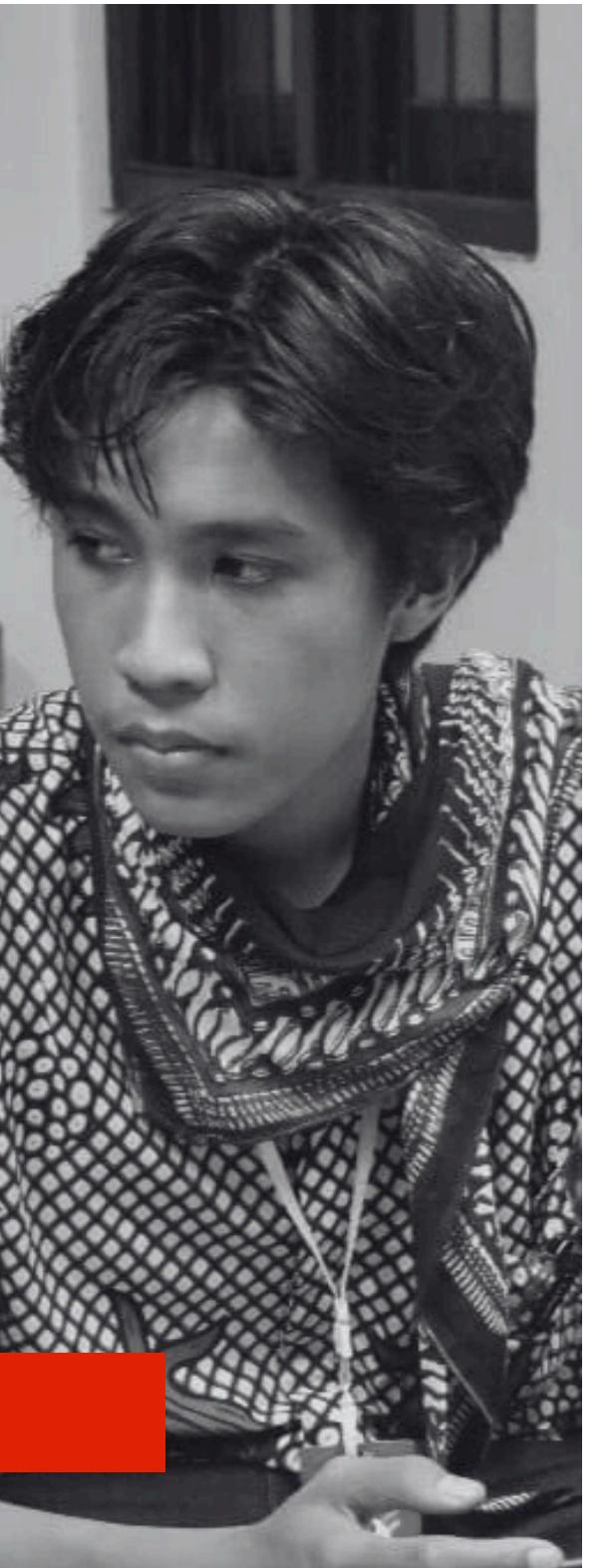


# FRAUD DETECTION PROJECT II

**Andrew NG**

11 October 2024

# Meet the Team



**Dwi Candra Nur Rohman**

**Vivaldy Marentino Dermagia**

**Cefiro Martha Yudhistira**

**I Gusti Ayu Meliniarayani**



THIS NOTE IS LEGAL  
FOR ALL DEPARTMENTS

# Background & Problem Statement

**Fraud** merupakan **tindakan ilegal dengan cara mengelabui**, menipu, atau memanipulasi sistem yang **bertujuan untuk mendapatkan keuntungan pribadi**. Tindakan ini dapat dilakukan oleh individu atau grup yang memiliki posisi kepercayaan atau otoritas tertentu dalam sebuah institusi.

Beberapa **dampak Fraud Transaction** antara lain: **kerugian finansial** yang signifikan baik individu maupun bisnis. **reputasi yang bisnis yang tercemar**, akibat sering terjadinya penipuan. Pelanggan mungkin merasa tidak aman melakukan transaksi dengan perusahaan yang sering mengalami fraud, dan Fraud transactions dapat mengakibatkan **proses hukum terhadap pelaku maupun perusahaan** yang gagal melindungi data konsumen.

Sebuah survei yang dilakukan pada periode post-pandemi COVID-19 menunjukkan bahwa **sebanyak 51% responden mengalami penipuan selama dua tahun terakhir sejak merebaknya pandemi COVID-19 pada awal tahun 2020**. Angka ini **merupakan presentase tertinggi dalam 20 tahun** penelitian fraud oleh PricewaterhouseCoopers (PwC)



# Objectives & Scope

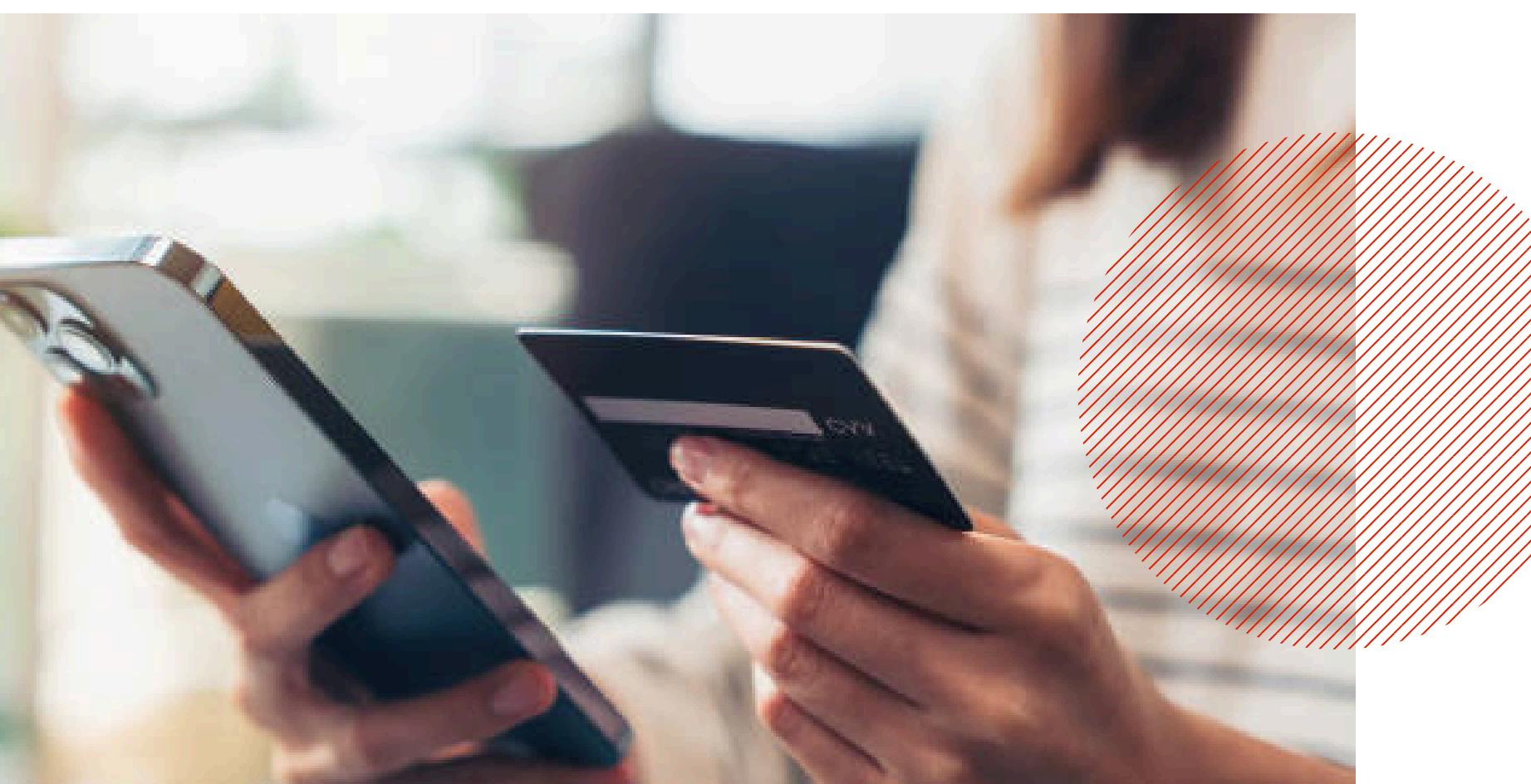
## Objectives:

- Membangun Sistem Deteksi Fraud:** Mengembangkan model machine learning yang mampu mendeteksi transaksi fraud secara akurat menggunakan data historis transaksi.
- Evaluasi Kinerja Model:** Mengevaluasi kinerja model dengan menggunakan berbagai metrik seperti ROC AUC, balanced accuracy, precision, recall, dan F1 score, untuk mendapatkan model terbaik dalam mendeteksi fraud.
- Optimisasi Model:** Menggunakan GridSearchCV untuk melakukan hyperparameter tuning pada beberapa model (Logistic Regression, Decision Tree, Random Forest, dan AdaBoost) agar performa model optimal.
- Menentukan Algoritma Terbaik:** Membandingkan beberapa algoritma machine learning untuk menemukan algoritma yang paling cocok dan efisien dalam mendeteksi fraud.



## Scope:

- Data Preprocessing:** Melakukan preprocessing pada data dengan encoding kategori, scaling data numerik, dan handling missing values jika ada. Penggunaan teknik manual encoding dan pipeline untuk mempermudah pengolahan data.
- Modeling:** Mengimplementasikan beberapa algoritma machine learning, termasuk Logistic Regression, Decision Tree, Random Forest, dan AdaBoost, sebagai model untuk deteksi fraud.
- Train-Test Split:** Membagi dataset menjadi data train, validation, dan test untuk memastikan model tidak overfitting serta memvalidasi kinerja model pada data yang belum pernah dilihat.
- Model Tuning:** Melakukan hyperparameter tuning menggunakan GridSearchCV untuk menemukan parameter optimal dari setiap model.
- Model Comparison:** Membandingkan model yang telah dibangun berdasarkan berbagai metrik evaluasi, serta memilih model terbaik berdasarkan performa pada data validation dan test.



