

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

On

MACHINE LEARNING (22CS6PCMAL)

Submitted by

SHIVANI SATHYANARAYANAN (1BM21CS203)

**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 -June 2024

**B. M. S. College of Engineering, Bull Temple Road, Bangalore 560019 (Affiliated To
Visvesvaraya Technological University, Belgaum) Department of Computer Science and
Engineering**



This is to certify that the Lab work entitled “**MACHINE LEARNING**” carried out by SHIVANI SATHYANARAYANAN (1BM21CS203), who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2023-24. The Lab report has been approved as it satisfies the academic requirements in respect of Machine Learning Lab - (22CS6PCMAL) work prescribed for the said degree.

Dr. Seema Patil
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

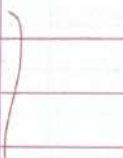
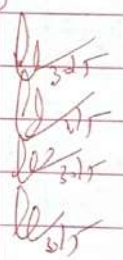
1BM21CS203

6D

INDEX

Name Shivani Sathyanarayanan Class _____

Roll No. _____ Subject _____ College _____

Sl. No.	Date	Title	Page No.	Teacher Sign / Remarks
1.	21.3	WEEK 1		 23-5-21
2.	28.3	Week 2		
3.	4.4	Week 3		
4.	18.4	Week 4		
5.	25.4	Week 5		
6.	29.5	Week 6		 23-5-21
7.	23.5	Week 7		
8.	23.5	Week 8		
9.	23.5	Program 9		
10.	30.5.24	Program 10		
11.	30.5.24	Program 11		

LAB-1

Importing and exporting a dataset

```
import pandas as pd
url = "https://..."
df = pd.read_csv(url, columns=['sepal-length',
                                'sepal-width', 'petal-length',
                                'petal-width', 'class'])
df.head()
```

	sepal-length	sepal-width	petal-length	petal-width
0	5.1	3.3	1.4	0.2
1	4.9	3.0	1.4	0.2

```
iris-data.to_csv('cleaned-iris-data.csv')
```

LAB - 2

1. Look at the big picture: Performance Measure

2. Get the data

```

fetch_housing_data()
import pandas as pd
def load_housing_data(housing_path = HOUSING_PATH):
    data_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(data_path)
housing = load_housing_data()
housing.head()
housing.info()
housing.describe()
import matplotlib.pyplot as plt
import seaborn as sns
housing.hist(bins=50, figsize=(20,15))
plt.show()
pd.cut
housing['income_cat'] = pd.cut(x=housing['median-
income'], bins=[0, 1.5, 3, 4.5, 6, np.inf],
labels=[1, 2, 3, 4, 5])
housing['income_cat'].hist()

```

3. Discover and visualize the data to gain insights

stat_train_set.shape, stat_test_set.shape
Visualizing

housing.plot

plt.show()

Correlation

corr_matrix

corr_matrix

sort_val

from pandas

attributes

scatter

plt.show

housing

4. Prepare the

from it

input

housing

input

input

housing

from it

ordinal

ordinal

alt


```
housing.plot(kind='scatter', x='longitude',
              y='latitude', alpha=0.1)
plt.show()
```

Correlation

```
corr_matrix = housing.corr()
corr_matrix['median-house-value']
sort_values(ascending=False)
from pandas.plotting import scatter_matrix
attributes = ['median-house-value',
              'median-income']
scatter_matrix(frame=housing[attributes],
               figsize=(12, 8))
plt.show()
housing['rooms-per-household'] =
    housing['total-rooms'] / housing['household']
```

4. Prepare the data for machine learning algorithms

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')
housing_num = housing.drop(["ocean-proximity",
                             axis=1])
imputer.fit(housing_num)
imputer.statistics_
housing_num.median().values
```

