2 MONTHS TRAINING PROGRAM

ON

AN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

K2N ACADEMY

GROUP 4

GROUP NAME:- TEAM MONK

GROUP MEMBERS:-

1. ANAND KUMAR
2. SANISH KUMAR
3. MD MAKKI
4. ABHISHEK ANAND
5. MD SHAKIR HASSAN
6. ANUKUL RAJ

Acknowledgement

I would like to extend my sincere gratitude to the training program organizers and instructors for their comprehensive and well-structured course content, which has helped me develop a solid foundation in machine learning concepts and techniques. Additionally, I would like to thank my colleagues and peers for their insightful discussions and exchange of ideas, which have enriched my learning experience. Finally, I would like to thank my teacher and friends for their unwavering support and encouragement throughout this journey.

TABLE OF CONTENTS

Introduction

An artificial intelligence and machine learning training program is an educational course designed to provide learners with an in-depth understanding of the concepts and techniques of AI and machine learning. Such a program typically covers a wide range of topics, including Python, artificial intelligence, machine learning algorithms, neural networks, natural language processing, and deep learning, among others. Participants in such programs will usually acquire the skills and knowledge necessary to develop and implement machine learning solutions for various applications in industries such as healthcare, finance, manufacturing, and transportation, among others.

1. **What are AI and ML?**

AI: (Artificial Intelligence) is a branch of computer science concerned with creating machines that can perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.

There are several techniques used in AI, including machine learning, deep learning, natural language processing, and computer vision, among others. Machine learning involves the use of algorithms that enable machines to learn from data and improve their performance over time. Deep learning is a subfield of machine learning that uses neural networks to model and process complex patterns and relationships in data.

AI has applications in a wide range of industries, including healthcare, finance, transportation, and manufacturing, among others. In healthcare, AI is being used to develop diagnostic tools and personalized treatments. In finance, AI is being used for fraud detection and risk analysis. In transportation, AI is being used to develop autonomous vehicles and optimize logistics operations. And in manufacturing, AI is being used to optimize production processes and reduce costs.

ML: (Machine Learning) is a subfield of AI that involves the development of algorithms that can learn from and make predictions or decisions based on data. In other words, ML is the process of training a computer to recognize patterns in data and make decisions based on that data, without being explicitly programmed to do so.

Machine learning algorithms can be classified into three main types: supervised learning, unsupervised learning, and reinforcement learning.

Machine learning has a wide range of applications in various industries, including healthcare, finance, transportation, and manufacturing, among others. Machine learning is being used to develop predictive models for disease diagnosis and treatment, fraud detection and risk analysis in finance, route optimization and demand forecasting in transportation, and predictive maintenance and quality control in manufacturing

2. Comparative study of these two

Artificial Intelligence (AI) and Machine Learning (ML) are related but distinct fields within computer science

AI is a broader concept that encompasses several technologies and approaches, including ML. AI refers to the development of

computer systems that can perform tasks that typically require human intelligence, such as speech recognition decision-making, and language translation.

Conversely, ML is a specific subset of AI that focuses on training algorithms to make predictions or decisions based on data. It involves using algorithms to learn patterns in data and using that knowledge to make decisions or predictions

3. Applications of AI & ML in five major domains of society.

AI and ML have a wide range of applications in various domains of society, and some of the most significant areas of application are

I. Healthcare: AI and ML are being used to analyze large amounts of medical data to help diagnose diseases, predict patient outcomes, and personalize treatment plans. AI is also being used in the development of medical imaging tools, such as computer-aided diagnosis systems and medical image analysis

II. Finance: AI and ML are being used in the finance industry to analyze large amounts of financial data, detect fraud, and make more informed investment decisions. They are also being used to develop personalized financial advice and investment recommendations for individuals

III. Retail: AI and ML are being used in retail to personalize customer experiences, such as product recommendations and targeted

marketing. They are also being used to optimize supply chain management and improve operational efficiency

IV. Transportation: AI and ML are being used to develop autonomous vehicles, improve traffic management, and optimize logistics and supply chain management

V. Education: AI and ML are being used to develop personalized learning experiences, such as adaptive learning systems, and to analyze student data to inform teaching and improve educational outcomes

4.Globally renowned companies using AI & ML in their business field

AI and ML are rapidly being adopted by companies across a wide range of industries, and there are many globally renowned companies that are using these technologies in their business operations. Here are a few examples:

I. **Google:** Google is a leader in the development of AI and ML, and it uses these technologies in a wide range of its products and services, including search, advertising, and image and speech recognition

II. **Amazon:** Amazon uses AI and ML to personalize customer experiences, such as product recommendations, and to optimize its supply chain and logistics operations

III. **Facebook:** Facebook is using AI and ML in several of its products and services, such as news feed customization, image and speech recognition, and natural language processing

5.Future scope of AI & ML

The future scope of AI and ML is both exciting and expansive. As these technologies continue to evolve and become more sophisticated, they will likely have an even greater impact on a wide range of industries and domains. Here are a few areas where AI and ML are expected to play a significant role in the future

I. **Healthcare:** AI and ML are expected to revolutionize the healthcare industry by enabling more accurate diagnoses, personalized treatment plans, and improved patient outcomes. They are also expected to play a key role in the development of new medical technologies and treatments.

II. **Transportation**: AI and ML are expected to play a significant role in the development of autonomous vehicles and the improvement of transportation systems, leading to safer and more efficient transportation.

III. **Finance:** AI and ML are expected to continue to shape the finance industry by enabling more accurate financial predictions, reducing fraud, and improving investment decisions.

IV. **Retail:** AI and ML are expected to continue to play a key role in the retail industry by enabling more personalized customer experiences and improving supply chain efficiency

V. **Education:** AI and ML are expected to play an increasingly important role in education by enabling personalized learning experiences. improving teaching outcomes.

6: Objectives of the course

The objectives of a course in AI and ML can vary depending on the level and focus of the course, but generally, the following are common objectives:

I. **Fundamentals of AI and ML:** A course in AI and ML typically aims to provide students with a comprehensive understanding of the basic concepts and techniques used in these fields, including supervised and unsupervised learning, decision trees, and neural networks.

II. **Applications of AI and ML:** The course will typically focus on the practical applications of AI and ML, including the use of these technologies in real-world scenarios, such as healthcare, finance, and retail.

III. **Problem-solving with AI and ML:** The course will usually aim to equip students with the skills and knowledge needed to apply AI and ML techniques to solve real-world problems, such as developing predictive models and optimizing decision-making processes.

IV. **Hands-on experience:** A course in AI and ML will typically include hands-on experience, such as coding and working with real datasets, to provide students with a practical understanding of the technologies.

7.Recent advancement of AI & ML

AI and ML have advanced rapidly in recent years, and there have been many exciting developments and breakthroughs. Here are a few of the most notable recent advancements in AI and ML:

I. **Deep learning**: Deep learning, a subset of machine learning, has made significant advances in recent years and is now being used for a wide range of applications, including computer vision, speech recognition, and natural language processing.

II. **Reinforcement learning:** Reinforcement learning is a type of machine learning that involves training agents to make decisions in an environment by receiving rewards or penalties. This area of AI has made significant advances in recent years, and it is now being used for a wide range of applications, including robotics and gaming.

III. **Generative models:** Generative models are a type of deep learning model that can generate new data that is similar to a given input. This area of AI has made significant advances in recent years, and it is now being used for a wide range of applications, including image and audio synthesis.

IV. **Transfer learning:** Transfer learning is a technique that allows deep learning models trained on one task to be fine-tuned for a different task, using a smaller amount of training data. This

area of AI has made significant advances in recent years, and it is now being used for a wide range of applications, including natural language processing and computer vision

8. Necessary tools for implementing ML

There are a variety of tools and technologies that are commonly used for implementing machine learning (ML) projects. Here are some of the most essential tools:

I. **Programming languages:** Python is ML's most commonly used programming language due to its simplicity, versatility, and abundance of libraries and frameworks. R and Julia are also used in ML, but Python is by far the most popular choice.

II. **Data analysis and visualization tools:** Tools such as Pandas, Matplotlib, and Seaborn are essential for analyzing and visualizing data in ML projects.

III. **ML libraries and frameworks:** TensorFlow, PyTorch, and scikit-learn are some of the most popular ML libraries and frameworks for developing ML models. They provide a wide range of algorithms and tools for building and training ML models.

IV. **Deep learning frameworks:** Keras and PyTorch are some of the most popular deep learning frameworks for developing complex deep learning models.

V. **Model deployment:** Platforms such as TensorFlow Serving, Flask, and Django can be used to deploy ML models in a production environment.

9. List of libraries used (need descriptions)

some of the most commonly used machine learning (ML) libraries and a brief description of each:

I. **NumPy:** NumPy is a Python numerical computing library that supports arrays, matrices, and multi-dimensional arrays. It is widely used in ML for array processing and computation.

II. **Pandas:** Pandas is Python's library for data analysis and manipulation. It provides data structures and functions for working with structured data, including data frames and series, which are commonly used in ML.

III. **Matplotlib:** Matplotlib is a plotting library in Python that is used for data visualization. It provides a wide range of plotting options and can be used to create visualizations such as line charts, scatter plots, histograms, and bar charts.

IV. **Seaborn:** Seaborn is a data visualization library in Python that is built on top of matplotlib. It provides advanced plotting options and is widely used for data visualization in ML.

V. **scikit-learn:** sci-kit-learn is a machine learning library in Python that provides a wide range of ML algorithms and tools. It is widely used in ML due to its simplicity, versatility, and abundance of algorithms and tools.

11. List of algorithms used (need descriptions)

Linear Regression: Linear Regression is a simple ML algorithm that models the relationship between two variables. It is used for regression problems where the goal is to predict a continuous output value based on one or more input variables

I. **Logistic Regression:** Logistic Regression is a type of regression algorithm that is used for classification problems. It models the probability of a binary outcome based on one or more input variables

II. **Decision Trees:** Decision Trees are a type of ML algorithm that is used for both classification and regression problems. They are based on a tree-like structure and use a series of rules to make predictions based on input variables

III. **k-Nearest Neighbors (k-NN):** k-NN is a simple ML algorithm that is used for classification problems. It makes predictions based on the k closest neighbors in the training data.

IV. **Support Vector Machines (SVMs):** SVMs are a type of ML algorithm that is used for classification and regression problems. They model the relationship between the input variables and the target variable by finding a boundary that separates the classes in the training data

V. **Neural Networks:** Neural Networks are a type of ML algorithm that are based on the structure of the human brain. They are widely used in deep learning and are used for a wide range of ML tasks, including image classification, natural language processing, and reinforcement learning

VI. **Convolutional Neural Networks (CNNs):** CNNs are a type of deep learning algorithm that is used for image classification and recognition. They are based on the idea of convolutional filters that scan the input image and identify features for classification

11. Write down sequentially all the class works / assignments clearly (** only error free code to be used)

Class 1

Q1-Remove duplicates from a list

In [1]:
```
l=[5,9,8,5,9,6,3,6,3,2]
for e in l:
    i=l.index(e)+1
    while i<len(l):
        if e==l[i]:
            l.pop(i)
        i+=1
print(l)
```

```
[5, 9, 8, 6, 3, 2]
```

Q2-Get the smallest number from a list

In [2]:
```
l=[5,7,8,5,9,6,4,6,3,2]
temp=l[0]
i=1
while i<len(l):
    if l[i]<temp:
        temp=l[i]
    i+=1
print(temp)
```

```
2
```

Q3- Print the numbers of a specified list after removing even numbers from it

In [3]:
```
l=[5,7,8,5,9,6,4,6,3,2]
for e in l:
    if e%2==0 :
        l.pop(l.index(e))
print(l)
```

```
[5, 7, 5, 9, 4, 3]
```

Q4-The program takes a number and reverses it.

In [4]:
```
N=input()
i=len(N)-1
while i>=0:
    print(N[i],end='')
    i-=1
```

```
10
01
```

Q5-Generate and print a list of first and last 5 elements where the values are square of numbers between two numbers

In [5]:
```
def printvalues():
    l = list()
    for i in range (0,10):
        l.append(i**2)
    print(l[:5])
    print(l[-5:])
```

In [6]:
```
printvalues()
```

```
[0, 1, 4, 9, 16]
[25, 36, 49, 64, 81]
```

Class 2

In [12]:

```
condition=input("has power?")

if condition=="yes":
    print("seek other help")

elif condition=="no":
    plugg_condition=input("Is plugged in?")
    if plugg_condition=="no":
        print("Plug it in")

    elif plugg_condition=="yes":
        switch_condition=input("Is the switch on?")
        if switch_condition=="no":
            print("Turn switch on")
        elif switch_condition=="yes":
            fuse_condition=input("Fuse OK?")
            if fuse_condition=="no":
                print("Check fuse")
            elif fuse_condition=="yes"
                 print("Seek other help")

            else:
                print("wrong input:please write no or yes")




        else:
             print("wrong input:please write no or yes")



    else:
        print("wrong input:please write no or yes")



else:
    print("wrong input:please write no or yes")
```

```
has power?no
Is plugged in?yes
Is the switch on?yes
Fuse OKyes
Seek other help
```

Class 3

use math library and evaluate the area of a triangle

In [14]:
```
import math
a = 5
b = 6
c = 7

# calculate the semi-perimeter
s = (a + b + c) / 2

# calculate the area
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
print('The area of the triangle is %0.2f' %area)
```

```
The area of the triangle is 14.70
```

use math library and evaluate the power and square root of numbers

In [2]:
```
import math

# Calculate the power of a number
result = math.pow(2, 3)
print(result)

# Calculate the square root of a number
result = math.sqrt(4)
print(result)  # 2.0
```

```
8.0
2.0
```

3. use math library and evaluate exp() and log() of an integer

In [11]:
```
import math
result = math.exp(64)
ans = math.log(1000, 10)
print(result)
print(ans)
```

```
6.235149080811617e+27
2.9999999999999996
```

4. use math library and evaluate log2 and log10

In [13]:
```
import math
x = 8
x = 100
log2x = math.log2(x)
log10x = math.log10(x)
print(log10x)
print(log2x)
```

```
2.0
6.643856189774724
```

5. use math library and evaluate pi and e

In [7]:
```
import math
print(math.e)
print(math.pi)
```

```
2.718281828459045
3.141592653589793
```

Class 4

In [5]:
```
import pandas as pd
titles=pd.read_csv('titles.csv')
titles.head(5)
```

Out[5]:

	title	year
0	The Rising Son	1990
1	The Thousand Plane Raid	1969
2	Crucea de piatra	1993
3	Country	2000
4	Gaiking II	2011

In [6]:
```
after85=titles[titles['year']>1985]
after85
```

Out[6]:

	title	year
0	The Rising Son	1990
2	Crucea de piatra	1993
3	Country	2000
4	Gaiking II	2011
5	Medusa (IV)	2015

In [6]:
```
# importing the module
import pandas as pd

# creating a sample dataframe
data = pd.DataFrame({'Brand': ['Maruti', 'Hyundai', 'Tata',
                               'Mahindra', 'Maruti', 'Hyundai',
                               'Renault', 'Tata', 'Maruti'],
                     'Year': [2012, 2014, 2011, 2015, 2012,
                              2016, 2014, 2018, 2019],
                     'Kms Driven': [50000, 30000, 60000,
                                    25000, 10000, 46000,
                                    31000, 15000, 12000],
                     'City': ['Gurgaon', 'Delhi', 'Mumbai',
                              'Delhi', 'Mumbai', 'Delhi',
                              'Mumbai', 'Chennai', 'Ghaziabad'],
                     'Mileage': [28, 27, 25, 26, 28,
                                 29, 24, 21, 24]})
```

In [7]:
```
data
```

Out[7]:

	Brand	Year	Kms Driven	City	Mileage
0	Maruti	2012	50000	Gurgaon	28
1	Hyundai	2014	30000	Delhi	27
2	Tata	2011	60000	Mumbai	25
3	Mahindra	2015	25000	Delhi	26
4	Maruti	2012	10000	Mumbai	28
5	Hyundai	2016	46000	Delhi	29
6	Renault	2014	31000	Mumbai	24
7	Tata	2018	15000	Chennai	21
8	Maruti	2019	12000	Ghaziabad	24

In [8]:
```
import numpy as np
```

In [11]:
```
a_1d=np.arange(3)
a_1d
```

Out[11]:
```
array([0, 1, 2])
```

In [14]:
```
a_2d=np.arange(12).reshape((3,4))
a_2d
```

Out[14]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [15]:
```
a_3d = np.arange(24).reshape((2, 3, 4))
print(a_3d)
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

In [17]:
```
arr_1D = np.array( [2,4,6,8] )     # create  NumPy 1D array which contain int value 2,4,6,8
print(arr_1D)         # print arr_1D
```

```
[2 4 6 8]
```

In [18]:
```
arr_2D = np.array ([ [0, 1, 1], [1, 0, 1], [1, 1, 0] ] )         # Create Numpy 2D array which contain inter type valye
print(arr_2D) # print arr_1D
```

```
[[0 1 1]
 [1 0 1]
 [1 1 0]]
```

In [19]:
```
c = np.array([(1,2,3),
              (4,5,6)])
print(c.shape)
(2, 3)
```

```
(2, 3)
```

Out[19]:
```
(2, 3)
```

In [20]:
```
import numpy as np
a = np.array([1, 2, 3], dtype = complex)
np.conj(a)
```

Out[20]:
```
array([1.-0.j, 2.-0.j, 3.-0.j])
```

In [21]:
```
a = np.array([[1, 2, 4], [5, 8, 7]], dtype = 'float')
print ("Array created using passed list:\n", a)
```

```
Array created using passed list:
 [[1. 2. 4.]
 [5. 8. 7.]]
```

In [22]:
```
import numpy as np
 arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newarr = arr.reshape(2, 3, 2)
 print(newarr)
```

```
[[[ 1  2]
  [ 3  4]
  [ 5  6]]

 [[ 7  8]
  [ 9 10]
  [11 12]]]
```

In [26]:
```
    arr = np.array([[1, 2, 3], [4, 5, 6]])
flarr = arr.flatten()
print ("\nOriginal array:\n", arr)


    print ("Fattened array:\n", flarr)
```

```
Original array:
 [[1 2 3]
 [4 5 6]]
Fattened array:
 [1 2 3 4 5 6]
```

In [27]:
```
np.random.rand(4)
```

Out[27]:
```
array([0.41158614, 0.0508884 , 0.29040874, 0.09685511])
```

In [28]:
```
np.random.rand(4,3)
```

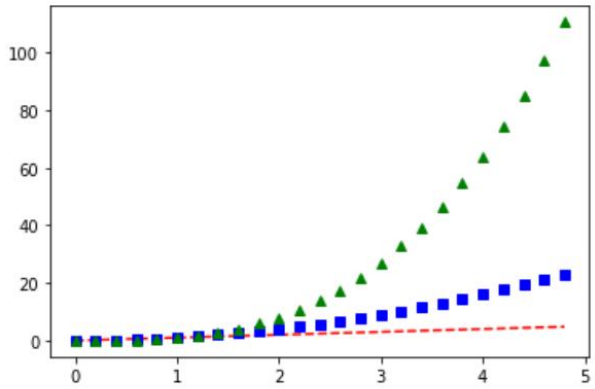
Out[28]:
```
array([[0.95068804, 0.29448749, 0.40663176],
       [0.62305187, 0.4949959 , 0.68233317],
       [0.29877634, 0.30492552, 0.65217195],
       [0.23596772, 0.2670643 , 0.66246387]])
```

Class 5

In [1]:
```
import numpy as np
 import matplotlib.pyplot as plt
```

In [2]:
```
# evenly sampled time at 200ms intervals

t = np.arange(0., 5., 0.2)
 # red dashes, blue squares and green triangles
    plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
 #plt.plot(x, y, linewidth=2)
```

Out[2]:
```
[<matplotlib.lines.Line2D at 0x23cf28e2c70>,
 <matplotlib.lines.Line2D at 0x23cf28e2d60>,
 <matplotlib.lines.Line2D at 0x23cf28e2cd0>]
```



In [4]:
```
import numpy as np
import matplotlib.pyplot as plt

