S-CSIS311_EA3_Regression_Imperial

September 11, 2024

De La Salle University – Dasmariñas
College of Information and Computer Studies

S-CSIS311 / S-CSIS311LA
Introduction to Machine Learning

Enabling Assessment: Regression
Wednesday, September 11, 2024

Luis Anton P. Imperial
BCS32

1 Enabling Assessment – Regression

1.1 Learning Outcome:

1. Students will be able to explain the basic concept of linear regression.
2. Students will be able to build a simple regression model in Google Colab.
3. Students will be able to use Google Colab to visualize key metrics.

1.2 Direction:

Using Anaconda or Google Colab, solve the machine problem. Evaluate, analyze, and explain the steps using your chosen tool. Use the dataset housing. After completing the solution, create two copies: one in PDF format and one in Python (.py) format. Submit both files to the folder provided in MS Teams.

1.3 Steps:

1. Import the libraries
2. Load the data
3. Setup the data
4. Prepare the data processing
5. Train the model
6. Test the model
7. Print the predictions
8. Visualize the model

1.4 Objective:

To predict the Median House Value based on Median Income

[1]:
```
# Importing libraries required
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

SEED = 42
np.random.seed(SEED)
```

1.5 Questions:

1. Load the dataset
2. Load the data frame.head()
3. Load the df = df[['median_income', 'median_house_value']]
4. Display dataframe information using df.info() and display df.head(10)
5. What values do you see?
6. What distributions do you see?
7. What relationships do you see?
8. What relationships do you think might benefit the prediction problem?
9. What ideas about the domain does the data spark?
10. Display all columns, head, tail and info using df.columns, df.head(), df.tail() and df.info()
11. Get an overall sense of the data shape using df.describe()
12. Group the data and sort in ascending order
13. Plot the data using scatterplot
14. Removing outliers
15. Visualize the trained model

[13]:
```
#### 1. Load the dataset
from google.colab import drive
drive.mount('/content/drive')
url = '/content/drive/MyDrive/Documents/Reference Documents/BCS3 CSIS311␣
↪Reference Documents/Supplementary BCS3 CSIS311 Reference Documents/housing.
↪csv'
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[14]:
```
#### 2. Load the data frame.head()
df = pd.read_csv(url)
df.head()
```

[14]:
```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
```

```
4    -122.25     37.85                 52.0        1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
```

1.5.1 (2/15) Loading the dataset

As we can see from the previous code block, we have properly loaded the Comma-Separated Values (CSV) file we will rely on as our dataset.

The `head()` function of the Pandas package displays the "head", meaning the first few rows, of a given dataset. It is often used to test for the program's access to the data.

[5]:
```python
#### 3. Load the df = df[['median_income', 'median_house_value']]
df = df[['median_income', 'median_house_value']]
```

[6]:
```python
#### 4. Display dataframe information using df.info() and display df.head(10)
df.info()
df.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 2 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   median_income       20640 non-null  float64
 1   median_house_value  20640 non-null  float64
dtypes: float64(2)
memory usage: 322.6 KB
```

[6]:
```
   median_income  median_house_value
0         8.3252            452600.0
1         8.3014            358500.0
2         7.2574            352100.0
3         5.6431            341300.0
4         3.8462            342200.0
5         4.0368            269700.0
6         3.6591            299200.0
7         3.1200            241400.0
8         2.0804            226700.0
9         3.6912            261100.0
```

1.5.2 (5/15) Peeking through people's houses

5. What values do you see?
6. What distributions do you see?
7. What relationships do you see?
8. What relationships do you think might benefit the prediction problem?
9. What ideas about the domain does the data spark?

The values provided are the median income and the median house value in each of the first ten (out of 20,640) households in the dataset.

The distribution would be somewhat of a straight line going from bottom-left to top-right, considering that a glance of the dataset's head gives the impression of a positive correlation between a neighborhood's median income and the home's median value.

Perhaps the relationships that can be formed are one-to-many and one-to-one. The median housing value can rise not just because it is located in a neighborhood with higher-income residents; it can also rise depending on other statistics such as its proximity to a body of water.

I believe one-to-many relationships that also include other factors such as the median age of the houses in each area, as well as the total number of rooms each house can provide, can help predict the value of newly built and planned houses.

In simpler terms, the data suggests that the richer the family, the more expensive their house can get. *Sounds like a fairly understood overview, isn't it?*

10. Display all columns, head, tail and info using df.columns, df.head(), df.tail() and df.info()
11. Get an overall sense of the data shape using df.describe()

[17]:
```
#### 10. Display all columns, head, tail and info using df.columns, df.head(),
↪df.tail() and df.info()
df = pd.read_csv(url)

