Data Science and Analytics (IT3080)

B.Sc (Hons) in IT (IT Specialization)

Year 3

Lab Sheet 02

Data Preprocessing and Feature Engineering

Real world data are generally incomplete, noisy, inconsistent and sometimes contains irrelevant information. To deal-with such issues, pre-processing of data expected to be analyzed is necessary. Data preprocessing consists of following tasks:

Data cleaning: fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.

Data integration: using multiple databases, data cubes, or files.

Data transformation: normalization and aggregation.

Data reduction: reducing the volume but producing the same or similar analytical results.

This lab sheet is mainly focused on cleaning raw data.

Activity 1

Download Loan.csv and LDataDictionary.xls from Courseweb. Spend few minutes to explore the data available in the Loan.csv. To understand the contents of the file, refer to the LDataDictionary.xls.

The file Loan.csv file consists of 145 columns. What type of issues do you find in the file that need addressing before proceeding to analyzing of data.

Open Jupyter Notebook and create a new Notebook. Name the Notebook as data_cleaning.

You need the following libraries for this activity

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

pandas library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming. matplotlib library provide support for generating plots and graphs in an easy manner.

Let's load data in the Loan.csv to the python program. You can use the following command to open the file.

```
raw_data=pd.read_csv("loan.csv")
```

Note that raw_data is a DataFrame object. Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Read_csv function in Pandas library is used to load the data. Use the following command to display the first five rows in the DataFrame.

```
raw_data.head()
```

Exploring the loan data

Try the following command to display the shape of the DataFrame. The shape function will display the number of rows and columns in the DataFrame.

```
raw_data.shape
```

You may also need to explore the data types of the columns which could be done using dtypes

```
raw_data.dtypes
```

If you want to explore a particular column in the data frame it could be done using the following.

```
raw_data['loan_amnt'].describe()
```

Note that this provides you descriptive statistics with relevance to the column.

Removing irrelevant columns

Following columns in the data set are irrelevant for our data analysis : zip_code, policy_code, application_type, last_pymnt_d, last_credit_pull_d, verification_status, pymnt_plan, funded_amnt_inv, sub_grade, out_prncp, out_prncp_inv, total_pymnt_inv, total_pymnt, total_pymnt_inv, total_rec_prncp, total_rec_int, total_rec_late_fee, recoveries, collection_recovery_fee, last_pymnt_amnt & initial_list_status

Therefore, they could be removed from the DataFrame. DataFrame.drop command in the pandas library could be used to remove the unwanted columns from the DataFrame. The list of columns should be inputted as parameters to the method along with the axis along which the data is to be deleted. In this case, we want to delete columns. Therefore, axis is 1. Another parameter to be used with drop is inplace. If, inplace=True nothing is returned after the

application of drop method. If inplace=Falses a copy of the object with the operation performed is returned. The default for inPlace is False.

Use the following statement to remove the columns which are not needed.

raw_data= raw_data.drop(['zip_code', 'policy_code', 'application_type', 'last_credit_pull_d', 'verification_status', 'pymnt_plan', 'funded_amnt_inv', 'sub_grade', 'out_prncp', 'out_prncp_inv', 'total_pymnt_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', 'initial_list_status'], axis =1)

Dealing with missing values

Having large number of missing values is going to adversely effect data analysis. In this example we will delete columns having more than 20 % of missing values. We will also remove empty rows if any.

View the content of the DataFrame by typing the name of the DataFrame on the Jupiter Notebook. Note that missing values are seen as 'NaN'. None (or float.NaN) is python's null datatype. NaN stands for 'Not a number'.

pandas.isnull method could be used to identify null or missing values. The method returns an array indicating whether each value is null or missing. The values for which isnull is true could be then counted using the sum method.

```
col_num=0
TotalObjects = raw_data.shape[0]
print ("Column\t\t\t\t\t Null Values%")
for x in raw_data:
   nullCount = raw_data[x].isnull().sum();
   nullPercent = nullCount*100 / (TotalObjects)
   if nullCount > 0 and nullPercent > 20 :
      col_num=col_num+1
      raw_data.drop(x, axis=1,inplace=True)
      print(str(x)+"\t\t\t\t\t "+str(nullPercent))
print ("A total of "+str(col_num)+" deleted !")
```

Use the shape and head methods to explore the resulted data.

Now, write a code segment to view remaining columns with missing data. We will replace these missing values rather than deleting the columns.