```
from sklearn.preprocessing import OrdinalEncoder
ordinal_encoder = OrdinalEncoder()
ordinal_encoder.categories_
a1tr.added = CombinedAttributesAdder(add_bedroom,
                                     purroom=False)
```


Transform
g. values)
non Transform
ns. to list ()

Linear Regression

y = housing -
(labels)

x = housing -
(prepared)

housing - labels
- predictions

housing - (labels)

1) housing

id, y =
housing - labels

1) housing

1) housing

1) housing

1) housing

1) housing

1) housing

1) housing

1) housing

1) housing

1) housing

(X = X test)

$$\text{final-rmse} = \text{np.sqrt}(\text{final-rmse})$$

final-rmse

$$\text{squared-errors} = (\text{y-test} - \text{final-predictions})^2$$

(y-test - final-predictions)

7 Launch, Monitor and Maintain your system

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

(y-test - final-predictions)

WEEK 3

SLR

```

df_sal = pd.read_csv('salary_data.csv')
df_sal.head()
df_sal.describe()
plt.title('Salary Dist plot')
sns.distplot(df_sal['salary'])
plt.show()
plt.scatter(df_sal['Years Exp'], df_sal['salary'], color='lightcoral')
plt.title('Sal v/h exp')
plt.ylabel('Salary')
plt.xlabel('Years of Exp')
plt.box(False)
plt.show()

```

```

x = df_sal.iloc[:, :1]
y = df_sal.iloc[:, 1]

```

```

x_train, x_test, y_train, y_test =
train_test_split(x, y, test_size=0.2, random_state=0)
regression = LinearRegression()
regression.fit(x_train, y_train)
plt.scatter(x_train, y_train, color='lightcoral')
plt.plot(x_train, y_train_pred, color='lightcoral')
print('Coeff: {regression.coef}')
print('Intercept: {regression.intercept}')

```

Multiple Li

```

df_sal
df_sal
df_sal
plt.title
plt.show
plt.
df_
plt
plt

```

x

y

Multiple Linear Regression

```
df_start = pd.read_csv('startups.csv')
```

```
df_start.head(6)
```

```
df_start.describe()
```

```
plt.title('Profit Dist Plot')
```

```
plt.show()
```

```
plt.scatter(df_start['spend'])
```

```
df_start['Profit'].color = 'light coral'
```

```
plt.box(False)
```

```
plt.show()
```

```
X = df_start.iloc[:, 1].values
```

```
Y = df_start.iloc[:, -1].values
```

```
regression = LinearRegression()
```

```
regressor.fit(X_train, Y_train)
```

```
Y_pred = regressor.predict(X_test)
```

```
result = np.concatenate((Y_pred.reshape(
    len(Y_pred), 1), Y_test.reshape(len(Y_test), 1)))
```

```
print(result)
```

Handwritten signature/initials

5880 - previous

9/0

WEEK 4

10.3

```

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
iris = load_iris()

```

```

X = iris.data

```

```

y = iris.target

```

```

feature_names = iris.feature_names

```

```

target_names = iris.target_names

```

```

X_train, X_test, y_train, y_test =

```

```

train_test_split(X, y, test_size=0.7,
                  random_state=42)

```

```

clf = DecisionTreeClassifier(criterion='gini')

```

```

accuracy = clf.score(X_test, y_test)

```

```

print("Accuracy", accuracy)

```

```

plt.figure(figsize=(12,8))

```

```

plt.show()

```

O/P Accuracy = 0.983

10/3-5-24
2

Logistic Reg

```

import pandas
from matplotlib
%matplotlib

```

```

df = pd.read_csv('iris.csv')
df.head()
plt.scatter(X_train, y_train)

```

from sklearn

perform

X_train, X_test, y_train, y_test =

train_test_split(X, y, test_size=0.7,

from sklearn

model = LogisticRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

model.score(X_test, y_test)

model.score(X_train, y_train)

model.score(X_test, y_test)

O/P :

Accuracy

10/3-5-24
2

WEEK 5

Logistic Regression