df.columns
df.head()
df.tail()
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
```

[18]:
```
#### 11. Get an overall sense of the data shape using df.describe()
df.describe()
```

[18]:
```
          longitude      latitude  housing_median_age   total_rooms  \
count  20640.000000  20640.000000        20640.000000  20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081
std        2.003532      2.135952           12.585558   2181.615252
min     -124.350000     32.540000            1.000000      2.000000
25%     -121.800000     33.930000           18.000000   1447.750000
50%     -118.490000     34.260000           29.000000   2127.000000
75%     -118.010000     37.710000           37.000000   3148.000000
max     -114.310000     41.950000           52.000000  39320.000000

       total_bedrooms    population    households  median_income  \
count    20433.000000  20640.000000  20640.000000   20640.000000
mean       537.870553   1425.476744    499.539680       3.870671
std        421.385070   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        296.000000    787.000000    280.000000       2.563400
50%        435.000000   1166.000000    409.000000       3.534800
75%        647.000000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
```

[19]:
```
##### This will show only features that have nonzero missing values.
df_na = df.isna().sum()
df_na
```

[19]:
```
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
```

[20]:
```
##### Limit to categorical data using df.select_dtypes()
df_cat = df.select_dtypes(include=['object'])
df_cat.nunique()
```

[20]:
```
ocean_proximity    5
dtype: int64
```

[21]:
```
##### Limit to numerical data df.select_dtypes()
df_num = df.select_dtypes(include=['number'])
df_num.nunique()
```

[21]:
```
longitude               844
latitude                862
housing_median_age       52
total_rooms            5926
total_bedrooms         1923
population             3888
households             1815
median_income         12928
median_house_value     3842
dtype: int64
```

[22]:
```
##### Look at correlations in the numerical independent variables as well as
 ↪the dependent variables by executing df_num.corr()
df_num.corr()
```

[22]:
```
                    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
population           0.099773 -0.108785           -0.296244     0.857126
households           0.055310 -0.071035           -0.302916     0.918484
median_income       -0.015176 -0.079809           -0.119034     0.198050
median_house_value  -0.045967 -0.144160            0.105623     0.134153

                    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
population               0.877747    1.000000    0.907222       0.004834
households               0.979728    0.907222    1.000000       0.013033
median_income           -0.007723    0.004834    0.013033       1.000000
median_house_value       0.049686   -0.024650    0.065843       0.688075