# Data Collection & Preparation

Data Fraud : 7506

Data Fraud 2019 : 5220

Data Fraud 2020: 2286

```
# cek data duplicated  
train.duplicated().sum()  
✓ 3.9s  
0
```

Feature:

- Identitas Pengguna
- Waktu transaksi
- Lokasi Transaksi
- Jumlah Transaksi
- Kategori barang transaksi
- Populasi suatu lokasi

Dataset terdiri dari 7506 entry data fraud dan 22 kolom dengan 5220 entry pada tahun 2019 dan 2286 entry di tahun 2020

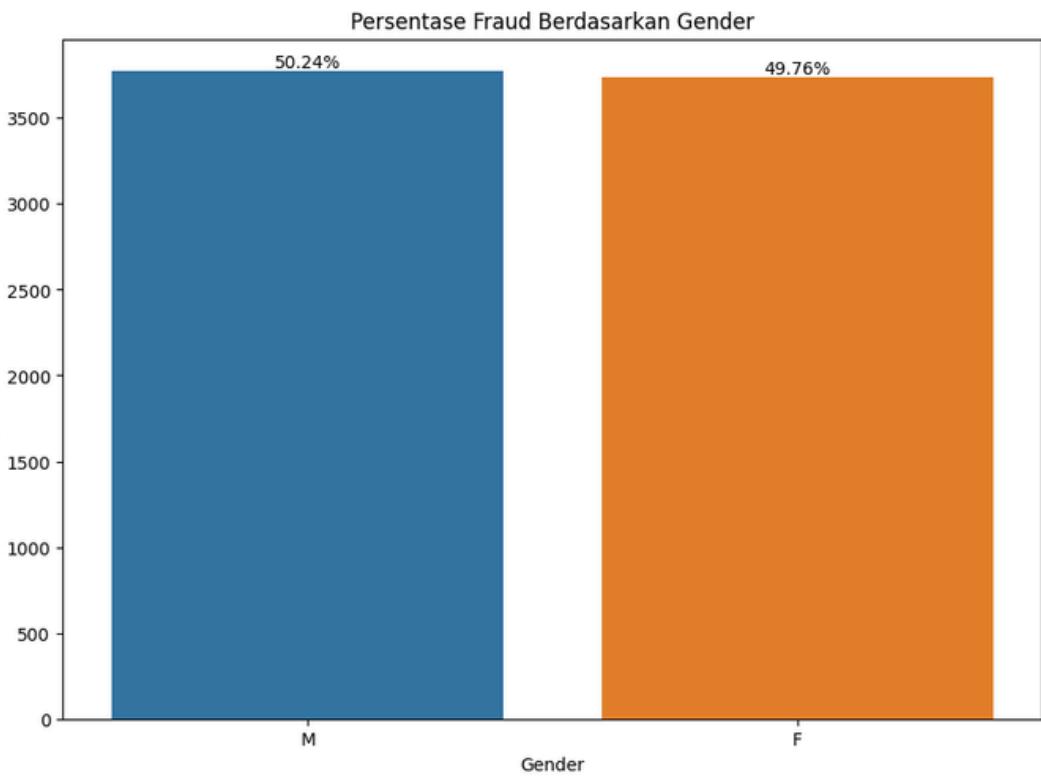
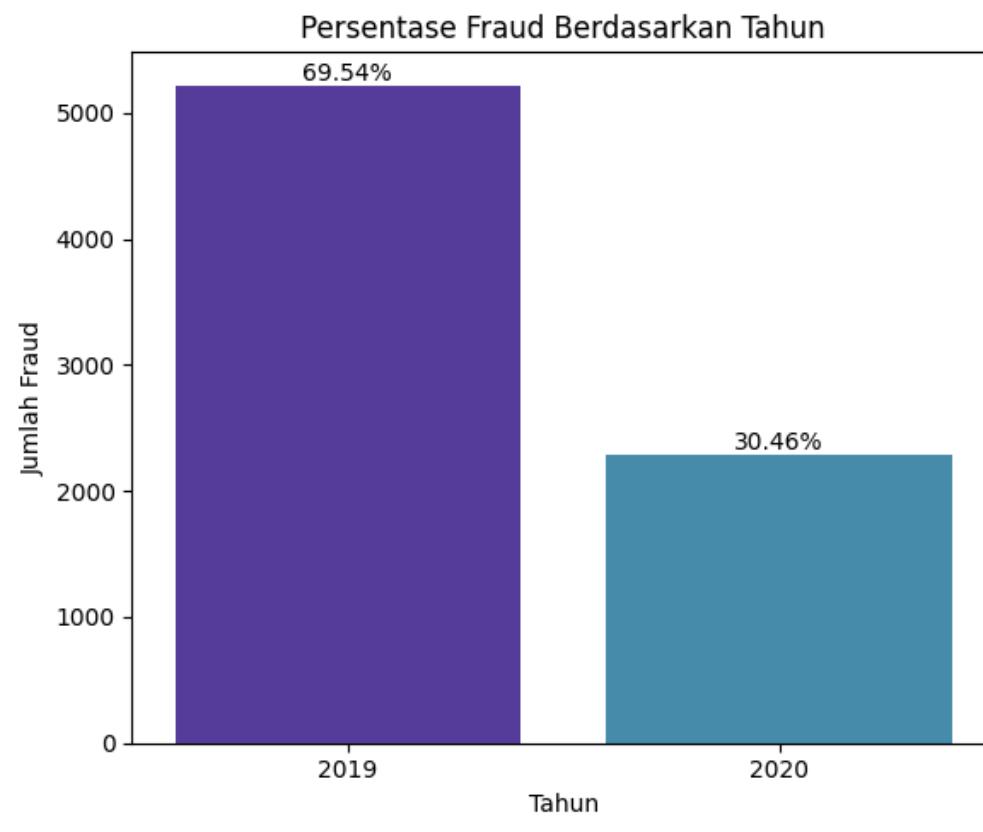
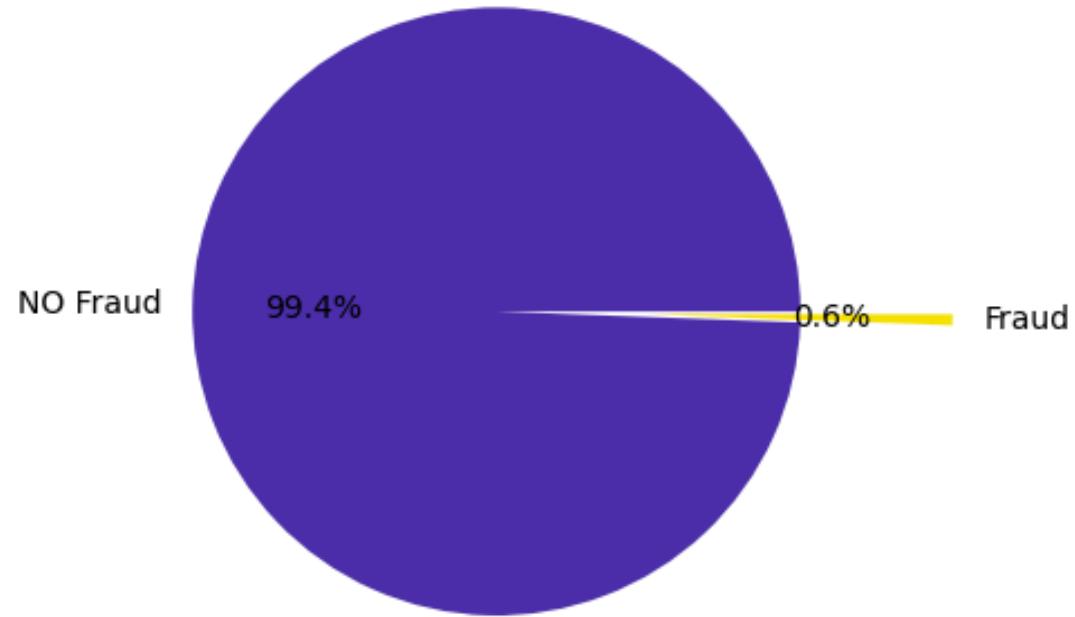
trans_date_trans_time	0
cc_num	0
merchant	0
category	0
amt	0
first	0
last	0
gender	0
street	0
city	0
state	0
zip	0
lat	0
long	0

Data tidak terdapat missing value dan data duplicated



# Data Analysis

## Proporsi Transaksi Fraud

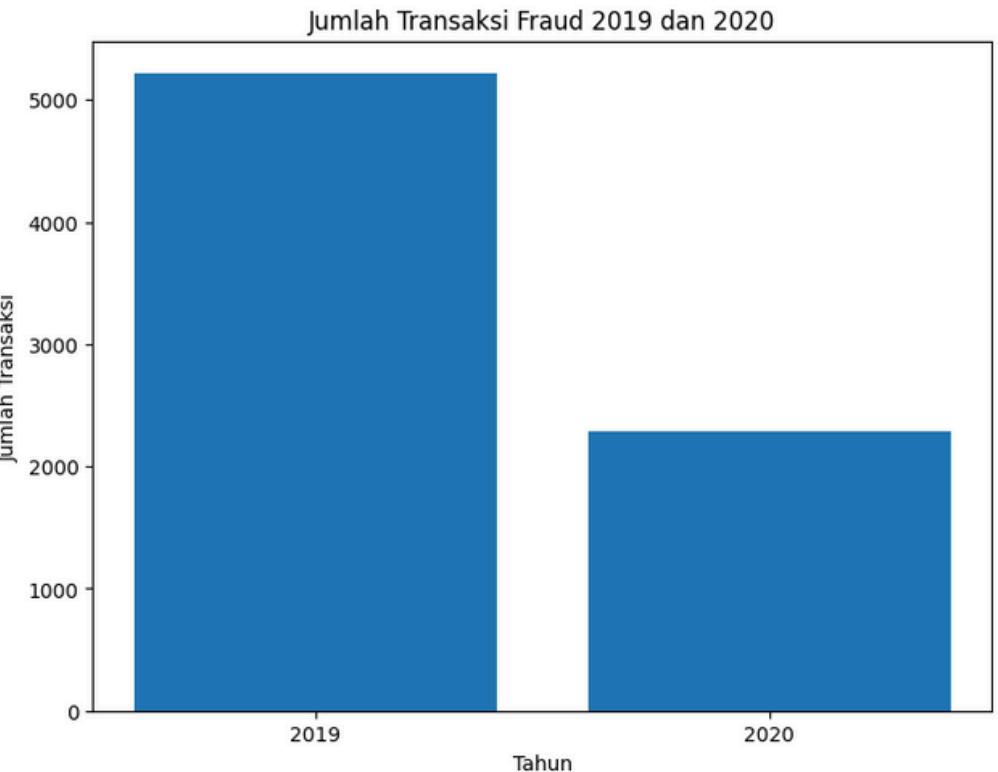
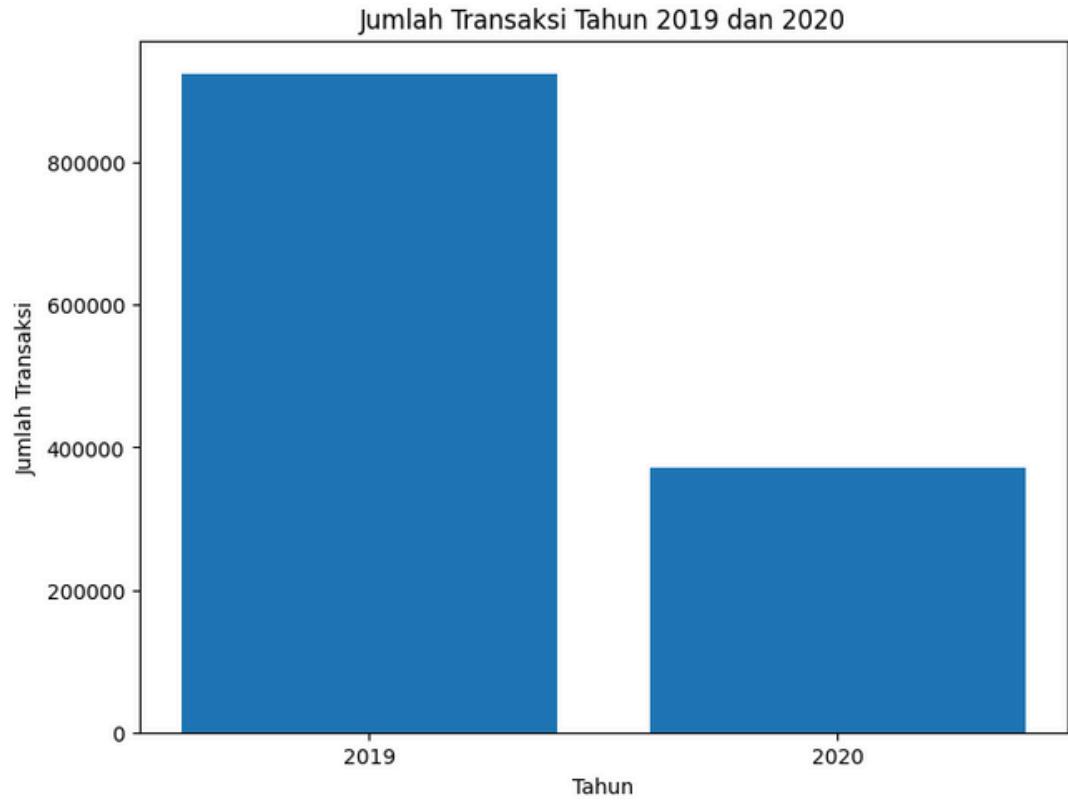


