Double-click (or enter) to edit

Name: Harshali Bhoye Roll No.: 21102A0040 BE CMPN A GitHub link:

```
import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'california-housing-prices:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F5227%2F7876%2Fbundle%2Farchive.zip%3FX-Goog-Algorithm%3DGOOG4-RSA

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading california-housing-prices, 409382 bytes compressed
[==================================================] 409382 bytes downloaded
Downloaded and uncompressed: california-housing-prices
Data source import complete.
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Read The Dataset

```
housing = pd.read_csv(r"/kaggle/input/california-housing-prices/housing.csv")
housing.head(10)
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_inco
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.32
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.30
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.25
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.64
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.84
5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.03
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.65
7	-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.12
8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.08
9	-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.69

EDA

```
housing.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
```

```
housing.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.00000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.53968
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.32975
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.00000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.00000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.00000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.00000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.00000

```
housing.nunique()
```

```
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
ocean_proximity           5
dtype: int64
```

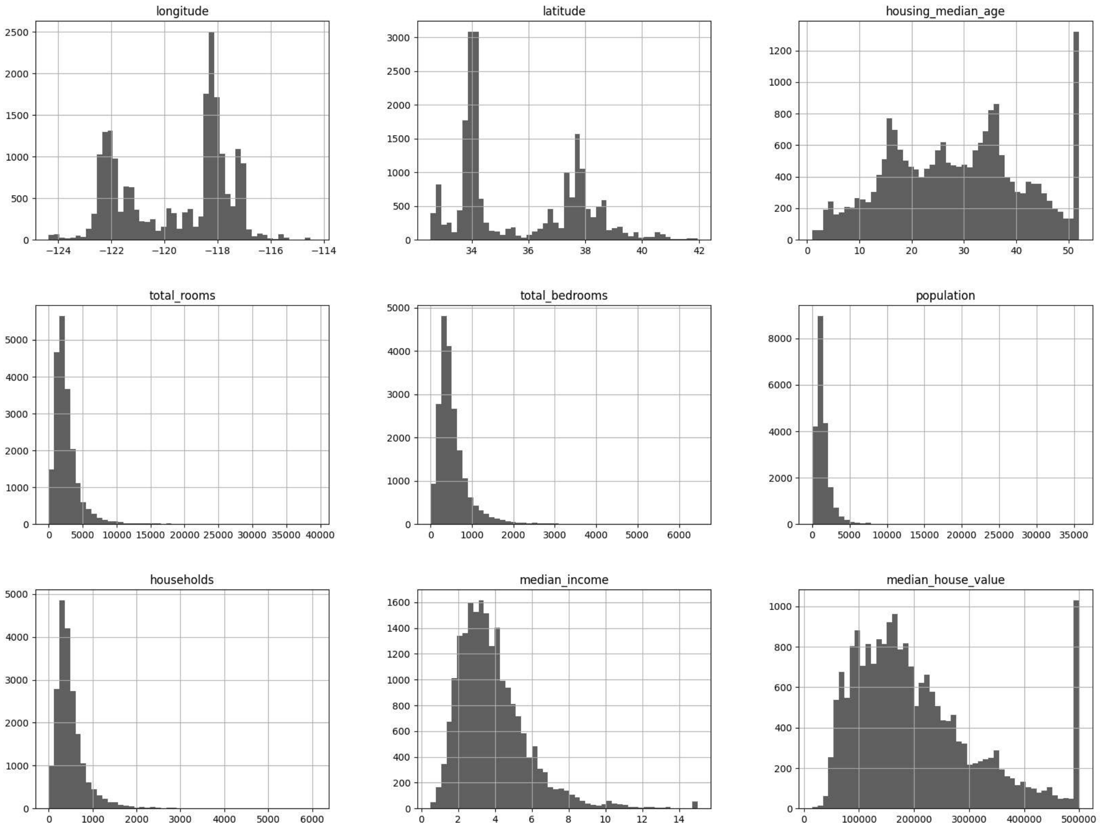
```
housing.isnull().sum()
```

