DOKUMEN PROYEK

12S4054 - PENAMBANGAN DATA

Binary Classification Using KNN for BPJS Fraud Prediction

Disusun Oleh:

12S18005 Lusiana Ros Romantika Siahaan

12S18042 Indah Oktavia M. Sibarani

12S18067 Grace Vidia Rosa Panjaitan

PROGRAM STUDI SARJANA SISTEM INFORMASI

FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO (FITE)

INSTITUT TEKNOLOGI DEL

TAHUN 2021/2022

DAFTAR ISI

DAFTAR GAMBAR

1. BUSINESS UNDERSTANDING

CRISP-DM or *Cross Industry Standard Process for Data Mining* merupakan serangkaian langkah/kerangka kerja yang jelas untuk melaksanakan proyek ilmu data/penambangan data apa pun. Penggunaan kerangka kerja ini akan memastikan kami memiliki proses yang efisien dalam mengerjakan proyek akhir *Data Mining*. *Business Understanding* adalah tahap pertama dalam CRISP-DM. Pada tahapan ini, untuk membangun model terbaik perlu untuk digali lebih dalam apa yang dibutuhkan dari proyek data mining. Untuk itu dibutuhkan pengetahuan mengenai objek bisnis, bagaimana membangun atau mendapatkan data, dan bagaimana menyesuaikan tujuan pemodelan terhadap tujuan bisnis untuk membangun pemodelan yang baik. Kegiatan yang dilakukan dalam tahapan ini antara lain: menetapkan tujuan dan persyaratan dengan jelas secara keseluruhan, menerjemahkan tujuan tersebut serta menentukan pembatasan dalam perumusan masalah data mining, dan mempersiapkan strategi awal untuk mencapai tujuan tersebut.

1.1 Determine Business Objective

Badan Penyelenggara Jaminan Sosial Kesehatan atau BPJS Kesehatan merupakan layanan kesehatan sosial yang berfungsi dalam memberikan jaminan kesehatan. Program ini merupakan program pemerintah yang berada dalam kesatuan bersama dengan JKN dan mulai beroperasi sejak 1 Januari 2014. BPJS Kesehatan memiliki wewenang terhadap seluruh wilayah Republik Indonesia dan memberikan fasilitas kesehatan pada para anggotanya.

Namun, terdapat permasalahan dimana pendapatan BPJS Kesehatan mengalami defisit. Salah satu dugaan penyebab terjadinya defisit pendapatan tersebut adalah dikarenakan beberapa peserta memalsukan status kepesertaannya, seperti adanya masyarakat yang bukan anggota BPJS Kesehatan, menggunakan layanan rumah sakit dengan memanfaatkan kartu orang lain yang merupakan anggota BPJS Kesehatan. Selain itu, terdapat juga dugaan masalah dimana terjadinya manipulasi terhadap klaim dalam pelayanan rumah sakit yang dilakukan oleh anggota BPJS Kesehatan.

Dalam mengatasi permasalahan yang dialami oleh BPJS Kesehatan, maka dilakukan prediksi terhadap *fraud* untuk klaim dalam Rumah Sakit. Analisis dilakukan dengan memanfaatkan algoritma klasifikasi *supervised learning,* yaitu K-Nearest Neighbors dan dilakukan terhadap dataset BPJS Kesehatan tahun 2021. Hasil dari pengklasifikasian ini diharapkan membantu BPJS Kesehatan dalam mengatasi permasalahan terkait kemungkinan terjadinya *fraud* pada klaim dalam rumah sakit.

1.2 Situation Assessment

Dalam pengerjaan proyek ini, dibutuhkan sumber daya yang terdiri dari *hardware*, *data sources*, dan personel. *Hardware* yang digunakan selama pengerjaan proyek berupa Laptop Lenovo Ideapad dengan 8GB RAM. Dataset yang digunakan pengerjaan proyek ini adalah data BPJS yang digunakan dalam kegiatan BPJS Hackathon dengan format file adalah CSV. Dataset train tersebut terdiri dari 200217 observasi dan 53 variable. Jumlah personil yang dibutuhkan dalam pengerjaan proyek ini adalah 3 orang mahasiswa yang terlibat dalam setiap proses dalam proyek ini, baik itu dalam proses persiapan yaitu pemilihan kasus dan algoritma, serta proses pelaksanaan yang terjadi *business understanding*, *data understanding*, *data preparation*, *modeling*, *evaluation*, dan *deployment*. Pengerjaan proyek data mining ini dilakukan selama kurang lebih 4 minggu, secara daring.

1.3 Determine Data Mining Goal

Sektor kesehatan menjadi target yang menarik bagi para *fraudsters*. Ketersediaan sejumlah besar data memungkinkan untuk mengatasi masalah ini dengan penerapan teknik *data mining*, membuat proses audit lebih efisien dan efektif. Penelitian ini bertujuan untuk mengembangkan model data mining yang ditujukan untuk membantu rumah sakit dalam melakukan pendeteksian terhadap penipuan yang terjadi di rumah sakit terkait klaim pelayanan, menggunakan algoritma KNN (*K-Nearest Neighbors*) pada data BPJS (Badan Penyelenggaraan Jaminan sosial) dan menerapkan metodologi CRISP-DM (Standar Kompetensi Kerja Nasional: KepMen Ketenagakerjaan No 299 thn 2020).

Metode atau algoritma yang akan digunakan dalam proyek ini adalah *k-nearest neighbor* (KNN) yaitu algoritma machine learning sederhana dan terawasi yang dapat digunakan untuk menyelesaikan masalah klasifikasi dan regresi. *K-nearest neighbor* (KNN) bekerja dengan mencari jarak antara query dan semua contoh dalam data, memilih contoh nomor tertentu (K) yang paling dekat dengan query, kemudian memilih label yang paling sering (dalam kasus klasifikasi) atau rata-rata label (dalam kasus regresi). Dalam kasus klasifikasi dan regresi, memilih K yang tepat untuk data dapat dilakukan dengan mencoba beberapa K dan memilih salah satu yang terbaik.

Pengerjaan proyek ini juga bertujuan memberikan tampilan hasil *classification* dalam bentuk visualisasi untuk memudahkan membaca hasil *classification.* dalam memprediksi besar fraud yang mungkin terjadi, maka dalam penelitian ini diterapkan suatu model yang akan dibangun dengan *data mining*.

1.4 Produce Project Plan

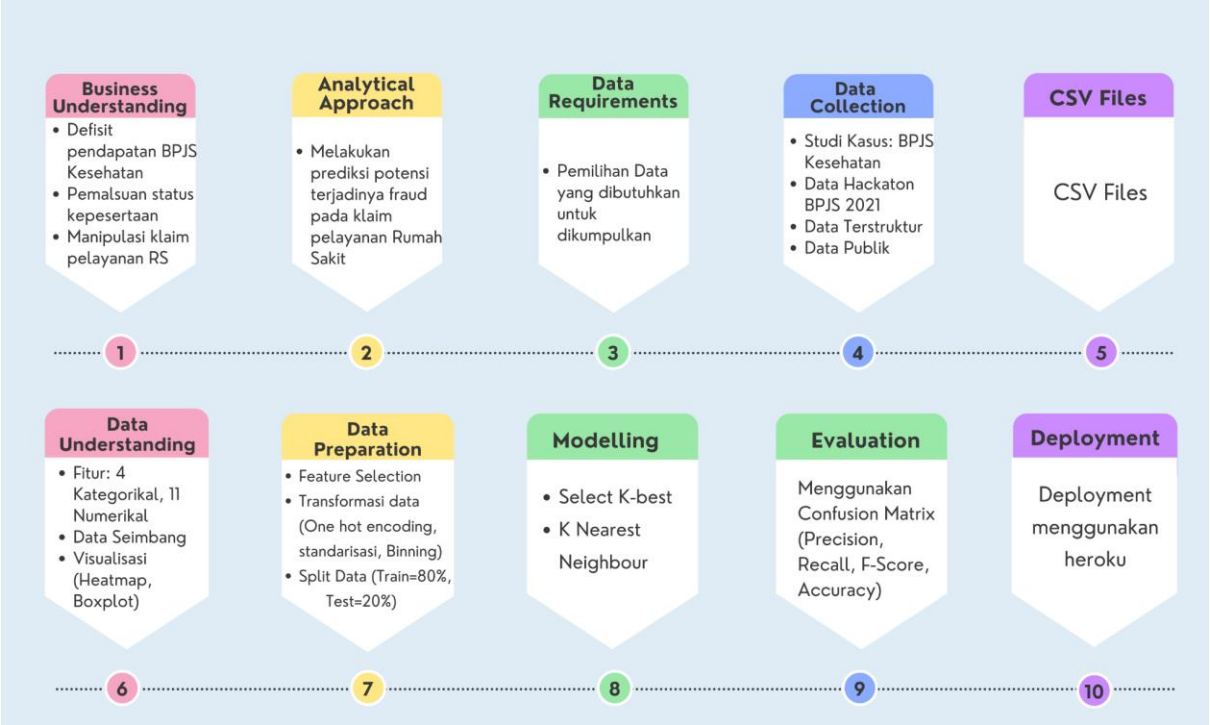
Tahap perencanaan yang dilakukan untuk mencapai tujuan data mining dan mencapai tujuan bisnis pada penelitian "*Binary Classification Using KNN for BPJS Fraud Prediction*" ini adalah sebagai berikut.

Aktivitas	Sub Aktivitas	Detail	Week																						
			12		13							14							15						
			November																			Desember			
			12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4
Persiapan	Pemilihan Kasus dan Algoritma	Pemilihan Kasus																							
		Penentuan Algoritma																							
Pelaksanaan	Business Understanding	Menentukan Objektif Bisnis																							
		Menentukan Tujuan Bisnis																							
		Membuat Rencana Proyek																							
	Data Understanding	Mengumpulkan Data																							
		Menelaah Data																							
		Memvalidasi Data																							
	Data Preparation	Memilah Data																							
		Membersihkan Data																							
		Mengkonstruksi Data																							
		Menentukan Label Data																							
		Menintegrasikan Data																							
	Modeling	Membangun Skenario Pengujian																							
		Membangun Model																							
	Model Evaluation	Mengevaluasi Hasil Pemodelan																							
		Melakukan Review Proses Pemodelan																							
	Deployment	Melakukan Deploymen Model																							
		Membuat laporan akhir Proyek																							

Gambar 1 Jadwal rencana pelaksanaan proyek

Dalam pelaksanaan proyek penelitian *data mining* ini, diperlukan *tools* dan teknik yang mendukung berbagai tahapan proses pengerjaan. *Tool* yang digunakan dalam mengerjakan proyek ini adalah Python, yaitu bahasa pemrograman berorientasi objek yang digunakan dalam pengembangan perangkat lunak maupun dalam analisis dan data science.