 def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)
```

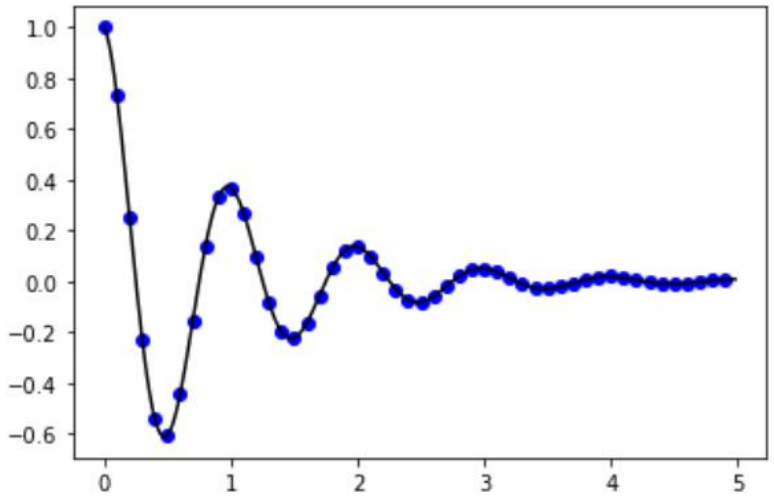
In [5]:
```
t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)
```

In [6]:
```
t1
t2
```

Out[6]:
```
array([0.  , 0.02, 0.04, 0.06, 0.08, 0.1 , 0.12, 0.14, 0.16, 0.18, 0.2 ,
       0.22, 0.24, 0.26, 0.28, 0.3 , 0.32, 0.34, 0.36, 0.38, 0.4 , 0.42,
       0.44, 0.46, 0.48, 0.5 , 0.52, 0.54, 0.56, 0.58, 0.6 , 0.62, 0.64,
       0.66, 0.68, 0.7 , 0.72, 0.74, 0.76, 0.78, 0.8 , 0.82, 0.84, 0.86,
       0.88, 0.9 , 0.92, 0.94, 0.96, 0.98, 1.  , 1.02, 1.04, 1.06, 1.08,
       1.1 , 1.12, 1.14, 1.16, 1.18, 1.2 , 1.22, 1.24, 1.26, 1.28, 1.3 ,
       1.32, 1.34, 1.36, 1.38, 1.4 , 1.42, 1.44, 1.46, 1.48, 1.5 , 1.52,
       1.54, 1.56, 1.58, 1.6 , 1.62, 1.64, 1.66, 1.68, 1.7 , 1.72, 1.74,
       1.76, 1.78, 1.8 , 1.82, 1.84, 1.86, 1.88, 1.9 , 1.92, 1.94, 1.96,
       1.98, 2.  , 2.02, 2.04, 2.06, 2.08, 2.1 , 2.12, 2.14, 2.16, 2.18,
       2.2 , 2.22, 2.24, 2.26, 2.28, 2.3 , 2.32, 2.34, 2.36, 2.38, 2.4 ,
       2.42, 2.44, 2.46, 2.48, 2.5 , 2.52, 2.54, 2.56, 2.58, 2.6 , 2.62,
```

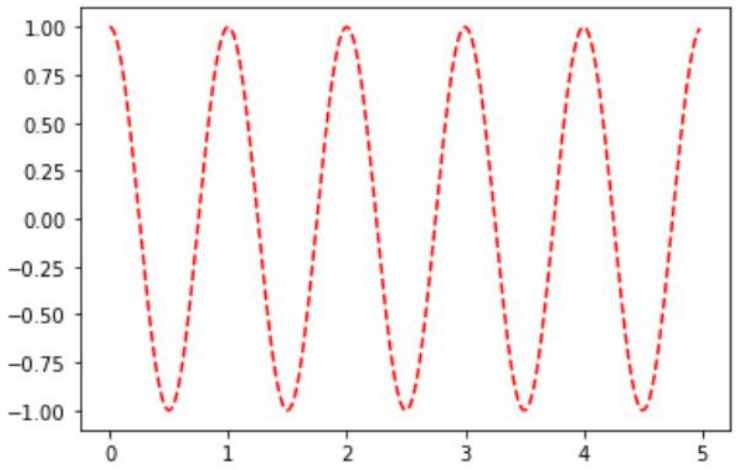
In [7]:
```
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x23cf295b100>,
 <matplotlib.lines.Line2D at 0x23cf295b160>]
```

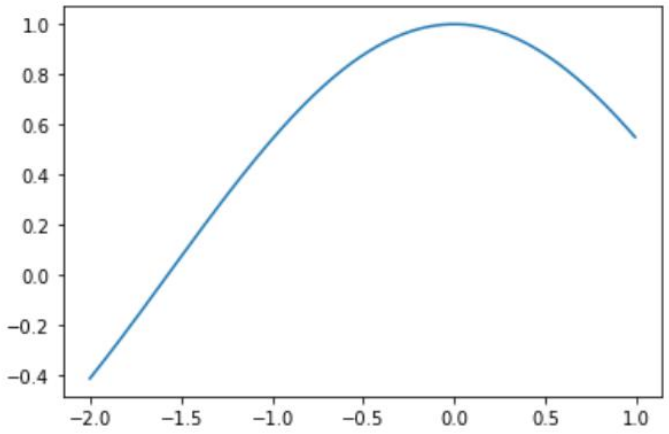


In [8]:
```
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0x23cf29e9850>]
```

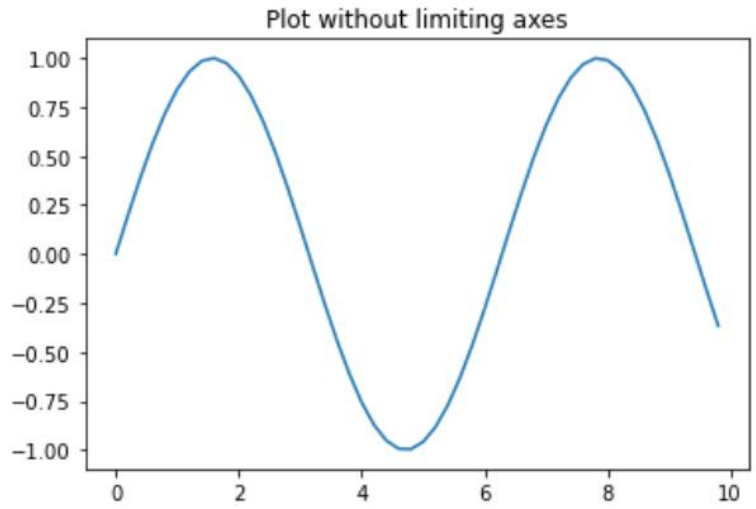


In [11]:
```
xvals = np.arange(-2, 1, 0.01) # Grid of 0.01 spacing from -2 to 10
yvals = np.cos(xvals) # Evaluate function on xvals
plt.plot(xvals, yvals) # Create line plot with yvals against xvals
plt.show() # Show the figure
```



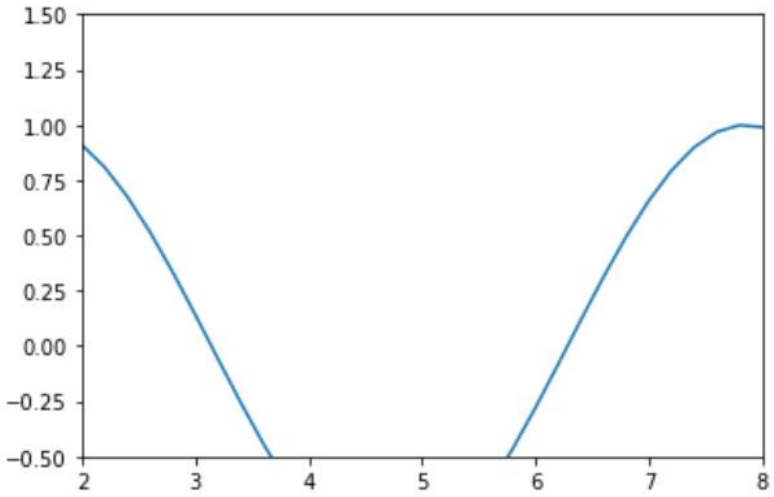
In [12]:
```
x = np.arange(0, 10, 0.2)
y = np.sin(x)
 # Plot
    plt.plot(x, y)
 # Add title
 plt.title("Plot without limiting axes")
```

Out[12]:
```
Text(0.5, 1.0, 'Plot without limiting axes')
```



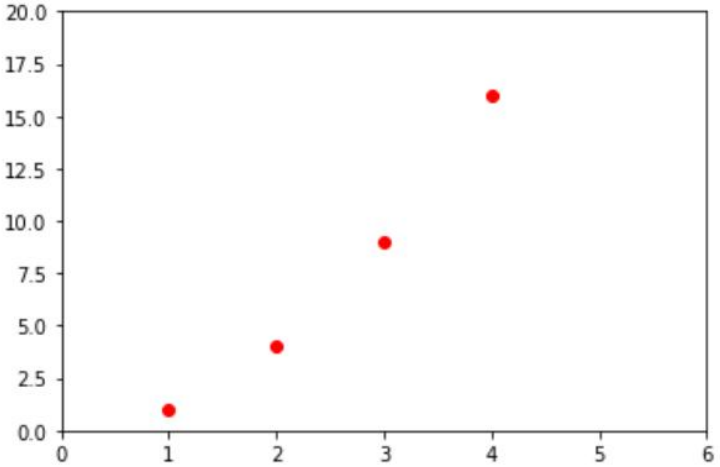
In [13]:
```
x = np.arange(0, 10, 0.2)
y = np.sin(x)
 # Plot
    plt.plot(x, y)
 # Set axes limit
 plt.xlim(2, 8)
plt.ylim(-0.50,1.5)
```

Out[13]: (-0.5, 1.5)

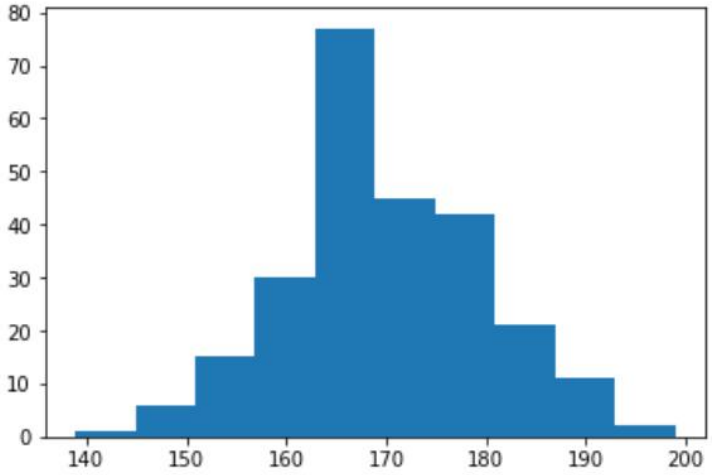


In [30]:
```
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20])
```

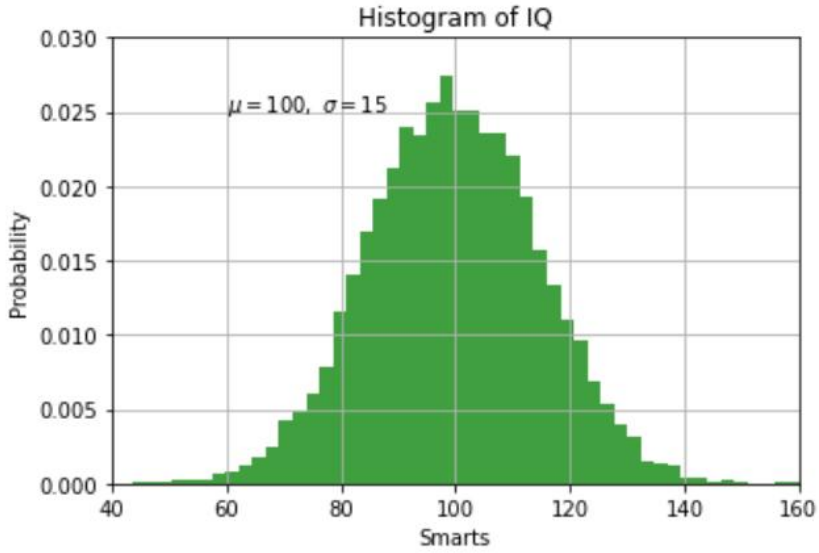
Out[30]: (0.0, 6.0, 0.0, 20.0)



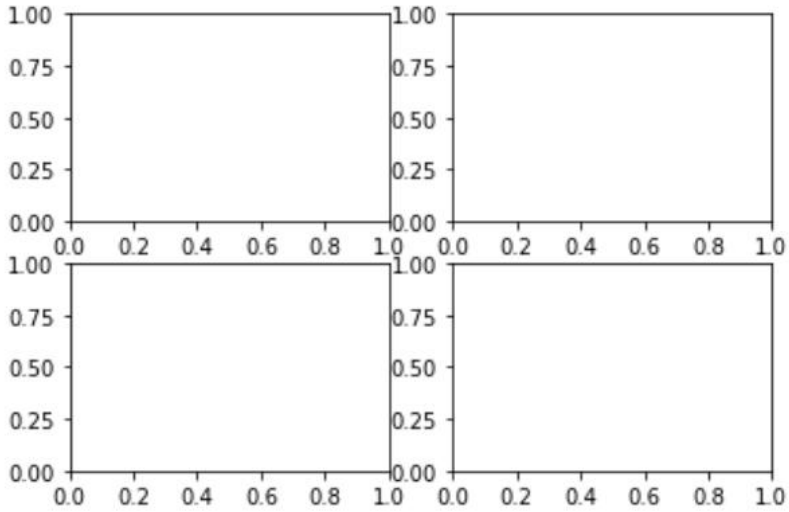
In [31]:
```
x=np.random.normal(170,10,250)
plt.hist(x)
plt.show()
```



In [44]:
```
np.random.seed(19680801)
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)
 # the histogram of the data
n, bins, patches = plt.hist(x, 50, density=1, facecolor='g', alpha=0.75)
 plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```



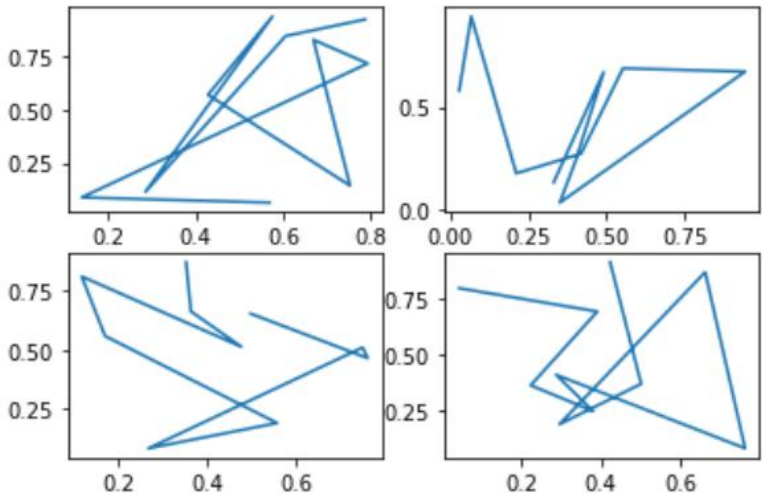
In [56]:
```
fig1, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
```



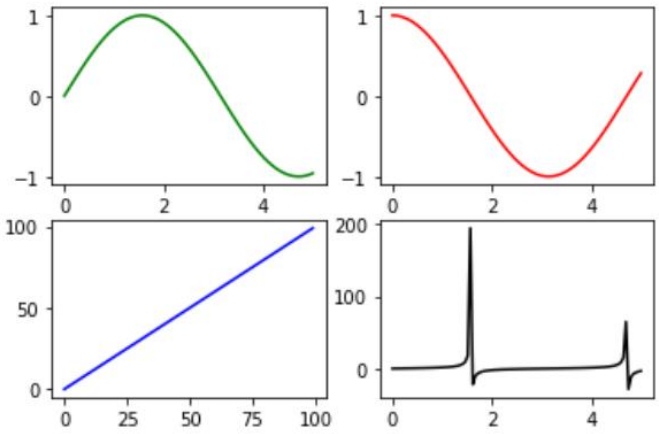
In [57]:
```
x1 = np.random.rand(10)
x2 = np.random.rand(10)
x3 = np.random.rand(10)
x4 = np.random.rand(10)
y1 = np.random.rand(10)
y2 = np.random.rand(10)
y3 = np.random.rand(10)
y4 = np.random.rand(10)
```

In [58]:
```
figure2, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)
ax1.plot(x1,y1)
ax2.plot(x2,y2)
ax3.plot(x3,y3)
ax4.plot(x4,y4)
```

Out[58]: [<matplotlib.lines.Line2D at 0x23cf29bdb20>]



In [59]:
```
# Create a Figure with 2 rows and 2 columns of subplots:
fig, ax = plt.subplots(2, 2)
 x = np.linspace(0, 5, 100)
 # Index 4 Axes arrays in 4 subplots within 1 Figure:
ax[0, 0].plot(x, np.sin(x), 'g') #row=0, column=0
ax[1, 0].plot(range(100), 'b') #row=1, column=0
ax[0, 1].plot(x, np.cos(x), 'r') #row=0, column=1
ax[1, 1].plot(x, np.tan(x), 'k') #row=1, column=1
 plt.show()
```



Class6

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn import datasets
import seaborn as sns
```

In [2]:
```
dataset = pd.read_csv('housing.csv')  # Load data set
print(dataset)
```

```
     Unnamed: 0     price  lotsize  bedrooms  bathrms  stories driveway  \
0             1   42000.0     5850         3        1        2      yes
1             2   38500.0     4000         2        1        1      yes
2             3   49500.0     3060         3        1        1      yes
3             4   60500.0     6650         3        1        2      yes
4             5   61000.0     6360         2        1        1      yes
..          ...       ...      ...       ...      ...      ...      ...
541         542   91500.0     4800         3        2        4      yes
542         543   94000.0     6000         3        2        4      yes
543         544  103000.0     6000         3        2        4      yes
544         545  105000.0     6000         3        2        2      yes
545         546  105000.0     6000         3        1        2      yes

    recroom fullbase gashw airco  garagepl prefarea
0        no      yes    no    no         1       no
1        no       no    no    no         0       no
2        no       no    no    no         0       no
3       yes       no    no    no         0       no
4        no       no    no    no         0       no
..      ...      ...   ...   ...       ...      ...
541     yes       no    no   yes         0       no
542      no       no    no   yes         0       no
543     yes       no    no   yes         1       no
544     yes       no    no   yes         1       no
545      no       no    no   yes         1       no
```

In [3]:
```
dataset.shape
```

Out[3]: (546, 13)

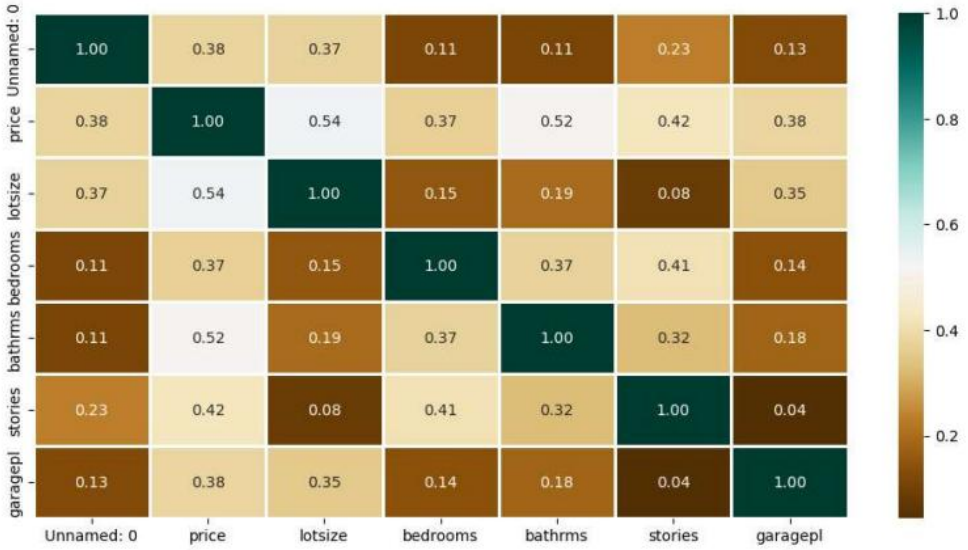
In [4]:
```
X = dataset.iloc[:, 0].values.reshape(-1, 1)
Y = dataset.iloc[:, 1].values.reshape(-1, 1)
```

In [7]:
```
obj = (dataset.dtypes == 'object')

object_cols = list(obj[obj].index)

print("Categorical variables:",len(object_cols))


int_ = (dataset.dtypes == 'int')

num_cols = list(int_[int_].index)

print("Integer variables:",len(num_cols))


fl = (dataset.dtypes == 'float')

fl_cols = list(fl[fl].index)

print("Float variables:",len(fl_cols))
```

```
Categorical variables: 6
Integer variables: 0
Float variables: 1
```

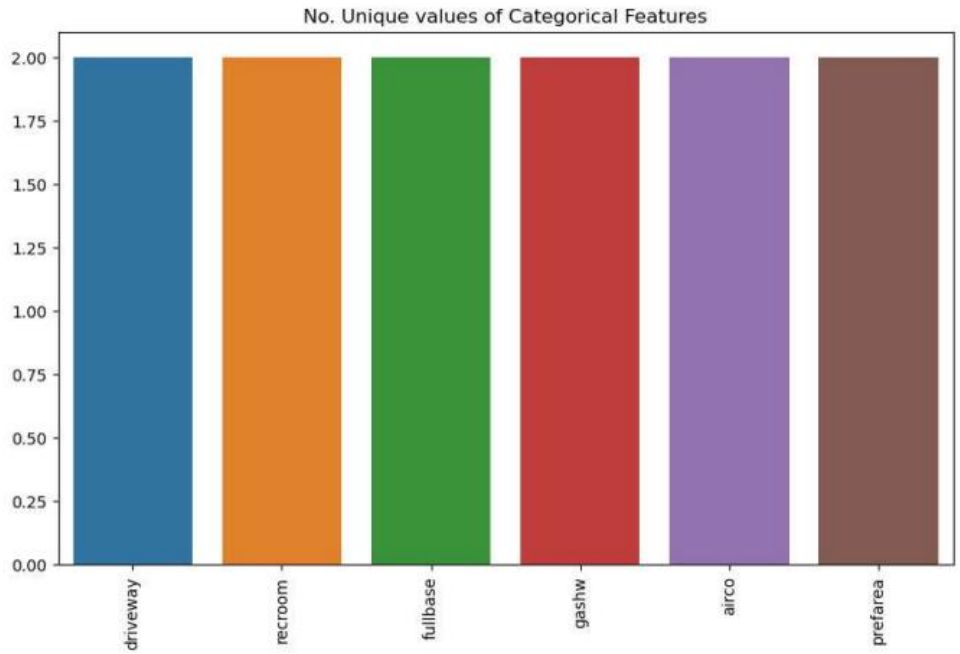
In [8]:
```
plt.figure(figsize=(12, 6))
sns.heatmap(dataset.corr(),

            cmap = 'BrBG',

            fmt = '.2f',

            linewidths = 2,

            annot = True)
```

Out[8]: <AxesSubplot:>



In [9]:
```
unique_values = []
for col in object_cols:
  unique_values.append(dataset[col].unique().size)
plt.figure(figsize=(10,6))
plt.title('No. Unique values of Categorical Features')
plt.xticks(rotation=90)
sns.barplot(x=object_cols,y=unique_values)
```

Out[9]: <AxesSubplot:title={'center':'No. Unique values of Categorical Features'}>



Class 7

In [1]:
```
true=[100,50,30,20]
pred=[90,50,50,30]
```

In [2]:
```
# calculate MAE
print((10+0+20+10)/4)
```

```
10.0
```

In [3]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(true,pred))
```

```
10.0
```

In [4]:
```
print(metrics.mean_squared_error(true,pred))
```

```
150.0
```

In [5]:
```
import numpy as np
print(np.sqrt(metrics.mean_squared_error(true,pred)))
```

```
12.24744871391589
```

In [6]:
```
import numpy as np
x=np.array([5,15,25,35,45,55]).reshape(-1,1)

y=np.array([5,20,14,32,22,28]).reshape(-1,1)
```

In [7]:
```
print(x)
```

```
[[ 5]
 [15]
 [25]
 [35]
 [45]
 [55]]
```

In [8]:
```
print(y)
```

```
[[ 5]
 [20]
 [14]
 [32]
 [22]
 [28]]
```

In [9]:
```
from sklearn.linear_model import LinearRegression
```

In [10]:
```
model=LinearRegression()
model.fit(x,y)
```

Out[10]:
```
LinearRegression()
```

In [11]:
```
rsq=model.score(x,y)
print('Coefficient of determination: ',rsq)
```

```
Coefficient of determination:  0.5837454051059972
```

