

**Laporan Tugas Besar Penambangan Data**  
**IF-41-GAB 01**



**Disusun Oleh :**

1301184219	Sya Raihan Heggi
------------	------------------

**S1 INFORMATIKA**  
**FAKULTAS INFORMATIKA**  
**UNIVERSITAS TELKOM**  
**BANDUNG**  
**2021**

## 1. Latar Belakang Masalah

*Data mining* adalah sebuah studi ilmu teknologi yang mempelajari data dalam jumlah yang besar, mulai dari mengumpulkan (collecting) data, membersihkan (cleaning) data, processing data, menganalisis data, sampai dengan mengekstraksi knowledge, di dalam data mining sendiri terdapat berbagai permasalahan yang dapat diselesaikan oleh algoritma data mining ini sendiri task-task tersebut adalah *Summarization*, *Clustering*, *Classification*, *Regression*, dan *Association* [1], dalam tugas besar kali ini diberikan sebuah dataset yang memiliki judul ***Fraud Detection at Self-Checkout in Retail*** dimana kasus yang dihadapi dikarenakan pertumbuhan dan penggunaan dari mesin pembayaran otomatis yang ada di retail, sebenarnya sistem yang dibangun cukup baik karena dapat mengurangi penggunaan atau antrian namun memiliki celah untuk dilakukan kejahatan karena tidak ada manusia yang mengawasi kecuali pembeli itu sendiri.

Kegiatan ini memanglah menguntungkan bagi pihak pembeli dimana tidak perlu melakukan antrian panjang kemudian juga menguntungkan untuk penjual karena tingkat penjualan bisa semakin banyak karena antrian lancar, namun hal ini juga menyebabkan pemanfaatan celah sehingga dapat merugikan pihak penjual, dikarenakan oleh itu maka perlu dibuatkan sebuah sistem yang dapat melakukan prediksi/klasifikasi terhadap kegiatan yang dilakukan pembeli di mesin pembayaran otomatis dan dikategorikan menjadi penipuan atau tidak, untuk melakukan klasifikasi ini akan dilakukan menggunakan dua buah algoritma yaitu Decision Tree, dan ANN.

## 2. Tujuan

Tujuan yang ingin dicapai dari tugas besar ini adalah.

- Melakukan proses klasifikasi data dengan menggunakan algoritma Decision Tree dan ANN..
- Membandingkan performansi algoritma yang digunakan dalam melakukan klasifikasi penipuan atau tidak
- Menentukan fitur apa saja yang optimum dalam membangun sebuah sistem
- Mendapatkan akurasi terbaik dari algoritma yang dipilih.

## 3. Pra-processing data.

Seperti yang kita ketahui pada tahapan preprocessing ini akan dilakukan proses eksplorasi dan penanganan dari nilai yang kita akan lakukan hal-hal yang ingin dilakukan adalah pertama kita akan melakukan *data exploration*, *data cleansing*, dan *data transformation* dimana nantinya akan meningkatkan kualitas dari dataset tersebut.

### 3.1. Data Exploration

Pada tahapan ini akan di cek apakah dataset ini memiliki kekurangan seperti *missing value*, *attribute number*, *sum*, *duplicate value*, dan lain-lain, dataset ini sendiri ketika diunduh akan berisi dua buah file yaitu *train.csv* dan *test.csv*, data training terdiri dari 1879 baris data dan jumlah atribut 10. Untuk data testing terdiri dari 498121 baris data dengan jumlah

atribut 9, mengapa pada dataset test ada 9 karena pada dataset ini belum ada pelabelan dan nantinya digunakan untuk pembangunan sistem.

The screenshot shows a Jupyter Notebook interface with a dark theme. At the top, the text 'SHAPE DATASET' is displayed in white. Below it, two code cells are visible. The first cell, labeled '[9]', contains the code 'df\_train.shape' and shows the output '(1879, 10)'. The second cell, labeled '[12]', contains the code 'df\_test.shape' and shows the output '(498121, 9)'. Each code cell has a green play button icon to its left.

```
[9] df_train.shape
(1879, 10)

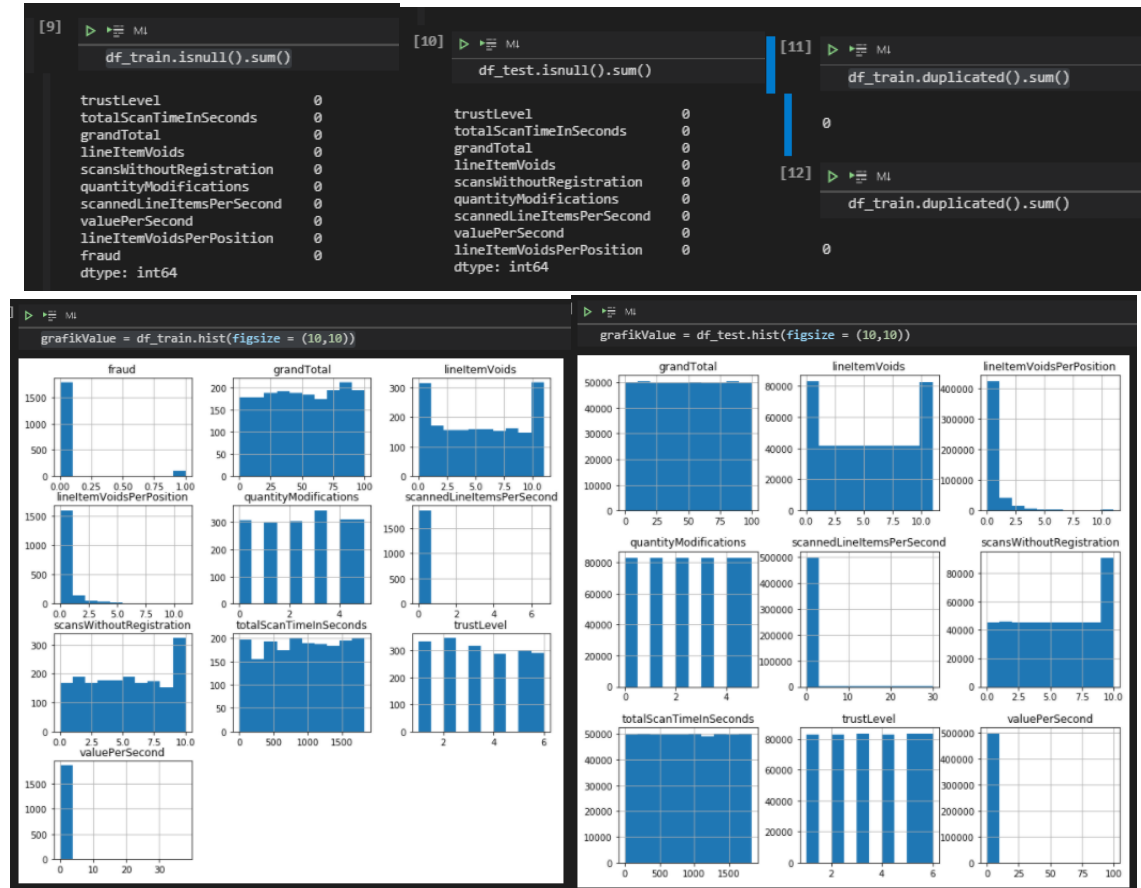
[12] df_test.shape
(498121, 9)
```

Berikut merupakan penjelasan karakteristik dari dataset yang digunakan dalam tugas besar ini.

