22A81A05L0

22A81A05L1

22A81A05L2

22A81A05L3

CSE 2ND YEAR TEAM-2

22A81A05L4

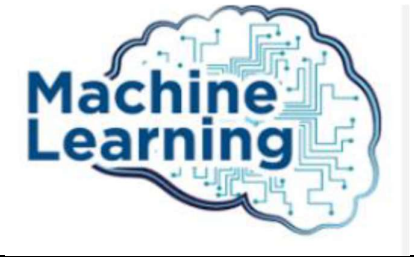
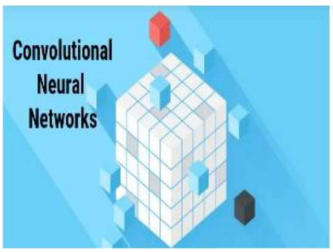
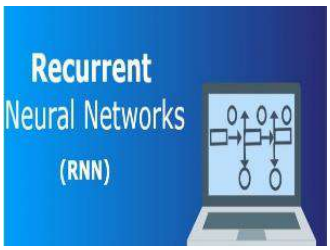
22A81A05L5

22A81A05L6

22A81A05L7

22A81A05L8

22A81A05L9

S.no.	Domain	Project title
1		1.By using the given data, find the city is good or bad with the help of crime rate 2.Predicting Employee Attrition using Random Forest and Decision tree classifiers 3.Predictive Modeling for Obesity: Harnessing Machine Learning for Diagnosis and Prevention
2		1.Finding the weather report of a given city by using Weather API 2.Convert the text prompt to an image using API
3		1.With the help of given images check whether the person having brain tumor or not by using CNN 2.Identify whether the person having pneumonia with the help of given data by using CNN
4		1.Predicting the next words using a trained model with the help of RNN 2.Predicting the next sequence of words using a trained model with the help of RNN

5		1.Retrieving the data of richest people in india according to forbes 2023 magazine 2.Retrieving the data of top 10 tallest buildings in the world
6		1.Searching and saving the dell laptops in amazon using selenium 2.Taking the details from the user and searching myntra products by using selenium 3.Using selenium, Extract the data of smart watch products in myntra website
7		1.Hypothesis Analysis 2.ANOVA 3.Interpolation

Project:1:

By using the given data, find the city is good or bad with the help of crime rate


```
1 import numpy as np
2 import pandas as pd
3 from sklearn.tree import DecisionTreeClassifier
```

```
[ ]  1 demodt=pd.read_csv("/content/demodt.txt",sep=",")
     2 print(demodt)
```

	State	Literacy	Cleanliness	Crime_Rate	Good
0	A	92	90	54	0
1	B	56	67	50	1
2	C	78	85	62	0
3	D	63	72	48	1
4	E	85	79	55	0
5	F	71	68	58	0
6	G	80	83	51	0
7	H	67	74	47	1
8	I	89	88	53	0
9	J	58	65	49	1
10	K	82	81	60	0
11	L	75	78	57	0
12	M	69	70	46	1
13	N	87	86	52	0
14	O	61	63	45	1
15	P	93	91	56	0

Output:

```
[ ]  1 feature_cols=["Literacy","Cleanliness","Crime_Rate"]
     2 feature=demodt[feature_cols]
     3 target=demodt.Good
```

```
[ ]  1 logr=DecisionTreeClassifier()
     2 logr.fit(feature,target)
```

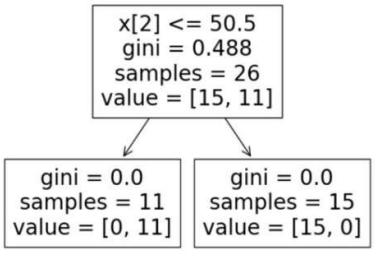
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
  1 a=int(input("Enter literacy:"))
  2 b=int(input("Enter cleanliness:"))
  3 c=int(input("Enter crimerate:"))
  4 p=logr.predict([[a,b,c]])
  5 if p==1:
  6   print("GOOD")
  7 else:
  8   print("BAD")
```

```
Enter literacy:9
Enter cleanliness:78
Enter crimerate:87
BAD
```

```
1 plot_tree(logr)   #plot_tree(model_name)
```

```
[Text(0.5, 0.75, 'x[2] <= 50.5\ngini = 0.488\nsamples = 26\nvalue = [15, 11]'),
 Text(0.25, 0.25, 'gini = 0.0\nsamples = 11\nvalue = [0, 11]'),
 Text(0.75, 0.25, 'gini = 0.0\nsamples = 15\nvalue = [15, 0]')]
```



Project 2:

With the help of given data predict whether the employee will continue in the job or not


```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
```

```
dt=pd.read_csv("/content/WA_Fn-UseC_-HR-Employee-Attrition.csv",sep=",")
print(dt)
```

DAY-6-PRO.ipynb

File Edit View Insert Runtime Tools Help Last saved at March 2

+ Code + Text

```
feature_cols=["YearsAtCompany","Age","TotalWorkingYears"]
feature=dt[feature_cols]
target=dt.Attrition
```

```
logr=DecisionTreeClassifier()
logr.fit(feature,target)
```

DecisionTreeClassifier
DecisionTreeClassifier()

```
d=int(input("Enter your working years in this company:"))
e=int(input("Enter your age:"))
f=int(input("Enter your total working years:"))
p=logr.predict([[d,e,f]])
if p=='Yes':
  print("YES")
else:
  print("NO")
```

```
Enter your working years in this company:8
Enter your age:45
Enter your total working years:16
NO
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
lr=RandomForestClassifier(n_estimators=60)
lr.fit(feature,target)
```

RandomForestClassifier
RandomForestClassifier(n_estimators=60)

DAY-6-PRO.ipynb

File Edit View Insert Runtime Tools Help Last saved at March 2

+ Code + Text

```
e=int(input("Enter your age:"))
f=int(input("Enter your total working years:"))
p=lr.predict([[d,e,f]])
if(p=='Yes1'):
  print("GOOD")
else:
  print("BAD")
```

```
Enter your working years in this company:8
Enter your age:45
Enter your total working years:16
BAD
```

```
import os
output_dir="tree visualisations"
os.makedirs(output_dir,exist_ok=True)
```

```
for i,tree in enumerate(lr.estimators_):
  tree_dot_file=os.path.join(output_dir,f"tree_{i}.dot")
  tree_png_file=os.path.join(output_dir,f"tree_{i}.png")
```

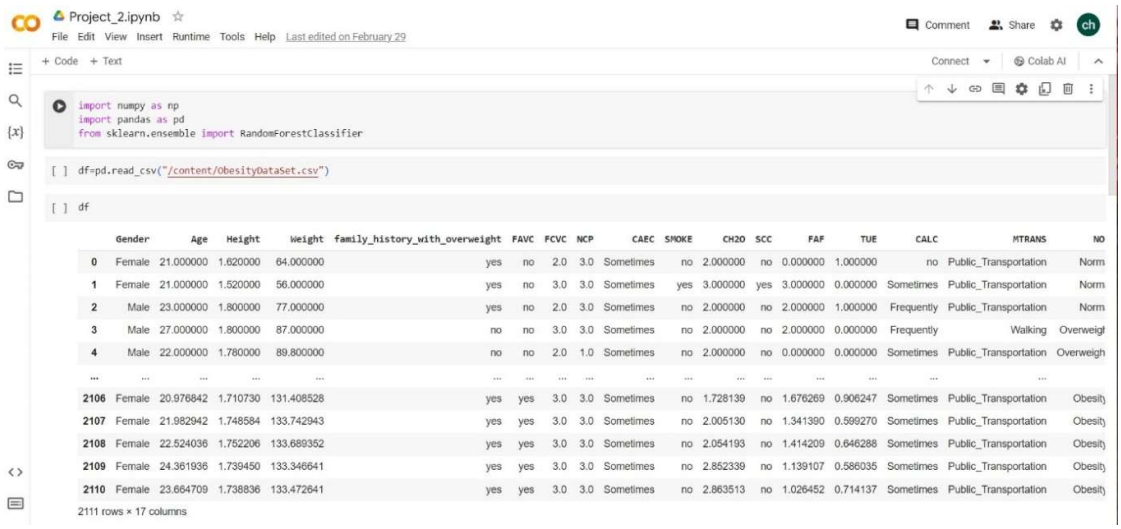
```
from sklearn.tree import export_graphviz
export_graphviz(tree, out_file=tree_dot_file , feature_names=["YearsAtCompany","Age","TotalWorkingYears"],
                class_names=[str(cls) for cls in lr.classes_], filled=True, rounded=True)
```