DataFrame.fillna could be used be used to fill the NaN values indicating missing values in a column. Most common approaches are replacing missing values with constants or mean for that column. Consider the following code segment :

```
raw_data['emp_title'].fillna('Unknown',inplace = True)
raw_data['dti'].fillna(0,inplace=True)
raw_data['revol_util'].fillna(raw_data['revol_util'].mean(),inplace = True)
```

Now do the following :

- Set the missing values of the next_pymnt_d column assuming that these are missing because payment is already completed.
- Set the missing values of columns mths_since_rcnt_il,mo_sin_old_il_acct, bc_open_to_buy,num_tl_120dpd_2m,bc_util and percent_bc_gt_75 to 0
- Set the missing values of columns all_util and avg_cur_bal with the mean value of the column

Formatting data

Emp_length column of the DataFrame contains information on how long an employee has been employed. Observe the content of the column. Unique function in numpy would be helpful here.

Try the code segment below :

```
pd.unique(raw_data['emp_length'].values)
```

It returns the unique values in the dataFrame as an array. Note that the content of the column includes text as well. For calculations we need the emp_length in numerical format rather than in a textual format. Therefore, let's format the data in the column in a way that the emp_length contains only numeric values.

Start by replacing the NaN values in the column with 0s. Then, use the following function in python to replace years with numeric values. The rstrip() method returns a copy of the string with trailing characters removed (based on the string argument passed)

```
def CalculateEmployeeLength(year):
        if year == '< 1 year':
                return 0.5
         elif year == '10+ years':
                 return 10
        else:
                yr=str(year)
                return yr.rstrip(' years')
```

Now apply the created function on emp_length column to convert the years in to numeric format.

```
raw_data['emp_length']=raw_data['emp_length'].apply(CalculateEmployeeLength)
```

Visualizing data

Sometimes rather than looking at values of a certain column, looking at summarized view of data in forms of plots and charts would be helpful to get insights of data before analysis.

Suppose we want to look at a summarized view on loan_amnt column. This could be done using a pie char or a bar chart.

Type the following function on JupyterNotebook. The function takes a loan value and returns the range the loan amount is.

```
def CalculateLoanRanges(value):
   if value <= 5000:
    return '5K and Below'
   if value > 5000 and value <= 10000:
    return '5K-10K'
   if value > 10000 and value <= 15000:
    return '10K-15K'
   if value > 15000 and value <= 20000:
    return '15K-20K'
   if value > 20000 and value <= 25000:
    return '20K-25K'
   if value > 25000 and value <= 30000:
    return '25K-30K'
   if value > 30000 :
    return '30K and Above'
   return 'Other'
```

Lets call the function above on the values of loan_amnt column. This could be done using the apply function. In this instance it the function returns the list of resulted values.

```
loan_ranges = raw_data['loan_amnt'].apply(CalculateLoanRanges)
```

Now let's count how many loan amounts are there in each range using the following statement.

```
loan_ranges.value_counts()
```

Use the code segment below to create a pie chart from the data above. In python matplotlib.pyplot.figure function is used to create a figure. Plot.pie will draw a pie chart using the given values. The autopct parameter is used to create labels with percentages inside the wedges of the pie char. Finally plt.title will set a title for the pie chart.

```
f = plt.figure()
loan_ranges.value_counts().plot.pie(autopct='%1.0f%%',)
plt.title('Pie Chart of Loan Amount')
```

Similar to the above, lets draw a bar chart using matplotlib for the purposes loans are applied for using the following code segment.

```
pur = raw_data['purpose'].value_counts()
pur.plot(kind='bar')
```

Finally save the cleaned data in a new csv file for the use of data analysis using the following statement.

```
raw_data.to_csv('cleaned_loans2007.csv', index=False,encoding='utf-8')
```

Activity 2

How to Detect and Remove Outliers

Step-1: Importing Necessary Dependencies and load the dataset

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

df_boston = pd.read_csv('Airdata.csv')
```

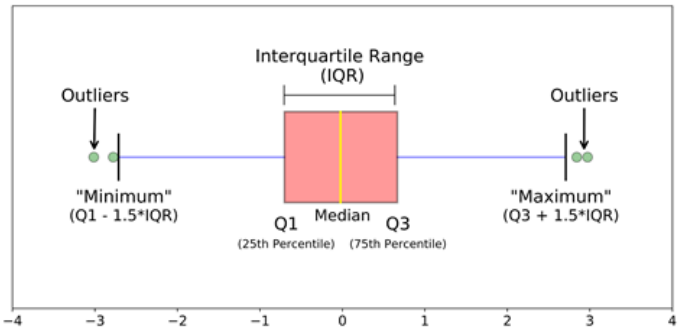
Step-2: Show a preview of the dataset

```
df_boston.head()
```

Step-3: Generate a Box-plot for the NO2_Location_A feature

```
sns.boxplot(df_boston['NO2_Location_A'])
plt.show()
```

Step-4: Finding the IQR




```
percentile25 = df_boston['NO2_Location_A'].quantile(0.25)
percentile75 = df_boston['NO2_Location_A'].quantile(0.75)

iqr = percentile75-percentile25
```

Step-5: Finding upper and lower limit

```
upper_limit = percentile75 + (1.5 * iqr)
lower_limit = percentile25 - (1.5 * iqr)
```

Step-6: Capping

