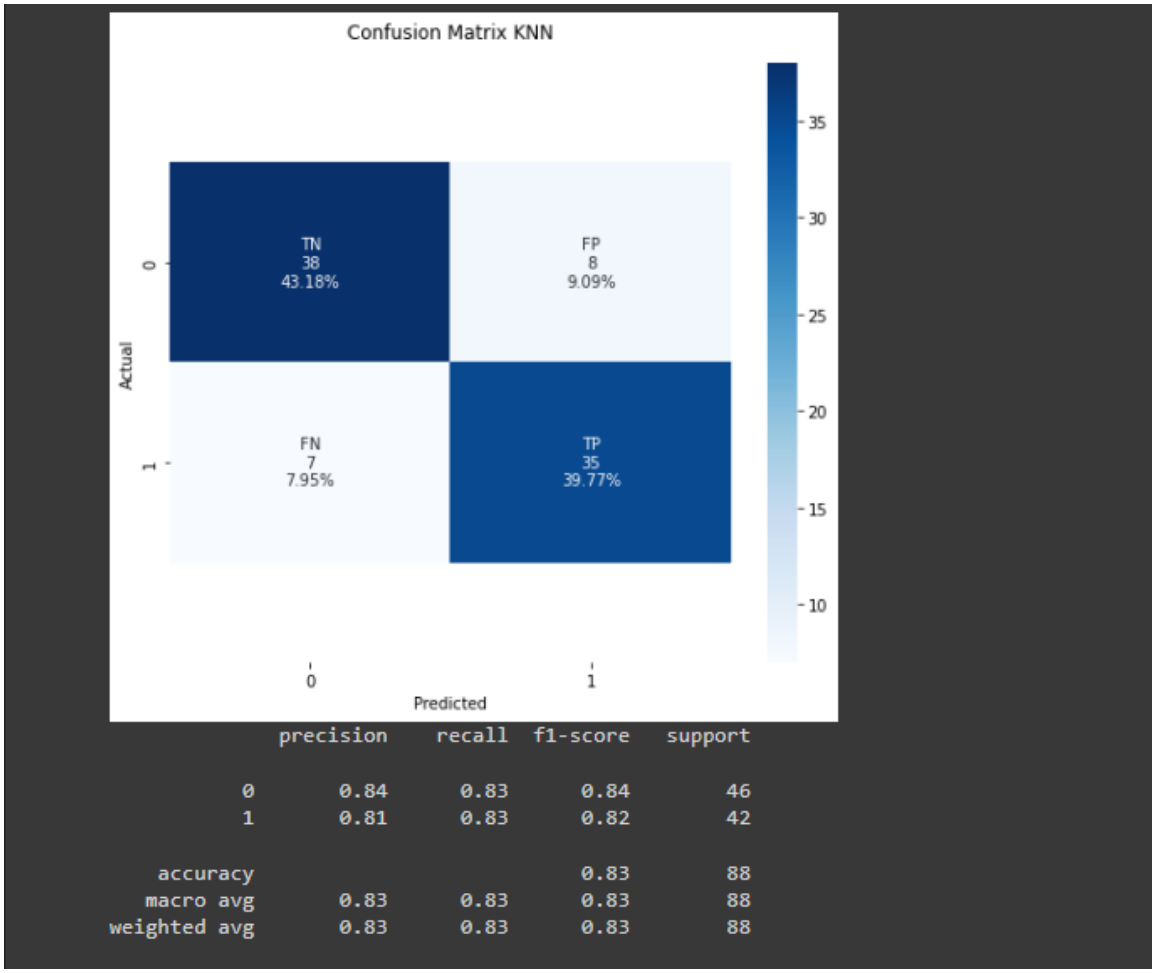
Berdasarkan hasil prediksi (hasil_knn) dan data sebenarnya (y_test) diketahui bahwa hasilnya tidak selalu sama, dikarenakan hasil prediksi diperoleh dari hasil voting secara random.