                    median_house_value
longitude                    -0.045967
latitude                     -0.144160
housing_median_age            0.105623
total_rooms                   0.134153
total_bedrooms                0.049686
population                   -0.024650
households                    0.065843
median_income                 0.688075
median_house_value            1.000000
```

[23]:

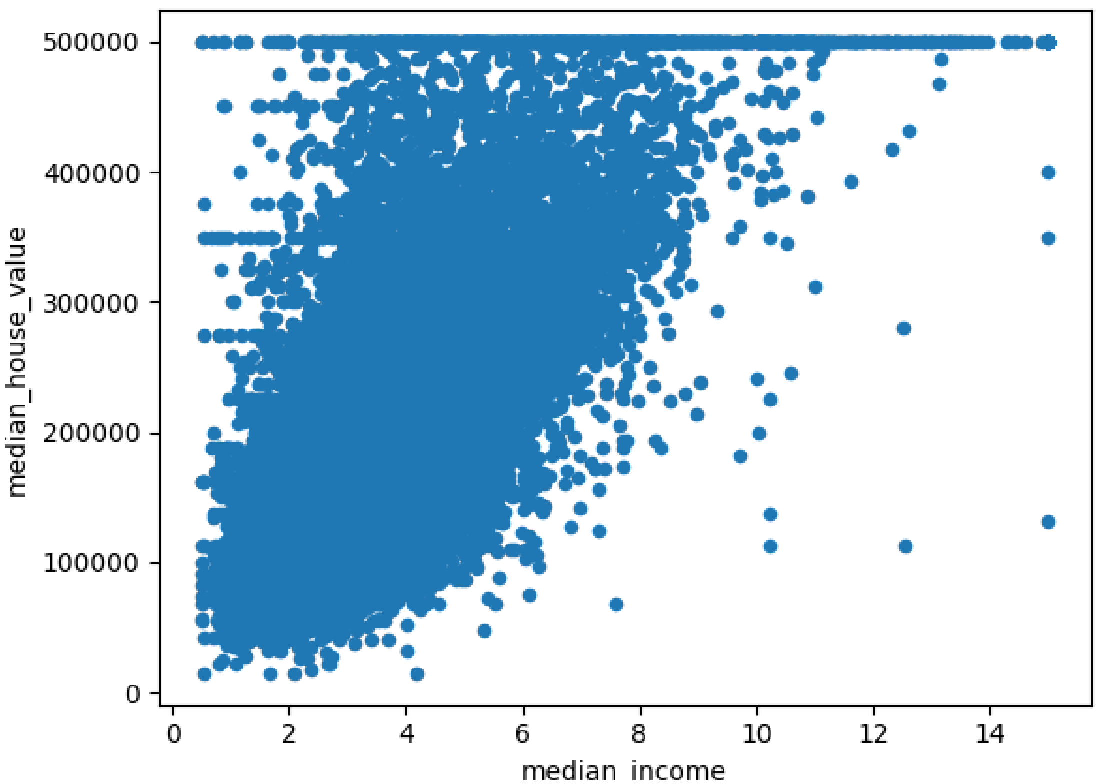
```python
#### 12. Group the data and sort in ascending order
df.groupby(by='median_house_value').count().sort_values('median_house_value',␣
 ↪ascending=False).head(10)
```

[23]:

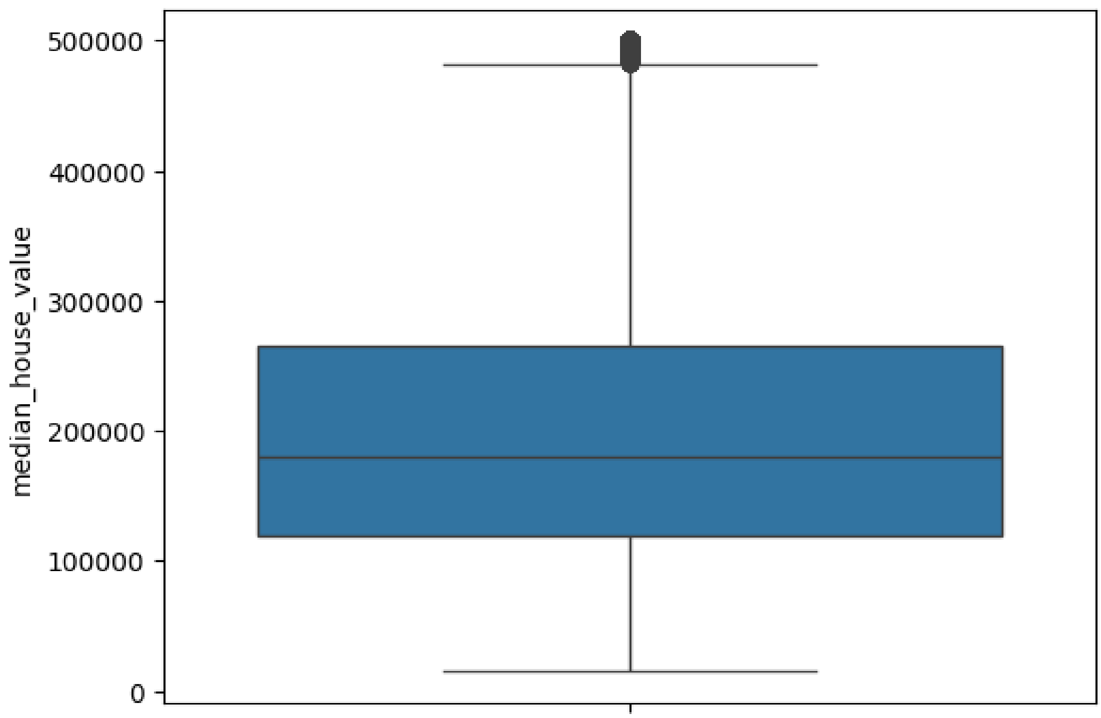
```
                    longitude  latitude  housing_median_age  total_rooms  \
median_house_value
500001.0                  965       965                 965          965
500000.0                   27        27                  27           27
499100.0                    1         1                   1            1
499000.0                    1         1                   1            1
498800.0                    1         1                   1            1
498700.0                    1         1                   1            1
498600.0                    1         1                   1            1
498400.0                    1         1                   1            1
497600.0                    1         1                   1            1
497400.0                    1         1                   1            1

