National Institute of Technology
Tiruchirappalli, Tamil Nadu – 620 015

Data Preprocessing

Date: 16.03.2022

Exercise 1: Load from Built-In Dataset

a. Normal Load:

```
from sklearn import datasets
iris = datasets.load_iris()
```

#Type the following in console

```
iris
iris.data
iris.target
```

del iris

clear

---

b. Using Pandas:

```
from sklearn import datasets
import pandas as pd
df=pd.DataFrame(datasets.load_iris()['data'],columns=datasets.load_iris()['feature_names'])
```

#Type the following in console

```
df.head()
df.tail()
df.shape
```

del df

clear

---------- **XXX** ----------

Exercise 2: Load from URL

```
import numpy as np
import urllib.request
```

 # URL for the Pima Indians Diabetes dataset

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"

 # Interface to fetch the URL. It can handle basic authentication, cookies, proxies, etc.

raw_data = urllib.request.urlopen(url)

 # Load the content as a numpy matrix

dataset = np.loadtxt(raw_data, delimiter=",")

 # Separate the data from the target attribute

X = dataset[:,0:8]

y = dataset[:,8]

#Type the following in console

del url, dataset, X, y

clear

---------- **XXX** ----------

Exercise 3: Load from File

import pandas as pd

dataset = pd.read_csv('D:/Dataset/cars.csv', delimiter=";")

#Type the following in console

del dataset

---------- **XXX** ----------

Exercise 4: Data Preprocessing

Remember, this dataset is going to be used by a machine learning algorithm that learns from past loans and predict which loans will be paid off and which won't.

a. Initial Step:

import pandas as pd

from sklearn import preprocessing

import matplotlib.pyplot as plt

import seaborn as sns

 # Load Dataset from File

loans_2007 = pd.read_csv('D:/Dataset/raw-lending-club-loans.csv', low_memory=False)

loans_2007 = pd.read_csv('D:/Dataset/lending-club-loans.csv', low_memory=False)

 # To define the Datatype of each column while loading the file: pd.read_csv(sio, dtype={"user_id": int, "username": object})

 # Type the following line in console

loans_2007.loc[2,'id']

 # To Check whether duplicate rows exist or not, type the following in console

any(loans_2007.duplicated())

 # Create a duplicate DataFrame and concat along the rows

loans_2007_repeated = pd.concat([loans_2007]*2, ignore_index=True)

 # Type the following in console

any(loans_2007_repeated.duplicated())

 # Drop duplicate rows

drop_dup=loans_2007_repeated.drop_duplicates(subset=['id'], keep='first')

 # Type the following line in console

del drop_dup, loans_2007_repeated

 # To see the attribute names

loans_2007.columns.tolist()

 # To find the number of NULL values in each Row and Column, type the following in console

loans_2007.isnull().sum(axis = 1) # Returns the no. of NULL values in each Row

 # Returns the no. of NULL values in each column

loans_2007.isnull().sum(axis = 0)

 # Check and drop columns with more than half NULL values

half_count = len(loans_2007) / 2

loans_2007 = loans_2007.dropna(thresh=half_count,axis=1) # Drop any column with more than 50% missing values

 # Load the dataset description file

data_dictionary = pd.read_csv('D:/Dataset/LCDataDictionary.csv') # Loading in the data dictionary

 # To rename any column

data_dictionary = data_dictionary.rename(columns={'LoanStatNew': 'name'})

 # Find data types of all columns

loans_2007.dtypes

(or)

loans_2007.info

 # Find data types of particular column

loans_2007['member_id'].dtypes

 # Change data type of a particular column

loans_2007['id'].dtypes = loans_2007['id'].astype('int')

 # Merge both data_dictionary and loans_2007 DataFrames

loans_2007_dtypes = pd.DataFrame(loans_2007.dtypes,columns=['dtypes'])

loans_2007_dtypes = loans_2007_dtypes.reset_index()

loans_2007_dtypes['name'] = loans_2007_dtypes['index'] # Copy column values

loans_2007_dtypes = loans_2007_dtypes[['name','dtypes']]

loans_2007_dtypes['first value'] = loans_2007.loc[0].values

preview = loans_2007_dtypes.merge(data_dictionary, on='name',how='left')

 # View the first 19 rows, type the following in Console

preview[:19]

 # Identify and drop redundant columns

drop_list=['url','desc','id','member_id','funded_amnt','funded_amnt_inv','int_rate','sub_grade','emp_title','issue_d','zip_code','out_prncp','out_prncp_inv','total_pymnt','total_pymnt_inv','total_rec_prncp','total_rec_int','total_rec_late_fee','recoveries','collection_recovery_fee','last_pymnt_d','last_pymnt_amnt']

loans_2007 = loans_2007.drop(drop_list,axis=1)

 # Fix a target column. To view a particular row, type the following line in console

preview[preview.name == 'loan_status']

