INDEX

Name: R. Kumar. Raghav. Std: Sec:

Roll No. Subject: ML lab School/College: BMSCE

School/College Tel. No. Parents Tel. No.

Sl. No.	Date	Title	Page No.	Teacher Sign / Remarks
1.	21/3/24	lab 1 — NL 21/3/2024	0-2	
2	28/3/24	ML-lab-2 exploration of Github	2-4	
		Projection		
3	18/4/24	Linear regression	4-6	
		Multiple regression	6-8	
4	25/4/24	Decision tree	8-10	
5.	9/8/24	logistic regression	10-12	
		KNN		
6	23/5/24	K-means	12-14	30.05.24
		SVM		
		PCA.		
7.	30 may	random forest ensemble method	14-16	
		Boosting ensemble method		
		Do upload everything		
		Completed everything		30.8.24

lab : 1

```
# import csv file from drive
import pandas as pd
airbnb_data = pd.read_csv("/content/cleaned_data.csv").
airbnb_data.head()
```

unnamed :	0	Sepal length in cm	Sepal_width	petal_length	petal width	class
0	0	5.1	3.5	1.4	0.2	iris_Setosa
1	1					
2	2					
3	3					
4	4					

```
# import from url
import pandas as pd
url = "https://archive.ics.uci.edu/ml/data")
# define columns
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)
iris_data.head()
```

	Sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	iris-Setosa.
1					
2					
3					
4					

```
# export the dataframe to the csv file
iris_data.to_csv("cleaned_iris_data.csv")
```

21/3/2024

lab : 1

```
# import csv file from drive
import pandas as pd
airbnb_data = pd.read_csv("/content/cleaned_data.csv")
airbnb_data.head()
```

unnamed :	0	Sepal length in cm	Sepal_width	petal_length	petal width	class
0	0	5.1	3.5	1.4	0.2	iris_Setosa
1	1					
2	2					
3	3					
4	4					

```
# import from url
import pandas as pd
url = "https://archive.ics.uci.edu/ml/data")
# define coulumns
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)
iris_data.head()
```

	Sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	8.5	1.4	0.2	iris-Setosa
1					
2					
3					
4					

```
# export the dataframe to the csv_file
iris_data.to_csv("cleaned_iris_data.csv")
```

21/3/2024

lab - 2. HANDS ON LEARNING WITH SCIKIT-LEARN-KERAS & TF.

1. Framing the problem & looking at the bigger Picture

about the data : California Census data to build model of housing prices in the state. features of the data include

- Population
- Median income
- Median housing price for each blk group in California

MEC

- The problem statement involves creating or building a pipeline so that this model output can be fed into another model that attempts to increase the ROI of the company on investing on a given district
- As most of the data is numerical in nature, except one feature, the most obvious method of choice is linear regression & the performance measure chosen to visualize or interpret the data is root mean square

$x^{(i)}$ → is a vector containing all of the input feature values excluding label

$y^{(i)}$ is the desired output of input $x^{(i)}$

x matrix → contains all feature values excluding labels & targets

- describe ignores null values
- To plot histogram, you need to use .hist() with parameters, bin & fig size
- The CRC32 function converts a variable-length string into 8-character string that is a text representation of the hex value of 32 bit Binary seq

get the data:

Download the data.

~~import os~~

import tarfile

import urllib

we have diff lib to download/extract web-based dataset that is housing dataset

retrieve the data into housing dataset retrieve that data into 'housing-lgs' and read 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 the matplotlib & seaborn libraries detecting the outliers

lab - 3

Create testset

- splitting the dataset on test ratio = 0.2 ie, training data is 20% of the dataset
- Stratified Sampling is when random chosen data are representation of a whole target population. each homogenous subgroup is called strata

Discover & validate the Data to gain insights.

- Visualize the data using matplotlib & seaborn libraries
- Calculating the standard corr co-eff of every pair of columns

Prepare the data for machine learning algorithm.

