**Machine Learning**

**1)Design the simple ML model to train the training instances and test the same**

import pandas as pd

df= pd.read\_csv('data.csv')

df.head(20)

import matplotlib.pyplot as plt

plt.scatter(df['mileage'],df['cell\_price'])

x=df[['mileage','age']] #Independent variable

y=df[['cell\_price']] #Dependent variable

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

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y,test\_size=0.30,random\_state=10)

from sklearn.linear\_model import LinearRegression

model=LinearRegression()

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

model.predict(X\_test)

X\_test

model.score(X\_test,Y\_test)

**2) For a given set of training data example stored in a.csv file implement linear regression algorithm.**

import pandas as pd

import numpy as np

import seaborn as sns

from sklearn.datasets import load\_boston

df=load\_boston()

dataset=pd.DataFrame(df.data)

dataset.columns=df.feature\_names

dataset.head(20)

x=dataset

y=df.target

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

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y,test\_size=0.30,random\_state=10)

print(X\_test)

from sklearn.linear\_model import LinearRegression

model=LinearRegression()

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

print('length of X\_train ',len(X\_train))

print('length of X\_test ',len(X\_test))

model.predict(X\_test)

model.score(X\_test,Y\_test)

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

sns.displot(y\_pred-Y\_test)

**3) .CSV file implement Logistic Regression algorithm.**

import pandas as pd

df=pd.read\_csv('iris.csv')

df.head()

df['Species'].unique()

df=df[df['Species']!='Iris-setosa']

df['Species'].unique()

df['Species']=df['Species'].map({'Iris-versicolor':0,'Iris-virginica':1})

x=df[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]

y=df['Species']

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=10)

from sklearn.linear\_model import LogisticRegression

model=LogisticRegression()

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

model.predict(X\_test)

model.score(X\_test,Y\_test)

import seaborn as plt

plt.pairplot(df,hue='Species')

**4) Build an Artificial Neural Network**

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

class Neuralnetwork:

def \_\_init\_\_(self):

self.input\_size=2

self.output\_size=1

self.hidden\_size=3

self.w1=np.random.randn(self.input\_size,self.hidden\_size)

self.w2=np.random.randn(self.hidden\_size,self.output\_size)

def Forward(self,x):

self.z=np.dot(x,self.w1)

self.z2=self.sigmoid(self.z)

self.z3=np.dot(self.z2,self.w2)

output=self.sigmoid(self.z3)

return output

def sigmoid(self,s,deriv=False):

#if(deriv==True):

#return s\*(1-s)

return 1/(1+np.exp(-s))

def backword(self,x,y,output):

self.output\_error=y-output

def train(self,x,y):

output=self.Forward(x)

self.backword(x,y,output)

NN=Neuralnetwork()

for i in range(10):

NN.train(x,y)

print('predicted output '+str(NN.Forward(x)))

print('loss '+str(np.mean(np.square(y-NN.Forward(x)))))

**5)Decision tree and confusion matrix**

import pandas as pd

from sklearn.datasets import load\_breast\_cancer

data=load\_breast\_cancer()

dataset=pd.DataFrame(data.data)

dataset.columns=data.feature\_names

dataset

x=dataset.copy()

y=data.target

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

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y,test\_size=0.33)

from sklearn.tree import DecisionTreeClassifier

clf=DecisionTreeClassifier()

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

clf.get\_params()

prediction=clf.predict(X\_test)

prediction

clf.predict\_proba(X\_test)

from sklearn.metrics import accuracy\_score

accuracy\_score(Y\_test,prediction)

from sklearn.metrics import confusion\_matrix

confusion\_matrix(Y\_test,prediction,labels=[0,1])

from sklearn.metrics import precision\_score

precision\_score(Y\_test,prediction)

from sklearn.metrics import classification\_report

print(classification\_report(Y\_test,prediction,target\_names=['Maligant','Benign']))

clf.feature\_importances\_

from sklearn import tree

from matplotlib import pyplot as plt

fig=plt.figure(figsize=(20,25))

feature\_names=x.columns

tree.plot\_tree(clf,feature\_names=feature\_names,class\_names={0:'Maligant',1:'Benign'},filled=True,

fontsize=12)

**6)Naïve\_bayes**

import pandas as pd

from sklearn.datasets import load\_iris

df=load\_iris()

dataset=pd.DataFrame(df['data'])

dataset.columns=df.feature\_names

dataset

x=dataset

y=df.target

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,random\_state=100)

from sklearn.naive\_bayes import GaussianNB

model=GaussianNB()

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

prediction=model.predict(X\_test)

print(model.score(X\_test,Y\_test))

from sklearn.metrics import accuracy\_score

accuracy\_score(prediction,Y\_test)