```
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
```

Visualization

```
housing.hist(bins=50,figsize=(20,15))
```

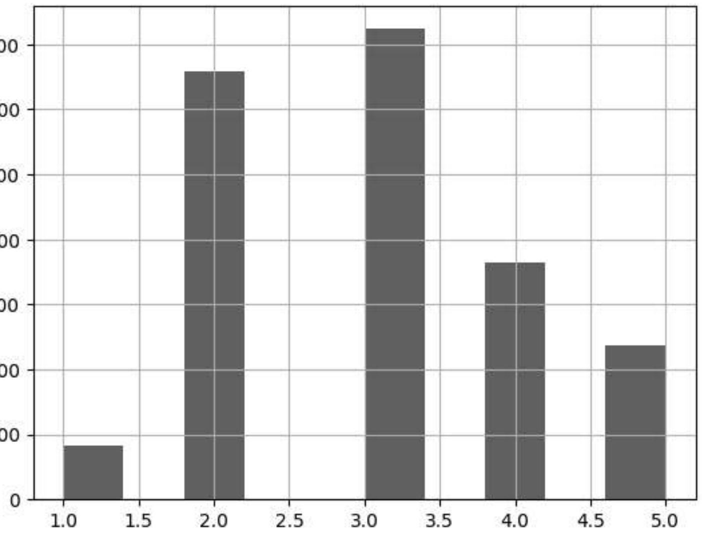
```
array([[<Axes: title={'center': 'longitude'}>,
        <Axes: title={'center': 'latitude'}>,
        <Axes: title={'center': 'housing_median_age'}>],
       [<Axes: title={'center': 'total_rooms'}>,
        <Axes: title={'center': 'total_bedrooms'}>,
        <Axes: title={'center': 'population'}>],
       [<Axes: title={'center': 'households'}>,
        <Axes: title={'center': 'median_income'}>,
        <Axes: title={'center': 'median_house_value'}>]], dtype=object)
```



```
housing['income_category'] = pd.cut(housing['median_income'], bins = [0,1.5,3,4.5,6, np.inf], labels=[1,2,3,4,5])
```

```
housing['income_category'].hist()
```

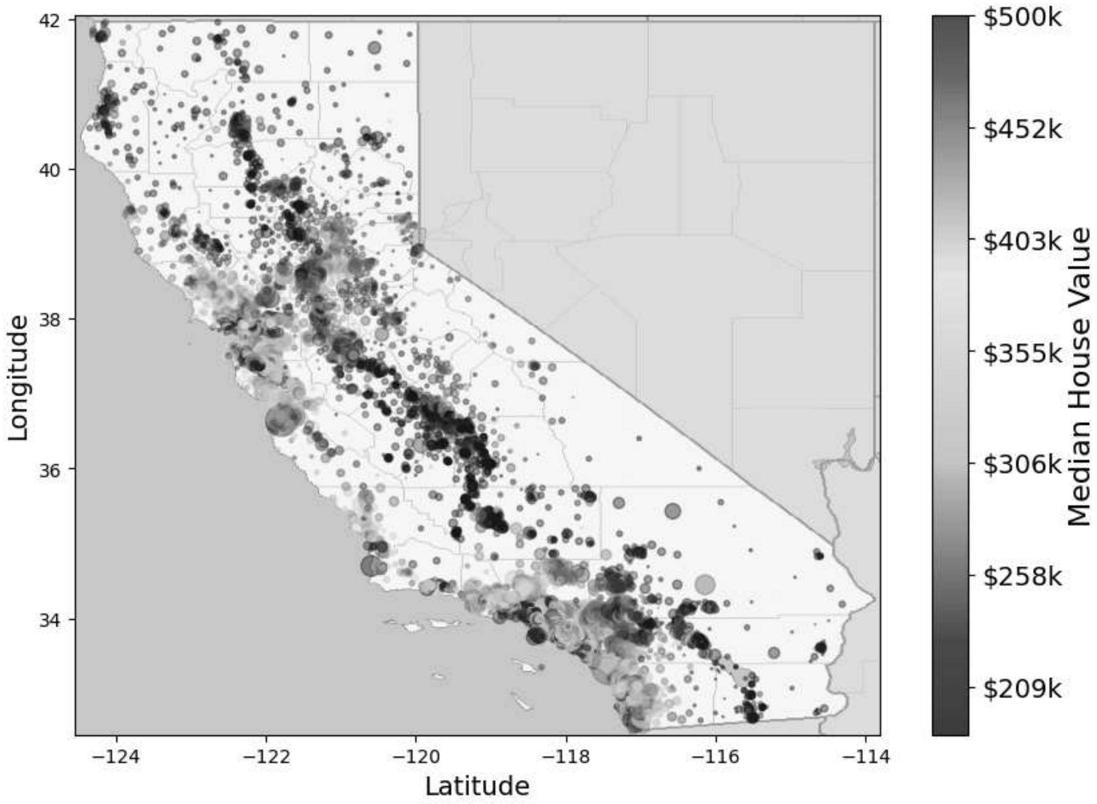
```
<Axes: >
```



