#Preprocessed data in Assignment 4

from google.colab import drive

drive.mount('/content/drive')

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

file\_path = '/content/drive/My Drive/CSE422 LAB/Assignment 4 (ML)/Dataset/Housing Price.xlsx'

df = pd.read\_excel(file\_path)

# Task 1: Remove null values

df\_cleaned = df.dropna()

# Task 2: Remove duplicate rows

df\_cleaned = df\_cleaned.drop\_duplicates()

# Task 3: Handle categorical variables (Binary encoding and One-Hot Encoding)

binary\_columns = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']

df\_cleaned[binary\_columns] = df\_cleaned[binary\_columns].replace({'yes': 1, 'no': 0})

# One-Hot Encoding for 'furnishingstatus'

df\_cleaned = pd.get\_dummies(df\_cleaned, columns=['furnishingstatus'], drop\_first=True)

# Task 4: Feature scaling for continuous variables

scaler = StandardScaler()

df\_cleaned[['price', 'area', 'parking']] = scaler.fit\_transform(df\_cleaned[['price', 'area', 'parking']])

# Task 5: Remove variables with high correlation (threshold > 0.8)

corr\_matrix = df\_cleaned.corr().abs()

upper = corr\_matrix.where(np.triu(np.ones(corr\_matrix.shape), k=1).astype(bool))

to\_drop = [column for column in upper.columns if any(upper[column] > 0.8)]

df\_cleaned.drop(columns=to\_drop, inplace=True)

#Assignment 5 tasks

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression, LogisticRegression

from sklearn.metrics import mean\_squared\_error, accuracy\_score

#Linear Regression

X = df\_cleaned.drop(columns=['price'])

y = df\_cleaned['price']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

linear\_model = LinearRegression()

linear\_model.fit(X\_train, y\_train)

y\_pred\_linear = linear\_model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred\_linear)

rmse = np.sqrt(mse)

print(f"Linear Regression MSE: {mse}")

print(f"Linear Regression RMSE: {rmse}")

#Logistic Regression

threshold = y.median()

y\_binary = (y > threshold).astype(int)

X\_train\_log, X\_test\_log, y\_train\_log, y\_test\_log = train\_test\_split(X, y\_binary, test\_size=0.3, random\_state=42)

logistic\_model = LogisticRegression()

logistic\_model.fit(X\_train\_log, y\_train\_log)

y\_pred\_log = logistic\_model.predict(X\_test\_log)

accuracy = accuracy\_score(y\_test\_log, y\_pred\_log)

print(f"Logistic Regression Accuracy: {accuracy}")