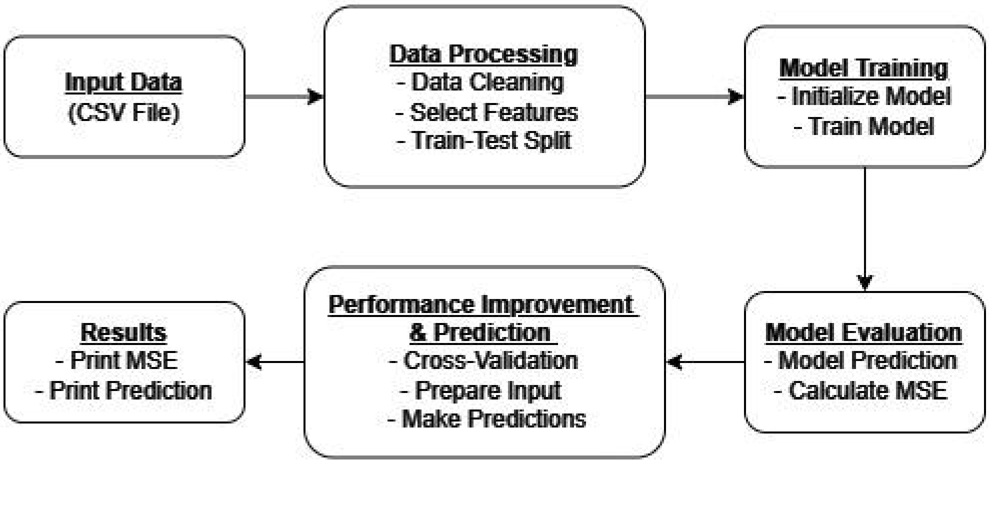
Practical 1

To study and implement linear regression on house price prediction and smartphone datasets.

Problem Description:

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}')
```

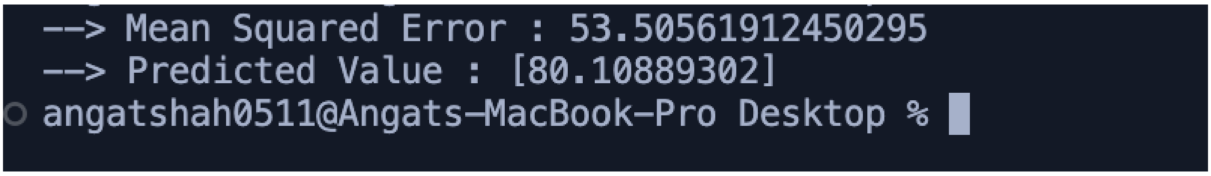
→ 2. Smart Phone Price Prediction Implementation

```
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))
```

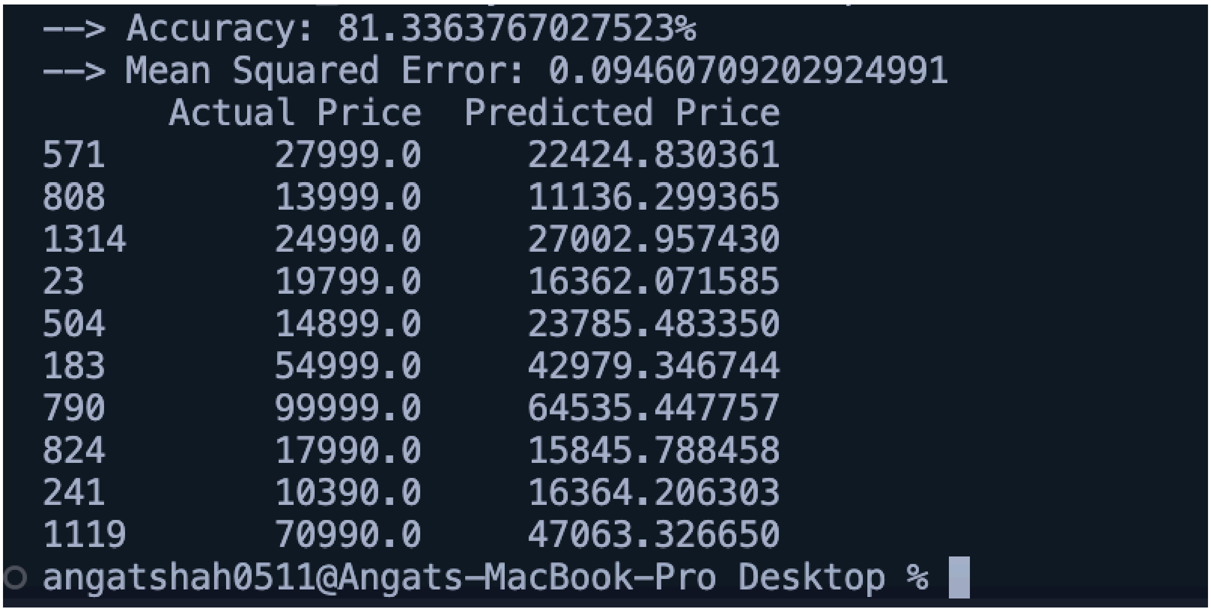
Results:

→ **1. House Price Prediction Output**



```
  --> Mean Squared Error : 53.50561912450295
  --> Predicted Value : [80.10889302]
○ angatshah0511@Angats-MacBook-Pro Desktop %
```

→ **2. Smart Phone Price Prediction Output**



```
  --> Accuracy: 81.3363767027523%
  --> Mean Squared Error: 0.09460709202924991
        Actual Price  Predicted Price
  571        27999.0     22424.830361
  808        13999.0     11136.299365
  1314       24990.0     27002.957430
  23         19799.0     16362.071585
  504        14899.0     23785.483350
  183        54999.0     42979.346744
  790        99999.0     64535.447757
  824        17990.0     15845.788458
  241        10390.0     16364.206303
  1119       70990.0     47063.326650
○ angatshah0511@Angats-MacBook-Pro Desktop %
```