***Internship:Python with data science at BrainyBeam technologies pvt. Ltd.***

***Name: satani jainish s.***

***College : Dr.jivraj mehta institute of technology***

***Branch : computer engineering(B.E.)***

***Semester : 7 th***

***Day 1 Task :build python program which can take input of students with their subject marks, and gives their total marks obtained.***

n = **int**(**input**("Enter number of elements : "))

lst = []

for i in **range**(0, n):

    ele = **int**(**input**())

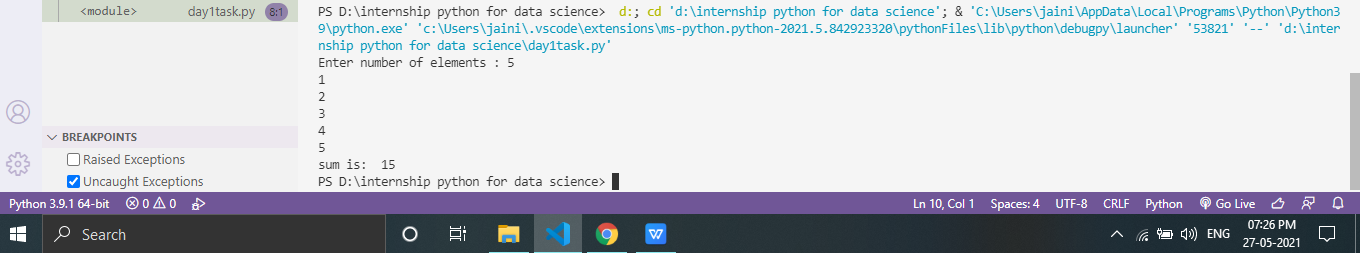
    lst.**append**(ele)

**sum** = 0

for i in **range**(**len**(lst)):

**sum** = **sum** + lst[i]

**print**( "sum is: ", **sum**)



***Day 2 Task :list out the methods used commonly In list,set,tuple,dictionary with their rules.***

|  |  |
| --- | --- |
| len() | Returns number of elements in list/tuple |
| max() | If list/tuple contains numbers, largest number will be returned. If list/tuple contains strings,  one that comes last in alphabetical order will be returned. |
| min() | If list/tuple contains numbers, smallest number will be returned. If list/tuple contains strings,  one that comes first in alphabetical order will be returned. |
| sum() | Returns addition of all elements in list/tuple |
| sorted() | sorts the elements in list/tuple |

List and tuple common method:

List method

|  |  |
| --- | --- |
| append() | appends an object to end  of list |
| copy() | makes a shallow copy of list |
| count() | return number of occurrences of value in list |
| extend() | extends the list by appending elements from another list/tuple |
| insert() | inserts object in the list before given index |

|  |  |
| --- | --- |
| pop() | removes and returns item at given index .  Raises IndexError if list is  empty or index is out of range. |
| remove() | removes first occurrence of value in the list. Raises ValueError  if the value is not present. |
| clear() | remove all items from the list |

|  |  |
| --- | --- |
| reverse() | reverses the list in place |
| sort() | sorts the list in place |

Dictionary:

|  |  |
| --- | --- |
| dict() | creates a new dictionary object |
| clear() | Remove all items from dictionary. |
| copy() | a shallow copy of dictionary |
| fromkeys() | Returns a new dict with keys from iterable and values equal to value. |
| get() | returns value associated with key |
| items() | list of tuples – each tuple is key value pair |
| keys() | list of dictionary keys |
| pop() | remove specified key and return the corresponding value. |
| popitem() | remove and return some (key, value) pair as a 2-tuple. |
| setdefault() | dictionary.get(k,dictionary), also set DICTIONARY[k]=dictionary if k not in dictionary |
| update() | Update dictionary from another dict/iterable |
| values() | list of dictionary values |

|  |  |
| --- | --- |
| len() | returns number of key: value pairs in dictionary |
| max() | If all keys in dictionary are numbers, largest number will be returned. If all keys in  dictionary are strings, one that comes last in alphabetical order will be returned. |
| min() | If all keys in dictionary are numbers, smallest number will be returned. If all keys  in dictionary are strings, one that comes first in alphabetical order will be returned. |

Set

|  |  |
| --- | --- |
| add() | Add an element to a set.This has no effect if the element is already present. |
| pop() | Remove and return an arbitrary set element.RaisesKeyError if the set is empty. |
| remove() | Remove an element from a set.If the element is not a member, raise a KeyError. |
| discard() | Remove an element from a set if it is a member.If the element is not a member,  do nothing. |
| update() | Update a set with the union of itself and others. |

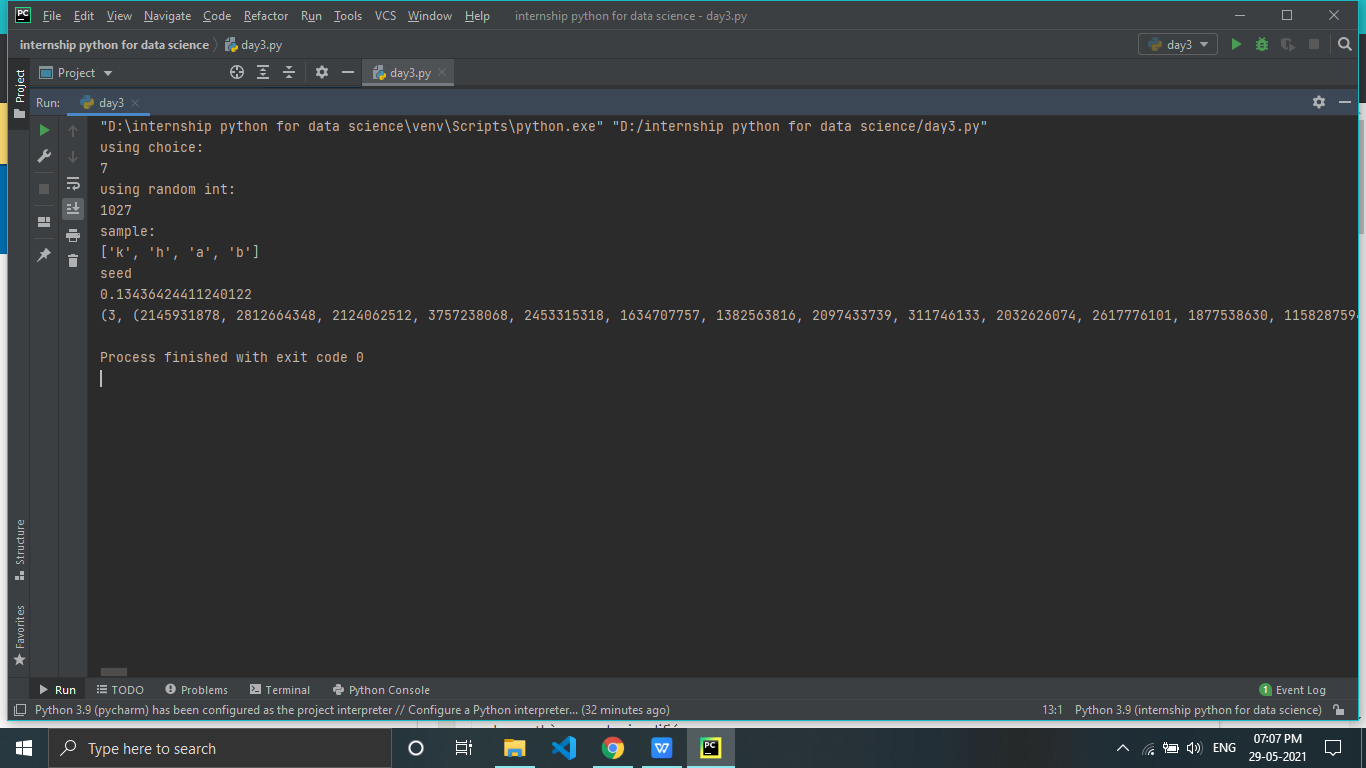
|  |  |
| --- | --- |
| difference() | Return the difference of two or more sets as a new set. |
| difference\_update() | Remove all elements of another set from this set. |
| intersection() | Return the intersection of two sets as a new set. |
| intersection\_update() | Update a set with the intersection of itself and another. |
| symmetric\_difference() | Return the symmetric difference of two sets as a new set. |
| symmetric\_difference\_  update() | Update a set with the symmetric difference of itself and another. |
| union() | Return the union of sets as a new set |

***Day 3 Task:***

***I)random module functions with explanation:***

|  |  |
| --- | --- |
| **Method** | **Description** |
| [seed()](https://www.w3schools.com/python/ref_random_seed.asp) | Initialize the random number generator |
| [getstate()](https://www.w3schools.com/python/ref_random_getstate.asp) | Returns the current internal state of the random number generator |
| [choice()](https://www.w3schools.com/python/ref_random_setstate.asp) | |  | | --- | | Returns a random element from the given sequence | |
| [randint()](https://www.w3schools.com/python/ref_random_randint.asp) | Returns a random number between the given range |
| [sample](https://www.w3schools.com/python/ref_random_sample.asp)[()](https://www.w3schools.com/python/ref_random_choice.asp) | |  | | --- | | Returns a given sample of a sequence | |

import random  
a=[1,23,7,6,88,4,78]  
print("using choice:")  
print(random.choice(a))  
print("using random int:")  
print(random.randint(100,5000))  
len=4  
num='abcdhgk'  
print("sample:")  
print(random.sample(num,len))  
random.seed(1)  
print("seed")  
print(random.random())  
x = random.getstate()  
print("getstate:")  
print(x)



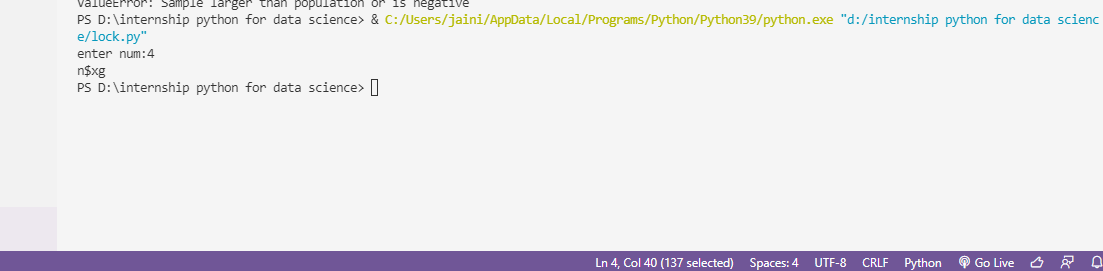
**ii)Build password generator program containing numbers,alphabets and**

**Special characters.**

import **random**

**len**=**int**(**input**("enter num:"))

num='123456789qwertyuiopasdfghjklzxcvbnm!@#$%^&\*()'

**print**(" ".**join**(**random**.**sample**(num,**len**)))

**Iii)write note about NLU,NLP,NLG with example.**

1. ****NLP**** — Natural Language “****Processing”****
2. ****NLU**** — Natural Language “****Understanding”****
3. ****NLG**** — Natural Language “****Generation”****

## What is natural language processing?

From the computer’s point of view, any natural language is a [free form text](https://www.pcmag.com/encyclopedia/term/free-form-text" \t "https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/_blank). That means there are no set keywords at set positions when providing an input.

Beyond the unstructured nature, there can also be multiple ways to express something using a natural language. For example, consider these three sentences:

* How is the weather today?
* Is it going to rain today?
* Do I need to take my umbrella today?

All these sentences have the same underlying question, which is to enquire about today’s weather forecast.

As humans, we can identify such underlying similarities almost effortlessly and respond accordingly. But this is a problem for machines—any algorithm will need the input to be in a set format, and these three sentences vary in their structure and format. And if we decide to code rules for each and every combination of words in any natural language to help a machine understand, then things will get very complicated very quickly.

This is where NLP enters the picture.

NLP is a subset of AI tasked with enabling machines to interact using natural languages. The domain of NLP also ensures that machines can:

* Process large amounts of natural language data
* Derive insights and information

But before any of this natural language processing can happen, the text needs to be standardized.

In [machine learning (ML)](https://www.bmc.com/blogs/machine-learning-hype-vs-reality/" \t "https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/_self) jargon, the series of steps taken are called [data pre-processing](https://www.bmc.com/blogs/data-annotation/" \t "https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/_self). The idea is to break down the natural language text into smaller and more manageable chunks. These can then be analyzed by ML algorithms to find relations, dependencies, and context among various chunks.

Some examples of pre-processing steps are:

* Parsing
* Stop-word removal
* Part-of-speech (POS) tagging
* Tokenization
* Many more

Thus, we can sum up: The aim of NLP is to process the free form natural language text so that it gets transformed into a standardized structure.

## What is natural language understanding (NLU)?

Considered [a subtopic](https://en.wikipedia.org/wiki/Natural-language_understanding" \t "https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/_blank) of NLP, the main focus of natural language understanding is to make machines:

* Interpret the natural language
* Derive meaning
* Identify context
* Draw insights

For example, in NLU, various ML algorithms are used to [identify the sentiment](https://en.wikipedia.org/wiki/Sentiment_analysis" \t "https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/_blank), perform [Name Entity Recognition](https://en.wikipedia.org/wiki/Named-entity_recognition" \t "https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/_blank) (NER), process semantics, etc. NLU algorithms often operate on text that has already been standardized by text pre-processing steps.

Going back to our weather enquiry example, it is NLU which enables the machine to understand that those three different questions have the same underlying weather forecast query. After all, different sentences can mean the same thing, and, vice versa, the same words can mean different things depending on how they are used.

Let’s take another example:

* The banks will be closed for Thanksgiving.
* The river will overflow the banks during floods.

A task called [word sense disambiguation](https://en.wikipedia.org/wiki/Word-sense_disambiguation" \t "https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/_blank), which sits under the NLU umbrella, makes sure that the machine is able to understand the two different senses that the word “bank” is used.

Iv)perform text to speech examples using gtts.

from gtts import gTTS

import os

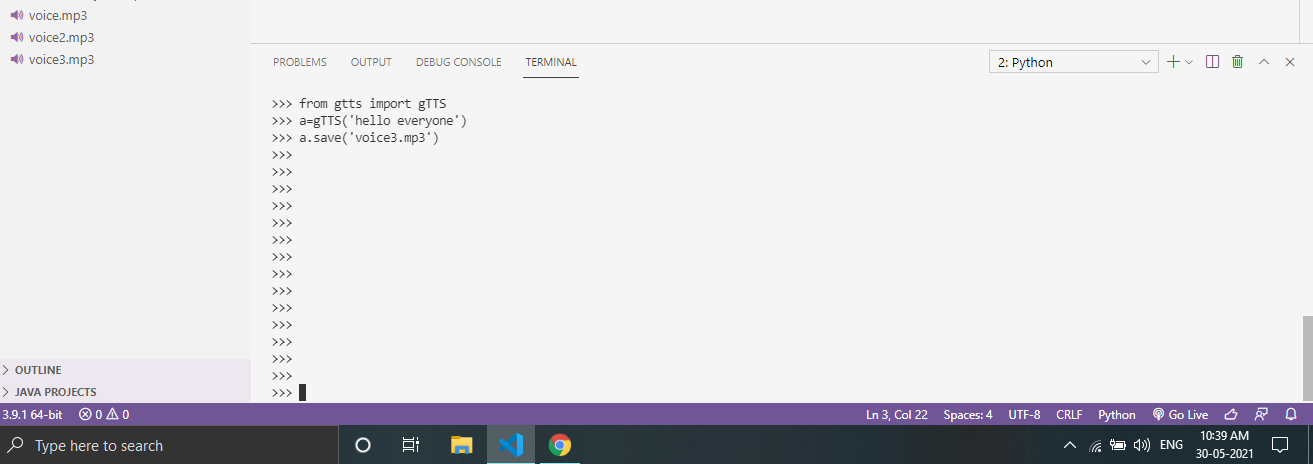
text = "hello everyone"

language = 'en'

output = gTTS(text = text, lang = language, slow = False)

output.save("voice.mp3")

os.system("start voice3.mp3")



***Day 4 Task :***

1. ***list out 5 method of pandas and numpy with output.***

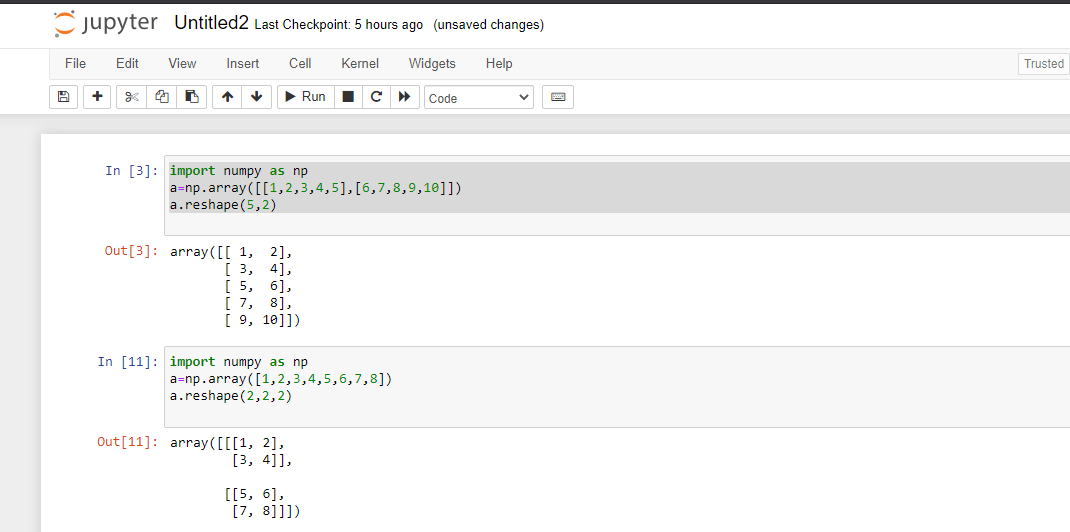
***Numpy methods:***

***Reshape:***

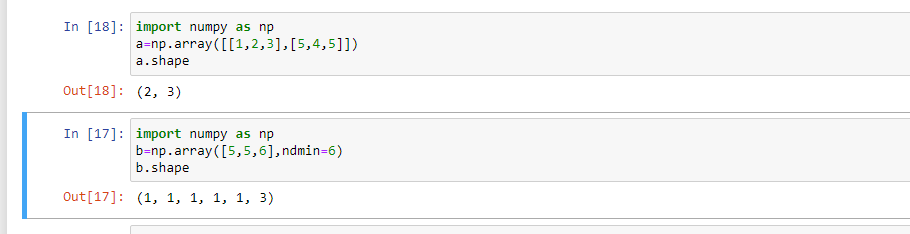
Reshaping means changing the shape of an array.

The shape of an array is the number of elements in each dimension.

By reshaping we can add or remove dimensions or change number of elements in each dimension.

******

***Shape:***

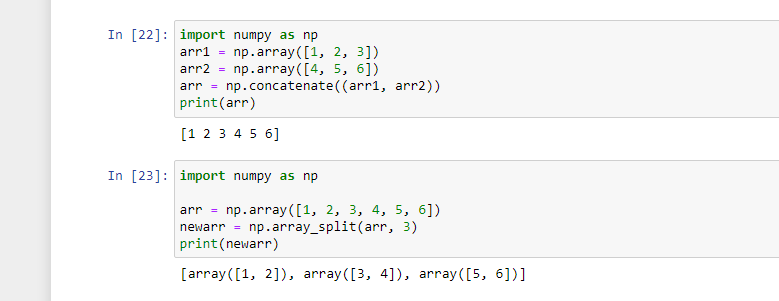
******The shape of an array is the number of elements in each dimension.

