LAPORAN

TUGAS SUB-CPMK 3

MATA KULIAH DATA MINING

DISUSUN OLEH:

Muhammad Aryasatya Nugroho (22083010085)

DOSEN PENGAMPU:

Trimono, S.Si., M.Si. (NIP. 211199 50 908269)

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN" JAWA TIMUR

2024

STUDI KASUS

Diberikan sebuah data yang terdiri dari sepuluh variabel. Dari data tersebut, lakukan deteksi *outlier* dengan menggunakan dua pendekatan, yaitu pendekatan secara visual dan teoritis. Jika hasil pengujian menunjukkan adanya *outlier*, lakukan penanganan *outlier* tersebut dengan menggunakan metode yang paling tepat.

PENYELESAIAN

Outlier

Outlier, yang juga dikenal sebagai anomali, merujuk pada nilai atau data yang secara signifikan berbeda dari mayoritas data lainnya dalam sebuah kumpulan data. Dalam konteks analisis data, keberadaan *outlier* bisa menjadi hal yang menarik karena mereka seringkali mengandung informasi yang berharga atau menandakan situasi yang istimewa. Analisis *outlier* merupakan upaya untuk mengidentifikasi, memahami, dan mengelola nilai-nilai yang menyimpang dari pola umum atau ekspetasi dalam data. Dengan deteksi *outlier*, para analis dapat memperoleh *insight* yang lebih mendalam tentang karakteristik data, menghadapi kejadian yang tidak biasa, dan mengambil tindakan yang sesuai dalam pengambilan keputusan.

Proses identifikasi *outlier* dan pemahaman terhadapnya menjadi kunci dalam berbagai domain, mulai dari ilmu sosial hingga ilmu pengetahuan alam. *Outlier* dapat muncul dalam berbagai bentuk dari ilmu sosial hingga ilmu pengetahuan alam. *Outlier* juga dapat muncul dalam berbagai bentuk dan dapat memiliki dampak yang signifikan terhadap hasil analisis. Oleh karena itu, analisis *outlier* tidak hanya merupakan langkah penting dalam pemrosesan data, tetapi juga merupakan area penelitian yang aktif dan penting dalam ilmu data modern. Dengan pemahaman yang mendalam tentang *outlier*, kita dapat memastikan keakuratan dan keandalan analisis data serta menghindari kesimpulan yang salah atau tidak akurat.

Implementasi Studi Kasus Menggunakan Pemrograman Python

Dataset yang digunakan terdiri dari sepuluh variabel numerik, masing-masing disebut sebgai Var 1 urut hingga Var 10 dengan jumlah total kolom sebanyak sepuluh dan total baris sebanyak 1000 per kolomnya. Deteksi *outlier* dilakukan menggunakan dua pendekatan yaitu pendekatan visual yang dapat dilakukan dengan menggunakan *boxplot* dan pendekatan teoritis yang dapat dilakukan dengan menggunakan metode IQR. Setelah *outlier* terdeteksi, langkah selanjutnya adalah menangani *outlier* tersebut dengan menggunakan metode yang paling sesuai dengan kebutuhan kita.