```

import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline

df = pd.read_csv("insurance-data.csv")
df.head()
plt.scatter(df.age, df.bought_insurance, marker='x',
            color='red')

from sklearn.model_selection import train_test_split
# Perform the split
X_train, X_test, y_train, y_test = train_test_split(
    df[['age']], df.bought_insurance,
    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.predict_proba(X_test)
model.score(X_test, y_test)

o/p :
Accuracy : 0.83259147

```

10/5/24

Analyze Through Linear Regression

model.coef - indicates value of $y = mx + b$ eq.

model.coef - coef (slope) = $[[0.14544702]]$

model.intercept - indicates value of b in $y = mx + b$

model.intercept: $[-5.44733781]$

Analyze

Analyze through sigmoid function:

import math

def sigmoid(x):
 return $1 / (1 + \text{math.exp}(-x))$

def prediction - function (age):

$z = 0.042 * \text{age} - 1.53$ # 0.04150133

0.042 and -1.53 # 2.6963 and -1.53

$y = \text{sigmoid}(z)$

return y

age = 35

prediction - function (age)

prob: 0.4880044983805889

age = 43

prediction - function (age)

prob: 0.568565299077705

from
neigh
train -
test -

knn.f
train
test - ac

23-5-24

WEEK 6

KNN Classifier

```

import numpy as np
import pandas as pd
import matplotlib as plt
plt.style.use('ggplot')

data = pd.read_csv('1/diabetes.csv')
df.head()
df.shape

X = df.drop('Outcome', axis=1).values
Y = df['Outcome'].values

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.4,
                                                    random_state=42,
                                                    stratify=Y)

from sklearn.neighbors import KNeighborsClassifier
neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

knn.fit(X_train, Y_train)
train_accuracy[i] = knn.score(X_train, Y_train)
test_accuracy[i] = knn.score(X_test, Y_test)

```

```
plt.plot(neighbors, train-accuracy, label = 'train')
plt.plot(neighbors, test-accuracy, label = 'test')
plt.show()
```

```
plt.scatter(neighbors, train-accuracy, color = 'k')
plt.show()
```

```
knn.fit(X_train, Y_train)
knn.score(X_test, Y_test)
```

o/p:

Accuracy = 0.7348

o/p:

A

WEEK 7SVM

```

from sklearn import datasets
cancer = datasets.load_breast_cancer()
# enter features & print
print(cancer.target)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, test_size=0.3,
    random_state=100)
from sklearn import svm
clf = svm.SVC(kernel='linear')
y_pred = clf.predict(X_test)
from sklearn import metrics
print(metrics.accuracy_score(y_test, y_pred))
print(metrics.precision_score(y_test, y_pred))
print(metrics.recall_score(y_test, y_pred))
import seaborn as sns
sns.scatterplot(x=cancer.data[:, 0],
                y=cancer.data[:, 1], hue=cancer.target)
plt.show()

```

o/p :

Accuracy : 0.9764

WEEK 8

```

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
iplayer_neurons = 32
hiddenlayer_neurons = 3
output_neurons = 1
wh = np.random.uniform
bh = np.random.uniform
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
    hlayer_act = sigmoid(hinp)
    output = sigmoid(hlayer_act)
    E0 = y - output
    outgrad = derivatives_sigmoid(output)
    E1 = E0 * output * (1 - output)
    hiddengrad = derivatives_sigmoid(hlayer_act)
    d_hiddenlayer = E1 * hiddengrad
    wh += x.T.dot(d_hiddenlayer * lr)

```

o/p:

C 2049

Input

[0.86667 1.]

[0.3333 0.5556]

[1. 0.666677]

desired network (OP

desired target

desired target

desired target

desired target

Actual output:

[0.92]

[0.86]

[0.89]

Predicted output

[0.8196977]

[0.7969237]

[0.81616067]

30/5/24

(jg - wntropi) lary

2/0

for cor' pntura

Prob. 9

9a) Random Forest