Python memiliki berbagai *library* yang menyediakan fungsi untuk melakukan analisis data, memproses data, memvisualisasikan data, dll. *Library* yang disediakan diantaranya scikit-learn, Keras, dan TensorFlow untuk membantu dalam pembuatan model data mining dengan cepat. Selain itu, terdapat juga library yang dapat digunakan untuk membagi dataset menjadi data training dan data test, misalnya menggunakan *cross-validation*.



Gambar 2 Tahapan proses data mining

2. DATA UNDERSTANDING

Data Understanding adalah tahapan selanjutnya yang dilakukan setelah *business understanding.* Pada tahapan ini dilakukan pengumpulan data, eksplorasi data, dan validasi data guna memahami kondisi dari dataset.

2.1 Collect Initial Data

Pengumpulan data adalah tahapm an pertama yang dilakukan pada kegiatan *data understanding*. Dataset yang digunakan dan dikumpulkan pada proyek ini adalah dataset BPJS Kesehatan dalam kegiatan BPJS Hackathon.

2.2. Describe Data

Dataset yang digunakan dan dikumpulkan pada proyek ini adalah dataset BPJS Kesehatan yang digunakan dalam kegiatan BPJS Hackathon. Format dari data yang akan digunakan adalah dalam bentuk .CSV. Untuk membaca dataset yang akan digunakan, terlebih dahulu import library pandas untuk membaca data.

```
import pandas as pd
import numpy as np

file = ("fraud_detection_train.csv")
df = pd.read_csv(file)
```

Gambar 3 Kode untuk membaca dataset

Dataset yang digunakan terdiri dari 200217 *record* dengan 53 kolom. Untuk melihat dimensi dataset BPJS Kesehatan, yaitu mendapatkan jumlah baris dan kolom digunakan fungsi `df.shape`. Fungsi `df.columns` pada pandas digunakan untuk melihat kolom yang ada pada dataset.

```
df.shape
```

```
(200217, 53)
```

```
print(df.columns)
```

```
Index(['visit_id', 'kdkc', 'dati2', 'typeppk', 'jkpst', 'umur', 'jnspelsep',
       'los', 'cmg', 'severitylevel', 'diagprimer', 'dx2_a00_b99',
       'dx2_c00_d48', 'dx2_d50_d89', 'dx2_e00_e90', 'dx2_f00_f99',
       'dx2_g00_g99', 'dx2_h00_h59', 'dx2_h60_h95', 'dx2_i00_i99',
       'dx2_j00_j99', 'dx2_koo_k93', 'dx2_l00_l99', 'dx2_m00_m99',
       'dx2_n00_n99', 'dx2_o00_o99', 'dx2_p00_p96', 'dx2_q00_q99',
       'dx2_r00_r99', 'dx2_s00_t98', 'dx2_u00_u99', 'dx2_v01_y98',
       'dx2_z00_z99', 'proc00_13', 'proc14_23', 'proc24_27', 'proc28_28',
       'proc29_31', 'proc_32_38', 'proc39_45', 'proc46_51', 'proc52_57',
       'proc58_62', 'proc63_67', 'proc68_70', 'proc71_73', 'proc74_75',
       'proc76_77', 'proc78_79', 'proc80_99', 'proce00_e99', 'procv00_v89',
       'label'],
      dtype='object')
```

Gambar 4 Kode untuk melihat kolom pada dataset

Pada hasil yang diberikan dapat dilihat nama dari 53 atribut yang ada pada dataset. Selanjutnya, untuk melihat detail statistik seperti persentil, rata-rata, standar deviasi, dan lain-lain dari atribut dalam dataset, digunakan fungsi `df.describe()`.

```
#descriptive data
df.describe()
```

	visit_id	kdkc	dati2	umur	jnspelsep	los	severitylevel	dx2_a00_b99	dx2_c00_d48	dx2_d50_d89	dx2_e00
count	200217.000000	200217.000000	200217.000000	200217.000000	200217.000000	200217.000000	200217.000000	200217.000000	200217.000000	200217.000000	200217.00
mean	100109.000000	1147.367816	184.793309	36.850602	1.669778	1.303356	0.444003	0.024893	0.008341	0.020703	0.04
std	57797.813761	574.486224	107.226676	23.095928	0.470294	5.639751	0.725227	0.162484	0.093386	0.146842	0.24
min	1.000000	101.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	50055.000000	903.000000	114.000000	18.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	100109.000000	1101.000000	169.000000	39.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
75%	150163.000000	1314.000000	232.000000	56.000000	2.000000	2.000000	1.000000	0.000000	0.000000	0.000000	0.00
max	200217.000000	2606.000000	528.000000	109.000000	2.000000	592.000000	3.000000	4.000000	3.000000	3.000000	7.00

Gambar 5 Kode untuk detail statistik dataset

2.3 Explore Data

Pada bagian ini, dataset ditelaah untuk memperoleh informasi terkait kondisi dari dataset. Penelaahan terhadap data dimulai dengan memperlihatkan informasi terkait kolom yang terdapat di dalam dataset beserta dengan data descriptionnya. Perintah ini dilakukan dengan menjalankan fungsi `df.info()`.

```
df.info()
```

```
Data columns (total 53 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   visit_id        200217 non-null  int64
 1   kdkc            200217 non-null  int64
 2   dati2           200217 non-null  int64
 3   typeppk         200217 non-null  object
 4   jkpst           200217 non-null  object
 5   umur            200217 non-null  int64
 6   jnspelsep       200217 non-null  int64
 7   los             200217 non-null  int64
 8   cmg             200217 non-null  object
 9   severitylevel   200217 non-null  int64
 10  diagprimer      200217 non-null  object
 11  dx2_a00_b99     200217 non-null  int64
 12  dx2_c00_d48     200217 non-null  int64
 13  dx2_d50_d89     200217 non-null  int64
 14  dx2_e00_e90     200217 non-null  int64
 15  dx2_f00_f99     200217 non-null  int64
```

Gambar 6 Kode untuk melihat informasi kolom

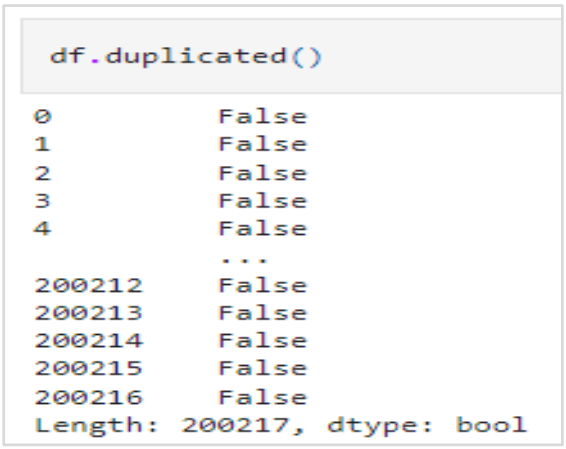
Untuk keterangan pada setiap atribut dapat dilihat pada table berikut ini.

Table 1. Keterangan Atribut

No	Atribute	Keterangan	Tipe	Nilai
1	Visit_id	id kunjungan	Int64	Id numerik
2	kdkc	kode wilayah kantor cabang BPJS Kesehatan	Int64	Kode kdkc numerik
3	Dati2	kode kabupaten/kota	Int64	Kode dati2 numerik
4	Typeppk	kode tipe Rumah Sakit	Object	SC, C, B, SD, A, D, I3, KM, KI, I2, K4, KJ, KL, I1, KB, KC, GD, SA, KP, KO, KG, HD, KT, KU
5	Jkpst	jenis kelamin peserta JKN-KIS	Object	P dan L, dimana P (Perempuan) dan L (Laki-Laki)
6	Umur	umur peserta saat mendapatkan pelayanan rumah sakit	Int64	0-109
7	Jnspelsep	tingkat pelayanan; 1:rawat inap; 2. rawat jalan	Int64	1 dan 2, dimana 1 adalah rawat inap dan 2 adalah rawat jalan
8	Los	lama peserta dirawat di rumah sakit	Int64	‘F’, ‘E’, ‘Q’, ‘L’, ‘H’, ‘W’, ‘P’, ‘U’, ‘K’, ‘G’, ‘M’, ‘N’, ‘A’, ‘C’, ‘D’, ‘Z’, ‘J’, ‘O’, ‘S’, ‘T’, ‘V’, ‘T’, ‘B’

9	Cmg	klasifikasi CMG (Case Mix Group)	Object	0, 1, 2, 3
10	Severitylevel	tingkat urgensi	Int64	
11	Diagprimer	diagnosa primer	Object	‘a00_b99’, ‘c00_d48’,’ d50_d89’, ‘e00_e90’, ‘f00_f99’, ‘g00_g99’, ‘h00_h59’, ‘h60_h95’, ‘i00_i99’, ‘j00_j99’, ‘koo_k93’, ‘l00_l99’, ‘m00_m99’, ‘n00_n99’, ‘o00_o99 ‘, ‘p00_p96`, ‘q00_q99’, ‘r00_r99’, ‘s00_t98’, ‘u00_u99’, ‘v01_y98’, ‘z00_z99’.
12	dx2_..._...	diagnosa sekunder	Int64	Terdiri dari 22 atribut dengan nilai berada pada rentang 0-13
13	proc_..._...	kode kelompok procedure	Int64	Terdiri dari 19 atribut dengan nilai berada pada rentang 0-13
14	label	flag fraud; 1:fraud; 0:tidak fraud	Int64	1 dan 0 dimana 1 adalah fraud, dan 0 adalah tidak fraud.

