Nama : Al Fitra Nur Ramadhani

NIM : 202210370311264

Mata Kuliah : Data, Information & Knowledge

Laporan PreProcessing Metadata Aplikasi Google Play Store

1. Deskripsi

Laporan ini bertujuan untuk menjelaskan tahapan PreProcessing dataset metadata aplikasi dari Google Play Store yang berisi informasi seperti nama aplikasi, kategori, rating, jumlah ulasan, ukuran aplikasi, jumlah instalasi, tipe (gratis/berbayar), harga, rating konten, genre, tanggal pembaruan terakhir, versi saat ini, dan versi Android yang didukung. Dataset ini diambil dari Kaggle yang awalnya ada 10.842 rows , kemudian setelah melewati beberapa tahapan preprocessing menjadi sekitar 9660 rows.

2. Link Kaggle = Google Play Store Apps

3. Link GitHub = DATA-INFORMATION-KNOWLEDGE/GooglePlayStore at main · alfitranurr/DATA-INFORMATION-KNOWLEDGE

4. Implementasi Step by Step

a) Menampilkan metadata Information Details

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
✓ 1.3s
```

- Import beberapa library yang dibutuhkan, pandas (Pengolahan data), numpy (Perhitungan Numerik), matplotlib (visualisasi Grafik), seaborn (Visualisasi Statistik), re (Ekspresi Reguler)

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
✓ 0.0s
```

- Atur untuk menampilkan semua baris, kolom dan tampilan penuh

```
df = pd.read_csv('googleplaystore.csv')
✓ 0.0s
```

- Import data csv nya untuk dibaca di kernel python

```
df.info()
✓ 0.0s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   App             10841 non-null  object
 1   Category        10841 non-null  object
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  object
 4   Size            10841 non-null  object
 5   Installs        10841 non-null  object
 6   Type            10840 non-null  object
 7   Price           10841 non-null  object
 8   Content Rating  10840 non-null  object
 9   Genres          10841 non-null  object
 10  Last Updated    10841 non-null  object
 11  Current Ver     10833 non-null  object
 12  Android Ver     10838 non-null  object
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

- Menampilkan Informasi Dataframe
- Jumlah data ada 10841 entries dan 13 kolom
- Beberapa tipe data tidak sesuai dengan hakikatnya seperti Rating, Reviews, Size, Installs, Price, dan Last Updated
- Beberapa data ditemukan missing value di kolom Rating, Type, Content Rating, Cureent Ver, dan Android Ver

```
df.describe()
```
✓ 0.0s

	# Rating
count	9367.0
mean	4.193338315362443
std	0.5374313031477587
min	1.0
25%	4.0
50%	4.3
75%	4.5
max	19.0

8 rows x 1 cols 10 per page

- Statistik Deskriptif, menyangkut mean , count, standar deviasi, dll untuk melihat sebaran data

```
nama_kolom = df.columns.tolist()
print(nama_kolom)
```
✓ 0.0s Open 'nama_kolom' in Data Wrangler

```
['App', 'Category', 'Rating', 'Reviews', 'Size', 'Installs', 'Type', 'Price', 'Content Rating', 'Genres', 'Last Updated', 'Current Ver', 'Android Ver']
```

- Semua nama kolom yang ada, agar mudah untuk di analisis
- ==========

Cek Missing Value

```
print(df.isnull().sum())
```
[10] ✓ 0.0s

```
App                1
Category           0
Rating          1474
Reviews            0
Size               0
Installs           0
Type               1
Price              0
Content Rating     1
Genres             0
Last Updated       0
Current Ver        8
Android Ver        3
dtype: int64
```

```
df.isnull().sum() / len(df) * 100
```
✓ 0.0s

```
App               0.000000
Category          0.000000
Rating           13.596532
Reviews           0.000000
Size              0.000000
Installs          0.000000
Type              0.009224
Price             0.000000
Content Rating    0.009224
Genres            0.000000
Last Updated      0.000000
Current Ver       0.073794
Android Ver       0.027673
dtype: float64
```

