Pradhumna Singh 201B182

1. Load Housing dataset

```
import pandas as pd
df = pd.read_csv('/content/housing.csv')
```

2.Display Brief information about Dataset.

```
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
```

Display number of Rows and columns

```
# 3.rows
df.shape[0]
#columns
df.shape[1]
```

```
10
```

```
# 4.Find Target Variable
target = 'median_house_value'
df[target]
```

```
0       452600.0
```

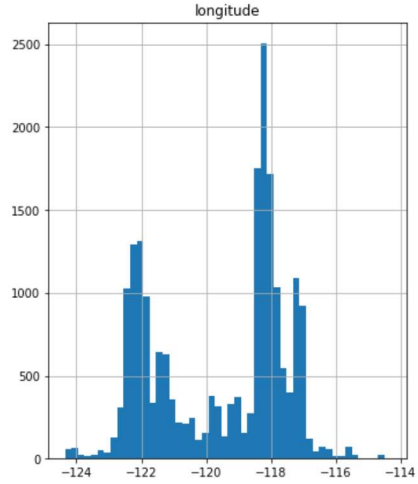
```
1        358500.0
2        352100.0
3        341300.0
4        342200.0
           ...
20635     78100.0
20636     77100.0
20637     92300.0
20638     84700.0
20639     89400.0
Name: median_house_value, Length: 20640, dtype: float64
```

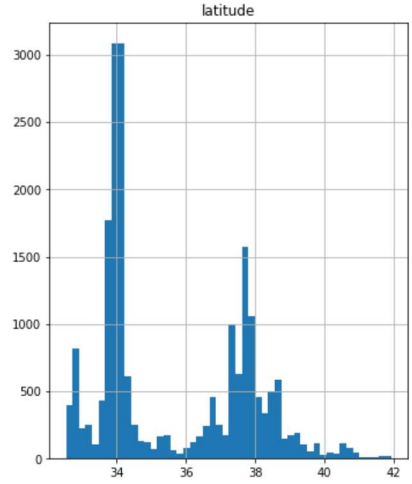
```
# 5.Show first few rws of dataset
f=df.head(6)
```

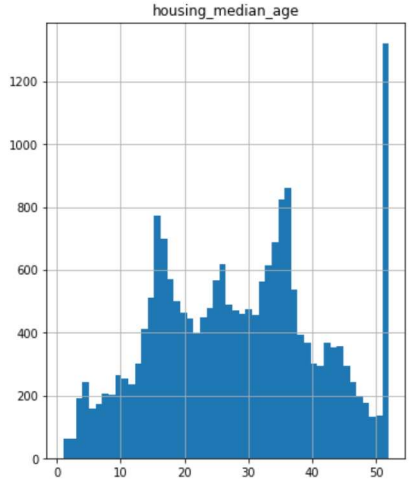
```
# 6.Display the Summary Statistic about all the features of Datas
df.describe()
```

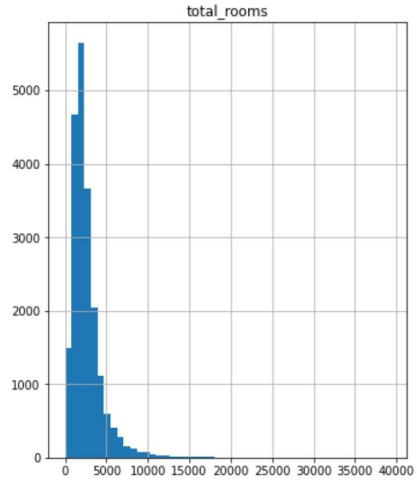
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	pc
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	206
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	14
std	2.003532	2.135952	12.585558	2181.615252	421.385070	11
min	-124.350000	32.540000	1.000000	2.000000	1.000000	
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	7
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	11
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	17
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	356