gender	mean	diff	risk
F	0.005262	-0.000527	0.908947
M	0.006426	0.000638	1.110146

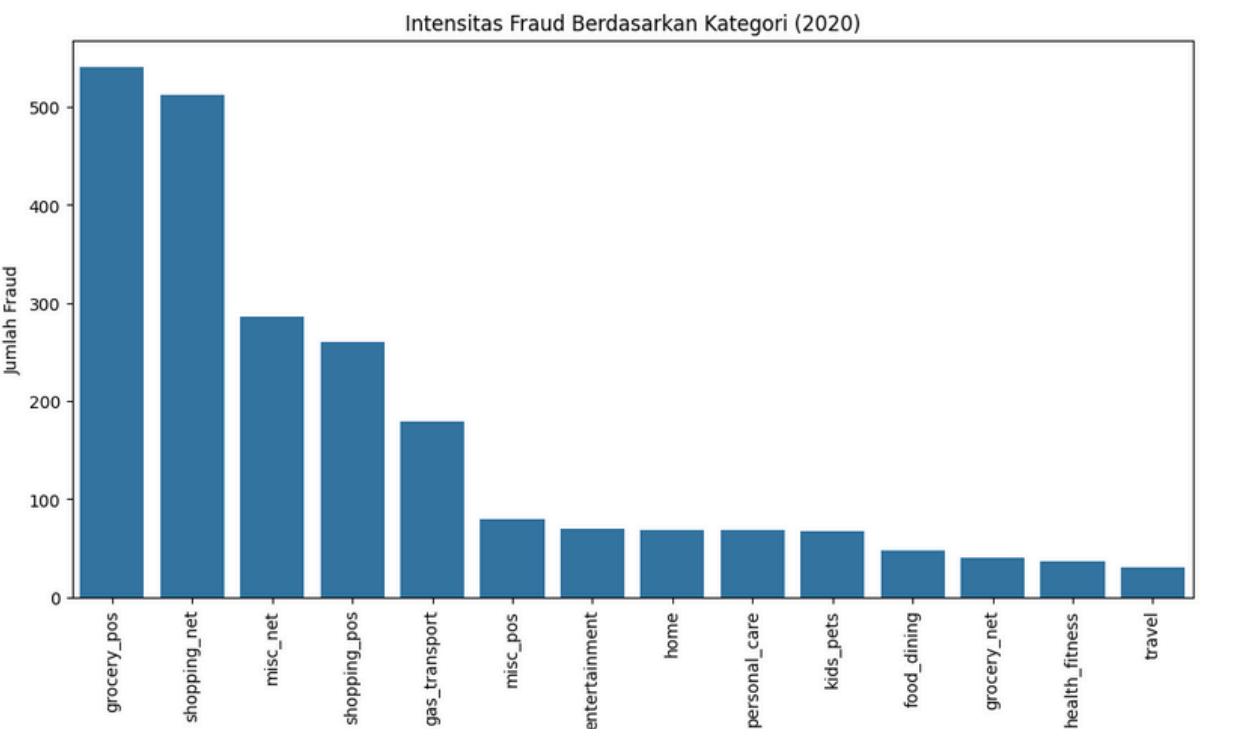
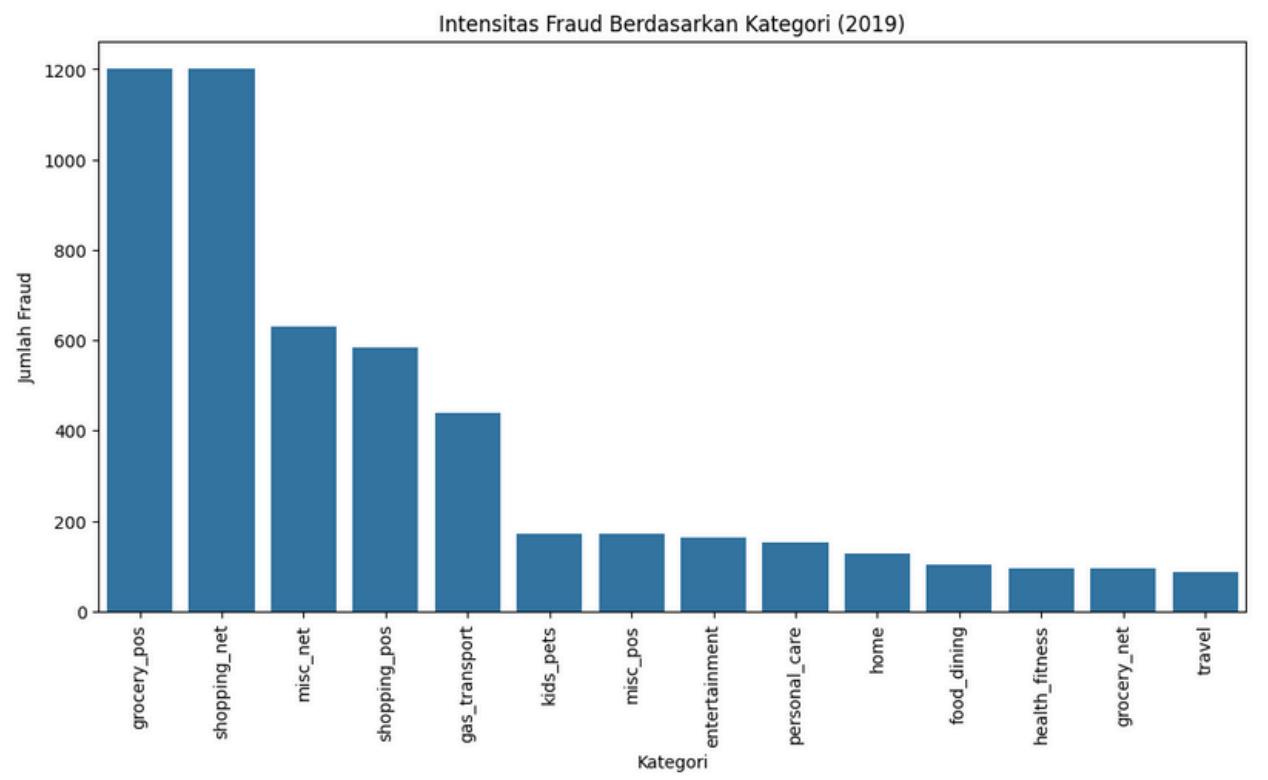
### Insight :

- Hanya **0.6%** dari seluruh transaksi dalam dataset yang terindikasi **fraud** selama **periode 2019-2020**.
- Fraud lebih sering terjadi pada tahun 2019**, terutama pada bulan **Desember**, sementara di **tahun 2020**, puncak kasus fraud terjadi pada bulan **Mei**. **Penting untuk dicatat** bahwa transaksi di **tahun 2019 berlangsung hingga Desember penuh**, sedangkan transaksi di tahun **2020 hanya tercatat hingga bulan Juni**.
- Fraud cenderung tersebar merata antara **pria dan wanita**, namun **laki-laki** menunjukkan sedikit **kecenderungan lebih tinggi untuk terlibat dalam transaksi fraud dibandingkan dengan perempuan**.