 # To find unique values in loan_status column, type the following in console

loans_2007["loan_status"].value_counts()

```
# Fill the meaning
meaning = ["Loan has been fully paid off.", "Loan for which there is no longer a reasonable expectation of further payments.", "While the loan was paid off, the loan application today would no longer meet the credit policy and wouldn't be approved on to the marketplace.", "While the loan was charged off, the loan application today would no longer meet the credit policy and wouldn't be approved on to the marketplace.", "Loan is up to date on current payments.", "The loan is past due but still in the grace period of 15 days.", "Loan hasn't been paid in 31 to 120 days (late on the current payment).", "Loan hasn't been paid in 16 to 30 days (late on the current payment).", "Loan is defaulted on and no payment has been made for more than 121 days."]

status, count =
loans_2007["loan_status"].value_counts().index,loans_2007["loan_status"].value_counts().value
s

loan_statuses_explanation = pd.DataFrame({'Loan Status': status,'Count': count,
'Meaning':meaning})

loan_statuses_explanation

# Filter the necessary rows and Change target column values to either "0" 0r "1"

loans_2007 = loans_2007[(loans_2007["loan_status"] == "Fully Paid") |

(loans_2007["loan_status"] == "Charged Off")]

mapping_dictionary = {"loan_status":{ "Fully Paid": 1, "Charged Off": 0}}

loans_2007 = loans_2007.replace(mapping_dictionary)

# To visualize the target column outcomes

fig, axs = plt.subplots(1,2,figsize=(14,7))

sns.countplot(x='loan_status',data=loans_2007,ax=axs[0])

axs[0].set_title("Frequency of each Loan Status")

loans_2007.loan_status.value_counts().plot(x=None,y=None, kind='pie',

ax=axs[1],autopct='%1.2f%%')

axs[1].set_title("Percentage of each Loan status")

plt.show()

# Remove columns having only one value throughout

loans_2007 = loans_2007.loc[:,loans_2007.apply(pd.Series.nunique) != 1]

# Remove columns having less than 4 unique values

for col in loans_2007.columns:

    if (len(loans_2007[col].unique()) < 4):
```

```
    print(loans_2007[col].value_counts())
```

 # paymt_plan has only one True value. so, lets drop that column

```
print (loans_2007['pymnt_plan'].value_counts())
loans_2007 = loans_2007.drop('pymnt_plan', axis=1)
```

 # Save cleaned Dataset to file

```
loans_2007.to_csv('D:/Dataset/filtered_loans_2007.csv',index=False)
```

---

b. <u>Second Step:</u>

Note: Before starting this exercise, please do the following: Go to the location where the file "*filtered_loans_2007.csv*" is stored and open the file using Microsoft Excel or Spreadsheet. Delete few (two or three) values in the column "*loan_amnt*". This will create some MISSING VALUES in that column.

```
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
filtered_loans = pd.read_csv('D:/Dataset/filtered_loans_2007.csv', low_memory=False)
```

 # Find no. of NULL values in each column

```
null_counts = filtered_loans.isnull().sum(axis=0)    # Column
```

 # To print the no. of NULL in row 845 type the following line in Console

```
filtered_loans.iloc[845].isnull().sum()          # Prints no. of NULL values in row 845
```

 # To find the datatype of emp_length, type the following line in console

```
null_counts.emp_length.dtype
```

 # Type the following in console and check for NULL values in loan_amnt column

```
print(filtered_loans.isnull().sum(axis=0))
```

 # Replace NULL values in loan_amnt" with 0

```
filtered_loans['loan_amnt'].fillna(0, inplace=True)
print("After replacing null")
print(filtered_loans.isnull().sum(axis=0))
```

 # Find the various datatypes in DataFrame by typing the following line in console

```
filtered_loans.dtypes.value_counts()
```

 # Find the columns with datatype OBJECT and print them

```
object_columns_df = filtered_loans.select_dtypes(include=['object'])
print(object_columns_df.iloc[0])
```

 # Remove months from term column

```
filtered_loans['term'] = filtered_loans['term'].str.rstrip(' months').astype('float')
```

 # Find unique values in each column

```
cols = ['home_ownership', 'grade','verification_status', 'emp_length', 'term', 'addr_state']
for name in cols:
    print(name,':')
    print(object_columns_df[name].value_counts(),'\n')
filtered_loans['loan_amnt'].fillna(0, inplace=True)
```

 # Replace categorical values

```
mapping_dict = {"emp_length": {"10+ years": 10,"9 years": 9,"8 years": 8,"7 years": 7,"6 years": 6, "5 years": 5,"4 years": 4,"3 years": 3,"2 years": 2,"1 year": 1,"< 1 year": 0,"n/a": 0},"grade":{"A": 1,"B": 2,"C": 3,"D": 4,"E": 5,"F": 6,"G": 7}}
filtered_loans = filtered_loans.replace(mapping_dict)
filtered_loans[['emp_length','grade']].head()
```

 # Identify Minimum and Maximum Values of Columns

```
object_columns_df['term'].min()
loans_2007[['term', 'loan_amnt']].min()
object_columns_df['term'].max()
```

---------- **XXX** ----------