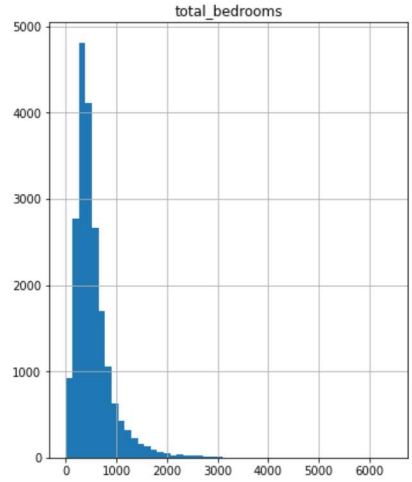
```
# 7.Show histogram of each attribute
import matplotlib.pyplot as plt
df.hist(bins =50, figsize=(20,25))
plt.show()
```

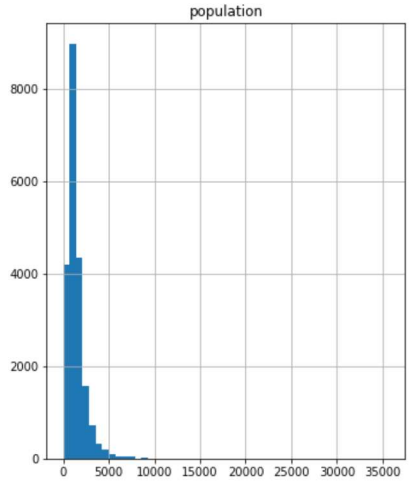

longitude
2500
2000
1500
1000
500
0
−124
−122
−120
−118
−116
−114

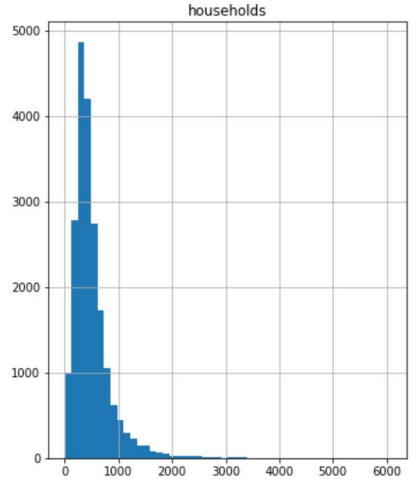

latitude
3000
2500
2000
1500
1000
500
0
34
36
38
40
42

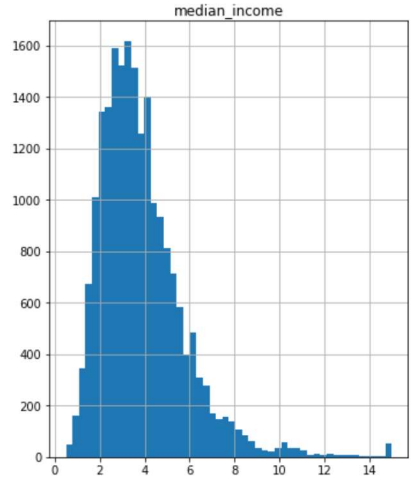

housing_median_age
1200
1000
800
600
400
200
0
0
10
20
30
40
50

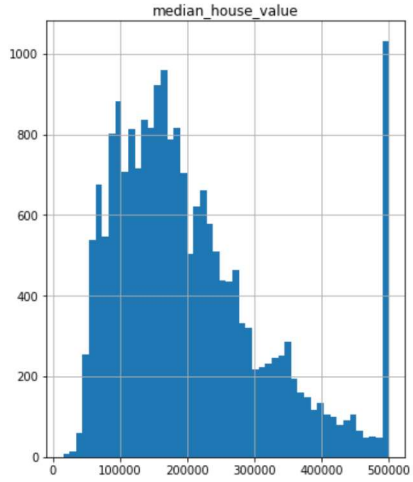

total_rooms
5000
4000
3000
2000
1000
0
0
5000
10000
15000
20000
25000
30000
35000
40000


total_bedrooms
5000
4000
3000
2000
1000
0
0
1000
2000
3000
4000
5000
6000


population
8000
6000
4000
2000
0
0
5000
10000
15000
20000
25000
30000
35000


households
5000
4000
3000
2000
1000
0
0
1000
2000
3000
4000
5000
6000


median_income
1600
1400
1200
1000
800
600
400
200
0
0
2
4
6
8
10
12
14


median_house_value
1000
800
600
400
200
0
0
100000
200000
300000
400000
500000

```
#8.Show Null values in dataset
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
```

```
#9. Show different types of values in categorical attributes alon
df['ocean_proximity'].value_counts()
```

```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