In [12]:
```
y_pred=model.predict(x)
```

In [13]:
```
y_pred
```

Out[13]:
```
array([[10.23809524],
       [14.20952381],
       [18.18095238],
       [22.15238095],
       [26.12380952],
       [30.0952381 ]])
```

In [14]:
```
print('intercept:',model.intercept_)
```

```
intercept: [8.25238095]
```

In [15]:
```
print('coeff-x:',model.coef_)
```

```
coeff-x: [[0.39714286]]
```

In [16]:
```
import numpy as np
import matplotlib.pyplot as plt  # To visualize
import pandas as pd  # To read data
from sklearn.linear_model import LinearRegression
```

In [17]:
```
dataset = pd.read_csv('student_scores.csv')  # Load data set
print(dataset)
```

```
    Hours  Scores
0     2.5      21
1     5.1      47
2     3.2      27
3     8.5      75
4     3.5      30
5     1.5      20
6     9.2      88
7     5.5      60
8     8.3      81
9     2.7      25
10    7.7      85
11    5.9      62
12    4.5      41
13    3.3      42
14    1.1      17
15    8.9      95
16    2.5      30
17    1.9      24
18    6.1      67
19    7.4      69
```

In [18]:
```
dataset.shape
```

Out[18]:
```
(25, 2)
```

In [19]:
```
A = dataset.iloc[:, 0].values.reshape(-1, 1)
B = dataset.iloc[:, 1].values.reshape(-1, 1)
```

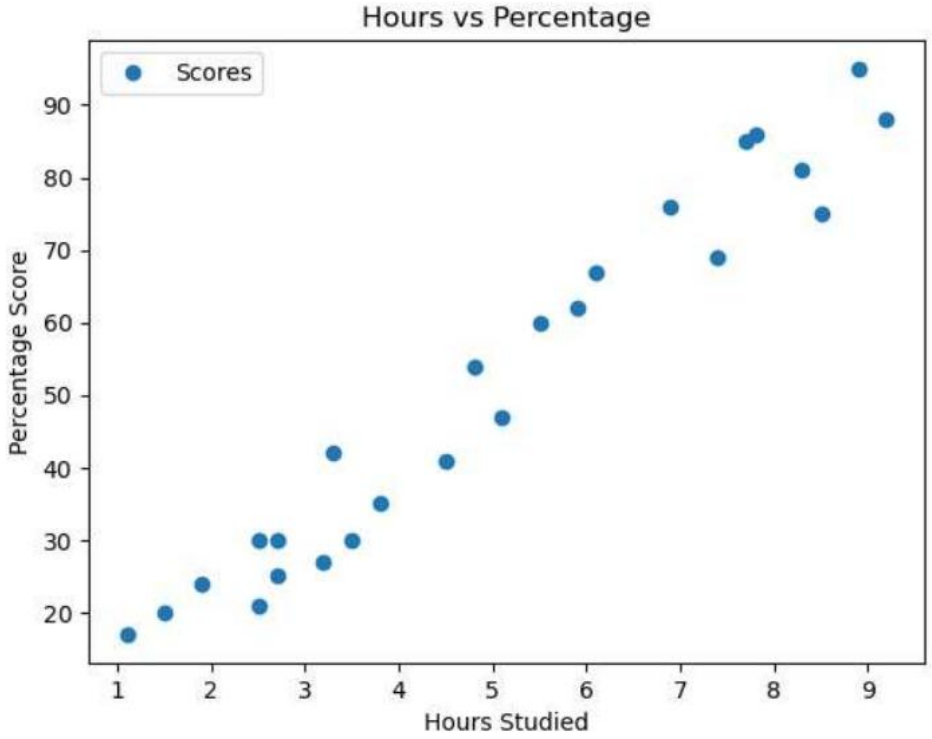
In [20]:
```
A
```

Out[20]:
```
array([[2.5],
       [5.1],
       [3.2],
       [8.5],
       [3.5],
       [1.5],
       [9.2],
       [5.5],
       [8.3],
       [2.7],
```

In [21]:
```
B
```

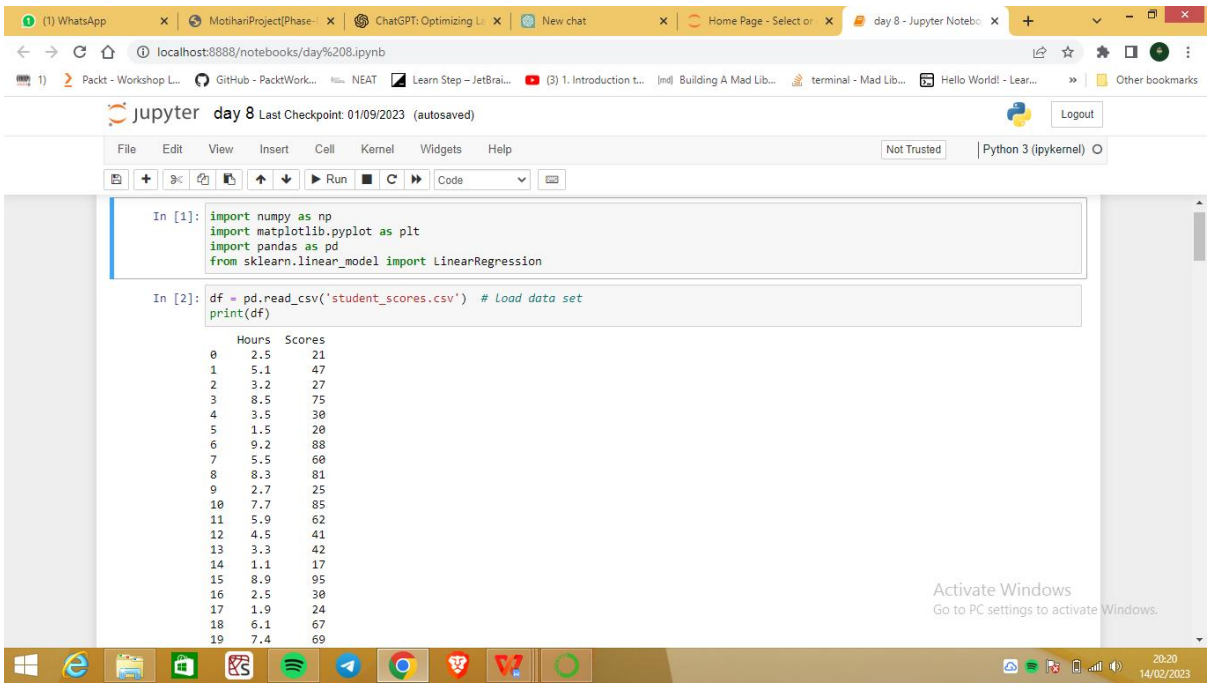
Out[21]:
```
array([[21],
       [47],
       [27],
       [75],
       [30],
       [20],
       [88],
       [60],
       [81],
       [25],
       [85],
       [62],
```

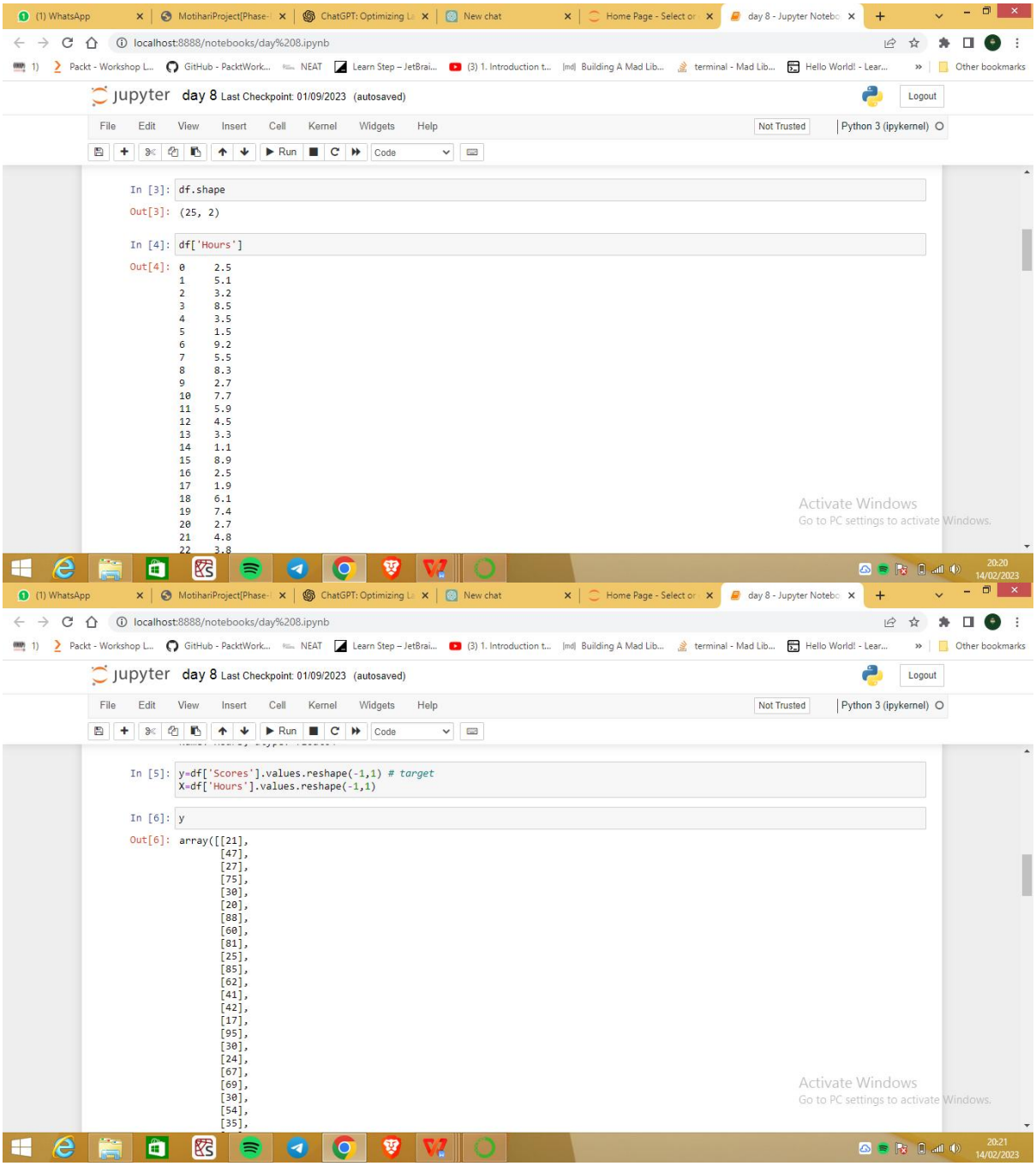
In [24]:
```
dataset.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



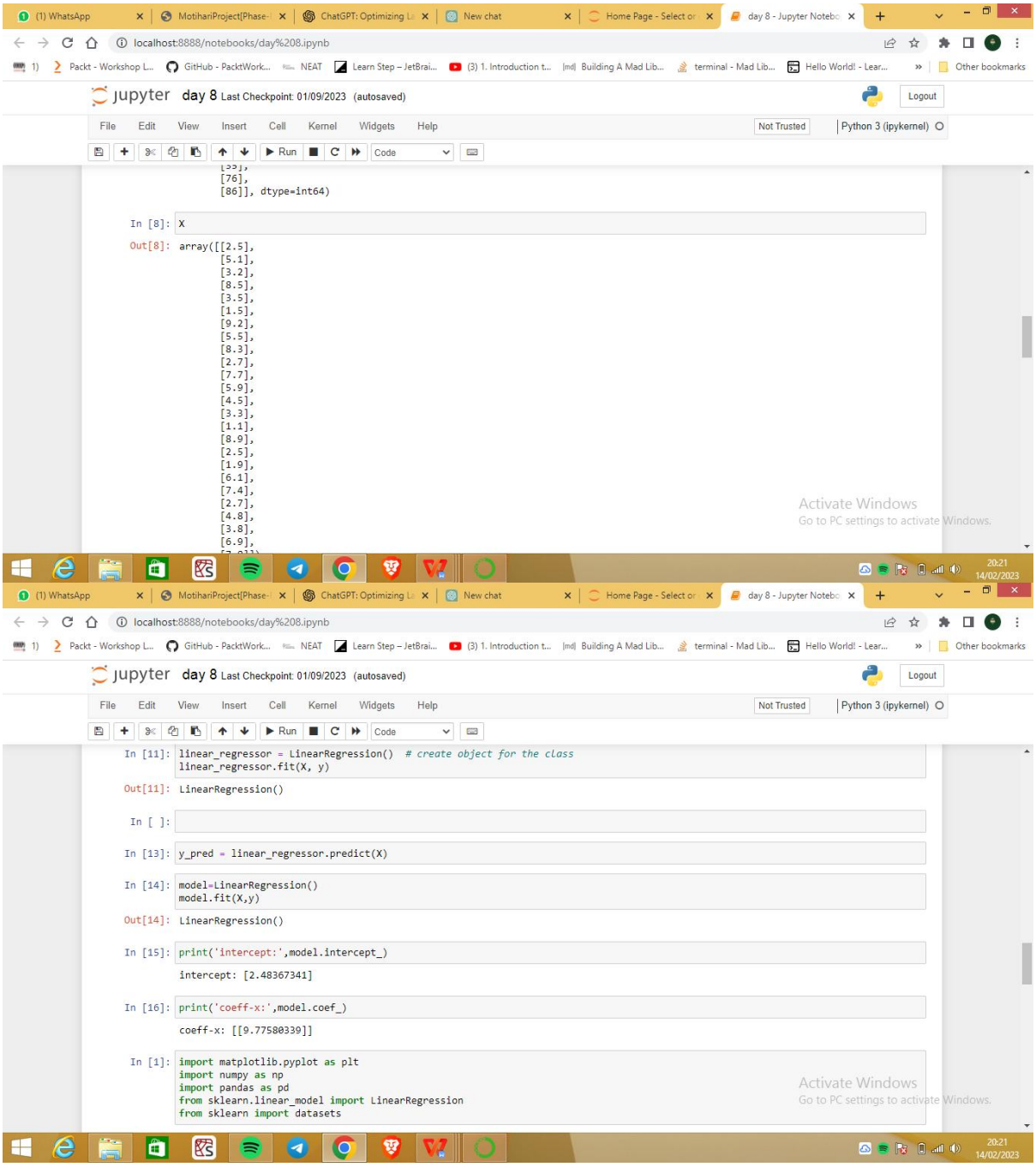
```
In [25]: model=LinearRegression()
         model.fit(A,B)

Out[25]: LinearRegression()

In [26]: print('coeff-x:',model.coef_)

         coeff-x: [[9.77580339]]

In [27]: print('intercept:',model.intercept_)

         intercept: [2.48367341]

In [28]: score9_5=model.predict([[5.4]])

