VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

ML Lab Report

Submitted by

PRAKHYATI BANSAL
(1BM21CS136)

Under the Guidance of

M Lakshmi Neelima
Assistant Professor
BMSCE

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

B. M. S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)

BENGALURU-560019

March 2024 to June 2024

Date :

Lab1

COMPLETE UPLOAD OF READING OF CSV FILE.

```
import pandas as pd
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)
print (iris_data.head())
```

OUTPUT-

sepal_length_in_cm	sepal_width_in_cm	...	petal_width_in_cm	class
5.1	3.5		0.2	Iris_setosa

21/3/24

Date :

Lab-2

Housing Dataset

End End Machine Learning Project

1) We frame the problem, use median & RMSE regression model.

2) Getting the data-
Download-Root holds URL of housing data
Housing-Path is used to join "data" & "01" directories.
Housing-URL has it combined.
OS mandirs for creating a directory.
Extract the files using tarfile.open()
.extractall()

3) imports required-
os for joining paths
tarfile for opening tarfiles
urllib for downloading data
matplotlib.pyplot for plotting data

4) load-dataframe-
pd.read-csv(data-path) to create a dataframe.

Date : _____

5) Create Test-data:

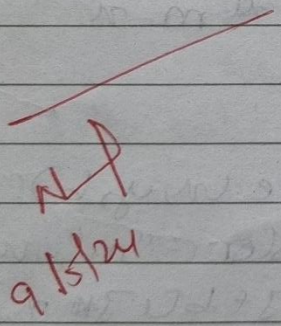
np.random.permutations() ⇒ generates random numbers.

test_size ⇒ store the % of data in test_size

indices[: test_set_size] ⇒ first set.

indices[test_set_size:] ⇒ second set

use a seed to get the same data always or store in a file.



9/5/24

Date :

Python Implementation of Linear Regression

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

def estimate_coeff(x, y):
    n = np.size(x)
    m_x = np.mean(x)
    m_y = np.mean(y)
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)

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

Date :

```
b = estimate_coef(x,y)
print(b)
plot_regression_line(x,y,b)
```

output:

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

→ Multiple Linear Regression:

```
from sklearn.model_selection import
    train_test_split
    import matplotlib.pyplot
    import numpy as np
from sklearn import datasets, linear_model,
        metrics
data_url = "http://lib.stat.cmu.edu/datasets
                    /boston"
raw_dt = 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 = rawdf.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)
```

Date :

```
print ("coefficient:", reg.coeff)
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.hlines (y=0, xmin=0, xmax=50,
            linewidth=2)
plt.legend (loc="upper right")
plt.title ("residual errors")
plt.show()
```

Date :

Decision Trees (ID3)

```
import numpy as np
import pandas as pd
df = pd.read_csv('url')
df.head()
df.info()
df.describe()
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 BuildTree(df, tree=None):
    target = df.keys()[-1]
    node = find_winner(df)
    att = np.unique(df[node])
if tree is None:
    tree = {}
    tree[node] = {}
for value in att:
    sub = get_subtable(df, node, value)
    dValue, counts = np.unique(subtable
                     [target], return_counts=True)
```

9/5/24

Date :

```
        if len(counts) == 1:
            tree[node][value] = value[0]
        else:
            tree[node][value] = buildTree(subtable)
    return tree
tree = buildTree(df)
import pprint
pprint.print(tree)
```

Decision tree (sklearn):-

```
import pandas as pd
import numpy as np
import sklearn.model_selection import
                                DecisionTree
from sklearn.tree import plot_tree.
df = pd.read_csv('url')
df.head()
df.info()
df.isnull().sum()
cols = df.columns[0:-1]
for i in cols:
    sns.boxplot(y=df[i])
    plt.show()
x = df.drop('species', axis=1)
y = df['species']
```

Date : _____

```
x_train, x_test, y_train =
        train_test_split(x, y, test_size=0.3)
dt = DecisionTreeClassifier(max_depth=3)
dt.fit(x,y)
y_pred_train = dt.predict(x_train)
y_pred = dt.predict(x_test)
accuracy = score(y_pred, y_test)
```

NL
9/5/24

Date :

Logistic Regression

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

file = "insurance.csv"
df = pd.read_csv(file)
from sklearn.linear_model import
Logistic Regression
model = Logistic Regression()
model.fit(x_train, y_train)
print(x_test)

y_predicted = model.predict(x_test)
print(y_predicted)
```

o/p: Age: 22
probability: 0.1059

Date :

KNN

```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()

df = pd.DataFrame(iris.data, columns=
                  iris.feature_names)
df['target'] = iris.target
df['flower_name'] = df.target.apply
        (lamda x: iris.target_name
                  (x))

from sklearn.model_selection import
                  train_test_split
x = df.drop(['target', 'flower_name'],
        axis = 'columns')
y = df.target

x_train, x_test, y_train, y_test =
        train_test_split(x, y, test_size=0.2)
from sklearn.neighbours import
        KNeighboursClassifier(
        KNN = KNNeighbors_Classifier(
                n_neighbours=10)
        knn.fit(x_train, y_train)
```

Date :

```
knn.score(x_test, y_test)
0.966

knn.predict([[4.8, 3.0, 1.5, 0.3]])
```

5

o/p: 0 (setosa)

10

15

20

25

Date :

K-means Implementation

```
import pandas as pd
data = pd.read_csv("iris.csv")
data.head()

import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans

x, y = load_iris(return_x_y = True)

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

pred = kMeans.fit_predict(x)
pred
```

Date :

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()
```

Date : _____

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()
data.keys()
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)
plt.figure(figsize=(10,10))
plt.scatter(x[:,0], x[:,1], c=['target'],
                                cmap='plasma')
plt.xlabel('pc1')
plt.ylabel('pc2')
```

Date :

Random Forest Ensemble & Ada Boost

```
from sklearn.datasets import make_moons
                              ada boost
from sklearn.metrics import accuracy_
                              score
from sklearn.model_selection import
                              train_test_split
from sklearn.tree import DecisionTree
                              classifier
from sklearn.ensemble import RandomForest
                              Classifier,
                              AdaBoostClassifier

X,y = make_moons(n_examples=10000,
      noise=0.5, random_state=0)
x_train, x_test, y_train, y_test =
        train_test_split
    (X,y, test_size=0.2, random_state=42)
clf = RandomForestClassifier(n_estimators=100,
    max_features="auto", random_state=0)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
accuracy_score(y_test, y_pred)
```

30/5/24

O/P: 0.7965

Date : _____

AdaBoost:

```
clf = AdaboostClassifier(n_estimators=100)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
accuracy_score(y_test, y_pred)
```

o/p: 0.8393