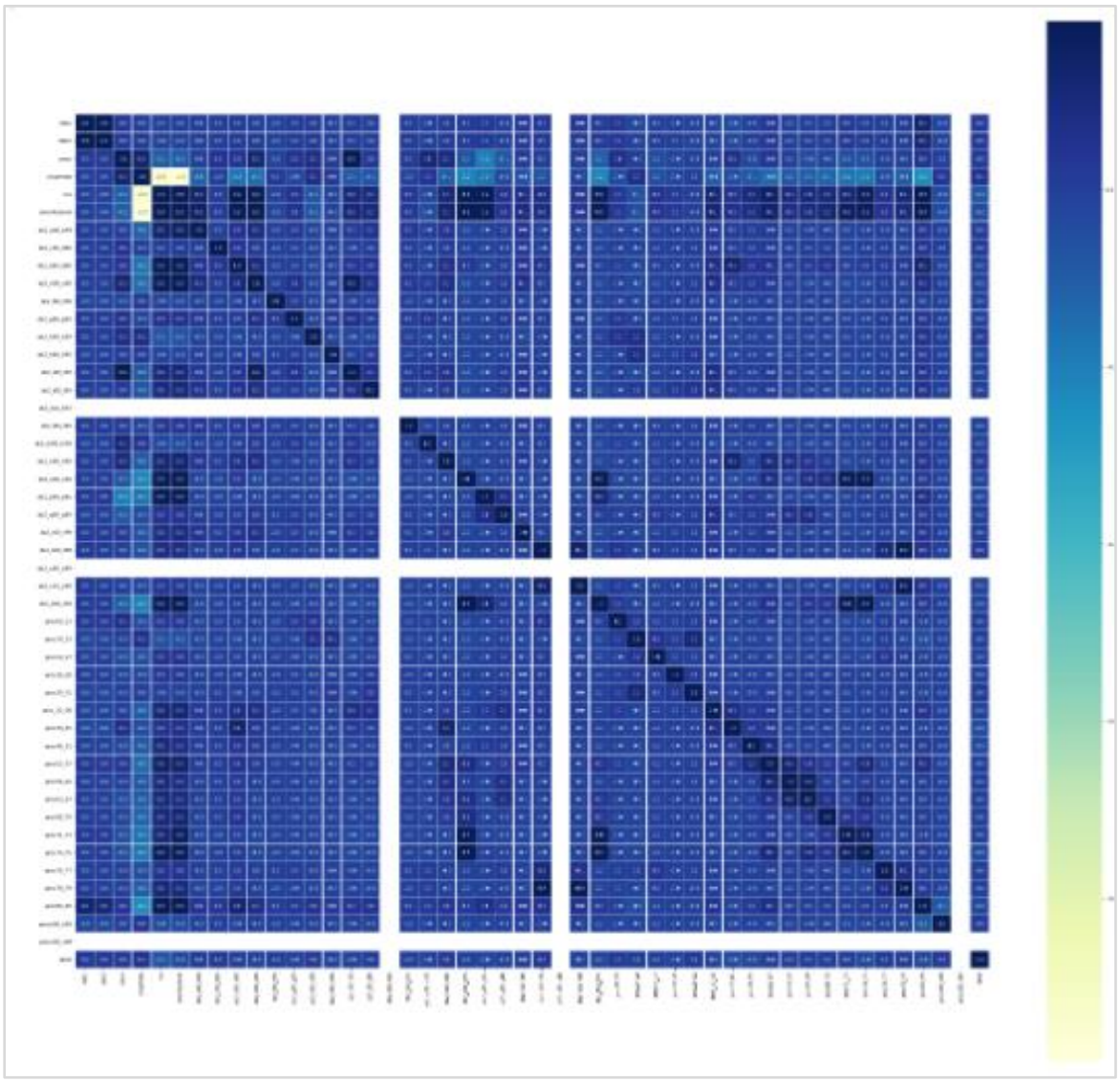
Setelah itu, dataset dicek kembali apakah terdapat duplikasi data, menggunakan fungsi `df.duplicate()`, dapat dilihat dari data tersebut bahwa tidak ada data yang duplikasi.



Gambar 7 Kode untuk melihat data duplikat

Korelasi antar atribut juga perlu untuk ditinjau. Oleh karena itu, korelasi antar masing-masing atribut divisualisasikan menggunakan visualisasi *heatmap*. Ukuran dari visualisasi diatur menggunakan *library* matplotlib, sementara untuk menampilkan heatmap menggunakan library

seaborn. Melalui korelasi ini diperlihatkan bahwa setiap atribut memiliki korelasi sebesar 0. Semakin terang hasil pemetaan dari heatmap, semakin rendah korelasi antara atribut tersebut.

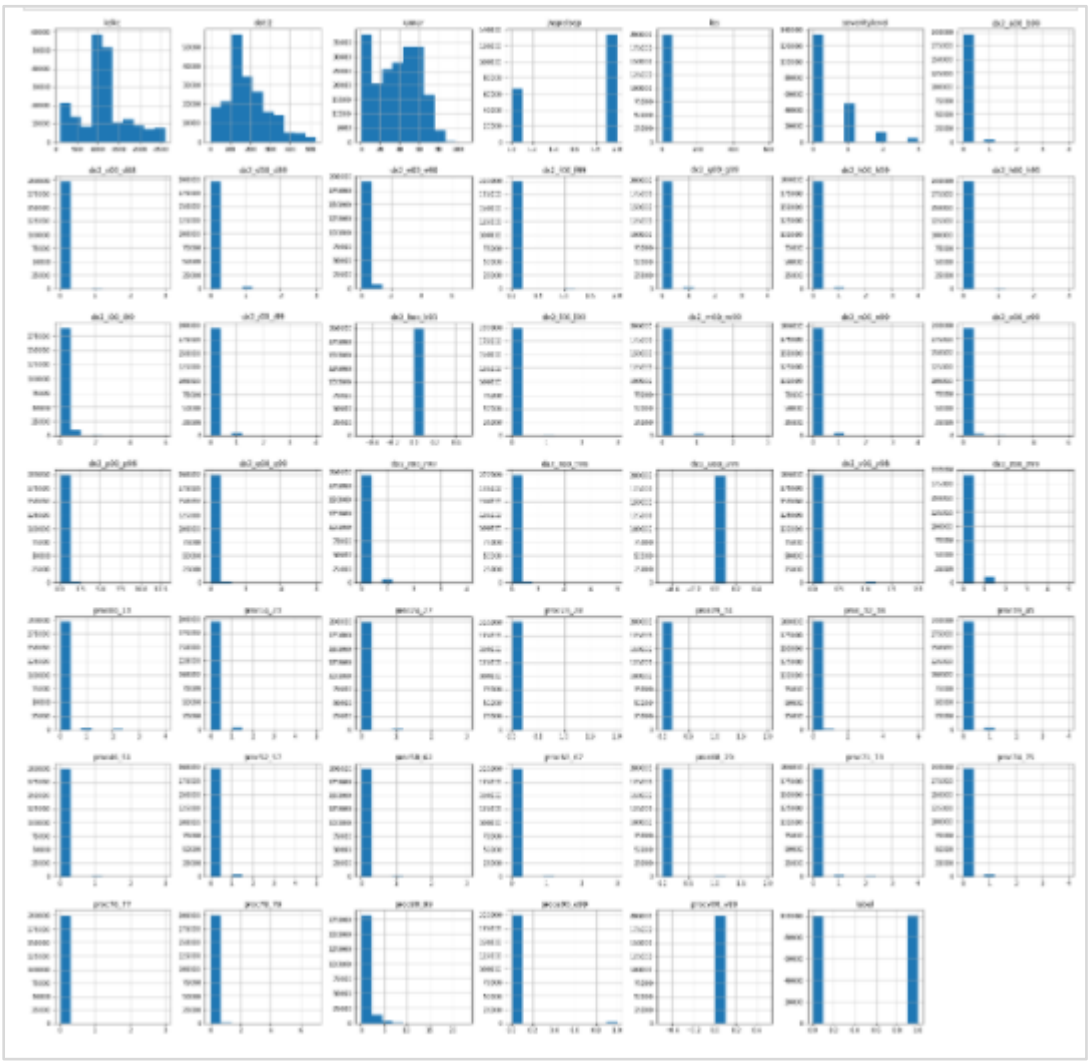


Gambar 8 Visualisasi korelasi antar atribut

Setelah berhasil meninjau korelasi antar atribut, selanjutnya dilakukan penelaahan terhadap *value* dari masing-masing atribut. Dari tampilan visualisasi histogram pada Gambar 9, yaitu tampilan *value* setiap atribut dalam dataset, maka didapatkan hal-hal berikut ini:

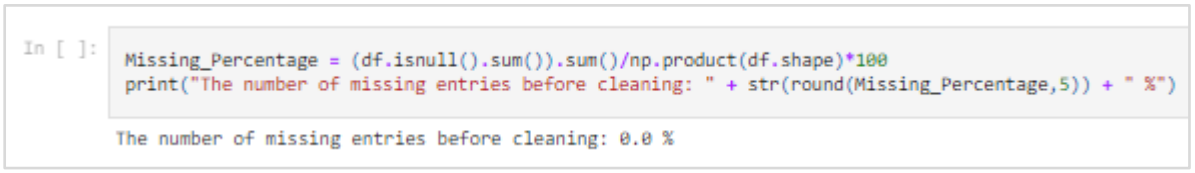
1. Atribut yang memiliki variasi *value* terbanyak adalah atribut kdkc, dati2, dan umur.
2. Atribut kdkc dengan *value* 1000 memiliki frekuensi tertinggi dan *value* 2250 memiliki frekuensi terendah.

3. Atribut kdkc menunjukkan kode wilayah kantor cabang BPJS Kesehatan, yang menunjukkan bahwa kode kdkc dengan value sekitar 1000 memiliki jumlah pasien terbanyak.
4. Atribut dati2 dengan *value* 100 memiliki frekuensi tertinggi dan *value* 500 terendah yang menunjukkan bahwa kabupaten dengan kode 100 memiliki jumlah pasien tertinggi.
5. Untuk umur pasien dengan jumlah terbanyak adalah pasien dengan umur sekitar 0 bulan dan umum pasien dengan jumlah terkecil adalah umur 80.



Gambar 9 Value masing-masing atribut

Gambar 9 memperlihatkan visualisasi dari keseimbangan data yang ada di masing-masing fitur. Seperti yang dapat dilihat, data di atas dianggap seimbang pada fitur label. Label adalah fitur acuan dalam melatih dan menguji data terkait data tersebut fraud atau tidak fraud.



Gambar 10 Kode untuk melihat *missing entries*

2.4 Verify Data Quality

Data validation dilakukan dengan tujuan untuk memastikan bahwa pemodelan terjadi pada data yang benar. Data yang salah yang digunakan sebagai data pelatihan untuk model akan menghasilkan pengetahuan yang salah. Validasi data dilakukan segera setelah persiapan data, dan sebelum pemodelan data. Itu karena selama persiapan data ada kemungkinan besar terjadi kesalahan terutama dalam skenario yang kompleks. Validasi data harus dilakukan dengan melibatkan minimal satu orang eksternal yang memiliki pemahaman yang tepat tentang data dan bisnis. Dataset yang digunakan terdiri atas data kuantitatif, yaitu data yang dapat diukur (measurable) atau dapat dihitung sebagai angka atau bilangan. Data tersebut dapat berupa bilangan diskrit atau bilangan kontinu. Data kuantitatif memiliki kecenderungan dapat dianalisis dengan teknik statistik. Data yang termasuk kuantitatif pada dataset adalah Quantity (QTY) dan Value.

3. DATA PREPARATION

Setelah data diperiksa dan dikarakterisasi selama tahap *data understanding*, data tersebut kemudian disiapkan untuk tahapan *data mining* berikutnya yaitu *data preparation*, tahap selanjutnya pada fase CRISP-DM. Tahap *data preparation* merupakan tahapan untuk menyiapkan data awal, memilih variabel yang akan dianalisis dan membersihkan data. Dalam pengerjaan proyek, bahasa pemrograman yang digunakan adalah pemrograman *Python* dengan *software* pengolah data *Jupyter Notebook*.

3.1 Data Cleaning

Pada tahapan Data Cleaning, akan dilakukan pengecekan adanya nilai null pada dataset dan mendeteksi adanya outlier di dalam dataset. Tahapan data cleaning merupakan bagian dari Exploratory Data Analysis untuk menghasilkan dataset yang tidak mengandung missing value.

