**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 ISI 2](#_Toc90934714)

[DAFTAR GAMBAR 3](#_Toc90934715)

[1. BUSINESS UNDERSTANDING 5](#_Toc90934716)

[**1.1 Determine Business Objective 5**](#_Toc90934717)

[**1.2 Situation Assessment 6**](#_Toc90934718)

[**1.3 Determine Data Mining Goal 6**](#_Toc90934719)

[**1.4 Produce Project Plan 7**](#_Toc90934720)

[2. DATA UNDERSTANDING 9](#_Toc90934721)

[**2.1 Collect Initial Data 9**](#_Toc90934722)

[**2.2. Describe Data 9**](#_Toc90934723)

[**2.3 Explore Data 10**](#_Toc90934724)

[**2.4 Verify Data Quality 15**](#_Toc90934725)

[3. DATA PREPARATION 16](#_Toc90934726)

[**3.1 Data Cleaning 16**](#_Toc90934727)

[**3.1.1 Check Null Value 16**](#_Toc90934728)

[**3.1.2. Check Outlier 16**](#_Toc90934729)

[**3.2 Feature Selection 21**](#_Toc90934730)

[**3.2 Data Transformation 23**](#_Toc90934731)

[**3.2.1 One Hot Encoding 23**](#_Toc90934732)

[**3.2.2 Standarisasi 25**](#_Toc90934733)

[**3.2.3 Binning 25**](#_Toc90934734)

[**3.3 Data Labelling 27**](#_Toc90934735)

[4. MODELLING DAN MODEL EVALUATION 28](#_Toc90934736)

[**4.1 Select Modelling Technique 28**](#_Toc90934737)

[**4.2 Generate Test Design 28**](#_Toc90934738)

[**4.3 Build Model 29**](#_Toc90934739)

[5. DEPLOYMENT 32](#_Toc90934740)

[Lampiran 37](#_Toc90934741)

# DAFTAR GAMBAR

[Gambar 1 Jadwal rencana pelaksanaan proyek 7](#_Toc89883023)

[Gambar 2 Tahapan proses *data mining* 8](#_Toc89883024)

[Gambar 3 Kode untuk membaca dataset 9](#_Toc89883025)

[Gambar 4 Kode untuk melihat kolom pada dataset 10](#_Toc89883026)

[Gambar 5 Kode untuk detail statistik dataset 10](#_Toc89883027)

[Gambar 6 Kode untuk melihat informasi kolom 11](#_Toc89883028)

[Gambar 7 Kode untuk melihat data duplikat 12](#_Toc89883029)

[Gambar 8 Visualisasi korelasi antar atribut 13](#_Toc89883030)

[Gambar 9 Value masing-masing atribut 14](#_Toc89883031)

[Gambar 10 Kode untuk melihat missing entries 15](#_Toc89883032)

[Gambar 11 Kode untuk melihat missing entries 16](#_Toc89883033)

[Gambar 12 Visualisasi *outlier* pada *feature* kdkc 17](#_Toc89883034)

[Gambar 13 Rentang *outlier* yang terdapat pada *feature* kdkc 17](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883035)

[Gambar 14 Visualisasi *outlier* pada *feature* severitylevel 17](#_Toc89883036)

[Gambar 15 Rentang *outlier* yang terdapat pada *feature* severitylevel 18](#_Toc89883037)

[Gambar 16 Visualisasi *outlier* pada *feature* los 18](#_Toc89883038)

[Gambar 17 Rentang *outlier* yang terdapat pada *feature* los 18](#_Toc89883039)

[Gambar 18 Visualisasi *outlier* pada *feature* proc80\_99 18](#_Toc89883040)

[Gambar 19 Rentang *outlier* yang terdapat pada *feature* proc80\_99 19](#_Toc89883041)

[Gambar 20 Perhitungan *quantile* pada *features* 19](#_Toc89883042)

[Gambar 21 Perhitungan nilai batas bawah dan batas atas *feature* kdkc 19](#_Toc89883043)

[Gambar 22 Indeks elemen dalam array *input* 20](#_Toc89883044)

[Gambar 23 *Chart boxplot* kdkc dengan *outlier* yang telah dihapus 20](#_Toc89883045)

[Gambar 24 *Chart boxplot* severitylevel dengan *outlier* yang telah dihapus 20](#_Toc89883046)

[Gambar 25 *Chart boxplot* los dengan *outlier* yang telah dihapus 21](#_Toc89883047)

[Gambar 26 *Chart boxplot* proc80\_99 dengan *outlier* yang telah dihapus 21](#_Toc89883048)

[Gambar 27 Kode untuk memilih *feature* yang akan dieliminasi 22](#_Toc89883049)

[Gambar 28 *Output* dari kode eliminasi 22](#_Toc89883050)

[Gambar 29 *Output* dari kode eliminasi *cont'd* 22](#_Toc89883051)

[Gambar 30 Eleminasi *feature* 23](#_Toc89883052)

[Gambar 31 Eleminasi *feature cont'd* 23](#_Toc89883053)

[Gambar 32 *Features dataset* setelah eliminasi 23](#_Toc89883054)

[Gambar 33 *Features data type* 24](#_Toc89883055)

[Gambar 34 Transformasi pada *features datatype object* 24](#_Toc89883056)

[Gambar 35 Bentuk numerik *feature* cmg 24](#_Toc89883057)

[Gambar 36 Proses standarisasi 25](#_Toc89883058)

[Gambar 37 Proses binning atribut umur 26](#_Toc89883059)

[Gambar 38 *Output* proses *binning* atribut umur 26](#_Toc89883060)

[Gambar 39 Proses LoS 26](#_Toc89883061)

[Gambar 40 *Output* proses LoS 27](#_Toc89883062)

[Gambar 41 Value pada label 27](#_Toc89883063)

[Gambar 42 Membagi data 29](#_Toc89883064)

[Gambar 43 Nilai K optimal 29](#_Toc89883065)

[Gambar 44 Hasil train dan test split 29](#_Toc89883066)

[Gambar 45 Proses *Tunning* 30](#_Toc89883067)

[Gambar 46 *Output* proses *tunning* 30](#_Toc89883068)

[Gambar 47 Nilai akurasi 30](#_Toc89883069)

[Gambar 48 *Classification report, Confusion matrix* dan *Accuracy score* dari model 31](#_Toc89883070)

[Gambar 49 *Command prompt* VSCode 32](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883071)

[Gambar 50 *Virtual Env* 32](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883072)

[Gambar 51 *Install dependencies* 32](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883073)

[Gambar 52 Heroku 33](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883074)

[Gambar 53 *Login* Heroku 33](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883075)

[Gambar 54 *Upload file* dalam Heroku 33](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883076)

[Gambar 55 Heroku git:remote 34](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883077)

[Gambar 56 Git *add* 34](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883078)

[Gambar 57 Git *commit one* 34](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883079)

[Gambar 58 Git *commit two* 34](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883080)

[Gambar 59 Git *push* 34](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883081)

[Gambar 60 *Deployment* berhasil 35](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883082)

[Gambar 61 *Input not fraud* 35](#_Toc89883083)

[Gambar 62 *Input fraud* 36](#_Toc89883084)

[Gambar 63 Hasil turnitin laporan 37](file:///E:\MateriKuliah\Semester%207\DAMI\Proyek\01-LAP-DAMI.docx#_Toc89883085)