***Join-Concate :***Joining means putting contents of two or more arrays in a single array.

In SQL we join tables based on a key, whereas in NumPy we join arrays by axes.

We pass a sequence of arrays that we want to join to the concatenate() function, along with the axis. If axis is not explicitly passed, it is taken as 0.

***split:***

******Splitting is reverse operation of Joining.

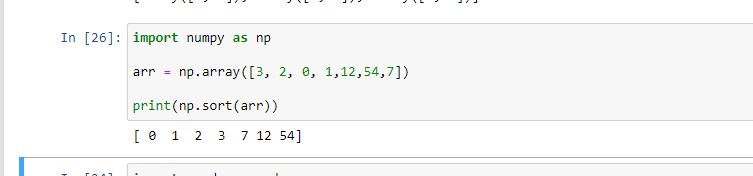
Joining merges multiple arrays into one and Splitting breaks one array into multiple.

We use array\_split() for splitting arrays, we pass it the array we want to split and the number of splits.

***Sort:***Sorting means putting elements in an ordered sequence.

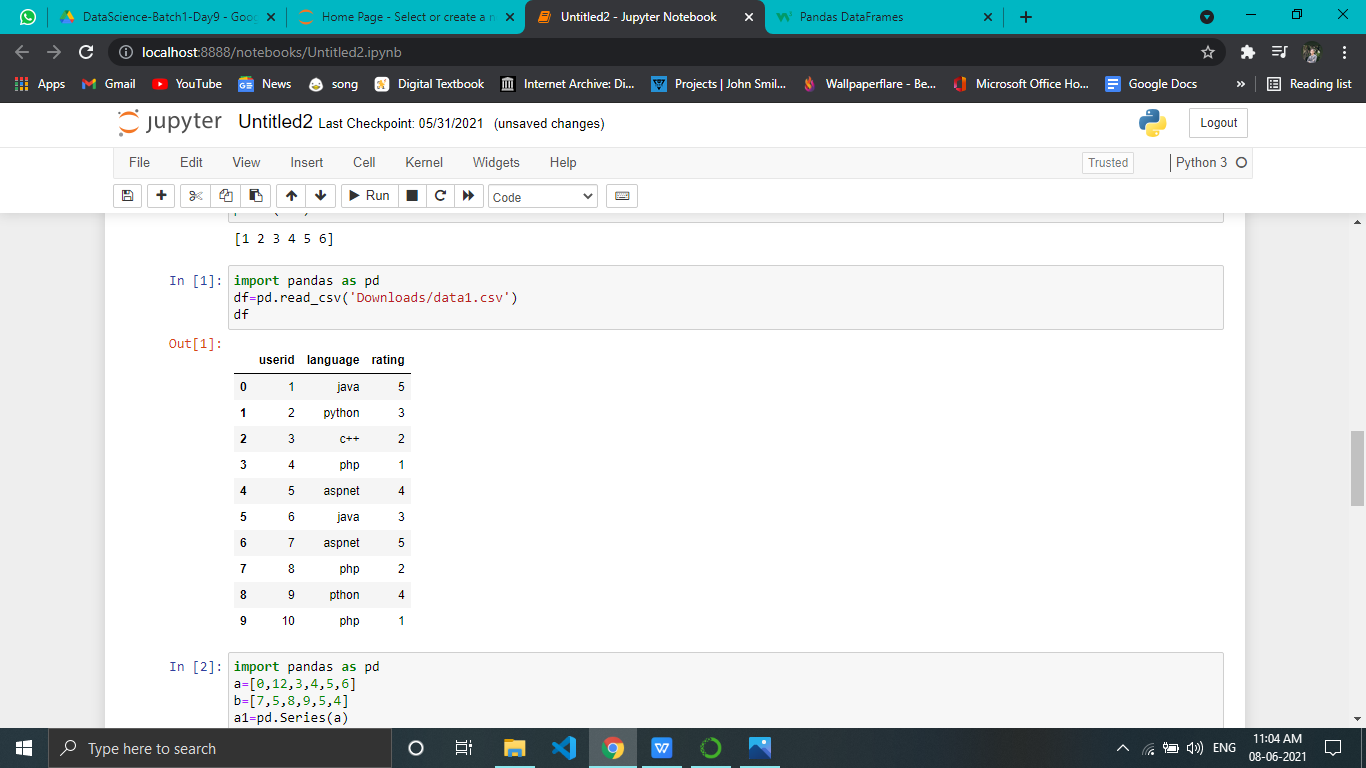
Ordered sequence is any sequence that has an order corresponding to elements, like numeric or alphabetical, ascending or descending.

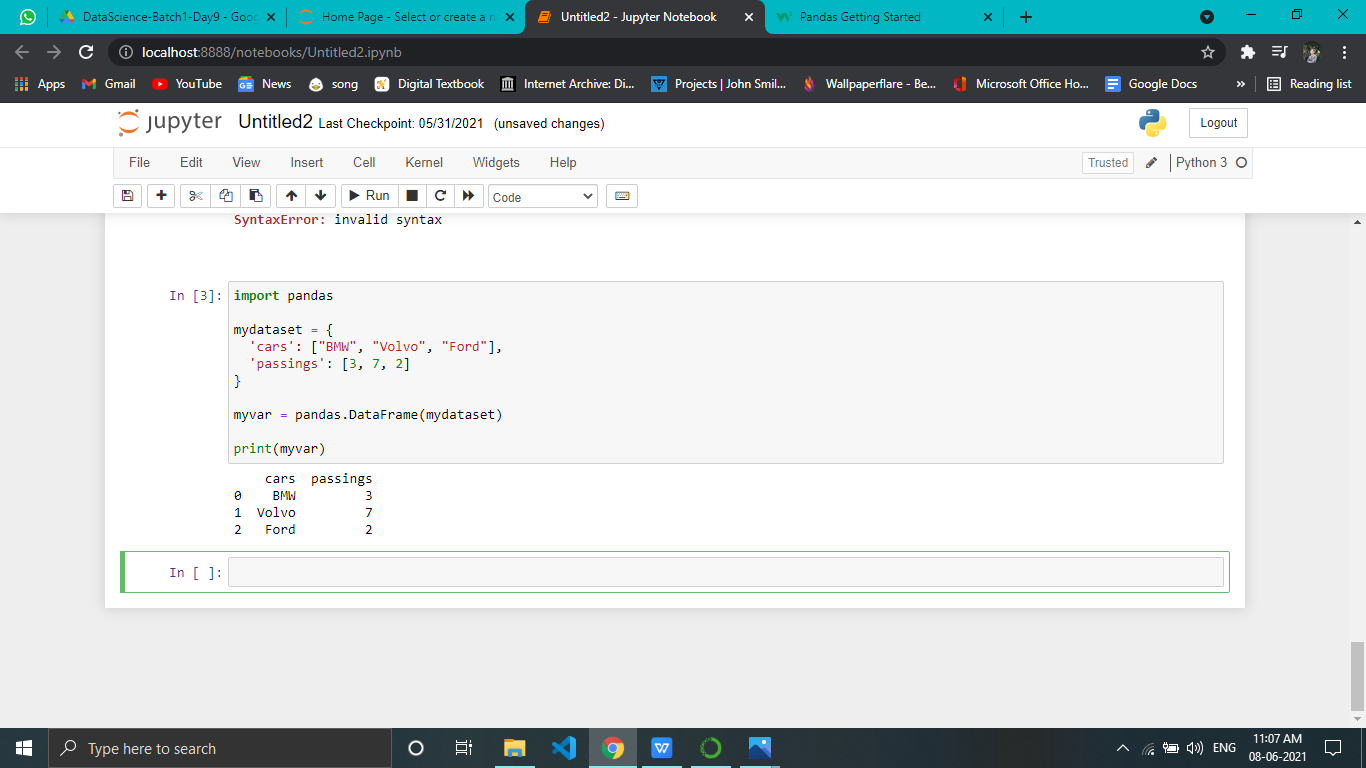
The NumPy ndarray object has a function called sort(), that will sort a specified array.

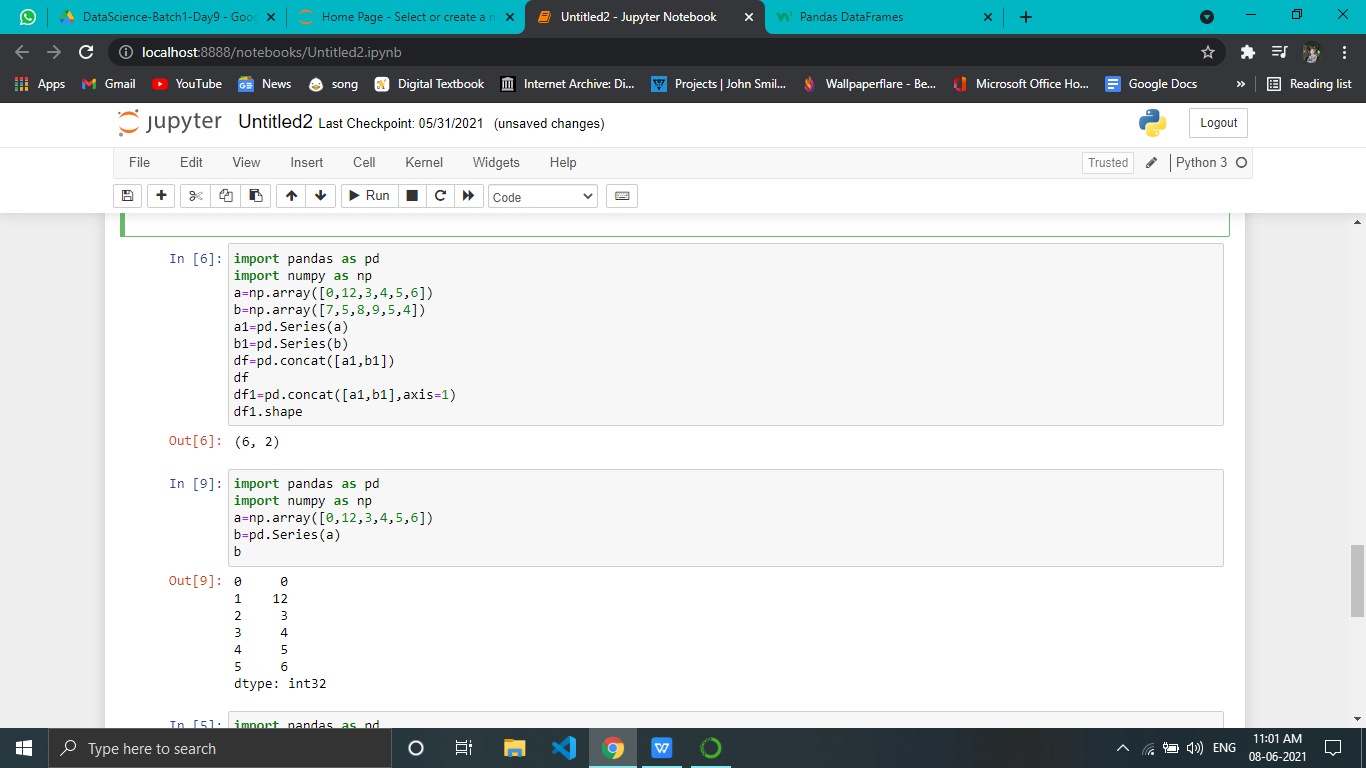
******

***Pandas methods:***

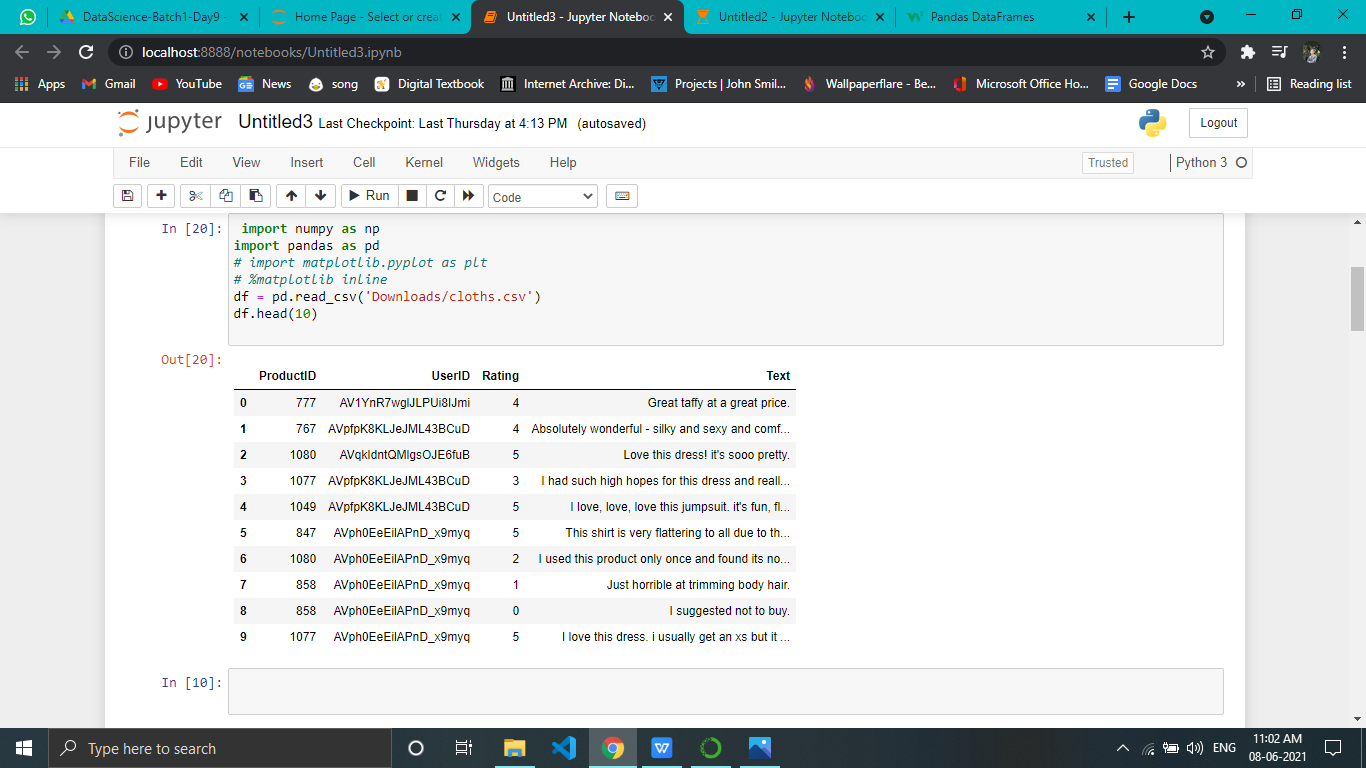
1. **read\_csv:** is an important pandas function to read csv files and do operations on it.



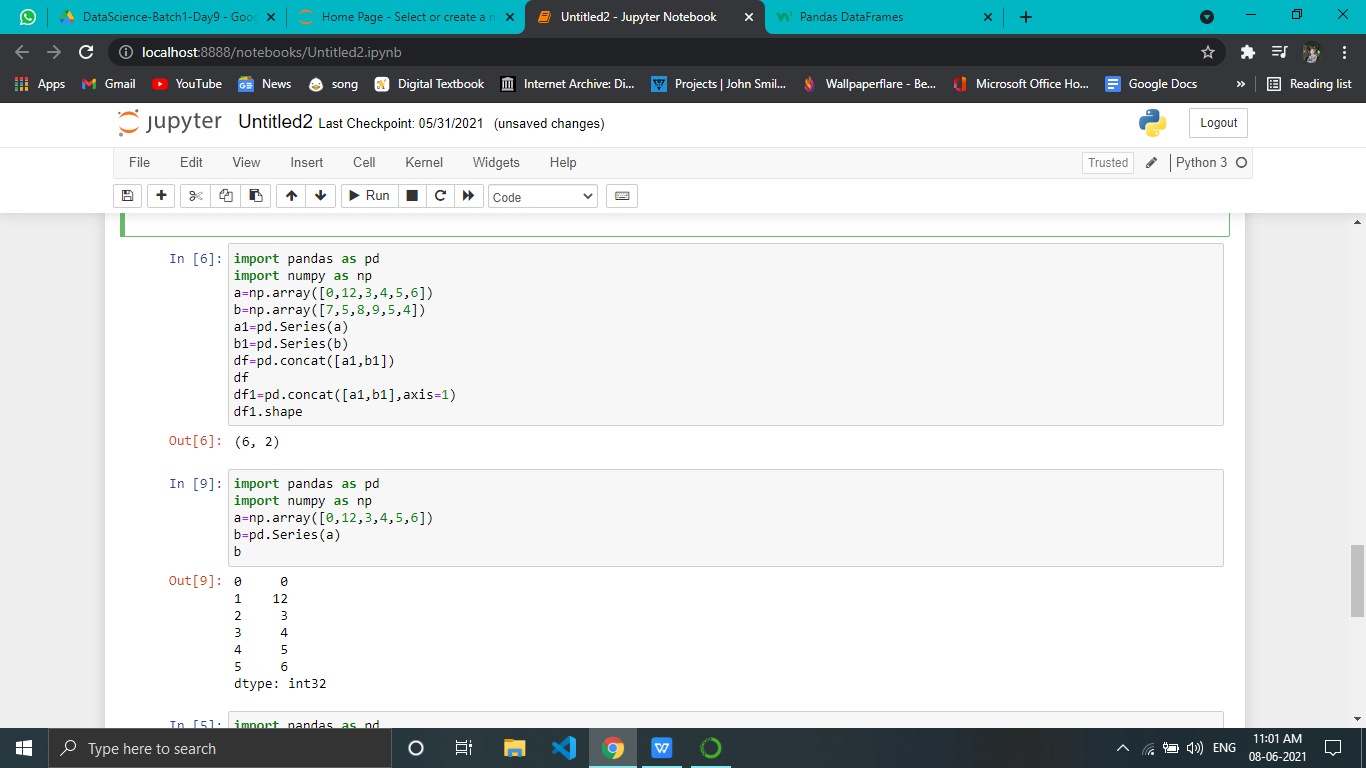
1. **Pandas DataFrame** is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows andcolumns.******
2. **Creating a series from array:** In order to create a series from array, we have to import a numpy module and have to use array() function.



