Student Name:_____Roll No:_____Section:_____

CS334 - Machine Learning

Lab 03

Instructor: Ms. Maham Ashraf
E-mail: mashraf@uit.edu

Semester: Fall, 2023

Objective

The purpose of this lab session is to introduce Analytical Base Table (ABT) for Machine Learning(ML) projects.

Instructions

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

How to Submit

- Submit lab work in a single .py file on Microsoft Teams. (No other format will be accepted)
- Lab work file name should be saved with your roll number (e.g. 19a-001-SE_LW01.py)
- Submit home work in a single .py file on Microsoft Teams. (No other format will be accepted)
- Home work file name should be saved with your roll number (e.g. 19a-001-SE_HW01.py)

1 Data Processing

1.1 Analytics Base Table (ABT)

An **Analytics Base Table (ABT)** is a simple, flat, tabular data structure made up of rows and columns. The columns are divided into a set of descriptive features and a single target feature. Each row contains a value for each descriptive feature and the target feature and represents an instance about which a prediction can be made.
Pandas-DataFrame is a Python library specially designed to load and manipulate data in ABT.

```
import pandas as pd
data = pd.read_csv("Motor Insurance Fraud Claim ABT Full .csv")
df = pd.DataFrame(data)
df.head()
```

Listing 1: Load data in data frame.

The above code load data from a csv file and place in a pandas data frame object. Now data is in ready for further processing.

1.2 Data properties

It is very important for data scientists to know about the data properties in terms of their data types and its statistical nature. The following line of code will display the data properties of the data (e.g. Motor Insurance Fraud Claim).

```
df.dtypes
```

Table 1: Data Properties

ID	int64
InsuranceType	object
IncomeofPolicyHolder	int64
MaritalStatus	object
NumClaimants	int64
InjuryType	object
OvernightHospitalStay	object
ClaimAmount	int64
TotalClaimed	int64
NumClaims	int64
NumSoftTissue	float64
%SoftTissue	float64
ClaimAmountReceived	int64
FraudFlag	int64
dtype: object	

To explore the data in terms of its statistical properties, write the following line;

```
df.describe()
```

Table 2: Statistical properties of Data

	ID	Income of Policy Holder	Num Claimants	Claim Amount	Total Claimed	Num Claims	Num Soft Tissue	% Soft Tissue	Claim Amount Received
count	500.00	500.00	500.00	500.00	500.00	500.00	490.00	500.00	500.00
mean	250.50	13739.99	1.91	16373.20	9597.19	0.80	0.23	0.17	13051.94
std	144.48	20081.54	1.01	29426.28	35655.69	2.67	0.59	0.43	30547.19
min	1.00	0.00	1.00	-99999	0.00	0.00	0.00	0.00	0.00
25%	125.75	0.00	1.00	3322.25	0.00	0.00	0.00	0.00	0.00
50%	250.50	0.00	2.00	5663.00	0.00	0.00	0.00	0.00	3253.50
75%	375.25	33918.50	3.00	12245.50	11282.75	1.00	0.00	0.00	8191.75
max	500.00	71284.00	4.00	270200	729792	56.00	5.00	2.00	295303.00

1.3 data Selection

To select specific column of the data, write the following code;

```
data2 = pd.read_csv("diabetes.csv")
df2 = pd.DataFrame(data2, columns=['Glucose',"BMI"])
df2.head()
```

Listing 2: Select specific column

only the two columns will be selected and loaded into ABT. for selecting specific rows from the table, use the following code;

```
df2.loc[df2['BMI'] >= 50]
```

In the Table-4, we have selected those rows where BMI index is greater than or equal to 50.

Table 3: Select two columns from the table

	Glucose	BMI
0	148	33.6
1	85	26.6
2	183	23.3
3	89	28.1
4	137	43.1

Table 4: Select specific rows from the table

	Glucose	BMI
120	162	53.2
125	88	55.0
155	152	50.0
177	129	67.1
193	135	52.3
247	165	52.3
303	115	52.9
445	180	59.4
673	123	57.3

2 Drop/Eliminate Data

Some time it is required to remove some data from the ABT. For removing some specific columns from the data use drop method. In the example code mentioned below, is removing *Pregnancies, SkinThickness, and DiabetesPedigreeFunction* columns from the diabetes data.

```
df2.drop(columns=['Pregnancies','SkinThickness','DiabetesPedigreeFunction'])
```

2.1 Drop Duplicate Values

Some time you want to drop duplicate values from data,

```
df = pd.DataFrame({
  'brand': ['Yum Yum', 'Yum Yum', 'Indomie', 'Indomie', 'Indomie'],
  'style': ['cup', 'cup', 'cup', 'pack', 'pack'],
  'rating': [4, 4, 3.5, 15, 5]
})
df.drop_duplicates()
```