3.1.1 Check Null Value

Data yang baru saja dikumpulkan kemungkinan besar memiliki banyak bagian yang tidak relevan bahkan ada bagian yang hilang. Proses cleaning yang dilakukan adalah dengan memeriksa adanya data yang memiliki nilai null. *Data cleaning* diperlukan untuk menjaga kekonsistenan dan menghilangkan data tidak relevan. Pada proses data mining, *data cleaning* dapat mengurangi jumlah dan kompleksitas data. *Data Cleaning* termasuk kedalam bagian dari *Exploratory Data Analysis*, *dimana* akan dihasilkan data yang tidak mengandung *missing value* pada dataset tersebut. Setelah data cleaning dilakukan, pada dataset yang digunakan ditemukan bahwa data tidak mengandung *missing value*. Pada dataset yang digunakan tidak terdapat nilai null *value* ataupun *missing values*.

```
Missing_Percentage = (df.isnull().sum()).sum()/np.product(df.shape)*100
print("The number of missing entries before cleaning: " + str(round(Missing_Percentage,5)) + " %")
```

```
The number of missing entries before cleaning: 0.0 %
```

Gambar 11 Kode untuk melihat *missing entries*

3.1.2. Check Outlier

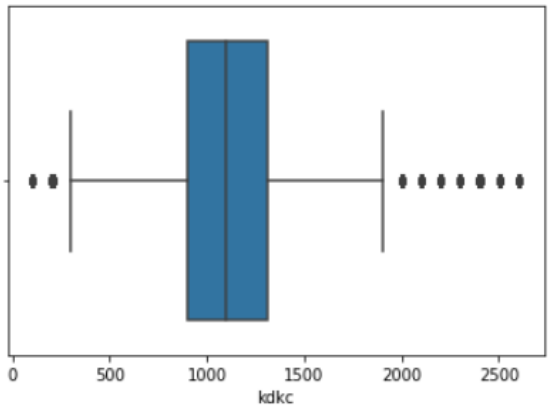
Outlier adalah pengamatan yang sangat menyimpang dari pengamatan lain dalam sampel. Pengecek *outlier* dilakukan dikarenakan *outlier* mungkin menunjukkan data yang buruk. Misalnya, data mungkin salah dikodekan atau eksperimen mungkin tidak dijalankan dengan benar. Jika dapat ditentukan bahwa titik *outlying* sebenarnya salah, maka nilai *outlying* harus

dihapus dari analisis (atau dikoreksi jika memungkinkan). Dari dataset yang digunakan terdapat *outlier* pada *features* kdkc, severitylevel, los, dan proc80 .

1. Berikut merupakan visualisasi *outlier* pada *feature* kdkc

```
import seaborn as sns

sns.boxplot(x=df['kdkc'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af8b6b790>
```



Gambar 12 Visualisasi *outlier* pada *feature* kdkc

Dan rentang dari *outlier* yang terdapat pada *feature* kdkc adalah sebagai berikut:

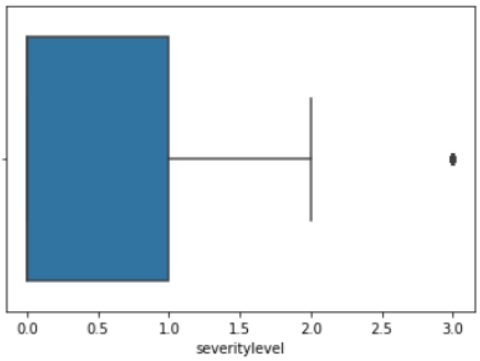
```
In [ ]:  print(np.where(df['kdkc']>2000))

(array([     7,     10,     45, ..., 200190, 200204, 200212]),)
```

Gambar 13 Rentang *outlier* yang terdapat pada *feature* kdkc

2. Berikut merupakan visualisasi *outlier* pada *feature* severitylevel.

```
sns.boxplot(x=df['severitylevel'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af87d23d0>
```



Gambar 14 Visualisasi *outlier* pada *feature* severitylevel

Dan rentang dari *outlier* yang terdapat pada *feature* severitylevel adalah sebagai berikut:

```
print(np.where(df['severitylevel']>1))
```

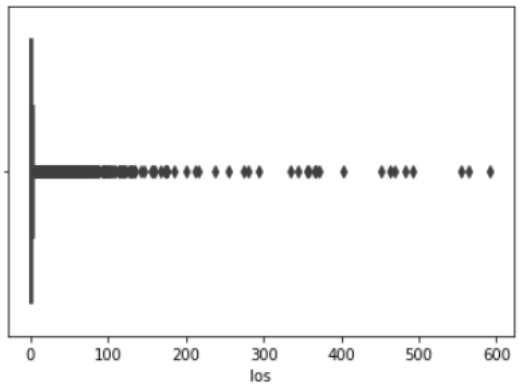
```
(array([     1,      9,     37, ..., 200177, 200197, 200205]),)
```

Gambar 15 Rentang *outlier* yang terdapat pada *feature* severitylevel

3. Berikut merupakan visualisasi *outlier* pada *feature* los.

```
sns.boxplot(x=df['los'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af87fca10>
```



Gambar 16 Visualisasi *outlier* pada *feature* los

Dan rentang dari *outlier* yang terdapat pada *feature* los adalah sebagai berikut:

```
print(np.where(df['los']>0))
```

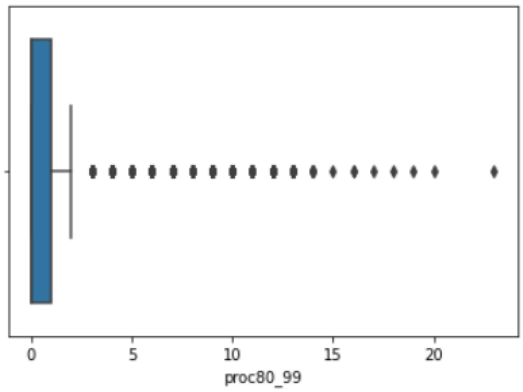
```
(array([     1,      6,      9, ..., 200210, 200215, 200216]),)
```

Gambar 17 Rentang *outlier* yang terdapat pada *feature* los

4. Berikut merupakan visualisasi *outlier* pada proc80_99.

```
sns.boxplot(x=df['proc80_99'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af773a2d0>
```



Gambar 18 Visualisasi *outlier* pada *feature* proc80_99

Dan rentang dari *outlier* yang terdapat pada proc80_99 adalah sebagai berikut:

```
print(np.where(df['los']>1))
```

```
(array([     1,      6,      9, ..., 200198, 200205, 200210]),)
```

Gambar 19 Rentang *outlier* yang terdapat pada *feature* proc80_99

Quantile mendefinisikan bagian tertentu dari kumpulan data, yaitu *quantile* menentukan berapa banyak nilai dalam distribusi yang berada di batas atas atau batas bawah. Untuk mengatasi *outlier* tersebut maka pertama sekali menghitung *quantile* pada setiap *features*.

```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
visit_id         100108.0
kdkc                411.0
umur                 38.0
jnspelsep             1.0
los                   2.0
severitylevel         1.0
dx2_a00_b99           0.0
dx2_c00_d48           0.0
dx2_d50_d89           0.0
dx2_e00_e90           0.0
dx2_f00_f99           0.0
dx2_g00_g99           0.0
dx2_h00_h59           0.0
dx2_h60_h95           0.0
dx2_i00_i99           0.0
dx2_j00_j99           0.0
dx2_l00_l99           0.0
dx2_m00_m99           0.0
dx2_n00_n99           0.0
dx2_o00_o99           0.0
```

Gambar 20 Perhitungan *quantile* pada *features*

Kemudian melakukan penghapusan terhadap *outlier* pada *features* yang telah dilakukan pengecekan *outlier*. Dengan melakukan quantile atau menghitung nilai batas bawah dan batas atas pada fitur yang outliernya ingin dihapus misalnya untuk kdkc.

```
print(df['kdkc'].quantile(0.10))
print(df['kdkc'].quantile(0.90))
```

```
303.0
2101.0
```

Gambar 21 Perhitungan nilai batas bawah dan batas atas *feature* kdkc

Fungsi `where()` bertujuan untuk mengembalikan indeks elemen dalam array input dimana kondisi yang diberikan terpenuhi, misalnya untuk fitur kdkc dengan batas bawah lebih kecil dari 303.0 dan batas atas lebih besar dari 1750.0.

```
df["kdkc"] = np.where(df["kdkc"] <303.0, 303.0,df['kdkc'])
df["kdkc"] = np.where(df["kdkc"] >1750.0, 1750.0,df['kdkc'])
print(df['kdkc'].skew())
```

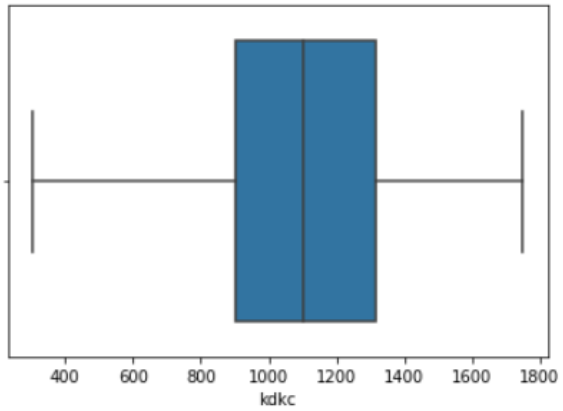
```
-0.1903395687739066
```

Gambar 22 Indeks elemen dalam array input

Berikut merupakan tampilan *chart* boxplot dengan *outlier* yang sudah dihapus.

