Anvesh Khode

PRN: 22070521021 | Practical 2: Data preprocessing and cleaning

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, MinMaxScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from scipy.stats import zscore
```

```
# ---- Load Boston Housing from CMU URL ----
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)

# Data wrangling: the dataset is split oddly across rows
X = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
y = raw_df.values[1::2, 2]

# Convert to DataFrame
feature_names = [
    "CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE",
    "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT"
]
X = pd.DataFrame(X, columns=feature_names)
y = pd.Series(y, name="MEDV")
```

```
X
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

506 rows × 13 columns

Next steps: Generate code with X | View recommended plots | New interactive sheet

```
# Step 1: Handle Missing Values
# Introduce some artificial missing values (since Boston Housing usually has none)
X.iloc[0:5, 0] = np.nan
# We use **imputation (mean)** instead of deletion (loses data) or prediction (complex)
imputer = SimpleImputer(strategy="mean")
X = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)
```

```
# Step 2: Handle Outliers
# Outlier detection in "MEDV" target (house prices)
# We will use **removal** method (drop rows with Z-score > 3)

z_scores = np.abs(zscore(X))
X = X[(z_scores < 3).all(axis=1)]
y = y.loc[X.index]
```

```
# Step 3: Encoding categorical data
# Boston has one categorical feature: "CHAS" (Charles River dummy variable: 0 or 1)
# Since it's binary, we’ll use **Label Encoding** (OneHot not needed)
# But for multi-class, OneHot would be used
# Already encoded as 0/1, so nothing more needed
```

```
# Step 4: Feature Scaling
# We'll use **Z-score normalization (StandardScaler)**
scaler = StandardScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)

print("Final cleaned and preprocessed dataset shape:", X_scaled.shape)
print()

X_scaled
```

```
Final cleaned and preprocessed dataset shape: (415, 13)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LST
0	0.236041	0.427278	-1.254663	0.0	-0.083120	0.503296	-0.077906	0.120954	-0.938612	-0.607776	-1.483794	0.439914	-1.1282
1	0.236041	-0.488581	-0.553675	0.0	-0.708718	0.251194	0.410713	0.567909	-0.816465	-0.943176	-0.312853	0.439914	-0.4849
2	0.236041	-0.488581	-0.553675	0.0	-0.708718	1.501880	-0.224135	0.567909	-0.816465	-0.943176	-0.312853	0.360916	-1.2751
3	0.236041	-0.488581	-1.273808	0.0	-0.808451	1.195757	-0.769819	1.125951	-0.694319	-1.067398	0.108686	0.395854	-1.4436
4	0.236041	-0.488581	-1.273808	0.0	-0.808451	1.439673	-0.470228	1.125951	-0.694319	-1.067398	0.108686	0.439914	-1.0740
...	...	...	...	...	...	...	...	...	...	...	...	...	
410	-0.498068	-0.488581	0.162039	0.0	0.234212	0.532762	0.061190	-0.700185	-0.938612	-0.750631	1.185952	0.344612	-0.4029
411	-0.501621	-0.488581	0.162039	0.0	0.234212	-0.241550	0.332249	-0.797567	-0.938612	-0.750631	1.185952	0.439914	-0.4942
412	-0.498451	-0.488581	0.162039	0.0	0.234212	1.159742	0.842267	-0.858716	-0.938612	-0.750631	1.185952	0.439914	-1.0261
413	-0.488456	-0.488581	0.162039	0.0	0.234212	0.861804	0.781636	-0.745895	-0.938612	-0.750631	1.185952	0.372950	-0.8962
414	-0.501183	-0.488581	0.162039	0.0	0.234212	-0.388882	0.478478	-0.686733	-0.938612	-0.750631	1.185952	0.439914	-0.6797

415 rows × 13 columns

Next steps: Generate code with X_scaled | View recommended plots | New interactive sheet