untitled4

November 13, 2025

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv('/content/housing[1].csv')

display(df.head())
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

[2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
```

```
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

[8]:
```
print(df.isnull().sum())
```

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

[29]:
```
df_cleaned = df.dropna()
print(f"Original DataFrame shape: {df.shape}")
print(f"DataFrame shape after dropping nulls: {df_cleaned.shape}")

display(df_cleaned.info())
```

```
Original DataFrame shape: (20640, 14)
DataFrame shape after dropping nulls: (20640, 14)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 14 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   longitude                   20640 non-null  float64
 1   latitude                    20640 non-null  float64
 2   housing_median_age          20640 non-null  float64
 3   total_rooms                 20640 non-null  float64
 4   total_bedrooms              20640 non-null  float64
 5   population                  20640 non-null  float64
 6   households                  20640 non-null  float64
 7   median_income               20640 non-null  float64
 8   median_house_value          20640 non-null  float64
 9   ocean_proximity_<1H OCEAN   20640 non-null  bool
 10  ocean_proximity_INLAND      20640 non-null  bool
 11  ocean_proximity_ISLAND      20640 non-null  bool
 12  ocean_proximity_NEAR BAY    20640 non-null  bool
```

```
 13  ocean_proximity_NEAR OCEAN  20640 non-null  bool
dtypes: bool(5), float64(9)
memory usage: 1.5 MB
```

None

[9]:
```
print(f"Number of duplicate rows: {df.duplicated().sum()}")
```

```
Number of duplicate rows: 0
```

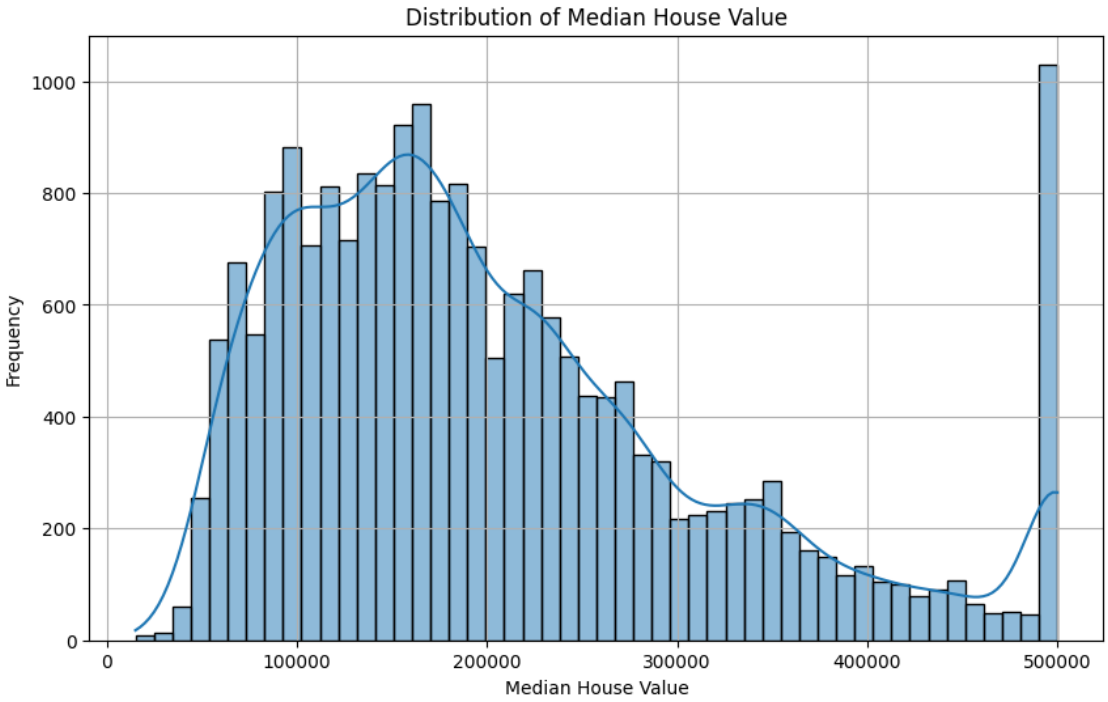
[4]:
```
X = df[['median_income']]
y = df['median_house_value']

if X.isnull().sum().any() or y.isnull().sum().any():
    print("Warning: Missing values detected. Dropping rows with missing data␣
 ↪for selected feature/target.")
    combined = pd.concat([X, y], axis=1).dropna()
    X = combined[['median_income']]
    y = combined['median_house_value']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

print(f"Training data size: {len(X_train)} samples")
print(f"Testing data size: {len(X_test)} samples")
```

```
Training data size: 16512 samples
Testing data size: 4128 samples
```

