**EXP-1(FINDS Algorithm)**

import pandas as pd

import numpy as np

d=pd.read\_csv(r"D:\machine learning lab\Data.csv")

print(d)

a=np.array(d)[:,:-1]

print("The attributes are:",a)

t=np.array(d)[:,-1]

print("The target is:",t)

def fun(c,t):

for i, val in enumerate(t):

if val == "Yes":

specific\_hypothesis = c[i].copy()

break

for i, val in enumerate(c):

if t[i] == "Yes":

for x in range(len(specific\_hypothesis)):

if val[x] != specific\_hypothesis[x]:

specific\_hypothesis[x] = '?'

else:

pass

return specific\_hypothesis

print(" The final hypothesis is:",fun(a,t))

**EXP-2(Candidate key elimination)**

import numpy as np

import pandas as pd

data = pd.read\_csv(r"D:\machine learning lab\Data.csv")

concepts = np.array(data.iloc[:,0:-1])

print(concepts)

target = np.array(data.iloc[:,-1])

print(target)

def learn(concepts, target):

specific\_h = concepts[0].copy()

print("initialization of specific\_h and general\_h")

print(specific\_h)

general\_h = [["?" for i in range(len(specific\_h))] for i in range(len(specific\_h))]

print(general\_h)

for i, h in enumerate(concepts):

if target[i] == "Yes":

for x in range(len(specific\_h)):

if h[x]!= specific\_h[x]:

specific\_h[x] ='?'

general\_h[x][x] ='?'

print(specific\_h)

print(specific\_h)

if target[i] == "No":

for x in range(len(specific\_h)):

if h[x]!= specific\_h[x]:

general\_h[x][x] = specific\_h[x]

else:

general\_h[x][x] = '?'

print(" steps of Candidate Elimination Algorithm",i+1)

print(specific\_h)

print(general\_h)

indices = [i for i, val in enumerate(general\_h) if val == ['?', '?', '?', '?', '?', '?']]

for i in indices:

general\_h.remove(['?', '?', '?', '?', '?', '?'])

return specific\_h, general\_h

s\_final, g\_final = learn(concepts, target)

print("Final Specific\_h:", s\_final, sep="\n")

print("Final General\_h:", g\_final, sep="\n")

**EXP-3(Decision tree)**

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.tree import DecisionTreeClassifier

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

import pandas as pd

import numpy as np

from sklearn import tree

from sklearn.datasets import load\_iris

data =load\_iris()

df=pd.DataFrame(data.data,columns=data.feature\_names)

df['target'] =data.target

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(df[data.feature\_names],df['target'],random\_state=0)

clf=DecisionTreeClassifier(max\_depth=2,random\_state=0)

clf.fit(X\_train,Y\_train)

clf.predict(X\_test)

tree.plot\_tree(clf);

fn=['sepal length(cm)','sepal width(cm)','petal length(cm)','petal width(cm)']

cn=['setosa','versicolor','verginica']

fig,axes=plt.subplots(nrows=1,ncols=1,figsize=(4,4),dpi=300)

tree.plot\_tree(clf,feature\_names=fn,class\_names=cn,filled =True);

fig.savefig('imagename.png')

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

print(y\_pred)

from sklearn.metrics import confusion\_matrix

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

print(cm)

**Exp-4(Linear Regression)**

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

data\_set=pd.read\_csv(r"D:\machine learning lab\Salary\_Data.csv")

x=data\_set.iloc[:,:-1].values

y=data\_set.iloc[:,1].values

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=1/3,random\_state=0)

from sklearn.linear\_model import LinearRegression

regressor=LinearRegression()

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

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

x\_pred=regressor.predict(x\_train)

mtp.scatter(x\_train,y\_train,color="green")

mtp.plot(x\_train,x\_pred,color="red")

mtp.title("Salary vs experience(Training dataset)")

mtp.xlabel("Years of experience")

mtp.ylabel("salary(In Rupees)")

mtp.show()

**Logistic Regression**

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

#importing datasets

data\_set=pd.read\_csv(r"D:\machine learning lab\User\_Data.csv")

print(data\_set)

x=data\_set.iloc[:,[2,3]].values

y=data\_set.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

st\_x=StandardScaler()

x\_train=st\_x.fit\_transform(x\_train)

x\_test=st\_x.fit\_transform(x\_test)

from sklearn.linear\_model import LogisticRegression

classifier=LogisticRegression(random\_state=0)

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("Confusion matrix:\n",cm)

from sklearn.metrics import accuracy\_score

print("Accuracy:",accuracy\_score(y\_test,y\_pred))

**EXP-5(REMOVE DUPLICATES)**

import pandas as pd

data={"A":["TeamA","TeamB","TeamB","TeamC","TeamA"],"B":[50,40,40,30,50], "C":[True,False,False,False,True] }

df=pd.DataFrame(data)

print(df)

display(df.drop\_duplicates())

**CROSS VALIDATION**

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import KFold, cross\_val\_score

