B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

Lab Record

Machine Learning

Submitted in partial fulfillment for the 6th Semester Laboratory

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

R SHREYA

1BM21CS152

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
Mar-June 2024

Lab	**Program**
1	Write a python program to import and export data using Pandas library functions
2	Demonstrate various data pre-processing techniques for a given dataset
3	Demonstrate various data pre-processing techniques for a given dataset
4	Implement Linear and Multi-Linear Regression algorithm using appropriate dataset
5	Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.
6	Build Logistic Regression Model for a given dataset and 6 Build KNN Classification model for a given dataset.
7	Build Support vector machine model for a given dataset and Build k-Means algorithm to cluster a set of data stored in a .CSV file. And Implement Dimensionality reduction using Principle Component Analysis (PCA)
8	Implement Random forest ensemble method on a given dataset And Implement Boosting ensemble method on a given dataset.
9	Build Artificial Neural Network model with back propagation on a given dataset

LAB-1

Date 21/3/24
Page

Write a python programming to import and export data using pandas library functions

Code:

```
import pandas as pd
x = pd.read_csv('C:\\Users\\Admin\\Downloads\\iris\\iris.csv')
x.columns = col_names
x.head()
```

Output:

	Sepal_length_in_cm	sepal_width_in_cm	petal length	petal width	class
0	4.9	3.0	1.4	0.2	Iris-Setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

Code:

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
col_names = ["sepal_length_in_cm", "Sepal_width_in_cm",
             "petal_length_in_cm", "petal_width_in_cm",
             "class"]

iris_data = pd.read_csv(url, names = col_names)
```

21/3/24

Output.

	Sepal length in cm	Sepal width in -cm	petal length in-cm	petal width in cm	class
0	5.1	3.5	1.4	0.2	Iris-Setosa

```
iris_data.to_csv("cleaned_iris_data.csv")
```

LAB-2

1. Get the Data

Download the Data

```
import os
import tarfile
import urllib
```

Use these libraries to download /extract web-based dataset that is housing dataset
Retrieve the data into "housing.tgz"
load the data into housing.csv

```
housing.head()
housing.info()
housing.describe()
```

→ Using these commands, inspect the attributes of the dataset

```
import matplotlib.pyplot as plt
import seaborn as sns

housing.hist (bins=50, figsize=(20,15))
plt.show()
```

→ Using matplotlib & seaborn libraries detecting the outliers

Create

LAB-3

Create Test Set

→ Splitting the dataset on test ratio = 0.2, i.e., training data is 80% of dataset and testing data is 20% of dataset

→ Stratified sampling is when random chosen data are representation of a whole target population. Each homogenous subgroup is called Strata.

Discover and Visualize the Data to gain insights

→ Visualize the data using matplotlib and seaborn libraries

→ Calculating the standard correlation coefficient of every pair of column.

Prepare the data for Machine learning algorithm

Data cleaning, handling text and categorical data, Custom Transformers, feature scaling, transformation pipelines etc, are done here

Select and Train model

→ At first Linear regression model is used to train but the model is overfitting the data.

→ To Tackle this, DecisionTreeRegressor Model is used as it is capable of finding non-linear relationships within the data.

Date
Page

But, the decisiontree model is also overfitting so badly that it performs worse than the linear regression model.

At last Randomforest regressor model is used. It is much better.

fine tune your model:

→ At last fine tuning of model is carried out, evaluating on the test set and then launch, monitor and maintaining the system

→ Evaluate your system on the test set by using mean-squared error method.

Launch, Monitor & Maintain your system

We can automate this process by
- Collecting fresh data regularly and labeling it
- Writing script to train model and fine tune the hyper parameters
- Writing script to evaluate the model.

18/4/24

LAB-4

Python Implementation of Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt

def estimate_coef(x,y):
        n = np.size(x)
        mx = np.mean(x)
        my = np.mean(y)
        ss_xy = np.sum(y*x) - n*my*mx
        ss_xx = np.sum(x*x) - n*mx*mx
        b_1 = ss_xy / ss_xx
        b_0 = my - b_1*mx
        return (b_0, b_1)

def plot_regression_line (x,y,b):
        plot.scatter (x,y, color = "m", marker = "o", s=30)
        y_pred = b[0] + b[1] *x
        plt.plot (x,y_pred, color="g")
        plt.xlabel('x')
        plt.ylabel('y')

def main()
        x = np.array([0,1,2,...9])
        y = np.array([1,3,2,...12])
        b = estimate_coef(x,y)
        print(b)
        plot_regression_line(x,y,b)
```

Output:

(b_0, b_1) = (1.2363,..., 1.16969,...)