```
df.isnull().sum().sum()
```
✓ 0.0s

```
np.int64(1487)
```

Cek Missing Value

- Informasi detail mengenai kolom mana saja yang missing, ada di kolom Rating, Type, Content rating, Current Ver dan Android Ver
- Total Missing Value ada 1487 cell
- Dan ditampilkan persenan missing value terhadap keseluruhan data , paling tinggi ada di kolom Reviews yaitu 13,6%

==========

```
# 1. Cek total baris
print("Total entri:", len(df))

# 2. Cek jumlah duplikat total
print("Jumlah baris duplikat:", df.duplicated().sum())

# 3. Cek jumlah duplikat di kolom 'App'
print("Jumlah duplikat di kolom App:", df['App'].duplicated().sum())

# 4. Cek jumlah nilai unik di kolom 'App'
print("Jumlah aplikasi unik:", df['App'].nunique())

# 5. Cek nilai kosong di kolom 'App'
print("Jumlah nilai kosong di kolom App:", df['App'].isna().sum())
```
✓ 0.0s

```
Total entri: 10841
Jumlah baris duplikat: 483
Jumlah duplikat di kolom App: 1181
Jumlah aplikasi unik: 9660
Jumlah nilai kosong di kolom App: 0
```

Cek Duplikat Data

```
jumlah_duplikat = df.duplicated(keep='first').sum()
print(jumlah_duplikat)
```
✓ 0.0s

```
483
```

Cek Data Duplikat

- Setelah di cek duplikat data ternyata ada 1181 duplikat dengan mengacu pada kolom App

```
df_duplikat = df[df.duplicated(keep=False)]
display(df_duplikat)
```
✓ 0.1s Open 'df_duplikat' in Data Wrangler Python

	App	Category	# Rating	Reviews	Size	Installs	
164	Ebook Reader	BOOKS_AND_REFERENCE	4.1	85842	37M	5,000,000+	Fre
192	Docs To Go™ Free Office Suite	BUSINESS	4.1	217730	Varies with device	50,000,000+	Fre
193	Google My Business	BUSINESS	4.4	70991	Varies with device	5,000,000+	Fre

b) Data Preprocessing

- **Ubah Data Type**

1. Pastikan Format data untuk Rating adalah float

```
df['Rating'] = pd.to_numeric(df['Rating'])
```

✓ 0.0s

- Tipe data numeric untuk Rating

2. Tipe Data Reviews dari Object menjadi Integer

```
# Temukan indeks baris yang bermasalah
idx = df.index[df['App'] == 'Life Made WI-Fi Touchscreen Photo Frame'].tolist()[0]

# Ambil data baris sebagai list
row_data = df.loc[idx].tolist()

# Kolom yang diharapkan
columns = ['App']

# Geser data ke kanan dan tambahkan 'Category' dengan nilai 'Unknown'
corrected_data = ['Life Made WI-Fi Touchscreen Photo Frame', 'Unknown'] + row_data[1:-1]

# Pastikan panjang data sesuai dengan jumlah kolom
if len(corrected_data) < len(columns):
    corrected_data.extend([None] * (len(columns) - len(corrected_data)))

# Perbarui baris di DataFrame
df.loc[idx] = corrected_data

# Verifikasi
print("Data setelah diperbaiki:")
print(df.loc[idx])
```

✓ 0.0s

```
df['Reviews'] = df['Reviews'].astype(int)
```

✓ 0.0s

- Ternyata setelah di cek di Kolom Reviews ada di kolom App (Life Made WI-Fi Touchscreen Photo Frame) datanya kegeser ke kanan sebanyak 1
- Setelah di proses kemudian ubah tipe datanya menjadi integer

3. Tipe Data Size dari Object menjadi float64

```
# Ubah nama kolom Size
df = df.rename(columns={'Size': 'Size (M)'})
```

✓ 0.0s

