Department of Computer Engineering
University of Peradeniya

CO 544 Machine Learning and Data Mining

Lab 03

30^{th} of April 2020

1. Objective

Getting hands-on experience on Linear models and Ensemble learning using Python.

2. Linear Regression

Linear regression is commonly used for predictive analysis. Linear regression attempts to model the relationship between two or more variables by fitting a linear equation to observed data.

(a) Simple Linear regression

Simple linear regression involves single independent variable. The line which fits best to the given data points is called the 'Regression Line'. The equation of the regression line as follows:

$$y = \beta_0 + \beta_1 x$$

Where, y - predictive variable (dependent), x - independent variable

β_0, β_1 - regression coefficients

In order to determine the regression line first we have to determine the regression coefficients. Here 'Least Square Method' can be used to determine the regression coefficients.

$$\hat{\beta}_1 = \frac{\sum(x_i y_i) - (n\bar{x}\bar{y})}{\sum(x_i)^2 - n(\bar{x})^2} \tag{1}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{2}$$

Where, $\bar{y}, \bar{x}$ - mean values of y and x variables & n - number of observations

Now we can implement this in python using Numpy as below.

```
import numpy as np  #importing numpy module
import matplotlib.pyplot as plt     #importing matplotlib modules to plot

x= np.array([1,2,3,4,5,6,7,8,9,10])     #array of independent variable values
y= np.array([2,5,7,8,9,11,14,15,17,19]) #array of dependent variable values

n= np.size(x)   #get the number of observations

m_x=np.mean(x)  #determining the mean values of variables
m_y=np.mean(y)

SS_xy = np.sum(y*x) - n*m_y*m_x     #to find b_1 estimator value
```

```
SS_xx = np.sum(x*x) - n*m_x*m_x

b_1 = SS_xy / SS_xx   #determining the parameter values
b_0 = m_y - b_1*m_x

plt.scatter(x, y,color = "b", marker = "*", s = 60)     #plotting a scatter plot
plt.title('Simple Linear Regression')   #adding a title to the graph
plt.xlabel('Independent Variable')   #adding axis labels
plt.ylabel('Dependent Variable')


y_pred = b_0 + b_1*x    #predicting response variable values
plt.plot(x, y_pred, color = "r")    #plotting the predicted line
plt.show()  #displaying the plot
```

(b) Exercise 01

i. Import the Boston_Housing.csv file.
(This is a data set from Kaggle open datasets)

ii. Split 80% of data as the training set and rest as the test set.

iii. Divide the data into feature matrix (x) and response vector (y).
Features: RM, LSTAT, PTRATIO Response: MEDV

iv. Use following expressions to estimate the parameter values:

$$\hat{\beta} = (X^\intercal X)^{-1} X^\intercal y \tag{3}$$

where,y= $\begin{pmatrix} y_1 \\ . \\ . \\ y_n \end{pmatrix}$, $\hat{\beta}$= $\begin{pmatrix} \beta_0 \\ . \\ . \\ \beta_p \end{pmatrix}$, X= $\begin{pmatrix} 1 & x_{11} & . & x_{1p} \\ . & . & . & . \\ 1 & x_{n1} & . & x_{np} \end{pmatrix}$

v. Predict the response values for the training and test sets using the derived regression model.

$$\hat{y} = X\hat{\beta}$$

vi. Visualize the residual errors for both train and test sets in one graph.
(x axis- predicted value, y-axis- actual value)

$$\text{Residual error} = y_i - \hat{y}$$

3. Logistic Regression

Logistic regression is a classification algorithm. In a classification problem, the response variable y can take only discrete values for a given set of features x.

scikit-learn is a python module with simple and efficient tools for predictive analysis. In sklearn, all machine learning models are implemented as Python classes.

It also includes some standard data sets for use. The wine data set is one such standard data set of 13 features(key name-'data') and a target variable(key name- 'target') with 3 classes.

```
#import standard data sets
from sklearn import datasets

#import the Logistic regression model
from sklearn.linear_model import LogisticRegression
```

```
#split data set into a train and test set
from sklearn.model_selection import train_test_split

#importing modules to measure classification performance
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score

wine_dataset =datasets.load_wine()  #load 'wine' data set from standard data sets
x=wine_dataset["data"]  #defining features values
y =wine_dataset["target"]   #defining target variable values

#splitting data set into a train and test set with 70% and 30%
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)

log_reg = LogisticRegression()  #creating an instance of the model
log_reg.fit(x_train,y_train)    #fitting the relationship between data

predictions = log_reg.predict(x_test)   #predict labels for test data

print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

(a) Exercise 02

i. Load the 'digits' data set from the scikit-learn standard data sets.

```
dataset =datasets.load_digits()
```

ii. Split 80% of the data set to train and rest for the test set

iii. Train a Logistic regression model and predict values for the test set.

iv. Find the accuracy and confusion matrix.

4. Ensemble Learning (Bagging)

Ensemble learning is a machine learning technique where multiple models are trained to solve the same problem and combined to get better results. The main hypothesis is that by combining models we can obtain more accurate and/or robust models.

(a) Exercise 03

In this exercise we will apply bagging technique to learn from the data set considered in Exercise 01.

i. Create 50 random samples (number of instances, n=100) by sampling with replacement (bootstrap) from the training data used in Exercise 01.

ii. Determine the linear regression models for each sample.
(Hint: One approach would be to use the method from Exercise 01)

iii. Find response variable (y) values for test data using each model developed.

iv. Make the predictions for test data, by taking the average (i.e. simple voting)

v. Visualize the residual error using a scatter plot.

5. Submission

Submit Python files with all the commands to the exercises separately (i.e. three .py files, one each per exercise) as a **zipped** folder. Name it as **e15xxxlab3**.zip where **xxx** is your registration number.