4)head():This function returns the first n rows for the object based on position. It is useful for quickly testing if your object has the right type of data in it

******

***5)shape():****provide shape of array*

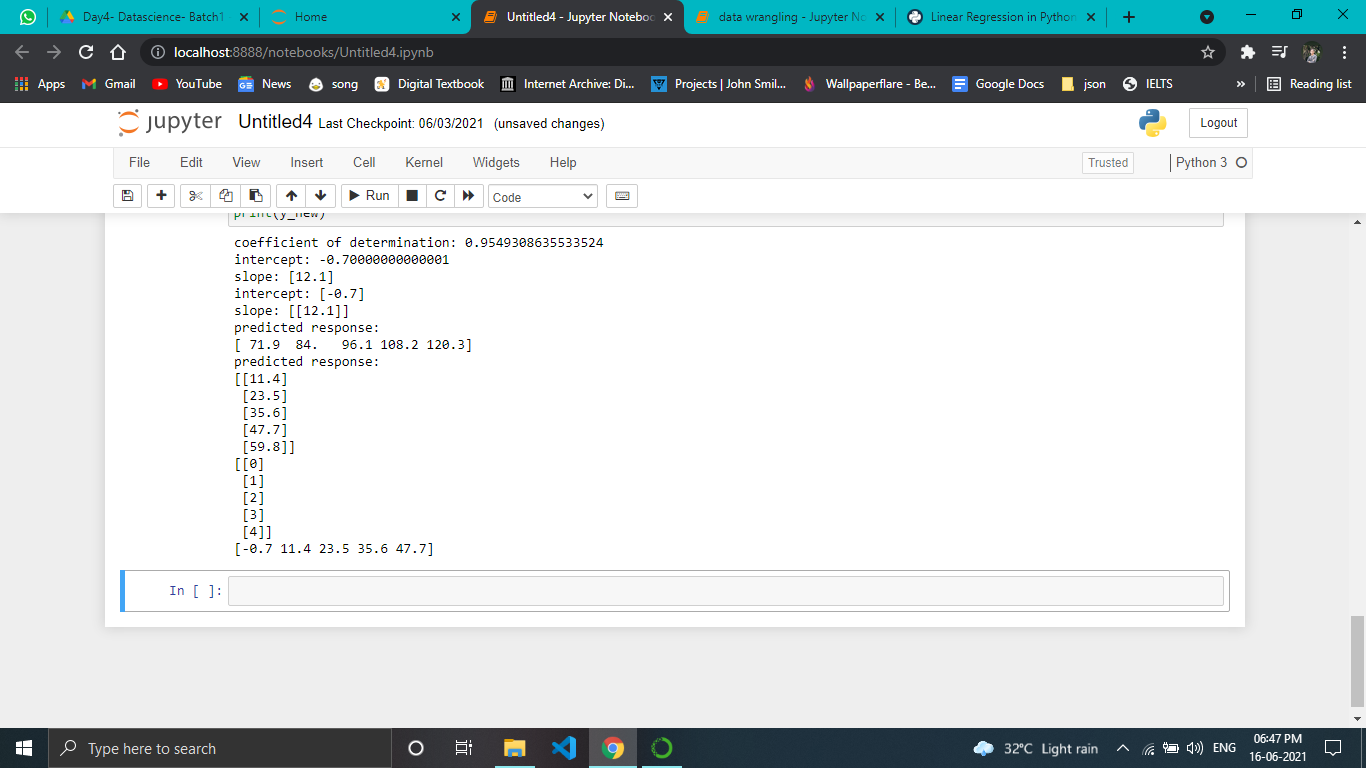
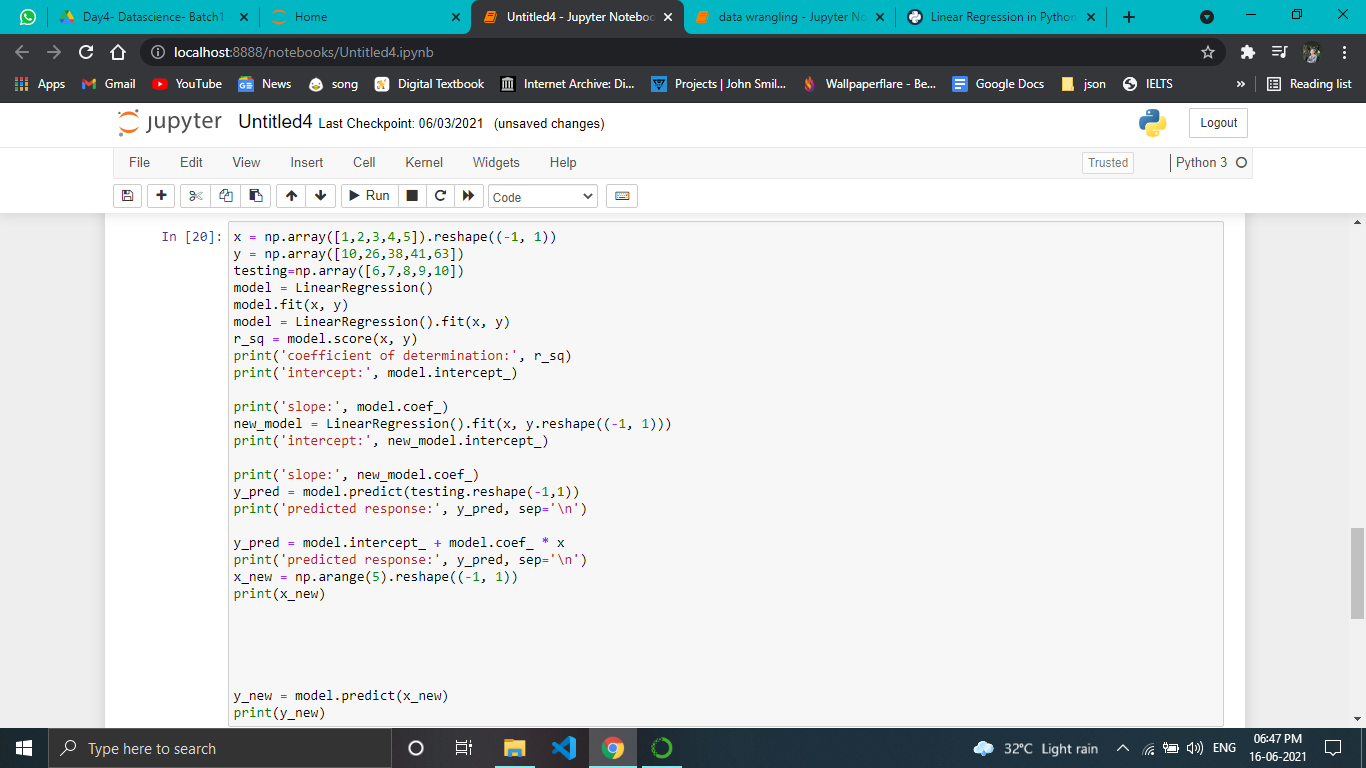
******

***2)Reshape(-1,1) explanation:***

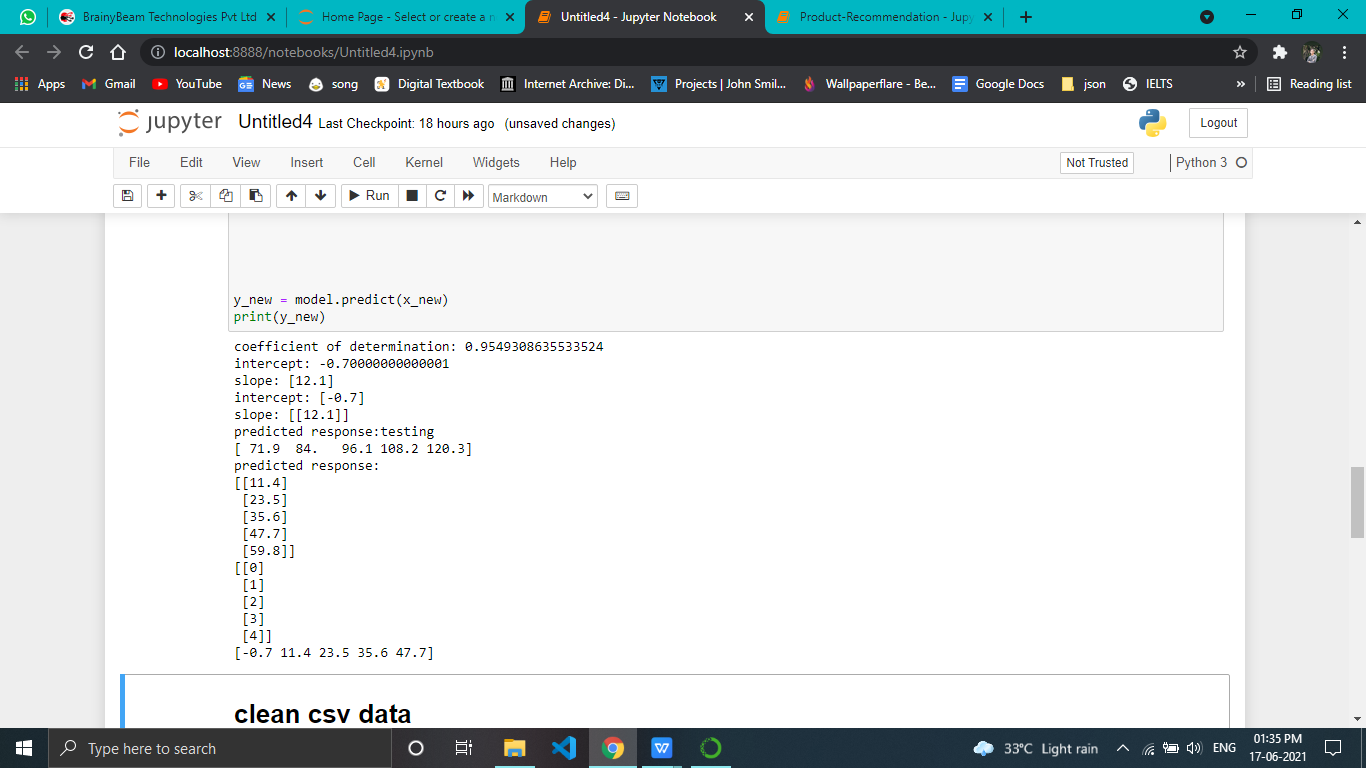
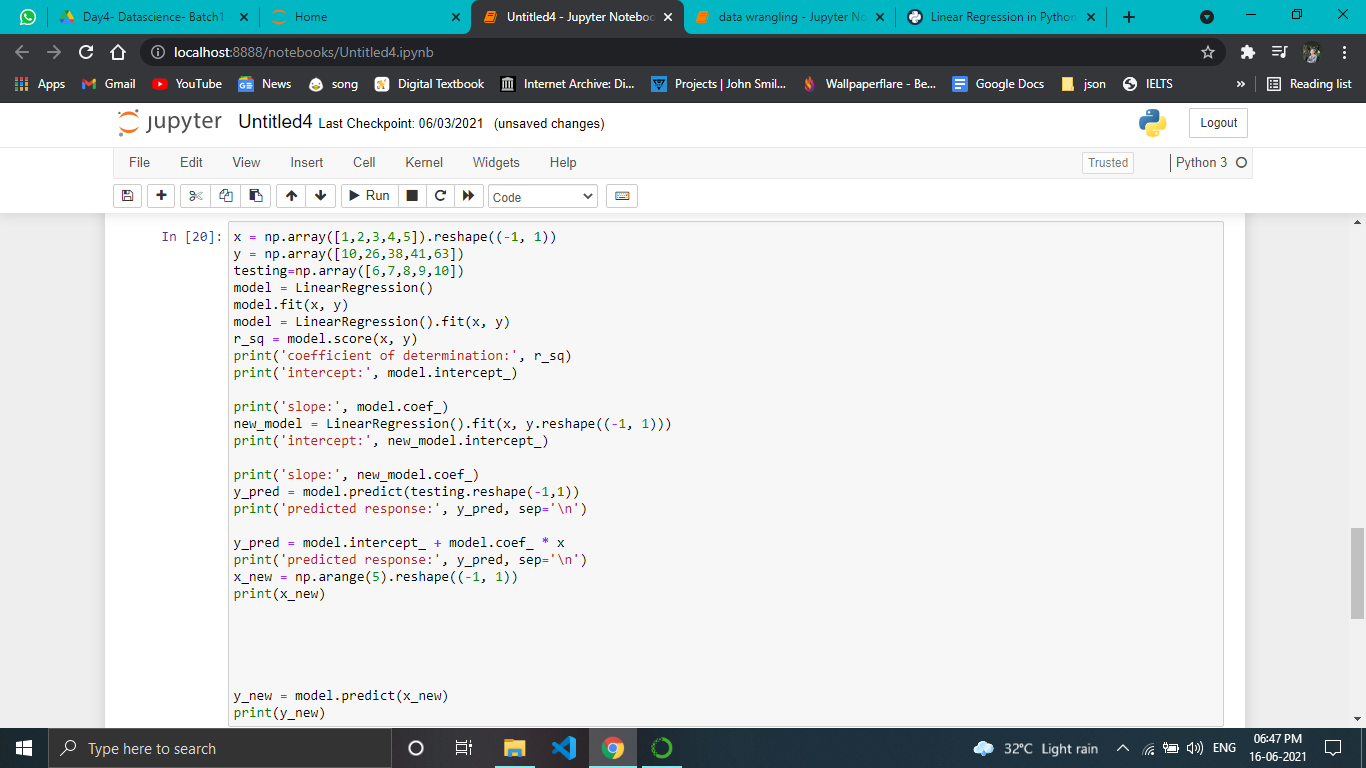
numpy allow us to give one of new shape parameter as -1 (eg: (2,-1) or (-1,3) but not (-1, -1)). It simply means that it is an unknown dimension and we want numpy to figure it out. And numpy will figure this by looking at the 'length of the array and remaining dimensions' and making sure it satisfies the above mentioned criteria.

**Reshape** your data using array.**reshape(-1**, **1**) if your data has a single feature. New shape as **(-1**, 2). row unknown, column 2. we get result new shape as (6, 2) z.**reshape(-1**, 2) array([[ **1**, 2], [ 3, 4], [ 5, 6], [ 7, 8], [ 9, 10], [11, 12]])

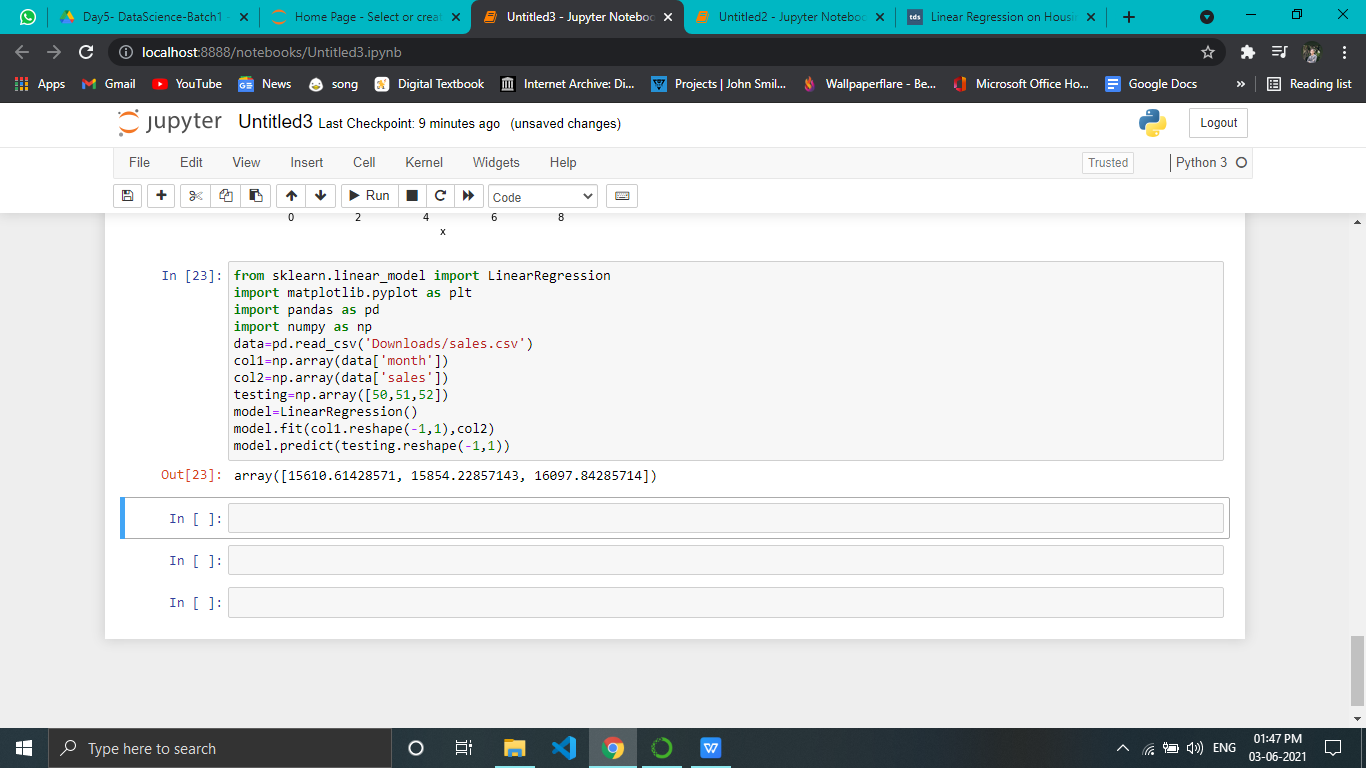
1. ***Linear regression working with mathematical equation:***

