Linear Regression Assignment

Certainly! To create a model for house price prediction using linear regression, we'll follow these steps:

1. \*\*Understand the Dataset\*\*: Assess the features (independent variables) and the target variable (house prices).

2. \*\*Preprocess the Data\*\*: Handle missing values, encode categorical variables, and scale numerical features if necessary.

3. \*\*Split the Data\*\*: Divide the data into training and testing sets.

4. \*\*Build the Model\*\*: Implement linear regression.

5. \*\*Evaluate the Model\*\*: Assess the performance using metrics like R-squared, Mean Absolute Error (MAE), or Mean Squared Error (MSE).

6. \*\*Fine-tune and Optimize\*\*: Perform any necessary adjustments to improve the model.

Assuming we have a dataset with features such as number of rooms, square footage, location, etc., let's go through the steps in Python.

### Step 1: Understand the Dataset

For example, let’s assume your dataset (`house\_prices.csv`) has the following columns:

- `num\_rooms`: Number of rooms

- `square\_footage`: Size of the house in square feet

- `location`: Categorical variable indicating the location of the house

- `price`: Target variable indicating the house price

### Step 2: Preprocess the Data

First, we will load and inspect the dataset, then handle missing values and encode categorical variables.

```python

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import OneHotEncoder

# Load dataset

df = pd.read\_csv('house\_prices.csv')

# Inspect the dataset

print(df.head())

print(df.info())

print(df.describe())

# Handle missing values (if any)

df = df.dropna()

# Encode categorical variables (OneHotEncoding for location)

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

# Separate features and target variable

X = df.drop('price', axis=1)

y = df['price']

```

### Step 3: Split the Data

```python

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

```

### Step 4: Build the Model

```python

# Initialize the linear regression model

model = LinearRegression()

# Train the model

model.fit(X\_train, y\_train)

```

### Step 5: Evaluate the Model

```python

# Make predictions

y\_pred = model.predict(X\_test)

# Calculate performance metrics

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

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

print(f'R-squared: {r2}')

```

### Step 6: Fine-tune and Optimize

You might consider additional steps such as feature scaling, polynomial features, or cross-validation for fine-tuning:

```python

# Example of feature scaling

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Retrain the model with scaled features

model.fit(X\_train\_scaled, y\_train)

# Re-evaluate the model

y\_pred\_scaled = model.predict(X\_test\_scaled)

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

r2\_scaled = r2\_score(y\_test, y\_pred\_scaled)

print(f'Mean Squared Error (scaled): {mse\_scaled}')

print(f'R-squared (scaled): {r2\_scaled}')

```

### Complete Example Code

Here’s the complete code for clarity:

```python

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import OneHotEncoder

# Load dataset

df = pd.read\_csv('house\_prices.csv')

# Inspect the dataset

print(df.head())

print(df.info())

print(df.describe())

# Handle missing values (if any)

df = df.dropna()

# Encode categorical variables (OneHotEncoding for location)

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

# Separate features and target variable

X = df.drop('price', axis=1)

y = df['price']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize the linear regression model

model = LinearRegression()

# Train the model

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Calculate performance metrics

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

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

print(f'R-squared: {r2}')

# Example of feature scaling

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Retrain the model with scaled features

model.fit(X\_train\_scaled, y\_train)

# Re-evaluate the model

y\_pred\_scaled = model.predict(X\_test\_scaled)

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

r2\_scaled = r2\_score(y\_test, y\_pred\_scaled)

print(f'Mean Squared Error (scaled): {mse\_scaled}')

print(f'R-squared (scaled): {r2\_scaled}')

```

This code provides a straightforward example of building a linear regression model for house price prediction. You can further improve and fine-tune the model based on your specific dataset and requirements.