                    total_bedrooms  population  households  median_income  \
median_house_value
500001.0                       958         965         965            965
500000.0                        27          27          27             27
499100.0                         1           1           1              1
499000.0                         1           1           1              1
498800.0                         1           1           1              1
498700.0                         1           1           1              1
498600.0                         1           1           1              1
498400.0                         1           1           1              1
497600.0                         1           1           1              1
```

```
497400.0                          1            1            1               1

                    ocean_proximity
median_house_value
500001.0                        965
500000.0                         27
499100.0                          1
499000.0                          1
498800.0                          1
498700.0                          1
498600.0                          1
498400.0                          1
497600.0                          1
497400.0                          1
```

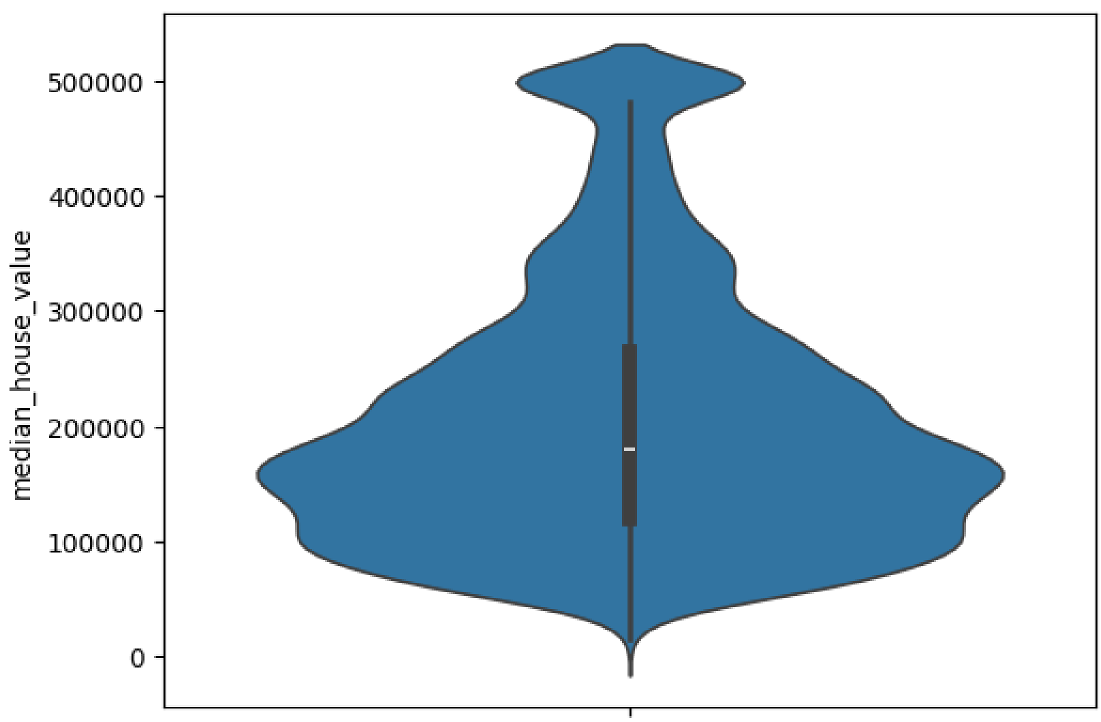
[25]:
```
#### 13. Plot the data using scatterplot
_ = df.plot.scatter('median_income', 'median_house_value')
```



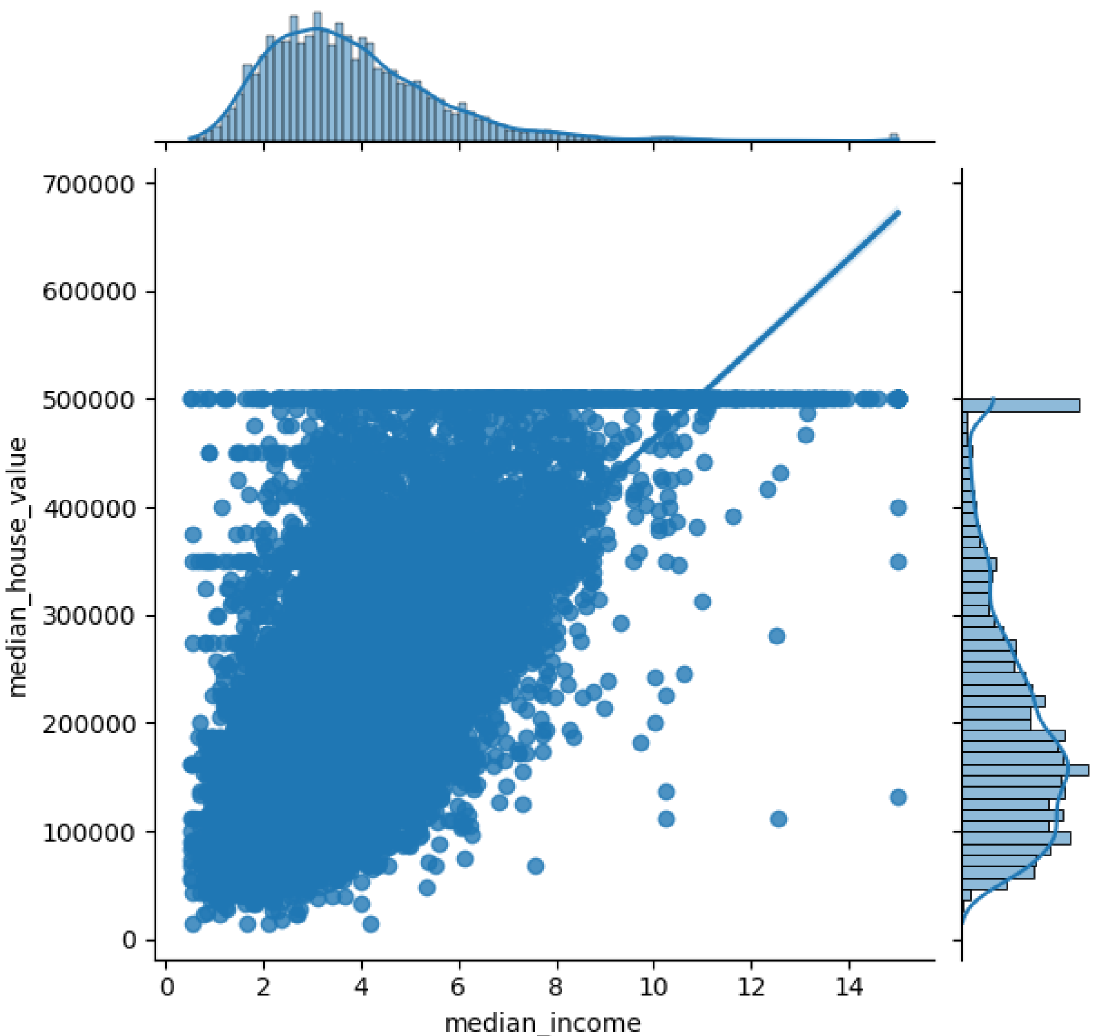
[26]:
```
_ = sns.boxplot(y='median_house_value', data=df)