In [29]: score9_5

Out[29]: array([[55.27301172]])
```

Class8

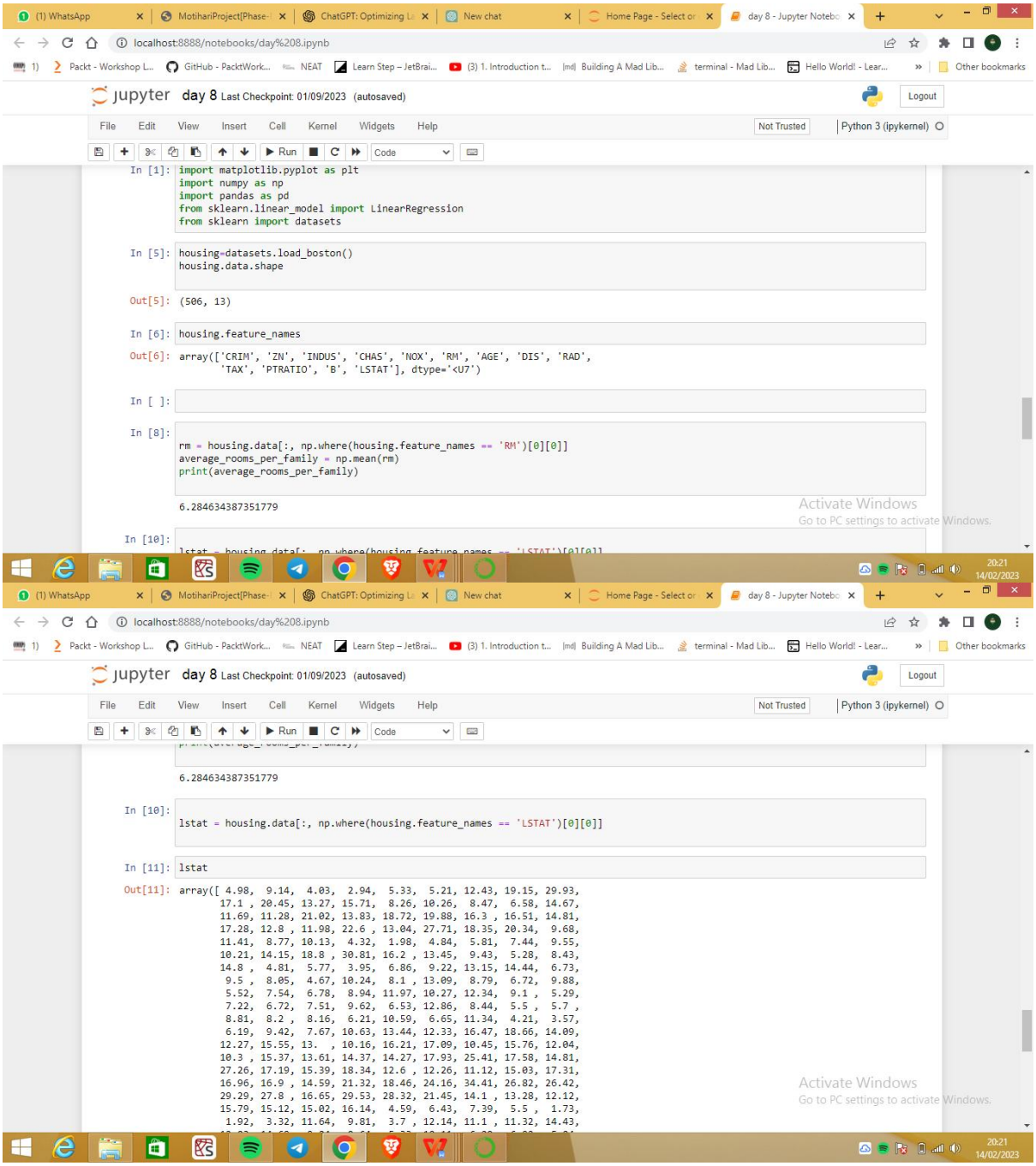



jupyter day 8 Last Checkpoint: 01/09/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [3]: df.shape
Out[3]: (25, 2)
In [4]: df['Hours']
Out[4]: 0 2.5
1 5.1
2 3.2
3 8.5
4 3.5
5 1.5
6 9.2
7 5.5
8 8.3
9 2.7
10 7.7
11 5.9
12 4.5
13 3.3
14 1.1
15 8.9
16 2.5
17 1.9
18 6.1
19 7.4
20 2.7
21 4.8
22 3.8
Activate Windows
Go to PC settings to activate Windows.
20:20
14/02/2023

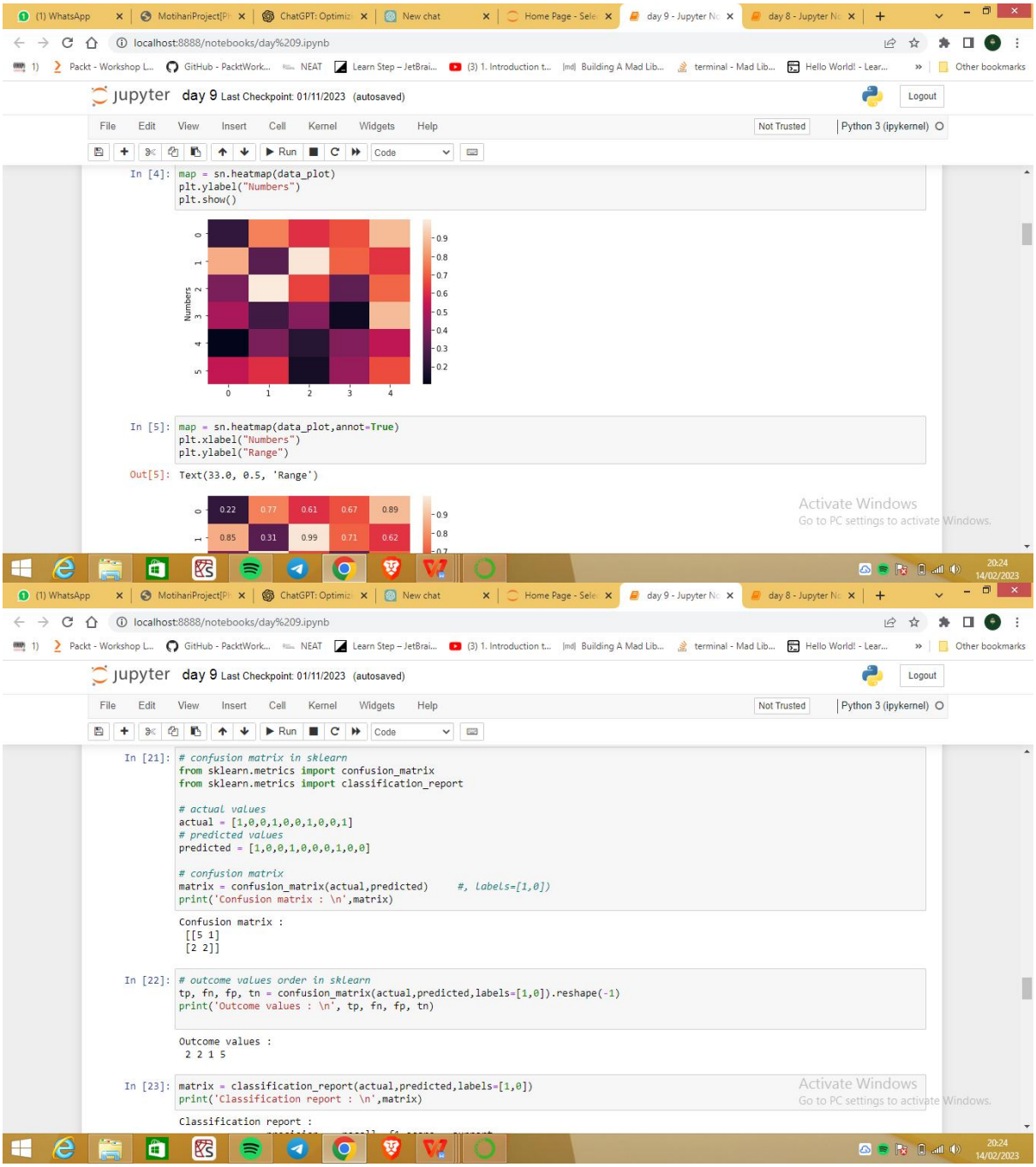
jupyter day 8 Last Checkpoint: 01/09/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [5]: y=df['Scores'].values.reshape(-1,1) # target
X=df['Hours'].values.reshape(-1,1)
In [6]: y
Out[6]: array([[21],
[47],
[27],
[75],
[30],
[20],
[88],
[60],
[81],
[25],
[85],
[62],
[41],
[42],
[17],
[95],
[30],
[24],
[67],
[69],
[30],
[54],
[35],
Activate Windows
Go to PC settings to activate Windows.
20:21
14/02/2023

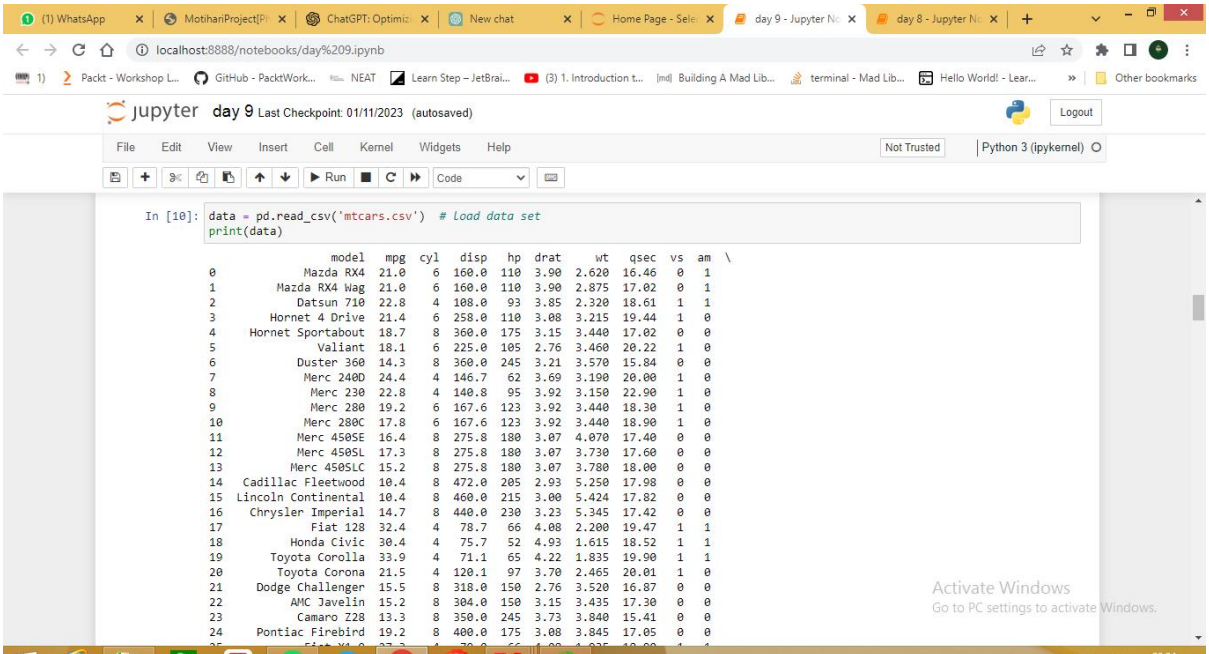

jupyter day 8 Last Checkpoint: 01/09/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
[76],
[86]], dtype=int64)
In [8]: X
Out[8]: array([[2.5],
[5.1],
[3.2],
[8.5],
[3.5],
[1.5],
[9.2],
[5.5],
[8.3],
[2.7],
[7.7],
[5.9],
[4.5],
[3.3],
[1.1],
[8.9],
[2.5],
[1.9],
[6.1],
[7.4],
[2.7],
[4.8],
[3.8],
[6.9],
Activate Windows
Go to PC settings to activate Windows.
20:21
14/02/2023

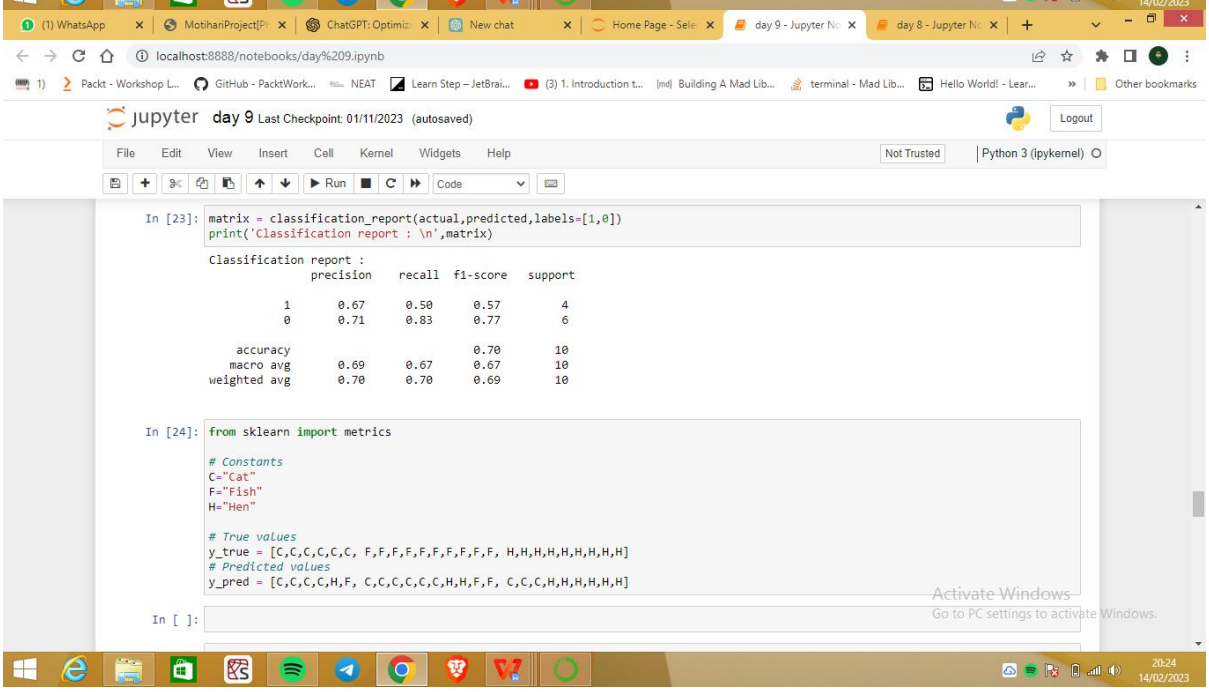
jupyter day 8 Last Checkpoint: 01/09/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [11]: linear_regressor = LinearRegression() # create object for the class
linear_regressor.fit(X, y)
Out[11]: LinearRegression()
In [ ]:
In [13]: y_pred = linear_regressor.predict(X)
In [14]: model=LinearRegression()
model.fit(X,y)
Out[14]: LinearRegression()
In [15]: print('intercept:',model.intercept_)
intercept: [2.48367341]
In [16]: print('coeff-x:',model.coef_)
coeff-x: [[9.77580339]]
In [1]: import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn import datasets
Activate Windows
Go to PC settings to activate Windows.
20:21
14/02/2023

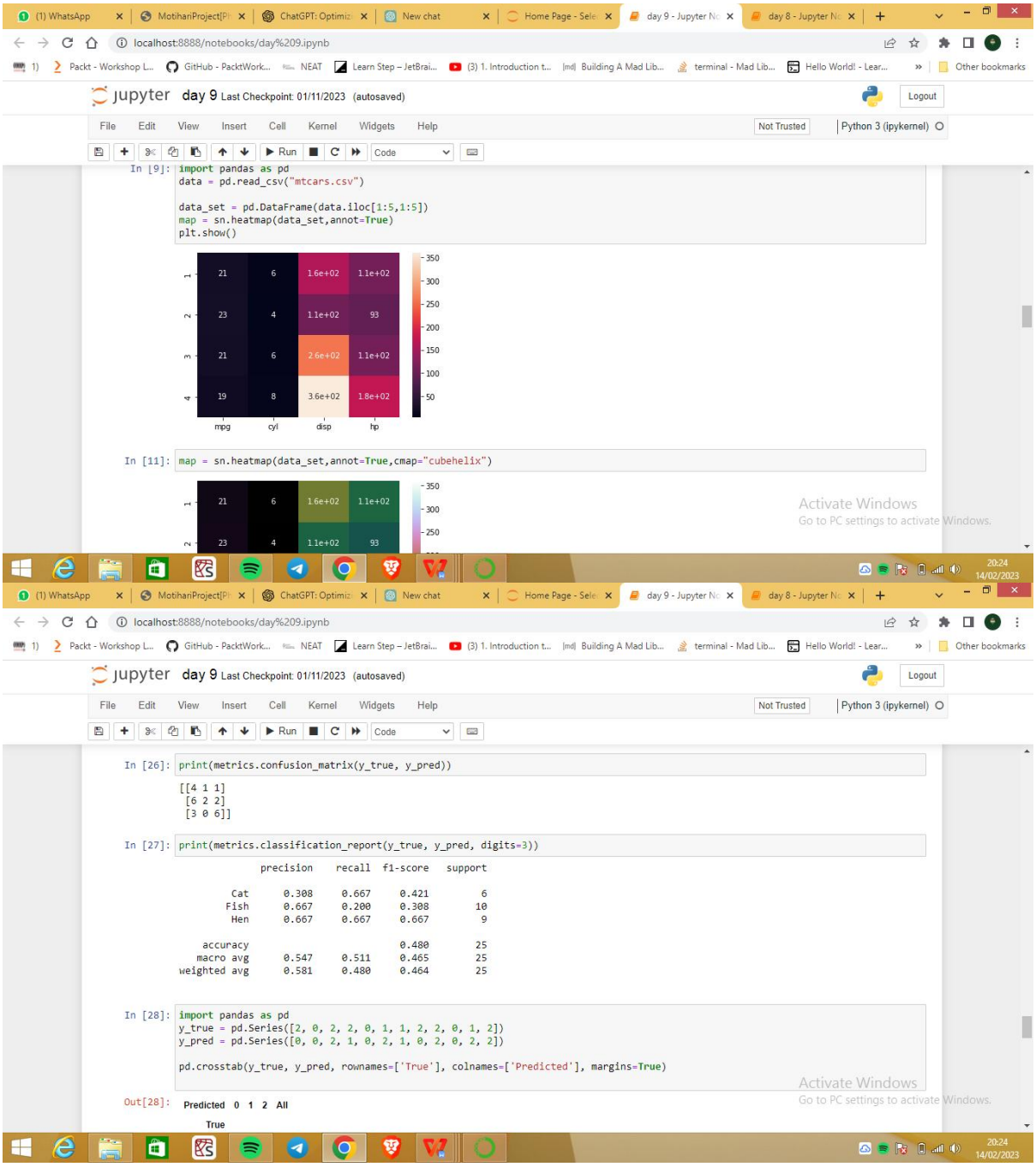

day 8 - Jupyter Notebook
localhost:8888/notebooks/day%208.ipynb
jupyter day 8 Last Checkpoint: 01/09/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [1]:
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn import datasets
In [5]:
housing=datasets.load_boston()
housing.data.shape
Out[5]:
(506, 13)
In [6]:
housing.feature_names
Out[6]:
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
In [ ]:
In [8]:
rm = housing.data[:, np.where(housing.feature_names == 'RM')[0][0]]
average_rooms_per_family = np.mean(rm)
print(average_rooms_per_family)
6.284634387351779
In [10]:
Activate Windows
Go to PC settings to activate Windows.
20:21
14/02/2023
6.284634387351779
In [10]:
lstat = housing.data[:, np.where(housing.feature_names == 'LSTAT')[0][0]]
In [11]:
lstat
Out[11]:
array([4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93,
17.1 , 20.45, 13.27, 15.71, 8.26, 10.26, 8.47, 6.58, 14.67,
11.69, 11.28, 21.02, 13.83, 18.72, 19.88, 16.3 , 16.51, 14.81,
17.28, 12.8 , 11.98, 22.6 , 13.04, 27.71, 18.35, 20.34, 9.68,
11.41, 8.77, 10.13, 4.32, 1.98, 4.84, 5.81, 7.44, 9.55,
10.21, 14.15, 18.8 , 30.81, 16.2 , 13.45, 9.43, 5.28, 8.43,
14.8 , 4.81, 5.77, 3.95, 6.86, 9.22, 13.15, 14.44, 6.73,
9.5 , 8.05, 4.67, 10.24, 8.1 , 13.09, 8.79, 6.72, 9.88,
5.52, 7.54, 6.78, 8.94, 11.97, 10.27, 12.34, 9.1 , 5.29,
7.22, 6.72, 7.51, 9.62, 6.53, 12.86, 8.44, 5.5 , 5.7 ,
8.81, 8.2 , 8.16, 6.21, 10.59, 6.65, 11.34, 4.21, 3.57,
6.19, 9.42, 7.67, 10.63, 13.44, 12.33, 16.47, 18.66, 14.09,
12.27, 15.55, 13. , 10.16, 16.21, 17.09, 10.45, 15.76, 12.04,
10.3 , 15.37, 13.61, 14.37, 14.27, 17.93, 25.41, 17.58, 14.81,
27.26, 17.19, 15.39, 18.34, 12.6 , 12.26, 11.12, 15.03, 17.31,
16.96, 16.9 , 14.59, 21.32, 18.46, 24.16, 34.41, 26.82, 26.42,
29.29, 27.8 , 16.65, 29.53, 28.32, 21.45, 14.1 , 13.28, 12.12,
15.79, 15.12, 15.02, 16.14, 4.59, 6.43, 7.39, 5.5 , 1.73,
1.92, 3.32, 11.64, 9.81, 3.7 , 12.14, 11.1 , 11.32, 14.43,
Activate Windows
Go to PC settings to activate Windows.
20:21
14/02/2023

Class 9

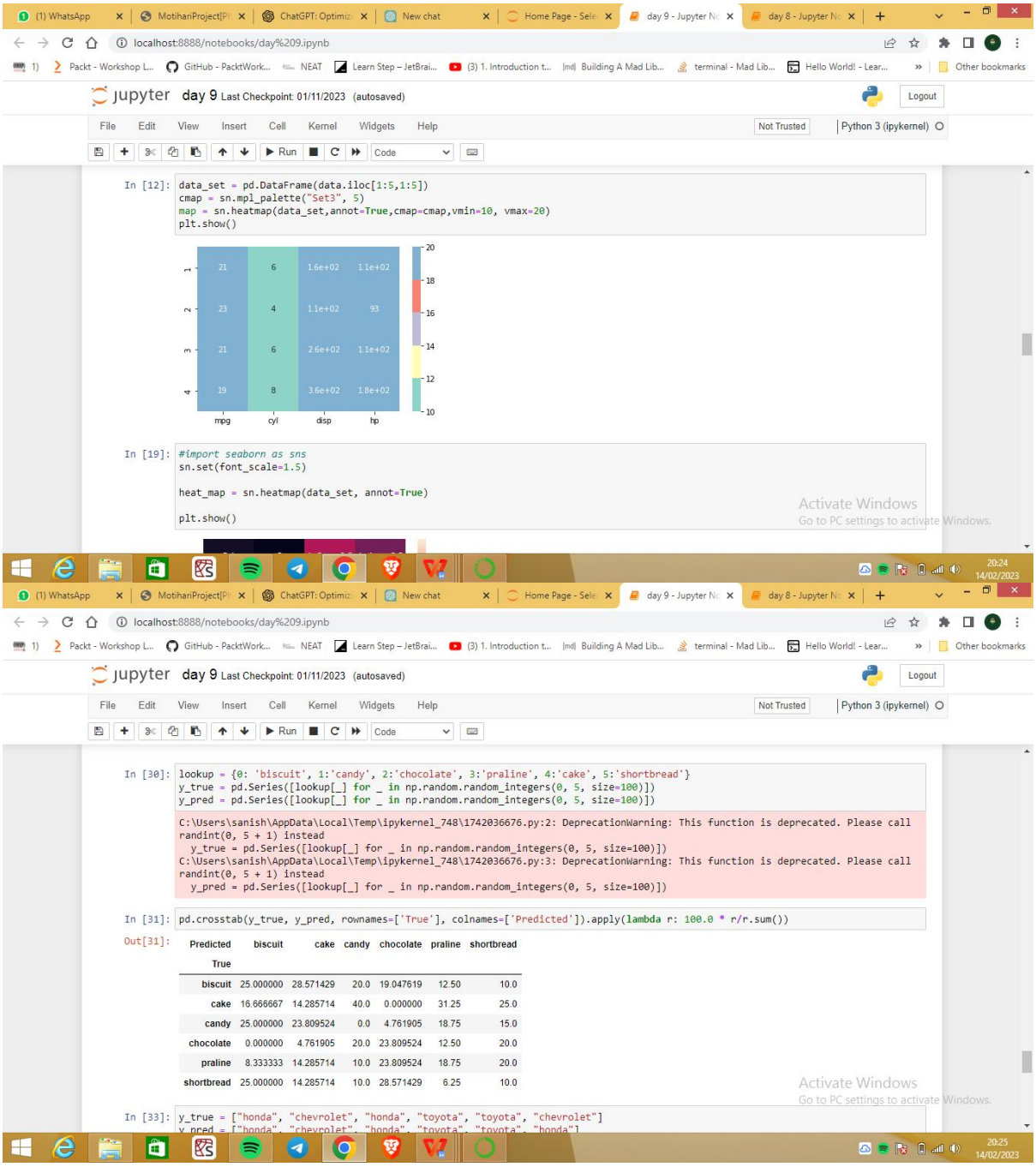

jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [4]:
map = sn.heatmap(data_plot)
plt.ylabel("Numbers")
plt.show()
Numbers
In [5]:
map = sn.heatmap(data_plot,annot=True)
plt.xlabel("Numbers")
plt.ylabel("Range")
Out[5]:
Text(33.0, 0.5, 'Range')
0.22 0.77 0.61 0.67 0.89
0.85 0.31 0.99 0.71 0.62
Activate Windows
Go to PC settings to activate Windows.
20:24
14/02/2023
jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [21]:
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = [1,0,0,1,0,0,1,0,0,1]
# predicted values
predicted = [1,0,0,1,0,0,0,1,0,0]
# confusion matrix
matrix = confusion_matrix(actual,predicted)
#, labels=[1,0])
print('Confusion matrix : \n',matrix)
Confusion matrix :
[[5 1]
[2 2]]
In [22]:
outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
Outcome values :
2 2 1 5
In [23]:
matrix = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n',matrix)
Classification report :
Activate Windows
Go to PC settings to activate Windows.
20:24
14/02/2023

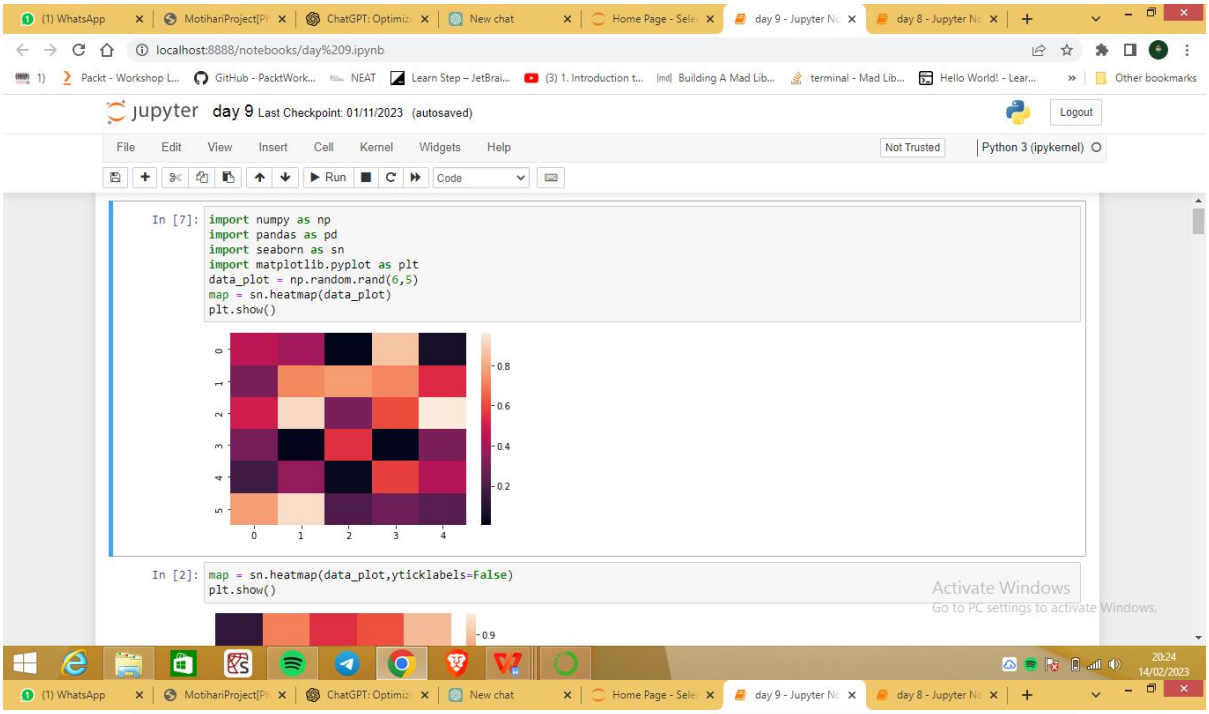

jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [10]:
data = pd.read_csv('mtcars.csv') # Load data set
print(data)
model mpg cyl disp hp drat wt qsec vs am \
0 Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 0 1
1 Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 0 1
2 Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 1 1
3 Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0
4 Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0
5 Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1 0
6 Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 0 0
7 Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 1 0
8 Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0
9 Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 1 0
10 Merc 280C 17.8 6 167.6 123 3.92 3.440 18.90 1 0
11 Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0
12 Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0
13 Merc 450SLC 15.2 8 275.8 180 3.07 3.780 18.00 0 0
14 Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0
15 Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0
16 Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0
17 Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1
18 Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1
19 Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1
20 Toyota Corona 21.5 4 120.1 97 3.70 2.465 20.01 1 0
21 Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0
22 AMC Javelin 15.2 8 304.0 150 3.15 3.435 17.30 0 0
23 Camaro Z28 13.3 8 350.0 245 3.73 3.840 15.41 0 0
24 Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0