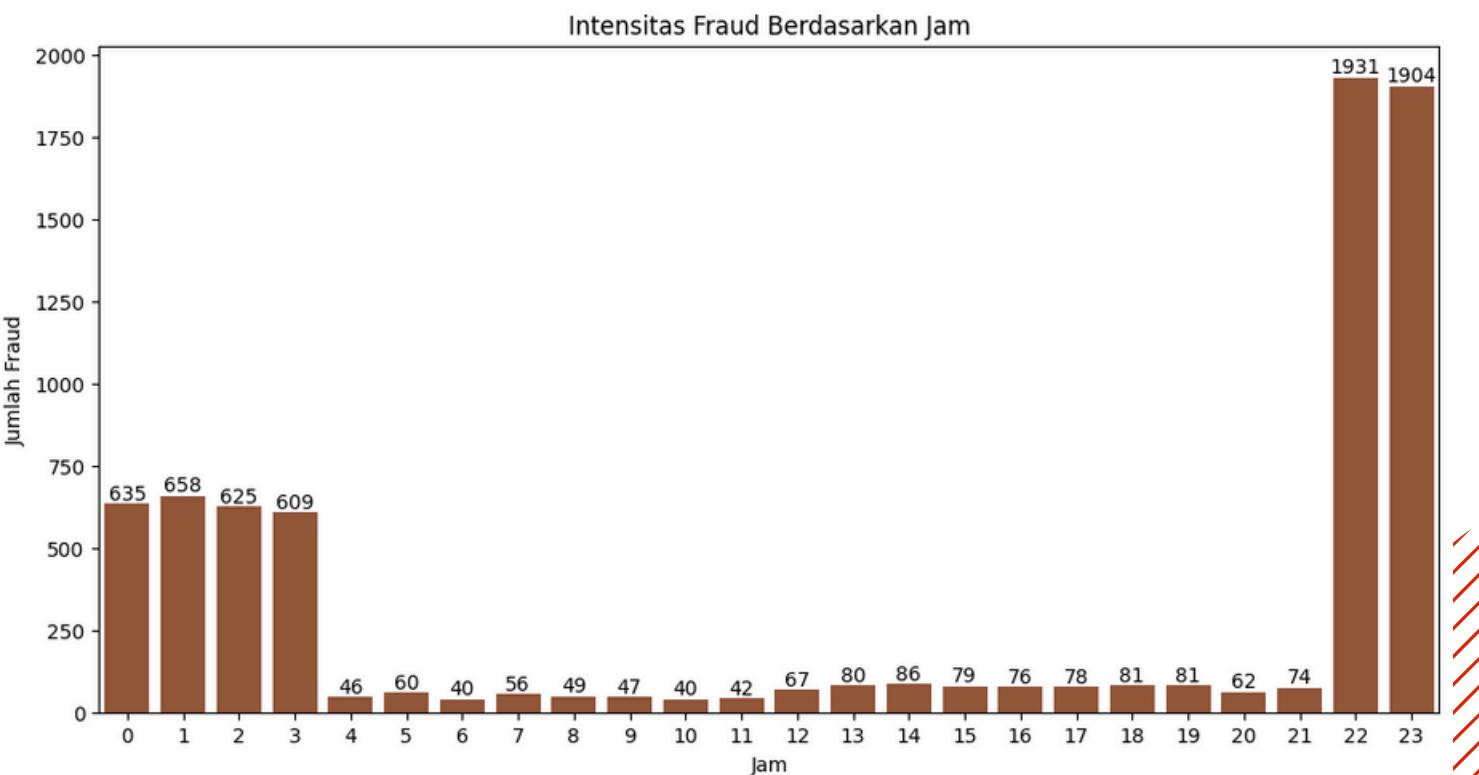
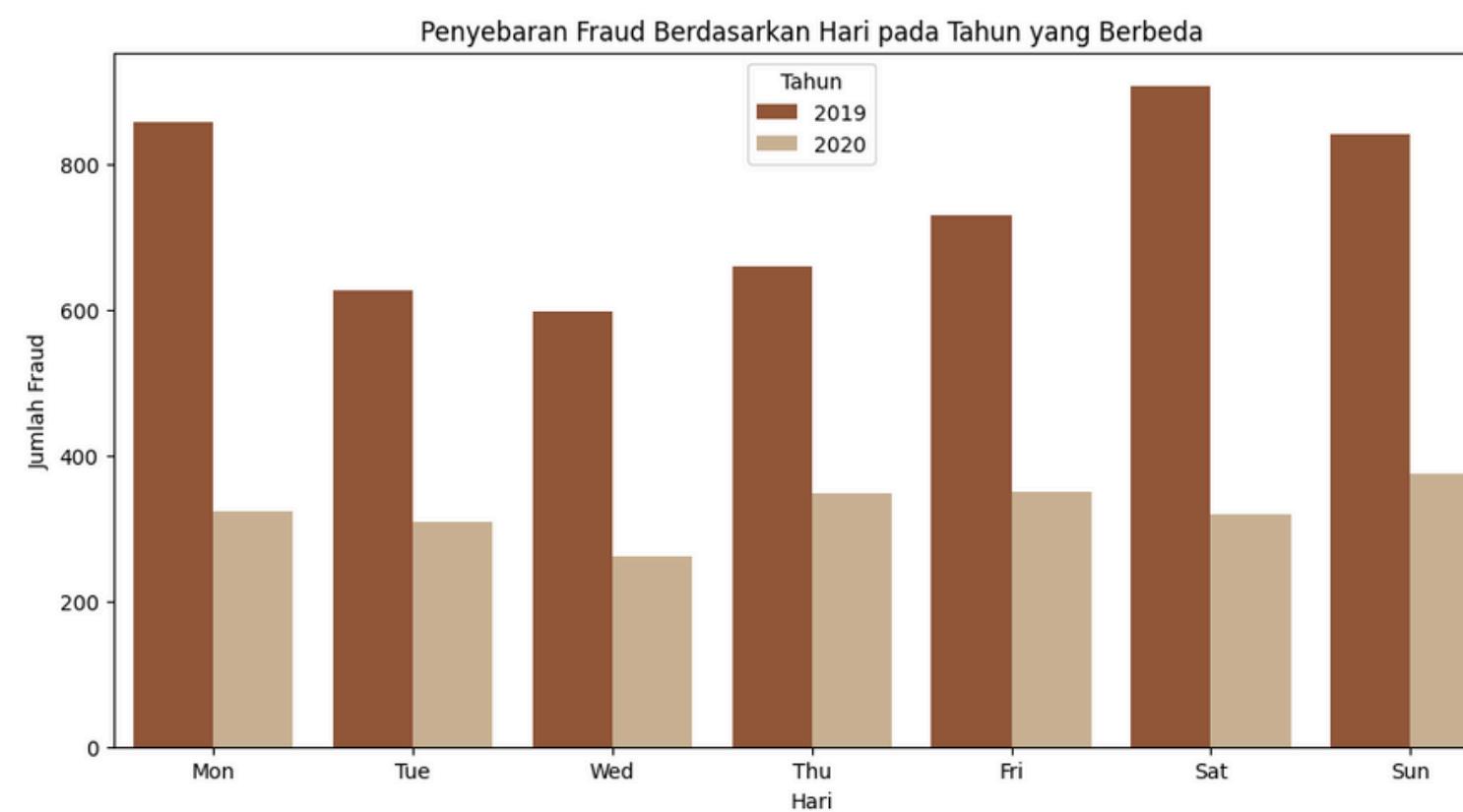
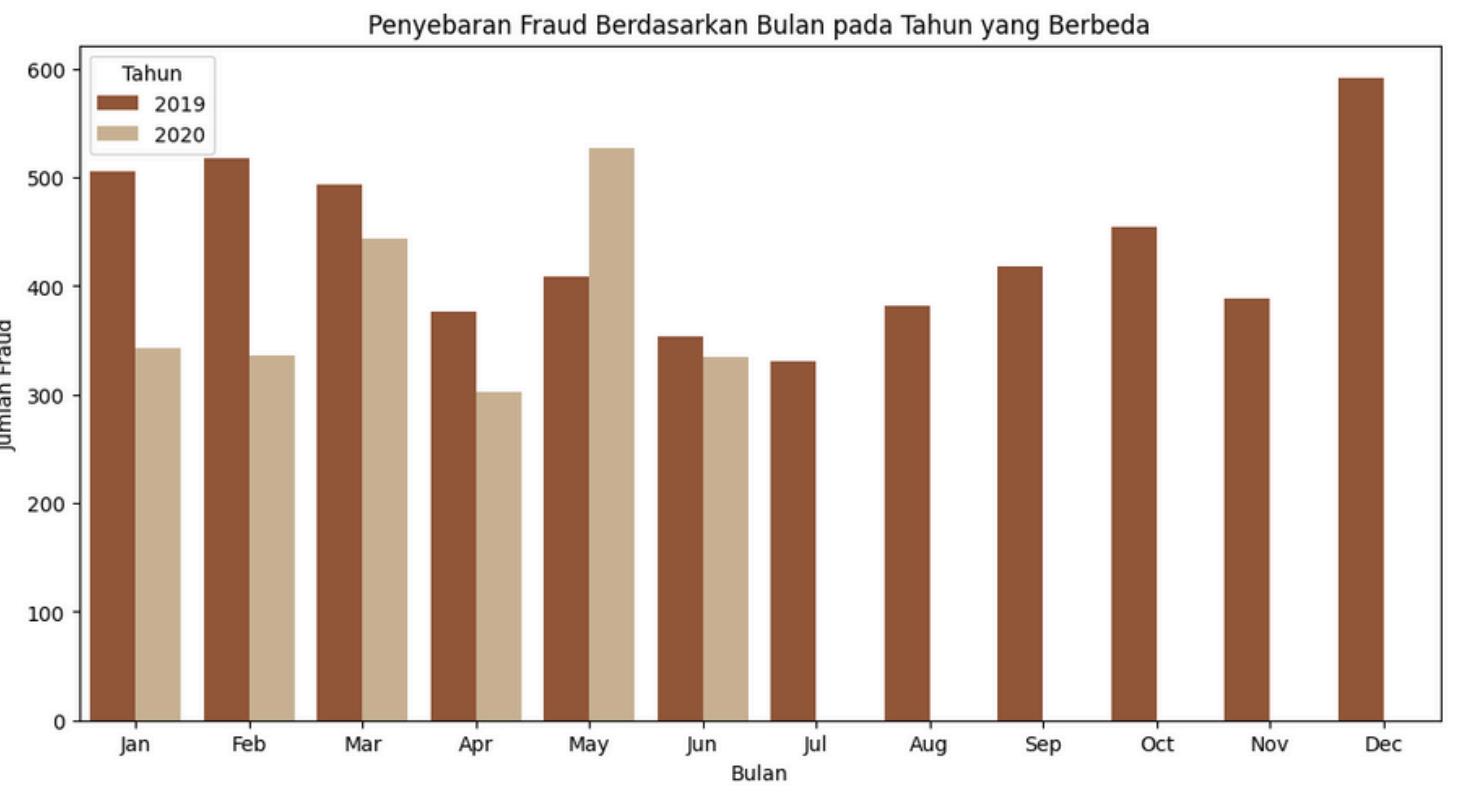
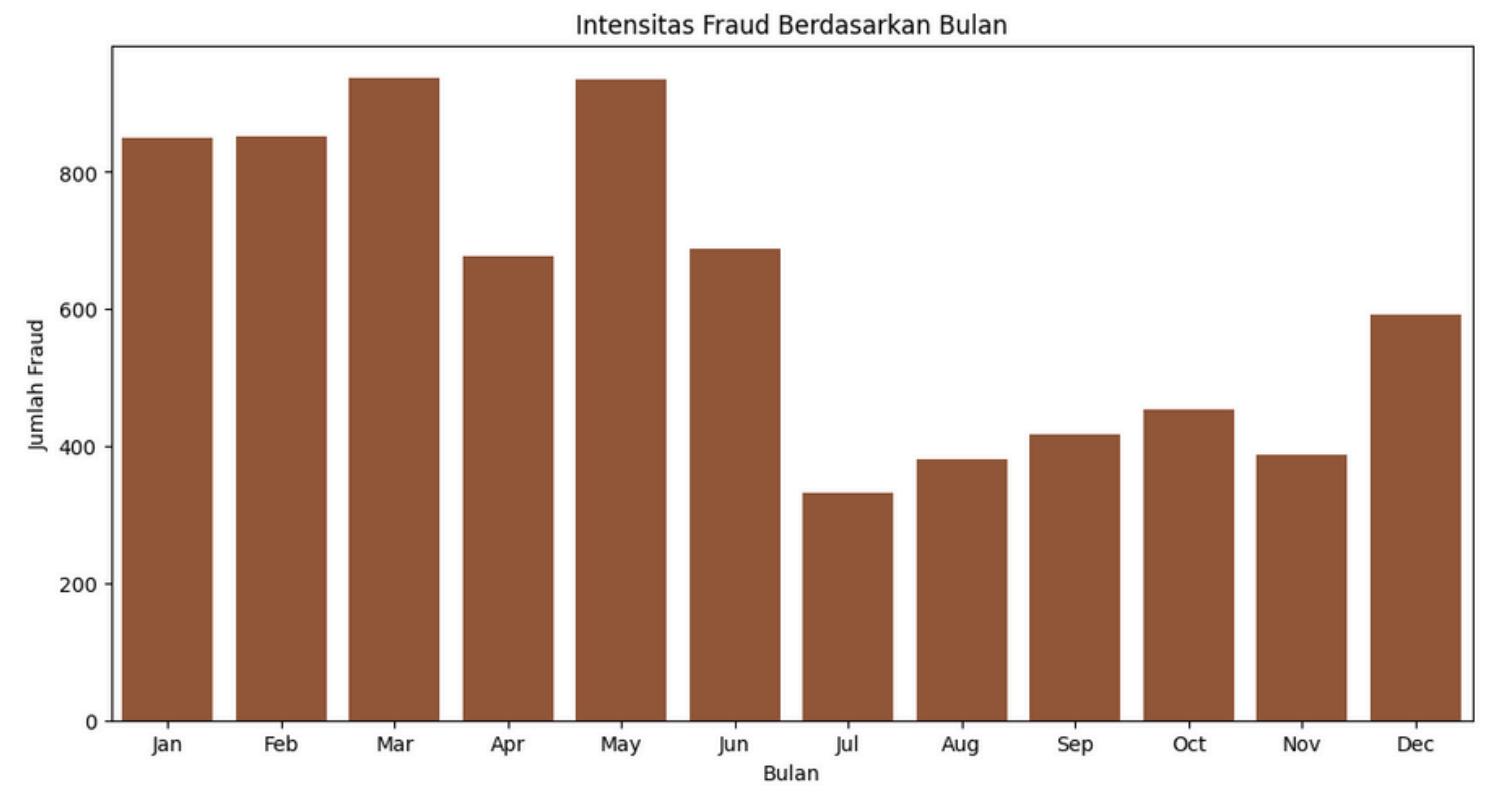
# Data Analysis



