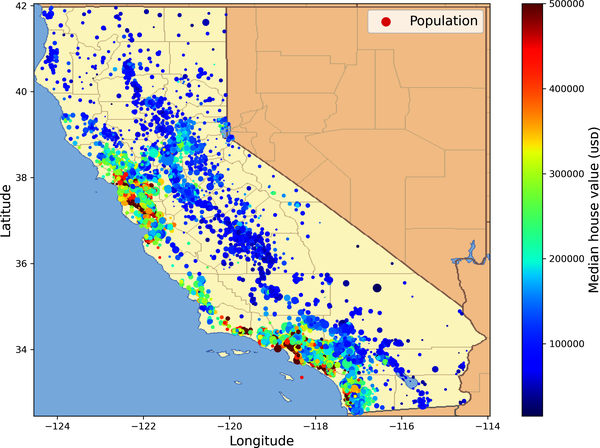
BU.330.775 Machine Learning: Design and Deployment

**Lab 2. Explore and clean a housing dataset**

Learning Goal: practice data preparation and exploratory data analysis on a California housing dataset

Background: This example is curated from Geron (2022). The California housing dataset includes metrics such as the population, median income, and median housing price for each block group. Block groups are the smallest geographical unit used by the US Census Bureau. A block group typically has a population of 600 to 3,000 people. We will follow the author in referring to them as “districts” in the exercise.



1. First let’s load the dataset.

from pathlib import Path

import pandas as pd

import tarfile

import urllib.request

def load\_housing\_data():

tarball\_path = Path("datasets/housing.tgz")

if not tarball\_path.is\_file():

Path("datasets").mkdir(parents=True, exist\_ok=True)

url = "https://github.com/ageron/data/raw/main/housing.tgz"

urllib.request.urlretrieve(url, tarball\_path)

with tarfile.open(tarball\_path) as housing\_tarball:

housing\_tarball.extractall(path="datasets")

return pd.read\_csv(Path("datasets/housing/housing.csv"))

housing = load\_housing\_data()

1. We will start by looking at the top five rows of data.

housing.head()

1. Get a description of the dataset, such as the total number of rows, each attribute’s type, and the number of non-empty values.

housing.info()

1. Obtain a count for each value in the ocean\_proximity column.



housing["ocean\_proximity"].value\_counts()

1. We can also return the statistical summary of each column. Note that it only works on numeric features.



housing.describe()



1. We will use the matplotlib to generate histograms for the numeric columns in the dataset to visualize the distributions.

import matplotlib.pyplot as plt

# this is a comment – the next 5 lines define the plot style

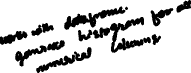
plt.rc('font', size=14)

plt.rc('axes', labelsize=14, titlesize=14)

plt.rc('legend', fontsize=14)

plt.rc('xtick', labelsize=10)

plt.rc('ytick', labelsize=10)



housing.hist(bins=50, figsize=(12, 8))



plt.show()

**Homework Question 1 (7pt):** Describe each of the seven plots: What information can you observe?5

* 1. housing\_median\_age
  2. total\_rooms
  3. total\_bedrooms
  4. population
  5. households
  6. median\_income
  7. median\_house\_value

1. Next, let’s visualize the geographical data to gain more insights, showing the latitude and longitude of individual data points.

housing.plot(kind="scatter", x="longitude", y="latitude", grid=True)

plt.show()

1. Add a new parameter, alpha, to control the transparency of the scatter. Transparency provides a better view of how data points overlap. If multiple data points are located in the same or close proximity, you can make the overlap of these points more obvious by reducing transparency. Multiple overlapping points are darkened, showing dense areas of data.

housing.plot(kind="scatter", x="longitude", y="latitude", grid=True, alpha=0.2)

plt.show()

1. Generate a color and size scatter plot that shows the relationship between geographic location, population, and the median home value.

Set the size of each scatter, which is proportional to the population. Here the population is scaled by dividing it by 100 to prevent the scatter from getting too large. Choose a colormap, jet is a common color gradient, usually from blue to red. Blue means low house prices, red means high house prices.

housing.plot(kind="scatter", x="longitude", y="latitude", grid=True,

s=housing["population"] / 100, label="population",

c="median\_house\_value", cmap="jet", colorbar=True,

legend=True, sharex=False, figsize=(10, 7))

plt.show()

**Homework Question 2 (1pt):** What insights can you derive from the last visualization?

1. By calculating the correlation matrix in the dataset, we look for which features have a strong correlation with median\_house\_value. Correlation is a statistic that measures the strength of the linear relationship between two variables, with values ranging between

-1 and 1. We perform the task only on numeric features, and sort them in descending order.



corr\_matrix = housing.corr(numeric\_only=True)

corr\_matrix["median\_house\_value"].sort\_values(ascending=False)



1. The relationship between median\_income and median\_house\_value is shown by generating a scatter plot. Refer to [https://en.wikipedia.org/wiki/Correlation](https://www.google.com/url?q=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FCorrelation) for sample scatter plots illustrating different correlations.

housing.plot(kind="scatter", x="median\_income",

y="median\_house\_value",

alpha=0.1, grid=True)

plt.show()

1. Now, we can experiment with creating new attributes, such as the average number of rooms per household, the ratio of bedrooms to total rooms, and the average number of households. And then calculate the correlations again.

housing["rooms\_per\_house"] = housing["total\_rooms"] / housing["households"]

housing["bedrooms\_ratio"] = housing["total\_bedrooms"] / housing["total\_rooms"]

housing["people\_per\_house"] = housing["population"] / housing["households"]

corr\_matrix = housing.corr(numeric\_only=True)

corr\_matrix["median\_house\_value"].sort\_values(ascending=False)



**Homework Question 3 (2pt):** Compare the correlations to those in step 10. What can you say about the three new features: rooms\_per\_house, bedroom\_ratio, and people\_per\_house?