```
// import libraries
```

```
data = pd.read_csv('Iris.csv')
```

```
X = data.drop(columns=['species'])
```

```
y = data['species']
```

```
X_train, X_test, y_train, y_test =
```

```
train_test_split(X, y, test_size=0.2, random_state=42)
```

```
rf_classifier = RandomForestClassifier(n_estimators=100  
                                     random_state=42)
```

```
rf_classifier.fit(X_train, y_train)
```

```
y_pred = rf_classifier.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
```

```
feature_importances = rf_classifier.feature_importances_
```

```
features = X.columns
```

```
importance_df = pd.DataFrame({'Feature': features,  
                             'Importance': feature_importances})
```

```
sort_values(by='Importance'
```

```
ascending=False)
```

```
print(importance_df)
```

O/P:

Accuracy: 100.00%

9b) Adaboost

```
// import modules
```

```
data = pd.read_csv('Iris.csv')
```

```
X = data.drop(columns=['species'])
```

```
y = data['species']
```

```
X_train, X_test, y_train, y_test =
```

```
train_test_split(X, y, test_size=0.2, random_state=42)
```

```
adaboost_classifier =
```

```
AdaboostClassifier(n_estimators=100,  
                  random_state=42)
```

```
adaboost_classifier.fit(X_train, y_train)
```

```
y_pred = adaboost_classifier.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
```

```
O/P:
```

```
Accuracy: 100.00%
```


9b) Adaboost

// import necessary libraries

data = pd.read_csv('/content/iris.csv')

X = data.drop(columns=['species'])

y = data['species']

X_train, X_test, y_train, y_test =

train_test_split(X, y, test_size=0.2, random_state=42)

adaboost_df = AdaBoostClassifier(n_estimators=150, learning_rate=1)

adaboost_df.fit(X_train, y_train)

y_pred = adaboost_df.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

// print Acc

O/P:

Accuracy: 1.0

3/25/24

PROGRAM 10

```
// import necessary libraries
```

```
chunk size = 40960
```

```
Data source mapping = 'iris_flowers_dataset.csv'
```

```
shutil.rmtree
```

```
os.makedirs
```

```
os.makedirs
```

```
iris = pd.read_csv("iris.csv")
```

```
x = iris.iloc[:, [0, 1, 2, 3]].values
```

```
iris.info()
```

```
iris[0:10]
```

```
iris_outcome = pd.crosstab(index=iris["species"],  
                           columns="count")
```

```
sns.violinplot(x="species", y="petal length", data=iris)  
plt.show
```

```
sns.set_style("whitegrid")
```

```
sns.pairplot(iris, hue="species", size=3)
```

```
plt.show()
```

```
// import all nec
```

```
cancer = load_bre
```

```
df = pd.DataFrame
```

```
df.head()
```

```
cancer["target"]
```

```
scaler = Standard
```

```
scaler.fit(df)
```

```
scaled_data = sca
```

```
pta
```

```
pca = PCA(n=cor
```

```
pca.fit(scaled
```

```
x_pca = pca.to
```

```
scaled_data.s
```

```
x_pca.shape
```

```
pca.components
```

```
df_comp = pd.Da
```

```
df_comp
```

```
plt.figure(fi
```

```
sns.heatmap(d)
```

5/5/14

5/5/14

PROGRAM 11

// Import all necessary libraries

cancer = load_breast_cancer()

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

df.head()

cancer['target_names']

scaler = StandardScaler()

scaler.fit(df)

scaled_data = scaler.transform(df)

~~plt~~

pca = PCA(n_components=2)

pca.fit(scaled_data)

X_pca = pca.transform(scaled_data)

scaled_data.shape

X_pca.shape

pca.components_

df_comp = pd.DataFrame(pca.components_, columns=cancer['feature_names'])

df_comp

plt.figure(figsize=(12,6))

sns.heatmap(df_comp, cmap='plasma')

le

5/11/20