2.5. Evaluasi Model

Evaluasi model untuk melihat model terbaik menggunakan confusion_matrix. Kemudian mengelompokkan *True Negative* (TN), *False Positive* (FP), *False Negative* (FN), dan *True Positive* (TP). Dan akan menampilkan klasifikasi dari data y_test dan hasil_knn.

```
conf_matrix_knn = confusion_matrix(y_test, hasil_knn)

plt.figure(figsize=(8, 7))

group_names = ['TN', 'FP', 'FN', 'TP']
group_counts = ["{0:0.0f}".format(value) for value in conf_matrix_knn.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in conf_matrix_knn.flatten() / np.sum(conf_matrix_knn)]

labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in zip(group_names, group_counts, group_percentages)]
labels = np.asarray(labels).reshape(2,2)

ax = sns.heatmap(conf_matrix_knn, annot=labels, xticklabels=[0, 1], yticklabels=[0, 1], cmap='Blues', fmt='')
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)

plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix KNN', pad=16)
plt.show()

print(classification_report(y_test, hasil_knn))
```

Sehingga didapatkan klasifikasi confusion matrix KNN sebagai berikut.



	precision	recall	f1-score	support
0	0.84	0.83	0.84	46
1	0.81	0.83	0.82	42
accuracy			0.83	88
macro avg	0.83	0.83	0.83	88
weighted avg	0.83	0.83	0.83	88

Di mana,

1) precision merupakan berapa banyak instansi yang dipilih yang relevan,
2) *recall* merupakan berapa banyak instansi relevan yang dipilih, serta
3) f1-score merupakan rata-rata harmonik *precision* dan *recall*.

Berdasarkan hasil di atas, didapatkan akurasinya sebesar 0.83 dan presentase *precision*, *recall*, dan *f1-score* antara biner 0 dan 1 tidak terlalu jauh. Persentase pada *True Negative* (TN) dan *True Positive* (TP) tidak terlalu jauh atau seimbang. Ini sudah termasuk bagus, karena kita sudah bisa memprediksi 2 model.

Fungsi di bawah digunakan untuk menghitung nilai akurasi data sebenarnya dan data prediksi. Kemudian menampilkan nilai akurasi yang diperoleh berdasarkan y_test dan hasil_knn yaitu sebesar 82,9.

```
[ ]  def accuracy_metric(actual, predicted):
       correct = 0

       for i in range(len(actual)):
         if actual[i] == predicted[i]:
           correct = correct + 1

       return correct / float(len(actual)) * 100.0
```

```
[ ]  print("Accuracy : ", accuracy_metric(y_test, hasil_knn))
```

```
Accuracy :  82.95454545454545
```

2.6. Hasil Prediksi dari Model

Berikut ini variabel df_test untuk membaca data pada file traintest.xlsx bagian test. Lalu ditampilkan 5 data teratas dari data test tersebut.

```
[ ]  df_test = pd.read_excel('./traintest.xlsx', 'test')
     df_test.head()
```

	id	x1	x2	x3	y
0	297	43	59	2	?
1	298	67	66	0	?
2	299	58	60	3	?
3	300	49	63	3	?
4	301	45	60	0	?

Kemudian berikut merupakan df_id untuk membaca data test bagian id. Kemudian diambil 5 data pertama dari data test bagian id menggunakan df_id.head. Untuk df_out digunakan untuk membaca data test bagian x1,x2,x3. Lalu akan ditampilkan 5 data pertama dari data test bagian x1,x2,x3 tersebut menggunakan df_out.head.

```
[ ] df_id = df_test[['id']]
    df_out = df_test[['x1', 'x2', 'x3']]
```

```
[ ] df_id.head()
```

	id
0	297
1	298
2	299
3	300
4	301

```
[ ] df_out.head()
```

	x1	x2	x3
0	43	59	2
1	67	66	0
2	58	60	3
3	49	63	3
4	45	60	0

Di sini akan ditampilkan hasil data prediksi yang didapatkan dari pemanggilan fungsi *predict*. print(hasil_prediksi) digunakan untuk menampilkan bukti dari isi y. Hasilnya akan selalu berbeda, karena bersifat random.

```
[ ] df_out_arr = df_out.to_numpy()
    hasil_prediksi = model_knn.predict(df_out_arr)
```

```
[ ] print(hasil_prediksi)

    [1 1 0 1 1 1 0 0 1 1]
```

Untuk df_out.insert digunakan untuk memasukkan hasil prediksi ke kolom y, dan memasukkan df_id ke kolom id. Sehingga didapatkan tabel untuk data traintes.xlsx bagian test seperti berikut.

```
[ ]  df_out.insert(3, 'y', hasil_prediksi)
     df_out.insert(0, "id", df_id)
     df_out
```