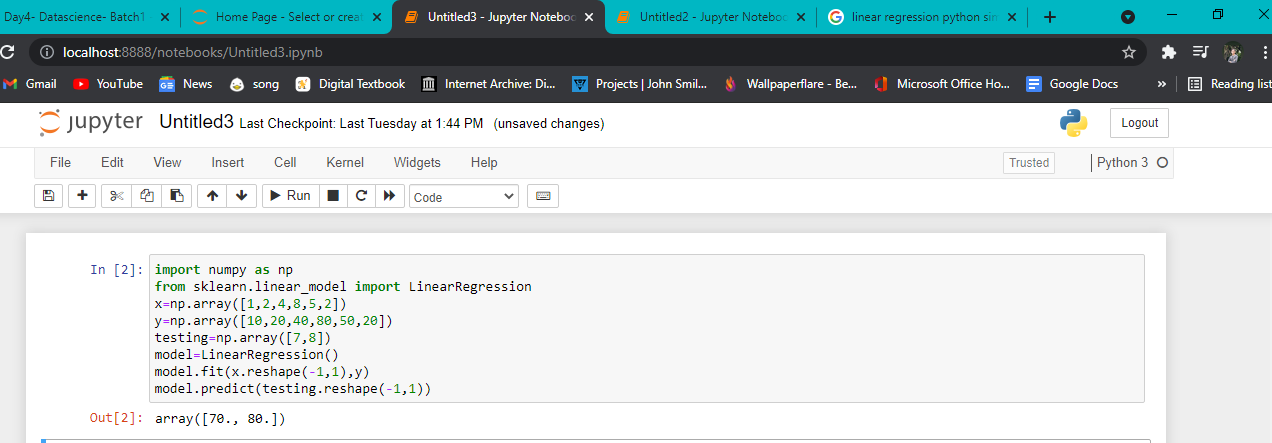
******

***Linear regression working with mathematical equation:***

******

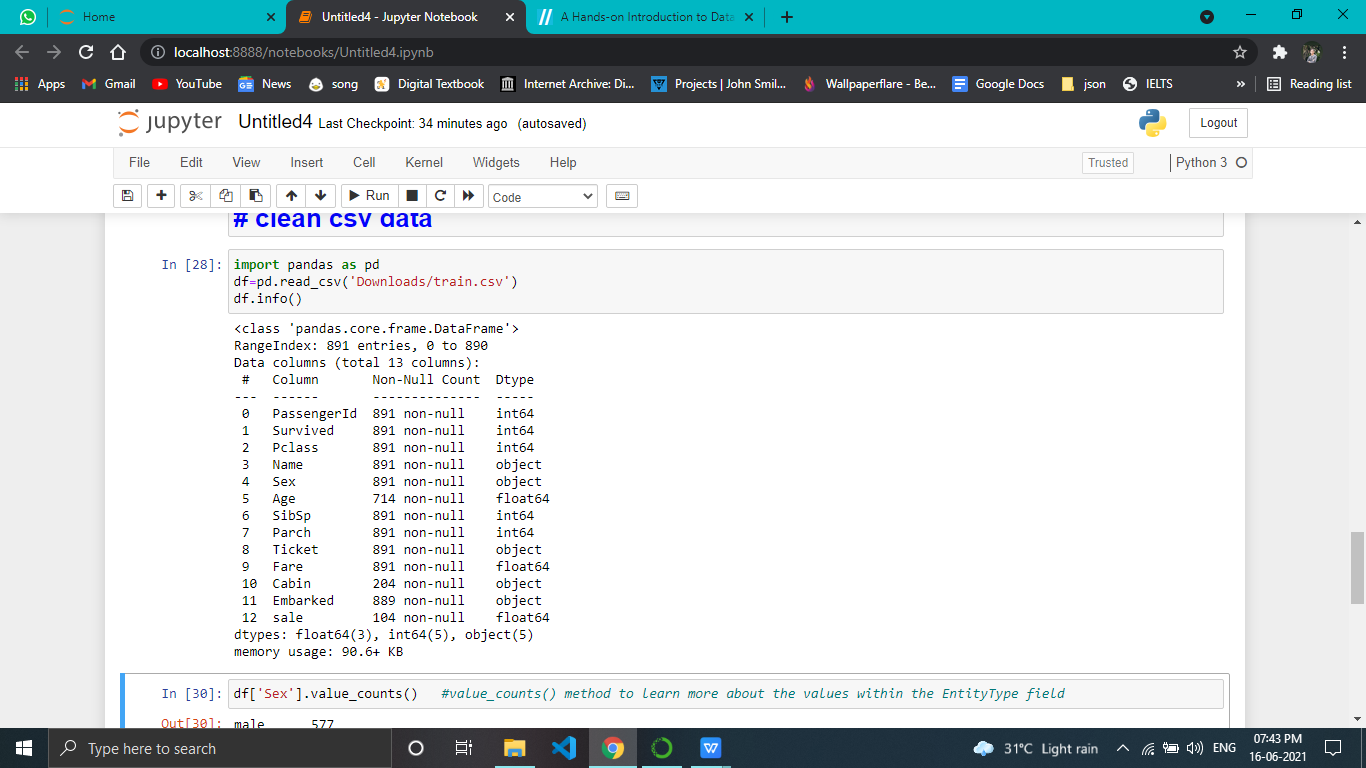
1. ***Sales prediction with csv:***

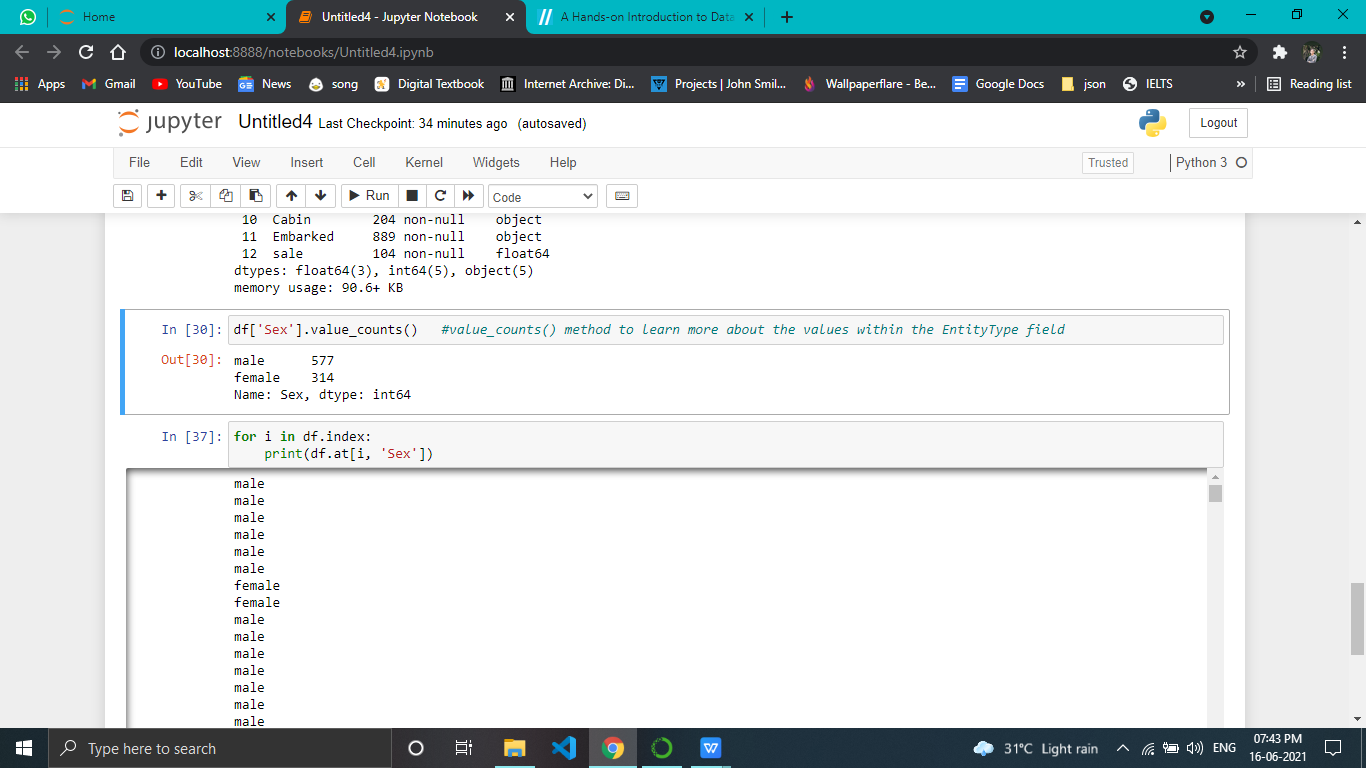


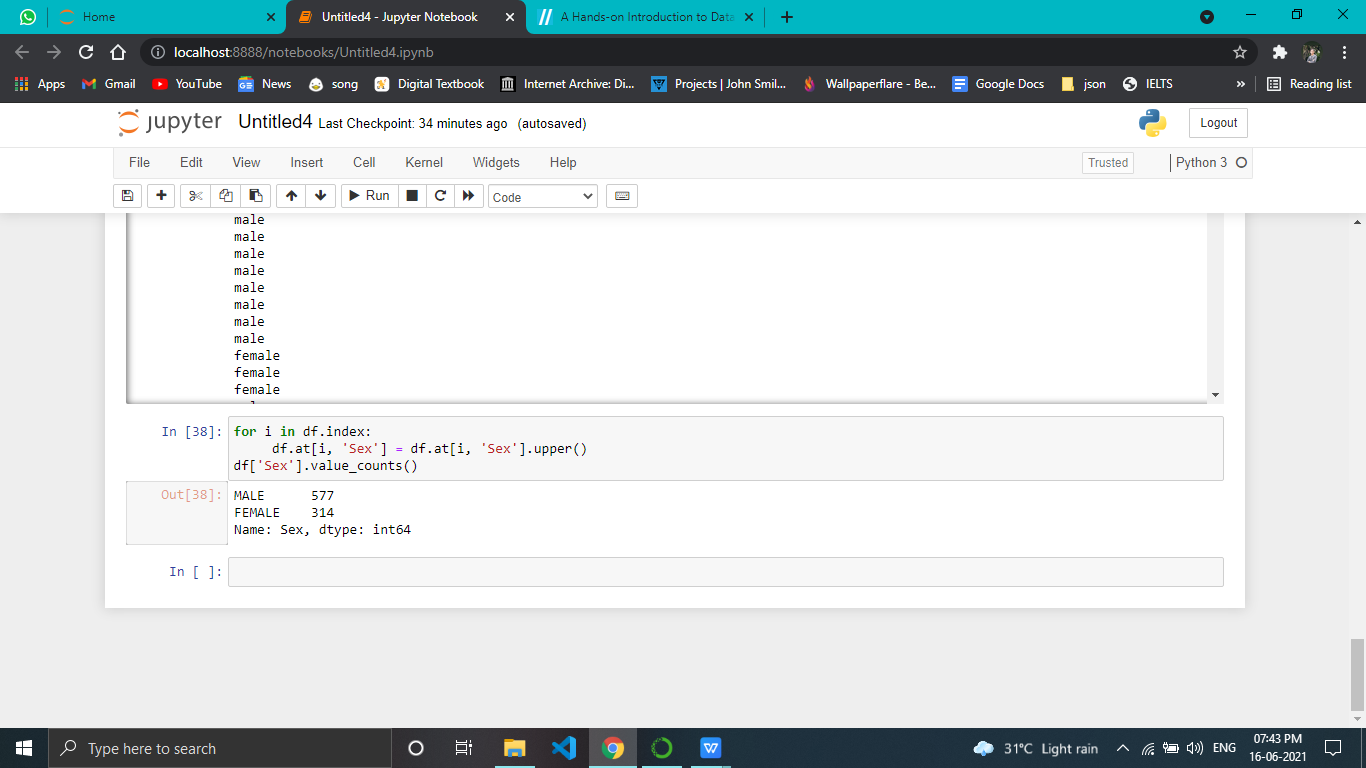
******

***Day 5 Task :1)Try to clean this data :***

***Sex data male and female convert into uppercase***

******

******

******

***2)decision tree explanation***

## **Decision Tree Algorithm**

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

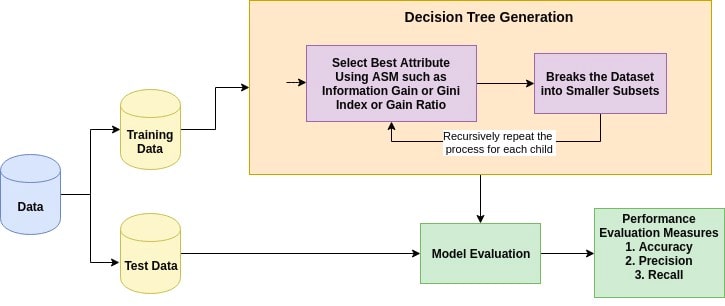


Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

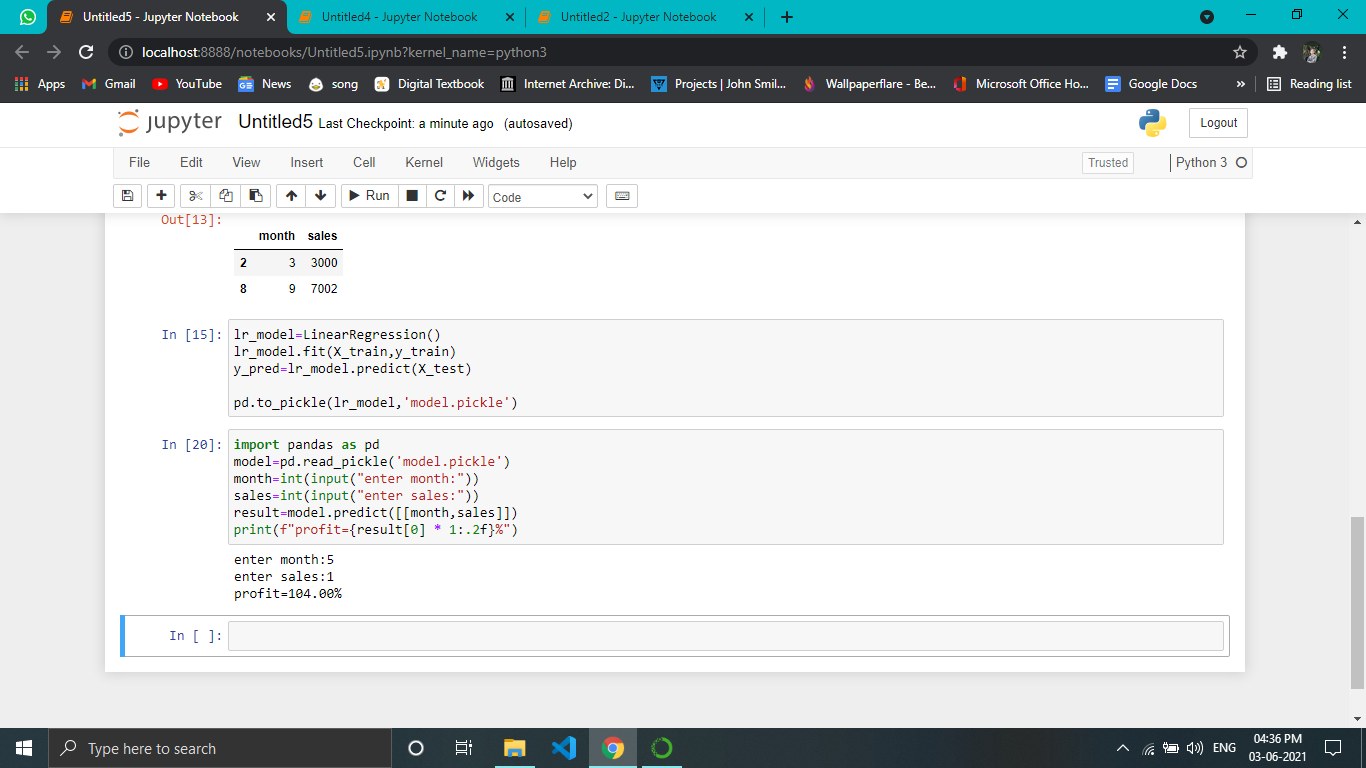
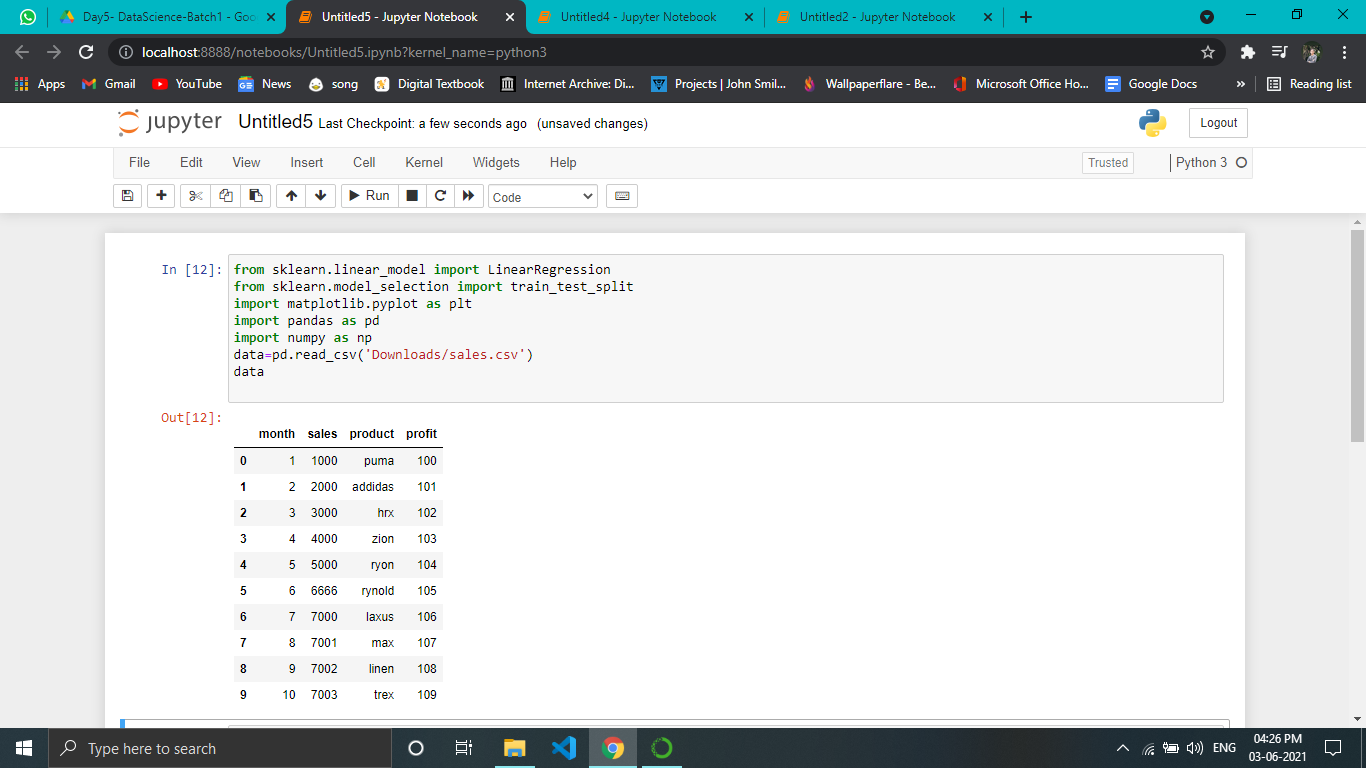
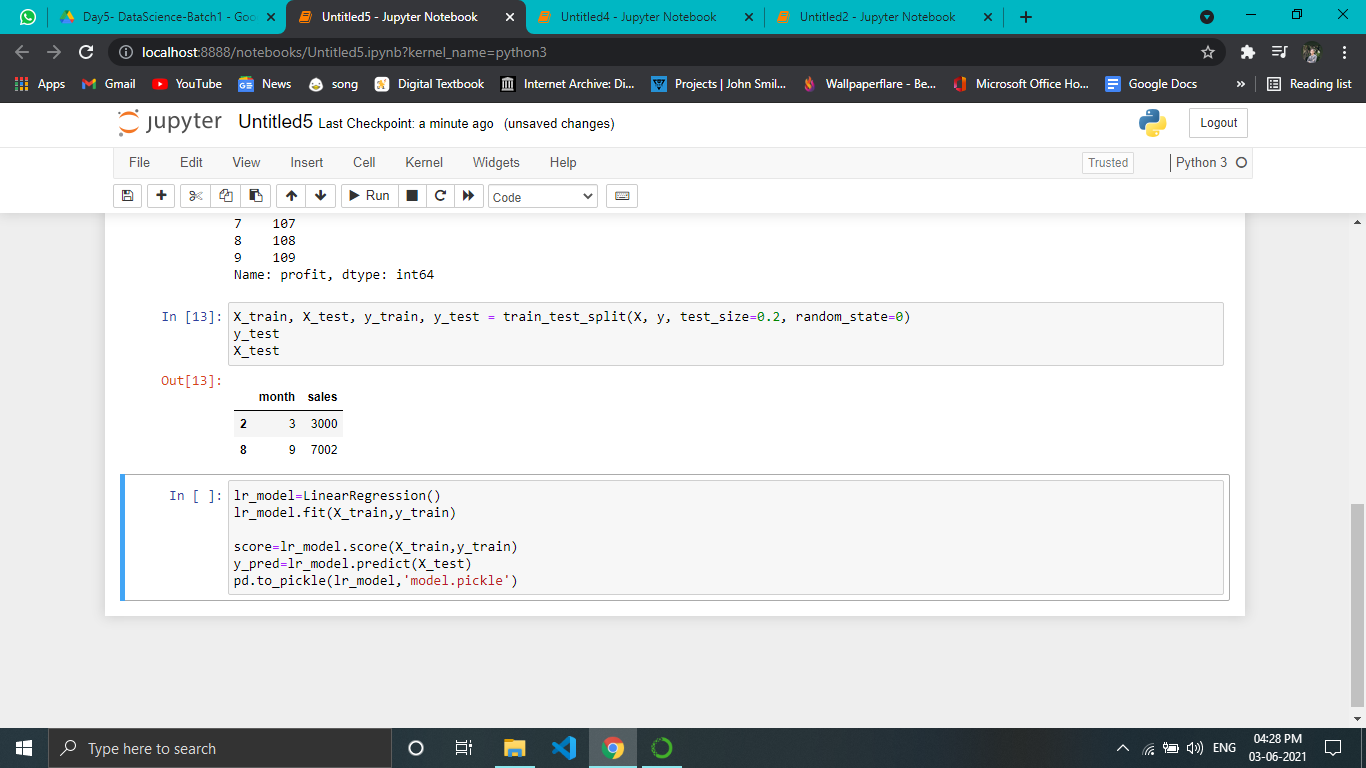
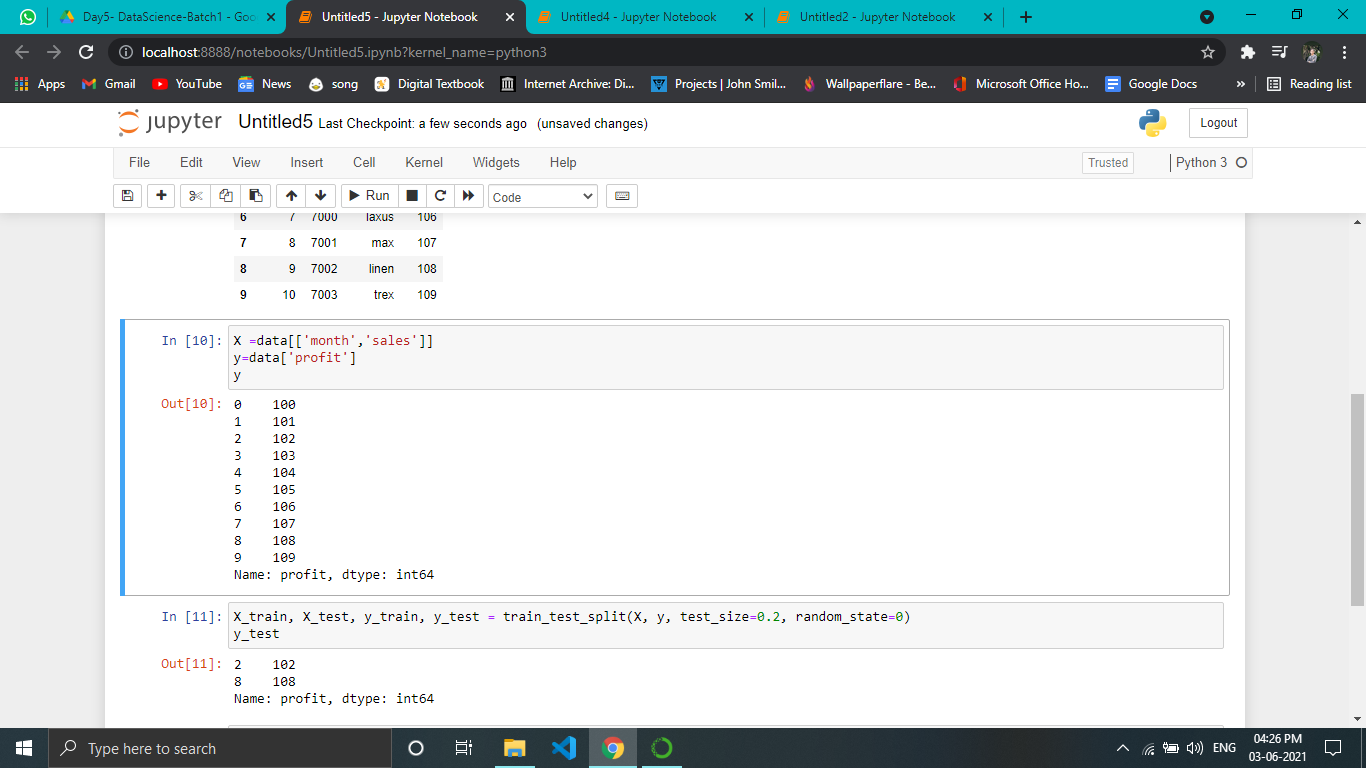
## **How does the Decision Tree algorithm work?**