Nama Atribut	Deskripsi	Range Nilai
trustLevel	Tingkat kepercayaan individu pelanggan.  <b>*6 : Nilai Tertinggi</b>	{ 1,2,3,4,5,6}
totalScanTimeInSeconds	Total waktu dalam detik antara produk pertama dan terakhir dipindai.	Bilangan bulat positif.
grandTotal	Total keseluruhan produk yang dipindai.	Angka desimal positif dengan maksimum dua tempat desimal.
lineItemVoids	Jumlah pemindaian yang dibatalkan	Bilangan bulat positif.
scansWithoutRegistration	Jumlah upaya untuk mengaktifkan pemindai tanpa benar-benar memindai apa pun.	Bilangan bulat positif atau 0.
quantityModification	Jumlah kuantitas yang dimodifikasi untuk salah satu produk yang dipindai.	Bilangan bulat positif atau 0.
scannedLineItemsPerSecond	Jumlah rata-rata produk yang dipindai per detik.	Angka desimal positif.

valuePerSecond	Nilai total rata-rata produk yang dipindai per detik.	Angka desimal positif.
lineItemVoidsPerPosition	Jumlah rata-rata kekosongan item per jumlah total semua produk yang dipindai dan tidak dibatalkan.	Angka desimal positif.
fraud	Klasifikasi sebagai fraud/penipuan (1) atau bukan fraud/penipuan (0)	{0,1}

Pada Dataset ini tidak ditemukan nilai-nilai berikut ini *Missing Value*, *Duplicate Value*, dan *Wrong Value*, inovasi yang digunakan (menggunakan visualisasi untuk melihat sebaran data sehingga diketahui jika ada nilai yang salah, karena bila menggunakan value\_counts saja susah terlihat karena ada data yang jumlahnya banyak).



Kemudian untuk dataset ini sendiri memiliki fitur-fitur yang memiliki jenis tipe data `[int64,float64]`, dengan perincian seperti berikut ini.

```
INFORMASI DATASET

[162] In [162]: df_train.dtypes

trustlevel          int64
totalScanTimeInSeconds  int64
grandTotal          float64
lineItemVoids       int64
scansWithoutRegistration  int64
quantityModifications  int64
scannedLineItemsPerSecond float64
valuePerSecond       float64
lineItemVoidsPerPosition float64
fraud               int64
dtype: object

[163] In [163]: df_test.dtypes

trustlevel          int64
totalScanTimeInSeconds  int64
grandTotal          float64
lineItemVoids       int64
scansWithoutRegistration  int64
quantityModifications  int64
scannedLineItemsPerSecond float64
valuePerSecond       float64
lineItemVoidsPerPosition float64
dtype: object
```

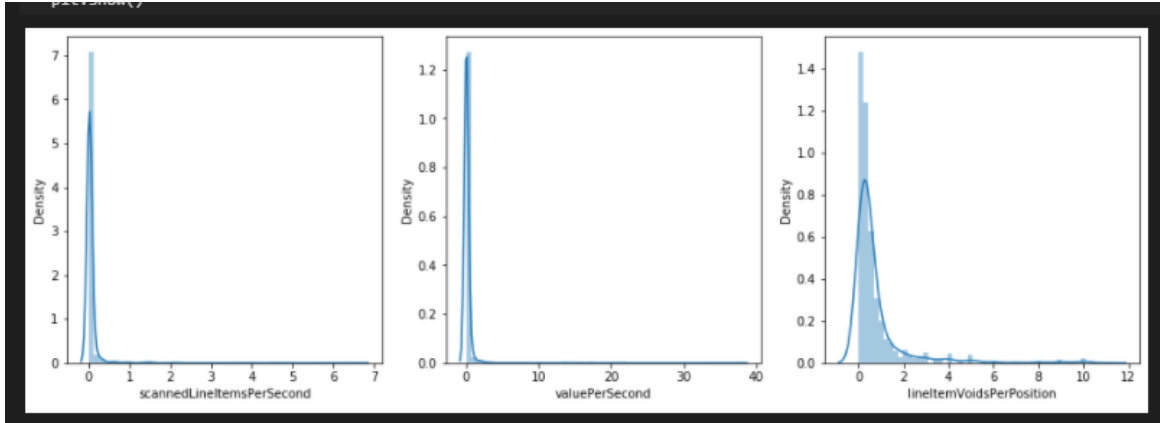
Selanjutnya kita akan mengecek terakhir kali yang berkaitan dengan kualitas dari dataset ini yaitu outlier, pada dataset ini ditemukan outlier pada data yang ditemukan pada tiga variabel terakhir oleh karena itu akan dilakukan pembersihan dengan melakukan *drop data*



Sehingga dari pengujian ini bisa disimpulkan bahwa **data ini berkualitas** dan siap untuk digunakan mungkin yang perlu dilakukan adalah membersihkan outlier yang ada dan mungkin bisa menjadi analisis apakah akan mempengaruhi nantinya.

### 3.2. Data Cleansing

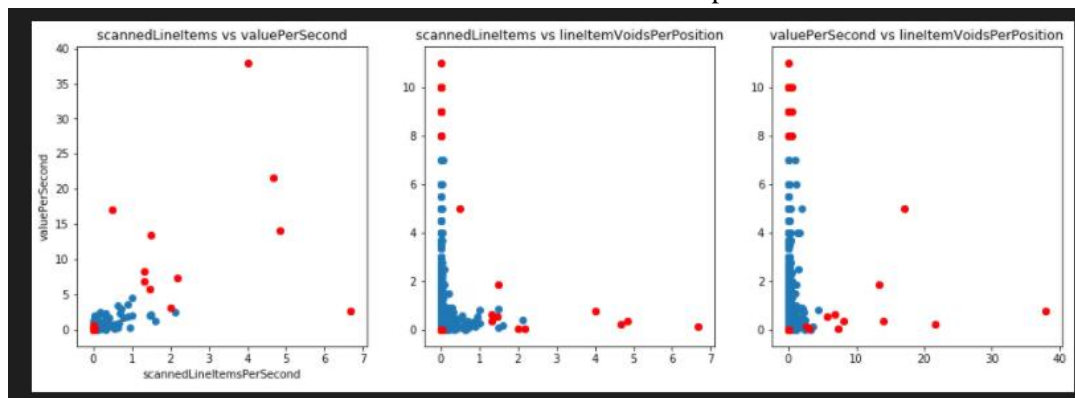
Setelah melakukan Data Exploration kita mendapatkan ada outlier yang harus ditangani untuk penanganan dari outlier sendiri, sebelumnya untuk meyakinkan apakah itu benar outlier kita bisa menggunakan cara melihat distribusinya, dan hasilnya seperti ini.



Dari gambar yang terlihat memang benar ada outlier karena itu akan dilakukan penanganan dengan menggunakan drop data yang dilakukan dengan menghapus outlier tersebut hal ini diperkuat dengan nilai skewnya sehingga mengincar untuk membuat data menjadi distribusi normal.

```
[66] ▶ ML  
df.skew()  
  
scannedLineItemsPerSecond    15.078211  
valuePerSecond                20.771462  
lineItemVoidsPerPosition      4.315624  
dtype: float64
```

Selain itu dilakukan inovasi juga tidak hanya menggunakan boxplot namun untuk memastikan kembali apakah outlier memang akan terjadi bila menggunakan tiga fitur tersebut maka digunakan SVM untuk mendeteksi anomali outlier dan divisualisasikan seperti berikut ini.



Sehingga perlu dibersihkan maka akan dilakukan pembersihan dengan menggunakan metode melakukan drop data yang berada di luar area *IQR* (Interquartile Range), dengan harapan akan mengembalikan kembali distribusinya.