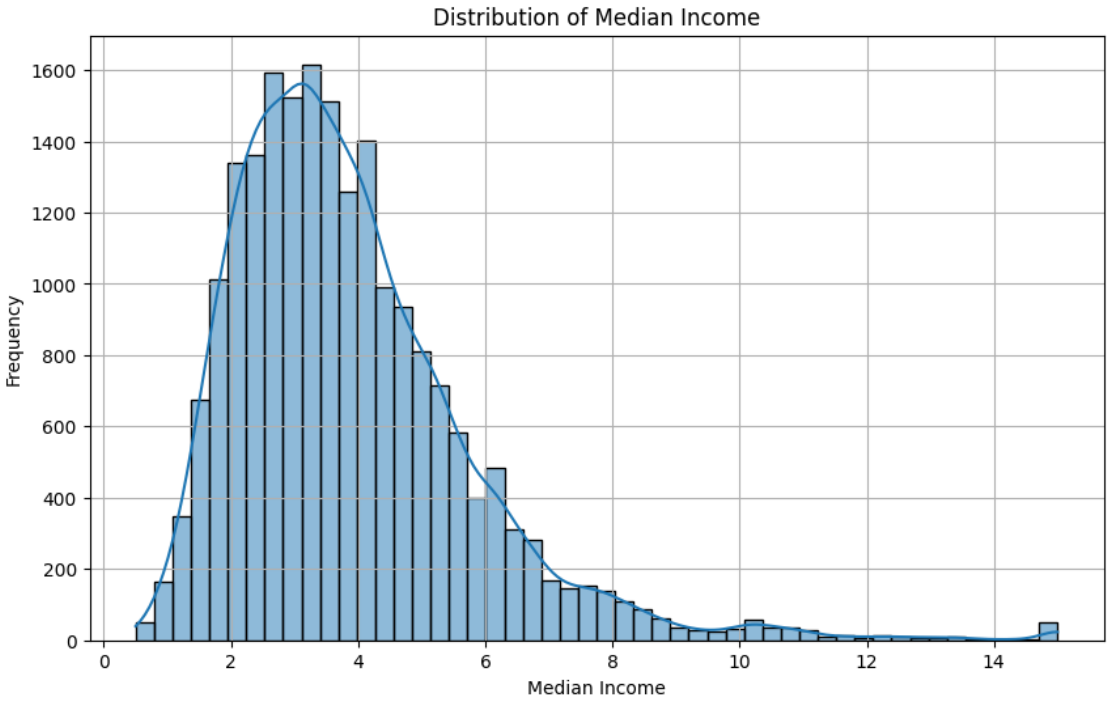
[5]:
```
model = LinearRegression()
model.fit(X_train, y_train)

print(f"Model trained successfully. Coefficients: {model.coef_[0]:.2f},␣
 ↪Intercept: {model.intercept_:.2f}")
```

```
Model trained successfully. Coefficients: 41933.85, Intercept: 44459.73
```

[10]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df['median_house_value'], bins=50, kde=True)
plt.title('Distribution of Median House Value')
plt.xlabel('Median House Value')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



[11]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df['median_income'], bins=50, kde=True)
plt.title('Distribution of Median Income')
plt.xlabel('Median Income')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

