| **Practical 1**  To study and implement linear regression on house price prediction and  smartphone datasets. |
| --- |

| **Problem Description:** The objective is to understand how various features within the house price and smartphone datasets influence their pricing. By implementing a Linear Regression model, we aim to predict these prices accurately. The approach involves cleaning and preparing the data, selecting the appropriate features and evaluating the model's performance using MSE to obtain reliable predictions. |
| --- |

| **Solution Architecture:** |
| --- |
| **Code:**  **→ 1. House Price Prediction Implementation**  import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import mean\_squared\_error  import numpy as np  def train\_multiple\_linear\_regression(csv\_file):  data = pd.read\_csv(csv\_file, encoding='ISO-8859-1')    X = data.iloc[:, :6].values # Features  Y = data.iloc[:, 6].values # Target\_value    # Train-Test Splitting  X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)    # Initializing the LR Model  model = LinearRegression()    # Training the Model  model.fit(X\_train, Y\_train)    # prediction on test data  Y\_pred = model.predict(X\_test)    # Calculation of the Mean Squared Error  mse = mean\_squared\_error(Y\_test, Y\_pred)    return model, mse  def predict\_new\_input(model, new\_input):  new\_input\_array = np.array(new\_input).reshape(1, -1)  predicted\_value = model.predict(new\_input\_array)    return predicted\_value  # Demo Use  csv\_file = '/Users/angatshah0511/Desktop/re - re.csv'  model, mse = train\_multiple\_linear\_regression(csv\_file)  print(f'--> Mean Squared Error : {mse}')  new\_input = [2018, 5, 20, 8, 24.98298, 121.54024]  predicted\_value = predict\_new\_input(model, new\_input)  print(f'--> Predicted Value : {predicted\_value}')  **→ 2. Smart Phone Price Prediction Implementation**  import pandas as pd  import numpy as np  import re  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import mean\_squared\_error  from scipy.stats import rankdata  data = pd.read\_csv('/Users/angatshah0511/Desktop/smart\_phone\_dataset.csv')  data = data.drop(columns=['Unnamed: 0'])  data.head()  def extract\_numeric(value):  match = re.search(r'\d+', str(value))  return int(match.group()) if match else None  def extract\_android\_version(memory\_info):  match = re.search(r'Android v(\d+)', memory\_info)  match2 = re.search(r'HarmonyOS v(\d+)', memory\_info)  match3 = re.search(r'EMUI v(\d+)', memory\_info)  if match:  return match.group(1)  elif match2:  return match2.group(1)  elif match3:  return match3.group(1)  return None  def extract\_max\_megapixel(value):  megapixels = re.findall(r'\d+\.?\d\*', value)  return max(map(float, megapixels)) if megapixels else None  def data\_cleaning(data):  data['Ram'] = data['Ram'].apply(extract\_numeric)  data['Battery'] = data['Battery'].apply(extract\_numeric)  data['Display'] = data['Display'].apply(lambda x: float(re.search(r'\d+(\.\d+)?', str(x)).group()))  data['Rating'] = pd.to\_numeric(data['Rating'], errors='coerce')  data['Spec\_score'] = pd.to\_numeric(data['Spec\_score'], errors='coerce')  data['fast\_charging'] = data['fast\_charging'].apply(extract\_numeric)  data['Processor'] = data['Processor'].apply(lambda x: 'Octa Core' in x if isinstance(x, str) else False)  data['Inbuilt\_memory'] = data['Inbuilt\_memory'].apply(extract\_numeric)  missing\_android\_version = data['Android\_version'].isnull()  extracted\_versions = data.loc[missing\_android\_version, 'External\_Memory'].apply(extract\_android\_version)  data.loc[missing\_android\_version, 'Android\_version'] = extracted\_versions  data.loc[missing\_android\_version & extracted\_versions.notnull(), 'External\_Memory'] = 'Memory Card Not Supported'  data['Android\_version'] = data['Android\_version'].apply(extract\_numeric)  data = data.dropna(subset=['Android\_version'])  data = data.dropna(subset=['Inbuilt\_memory', 'No\_of\_sim'])  data['fast\_charging'].fillna(5, inplace=True)  data['fast\_charging'] = data['fast\_charging'].astype(float)  data['Price'] = data['Price'].str.replace(',', '').astype(float)  data['Camera'] = data['Camera'].apply(extract\_max\_megapixel)  data['External\_Memory\_GB'] = data['External\_Memory'].str.extract(r'(\d+) TB|(\d+) GB').apply(lambda x: x[0] if pd.notna(x[0]) else x[1], axis=1).astype(float)  data['External\_Memory\_GB'] = data['External\_Memory\_GB'].fillna(0) \* np.where(data['External\_Memory'].str.contains('TB'), 1024, 1)  data = data.drop(columns=['External\_Memory'])  data['Company'] = data['Name'].str.split().str[0]  data = data.drop(columns=['Name'])  brand\_priority = {  'Samsung': 95, 'Google': 90, 'OnePlus': 85, 'Sony': 80, 'Xiaomi': 75, 'Motorola': 70, 'Nokia': 65, 'Realme': 60, 'Oppo': 60, 'Vivo': 60,  }  data['Brand\_Priority'] = data['Company'].map(brand\_priority)  data['Brand\_Priority'].fillna(50, inplace=True)  data = data.dropna()  return data  def data\_preprocessing(data):  for column in ['Ram', 'Battery', 'Display', 'Rating', 'Spec\_score', 'fast\_charging', 'Inbuilt\_memory', 'Android\_version', 'Camera']:  data[f'{column}'] = rankdata(data[column]) / len(data[column]) \* 100  X = data[['Ram', 'Battery', 'Display', 'Rating', 'Spec\_score', 'fast\_charging', 'Processor', 'Inbuilt\_memory', 'Android\_version', 'Camera', 'Brand\_Priority']]  y = np.log(data['Price'])  return X, y  cleaned\_data = data\_cleaning(data)  cleaned\_data.head()  X\_processed, Y\_processed = data\_preprocessing(cleaned\_data)  X\_processed.head()  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_processed, Y\_processed, test\_size=0.20, random\_state=21)  model = LinearRegression()  model.fit(X\_train, y\_train)  y\_pred = model.predict(X\_test)  mse = mean\_squared\_error(y\_test, y\_pred)  accuracy = model.score(X\_test, y\_test)  print(f'--> Accuracy: {accuracy \* 100}%')  print(f'--> Mean Squared Error: {mse}')  test\_df = pd.DataFrame({'Actual Price': np.exp(y\_test), 'Predicted Price': np.exp(y\_pred)})  print(test\_df.head(10)) |

| **Results:**  **→ 1. House Price Prediction Output**    **→ 2. Smart Phone Price Prediction Output** |
| --- |