1. Persiapan Dataset

```
data = pd.read_csv("Data Outlier Testing.csv")
data
```

	Var 1	Var 2	Var 3	Var 4	Var 5	Var 6	Var 7	Var 8	Var 9	Var 10
0	0.435518	0.038492	0.551343	0.140049	0.899545	0.588684	0.299706	0.245713	0.367375	0.452970
1	0.633197	0.034490	0.319406	0.879141	0.163079	0.184356	0.160583	0.104973	0.294980	0.429709
2	0.421558	0.299824	0.602220	0.521654	0.954621	0.547448	0.882898	0.586641	0.840204	0.212529
3	0.817491	0.647528	0.046214	0.487270	0.053872	0.817499	0.390589	0.394750	0.736854	0.442689
4	0.291513	0.474018	0.065267	0.410573	0.903696	0.466520	0.196878	0.165370	0.297764	0.467911
...	...	...	...	...	...	...	...	...	...	...
995	0.088368	0.402631	0.726744	0.053558	0.920036	0.599621	0.132750	0.119502	0.251470	0.482979
996	0.709031	0.869773	0.563853	0.667340	0.144001	0.116051	0.352950	0.372501	0.343832	0.648546
997	0.281746	0.523779	0.290838	0.837198	0.079527	0.166336	0.468142	0.545292	0.352506	0.686788
998	0.159682	0.825127	0.716445	0.069445	0.032148	0.654702	0.028203	0.886761	0.165482	0.116204
999	0.297497	0.464243	0.577540	0.153434	0.761632	0.549225	0.788681	0.755917	0.611247	0.555821

1000 rows × 10 columns

Load Dataset dan Output DataFrame

Kode tersebut menggunakan *library* pandas untuk membaca *file* CSV yang berisi data. Output dari kode tersebut adalah tabel yang menampilkan data dari *file* CSV 'Data Outlier Testing.csv'. Tabel tersebut terdiri dari sepuluh kolom dan seribu baris data numerik. Setiap baris mewakili sampel data, sedangkan setiap kolom mewakili atribut atau fungsi tertentu dari sampel tersebut. Berikut persebaran data sebelum penanganan *outlier*.

```
print('\nPersebaran data sebelum ditangani Outlier: ')
print(data.describe())
```

	Var 1	Var 2	Var 3	Var 4	Var 5	Var 6	Var 7	Var 8	Var 9	Var 10
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.491362	0.490200	0.509077	0.497362	0.586120	0.514636	0.508270	0.457541	0.477685	0.495760
std	0.259138	0.251931	0.256606	0.263562	0.334658	0.317470	0.278483	0.220129	0.241432	0.211677
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.014495
25%	0.287458	0.291449	0.337802	0.256147	0.169680	0.142587	0.246021	0.245539	0.291452	0.329904
50%	0.492070	0.488656	0.510077	0.497537	0.782019	0.537953	0.465679	0.515619	0.387753	0.488891
75%	0.694192	0.686531	0.686914	0.731949	0.847956	0.856512	0.804935	0.626757	0.723674	0.659528
max	0.994431	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.988732	1.000000

Persebaran Data Awal

Dapat dilihat output dari persebaran data untuk memeriksa *outlier*.

2. Visualisasi Untuk Mendeteksi Outlier

a) Boxplot

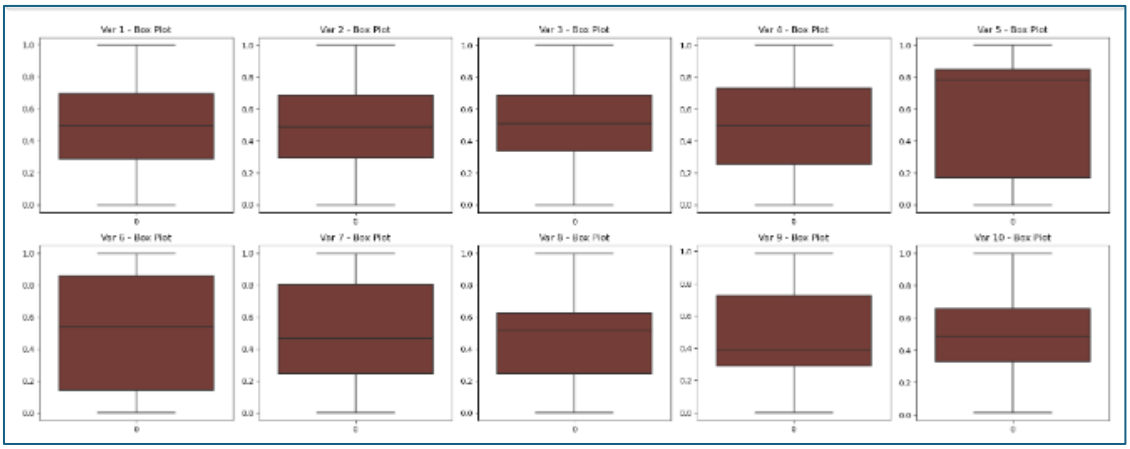
```
# Ukuran plot
plt.figure(figsize=(20, 15))

# Loop melalui setiap kolom dari Var 1 hingga Var 10
for i in range(1, 11):
    # Memilih kolom yang akan divisualisasikan
    data_col = data[f'Var {i}']

    # Membuat subplot untuk setiap kolom
    plt.subplot(4, 5, i)

    # Box Plot
    sns.boxplot(data=data_col, color='#7E352D')
    plt.title(f'Var {i} - Box Plot')

plt.tight_layout()
plt.show()
```



