10/03/25

ML-LAB-01

⇒ housing.csv

```
import pandas as pd

housing_df = pd.read_csv("/content/housing.csv")

print("Information of all columns:")
print(housing_df.info())

print("\nStatistical information of numerical columns")
print(housing_df.describe())

print("\nCount of unique labels for 'Ocean Proximity' column:")
if 'ocean_proximity' in housing_df.columns:
    print(housing_df['ocean_proximity'].value_counts())
else:
    print(f"Column 'ocean_proximity' not found in the dataFrame. Available columns: {housing_df.columns.tolist()}")
print("\nColumns with missing values:")
missing_values = housing_df.isnull().sum()
print(missing_values[missing_values>0])
```

__/__/____

2

⟹ *Handling Missing data,
- Data processing techniques for Diabetes and Adult income datasets.

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

diabetes_file_path = '/content/Dataset of Diabetes.csv'
diabetes_df = pd.read_csv(diabetes_file_path)

numeric_cols = diabetes_df.select_dtypes(include=
    np.number).columns
categorical_cols = diabetes_df.select_dtypes(exclude
    = np.number).columns

imputer_numeric = SimpleImputer(strategy='mean')
diabetes_df[numeric_cols] = imputer_numeric.fit_
    transform(diabetes_df[numeric_cols])

imputer_categorical = SimpleImputer(strategy='most_
    frequent')
diabetes_df[categorical_cols] = imputer_categorical.
    fit_transform(diabetes_df[categorical_cols])
```

=> observations on Data processing techniques for the Diabetes & Adult income datasets

1. which columns in the Data set had missing values? How did you handle them?

Diabetes dataset:

→ columns with missing value: Urea, Cr, HbA1c, Chol, TG, HDL, LDL, VLDL, BMI.

→ For numeric columns for missing ~~dat~~ values, we applied mean imputation. The missing values were replaced with the mean of each respective column.

→ For categorical columns missing values were handled using most ~~fe~~ frequent value (mode) imputation.

Adult income Dataset:

→ ~~columns~~: In this dataset, missing values were represented by ? in the og dataset

→ we replaced all occurances of ? with NaN to make the missing values identifiable

→ After this, we applied mode imputation for categorical columns to replace missing val

2. which categorical columns did you identify in the dataset. How did you encode them?

Diabetes:

→ columns: gender, Race, class

→ For encoding categorical variables, one not encoding or label encoding could be applied to convert into binary columns.

Adult Income Dataset:

→ columns: work class, education, marital-status, occupation, relationship, race, sex, native-country

→ one-hot encoding was applied to convert the categorical columns into multiple binary columns, For instance, work class would be converted into multiple binary columns

→ Using pd.get_dummies() with drop_first=True allowed us to avoid multicollinearity by removing the first column from each categorical variable set.

3. what is the difference b/w Min-Max scaling and standardization? when would you use one over the other?

→ Min-Max Scaling:

(or) normalization transforms data to a specific range usually b/w 0 & 1

__/__/____

- using the formula

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

<u>when to use</u>: Min-max scaling is prefered when the dataset contains features with varying units @ scales & you want to scale them to a fixed range.

<u>Standerdization</u>:

(or) z-score normalization transforms data to ha a mean of 0 & a standard deviation of 1. If 'i' calculated using the formula:

$$X_{scaled} = \frac{X - \mu}{\sigma}$$

μ - mean & σ - standard deviation.

→ standerdisation is typically used when the data is assumed to follow a gaussian distribution, especially when working with algorithms that assume normally distributed data regression, logistic regression.

Snehal B
10/3/25