```
def clean_size(size):
    if pd.isna(size) or str(size).lower() == 'varies with device':
        return None
    size_str = str(size).replace('M', '').replace('k', 'e-3')
    try:
        return float(size_str)
    except ValueError:
        return None

df['Size (M)'] = df['Size (M)'].apply(clean_size)
```

✓ 0.0s

```
df['Size (M)'] = df['Size (M)'].astype(float)
```

✓ 0.0s

- Rename kolom untuk Size menjadi Size (M) untuk memberikan penjelasan lebih baik, karena akan di ubah tipe data nya menjadi float
- Cleaning format data yang Dimana yang mengandung huruf M maka akan dihapus

4. Ubah tipe data Installs dari Object menjadi Integer

```
# Konversi kolom Installs (misalnya "10,000+" ke 10000)
df['Installs'] = df['Installs'].str.replace('[\+,]', '', regex=True)
df['Installs'] = pd.to_numeric(df['Installs'], errors='coerce').fillna(0)
```

✓ 0.0s

```
<>:2: SyntaxWarning: invalid escape sequence '\+'
<>:2: SyntaxWarning: invalid escape sequence '\+'
C:\Users\ASUS\AppData\Local\Temp\ipykernel_24388\2524618794.py:2: SyntaxWarning: invalid escape sequence '\+'
  df['Installs'] = df['Installs'].str.replace('[\+,]', '', regex=True)
```

```
# Konversi ke tipe integer (int64)
df['Installs'] = df['Installs'].astype('int64')
```

✓ 0.0s

- Hapus karakter symbol yang ada di kolom Installs dan ubah tipe data menjadi Integer

5. Ubah tipe data Price dari Object menjadi float64

```
# Bersihkan kolom Price: hapus simbol "$" dan konversi ke float
df['Price'] = df['Price'].replace('[\$,]', '', regex=True)
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')

print("Tipe data kolom Price:", df['Price'].dtype)
```

✓ 0.0s

```
<>:2: SyntaxWarning: invalid escape sequence '\$'
<>:2: SyntaxWarning: invalid escape sequence '\$'
Tipe data kolom Price: float64
C:\Users\ASUS\AppData\Local\Temp\ipykernel_24388\1313369473.py:2: SyntaxWarning: invalid escape sequence '\$'
  df['Price'] = df['Price'].replace('[\$,]', '', regex=True)
```

- Ubah tipe data Price menjadi float dan menghapus symbol symbol agar lebih konsisten untuk numeric

6. Ubah tipe data Last Updated dari Object menjadi Date Time

```
df['Last Updated'] = pd.to_datetime(df['Last Updated'], format='%B %d, %Y')
```

✓ 0.0s

- Ubah tipe data Last Updated menjadi Date Time

- **Cek & Drop Duplikat**

-

```
df = df.drop_duplicates(subset=['App'])
```

✓ 0.0s

```
df = df.drop_duplicates(keep='first')
```

✓ 0.0s

```
jumlah_duplikat = df.duplicated(keep='first').sum()
print(jumlah_duplikat)
```

✓ 0.0s

```
0
```

```
print("Total entri setelah hapus duplikat:", len(df))
```

✓ 0.0s

```
Total entri setelah hapus duplikat: 9660
```

- Drop duplikat di kolom App , dan setiap baris yang duplikat dan gunakan first untuk menjaga data pertama tidak ke drop
- Periksa lagi data nya apakah sudah clear
- Total entri setelah duplikat ada 9660 rows

- **Cek Missing Value Again**

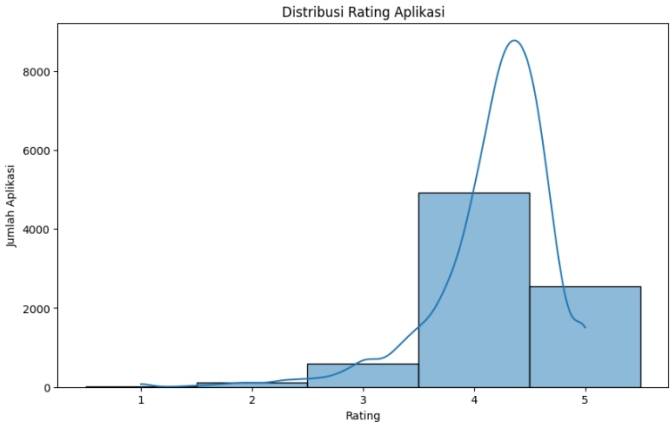
Cek Missing Value