Visualisasi Boxplot

Visualisasi menggunakan *boxplot* adalah salah satu metode yang efektif untuk mendeteksi *outlier* dalam dataset. Pada setiap *boxplot* untuk setiap variabel, titik-titik data yang berada jauh dari jangkauan kuartil dapat dengan mudah diidentifikasi sebagai *outlier*. Titik *outlier* ini muncul sebagai titik-titik yang terletak jauh di luar batas atas atau batas bawah dari "*whiskers*" pada *boxplot*. Dengan menganalisis *boxplot* untuk setiap variabel, kita dapat secara visual mengetahui distribusi nilai-nilai dalam dataset dan mengidentifikasi observasi yang berpotensi tidak biasa atau tidak wajar yang mungkin perlu diselidiki lebih lanjut.

b) Scatter Plot

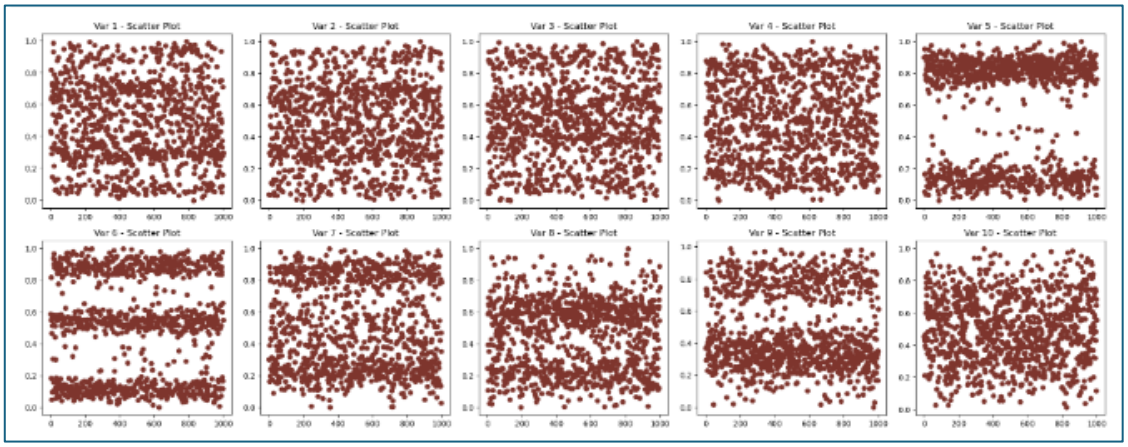
```
# Ukuran plot
plt.figure(figsize=(20, 15))

# Loop melalui setiap kolom dari Var 1 hingga Var 10
for i in range(1, 11):
    # Memilih kolom yang akan divisualisasikan
    data_col = data[f'Var {i}']

    # Membuat subplot untuk setiap kolom
    plt.subplot(4, 5, i)

    # Scatter Plot
    plt.scatter(np.arange(len(data_col)), data_col, color='#7E352D')
    plt.title(f'Var {i} - Scatter Plot')

plt.tight_layout()
plt.show()
```