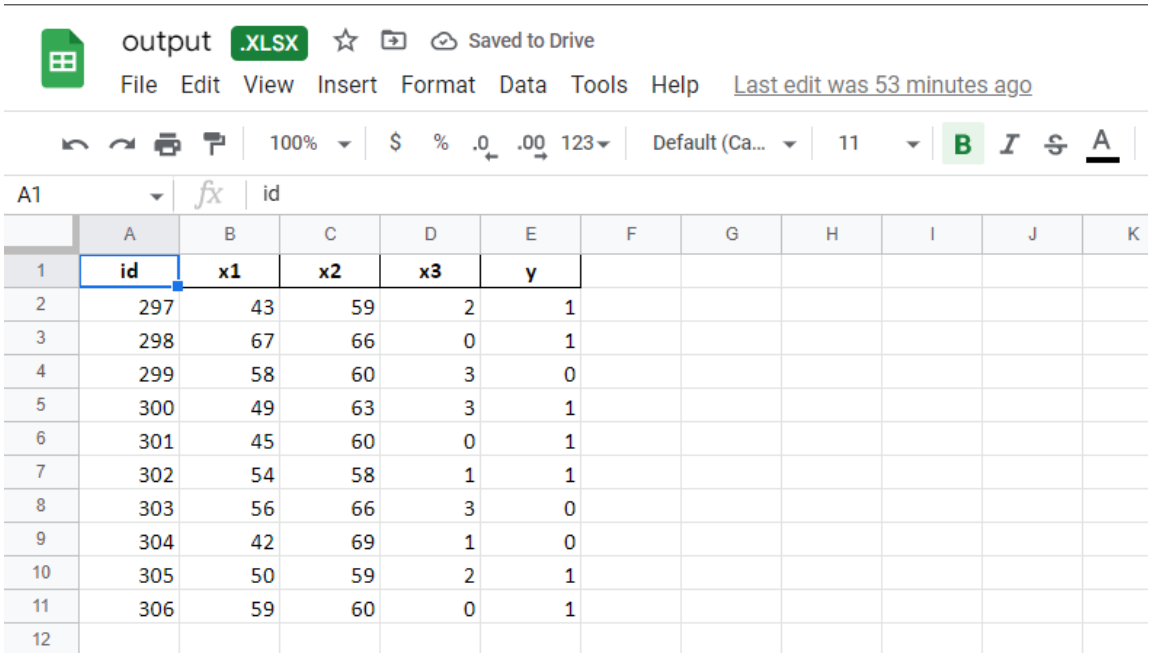
	id	x1	x2	x3	y
0	297	43	59	2	1
1	298	67	66	0	1
2	299	58	60	3	0
3	300	49	63	3	1
4	301	45	60	0	1
5	302	54	58	1	1
6	303	56	66	3	0
7	304	42	69	1	0
8	305	50	59	2	1
9	306	59	60	0	1

2.7. Menyimpan Output ke File

Menyimpan data output yang dihasilkan ke file yang telah dibuat dalam folder drive TUPRO3 dengan nama file output.xlsx.

```
[ ]  drive.mount('/drive')
     df_out.to_excel('/drive/My Drive/TUPRO3/output.xlsx', index=False)
```

```
Mounted at /drive
```



	A	B	C	D	E	F	G	H	I	J	K
1	**id**	**x1**	**x2**	**x3**	**y**						
2	297	43	59	2	1						
3	298	67	66	0	1						
4	299	58	60	3	0						
5	300	49	63	3	1						
6	301	45	60	0	1						
7	302	54	58	1	1						
8	303	56	66	3	0						
9	304	42	69	1	0						
10	305	50	59	2	1						
11	306	59	60	0	1						
12											

BAB III
KESIMPULAN

Dari hasil pengamatan kami, dapat disimpulkan bahwa hasil dari pelatihan atau training model dengan menggunakan metode KNN (*K-nearest neighbours*) dan pengujian atau *testing* model didapatkan output kelas y yaitu 1 1 0 1 1 1 0 0 1 1.

DAFTAR PUSTAKA

K-Nearest Neighbor (K-NN) - achmadrizal's blog. (2011, July 26). Achmadrizal's Blog; achmadrizal.staff.telkomuniversity.ac.id. https://achmadrizal.staff.telkomuniversity.ac.id/k-nearest-neighbor-k-nn/

Pengertian dan Cara Kerja Algoritma K-Nearest Neighbors (KNN). (2018, May 11). Advernesia; www.advernesia.com. https://www.advernesia.com/blog/data-science/pengertian-dan-cara-kerja-algoritma-k-nearest-neighbours-knn/

Afifah, L. (2020, November 23). *Algoritma K-Nearest Neighbor (KNN) untuk Klasifikasi - IlmudataPy*. IlmudataPy; ilmudatapy.com. https://ilmudatapy.com/algoritma-k-nearest-neighbor-knn-untuk-klasifikasi/