

# Adaboost for Fraud Detection

See jupyter notebook [on github](#)  
[This Slide Link](#)

Machine Learning Project 2  
IndonesiaAI ML Batch 7



# Introduction

- **Project Goal:** Develop a model to detect fraudulent transactions.
- **Dataset:** 'fraudTrain.csv' containing transaction data.
- **Approach:** Data analysis, preprocessing, modeling, and evaluation.



# Background

As more and more people rely on digital payments, fraud is getting smarter, sneakier, and harder to catch. Traditional systems? They just can't keep up anymore.

What if we could use machine learning to *really* understand the patterns behind these transactions and spot the shady ones before they do damage?

That's where this project comes in. I took the [fraudTrain.csv](#) dataset — real transaction data — and started digging. The goal? Build a model that doesn't just guess, but learns how fraud behaves, and catches it with speed and precision.



# Libraries and Tools

- **Data Handling:** pandas, numpy
- **Data Splitting:** train\_test\_split (sklearn)
- **Imbalanced Data Handling:** SMOTE (imblearn)
- **Modeling:** AdaBoostClassifier (sklearn)
- **Evaluation:** classification\_report, confusion\_matrix, roc\_auc\_score (sklearn)
- **Visualization:** matplotlib, seaborn
- **Model Serialization:** pickle



# Data Overview

- **Total Columns:** 22
- **Key Columns:**
  - Transaction Details: `trans_date_trans_time`, `amt`
  - User Details: `cc_num`, `first`, `last`, `gender`, `dob`
  - Merchant Details: `merchant`, `category`
  - Geographical Info: `street`, `city`, `state`, `zip`, `lat`, `long`
  - Target Variable: `is_fraud`



# Data Preparation

- **Sampling:** Used 1000 rows initially for quick analysis
- **EDA:**
  - Generated a PDF report (`eda_report__df_train.pdf`)
  - Non-null counts, missing values, distinct values
  - Numerical & categorical summaries
- **Creating New Features:**