Activate Windows
Go to PC settings to activate Windows.
20:24
14/02/2023

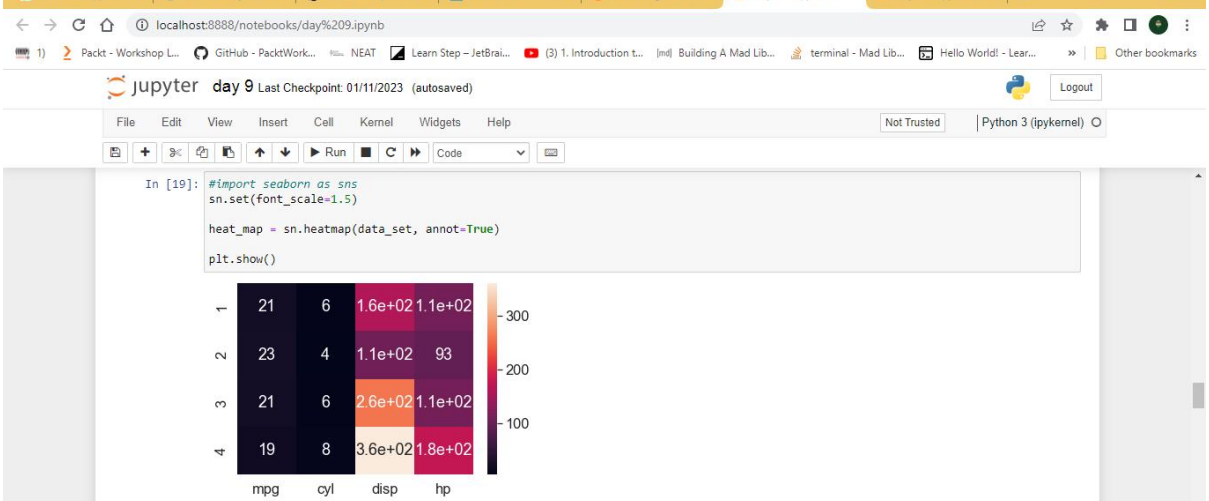

jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [23]:
matrix = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n',matrix)
Classification report :
precision recall f1-score support
1 0.67 0.50 0.57 4
0 0.71 0.83 0.77 6
accuracy 0.70 10
macro avg 0.69 0.67 0.67 10
weighted avg 0.70 0.70 0.69 10
In [24]:
from sklearn import metrics
# Constants
C="Cat"
F="Fish"
H="Hen"
# True values
y_true = [C,C,C,C,C, F,F,F,F,F,F,F,F,F, H,H,H,H,H,H,H,H,H]
# Predicted values
y_pred = [C,C,C,C,H,F, C,C,C,C,C,H,H,F,F, C,C,C,H,H,H,H,H,H]
In [ ]:
Activate Windows
Go to PC settings to activate Windows.
20:24
14/02/2023

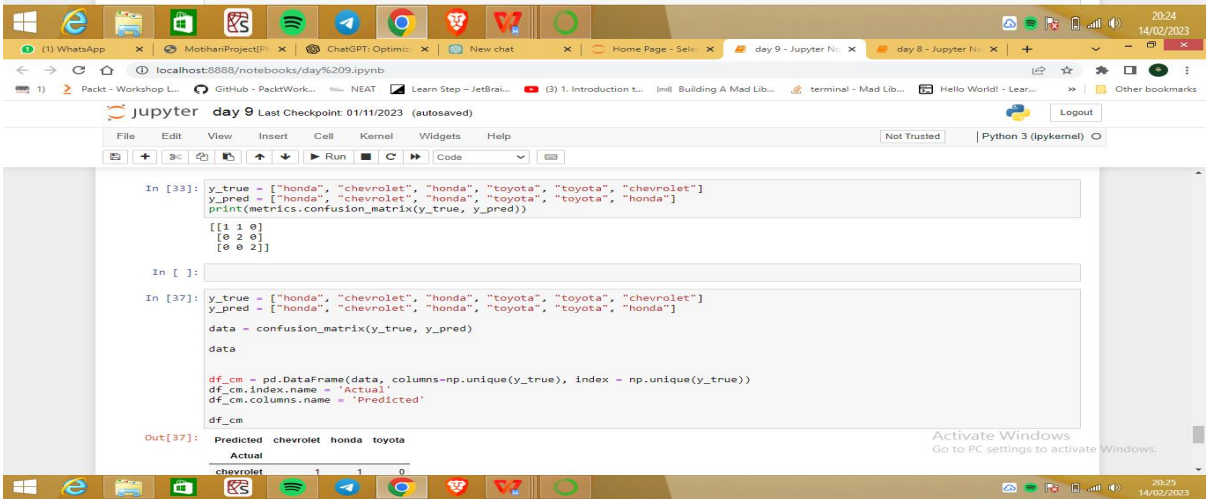

jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [9]:
import pandas as pd
data = pd.read_csv("mtcars.csv")
data_set = pd.DataFrame(data.iloc[1:5,1:5])
map = sn.heatmap(data_set,annot=True)
plt.show()
21 6 1.6e+02 1.1e+02
23 4 1.1e+02 93
21 6 2.6e+02 1.1e+02
19 8 3.6e+02 1.8e+02
1 2 3 4
mpg cyl disp hp
350 300 250 200 150 100 50
In [11]:
map = sn.heatmap(data_set,annot=True,cmap="cubehelix")
Activate Windows
Go to PC settings to activate Windows.
20:24
14/02/2023

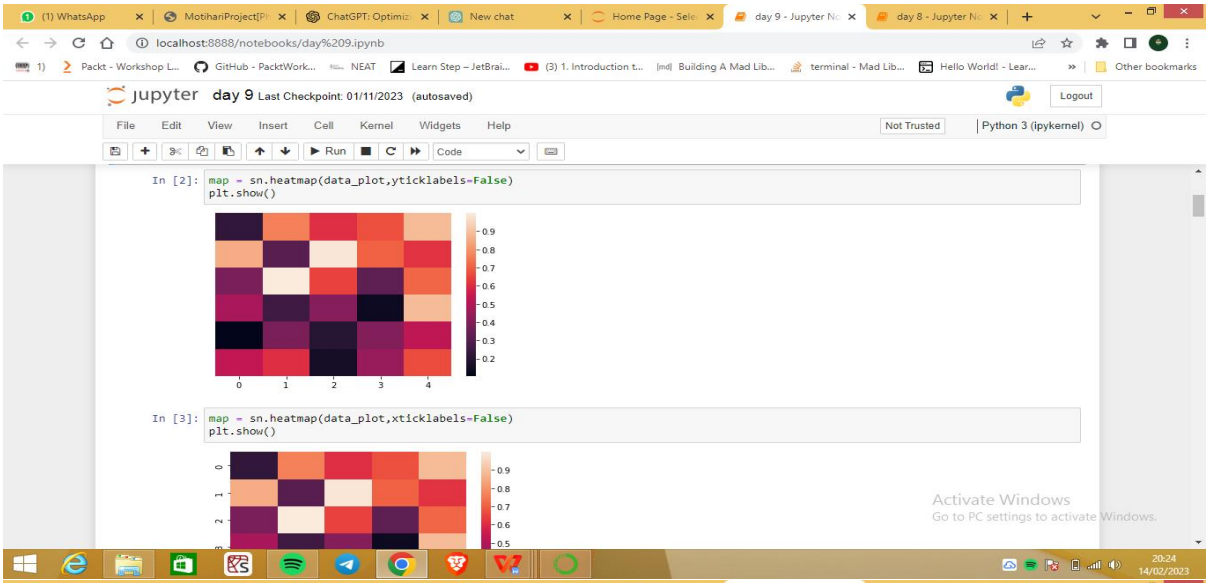
jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [26]:
print(metrics.confusion_matrix(y_true, y_pred))
[[4 1 1]
[6 2 2]
[3 0 6]]
In [27]:
print(metrics.classification_report(y_true, y_pred, digits=3))
precision recall f1-score support
Cat 0.308 0.667 0.421 6
Fish 0.667 0.200 0.308 10
Hen 0.667 0.667 0.667 9
accuracy 0.480 25
macro avg 0.547 0.511 0.465 25
weighted avg 0.581 0.480 0.464 25
In [28]:
import pandas as pd
y_true = pd.Series([2, 0, 2, 2, 0, 1, 1, 2, 2, 0, 1, 2])
y_pred = pd.Series([0, 0, 2, 1, 0, 2, 1, 0, 2, 0, 2, 2])
pd.crosstab(y_true, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
Out[28]:
Predicted 0 1 2 All
True
Activate Windows
Go to PC settings to activate Windows.
20:24
14/02/2023

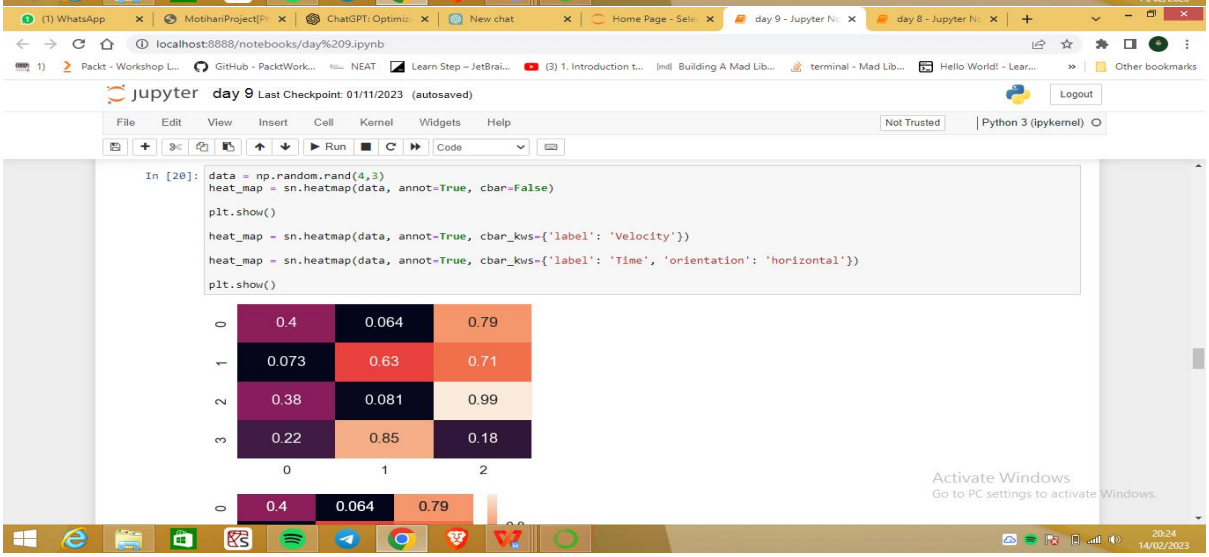

jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [12]:
data_set = pd.DataFrame(data.iloc[1:5,1:5])
cmap = sn.mpl_palette("Set3", 5)
map = sn.heatmap(data_set,annot=True,cmap=cmap,vmin=10, vmax=20)
plt.show()
mpg
cyl
disp
hp
In [19]:
#import seaborn as sns
sn.set(font_scale=1.5)
heat_map = sn.heatmap(data_set, annot=True)
plt.show()
Activate Windows
Go to PC settings to activate Windows.
In [30]:
lookup = {0: 'biscuit', 1:'candy', 2:'chocolate', 3:'praline', 4:'cake', 5:'shortbread'}
y_true = pd.Series([lookup[_] for _ in np.random.random_integers(0, 5, size=100)])
y_pred = pd.Series([lookup[_] for _ in np.random.random_integers(0, 5, size=100)])
C:\Users\sanish\AppData\Local\Temp\ipykernel_748\1742036676.py:2: DeprecationWarning: This function is deprecated. Please call randint(0, 5 + 1) instead
y_true = pd.Series([lookup[_] for _ in np.random.random_integers(0, 5, size=100)])
C:\Users\sanish\AppData\Local\Temp\ipykernel_748\1742036676.py:3: DeprecationWarning: This function is deprecated. Please call randint(0, 5 + 1) instead
y_pred = pd.Series([lookup[_] for _ in np.random.random_integers(0, 5, size=100)])
In [31]:
pd.crosstab(y_true, y_pred, rownames=['True'], colnames=['Predicted']).apply(lambda r: 100.0 * r/r.sum())
Out[31]:
Predicted biscuit cake candy chocolate praline shortbread
True
biscuit 25.000000 28.571429 20.0 19.047619 12.50 10.0
cake 16.666667 14.285714 40.0 0.000000 31.25 25.0
candy 25.000000 23.809524 0.0 4.761905 18.75 15.0
chocolate 0.000000 4.761905 20.0 23.809524 12.50 20.0
praline 8.333333 14.285714 10.0 23.809524 18.75 20.0
shortbread 25.000000 14.285714 10.0 28.571429 6.25 10.0
Activate Windows
Go to PC settings to activate Windows.
In [33]:
y_true = ["honda", "chevrolet", "honda", "toyota", "toyota", "chevrolet"]

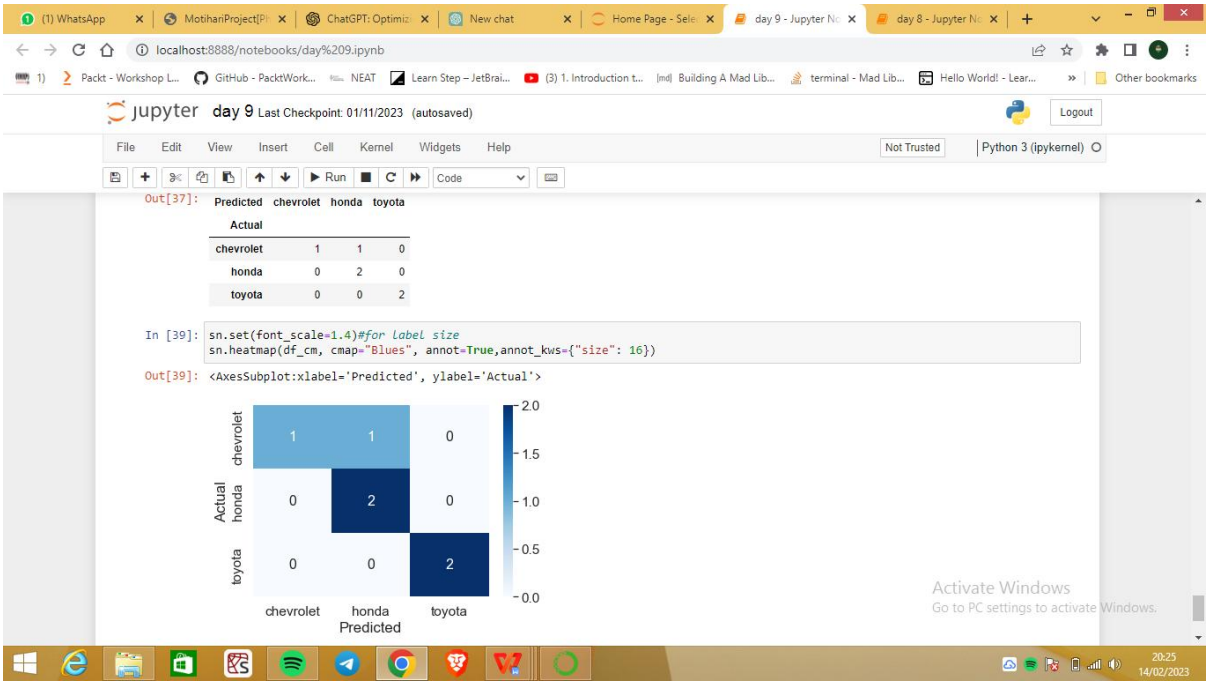

jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [7]:
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
data_plot = np.random.rand(6,5)
map = sn.heatmap(data_plot)
plt.show()
In [2]:
map = sn.heatmap(data_plot,yticklabels=False)
plt.show()
Activate Windows
Go to PC settings to activate Windows.


jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [19]:
#import seaborn as sns
sn.set(font_scale=1.5)
heat_map = sn.heatmap(data_set, annot=True)
plt.show()
21 6 1.6e+02 1.1e+02
23 4 1.1e+02 93
21 6 2.6e+02 1.1e+02
19 8 3.6e+02 1.8e+02
mpg cyl disp hp
In [20]:
data = np.random.rand(4,3)
heat_map = sn.heatmap(data, annot=True, cbar=False)
plt.show()
heat_map = sn.heatmap(data, annot=True, cbar_kws={'label': 'Velocity'})
Activate Windows
Go to PC settings to activate Windows.

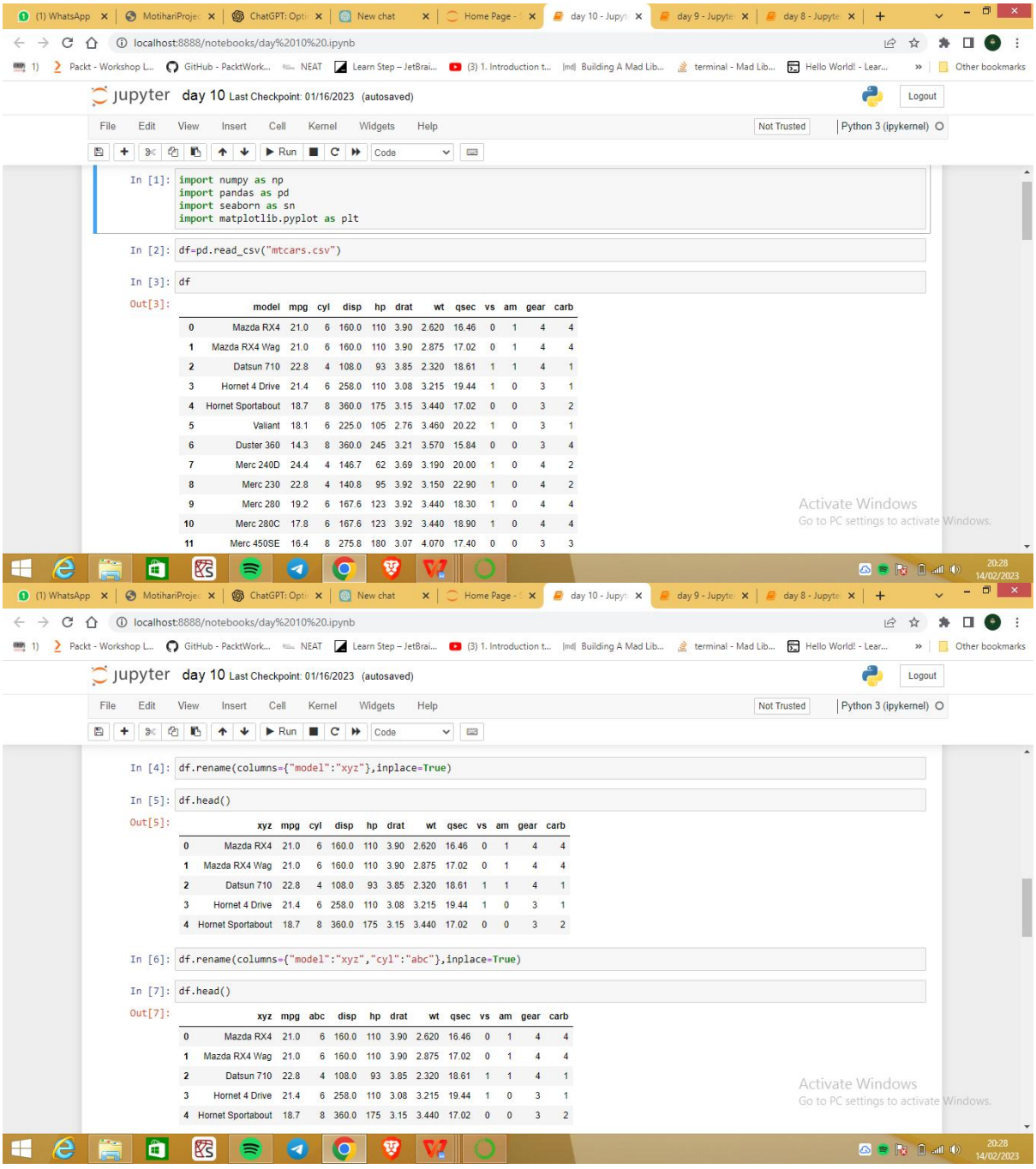

jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [33]:
y_true = ["honda", "chevrolet", "honda", "toyota", "toyota", "chevrolet"]
y_pred = ["honda", "chevrolet", "honda", "toyota", "toyota", "honda"]
print(metrics.confusion_matrix(y_true, y_pred))
[[1 1 0]
[0 2 0]
[0 0 2]]
In [ ]:
In [37]:
y_true = ["honda", "chevrolet", "honda", "toyota", "toyota", "chevrolet"]
y_pred = ["honda", "chevrolet", "honda", "toyota", "toyota", "honda"]
data = confusion_matrix(y_true, y_pred)
data
df_cm = pd.DataFrame(data, columns=np.unique(y_true), index = np.unique(y_true))
df_cm.index.name = 'Actual'
df_cm.columns.name = 'Predicted'
df_cm
Out[37]:
Predicted chevrolet honda toyota
Actual
chevrolet 1 1 0
Activate Windows
Go to PC settings to activate Windows.


jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [2]:
map = sn.heatmap(data_plot,yticklabels=False)
plt.show()
In [3]:
map = sn.heatmap(data_plot,xticklabels=False)
plt.show()
Activate Windows
Go to PC settings to activate Windows.


jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [20]:
data = np.random.rand(4,3)
heat_map = sn.heatmap(data, annot=True, cbar=False)
plt.show()
heat_map = sn.heatmap(data, annot=True, cbar_kws={'label': 'Velocity'})
heat_map = sn.heatmap(data, annot=True, cbar_kws={'label': 'Time', 'orientation': 'horizontal'})
plt.show()
0.4 0.064 0.79
0.073 0.63 0.71
0.38 0.081 0.99
0.22 0.85 0.18
Activate Windows
Go to PC settings to activate Windows.


jupyter day 9 Last Checkpoint: 01/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Out[37]:
Predicted chevrolet honda toyota
Actual
chevrolet 1 1 0
honda 0 2 0
toyota 0 0 2
In [39]:
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, cmap="Blues", annot=True,annot_kws={"size": 16})
Out[39]:
<AxesSubplot:xlabel='Predicted', ylabel='Actual'>
chevrolet
honda
toyota
Actual
Predicted
Activate Windows
Go to PC settings to activate Windows.

CLASS 10



jupyter day 10 Last Checkpoint: 01/16/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv("mtcars.csv")
```

