PMDS505P Data Mining and Machine Learning Lab
Experiment 1

December 2024

1 Recap of working with pandas and certain Data Preprocessing techniques, mainly data transformation techniques

Today we will have a recap of pandas and certain data preprocessing techniques that would be used in the our algorithms and few other concepts as well.

Note: Take screenshots of all the output and make a single PDF file.

- Download the csv file from the Coursepage which is a dataset on a housing information.
- Load the file to your python program in to a data frame DF1.
- Print the following details of the data frame DF1.
    1. First five observations from your dataset
    2. Last five observations from our dataset
    3. Shape of your dataset
    4. info of your dataset
- Create a new dataframe DF2 with the first 100 observations of your dataset with only the columns 'LotArea' and 'BedroomAbvGr' [Use iloc operator]
- Write or export your dataset DF2 to a csv file DF11.csv and save it.
- Find the maximum and minimum of the 'LotArea' column for your dataset DF1
- Find the observations from your dataset with LotArea $>$ 10650 from your DF1 data frame.
- Find the mean, median of your column 'TotalBsmtSF' and find the unique entries (non-repeated ones)
- Sort the dataset DF1 according to the 'TotalBsmtSF' column of your dataset DF1 in ascending and descending order

- Find the empty cells in 'GarageArea' column of your dataset DF1 and fill it with the average value of the column 'GarageArea'
- Replace the column named Above median price in your dataframe with 1's where ever you habe Yes and 0 where ever you have No.
- Draw a scatterplot with columns 'GarageArea' on x axis and 'LotArea' on y-axis
- Drop the column 'GarageArea' from your dataset DF1

Data Transformation: Data Normalization

Some data transformation techniques that we will see next. In Machine Learning, we often need to deal with normalization of data improving the training process and better prediction of models.

Min-Max normalization Normalization is the process of scaling the data to the range of 0 to 1. or to a new range, say [a,b]

The MinMaxScaler() uses the following idea as given below:

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

and converts the values in the columns in the dataframe to a range of [0,1].

Q1. Now, normalize the columns of the dataset1 using the above technique and save it to a new csv file DF3.csv

You can refer this code for the same.

```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
colname = DF1.columns[:]
x = DF1.loc[:, colname]
DD = min_max_scaler.fit_transform(x)
x = pd.DataFrame(DD,columns = colname)
print(x)
```

Now if you want to make the data to a range of [-1,1] you can use

```
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-1,1))
```

while creating an object of your class MinMaxScaler.

Q2. Now normalize the whole data to the range (2,3) using Min-Max normalization

Decimal scaling Decimal Scaling is a normalization technique that scales data by moving the decimal point. It is done by dividing each value of a column by 10^j, where j is the smallest integer such that the maximum absolute value of the scaled data is less than 1.

In other words:

$$X_{\text{scaled}} = \frac{X}{10^j}$$

Here:
X : Original value.
j : The number of digits in the largest absolute value of the column.

Steps to Perform Decimal Scaling for a Column in a DataFrame:

1. Find the maximum absolute value of the column.
2. Determine j, the number of digits in the largest absolute value.
3. Scale the column by dividing each value by 10^j

Q3. Now do decimal scaling for the original column data of the column LotArea of your initial dataframe and print the results.

ZScore normalization

Here we do standardization of the data. That is StandardScaler is a preprocessing technique provided by sklearn.preprocessing that standardizes data by transforming it to have a mean of 0 and a standard deviation of

Transformation is done as:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

Where:
- X = Original feature value
- μ = Mean of the feature
- σ = Standard deviation of the feature.

Q4. Now try to standardize the whole data in the dataframe and print the dataframe.

here you have to use the class **StandardScaler()** instead of **MinMaxScaler()**. Proceed as in question 1 and make the above change.

Train Test splitting of data for model training.

Now we will see how the splitting of data is done into train and testing parts using some inbuilt functions. In Python, the train_test_split function from the sklearn.model_selection module is commonly used to split a dataset into training and testing subsets.

Here's an example of how to use it with a Pandas DataFrame:
Import necessary libraries: train_test_split from sklearn.model_selection
pandas for DataFrame manipulation
Load your DataFrame (assuming you already have a DataFrame called df).
Split your data using train_test_split

Example code

```
import pandas as pd
from sklearn.model_selection import train_test_split
# Example DataFrame (you can replace this with your own DataFrame)
data = { 'Feature1': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'Feature2': [10, 9, 8, 7, 6, 5, 4, 3, 2, 1],
'Target': [0, 1, 0, 1, 0, 1, 0, 1, 0, 1] }
df = pd.DataFrame(data)
# Define your features (X) and target (y)
X = df.drop('Target', axis=1)
# Features (everything except 'Target')
y = df['Target']
# Target (the 'Target' column)

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Print the testing and training data
print("Training Features:", X_train)
print("Testing Features:", X_test)
print("Training Target:", y_train)
print("Testing Target:", y_test)
```

Now perfrom 70:30 train test split for our dataframe data with the target variable or output variable as LotArea and print the training and testing data which may be we can use for fitting a model for this data. Say like trying to predict the LotArea for a house based on all the other features.

Solution:

1. To load the dataset in to a dataframe DF2 use the code:

   DF1 = pd.read_csv("housepricedata.csv")

2. Use DF1.head(). DF1.tail(), DF1.info() and DF1.shape to obtain the details of the data frame DF1.

3. DF2 = DF1.loc[0:100,['LotArea','BedroomAbvGr']]

4. To create or export a csv file named DF11 use the code

   DF2.to_csv('DF11.csv')

5. use DF1.LotArea.max(), DF1.LotArea.min()

6. use the code DF1[DF1.LotArea>10650]

7. use the code DF1.TotalBsmtSF.mean(), DF1.TotalBsmtSF.median(), DF1.TotalBsmtSF.unique()

8. use the code

   DF1.sort_values(by=['TotalBsmtSF'],ascending=False)

   DF1.sort_values(by=['TotalBsmtSF'])

9. use the code

   DF1.GarageArea.isnull()

   DF1.GarageArea.fillna(472.614,inplace=True)

   print(DF1)

10. DF1['Abovemedianprice'] = DF1['Abovemedianprice'].replace("Yes": 1, "no": 0)
    print(DF1)

11. import matplotlib.pyplot as plt
    plt.scatter(DF1.GarageArea, DF1.LotArea, label = 'points', color = 'r')
    plt.legend()

12. use the code

    DF1.drop('GarageArea', axis=1, inplace = True)

    print(DF1)

Normalizing the data code: use the following code

from sklearn import preprocessing

min_max_scaler = preprocessing.MinMaxScaler()

col_name = DF1.columns[:]

x = DF1.loc[:, col_name]

x = pd.DataFrame(data = min_max_scaler.fit_transform(x), columns = col_name)

print(DF1.head())

print(x.head())

To export a new csv file use the code

x.to_csv('DF3.csv')