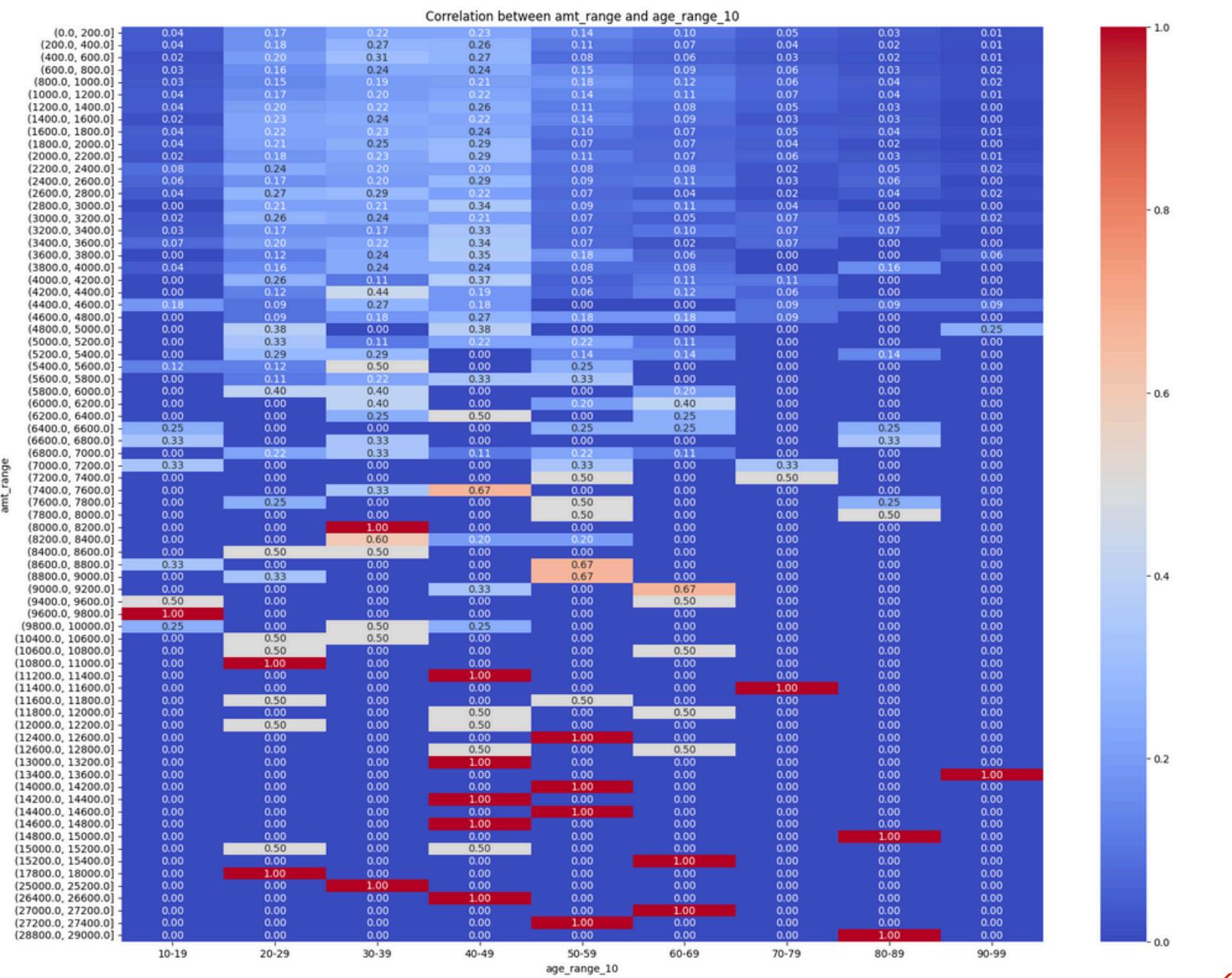
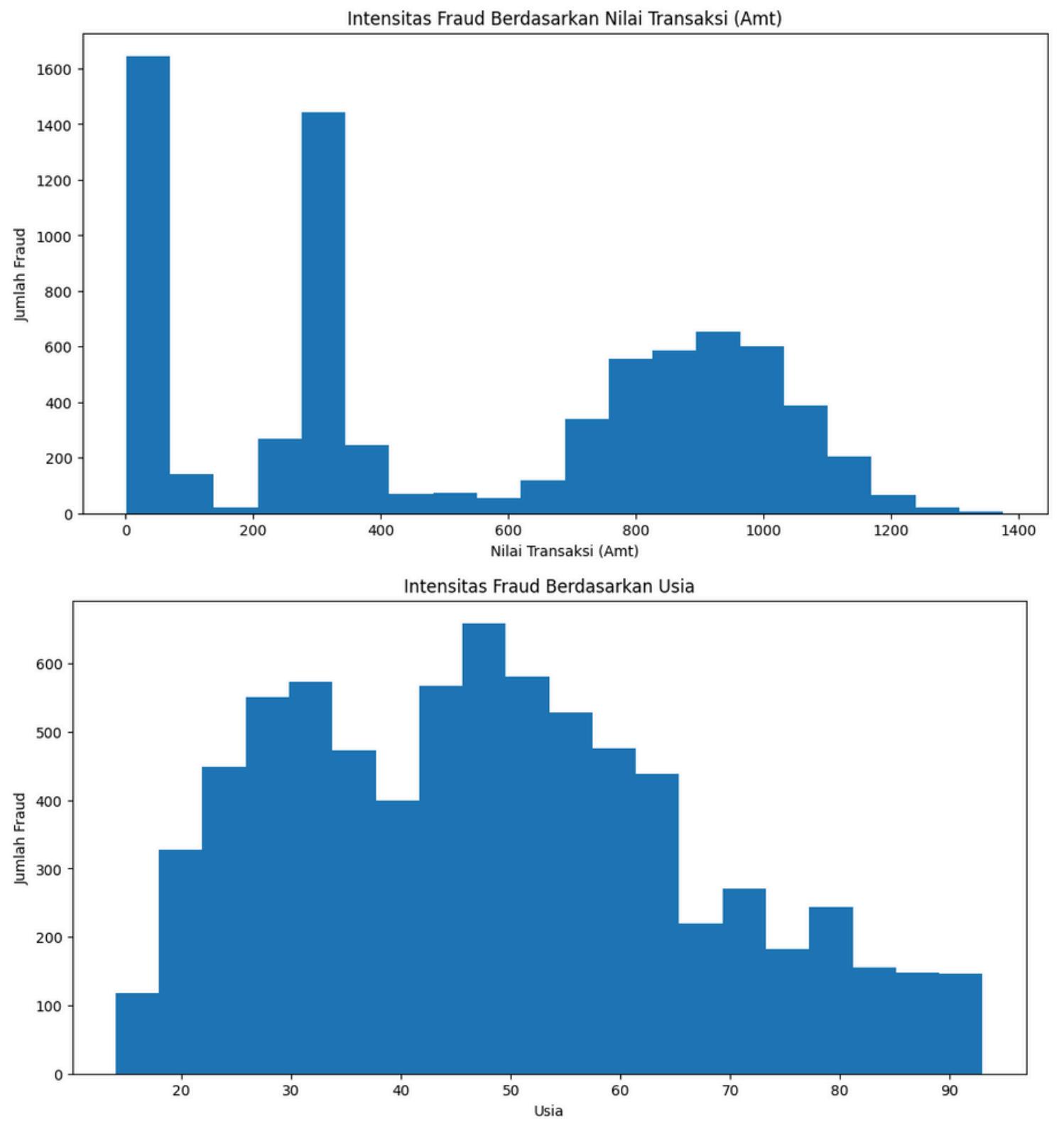
```
print(f"Percentase data fraud di transaksi 2020: {fraud_percent}")  
# Filter data yang fraud
```



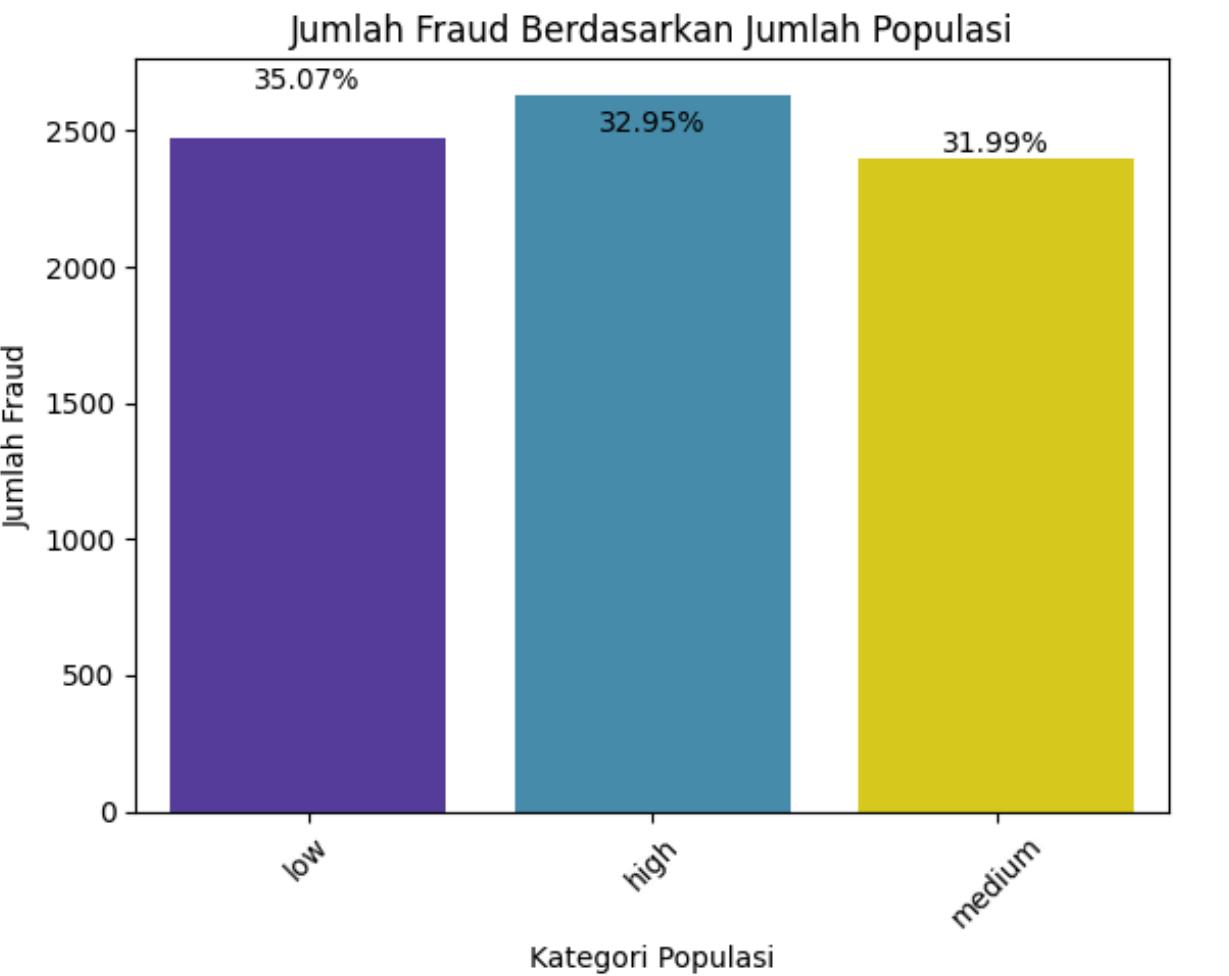
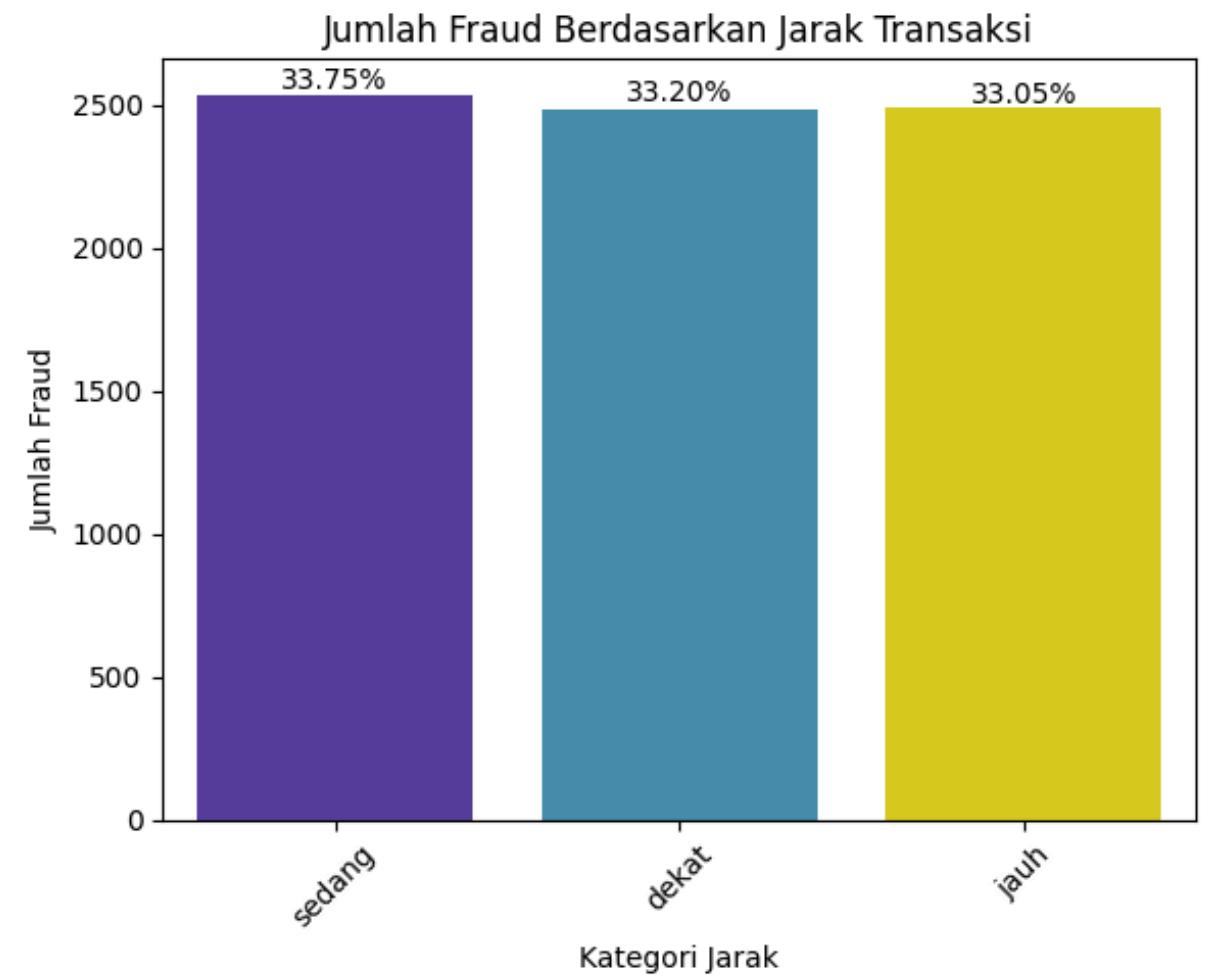
# Data Analysis