```
command= f"dot -Tpng {tree_dot_file} -o {tree_png_file}"
os.system(command)
print(f"Tree {i} visualization saved to {tree_png_file}")
```

```
Tree 59 visualization saved to tree visualisations/tree_59.png
```

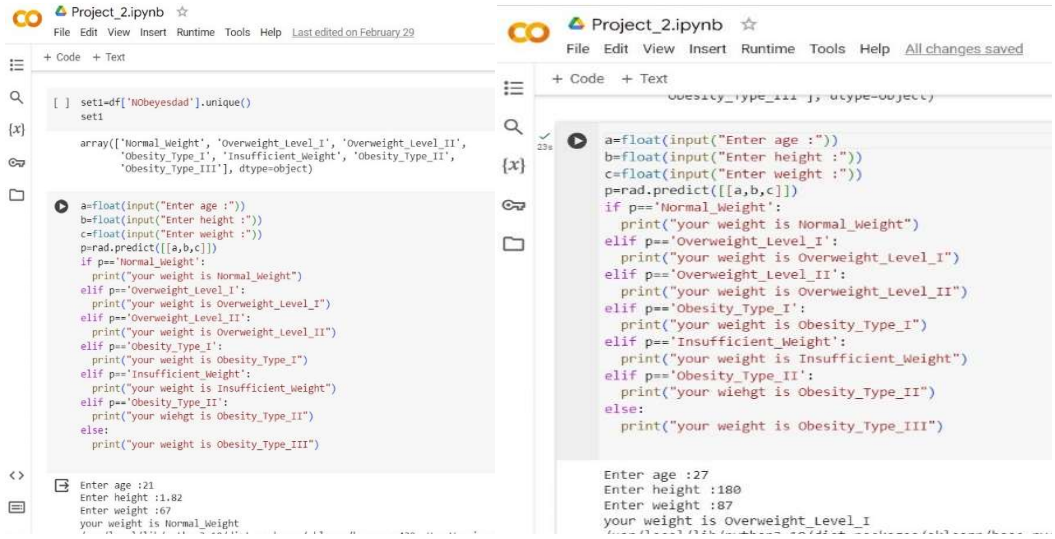
Project 3:

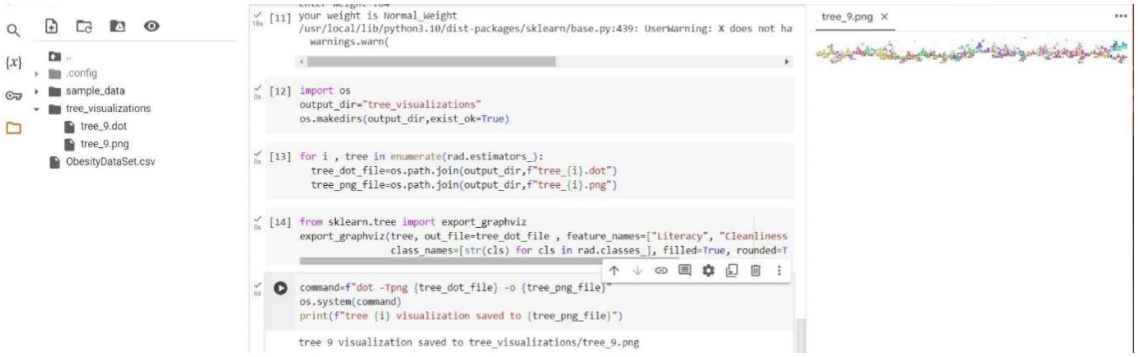
Predictive Modeling for Obesity: Harnessing MachineLearning for Diagnosis and Prevention



	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NO
0	Female	21.000000	1.620000	64.000000	yes	no	2.0	3.0	Sometimes	no	2.000000	no	0.000000	1.000000	no	Public_Transportation	Norm
1	Female	21.000000	1.520000	56.000000	yes	no	3.0	3.0	Sometimes	yes	3.000000	yes	3.000000	0.000000	Sometimes	Public_Transportation	Norm
2	Male	23.000000	1.800000	77.000000	yes	no	2.0	3.0	Sometimes	no	2.000000	no	2.000000	1.000000	Frequently	Public_Transportation	Norm
3	Male	27.000000	1.800000	87.000000	no	no	3.0	3.0	Sometimes	no	2.000000	no	2.000000	0.000000	Frequently	Walking	Overweigl
4	Male	22.000000	1.780000	89.800000	no	no	2.0	1.0	Sometimes	no	2.000000	no	0.000000	0.000000	Sometimes	Public_Transportation	Overweigh
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.0	3.0	Sometimes	no	1.728139	no	1.676269	0.906247	Sometimes	Public_Transportation	Obesity
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.0	3.0	Sometimes	no	2.005130	no	1.341390	0.599270	Sometimes	Public_Transportation	Obesity
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.0	3.0	Sometimes	no	2.054193	no	1.414209	0.646288	Sometimes	Public_Transportation	Obesity
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.0	3.0	Sometimes	no	2.852339	no	1.139107	0.586035	Sometimes	Public_Transportation	Obesity
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.0	3.0	Sometimes	no	2.863513	no	1.026452	0.714137	Sometimes	Public_Transportation	Obesity

2111 rows × 17 columns


Project_2.ipynb
File Edit View Insert Runtime Tools Help Last edited on February 29
+ Code + Text
[ ] set1=df['NObeyesdad'].unique()
set1
array(['Normal_Weight', 'Overweight_Level_I', 'Overweight_Level_II',
'Obesity_Type_I', 'Insufficient_Weight', 'Obesity_Type_II',
'Obesity_Type_III'], dtype=object)
a=float(input("Enter age :"))
b=float(input("Enter height :"))
c=float(input("Enter weight :"))
p=rad.predict([[a,b,c]])
if p=='Normal_Weight':
print("your weight is Normal_Weight")
elif p=='Overweight_Level_I':
print("your weight is Overweight_Level_I")
elif p=='Overweight_Level_II':
print("your weight is Overweight_Level_II")
elif p=='Obesity_Type_I':
print("your weight is Obesity_Type_I")
elif p=='Insufficient_Weight':
print("your weight is Insufficient_Weight")
elif p=='Obesity_Type_II':
print("your wiehgt is Obesity_Type_II")
else:
print("your weight is Obesity_Type_III")
Enter age :21
Enter height :1.82
Enter weight :67
your weight is Normal_Weight
Project_2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
a=float(input("Enter age :"))
b=float(input("Enter height :"))
c=float(input("Enter weight :"))
p=rad.predict([[a,b,c]])
if p=='Normal_Weight':
print("your weight is Normal_Weight")
elif p=='Overweight_Level_I':
print("your weight is Overweight_Level_I")
elif p=='Overweight_Level_II':
print("your weight is Overweight_Level_II")
elif p=='Obesity_Type_I':
print("your weight is Obesity_Type_I")
elif p=='Insufficient_Weight':
print("your weight is Insufficient_Weight")
elif p=='Obesity_Type_II':
print("your wiehgt is Obesity_Type_II")
else:
print("your weight is Obesity_Type_III")
Enter age :27
Enter height :180
Enter weight :87
your weight is Overweight_Level_I

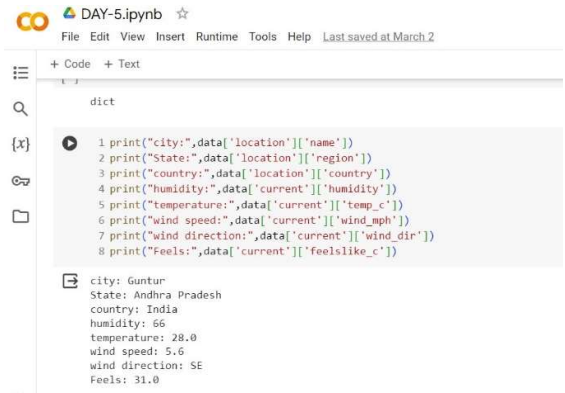

.config
sample_data
tree_visualizations
tree_9.dot
tree_9.png
ObesityDataSet.csv
[11] your weight is Normal_Weight
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not ha
warnings.warn(
[12] import os
output_dir="tree_visualizations"
os.makedirs(output_dir,exist_ok=True)
[13] for i , tree in enumerate(rad.estimators_):
tree_dot_file=os.path.join(output_dir,f"tree_{i}.dot")
tree_png_file=os.path.join(output_dir,f"tree_{i}.png")
[14] from sklearn.tree import export_graphviz
export_graphviz(tree, out_file=tree_dot_file , feature_names=["Literacy", "Cleanliness
class_names=[str(cls) for cls in rad.classes_], filled=True, rounded=T
command=f"dot -Tpng {tree_dot_file} -o {tree_png_file}"
os.system(command)
print(f"tree {i} visualization saved to {tree_png_file}")
tree 9 visualization saved to tree_visualizations/tree_9.png
tree_9.png

Project 4:

Finding the weather report of a given city by using Weather API


DAY-5.ipynb
File Edit View Insert Runtime Tools Help Last saved at March 2
Comment
Share
+ Code + Text
Connect
import requests
api_key = "9f5ea1f4733d440582e53549242802"
city = input("Enter your favourite city:\n")
url = f"http://api.weatherapi.com/v1/current.json?key={api_key}&q={city}"
response = requests.get(url)
data = response.json()
print(data)
if response.status_code==200:
print("city:",data['location']['name'])
else:
print(data['error']['message'])
Enter your favourite city:
guntur
{'location': {'name': 'Guntur', 'region': 'Andhra Pradesh', 'country': 'India', 'lat': 16.3, 'lon': 80.45, 'tz_id': 'Asia/Kolkata', 'localtime_epoch': 1709390159,
city: Guntur
print("State:",data['location']['region'])
State: Andhra Pradesh


DAY-5.ipynb
File Edit View Insert Runtime Tools Help Last saved at March 2
+ Code + Text
print("State:",data['location']['region'])
State: Andhra Pradesh
print("country:",data['location']['country'])
country: India
print("temperature:",data['current']['temp_c'])
temperature: 28.0
print("humidity:",data['current']['humidity'])
humidity: 66
print("wind speed:",data['current']['wind_mph'])
wind speed: 5.6
print("wind direction:",data['current']['wind_dir'])
wind direction: SE


DAY-5.ipynb
File Edit View Insert Runtime Tools Help Last saved at March 2
+ Code + Text
dict
print("city:",data['location']['name'])
print("State:",data['location']['region'])
print("country:",data['location']['country'])
print("humidity:",data['current']['humidity'])
print("temperature:",data['current']['temp_c'])
print("wind speed:",data['current']['wind_mph'])
print("wind direction:",data['current']['wind_dir'])
print("Feels:",data['current']['feelslike_c'])
city: Guntur
State: Andhra Pradesh
country: India
humidity: 66
temperature: 28.0
wind speed: 5.6
wind direction: SE
Feels: 31.0

Project 5:

Convert the text prompt to an image using API

DAY-5-HW.ipynb

File Edit View Insert Runtime Tools Help Last saved at 12:21 PM

Comment Share D

+ Code + Text Connect