- Data cleaning, handling text & categorical data, custom Transformers, feature scaling, transformation pipelines etc are done here

Select & train model.

- at first linear regression model is used to train but the model is overfitting the data.
- To tackle this, Descicion tree Regressor model is used as it is capable of finding non-linear relationships within the data
  But the decision tree model is due 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 & then launch monitor and maintaining the system
- evaluate your system on the test set by using mean-squared error

Launch Monitor & maintain your System

we can automate the process by

- collecting fresh data regularly & labelling it
- writing script to train model & re fine tune the hyper parameters
- writing script to evaluate the model

Week - 4

Python implementation of linear regression

```
import numpy as np
import matplotlib.pyplot as plt
def estimate_coef(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", 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, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 9, 10, 12])
    b = estimate_coef(x, y)
    print("Estimate co-eff : \nb_0 = {} \nb_1 = {}"
          format(b))
```

output :

(b_0, b_1) = (1.2,363 . 1.16969 .)

multiple linear regression

```
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from sklearn import dataset, linear model, multilinear

data_url = "url"
raw_df = pd.read_csv (data_url, sep="\s+", skiprows=20, header
        = none)
    x = np.stack ([raw_df.values [::2,:], raw_df.values[1::2
    y = raw_df.values [1::2, 2]
x_train, x_test, y_train, y_test = train_test_split (x, y, test_size=
    random_state = 1)
reg = linear_model.LinearRegression ()
reg.fit (x_train, y_train)
print ("coeff =", 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=0, xmax=50, linewidth=2)
plt.legend (loc= 'upper right')
plt.title ('residual error')
plt.show ()
```

week - 5

Decision tree ID3

```
import numpy as np
impost pandas as pd
df = pd.read_csv ("w4l")
df.head ().
df.info()
df.describe ().
def find_entropy (df):
    target = df.key () [-1].
    entropy = 0
    value = df [target].unique ()
    for value in values:
        fraction = df [target].value_counts () [value] / len (df [target])
        entropy += fraction * np.log2 (fraction)
    return entropy.
def built Tree (df, tree = None):
    target = df.keys () [-1]
    node = find_minner (df)
    att = np.unique (df [node])
    if tree is None:
        tree = {}
        tree [Node] = {}
    for value in att:
        sub = get_subtable (df, node, value)
        df value, count = np.unique ( suitable [target], return,
                                    count = True )
        if len (count) = 1:
            tree [node] [value] = .ClValue [ ]
        else:
            tree [node] [value] = builtTree ( suitable
    return tree
tree = build Tree (df)
import pprint
Pprint.print (tree)
```

Decision tree (sklearn):

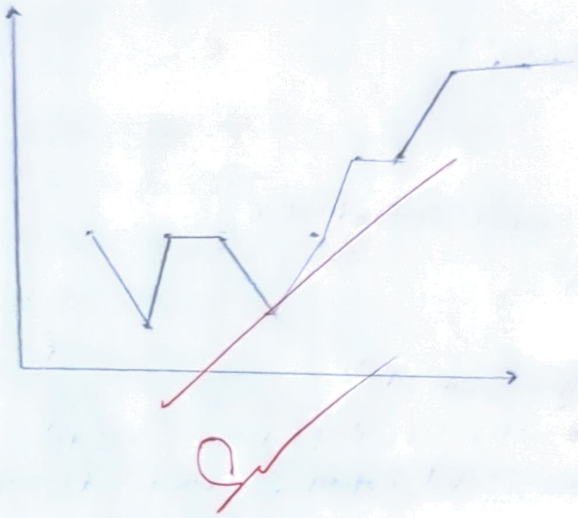
```
import pandas as pd
import numpy as np
import sklearn.model_selection import DecisionTree
from sklearn.tree import plot_tree
df = pd.read_csv('path')
df.head()
df.info()
df.isNull().sum()
cols = df.columns[0:-1]
for i in cols:
    sns.boxplot(y = df[i])
    plt.show()
x = df.drop('species', axis=1)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

09.05.24

KNN.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv('prostate.csv')
df.head()

from sklearn preprocessing import StandardScaler
Scaler = StandardScaler()

Scaler.fit(df.drop('Target', axis=1))
Scaled_features = scaler.transform(df.drop('Target', axis=1))
df_feat = pd.DataFrame(scaled_features, columns = df.columns[:-1])
df_feat.head()

from sklearn import *

x_train, x_test, y_train, y_test = train_test_split(scaled_features,
                                                    df['Target'],
                                                    test_size=0.30)
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(x_train, y_train)
pred = knn.predict(x_test)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))

error_rate = []

for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    pred_i = knn.predict(x_test)
    error_rate.append(np.mean(pred_i != y_test))

plt.figure(figsize=(10,6))
plt.plot(range(1,40), error_rate, color='blue',
    linestyle='dashed', marker='o')
```

```
plt.title ('Error Rate vs k value')
plt.xlabel ('k')
plt.ylabel ('error Rate')
plt.show ()
```