```
print(df.isnull().sum())
```

```
App                0
Category           0
Rating          1463
Reviews            0
Size (M)        1227
Installs           0
Type               1
Price              0
Content Rating     0
Genres             1
Last Updated       0
Current Ver        8
Android Ver        2
dtype: int64
```

- Missing Value = Rating, Size (M), Type, Genres, Current Ver, Android Ver

 # Kolom Rating



1. Missing Value Kolom Rating

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Rating', bins=20, kde=True)
plt.title('Distribusi Rating Aplikasi')
plt.xlabel('Rating')
plt.ylabel('Jumlah Aplikasi')
plt.show()
```

- Setelah di cek distribusi data lewat histogram ternyata ada data outlier , data lebih dari 5, yang seharusnya rentang value untuk rating kisaran 1-5

```
df['Rating'] = pd.to_numeric(df['Rating'])
```

```
df['Rating'] = df['Rating'].replace('', pd.NA)
```

```
# Mengisi missing value dengan median per kategori
df['Rating'] = df.groupby('Category')['Rating'].transform(lambda x: x.fillna(x.median()))
```

- Mengisi missing value di Kolom Rating dengan median karena lebih robust untuk distribusi data dan tidak terpengaruh oleh outlier nya

 # Kolom Size (M)

2. Missing Value pada Kolom Size (M)

```
df['Size (M)'] = df['Size (M)'].fillna(df['Size (M)'].median())
```

- Isi missing value dengan median karena lebih robust terhadap distribusi data dan data di kolom Size (M) bersifat ordinal

 # Kolom Type

3. Missing Value pada Kolom Type

```
# Menghitung distribusi nilai pada kolom Type
type_distribution = df['Type'].value_counts(dropna=False)
type_percentage = df['Type'].value_counts(normalize=True, dropna=False) * 100

print("Distribusi nilai pada kolom Type:")
print(type_distribution)
print("\nPersentase distribusi nilai pada kolom Type:")
print(type_percentage)
```

```
Distribusi nilai pada kolom Type:
Type
Free    8903
Paid     756
NaN        1
Name: count, dtype: int64

Persentase distribusi nilai pada kolom Type:
Type
Free    92.163561
Paid     7.826087
NaN      0.010352
Name: proportion, dtype: float64
```

```python
# Mengganti nilai kosong atau "" dengan NaN
df['Type'] = df['Type'].replace('', pd.NA)

# Menemukan baris dengan NaN di kolom Type
missing_type_rows = df[df['Type'].isna()]

# Menampilkan baris dengan missing value
print("Baris dengan missing value di kolom Type (NaN):")
display(missing_type_rows)
```

Python

```
Baris dengan missing value di kolom Type (NaN):
```

	App	Category	# Rating	# Reviews	# Size (M)	# Installs	Type
9148	Command & Conquer: Rivals	FAMILY	4.3	0	12.0	0	*Missing value*

```python
# Opsi 2 Mengisi Berdasarkan Kolom Price (Alternatif)

# Mengganti nilai kosong atau "" dengan NaN
df['Type'] = df['Type'].replace('', pd.NA)

# Mengganti "0" dengan NaN (karena tidak valid)
df['Type'] = df['Type'].replace('0', pd.NA)

# Mengisi Type berdasarkan Price
df.loc[(df['Type'].isna()) & (df['Price'] == 0), 'Type'] = 'Free'
df.loc[(df['Type'].isna()) & (df['Price'] > 0), 'Type'] = 'Paid'
```

- Mengisi missing value dengan melihat kolom price , karena price nya 0 maka diisi dengan free, jika > 0 maka diisi dengan paid