```
import urllib.request
import io
import matplotlib.image as mpimg

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml2/master/"
filename = "california.png"
print("Downloading", filename)
url = DOWNLOAD_ROOT + "images/end_to_end_project/" + filename
with urllib.request.urlopen(url) as url_request:
    image_data = url_request.read()

image_data = io.BytesIO(image_data)
california_img = mpimg.imread(image_data, format='png')
```

```
Downloading california.png
```

```
ax = housing.plot(kind='scatter', x='longitude',y='latitude', figsize=(10,7),s=housing['population']/100,
                  c='median_house_value',colorbar=False,cmap=plt.get_cmap('jet'),alpha=0.4)

plt.imshow(california_img,alpha=0.8, extent=[-124.55, -113.80, 32.45, 42.05], cmap=plt.get_cmap('jet'))
plt.xlabel('Latitude',fontsize=14)
plt.ylabel('Longitude',fontsize=14)

prices = housing["median_house_value"]
tick_values = np.linspace(prices.min(), prices.max(), 11)
cbar = plt.colorbar(ticks=tick_values/prices.max())
cbar.ax.set_yticklabels(["$%dk"%(round(v/1000)) for v in tick_values], fontsize=14)
cbar.set_label('Median House Value', fontsize=16)
#plt.legend(fontsize=16)
```

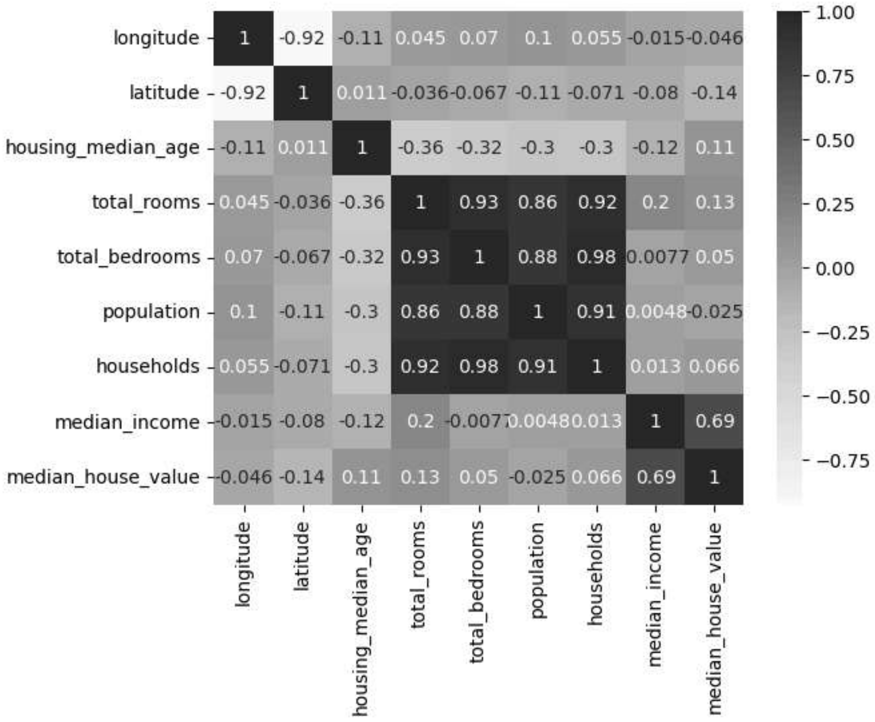