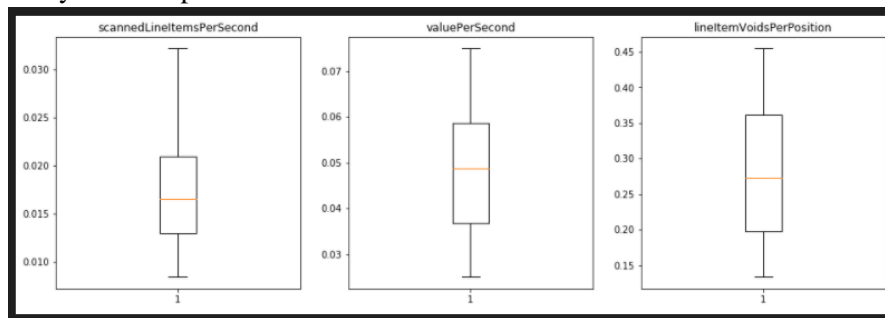
### 3.2.1 Outlier Cleaning

```
Quartile_Bawah = np.percentile(df_train['lineItemVoidsPerPosition'],25)
Quartile_Atas = np.percentile(df_train['lineItemVoidsPerPosition'],75)
IQR = Quartile_Atas-Quartile_Bawah
Upper = Quartile_Atas+1.5*IQR
Lower = Quartile_Bawah+1.5*IQR
Upper,Lower

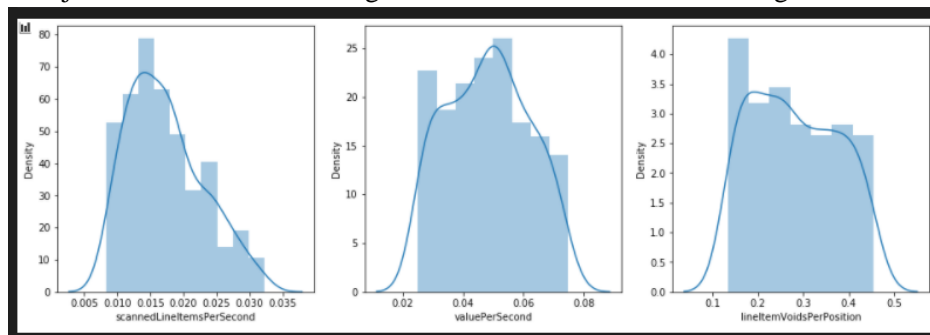
(0.9363636363636381, 0.6151515151515161)

warnings.filterwarnings('ignore')
indexUpper= df_train[(df_train['lineItemVoidsPerPosition'] > df_train['lineItemVoidsPerPosition'].quantile(0.75))]
indexLower= df_train[(df_train['lineItemVoidsPerPosition'] < df_train['lineItemVoidsPerPosition'].quantile(0.25))]
df_train.drop(indexUpper, inplace=True)
df_train.drop(indexLower, inplace=True)
plt.boxplot(df_train['lineItemVoidsPerPosition'])
```

Seperti yang sudah dijelaskan sebelumnya proses pembersihan dilakukan dengan menghitung wilayah *IQR*, *Lower*, *Upper* sehingga nantinya akan di drop data yang berada diluar wilayah *IQR* sehingga hasilnya akan seperti berikut ini.



Dan seperti tujuan awal mari kita cek lagi distribusi dan *skewness* dari ketiga fitur ini.

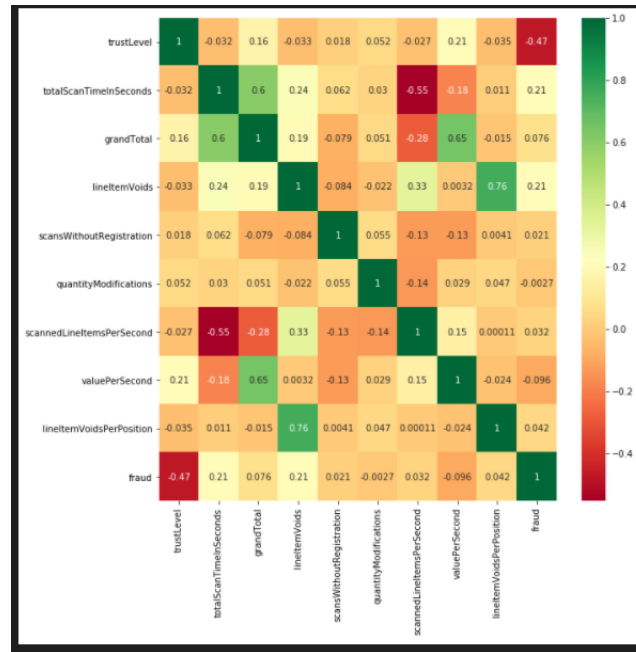


Dari plot distribusi setelah pembersihan bisa dilihat persebaran data sudah mulai mengarah kearah normal walaupun masih ada skewness, untuk catatan distribusi normal akan bernilai 0, kemudian cek skewnessnya.

```
trustLevel -0.055284
totalScanTimeInSeconds -0.457393
grandTotal -0.060956
lineItemVoids 0.218349
scansWithoutRegistration 0.157187
quantityModifications 0.039724
scannedLineItemsPerSecond 0.584444
valuePerSecond 0.051412
lineItemVoidsPerPosition 0.166074
fraud 2.536249
dtype: float64
```

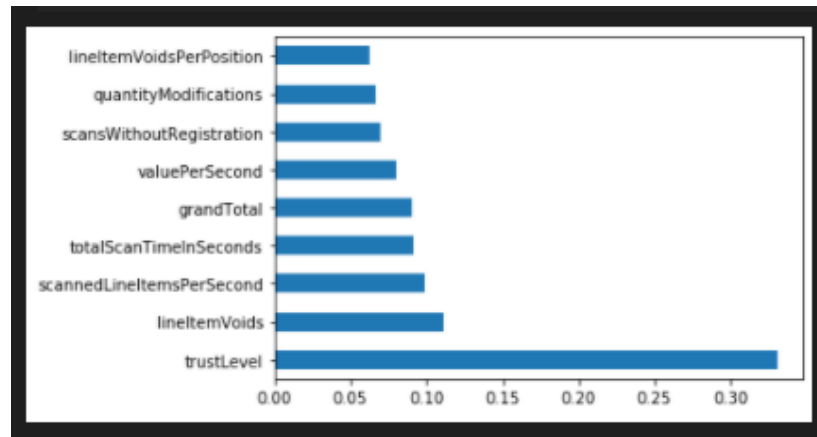
Dari hasil pembersihan sudah mendekati 0.0 dan sudah cukup memenuhi standar yang diinginkan.

### 3.2.2 Correlation Check



Dari perhitungan korelasi dapat dilihat pengambilan feature yang dapat digunakan **lineItemVoids**, **totalScanTimeInSeconds**, **grandTotal**, **lineItemVoidsPerPosition**, namun untuk inovasi yang dihadirkan untuk memverifikasi untuk melakukan pemilihan features oleh karena itu akan dilakukan pengecekan kembali menggunakan metode *feature importance*, dan *univariate selection*.

### 3.2.3 Feature Importance



Dari perhitungan korelasi dapat dilihat pengambilan feature yang dapat digunakan **trustLevel**, **lineItemVoids**, **totalScanTimeInSeconds**, **grandTotal**, **scannedLineItemsPerSecond**.



### 3.2.4 Univariate Selection

	Feature	Score
1	totalScanTimeInSeconds	921.331002
0	trustLevel	41.718712
3	lineItemVoids	11.535654
2	grandTotal	10.224953
4	scansWithoutRegistration	0.241275
8	lineItemVoidsPerPosition	0.014016
7	valuePerSecond	0.008598
5	quantityModifications	0.002135
6	scannedLineItemsPerSecond	0.000452

Dari perhitungan korelasi dapat dilihat pengambilan feature yang dapat digunakan **trustLevel**, **lineItemVoids**, **totalScanTimeInSeconds**, **grandTotal**.

### 3.3. Data Transformation

Karena tipe data sudah sesuai dan sifatnya **int64** dan **float64** maka tidak perlu dilakukan perubahan namun untuk transformation yang dilakukan adalah melakukan scaling agar data yang digunakan dapat scaling untuk menentukan rangenya, untuk scaling dilakukan menggunakan Min-Max Scaler yang sudah disediakan oleh sklearn.

```
[1514] > #> MI
from sklearn.preprocessing import MinMaxScaler

[1515] > #> MI
normalize = MinMaxScaler()
d = preprocessing.normalize(df_train[df_train.columns[9]], axis=0)
scaled_df = pd.DataFrame(d, columns=['trustLevel', 'scannedLineItemsPerSecond', 'totalScanTimeInSeconds', 'quantityModifications', 'scansWithoutRegistration', 'lineItemVoids', 'grandTotal', 'valuePerSecond', 'lineItemVoidsPerPosition'])
scaled_df.head()
```