 # Kolom Genres

4. Missing Value pada Kolom Genres

```python
# Menemukan baris dengan missing value di kolom Price
missing_price_rows = df[df['Genres'].isna()]

# Menampilkan baris dengan missing value
print("Baris dengan missing value di kolom Genres:")
display(missing_price_rows[['App', 'Category', 'Genres']])
```

```
Baris dengan missing value di kolom Genres:
```

	App	Category	Genres
10472	Life Made WI-Fi Touchscreen Photo F	Unknown	*Missing value*

1 rows x 3 cols 10 per page

```python
df['Genres'] = df['Genres'].fillna('Unknown')
```

- Karena Data tidak tahu source nya maka diisi Unknown agar cari aman

Kolom Current Ver

5. Missing Value pada Kolom Current Ver

```python
# Menemukan baris dengan missing value
missing_curr_ver_rows = df[df['Current Ver'].isna()]
print("\nBaris dengan missing value di kolom Content Rating:")
display(missing_curr_ver_rows[['App', 'Category', 'Current Ver', 'Last Updated']])
```

```
Baris dengan missing value di kolom Content Rating:
```

	App	Category	Current Ver
15	Learn To Draw Kawaii Characters	ART_AND_DESIGN	*Missing value*
1553	Market Update Helper	LIBRARIES_AND_DEMO	*Missing value*
6322	Virtual DJ Sound Mixer	TOOLS	*Missing value*
6803	BT Master	FAMILY	*Missing value*
7333	Dots puzzle	FAMILY	*Missing value*
7407	Calculate My IQ	FAMILY	*Missing value*
7730	UFO-CQ	TOOLS	*Missing value*
10342	La Fe de Jesus	BOOKS_AND_REFERENCE	*Missing value*

8 rows x 4 cols 10 per page

```python
# Mengisi missing value dengan "Unknown"
df['Current Ver'] = df['Current Ver'].fillna('Unknown')
```

- Mengisi missing value dengan Unknown karena tidak tahu source nya seperti apa

 # Kolom Android Ver

6. Missing Value pada Kolom Android Ver

```
# Menemukan baris dengan missing value
missing_ver_rows = df[df['Android Ver'].isna()]
print("\nBaris dengan missing value di kolom Android Ver:")
display(missing_ver_rows[['App', 'Android Ver']])
```

```
Baris dengan missing value di kolom Android Ver:
```

	App	Android Ver
4453	[substratum] Vacuum: P	*Missing value*
4490	Pi Dark [substratum]	*Missing value*

2 rows x 2 cols 10 per page

```
# Mengisi missing value dengan "Unknown"
df['Android Ver'] = df['Android Ver'].fillna('Unknown')
```

- Isi missing value dengan Unknown agar cari aman karena source nya tidak diketahui, karena data nya kategorikal maka ini opsi terbaik menurut saya

