**Programming**

* **Arithmetic and Variable**

 we use variables in programming to easily manipulate values

n=5

n+3

type(n)

n=10

#Print

print(n)

print ("The value of n is", n)

print (1 + 2)

s='yellow'

s.upper()

s+' submarine'

*# Set the value of a new variable to 3*

my\_var = 3

*# Print the value assigned to my\_var*

print(my\_var)

*# Change the value of the variable to 100*

my\_var = 100

*# Print the new value assigned to my\_var*

print(my\_var)

* **Functions**

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

*# Define the function*

def add\_three(input\_var):

output\_var = input\_var + 3

return output\_var

*# Run the function with 10 as input*

new\_number = add\_three(10)

*# Check that the value is 13, as expected*

print(new\_number)

def get\_pay\_with\_more\_inputs (num\_hours, hourly\_wage, tax\_bracket):

*# Pre-tax pay*

pay\_pretax = num\_hours \* hourly\_wage

*# After-tax pay*

pay\_aftertax = pay\_pretax \* (1 - tax\_bracket)

return pay\_aftertax

higher\_pay\_aftertax = get\_pay\_with\_more\_inputs(40, 24, .22)

print(higher\_pay\_aftertax)

* Python data structures

The basic Python data structures in Python include list, set, tuples, and dictionary. Each of the data structures is unique in its own way. Data structures are “containers” that organize and group data according to type

*my\_tuple = (120, 80, 100, 120, 35, 140, 120)*

*my\_tuple*

print(higher\_pay\_aftertax)

my\_tuple [1]

#List

animal\_list = ['cat', 'dog', 'lion', 'spider', 'eagle']

animal\_list

animal\_list [2]

animal\_list [2]='roma'

#Dictionary

ages = {"Ian" : 40, "Alice" : 25, "Kate" : 65}

ages

ages["Kate"]

* Pandas Library

fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

*#Series*

import pandas as pd

my\_grades = pd. Series ([89, 72, 55, 93], index= ['Math', 'English', 'French', 'Chemistry'])

my\_grades

my\_grades.index

#dataframe

data = {'studentID': ['s08549383', 's062184743', 's17758784', 's17439450'],

        'name': ['John', 'Rhona', 'Clara', 'Dave'],

        'age': [22, 20, 36, 22],

        'grade': [55, 86, 62, 38]}

physics\_class = pd.DataFrame(data)

physics\_class

physics\_class. Head ()

physics\_class. Tail ()

physics\_class. Columns

physics\_class. mean ()

physics\_class['age'].max ()

physics\_class [1:2]['name']

#Native accessors>>>>Native Python objects provide good ways of indexing data. Pandas carries all of these over,

#access the property of an object by accessing it as an attribute.

physics\_class.name

physics\_class['name']

physics\_class[1:2]['name']

physics\_class['name'][0]

# index-based selection: selecting data based on its numerical position in the data

#firstrow

physics\_class.iloc[0]

#all rows firstcolumn

physics\_class.iloc[:, 0]

#another awy all3 rows firstcolumn

physics\_class.iloc[[0, 1, 2], 0]

#- negtive like tail the 5rows from the end allcolumn

physics\_class.iloc[-5:]

#Conditional selection

physics\_class.name == 'Dave'

#label-based selection. In this paradigm, it's the data index value, not its position, which matters.

physics\_class.loc[:, ['name', 'age']]

#grouping

#groupby() created a group of which allotted the same name values to the given wines. Then, for each of these groups, we grabbed the points() column and counted how many times it appeared

physics\_class.groupby('grade').grade.count()

#result 2column grade,second column gradecount

* **visulization**

the physical or imagining creation of images, diagrams, or animations to communicate a message

import matplotlib

%Matplotlib inline

*#Histogram*

student ages = physics\_class['age']

student ages. hist(bins=2)

#Boxplot to ignore outlier

physics\_class.plot.box ()

**………………………………………………………………………………………………………………………………………………………**

**machine learning**

* The first step in any machine learning project is familiarize yourself with the data. You'll use the Pandas library for this

import pandas as pd

import numpy as np

data=pd.read\_csv("/content/drive/MyDrive/heathml/cancer.csv")