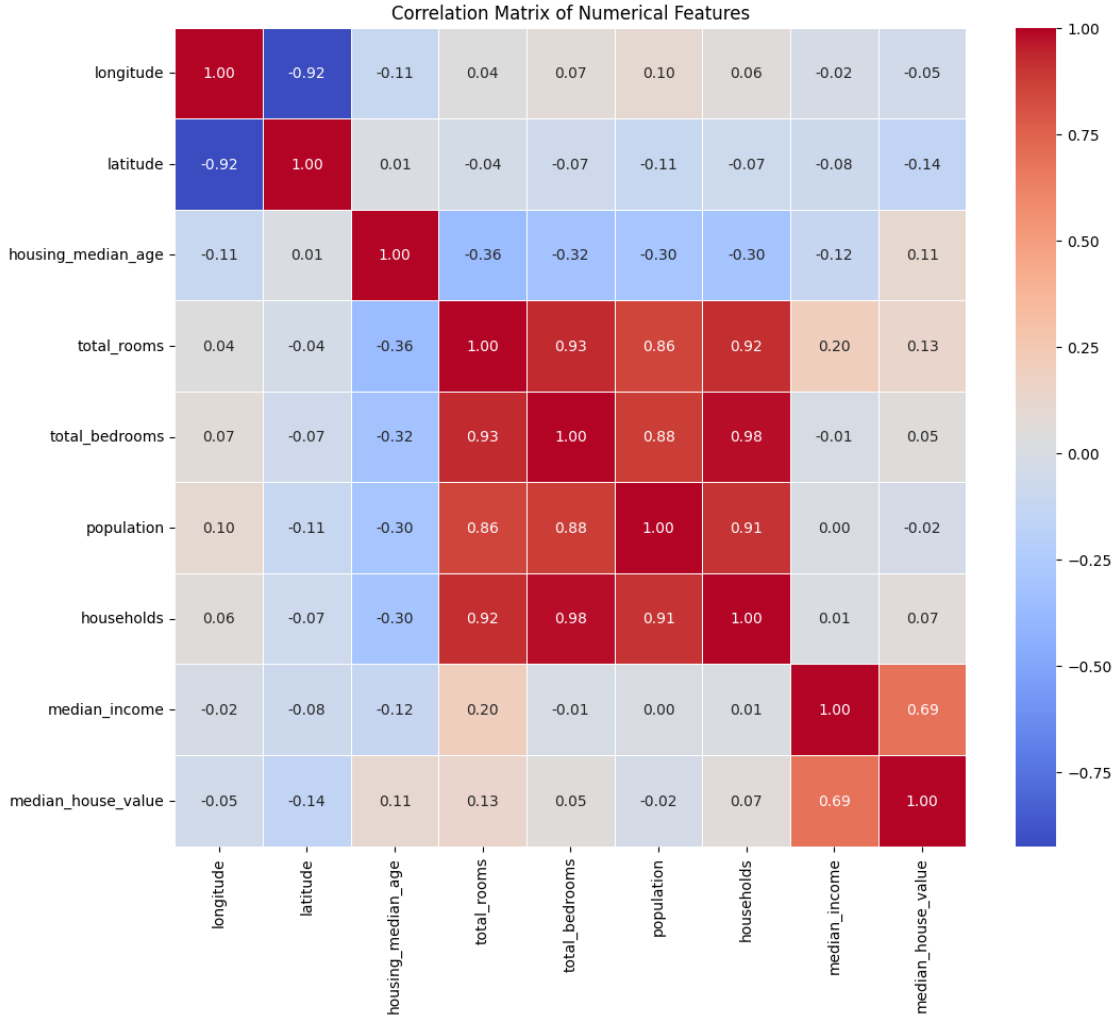


[12]:
```
residuals = y_test - y_pred

plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_pred, y=residuals, alpha=0.6)
plt.axhline(y=0, color='red', linestyle='--', label='Zero Residuals')
plt.title('Residuals Plot')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.legend()
plt.grid(True)
plt.show()
```