Date
Page

Multiple linear regression

```
from sklearn.model-selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear model, metrics

data_url = " http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep = "\s+", skiprows=22
             header = None)

x = np.hstack((raw_df.values[::2, :], raw_df.values[1::2,:2]))
y = raw_df.values[1::2, 2]

x_train, x_test, y_train, y_test = train_test_split(x, y,
             test_size = 0.4, random_state = 1)

reg = linear_model.LinearRegression()
reg.fit(x train, y train)

print("Coefficients : ", reg.coef_)
print(" Variance Score : {}".format(reg.score(x_test,
                                   y_test)))

plt.style.use ('fivethirtyeight')

plt.scatter( reg.predict (x train), reg.predict (x train) -
             y_train, color = "green", s=10,
             label ="Train data")

plt.scatter( reg.predict (x_test),
             reg.predict( x_test) - y_test
             color = "blue", s=10, label= "Test data")
```

```
plt.lines (y=0, xmin=0 , xmax = 50, linewidth = 2)

plt.legend (loc= 'upper right')
plt.title ("Residual error")
plt.show()
```

LAB-5

Date ____
Page ____

Implementation of ID3

```
import numpy as np
import pandas as pd
eps = np.info(float).eps
from numpy import log2 as log

from google.colab import drive
drive.mount('/content/drive')
Path = 'drive/My Drive/ml datasets/PlayTennis.csv'
df = pd.read_csv(Path)

def find_entropy(df):
    target = df.keys()[-1]
    entropy = 0
    values = df[target].unique()

    for value in values:
        fraction = df[target].value_counts()[value]/len(df[target])
        entropy += -fraction * np.log2(fraction)
return entropy

def average_information(df, attribute)
    target = df.keys()[-1]
    target_variables = df[target].unique()
    variables = df[attribute].unique()
    entropy2 = 0
    for variable in variables:
        entropy = 0
        for target_variable in target_variables:
            num = len(df[attribute][df[attribute] == variable]
                      [df[target] == target_variable])
```

```
    den = len(df[attribute][df[attribute] == variable])
    fraction = num / (den+eps)
    entropy += - fraction * log(fraction+eps)
  fraction2 = den / len(df)
  entropy2 += - fraction2 * entropy
return abs(entropy2)

tree = buildTree(df)

import pprint
pprint.pprint(tree)
```

<u>Output</u>: {'outlook': {'overcast': 'yes',
'rainy': {'windy': {False: 'yes',
True: 'no'}},
'sunny': {'humidity': {'high': 'no',
'normal': 'yes'}}}}

09.05.24

9/5/24

LAB 6

Date
Page

Logistic Regression

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.metrics import accuracy-score
Path = 'drive/MyDrive/mldatasets/insurance data.csv'
df = pd.read_csv (Path)

from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
% matplotlib inline,

plt.scatter (data ['CET SCORE'], data ['Admitted'],
    marker = '.', color = 'purple')
x_train , x_test, y_train, y_test = train_test_split(data,
    [['CET SCORE']], data ['Admitted'] ,train_size = 0.8)

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit (x_train, y_train)
y_predicted = model.predict (x_test)
model.score ( x_test , y_test)
print (y_predicted)
print ( Xtest)

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit (x_train, y_train)
print(" coefficient (m)": model.coef_)
print(" Intercept (b):" , model.intercept_)
```

output:

KNN Implementation

```
import numpy as np
import pandas as pd

from google.colab import drive
drive.mount ('/content/drive')
dataset = pd.read_csv( '/content/drive/MyDrive/iris.csv')
dataset.head()

dataset.groupby ('species').size()

feature_columns = ['sepallength', 'sepalwidth', 'petallength',
                   'petalwidth']

x = dataset[feature_columns].values
y = dataset['species'].values

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform (y)
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y,
                   test_size = 0.2, random_state = 0)

import matplotlib.pyplot as plt
import seaborn as sns
% matplotlib inline

from sklearn.neighbours import KNeighbourClassifier
```

```
from sklearn . metrics import confusion_matrix, accuracy_score
from sklearn . model_selection import cross_val_score

classifier = KNeighborsClassifier( n_neighbors = 3)
classifier . fit (X_train, y_train)

y_pred = classifier .predict (x_test)
accuracy = accuracy_score( y_test, y_pred) * 100
print ( 'Accuracy of our model is equal' +
        str( round (accuracy, 2)) + '%')
```

Output

Accuracy of our model is equal 96.67%

23/5/24

Lab-7

SVM