```
import requests

url = "https://text-to-image7.p.rapidapi.com/"

querystring = {"prompt":"a painting of foxes playing poker, disney, zootopia","batch_size":"1","negative_prompt":"ugly, duplicate, morbid, mutilated, [out of frar

headers = {
  "X-RapidAPI-Key": "abda0da374msh31c763199500085p12783bjsn89e02ed2d036",
  "X-RapidAPI-Host": "text-to-image7.p.rapidapi.com"
}

response = requests.get(url, headers=headers, params=querystring)

print(response.json())
```

```
{'data': ['https://api.haxed.net/tmp/5WnZ003yVt.png']}
```

Project 6:

With the help of given images check whether the person having brain tumor or not by using CNN

DAY-7.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at March 2

+ Code + Text

```
[ ] 1 import tensorflow as tf
    2 from tensorflow import keras
    3 from tensorflow.keras import layers
    4 from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
[ ] 1 train_datagen = ImageDataGenerator(
    2     rescale=1./255,
    3     validation_split=0.2
    4 )
```

```
    1 IMG_SIZE=224
    2 BATCH_SIZE=32
```

```
[ ] 1 train_generator = train_datagen.flow_from_directory(
    2     r'/content/drive/MyDrive/Brain_Tumor_Detection-20240301T063135Z-001/Brain_Tumor_Detection/Train',
    3     target_size=(IMG_SIZE,IMG_SIZE),
    4     batch_size=BATCH_SIZE,
    5     class_mode='binary',
    6     subset='training'
    7 )
```

```
Found 2408 images belonging to 2 classes.
```

```
[ ] 1 val_generator = train_datagen.flow_from_directory(
    2     r'/content/drive/MyDrive/Brain_Tumor_Detection-20240301T063135Z-001/Brain_Tumor_Detection/Train',
    3     target_size=(IMG_SIZE,IMG_SIZE),
    4     batch_size=BATCH_SIZE,
    5     class_mode='binary',
    6     subset='validation'
    7 )
```

```
Found 602 images belonging to 2 classes.
```

```
[ ] 1 model=keras.Sequential([
    2     layers.Conv2D(32,(3,3),activation='relu',input_shape=(IMG_SIZE,IMG_SIZE,3)),
    3     layers.MaxPooling2D((2,2)),
    4     layers.Conv2D(64,(3,3),activation='relu'),
    5     layers.MaxPooling2D((2,2)),
    6     layers.Conv2D(128,(3,3),activation='relu'),
    7     layers.MaxPooling2D((2,2)),
    8     layers.Flatten(),
    9     layers.Dense(128,activation='relu'),
   10     layers.Dense(1,activation='sigmoid')
   11 ])
```

```
[ ] 1 model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
[ ] 1 model.fit(train_generator,validation_data=val_generator,epochs=5)
```

```
Epoch 1/5
76/76 [==============================] - 464s 6s/step - loss: 0.5757 - accuracy: 0.7683 - val_loss: 0.4292 - val_accuracy: 0.8156
Epoch 2/5
76/76 [==============================] - 289s 4s/step - loss: 0.2390 - accuracy: 0.9128 - val_loss: 0.1545 - val_accuracy: 0.9419
Epoch 3/5
76/76 [==============================] - 306s 4s/step - loss: 0.1170 - accuracy: 0.9614 - val_loss: 0.0721 - val_accuracy: 0.9751
Epoch 4/5
76/76 [==============================] - 284s 4s/step - loss: 0.0716 - accuracy: 0.9788 - val_loss: 0.0687 - val_accuracy: 0.9784
Epoch 5/5
76/76 [==============================] - 287s 4s/step - loss: 0.0392 - accuracy: 0.9875 - val_loss: 0.0114 - val_accuracy: 0.9967
<keras.src.callbacks.History at 0x7bafe5c42980>
```

```
[ ] 1 model.save("Model.h5",'label.txt')
```

Day-8_BrainTumour.ipynb ☆

File Edit View Insert Runtime Tools Help Last edited on 2 March

+ Code + Text

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

```
[ ] #load the saved model
    model=load_model("/content/Model.h5")
```

```
[ ] #load and preprocess the test image
    test_image_path='/content/drive/MyDrive/Brain_Tumor_Detection-20240301T063040Z-001/Test/pred/pred1.jpg'
    img=image.load_img(test_image_path,target_size=(224,224))
    img_array=image.img_to_array(img)
    img_array=np.expand_dims(img_array,axis=0)