```



[27]:
```
_ = sns.violinplot(y='median_house_value', data=df)
```

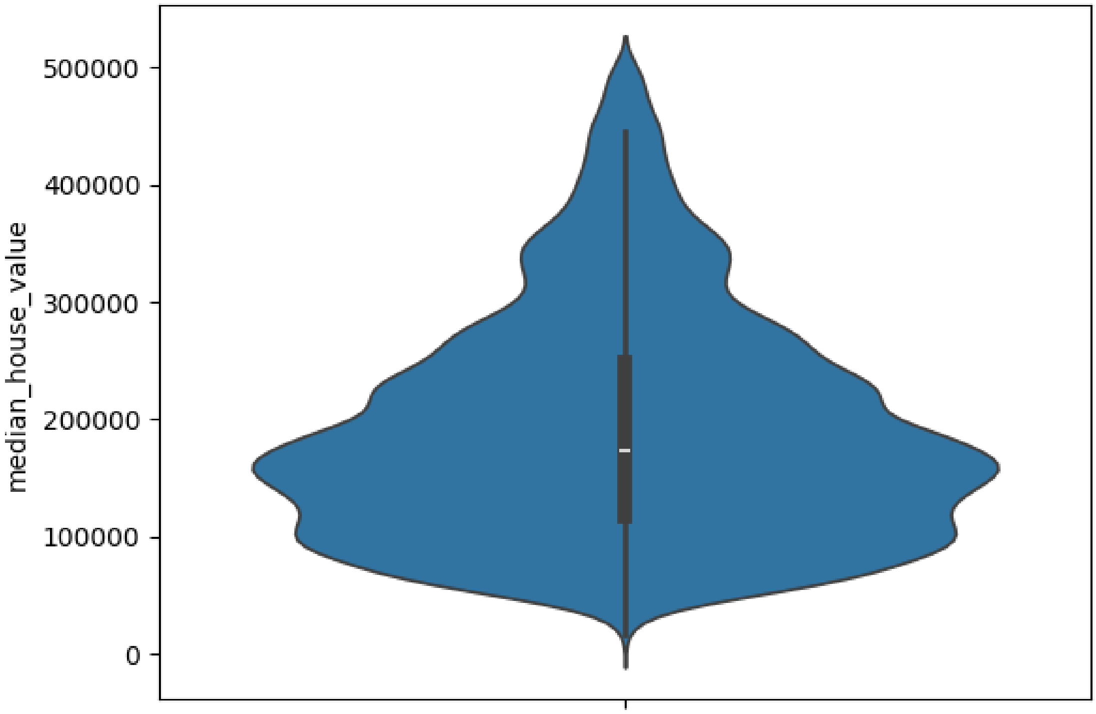


[28]:
```
_ = sns.jointplot(x="median_income", y="median_house_value", data=df,␣
↪kind="reg")
```

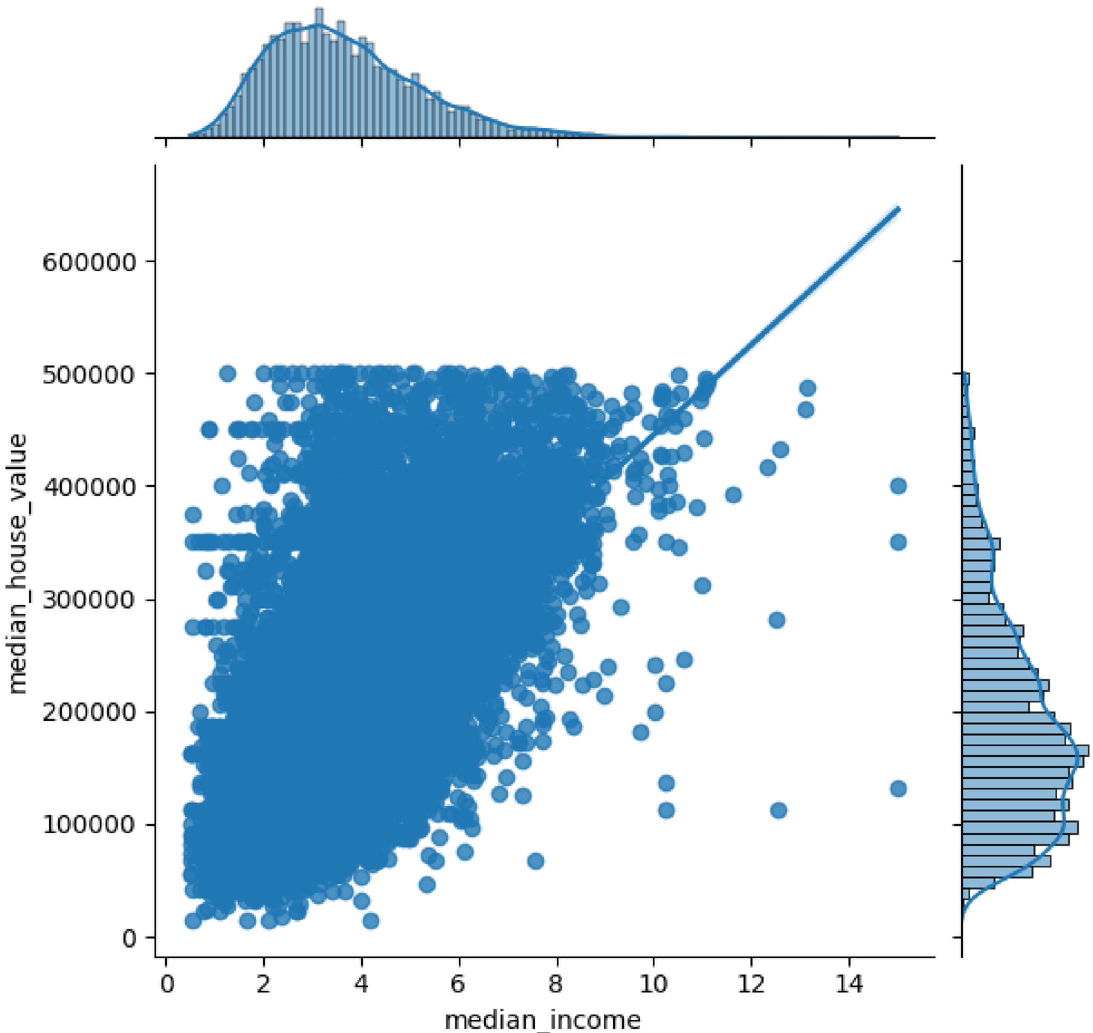


[30]:
```
#### 14. Removing outliers
df.drop(df[df['median_house_value']>500000].index, inplace=True)
```

[31]:
```
_ = sns.violinplot(y='median_house_value', data=df)
```



[32]:
```
_ = sns.jointplot(x="median_income", y="median_house_value", data=df,␣
↪kind="reg")
```



Preparing the data for training and testing

1. Divide our independent and dependent variable into two separate variables.
2. Split the data into training and testing datasets.

[42]:
```
# 1)