X, y = datasets.load\_iris(return\_X\_y=True)

clf = DecisionTreeClassifier(random\_state=42)

k\_folds = KFold(n\_splits = 5)

scores = cross\_val\_score(clf, X, y, cv = k\_folds)

print("Cross Validation Scores: ", scores)

print("Average CV Score: ", scores.mean())

print("Number of CV Scores used in Average: ", len(scores))

**BIAS,VARIANCE**

from mlxtend.evaluate import bias\_variance\_decomp

from sklearn.tree import DecisionTreeClassifier

from mlxtend.data import iris\_data

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

from sklearn.ensemble import BaggingClassifier

X, y = iris\_data()

X\_train\_ds, X\_test\_ds, y\_train\_ds, y\_test\_ds = train\_test\_split(X, y,

test\_size=0.3,random\_state=123,shuffle=True,stratify=y)

tree = DecisionTreeClassifier(random\_state=123)

bag=BaggingClassifier(base\_estimator=tree,n\_estimators=100,random\_state=123)

avg\_expected\_loss, avg\_bias, avg\_var = bias\_variance\_decomp(bag, X\_train\_ds, y\_train\_ds, X\_test\_ds, y\_test\_ds, loss='0-1\_loss',

random\_seed=123,num\_rounds=1000)

print(f'Average Expected Loss: {round(avg\_expected\_loss, 4)}n')

print(f'Average Bias: {round(avg\_bias, 4)}')

print(f'Average Variance: {round(avg\_var, 4)}')

**EXP-6(Categorical and onehot encoder)**

from numpy import asarray

from sklearn.preprocessing import OneHotEncoder

data=asarray([['red'],['green'],['blue']])

print(data)

encoder=OneHotEncoder(sparse=False)

onehot=encoder.fit\_transform(data)

print(onehot)

**Exp-8(KNN)**

import numpy as np

import matplotlib.pyplot as mtp

import pandas as pd

data\_set = pd.read\_csv(r'D:\machine learning lab\User\_Data.csv')

x=data\_set.iloc[:,[2,3]].values

y=data\_set.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

st\_x=StandardScaler()

x\_train = st\_x.fit\_transform(x\_train)

x\_test = st\_x.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)

print(y\_pred)

from sklearn.metrics import confusion\_matrix

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

print(cm)

**Exp-9(Non-Parametric)**

!pip install moepy

import numpy as np

import matplotlib.pyplot as plt

from moepy import lowess

x=np.linspace(0,5,num=150)

y=np.sin(x)+(np.random.normal(size=len(x)))/10

lowess\_model=lowess.Lowess()

lowess\_model.fit(x,y)

x\_pred=np.linspace(0,5,26)

y\_pred=lowess\_model.predict(x\_pred)

plt.plot(x\_pred,y\_pred,'--',label='Lowess',color='r',zorder=3)

plt.scatter(x,y,label='Noisy Sin Wave',color='m',s=15,zorder=1)

**EXP-10(Naïve Bayesian)**

import numpy as np

import matplotlib.pyplot as mtp

import pandas as pd

data\_set = pd.read\_csv(r'D:\machine learning lab\User\_Data.csv')

x=data\_set.iloc[:,[2,3]].values

y=data\_set.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

st\_x=StandardScaler()

x\_train = st\_x.fit\_transform(x\_train)

x\_test = st\_x.transform(x\_test)

from sklearn.naive\_bayes import GaussianNB

classifier=GaussianNB()

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

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

print(y\_pred)

from sklearn.metrics import confusion\_matrix

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

print(cm)

from sklearn.metrics import precision\_score,recall\_score

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

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

print("Precision:",precision,”\nRecall”,recall)

**EXP-12(Classification Using Pandas and matplotlib)**

import pandas as pd

import numpy as np

import matplotlib.pyplot as mtp

import seaborn as sns

df=pd.read\_csv(r'D:\machine learning lab\User\_Data.csv')

print(df)

print(df.head())

print(df.tail())

print(df.Age.describe())

print(df.info())

print(df.Gender.value\_counts())

sns.catplot(data=df,x="Purchased",y="EstimatedSalary",kind="box",aspect=1.5)

mtp.title("Boxplot for target vs protine")

mtp.show()

**EXP-14(Support Vector machine)**

import pandas as pd

from sklearn import svm

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

from sklearn.metrics import accuracy\_score

iris = pd.read\_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',names=['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'class'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(iris.drop('class', axis=1), iris['class'], test\_size=0.3, random\_state=42)

clf = svm.SVC(kernel='rbf')

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

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

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

print("Accuracy:",accuracy)

**EXP-15(Principle component analysis)**

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import seaborn as sns

from sklearn.datasets import load\_breast\_cancer

cancer = load\_breast\_cancer()

df = pd.DataFrame(cancer['data'],columns=cancer['feature\_names'])

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(df)

scaled\_data =scaler.transform(df)

from sklearn.decomposition import PCA

pca = PCA(n\_components=2)

pca.fit(scaled\_data)

PCA(n\_components=2)

x\_pca = pca.transform(scaled\_data)

x\_pca.shape

plt.figure(figsize=(8,6))

plt.scatter(x\_pca[:,0],x\_pca[:,1],c=cancer['target'],cmap='prism')

plt.xlabel('First Principle Component')

plt.ylabel('Second Principle Component')