**#Import Data and Clean it in Pandas**

**#Data Cleaning Process**

import pandas as pd

**#Way to Include data in Local machine**

#df=pd.read\_csv('F:\Sample.csv')

#df

**#Upload Way in Google Colab**

from google.colab import files

uploaded = files.upload()

**#Way to use data in Google Colab same as in local machine**

df2 = pd.read\_csv('sample.csv')

df2

**#Remove Row data df.drop(df.index[2])**

df3=df2.drop(df2.index[0])

df3

df2

**# Add new Data**

df3=df2

row = ['Daraz', 'Hanif', 90,-1,-1]

df3.loc[df3.index[0]]=row#df3.loc[5]=row\_value

df3

**#Top heading remove**

df2 = pd.read\_csv('sample1.csv',header=1)

df2

**#Same Result as above command**

df2 = pd.read\_csv('sample1.csv',skiprows=1)

df2

**#Add header of choice**

df2 = pd.read\_csv('sample2.csv',header=0,names=['Company', 'Manager' ,'Revenue', 'Expenses', 'Profit'])

df2

**#fix no of rows**

df2 = pd.read\_csv('sample2.csv',header=0,nrows=3,names=['Company', 'Manager' ,'Revenue', 'Expenses', 'Profit'])

df2

df3

df2 = pd.read\_csv('sample1.csv')

df2

**#replace -1,80 with NAN**

df2 = pd.read\_csv('sample1.csv',na\_values=[-1,80])

df2

**#replace NAN in specific col**

df2 = pd.read\_csv('sample1.csv',na\_values={'Revenue':['80']})

df2

**#save to csv**

df2.to\_csv('ssample.csv')

df2 = pd.read\_csv('ssample.csv')

df2

**#Read from Excel**

from google.colab import files

uploaded = files.upload()

df2 = pd.read\_excel('esample.xls')

df2

**#Read from Excel workbook**

from google.colab import files

uploaded = files.upload()

df2 = pd.read\_excel('esample1.xls')

df2

**# Read specific Sheet in Excel File**

df2 = pd.read\_excel('esample1.xls','Sheet2')

df2

**# Replace specific data**

def convert\_Manager\_cell(cell):

if cell == 'NA':

return 'Rizwan'

return cell

df2 = pd.read\_excel('esample1.xls','Sheet2',converters={'Manager':convert\_Manager\_cell})

df2

**#cleaning data by fill empty place**

from google.colab import files

uploaded = files.upload()

df2 = pd.read\_excel('iris.xlsx','Sheet1')

df2

**#check data type**

df2.sepal\_length

df2.Date

**# Change index values**

df2.set\_index('Date',inplace=True)

df2

**# Fill missing values**

ndf=df2.fillna(1)

ndf

**# top to bottom forward fill Row wise**

print(df2)

ndf=df2.fillna(method='ffill')

ndf

**# bottom to top backward fill**

print(df2)

ndf=df2.fillna(method='bfill')

ndf

**# top to bottom forward fill by Column wise same like this backward fill also**

print(df2)

ndf=df2.fillna(method='ffill',axis='columns')

ndf

**# top to bottom forward fill with limit of copy**

print(df2)

ndf=df2.fillna(method='ffill',limit=1)

ndf

**# Guess based filling missing values**

print(df2)

ndf=df2.interpolate()

ndf

print(df2)

ndf=df2.interpolate(method='time')

ndf

**#Drop rows of missing values/Data**

print(df2)

ndf=df2.dropna()

ndf

**#Drop only if all missing**

print(df2)

ndf=df2.dropna(how='all')#thresh for same purpose

ndf

**# For Sheet Dimensions**

df2.shape

**#Fill missing value with mean, median, sum**

df2.fillna(df2.sum())

**# Count missing values**

df2.isnull().sum()

**#Show null in col**

df2[df2.petal\_width.isnull()]

**#Shape of data with out missing value**

df2.dropna(how='any').shape

df2.dropna()

**# For fix col shape with out missing values**

df2.dropna(subset=['sepal\_width','petal\_width'],how='any').shape

df2.dropna(subset=['sepal\_width','petal\_width'],how='any')

df2.dropna(subset=['sepal\_width','petal\_width'],how='all').shape

**#Numpy for NAN values**

import numpy as np

df2

**#for replace specific value with NAN**

df3=df2.replace(0.2,np.NaN)

df3

df3.replace(np.NaN,0.2)

**#set of values replacement**

df2.replace([0.2,1.4],np.NaN)

**#if want to replace in specific col**

df2.replace({'sepal\_width':np.NaN,'petal\_width':np.NaN},0)

**#each value with different assignment**

df2.replace({1.4:np.NaN,0.2:22})

**# Record the data in Groups**

data=df2.groupby('species')

data

**# Print elements in each group**

for species, species\_df in data:

print(species)

print(species\_df)

**#Display specific group**

data.get\_group('setosa')

**#Maximum value in each group**

print(data.max())

print(data.min())

**#Above is a rule of split apply and combine and here its detail**

data.describe()

**#Plot Data**

data.plot(x='species',y='sepal\_width')

**#Perceptron**

input\_features=np.array([[0,0],[0,1],[1,0],[1,1]])

print(input\_features.shape)

input\_features

target\_output=np.array([[0,1,1,1]])

target\_output=target\_output.reshape(4,1)

print(target\_output.shape)

target\_output

#np.random.seed(1)

#weights=2\*np.random.random((3,1))-1

weights=np.array([[0.2],[0.3]])

print(weights.shape)

weights

#syn0 = 2\*np.random.random((3,4)) - 1

#syn1 = 2\*np.random.random((4,1)) - 1

def sigmoid(x):

  return 1/(1+np.exp(-x))

def sigmoid\_der(x):

  return sigmoid(x)\*(1-sigmoid(x))

  #return x\*(1)

for epoch in range(20):

  inputs=input\_features

  in\_o=np.dot(inputs,weights) + bias

  out\_o=sigmoid(in\_o)

  #error=out\_o-target\_output

  '''outputs=sigmoid(np.dot(input\_features,weights))

  error=target\_output-outputs

  adjustments=error\*sigmoid\_der(outputs)

  weights=weights+np.dot(input\_features.T,adjustments)'''

print("Output After Training")

#print(outputs)

print(out\_o)