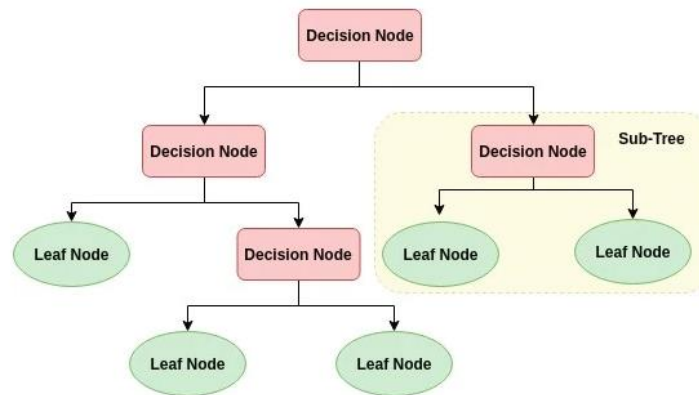
	trustLevel	scannedLineItemsPerSecond	totalScanTimeInSeconds	quantityModifications	scansWithoutRegistration	lineItemVoids	grandTotal	valuePerSecond	lineItemVoidsPerPosition
0	0.081868	0.851421	0.854872	0.070456	0.000000	0.06396	0.097699	0.066916	0.852572
1	0.049121	0.873969	0.862355	0.030195	0.114512	0.10660	0.030440	0.052869	0.050261
2	0.008242	0.887376	0.892680	0.009521	0.045805	0.08528	0.057496	0.066457	0.860883
3	0.008242	0.866933	0.865645	0.070456	0.000000	0.04204	0.099668	0.087711	0.056467
4	0.081868	0.874798	0.884996	0.040261	0.022902	0.08528	0.037808	0.071266	0.054458

Selanjutnya dilakukan *feature selection* dengan mempertimbangkan nilai korelasi, *univariate selection*, dan *feature selection*, karena dengan mengecek beberapa metode secara logika kalau muncul beberapa kali maka kemungkinan memang fitur tersebut memang terkait, sehingga dipilih **trustLevel**, **lineItemVoids**, **totalScanTimeInSeconds**, **grandTotal**.

Akhir dari tahapan ini adalah data disimpan agar dapat digunakan pada proses klasifikasi yang akan digunakan untuk memprediksi dan klasifikasi.

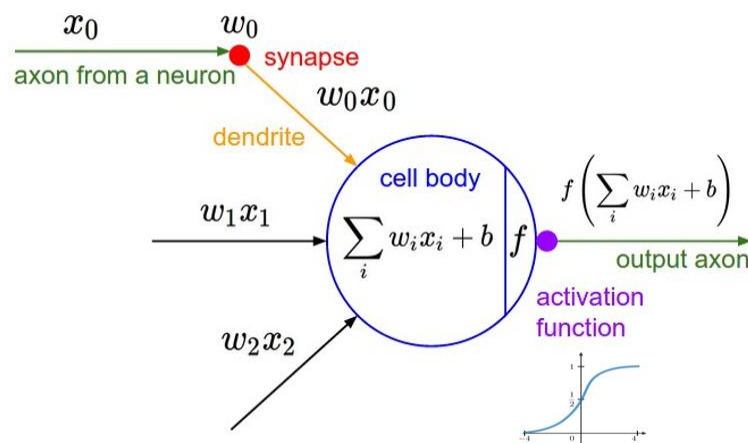
## 4. Analisis Pemilihan Algoritma.

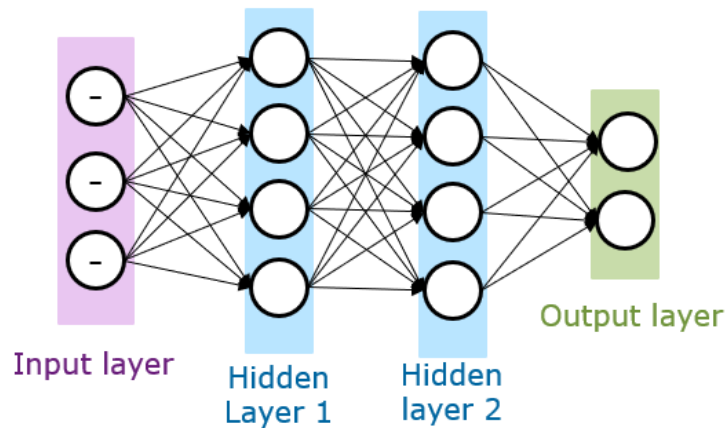
Task yang ada dalam data mining yaitu prediksi. Task ini menggunakan atribut untuk mengetahui atau memprediksi nilai yang sebelumnya belum diketahui (unknown) dari atribut lain. Contoh dari task ini yaitu klasifikasi, regresi, dan deviation detection, pada tugas besar ini digunakan dua buah algoritma untuk melakukan klasifikasi ini yaitu ANN (JST/Jaringan Syaraf Tiruan) dan Decision Tree, pada tugas besar memiliki tujuan untuk mempertimbangkan performansi keduanya sehingga dipilih dua buah algoritma yang menggunakan metode yang berbeda sehingga diketahui perbedaannya.



*Decision tree* adalah salah satu metode klasifikasi yang paling populer, karena mudah untuk diinterpretasi oleh manusia. Decision tree adalah model prediksi dengan membangun pohon keputusan, yang nantinya akan membentuk aturan yang digunakan, dan pembangunan cabang-cabang ini biasanya dilakukan dengan kriteria tertentu, dan kemudian akan menghasilkan keputusan yang kompleks menjadi lebih simpel dan dapat diimplementasikan, keuntungan menggunakan algoritma ini bisa menemukan decision yang mungkin tidak dapat ditemukan oleh Jaringan Syaraf Tiruan, selain itu untuk membuat model dengan metode ini bisa dikatakan cukup cepat dikarenakan memiliki kompleksitas algoritma yang tidak lebih rumit dari Jaringan Syaraf Tiruan dan terakhir algoritma ini dapat menangani data berdimensi tinggi dengan cukup baik [2].

*ANN (Artificial Neural Network)* adalah sebuah model Jaringan Syaraf Tiruan yang terinspirasi dari cara kerja otak manusia sebagai contohnya adalah Neuron pada otak, Neuron merupakan bagian yang menyelesaikan sebuah permasalahan yang ada (berpikir) [3], selain itu Neuron mempunyai tiga komponen penting yang membuatnya dapat memberikan respons yaitu *dendrit*, *soma* (badan), *axon* [3], hal ini dimodelkan kembali oleh ANN dengan membangun sebuah neuron atau lebih dikenal perceptron untuk melakukan pembelajaran dan nantinya berguna untuk penyelesaian masalah, untuk gambaran bentuk perceptron akan seperti berikut ini.





Seperti layaknya Jaringan Saraf pada implementasinya ANN biasanya membuat menjadi 3 layer utama yaitu *Input Layer*, *Hidden Layer*, *Output* untuk kelebihan ANN sendiri dapat menggunakan supervised atau unsupervised learning, namun memiliki kekurangan yaitu kompleksitas algoritma lebih tinggi jika dibandingkan dengan *Decision Tree* namun ANN sendiri cocok untuk digunakan pada permasalahan *Bankruptcy prediction*, *Speech recognition*, *Product inspection*, *Fault detection* dengan tipe data Tabular, Gambar, dan Text.

Didasarkan dari penjelasan dan latar belakang algoritma ini lah yang membuat penulis ingin menggunakan dan membandingkan dan bagaimana performansinya terhadap klasifikasi kasus penipuan atau tidak

## 5. Hasil Percobaan.

Pada tugas besar ini akan dilakukan percobaan dengan menggunakan dua jenis algoritma klasifikasi yaitu *Decision Tree* dan *ANN*, untuk hasil percobaan akan berupa data biner 0/1 dimana 0 berarti **bukan penipuan** dan 1 berarti **penipuan**, dan kemudian hasilnya akan dievaluasi menggunakan parameter *Precision*, *Recall*, *F1-Score*, dan *Accuracy*, kemudian tahapan yang dilakukan adalah **Pembuatan Model** dan **Implementasi Model**, sebelum masuk ke tahapan akan dijelaskan apa saja parameter evaluasinya itu.

- **Precision**

Presisi adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Persamaan presisi adalah sebagai berikut.

$$Precision = \frac{TP}{FP+TP}$$

- **Recall**

Recall adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Persamaan recall adalah sebagai berikut.

$$Recall = \frac{TP}{FN + TP}$$

- **Accuracy**

Akurasi didefinisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai aktual. Semakin besar nilai akurasi, maka performansi sistem klasifikasi semakin baik. Persamaan akurasi adalah sebagai berikut.

$$Accuracy = \frac{TP + TN}{TN + FN + TP + FP}$$

- **F1-Score**

F1 Score adalah perbandingan rata-rata antara precision dan recall. Score ini akan memperhitungkan false positive dan false negative. Persamaan F1 Score adalah sebagai berikut.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Setelah mengetahui metode evaluasi yang akan digunakan maka bisa dilanjutkan ke tahapan membuat model.