# 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

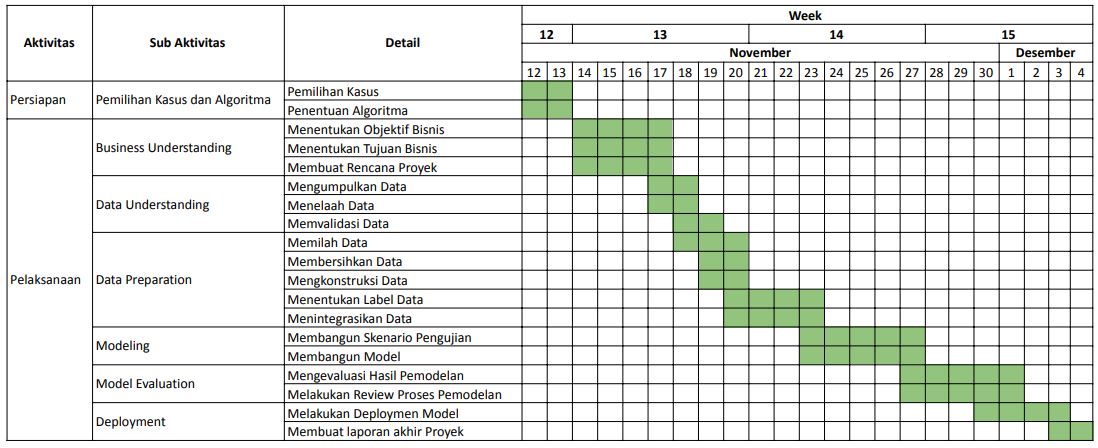
Sektor kesehatan menjadi target yang menarik bagi para *fraudsters*. Ketersediaan sejumlah besar data memungkinkan untuk mengatasi masalah ini dengan penerapan teknik *data mining*, membuat proses audit lebih efisien dan efektif. Penelitian ini bertujuan untuk mengembangkan model data mining yang ditujukan 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

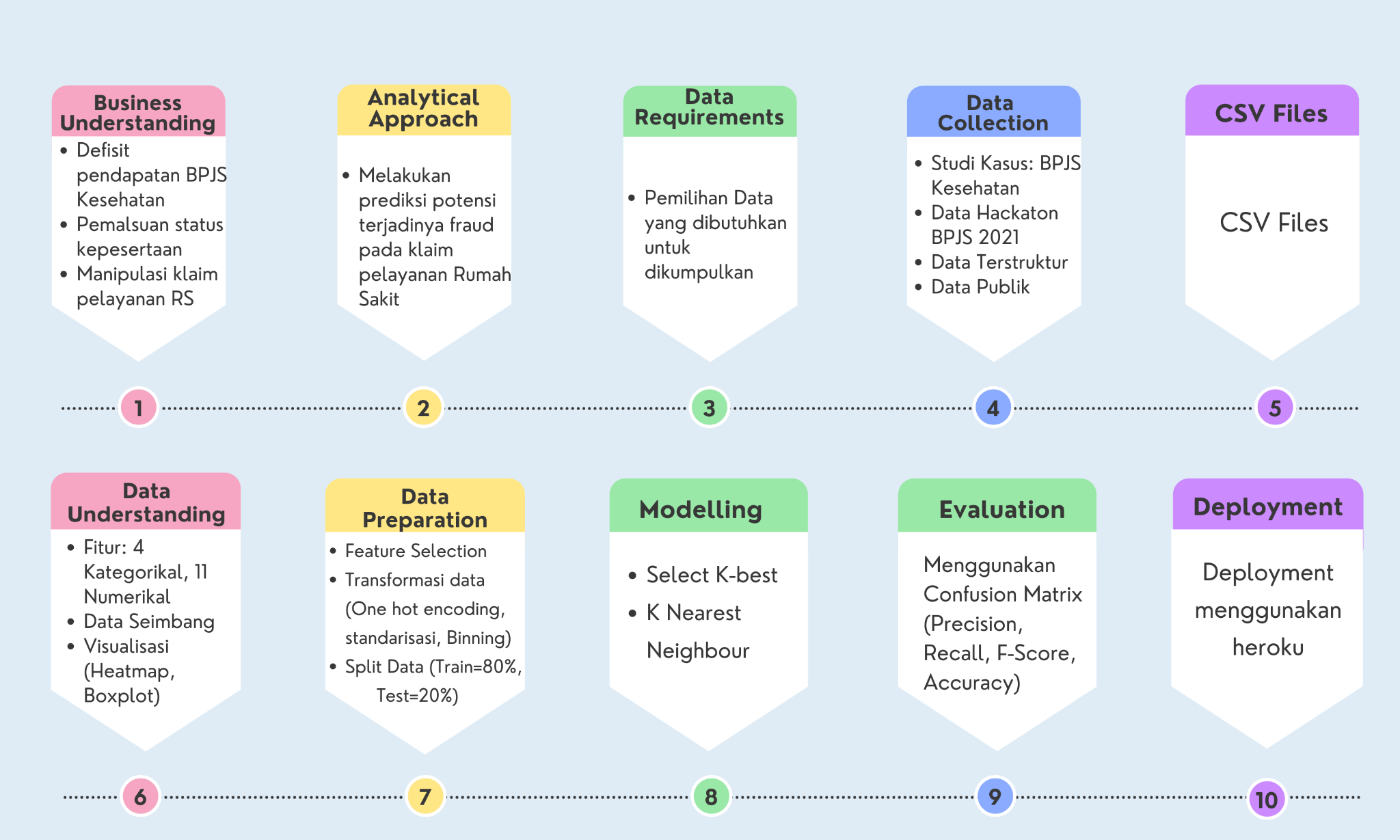
Tahap perencanaan yang dilakukan untuk mencapai tujuan data mining dan mencapai tujuan bisnis pada penelitian “*Binary Classification Using KNN for BPJS Fraud Prediction*” ini adalah sebagai berikut.



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

# DATA UNDERSTANDING

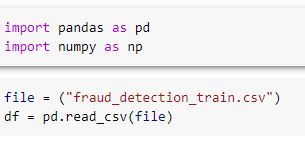
Data Understanding adalah tahapan selanjutnya yang dilakukan setelah *business understanding.* Pada tahapan ini dilakukan pengumpulan data, eksplorasi data, dan validasi data guna memahami kondisi dari dataset.

## 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.

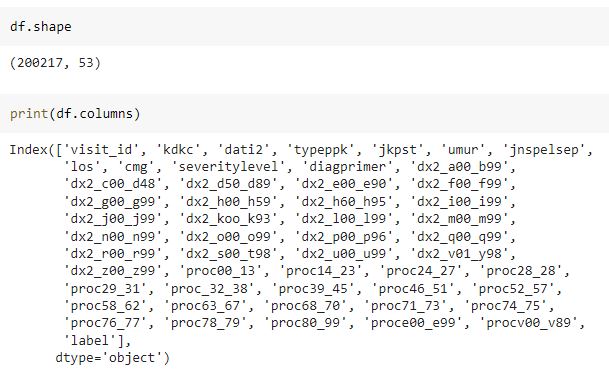
## 2.2. Describe Data

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.



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.



Gambar 4 Kode untuk melihat kolom pada dataset

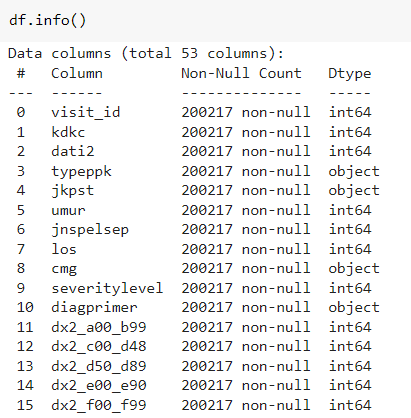
Pada hasil yang diberikan dapat dilihat nama dari 53 atribut yang ada pada dataset. Selanjutnya, untuk melihat detail statistik seperti persentil, rata-rata, standar deviasi, dan lain-lain dari atribut dalam dataset, digunakan fungsi df.describe().



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().