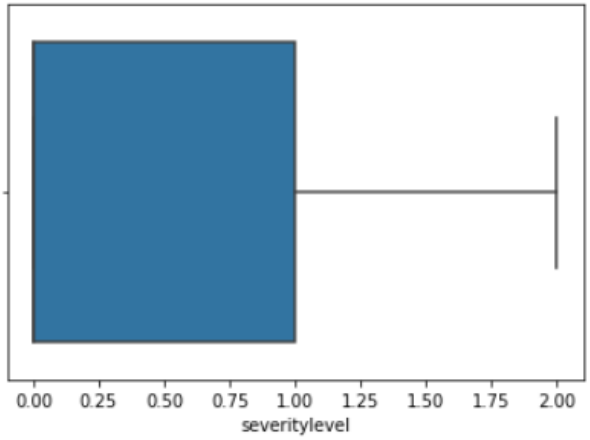
```
sns.boxplot(x=df['kdkc'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af76b1f90>
```

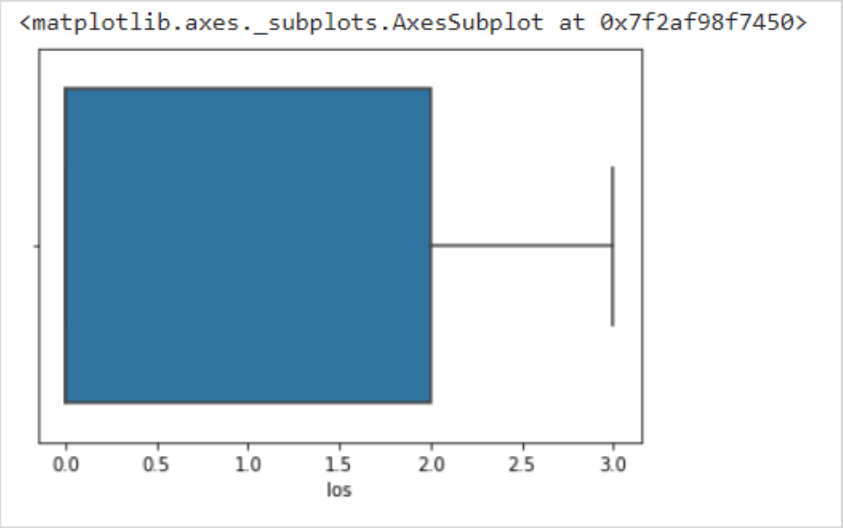


Gambar 23 *Chart boxplot* kdkc dengan *outlier* yang telah dihapus

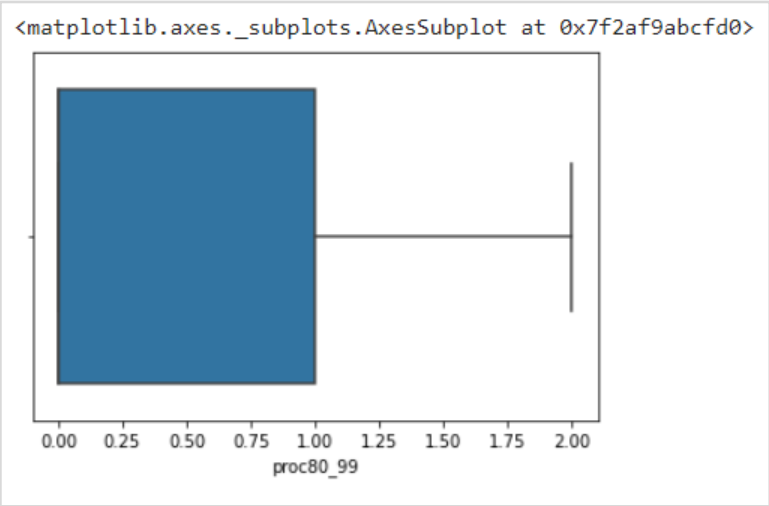
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2af757e290>
```



Gambar 24 *Chart boxplot* severitylevel dengan *outlier* yang telah dihapus



Gambar 25 *Chart boxplot* los dengan *outlier* yang telah dihapus



Gambar 26 *Chart boxplot* proc80_99 dengan *outlier* yang telah dihapus

3.2 Feature Selection

Feature selection dikenal juga sebagai seleksi Variabel atau seleksi Atribut. Merupakan memilih fitur-fitur secara otomatis yang terdapat dalam data yang digunakan yang berkontribusi paling besar pada variabel prediksi. *Feature selection* ini digunakan ketika berhadapan dengan dataset berdimensi tinggi. *Feature selection* penting dilakukan karena sebagai berikut:

- Memungkinkan algoritma pembelajaran mesin untuk berlatih lebih cepat.
- Mengurangi kompleksitas model dan membuatnya lebih mudah untuk diinterpretasikan.

- Meningkatkan akurasi model jika subset yang tepat dipilih.
- Mengurangi *overfitting*.

Kode dibawah ini merupakan kode yang digunakan untuk memilih *feature* yang akan dieliminasi, dimana suatu feature akan di seleksi jika nilai unique pada *feature* tersebut hanya 1.

```
def print_str_unique(df):
    for col in df.columns:
            unique_cnt = len(df[col].unique())
            print(f'Column {col.rjust(13)} has {unique_cnt:5} unique values.')
            if unique_cnt == 1:
              print(f'{col}-->> need to be dropped.')
print_str_unique(df)
```

Gambar 27 Kode untuk memilih *feature* yang akan dieliminasi

Berikut merupakan *output* dari kode diatas. Berdasarkan *output* tersebut dapat dilihat terdapat 3 *features* yang akan dieliminasi dikarenkan jumlah *unique values*nya = 1 yaitu dx2_u00_u99, dx2_koo_k93, dan procv00_v89. Fitur tersebut dieliminasi guna untuk mengurangi fitur yang tidak memiliki pengaruh dalam pembangunan model.

```
Column    dx2_koo_k93 has     1 unique values.
dx2_koo_k93-->> need to be dropped.
Column    dx2_l00_l99 has     4 unique values.
Column    dx2_m00_m99 has     4 unique values.
Column    dx2_n00_n99 has     5 unique values.
Column    dx2_o00_o99 has     7 unique values.
Column    dx2_p00_p96 has    14 unique values.
Column    dx2_q00_q99 has     7 unique values.
Column    dx2_r00_r99 has     5 unique values.
Column    dx2_s00_t98 has     8 unique values.
Column    dx2_u00_u99 has     1 unique values.
dx2_u00_u99-->> need to be dropped.
Column    dx2_v01_y98 has     3 unique values.
```

Gambar 28 *Output* dari kode eliminasi

```
Column     proc76_77 has     4 unique values.
Column     proc78_79 has     7 unique values.
Column     proc80_99 has    22 unique values.
Column   proce00_e99 has     2 unique values.
Column   procv00_v89 has     1 unique values.
procv00_v89-->> need to be dropped.
Column         label has     2 unique values.
```

Gambar 29 *Output* dari kode eliminasi *cont'd*

Berdasarkan hasil korelasi antar *features*, *feature* yang memiliki korelasi tertinggi akan dieliminasi yaitu 0.8. Eliminasi dilakukan dengan cara drop column x2_u00_u99, dx2_koo_k93, procv00_v89, dati2, visit_id

```
df = df.drop(df.columns[[20,29,50]], axis = 1)
```

Gambar 30 Eleminasi *feature*

```
df = df.drop(['dati2', 'visit_id'], axis=1)
```

Gambar 31 Eleminasi *feature cont'd*

Setelah dilakukan proses eliminasi, maka feature yang terdapat dalam dataset saat ini adalah sebagai berikut.

```
df.columns
```

```
Index(['kdkc', 'typeppk', 'jkpst', 'umur', 'jnspelsep', 'los', 'cmg',
       'severitylevel', 'diagprimer', 'dx2_a00_b99', 'dx2_c00_d48',
       'dx2_d50_d89', 'dx2_e00_e90', 'dx2_f00_f99', 'dx2_g00_g99',
       'dx2_h00_h59', 'dx2_h60_h95', 'dx2_i00_i99', 'dx2_j00_j99',
       'dx2_l00_l99', 'dx2_m00_m99', 'dx2_n00_n99', 'dx2_o00_o99',
       'dx2_p00_p96', 'dx2_q00_q99', 'dx2_r00_r99', 'dx2_s00_t98',
       'dx2_v01_y98', 'dx2_z00_z99', 'proc00_13', 'proc14_23', 'proc24_27',
       'proc28_28', 'proc29_31', 'proc_32_38', 'proc39_45', 'proc46_51',
       'proc52_57', 'proc58_62', 'proc63_67', 'proc68_70', 'proc71_73',
       'proc74_75', 'proc76_77', 'proc78_79', 'proc80_99', 'proce00_e99',
       'label'],
      dtype='object')
```

Gambar 32 *Features dataset* setelah eliminasi

3.2 Data Transformation

Pada *data transformation*, data diubah dari satu format ke format lainnya. Proses transformasi data melibatkan pengumpulan data mentah dan mengubahnya menjadi data yang bersih dan dapat digunakan. Transformasi data meningkatkan efisiensi proses bisnis dan analitik, dan memungkinkan bisnis membuat keputusan berdasarkan data yang lebih baik. Dalam tahap *data transformation* melibatkan *one hot encoding*, *standarisasi*, dan *binning*).

3.2.1 One Hot Encoding

Teknik One-Hot Encoding merupakan teknik mewakili yang variabel/fitur dengan jenis kategori sebagai sekelompok nilai biner, di mana setiap nilai biner ini mewakili satu kategori . Variabel biner menunjukkan apakah kategori tersebut ada dalam suatu pengamatan (1) atau tidak (0). Pada gambar dibawah ini dapat kita lihat terdapat 3 fitur dengan data type object, sehingga harus dilakukan transformasi.

```
 1   kdkc           200217 non-null  int64
 2   dati2          200217 non-null  int64
 3   typeppk        200217 non-null  object
 4   jkpst          200217 non-null  object
 5   umur           200217 non-null  int64
 6   jnspelsep      200217 non-null  int64
 7   los            200217 non-null  int64
 8   cmg            200217 non-null  object
 9   severitylevel  200217 non-null  int64
 10  diagprimer     200217 non-null  object
```

Gambar 33 Features data type

Maka dari itu berdasarkan jenis data type tersebut, akan dilakukan proses transformasi terhadap fitur yang bertype data object yaitu typeppk, jkpst, cmg, diagprimer yang ditampung pada variabel vars_categorical. *One hot encoding* ini merupakan transformasi terhadap tipe data berkategori numerik. Teknik encoding ini penting untuk data type *categorical data*, dikarenakan dalam machine learning tidak menerima inputan dalam bentuk *string* melainkan bentuk data numerik.

```
vars_categorical = ['typeppk', 'jkpst', 'cmg', 'diagprimer']
data_enc = pd.get_dummies(df[vars_categorical], drop_first=True)
encoder = OneHotEncoder(categories='auto', drop='first', sparse=False)
encoder.fit(df[vars_categorical])
data_enc = encoder.transform(df[vars_categorical])

from feature_engine.categorical_encoders import OneHotCategoricalEncoder
one_enc = OneHotCategoricalEncoder(top_categories=None, drop_last=True)
one_enc.fit(df)
```

Gambar 34 Transformasi pada *features datatype object*

Berdasarkan hasil output dibawah ini dapat dilihat bahwa seperti feature cmg yang telah diubah kedalam bentuk numerik, maka cmg tersebut akan terbagi-bagi menjadi beberapa bagian dengan nilai 0 dan 1.

