

# DATA SCIENCE

## ANALISIS DATA HISTORIS BITCOIN MENGGUNAKAN METODE DATA SCIENCE UNTUK MENGIDENTIFIKASI POLA HARGA DAN HUBUNGAN ANTARVARIABEL

# LATAR BELAKANG

Bitcoin mengalami pertumbuhan pesat seiring perkembangan teknologi blockchain dan meningkatnya minat masyarakat terhadap kripto. Sebagai mata uang kripto pertama, Bitcoin memiliki peran penting dalam ekosistem keuangan digital, namun harganya yang fluktuatif menimbulkan tantangan bagi investor dan peneliti. Proyek ini melakukan analisis data Bitcoin melalui tahap Data Collection, Data Visualization, Advanced Preprocessing, dan Statistical Analysis untuk memahami tren pasar, aktivitas jaringan, dan faktor-faktor yang mempengaruhi nilai Bitcoin.

# METODE PENGERJAAN PROYEK ANALISIS DATA BITCOIN

## Lingkungan Pemrograman:

Menggunakan Google Colab karena berbasis cloud, mendukung Python, dan tidak perlu instalasi tambahan.

## Data Collection

- Dataset berasal dari Kaggle: “Bitcoin Price Prediction” oleh @aditeloo.
- Format CSV, siap pakai, tidak perlu scraping.
- Dataset berisi harga, volume transaksi, dan parameter teknis jaringan Bitcoin.

# METODE PENGERJAAN PROYEK ANALISIS DATA BITCOIN

## Data Processing

- **Data Cleaning:**

- Mengecek dan menangani missing values

```
[ ] #Cek Missing Value
print("\n==== CEK MISSING VALUES ===")
missing_values = df.isnull().sum()
print(missing_values[missing_values > 0])

[ ] === CEK MISSING VALUES ===
Date 1140
btc_total_bitcoins 27
btc_trade_volume 21
btc_blocks_size 29
btc_median_confirmation_time 12
btc_difficulty 16
btc_transaction_fees 10
dtype: int64
```

```
[ ] #Handling Missing Value
print("==== HANDLING MISSING VALUES ===")

# Select only numeric columns for imputation
df_num = df.select_dtypes(include=np.number)

df_filled = df_num.fillna(df_num.mean())

print("\nMissing values after imputation:\n", df_filled.isnull().sum())

[ ] === HANDLING MISSING VALUES ===

Missing values after imputation:
btc_market_price 0
btc_total_bitcoins 0
btc_market_cap 0
btc_trade_volume 0
btc_blocks_size 0
btc_avg_block_size 0
btc_n_orphaned_blocks 0
btc_n_transactions_per_block 0
btc_median_confirmation_time 0
btc_hash_rate 0
btc_difficulty 0
btc_miners_revenue 0
btc_transaction_fees 0
btc_cost_per_transaction_percent 0
btc_cost_per_transaction 0
btc_n_unique_addresses 0
btc_n_transactions 0
btc_n_transactions_total 0
btc_n_transactions_excluding_popular 0
btc_n_transactions_excluding_chains_longer_than_100 0
btc_output_volume 0
btc_estimated_transaction_volume 0
btc_estimated_transaction_volume_usd 0
dtype: int64
```

Kode pertama memeriksa apakah ada nilai yang hilang (missing values) di dataset df. df.isnull().sum() menghitung jumlah nilai kosong di tiap kolom, dan hanya menampilkan kolom yang memiliki nilai kosong saja.

Kode kedua menangani missing values pada kolom numerik dengan metode imputasi rata-rata (mean). Kolom numerik diambil menggunakan select\_dtypes, lalu fillna(df\_num.mean()) mengganti nilai kosong dengan rata-rata kolom tersebut. Setelah itu dicek kembali, dan semua missing value sudah terisi.

# METODE PENGERJAAN PROYEK ANALISIS DATA BITCOIN

## Data Processing

- Data Cleaning:

- Mengecek dan menangani missing Outlier.

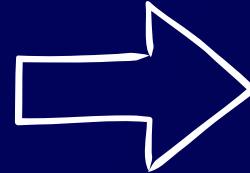
```
[1] #Cek Outliers

print("\n==== CEK OUTLIERS (IQR METHOD) ===")
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
outlier_counts = {}

for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    outliers = df[(df[col] < (Q1 - 1.5 * IQR)) | (df[col] > (Q3 + 1.5 * IQR))]
    if len(outliers) > 0:
        outlier_counts[col] = len(outliers)

print(pd.Series(outlier_counts).sort_values(ascending=False))

[2]
==== CEK OUTLIERS (IQR METHOD) ===
btc_n_orphaned_blocks          607
btc_cost_per_transaction_percent 434
btc_difficulty                  410
btc_cost_per_transaction         409
btc_hash_rate                     388
btc_transaction_fees              387
btc_trade_volume                  360
btc_estimated_transaction_volume_usd 323
btc_market_cap                     298
btc_market_price                   274
btc_miners_revenue                  260
btc_output_volume                   114
btc_n_transactions_total             95
btc_median_confirmation_time            91
btc_estimated_transaction_volume           63
btc_blocks_size                      58
btc_n_unique_addresses                 21
btc_n_transactions_excluding_chains_longer_than_100 2
btc_n_transactions_excluding_popular      1
btc_n_transactions                         1
dtype: int64
```



```
[1] #Handling Outlier
outlier_counts_after = {}

for col in numeric_cols:
    Q1 = df_clean[col].quantile(0.25)
    Q3 = df_clean[col].quantile(0.75)
    IQR = Q3 - Q1
    outliers = df_clean[(df_clean[col] < (Q1 - 1.5 * IQR)) | (df_clean[col] > (Q3 + 1.5 * IQR))]
    if len(outliers) > 0:
        outlier_counts_after[col] = len(outliers)

if len(outlier_counts_after) == 0:
    print("Tidak ada outlier terdeteks")
else:
    print(pd.Series(outlier_counts_after).sort_values(ascending=False))

[2]
Tidak ada outlier terdeteks
```

Kode pertama digunakan untuk mendeteksi outlier pada setiap kolom numerik dalam dataset menggunakan metode IQR (Interquartile Range).

Kode kedua menunjukkan hasil setelah pembersihan data (df\_clean), yaitu outlier yang telah dihapus. Jika tidak ada outlier yang tersisa, akan muncul pesan "Tidak ada outlier terdeteksi", menandakan dataset kini sudah bersih dari nilai ekstrem dan siap untuk analisis lebih lanjut.

# METODE PENGERJAAN PROYEK ANALISIS DATA BITCOIN

## Data Processing

- **Data Transformation:**

- Melakukan Feature Scaling dengan Standardization

```
#Cek Feature Scaling
feature_scaling = df_filled[numERIC_COLs].describe().T[['min', 'max', 'mean', 'std']]
print(feature_scaling)

      min \
btc_market_price          0.000000e+00
btc_total_bitcoins         2.043200e+06
btc_market_cap              0.000000e+00
btc_trade_volume            0.000000e+00
btc_blocks_size             0.000000e+00
btc_avg_block_size          2.163350e-04
btc_n_orphaned_blocks       0.000000e+00
btc_n_transactions_per_block 1.000000e+00
btc_median_confirmation_time 0.000000e+00
btc_hash_rate                2.250000e-05
btc_difficulty               2.527738e+00
btc_miners_revenue           0.000000e+00
btc_transaction_fees          0.000000e+00
btc_cost_per_transaction_percent 1.365306e-01
btc_cost_per_transaction                 0.000000e+00
```

Baris ini menampilkan ringkasan statistik untuk setiap kolom numerik sebelum dilakukan scaling, termasuk nilai minimum, maksimum, rata-rata (mean), dan standar deviasi (std). Tujuannya untuk melihat perbedaan skala antar fitur dan memastikan apakah standardisasi diperlukan.

```
#Feature scaling (Standardization).
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_filled[numERIC_COLs])
df_scaled = pd.DataFrame(scaled_data, columns=numERIC_COLs)

after_scaling = df_scaled.describe().T[['min', 'max', 'mean', 'std']]
print(after_scaling)

      min        max \
btc_market_price     -0.364104    8.096752
btc_total_bitcoins   -2.265254   1.274316
btc_market_cap        -0.347722   8.110249
btc_trade_volume      -0.253966  18.118030
btc_blocks_size        -0.818233   2.740980
btc_avg_block_size     -0.991623   2.152207
btc_n_orphaned_blocks  -0.432334   7.880076
btc_n_transactions_per_block -0.972777   2.974796
btc_median_confirmation_time -1.511282   8.105762
btc_hash_rate          -0.425521   6.965869
btc_difficulty          -0.432486   6.567587
btc_miners_revenue      -0.385318   8.997629
btc_transaction_fees     -0.514274  12.215656
btc_cost_per_transaction_percent -0.037813  50.241320
btc_cost_per_transaction      -0.712972   7.161651
btc_n_unique_addresses     -0.927218   4.208543
btc_n_transactions          -0.981556   3.740531
btc_n_transactions_total     -0.825749   2.755258
btc_n_transactions_excluding_popular -0.906517   3.620083
btc_n_transactions_excluding_chains_longer_than... -0.984518   3.670692
btc_output_volume            -0.684685  19.497765
btc_estimated_transaction_volume -0.759195  20.957304
btc_estimated_transaction_volume_usd -0.349053   9.583236

      mean        std \
btc_market_price     -3.912142e-17  1.000172
btc_total_bitcoins   3.129713e-16  1.000172
btc_market_cap        -7.824283e-17  1.000172
btc_trade_volume      1.956071e-17  1.000172
btc_blocks_size        -1.564857e-16  1.000172
btc_avg_block_size      0.000000e+00  1.000172
btc_n_orphaned_blocks  -3.912142e-17  1.000172
```

Kode ini melakukan feature scaling menggunakan standardisasi. StandardScaler mengubah setiap kolom numerik sehingga memiliki mean = 0 dan standar deviasi = 1, sehingga semua fitur berada pada skala yang sama. Hasil scaling disimpan di df\_scaled dan ringkasan statistiknya (min, max, mean, std) ditampilkan untuk memastikan transformasi berhasil.