```
# 10. Fill the missing values with most frequently used value for
# numerical attribute fill median value.
df_num =pd.read_csv(r'/content/housing.csv')
median = df_num["total_bedrooms"].median()
df_num["total_bedrooms"].fillna(median, inplace=True)
mode = df_num["ocean_proximity"].mode()
df_num["ocean_proximity"].fillna(mode, inplace=True)
```

```
df_num.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      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     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

```
#11.Display the sum of missing values after filling the values
((df.isnull().sum().sum())*median)
```

```
90045.0
```

```
# 12. Transform “median_income” attribute into a new attribute “i
#from 0-1.5, 1.5-3.0, 3.0-4.5, 4.5-6.0, 6.0-np.inf respectively.
# Use pd.cut(df["median_income"], bins=[0., 1.5, 3.0, 4.5, 6., np
import numpy as np
df["income_cat"] = pd.cut(df["median_income"],
bins=[0., 1.5, 3.0, 4.5, 6., np.inf],
labels=[1, 2, 3, 4, 5])
```

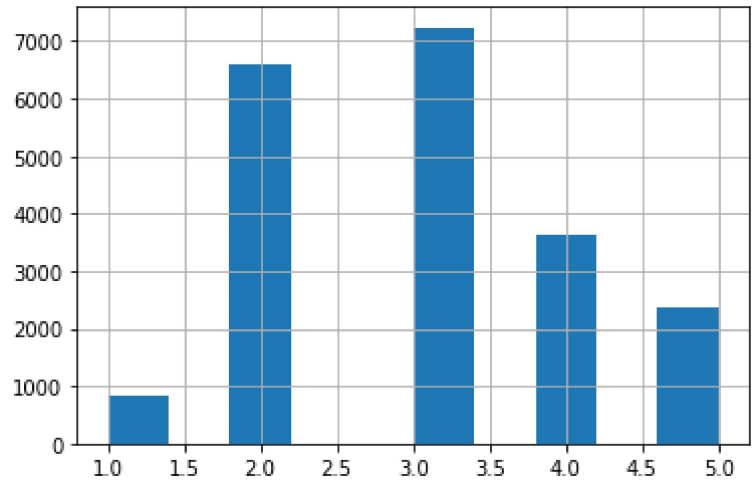
```
# 13. Find the distribution based on “income_cat” in the entire d
df["income_cat"].value_counts() / len(df)
```

```
3    0.350581
2    0.318847
4    0.176308
5    0.114438
1    0.039826
Name: income_cat, dtype: float64
```

```
# 14. Plot histogram of “income_cat” attributes. (use df[‘attribu
```

```
df["income_cat"].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbffd9c81d0>
```




```
# 15. Split the dataset 80% of rows for training, and 20% of rows
#of learning take first 80% rows as training and, rest 20% rows a
#train and test datasets in temp_train and temp_test variables.
def split_train_test(data, test_ratio):
        shuffled_indices = np.random.permutation(len(data))
        test_set_size = int(len(data) * test_ratio)
        test_indices = shuffled_indices[:test_set_size]
        train_indices = shuffled_indices[test_set_size:]
        return data.iloc[train_indices], data.iloc[test_indices]
temp_train, temp_test = split_train_test(df, 0.2)
print(len(temp_train))
print(len(temp_test))
```

```
16512
4128
```

```
# 16. Check the distribution based on “income_cat” in train and t
df["income_cat"].value_counts() / len(df)
```

```
3    0.350581
2    0.318847
4    0.176308
5    0.114438
1    0.039826
Name: income_cat, dtype: float64
```

```
temp_train["income_cat"].value_counts() / len(df)
```

```
3    0.281638
2    0.252665
4    0.141279
5    0.092103
1    0.032316
Name: income_cat, dtype: float64
```

```
temp_test["income_cat"].value_counts() / len(df)
```

```
3    0.068944
2    0.066182
4    0.035029
5    0.022335
1    0.007510
Name: income_cat, dtype: float64
```

```
# 17. Reshufle the dataset to have stratified distribution of ‘in
from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_
for train_index, test_index in split.split(df, df["income_cat"]):
  train = df.loc[train_index]
  test = df.loc[test_index]
print(len(train))
print(len(test))
```

```
16512
4128
```

```
# 18. Check again the distribution based on “income_cat” in train
df["income_cat"].value_counts() / len(df)
```

```
3    0.350581
2    0.318847
4    0.176308
5    0.114438
1    0.039826
Name: income_cat, dtype: float64
```

```
train["income_cat"].value_counts() / len(train)