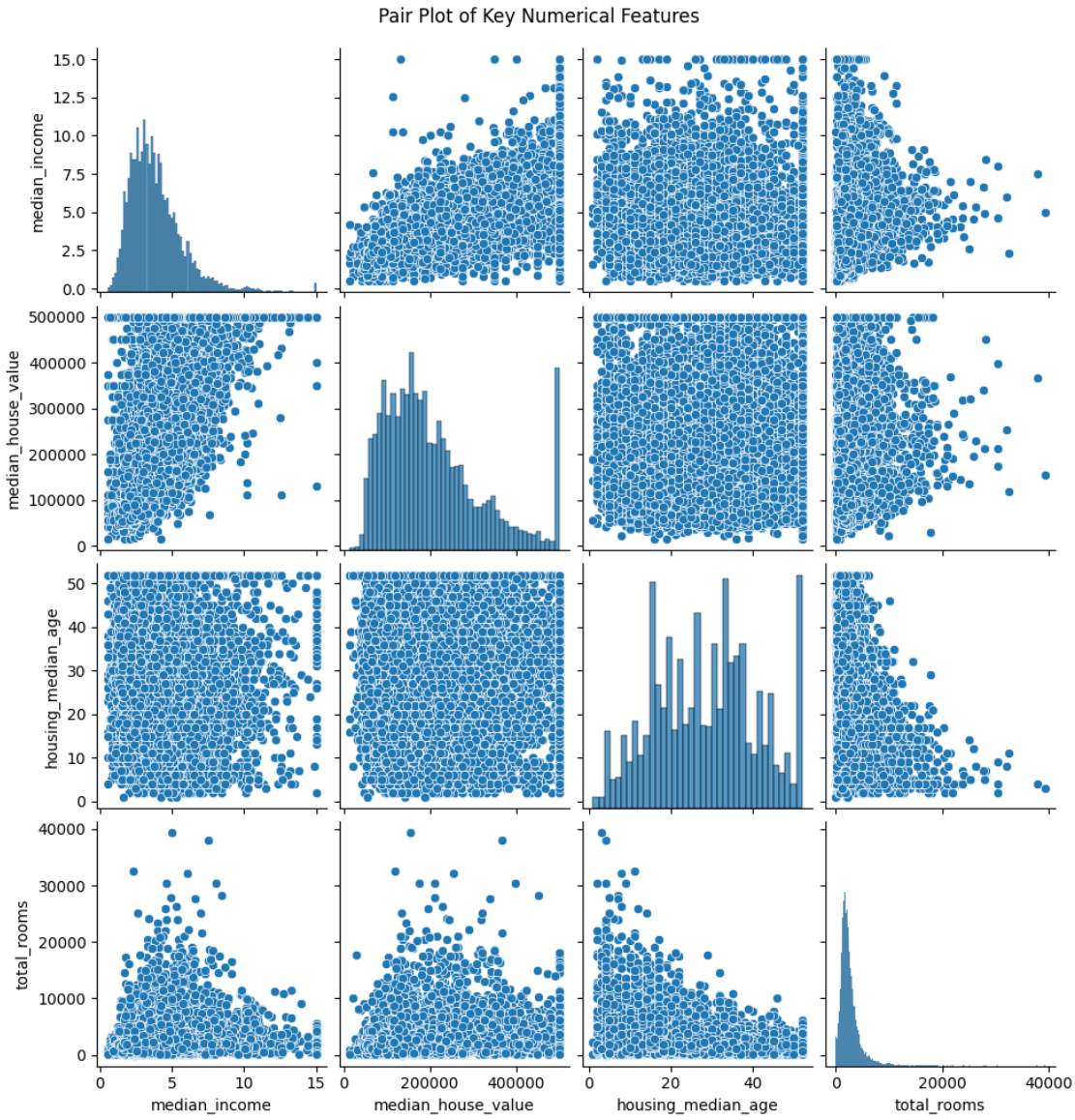
[13]:
```
numerical_df = df.select_dtypes(include=np.number)
correlation_matrix = numerical_df.corr()
display(correlation_matrix.head())
```

```
                    longitude  latitude  housing_median_age  total_rooms  \
longitude            1.000000 -0.924664           -0.108197     0.044568
latitude            -0.924664  1.000000            0.011173    -0.036100
housing_median_age  -0.108197  0.011173            1.000000    -0.361262
total_rooms          0.044568 -0.036100           -0.361262     1.000000
total_bedrooms       0.069608 -0.066983           -0.320451     0.930380

                    total_bedrooms  population  households  median_income  \
longitude                 0.069608    0.099773    0.055310      -0.015176
latitude                 -0.066983   -0.108785   -0.071035      -0.079809
housing_median_age       -0.320451   -0.296244   -0.302916      -0.119034
total_rooms               0.930380    0.857126    0.918484       0.198050
total_bedrooms            1.000000    0.877747    0.979728      -0.007723

                    median_house_value
longitude                    -0.045967
latitude                     -0.144160
housing_median_age            0.105623
total_rooms                   0.134153
total_bedrooms                0.049686
```

[14]:
```
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
↪linewidths=.5)
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```



[15]:
```
selected_features = ['median_income', 'median_house_value',␣
↪'housing_median_age', 'total_rooms']
sns.pairplot(df[selected_features])
plt.suptitle('Pair Plot of Key Numerical Features', y=1.02)
```



[6]:
```
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
Mean Squared Error: 7091157771.77
R-squared: 0.46
```

[7]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_test['median_income'], y=y_test, alpha=0.6, label='Actual␣
 ↪Values')
plt.plot(X_test['median_income'], y_pred, color='red', linewidth=2,␣
 ↪label='Regression Line')
plt.title('Simple Linear Regression: Median Income vs. Median House Value')
plt.xlabel('Median Income')
plt.ylabel('Median House Value')
plt.legend()
plt.grid(True)
plt.show()
```