The basic idea behind any decision tree algorithm is as follows:

1. Select the best attribute using Attribute Selection Measures(ASM) to split the records.
2. Make that attribute a decision node and breaks the dataset into smaller subsets.
3. Starts tree building by repeating this process recursively for each child until one of the condition will match:
   1. All the tuples belong to the same attribute value.
   2. There are no more remaining attributes.
   3. There are no more instances.



***3) random state working***

******

***Day 6 Task :***

***1)5 parameter used in decision tree in python***

****criterion***{“gini”, “entropy”}, default=”gini”***

The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

****splitter***{“best”, “random”}, default=”best”***

The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.

****max\_depth :***int, default=None***

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

****min\_samples\_split :** *int or float, default=2***

The minimum number of samples required to split an internal node:

If int, then consider min\_samples\_split as the minimum number.

If float, then min\_samples\_split is a fraction and ceil(min\_samples\_split \* n\_samples) are the minimum number of samples for each split.

*Changed in version 0.18:*Added float values for fractions.

****max\_features***int, float or {“auto”, “sqrt”, “log2”}, default=None***

The number of features to consider when looking for the best split:

If int, then consider max\_features features at each split.

If float, then max\_features is a fraction and int(max\_features \* n\_features) features are considered at each split.

If “auto”, then max\_features=sqrt(n\_features).

If “sqrt”, then max\_features=sqrt(n\_features).

If “log2”, then max\_features=log2(n\_features).

If None, then max\_features=n\_features.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than max\_features features.

1. ***mention data cleaning methods and its working.***

**DataFrame.isnull()**

Detect missing values.

Return a boolean same-sized object indicating if the values are NA. NA values, such as None or **numpy.NaN**, gets mapped to True values. Everything else gets mapped to False values. Characters such as empty strings '' or **numpy.inf** are not considered NA values

# DataFrame.dropna()

If your dataset consists of null values, we can use the dropna() function to analyze and drop the rows/columns in the dataset.

## Syntax:

1. DataFrameName.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)

# DataFrame.fillna()

We can use the fillna() function to fill the null values in the dataset.

## Syntax:

1. DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None,

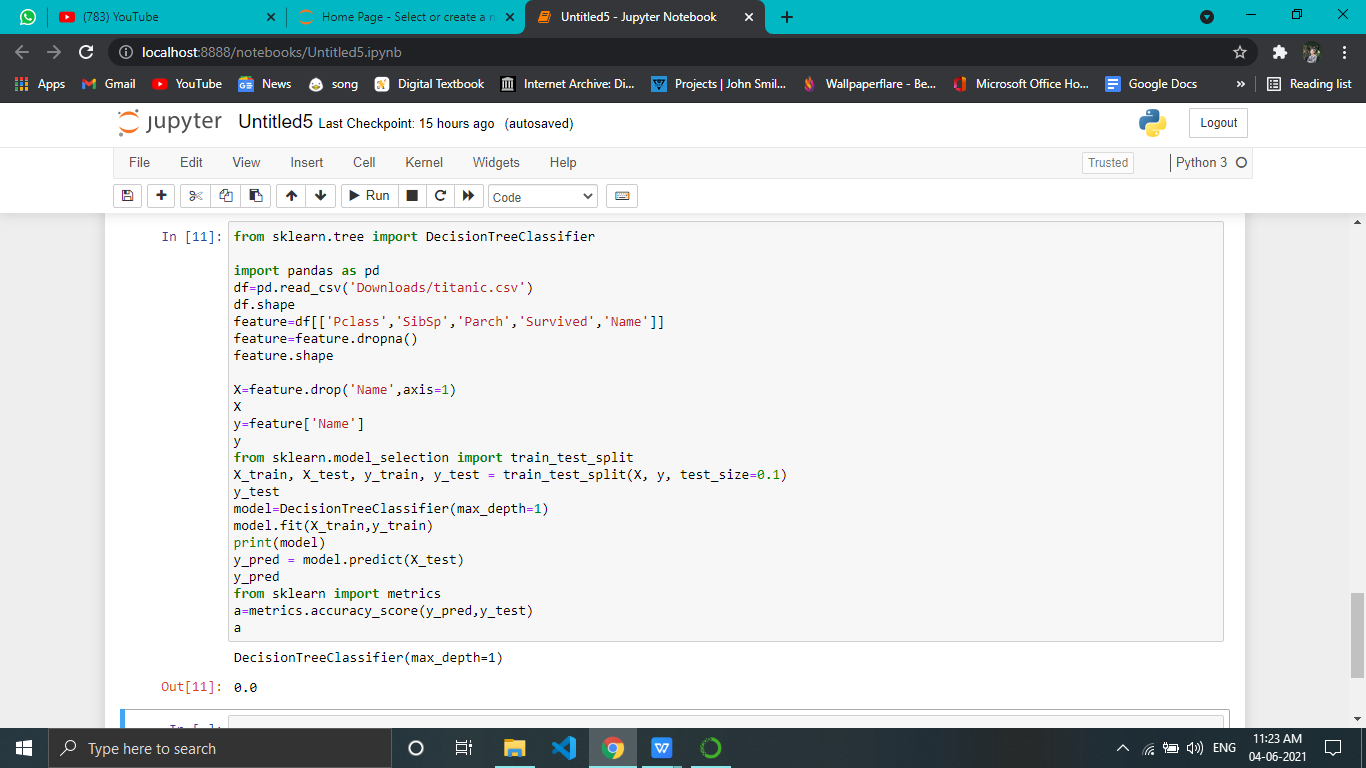
# Pandas DataFrame.replace()

Pandas replace() is a very rich function that is used to replace a **string, regex, dictionary, list,** and **series** from the DataFrame. The values of the DataFrame can be replaced with other values dynamically. It is capable of working with the Python regex(regular expression).

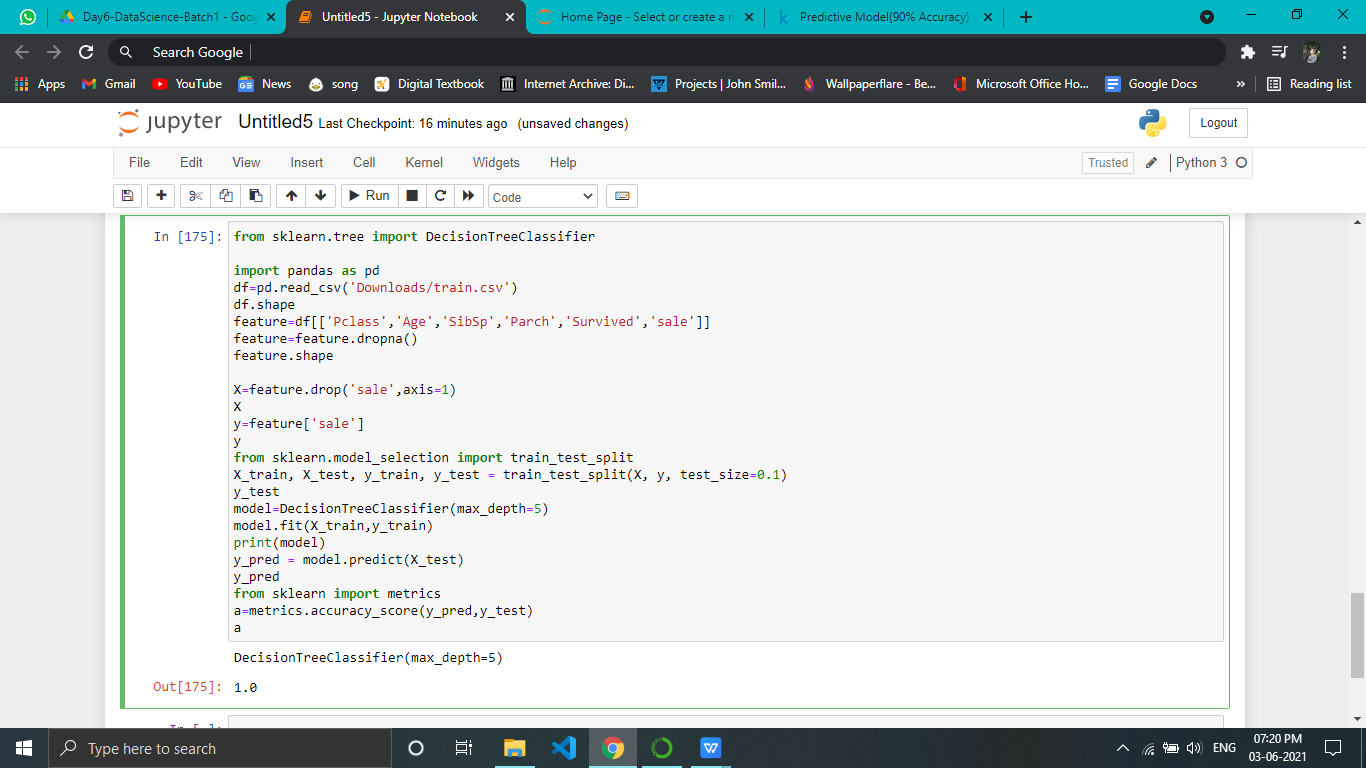
It differs from updating with **.loc** or **.iloc**, which requires you to specify a location where you want to update with some value.

## Syntax:

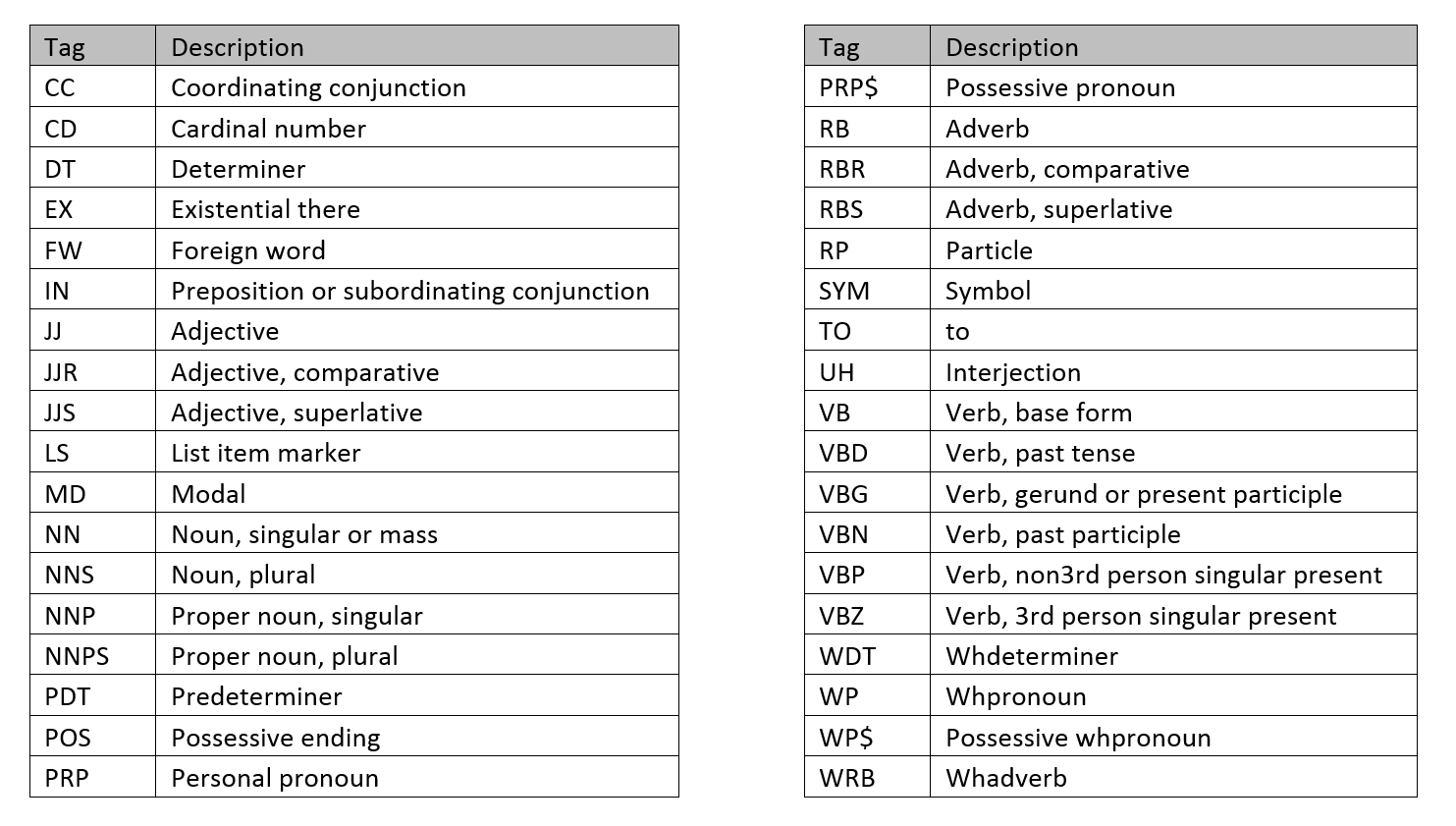
1. DataFrame.replace(to\_replace=None, value=None, inplace=False, limit=None, regex=False, method='pad', axis=None)
2. ***make a diagram explainig dicision tree parameter with titanic dataset equation***