# Data Preparation

```
1 df_train.info()

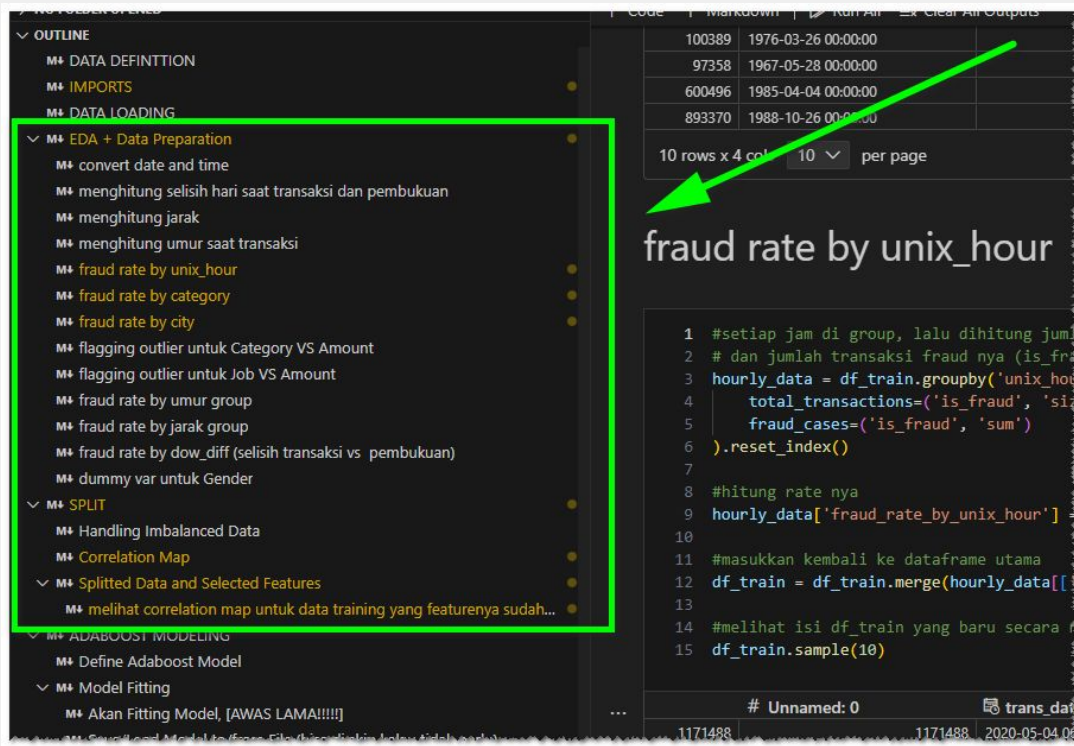
[238]

... <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1296675 entries, 0 to 1296674
Data columns (total 23 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Unnamed: 0             1296675 non-null   int64
1   trans_date_trans_time  1296675 non-null   object
2   cc_num                 1296675 non-null   int64
3   merchant               1296675 non-null   object
4   category               1296675 non-null   object
5   amt                    1296675 non-null   float64
6   first                  1296675 non-null   object
7   last                   1296675 non-null   object
8   gender                 1296675 non-null   object
9   street                 1296675 non-null   object
10  city                   1296675 non-null   object
11  state                  1296675 non-null   object
12  zip                    1296675 non-null   int64
13  lat                    1296675 non-null   float64
14  long                   1296675 non-null   float64
15  city_pop               1296675 non-null   int64
16  job                    1296675 non-null   object
17  dob                   1296675 non-null   object
18  trans_num              1296675 non-null   object
19  unix_time              1296675 non-null   int64
...
21  merch_long             1296675 non-null   float64
22  is_fraud                1296675 non-null   int64
dtypes: float64(5), int64(6), object(12)
memory usage: 227.5+ MB

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```



# Data Preparation



The screenshot shows a Jupyter Notebook with the following structure:

- OUTLINE
  - DATA DEFINITION
  - IMPORTS
  - DATA LOADING
  - EDA + Data Preparation**
    - convert date and time
    - menghitung selisih hari saat transaksi dan pembukuan
    - menghitung jarak
    - menghitung umur saat transaksi
    - fraud rate by unix\_hour**
    - fraud rate by category
    - fraud rate by city
    - flagging outlier untuk Category VS Amount
    - flagging outlier untuk Job VS Amount
    - fraud rate by umur group
    - fraud rate by jarak group
    - fraud rate by dow\_diff (selisih transaksi vs pembukuan)
    - dummy var untuk Gender
  - SPLIT
    - Handling Imbalanced Data
    - Correlation Map
    - Splitted Data and Selected Features**
      - melihat correlation map untuk data training yang featurenya sudah...
  - ADABOOST MODELING
    - Define Adaboost Model
  - Model Fitting
    - Akan Fitting Model, [AWAS LAMA!!!!]

The notebook displays a table of transaction data:

id	unix_timestamp	amount	is_fraud
100389	1976-03-26 00:00:00		
97358	1967-05-28 00:00:00		
600496	1985-04-04 00:00:00		
893370	1988-10-26 00:00:00		

Below the table, it shows the text "10 rows x 4 columns" and "10 per page".

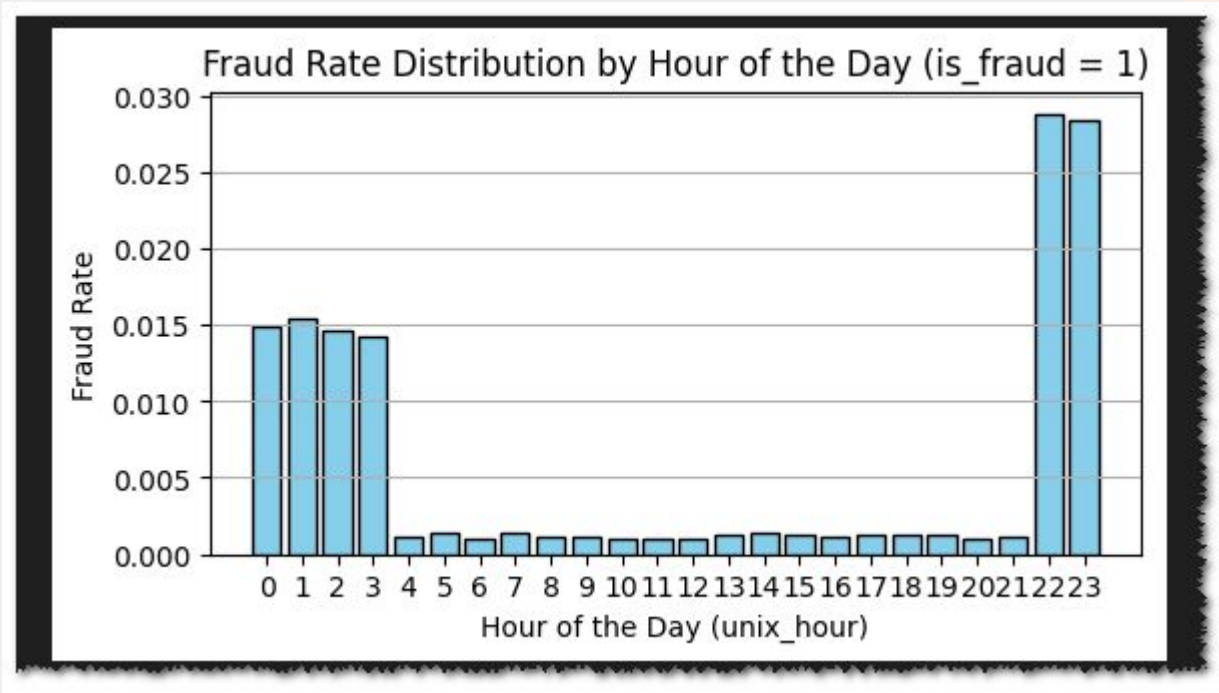
The code cell titled "fraud rate by unix\_hour" contains the following Python code:

```
1 #setiap jam di group, lalu dihitung jumlah
2 # dan jumlah transaksi fraud nya (is_fraud)
3 hourly_data = df_train.groupby('unix_hour')
4     .agg(total_transactions=('is_fraud', 'sum'))
5     .reset_index()
6
7
8 #hitung rate nya
9 hourly_data['fraud_rate_by_unix_hour'] =
10
11 #masukkan kembali ke dataframe utama
12 df_train = df_train.merge(hourly_data[['unix_hour', 'fraud_rate_by_unix_hour']],
13
14 #melihat isi df_train yang baru secara acak
15 df_train.sample(10))
```

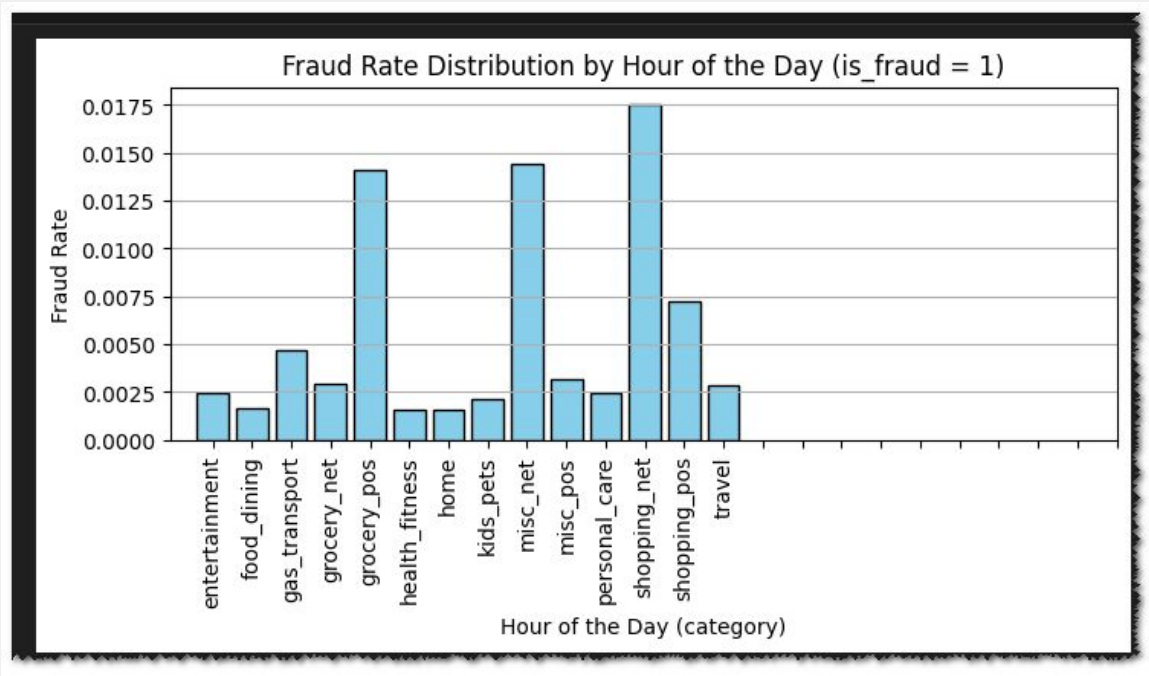




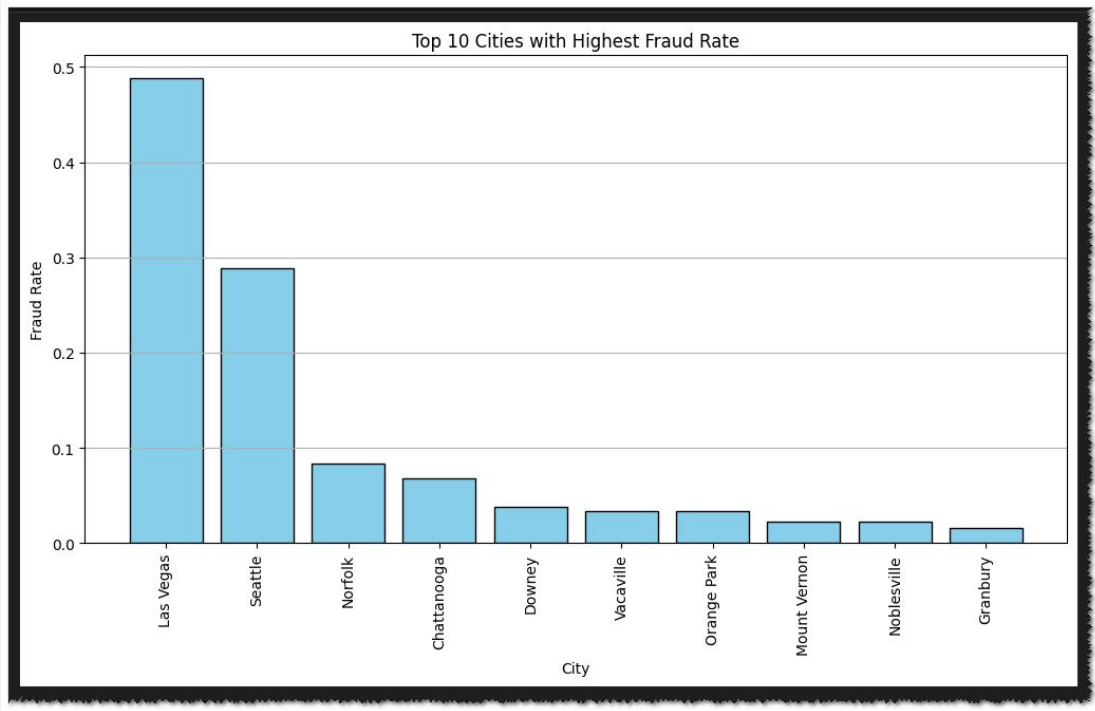