c) Deteksi Outlier

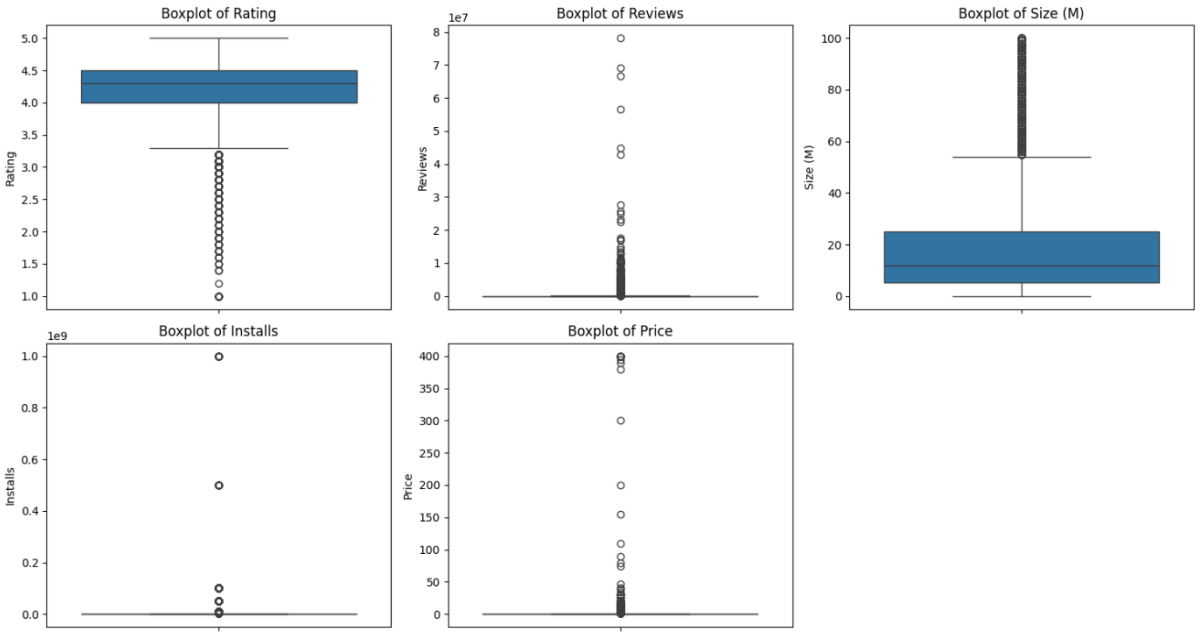
Deteksi Outlier

```
numeric_cols = ['Rating', 'Reviews', 'Size (M)', 'Installs', 'Price']
```

```
# Membuat boxplot untuk setiap kolom numerik
plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot of {col}')
    plt.tight_layout()

plt.show()
```

- Deteksi Outlier dengan boxplot untuk kolom numerik = Rating, Reviews, Size (M), Installs, price
- Untuk Distribusi data nya masih aman jika dilihat dari konteks kolom nya



Metode IQR for Outlier

Metode IQR for Outlier

```python
# Fungsi untuk menghitung batas IQR dan mendeteksi outlier
def detect_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)][column]

    return lower_bound, upper_bound, outliers
```

```python
# Menyimpan hasil batas dan outlier
outlier_info = {}
for col in numeric_cols:
    lower, upper, outliers = detect_outliers_iqr(df, col)
    outlier_info[col] = {
        'Lower Bound': lower,
        'Upper Bound': upper,
        'Outliers Count': len(outliers),
        'Outliers': outliers.tolist()[:5]
    }
```

```python
# Tampilkan batas bawah dan atas serta informasi outlier
for col, info in outlier_info.items():
    print(f"\nKolom: {col}")
    print(f"Batas Bawah: {info['Lower Bound']:.2f}")
    print(f"Batas Atas: {info['Upper Bound']:.2f}")
    print(f"Jumlah Outlier: {info['Outliers Count']}")
    print(f"Contoh Outlier (5 pertama): {info['Outliers']}")
```

```
Kolom: Rating
Batas Bawah: 3.25
Batas Atas: 5.25
Jumlah Outlier: 493
Contoh Outlier (5 pertama): [3.2, 3.2, 3.1, 3.2, 3.2]

Kolom: Reviews
Batas Bawah: -44028.50
Batas Atas: 73447.50
Jumlah Outlier: 1656
Contoh Outlier (5 pertama): [87510, 215644, 194216, 224399, 295221]

Kolom: Size (M)
Batas Bawah: -24.25
Batas Atas: 54.55
Jumlah Outlier: 762
Contoh Outlier (5 pertama): [56.0, 57.0, 57.0, 73.0, 55.0]

Kolom: Installs
Batas Bawah: -1497500.00
Batas Atas: 2498500.00
Jumlah Outlier: 1978
Contoh Outlier (5 pertama): [5000000, 50000000, 10000000, 5000000, 10000000]

Kolom: Price
Batas Bawah: 0.00
Batas Atas: 0.00
Jumlah Outlier: 756
Contoh Outlier (5 pertama): [4.99, 4.99, 3.99, 3.99, 6.99]
```

Histogram untuk melihat distribusi data