```
data_enc.head()
```

7	proc68_70	...	cmg_Q	cmg_L	cmg_H	cmg_W	cmg_P	cmg_U	cmg_K	cmg_G	cmg_M	cmg_N	cmg_A	cmg_C	cmg_D	cmg_Z	cmg_J	cmg_O	cmg_S	cmg_I	cmg_V	cmg_T
0	0	...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	...	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	...	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Gambar 35 Bentuk numerik *feature* cmg

3.2.2 Standarisasi

Scaling standardisasi berfokus pada mengubah data mentah menjadi informasi yang dapat digunakan sebelum dianalisis. Standarisasi merupakan teknik yang menskalakan distribusi setiap atribut untuk memiliki rata-rata nol (mean = 0) dan standar deviasi satu (satuan varians). Dalam melakukan standarisasi menggunakan fungsi standardscaler(). Melalui hasil output seperti dibawah ini dapat kita lihat rata nilai berada di antara rentang -1 sampaikan 1.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(data_enc)
train_scaled = scaler.transform(data_enc)
```

```
train_scaled
```

```
array([[-1.73204216,  0.02347536,  1.17550878, ..., -0.22975506,
        -0.22893468, -0.22543364],
       [-1.73202485,  0.46477801,  0.35285088, ..., -0.22975506,
        -0.22893468, -0.22543364],
       [-1.73200755,  0.03923617, -0.12342476, ..., -0.22975506,
        -0.22893468, -0.22543364],
       ...,
       [ 1.73200755, -1.78676611, -1.46565608, ..., -0.22975506,
        -0.22893468, -0.22543364],
       [ 1.73202485, -0.19942751,  0.65593537, ..., -0.22975506,
        -0.22893468, -0.22543364],
       [ 1.73204216, -0.18141516, -1.42235829, ..., -0.22975506,
        -0.22893468, -0.22543364]])
```

Gambar 36 Proses standarisasi

3.2.3 Binning

Binning merupakan proses dalam melakukan pengelompokan data dalam bin (atau bucket), yang artinya menggantikan nilai yang terdapat dalam interval kecil dengan nilai perwakilan tunggal untuk interval tersebut. *Binning* ini digunakan untuk melakukan transformasi terhadap variabel numerik ke dalam bentuk kategoris. *Binning* variabel numerik cenderung meningkatkan kinerja model dan dapat digunakan untuk melakukan identifikasi terhadap nilai-nilai yang hilang atau *outlier*. Untuk melakukan binning menggunakan fungsi cut() untuk melakukan transformasi. Dalam proses ini dilakukan proses binning pada dua atribut yaitu:

1. Umur

   Pada atribut umur akan diubah kedalam beberapa kategori dengan interval tertentu yaitu 1: umur <=1, 2: 2<= umur >=10, 3: 11<= umur >=19, 4: 20<= umur >=60, dan 5: umur > 60. Bins merupakan interval antar label dimana nilai terendah adalah 0 sehingga nilai terendah dalam bins adalah -1 dan tertinggi adalah 109.

```
umur = df['umur'].to_numpy()
bins = [-1, 1, 10, 20, 60,109]
labels = [1, 2, 3, 4, 5 ]
data_enc['Umur Binning'] = pd.cut(data_enc['umur'], bins=bins,labels=labels)
```

Gambar 37 Proses *binning* atribut umur

Berdasarkan hasil *output* kode diatas, melalui gambar berikut ini dapat kita lihat terdapat atribut baru yaitu Umur *Binning* yang menampung nilai dari label tersebut dimana label tersebut terdiri dari 1-5. Dimana dari *output* tersebut sudah dikategorikan kedalam beberapa bagian sesuai dengan *labels* yang telah dideklarasikan.

```
data_enc[['umur','Umur Binning']].head(10)
```

	umur	Umur Binning
0	64	5
1	45	4
2	34	4
3	34	4
4	27	4
5	0	1
6	73	5
7	64	5
8	21	4
9	44	4

Gambar 38 *Output* proses *binning* atribut umur

.

2. LoS (*Length of Stay*)

   Pada atribut LoS (*Length of Stay*) akan diubah kedalam beberapa kategori dengan interval tertentu yaitu 1-5: short stay, 6-10: medium stay, dan > 10: long stay. Bins merupakan interval antar label dimana nilai terendah adalah 1 sehingga nilai terendah dalam bins adalah -1.

```
umur = df['los'].to_numpy()
bins = [-1, 5, 10, 11]
labels = [1, 2, 3]
data_enc['Los Binning'] = pd.cut(data_enc['los'], bins=bins,labels=labels)
```

Gambar 39 Proses LoS

Berdasarkan hasil *output* kode diatas, melalui gambar berikut ini dapat kita lihat terdapat atribut baru yaitu *Los Binning* yang menampung nilai dari label tersebut dimana

label tersebut terdiri dari 1-3. Dimana dari *output* tersebut sudah dikategorikan kedalam beberapa bagian sesuai dengan labels yang telah dideklarasikan.

```
data_enc[['los','Los Binning']].head(10)
```

	los	Los Binning
0	0.0	1
1	2.0	1
2	0.0	1
3	0.0	1
4	0.0	1
5	0.0	1
6	2.0	1
7	0.0	1
8	0.0	1
9	2.0	1

Gambar 40 *Output* proses LoS

3.3 Data Labelling

Pelabelan data adalah proses menetapkan tag ataupun label dimana mengubah label kata menjadi bentuk numerik. Pada dataset yang digunakan sudah terdapat label dengan tipe datanya adalah numerik. Jumlah nilai unik pada atribut ini terdiri dari 2 *value* yaitu 0 dan 1. Dimana 0 adalah tidak *fraud* dan 1 adalah *fraud*.

label
1
1
1
1
1

Gambar 41 *Value* pada label

4. MODELLING DAN MODEL EVALUATION

Tahap keempat pada metodologi CRISP-DM untuk melakukan prediksi terhadap *fraud* untuk klaim dalam Rumah Sakit adalah *modelling*. Pembangunan model dengan menggunakan metode *k-nearest neighbor* (KNN) yaitu algoritma *machine learning* sederhana dan terawasi yang dapat digunakan untuk menyelesaikan masalah klasifikasi dan regresi. *K-nearest neighbor* (KNN) bekerja dengan mencari jarak antara query dan semua contoh dalam data, memilih contoh nomor tertentu (K) yang paling dekat dengan query, kemudian memilih label yang paling sering (dalam kasus klasifikasi) atau rata-rata label (dalam kasus regresi). Dalam kasus klasifikasi dan regresi, memilih K yang tepat untuk data dapat dilakukan dengan mencoba beberapa K dan memilih salah satu yang terbaik.

4.1 Select Modelling Technique

Sesuai dengan tujuan penelitian, yaitu untuk mengembangkan model *data mining* yang ditujukan untuk membantu rumah sakit dalam melakukan pendeteksian terhadap penipuan yang terjadi di rumah sakit terkait klaim pelayanan. Pada pengerjaan proyek data mining ini, model yang diterapkan dalam menyelesaikan masalah klasifikasi dan regresi yaitu menggunakan menggunakan algoritma KNN (*K-Nearest Neighbors*) pada data BPJS (Badan Penyelenggaraan Jaminan sosial). Teknik *k-nearest neighbors* (KNN) yaitu algoritma *machine learning* sederhana dan terawasi yang bekerja dengan mencari jarak antara *query* dan semua contoh dalam data, memilih contoh nomor tertentu (K) yang paling dekat dengan *query*, kemudian memilih label yang paling sering (dalam kasus klasifikasi) atau rata-rata label (dalam kasus regresi). Dalam kasus c, memilih K yang tepat untuk data dapat dilakukan dengan mencoba beberapa K dan memilih salah satu yang terbaik. Algoritma KNN menggunakan '*feature similarity*' untuk memprediksi nilai dari setiap titik data baru. Ini berarti bahwa titik baru diberi nilai berdasarkan seberapa miripnya titik tersebut dalam *training set*.

4.2 Generate Test Design

Sebelum membangun model klasifikasi dan regresi menggunakan *k-nearest neighbors*, perlu dilakukan pembuatan prosedur atau mekanisme untuk menguji kualitas dan validitas model. Terlebih dulu, untuk mengurangi tingkat kesalahan maka dataset yang digunakan dipisahkan menjadi dua bagian yaitu *train* dan *test*.

4.3 Build Model

Dalam proses ini, terlebih dahulu variabel bebas (X) dan variabel terikat (Y) akan didefinisikan. Menggunakan variabel yang ditentukan, kami membagi data menjadi training set dan testing set yang akan digunakan untuk pemodelan dan evaluasi. Proses membagi data dilakukan dengan menggunakan algoritma 'train_test_split' dengan python.

```
from sklearn.model_selection import train_test_split
data_enc_new = data_enc.drop(labels=['label'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(data_enc_new, data_enc['label'], test_size=0.2)
```

Gambar 42 Membagi data

Pada pengerjaan proyek data mining ini, model yang diterapkan dalam menyelesaikan masalah klasifikasi dan regresi adalah KNN (*K-Nearest Neighbors*). Jadi perlu untuk menemukan nilai K optimal untuk mendapatkan yang terbaik.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_regression

selector = SelectKBest(score_func=chi2, k=10)
selector.fit(X_train, y_train)
```

```
SelectKBest(score_func=<function chi2 at 0x7fd4e7013560>)
```

Gambar 43 Nilai K optimal

Kemudian hasil train split tersebut trainX_best = 160173, dan testX_best 40044.

```
vector_names = list(X_train.columns[selector.get_support(indices=True)])
print(vector_names)
```

```
['kdkc', 'umur', 'los', 'dx2_p00_p96', 'proc39_45', 'typeppk_C ', 'typeppk_SC', 'typeppk_D ', 'cmg_H', 'cmg_O']
```

```
trainX_best = X_train[vector_names]
testX_best = X_test[vector_names]

print(trainX_best.shape)
print(testX_best.shape)
```

```
(160173, 10)
(40044, 10)
```

Gambar 44 Hasil train dan test split

Untuk menentukan nilai n_neighbours yang paling tepat, digunakan *code* berikut yang bertujuan men-*tuning* nilai n dimulai dari nilai 1 sampai nilai 30.

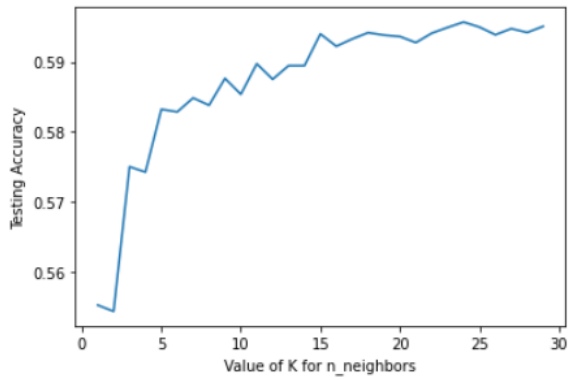
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
k_range = range(1,30)
scores = {}
scores_list = []

for k in k_range:
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train, y_train)
  y_pred = knn.predict(X_test)
  scores[k] = metrics.accuracy_score(y_test, y_pred)
  scores_list.append(metrics.accuracy_score(y_test, y_pred))
```

Gambar 45 Proses *Tunning*

Dari hasil *tunning* berikut, diperlihatkan bahwa nilai n dengan hasil akurasi tertinggi adalah pada nilai n yang ke-30, sehingga nilai n yang digunakan adalah nilai n = 30.