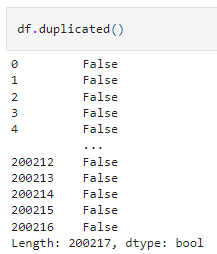
Gambar 6 Kode untuk melihat informasi kolom

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

**Table 1. Keterangan Atribut**

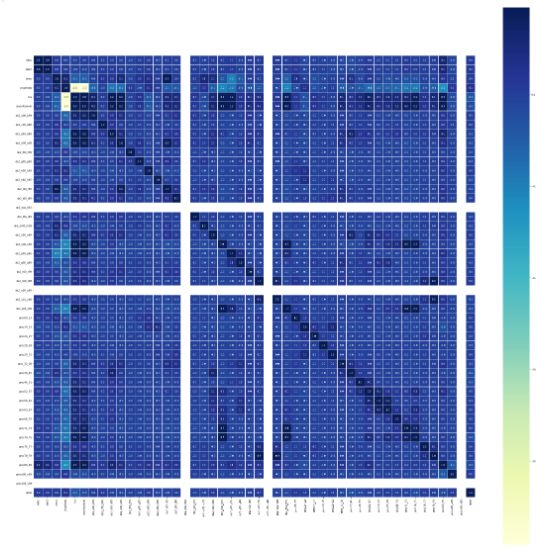
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Atribute** | **Keterangan** | **Tipe** | **Nilai** |
| 1 | Visit\_id | id kunjungan | Int64 | Id numerik |
| 2 | kdkc | kode wilayah kantor cabang BPJS Kesehatan | Int64 | Kode kdkc numerik |
| 3 | Dati2 | kode kabupaten/kota | Int64 | Kode 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

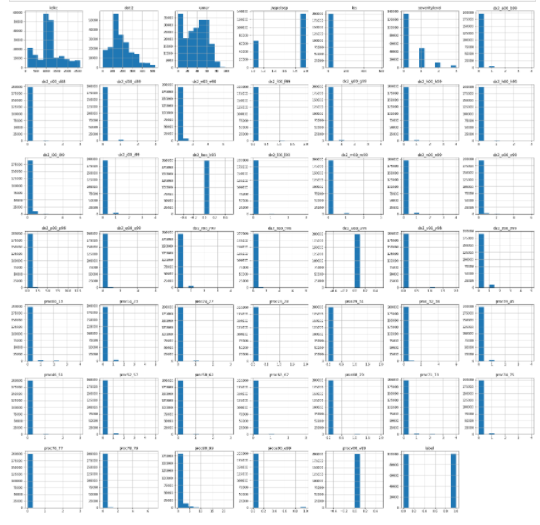
Korelasi antar atribut juga perlu untuk ditinjau. Oleh karena itu, korelasi antar masing-masing atribut divisualisasikan menggunakan visualisasi *heatmap*. Ukuran dari visualisasi diatur menggunakan *library* matplotlib, sementara untuk menampilkan heatmap menggunakan library seaborn. Melalui korelasi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

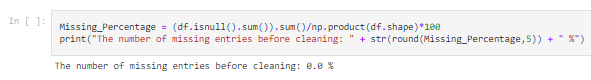
Setelah berhasil meninjau korelasi antar atribut, selanjutnya dilakukan penelaahan terhadap *value* dari masing-masing atribut. Dari tampilan visualisasi histogram pada Gambar 9, yaitu tampilan *value* setiap atribut dalam dataset, maka didapatkan hal-hal berikut ini:

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.

# DATA PREPARATION

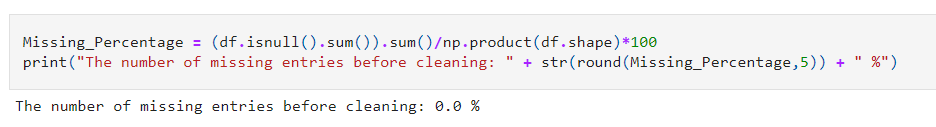
Setelah data diperiksa dan dikarakterisasi selama tahap *data understanding*, data tersebut kemudian disiapkan untuk tahapan *data mining* berikutnya yaitu *data preparation,* tahap selanjutnya pada fase CRISP-DM. Tahap *data preparation* merupakan tahapan untuk menyiapkan data awal, memilih variabel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

## 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*.

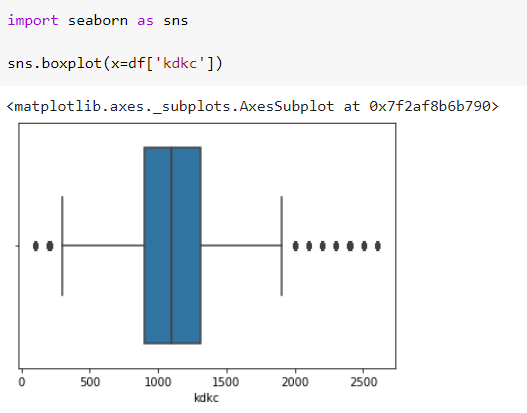


Gambar 11 Kode untuk melihat missing entries

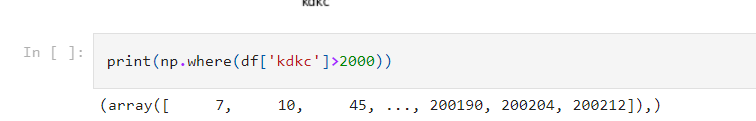
### **3.1.2. Check Outlier**

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

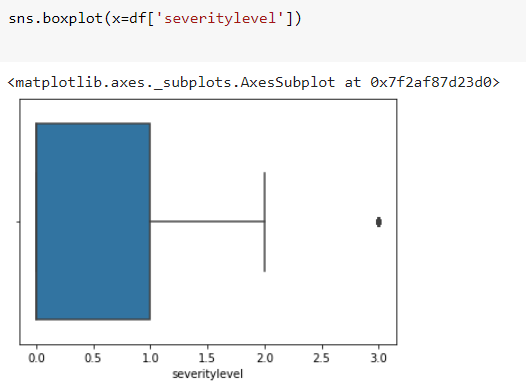


Gambar 12 Visualisasi outlier pada feature kdkc

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

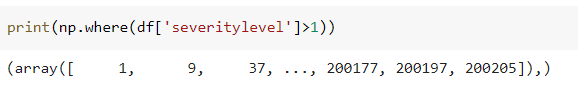
Gambar 13 Rentang outlier yang terdapat pada feature kdkc

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



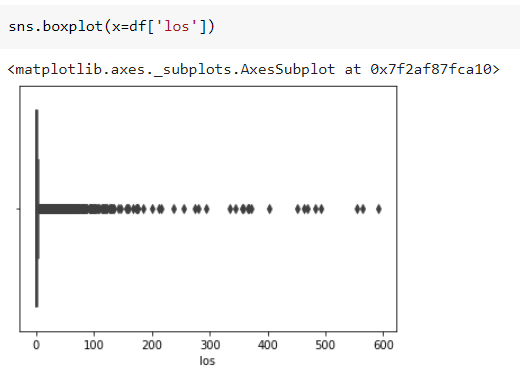
Gambar 14 Visualisasi outlier pada feature severitylevel

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



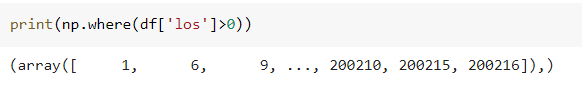
Gambar 15 Rentang outlier yang terdapat pada feature severitylevel

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



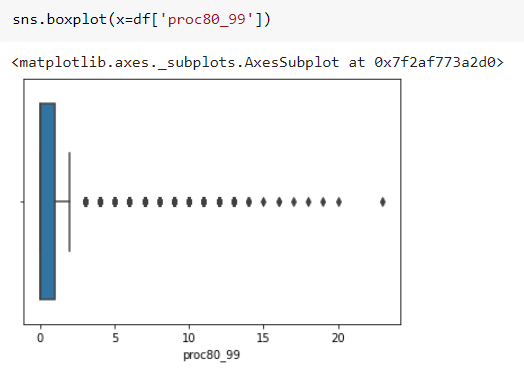
Gambar 16 Visualisasi outlier pada feature los

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



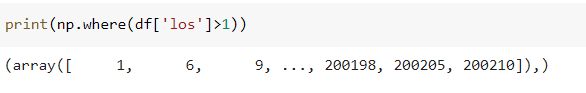
Gambar 17 Rentang outlier yang terdapat pada feature los

1. Berikut merupakan visualisasi *outlier* pada proc80\_99.



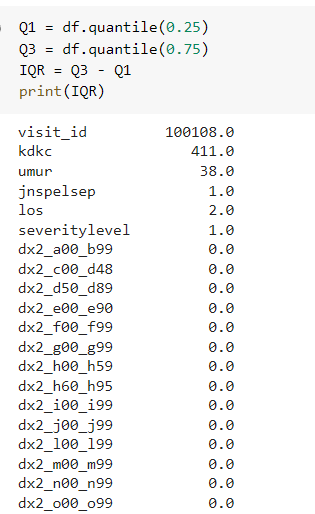
Gambar 18 Visualisasi outlier pada feature proc80\_99

Dan rentang dari *outlier* yang terdapat pada proc80\_99 adalah sebagai berikut:



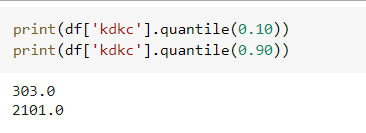
Gambar 19 Rentang outlier yang terdapat pada feature proc80\_99

*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*.