******

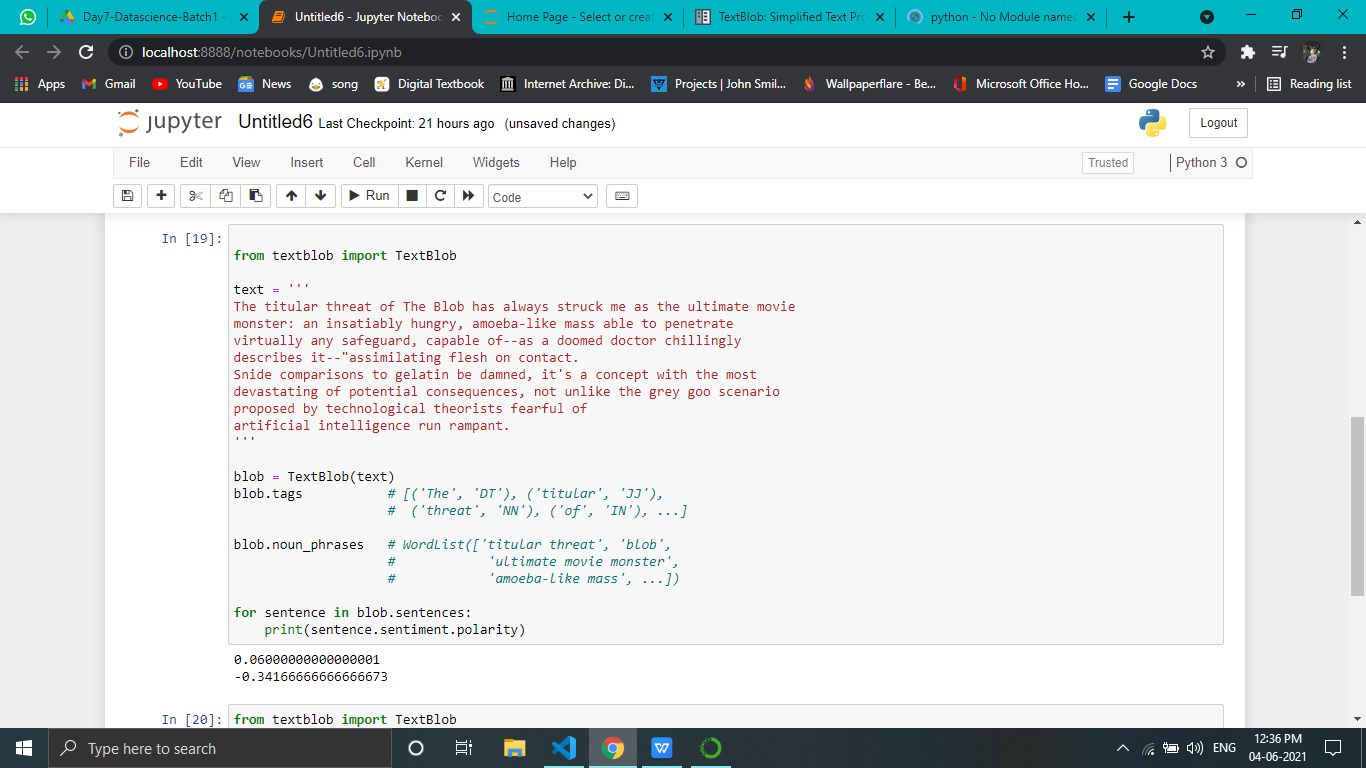
***4)90% accuracy from dataset***

******

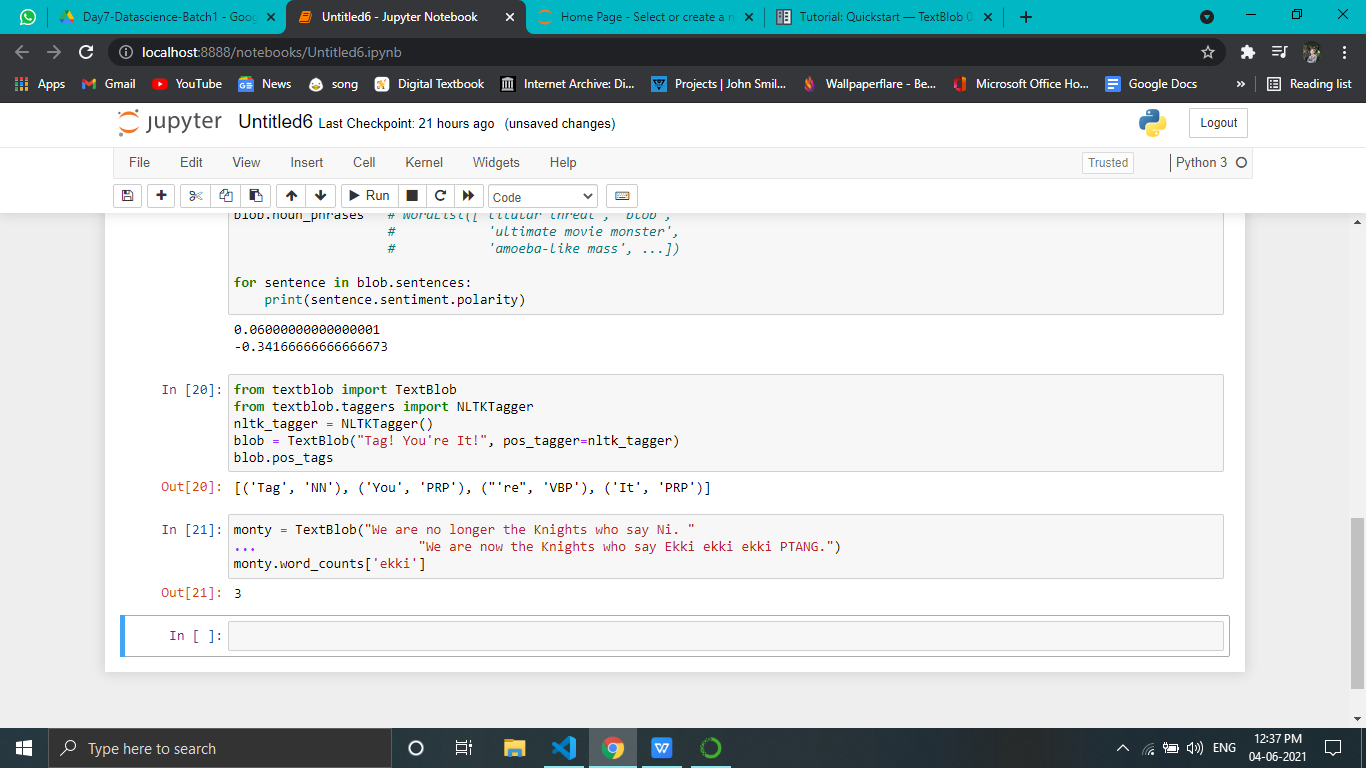
***Day 7 Task:***

******

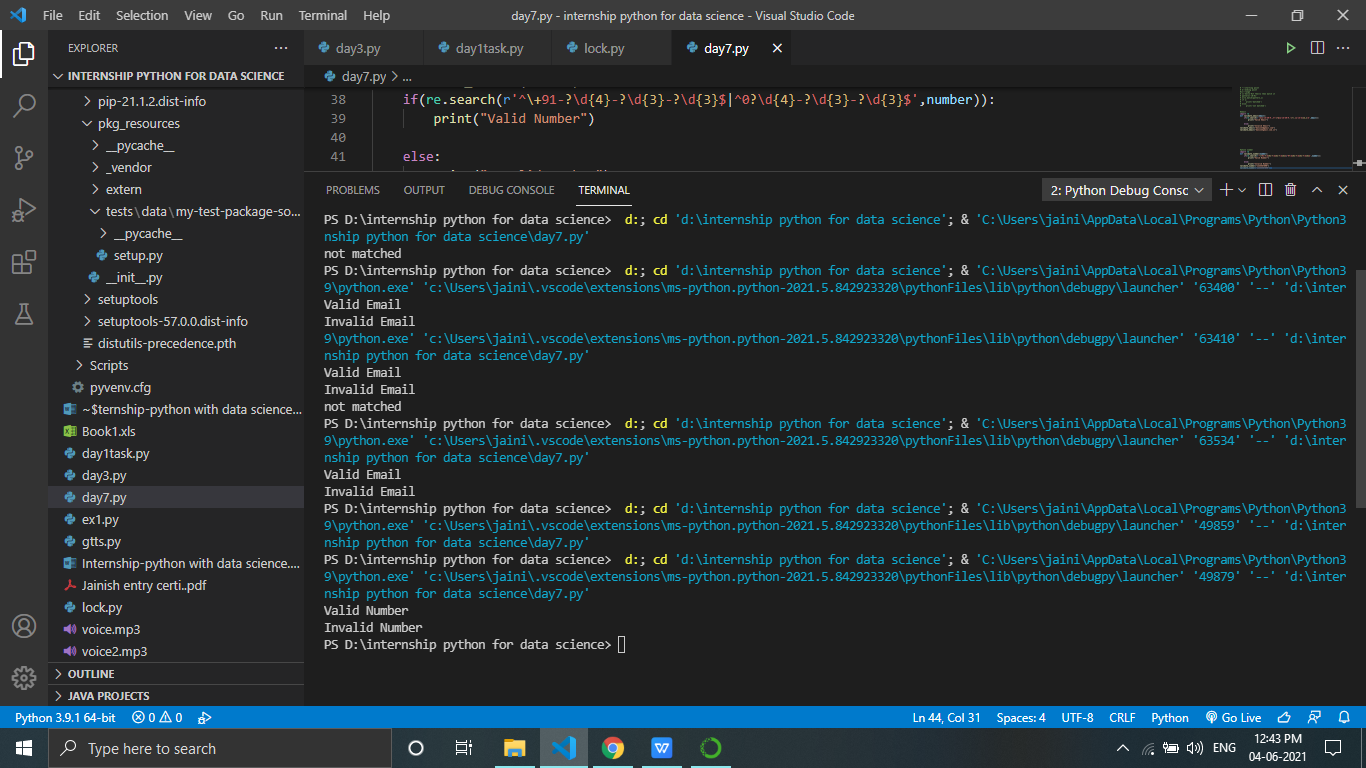
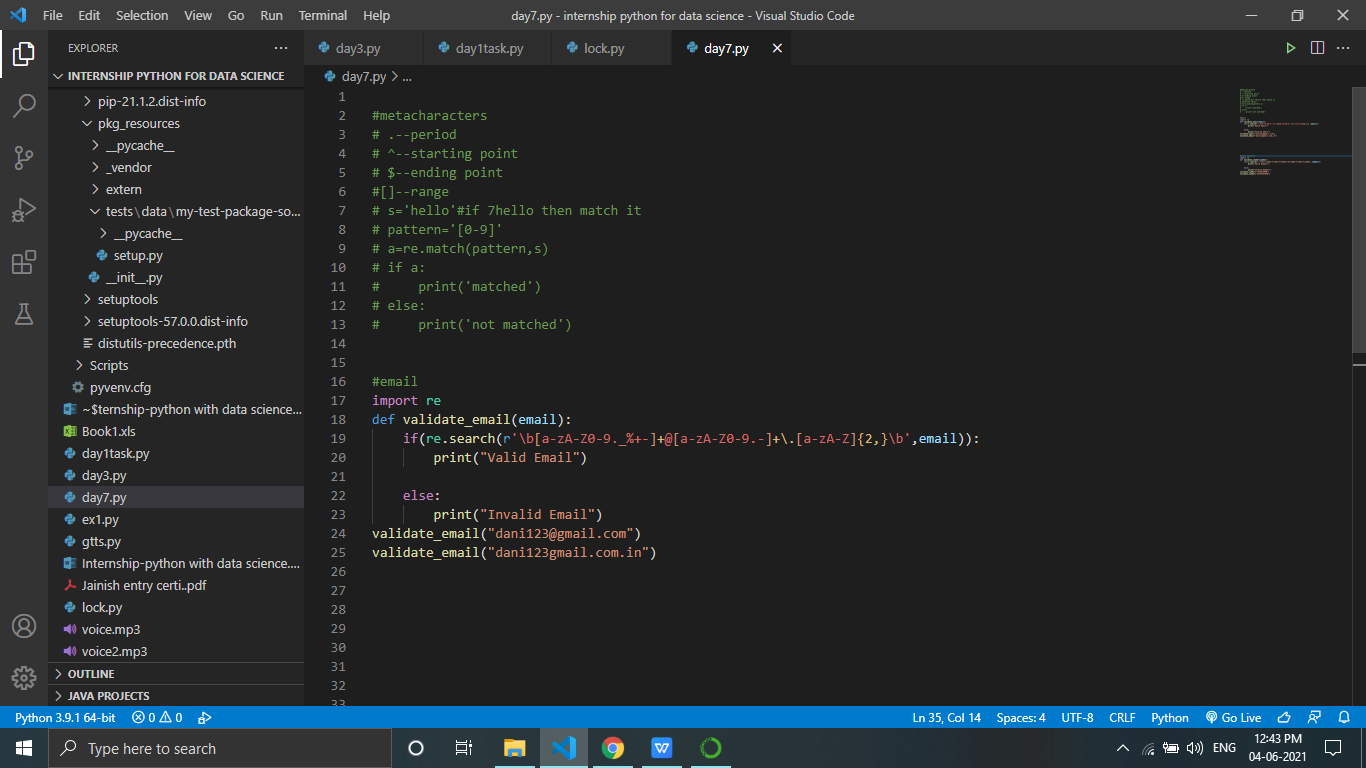
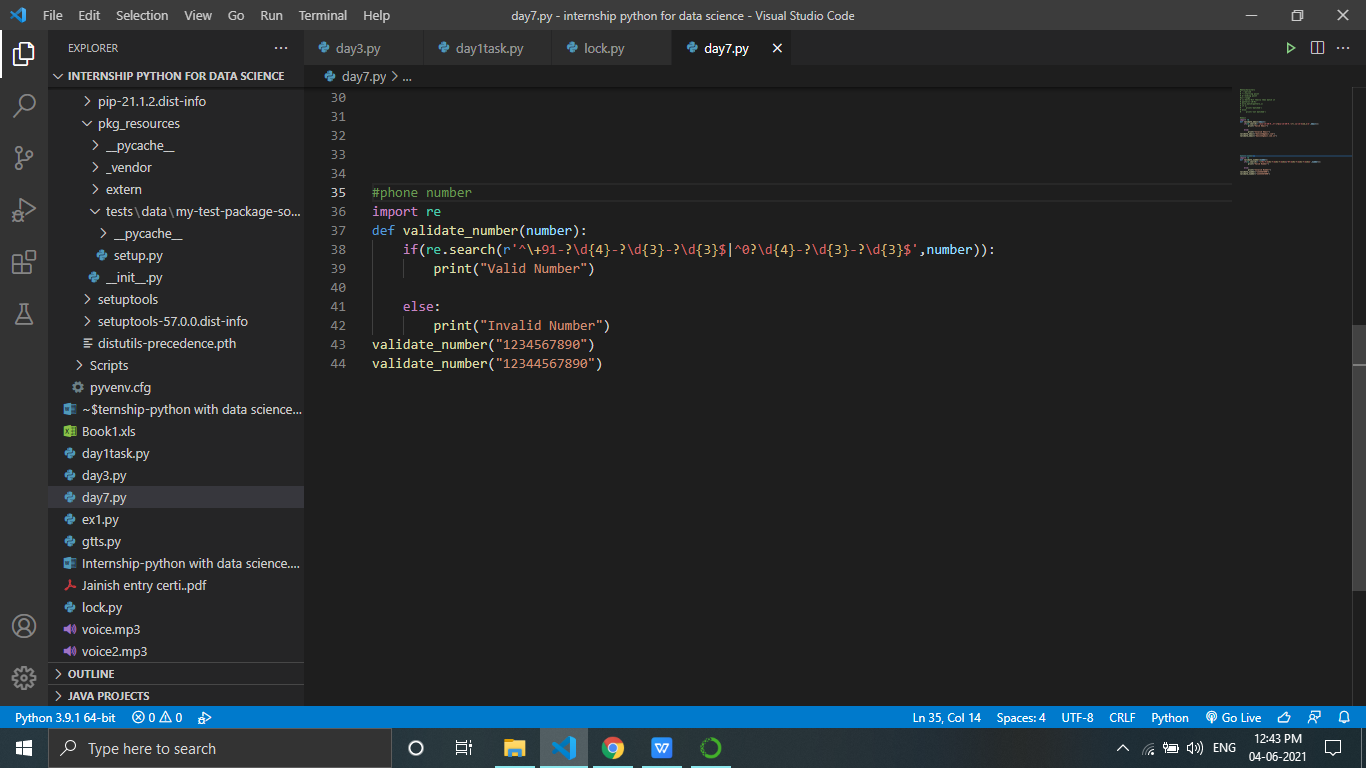
1. ***find polarity of unique products using apply with function.***

******

***2)frequency distribution and pos tag from nltk.***

******

***3)make list of meta characters and make the pattern of email and phone number.***

******

1. ***explainTF-IDF with example***

# Introduction: TF-IDF

TF-IDF stands for ****“Term Frequency — Inverse Document Frequency”****. This is a technique to quantify a word in documents, we generally compute a weight to each word which signifies the importance of the word in the document and corpus. This method is a widely used technique in Information Retrieval and Text Mining.

TF-IDF = Term Frequency (TF) \* Inverse Document Frequency (IDF)

## Terminology

* t — term (word)
* d — document (set of words)
* N — count of corpus
* corpus — the total document set

## TF-IDF

TF-IDF is the product of Term Frequency and Inverse Document Frequency. Here’s the formula for TF-IDF calculation.

TF-IDF = Term Frequency (TF) \* Inverse Document Frequency (IDF)

What are Term Frequency and Inverse Document Frequency you ask? let’s see what they actually are.

### What is Term Frequency?

It is the measure of the frequency of words in a document. It is the ratio of the number of times the word appears in a document compared to the total number of words in that document.

tf(t,d) = count of t in d / number of words in d

### What is ****Inverse Document Frequency****?

The words that occur rarely in the corpus have a high IDF score. It is the log of the ratio of the number of documents to the number of documents containing the word.

We take log of this ratio because when the corpus becomes large IDF values can get large causing it to explode hence taking log will dampen this effect.

we cannot divide by 0, we smoothen the value by adding 1 to the denominator.

idf(t) = log(N/(df + 1))

## Step by Step Implementation of the TF-IDF Model

Let’s get right to the implementation part of the TF-IDF Model in Python.