# METODE PENGERJAAN PROYEK ANALISIS DATA BITCOIN

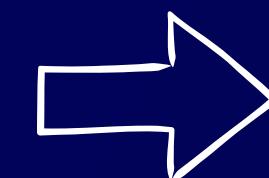
## Data Processing

- Data Transformation :

- Visualisasi Perbandingan Setelah Melakukan Scaling

```
#Cek Feature Scaling
feature_scaling = df_filled[numeric_cols].describe().T[['min', 'max', 'mean', 'std']]
print(feature_scaling)
```

	min	max
btc_market_price	0.000000e+00	1.949868e+04
btc_total_bitcoins	2.043200e+06	1.683769e+07
btc_market_cap	0.000000e+00	3.270000e+11
btc_trade_volume	0.000000e+00	5.352016e+09
btc_blocks_size	0.000000e+00	1.544446e+05
btc_avg_block_size	2.163350e-04	1.118327e+00
btc_n_orphaned_blocks	0.000000e+00	7.000000e+00
btc_n_transactions_per_block	1.000000e+00	2.722625e+03
btc_median_confirmation_time	0.000000e+00	4.773333e+01
btc_hash_rate	2.250000e-05	2.160975e+07
btc_difficulty	2.527738e+00	2.600000e+12
btc_miners_revenue	0.000000e+00	5.319158e+07
btc_transaction_fees	0.000000e+00	1.495946e+03
btc_cost_per_transaction_percent	1.365306e-01	8.857143e+04
btc_cost_per_transaction	0.000000e+00	1.616861e+02
btc_n_unique_addresses	1.100000e+02	
btc_n_transactions	1.180000e+02	
btc_n_transactions_total	4.124000e+04	
btc_n_transactions_excluding_popular	1.180000e+02	
btc_n_transactions_excluding_chains_longer_than...	1.180000e+02	
btc_output_volume	6.150000e+03	
btc_estimated_transaction_volume	7.000000e+00	
btc_estimated_transaction_volume_usd	0.000000e+00	
	mean	std
btc_market_price	1.949868e+04	1.912142e-17
btc_total_bitcoins	1.683769e+07	3.129713e-16
btc_market_cap	3.270000e+11	7.824283e-17
btc_trade_volume	5.352016e+09	1.000172
btc_blocks_size	1.544446e+05	1.000172
btc_avg_block_size	1.118327e+00	1.000172
btc_n_orphaned_blocks	7.000000e+00	1.000172
btc_n_transactions_per_block	2.722625e+03	7.824283e-17
btc_median_confirmation_time	4.773333e+01	7.824283e-17
btc_hash_rate	2.160975e+07	1.000172
btc_difficulty	2.600000e+12	1.000172
btc_miners_revenue	5.319158e+07	1.000172
btc_transaction_fees	1.495946e+03	1.000172
btc_cost_per_transaction_percent	8.857143e+04	1.000172
btc_cost_per_transaction	1.616861e+02	1.000172



```
#Feature scaling (Standardization).
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_filled[numeric_cols])
df_scaled = pd.DataFrame(scaled_data, columns=numerical_cols)

after_scaling = df_scaled.describe().T[['min', 'max', 'mean', 'std']]
print(after_scaling)
```

	min	max	mean	std
btc_market_price	-0.364104	8.096752	-3.912142e-17	1.000172
btc_total_bitcoins	-2.265254	1.274316	3.129713e-16	1.000172
btc_market_cap	-0.347722	8.110249	-7.824283e-17	1.000172
btc_trade_volume	-0.253966	18.118030	1.000172	1.000172
btc_blocks_size	-0.818233	2.740908	1.000172	1.000172
btc_avg_block_size	-0.991623	2.152207	1.000172	1.000172
btc_n_orphaned_blocks	-0.432334	7.880076	1.000172	1.000172
btc_n_transactions_per_block	-0.972777	2.974796	1.000172	1.000172
btc_median_confirmation_time	-1.511282	8.105762	1.000172	1.000172
btc_hash_rate	-0.425521	6.965869	1.000172	1.000172
btc_difficulty	-0.432486	6.567587	1.000172	1.000172
btc_miners_revenue	-0.385318	8.997629	1.000172	1.000172
btc_transaction_fees	-0.514274	12.215656	1.000172	1.000172
btc_cost_per_transaction_percent	-0.037813	50.241320	1.000172	1.000172
btc_cost_per_transaction	-0.712972	7.161651	1.000172	1.000172
btc_n_unique_addresses	-0.927218	4.208543	1.000172	1.000172
btc_n_transactions	-0.981556	3.740531	1.000172	1.000172
btc_n_transactions_total	-0.825749	2.755258	1.000172	1.000172
btc_n_transactions_excluding_popular	-0.906517	3.620083	1.000172	1.000172
btc_n_transactions_excluding_chains_longer_than...	-0.984518	3.670692	1.000172	1.000172
btc_output_volume	-0.684685	19.497765	1.000172	1.000172
btc_estimated_transaction_volume	-0.759195	20.957304	1.000172	1.000172
btc_estimated_transaction_volume_usd	-0.349053	9.583236	1.000172	1.000172
	mean	std	mean	std
btc_market_price	-3.912142e-17	1.000172	-3.912142e-17	1.000172
btc_total_bitcoins	3.129713e-16	1.000172	3.129713e-16	1.000172
btc_market_cap	-7.824283e-17	1.000172	-7.824283e-17	1.000172
btc_trade_volume	1.956071e-17	1.000172	1.956071e-17	1.000172
btc_blocks_size	-1.564857e-16	1.000172	-1.564857e-16	1.000172
btc_avg_block_size	0.000000e+00	1.000172	0.000000e+00	1.000172
btc_n_orphaned_blocks	3.912142e-17	1.000172	3.912142e-17	1.000172
btc_n_transactions_per_block	-7.824283e-17	1.000172	-7.824283e-17	1.000172
btc_median_confirmation_time	7.824283e-17	1.000172	7.824283e-17	1.000172
btc_hash_rate	-3.912142e-17	1.000172	-3.912142e-17	1.000172
btc_difficulty	0.000000e+00	1.000172	0.000000e+00	1.000172
btc_miners_revenue	0.000000e+00	1.000172	0.000000e+00	1.000172
btc_transaction_fees	-7.824283e-17	1.000172	-7.824283e-17	1.000172
btc_cost_per_transaction_percent	0.000000e+00	1.000172	0.000000e+00	1.000172
btc_cost_per_transaction	-9.780354e-17	1.000172	-9.780354e-17	1.000172
btc_n_unique_addresses	-7.824283e-17	1.000172	-7.824283e-17	1.000172
btc_n_transactions	-1.564857e-16	1.000172	-1.564857e-16	1.000172
btc_n_transactions_total	0.000000e+00	1.000172	7.824283e-17	1.000172
btc_n_transactions_excluding_popular	7.824283e-17	1.000172	7.824283e-17	1.000172
btc_n_transactions_excluding_chains_longer_than...	0.000000e+00	1.000172	0.000000e+00	1.000172
btc_output_volume	-5.868213e-17	1.000172	-5.868213e-17	1.000172
btc_estimated_transaction_volume	5.868213e-17	1.000172	5.868213e-17	1.000172
btc_estimated_transaction_volume_usd	-7.824283e-17	1.000172	-7.824283e-17	1.000172

Sebelum scaling, setiap fitur memiliki skala dan rentang nilai berbeda. Setelah scaling dengan StandardScaler, semua fitur distandarisasi sehingga mean  $\approx 0$  dan standar deviasi  $\approx 1$ , membuat tiap fitur seimbang untuk analisis atau pemodelan.

Pemilihan standardization (standarisasi) dilakukan karena fitur-fitur numerik pada dataset Bitcoin memiliki skala dan satuan yang berbeda-beda.. Jika tidak distandarisasi, fitur dengan skala besar akan mendominasi analisis atau model, sedangkan fitur kecil hampir tidak berpengaruh.