```
corr_matrix = housing.corr(numeric_only=True)
corr_matrix['median_house_value'].sort_values(ascending=False)
```

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

```
sns.heatmap(corr_matrix,annot=True,cmap='Blues')
```

```
<Axes: >
```



```
housing_eda = housing.copy()
```

Preprocessing

```
housing_eda['rooms_per_household'] = housing_eda['total_rooms'] / housing_eda['households']
housing_eda['bedrooms_per_room'] = housing_eda['total_bedrooms'] / housing_eda['total_rooms']
housing_eda['population_per_houshold'] = housing_eda['population'] / housing_eda['households']
```

```
corr_matrix = housing_eda.corr(numeric_only=True)
corr_matrix['median_house_value'].sort_values(ascending=False)
```

```
median_house_value         1.000000
median_income              0.688075
rooms_per_household        0.151948
total_rooms                0.134153
housing_median_age         0.105623
households                 0.065843
total_bedrooms             0.049686
population_per_houshold   -0.023737
population                -0.024650
longitude                 -0.045967
latitude                  -0.144160
bedrooms_per_room         -0.255880
Name: median_house_value, dtype: float64
```

split the data

```
from sklearn.model_selection import train_test_split

x = housing.drop(columns='median_house_value')
y = housing['median_house_value']

X_train,X_test, Y_train, Y_test = train_test_split(x,y,test_size=0.2)
```

Feature engineering

```
from sklearn.base import BaseEstimator, TransformerMixin
rooms_ix, bedrooms_ix, population_ix, households_ix = 3, 4, 5, 6
```

Pipline

Handling missing values and scaling

```
        return self
```

```
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline= Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('attrib_adder', CombinedAttributeAdder(add_bedrooms_per_room=True, add_rooms_per_household=True, add_population_per_household=False)),
    ('std_scaler', StandardScaler()),
])
```

Transformation

Transform numeric and categorical columns and encoding

```
            X[households_ix]=np.log(X[households_ix]+1)
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
num_attribs = X_train.select_dtypes(include=['int64', 'float64']).columns.tolist()
cat_attribs=['ocean_proximity']
full_pipline=ColumnTransformer([
    ('num', num_pipeline, num_attribs),
    ('cat', OneHotEncoder(), cat_attribs)
])
processed_X_train=full_pipline.fit_transform(X_train)
processed_X_test =full_pipline.transform(X_test)
```

Modeling

Linear regression

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
reg=LinearRegression()
reg.fit(processed_X_train, Y_train)


scores = cross_val_score(reg,processed_X_train,Y_train, scoring='neg_mean_squared_error',cv=10)
print('cross validation scores :',np.sqrt(-scores).mean(),'\n')
```

```
cross validation scores : 67888.96025218003
```

Random forest

```
from sklearn.ensemble import RandomForestRegressor

forest_reg = RandomForestRegressor(n_estimators=100, random_state=42)

forest_reg.fit(processed_X_train, Y_train)

scores = cross_val_score(forest_reg,processed_X_train,Y_train, scoring='neg_mean_squared_error',cv=10)
print('cross validation scores :',np.sqrt(-scores).mean(),'\n')
```

```
cross validation scores : 49322.91141668432
```

Evaluation

```
from sklearn.metrics import accuracy_score

train_score = reg.score(processed_X_train,Y_train)
test_score = reg.score(processed_X_test,Y_test)

print('Linear regression score: \n')
print('Train score : ',round(train_score*100),'%')
print('Test score : ',round(test_score*100),'%')
```

```
Linear regression score:

Train score :  65 %
Test score :  64 %
```