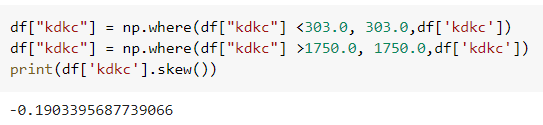
Gambar 20 Perhitungan quantile pada features

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.



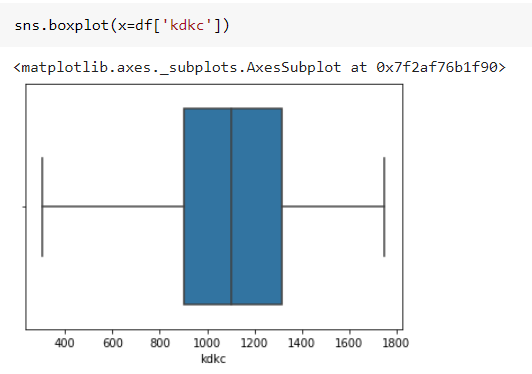
Gambar 21 Perhitungan nilai batas bawah dan batas atas feature kdkc

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.

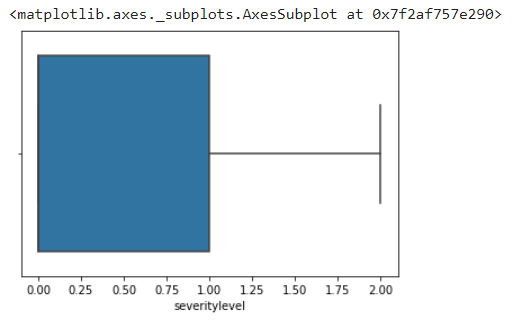


Gambar 22 Indeks elemen dalam array input

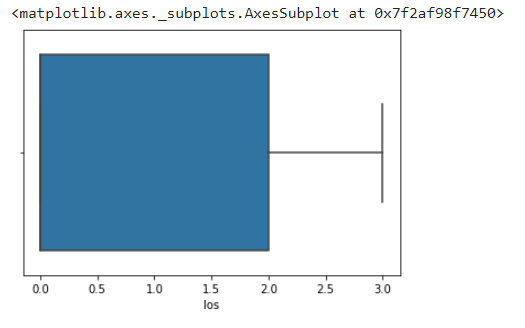
Berikut merupakan tampilan *chart* boxplot dengan *outlier* yang sudah dihapus.



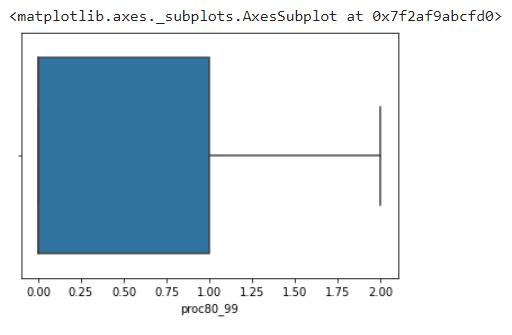
Gambar 23 Chart boxplot kdkc dengan outlier yang telah dihapus



Gambar 24 Chart boxplot severitylevel dengan outlier yang telah dihapus



Gambar 25 Chart boxplot los dengan outlier yang telah dihapus



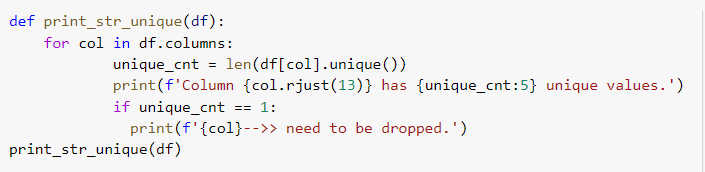
Gambar 26 Chart boxplot proc80\_99 dengan outlier yang telah dihapus

## 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:

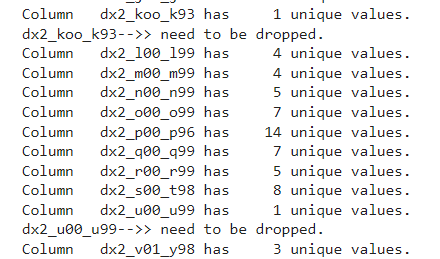
* Memungkinkan algoritma pembelajaran mesin untuk berlatih lebih cepat.
* Mengurangi kompleksitas model dan membuatnya lebih mudah untuk diinterpretasikan.
* Meningkatkan akurasi model jika subset yang tepat dipilih.
* Mengurangi *overfitting*.

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.

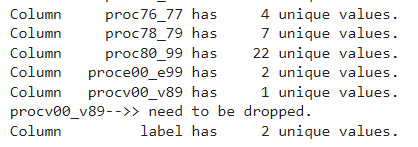


Gambar 27 Kode untuk memilih feature yang akan dieliminasi

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.



Gambar 28 Output dari kode eliminasi



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

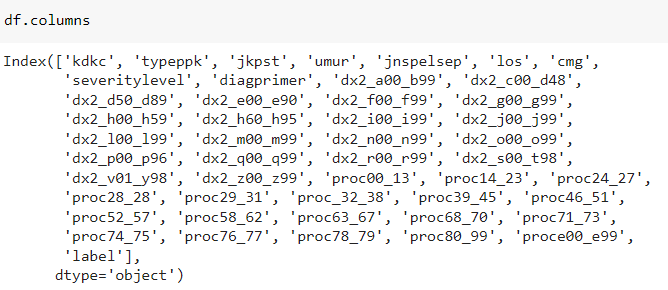


Gambar 30 Eleminasi feature



Gambar 31 Eleminasi feature cont'd

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



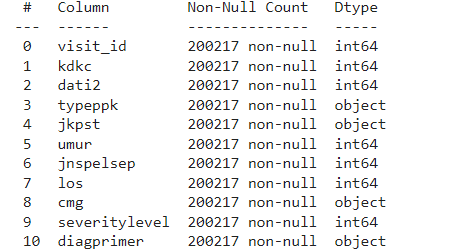
Gambar 32 Features dataset setelah eliminasi

## 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*).

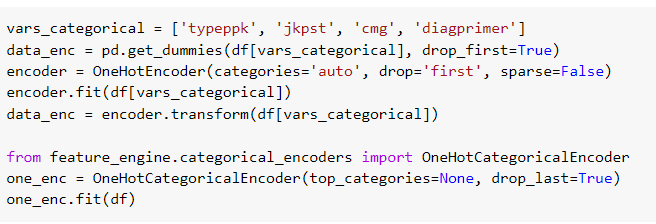
### **3.2.1 One Hot Encoding**

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.



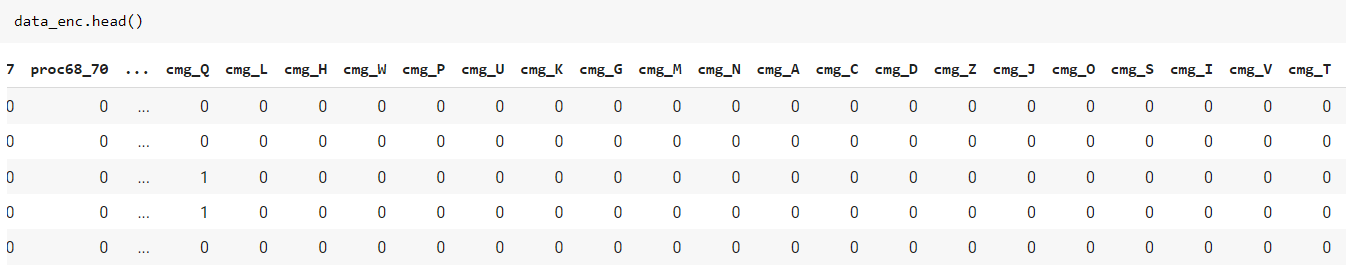
Gambar 33 Features data type

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.