    #Add batch dimension
    img_array/=225.   #Normalize the pixel values
    #make predictions
    prediction=model.predict(img_array)
    #print the prediction
    print(prediction)
```

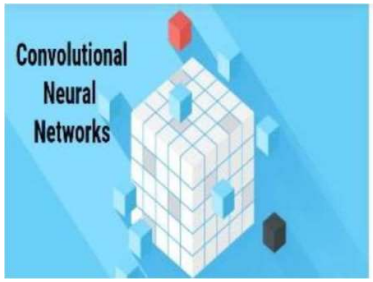
```
1/1 [==============================] - 0s 50ms/step
[[1.5095505e-06]]
```

```
if prediction<0.5:
  print("Prediction:No tumor")
else:
  print("Prediction:Tumor present")
```

```
Prediction:No tumor
```

Project 7:

Identify whether the person having pneumonia with the help of given data by using CNN



jupyter Day_8 Last Checkpoint: Yesterday at 4:33 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run Code

In [2]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
WARNING:tensorflow:From D:\Users\darla\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_
cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [3]:
```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)
```

In [4]:
```
IMG_SIZE=224
BATCH_SIZE=32
```

In [10]:
```
train_generator = train_datagen.flow_from_directory(
    r"C:\Users\darla\Downloads\archive (3)\chest_xray\train",
    target_size=(IMG_SIZE,IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)
```

```
Found 4173 images belonging to 2 classes.
```

In [11]:
```
val_generator = train_datagen.flow_from_directory(
    r'C:\Users\darla\Downloads\archive (3)\chest_xray\train',
    target_size=(IMG_SIZE,IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)
```

```
Found 1043 images belonging to 2 classes.
```

In [12]:
```
model=keras.Sequential([
    layers.Conv2D(32,(3,3),activation='relu',input_shape=(IMG_SIZE,IMG_SIZE,3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(128,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(128,activation='relu'),
    layers.Dense(1,activation='sigmoid')
])
```

In [14]:
```
model.fit(train_generator,validation_data=val_generator,epochs=5)
```

```
Epoch 1/5
WARNING:tensorflow:From D:\Users\darla\anaconda3\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTe
nsorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From D:\Users\darla\anaconda3\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.execut
ing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

131/131 [==============================] - 303s 2s/step - loss: 0.3041 - accuracy: 0.8756 - val_loss: 0.1965 - val_accuracy: 0.
9271
Epoch 2/5
131/131 [==============================] - 263s 2s/step - loss: 0.1240 - accuracy: 0.9538 - val_loss: 0.1275 - val_accuracy: 0.
9511
Epoch 3/5
131/131 [==============================] - 252s 2s/step - loss: 0.0839 - accuracy: 0.9686 - val_loss: 0.1229 - val_accuracy: 0.
9616
Epoch 4/5
131/131 [==============================] - 270s 2s/step - loss: 0.0635 - accuracy: 0.9782 - val_loss: 0.1285 - val_accuracy: 0.
9540
Epoch 5/5
131/131 [==============================] - 254s 2s/step - loss: 0.0520 - accuracy: 0.9815 - val_loss: 0.1073 - val_accuracy: 0.
9626
```

Out[14]:
```
<keras.src.callbacks.History at 0x17f13f0e4d0>
```

jupyter Day_8 Last Checkpoint: Yesterday at 4:33 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ip

Run Code

In [18]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

In [23]:
```
#Load the saved model
model = load_model(r'C:\Users\darla\Pneumonia.h5')  #path of model file
```

In [32]:
```
#Load and preprocess the test image
test_image_path = r"C:\Users\darla\Downloads\archive (3)\chest_xray\train\NORMAL\NORMAL2-IM-1335-0001.jpeg"
img=image.load_img(test_image_path, target_size=(224,224))
img_array=image.img_to_array(img)
img_array=np.expand_dims(img_array,axis=0)
```

In [33]:
```
#Add batch dimension
img_array /=255.
prediction=model.predict(img_array)
print(prediction)
```

```
1/1 [==============================] - 0s 118ms/step
[[0.01437457]]
```

In [34]:
```
if prediction < 0.6:
  print("Prediction : No pneumonia Probability : ",prediction[0][0])
else:
  print("Prediction : pneumonia Probability : ",prediction[0][0])
```

```
Prediction : No pneumonia Probability :  0.014374572
```

Project 8:

Predicting the next words using a trained model withthe help of RNN


```
[ ]  1 import numpy as np
     2 from keras.models import Sequential
     3 from keras.layers import SimpleRNN,Dense,Embedding
     4 from keras.preprocessing.text import Tokenizer
     5 from keras.preprocessing.sequence import pad_sequences
     6 from keras.utils import to_categorical
```

```
[ ]  1 sentence=['I Hate college','I want to study','Python is useful']
```

```
[ ]  1 tokenizer=Tokenizer()
     2 tokenizer.fit_on_texts(sentence)
     3 total_words=len(tokenizer.word_index)+1
     4 print(total_words)
```

```
10
```

```
     1 # Creating input sequences and their corresponding next words
     2 input_sequences = []
     3 for sentences in sentence:
     4     tokenized_sentence = tokenizer.texts_to_sequences([sentences])[0]
     5     for i in range(1, len(tokenized_sentence)):
     6         n_gram_sequence = tokenized_sentence[:i+1]
     7         input_sequences.append(n_gram_sequence)
     8 input_sequences
```

```
[[1, 2], [1, 2, 3], [1, 4], [1, 4, 5], [1, 4, 5, 6], [7, 8], [7, 8, 9]]
```

```
     1 # Padding sequences for consistent input size
     2 max_sequence_length = max([len(seq) for seq in input_sequences])
     3 input_sequences = pad_sequences(input_sequences, maxlen=max_sequence_length, padding='pre')
```

```
[ ]  1 # Creating input and output data
     2 X, y = input_sequences[:, :-1], input_sequences[:, -1]
     3 y = to_categorical(y, num_classes=total_words)
```

```
[ ]  1 # Building a simple RNN model
     2 model = Sequential()
     3 model.add(Embedding(input_dim=total_words, output_dim=50, input_length=max_sequence_length-1))
     4 model.add(SimpleRNN(100, return_sequences=True))
     5 model.add(SimpleRNN(100))
     6 model.add(Dense(total_words, activation='softmax'))
```

```
[ ]  1 model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
     2 model.fit(X,y,epochs=5,verbose=2)
```

```
Epoch 1/5
1/1 - 2s - loss: 2.3115 - accuracy: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 2/5
1/1 - 0s - loss: 2.2328 - accuracy: 0.4286 - 10ms/epoch - 10ms/step
Epoch 3/5
1/1 - 0s - loss: 2.1575 - accuracy: 0.5714 - 9ms/epoch - 9ms/step
Epoch 4/5
1/1 - 0s - loss: 2.0835 - accuracy: 0.7143 - 10ms/epoch - 10ms/step
```

```
[ ]  1 # Generating text using the trained model
     2 seed_text = input("Enter the starting word: ")
     3 next_words = int(input("Enter how many words to predict: "))
     4 for _ in range(next_words):
     5     tokenized_seed = tokenizer.texts_to_sequences([seed_text])[0]
     6     tokenized_seed = pad_sequences([tokenized_seed], maxlen=max_sequence_length-1, padding='pre')
     7     predicted_word_index = np.argmax(model.predict(tokenized_seed), axis=-1)
     8     predicted_word = tokenizer.index_word[predicted_word_index[0]]
     9     seed_text += " " + predicted_word
    10 print(seed_text)
```

```
Enter the starting word: i
Enter how many words to predict: 5
1/1 [==============================] - 0s 275ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
i want to study study want
```

Project 9:

Predicting the next sequence of words using a trained model with the help of RNN

```
import pandas as pd
e_sentences=pd.read_csv('/content/earth1.txt')
e_sentences
```

Earth has a dynamic atmosphere	which sustains Earth's surface conditions and protects it from most meteoroids and UV-light at entry. It has a composition of primarily nitrogen and oxygen. Water vapor is widely present in the atmosphere
	forming clouds that cover most of the planet. The water vapor acts as a greenhouse gas and

```
tokenizer=Tokenizer()
tokenizer.fit_on_texts(e_sentences)
total_words=len(tokenizer.word_index)+1
print(total_words)
```