### 1. Preprocess the data

We’ll start with preprocessing the text data, and make a vocabulary set of the words in our training data and assign a unique index for each word in the set.

|  |
| --- |
| #Importing required module  **import** numpy as np  **from** nltk.tokenize **import**  word\_tokenize    #Example text corpus for our tutorial  text **=** ['Topic sentences are similar to mini thesis statements.\          Like a thesis statement, a topic sentence has a specific \          main point. Whereas the thesis **is** the main point of the essay',\          'the topic sentence **is** the main point of the paragraph.\          Like the thesis statement, a topic sentence has a unifying function. \          But a thesis statement **or** topic sentence alone doesn’t guarantee unity.', \          'An essay **is** unified **if** all the paragraphs relate to the thesis,\          whereas a paragraph **is** unified **if** all the sentences relate to the topic sentence.']    #Preprocessing the text data  sentences **=** []  word\_set **=** []    **for** sent **in** text:      x **=** [i.lower() **for**  i **in** word\_tokenize(sent) **if** i.isalpha()]      sentences.append(x)  **for** word **in** x:  **if** word **not** **in** word\_set:              word\_set.append(word)    #Set of vocab  word\_set **=** set(word\_set)  #Total documents in our corpus  total\_documents **=** len(sentences)    #Creating an index for each word in our vocab.  index\_dict **=** {} #Dictionary to store index for each word  i **=** 0  **for** word **in** word\_set:      index\_dict[word] **=** i      i **+=** 1 |

### 2. Create a dictionary for keeping count

We then [create a dictionary](https://www.askpython.com/python/dictionary/python-dictionary-dict-tutorial) to keep the count of the number of documents containing the given word.

|  |
| --- |
| #Create a count dictionary    **def** count\_dict(sentences):      word\_count **=** {}  **for** word **in** word\_set:          word\_count[word] **=** 0  **for** sent **in** sentences:  **if** word **in** sent:                  word\_count[word] **+=** 1  **return** word\_count    word\_count **=** count\_dict(sentences) |

### 3. Define a function to calculate Term Frequency

Now, let’s define a function to count the term frequency (TF) first.

|  |
| --- |
| #Term Frequency  **def** termfreq(document, word):      N **=** len(document)      occurance **=** len([token **for** token **in** document **if** token **==** word])  **return** occurance**/**N |

### 4. Define a function calculate Inverse Document Frequency

Now, with the term frequency function set, let’s define another function for the Inverse Document Frequency (IDF)

|  |
| --- |
| #Inverse Document Frequency    **def** inverse\_doc\_freq(word):  **try**:          word\_occurance **=** word\_count[word] **+** 1  **except**:          word\_occurance **=** 1  **return** np.log(total\_documents**/**word\_occurance) |

### 5. Combining the TF-IDF functions

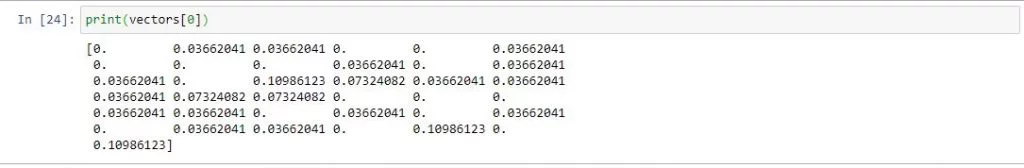
Let’s create another function to combine both the TF and IDF functions from above to give us our desired output for the TF-IDF model.

|  |
| --- |
| **def** tf\_idf(sentence):      tf\_idf\_vec **=** np.zeros((len(word\_set),))  **for** word **in** sentence:          tf **=** termfreq(sentence,word)          idf **=** inverse\_doc\_freq(word)            value **=** tf**\***idf          tf\_idf\_vec[index\_dict[word]] **=** value  **return** tf\_idf\_vec |

### 6. Apply the TF-IDF Model to our text

The implementation of the TF-IDF model in Python is complete. Now, let’s pass the text corpus to the function and see what the output vector looks like.

|  |
| --- |
| #TF-IDF Encoded text corpus  vectors **=** []  **for** sent **in** sentences:      vec **=** tf\_idf(sent)      vectors.append(vec)    print(vectors[0]) |

TF-IDF Encoded Vector

Now, if the model encounters an unknown word other than the vocab, it will give us a Key error as we did not account for any unknown tokens.

## **Day9-task 1) Explain three techniques of stemming.**

## **What is Stemming?**

****Stemming**** is a method of normalization of words in Natural Language Processing. It is a technique in which a set of words in a sentence are converted into a sequence to shorten its lookup. In this method, the words having the same meaning but have some variations according to the context or sentence are normalized.

### PorterStemmer class

NLTK has **PorterStemmer** class with the help of which we can easily implement Porter Stemmer algorithms for the word we want to stem. This class knows several regular word forms and suffixes with the help of which it can transform the input word to a final stem. The resulting stem is often a shorter word having the same root meaning

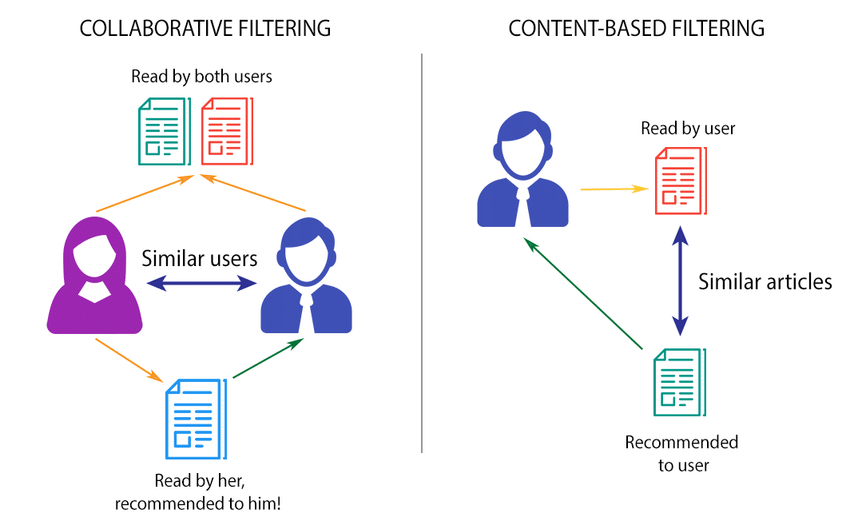
### LancasterStemmer class

NLTK has **LancasterStemmer** class with the help of which we can easily implement Lancaster Stemmer algorithms for the word we want to stem.

### RegexpStemmer class

NLTK has **RegexpStemmer** class with the help of which we can easily implement Regular Expression Stemmer algorithms. It basically takes a single regular expression and removes any prefix or suffix that matches the expression.

**Day10-task 1)explain collaborative and content based filtering with example.**

******

Collaborative Filtering

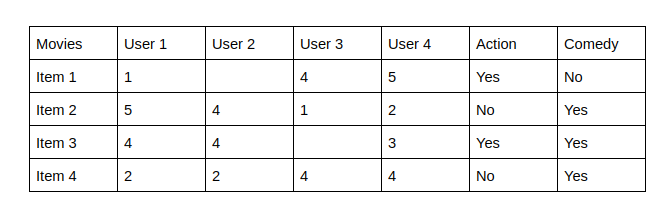
There are two classes of Collaborative Filtering:

User-based, which measures the similarity between target users and other users.

Item-based, which measures the similarity between the items that target users rate or interact with and other items.

 Collaborative filtering encompasses techniques for matching people with similar interests and making recommendations on this basis.

****Content-based filtering system:****Content-Based recommender system tries to guess the features or behavior of a user given the item’s features, he/she reacts positively to.



The last two columns Action and Comedy Describe the Genres of the movies. Now, given these genres, we can know which users like which genre, as a result, we can obtain features corresponding to that particular user, depending on how he/she reacts to movies of that genre.

Once, we know the likings of the user we can embed him/her in an embedding space using the feature vector generated and recommend him/her according to his/her choice. During recommendation, the similarity metrics (We will talk about it in a bit) are calculated from the item’s feature vectors and the user’s preferred feature vectors from his/her previous records. Then, the top few are recommended.

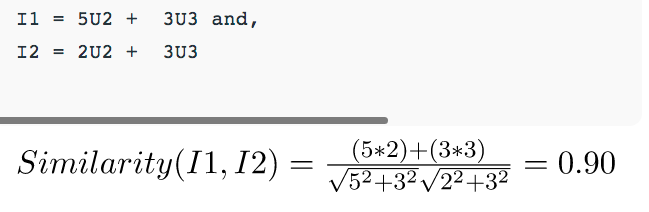
Content-based filtering does not require other users' data during recommendations to one user.

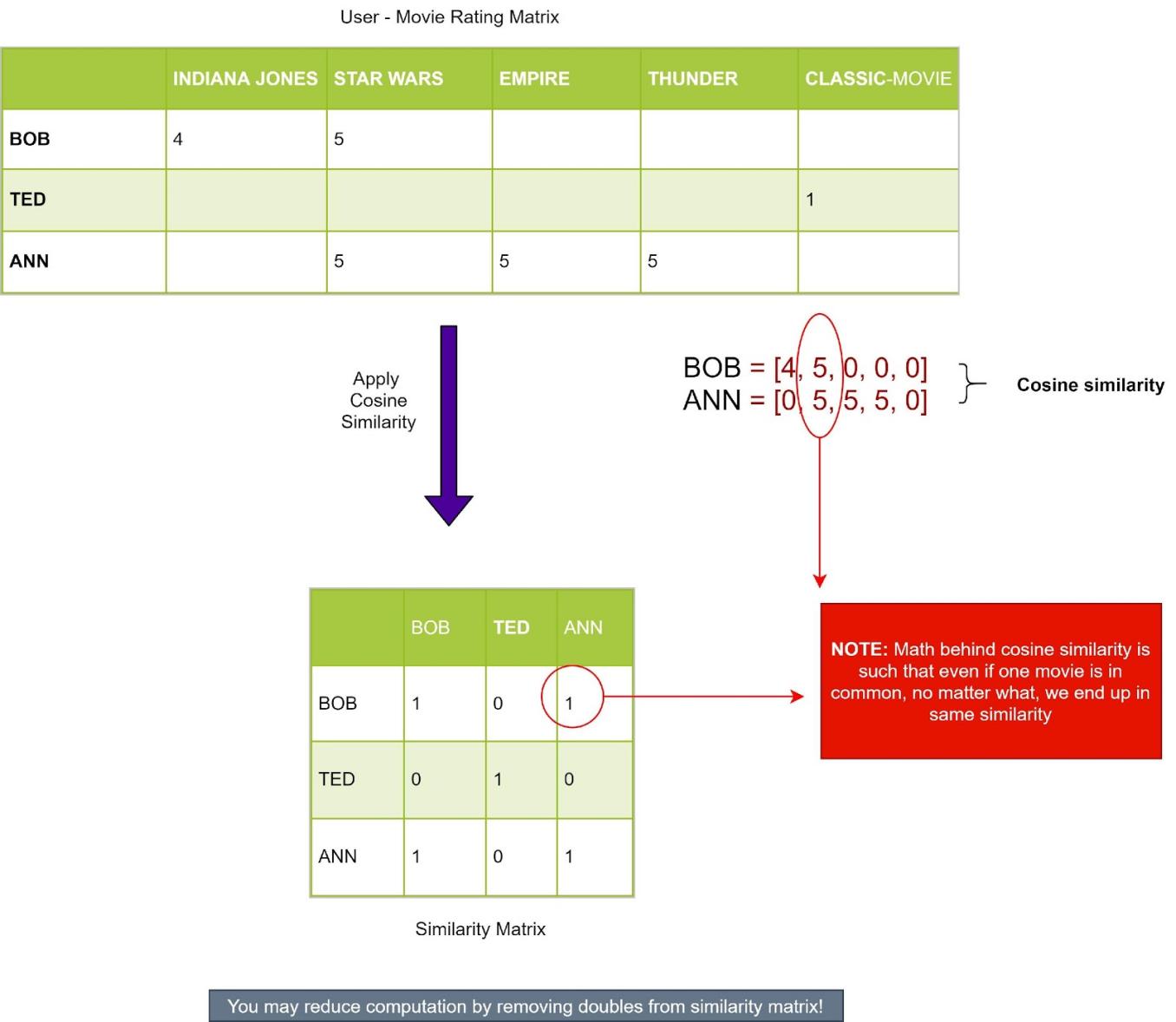
**2)explain cosine similarity equation.**

Cosine similarity is a metric used to determine how similar the documents are irrespective of their size.

Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. In this context, the two vectors I am talking about are arrays containing the word counts of two documents.

Item to Item Similarity: The very first step is to build the model by finding similarity between all the item pairs. The similarity between item pairs can be found in different ways. One of the most common methods is to use cosine similarity.





**3)explain RMSE and MSE with mathematical equation.**

computing the RMSE using the knn\_model that you fitted in the previous code block. You compute the RMSE on the training data for now. For a more realistic result, evaluate the performances on data that aren’t included in the model. This is why kept the test set separate for now.

test\_preds = knn\_model.predict(X\_test)

mse = mean\_squared\_error(y\_test, test\_preds)

rmse = sqrt(mse)

**RMSE:**

import matplotlib.pyplot as plt

import math

x = [1, 2, 2, 3]

y = [1, 2, 3, 6]

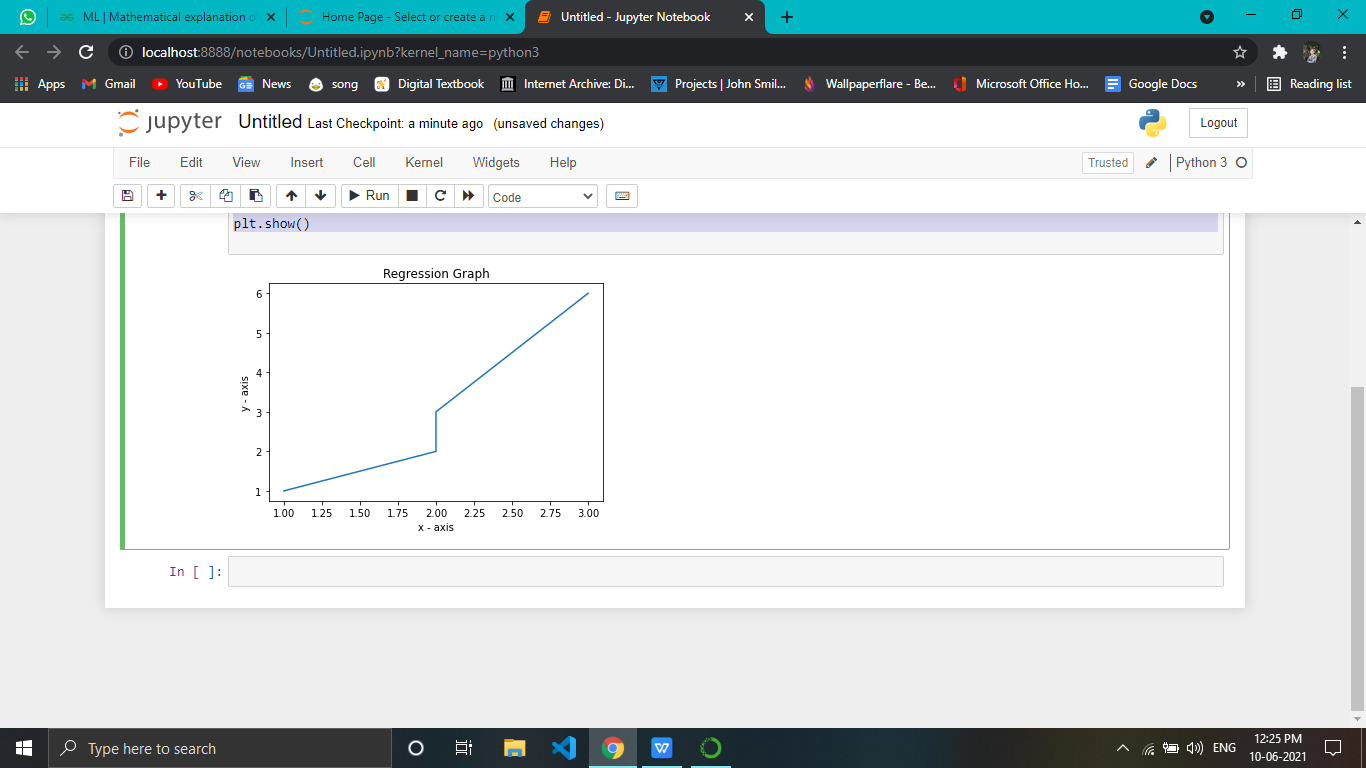
plt.plot(x, y)

plt.xlabel('x - axis')

plt.ylabel('y - axis')

plt.title('Regression Graph')

plt.show()

