# HOUSING LINEAR REGRESSION PROJECT

In [1]:
```
#importing the libraries
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#load dataset
data = 'https://raw.githubusercontent.com/alexeygrigorev/datasets/master/housing.csv'
```

In [3]:
```
#read the dataset
df = pd.read_csv(data)
```

In [4]:
```
print(df.shape)
```

```
(20640, 10)
```

In [5]:
```
df.head()
```

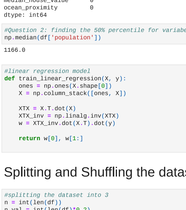
Out[5]:

| | 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 |

# EDA

In [6]:
```
sns.histplot(df.median_house_value, bins=50)
#the median_house_value doesnt have a long tail
```

Out[6]:
```
<AxesSubplot:xlabel='median_house_value', ylabel='Count'>
```




In [7]:
```
#question 1: finding a feature with missing values
df.isnull().sum()
```

Out[7]:
```
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
```

In [8]:
```
#Question 2: finding the 50% percentile for variabe 'population'
np.median(df['population'])
```

Out[8]:
```
1166.0
```

In [9]:
```
#linear regression model
def train_linear_regression(X, y):
    ones = np.ones(X.shape[0])
    X = np.column_stack([ones, X])

    XTX = X.T.dot(X)
    XTX_inv = np.linalg.inv(XTX)
    w = XTX_inv.dot(X.T).dot(y)

    return w[0], w[1:]
```

# Splitting and Shuffling the dataset

In [10]:
```
#splitting the dataset into 3
n = int(len(df))
n_val = int(len(df)*0.2)
n_test = int(len(df)*0.2)
n_train = n - n_val - n_test
n_val, n_test, n_train
```

Out[10]:
```
(4128, 4128, 12384)
```

In [11]:
```
#shuffling the dataset
idx = np.arange(n)
shuffle = np.random.seed(42)
np.random.shuffle(idx)

df_train = df.iloc[idx[:n_train]].copy()
df_val = df.iloc[n_train:n_train + n_val].copy()
df_test = df.iloc[n_train + n_val:].copy()
df_train.head()
```

Out[11]:

| | longitude | latitude | housing_median_age | total_rooms | total_bedrooms | population | households | median_income | median_house_value | ocean_proximity |
|---|---|---|---|---|---|---|---|---|---|---|
| **20046** | -119.01 | 36.06 | 25.0 | 1505.0 | NaN | 1392.0 | 359.0 | 1.6812 | 47700.0 | INLAND |
| **3024** | -119.46 | 35.14 | 30.0 | 2943.0 | NaN | 1565.0 | 584.0 | 2.5313 | 45800.0 | INLAND |
| **15663** | -122.44 | 37.80 | 52.0 | 3830.0 | NaN | 1310.0 | 963.0 | 3.4801 | 500001.0 | NEAR BAY |
| **20484** | -118.72 | 34.28 | 17.0 | 3051.0 | NaN | 1705.0 | 495.0 | 5.7376 | 218600.0 | <1H OCEAN |
| **9814** | -121.93 | 36.62 | 34.0 | 2351.0 | NaN | 1063.0 | 428.0 | 3.7250 | 278000.0 | NEAR OCEAN |

In [12]:
```
df_train = df_train.reset_index(drop = True)
df_val = df_val.reset_index(drop = True)
df_test = df_test.reset_index(drop = True)
```

In [13]:
```
y_train = np.log1p(df_train.median_house_value.values)
y_val = np.log1p(df_val.median_house_value.values)
y_test = np.log1p(df_test.median_house_value.values)
```

In [14]:
```
#deleting the median_house_value from our dataset
del df_train['median_house_value']
del df_val['median_house_value']
del df_test['median_house_value']
```

In [15]:
```
#filling the missing data
x_train = df_train['total_bedrooms'].fillna(0).values
```

In [16]:
```
df_train
```

Out[16]:

| | longitude | latitude | housing_median_age | total_rooms | total_bedrooms | population | households | median_income | ocean_proximity |
|---|---|---|---|---|---|---|---|---|---|
| **0** | -119.01 | 36.06 | 25.0 | 1505.0 | NaN | 1392.0 | 359.0 | 1.6812 | INLAND |
| **1** | -119.46 | 35.14 | 30.0 | 2943.0 | NaN | 1565.0 | 584.0 | 2.5313 | INLAND |
| **2** | -122.44 | 37.80 | 52.0 | 3830.0 | NaN | 1310.0 | 963.0 | 3.4801 | NEAR BAY |
| **3** | -118.72 | 34.28 | 17.0 | 3051.0 | NaN | 1705.0 | 495.0 | 5.7376 | <1H OCEAN |
| **4** | -121.93 | 36.62 | 34.0 | 2351.0 | NaN | 1063.0 | 428.0 | 3.7250 | NEAR OCEAN |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **12379** | -117.81 | 33.88 | 19.0 | 2265.0 | 283.0 | 904.0 | 279.0 | 9.2327 | <1H OCEAN |
| **12380** | -120.68 | 35.48 | 15.0 | 2608.0 | 525.0 | 1351.0 | 502.0 | 2.7798 | <1H OCEAN |
| **12381** | -120.91 | 38.98 | 13.0 | 7689.0 | 1415.0 | 3264.0 | 1198.0 | 3.6530 | INLAND |
| **12382** | -117.72 | 34.09 | 36.0 | 1473.0 | 326.0 | 785.0 | 299.0 | 3.2566 | INLAND |
| **12383** | -122.47 | 37.76 | 34.0 | 2807.0 | 487.0 | 1152.0 | 445.0 | 5.1893 | NEAR BAY |