In [3]:
```
df
```

Out[3]:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
5	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
6	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
7	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
8	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
9	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
10	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
11	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3

Activate Windows
Go to PC settings to activate Windows.

jupyter day 10 Last Checkpoint: 01/16/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [4]:
```
df.rename(columns={"model":"xyz"},inplace=True)
```

In [5]:
```
df.head()
```

Out[5]:

	xyz	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

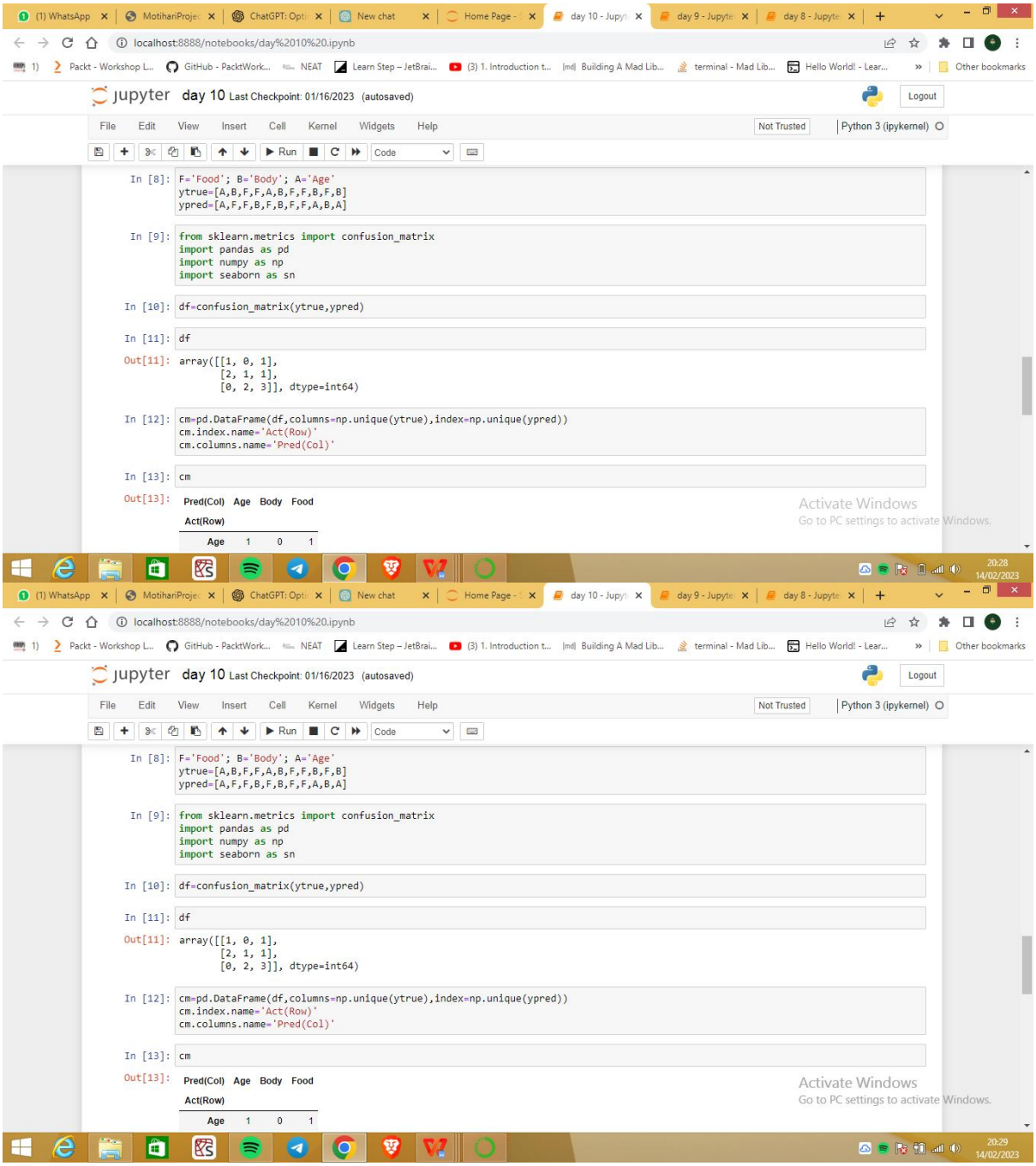
In [6]:
```
df.rename(columns={"model":"xyz","cyl":"abc"},inplace=True)
```

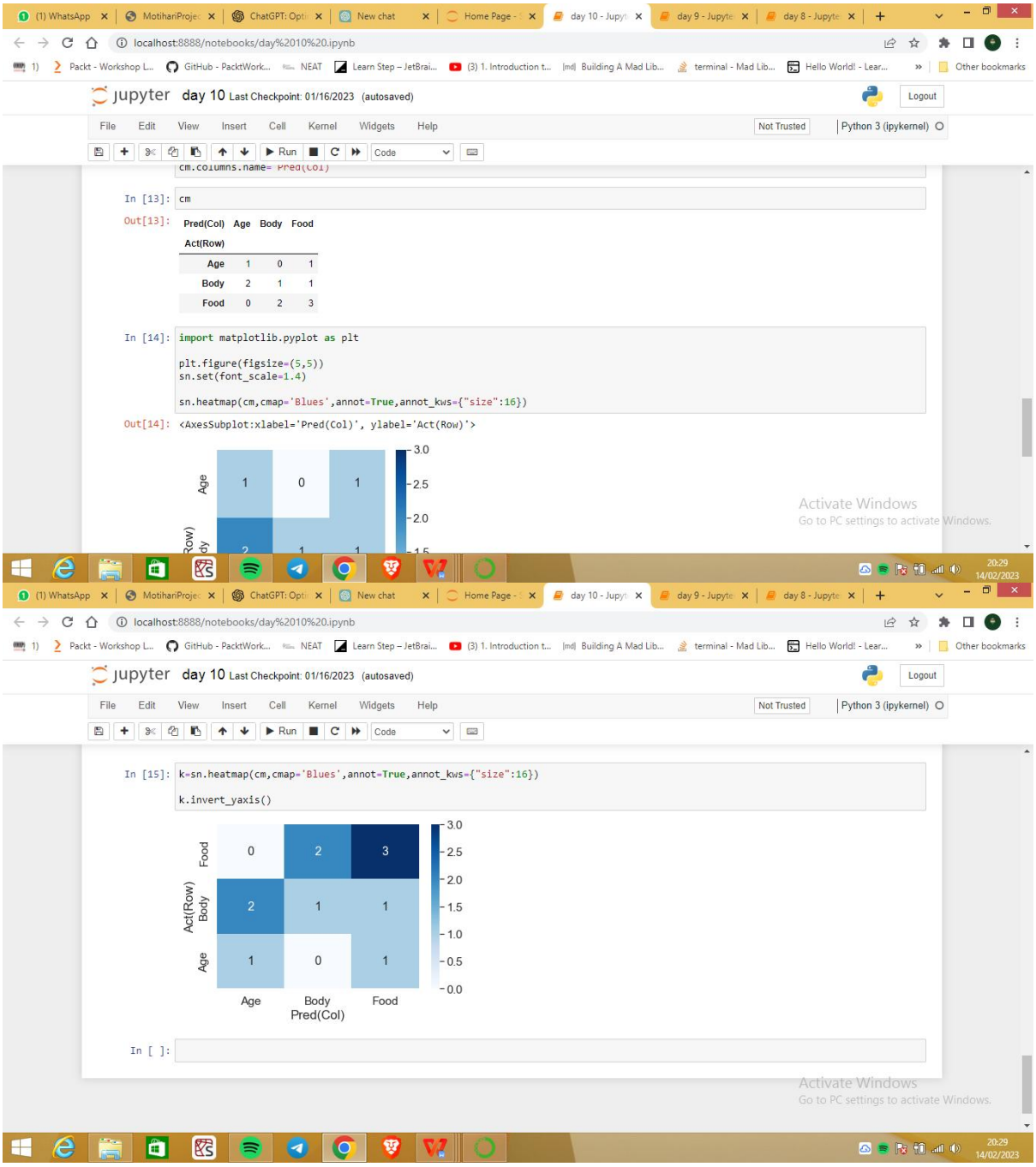
In [7]:
```
df.head()
```

Out[7]:

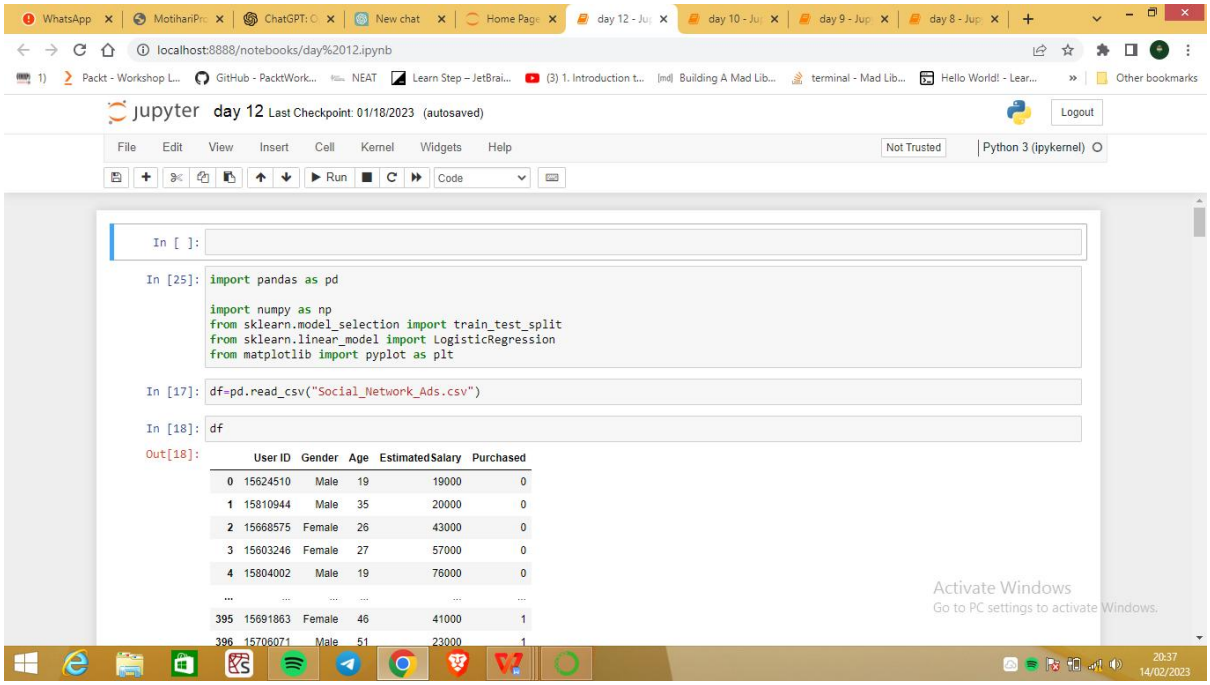
	xyz	mpg	abc	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

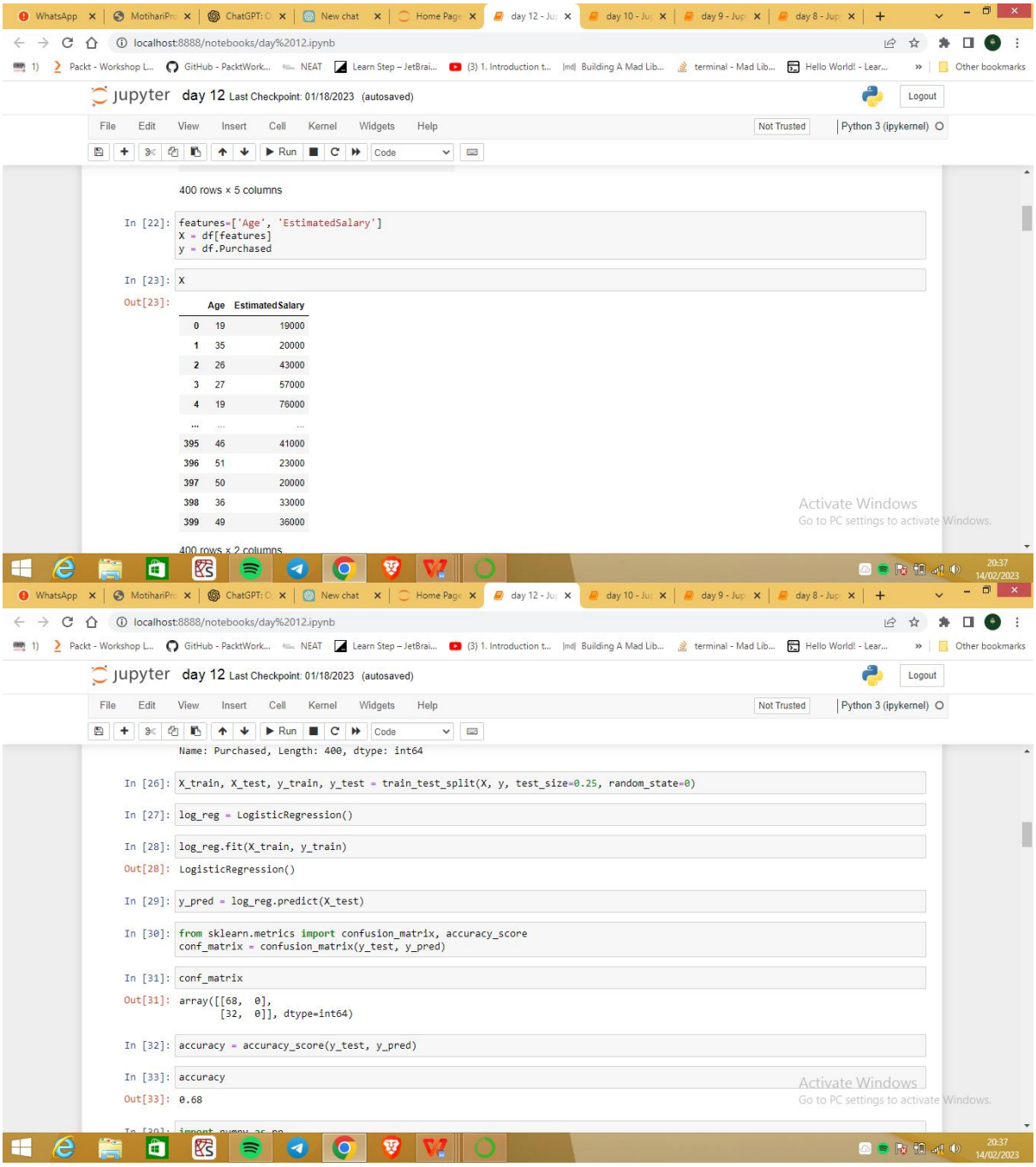
Activate Windows
Go to PC settings to activate Windows.

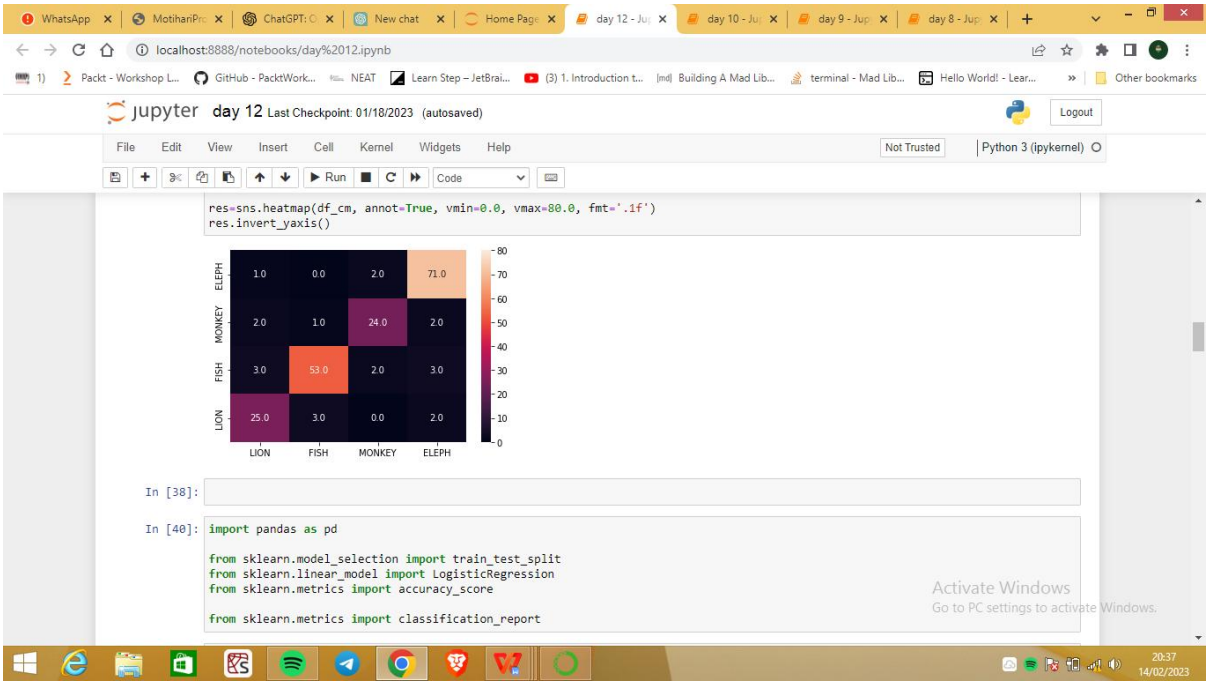

jupyter day 10 Last Checkpoint: 01/16/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [8]:
F='Food'; B='Body'; A='Age'
ytrue=[A,B,F,F,A,B,F,F,B,F,B]
ypred=[A,F,F,B,F,B,F,F,A,B,A]
In [9]:
from sklearn.metrics import confusion_matrix
import pandas as pd
import numpy as np
import seaborn as sn
In [10]:
df=confusion_matrix(ytrue,ypred)
In [11]:
df
Out[11]:
array([[1, 0, 1],
[2, 1, 1],
[0, 2, 3]], dtype=int64)
In [12]:
cm=pd.DataFrame(df,columns=np.unique(ytrue),index=np.unique(ypred))
cm.index.name='Act(Row)'
cm.columns.name='Pred(Col)'
In [13]:
cm
Out[13]:
Pred(Col) Age Body Food
Act(Row)
Age 1 0 1
Activate Windows
Go to PC settings to activate Windows.
20:28
14/02/2023
jupyter day 10 Last Checkpoint: 01/16/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [8]:
F='Food'; B='Body'; A='Age'
ytrue=[A,B,F,F,A,B,F,F,B,F,B]
ypred=[A,F,F,B,F,B,F,F,A,B,A]
In [9]:
from sklearn.metrics import confusion_matrix
import pandas as pd
import numpy as np
import seaborn as sn
In [10]:
df=confusion_matrix(ytrue,ypred)
In [11]:
df
Out[11]:
array([[1, 0, 1],
[2, 1, 1],
[0, 2, 3]], dtype=int64)
In [12]:
cm=pd.DataFrame(df,columns=np.unique(ytrue),index=np.unique(ypred))
cm.index.name='Act(Row)'
cm.columns.name='Pred(Col)'
In [13]:
cm
Out[13]:
Pred(Col) Age Body Food
Act(Row)
Age 1 0 1
Activate Windows
Go to PC settings to activate Windows.
20:29
14/02/2023

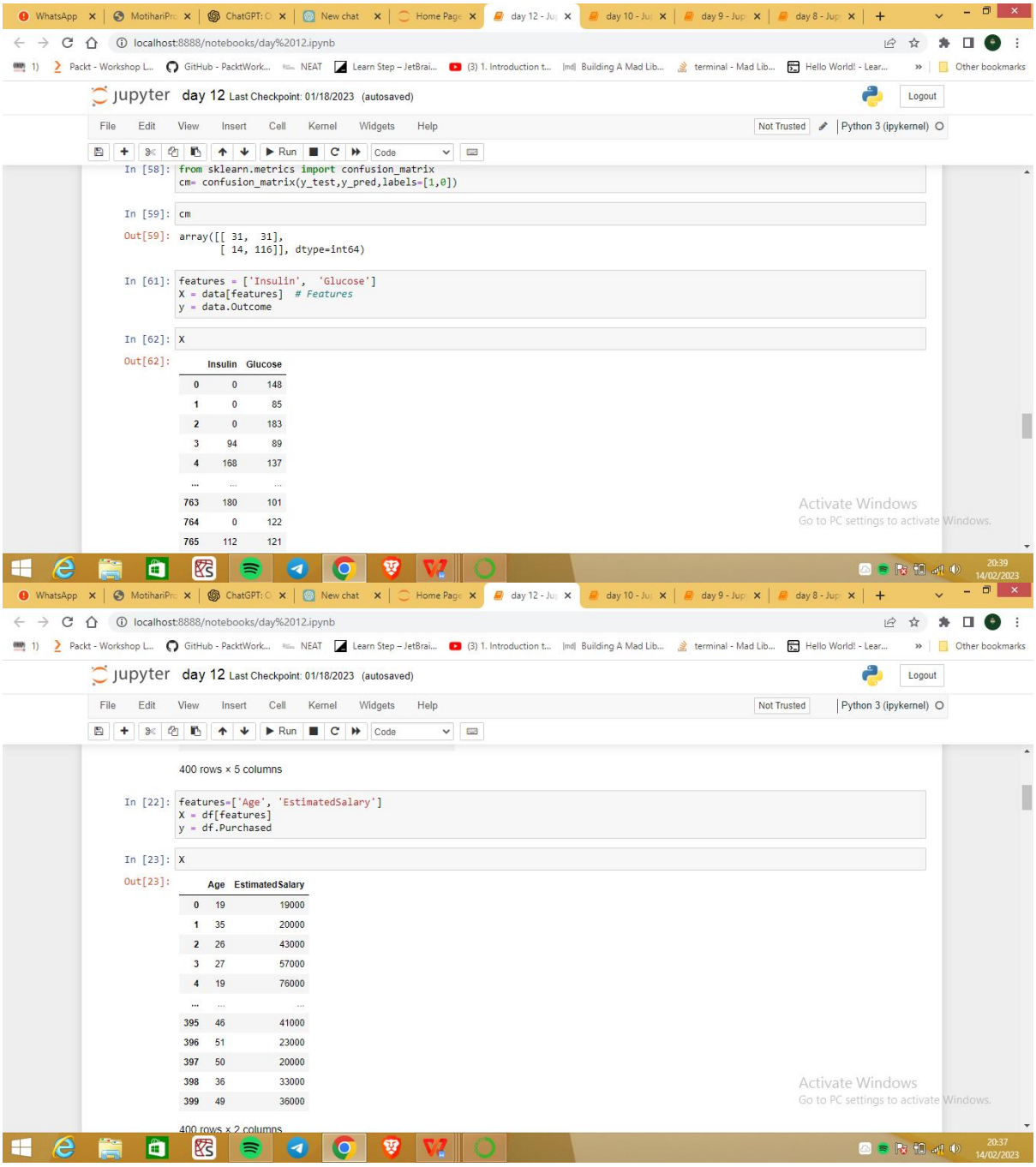

jupyter day 10 Last Checkpoint: 01/16/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [13]: cm
Out[13]:
Pred(Col) Age Body Food
Act(Row)
Age 1 0 1
Body 2 1 1
Food 0 2 3
In [14]: import matplotlib.pyplot as plt
plt.figure(figsize=(5,5))
sn.set(font_scale=1.4)
sn.heatmap(cm,cmap='Blues',annot=True,annot_kws={"size":16})
Out[14]: <AxesSubplot:xlabel='Pred(Col)', ylabel='Act(Row)'>
Age
1 0 1
3.0
2.5
2.0
Activate Windows
Go to PC settings to activate Windows.
20:29
14/02/2023
jupyter day 10 Last Checkpoint: 01/16/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [15]: k=sn.heatmap(cm,cmap='Blues',annot=True,annot_kws={"size":16})
k.invert_yaxis()
Food
Act(Row) Body
Age
0 2 3
2 1 1
1 0 1
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Age Body Food
Pred(Col)
In [ ]:
Activate Windows
Go to PC settings to activate Windows.
20:29
14/02/2023