Logistic Regression

```
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_ import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

x, y = load_breast_cancer(return_x_y = True)
x_train, x_test, \ y_train, y_test = train_test_split(x, y, test_size = 0.20,
    random_state = 23)
clf = LogisticRegression (random_state =0)
clf.fit (x_train, y_train)
y_pred = clf.predict (x_test)

acc = accuracy_score (y_test, y_pred)
print ("logistic Regression model accuracy (in %):", acc*100)
digits = datasets.load_digits()
x = digits.data
y = digits.target
x_train, x_test, \ y_train, y_test = train_test_split(x, y, test_size=0.4,
    random_state = 1)
reg = linear_model.LogisticRegression ()
reg.fit ( x_train , y_train)
y_pred = reg.predict (x_test)
print ("logic Regression model accuracy (in %):",
```

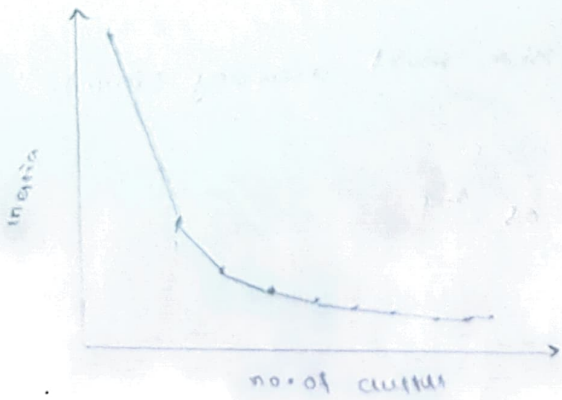
23 05 24

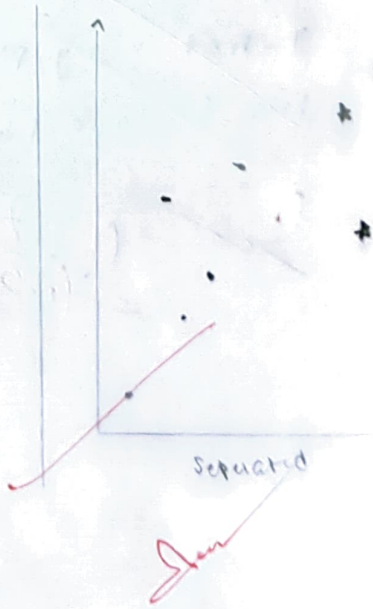
K-means clustering

```
import matplotlib.pyplot as plt

x = [4, 5, 10, 4, 3, 11, 14, 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
plt.scatter(x, y)
plt.show()
```

```
from sklearn.cluster import KMeans

data = list(zip(x, y))
inertias = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)

plt.plot(range(1, 11), inertias, marker = 'o')
plt.title('elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```

Elbow method