12384 rows × 9 columns

In [18]:
```
#validating the model
check = ['latitude', 'longitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income']

def prepare_X(df, fillna_value):
    df_num = df[check]
    df_num = df_num.fillna(fillna_value)
    X = df_num.values

    return X
```

In [19]:
```
#evaluating linear regression mofel using 'rmse' root means squared error
def rmse(y, y_pred):
    error = y_pred - y
    mse = (error ** 2).mean()
    return np.sqrt(mse)
```

In [20]:
```
#calculating the mean
mean = df_train.housing_median_age.mean()

X_mean_train = prepare_X(df_train, fillna_value=mean)
w_0_mean, w_mean = train_linear_regression(X_mean_train, y_train)
```

In [21]:
```
X_mean_val = prepare_X(df_val, fillna_value=mean)
y_mean_pred_val = w_0_mean + X_mean_val.dot(w_mean)
```

In [22]:
```
np.round(rmse(y_val, y_mean_pred_val),2)
```

Out[22]:
```
0.33
```

In [23]:
```
X_null_train = prepare_X(df_train, fillna_value=0)
w_0_null, w_null = train_linear_regression(X_null_train, y_train)
```

In [24]:
```
X_null_val = prepare_X(df_val, fillna_value=0)
y_null_pred_val = w_0_null + X_null_val.dot(w_null)
```

In [25]:
```
np.round(rmse(y_val, y_null_pred_val),2)
```

Out[25]:
```
0.33
```

In [26]:
```
#answer to number 3: both options are good
```

In [27]:
```
#regularization model
def train_linear_regression_reg(X, y, r=0.1):
    ones = np.ones(X.shape[0])
    X = np.column_stack([ones, X])

    XTX = X.T.dot(X)
    reg = r*np.eye(XTX.shape[0])
    XTX = XTX + reg

    XTX_inv = np.linalg.inv(XTX)
    w = XTX_inv.dot(X.T).dot(y)

    return w[0], w[1:]
```

In [28]:
```
for r in [0, 0.000001, 0.0001, 0.001, 0.01, 0.1, 1, 5, 10]:
    w_0, w = train_linear_regression_reg(X_null_train, y_train, rrr)
    y_null_reg_val = w_0 + X_null_val.dot(w)
    rmse_val = np.round(rmse(y_val, y_null_reg_val),2)
    print(r, w_0, rmse_val)
```

```
0 -11.686975241713924 0.33
1e-06 -11.686959175539379 0.33
0.0001 -11.685368865728295 0.33
0.001 -11.670931317955695 0.33
0.01 -11.528493585709422 0.33
0.1 -10.274500282066442 0.34
1 -4.920488097806387 0.35
5 -1.4829957458166324 0.36
10 -0.7899311832497758 0.36
```

In [29]:
```
#answer to Q4: 0
```

In [39]:
```
#trying random seed values
rmse_list = []

for r in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:
    idx = np.arange(n)
    shuffle = np.random.seed(r)
    np.random.shuffle(idx)

    df_train = df.iloc[idx[:n_train]].copy()
    df_val = df.iloc[n_train:n_train + n_val].copy()
    df_test = df.iloc[n_train + n_val:].copy()

    df_train = df_train.reset_index(drop = True)
    df_val = df_val.reset_index(drop = True)
    df_test = df_test.reset_index(drop = True)

    y_train = np.log1p(df_train.median_house_value.values)
    y_val = np.log1p(df_val.median_house_value.values)
    y_test = np.log1p(df_test.median_house_value.values)

    del df_train['median_house_value']
    del df_val['median_house_value']
    del df_test['median_house_value']

    X_null_train = prepare_X(df_train, fillna_value=0)
    w_0, w = train_linear_regression(X_null_train, y_train)

    X_null_val = prepare_X(df_val, fillna_value=0)
    y_null_reg_val = w_0 + X_null_val.dot(w)
    rmse_val = np.round(rmse(y_val, y_null_reg_val),2)


    rmse_list.append(rmse_val)

    print(r, w_0, rmse_val)
```

```
0 -11.90038213944327 0.33
1 -11.732757373641363 0.33
2 -11.806729361482816 0.33
3 -11.587900348548297 0.33
4 -11.389470588931339 0.33
5 -11.447114276367927 0.33
6 -11.370516352206277 0.33
7 -12.473448918967847 0.33
8 -11.880287439019588 0.33
9 -11.459046830955497 0.33
```

In [40]:
```
rmse_list
```

Out[40]:
```
[0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33, 0.33]
```

In [41]:
```
np.round(np.std(rmse_list),3)
```

Out[41]:
```
0.0
```

In [42]:
```
# use seed 9,Combine train and validation datasets

r = 9
idx = np.arange(n)
shuffle = np.random.seed(r)
np.random.shuffle(idx)

df_train = df.iloc[idx[:n_train]].copy()
df_val = df.iloc[n_train:n_train + n_val].copy()
df_test = df.iloc[n_train + n_val:].copy()

combine = [df_train, df_val]
df_train_val = pd.concat(combine)

df_train_val = df_train_val.reset_index(drop = True)
df_test = df_test.reset_index(drop = True)

y_train_val = np.log1p(df_train_val.median_house_value.values)
y_test = np.log1p(df_test.median_house_value.values)

del df_train_val['median_house_value']
del df_test['median_house_value']
```

In [45]:
```
X_null_train_val = prepare_X(df_train_val, fillna_value=0)
w_0_train_val, w_train_val = train_linear_regression_reg(X_null_train_val, y_train_val, rr=0.001)

X_null_test = prepare_X(df_test, fillna_value=0)
y_null_pred_test = w_0_train_val + X_null_test.dot(w_train_val)

np.round(rmse(y_test, y_null_pred_test),2)
```

Out[45]:
```
0.32
```

In [46]:
```
#answer = 0_32
```

In [ ]:
```

```