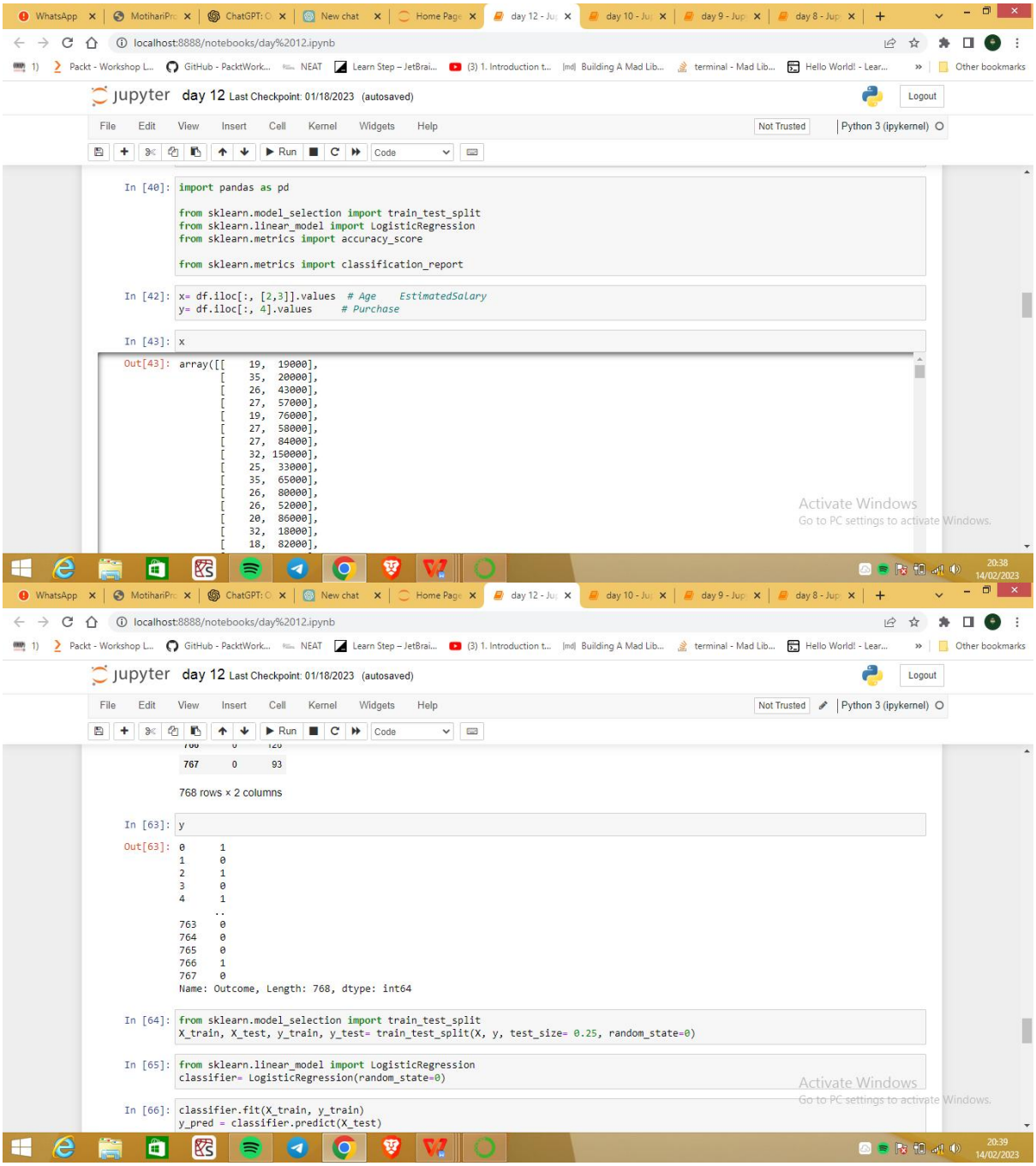
CLASS 11 & 12

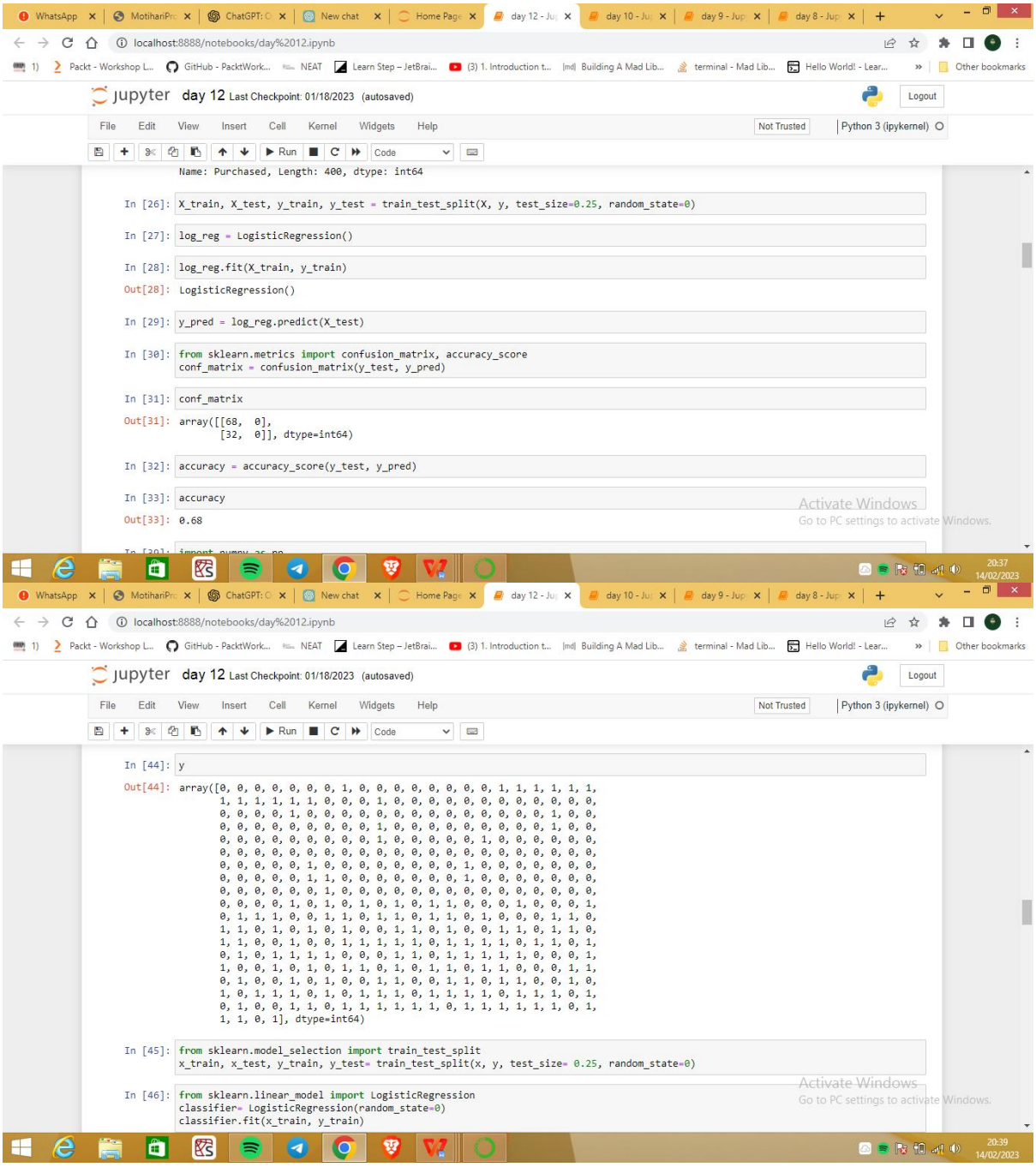

jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [ ]:
In [25]:
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from matplotlib import pyplot as plt
In [17]:
df=pd.read_csv("Social_Network_Ads.csv")
In [18]:
df
Out[18]:
User ID Gender Age EstimatedSalary Purchased
0 15624510 Male 19 19000 0
1 15810944 Male 35 20000 0
2 15668575 Female 26 43000 0
3 15603246 Female 27 57000 0
4 15804002 Male 19 76000 0
395 15691863 Female 46 41000 1
Activate Windows
Go to PC settings to activate Windows.
20:37
14/02/2023


jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
400 rows × 5 columns
In [22]:
features=['Age', 'EstimatedSalary']
X = df[features]
y = df.Purchased
In [23]:
X
Out[23]:
Age EstimatedSalary
0 19 19000
1 35 20000
2 26 43000
3 27 57000
4 19 76000
... ...
395 46 41000
396 51 23000
397 50 20000
398 36 33000
399 49 36000
400 rows × 2 columns
Activate Windows
Go to PC settings to activate Windows.
20:37
14/02/2023
jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Name: Purchased, Length: 400, dtype: int64
In [26]:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
In [27]:
log_reg = LogisticRegression()
In [28]:
log_reg.fit(X_train, y_train)
Out[28]:
LogisticRegression()
In [29]:
y_pred = log_reg.predict(X_test)
In [30]:
from sklearn.metrics import confusion_matrix, accuracy_score
conf_matrix = confusion_matrix(y_test, y_pred)
In [31]:
conf_matrix
Out[31]:
array([[68, 0],
[32, 0]], dtype=int64)
In [32]:
accuracy = accuracy_score(y_test, y_pred)
In [33]:
accuracy
Out[33]:
0.68
Activate Windows
Go to PC settings to activate Windows.
20:37
14/02/2023

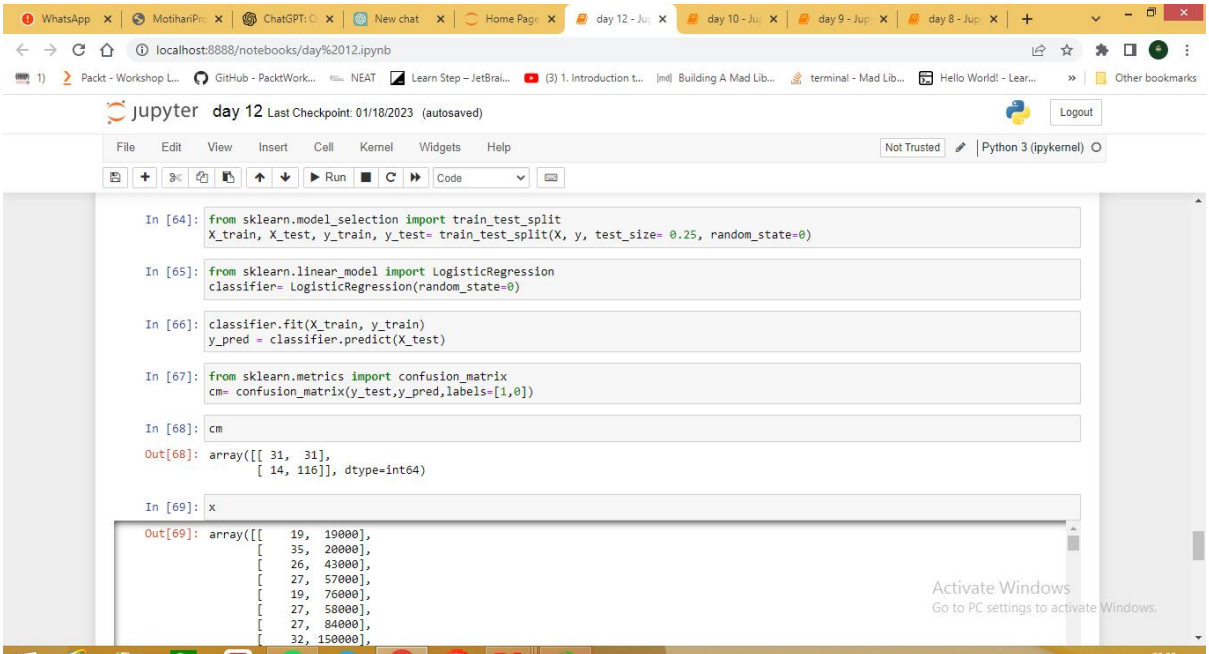

jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
res=sns.heatmap(df_cm, annot=True, vmin=0.0, vmax=80.0, fmt='.1f')
res.invert_yaxis()
ELEPH
MONKEY
FISH
LION
1.0 0.0 2.0 71.0
2.0 1.0 24.0 2.0
3.0 53.0 2.0 3.0
25.0 3.0 0.0 2.0
LION FISH MONKEY ELEPH
80 70 60 50 40 30 20 10 0
In [38]:
In [40]:
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
Activate Windows
Go to PC settings to activate Windows.
20:37
14/02/2023

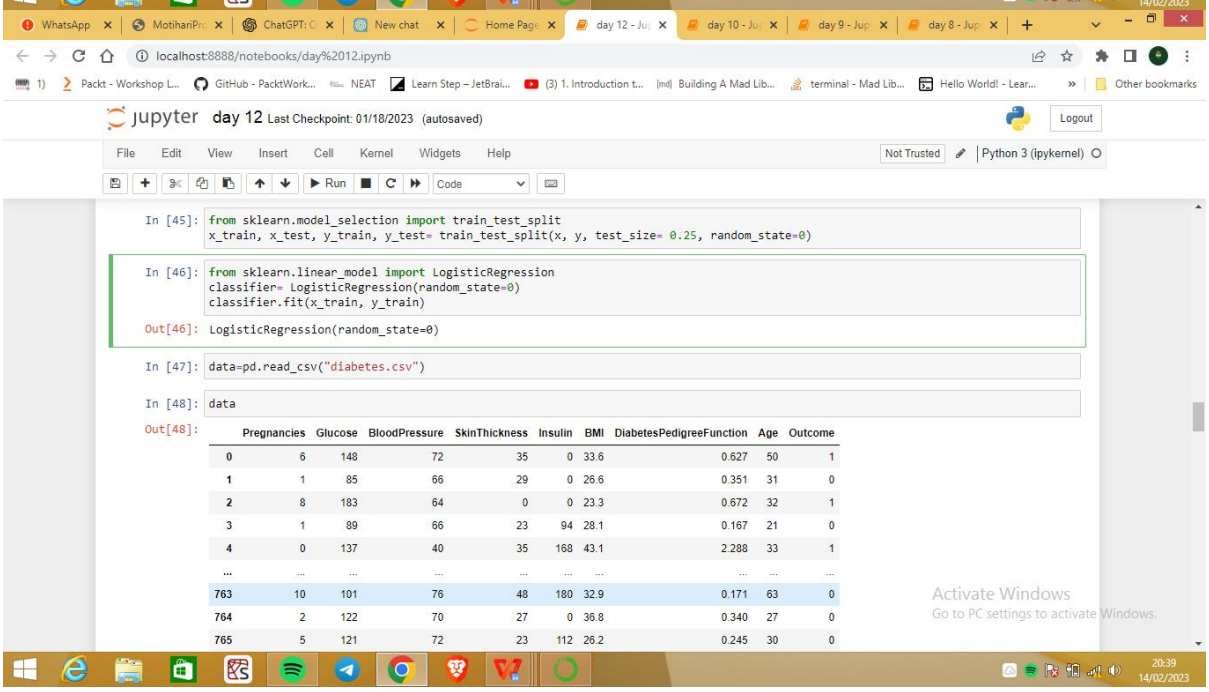

jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [58]:
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred,labels=[1,0])
In [59]:
cm
Out[59]:
array([[31, 31],
[14, 116]], dtype=int64)
In [61]:
features = ['Insulin', 'Glucose']
X = data[features] # Features
y = data.Outcome
In [62]:
X
Out[62]:
Insulin Glucose
0 0 148
1 0 85
2 0 183
3 94 89
4 168 137
... ...
763 180 101
764 0 122
765 112 121
Activate Windows
Go to PC settings to activate Windows.
jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
400 rows × 5 columns
In [22]:
features=['Age', 'EstimatedSalary']
X = df[features]
y = df.Purchased
In [23]:
X
Out[23]:
Age EstimatedSalary
0 19 19000
1 35 20000
2 26 43000
3 27 57000
4 19 76000
... ...
395 46 41000
396 51 23000
397 50 20000
398 36 33000
399 49 36000
400 rows × 2 columns
Activate Windows
Go to PC settings to activate Windows.

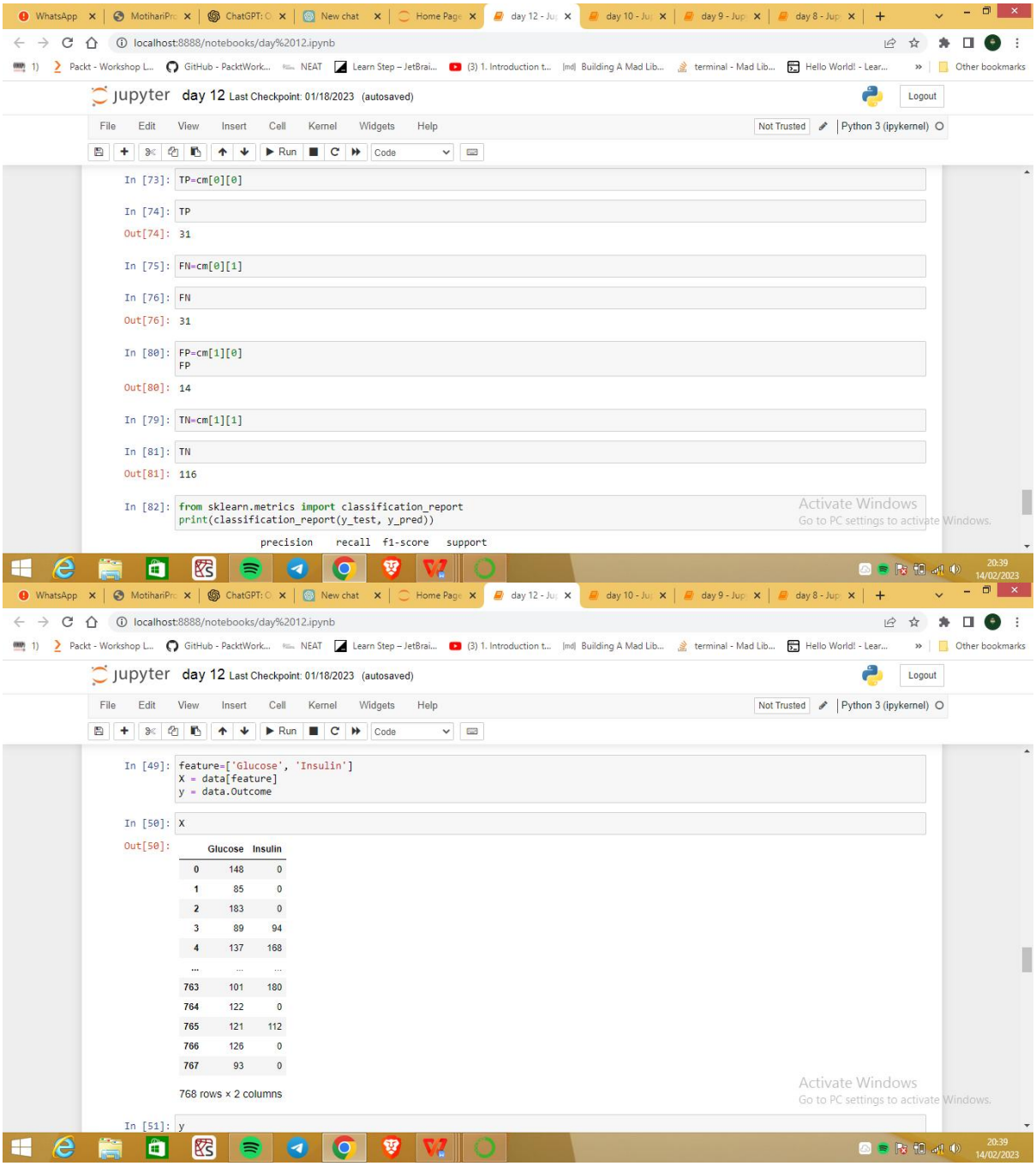

jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [40]:
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
In [42]:
x= df.iloc[:, [2,3]].values # Age EstimatedSalary
y= df.iloc[:, 4].values # Purchase
In [43]:
x
Out[43]:
array([[19, 19000],
[35, 20000],
[26, 43000],
[27, 57000],
[19, 76000],
[27, 58000],
[27, 84000],
[32, 150000],
[25, 33000],
[35, 65000],
[26, 80000],
[26, 52000],
[20, 86000],
[32, 18000],
[18, 82000],
Activate Windows
Go to PC settings to activate Windows.
20:38
14/02/2023
jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
767 0 93
768 rows × 2 columns
In [63]:
y
Out[63]:
0 1
1 0
2 1
3 0
4 1
..
763 0
764 0
765 0
766 1
767 0
Name: Outcome, Length: 768, dtype: int64
In [64]:
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)
In [65]:
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
In [66]:
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
Activate Windows
Go to PC settings to activate Windows.
20:39
14/02/2023