# Data Preparation



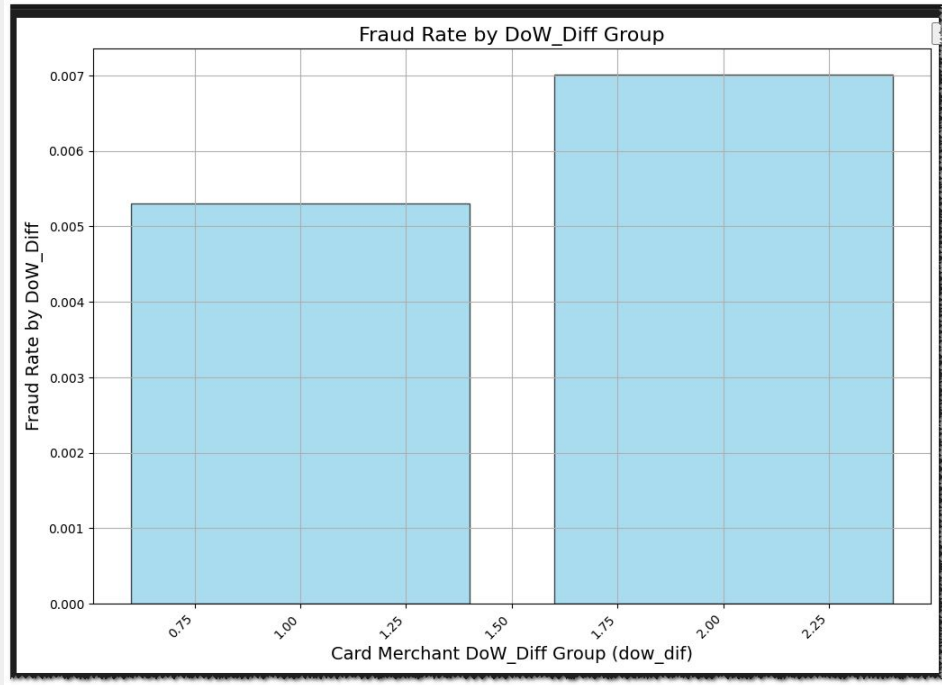
# Data Preparation



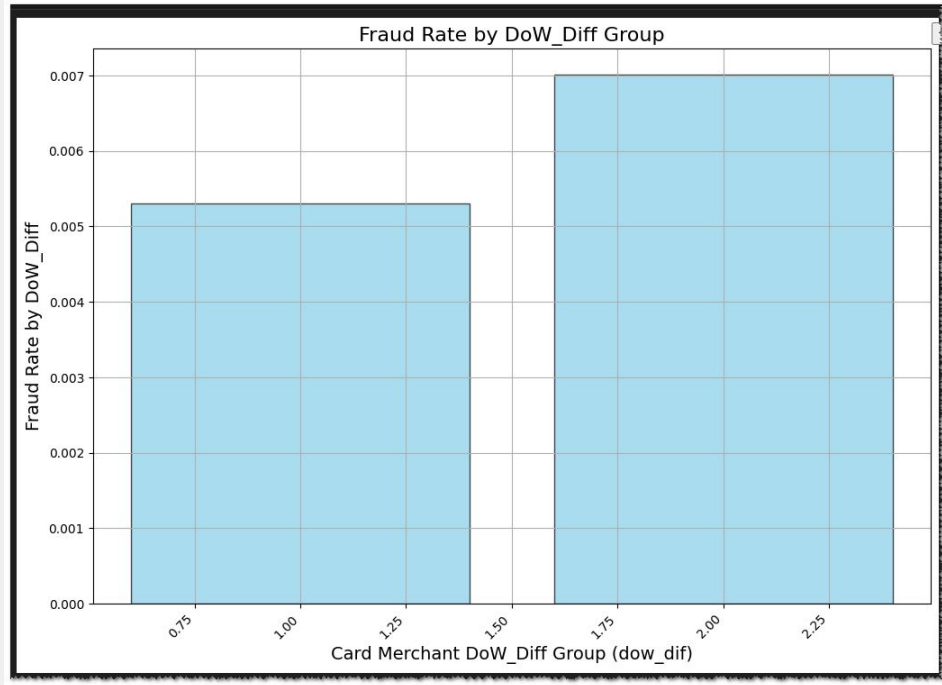
# Data Preparation



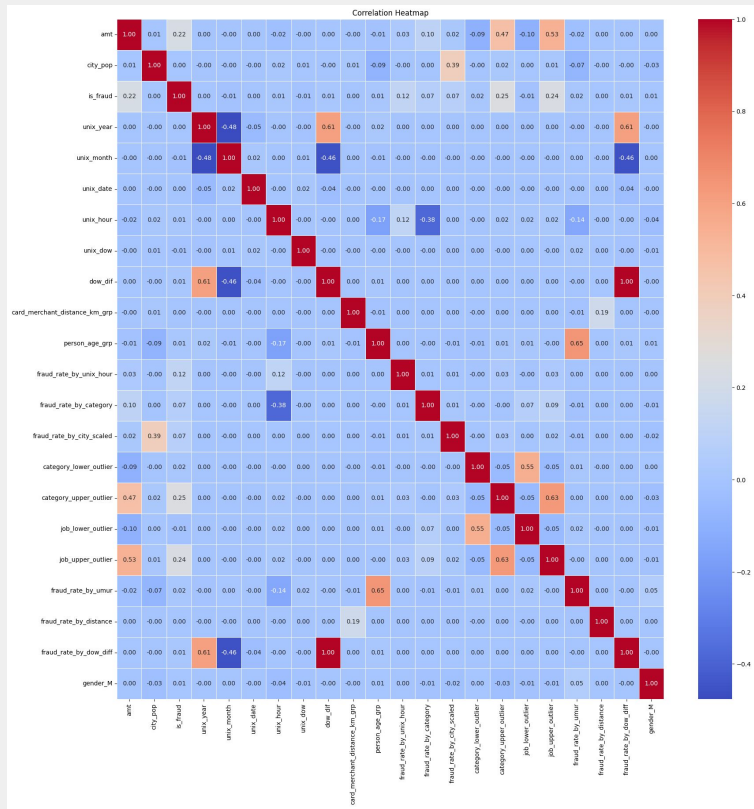
# Data Preparation



# Data Preparation



# Data Preparation



# Modeling

- **Models Used:**
  - a. `AdaBoostClassifier`
- **Handling Imbalanced Data:** SMOTE
- **Model Evaluation Metrics:**
  - a. Classification report
  - b. Confusion matrix
  - c. ROC AUC score
- **Feature Importance**
- **Run Model with Important Feature Only**



# Modeling

## Define Adaboost Model