# Data Analysis



# Data Analysis



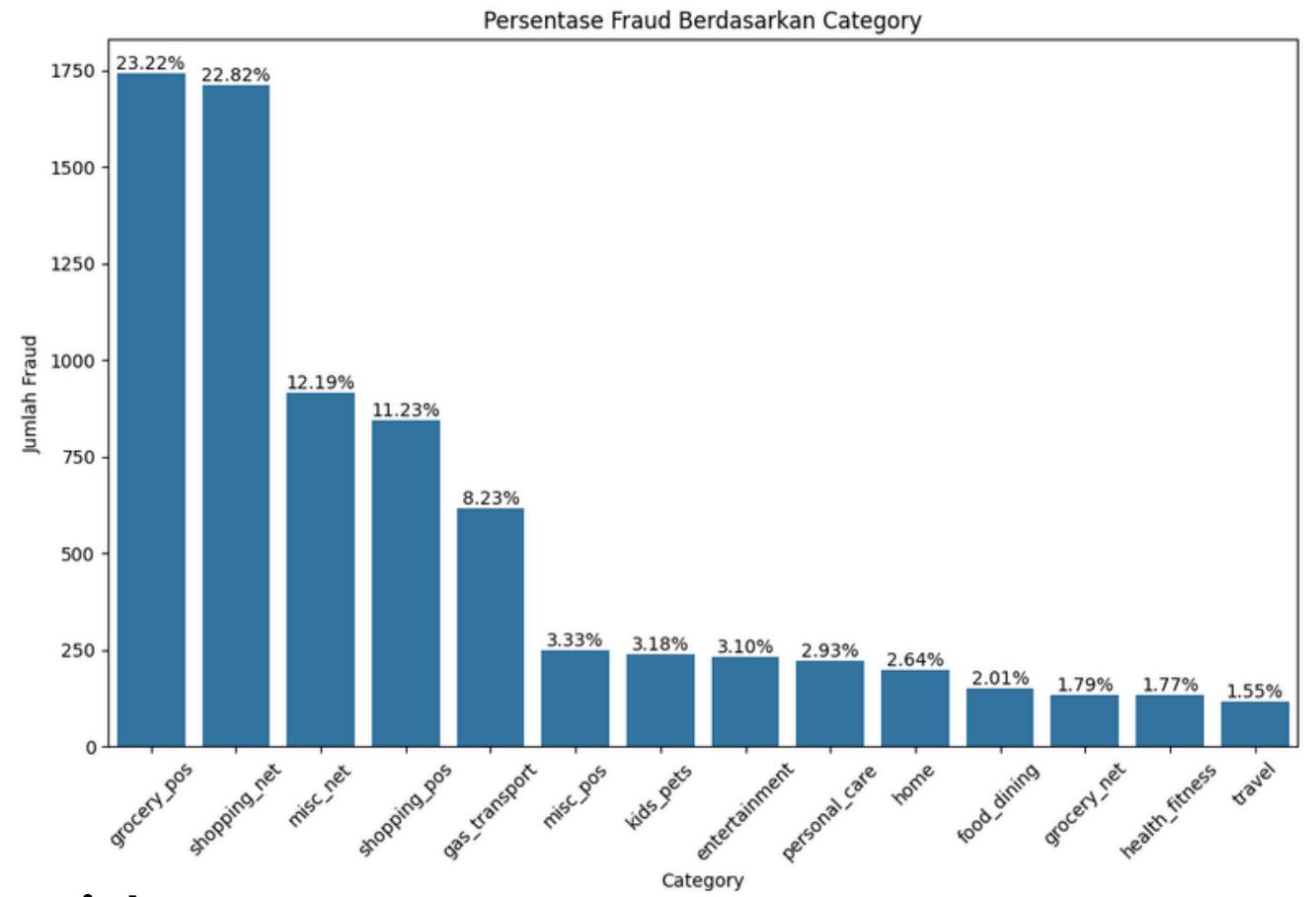
	mean	diff	risk
distance_category_1			
dekat	0.005740	-0.000049	0.991607
jauh	0.005766	-0.000023	0.996003
sedang	0.005860	0.000072	1.012390

	mean	diff	risk
city_pop_category_1			
high	0.006097	0.000309	1.053299
low	0.005715	-0.000074	0.987192
medium	0.005555	-0.000234	0.959595

## Insight :

- **Kota dengan populasi tinggi dan rendah cenderung memiliki tingkat fraud lebih tinggi dibandingkan kota dengan populasi menengah.**
- Pelanggan yang tinggal pada **jarak sedang dan jarak jauh dari merchant** lebih sering terlibat dalam **transaksi fraud** dibandingkan mereka yang tinggal lebih dekat.

# Data Analysis



## Insight :

state	city	city_pop	fraud_count
615	TX	Houston	2906700
104	FL	Naples	276002
12	AL	Huntsville	190178
196	KS	Topeka	163415
606	TX	Dallas	1263321
499	OK	Tulsa	413574
259	MI	Detroit	673342

DC	0.005812	0.000024	1.004093
DE	1.000000	0.994211	172.751799

- Kategori transaksi dengan tingkat fraud tertinggi meliputi **grocery\_pos, shopping\_net, misc\_net, dan shopping\_pos**.
- Texas, khususnya kota Houston dengan populasi 2.9 juta jiwa, memiliki jumlah kasus fraud terbanyak, yaitu **39** kasus.
- Beberapa kota dengan kecenderungan fraud yang lebih tinggi di antaranya: Angwin, Ashland, Beacon, Brookfield, Bruce, Buellton, Byesville, Chattanooga, Clarion, Claypool, Clinton, Coulee Dam, Crouse, Downey, East China, Freeport, Gaines, Granbury, Greenport, dan masih banyak lagi.
- Delaware (**DE**) menunjukkan kecenderungan fraud yang lebih tinggi dibandingkan negara bagian lainnya.

# Model Development

## Data Preprocessing:

- **Handling Missing Value & Duplicate**: Tidak dilakukan karena tidak terdapat Misssing value dan data duplicate
- **Encoding Categorical Variables**: One-Hot Encoding pada kolom ['category', 'gender'] dan Ordinal Encoding secara manual pada kolom ['city\_pop\_category\_1', 'distance\_category\_1']
- **Feature Scaling**: Metode scaling yang digunakan adalah StandardScaler saat melatih model Logistik Regression

## Code Class encoding yang dibuat :

```
class OneHotManual(BaseEstimator, TransformerMixin):  
    def __init__(self, variables):  
        self.variables = variables  
  
    def fit(self, X, y=None):  
        return self  
  
    def transform(self, X):  
        # Lakukan OneHotEncoding pada kolom yang dipilih  
        X_encoded = pd.get_dummies(X[self.variables], drop_first=False)  
  
        # Pastikan untuk menghapus kolom 'gender_F' jika ada  
        if 'gender_F' in X_encoded.columns:  
            X_encoded.drop(columns='gender_F', inplace=True)  
  
        # Ganti labels  
        X_encoded = X_encoded.astype(int)  
        return X_encoded
```

```
class OrdinalManual(BaseEstimator, TransformerMixin):  
    def __init__(self, variables):  
        self.variables = variables  
  
    def fit(self, X, y=None):  
        return self  
  
    def transform(self, X):  
        # Membuat label untuk pengkodean ordinal  
        label_mapping = {  
            'high': 3, 'jauh': 3,  
            'medium': 2, 'sedang': 2,  
            'low': 1, 'dekat': 1  
        }  
  
        # Mengubah setiap kolom sesuai label_mapping  
        for i in self.variables:  
            X[i] = X[i].apply(lambda row: label_mapping.get(row, row))  
  
        return X
```

# Future Improvement

- **Feature Split (Pemisahan Fitur)**: Membuat beberapa kolom waktu dari trans\_date\_trans\_time, seperti:
  - year (tahun)
  - month (bulan)
  - day (hari)
  - day\_of\_week (hari dalam seminggu)
  - hour (jam)
- **Feature Transformation (Transformasi Fitur)**: Mengubah kolom koordinat geografis:
  - lat, long, merch\_lat, dan merch\_long menjadi kolom baru, yaitu distance (jarak) antara pelanggan dan merchant.
- **Feature Categorization (Kategorisasi Fitur)**: Mengkategorikan beberapa kolom numerik ke dalam beberapa kelas:
  - Kolom distance dikategorikan menjadi tiga kelas: dekat, sedang, dan jauh.
  - Kolom city\_pop (populasi kota) dikategorikan menjadi tiga kelas: low, medium, dan high.
- **Feature Engineering (Rekayasa Fitur)**: Menambahkan beberapa kolom baru berdasarkan analisis dan transformasi:
  - risk\_time: Menandai waktu transaksi yang dianggap berisiko.
  - category\_risk: Menandai kategori yang dianggap berisiko untuk fraud.
  - fraud\_category\_count: Menghitung jumlah fraud yang terjadi per kategori transaksi.