Gambar 34 Transformasi pada features datatype object

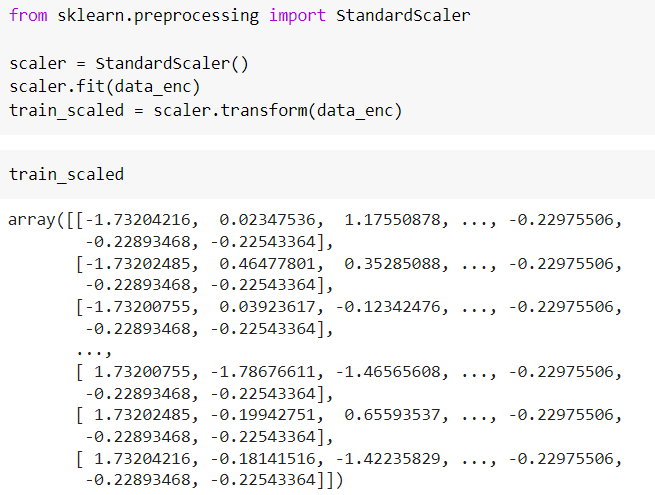
Berdasarkan hasil output dibawah ini dapat dilihat bahwa seperti feature cmg yang telah diubah kedalam bentuk numerik, maka cmg tersebut akan terbagi-bagi menjadi beberapa bagian dengan nilai 0 dan 1.



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.



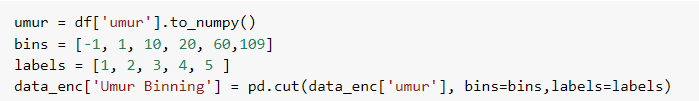
Gambar 36 Proses standarisasi

### **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:

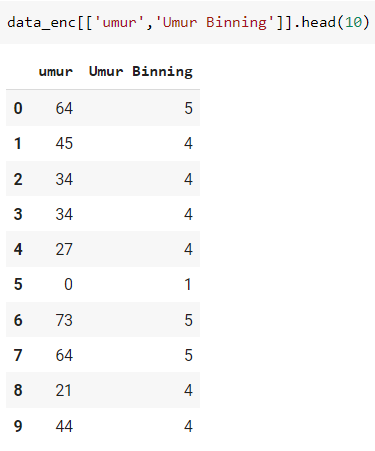
1. Umur

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.



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.

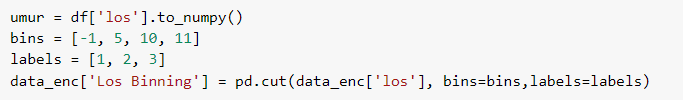
****

Gambar 38 Output proses binning atribut umur

.

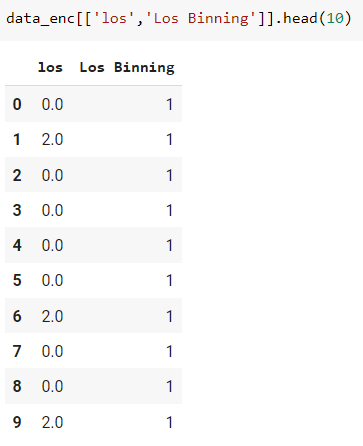
1. LoS (*Length of Stay*)

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.



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.



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*.



Gambar 41 Value pada label

# 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

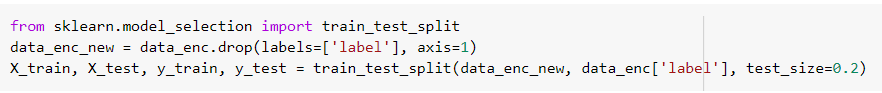
Sesuai dengan tujuan penelitian, yaitu untuk mengembangkan model *data mining* yang ditujukan untuk membantu rumah sakit dalam melakukan pendeteksian terhadap penipuan yang terjadi di rumah sakit terkait klaim pelayanan. Pada pengerjaan proyek data mining ini, model yang diterapkan dalam menyelesaikan 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*.

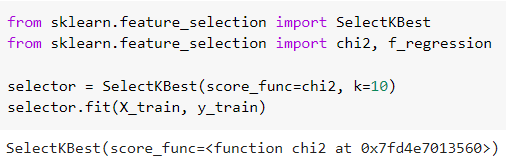
## 4.3 Build Model

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.



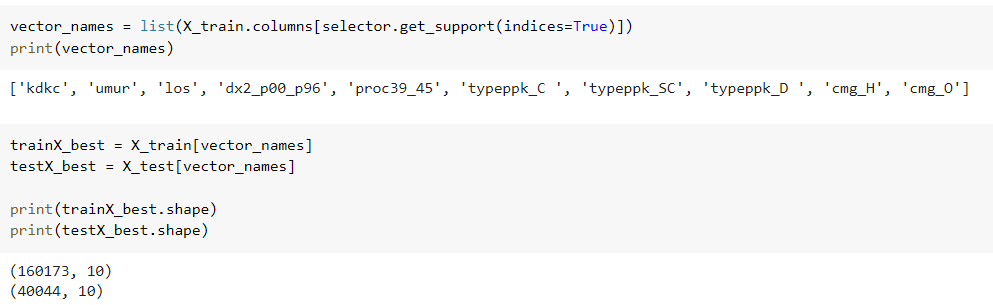
Gambar 42 Membagi data

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.



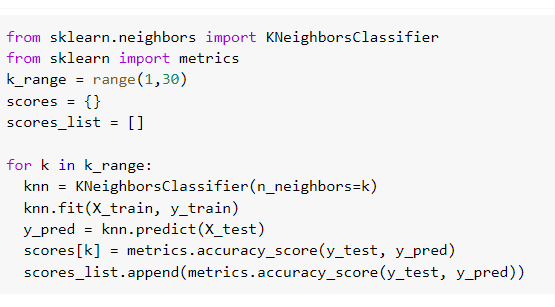
Gambar 43 Nilai K optimal

Kemudian hasil train split tersebut trainX\_best = 160173, dan testX\_best 40044.



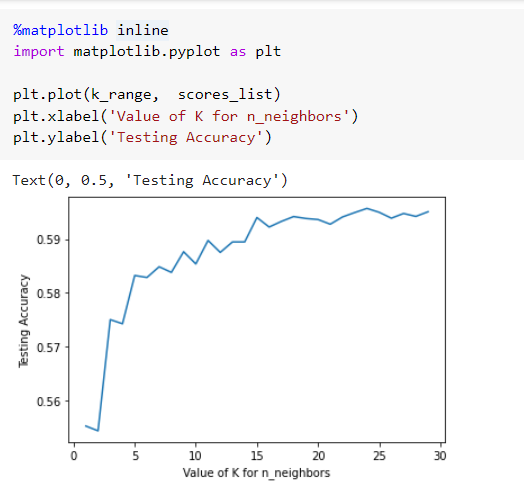
Gambar 44 Hasil train dan test split

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

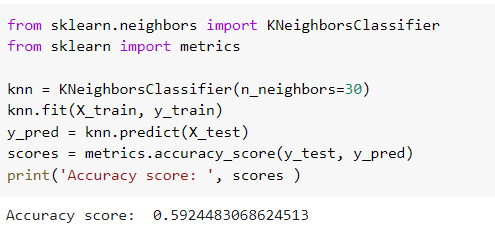


Gambar 45 Proses Tunning

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.

