Data Cleaning

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('housing.csv')
df
```

Out[ ]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	po
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	
...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	
20636	-121.21	39.49	18.0	697.0	150.0	
20637	-121.22	39.43	17.0	2254.0	485.0	
20638	-121.32	39.43	18.0	1860.0	409.0	
20639	-121.24	39.37	16.0	2785.0	616.0	

20640 rows × 10 columns

In [ ]:
```
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
```

In [ ]:
```
print("Sum of null values of the column Total_bedrooms : ", df['total_bed
x_mean = df['total_bedrooms'].mean()
df['total_bedrooms'] = df['total_bedrooms'].fillna(x_mean)
print("Sum of null values of the column Total_bedrooms : ", df['total_bed
```

```
Sum of null values of the column Total_bedrooms :  207
Sum of null values of the column Total_bedrooms :  0
```

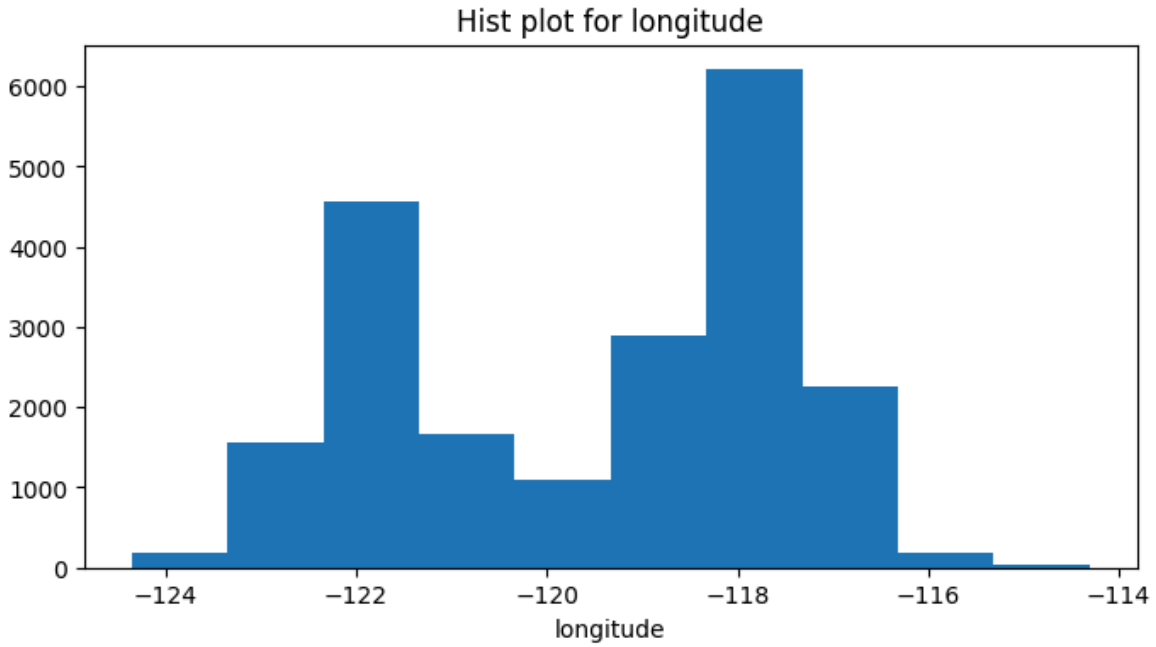
In [ ]:
```
df['ocean_proximity'].unique()
```

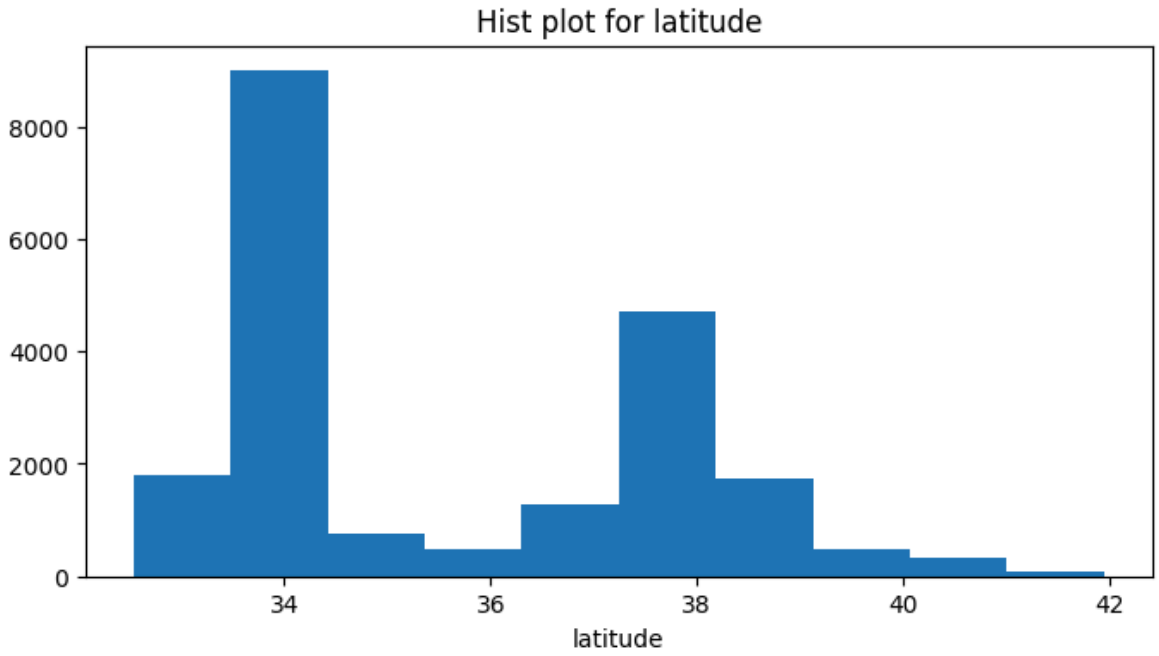
Out[ ]:
```
array(['NEAR BAY', '<1H OCEAN', 'INLAND', 'NEAR OCEAN', 'ISLAND'],
      dtype=object)
```

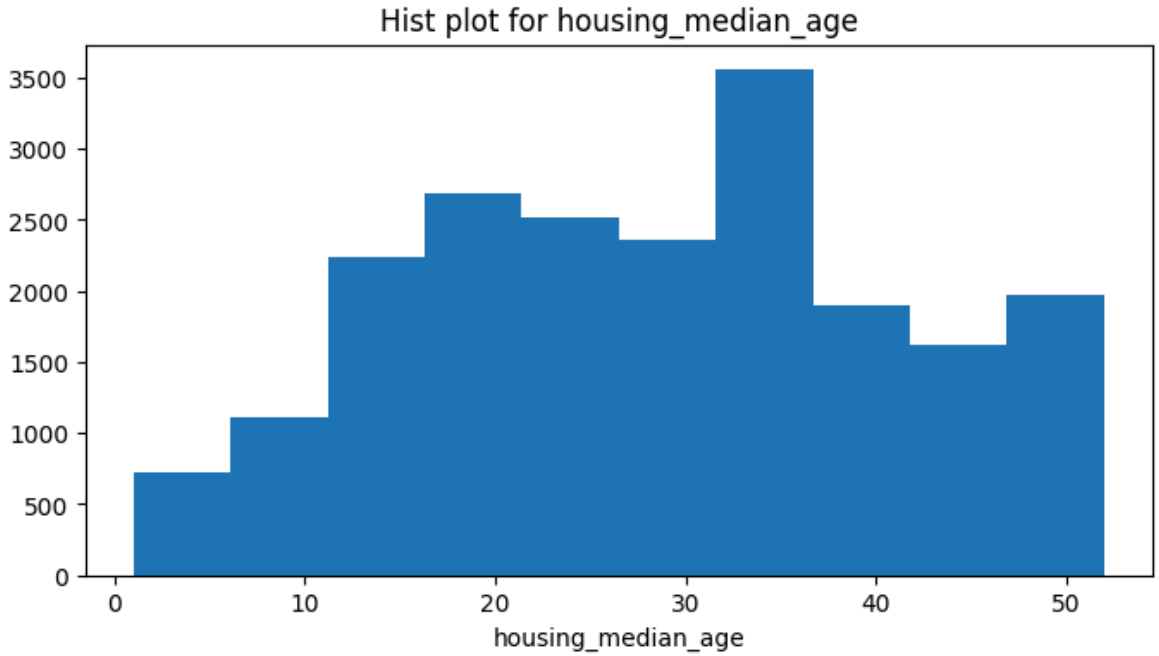
In [ ]:
```
ocean_prox_map = {
    "NEAR BAY": 0,
    "<1H OCEAN": 1,
    "INLAND": 2,
    "NEAR OCEAN": 3,
    "ISLAND": 4
}

