MACHINE LEARNING (ALGORITHMS USING PYTHON) LAB MANUAL

1.LINEAR REGRESSION DATASET

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

d=pd.read\_csv(r'Salary\_data.csv')

x=d.iloc[:,:-1].values

y=d.iloc[:,1].values

plt.scatter(x,y)

plt.xlabel("Years of exp")

plt.ylabel("Salary")

plt.title("salary and Years of exp")

plt.show()

from sklearn.model\_selection import train\_test\_split

xtrain,xtest,ytrain,ytest=train\_test\_split(x,y,test\_size=0.1,random\_state=0)

from sklearn.linear\_model import LinearRegression

l = LinearRegression()

l.fit(xtrain, ytrain)

print(l.coef\_)

ypred=l.predict(xtest)

df=pd.DataFrame({'Actual':ytest,'Predicted':ypred})

from sklearn import metrics

print("Mean Absolute error",metrics.mean\_absolute\_error(ytest,ypred))

print("Mean Squared error",metrics.mean\_squared\_error(ytest,ypred))

print("Root Mean squared error",np.sqrt(metrics.mean\_squared\_error(ytest,ypred)))

plt.scatter(xtrain,ytrain)

plt.xlabel("Years of Experience")

plt.ylabel("salary")

plt.title("Salary data(training data)")

plt.show()

plt.scatter(xtest,ytest)

plt.xlabel("Years of Experience")

plt.ylabel("salary")

plt.title("Salary data(testing data)")

plt.show()

2.LINEAR REGRESSION DATAFRAME

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

df = pd.read\_csv('50\_Startups.csv')

df.head()

df.describe()

x = df[['R&D Spend','Administration','Marketing Spend']]

y = df['Profit']

print(x.head())

print(y.head())

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)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(x\_train, y\_train)

coeff\_df = pd.DataFrame(regressor.coef\_, x.columns, columns = ['coefficent'])

coeff\_df

y\_pred = regressor.predict(x\_test)

df = pd.DataFrame({'Actual':y\_test , 'Predicted':y\_pred})

df

from sklearn import metrics

print('Mean Absolute error :',metrics.mean\_absolute\_error(y\_test, y\_pred))

print('Mean squared error :',metrics.mean\_squared\_error(y\_test, y\_pred))

print('Root mean squared error:',np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

3.LOGISTIC REGRESSION

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

d=pd.read\_csv(r'userdata.csv')

x=d.iloc[:,[2,3]].values

y=d.iloc[:,[4]].values

from sklearn.model\_selection import train\_test\_split

xtrain,xtest,ytrain,ytest=train\_test\_split(x,y,test\_size=0.2,random\_state=0)

from sklearn.preprocessing import StandardScaler

s=StandardScaler()

xtrain=s.fit\_transform(xtrain)

xtest=s.transform(xtest)

from sklearn.linear\_model import LogisticRegression

l= LogisticRegression()

l.fit(xtrain, ytrain)

ypred=l.predict(xtest)

from sklearn.metrics import confusion\_matrix

cm=confusion\_matrix(ytest,ypred)

print(cm)

from sklearn import metrics

print("Precision",metrics.precision\_score(ytest,ypred))

print("recall",metrics.recall\_score(ytest,ypred))

print("accuracy",metrics.accuracy\_score(ytest,ypred))

4.DECISION TREE

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

d=pd.read\_csv(r'social.csv')

x=d.iloc[:,[2,3]].values

y=d.iloc[:,[4]].values

from sklearn.model\_selection import train\_test\_split

xtrain,xtest,ytrain,ytest=train\_test\_split(x,y,test\_size=0.2,random\_state=0)

from sklearn.preprocessing import StandardScaler

s=StandardScaler()

xtrain=s.fit\_transform(xtrain)

xtest=s.transform(xtest)

from sklearn.ensemble import RandomForestClassifier

c= RandomForestClassifier(n\_estimators = 10, criterion = 'entropy')

c.fit(xtrain,ytrain)

ypred=c.predict(xtest)

from sklearn import metrics

print("confusion matrix\n",metrics.confusion\_matrix(ytest,ypred))

5.kNEAREST  
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Social.csv')

