Technical Test Data Science PT Bina Pertiwi

Enrico Farizky Rustam

Latihan 1

- Pada kasus ini, akan diterapkan proses data cleansing, dimana data dibersihkan agar dapat dianalisis lebih lanjut dengan tepat.
- Penanganan nilai hilang dapat dilakukan dengan berbagai cara, dari menghilangkan row tersebut sampai mengisi nilai yang hilang
- Ketidaksesuaian format pada data dapat dilakukan secara manual atau dengan cara normalisasi jika diperlukan
- Penghapusan duplikasi dilakukan dengan mengidentifikasi duplikasi dan menghapus row duplikasi tersebut.

Latihan 1

```python
import pandas as pd
import numpy as np

# Contoh data dummy
data = {
    'Tanggal_Film': ['2023-08-01', '2023-08-02', '2023-08-01', '2023-08-01', '2023-08-01'],
    'Judul_Film': ['Film A', 'Film B', 'Film A', 'Film A', 'Film A'],
    'Durasi_Film': [120, 100, 120, 120, np.nan],
    'Kapasitas_Auditorium': [150, 200, 150, 150, 120],
    'Tiket_Terjual': [120, 180, 130, 130, 120],
    'Harga_Tiket': [10, 15, 10, 10, 120]
}

df = pd.DataFrame(data)
df
```

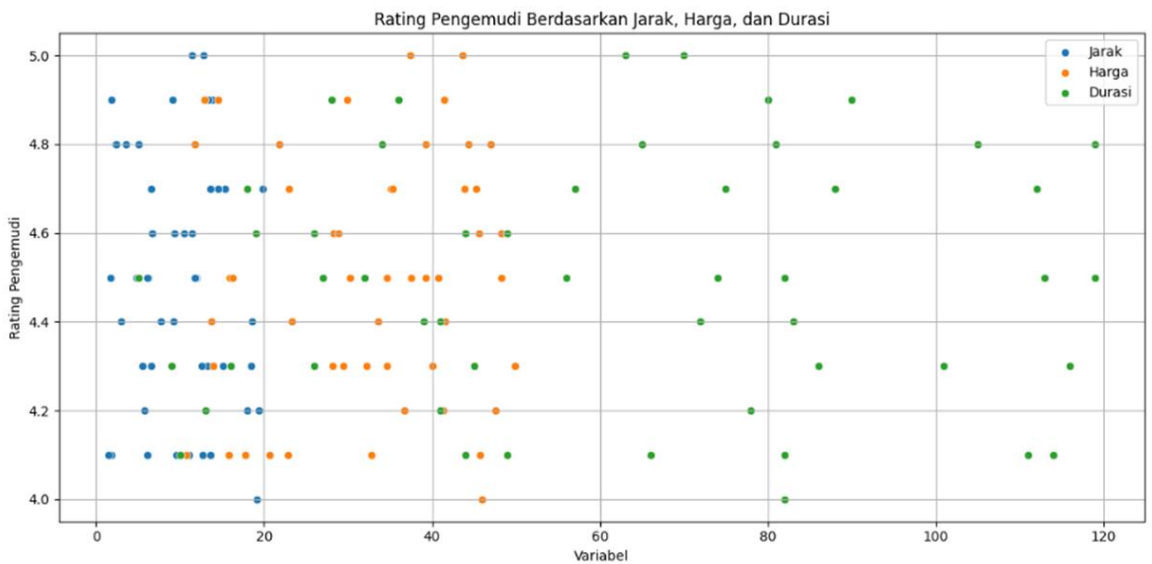
	Tanggal_Film	Judul_Film	Durasi_Film	Kapasitas_Auditorium	Tiket_Terjual	Harga_Tiket
0	2023-08-01	Film A	120.0	150	120	10
1	2023-08-02	Film B	100.0	200	180	15
2	2023-08-01	Film A	120.0	150	130	10
3	2023-08-01	Film A	120.0	150	130	10
4	2023-08-01	Film A	NaN	120	120	120

```python
# Mengubah kolom tanggal menjadi tipe datetime
df['Tanggal_Film'] = pd.to_datetime(df['Tanggal_Film'])

# Menghapus duplikasi berdasarkan kolom tertentu
df_cleaned = df.drop_duplicates(subset=['Tanggal_Film', 'Judul_Film', 'Durasi_Fil

# Mengisi nilai hilang jika diperlukan
#df_cleaned = df_cleaned.fillna(0)
df_cleaned = df_cleaned.fillna(method ='pad')
df_cleaned
```

	Tanggal_Film	Judul_Film	Durasi_Film	Kapasitas_Auditorium	Tiket_Terjual	Har
0	2023-08-01	Film A	120.0	150	120	
1	2023-08-02	Film B	100.0	200	180	
2	2023-08-01	Film A	120.0	150	130	
4	2023-08-01	Film A	120.0	120	120	