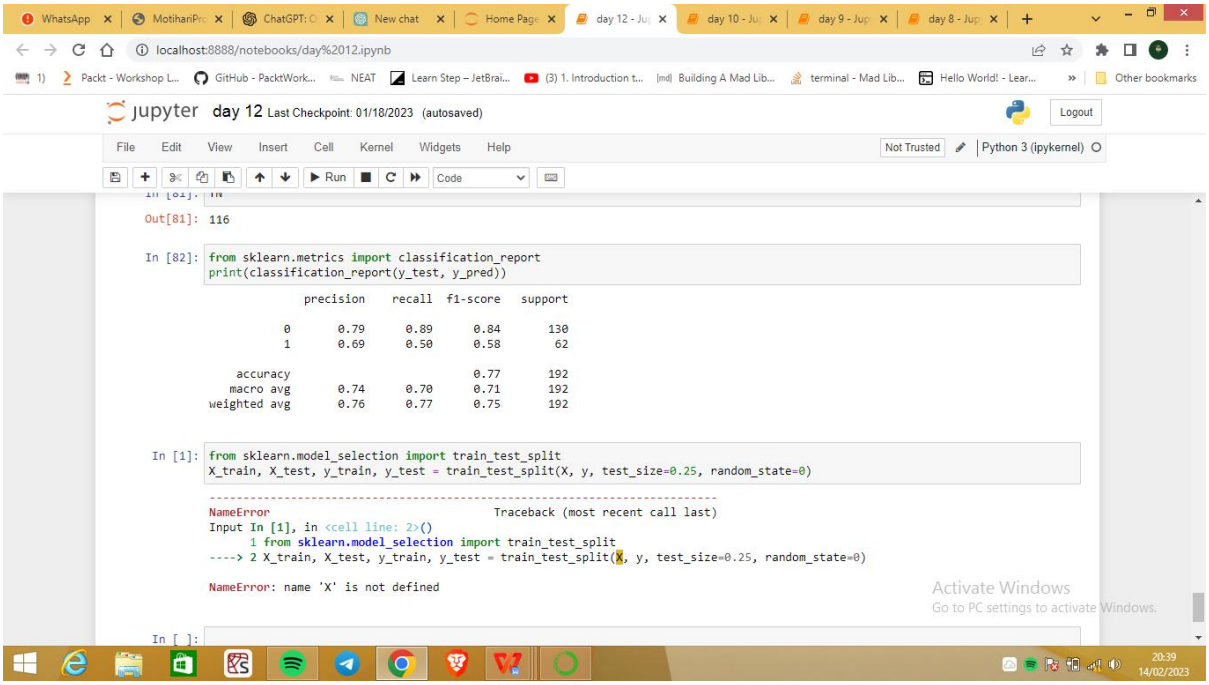

jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Name: Purchased, Length: 400, dtype: int64
In [26]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
In [27]: log_reg = LogisticRegression()
In [28]: log_reg.fit(X_train, y_train)
Out[28]: LogisticRegression()
In [29]: y_pred = log_reg.predict(X_test)
In [30]: from sklearn.metrics import confusion_matrix, accuracy_score
conf_matrix = confusion_matrix(y_test, y_pred)
In [31]: conf_matrix
Out[31]: array([[68, 0],
[32, 0]], dtype=int64)
In [32]: accuracy = accuracy_score(y_test, y_pred)
In [33]: accuracy
Out[33]: 0.68
Activate Windows
Go to PC settings to activate Windows.
20:37
14/02/2023

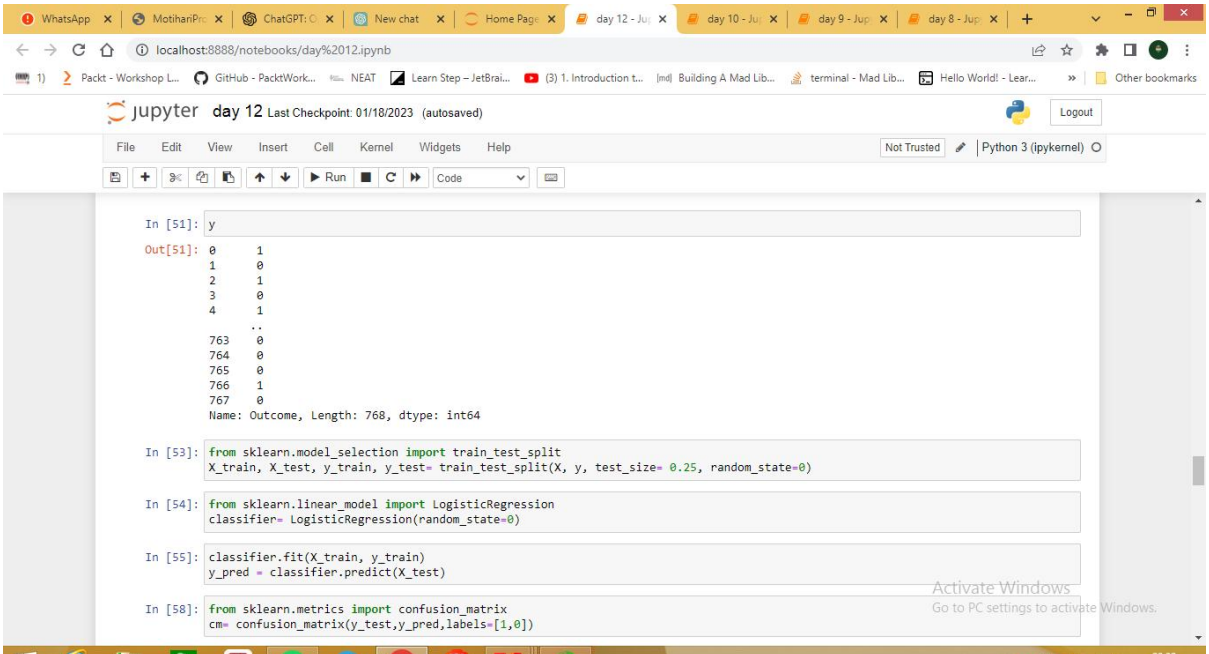
jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [44]: y
Out[44]: array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1,
1, 1, 0, 1], dtype=int64)
In [45]: 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)
In [46]: from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)
Activate Windows
Go to PC settings to activate Windows.
20:39
14/02/2023

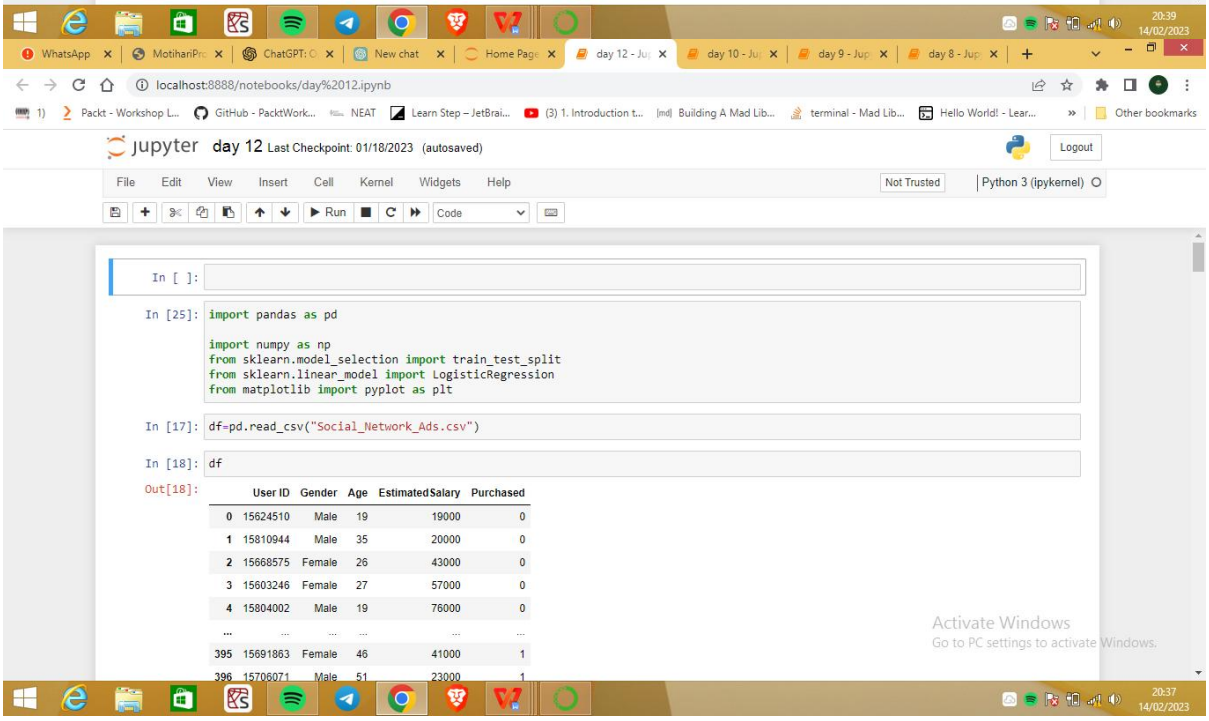

jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [64]:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(X, y, test_size= 0.25, random_state=0)
In [65]:
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
In [66]:
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
In [67]:
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred,labels=[1,0])
In [68]:
cm
Out[68]:
array([[31, 31],
[14, 116]], dtype=int64)
In [69]:
x
Out[69]:
array([[19, 19000],
[35, 20000],
[26, 43000],
[27, 57000],
[19, 76000],
[27, 58000],
[27, 84000],
[32, 150000],
Activate Windows
Go to PC settings to activate Windows.


jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [45]:
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)
In [46]:
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)
Out[46]:
LogisticRegression(random_state=0)
In [47]:
data=pd.read_csv("diabetes.csv")
In [48]:
data
Out[48]:
Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
0 6 148 72 35 0 33.6 0.627 50 1
1 1 85 66 29 0 26.6 0.351 31 0
2 8 183 64 0 0 23.3 0.672 32 1
3 1 89 66 23 94 28.1 0.167 21 0
4 0 137 40 35 168 43.1 2.288 33 1
763 10 101 76 48 180 32.9 0.171 63 0
764 2 122 70 27 0 36.8 0.340 27 0
765 5 121 72 23 112 26.2 0.245 30 0
Activate Windows
Go to PC settings to activate Windows.

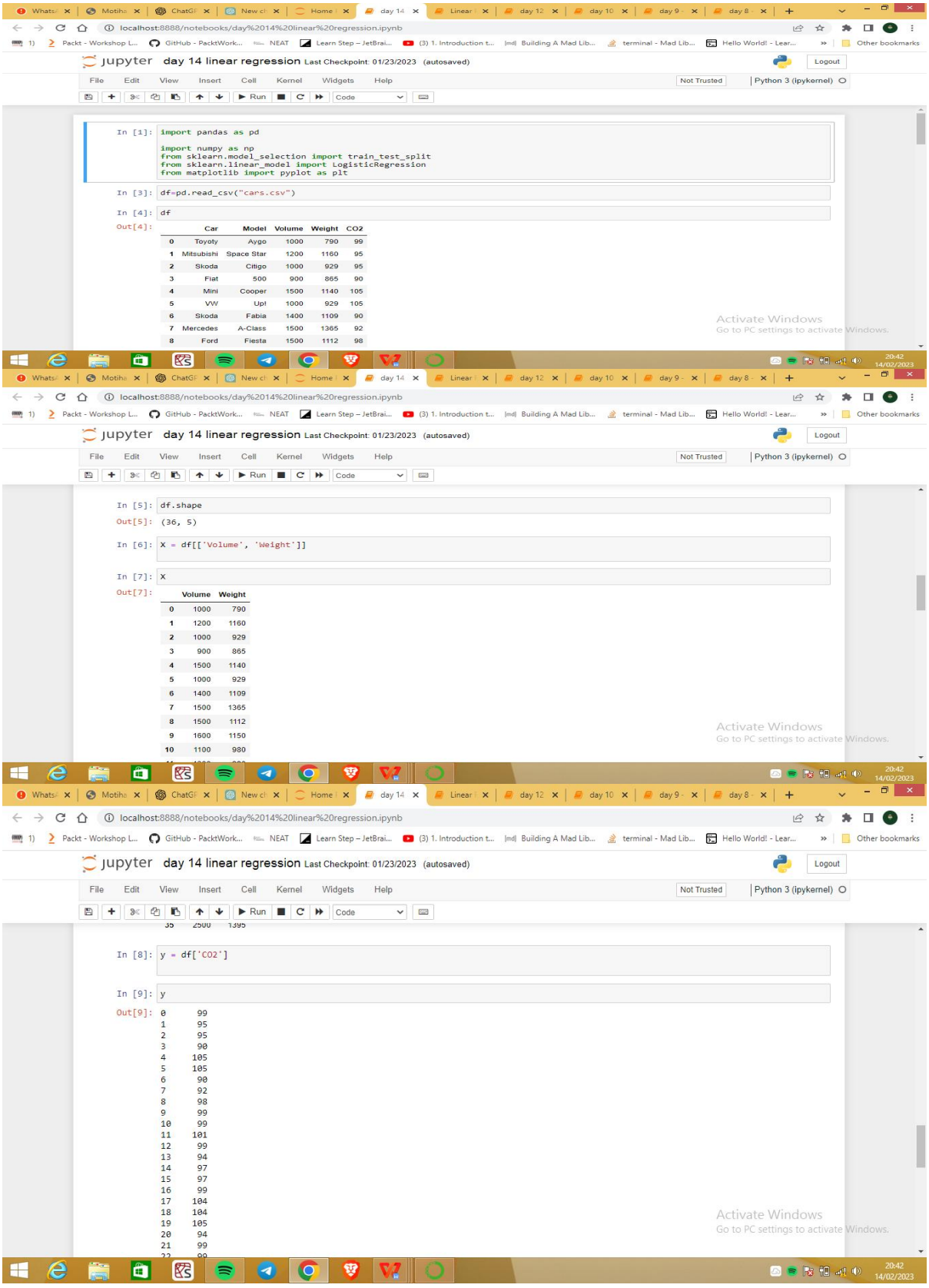

jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [73]: TP=cm[0][0]
In [74]: TP
Out[74]: 31
In [75]: FN=cm[0][1]
In [76]: FN
Out[76]: 31
In [80]: FP=cm[1][0]
FP
Out[80]: 14
In [79]: TN=cm[1][1]
In [81]: TN
Out[81]: 116
In [82]: from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
precision recall f1-score support
Activate Windows
Go to PC settings to activate Windows.
20:39
14/02/2023
jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [49]: feature=['Glucose', 'Insulin']
X = data[feature]
y = data.Outcome
In [50]: X
Out[50]:
Glucose Insulin
0 148 0
1 85 0
2 183 0
3 89 94
4 137 168
... ...
763 101 180
764 122 0
765 121 112
766 126 0
767 93 0
768 rows × 2 columns
Activate Windows
Go to PC settings to activate Windows.
In [51]: y
20:39
14/02/2023

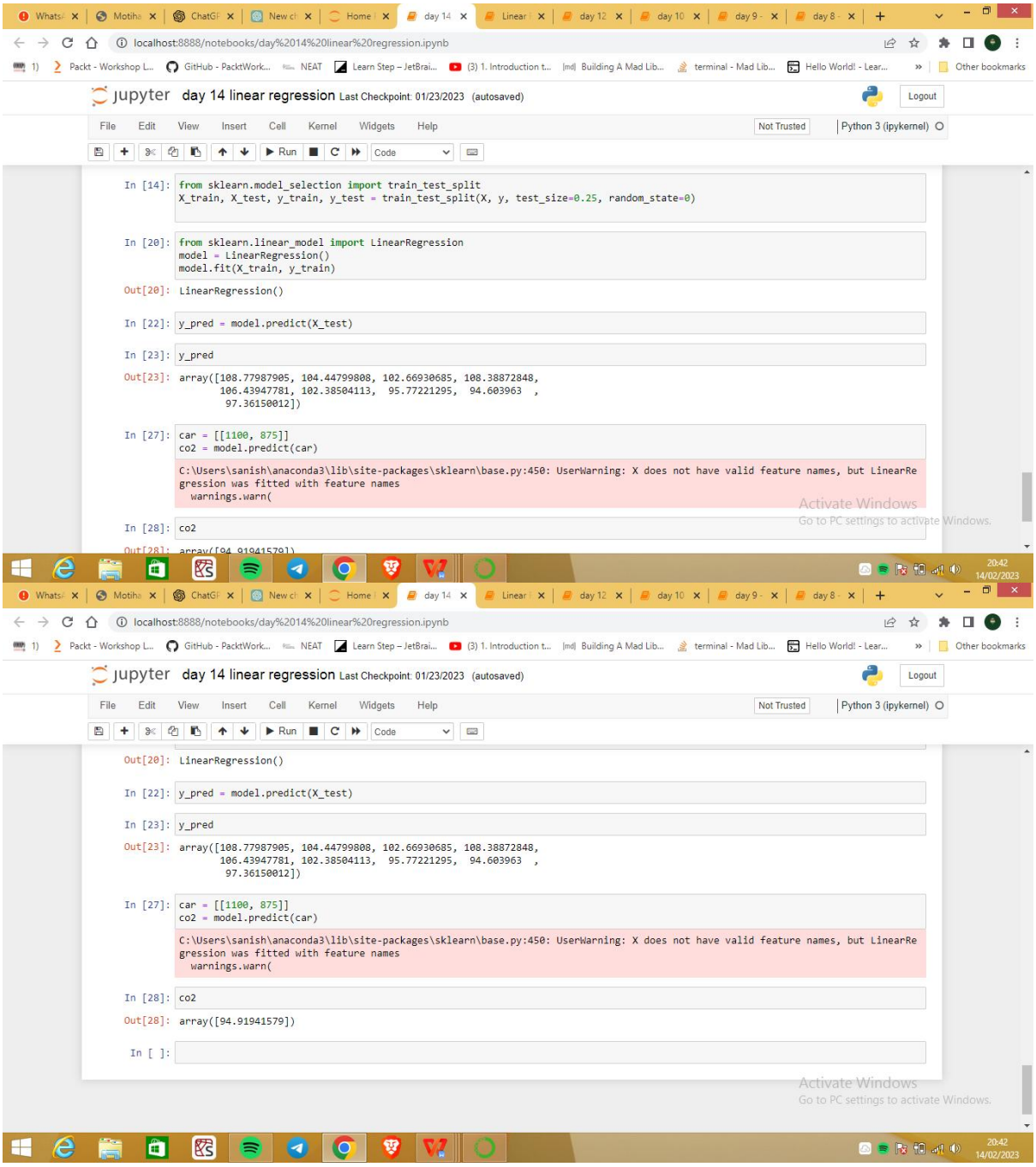

localhost:8888/notebooks/day%2012.ipynb
jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Out[81]: 116
In [82]:
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
precision recall f1-score support
0 0.79 0.89 0.84 130
1 0.69 0.50 0.58 62
accuracy 0.77 192
macro avg 0.74 0.70 0.71 192
weighted avg 0.76 0.77 0.75 192
In [1]:
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)
NameError Traceback (most recent call last)
Input In [1], in <cell line: 2>()
1 from sklearn.model_selection import train_test_split
----> 2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
NameError: name 'X' is not defined
Activate Windows
Go to PC settings to activate Windows.
In [ ]:
20:39
14/02/2023


jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [51]:
y
Out[51]:
0 1
1 0
2 1
3 0
4 1
..
763 0
764 0
765 0
766 1
767 0
Name: Outcome, Length: 768, dtype: int64
In [53]:
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)
In [54]:
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
In [55]:
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
In [58]:
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred,labels=[1,0])
Activate Windows
Go to PC settings to activate Windows.


jupyter day 12 Last Checkpoint: 01/18/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [ ]:
In [25]:
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from matplotlib import pyplot as plt
In [17]:
df=pd.read_csv("Social_Network_Ads.csv")
In [18]:
df
Out[18]:
User ID Gender Age EstimatedSalary Purchased
0 15624510 Male 19 19000 0
1 15810944 Male 35 20000 0
2 15668575 Female 26 43000 0
3 15603246 Female 27 57000 0
4 15804002 Male 19 76000 0
... ...
395 15691863 Female 46 41000 1
396 15706071 Male 51 23000 1
Activate Windows
Go to PC settings to activate Windows.

CLASS 13 & 14


jupyter day 14 linear regression Last Checkpoint: 01/23/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [1]:
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from matplotlib import pyplot as plt
In [3]:
df=pd.read_csv("cars.csv")
In [4]:
df
Out[4]:
Car Model Volume Weight CO2
0 Toyoty Aygo 1000 790 99
1 Mitsubishi Space Star 1200 1160 95
2 Skoda Citigo 1000 929 95
3 Fiat 500 900 865 90
4 Mini Cooper 1500 1140 105
5 VW Up! 1000 929 105
6 Skoda Fabia 1400 1109 90
7 Mercedes A-Class 1500 1365 92
8 Ford Fiesta 1500 1112 98
In [5]:
df.shape
Out[5]:
(36, 5)
In [6]:
X = df[['Volume', 'Weight']]
In [7]:
X
Out[7]:
Volume Weight
0 1000 790
1 1200 1160
2 1000 929
3 900 865
4 1500 1140
5 1000 929
6 1400 1109
7 1500 1365
8 1500 1112
9 1600 1150
10 1100 980
In [8]:
y = df['CO2']
In [9]:
y
Out[9]:
0 99
1 95
2 95
3 90
4 105
5 105
6 90
7 92
8 98
9 99
10 99
11 101
12 99
13 94
14 97
15 97
16 99
17 104
18 104
19 105
20 94
21 99
Activate Windows
Go to PC settings to activate Windows.


jupyter day 14 linear regression Last Checkpoint: 01/23/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [14]: 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)
In [20]: from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
Out[20]: LinearRegression()
In [22]: y_pred = model.predict(X_test)
In [23]: y_pred
Out[23]: array([108.77987905, 104.44799808, 102.66930685, 108.38872848,
106.43947781, 102.38504113, 95.77221295, 94.603963 ,
97.36150012])
In [27]: car = [[1100, 875]]
co2 = model.predict(car)
C:\Users\sanish\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
warnings.warn(
In [28]: co2
Out[28]: array([94.91941579])
In [ ]:
Activate Windows
Go to PC settings to activate Windows.

The End