Initial Data analysis, pre-processing, Regression and Decision tree induction in Python

LM 3.1 Objective: Statistical Functions

LM 3.2 Mean

It is calculated by taking the sum of the values and dividing with the number of values in a data series.

Syntax : mean([value1, value2, value3,….])

Example:

```
import numpy as np
from scipy import stats
dataset= [1,1,2,3,4,6,18]
#mean value
mean= np.mean(dataset)
print("Mean: ", mean)
```

LM 3.3 Median

The middle most value in a data series is called the median. The median() function is used in python using numpy to calculate this value.

Syntax : median([value1, value2, value3,….])

Example:

```
import numpy as np
from scipy import stats
dataset= [1,1,2,3,4,6,18]
#median value
median = np.median(dataset)
print("Median: ", median)
```

LM 3.4 Mode

The mode is the value that has highest number of occurrences in a set of data. Unlike mean and median, mode can have both numeric and character data. The mode() function is used in python using numpy to calculate this value.

Syntax: mode([value1, value2, value3,….])

Example:

```
import numpy as np
from scipy import stats
dataset= [1,1,2,3,4,6,18]
#mode value
mode= stats.mode(dataset)
print("Mode: ", mode)
```

LM 3.5 Standard Deviation

The standard deviation of an observation variable is the square root of its variance.

Syntax: Use **std()** function.

Example:

```
import numpy as np
from scipy import stats
dataset= [1,1,2,3,4,6,18]
std_result = np.std(dataset)
```

LM 3.6 Variance

The **variance** is a numerical measure of how the data values is dispersed around the mean.
Syntax: Use **var()** function.
Example:

```
import numpy as np
from scipy import stats
dataset= [1,1,2,3,4,6,18]
var_result = np.var(dataset)
```

LM 3.7 Five-point Summary

The five-point summary is a plot of 5 values consists Minimum, 1st Qu., Median, Mean, 3rd Qu. and Maximum.
Syntax: <No Syntax>, Custom Code.
Example:

```
# calculate a 5-number summary
from numpy import percentile
from numpy.random import rand
# generate data sample
data = rand(1000)
# calculate quartiles
quartiles = percentile(data, [25, 50, 75])
# calculate min/max
data_min, data_max = data.min(), data.max()
# print 5-number summary
print('Min: %.3f' % data_min)
print('Q1: %.3f' % quartiles[0])
print('Median: %.3f' % quartiles[1])
print('Q3: %.3f' % quartiles[2])
print('Max: %.3f' % data_max)
```

LM 4.1 Objective: Graphical representation of data

LM 4.2 Box-Plot

The box plot of an observation variable is a graphical representation based on its quartiles, as well as its smallest and largest values. It attempts to provide a visual shape of the data distribution. It is the visual representation of the statistical five number summary of a given data set.
A Five Number Summary includes:

- • Minimum
- • First Quartile
- • Median (Second Quartile)
- • Third Quartile
- • Maximum

Syntax: Use seaborn.box.plot() using seaborn library in python.
Example:

```
import seaborn as sns
sns.set(style="whitegrid")
tips = sns.load_dataset("tips")
```

```
ax = sns.boxplot(x=tips["total_bill"])
```

LM 4.3 Histogram
A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chart, but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range.
Syntax: Use **matplotlib.pyplot.hist()** function.
Example:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
x = np.random.random_integers(1, 100, 5)
plt.hist(x, bins=20)
plt.ylabel('Frequency')
plt.xlabel('Value')
plt.show()
```

LM 4.4 Heat Map
A heatmap contains values representing various shades of the same colour for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different colour can also be used.
Syntax: Use heatmap() function in seaborn library.
Dataset: Link to Flights Data
Example:

```
import numpy as np; np.random.seed(0)
import seaborn as sns; sns.set()
flights = sns.load_dataset("flights")
flights = flights.pivot("month", "year", "passengers")
ax = sns.heatmap(flights)
```

LM 4.5 Scatter Plot
Scatterplots show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.
Syntax: Use **plot.scatter()** function using pandas library.
Example:

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(50, 4), columns=['a', 'b', 'c', 'd'])
df.plot.scatter(x='a', y='b')
```

LM 4.6 Q-Q Plot
Quantile-quantile plots are a good way to compare the distribution of your data to a normal distribution (other distributions can also be used). Quantile-quantile plots are often used to test the assumptions of a data analysis (many common analysis methods assume the residuals from an analysis will be close to a normal distribution).
Syntax: Use **probplot()** function in scipy.stats library.
Example:

```
import numpy as np
import pylab
import scipy.stats as stats
measurements = np.random.normal(loc = 20, scale = 5, size=100)
stats.probplot(measurements, dist="norm", plot=pylab)
pylab.show()
```

LM 4.7 Bubble Chart

Bubble charts display data as a cluster of circles. The required data to create bubble chart needs to have the xy coordinates, size of the bubble and the colour of the bubbles. The colours can be supplied by the library itself.
Syntax: Use **plot.scatter()** function in pandas library.
Example:

```
import matplotlib.pyplot as plt
import numpy as np
# create data
x = np.random.rand(40)
y = np.random.rand(40)
z = np.random.rand(40)
colors = np.random.rand(40)
# use the scatter function
plt.scatter(x, y, s=z*1000,c=colors)
plt.show()
```

LM 5.1 Objective: Data Pre-processing – Initial Data Analysis (IDA)

LM 5.2 Data

Data is defined as facts or figures, or information that's stored in or used by a computer. Any information can be considered as data. It might be in structured format, like a matrix or unstructured format, like in txt file, video files, audio files, images etc. We are generating data every day from different platform.

- • 100 terabytes of data uploaded daily to Facebook.
- • More than 5 billion users are calling, texting, tweeting and browsing on mobile phones worldwide.
- • YouTube users watch 5.1 million videos, upload 58 hours of new video
- • every minute of the day.
- • Google delivers results for 3.6 million searches with every minute.
- • Wikipedia users publish 600 new edits with every minute.
- • 16.1 ZB of data generated in 2016.

LM 5.3 Data Pre-processing

Initial Data Analysis (IDA) also known as analysis of data or data analytics which includes the following

- • Cleansing
- • Transforming
- • Modeling data

LM 5.3.1 Need of Data Pre-processing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

LM 5.4 Initial data analysis (IDA)

Initial Data Analysis is an essential part of nearly every analysis. It is the process of data inspection steps to be carried out after the research plan and data collection have been finished but before formal statistical analyses. The purpose is to minimize the risk of incorrect or misleading results.

It is strongly recommended that its more systematic and thorough use of Data types, Forming new variables, Transforming variables, Sampling, Outliers, Binning data and Missing values.

LM 5.4.1 Special value(s) in data set – Missing Values/Wrong Inputs

There can be data record with missing values or doesn't exist or not applicable for the specific record which can mislead the data analysis. **"NA"** is treated automatically as not applicable value. For example, in a data set of patient data under health domain if some records of blood pressure is zero which is impossible or understood to be wrong input so we have to clean the same by replacing it with NA. Sometimes we have to use the mean values of the feature vector or sometimes guessing the value or it can be replaced as a global value used for replacing all missing records.

LE 5.4.1 Load the dataset

Reference Syntax:

```
# importing pandas module
import pandas as pd
# making data frame
data = pd.read_csv("<path to pima data set>")
# calling head() method
# storing in new variable
data_top = data.head()
# display
data_top
```

LE 5.4.3 Analyze the dataset and find the Headers, Dimensions (Number od Rows & Columns) and Structure of each variable, summary of the dataset

Reference Syntax

```
# calling head() method
# storing in new variable
data_top = data.head()
# display
data_top
data.info()
# check what each variables in the dataset represent
# check if variables are quantitative or quanlitative
# if quantitative, continous or discrete
# if quanlitative, whether order exists between levels
```

LE 5.4.5 Find out the min and max of the each variable. You will find weird some records with Zero (0) blood pressure or glucose or insulin. There is requirement of data cleansing with following steps.