```
60
```

```
# Creating input sequences and their corresponding next words
input_sequences = []
for sentences in e_sentences:
    tokenized_sentence = tokenizer.texts_to_sequences([sentences])[0]
    for i in range(1, len(tokenized_sentence)):
        n_gram_sequence = tokenized_sentence[:i+1]
        input_sequences.append(n_gram_sequence)
input_sequences
```

```
[[17, 8],
```

```
# Padding sequences for consistent input size
max_sequence_length = max([len(seq) for seq in input_sequences])
input_sequences = pad_sequences(input_sequences, maxlen=max_sequence_length, padding='pre')
```

```
# Creating input and output data
x, y = input_sequences[:, :-1], input_sequences[:, -1]
y = to_categorical(y, num_classes=total_words)
```

```
# Building a simple RNN model
model = Sequential()
model.add(Embedding(input_dim=total_words, output_dim=50, input_length=max_sequence_length-1))
model.add(SimpleRNN(100, return_sequences=True))
model.add(SimpleRNN(100))
model.add(Dense(total_words, activation='softmax'))
```

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
model.fit(x,y,epochs=5,verbose=2)
```

```
Epoch 1/5
3/3 - 2s - loss: 4.1138 - accuracy: 0.0000e+00 - 2s/epoch - 676ms/step
Epoch 2/5
3/3 - 0s - loss: 3.8571 - accuracy: 0.1807 - 62ms/epoch - 21ms/step
Epoch 3/5
3/3 - 0s - loss: 3.6594 - accuracy: 0.2410 - 62ms/epoch - 21ms/step
Epoch 4/5
3/3 - 0s - loss: 3.5055 - accuracy: 0.2289 - 61ms/epoch - 20ms/step
Epoch 5/5
3/3 - 0s - loss: 3.3738 - accuracy: 0.2530 - 67ms/epoch - 22ms/step
<keras.src.callbacks.History at 0x7b9f1e556590>
```

```
# Generating text using the trained model
seed_text = input("Enter the starting word: ")
next_words = int(input("Enter how many words to predict: "))
for _ in range(next_words):
    tokenized_seed = tokenizer.texts_to_sequences([seed_text])[0]
    tokenized_seed = pad_sequences([tokenized_seed], maxlen=max_sequence_length-1, padding='pre')
    predicted_word_index = np.argmax(model.predict(tokenized_seed), axis=-1)
    predicted_word = tokenizer.index_word[predicted_word_index[0]]
    seed_text += " " + predicted_word
print(seed_text)
```

```
Enter the starting word: earth
Enter how many words to predict: 4
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
earth the the the the
```

Project 10:

Retrieving the data of richest people in india according to forbes 2023 magazine

DAY-4-WS_BS.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Extracting a table from web page

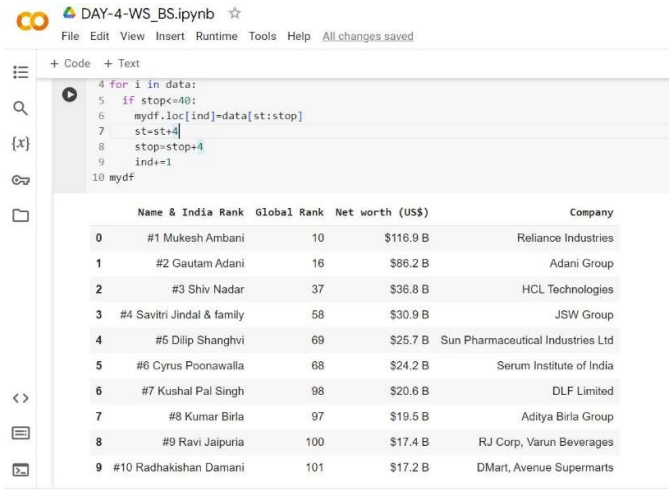
```
[ ] 1 from bs4 import BeautifulSoup
    2 import requests

    1 forbes="https://www.forbesindia.com/article/explainers/top-10-richest-people-india/85909/1"

[ ] 1 wp=requests.get(forbes)

    <Response [200]>

[ ] 1 htcode=code=BeautifulSoup(wp.text,'html')
    2 print(htcode)
                "@context": "https://schema.org",
                "@type": "ImageObject",
                "width": "800px",
                "height": "600px",
                "url":"https://www.forbesindia.com/media/images/2023/Jun/img_210953_top10.jpg"
              },
            "publisher":{
                "@context": "https://schema.org",
                "@type": "Organization",
                "name": "ForbesIndia",
```

DAY-4-WS_BS.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] 1 h=f.find_all("th")
    2 print(h)

    [<th style="border: 1px solid black; padding: 8px;"><strong>Name &amp;amp; India Rank</strong></th>, <th

[ ] 1 head=[i.text for i in h]
    2 print(head)

    ['Name & India Rank', 'Global Rank', 'Net worth (US$)', 'Company']

    1 import pandas as pd
    2 mydf=pd.DataFrame(columns=head)
    3 print(mydf)

    Empty DataFrame
    Columns: [Name & India Rank, Global Rank, Net worth (US$), Company]
    Index: []

    1 rows=f.find_all('td')
    2 print(rows)

    [<td style="border: 1px solid black; padding: 8px;">#1 Mukesh Ambani <br/></td>, <td style="border:
```



	Name & India Rank	Global Rank	Net worth (US$)	Company
0	#1 Mukesh Ambani	10	$116.9 B	Reliance Industries
1	#2 Gautam Adani	16	$86.2 B	Adani Group
2	#3 Shiv Nadar	37	$36.8 B	HCL Technologies
3	#4 Savitri Jindal & family	58	$30.9 B	JSW Group
4	#5 Dilip Shanghvi	69	$25.7 B	Sun Pharmaceutical Industries Ltd
5	#6 Cyrus Poonawalla	68	$24.2 B	Serum Institute of India
6	#7 Kushal Pal Singh	98	$20.6 B	DLF Limited
7	#8 Kumar Birla	97	$19.5 B	Aditya Birla Group
8	#9 Ravi Jaipuria	100	$17.4 B	RJ Corp, Varun Beverages
9	#10 Radhakishan Damani	101	$17.2 B	DMart, Avenue Supermarts

Project 11:

Retrieving the data of top 10 tallest buildings in theworld

jupyter DAY-4-HW-BeautifulSoup Last Checkpoint: Last Tuesday at 10:33 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [1]:
```
from bs4 import BeautifulSoup
import requests
```

In [2]:
```
url="https://en.m.wikipedia.org/wiki/List_of_tallest_buildings_in_India"
```

In [4]:
```
page=requests.get(url)
```

In [5]:
```
code=BeautifulSoup(page.text,'html')
print(code)
```

```
<!DOCTYPE html>
<html class="client-nojs mf-expand-sections-clientpref-0 mf-font-size-clientpref-small mw-mf-amc-clientpref-0 T357724" dir="l
tr" lang="en">
<head>
<meta charset="utf-8"/>
<title>List of tallest buildings in India - Wikipedia</title>
<script>(function(){var className="client-js mf-expand-sections-clientpref-0 mf-font-size-clientpref-small mw-mf-amc-clientpr
ef-0 T357724";var cookie=document.cookie.match(/(?:^|; )enwikimwclientpreferences=([^;]+)/);if(cookie){cookie[1].split('%2
C').forEach(function(pref){className=className.replace(new RegExp('(^| )'+pref.replace(/-clientpref-\w+$|[^\w-]+/g,'')+'-clie
ntpref-\\w+( |$)'),'$1'+pref+'$2');});}document.documentElement.className=className;}());RLCONF={"wgBreakFrames":false,"wgSep
aratorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","Febru
ary","March","April","May","June","July","August","September","October","November","December"],"wgRequestId":"c4610aa4-8d54-4
81d-a795-9aa2ac822f2d","wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNumber":0,"wgPageName":"List_
of_tallest_buildings_in_India","wgTitle":"List of tallest buildings in India","wgCurRevisionId":1210597935,"wgRevisionId":121
0597935,
"wgArticleId":904884,"wgIsArticle":true,"wgIsRedirect":false,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgPage
ViewLanguage":"en","wgPageContentLanguage":"en","wgPageContentModel":"wikitext","wgRelevantPageName":"List_of_tallest_buildin
gs_in_India","wgRelevantArticleId":904884,"wgIsProbablyEditable":true,"wgRelevantPageIsProbablyEditable":true,"wgRestrictionE
dit":[],"wgRestrictionMove":[],"wgNoticeProject":"wikipedia","wgFlaggedRevsParams":{"tags":{"status":{"levels":1}}},"wgMediaV
```

In [7]:
```
h=t.find_all("th")
print(h)
```