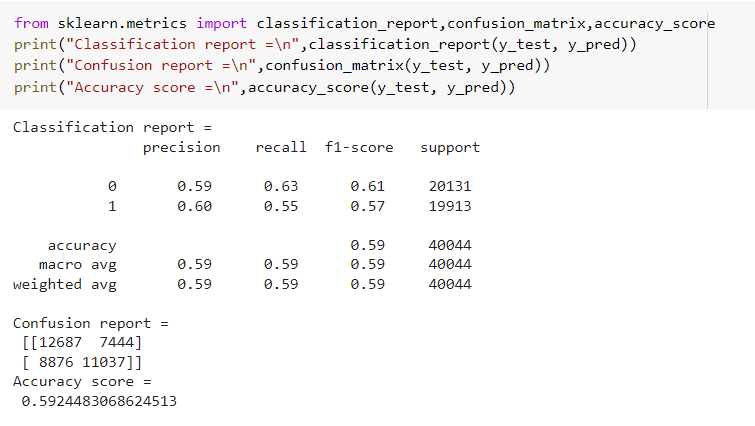


Gambar 46 Output proses tunning



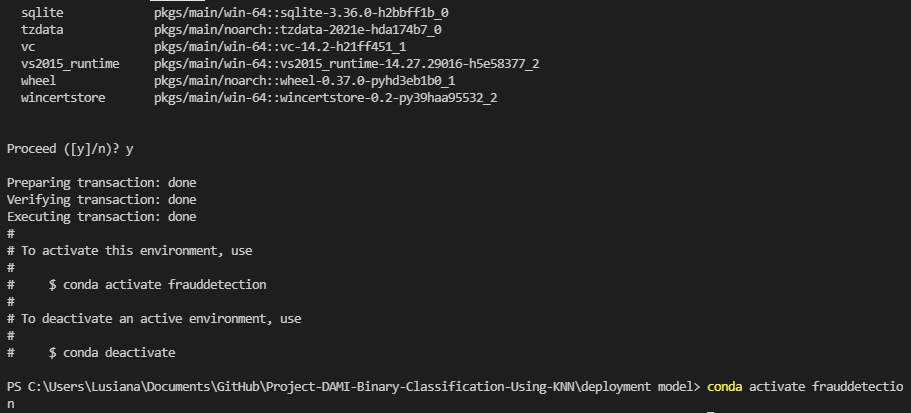
Gambar 47 Nilai akurasi

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.

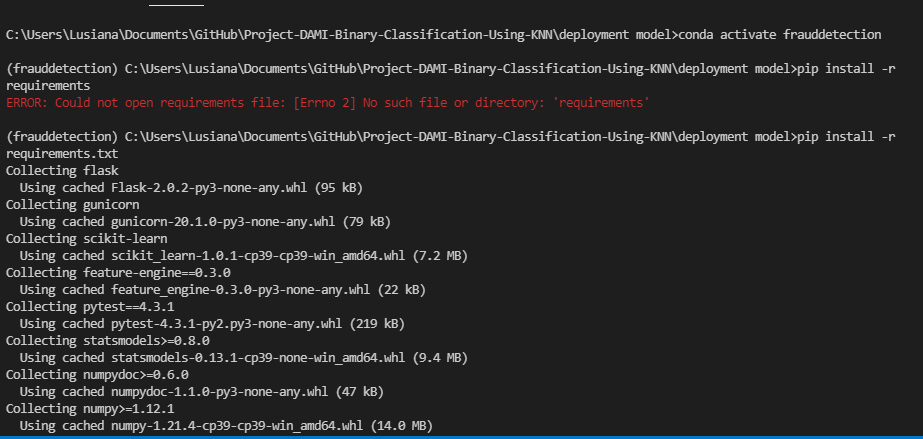


Gambar 48 Classification report, Confusion matrix dan Accuracy score dari model

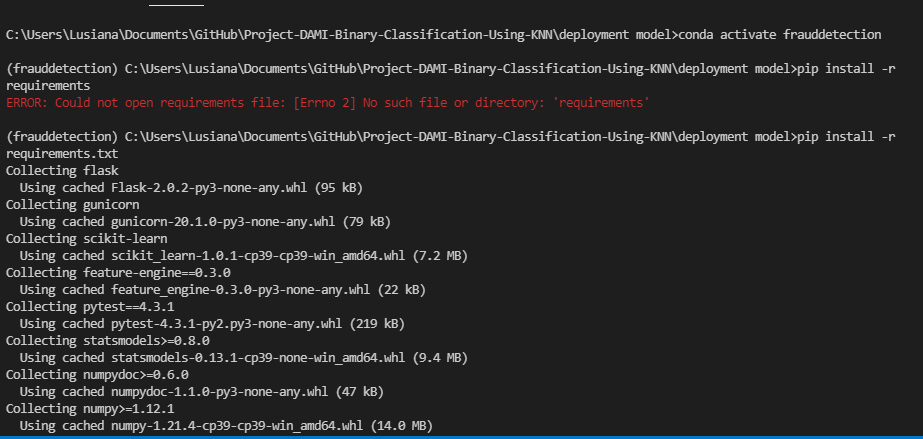
# DEPLOYMENT

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.