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(k_range,  scores_list)
plt.xlabel('Value of K for n_neighbors')
plt.ylabel('Testing Accuracy')
```

```
Text(0, 0.5, 'Testing Accuracy')
```



Gambar 46 *Output* proses *tunning*

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

knn = KNeighborsClassifier(n_neighbors=30)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
scores = metrics.accuracy_score(y_test, y_pred)
print('Accuracy score: ', scores )
```

```
Accuracy score:  0.5924483068624513
```

Gambar 47 Nilai akurasi

Pada tahap ini akan ditampilkan *Classification report*, *Confusion matrix* dan *Accuracy score* dari model. *Classification report* digunakan untuk mengukur kualitas prediksi dari algoritma klasifikasi. Berapa banyak prediksi yang Benar dan berapa banyak yang Salah. *Confusion matrix* digunakan sebagai visualisasi dari model klasifikasi untuk menunjukkan seberapa baik model dalam memprediksi hasil jika dibandingkan dengan yang asli. Biasanya, hasil prediksi disimpan dalam variabel yang kemudian diubah menjadi tabel korelasi. *Accuracy score* adalah salah satu metrik evaluasi paling dasar yang digunakan untuk mengevaluasi model klasifikasi. Skor akurasi dihitung hanya dengan membagi jumlah prediksi yang benar yang dibuat oleh model dengan jumlah total prediksi yang dibuat oleh model.

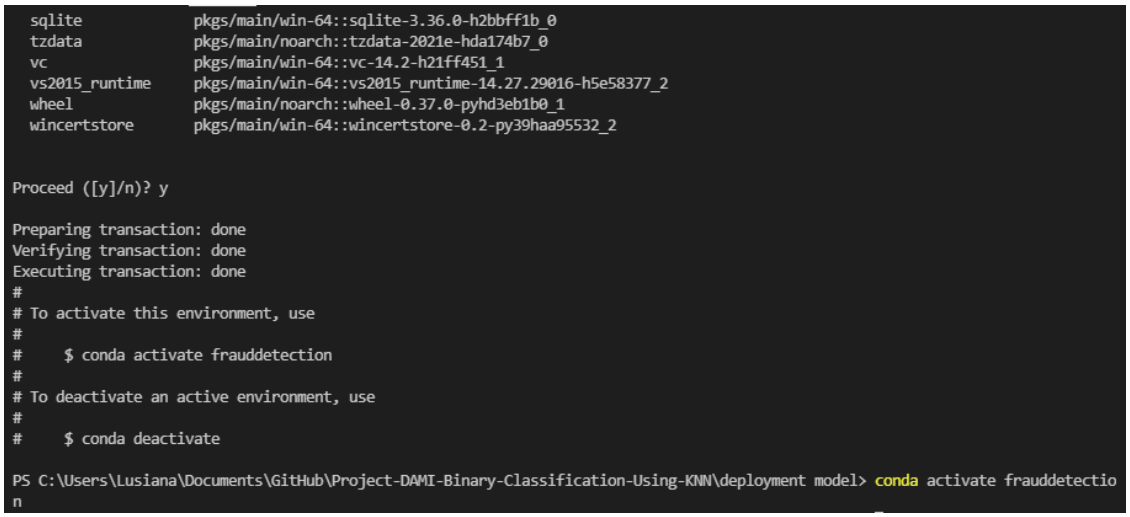
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
print("Classification report =\n",classification_report(y_test, y_pred))
print("Confusion report =\n",confusion_matrix(y_test, y_pred))
print("Accuracy score =\n",accuracy_score(y_test, y_pred))
```

```
Classification report =
               precision    recall  f1-score   support

           0       0.59      0.63      0.61     20131
           1       0.60      0.55      0.57     19913

    accuracy                           0.59     40044
   macro avg       0.59      0.59      0.59     40044
weighted avg       0.59      0.59      0.59     40044

Confusion report =
 [[12687  7444]
 [ 8876 11037]]
Accuracy score =
 0.5924483068624513
```

Gambar 48 *Classification report*, *Confusion matrix* dan *Accuracy score* dari model

5. DEPLOYMENT

1. Open folder deployment-model pada tools vscode
2. Buka command prompt pada visual studio code dan ketikkan conda create –name frauddetection python=3.9 dimana nama yang digunakan adalah frauddetection. Kemudian ketikan y dan enter jika terdapat pertanyaan.

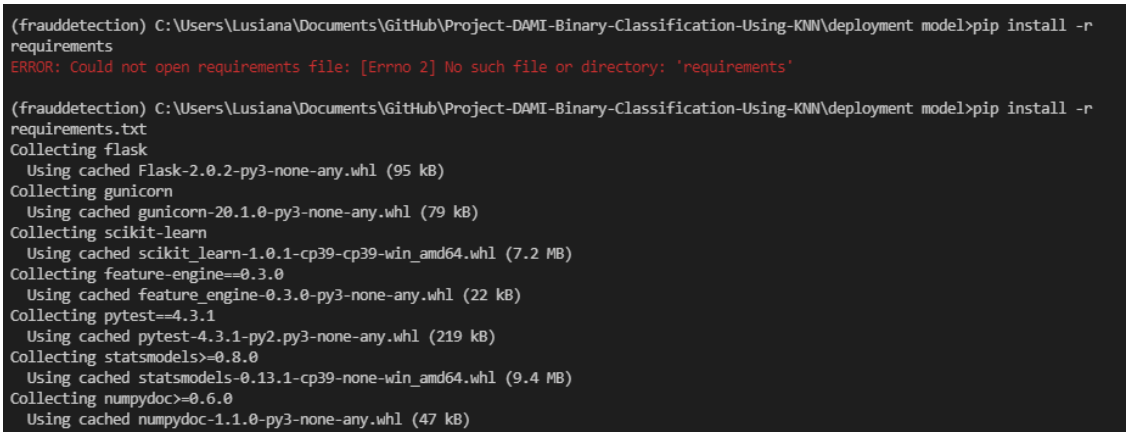


Gambar 49 *Command prompt* VSCode

3. Aktifkan virtual environment menggunakan perintah **conda activate frauddetection** dan tekan enter. Pada terminal yang digunakan akan terlihat nama virtual environment yang aktif yaitu frauddetection.

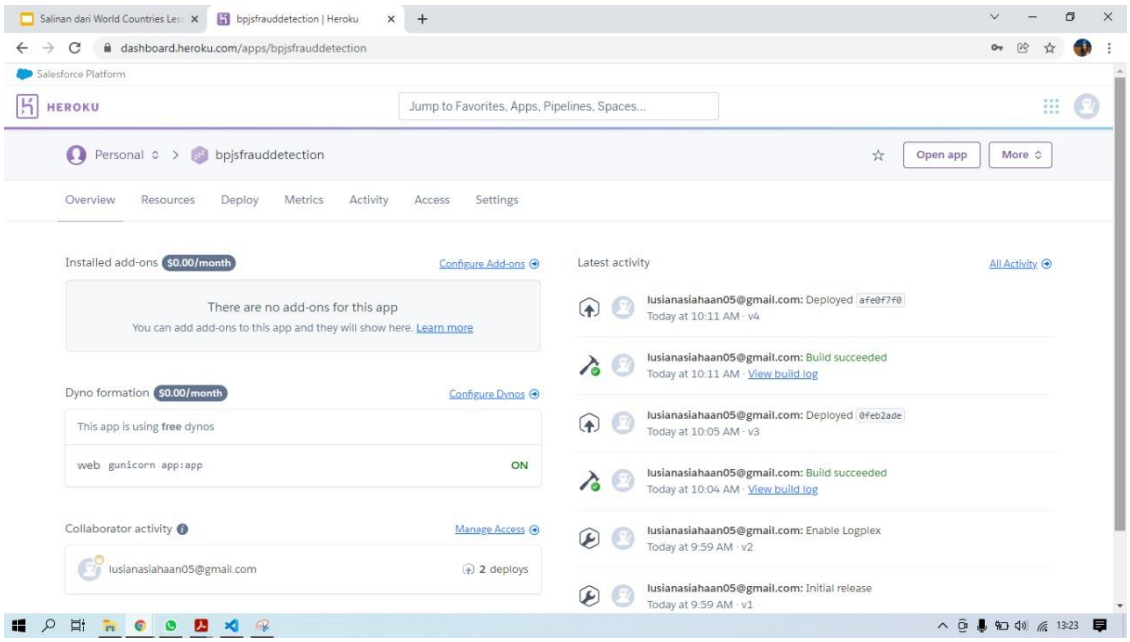
Gambar 50 Virtual Env

4. Install semua dependencies/library yang dibutuhkan dengan menggunakan perintah **pip install -r requirements.txt.**



Gambar 51 *Install dependencies*

5. Sebelum melakukan deploy website model fraud detection menggunakan Heroku, pastikan sudah memiliki akun dan login ke Heroku serta sudah menginstall Heroku CLI. Kemudian buka dashboard Heroku dan klik open app dan create new app.



Gambar 52 Heroku

6. Masukkan name aplikasi sesuai keinginan, lalu klik tombol create. Maka akan diarahkan ke halaman awal bagian deploy.
7. Login pada Heroku CLI dengan mengetikkan perintah heroku login dan tekan enter. Kemudian tekan sembarang kunci/tombol keyboard untuk membuka browser. Maka akan diarahkan untuk login melalui browser. Setelah selesai, buka kembali VSCode

```
(frauddetection) C:\Users\Lusiana\Documents\GitHub\Project-DAMI-Binary-Classification-Using-KNN\deployment model>heroku login -i

 »   Warning: heroku update available from 7.53.0 to 7.59.2.
heroku: Enter your login credentials
Email [lusianasiahaan05@gmail.com]: lusianasiahaan05@gmail.com
Password: **********
Logged in as lusianasiahaan05@gmail.com
```

Gambar 53 *Login* Heroku

8. Buat *git repository* dan *upload* semua *file* yang ada dalam Heroku dengan menggunakan perintah dibawah ini secara berurutan.

```
(frauddetection) C:\Users\Lusiana\Documents\GitHub\Project-DAMI-Binary-Classification-Using-KNN\deployment model>git init
Initialized empty Git repository in C:/Users/Lusiana/Documents/GitHub/Project-DAMI-Binary-Classification-Using-KNN/deployment mo
del/.git/
```

Gambar 54 *Upload file* dalam Heroku

- Git ini
- Heroku git:remote -a bpjsfrauddetection

```
(frauddetection) C:\Users\Lusiana\Documents\GitHub\Project-DAMI-Binary-Classification-Using-KNN\deployment model>heroku git:remo
te -a bpjsfrauddetection
 »   Warning: heroku update available from 7.53.0 to 7.59.2.
set git remote heroku to https://git.heroku.com/bpjsfrauddetection.git
```

Gambar 55 Heroku git:remote

- Git add .

