Data Cleaning Using Python for Data Science Project

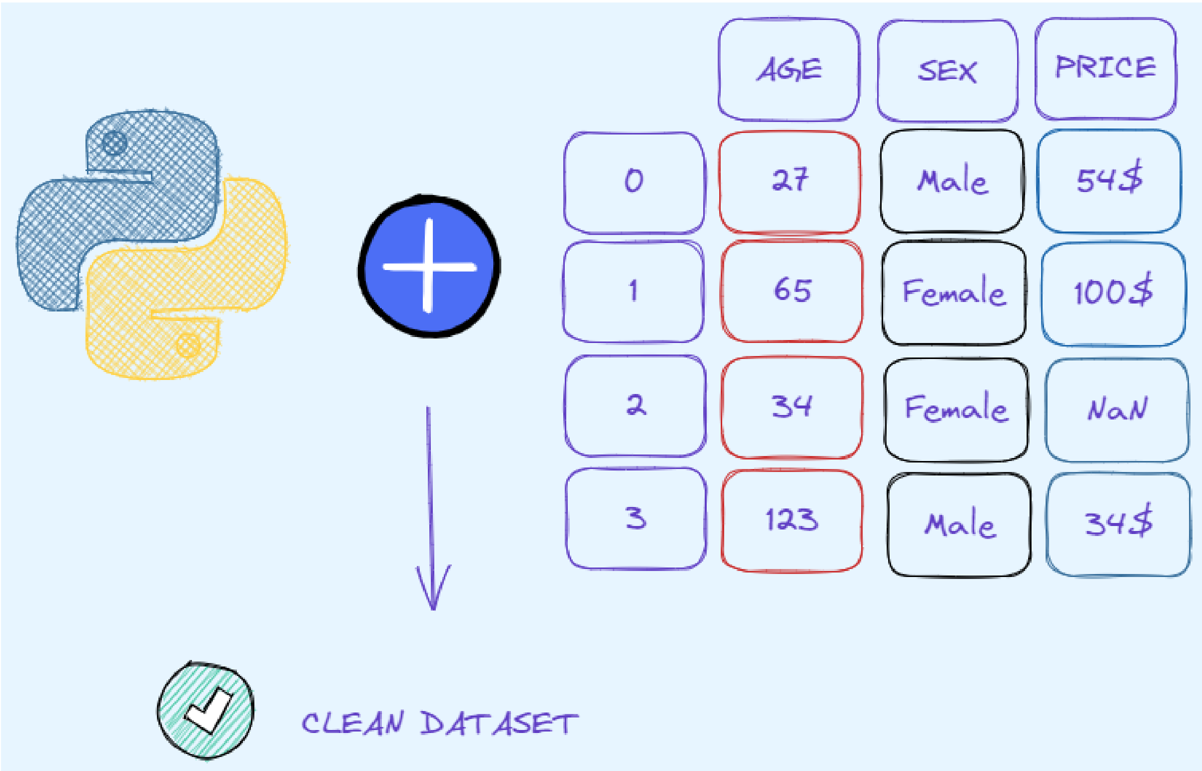

Table of Contents:

1. *Dealing with Missing Data*
2. *Dealing with Duplicates*
3. *Outlier Detection*
4. *Encode Categorical Features*
5. *Transformation*

1. Dealing with Missing data

Check missing data in each column of the dataset

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

Delete missing data

```
df.dropna(how='all')
```

Drop columns that have missing values

```
df.dropna(how='columns')
```

Drop specific columns that have missing values

```
df.dropna(subset=['municipal,'city'])
```

Replace missing values with Mean/Median/Mode

```
df['price'].fillna(df['price'].mean())
df['age'].fillna(df['age'].median())
df['type_building'].fillna(df['type_building'].mode())
```

Replace missing values with Mean/Median/Mode of the group

```
df['price'].fillna(df.group('type_building')['price'].transform('mean'),
inplace=True)
```

Forward Fill - Fill missing values with values before them

```
df['stock_price'].fillna(method='ffill')
```

id	stock_price
1	6.90 USD
2	NaN
3	NaN
4	6.88 USD
5	NaN

id	stock_price
1	6.90 USD
2	6.90 USD
3	6.90 USD
4	6.88 USD
5	6.88 USD

Forward Fill within Groups

```
df['stock_price'] = df.groupby('type_stock').ffill()
```

Backward Fill - FIll missing values with values after them

```
df['stock_price'].fillna(method='bfill')
```

id	stock_price
1	NaN
2	NaN
3	6.90 USD
4	NaN
5	6.88 USD

id	stock_price
1	6.90 USD
2	6.90 USD
3	6.90 USD
4	6.88 USD
5	6.88 USD

Backward Fill within Groups

```
df['stock_price'] = df.groupby('type_stock')['stock_price'].bfill()
```

Fill missing values using the interpolation method

```
df['stock_price'] =
df['stock_price'].interpolate(method='polynomial',order=2)
```

Fill missing values using the interpolation method within groups

```
df['stock_price'] = df.groupby('type_stock')['stock_price'].apply(lambda
x: x.interpolate(method='polynomial',order=2))
```

2. Dealing with Duplicates

Check if there are duplicates

```
df.duplicated().sum()
```

Extract duplicate rows from the dataframe

```
df[df.duplicated()]
```

Drop duplicates

```
df.drop_duplicates()
```

Aggregate data

```
df.groupby('id').agg({'price':'mean'}).reset_index()
```

3. Outlier detection

Detect range of values for each column of the dataset

```
df.describe([x*0.1 for x in range(10)])
```

Display boxplot to display the distribution of a column

```
import seaborn as sns
sns.boxplot(x=df['age'])
```

Display histogram to display the distribution of a column

```
sns.displot(data=df['column1'])
```

Remove outliers

```
df = df[df['age']<df['age'].quantile(0.9)]
```

Outlier detection with machine learning models, like Isolation Forest

```
if = IsolationForest(random_state=42)
if.fit(X)
y_pred = if.predict(X)
```

4. Encode categorical features

Apply one-hot-encoding to a categorical column

```
from sklearn.prepreprocessing import OneHotEncoder
ohe = OneHotEncoder()
encoded_data = pd.DataFrame(ohe.fit_transform(df[['type_build']]).toarray())
new_df = df.join(encoded_data)
```

id	type_build
1	flat
2	house
3	mansion

→

id	flat	house	mansion
1	1	0	0
2	0	1	0
3	0	0	1

Apply label-encoding to a categorical column

```
from sklearn.prepreprocessing import LabelEncoder
le = LabelEncoder()
df['type_build'] = le.fit_transform(df['type_build'])
```

id	type_build
1	flat
2	house
3	mansion

→

id	type_build_enc
1	0
2	1
3	2

Apply ordinal-encoding to a categorical column to retain its ordinal nature

```
from sklearn.prepreprocessing import OrdinalEncoder
le = OrdinalEncoder()
df['price_level'] = le.fit_transform(df['price_level'])
```

5. Transformation

Standardize features by removing the mean and scaling to unit variance

```
from sklearn.processing import StandardScaler
X_std = StandardScaler().transform(X)
```

Rescale features into the range [0,1]

```
from sklearn.processing import MinMaxScaler
X_mms = MinMaxScaler().transform(X)
```

Scale features exploiting statistics that are robust to outliers

```
from sklearn.processing import RobustScaler
X_rs = RobustScaler().transform(X)
```