```
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC

cancer = load_breast_cancer()
 X = cancer.data[:, :2]
 y = cancer.target

svm = SVC(kernel="rbf", gamma=0.5, C=1.0)
 svm.fit(x,y)

DecisionBoundaryDisplay.from_estimator(
    svm,
    x,
    response_method="predict",
    cmap = plt.cm.Spectral,
    alpha = 0.8,
    xlabel = cancer.feature_names[0],
    ylabel = cancer.feature_names[1],
)

plt.scatter(x[:,0], x[:,1],
            c=y,
            s=20, edgecolors="k")
plt.show
```

K-Means Clustering

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import sklearn . datasets import load_iris
import sklearn. cluster import KMeans

x, y = load_iris (return_X_y = True)

kmeans = KMeans (n_clusters = 3 , random_state = 2)
kmeans.fit (x)

kmeans. cluster_centers_

pred = kmeans -fit_predict (x)
pred
```

Dimensionality Reduction using PCA.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()

print (data['target_names'])
print(data['feature_names'])

df1 = pd.DataFrame(data['data'],
                   columns = data['feature_names'])

Scaling = StandardScaler()
scaling.fit(df1)

scaled_data = scaling.transform(df1)

principal = PCA(n_components = 3)
principal.fit(Scaled_data)

x = principal.transform(Scaled_data)
```

30/5/24

Lab-8

Date
Page

Implement Random forest ensemble method.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report

iris = datasets.load_iris()
iris_data = pd.DataFrame({
        'sepal length' : iris.data[:, 0],
        'sepal width' : iris.data[:, 1],
        'petal length' : iris.data[:, 2],
        'petal width' : iris.data[:, 3],
        'Species' : iris.target
    })

X = iris_data.iloc[:, :-1].values
y = iris_data.iloc[:, -1].values

x_train, x_test, y_train, y_test = train_test_split(x, y,
                    test_size = 0.33, random_state = 0)
model = RandomForestClassifier()
model.fit (x_train, y_train)
```

accuracy_score (y_test , y_pred)

Output : 0.56

Ada Boost

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

iris = load_iris()

X = iris.data
y = iris.target

x_train, x_test, y_train, y_test = train_test_split (x, y,
                      test_size = 0.4, random_state = 32)

adaboost_clf = AdaBoostClassifier (n_estimators = 30,
                learning_rate = 1.0, random_state = 42)

adaboost_clf.fit (x_train, y_train)

y_pred = adaboost_clf.predict (x_test)

Accuracy = accuracy_score (y_test , y_pred)
print ("Accuracy :", accuracy)
```

Output

Accuracy : 0.95667

20.05.24

Lab 9

Date
Page

ANN Implementation

```
import numpy as np
x = np.array(([2,9], [1,5], [3,6]) , dtype= float)
y = np.array(([92] , [86], [89]), dtype = float)
x = x / np.amax (x, axis=0)
y = y/100

epoch = 5000
lr = 0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
output_neurons = 1

wh = np.random.uniform (size = (inputlayer_neurons,
                    hiddenlayer_neurons))
bh = np.random.uniform ( size = ( 1, hiddenlayer_neurons))
wout = np.random.uniform ( size = ( hiddenlayer_neurons,
                    output_neurons))

bout = np.random.uniform (size = (1, output_neurons))

def sigmoid (x):
    return 1/(1+np.exp(-x))

def derivatives_sigmoid (x):
    return x * (1-x)

for i in range(epoch):
    hinp1 = np.dot(x, wh)
    hinp = hinp1 + bh
```

```
outinp1 = np.dot (hlayer_act, wout)
outinp  = outinp1 + bout
output  = sigmoid (outinp)

EO = y - output
outgrad = derivatives_sigmoid (output)
d_output = EO * outgrad
EH = d_output.dot (wout.T)

hiddengrad = derivatives_sigmoid (hlayer_act)
d_hiddenlayer = EH * hiddengrad

wout += hlayer_act.T.dot (d_output) * lr
wh += x.T.dot (d_hiddenlayer) * lr

print ("Input : \n" + str(X))
print("Actual Output : \n" + str(y))
print("Predicted output : \n" + output)
```

Input

```
[[0.66667  1.  ...]
 [0.33333  0.55556]
 [1. ...  , 0.66667]]
```

Actual Output	predicted output
[[0.92]	[[0.86729245]
[0.86]	[0.8451565]
[0.89]]	[0.864043]]