```
(frauddetection) C:\Users\Lusiana\Documents\GitHub\Project-DAMI-Binary-Classification-Using-KNN\deployment model>git add .
```

Gambar 56 Git *add*

- Git commit -am “bpjs commit one”

```
(frauddetection) C:\Users\Lusiana\Documents\GitHub\Project-DAMI-Binary-Classification-Using-KNN\deployment model>git commit -am
"bpjs commit one"
[master (root-commit) 0feb2ad] bpjs commit one
 6 files changed, 677 insertions(+)
 create mode 100644 BPJSFraudClassifier.pkl
 create mode 100644 Procfile.txt
 create mode 100644 app.py
 create mode 100644 requirements.txt
 create mode 100644 template/index.html
 create mode 100644 template/result.html
```

Gambar 57 Git *commit one*

- Git commit -am “bpjs commit two”

```
(frauddetection) C:\Users\Lusiana\Documents\GitHub\Project-DAMI-Binary-Classification-Using-KNN\deployment model>git commit -am
"bpjs commit two"
[master afe0f7f] bpjs commit two
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename Procfile.txt => Procfile (100%)
```

Gambar 58 Git *commit two*

- Git push Heroku master

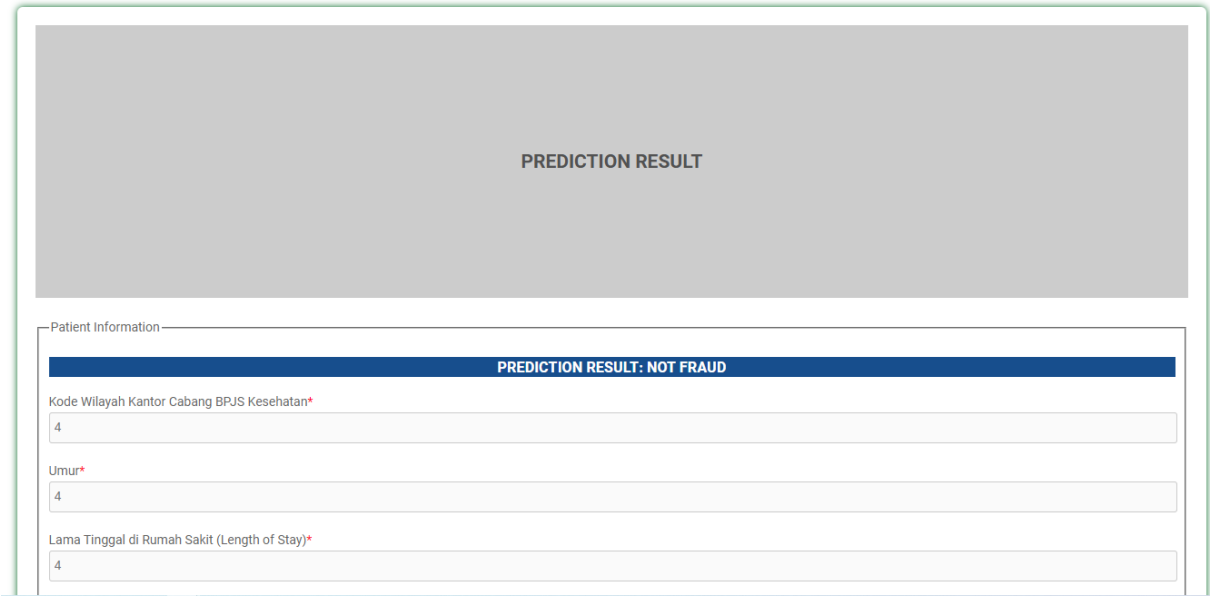
```
(frauddetection) C:\Users\Lusiana\Documents\GitHub\Project-DAMI-Binary-Classification-Using-KNN\deployment model>git push heroku
 master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 235 bytes | 235.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the same version as the last build: python-3.9.9
remote:        To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> No change in requirements detected, installing from cache
```

Gambar 59 Git *push*

```
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the same version as the last build: python-3.9.9
remote:        To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> No change in requirements detected, installing from cache
remote: -----> Using cached install of python-3.9.9
remote: -----> Installing pip 21.3.1, setuptools 57.5.0 and wheel 0.37.0
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote: -----> Discovering process types
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 190.7M
remote: -----> Launching...
remote:        Released v4
remote:        https://bpjsfrauddetection.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/bpjsfrauddetection.git
   0feb2ad..afe0f7f  master -> master
```

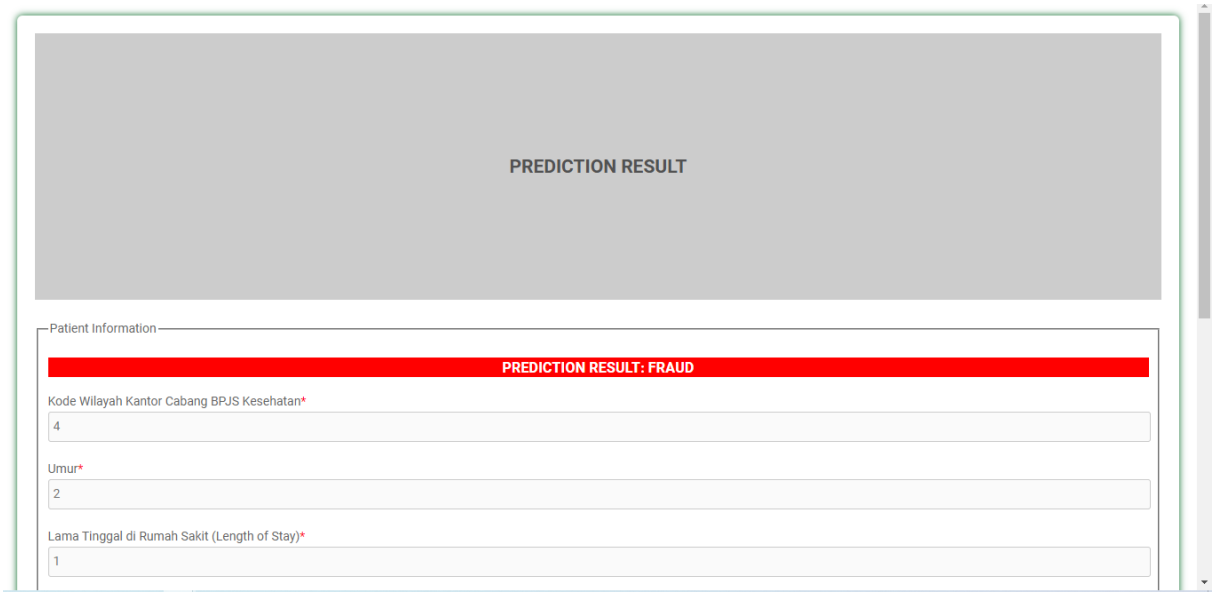
Gambar 60 *Deployment* berhasil

Gambar 60 menunjukkan tampilan jika berhasil melakukan deployment model bpjsfrauddetection dalam bentuk website menggunakan Heroku. Berikut merupakan tampilan ketika data input di prediksi dan menghasil keluaran berupa **Not Fraud**.



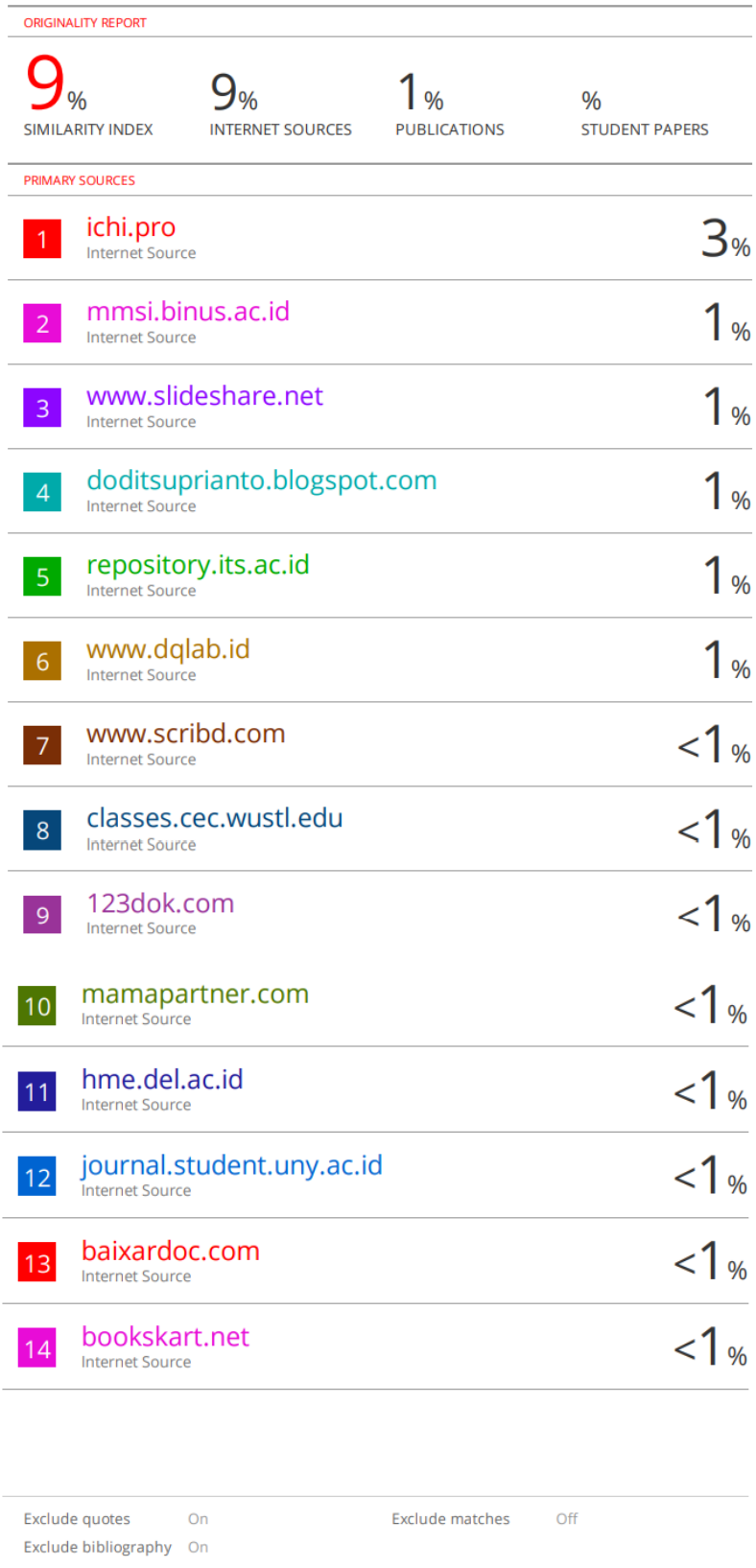
Gambar 61 *Input not fraud*

Berikut merupakan tampilan ketika data input di prediksi dan menghasil keluaran berupa **Fraud.**



Gambar 62 *Input fraud*

Lampiran



ORIGINALITY REPORT

9%	9%	1%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ichi.pro Internet Source	3%
2	mmsi.binus.ac.id Internet Source	1%
3	www.slideshare.net Internet Source	1%
4	doditsuprianto.blogspot.com Internet Source	1%
5	repository.its.ac.id Internet Source	1%
6	www.dqlab.id Internet Source	1%
7	www.scribd.com Internet Source	<1%
8	classes.cec.wustl.edu Internet Source	<1%
9	123dok.com Internet Source	<1%
10	mamapartner.com Internet Source	<1%
11	hme.del.ac.id Internet Source	<1%
12	journal.student.uny.ac.id Internet Source	<1%
13	baixardoc.com Internet Source	<1%
14	bookskart.net Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Gambar 63 Hasil turnitin laporan