## 5.1. Pembuatan Model

Model dibuat menggunakan data yang sudah dilakukan *preprocessing* sebelumnya sehingga pada tugas besar ini ada dua buah Model yang dibangun berdasarkan algoritma yang digunakan.

### 5.1.1 Decision Tree

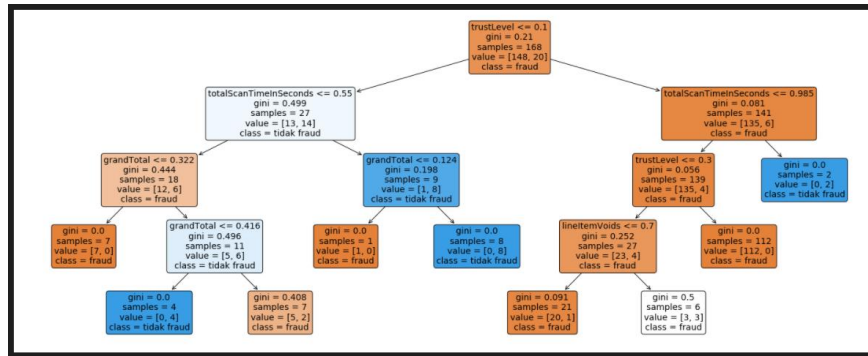
Model Dibangun dengan menggunakan parameter `max_depth = 5` dengan pembagian dataset train dengan validasi sebesar train 70 dan validasi 30% dan `random_state = 5`, kemudian dilakukan inovasi untuk memvisualisasikan bagaimana tree yang dibangun.

```
▶ ML
X_train,X_test,y_train,y_test = train_test_split(df_x,df_y,test_size=0.30,random_state=5)
{}

BUAT KLASIFIER

▶ ML
classifier = DecisionTreeClassifier(max_depth=4)
classifier.fit(X_train,y_train)

DecisionTreeClassifier(max_depth=4)
```



Kemudian didapati hasil dari pembangunan model adalah seperti berikut ini.

DATA TRAINING

```
[91] > M1
```

```
print(confusion_matrix(y_train, y_pred_train))
print(classification_report(y_train, y_pred_train))
print("Accuracy:", accuracy_score(y_train, y_pred_train)*100)
```

[[148 0]

[ 6 14]]

	precision	recall	f1-score	support
0	0.96	1.00	0.98	148
1	1.00	0.70	0.82	20
accuracy			0.96	168
macro avg	0.98	0.85	0.90	168
weighted avg	0.97	0.96	0.96	168

Accuracy: 96.42857142857143

DATA VALIDASI

```
> M1
```

```
print(confusion_matrix(y_test, y_pred_test))
print(classification_report(y_test, y_pred_test))
print("Accuracy:", accuracy_score(y_test, y_pred_test)*100)
```

[[63 3]

[ 3 3]]

	precision	recall	f1-score	support
0	0.95	0.95	0.95	66
1	0.50	0.50	0.50	6
accuracy			0.92	72
macro avg	0.73	0.73	0.73	72
weighted avg	0.92	0.92	0.92	72

Accuracy: 91.66666666666666

Ketika pembangunan ketika diuji ke data training sendiri mendapatkan akurasi 96% dan ketika diuji ke validasi 91%, dan nilai *precision*, *recall*, dan *f1-score* pada **data training** memiliki hasil 0 = (96% ,100% , 98%) dan 1 = (100% , 70% , 82%) sementara pada **data validasi** memiliki hasil 0 = (95%, 95% , 95%) dan 1 = (100% , 70% , 82%), bila diperhatikan ada penurunan nilai akurasi namun masih di jangka 5% maka diputuskan oleh penulis untuk tetap menggunakannya.

### 5.1.2 ANN

Model yang dibangun menggunakan algoritma ANN memiliki parameter untuk pembagian data training dan data validasi masih sama menggunakan training 70% dan validasi 30% dan `random_state = 5`, dibangun dengan satu layer input, empat hidden layer dengan parameter (**hidden layer pertama memiliki 2000 perceptron dengan fungsi aktivasi relu**, **hidden layer kedua memiliki 500 perceptron fungsi aktivasi sigmoid**, **hidden layer ketiga memiliki 500 perceptron dengan fungsi aktivasi relu**, dan **terakhir memiliki 2 perceptron dengan fungsi aktivasi softmax**), dan kemudian layer output, dan kemudian di compile dengan **optimizer = "adam"**, **loss = "sparse\_categorical\_crossentropy"**, dan **metrics = "accuracy"** dan kemudian dilakukan latihan sebanyak 10 epoch.

```

> MI
x_train,x_test,y_train,y_test = train_test_split(df_x,df_y,test_size=0.30,random_state=5)

ANN MODEL


> MI
model = models.Sequential([
    layers.Dense(2000, input_dim=4, activation='relu'),
    layers.Dense(500, activation='sigmoid'),
    layers.Dense(500, activation='relu'),
    layers.Dense(2, activation='softmax')
])
model.compile(optimizer="adam",loss="sparse_categorical_crossentropy",metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10)

Epoch 1/10
6/6 [=====] - 1s 8ms/step - loss: 0.7247 - accuracy: 0.6128
Epoch 2/10
6/6 [=====] - 0s 9ms/step - loss: 0.3814 - accuracy: 0.9070
Epoch 3/10
6/6 [=====] - 0s 8ms/step - loss: 0.4658 - accuracy: 0.9125
Epoch 4/10
6/6 [=====] - 0s 8ms/step - loss: 0.4363 - accuracy: 0.8770
Epoch 5/10
6/6 [=====] - 0s 11ms/step - loss: 0.3715 - accuracy: 0.8694
Epoch 6/10
6/6 [=====] - 0s 11ms/step - loss: 0.2884 - accuracy: 0.8895
Epoch 7/10
6/6 [=====] - 0s 9ms/step - loss: 0.2267 - accuracy: 0.9361
Epoch 8/10
6/6 [=====] - 0s 8ms/step - loss: 0.2170 - accuracy: 0.8990
Epoch 9/10
6/6 [=====] - 0s 8ms/step - loss: 0.2229 - accuracy: 0.8941
Epoch 10/10
6/6 [=====] - 0s 9ms/step - loss: 0.1548 - accuracy: 0.9220
<tensorflow.python.keras.callbacks.History at 0x1a92ce83a88>

```

Dan dari model ini dihasilkan hasil evaluasi model seperti berikut ini.

DATA TRAINING

▶  ML


```
import warnings
warnings.filterwarnings('ignore')
y_pred_train = [np.argmax(element) for element in y_pred_train]
print(confusion_matrix(y_train, y_pred_train))
print(classification_report(y_train, y_pred_train))
print("Accuracy:", accuracy_score(y_train, y_pred_train)*100)
```

```
[[148  0]
 [ 14  6]]
```

	precision	recall	f1-score	support
0	0.91	1.00	0.95	148
1	1.00	0.30	0.46	20
accuracy			0.92	168
macro avg	0.96	0.65	0.71	168
weighted avg	0.92	0.92	0.90	168

Accuracy: 91.66666666666666

DATA TESTING

▶  ML

```
warnings.filterwarnings('ignore')
y_pred_test = [np.argmax(element) for element in y_pred_test]
print(confusion_matrix(y_test, y_pred_test))
print(classification_report(y_test, y_pred_test))
print("Accuracy:", accuracy_score(y_test, y_pred_test)*100)
```

```
[[65  1]
 [ 5  1]]
```

	precision	recall	f1-score	support
0	0.93	0.98	0.96	66
1	0.50	0.17	0.25	6
accuracy			0.92	72
macro avg	0.71	0.58	0.60	72
weighted avg	0.89	0.92	0.90	72

Accuracy: 91.66666666666666

Dari model yang dibangun dengan ANN ketika diuji ke data training dan validasi sama-sama memiliki nilai akurasi 91,6% sehingga bisa dikatakan model yang dibuat cukup optimum dan untuk nilai *precision*, *recall*, dan *f1-score* pada **data training memiliki hasil 0 = (91%, 100% , 95%) dan 1 = (100% , 30%, 46%)** sementara pada **data validasi memiliki hasil 0 = (93%, 98%, 96%) dan 1 = (50%, 17% ,25%)**, sehingga dari pembangunan model sampai saat ini masih unggul menggunakan ANN.