# METODE PENGERJAAN PROYEK ANALISIS DATA BITCOIN

## Data Processing

- **Data Transformation:**

- Mengubah Kolom Date ke dalam Format Waktu
- 



#Mengubah kolom Date ke format waktu

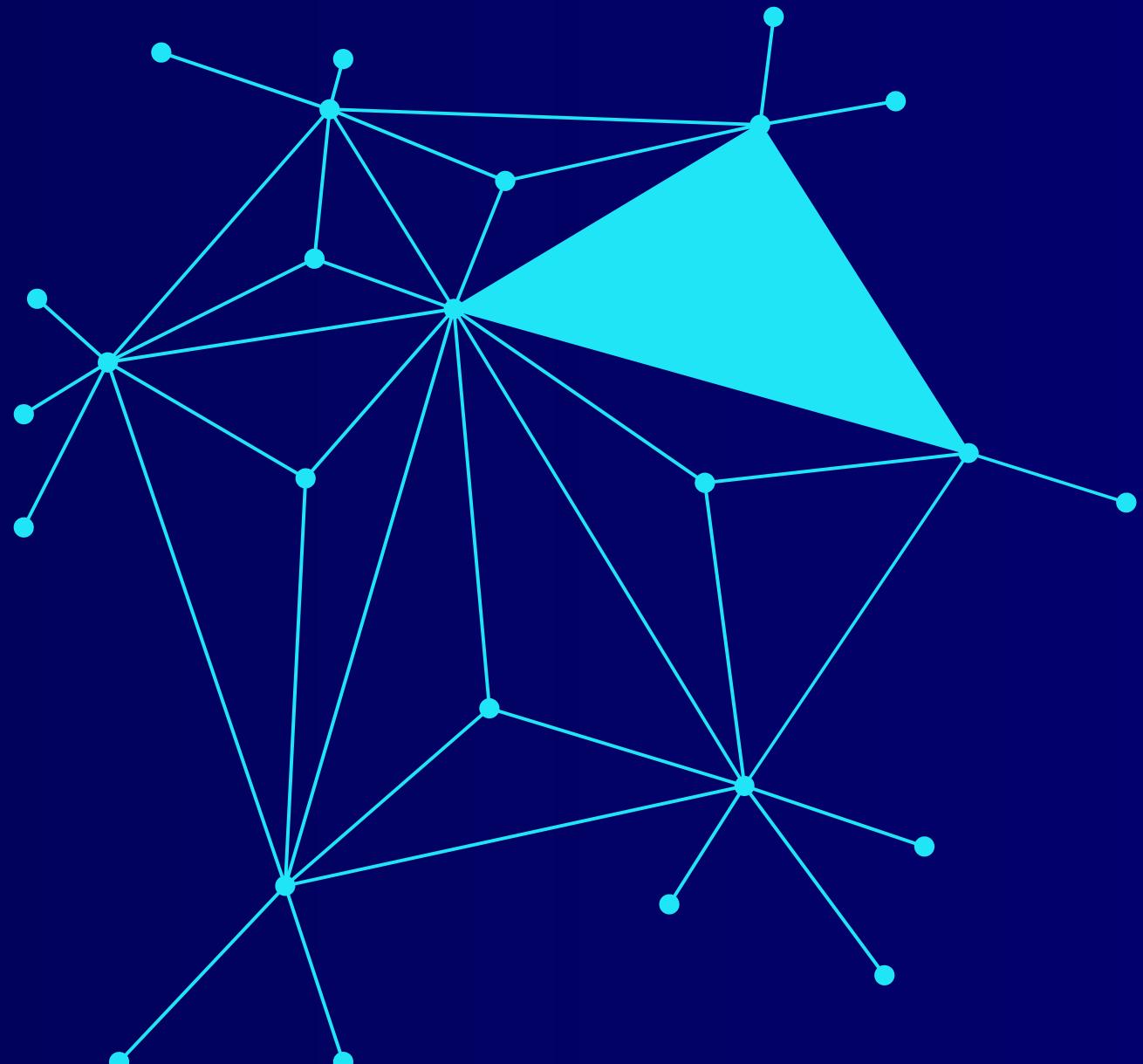
```
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df = df.dropna(subset=['Date'])
```

Kode ini digunakan untuk mengubah kolom `Date` ke format waktu yang valid dan menghapus baris dengan nilai tanggal tidak sah. Langkah ini berguna agar data waktu dapat diproses dengan benar dalam analisis berbasis tanggal, seperti melihat tren atau pola waktu.

Pemilihan standardization (standarisasi) dilakukan karena fitur-fitur numerik pada dataset Bitcoin memiliki skala dan satuan yang berbeda-beda.. Jika tidak distandarisasi, fitur dengan skala besar akan mendominasi analisis atau model, sedangkan fitur kecil hampir tidak berpengaruh.

## DATA INTEGRATION

Pada proyek ini, tidak dilakukan proses penggabungan data (data integration) karena dataset yang digunakan telah tersedia dalam kondisi lengkap dan terintegrasi dari sumber aslinya, yaitu Kaggle. Dataset tersebut sudah disusun secara sistematis oleh penyedia data, di mana seluruh informasi yang diperlukan telah tergabung dalam satu file utama tanpa adanya pemisahan antar tabel atau sumber data lain.



# DATA INTEGRATION

- **Feature Reduction**

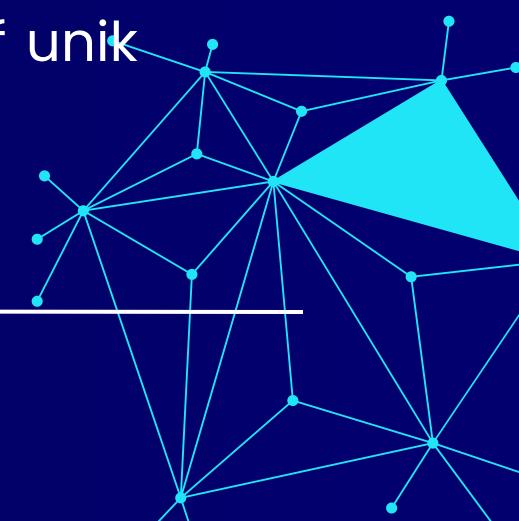
## 1. Mengecek Korelasi Antar Fitur (Kolom)

```
[ ] #Cek Feature Rediction

plt.figure(figsize=(15,8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title("Heatmap Korelasi Antar Fitur")
plt.show()

threshold = 0.9
high_corr = np.where(corr_matrix > threshold)
high_corr_pairs = [
    (corr_matrix.index[x], corr_matrix.columns[y])
    for x, y in zip(*high_corr)
    if x != y and x < y
]
if len(high_corr_pairs) > 0:
    for pair in high_corr_pairs:
        print(f"{pair[0]} <-> {pair[1]}")
else:
    print("Tidak ada fitur yang terlalu berkorelasi (redundansi rendah).")
```