```
Kmeans = KMeans(n_clusters = 2)
kmeans.fit(data)
plt.scatter(x, y, c = kmeans.labels_)
plt.show()
```

How principal component analysis (PCA)

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer(as_frame = True)
df = cancer.frame
print('original DataFrame shape:', df.shape)
x = df[cancer['feature_names']]
print('input DataFrame shape :', x.shape)
x_Mean = x.mean()
x_std = x.std()
z = (x - x_mean) / x_std
c = z.cov()

import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(c)
plt.show()

# compute eigen values & Eigen vectors
eigenvector = np.linalg.eig(c)
print(values :\n', eigenvalues) )
print(values shap : ', eigenvalues.shape') )
print(Eigen vector shape: ', eigenvector.shape)

# sort eigen values in descending order
idx = eigenvalues.argsort()[::-1]
eigenvalues = eigenvalues[idx]
eigenvectors = eigenvectors[:, idx]
explained_var = np.cumsum(eigenvalues) / np.sum(eigenvalues)
explained_var

# determine no of principal components
n_comp = np.argmax(explained_var >= 0.90)+1
n_comp
```

```
# pca component as unit matrix

u = eigenvectors[:, :n_components]

pca_comp = pd.DataFrame(u,
                        index = cancer['feature_names'],
                        columns = ['PC1', 'PC2']
                        )

# plot heatmap.

plt.figure(figsize=(5,7))

sns.heatmap(pca_component)

plt.title('PCA component')

plt.show()

# find projection in PCA.

z_pca = z @ pca_component

z_pca.rename({'PC1': 'PCA1', 'PC2': 'PCA2'}, axis=1, inplace=True)

print(z_pca)
```

PCA using sklearn

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
pca.fit(z)
x_pca = pca.transform(z)
df_pca1 = pd.DataFrame(x_pca,
                       columns = ['PC{}'.
                       format(i+1)
                       for i in range(n_components)])
                        print(df_pca1)

plt.figure(figsize = (8, 6))
plt.scatter(x_pca[:, 0], x_pca[:, 1],
            c = cancer['target'],
            cmap = 'plasma')
plt.xlabel('first PCA')
plt.ylabel('second PCA')
plt.show().
pca.components_
```

SVM support vector machine

```
from sklearn.datasets import load_dataset breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC
cancer = load_dataset_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_name[0],
    ylabel = cancer.feature_name[1],
    )
plt.scatter(x[:,0], x[:,1], c= y, s=20,
edge color = 'k')
plt.show()
```

lab-8

a) implement random forest ensemble method on a given dataset

b) Implement Boosting ensemble method on given dataset

a)
```
import pandas as pd
from sklearn model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn import datasets
iris = datasets.load_iris()
import pandas as pd
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
})
data.head.model_selection import train_test_split
x = data[['sepal length', 'sepal width', 'petal length', 'petal width']]
y = data['species']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
from sklearn import metrics
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
print(confusion matrix)
```

output ['setosa' 'versicolor' 'virginica']

['sepal length' 'sepal width' 'petal length' ...

Accuracy = 0.[illegible]

cont_mat = [[10 0 0]
[0 15 1]
[0 4 15]

Implement boosting ensemble method on a dataset

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

cricket_df = pd.read_csv('/content/cricket target.csv')
x = cricket_df.drop(columns=['player Name', 'Player type'])
y = cricket_df['player Type']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size
= 0.2, random_state = 42)

boosting = GradientBoostingClassifier(n_estimators=100, learning_rate
= 0.1, random_state=42)
boosting.fit(x_train, y_train)
boosting_prediction = boosting.predict(x_test)
boosting_accuracy = accuracy_score(y_test, boosting_prediction)
print("Boosting Accuracy:", boosting_accuracy)
```

Output Boosting Accuracy: 0.35

20.05.24