Reference Syntax:

```
# import the pandas library
```

```
import pandas as pd
import numpy as np
#Use YOUR Data Set, for reference below it's random numbers used.
df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f','h'],columns=['one', 'two',
'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df
#Pandas provides the isnull() and notnull() functions
print df['one'].isnull()
#can replace "NaN" with "0"
print ("NaN replaced with '0':")
print df.fillna(0)
#exclude the missing values, then use the dropna() function
print df.dropna()
#Replacing NA with a scalar value is equivalent behavior of the fillna() function.
print df.replace({1000:10,2000:60})
```

LM 6.1 Objective: Supervised Learning: Linear Regression Model

LM 6.2 Introduction

Data Type and Machine Learning

- • Labeled data: Supervised Learning

➢ Regression Model

➢ Classification Model

- • Unlabeled data: Unsupervised Learning

➢ Clustering Model

- • Partially labeled and unlabeled data: Semi supervised Learning

LM 6.3 Linear Regression

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

The general mathematical equation for a linear regression is −

$y = ax + b$

Following is the description of the parameters used −

− y is the response variable.

− x is the predictor variable.

− a and b are constants which are called the coefficients.

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

LE 6.1.1 Implement the linear regression for the dataset.

Syntax:

```
#dataset
from sklearn import datasets ## imports datasets from scikit-learn
dataset = datasets.load_YOURDATASETNAME() ## loads dataset from datasets
#Clean dataset
dataset.isnull().any()
dataset = dataset.fillna(method='ffill')
#Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
#Calling Linear Regression Algorithm
regressor = LinearRegression()
regressor.fit(X_train, y_train)
#calculate coefficients
coeff_df = pd.DataFrame(regressor.coef_, X.columns, columns=['Coefficient'])
coeff_df
y_pred = regressor.predict(X_test)
#Check Accuracy
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
```

LM 9.1 Objective: Supervised Learning: Classification Model with Decision Based Tree Method

LM 9.2 Introduction

Data Type and Machine Learning

• Labeled data: Supervised Learning

➢ Regression Model

➢ Classification Model

- • Unlabeled data: Unsupervised Learning

➢ Clustering Model

- • Partially labeled and unlabeled data: Semi supervised Learning

LM 9.3 Classification

Classification involves dividing up objects so that each is assigned to one of a number of mutually exhaustive and exclusive categories known as *classes.*

Many practical decision-making tasks can be formulated as classification problems

- • customers who are likely to buy or not buy a particular product in a supermarket
- • people who are at high, medium or low risk of acquiring a certain illness
- • student projects worthy of a distinction, merit, pass or fail grade
- • objects on a radar display which correspond to vehicles, people, buildings or trees
- • the likelihood of rain the next day for a weather forecast (very likely, likely, unlikely, very unlikely).

LM 9.3.1 Classification & Prediction

There are two forms of data analysis that can be used for extracting models describing important classes or to predict future data trends. These two forms are as follows:

- • Classification o predicts categorical class labels (discrete or nominal)

- o classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
-
- • Prediction o models continuous-valued functions, i.e., predicts unknown or missing values
-

Classification models predict categorical class labels; and prediction models predict continuous valued functions. For example, we can build a classification model to categorize bank loan applications as either safe or risky, or a prediction model to predict the expenditures in dollars of potential customers on computer equipment given their income and occupation.

LM 9.3.2 Classification – a Two Step process
Classification implementation is a two-step process as mentioned below:
• Model construction: describing a set of predetermined classes o Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute.