df["ocean_proximity"] = df["ocean_proximity"].map(ocean_prox_map)
```

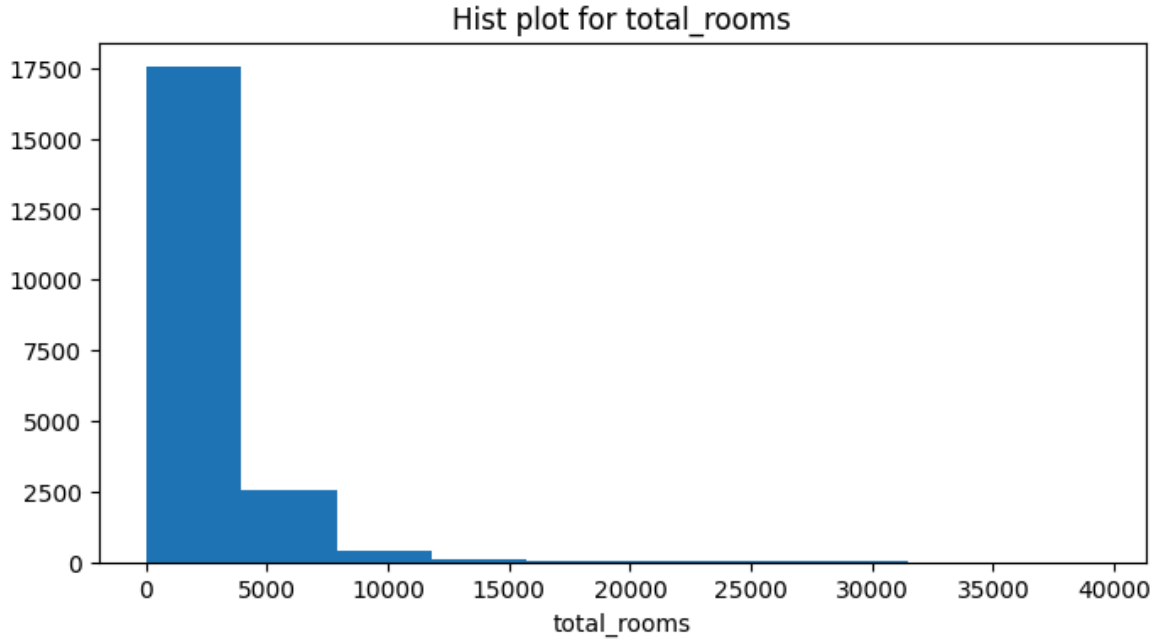
Feature Selection

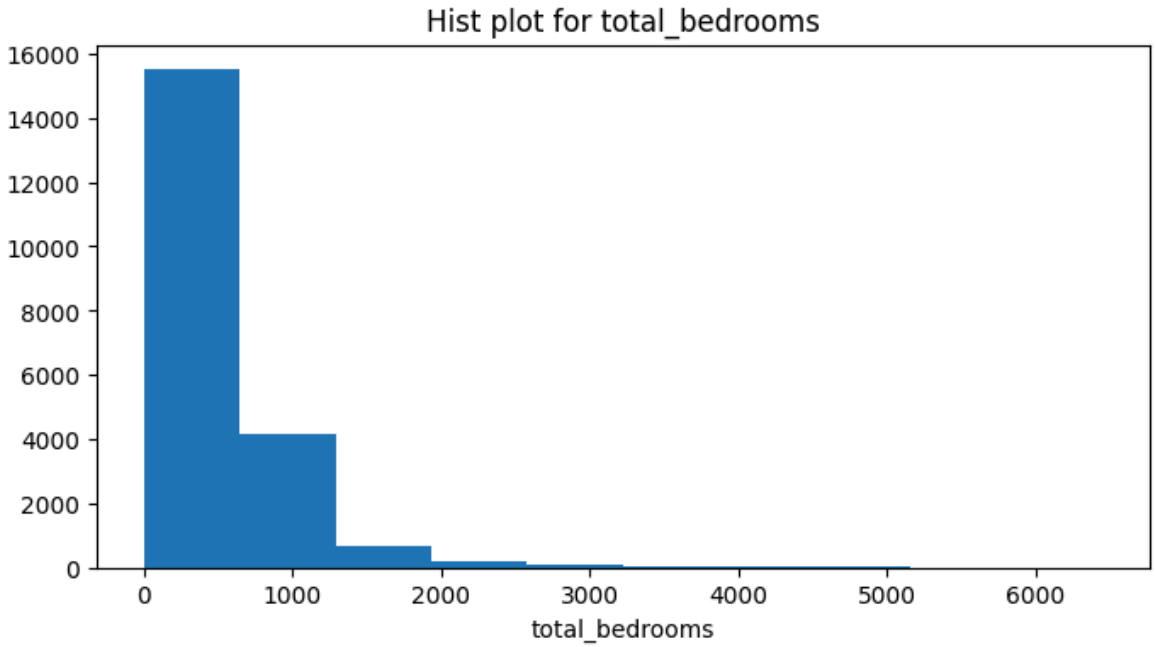
In [ ]:
```
features = ['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity']