Visualisasi Scatter Plot

Dalam visualisasi menggunakan *scatter plot* untuk setiap variabel, kita dapat dengan mudah melihat pola sebaran data dan mengidentifikasi titik-titik yang berpotensi menjadi *outlier*. Titik-titik yang jauh terpisah dari pola umum sebaran data dapat dianggap sebagai *outlier*. Dengan menganalisis *scatter plot* untuk setiap variabel, kita dapat memperoleh pemahaman yang lebih baik tentang hubungan antar variabel serta mengidentifikasi observasi yang mungkin tidak biasa atau tidak wajar dalam *dataset*, yang kemungkinan perlu diselidiki lebih lanjut.

c) Histogram

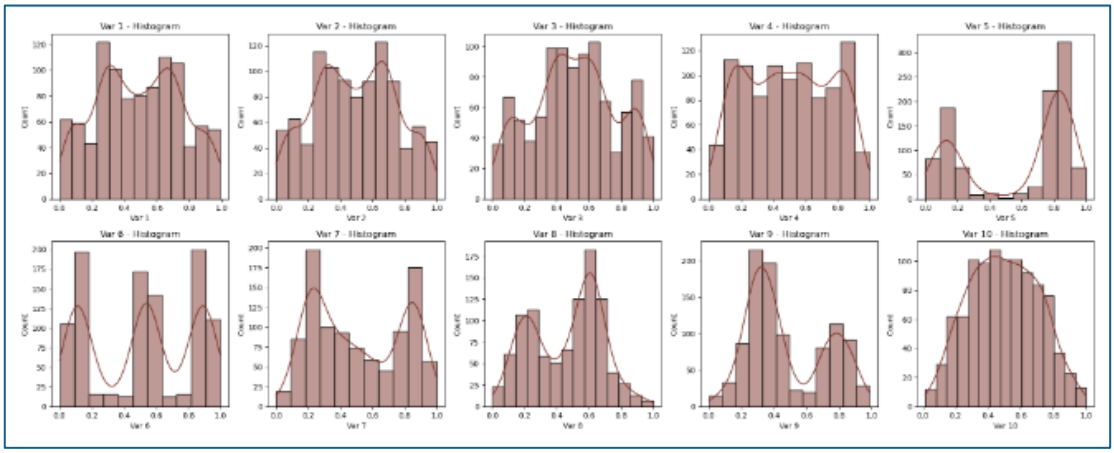
```
# Ukuran plot
plt.figure(figsize=(20, 15))

# Loop melalui setiap kolom dari Var 1 hingga Var 10
for i in range(1, 11):
    # Memilih kolom yang akan divisualisasikan
    data_col = data[f'Var {i}']

    # Membuat subplot untuk setiap kolom
    plt.subplot(4, 5, i)

    # Histogram
    sns.histplot(data_col, color='#7E352D', kde=True)
    plt.title(f'Var {i} - Histogram')

plt.tight_layout()
plt.show()
```



Visusalisasi Histogram

Dalam visualisasi menggunakan histogram untuk setiap variabel, kita dapat melihat distribusi frekuensi atau kejadian berbagai nilai dalam dataset. Dengan menampilkan histogram untuk setiap variabel, kita dapat dengan jelas melihat pola distribusi data dan mengidentifikasi apakah ada anomali atau ketidaknormalan dalam distribusi tersebut. Titik-titik yang memiliki frekuensi yang sangat tinggi atau sangat rendah dibandingkan dengan pola umum distribusi data dapat menjadi indikasi adanya *outlier*. Melalui analisis histogram untuk setiap variabel, kita dapat memperoleh wawasan yang lebih dalam tentang distribusi nilai dalam dataset dan mengidentifikasi observasi yang mungkin perlu ditinjau lebih lanjut.