```
[<th>Rank
</th>, <th>Name
</th>, <th>Image
</th>, <th>City
</th>, <th>Height
</th>, <th>Floors
</th>, <th>Year
</th>, <th>Building type
</th>, <th>Reference(s)
</th>]
```

In [8]:
```
head=[i.text for i in h]
print(head)
```

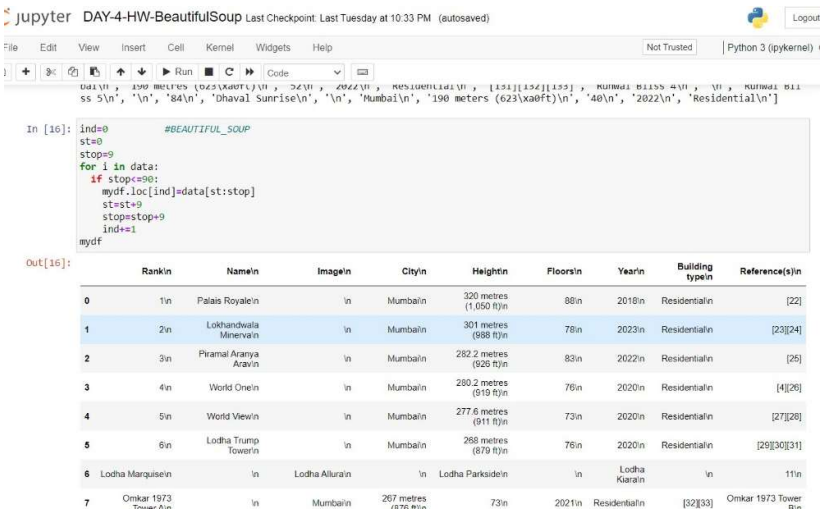
```
['Rank\n', 'Name\n', 'Image\n', 'City\n', 'Height\n', 'Floors\n', 'Year\n', 'Building type\n', 'Reference(s)\n']
```

In [9]:
```
import pandas as pd
mydf=pd.DataFrame(columns=head)
print(mydf)
```

```
Empty DataFrame
Columns: [Rank
, Name
, Image
, City
, Height
, Floors
, Year
, Building type
, Reference(s)
]
Index: []
```

In [10]:
```
rows=t.find_all('td')
print(rows)
```

```
[<td>1
</td>, <td><a href="/wiki/Palais_Royale,_Mumbai" title="Palais Royale, Mumbai">Palais Royale</a>
</td>, <td><span typeof="mw:File/Frameless"><a class="mw-file-description" href="/wiki/File:Palais_royale_worli_mumbai.png"><
noscript><img class="mw-file-element" data-file-height="1362" data-file-width="919" decoding="async" height="238" src="//uplo
```



	Rank\n	Name\n	Image\n	City\n	Height\n	Floors\n	Year\n	Building type\n	Reference(s)\n
0	1\n	Palais Royale\n	\n	Mumbai\n	320 metres (1,050 ft)\n	88\n	2018\n	Residential\n	[22]
1	2\n	Lokhandwala Minerva\n	\n	Mumbai\n	301 metres (988 ft)\n	78\n	2023\n	Residential\n	[23][24]
2	3\n	Piramal Aranya Arav\n	\n	Mumbai\n	282.2 metres (926 ft)\n	83\n	2022\n	Residential\n	[25]
3	4\n	World One\n	\n	Mumbai\n	280.2 metres (919 ft)\n	76\n	2020\n	Residential\n	[4][26]
4	5\n	World View\n	\n	Mumbai\n	277.6 metres (911 ft)\n	73\n	2020\n	Residential\n	[27][28]
5	6\n	Lodha Trump Tower\n	\n	Mumbai\n	268 metres (879 ft)\n	76\n	2020\n	Residential\n	[29][30][31]
6	Lodha Marquise\n	\n	Lodha Allura\n	\n	Lodha Parkside\n	\n	Lodha Kiara\n	\n	11\n
7	Omkar 1973	\n	Mumbai\n	267 metres	73\n	2021\n	Residential\n	[32][33]	Omkar 1973 Tower

Project 12:

Searching and saving the dell laptops in amazon using selenium


```
In [1]: import selenium

In [2]: from selenium import webdriver
        from selenium.webdriver.chrome.options import Options
        from selenium.webdriver.common.by import By
        from selenium.webdriver.common.keys import Keys
        from webdriver_manager.chrome import ChromeDriverManager
        import pandas as pd

In [3]: options=webdriver.ChromeOptions()
        driver=webdriver.Chrome(options=options)

        Error sending stats to Plausible: error sending request for url (https://plausible.io/api/event): operation timed out

In [4]: driver.get('https://www.amazon.in/')

In [5]: search=driver.find_element(By.XPATH,".//input[@class='nav-input nav-progressive-attribute']")

In [6]: search.send_keys("dell laptops")

In [7]: search.send_keys(Keys.ENTER)

In [18]: brand=driver.find_elements(By.XPATH,"//span[@class='a-size-medium a-color-base a-text-normal']")
         brands=[i.text for i in brand]
         brandss=brands[0:10]
         print(brandss)
```

In [10]:
```
rating=driver.find_elements(By.XPATH,".//i[@class='a-icon a-icon-star-small a-star-small-4 aok-align-bottom']")
ratings=[i.text for i in rating]
```

In [17]:
```
price=driver.find_elements(By.XPATH,".//div[@class='a-row']")
prices=[i.text for i in price]
pricing=prices[0:10]
print(pricing)
```

```
['', 'Deal of the Day', '₹35,990 M.R.P:\n₹48,692 (26% off)', '', 'FREE delivery Thu, 29 Feb', 'Or fastest delivery Tomorrow, 28 Feb', '', 'Deal of the Day', '₹35,990 M.R.P:\n₹57,778 (38% off)', '']
```

In [19]:
```
print(len(pricing))
print(len(brandss))
```

```
10
10
```

In [13]:
```
import pandas as pd
```

In [20]:
```
df=pd.DataFrame(columns=["Brand","Price"])
df
```

Out[20]:

Brand	Price

In [55]:
```
df=pd.DataFrame(columns=["Brand","Price"])
```

In [77]:
```
df['Brand']=final
df['Price']=price
```

In [78]:
```
df
```

Out[78]:

	Brand	Price
0	Dell 15 Laptop	34,990
1	Dell 14 Laptop	35,990
2	Dell 15 Laptop	47,990
3	Dell 14 Laptop	35,990
4	Dell 15 Laptop	33,990
5	Dell 15 Laptop	43,990
6	Dell 14 Laptop	49,990
7	Dell 14 Laptop	35,990
8	Dell Latitude	23,000
9	Dell G15 5520	76,990

In [81]:
```
df.to_csv("amazon.csv")
```

Project 13:

Taking the details from the user and searching myntra products by using selenium

In [ ]:
```
import selenium
```

In [16]:
```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from webdriver_manager.chrome import ChromeDriverManager
import pandas as pd
```

In [39]:
```
#define options and set browser capabilities
options=webdriver.ChromeOptions()
#options.add_argument('--some-option')
```

In [40]:
```
#create a webdriver instance with options
driver=webdriver.Chrome(options=options)
```

In [47]:
```
driver.get('https://www.myntra.com/')
```

In [48]:
```
search=driver.find_element(By.XPATH,".//input[@class='desktop-searchBar']")
```

In [49]:
```
search.send_keys("shirts")
```

In [50]:
```
search.send_keys(Keys.ENTER)
```

In [53]:
```
brand=driver.find_elements(By.XPATH,".//h3[@class='product-brand']")
brands=[i.text for i in brand]
```

In [60]:
```
print(prices)
```