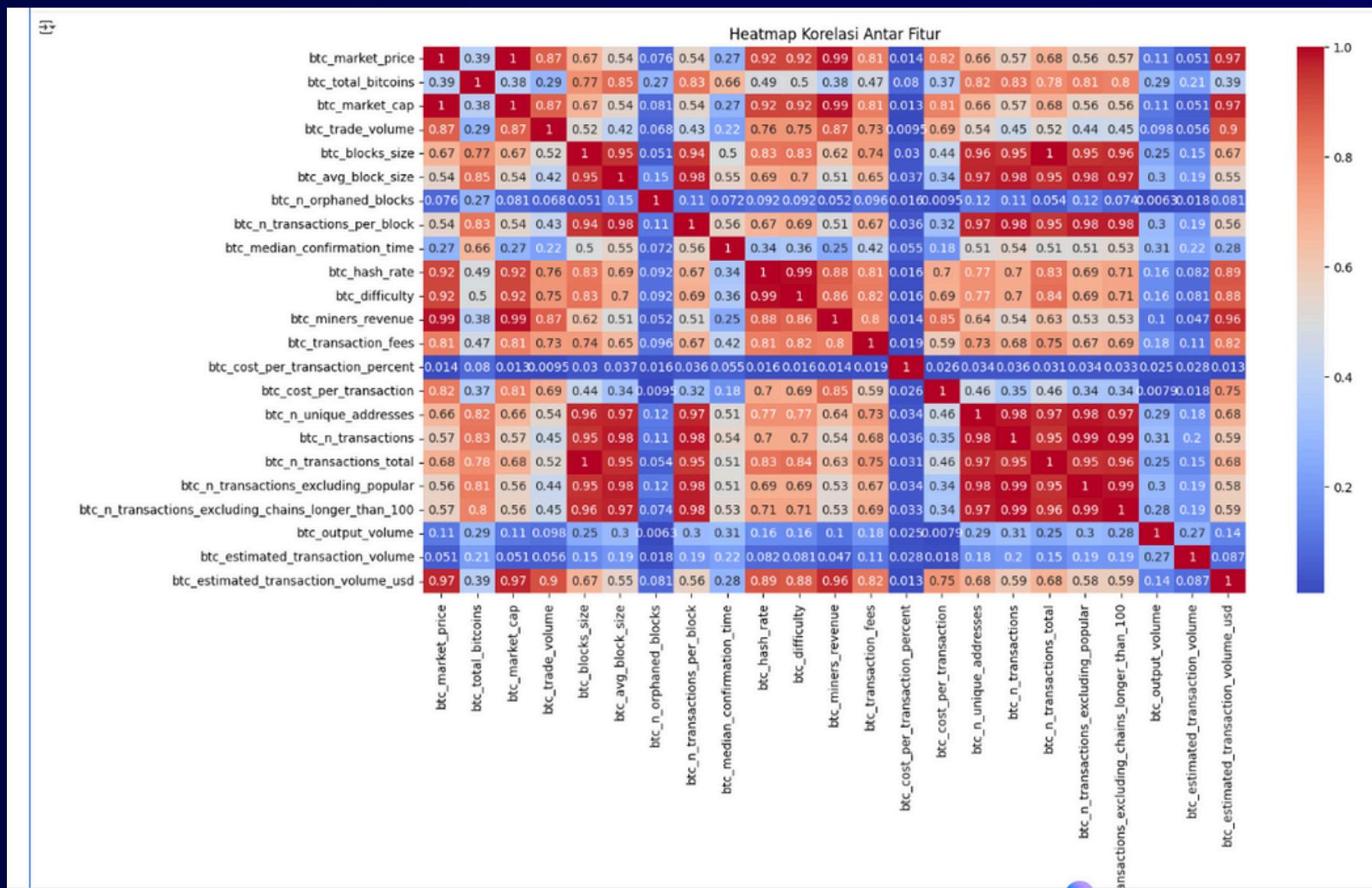
Kode ini bertujuan untuk mengurangi fitur yang berlebihan dengan cara mengecek apakah ada fitur yang hampir sama (sangat berkorelasi). Pertama, matriks korelasi divisualisasikan supaya kita bisa melihat hubungan antar fitur. Lalu, kode mencari semua pasangan fitur yang korelasinya lebih dari 0,9, mengabaikan korelasi fitur dengan dirinya sendiri dan menghindari duplikasi. Jika ada pasangan seperti ini, kode menampilkannya supaya bisa dipertimbangkan untuk dihapus. Jika tidak ada, artinya fitur-fitur data relatif unik dan tidak berlebihan.



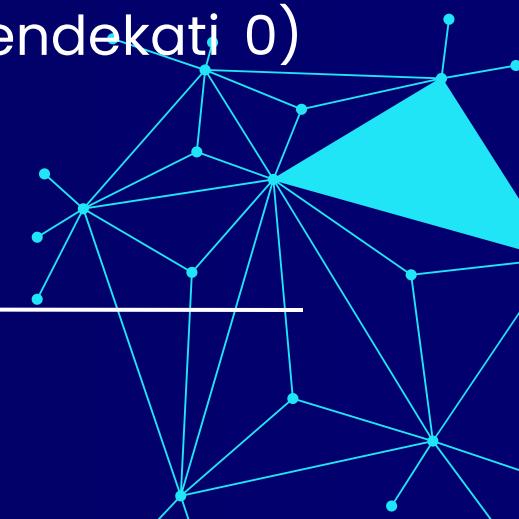
# DATA INTEGRATION

- **Feature Reduction**

1. Mengecek Korelasi Antar Fitur (Kolom)



Visualisasi matriks untuk menunjukkan koefisien korelasi antar setiap pasangan fitur dalam data. Setiap kotak pada grid mewakili hubungan antara dua variabel, di mana angka di dalamnya (koefisien korelasi) dan warnanya menunjukkan kekuatan dan arah hubungan tersebut. Skala warna di sebelah kanan (dari biru tua ke merah tua) adalah kuncinya: warna merah tua (mendekati 1.0) menunjukkan korelasi positif yang sangat kuat, sedangkan warna biru tua (mendekati -1.0) menunjukkan korelasi negatif yang kuat, dan warna pucat/putih (mendekati 0) berarti tidak ada korelasi linier.



# DATA INTEGRATION

- **Feature Reduction**

## 2. Mengurangi Fitur yang memiliki korelasi tinggi

```
[ ] ⏎ # Hilangkan satu fitur dari setiap pasangan yang berkorelasi tinggi
Features_to_drop = set()
for col1, col2 in high_corr_pairs:
    if col1 not in features_to_drop and col2 not in features_to_drop:
        # Tentukan mana yang akan dihilangkan - untuk lebih sederhana, mari kita hilangkan yang kedua
        features_to_drop.add(col2)

df_reduced = df_scaled.drop(columns=list(features_to_drop))

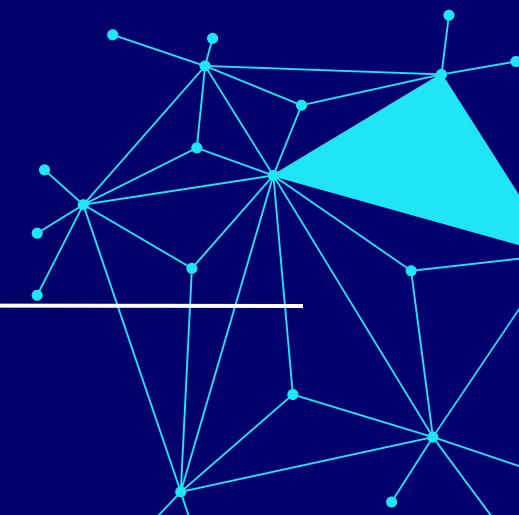
print(f"Original number of features: {df_scaled.shape[1]}")
print(f"Number of features after reduction: {df_reduced.shape[1]}")
display(df_reduced.head())

```

→ Original number of features: 23  
Number of features after reduction: 11

	btc_market_price	btc_total_bitcoins	btc_trade_volume	btc_blocks_size	btc_n_orphaned_blocks	btc
0	-0.364104	-2.265254	-0.253966	-0.818233	-0.432334	
1	-0.364104	-2.262515	-0.253966	-0.818233	-0.432334	
2	-0.364104	-2.260373	-0.253966	-0.818233	-0.432334	
3	-0.364104	-2.257718	-0.253966	-0.818233	-0.432334	
4	-0.364104	-2.255158	-0.253966	-0.818233	-0.432334	

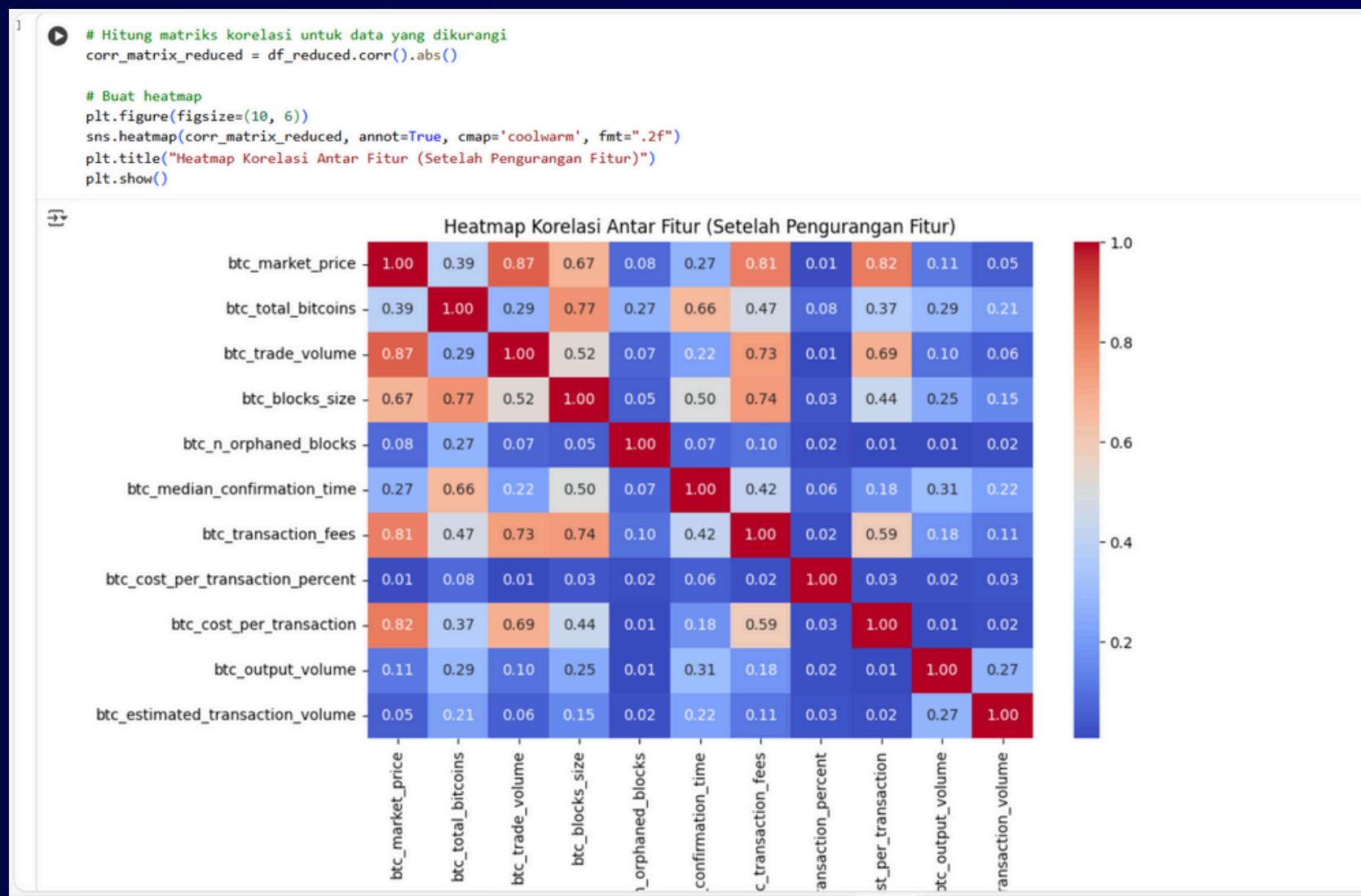
Kode ini menghapus satu fitur dari setiap pasangan fitur yang sangat berkorelasi untuk mengurangi redundansi. Fitur yang dihapus disimpan di `features_to_drop`, lalu dataset baru `df_reduced` dibuat tanpa fitur-fitur tersebut. Terakhir, kode menampilkan jumlah fitur sebelum dan sesudah reduksi serta contoh data awal dari dataset yang sudah dikurangi.