# Model Development

## Feature Selection :

### Mutual Information Score

	MI
city	0.004663
job	0.002173
category	0.002154
state	0.000151
gender	0.000029
city_pop_category_1	0.000004
city_pop_category_2	0.000002
distance_category_1	0.000000
distance_category_2	0.000000

### Variance Inflation Factor

	Feature	VIF
0	amt	1.195988
1	city_pop	1.078429
2	distance	4.506820
3	age	4.449302
4	fraud_category_count	1.980725

### Uji ANOVA

Fitur amt: Skor F = 65576.03468408215, p-value = 0.0

Fitur city\_pop: Skor F = 5.9155518320156775, p-value = 0.015007935755578803

Fitur distance: Skor F = 0.21031364935511054, p-value = 0.6465217565861164

Fitur age: Skor F = 194.54856908018786, p-value = 3.2564025166462755e-44

Fitur fraud\_category\_count: Skor F = 5912.727371962211, p-value = 0.0

- Secara keseluruhan, meskipun semua fitur memiliki nilai MI yang rendah, fitur kategorikal city adalah yang paling relevan terhadap fraud, namun kontribusinya tetap kecil, menunjukkan bahwa variabel ini tidak terlalu kuat dalam menentukan fraud.
- Tidak ada multikolinearitas signifikan antara variabel kontinu, meskipun distance dan age memiliki nilai VIF yang lebih tinggi dari variabel lain, tetapi masih dalam batas wajar (VIF < 5).
- Berdasarkan uji ANOVA, fitur kontinu yang paling signifikan dan penting bagi model adalah amt, fraud\_category\_count, dan age.

### Kolom yang digunakan dalam melatih model :

**Kolom Kategorik :** category, gender

city\_pop\_category\_1, distance\_category\_1

**Kolom Diskrit :** risk\_time, category\_risk

**Kolom Continues :** amt, age , fraud\_category\_count

**Kolom Time :** month, day, day\_of\_week, hour

**Kita menggunakan feature kategori dengan threshold < 20 dan beberapa feature yang signifikan dalam menentukan fraud berdasarkan beberapa teknik feature selection**

# Training & Optimization

## Split data menjadi train dan test

```
# split data feature dan target
x = pd.concat([df_train[ohe_colom], df_train[ordinal_colom], df_train[diskrit_colom], df_train[con_colom], df_train[time_cols]], axis=1)
y = df_train['is_fraud']

# Buat train dan test
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=79, stratify=y)
```

Data train : 1037340  
Data test : 259335

## Split data train menjadi train dan validation untuk training model

```
# Memisahkan data menjadi train dan test
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=79, stratify=y_train)
print(f'Data train : {len(X_train)} ')
print(f'Data validation : {len(X_val)} ')
```

Data train : 829872  
Data validation : 207468

# Training & Optimization

## Logistic Regression

```
# Membuat column transformer untuk preprocessing
preprocessor = ColumnTransformer(
    transformers=[
        ('ohe', OneHotManual(variables=ohe_cols), ohe_cols), # One Hot Encoding
        ('ord', OrdinalManual(variables=ordinal_cols), ordinal_cols), # Ordinal Encoding
        ('scaler', StandardScaler(), con_cols) # Scaling
    ],
    remainder='passthrough' # Menjaga kolom lain yang tidak diproses
)

# Membangun pipeline untuk Logistic Regression
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor), # Proses preprocessing
    ('classifier', LogisticRegression()) # Model klasifikasi
])

# Parameter grid untuk GridSearch
param_grid = {
    'classifier_C': [0.01, 0.1, 1, 10, 100], # Hyperparameter untuk Logistic Regression
    'classifier_solver': ['liblinear', 'lbfgs'] # Solver yang digunakan
}

Best parameters found: {'classifier_C': 1, 'classifier_solver': 'liblinear'}
Best cross-validated f1 Score: 0.18955910723298558
```

**Preprocessing :** Logistic Regression dilakukan one hot dan ordinal encoding serta scaling.

**Training :** Menggunakan Gridsearch untuk mencari best parameter dengan cross validation 5 untuk mendapat model yang cenderung stabil kinerjanya

**Scoring :** F1 score, tujuan utamanya adalah keseimbangan antara precision dan recall. F1 score merupakan rata-rata harmonis dari **precision** dan **recall**, sehingga ideal untuk situasi di mana keduanya sama pentingnya, **terutama jika dataset memiliki ketidakseimbangan kelas**, seperti dalam kasus fraud detection.

**Parameter 'classifier\_C': 1 pada Logistic Regression mengatur kekuatan regularisasi.** Nilai C=1 berarti regularisasi sedang, yang menjaga keseimbangan antara memaksimalkan akurasi dan menghindari overfitting.

**Parameter 'classifier\_solver': 'liblinear' menentukan algoritma optimasi.** liblinear cocok untuk dataset kecil hingga sedang, dan mendukung regularisasi L1 atau L2.

# Training & Optimization

## Decision Tree

```
# Membuat column transformer untuk preprocessing
preprocessor = ColumnTransformer(
    transformers=[
        ('ohe', OneHotManual(variables=ohe_cols), ohe_cols), # Menggunakan OneHotManual
        ('ord', OrdinalManual(variables=ordinal_cols), ordinal_cols), # Menggunakan OrdinalManual
    ],
    remainder='passthrough' # Menjaga kolom lain yang tidak diproses
)

# Membangun pipeline untuk Decision Tree
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor), # Proses preprocessing
    ('classifier', DecisionTreeClassifier(random_state=42)) # Model klasifikasi Decision Tree
])

# Parameter grid untuk GridSearch
param_grid = {
    'classifier__max_depth': [3, 5, 10, None], # Hyperparameter max_depth
    'classifier__min_samples_split': [2, 10, 20], # Hyperparameter min_samples_split
    'classifier__min_samples_leaf': [1, 5, 10], # Hyperparameter min_samples_leaf
    'classifier__criterion': ['gini', 'entropy'] # Kriteria pemilihan node
}
```

```
Best parameters found: {'classifier__criterion': 'gini', 'classifier__max_depth': 10, 'classifier__min_samples_leaf': 1, 'classifier__min_samples_split': 20}
Best cross-validated F1 score: 0.8549442142315025
```

**Preprocessing : Decision Tree** dilakukan one hot dan ordinal encoding.

**Training :** Menggunakan Gridsearch untuk mencari best parameter dengan cross validation 5 untuk mendapat model yang cenderung stabil kinerjanya

**Scoring :** F1 score, tujuan utamanya adalah keseimbangan antara precision dan recall. F1 score merupakan rata-rata harmonis dari **precision** dan **recall**, sehingga ideal untuk situasi di mana keduanya sama pentingnya, **terutama jika dataset memiliki ketidakseimbangan kelas**, seperti dalam kasus fraud detection.

- **classifier\_\_criterion: 'gini'** : Menunjukkan fungsi yang digunakan untuk mengukur kualitas pemisahan di setiap node.
- **classifier\_\_max\_depth: 10** : Menentukan kedalaman maksimum dari pohon keputusan..
- **classifier\_\_min\_samples\_leaf: 1** : Menetapkan jumlah minimum sampel yang harus ada di node daun.
- **classifier\_\_min\_samples\_split: 20** : Menentukan jumlah minimum sampel yang diperlukan untuk membagi node.

# Training & Optimization

## Random Forest

```
# Membuat column transformer untuk preprocessing tanpa scaler
preprocessor = ColumnTransformer(
    transformers=[
        ('ohe', OneHotManual(variables=ohe_cols), ohe_cols), # One Hot Encoding
        ('ord', OrdinalManual(variables=ordinal_cols), ordinal_cols) # Ordinal Encoding
        # Scaling dihapus karena tidak dibutuhkan oleh Random Forest
    ],
    remainder='passthrough' # Menjaga kolom lain yang tidak diproses
)

# Membangun pipeline untuk Random Forest
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor), # Proses preprocessing
    ('classifier', RandomForestClassifier(random_state=42)) # Model Random Forest
])

# Parameter grid untuk GridSearch pada Random Forest
param_grid = {
    'classifier__n_estimators': [5, 10, 15], # Jumlah trees di Random Forest
    'classifier__max_depth': [None, 10, 15], # Kedalaman maksimal setiap tree
    'classifier__min_samples_split': [3, 5, 10], # Jumlah minimal samples untuk split
    'classifier__min_samples_leaf': [1, 2, 4] # Jumlah minimal samples untuk leaf
}
```

```
Best parameters found: {'classifier__max_depth': None, 'classifier__min_samples_leaf': 1, 'classifier__min_samples_split': 3, 'classifier__n_estimators': 15}
Best cross-validated F1 score: 0.8385858346662012
```

**Preprocessing : Random Forest** dilakukan one hot dan ordinal encoding.

**Training :** Menggunakan Gridsearch untuk mencari best parameter dengan cross validation 5 untuk mendapat model yang cenderung stabil kinerjanya

**Scoring :** F1 score, tujuan utamanya adalah keseimbangan antara precision dan recall. F1 score merupakan rata-rata harmonis dari **precision dan recall**, sehingga ideal untuk situasi di mana keduanya sama pentingnya, **terutama jika dataset memiliki ketidakseimbangan kelas**, seperti dalam kasus fraud detection.

- **classifier\_\_max\_depth: None** : Menunjukkan bahwa tidak ada batasan pada kedalaman maksimum pohon keputusan.
- **classifier\_\_min\_samples\_leaf: 1** : Menetapkan jumlah minimum sampel yang harus ada di node daun.
- **classifier\_\_min\_samples\_split: 3** : Menentukan jumlah minimum sampel yang diperlukan untuk membagi node
- **classifier\_\_n\_estimators: 15** : Menunjukkan jumlah pohon dalam hutan (ensemble) yang akan dibangun.