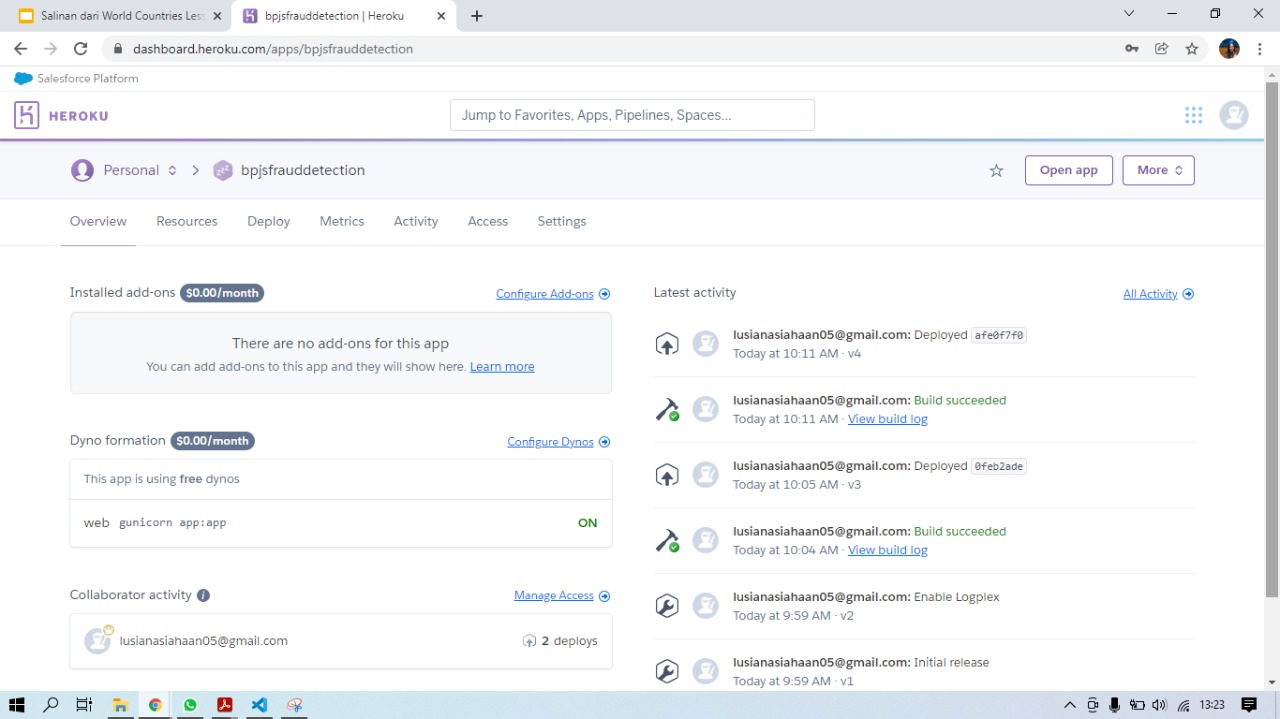
Gambar 49 Command prompt VSCode

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

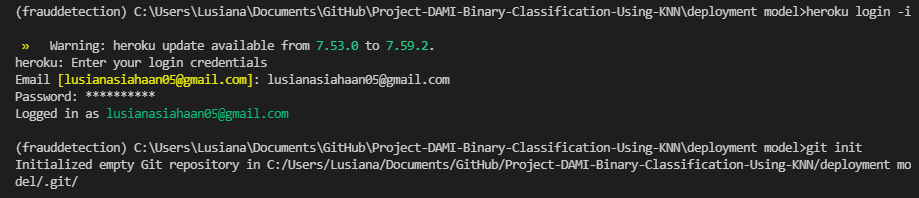
Gambar 50 Virtual Env

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

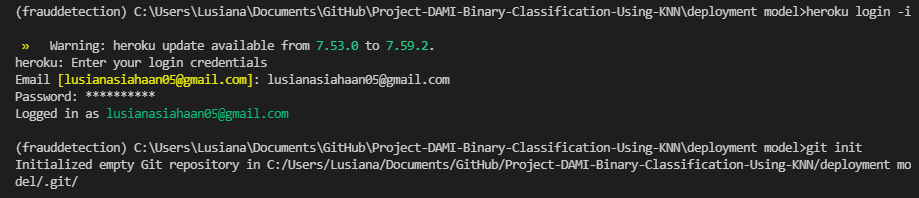
Gambar 51 Install dependencies

1. 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

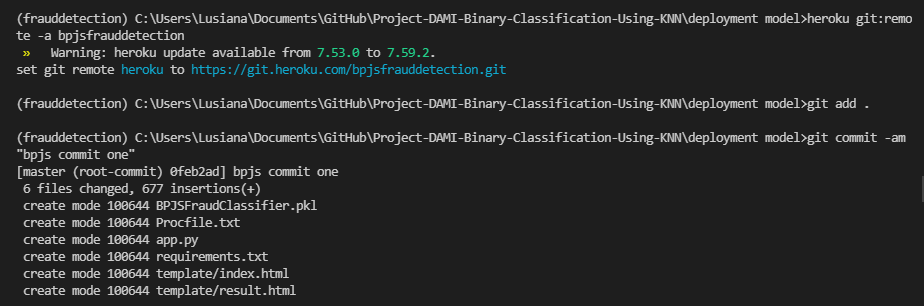
1. Masukkan name aplikasi sesuai keinginan, lalu klik tombol create. Maka akan diarahkan ke halaman awal bagian deploy.
2. Login pada Heroku CLI dengan mengetikkan perintah heroku login dan tekan enter. Kemudian tekan sembarang kunci/tombol keyboard untuk membuka browser. Maka akan diarahkan untuk login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

Gambar 53 Login Heroku

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

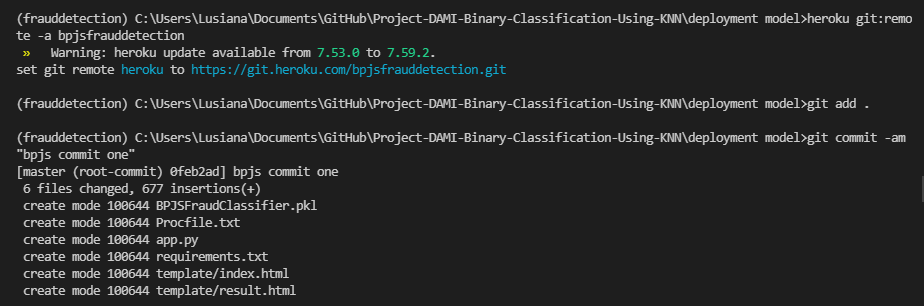
Gambar 54 Upload file dalam Heroku

* Git ini
* Heroku git:remote -a bpjsfrauddetection



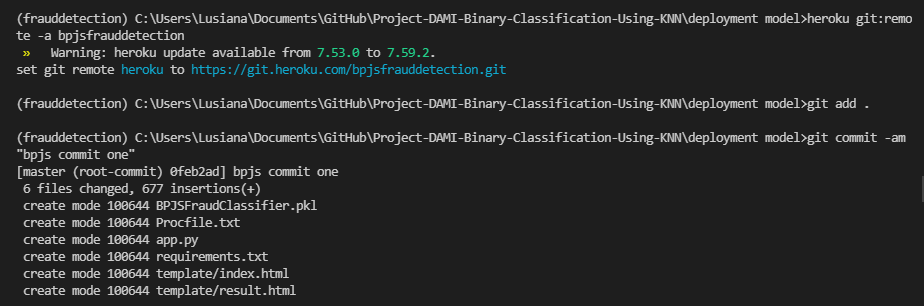
Gambar 55 Heroku git:remote

* Git add .



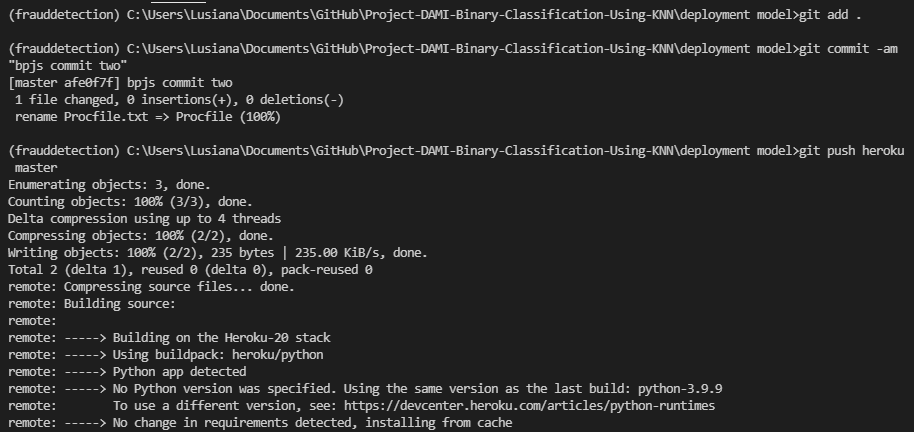
Gambar 56 Git add

* Git commit -am “bpjs commit one”



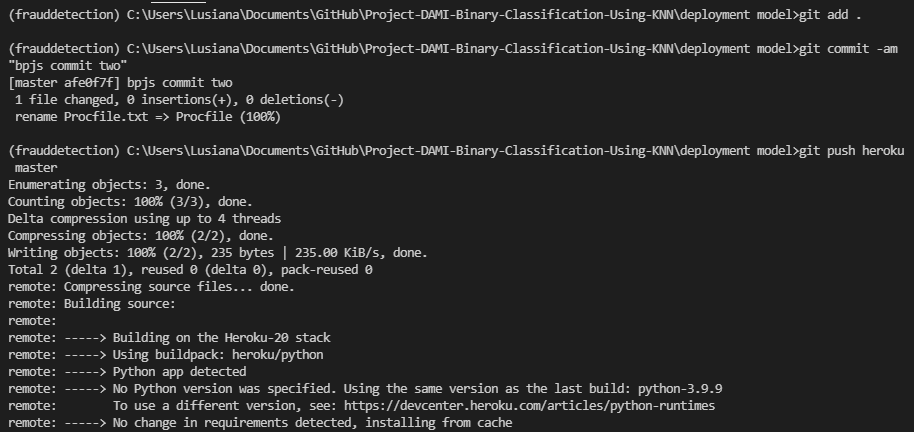
Gambar 57 Git commit one

* Git commit -am “bpjs commit two”