d) QQ-Plot

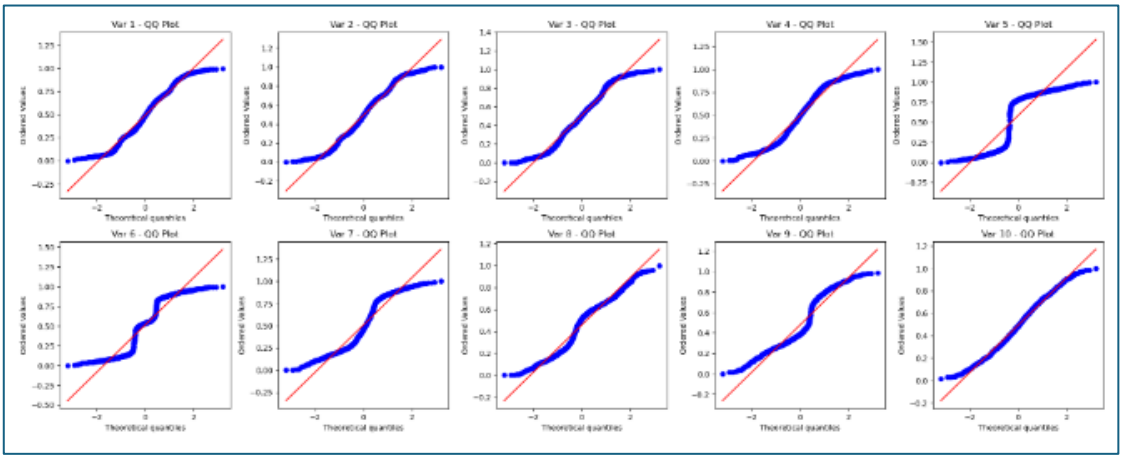
```
# Ukuran plot
plt.figure(figsize=(20, 15))

# Loop melalui setiap kolom dari Var 1 hingga Var 10
for i in range(1, 11):
    # Memilih kolom yang akan divisualisasikan
    data_col = data[f'Var {i}']

    # Membuat subplot untuk setiap kolom
    plt.subplot(4, 5, i)

    # QQ Plot
    stats.probplot(data_col, dist="norm", plot=plt)
    plt.title(f'Var {i} - QQ Plot')

plt.tight_layout()
plt.show()
```



Visualisasi QQ-Plot

Dalam analisis ini, digunakan visualisasi *QQ plot* untuk setiap kolom dari Var 1 hingga Var 10. Hasilnya menunjukkan bahwa data cenderung mengikuti pola distribusi normal, dengan titik-titik mendekati garis diagonal. Hal ini menandakan bahwa data relatif simetris dan tidak memiliki *outlier* yang signifikan dalam distribusinya. Dengan demikian, data tersebut dapat dianggap cukup bersih dan siap untuk digunakan dalam analisis lebih lanjut.

3. Pendekatan Teoritis Untuk Mendeteksi Outlier

a) Metode IQR

```
# Menghitung kuartil pertama (Q1) dan kuartil ketiga (Q3)
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)

# Menghitung IQR
IQR = Q3 - Q1

# Menghitung batas atas (upper fence) dan batas bawah (lower fence) untuk outlier
maximum = Q3 + (1.5 * IQR)
minimum = Q1 - (1.5 * IQR)

# Mengidentifikasi outlier
more_than = (data > maximum)
lower_than = (data < minimum)

# Menggantikan nilai outlier dengan nilai maksimum atau minimum yang telah ditentukan
data_no_outliers = data.mask(more_than, maximum, axis=1)
data_no_outliers = data_no_outliers.mask(lower_than, minimum, axis=1)

# Menampilkan persebaran data setelah ditangani outlier
if data.equals(data_no_outliers):
    print("Tidak ada outlier yang terdeteksi dalam dataset.")
else:
    print('\nPersebaran data setelah ditangani Outlier: ')
    print(data_no_outliers.describe())
```

```
Tidak ada outlier yang terdeteksi dalam dataset.
```