- o The set of tuples used for model construction is training set.
- o The model is represented as classification rules, decision trees, or mathematical formulae

- **Model usage:** for classifying future or unknown objects
- Estimate accuracy of the model
  - The known label of test sample is compared with the classified result from the model
  - Accuracy rate is the percentage of test set samples that are correctly classified by the model
  - ▪ Test set is independent of training set, otherwise over-fitting will occur
  -

If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

LM 9.3.3 Classification Techniques
Following are the various techniques with which you can you can implement the classification algorithm:

- • Decision Tree based Methods
- • Rule-based Methods
- • Neural Networks
- • Naïve Bayes and Bayesian Belief Networks
- • Support Vector Machines

LM 9.4 Decision Tree Method
It is a type of supervised learning algorithm. We use it for classification problems. It works for both types input and output variables. In this technique, we split the population into two or more homogeneous sets. Moreover, it is based on most significant splitter/differentiator in input variables. The decision tree is powerful non-linear classifiers. It uses a tree structure to model the relationships among the features. Also, potential outcomes. A decision tree classifier uses a structure of branching decisions.
In classifying data, the decision tree follows the steps mentioned below:

- • It puts all training examples to a root.

- • Decision tree divides training examples based on selected attributes.
- • Then it will select attributes by using some statistical measures.
- • Recursive partitioning continues until no training example remains.

LM 9.4.2 Types of Decision Tree

- • Categorical(classification) Variable Decision Tree: Decision Tree which has categorical target variable.
- • Continuous(Regression) Variable Decision Tree: Decision Tree has continuous target variable.

LM 9.4.3 Advantages of Decision Tree

The advantages of having a decision tree are as follows:

• It does not require any domain knowledge.
• It is easy to comprehend.
• The learning and classification steps of a decision tree are simple and fast.
Less data cleaning is required.
• It handles non-linearity
• It is robust
• It scales to big data.

LM 9.4.4 Disadvantages of Decision Tree

The disadvantages of having a decision tree are as follows:

- • Overfitting: It is one of the most practical difficulties for decision tree models. By setting constraints on model parameters and pruning we can solve this problem.
- • Not fit for continuous variables: While using continuous numerical variables. Whenever it categorizes variables in different categories, the decision tree loses information.

LM 9.4.5 Decision Tree Induction Algorithm

A machine researcher named J. Ross Quinlan in 1980 developed a decision tree algorithm known as ID3 (Iterative Dichotomiser). Later, he presented C4.5, which was the successor of ID3. ID3 and C4.5 adopt a greedy approach. In this algorithm, there is no backtracking; the trees are constructed in a top-down recursive divide-and-conquer manner.

Algorithm : Generate_decision_tree

Input:

Data partition, D, which is a set of training tuples and their associated class labels.
attribute_list, the set of candidate attributes.
Attribute selection method, a procedure to determine the
splitting criterion that best partitions that the data tuples into individual classes.

Output:

A decision tree;

Method

```
create a node N;
if tuples in D are all of the same class, C then
return N as leaf node labeled with class C;
if attribute_list is empty then
return N as leaf node with labeled with majority class in D;|| majority voting
apply attribute_selection_method(D, attribute_list)
to find the best splitting_criterion;
label node N with splitting_criterion;
```

```
if splitting_attribute is discrete-valued and
multiway splits allowed then // no restricted to binary trees
attribute_list = splitting attribute; // remove splitting attribute
for each outcome j of splitting criterion
// partition the tuples and grow subtrees for each partition
let Dj be the set of data tuples in D satisfying outcome j; // a partition
if Dj is empty then
attach a leaf labeled with the majority
class in D to node N;
else
attach the node returned by Generate
decision tree(Dj, attribute list) to node N;
end for
return N;
```

LE 9.4.1 Implement a Decision Tree Classification Algorithm in python .
Syntax:

```
from sklearn.datasets import load_iris
from sklearn import tree
iris = load_iris()
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
tree.plot_tree(clf.fit(iris.data, iris.target))
```