# DATA INTEGRATION

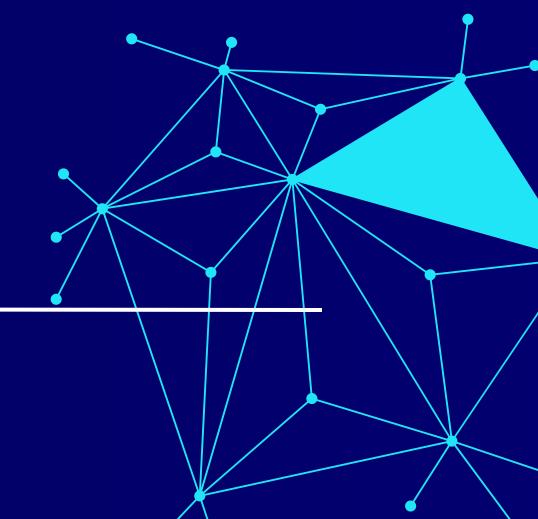
- **Feature Reduction**

## 2. Visualisasi Mengurangi Fitur yang memiliki korelasi tinggi



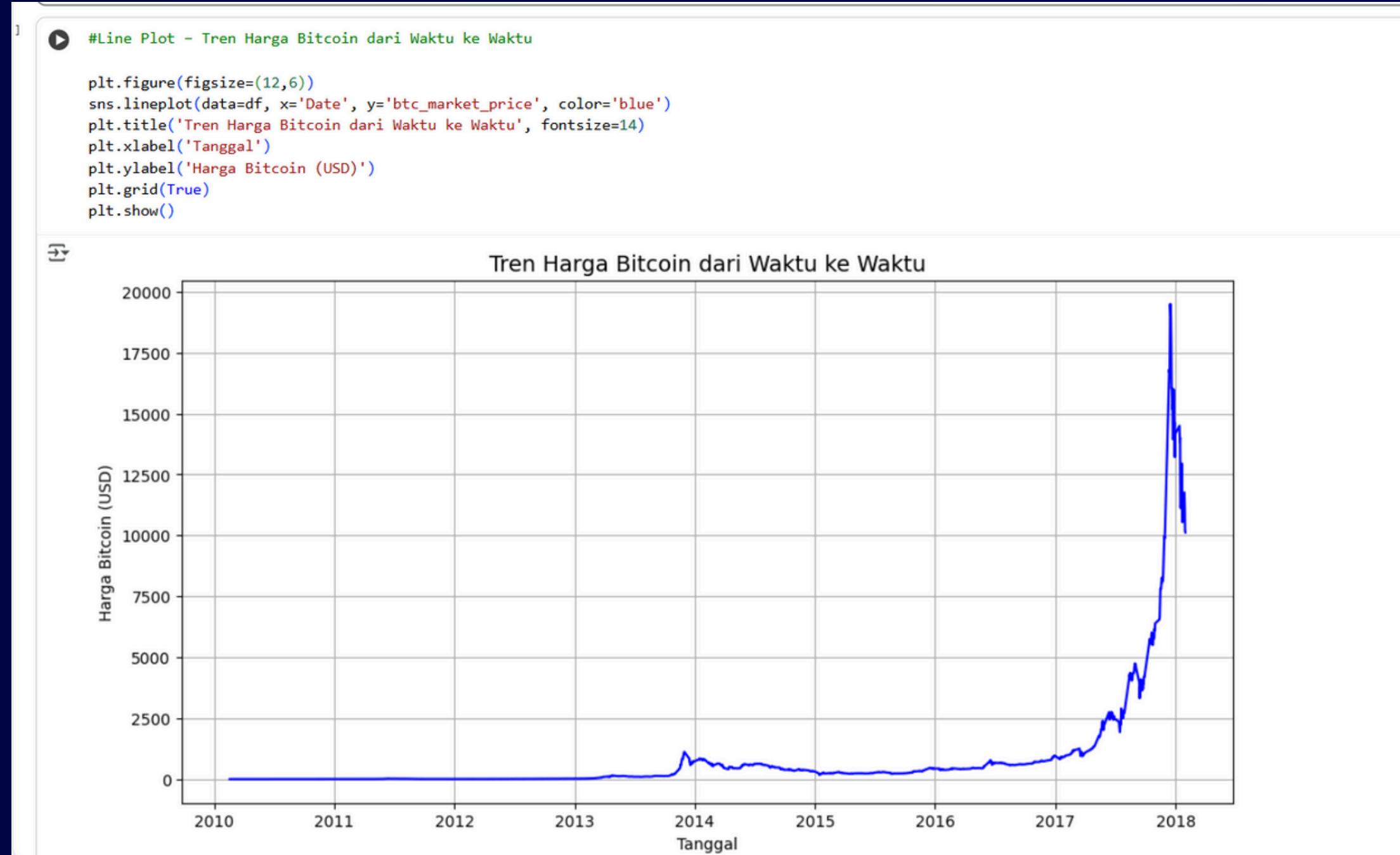
Kode ini menghitung kembali korelasi antar fitur setelah pengurangan fitur dan menampilkan heatmap baru untuk melihat apakah korelasi tinggi sudah berkurang, memastikan redundansi data lebih rendah.

Heatmap ini menunjukkan tingkat korelasi antar fitur setelah pengurangan fitur; warna lebih terang berarti korelasi tinggi, warna gelap berarti rendah, sehingga memudahkan melihat fitur yang masih saling terkait.



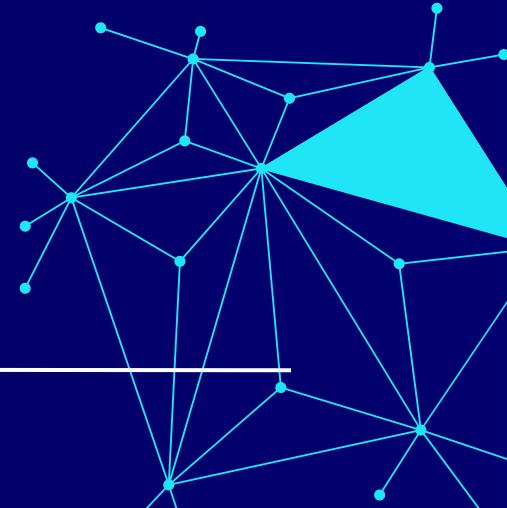
# DATA VISUALIZATION

## 1. Visualisasi Line Plot – Tren Harga Bitcoin dari Waktu ke Waktu



Kode ini membuat line plot untuk menampilkan pergerakan harga Bitcoin dari waktu ke waktu. Visualisasi garis dipilih karena mudah menunjukkan tren dan perubahan harga secara kontinu, sehingga tren naik, turun, atau stabil dapat terlihat dengan jelas sepanjang periode waktu.