Latihan 2

```
sns.scatterplot(x='jarak', y='driver_rating', data=dflat2, label='Jarak')
sns.scatterplot(x='harga', y='driver_rating', data=dflat2, label='Harga')
sns.scatterplot(x='durasi', y='driver_rating', data=dflat2, label='Durasi')

plt.xlabel('Variabel')
plt.ylabel('Rating Pengemudi')
plt.title('Rating Pengemudi Berdasarkan Jarak, Harga, dan Durasi')
plt.legend()
plt.grid(True)
plt.tight_layout()

plt.show()
```




```
jarak = [round(random.uniform(1, 20), 2) for _ in range(50)]
durasi = [random.randint(5, 120) for _ in range(50)]
harga = [round(random.uniform(10, 50), 2) for _ in range(50)]
driver_rating = [round(random.uniform(4, 5), 1) for _ in range(50)]
customer_rating = [round(random.uniform(4, 5), 1) for _ in range(50)]

# Membuat dataset dummy
datalat2 = {
    'tanggal_waktu': tanggal_waktu,
    'jarak': jarak,
    'durasi': durasi,
    'harga': harga,
    'driver_rating': driver_rating,
    'customer_rating': customer_rating
}

# Membuat DataFrame dari data dummy
dflat2 = pd.DataFrame(datalat2)
dflat2[:5]
```

	tanggal_waktu	jarak	durasi	harga	driver_rating	customer_rating

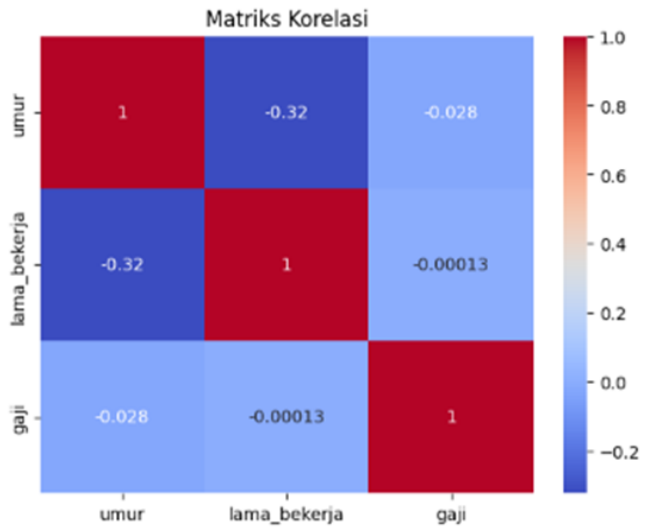
Latihan 3

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
import random

# Data Dummy
umur = [random.randint(20, 60) for _ in range(50)]
jenis_kelamin = ['Laki-laki', 'Perempuan']
pendidikan = ['SMA', 'D3', 'S1', 'S2']
lama_bekerja = [random.randint(1, 15) for _ in range(50)]
gaji = [random.randint(2000, 15000) for _ in range(50)]

# Membuat dataset
data2 = {
    'umur': umur,
    'jenis_kelamin': [random.choice(jenis_kelamin) for _ in range(50)],
    'pendidikan': [random.choice(pendidikan) for _ in range(50)],
    'lama_bekerja': lama_bekerja,
    'gaji': gaji
}

# Membuat DataFrame
df1 = pd.DataFrame(data2)
df1.head()
```

	umur	jenis_kelamin	pendidikan	lama_bekerja	gaji
0	45	Perempuan	S1	4	9991
1	56	Laki-laki	S2	13	8436
2	56	Laki-laki	SMA	6	12433
3	36	Laki-laki	SMA	3	13108
4	38	Perempuan	D3	10	12081

Latihan 3

```python
# Statistik Deskriptif
descriptive_stats = df1.describe()
print ("Statistik Deskriptif")
print(descriptive_stats)

# Distribusi dan Visualisasi
sns.histplot(data2['umur'], bins=20, kde=True)
plt.title('Distribusi Umur')
plt.show()

sns.histplot(data2['gaji'], bins=20, kde=True)
plt.title('Distribusi Gaji')
plt.show()

sns.histplot(data2['lama_bekerja'], bins=20, kde=True)
plt.title('Distribusi Lama Bekerja')
plt.show()

sns.countplot(x='jenis_kelamin', data=data2)
plt.title('Distribusi Jenis Kelamin')
plt.show()

sns.countplot(x='pendidikan', data=data2)
plt.title('Distribusi Pendidikan')
plt.xticks(rotation=45)
plt.show()

# Korelasi
correlation_matrix = df1.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Matriks Korelasi')
plt.show()

# Uji Hipotesis
t_statistic, p_value = stats.ttest_ind(df1[df1['jenis_kelamin'] == 'Laki-laki']['gaji'],
                                       df1[df1['jenis_kelamin'] == 'Perempuan']['gaji'])
print('T-statistic:', t_statistic)
print('P-value:', p_value)
if p_value < 0.05:
    print('Perbedaan gaji antara jenis kelamin signifikan.')
else:
    print('Tidak ada perbedaan gaji yang signifikan antara jenis kelamin.')