```
['Rs. 643', 'Rs. 664', 'Rs. 645', 'Rs. 599', 'Rs. 753', 'Rs. 799', 'Rs. 1407', 'Rs. 549', 'Rs. 607', 'Rs. 799', 'Rs. 682', 'Rs
799', 'Rs. 699', 'Rs. 699', 'Rs. 337', 'Rs. 545', 'Rs. 749', 'Rs. 619', 'Rs. 683', 'Rs. 699', 'Rs. 747', 'Rs. 1247', 'Rs. 539'
'Rs. 519', 'Rs. 599', 'Rs. 685', 'Rs. 998', 'Rs. 671', 'Rs. 569', 'Rs. 699', 'Rs. 599', 'Rs. 615', 'Rs. 699', 'Rs. 594', 'Rs.
40', 'Rs. 1122', 'Rs. 959', 'Rs. 740', 'Rs. 519', 'Rs. 879', 'Rs. 558', 'Rs. 683', 'Rs. 611', 'Rs. 939', 'Rs. 749', 'Rs. 675',
'Rs. 519', 'Rs. 993', 'Rs. 1279', 'Rs. 549']
```

In [61]:
```
df=pd.DataFrame(columns=["Brand","Description","Price"])
df
```

Out[61]:

Brand	Description	Price

In [62]:
```
df["Brand"]=brands
df["Description"]=products
df["Price"]=prices
```

In [63]:
```
df
```

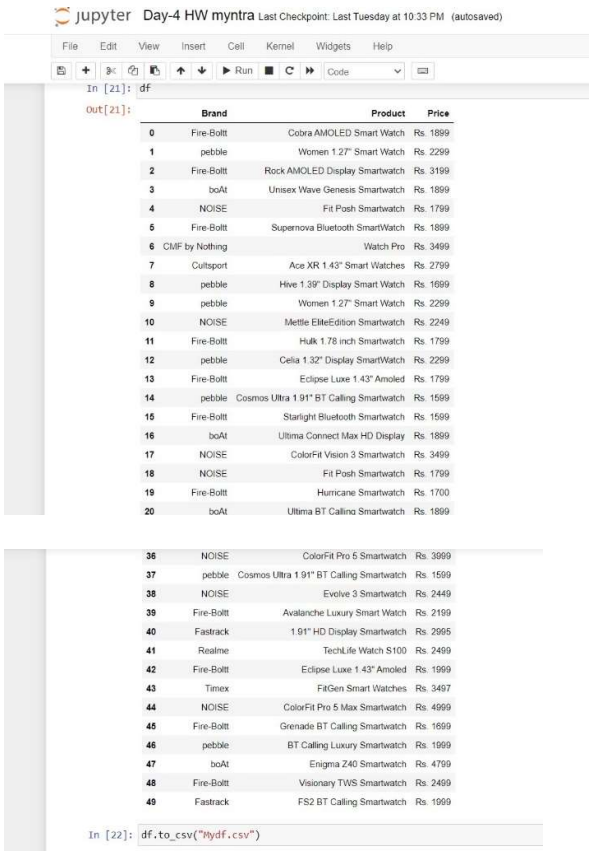
Out[63]:

	Brand	Description	Price
0	The Indian Garage Co	Men Slim Fit Casual Shirt	Rs. 643
1	Roadster	Men Casual Shirt	Rs. 664
2	Mast & Harbour	Men Slim Fit Casual Sustainable Shirt	Rs. 645
3	HERE&NOW	Slim Fit Casual Shirt	Rs. 599
4	Roadster	Men Pure Cotton Casual Shirt	Rs. 753
5	Mast & Harbour	Checked Regular Casual Sustainable Shirt	Rs. 799
6	WROGN	Pure Cotton Printed Shirt	Rs. 1407
7	LOCOMOTIVE	Men Slim Fit Casual Shirt	Rs. 549
8	Roadster	Men Cotton Casual Shirt	Rs. 607
9	HERE&NOW	Men Regular Fit Sustainable Casual Shirt	Rs. 799
10	The Indian Garage Co	Regular Fit Casual Shirt	Rs. 682
11	HERE&NOW	Slim Fit Cotton Casual Shirt	Rs. 799
12	HERE&NOW	Slim Fit Casual Shirt	Rs. 699
13	Dennis Lingo	Men Slim Fit Casual Shirt	Rs. 699
14	KETCH	Men Slim Fit Casual Shirt	Rs. 337
15	Roadster	Men Casual Shirt	Rs. 545
16	HERE&NOW	Slim Fit Striped Casual Shirt	Rs. 749

Project 14:

Using selenium, Extract the data of smart watchproducts in myntra website


```
In [1]: import selenium
        from selenium import webdriver
        from selenium.webdriver.chrome.options import Options
        from selenium.webdriver.common.by import By
        from selenium.webdriver.common.keys import Keys
        from webdriver_manager.chrome import ChromeDriverManager
        import pandas as pd

In [2]: options=webdriver.ChromeOptions()
        driver=webdriver.Chrome(options=options)

In [3]: driver.get('https://www.myntra.com/')

In [4]: search=driver.find_element(By.XPATH,".//input[@class='desktop-searchBar']")

In [5]: search.send_keys("smart watches")

In [6]: search.send_keys(Keys.ENTER)

In [7]: brand=driver.find_elements(By.XPATH,".//h3[@class='product-brand']")
        brands=[i.text for i in brand]

In [8]: product=driver.find_elements(By.XPATH,".//h4[@class='product-product']")
        products=[i.text for i in product]

In [11]: print(len(brands))
         print(len(products))
         print(len(prices))

         50
         50
         50

In [16]: df=pd.DataFrame(columns=["Brand","Product","Price"])
         df

Out[16]:
| Brand | Product | Price |
|---|---|---|

In [20]: df["Brand"]=brands          #WEB SCRAPPING USING SELENIUM
         df["Product"]=products
         df["Price"]=prices

In [21]: df
```



	Brand	Product	Price
0	Fire-Boltt	Cobra AMOLED Smart Watch	Rs. 1899
1	pebble	Women 1.27" Smart Watch	Rs. 2299
2	Fire-Boltt	Rock AMOLED Display Smartwatch	Rs. 3199
3	boAt	Unisex Wave Genesis Smartwatch	Rs. 1899
4	NOISE	Fit Posh Smartwatch	Rs. 1799
5	Fire-Boltt	Supernova Bluetooth SmartWatch	Rs. 1899
6	CMF by Nothing	Watch Pro	Rs. 3499
7	Cultsport	Ace XR 1.43" Smart Watches	Rs. 2799
8	pebble	Hive 1.39" Display Smart Watch	Rs. 1699
9	pebble	Women 1.27" Smart Watch	Rs. 2299
10	NOISE	Mettle EliteEdition Smartwatch	Rs. 2249
11	Fire-Boltt	Hulk 1.78 inch Smartwatch	Rs. 1799
12	pebble	Celia 1.32" Display SmartWatch	Rs. 2299
13	Fire-Boltt	Eclipse Luxe 1.43" Amoled	Rs. 1799
14	pebble	Cosmos Ultra 1.91" BT Calling Smartwatch	Rs. 1599
15	Fire-Boltt	Starlight Bluetooth Smartwatch	Rs. 1599
16	boAt	Ultima Connect Max HD Display	Rs. 1899
17	NOISE	ColorFit Vision 3 Smartwatch	Rs. 3499
18	NOISE	Fit Posh Smartwatch	Rs. 1799
19	Fire-Boltt	Hurricane Smartwatch	Rs. 1700
20	boAt	Ultima BT Calling Smartwatch	Rs. 1899

36	NOISE	ColorFit Pro 5 Smartwatch	Rs. 3999
37	pebble	Cosmos Ultra 1.91" BT Calling Smartwatch	Rs. 1599
38	NOISE	Evolve 3 Smartwatch	Rs. 2449
39	Fire-Boltt	Avalanche Luxury Smart Watch	Rs. 2199
40	Fastrack	1.91" HD Display Smartwatch	Rs. 2995
41	Realme	TechLife Watch S100	Rs. 2499
42	Fire-Boltt	Eclipse Luxe 1.43" Amoled	Rs. 1999
43	Timex	FitGen Smart Watches	Rs. 3497
44	NOISE	ColorFit Pro 5 Max Smartwatch	Rs. 4999
45	Fire-Boltt	Grenade BT Calling Smartwatch	Rs. 1699
46	pebble	BT Calling Luxury Smartwatch	Rs. 1999
47	boAt	Enigma Z40 Smartwatch	Rs. 4799
48	Fire-Boltt	Visionary TWS Smartwatch	Rs. 2499
49	Fastrack	FS2 BT Calling Smartwatch	Rs. 1999

```
In [22]: df.to_csv("Mydf.csv")
```

Project 15:

Hypothesis Analysis: Findingout the t-test and probability value.