```
1 #
2 base_estimator = RandomForestClassifier() # tidak ada parameter di set , saya ambil default saja semua
3
4 # Initialize AdaBoost with base estimator and n_estimators (number of trees)
5 model = AdaBoostClassifier(
6     estimator=base_estimator,
7     n_estimators=100,
8     random_state=42)
9
```

## Model Fitting

Akan Fitting Model, [AWAS LAMA!!!!!!]

skip saja kalau udah pernah , langsung load saja di bawah

```
1 X = df_train_balanced_smote.drop('is_fraud', axis=1)
2 y = df_train_balanced_smote['is_fraud']
```





# Modeling

[291] + Code + Markdown

1 feature\_importance\_df

[292]

...

	Feature	# Importance
10	fraud_rate_by_unix_hour	0.3050826510215114
0	amt	0.22111923750812024
14	category_upper_outlier	0.11910041584439623
11	fraud_rate_by_category	0.08741715006619467
16	job_upper_outlier	0.08566922496922108
5	unix_hour	0.04470641657439381
19	fraud_rate_by_dow_diff	0.017820550076904564
12	fraud_rate_by_city_scaled	0.017639410602988962
17	fraud_rate_by_umur	0.015927352302554394
1	city_pop	0.012412966599913604
13	category_lower_outlier	0.010854914583336766
6	unix_dow	0.010032869604976059
9	person_age_grp	0.008114118126266324
7	dow_dif	0.007207350363013228
20	gender_M	0.007009923678833331
18	fraud_rate_by_distance	0.006704709253933432
8	card_merchant_distance_km_grp	0.00595865717346875
3	unix_month	0.005759184413537581
4	unix_date	0.004428203334890267
15	job_lower_outlier	0.0037194998192841638
2	unix_year	0.003315194082261369

21 rows x 2 cols 50 per page

« < Page 1 of 1 > »



# Modeling

## ADABOOST MODELING (LAGI)

(kali ini akan dicoba menggunakan feature yang kontribusinya lebih dari 1% saja berdasar feature importance)

### Feature Selection based on Feature Importance