# Regresi Linier
from sklearn.linear_model import LinearRegression
model = LinearRegression()
X = df1[['umur', 'lama_bekerja']]
y = df1['gaji']
model.fit(X, y)
print('Koefisien Regresi:', model.coef_)
print('Intersep:', model.intercept_)
```



Statistik Deskriptif

	umur	lama_bekerja	gaji
count	50.000000	50.000000	50.000000
mean	37.920000	8.740000	8143.000000
std	12.064182	4.168933	3937.853984
min	20.000000	1.000000	2039.000000
25%	28.000000	5.250000	4851.000000
50%	35.500000	8.500000	8113.000000
75%	47.750000	12.750000	11317.000000
max	59.000000	15.000000	14796.000000

```
T-statistic: 0.5325351686618917
P-value: 0.5968121628066152
Tidak ada perbedaan gaji yang signifikan antara jenis kelamin.
Koefisien Regresi: [-10.28932855  -9.76291745]
Intersep: 8618.499237142752
```

Latihan 4

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Contoh data dummy
data1 = {
    'Usia': [25, 30, 22, 35, 28, 40, 27],
    'Pendapatan': [50000, 70000, 40000, 80000, 60000, 90000, 55000],
    'Jumlah_Kartu': [2, 1, 3, 2, 2, 4, 1],
    'Pengeluaran_Bulanan': [1000, 1200, 800, 1500, 1100, 2000, 900],
    'Tertarik': [0, 1, 0, 1, 0, 1, 0]  # 0 = Tidak tertarik, 1 = Tertarik
}

df = pd.DataFrame(data1)
df
```

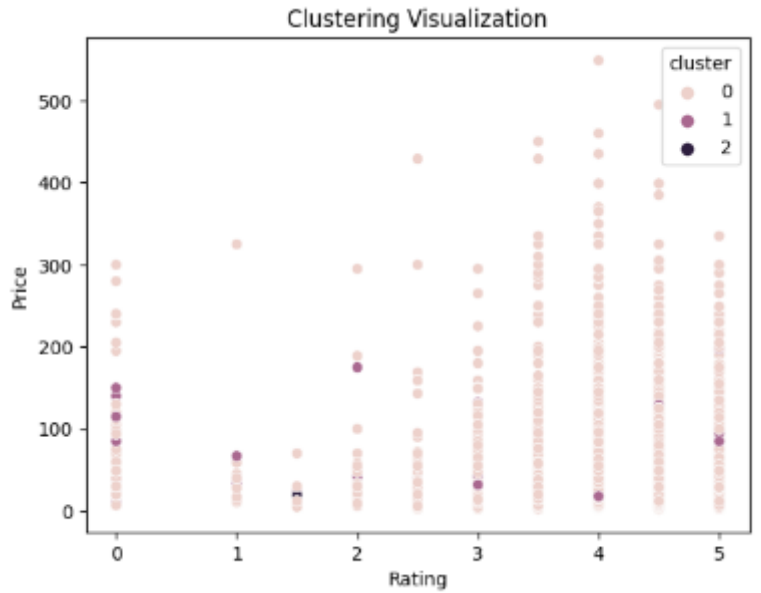
	Usia	Pendapatan	Jumlah_Kartu	Pengeluaran_Bulanan	Tertarik
0	25	50000	2	1000	0
1	30	70000	1	1200	1
2	22	40000	3	800	0
3	35	80000	2	1500	1
4	28	60000	2	1100	0
5	40	90000	4	2000	1
6	27	55000	1	900	0

```
# Pisahkan fitur dan label
X = df.drop('Tertarik', axis=1)
y = df['Tertarik']

# Pisahkan data untuk pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Inisialisasi dan latih model Logistic Regression
model = LogisticRegression()
model.fit(X_train, y_train)

# Prediksi dengan data pengujian
y_pred = model.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print(f'Akurasi model: {accuracy}')
```

```
Akurasi model: 0.5
```

Latihan 5

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Load the train and test datasets
train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')

# Preprocessing
columns_to_drop = ['id', 'name', 'URL', 'MarketingFlags', 'options', 'details', 'how_to_use', 'ingredients', 'size']
train_data = train_data.drop(columns=columns_to_drop)
test_data = test_data.drop(columns=columns_to_drop)

# Convert categorical variables to numerical using one-hot encoding
train_data = pd.get_dummies(train_data, columns=['brand', 'category'], drop_first=True)
test_data = pd.get_dummies(test_data, columns=['brand', 'category'], drop_first=True)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(train_data.drop(columns=['exclusive']))

# Model training (K-Means clustering)
num_clusters = 3  # You can adjust this number
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
train_data['cluster'] = kmeans.fit_predict(X_train)

# Visualize clustering
sns.scatterplot(data=train_data, x='rating', y='price', hue='cluster')
plt.title('Clustering Visualization')
plt.xlabel('Rating')
plt.ylabel('Price')
plt.show()
```



Github Source code

- https://github.com/enricorustam/Technical-Test-Data-Science-PT-Bina-Pertiwi