X = df.iloc[:,8].values.reshape(-1,1) # input
y = df.iloc[:,7].values # output (dependent variable)
```

[43]:
```
# 2) Splitting our data into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪shuffle=False, random_state=SEED)
```

Train the Model

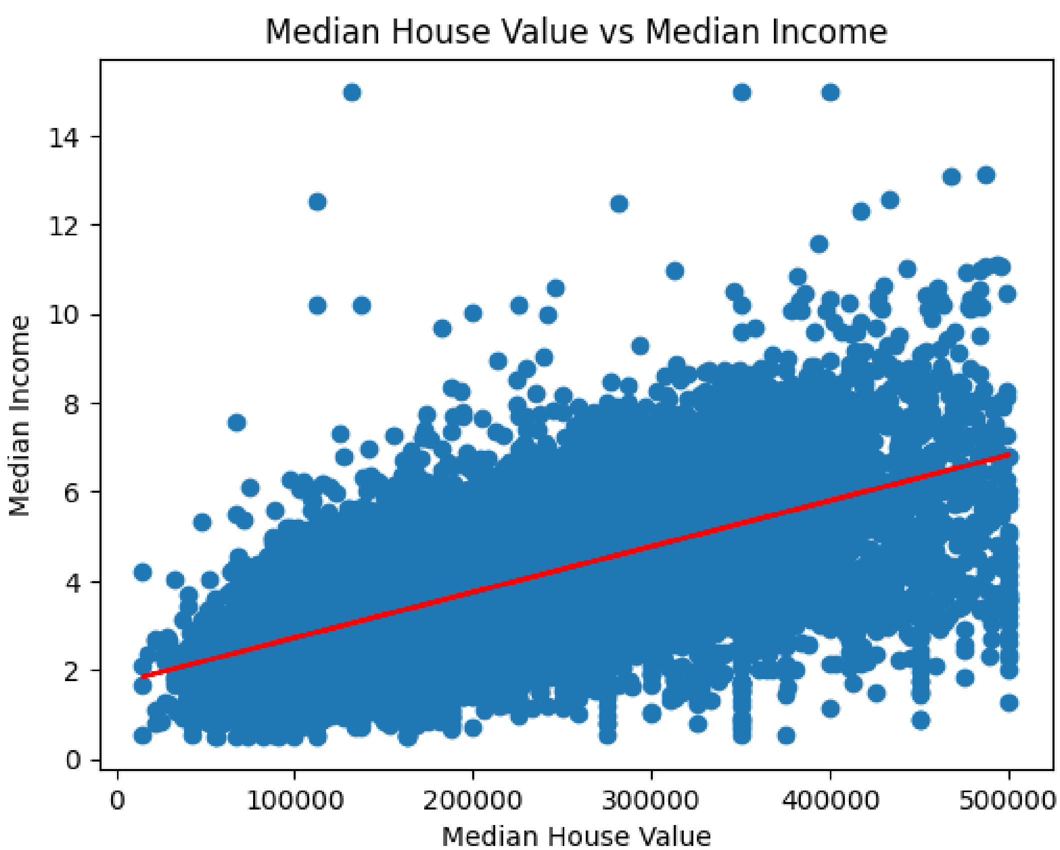
[44]:
```
# Import the linear regression algorithm
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

# Train the model
regressor.fit(X_train, y_train)
```

[44]:
```
LinearRegression()
```

15. Visualize the trained model

[45]:
```
# y=mx+c (Linear regression model)
line = regressor.coef_*X + regressor.intercept_
```

[46]:
```
# Lets plot this on the scatter plot
plt.scatter(X,y)
plt.plot(X, line, 'r')
plt.xlabel("Median House Value")
plt.ylabel("Median Income")
plt.title("Median House Value vs Median Income")
plt.show()
```



1.6 Rubrics:

Criteria	Scoring
Data Preprocessing	20 pts.
Training Performance	20 pts.
Model Evaluation Metrics	20 pts.
Visualization	20 pts.
Explanation & Analysis	20 pts.

[ ]:

```

```