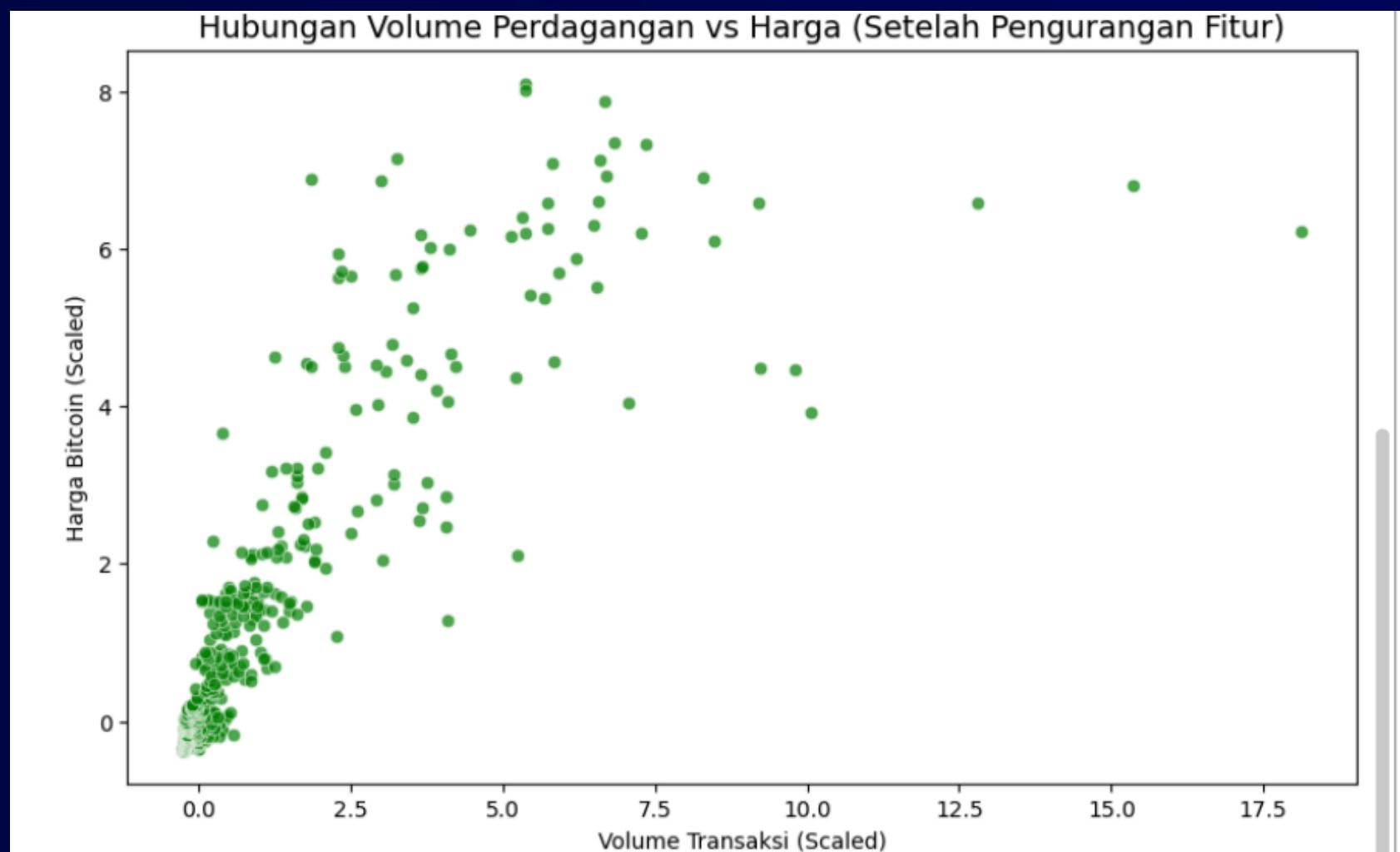
Heatmap ini menampilkan korelasi antar fitur dalam dataset. Setiap kotak menunjukkan tingkat hubungan antara dua fitur, dengan warna yang lebih gelap atau terang menandakan korelasi tinggi atau rendah.



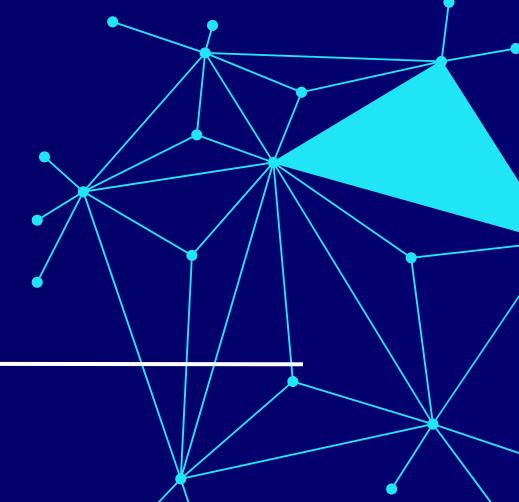
# DATA VISUALIZATION

## 2. visualisasi Hubungan antara Volume Perdagangan dan Harga Bitcoin Setelah Pengurangan Fitur

```
# Hubungan antara Volume Perdagangan dan Harga Bitcoin Setelah Pengurangan Fitur
plt.figure(figsize=(10, 6))
sns.scatterplot(x='btc_trade_volume', y='btc_market_price', data=df_reduced, alpha=0.7, color='green')
plt.title('Hubungan Volume Perdagangan vs Harga (Setelah Pengurangan Fitur)', fontsize=14)
plt.xlabel('Volume Transaksi (Scaled)')
plt.ylabel('Harga Bitcoin (Scaled)')
plt.show()
```

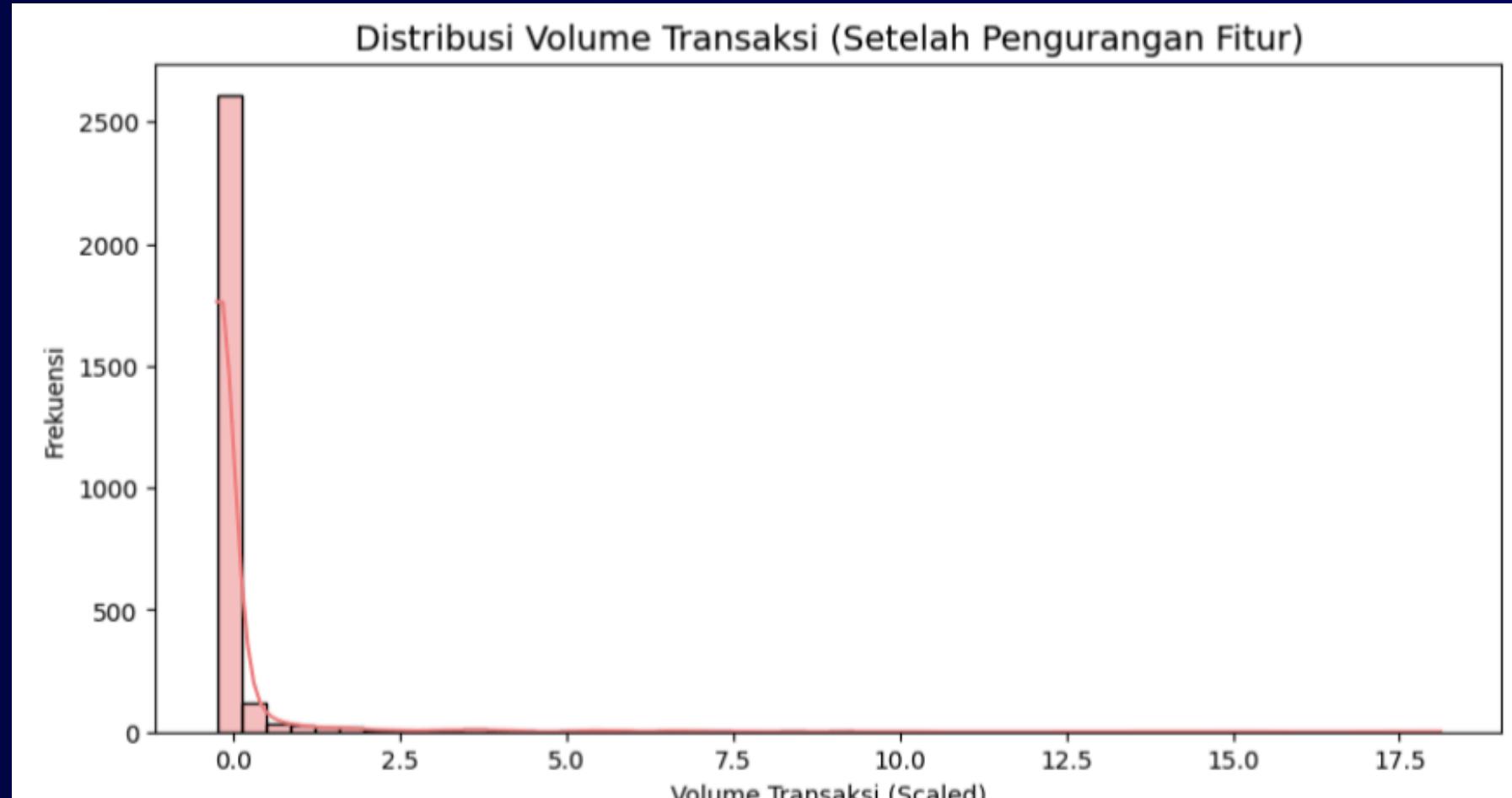


Kode ini membuat scatter plot untuk melihat hubungan antara volume perdagangan dan harga Bitcoin setelah pengurangan fitur. Titik-titik menunjukkan sebaran data, alpha=0.7 membuat titik agak transparan agar tumpang tindih terlihat. Visualisasi ini dipilih karena mudah menunjukkan pola korelasi atau tren antara dua variabel: titik naik bersama berarti hubungan positif, turun bersama berarti negatif, dan sebaran acak berarti tidak ada korelasi kuat.