Deteksi Outlier Menggunakan IQR

Dalam proses ini, pertama-tama kuartil pertama (Q1) dan kuartil ketiga (Q3) dari setiap variabel dalam dataset dihitung untuk menghitung Jangkauan Interkuartil (IQR). Selanjutnya, batas atas (*upper fence*) dan batas bawah (*lower fence*) untuk *outlier* dihitung dengan menambahkan dan mengurangkan 1.5 kali IQR dari Q3 dan Q1. Kemudian, *outlier* diidentifikasi dengan membandingkan nilai-nilai dalam *dataset* dengan batas atas dan batas bawah yang telah ditentukan. Jika nilai lebih besar dari batas atas atau lebih kecil dari batas bawah, maka nilai tersebut dianggap sebagai *outlier*. Selanjutnya, nilai *outlier* digantikan dengan nilai maksimum atau minimum yang telah ditentukan. *Output* dari proses ini menunjukkan persebaran data setelah *outlier* ditangani, dengan mencatat bahwa jika *dataset* tidak mengalami perubahan setelah penanganan *outlier*, maka disimpulkan bahwa tidak ada *outlier* yang terdeteksi dalam *dataset*.

b) Metode Z-Score

```python
def detect_outliers_zscore(data, threshold=3):
    z_scores = (data - data.mean()) / data.std()
    return np.abs(z_scores) > threshold

# Menggunakan threshold 3 (nilai default) untuk mendeteksi outlier
outliers_zscore = detect_outliers_zscore(data)

# Menampilkan indeks baris yang mengandung outlier
outlier_indices = np.where(outliers_zscore.any(axis=1))[0]
print("Indeks baris yang mengandung outlier:")
print(outlier_indices)
```

```
Indeks baris yang mengandung outlier:
[]
```

Deteksi Outlier Menggunakan Z-Score

Metode *z-score* digunakan untuk mendeteksi *outlier* dengan menghitung seberapa jauh suatu titik data berjarak dari *mean* populasi dalam satuan standar deviasi. Proses ini melibatkan perhitungan *z-score* untuk setiap titik data dalam *dataset*, di mana *z-score* dihitung sebagai selisih antara nilai titik data dan *mean*, dibagi dengan standar deviasi dari seluruh *dataset*. Nilai *z-scor*e yang tinggi menandakan bahwa titik data tersebut berjarak jauh dari *mean*, mengindikasikan kemungkinan adanya *outlier*. Dalam fungsi detect_outliers_zscore, nilai *threshold* digunakan untuk menentukan batas seberapa jauh titik data dapat dianggap sebagai *outlier*. Jika nilai *z-score* suatu titik data melebihi *threshold* yang ditentukan, titik data tersebut dianggap sebagai *outlier*. Dengan demikian, metode *z-score* memberikan kemampuan untuk mendeteksi *outlier* berdasarkan tingkat ekstrimitas titik data dalam distribusi.

KESIMPULAN

Dalam analisis ini, dilakukan dua pendekatan untuk mendeteksi *outlier*, yaitu secara visual dan teoritis. Pendekatan visual menggambarkan data melalui berbagai plot seperti box *plot*, *scatter plot*, histogram, dan *QQ plot*, sementara pendekatan teoritis menggunakan metode IQR dan *z-score*. Hasil dari kedua pendekatan menunjukkan bahwa tidak ada *outlier* yang terdeteksi dalam *dataset*, menegaskan konsistensi dan keandalan persebaran data yang digunakan.

Kesimpulannya, melalui kedua pendekatan tersebut, dapat disimpulkan bahwa *dataset* yang digunakan dalam analisis ini tidak mengandung *outlier* yang signifikan. Hal ini memberikan keyakinan bahwa data tersebut dapat dipercaya dan cocok untuk digunakan dalam analisis lebih lanjut tanpa perlunya penanganan *outlier* tambahan.