# Training & Optimization

## AdaBoost

```
# Membuat column transformer untuk preprocessing
preprocessor = ColumnTransformer(
    transformers=[
        ('ohe', OneHotManual(variables=ohe_cols), ohe_cols), # One Hot Encoding
        ('ord', OrdinalManual(variables=ordinal_cols), ordinal_cols) # Ordinal Encoding
        # Tidak membutuhkan scaling untuk AdaBoost
    ],
    remainder='passthrough' # Menjaga kolom lain yang tidak diproses
)

# Membangun pipeline untuk AdaBoost
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor), # Proses preprocessing
    ('classifier', AdaBoostClassifier(random_state=42)) # Model AdaBoost
])

# Parameter grid untuk GridSearch pada AdaBoost
param_grid = {
    'classifier__n_estimators': [5, 10, 15], # Jumlah estimators pada AdaBoost
    'classifier__learning_rate': [0.01, 0.1, 1.0] # Learning rate untuk boosting
}
```

```
Best parameters found: {'classifier__learning_rate': 1.0, 'classifier__n_estimators': 5}
Best cross-validated F1 score: 0.48444958643310476
```

**Preprocessing : AdaBoost** dilakukan one hot dan ordinal encoding.

**Training :** Menggunakan Gridsearch untuk mencari best parameter dengan cross validation 5 untuk mendapat model yang cenderung stabil kinerjanya

**Scoring :** F1 score, tujuan utamanya adalah keseimbangan antara precision dan recall. F1 score merupakan rata-rata harmonis dari **precision dan recall**, sehingga ideal untuk situasi di mana keduanya sama pentingnya, **terutama jika dataset memiliki ketidakseimbangan kelas**, seperti dalam kasus fraud detection.

- **classifier\_learning\_rate: 1.0**

Parameter ini mengontrol kontribusi setiap estimator (**pohon keputusan**) ke **model akhir**. Nilai 1.0 menunjukkan bahwa model memperhitungkan setiap estimator secara penuh, tanpa mengurangi bobotnya. Jika nilai ini terlalu tinggi, model dapat dengan mudah mengalami overfitting, tetapi dengan nilai yang lebih rendah, model dapat lebih stabil.

- **classifier\_n\_estimators: 5 Decision Tree**

Menunjukkan jumlah estimator (**pohon keputusan**) yang akan digunakan dalam **ensemble**. Dengan nilai 5, model menggunakan lima estimator untuk membangun model akhir. Jumlah yang lebih rendah ini mungkin lebih cepat dalam pelatihan tetapi bisa jadi kurang robust dibandingkan dengan jumlah yang lebih besar, yang sering digunakan untuk meningkatkan akurasi.

# Results

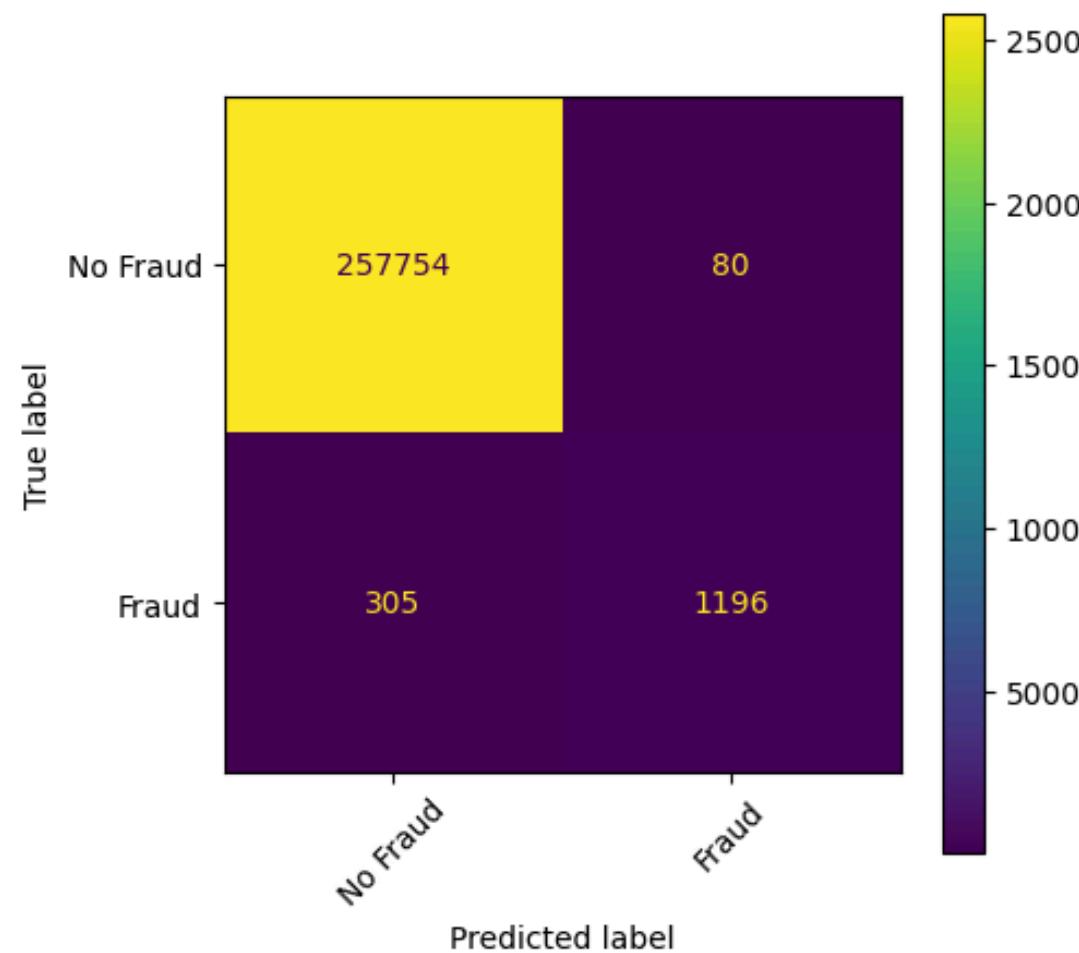
## Evaluasi Model :

Classifier	Train ROC AUC	Val ROC AUC	Test ROC AUC	Train Balanced Accuracy	Val Balanced Accuracy	Test Balanced Accuracy	Train Precision	Val Precision	Test Precision	Train Recall	Val Recall	Test Recall	Train F1 Score	Val F1 Score	Test F1 Score
LogisticRegression	0.553631	0.553930	0.555438	0.553631	0.553930	0.555438	0.636700	0.622010	0.627820	0.107619	0.108243	0.111259	0.184117	0.184397	0.189021
DecisionTreeClassifier	0.907265	0.900352	0.898246	0.907265	0.900352	0.898246	0.957905	0.940371	0.937304	0.814738	0.800999	0.796802	0.880540	0.865108	0.861361
RandomForestClassifier	0.974186	0.880845	0.876993	0.974186	0.880845	0.876993	0.999342	0.962145	0.960951	0.948376	0.761865	0.754164	0.973192	0.850372	0.845091
AdaBoostClassifier	0.686222	0.691838	0.687421	0.686222	0.691838	0.687421	0.686042	0.690583	0.706767	0.373439	0.384679	0.375750	0.483623	0.494118	0.490648

Dari hasil evaluasi beberapa model diatas model Decision Tree lah yang memiliki hasil lebih baik dengan score antara train, validation dan test tidak terlalu jauh dibandingkan dengan model yang lain. Hal ini mengartikan bahwa model tidak mengalami overfitting yang signifikan

# Results

## Confusion Metriks : Decision Tree



**Model mendeteksi 80 transaksi Non Fraud menjadi Fraud dan 305 transaksi Fraud menjadi Non Fraud**

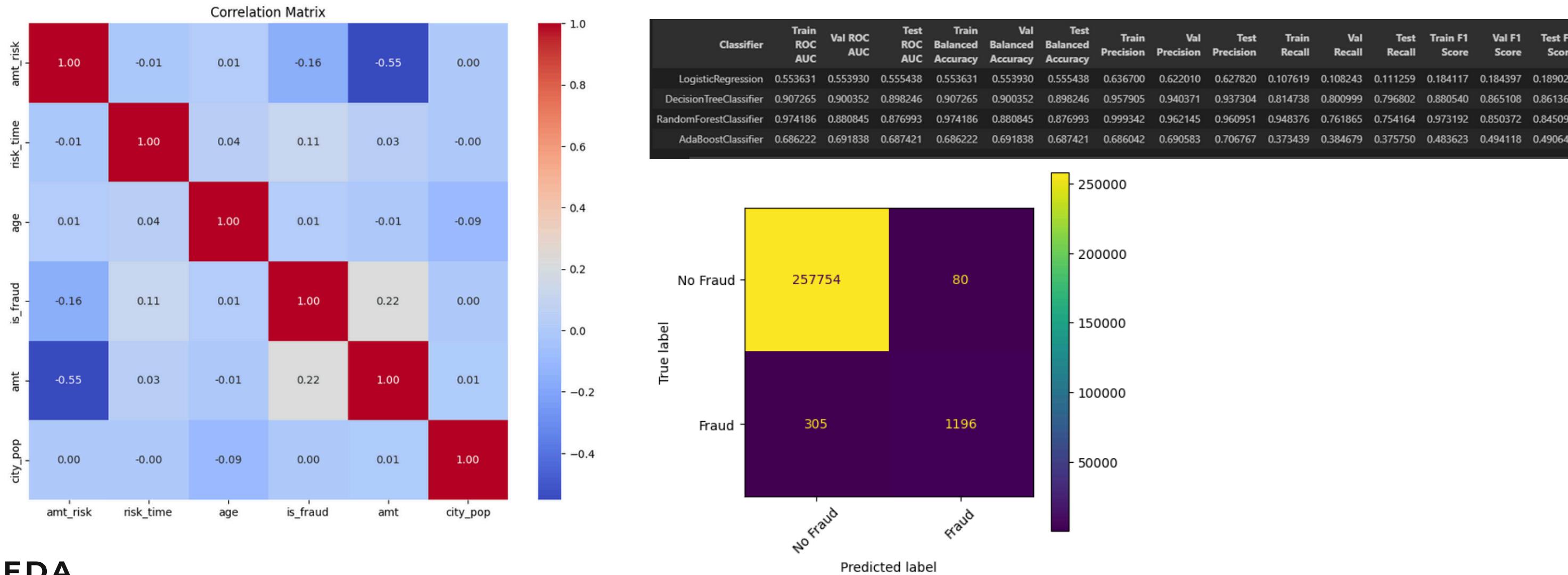
	Classifier	Test ROC AUC	Test Balanced Accuracy	Test Precision	Test Recall	Test F1 Score
1	DecisionTreeClassifier	0.898246	0.898246	0.937304	0.796802	0.861361

# Real-world Application

- Mengidentifikasi transaksi mencurigakan dalam akun bank atau kartu kredit.
- Mendeteksi penipuan dalam transaksi online, seperti pembayaran dengan kartu kredit curian atau pengembalian barang yang tidak sah.
- Menganalisis klaim untuk menemukan klaim palsu atau yang berlebihan. Misalnya, membandingkan klaim dengan data historis untuk mendeteksi ketidakcocokan.
- Mengidentifikasi penyalahgunaan layanan, seperti pencurian identitas untuk mendaftar layanan atau penggunaan telepon yang tidak sah.
- Mendeteksi penipuan dalam pengajuan pajak atau program bantuan sosial, memastikan bahwa dana disalurkan kepada yang berhak.
- Mengawasi klaim asuransi kesehatan untuk mendeteksi penipuan dalam perawatan medis atau klaim yang tidak sah.
- Mendeteksi akun atau aktivitas palsu, seperti bot yang berupaya memanipulasi opini publik atau menyebarkan informasi salah.



# Conclusion



## EDA

- Peristiwa mengakibatkan pergeseran perilaku salah satunya adalah tindakan kriminal.
- Fraud terjadi pada jam yang rawan
- Fraud dilakukan oleh rentang usia 30-50 tahun, dimana usia tersebut merupakan usia keluarga muda hingga keluarga dengan tanggungan biaya kebutuhan anak

## Modeling

Model Decision Tree memiliki hasil paling baik dengan score antara train, validation dan test tidak terlalu jauh dibandingkan dengan model yang lain