1. Now, we can divide the dataset into a training set and a testing set using an 80/20 split. Here random\_state is a parameter that can be used to seed the random number generator during the split process. This ensures that the random selection of data points to include in the training and testing sets is reproducible, which means, If you use the same random\_state value every time you run train\_test\_split, you will get the same split of data each time.

from sklearn.model\_selection import train\_test\_split

train\_set, test\_set = train\_test\_split(housing, test\_size=0.2, random\_state=42)

1. Let’s try stratified sampling based on the income category. Recall that the purpose of stratified sampling is to ensure that the proportion of data in each income range can be consistent in the training set and the test set, which can prevent the problem of uneven data distribution.

First create a new income category column based on median\_income, and check the distribution.

import numpy as np

housing["income\_cat"] = pd.cut(housing["median\_income"],

bins=[0., 1.5, 3.0, 4.5, 6., np.inf],

labels=[1, 2, 3, 4, 5])

housing["income\_cat"].value\_counts().sort\_index().plot.bar(rot=0, grid=True)



plt.xlabel("Income category")



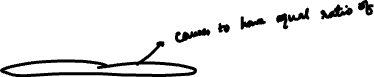
plt.ylabel("Number of districts")

plt.show()



1. Then perform the stratified sampling and check the distribution after sampling.

strat\_train\_set, strat\_test\_set = train\_test\_split(housing, test\_size=0.2, stratify=housing["income\_cat"], random\_state=42)



strat\_test\_set["income\_cat"].value\_counts() / len(strat\_test\_set)

1. Separate features and target labels.



housing = strat\_train\_set.drop("median\_house\_value", axis=1)



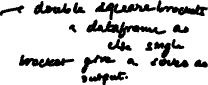
housing\_labels = strat\_train\_set["median\_house\_value"].copy()



1. Now, let’s handle text and categorical attribute in this dataset, which is the ocean\_proximity column.

housing\_cat = housing[["ocean\_proximity"]]

housing\_cat.head(8)



1. First try the ordinal encoding.



from sklearn.preprocessing import OrdinalEncoder

ordinal\_encoder = OrdinalEncoder()

housing\_cat\_encoded = ordinal\_encoder.fit\_transform(housing\_cat)

housing\_cat\_encoded[:8]

1. And then one-hot encoding which is a binary representation of each category.

from sklearn.preprocessing import OneHotEncoder



cat\_encoder = OneHotEncoder(sparse\_output=False)

housing\_cat\_1hot = cat\_encoder.fit\_transform(housing\_cat)



housing\_cat\_1hot[:8]

There are three ways to handle NaN (null or missing values) in a dataset. **The following code is for demonstration. DO NOT RUN!**



housing.dropna(subset=["total\_bedrooms"], inplace=True)    # option 1  
  
housing.drop("total\_bedrooms", axis=1)       # option 2  
  
median = housing["total\_bedrooms"].median()  # option 3  
housing["total\_bedrooms"].fillna(median, inplace=True)



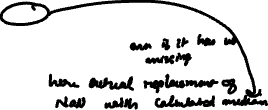
1. In this exercise, we will use an imputation strategy for all numeric features. *Which one should we choose: zero, mean, or median?*

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy="median")

housing\_num = housing.select\_dtypes(include=[np.number])

imputer.fit(housing\_num)



1. Now transform the training set and examine the basic information of the data.



X = imputer.transform(housing\_num)



housing\_tr = pd.DataFrame(X, columns=housing\_num.columns,



index=housing\_num.index)

housing\_tr.info()

1. For outliers, we can use a function to predict them.

from sklearn.ensemble import IsolationForest

isolation\_forest = IsolationForest(random\_state=42)

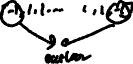
outlier\_pred = isolation\_forest.fit\_predict(X)



outlier\_pred



If you wanted to drop outliers, you would run the following code:



#housing = housing.iloc[outlier\_pred == 1]



#housing\_labels = housing\_labels.iloc[outlier\_pred == 1]

1. Let’s reprint the DataFrame and try a few scaling strategies.

print(housing\_num)

1. First, apply min-max scaling to the housing\_median\_age attribute. You can achieve this by either generating with AI or using the following code.

# prompt: convert housing\_median\_age column using minmaxscaler and print the whole dataframe housing\_num after convert

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

housing\_num['housing\_median\_age'] = scaler.fit\_transform(housing\_num[['housing\_median\_age']])

print(housing\_num)

1. Then, apply standardization on the median\_income column. There are two ways as well.

# prompt: convert median\_income column using standardscaler and print the whole dataframe housing\_num after convert

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

housing\_num['median\_income'] = scaler.fit\_transform(housing\_num[['median\_income']])

print(housing\_num)

1. Finally, let’s check the DataFrame after scaling. This is the end of Lab 2.

housing\_num.describe()

**Submission**: Complete all the lab steps and the 3 homework questions. Save your file as homework2\_yourname.ipynb and submit on Canvas by the beginning of class 3.