Table 5: Drop duplicate values

	brand	style	rating
0	Yum Yum	cup	4.0
2	Indomie	cup	3.5
3	Indomie	pack	15.0
4	Indomie	pack	5.0

similarly, remove duplicate values from specific column,

```
df.drop_duplicates(subset=['brand'])
```

2.2 Drop NaN values

The dataset may contains NaN (not a number) values in cells because of missing data items. In this situation, it would be get the real insight of the data. The one solutions of this problem to remove the 'NaN' values from the data by removing the row or by placing a alternate suitable value in it. look at the following example;

Table 6: Drop duplicate values from specific column

	brand	style	rating
0	Yum Yum	cup	4.0
2	Indomie	cup	3.5

```
import numpy as np
df = pd.DataFrame({"name": ['Alfred', 'Batman', 'Catwoman'],
  "toy": [np.nan, 'Batmobile', 'Bullwhip'],
  "born": [pd.NaT, pd.Timestamp("1940-04-25"),
  pd.NaT]})
df.dropna()
```

we have dropped all rows which contained NaN/NaT values in the cells. Now remove the column which contains 'NaN' values,

df.dropna(axis='columns')

2.3 Imputation

Fill the 'NaN' values with the suitable values, suppose we replace all 'NaN' values with '0.0'.

```
df = pd.DataFrame([[np.nan, 2, np.nan, 0],
[3, 4, np.nan, 1],
[np.nan, np.nan, np.nan, np.nan],
[np.nan, 3, np.nan, 4]],
columns=list("ABCD"))
df.fillna(0)
```

We can also propagate non-null values forward or backward.

`df.fillna(method="ffill")` OR `df.fillna(method="ffill")`

Similarly, Replace all NaN elements in column 'A', 'B', 'C', and 'D', with 0, 1, 2, and 3 respectively.

```
values = {"A": 0, "B": 1, "C": 2, "D": 3}
df.fillna(value=values)
```

use scikit-learn for replace missing values with suitable method,

```
import panada as pd
from sklearn.preprocessing import Imputer

data=array([[ 1., 2., 3., 4.],
  [ 5., 6., nan, 8.],
  [ 10., 11., 12., nan]])
df = pd.Data_Frame(data)
imr = Imputer(missing_values='NaN', strategy='mean', axis=0)
imr = imr.fit(df)
imputed_data = imr.transform(df.values)
print(imputed_data)
```

3 Grouping the Data

We can group the data by different methods, such as;

```
df = pd.DataFrame({'Animal': ['Falcon', 'Falcon',
  'Parrot', 'Parrot'],
  'Max Speed': [380., 370., 24., 26.]})
df.groupby(['Animal']).mean()
```

The above example grouped the data by animal names and show the mean of fly speed of animals.

4 Covariance and correlation among data variables

To find the covariance and correlation among the data variables, pandas data frame provides two important functions `cov()` and `corr()` for this purpose. try the following example on diabetes data.

```
data2 = pd.read_csv("diabetes.csv")
df2 = pd.DataFrame(data2)
df2.cov()
df2.corr()
```

5 Counting and Encoding

You can count the total number of records in each category of feature by using `count_value` method.

```
print(df.property_type.value_counts())
```

To convert categorical values into numerical values, use the `replace()` method.

```
# convert string and categorical values into integer values
cleanup_nums = {"property_type": {"House": 1, "Flat": 2,"Upper Portion
  ": 3,"Lower Portion": 4,"Room": 5,"Farm House": 6, "Penthouse": 7},
"province_name": {"Punjab": 1, "Sindh": 2, "KPK": 3, "Balochistan": 4,"GB"
  : 5, "Islamabad Capital": 6}}

encoded_df = df.replace(cleanup_nums)
encoded_df.head(n=100)
```

5.1 Encoding class labels

The Numpy API provide a efficient method to encode categorical class variable values, such as,

```
import numpy as np
class_mapping = {label:idx for idx,label in enumerate(np.unique(df['
    classlabel'])))}
print(class_mapping)
```

6 Lab Tasks

1 Download Zameen.com property data from https://www.kaggle.com/datasets/huzzefakhan/zameencom-property-data-pakistan

2 Describe the data properties of each column,

- Data type of each column
- missing values in each column
- null values in each column
- outliers in each columns

3 Handle null values by replacing with suitable values

4 Suppose you have to predict the cost of a house, for this purpose select the appropriate coulmns that will help you to develop machine laerning model. Save the selected coulmns dataset in a separate csv file.

5 List down the descriptive variables and target variable.

6 Describe the statistics of the new data.

7 Compute the covariance and correlation matrix among descriptive variables.

8 Group the data by city, location, and area.

9 Count the total values of each item of all attributes.

10 Encode categorical values of *'property_type'* and *'province_name'* fearures with numbers.