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

6.NAIVE BAYES

import matplotlib.pyplot as plt

import seaborn as sns;

from sklearn.metrics import confusion\_matrix,accuracy\_score

from sklearn.datasets import fetch\_20newsgroups

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.pipeline import make\_pipeline

data=fetch\_20newsgroups()

data.target\_names

k=5

for i in range(0,len(data.target\_names),k):

categories=data.target\_names[i:i+k]

train=fetch\_20newsgroups(subset='train',categories=categories)

test=fetch\_20newsgroups(subset='test',categories=categories)

model=make\_pipeline(TfidfVectorizer(),MultinomialNB())

model.fit(train.data,train.target)

labels=model.predict(test.data)

print("The accuracy:",i,(i+5),accuracy\_score(labels,test.target))

mat=confusion\_matrix(labels,test.target)

print(mat)

sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False, xticklabels=train.target\_names,

yticklabels=train.target\_names)

plt.xlabel('True label')

plt.ylabel('Predicted label')

k=4

for i in range(0,len(data.target\_names),k):

categories=data.target\_names[i:i+k]

train=fetch\_20newsgroups(subset='train',categories=categories)

test=fetch\_20newsgroups(subset='test',categories=categories)

model=make\_pipeline(TfidfVectorizer(),MultinomialNB())

model.fit(train.data,train.target)

labels=model.predict(test.data)

print("The accuracy:",i,(i+k),format(accuracy\_score(labels,test.target)))

mat=confusion\_matrix(labels,test.target)

print(mat)

sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False, xticklabels=train.target\_names, yticklabels=train.target\_names)

plt.xlabel('True label')

plt.ylabel('Predicted label')

k=3

for i in range(0,len(data.target\_names),k):

categories=data.target\_names[i:i+k]

train=fetch\_20newsgroups(subset='train',categories=categories)

test=fetch\_20newsgroups(subset='test',categories=categories)

model=make\_pipeline(TfidfVectorizer(),MultinomialNB())

model.fit(train.data,train.target)

labels=model.predict(test.data)

print("The accuracy:",i,(i+k),format(accuracy\_score(labels,test.target)))

mat=confusion\_matrix(labels,test.target)

print(mat)

sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False, xticklabels=train.target\_names, yticklabels=train.target\_names)

plt.xlabel('True label')

plt.ylabel('Predicted label')

k=2

for i in range(0,len(data.target\_names),k):

categories=data.target\_names[i:i+k]

train=fetch\_20newsgroups(subset='train',categories=categories)

test=fetch\_20newsgroups(subset='test',categories=categories)

model=make\_pipeline(TfidfVectorizer(),MultinomialNB())

model.fit(train.data,train.target)

labels=model.predict(test.data)

print("The accuracy:",i,(i+k),format(accuracy\_score(labels,test.target)))

mat=confusion\_matrix(labels,test.target)

print(mat)

sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False, xticklabels=train.target\_names, yticklabels=train.target\_names)

plt.xlabel('True label')

plt.ylabel('Predicted label')

7.PCA (k-Means)

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('wine.csv')

X = dataset.iloc[:, 0:13].values

y = dataset.iloc[:, 13].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

from sklearn.decomposition import PCA

pca = PCA(n\_components = 2)

X\_train = pca.fit\_transform(X\_train)

X\_test = pca.transform(X\_test)

explained\_variance = pca.explained\_variance\_ratio\_

from sklearn.linear\_model import LogisticRegression

classifier\_fit(X\_train,Y\_train)

y\_pred=classifier.predict(X\_test)

from sklearn.metrics import confussion\_matrix,accuracy\_score

accuracy=accuracy\_score(y\_test,y\_pred)

print("accuracy",accuracy)

cm=confussion\_matrix(y\_test,y\_pred)

print(cm)

8.LDA (ANN)

import pandas as pd

dataset=pd.read\_csv("wine.csv")

X=dataset.iloc[:,0:13].values

y=dataset.iloc[:,13].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n\_components=1)

X\_train = lda.fit\_transform(X\_train, y\_train)

X\_test = lda.transform(X\_test)

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(max\_depth=2, random\_state=0)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

print('Accuracy' + str(accuracy\_score(y\_test, y\_pred)))

9.MLP CLASSIFIER (SVM)

import pandas as pd

dataset=pd.read\_csv("wine.csv")

X=dataset.iloc[:,0:13].values

y=dataset.iloc[:,13].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

from sklearn.neural\_network import MLPClassifier

mlp = MLPClassifier(hidden\_layer\_sizes=(10, 10, 10), max\_iter=1000)

mlp.fit(X\_train, y\_train)

predictions = mlp.predict(X\_test)

from sklearn.metrics import classification\_report, confusion\_matrix

print(confusion\_matrix(y\_test,predictions))

print(classification\_report(y\_test,predictions))