## 5.2. Implementasi Model

Model yang sudah dibangun tadi kemudian akan digunakan untuk memprediksi data test dan kemudian akan dievaluasi dengan real class yang sudah disediakan, inovasi yang dihadirkan selain menampilkan data tetapi juga membuat visualisasi hasil.

### 5.2.1 Decision Tree

Implementasi menggunakan Decision Tree menghasilkan hasil berikut ini.

```
print(confusion_matrix(df_realclass, y_pred_testing))
print(classification_report(df_realclass, y_pred_testing))
print("Accuracy:", accuracy_score(df_realclass, y_pred_testing)*100)

[[443695  30699]
 [ 13067 10660]]

              precision    recall  f1-score   support

      0       0.97       0.94       0.95     474394
      1       0.26       0.45       0.33      23727

   accuracy          0.91     498121
  macro avg       0.61     0.69     0.64     498121
 weighted avg       0.94     0.91     0.92     498121

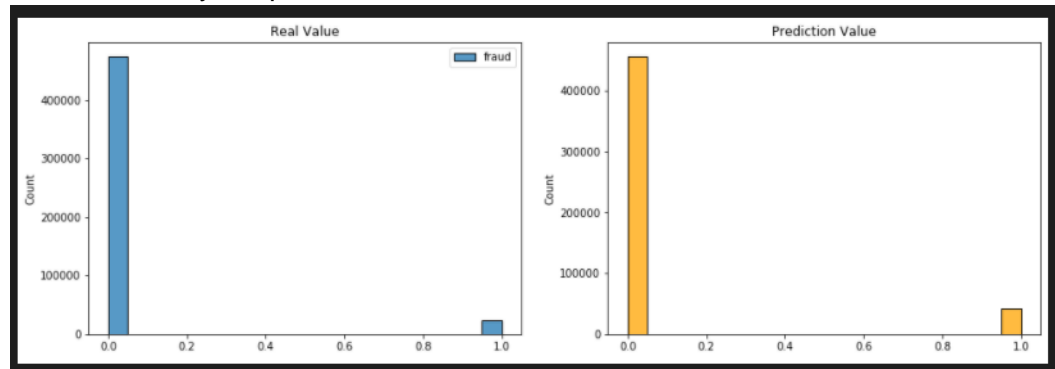
Accuracy: 91.21378139046537
```

Hasil klasifikasi yang dilakukan memiliki akurasi 91% dan bisa dibilang cukup bagus walaupun masih mengalami perbedaan 5% jika dibandingkan dengan akurasi pembuatan model di 96%, dan pada pengujian ini memiliki nilai confusion matrix yang membentuk nilai *precision*, *recall*, *f1-score* **0 = ( 97% , 94%, 95%) dan 1 = (26%, 45%, 33%)**.

Kategori	Kelas Aktual		
Kelas Prediksi		Positif	Negatif
	Positif	443695 (TP)	30699 (TN)
	Negatif	13067 (FN)	10660 (FN)

- **True Positive (TP)** : ketika prediksi yes dan faktanya yes (hasilnya benar). ‘
- **True Negative (TN)** : ketika prediksi no dan faktanya no (tidak ada hasil yang benar).
- **False Positive (FP)** : ketika prediksi yes dan faktanya no (hasilnya tidak diharapkan).
- **False Negative (FN)** : ketika prediksi no dan faktanya yes (hasilnya meleset).

Dan Visualisasi hasilnya seperti berikut ini



### 5.2.1 ANN

Implementasi menggunakan ANN menghasilkan hasil berikut ini.

```

> warnings.filterwarnings('ignore')
y_pred_testing = [np.argmax(element) for element in y_pred_testing]
print(confusion_matrix(df_realclass, y_pred_testing))
print(classification_report(df_realclass, y_pred_testing))
print("Accuracy:", accuracy_score(df_realclass, y_pred_testing)*100)

[[453699  20695]
 [ 13681  10046]]
              precision    recall  f1-score   support

      0       0.97       0.96       0.96    474394
      1       0.33       0.42       0.37     23727

   accuracy          0.93    498121
  macro avg       0.65    0.69    0.67    498121
 weighted avg       0.94    0.93    0.94    498121

Accuracy: 93.09886553668687

```

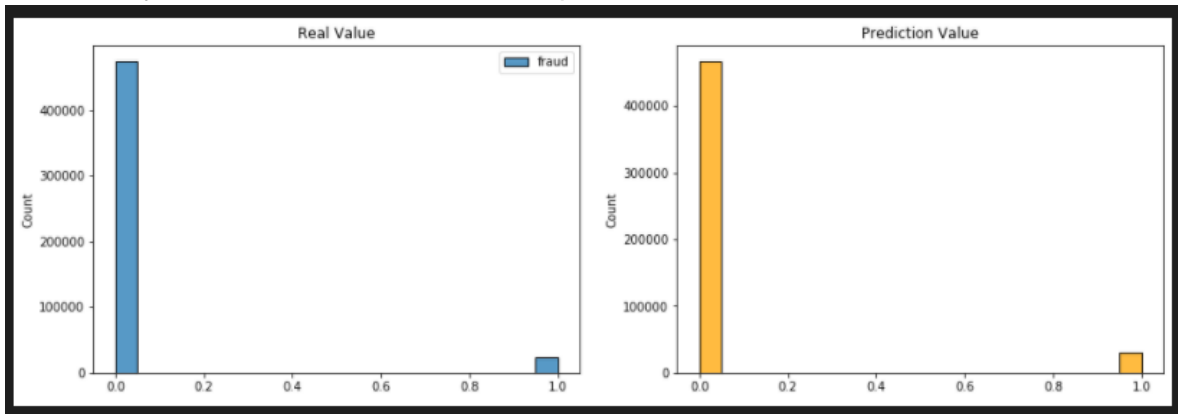
Hasil klasifikasi yang dilakukan memiliki akurasi 93% dan bisa dibilang cukup bagus dan tidak mengalami overfitting, dan pada pengujian ini memiliki nilai confusion matrix yang membentuk nilai *precision*, *recall*, *f1-score* **0 = ( 97%, 96% , 96%) dan 1 = (33%, 42%, 37%)**.

Kategori	Kelas Aktual		
Kelas Prediksi		Positif	Negatif
	Positif	453699 (TP)	20695 (TN)
	Negatif	13081 (FN)	10046 (FN)



- **True Positive (TP)** : ketika prediksi yes dan faktanya yes (hasilnya benar). ‘
- **True Negative (TN)** : ketika prediksi no dan faktanya no (tidak ada hasil yang benar).
- **False Positive (FP)** : ketika prediksi yes dan faktanya no (hasilnya tidak diharapkan).
- **False Negative (FN)** : ketika prediksi no dan faktanya yes (hasilnya meleset).

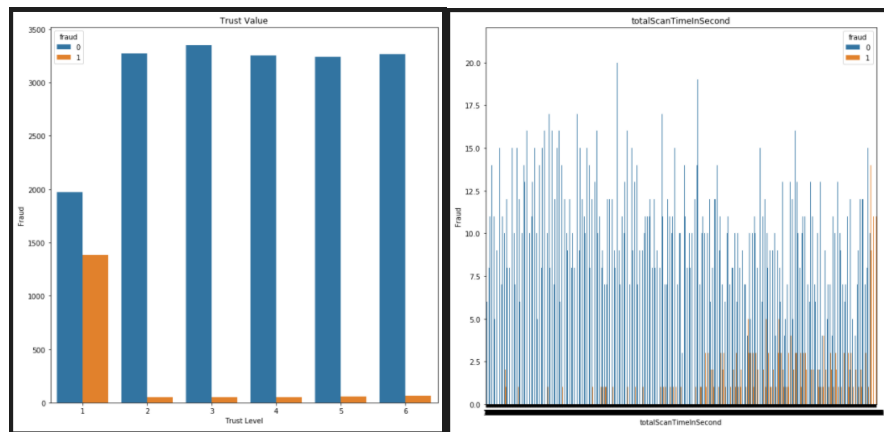
Dan hasilnya bila di visualisasikan akan seperti berikut ini.

