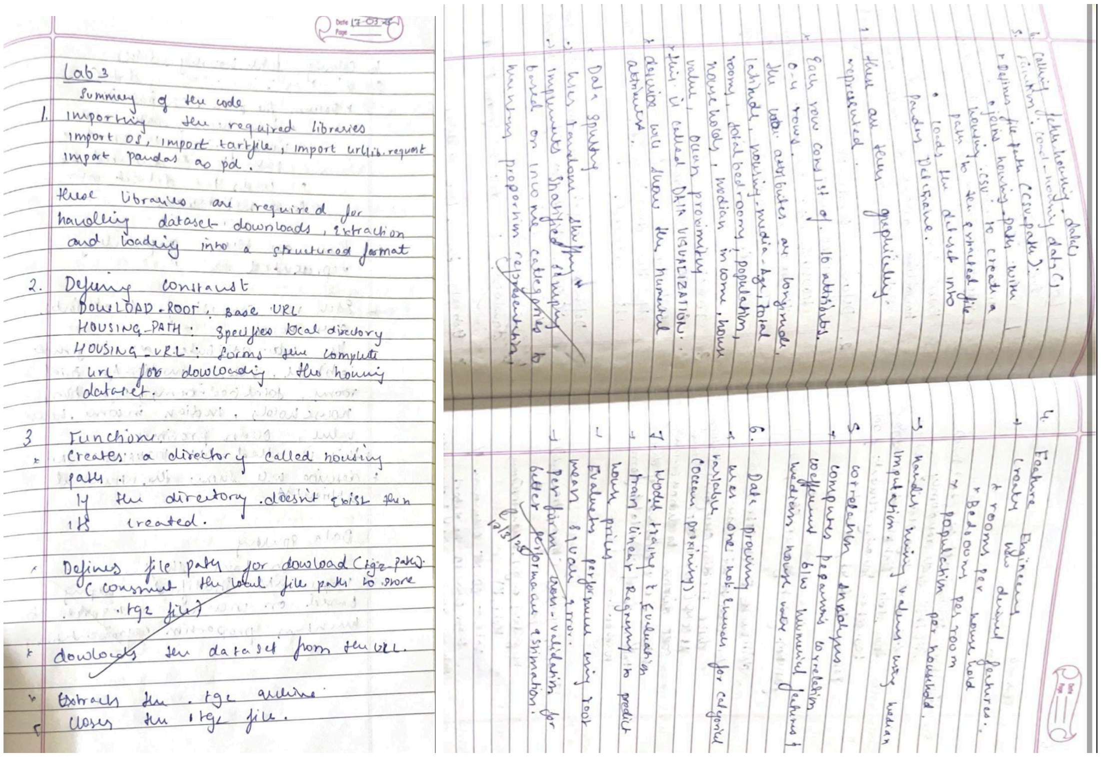
Demonstrate various data pre-processing techniques for a given dataset



Date 17-03-25

Lab 3

Summary of the code

1. importing the required libraries
import os, import tarfile, import urllib.request
import pandas as pd.

these libraries are required for handling dataset downloads, extraction and loading into a structured format

2. Defining constant
DOWNLOAD-ROOT: Base URL
HOUSING-PATH: Specifies local directory
HOUSING-URL: forms the complete url for downloading the housing dataset.

3 Function:
- Creates a directory called housing path.
If the directory doesn't exist then its created.

- Defines file path for download (tgz-path). (construct the local file path to store .tgz file)

- downloads the dataset from the URL.

- Extracts the .tgz archive
- closes the .tgz file.

4. calling fetch_housing_data()
5. Function: load-housing data():
- Defines file path (csv-path):
- joins housing-path with housing.csv to create a path to the extracted file
- loads the dataset into Pandas DataFrame.

- these are then graphically represented.

- Each row consist of 10 attributes.
0-4 rows.
the total attributes are longitude, latitude, housing-median-Age, total rooms, total bedrooms, population, households, median income, house value, ocean proximity
- this is called DATA VISUALIZATION.
- describe will show the numerical attributes.

- Data splitting
- uses random shuffling
- implements Stratified sampling based on income categories to maintain proportion representation.

4. Feature Engineering
- creates new derived features.
- rooms per household
- Bedrooms per room
- population per household.
- handles missing values using median imputation.

5 correlation analysis
- computes pearson's correlation coefficient b/w numerical features & median house value.