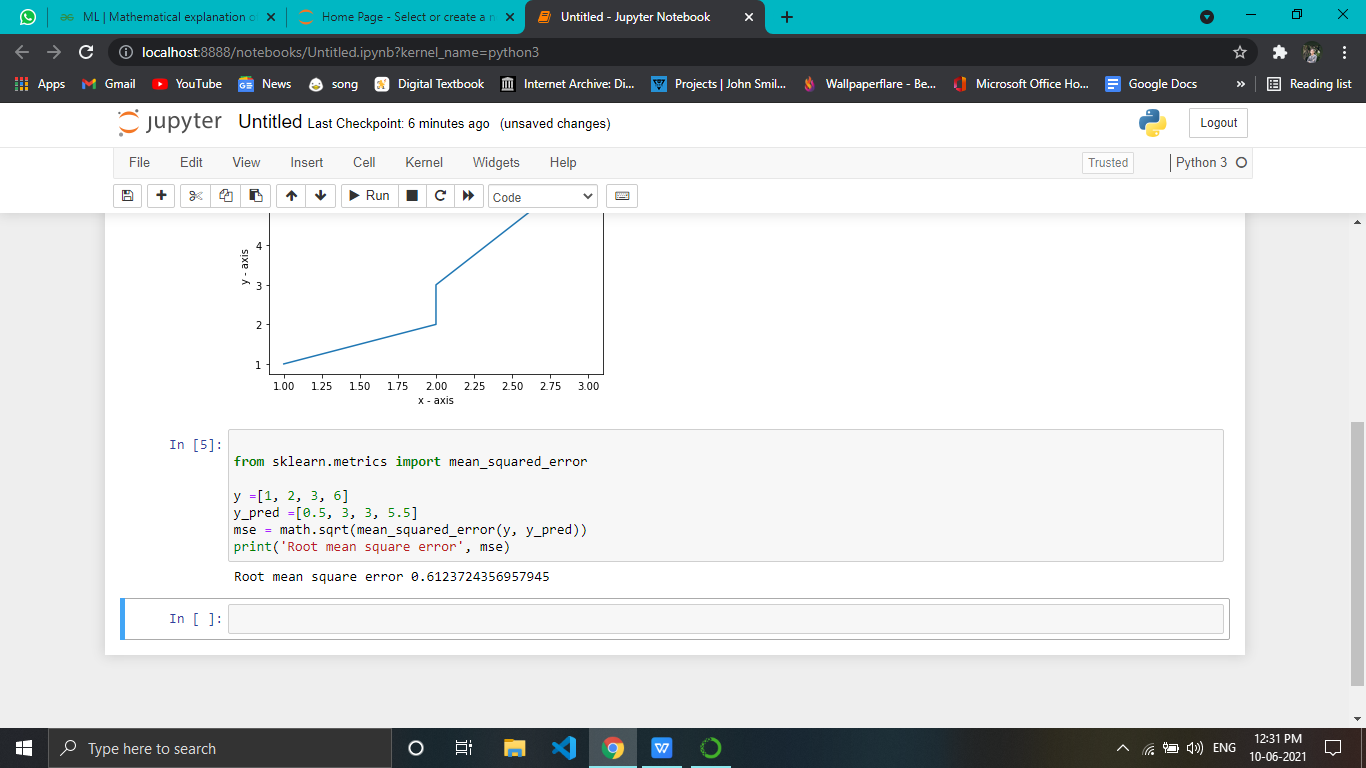


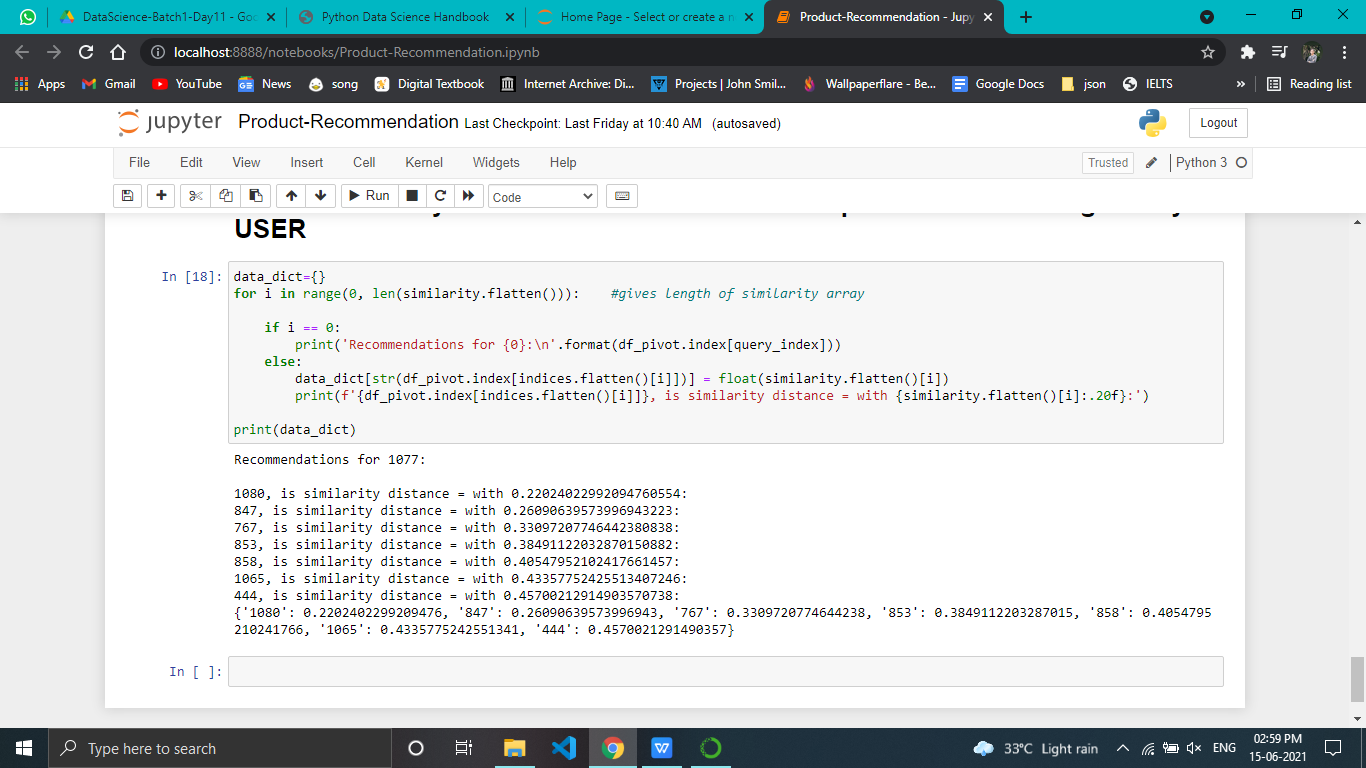
**MSE:**

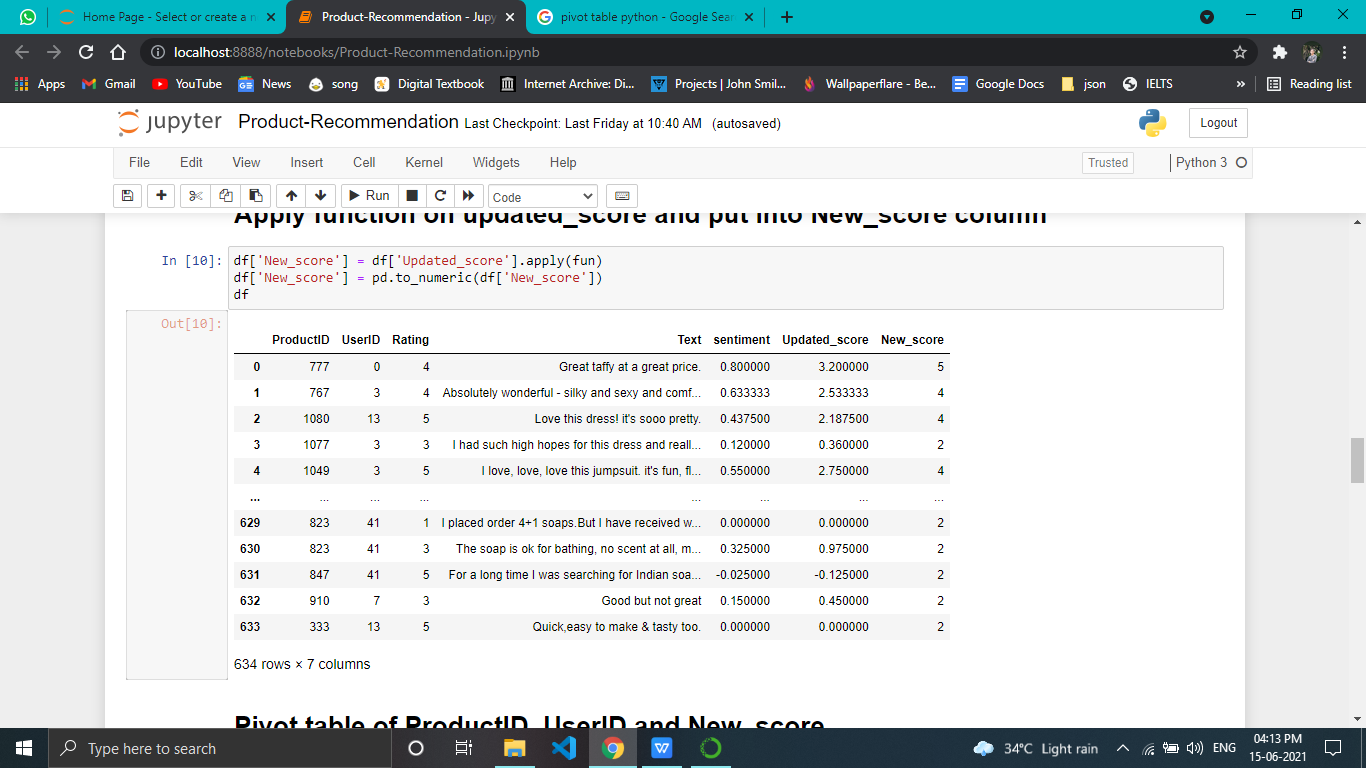
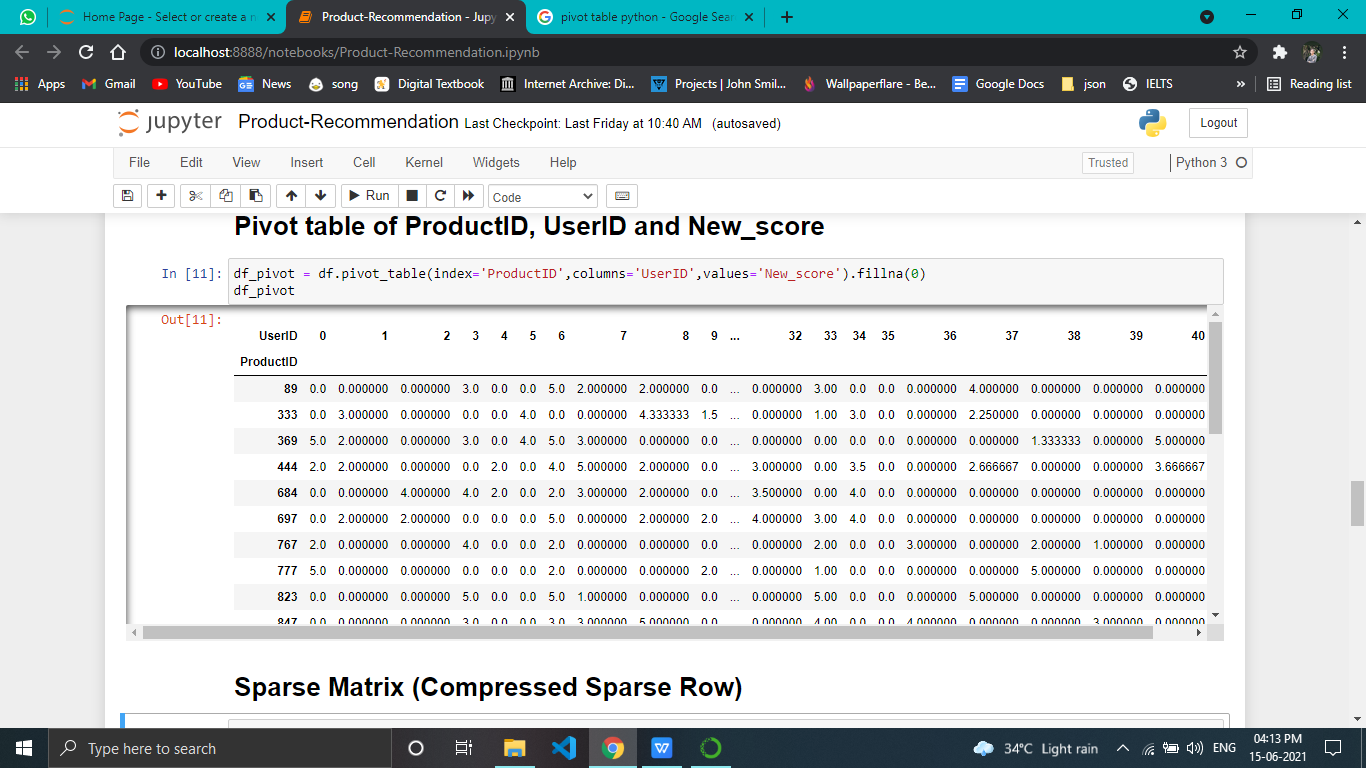
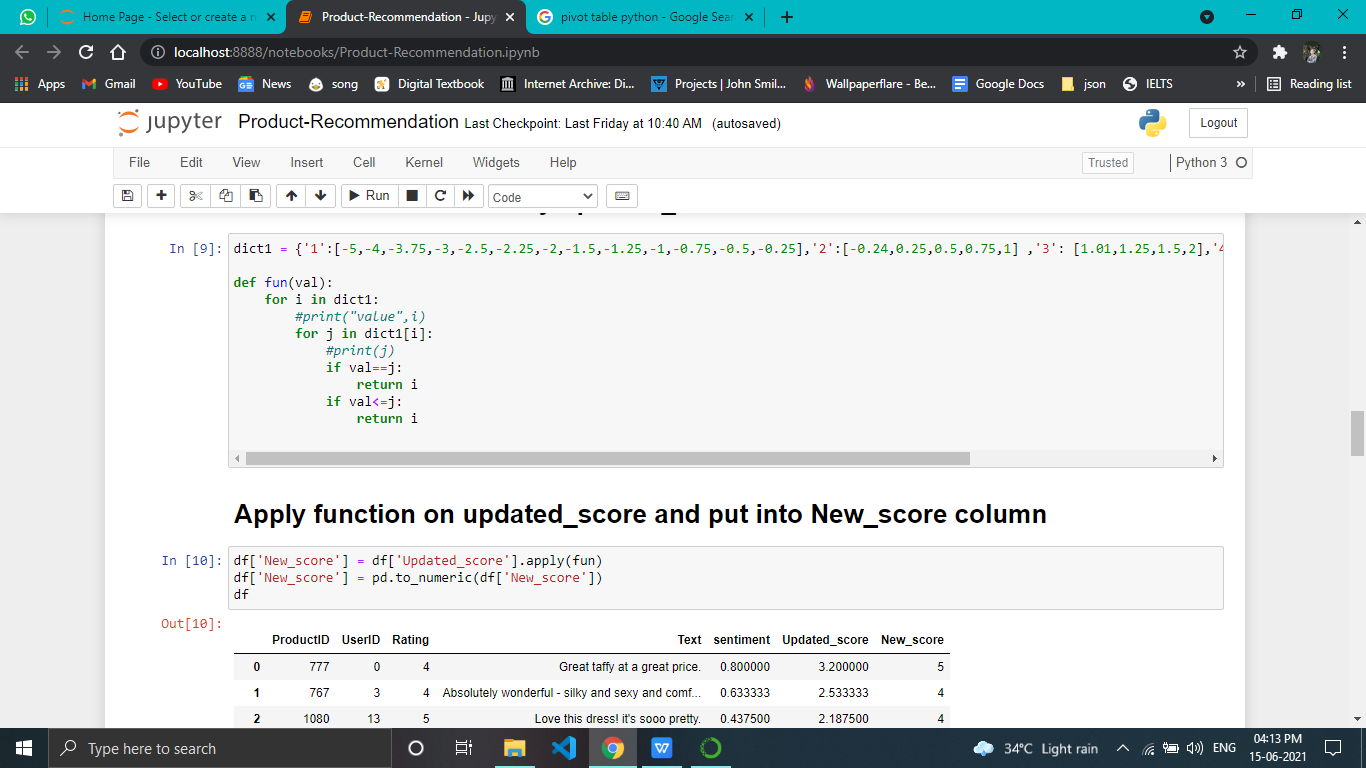
The measure of mean squared error needs a target of prediction or estimation along with a predictor or estimator, which is said to be the function of the given data. MSE is the average of squares of the “errors”.

Here, the error is the difference between the attribute which is to be estimated and the estimator. The mean square error may be called a risk function which agrees to the expected value of the loss of squared error. This difference or the loss could be developed due to the randomness or due to the estimator is not representing the information which could provide a more accurate estimate.

The mean squared error can also be referred to the second moment of the error, measured about the origin. It includes both the variance and bias of the estimator. If an estimator is an unbiased estimator, then its MSE is the same as the variance of the estimator. The unit of MSE is the same as the unit of measurement for the quantity which is being estimated.

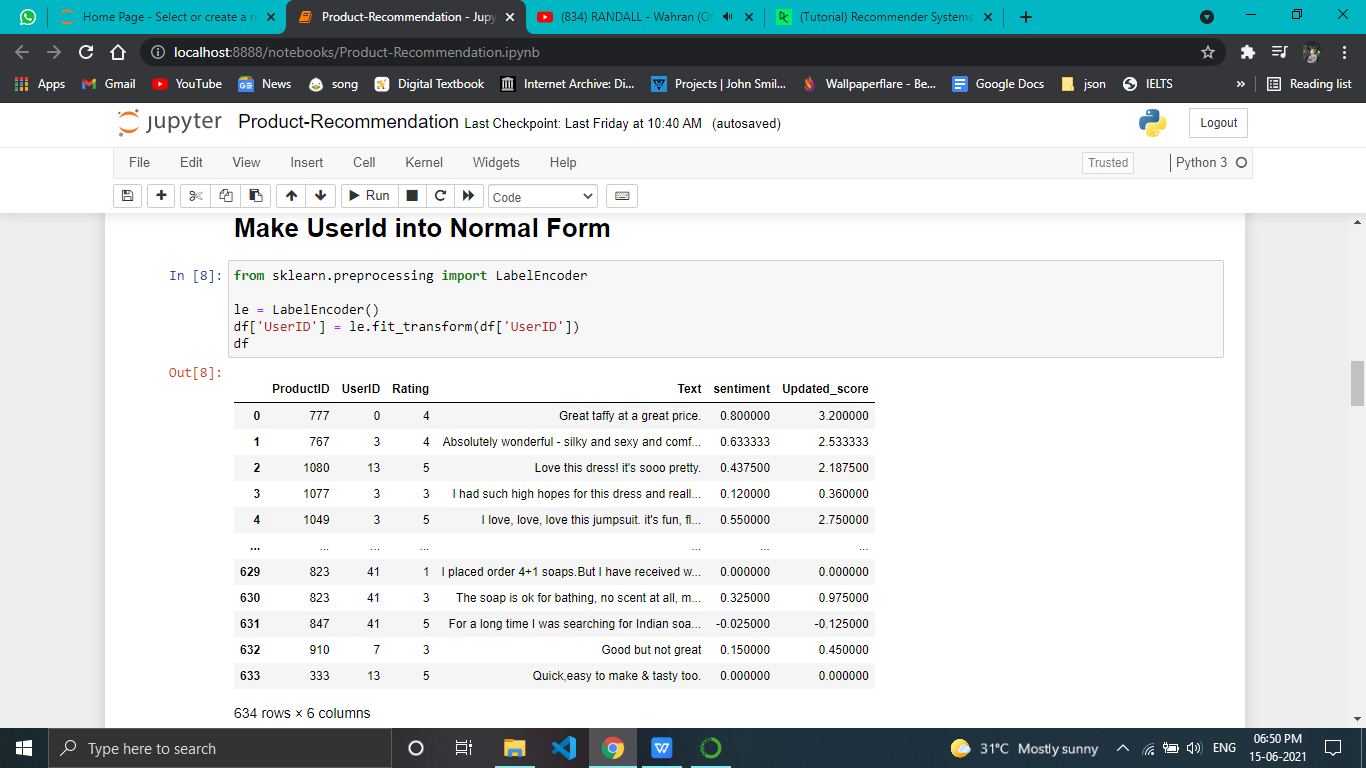


**Day11-task 1)perform recommendation with any dataset**

**Day12-find the key from the dictionary containing 1-5 ratings as keys and 40 values.**

**Day13-compare recommendation with rating**

**And newrating consist of sentiment score.**



**Day14-explain surprise package and its working:**

[Surprise](http://surpriselib.com/) is a Python [scikit](https://www.scipy.org/scikits.html) for building and analyzing recommender systems that deal with explicit rating data.

[Surprise](http://surpriselib.com/) ****was designed with the following purposes in mind****:

* Give users perfect control over their experiments. To this end, a strong emphasis is laid on [documentation](http://surprise.readthedocs.io/en/stable/index.html), which we have tried to make as clear and precise as possible by pointing out every detail of the algorithms.
* Alleviate the pain of [Dataset handling](http://surprise.readthedocs.io/en/stable/getting_started.html" \l "load-a-custom-dataset). Users can use both *built-in* datasets ([Movielens](http://grouplens.org/datasets/movielens/), [Jester](http://eigentaste.berkeley.edu/dataset/)), and their own *custom* datasets.
* Provide various ready-to-use [prediction algorithms](http://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html) such as [baseline algorithms](http://surprise.readthedocs.io/en/stable/basic_algorithms.html), [neighborhood methods](http://surprise.readthedocs.io/en/stable/knn_inspired.html), matrix factorization-based ( [SVD](http://surprise.readthedocs.io/en/stable/matrix_factorization.html" \l "surprise.prediction_algorithms.matrix_factorization.SVD), [PMF](http://surprise.readthedocs.io/en/stable/matrix_factorization.html" \l "unbiased-note), [SVD++](http://surprise.readthedocs.io/en/stable/matrix_factorization.html" \l "surprise.prediction_algorithms.matrix_factorization.SVDpp), [NMF](http://surprise.readthedocs.io/en/stable/matrix_factorization.html" \l "surprise.prediction_algorithms.matrix_factorization.NMF)), and [many others](http://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html). Also, various [similarity measures](http://surprise.readthedocs.io/en/stable/similarities.html) (cosine, MSD, pearson...) are built-in.
* Make it easy to implement [new algorithm ideas](http://surprise.readthedocs.io/en/stable/building_custom_algo.html).
* Provide tools to [evaluate](http://surprise.readthedocs.io/en/stable/model_selection.html), [analyse](http://nbviewer.jupyter.org/github/NicolasHug/Surprise/tree/master/examples/notebooks/KNNBasic_analysis.ipynb/) and [compare](http://nbviewer.jupyter.org/github/NicolasHug/Surprise/blob/master/examples/notebooks/Compare.ipynb) the algorithms' performance. Cross-validation procedures can be run very easily using powerful CV iterators (inspired by [scikit-learn](http://scikit-learn.org/) excellent tools), as well as [exhaustive search over a set of parameters](http://surprise.readthedocs.io/en/stable/getting_started.html" \l "tune-algorithm-parameters-with-gridsearchcv).

The name *SurPRISE* (roughly :) ) stands for *Simple Python RecommendatIon System Engine*.

Please note that surprise does not support implicit ratings or content-based information.