for i in features:
    plt.figure(figsize = (8, 4))
    plt.hist(df[i], linewidth=0.1)
    plt.title(f'Hist plot for {i}')
    plt.xlabel(i)
    plt.show()
```

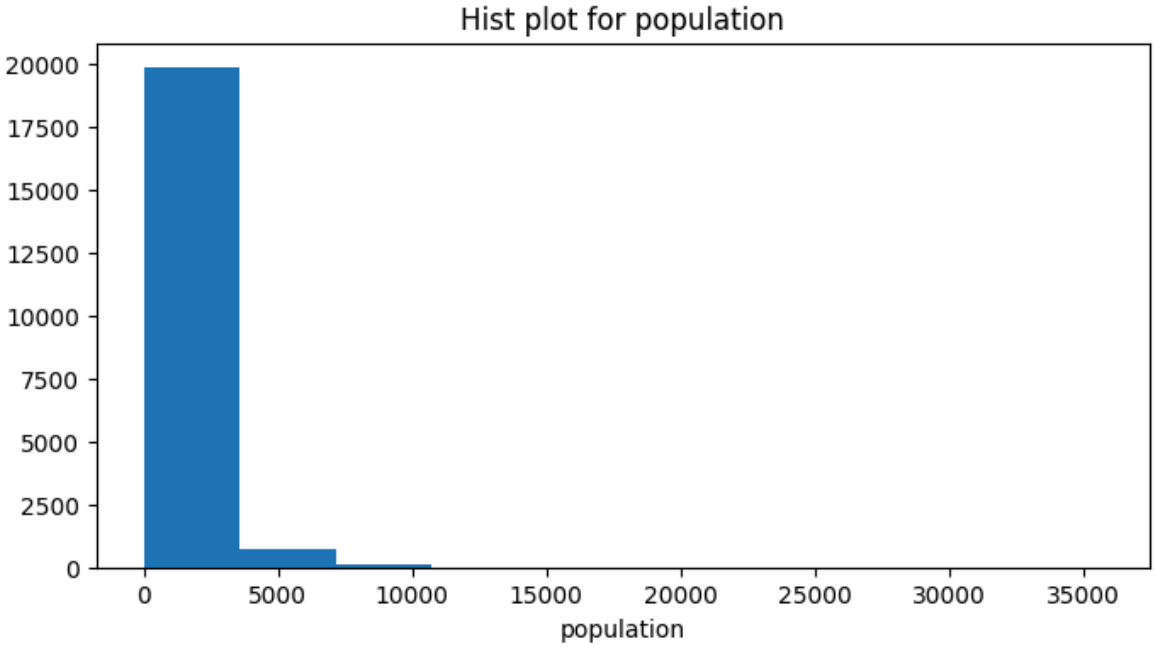


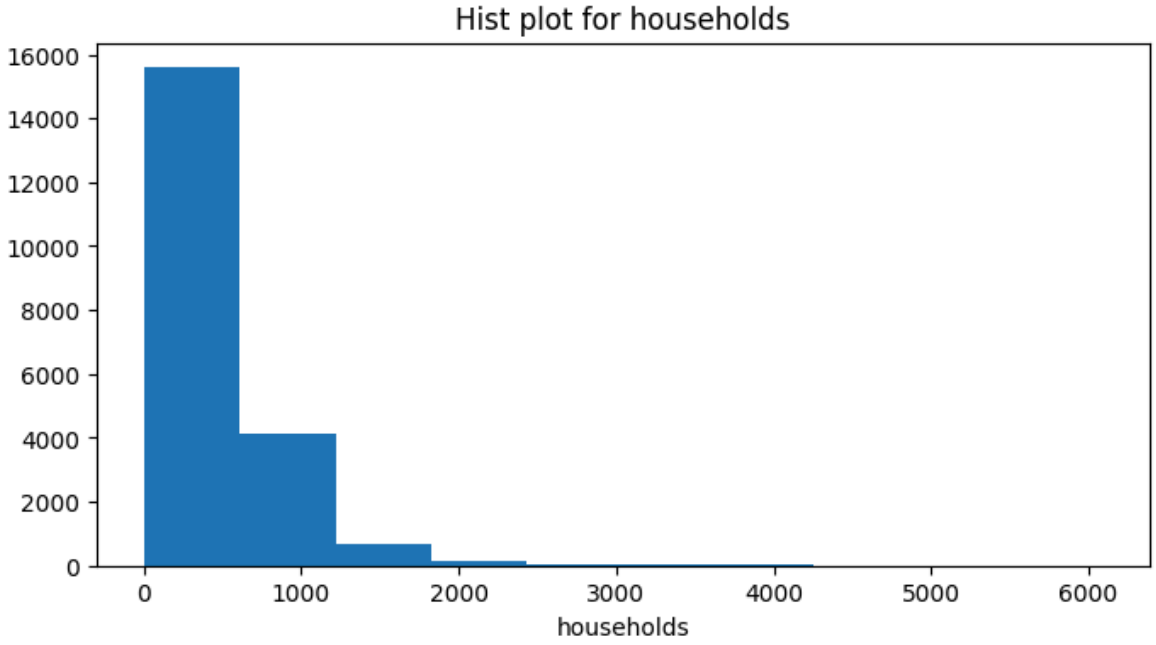

Hist plot for latitude
8000
6000
4000
2000
0
34
36
38
40
42
latitude

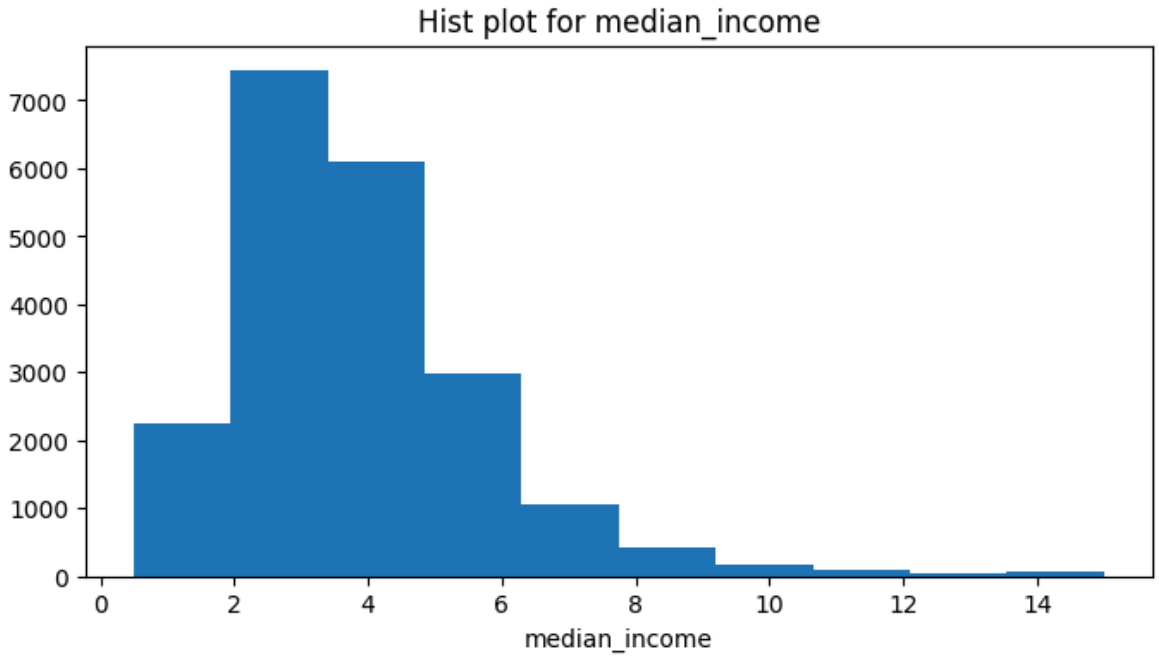

Hist plot for housing_median_age
3500
3000
2500
2000
1500
1000
500
0
0
10
20
30
40
50
housing_median_age

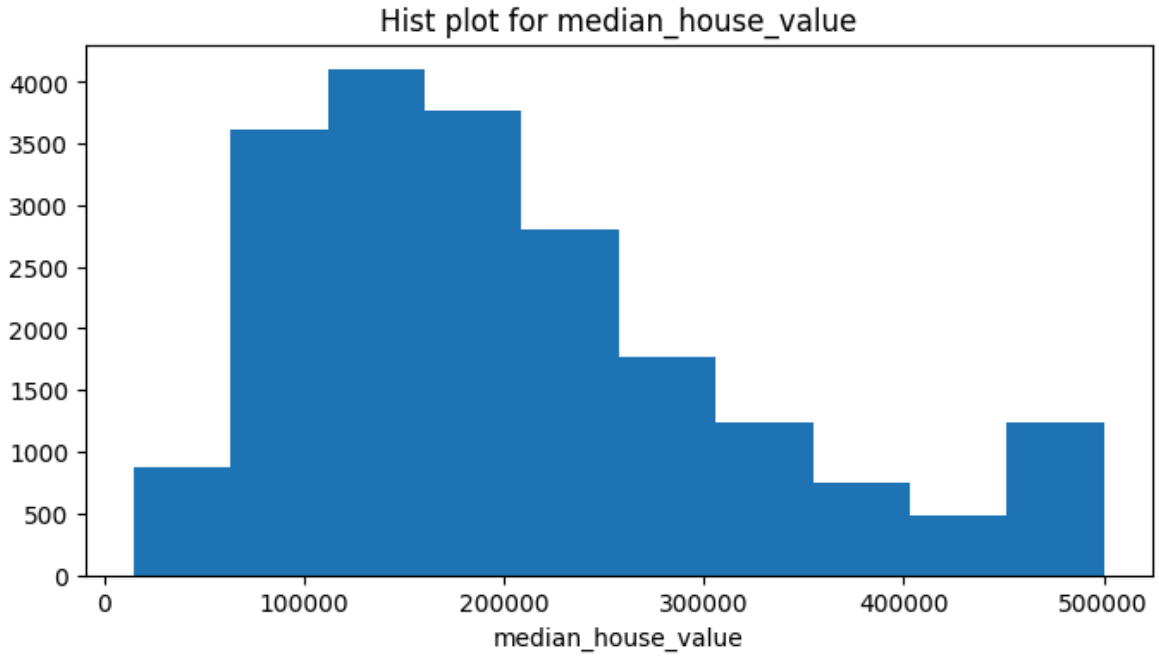

Hist plot for total_rooms
17500
15000
12500
10000