# Remove two columns name is 'C' and 'D'

data =data.drop(['id', 'Unnamed: 32'], axis = 1)

data.describe()

* **Selecting Data for Modeling**

x=data.drop(['diagnosis'], axis = 1)

y=data['diagnosis']

x.describe()

* **build model**

**use the scikit-learn library to create your models. When coding, this library is written as sklearn**

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, random\_state=42)

#trainging

from sklearn import svm

clf = svm.SVC()

clf.fit(X\_train, y\_train)

* **Model validation accuracy and evaluation**

pred=clf.predict(X\_test)

from sklearn.metrics import confusion\_matrix

confusion\_matrix(y\_test, pred)

from sklearn.metrics import classification\_report

target\_names = ['class 0', 'class 1']

print(classification\_report(y\_test, pred, target\_names=target\_names))

* **Model Deploy with vscode with anaconda**

**1**

conda activate Mlenv

conda install gunicorn

conda install pycryptodome

conda install flask

**2 html home in folder tamplets**

<html>

    <head>

        <meta charet="utf-8">

        <title>Iris</title>

    </head>

    <body  bgcolor="#8a2be2">

      <center>

    <h1> Iris Detection</h1>

    <form action="{{url\_for('predict')}}" method="post">

        <input type="text" name="sepal\_length" placeholder="enter here sepal\_length">

        <input type="text" name="sepal\_width" placeholder="enter here sepal\_width">

        <input type="text" name="petal\_length" placeholder="enter here petal\_length">

        <input type="text" name="petal\_width" placeholder="enter here petal\_width">

        <input type="submit" value="Submit">

    </form>

**3 in folder result .html**

<html>

<body>

    <center>

        <h1>

            Result:{{result}}

        </h1>

    </center>

</body>

</html>

4 model

from copyreg import pickle

from flask import Flask,render\_template

from requests import request

from sklearn.datasets import load\_iris

from sklearn.neighbors import KNeighborsClassifier

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

import pickle

irisdata=load\_iris()

model =KNeighborsClassifier(n\_neighbors=4)

xtrain,xtest,ytrain,ytest=train\_test\_split(irisdata.data,irisdata.target)

model.fit(xtrain,ytrain)

pickle.dump(model, open("model.pkl", 'wb'))

model=pickle.load(open('model.pkl','rb'))

 #not used to see on validation data

# some time later...

# load the model from disk

loaded\_model = pickle.load(open('model.pkl', 'rb'))

result3= loaded\_model.score(xtest, ytest)

print(result3)

5main that run all app >app.py

from copyreg import pickle

#from crypt import methods

from flask import Flask,render\_template

from requests import request

from sklearn.datasets import load\_iris

from sklearn.neighbors import KNeighborsClassifier

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

import numpy as np

import numpy as np

import pickle

import os

from flask import request

#1

app=Flask(\_\_name\_\_, template\_folder='templates')

loaded\_model = pickle.load(open('model.pkl', 'rb'))

@app.route("/")

def home():

#3rout cllassfiction

   # irisdata=load\_iris()

    #model =KNeighborsClassifier(n\_neighbors=4)

    #xtrain,xtest,ytrain,ytest=train\_test\_split(irisdata.data,iris.target)

    #model.fit(xtrain,ytrain)

    #pickle.dumps(model,open("iris.pkl","wb"))

    return render\_template("home.html")

#4route prediction

@app.route("/predict",methods=["get","post"])

def predict():

    sepal\_length=request.form['sepal\_length']

    sepal\_width=request.form['sepal\_width']

    petal\_length=request.form['petal\_length']

    petal\_width=request.form['petal\_width']

#take input from user form put in arry

    form\_arry=np.array([[sepal\_length,sepal\_width,petal\_length,petal\_width]])

    #model.pickle.load(open("model.pkl","rb"))

    #predict on arry of user input form

    loaded\_model = pickle.load(open('model.pkl', 'rb'))

    prediction=loaded\_model.predict(np.array(form\_arry.astype(int)))

    if  int(prediction)== 0:

        result="Iris setosa"

    elif int(prediction) == 1:

        result="Iris Veersiclor"

    else:

        result="Iris virginica"

    return render\_template("result.html",result=result)

if \_\_name\_\_ =="\_\_main\_\_":

    app.run(debug=True)