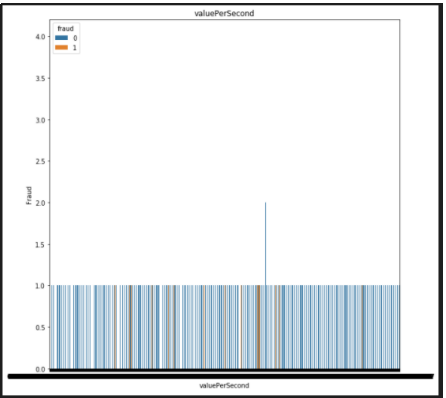
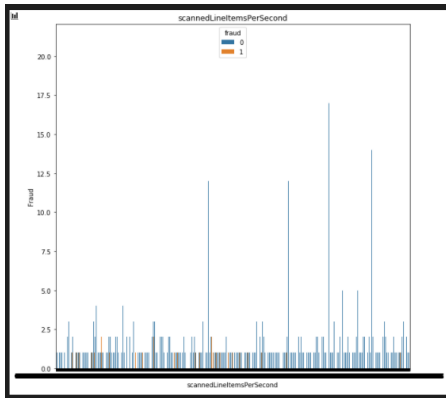
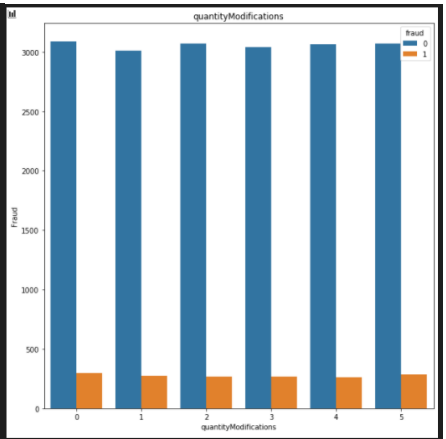
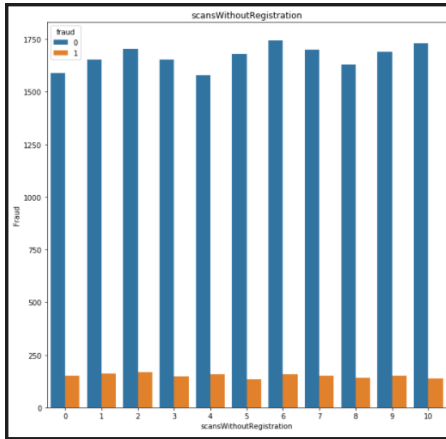
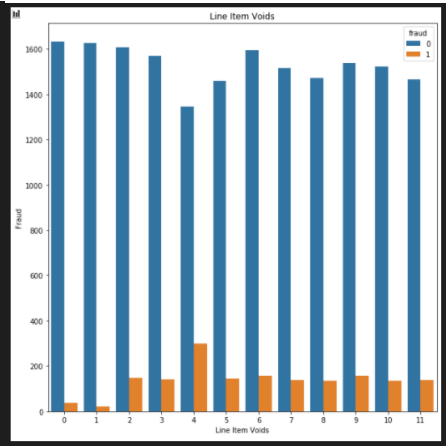
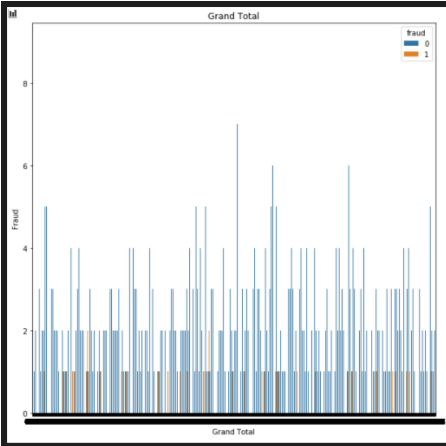


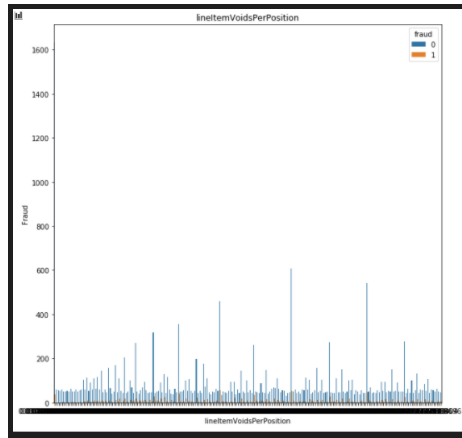
## 6. Interpretasi Hasil.

Untuk membantu proses interpretasi ini dilakukan visualisasi *count plot* dengan hue = class dan datanya berasal dari masing-masing fitur sehingga dapat diketahui apa makna yang dikandung oleh hasil tersebut.