7500
5000
2500
0
0
5000
10000
15000
20000
25000
30000
35000
40000
total_rooms

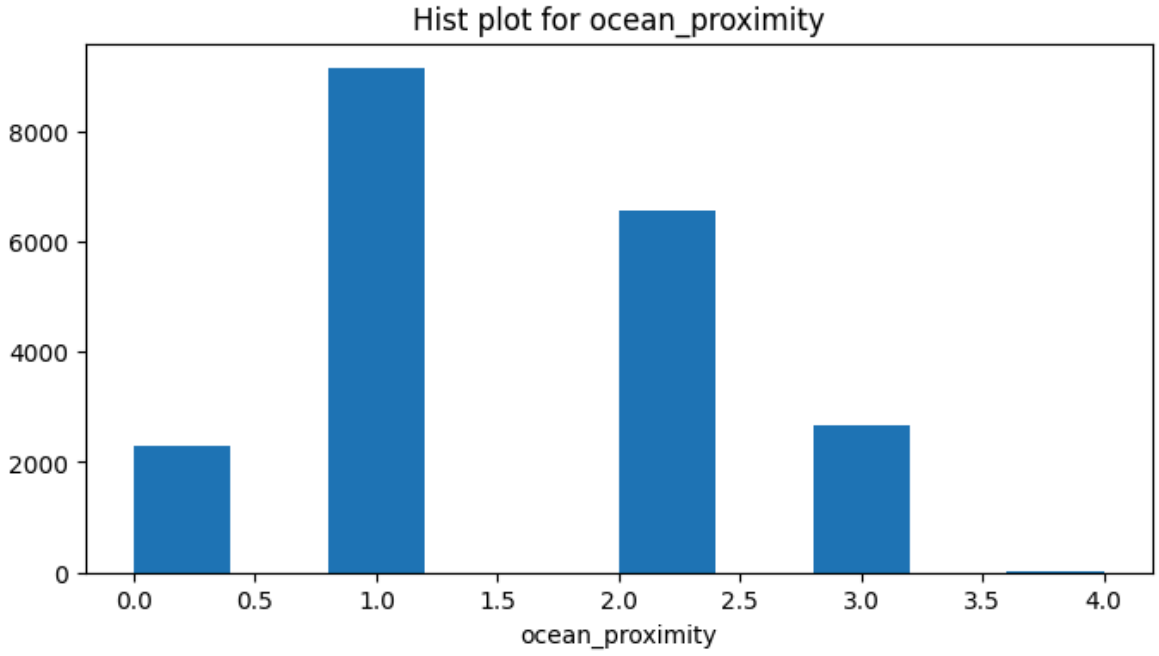

Hist plot for total_bedrooms
16000
14000
12000
10000
8000
6000
4000
2000
0
0
1000
2000
3000
4000
5000
6000
total_bedrooms


Hist plot for population
20000
17500
15000
12500
10000
7500
5000
2500
0
0
5000
10000
15000
20000
25000
30000
35000
population


Hist plot for households
16000
14000
12000
10000
8000
6000
4000
2000
0
0
1000
2000
3000
4000
5000
6000
households


Hist plot for median_income
7000
6000
5000
4000
3000
2000
1000
0
0
2
4
6
8
10
12
14
median_income


Hist plot for median_house_value
4000
3500
3000
2500
2000
1500
1000
500
0
0
100000
200000
300000
400000
500000
median_house_value


Hist plot for ocean_proximity
8000
6000
4000
2000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
ocean_proximity

In [ ]:
```
df.corr()
```

Out[ ]:

	longitude	latitude	housing_median_age	total_rooms	tot
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	0.069260	-0.066658	-0.318998	0.927253	
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	
ocean_proximity	0.180381	-0.067586	-0.204882	0.014818	

The columns latitude and longitude, total_rooms and total_bedrooms, total_rooms and population, total_rooms and household, total_bedrooms and population, total_bedrooms and households, population and households, are highly correlated. So we can neglect some of the columns as they have the same feature infos.

In [ ]:
```
drop_columns = ['total_bedrooms', 'population', 'households', 'latitude']
df = df.drop(drop_columns, axis=1)
```

Model Selection

In [ ]:
```
#Standardization...
from sklearn.preprocessing import StandardScaler