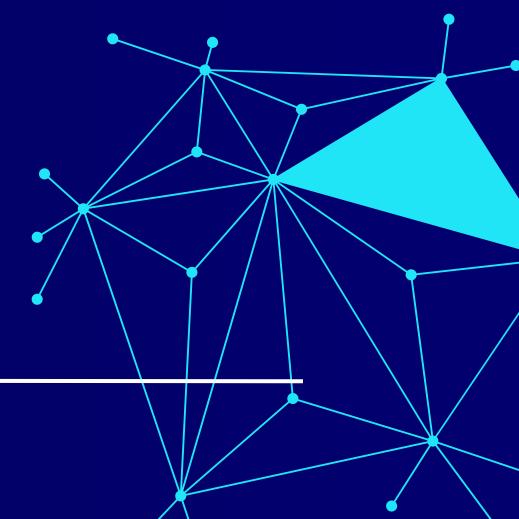


# DATA VISUALIZATION

## 3. Distribusi Volume Transaksi Setelah Pengurangan Fitur



Kode ini membuat histogram untuk menampilkan distribusi volume transaksi Bitcoin setelah pengurangan fitur. `bins=50` membagi data menjadi 50 interval, dan `kde=True` menambahkan kurva kepadatan sehingga tren distribusi lebih jelas. Visualisasi ini dipilih karena mudah menunjukkan seberapa sering nilai tertentu muncul dan bentuk distribusinya—apakah simetris, condong, atau memiliki outlier. Cara membacanya: tinggi tiap batang menunjukkan frekuensi nilai pada interval tertentu, sedangkan kurva KDE memberikan gambaran umum bentuk distribusi data.



# STATISTICAL ANALYSIS

## 1. Mengambil Parametrik yang Penting

```
[ ] ➔ df_stat = df_stat.select_dtypes(include=['float64', 'int64']).dropna()  
x = df_stat['btc_market_price']  
y = df_stat['btc_trade_volume']
```

Kode ini merupakan langkah persiapan data. Intinya, kode ini pertama-tama membersihkan DataFrame df\_stat dengan cara memilih hanya kolom-kolom bertipe numerik (yaitu float64 dan int64). Segera setelah itu, metode .dropna() diterapkan untuk menghapus baris manapun dari kolom-kolom numerik tersebut yang memiliki nilai kosong. Setelah DataFrame bersih, kode ini mengisolasi dua kolom spesifik untuk analisis: kolom btc\_market\_price ditugaskan ke variabel x, dan kolom btc\_trade\_volume ditugaskan ke variabel y. Hasilnya adalah dua variabel (x dan y) yang siap digunakan untuk analisis statistik atau visualisasi, seperti uji korelasi atau scatter plot.

# STATISTICAL ANALYSIS

Melakukan Uji Parametrik dan non-Parametrik

- Uji Parametrik dengan Menggunakan Pearson Correlation

```
[1] import math
from scipy import stats

# UJI PARAMETRIK: Pearson Correlation
pearson_corr, pearson_p = stats.pearsonr(x, y)

effect_size_pearson = abs(pearson_corr)

# Hitung Confidence Interval untuk korelasi (95%)
# Fisher transformation
def fisher_ci(r, n, alpha=0.05):
    z = np.arctanh(r)
    se = 1 / math.sqrt(n - 3)
    z_crit = stats.norm.ppf(1 - alpha/2)
    lo_z, hi_z = z - z_crit * se, z + z_crit * se
    lo, hi = np.tanh((lo_z, hi_z))
    return lo, hi

ci_low, ci_high = fisher_ci(pearson_corr, len(df_stat))

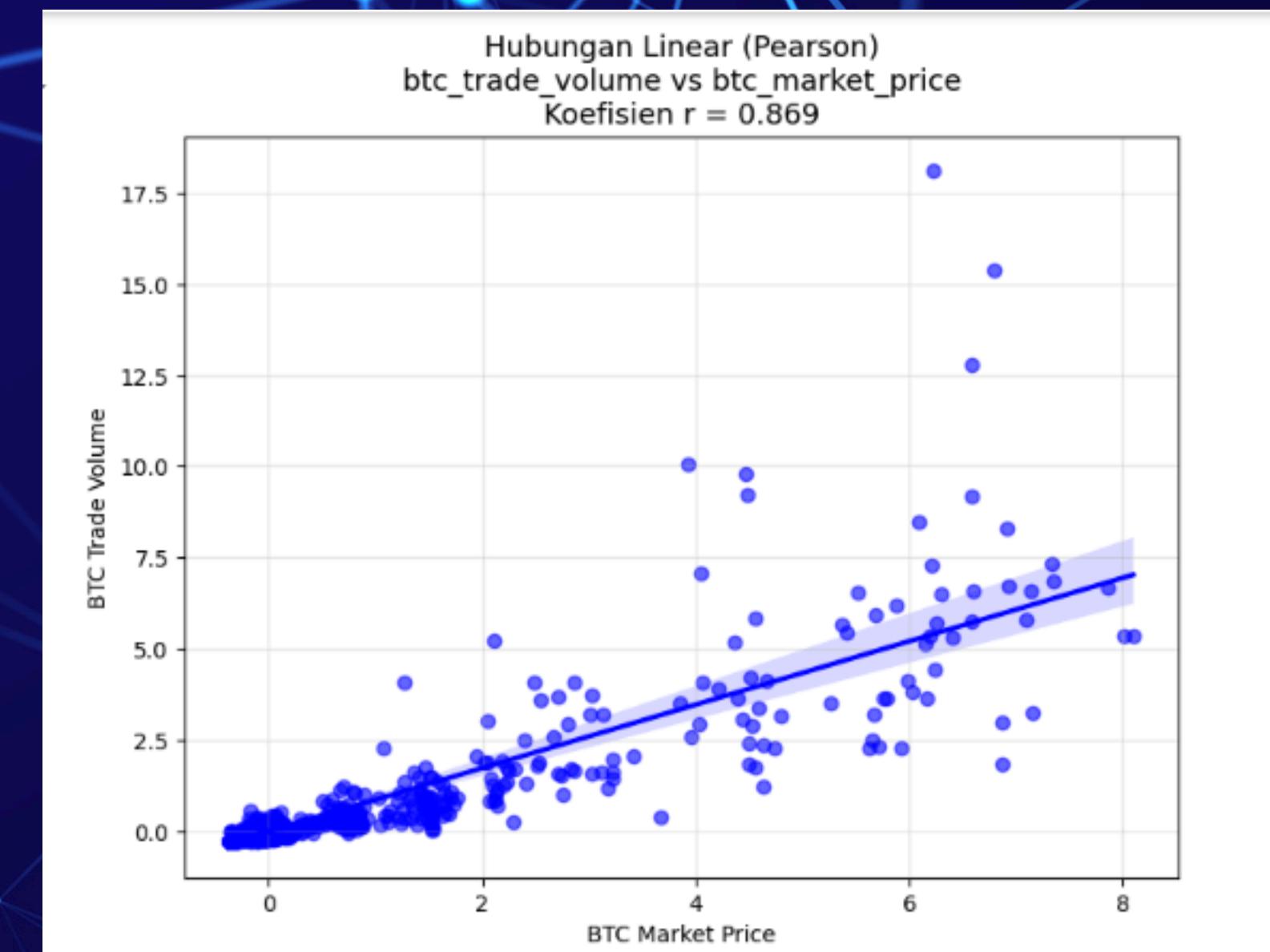
print("== PARAMETRIC TEST: Pearson Correlation ==")
print(f"Variabel: btc_market_price vs btc_trade_volume")
print(f"Koefisien Korelasi (r) = {pearson_corr:.4f}")
print(f"p-value = {pearson_p:.4e}")
print(f"Effect size (|r|) = {effect_size_pearson:.3f}")
print(f"95% Confidence Interval = [{ci_low:.3f}, {ci_high:.3f}]")

# Interpretasi sederhana
if pearson_p < 0.05:
    print("Interpretasi: Terdapat hubungan linear yang signifikan secara statistik antara kedua variabel (p < 0.05).")
else:
    print("Interpretasi: Tidak terdapat hubungan linear yang signifikan antara kedua variabel (p ≥ 0.05).")

# UJI NON-PARAMETRIK: Spearman Correlation
spearman_corr, spearman_p = stats.spearmanr(x, y)

print("\n== NON-PARAMETRIC TEST: Spearman Correlation ==")
print(f"Variabel: btc_market_price vs btc_trade_volume")
print(f"Koefisien Korelasi (ρ) = {spearman_corr:.4f}")
print(f"p-value = {spearman_p:.4e}")
print(f"Effect size (|ρ|) = {abs(spearman_corr):.3f}")

if spearman_p < 0.05:
    print("Interpretasi: Terdapat hubungan monoton signifikan antara kedua variabel (p < 0.05).")
else:
    print("Interpretasi: Tidak terdapat hubungan monoton signifikan antara kedua variabel (p ≥ 0.05).")
```



# STATISTICAL ANALYSIS

Melakukan Uji Parametrik dan non-Parametrik

- **Uji Parametrik dengan Menggunakan Pearson Correlation**

Kode ini menghitung korelasi antara harga Bitcoin (`btc_market_price`) dan volume transaksi (`btc_trade_volume`) menggunakan dua metode:

- Parametrik (Pearson): Mengukur hubungan linear antar variabel. Dihitung koefisien korelasi  $r$ , p-value, ukuran efek, dan interval kepercayaan 95% menggunakan transformasi Fisher. Hasil menunjukkan apakah hubungan linear signifikan secara statistik.
- Non-parametrik (Spearman): Mengukur hubungan monoton antar variabel, cocok jika data tidak normal atau memiliki outlier. Dihitung koefisien  $\rho$ , p-value, dan ukuran efek. Hasil menunjukkan apakah terdapat hubungan monoton signifikan.