### 6.1. Decision Tree

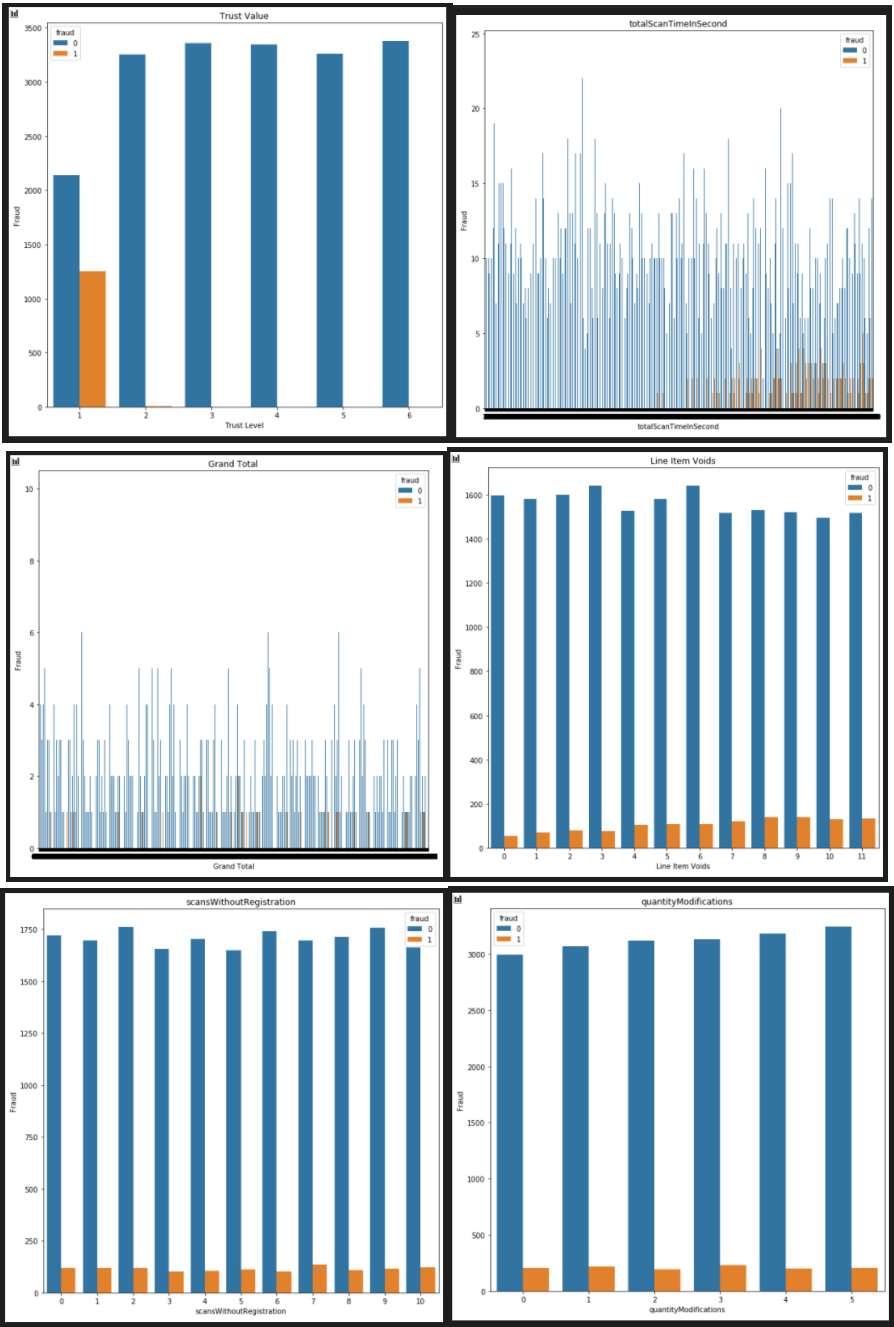


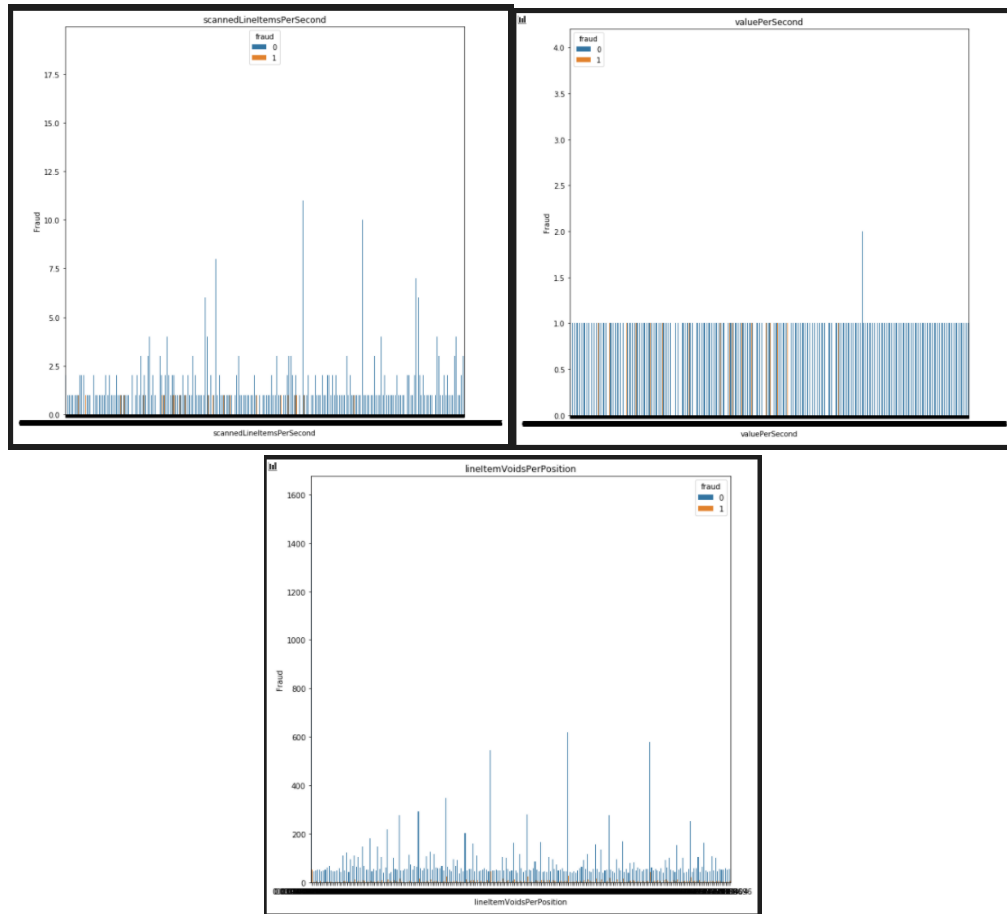




Jika Menggunakan Decision Tree Bisa dilihat hasilnya bahwa ada kecenderungan bila tingkat kepercayaan terhadap sistem rendah maka akan meningkatkan kegiatan penipuan pada sistem, kemudian bila mengacu terhadap kegiatan melakukan scan data tanpa ada informasi yang jelas atau valid secara keseluruhan dalam tingkatan kegiatannya selalu menghasilkan penipuan oleh karena itu kalau bisa kegiatan ini seharusnya di minimalisir agar setiap scan itu terdaftar dan valid, hal ini sama juga bila diperhatikan memiliki pola yang sama dengan setiap kegiatan modifikasi pesanan pada setiap tingkatan kegiatan hampir memiliki distribusi kegiatan penipuan yang sama, dan untuk produk yang dibatalkan bisa dilihat untuk kegiatan fraudnya bisa dilihat cukup kecil persebarannya, selanjutnya bila merujuk kepada banyak item yang discan, semakin banyak maka semakin meningkat pula kegiatan penipuan yang terjadi oleh karena itu bisa disimpulkan bahwa **jika keyakinan berada pada nilai yang rendah kemudian terdapat data yang discan dan tidak valid selanjutnya terjadi banyak modifikasi terhadap belanjaan dan memiliki jumlah belanjaan yang dibeli semakin meningkatkan kemungkinan bahwa itu adalah penipuan**, namun sebaliknya bila **keyakinan terhadap hasil belanja tinggi, data yang di scan menengah ke bawah, modifikasi belanjaan tidak terlalu banyak dan jumlahnya juga tidak terlalu banyak bisa meningkatkan bahwa kegiatan yang dilakukan bukan penipuan melainkan hasil yang asli**.

6.2. ANN





Jika Menggunakan ANN bisa dilihat akan mengklasifikasikan penipuan bila tingkat kepercayaan terhadap pembelian itu rendah, kemudian jika total dari barang yang di-scan persatuan waktu mulai dari menengah sampai keatas mengalami peningkatan bahwa itu kegiatan penipuan, untuk setiap barang yang dibeli ada kecenderungan untuk setiap delapan atau sembilan akan terjadi penipuan, kemudian untuk setiap scan yang tidak valid bisa dilihat persebarannya cukup merata dari seluruh kategori sehingga bisa dibilang jika ada ini maka ada kemungkinan kegiatan yang dilakukan itu penipuan, begitu pula pada data modification jika terjadi itu maka ada kemungkinan terjadi penipuan, sehingga bila disimpulkan **penipuan akan terjadi bila kepercayaan terhadap hasil transaksi rendah, kemudian barang yang discan memiliki nilai yang cukup banyak, ketika pembelian barang di kisaran 8-9 sering terjadi penipuan, dan jika terjadi pengubahan atau ada scan tidak valid maka itu ada kemungkinan terjadi penipuan**, selanjutnya bila data itu bukan penipuan bila tingkat kepercayaan terhadap pembelian cukup tinggi, barang yang dibeli tidak banyak yang discan, dan pembelian dilakukan secara wajar tanpa ada hal yang tidak valid dan dilakukan dengan cepat bisa dikatakan tidak terjadi **penipuan terhadap transaksi**, sehingga menurut data ini perlu diperhatikan jika terjadi transaksi yang tidak meyakinkan dan dilakukan dengan sangat cepat dan membludak dan sering terjadi kesalahan maka ini ada kemungkinan transaksi yang dilakukan palsu.

## 6. Kesimpulan.

Dari percobaan yang dilakukan bisa dikatakan untuk algoritma yang optimum dan dapat digunakan adalah ANN, namun Decision Tree juga dapat digunakan karena keduanya hanya berbeda akurasi sebesar 2%, dan juga setelah dilakukan beberapa kali percobaan ANN terkadang hanya dapat memprediksi satu nilai saja dalam kata lain tidak bisa memprediksi variabel **penipuan**, sehingga perlu dilakukan penyesuaian, untuk pemilihan fitur dalam pembangunan sendiri dapat menggunakan fitur **trustLevel**, **lineItemVoids**, **totalScanTimeInSeconds**, **grandTotal** hal ini diputuskan karena pemilihan dari fitur yang dikaji dari beberapa metode dan ditemukan kemunculan empat variabel tersebut di dalamnya, dan akurasi terbaik dari percobaan ini adalah dengan menggunakan ANN didapati hasil **93%** dan bila Decision Tree **91%** dengan penjelasan parameter seperti yang dituliskan pada laporan ini, namun kegiatan ini masih bisa dioptimalkan lagi karena menurut penulis masih bisa di atur lagi layer dan penggunaan fitur-fitur lainnya yang belum penulis lakukan.

## DAFTAR PUSTAKA

- [1] Gheware, S. D., Kejkar, A. S., & Tondare, S. M. (2014). Data mining: Task, tools, techniques and applications. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(10).
- [2] Navlani A, "Decision Tree Classification in Python", Python Decision Tree Classification with Scikit-Learn DecisionTreeClassifier - DataCamp, diakses pada 09/06/2021
- [3] Sidiropoulou, K., Pissadaki, E. K., & Poirazi, P. (2006). Inside the brain of a neuron. *EMBO reports*, 7(9), 886-892.