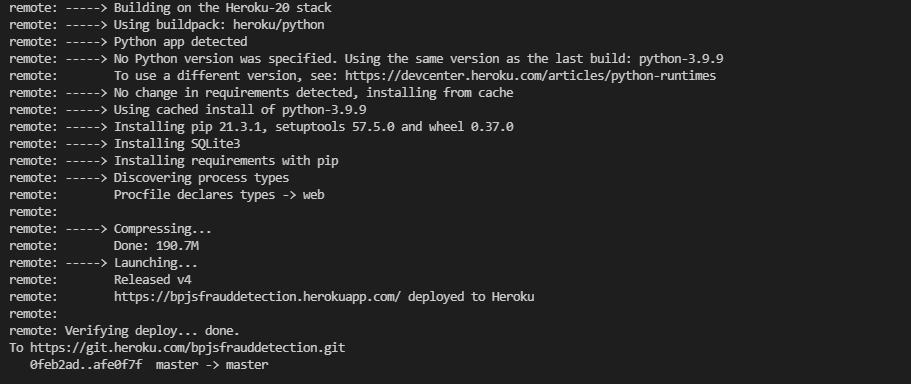


Gambar 58 Git commit two

* Git push Heroku master

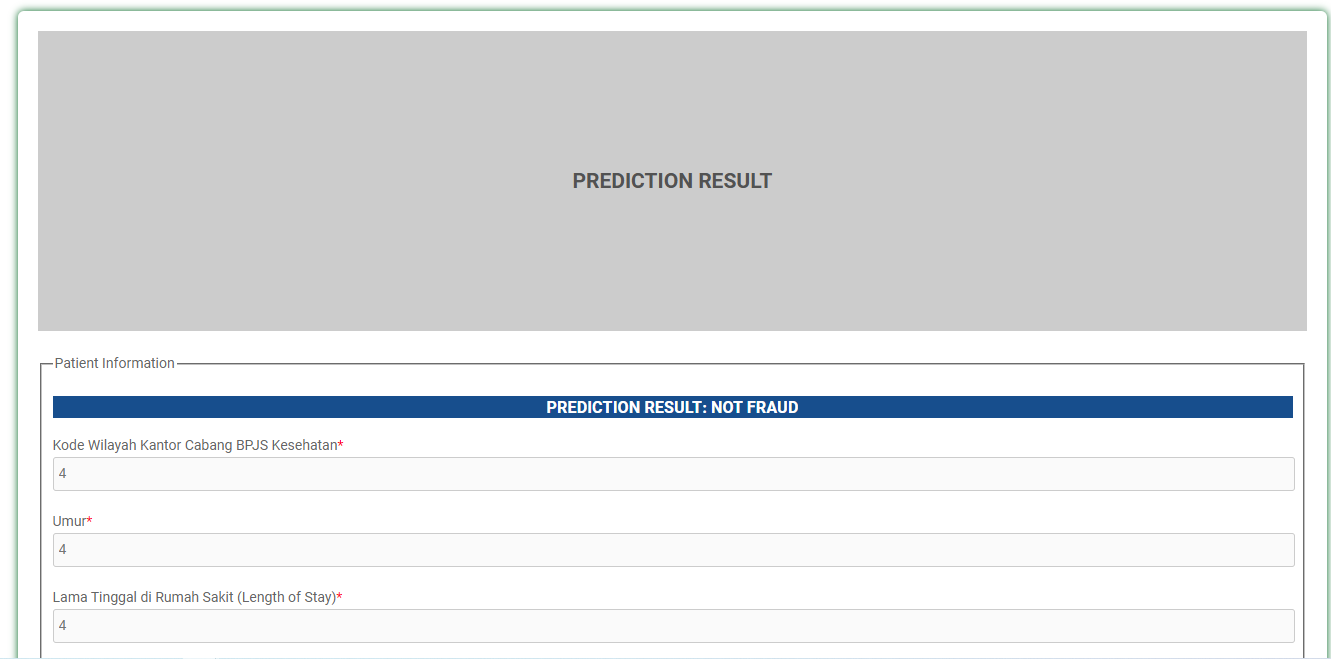


Gambar 59 Git push

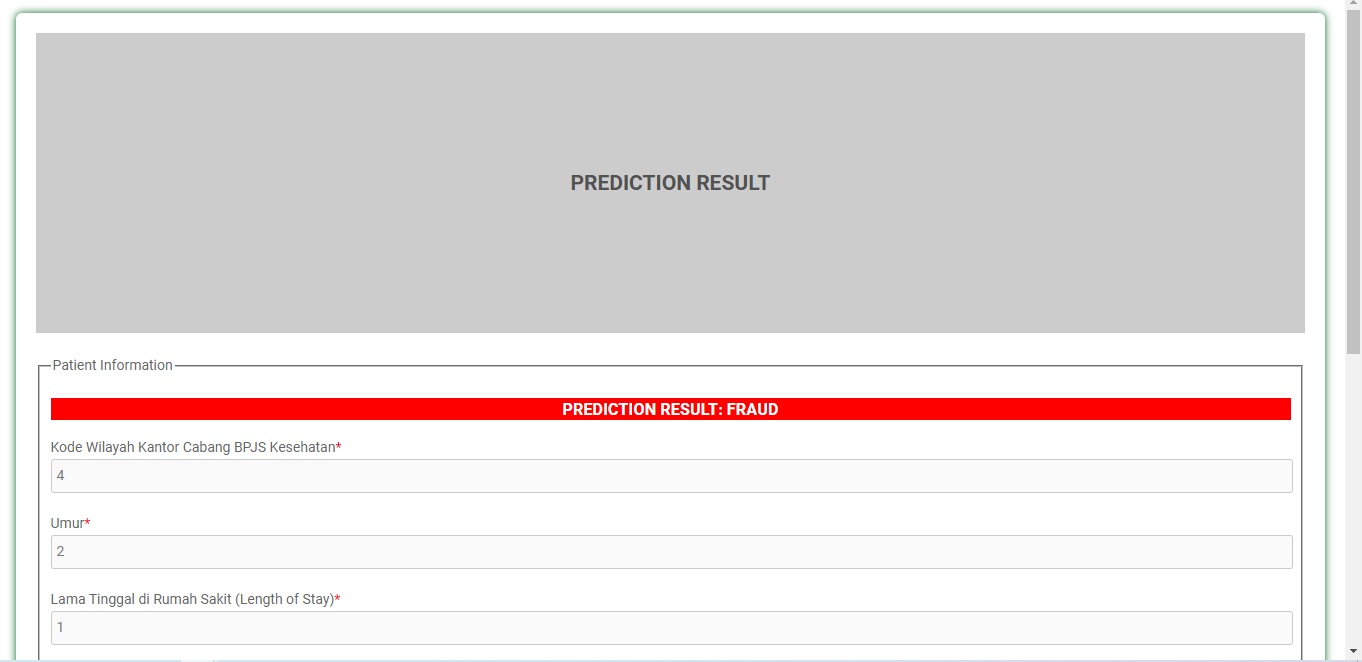


Gambar 60 Deployment berhasil

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**.

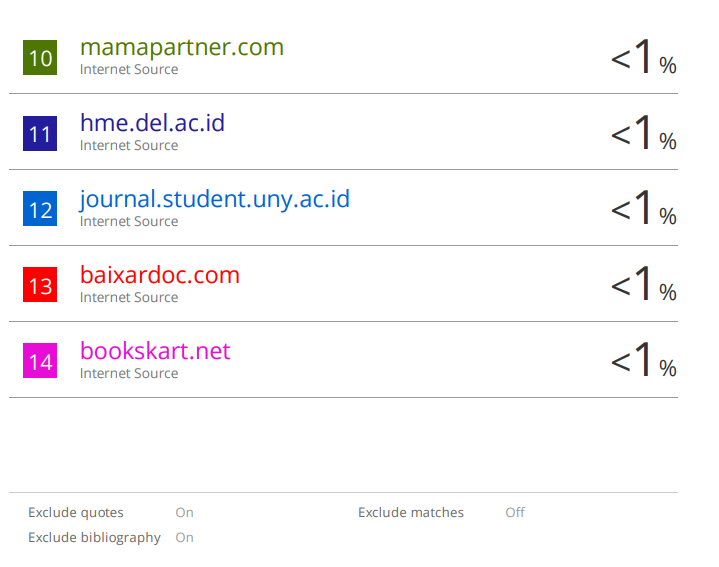


Gambar 61 Input not fraud

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

Gambar 62 Input fraud

# Lampiran



Gambar 63 Hasil turnitin laporan