```
new_df_cap = df_boston.copy()
new_df_cap['NO2_Location_A'] = np.where(
   new_df_cap['NO2_Location_A'] >= upper_limit, upper_limit,
   np.where(new_df_cap['NO2_Location_A'] <= lower_limit, lower_limit,
      new_df_cap['NO2_Location_A']
   )
)
```

Step-7: Draw boxplot after capping

```
sns.boxplot(new_df_cap['NO2_Location_A'])
plt.show()
```

Activity 3

Categorical Feature Selection using Chi squared Test

Here in chi squared test we decide whether a feature is correlated with target variable or not using p-value.

H0 :- There is no relationship between categorical feature and target variable

H1 :- There is some relationship between categorical feature and target variable

If **p-value ≥0.05**, failed to reject null hypothesis there is no any relationship between target variable and categorical features.

if **p_value <0.05**, Rejects null hypothesis and there will be some relationship between target variable and categorical variables/ features, and we will take all that features for further machine learning pipeline.

Step-1: Acquiring dataset and importing all the essential libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_selection import chi2
from sklearn.model_selection import train_test_split

df=pd.read_csv("https://raw.githubusercontent.com/srivatsan88/YouTubeLI/master/dataset/churn_data_st.csv")
```

Step-2: Show a preview of the dataset

```
df.head()
```

Step-3: Feature Encoding

a. Extract all the features which has categorical variables

```
df.dtypes
```

We will drop customerID because it will have null impact on target variable.

b. Extract all the features having categorical variable and then will do feature encoding on all of them.

```
df["gender"]=df["gender"].map({"Female":1,"Male":0})
df["Contract"]=df["Contract"].map({'Month-to-month':0, 'One year':1, 'Two year':2})
df["PaperlessBilling"]=df["PaperlessBilling"].map({"Yes":0,"No":1})

#select needed columns
cat_df=df[["gender","Contract","PaperlessBilling","Churn"]]
cat_df.head()
```

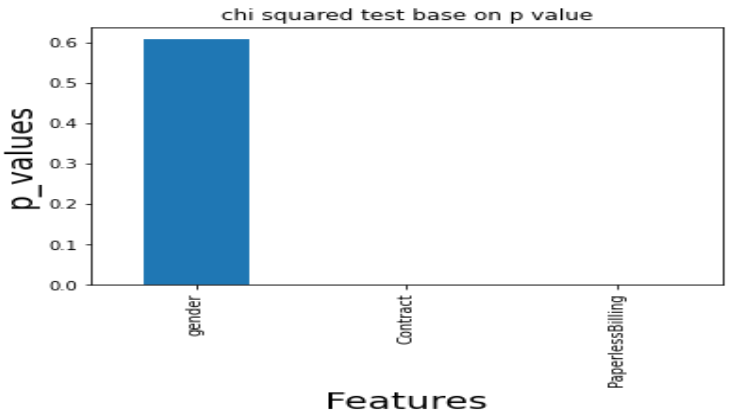
Step 4 : Applying Chi Squared test

```
#Independent variable
x=cat_df.iloc[:,:-1]
#Target variable
y=cat_df.iloc[:,-1]
#returns f score and p value
f_score=chi2(x,y)
f_score

p_value=pd.Series(f_score[1],index=x.columns)
p_value.sort_values(ascending=True)
```

Let's understand the p_value with the help of visualization.

```
p_value.plot(kind="bar")
plt.xlabel("Features",fontsize=20)
plt.ylabel("p_values",fontsize=20)
plt.title("chi squared test base on p value")
plt.show()
```



If we see above plot we can conclude that gender feature has p_value approximately equals to 0.6 means p_value >0.05 hence gender does not have significance on target variable.
So we will only select Contract and PaperlessBilling for further machine learning modeling.

Exercise

Download the Building_permits.csv from courseweb. Create a Python Notebook to clean the data in the csv file by performing following:

1. Remove columns with more than 20% of missing data
2. The columns Current Status Date, Description, Zipcode, Block, Lot, Supervisor District, Neighborhoods - Analysis Boundaries, Existing Construction Type, Existing Construction Type and Description, Number of Existing Stories and Location are irrelevant to the data analysis. Remove these columns from the dataset.

3. Replace the numeric columns Estimated Cost, Revised Cost and Number of Proposed Stories with mean of the column.
4. Replace missing values of columns Street Suffix, Existing Use, Proposed Use, Plansets, Proposed Construction Type and Proposed Construction Type Description with 'unknown'.
5. Replace missing values of Issued Date and First Construction Document Date with date of today.
6. Visualize estimated cost using a pie charts (select appropriate ranges by exploring data).
7. Visualize existing use of permits using a bar chart.