```
[ ] #Hypothesis testing:
    import numpy as np
    from scipy import stats
```

```
[ ] a=df['Efficiency'].mean()
    b=df['Rec_Rate'].mean()
    print("Mean Recovery",b)
    group_A=df['Efficiency']
    group_B=df['Rec_Rate']
    t_stat,p_value=stats.ttest_ind(group_A,group_B)
    print("T-statistic",t_stat)
    print("P-statistic",p_value)
```

```
Mean Recovery 10.033572680447492
T-statistic 43.144276880141796
P-statistic 8.95811181993363e-46
```

```
[ ] #set significance level
    alpha=0.05
```

```
[ ] #Interpret the values
    if p_value<alpha:
      print("H0- related")
    else:
      print("H1- not related")
```

```
H0- related
```

```
[ ] mean_efficiency=df['Efficiency'].mean()
    mean_rec_rate=df['Rec_Rate'].mean()
```

```
[ ] a=np.array(df.Efficiency)
    b=np.array(df.Rec_Rate)
```

```
[ ] t_stat,p_value=stats.ttest_ind(a,b,equal_var=False)
```

```
    print("T-statistic:",t_stat)
    print("P-value : ",p_value)
```

```
T-statistic: 43.144276880141796
P-value :  3.4493842608099316e-34
```

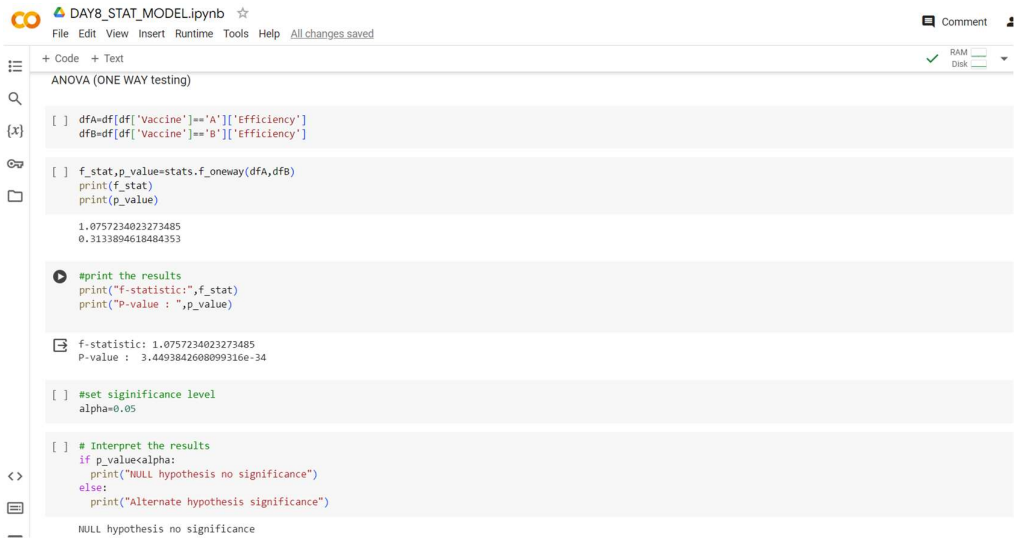
```
[ ] alpha=0.05
```

```
[ ] if p_value<alpha:
      print("NULL hypothesis no significance")
    else:
      print("Alternate hypothesis significance")
```

```
NULL hypothesis no significance
```

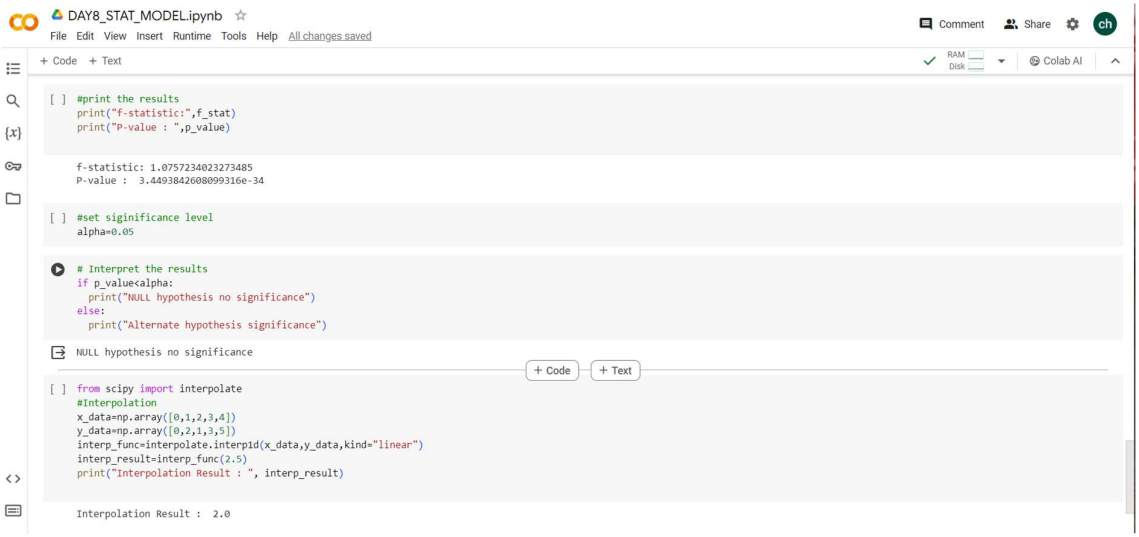
Project 16:

ANOVA Analysis: Finding out the f-stat and probability value.


DAY8_STAT_MODEL.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
+ Code + Text
RAM
Disk
ANOVA (ONE WAY testing)
[ ] dfA=df[df['Vaccine']=='A']['Efficiency']
dfB=df[df['Vaccine']=='B']['Efficiency']
[ ] f_stat,p_value=stats.f_oneway(dfA,dfB)
print(f_stat)
print(p_value)
1.0757234023273485
0.3133894618484353
#print the results
print("f-statistic:",f_stat)
print("P-value : ",p_value)
f-statistic: 1.0757234023273485
P-value : 3.449384260809931 6e-34
[ ] #set siginificance level
alpha=0.05
[ ] # Interpret the results
if p_value<alpha:
print("NULL hypothesis no significance")
else:
print("Alternate hypothesis significance")
NULL hypothesis no significance

Project 17:

Interpolation: Finding the value of Y for X input where X value should lie between the given boundaries.


DAY8_STAT_MODEL.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
ch
+ Code + Text
RAM
Disk
Colab AI
[ ] #print the results
print("f-statistic:",f_stat)
print("P-value : ",p_value)
f-statistic: 1.0757234023273485
P-value : 3.4493842608099316e-34
[ ] #set siginificance level
alpha=0.05
# Interpret the results
if p_value<alpha:
print("NULL hypothesis no significance")
else:
print("Alternate hypothesis significance")
NULL hypothesis no significance
+ Code
+ Text
[ ] from scipy import interpolate
#Interpolation
x_data=np.array([0,1,2,3,4])
y_data=np.array([0,2,1,3,5])
interp_func=interpolate.interp1d(x_data,y_data,kind="linear")
interp_result=interp_func(2.5)
print("Interpolation Result : ", interp_result)
Interpolation Result : 2.0