scalar = StandardScaler()
std_df = scalar.fit_transform(df.drop('ocean_proximity', axis=1))
df_std = pd.DataFrame(std_df, columns=['longitude', 'housing_median_age',
       'median_house_value'])
df
```

Out[ ]:

	longitude	housing_median_age	total_rooms	median_income	median_hou
0	-122.23	41.0	880.0	8.3252	
1	-122.22	21.0	7099.0	8.3014	
2	-122.24	52.0	1467.0	7.2574	
3	-122.25	52.0	1274.0	5.6431	
4	-122.25	52.0	1627.0	3.8462	
...	...	...	...	...	...
20635	-121.09	25.0	1665.0	1.5603	
20636	-121.21	18.0	697.0	2.5568	
20637	-121.22	17.0	2254.0	1.7000	
20638	-121.32	18.0	1860.0	1.8672	
20639	-121.24	16.0	2785.0	2.3886	

20640 rows × 6 columns

Training and Testing data

In [ ]:
```
from sklearn.model_selection import train_test_split

X = df.drop('ocean_proximity', axis=1)
y = df['ocean_proximity']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2
```

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X_train, y_train)
```

Out[ ]:
```
DecisionTreeClassifier(criterion='entropy')
```

In [ ]:
```
from sklearn.metrics import accuracy_score

y_pred = clf.predict(X_test)
accuracy_scr = accuracy_score(y_pred, y_test)
print("Accuracy : ", accuracy_scr)
```

```
Accuracy :  0.8401162790697675
```

Taking total_rooms column as feature

In [ ]:
```
conditions = [
    (df['total_rooms'] <= 2500),
    (df['total_rooms'] > 2500) & (df['total_rooms'] <= 5000),
    (df['total_rooms'] > 5000) & (df['total_rooms'] <= 7500),
    (df['total_rooms'] > 7500) & (df['total_rooms'] <= 10000),
    (df['total_rooms'] > 10000)
]
categories = [0, 1, 2, 3, 4]

# Use np.select to assign numerical categories based on conditions
df['new_total_rooms'] = np.select(conditions, categories, default=0)

df
```

Out[ ]:

	longitude	housing_median_age	total_rooms	median_income	median_hou
0	-122.23	41.0	880.0	8.3252	
1	-122.22	21.0	7099.0	8.3014	
2	-122.24	52.0	1467.0	7.2574	
3	-122.25	52.0	1274.0	5.6431	
4	-122.25	52.0	1627.0	3.8462	
...	...	...	...	...	
20635	-121.09	25.0	1665.0	1.5603	
20636	-121.21	18.0	697.0	2.5568	
20637	-121.22	17.0	2254.0	1.7000	
20638	-121.32	18.0	1860.0	1.8672	
20639	-121.24	16.0	2785.0	2.3886	

20640 rows × 7 columns

In [ ]:
```
cols = ['longitude', 'housing_median_age', 'median_income', 'median_house
y = df['new_total_rooms']

X = df[cols]
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2
clf.fit(X_train, y_train)
```

Out[ ]:
```
DecisionTreeClassifier(criterion='entropy')
```

In [ ]:
```
from sklearn.metrics import accuracy_score
y_pred = clf.predict(X_test)
print("The accuracy for this model : ", accuracy_score(y_test, y_pred))
```

```
The accuracy for this model :  0.5450581395348837
```

Inferences :

1. Ocean proximity as dependent variable.

As the dependent variable was alaready a catagorical value and some of the columns was dropped because pf their high correlation. Also the value was standerdized inorder to get all the values into a smaller ranged values. And finally we got an accuracy of 0.8

2. Total Rooms as the dependent variable.

As the total_rooms columns was not ment for the decision tree because of the continues values, we converted to classes of 5. Then without standerdizing normalizing the dataset, we performed the classifier on the training data. And we got anm accuracy of 0.5 which indicates that the column is not so good for classification.