6. Data processing
- uses one hot encoder for categorical variable (ocean proximity)

7. Model training & Evaluation
- train linear Regression to predict house prices
- Evaluates performance using root mean square error.
- Perform cross-validation for better performance estimation.

17/3/25

Code:

```
import pandas as pd
def load_housing_data():
  csv_path = r'/content/housing.csv'
  return pd.read_csv(csv_path)

housing = load_housing_data()
```

```
import numpy as np
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]
train_set, test_set = split_train_test(housing, 0.2)
```

```
from sklearn.model_selection import StratifiedShuffleSplit
```

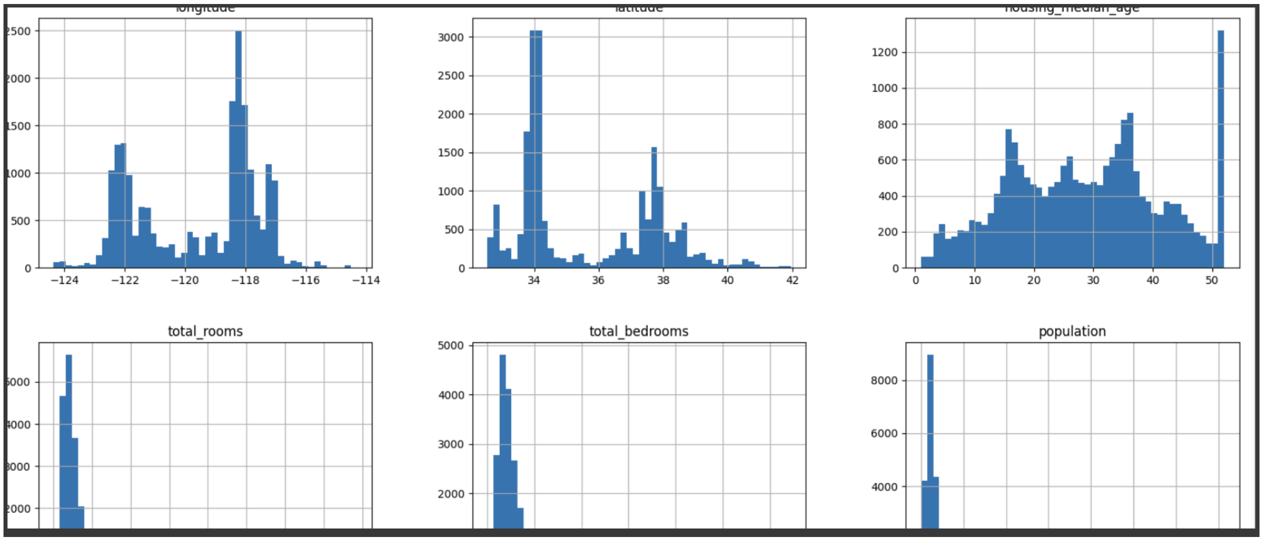
```
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(housing,
housing["income_cat"]):
  strat_train_set = housing.loc[train_index]
  strat_test_set = housing.loc[test_index]

for set_ in (strat_train_set, strat_test_set):
  set_.drop("income_cat", axis=1, inplace=True)
X = imputer.transform(housing_num)
housing_tr = pd.DataFrame(X, columns=housing_num.columns,
index=housing_num.index)
housing_cat = housing[["ocean_proximity"]]
housing_cat.head(10)
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)
some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
some_data_prepared = full_pipeline.transform(some_data)
print("Predictions:", lin_reg.predict(some_data_prepared))
print("Labels:", list(some_labels))

from sklearn.metrics import mean_squared_error
housing_predictions = lin_reg.predict(housing_prepared)
lin_mse = mean_squared_error(housing_labels, housing_predictions)
lin_rmse = np.sqrt(lin_mse)
print(f"lin_rmse: {lin_rmse}")
```

Output:

```
[[0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 ...
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]]
[[1.]
 [4.]
 [1.]
 [4.]
 [0.]
 [3.]
 [0.]
 [0.]
 [0.]
 [0.]]
```

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

```
ocean_proximity

     12655                INLAND

     15502            NEAR OCEAN

      2908                INLAND

     14053            NEAR OCEAN

     20496             <1H OCEAN

      1481              NEAR BAY

     18125             <1H OCEAN

      5830             <1H OCEAN

     17989             <1H OCEAN

                       <1H OCEAN
      4861
```


longitude
latitude
housing_median_age
total_rooms
total_bedrooms
population