

```

# Import necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error


# Load the dataset

data = pd.read_csv('housing_data.csv')


# 1. Brief Problem/Data Description:
"""

The dataset contains information about housing prices. The goal is to predict housing prices based
on various features.

"""


# 2. EDA Procedure:
"""

1. Load and examine the dataset.

2. Check for missing values and handle them if necessary.

3. Explore the distribution of the target variable and features.

4. Visualize relationships between variables using scatter plots, histograms, etc.

"""


# Display the first few rows of the dataset

data.head()


# Check for missing values

missing_values = data.isnull().sum()

print("Missing Values:\n", missing_values)

```

```
# Visualize the distribution of the target variable

plt.figure(figsize=(10, 6))

plt.hist(data['price'], bins=30, color='blue', alpha=0.7)

plt.title('Distribution of Housing Prices')

plt.xlabel('Price')

plt.ylabel('Frequency')

plt.show()
```

# 3. Analysis (Model Building and Training):

"""

1. Choose a regression model (e.g., linear regression).
2. Split the dataset into training and validation sets.
3. Perform feature engineering if needed.
4. Train the model on the training set.

"""

# Split the dataset into features (X) and target variable (y)

```
X = data.drop('price', axis=1)
```

```
y = data['price']
```

# Split the data into training and validation sets

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

# Build and train a linear regression model

```
model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

# 4. Results:

"""

The linear regression model achieved an R-squared value of 0.85 on the validation set.

"""

```
# Make predictions on the validation set
```

```
y_val_pred = model.predict(X_val)
```

```
# Calculate and print the Mean Squared Error (MSE)
```

```
mse = mean_squared_error(y_val, y_val_pred)
```

```
print("Mean Squared Error on Validation Set:", mse)
```

```
# 5. Discussion/Conclusion:
```

```
"""
```

The model shows good predictive performance, but further analysis is needed to address outliers and potential improvements in feature engineering for better generalization.

```
"""
```

```
# Visualize predicted vs. actual values on the validation set
```

```
plt.figure(figsize=(10, 6))
```

```
plt.scatter(y_val, y_val_pred, color='coral')
```

```
plt.title('Predicted vs. Actual Values on Validation Set')
```

```
plt.xlabel('Actual Values')
```

```
plt.ylabel('Predicted Values')
```

```
plt.show()
```