```
1 #memilih feature2 yang kontribusinya >= 1% saja
2 list_of_selected_field2 = feature_importance_df[feature_importance_df['Importance']>=0.01]
3 list_of_selected_field2
```

[299]

...

	Feature	# Importance
10	fraud_rate_by_unix_hour	0.3050826510215114
0	amt	0.22111923750812024
14	category_upper_outlier	0.11910041584439623
11	fraud_rate_by_category	0.08741715006619467
16	job_upper_outlier	0.08566922496922108
5	unix_hour	0.04470641657439381
19	fraud_rate_by_dow_diff	0.017820550076904564
12	fraud_rate_by_city_scaled	0.017639410602988962
17	fraud_rate_by_umur	0.015927352302554394
1	city_pop	0.012412966599913604

12 rows x 2 cols 10 ▾ per page

« < Page 1 of 2 > »



# Results

- **Key Metrics:**
  - Precision, Recall, F1-Score
  - ROC AUC



# Results

Classification Report untuk Test Data menggunakan model dengan feature pilihan manual

	precision	recall	f1-score	support
0	1.00	1.00	1.00	257815
1	0.94	0.77	0.85	1520
accuracy			1.00	259335
macro avg	0.97	0.89	0.92	259335
weighted avg	1.00	1.00	1.00	259335

Classification Report untuk Test Data menggunakan model dengan important feature saja

	precision	recall	f1-score	support
0	1.00	1.00	1.00	257815
1	0.91	0.80	0.85	1520
accuracy			1.00	259335
macro avg	0.95	0.90	0.92	259335
weighted avg	1.00	1.00	1.00	259335



# Conclusion

- **Model Performance:** Second Try with Important Feature only has similar result with faster processing (fewer feature being use)
- **Next Steps can be done:**
  - a. Improve feature engineering
  - b. Hyperparameter tuning
  - c. Implement in a real-time fraud detection system





# Thank You