# STATISTICAL ANALYSIS

## Melakukan Uji Parametrik dan non-Parametrik

Uji Parametrik dengan Menggunakan Spearman Correlation

```
#UJI NON-PARAMETRIK: Spearman Correlation

spearman_corr, spearman_p = stats.spearmanr(x, y)

print("\n*** NON-PARAMETRIC TEST: Spearman Correlation ***")
print(f"Variabel: btc_market_price vs btc_trade_volume")
print(f"Koefisien Korelasi ( $\rho$ ) = {spearman_corr:.4f}")
print(f"p-value = {spearman_p:.4e}")
print(f"Effect size ( $|p|$ ) = {abs(spearman_corr):.3f}")

if spearman_p < 0.05:
    print("Interpretasi: Terdapat hubungan monoton signifikan antara kedua variabel ( $p < 0.05$ ).")
else:
    print("Interpretasi: Tidak terdapat hubungan monoton signifikan antara kedua variabel ( $p \geq 0.05$ ).")
```

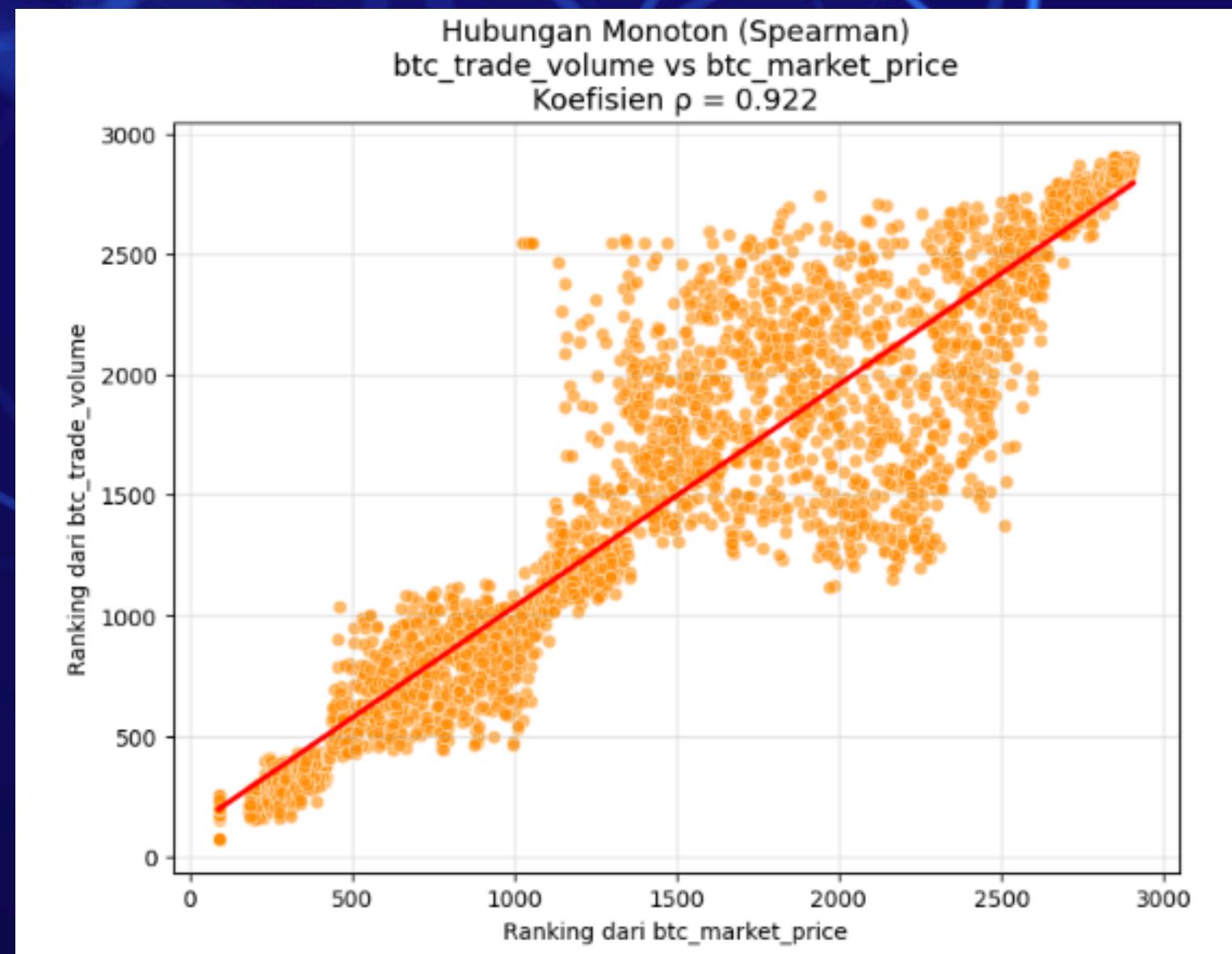
☰ **== PARAMETRIC TEST: Pearson Correlation ==**  
Variabel: btc\_market\_price vs btc\_trade\_volume  
Koefisien Korelasi ( $r$ ) = 0.8686  
p-value = 0.0000e+00  
Effect size ( $|r|$ ) = 0.869  
95% Confidence Interval = [0.859, 0.877]  
Interpretasi: Terdapat hubungan linear yang signifikan secara statistik antara kedua variabel ( $p < 0.05$ ).

**== NON-PARAMETRIC TEST: Spearman Correlation ==**  
Variabel: btc\_market\_price vs btc\_trade\_volume  
Koefisien Korelasi ( $\rho$ ) = 0.9224  
p-value = 0.0000e+00  
Effect size ( $|p|$ ) = 0.922  
Interpretasi: Terdapat hubungan monoton signifikan antara kedua variabel ( $p < 0.05$ ).

# STATISTICAL ANALYSIS

## Melakukan Uji Parametrik dan non-Parametrik

Uji Parametrik dengan Menggunakan Spearman Correlation



Hasil output menunjukkan Koefisien Korelasi ( $\rho$ ) = 0.9224. Nilai ini lebih tinggi daripada koefisien Pearson (0.8686). Plot visual "Hubungan Monoton (spearman)" (grafik oranye) menjelaskan alasannya: dengan mengubah nilai aktual menjadi peringkat, dampak dari outlier dan distribusi data yang miring (skewed) dapat diabaikan. Hasilnya, sebaran titik-titik peringkat terlihat jauh lebih rapat dan membentuk pola linier yang lebih konsisten, yang menunjukkan hubungan monotonik yang bahkan lebih kuat.

Insight utama dari Uji Spearman ini adalah bahwa terdapat hubungan monotonik positif yang hampir sempurna (dengan koefisien  $\rho = 0.9224$ ) antara harga Bitcoin dan volume perdagangannya.

# KESIMPULAN

## Temuan Utama Proyek

Analisis ini menyimpulkan beberapa temuan kunci. Pertama, data harga dan volume Bitcoin memiliki distribusi yang sangat miring ke kanan (right-skewed), yang berarti data terkonsentrasi di nilai rendah namun memiliki outlier bernilai ekstrem. Visualisasi line plot secara jelas mengkonfirmasi adanya volatilitas tinggi, terutama lonjakan besar pada periode 2017-2018.

Tahap preprocessing data terbukti krusial. Reduksi fitur berhasil mengatasi masalah multikolinearitas (fitur yang tumpang tindih dengan korelasi  $> 0.9$ ), sehingga menghasilkan dataset yang lebih efisien untuk analisis.

## Korelasi Harga dan Volume

Temuan utama adalah konfirmasi statistik mengenai hubungan antara harga dan volume:

- Uji Pearson (Linear): Menunjukkan korelasi positif yang sangat kuat ( $r = 0.8686$ ).
- Uji Spearman (Monotonik): Menunjukkan korelasi yang bahkan lebih kuat ( $\rho = 0.9224$ ).

Perbedaan ini adalah insight penting. Fakta bahwa koefisien Spearman lebih tinggi secara statistik mengkonfirmasi bahwa data tidak terdistribusi normal (akibat outlier yang ditemukan sebelumnya). Ini membuktikan Uji Spearman lebih robust (tahan banting) untuk menganalisis data ini.

# THANKYOU