```

```
3    0.350594
2    0.318859
```

```
4    0.176296
5    0.114462
1    0.039789
Name: income_cat, dtype: float64
```

```
test["income_cat"].value_counts() / len(train)
```

```
3    0.087633
2    0.079700
4    0.044089
5    0.028585
1    0.009993
Name: income_cat, dtype: float64
```

```
# 19. Find correlation of target attribute with rest of the attri
correlation=df.corr()
correlation["median_house_value"].sort_values()
```

```
latitude            -0.144160
longitude           -0.045967
population          -0.024650
total_bedrooms       0.049686
households           0.065843
housing_median_age   0.105623
total_rooms          0.134153
median_income        0.688075
median_house_value   1.000000
Name: median_house_value, dtype: float64
```

```
# 20. Convert categorical attribute to numeric using ordinal enco
from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder ()
df_cat_oe =oe.fit_transform(df[["ocean_proximity"]])
print(df_cat_oe)
```

```
[[3.]
 [3.]
 [3.]
 ...
 [1.]
 [1.]
 [1.]]
```

```
# 21. Add the new attribute that you have transformed into numeri
d = dict(enumerate(df_cat_oe.flatten(), 1))
```

```
df["df_cat_oe"]=d.values()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 12 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  
 10  income_cat          20640 non-null  category
 11  df_cat_oe           20640 non-null  float64 
dtypes: category(1), float64(10), object(1)
memory usage: 1.8+ MB
```

```
# 22. Drop the attribute which has categorical values from the da
df=df.drop("ocean_proximity" , axis=1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 11 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   income_cat          20640 non-null  category
 10  df_cat_oe           20640 non-null  float64 
dtypes: category(1), float64(10)
memory usage: 1.6 MB
```

```
# 23. Split the dataset. use sklearn.model_selection import train
# train_set, test_set = train_test_split(housing, test_size=0.2,r
from sklearn.model_selection import train_test_split
```

```
train_set, test_set = train_test_split(df, test_size=0.2, random_
print(len(train_set))
print(len(test_set))
```

```
16512
4128
```

```
# 24. Separate the target attribute and rest of the attributes fr
#test_target in two separate variables.
train_target = train_set[target]
train_rest=train_set.drop("median_house_value",axis=1)
test_target = test_set[target]
test_rest = test_set.drop("median_house_value",axis=1)
#train_target.info()
print("\n")
#test_target.info()
```

```
# 25. Take a linear regression mode and train it.
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
```

```
# 26. reg.fit(training_dataset_name, training_dataset_target)
reg.fit(train_set, train_target)
reg.intercept_
reg.coef_
```

```
array([-5.46419322e-11, -5.75836046e-11,  3.56863248e-14,  2.14793461e-14,
       -1.11005823e-13, -9.12811493e-15,  4.85739921e-14, -1.20912156e-11,
        1.00000000e+00,  5.65876585e-13, -1.01344104e-13])
```

```
# 27. Predict few values from the dataset. Use predict method and
train_set_rand = train_set.sample(frac=0.50)
reg.predict(train_set_rand)
```

```
array([256100., 325800.,  70700., ..., 154400., 283200., 271300.])
```

✓ 0s completed at 11:36 PM