**Charotar University of Science and Technology (CHARUSAT)**

Academic year

(2021-2022)

Devang Patel Institute of Advance Technology and Research

**INTERNSHIP REPORT**

UNDER SUBJECT OF

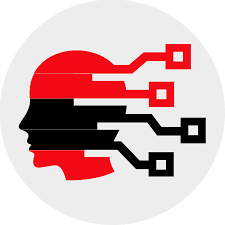
**SUMMER INTERNSHIP (3170001)**

B.E. SEMESTER-VII

**Information Technology**

**Submitted by :-**

**Nilay Nitinkumar Patel**



BrainyBeam Technologies Pvt. Ltd.

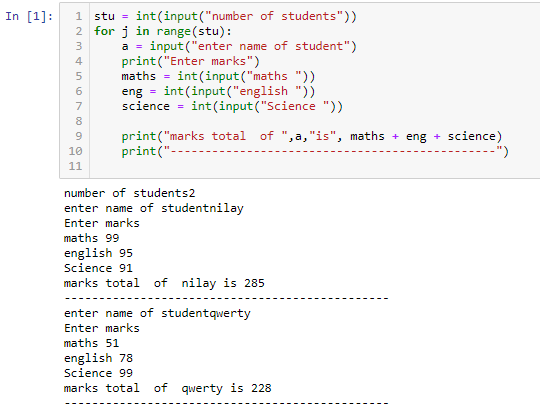
Prof. Chintal Raval, Mr. Raj Makhijani

(Internal Guide) (External Guide)

**Task 1 [26/05/2021]**

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

**Code & Output:**



**Task 2 [27/05/2021]**

**Task: List out the methods used commonly in list,set,tuple,dictionary with thier rules**

**Description:**

**List**

A data structure that stores an ordered collection of items in Python is called a list. In other words, a list holds a sequence of items. You need to put all the items, separated by commas, in square brackets to let Python know that a list has been specified.

* ***append () –****Adds an element to the end of the list*
* **clear () –** Removes all elements from the list
* **copy () –** Returns a shallow copy of the list
* **count () –** Returns the total number of items passed as an argument
* **extend () –** Adds all elements of a list to some other list
* **index () –** Returns the index of an element (Note: If the same element appears multiple times in the list then the index of the very first match is returned)
* **insert () –** Inserts an element to the list at the defined index
* **pop () –** Eliminates and returns an element from the list
* **remove () –** Eliminates an element from the list
* **reverse () –** Reverses the order of all elements of the list
* **sort () –** Sort all elements of a list in the ascending order

**Set**

Similar to a list, the tuple is a built-in data structure in Python. However, it doesn’t support the same level of extensive functionality. The most important difference between a list and a tuple is mutability. Unlike lists, tuples are immutable i.e. they can’t be modified.

Typically, a tuple in the Python programming language is defined using parentheses with each item separated by commas. Though adding parentheses to the tuple is optional, its use is recommended to clearly define the end and start of the tuple. The general syntax of a tuple is:

Similar to string indices, tuple indices start at 0. Like strings, tuples can be concatenated and sliced. Methods available for tuples are:

1. ***cmp ()* –** Compares elements of two tuples
2. ***len ()* –** Gives out the total length of some tuple
3. ***max ()* –** Returns the biggest value from a tuple
4. ***min ()* –** Returns the smallest value from a tuple
5. ***tuple ()* –** Converts some list into a tuple

**Tuple**

**Python has two built-in methods that you can use on tuples.**

**Method Description**

**count() Returns the number of times a specified value occurs in a tuple**

**index() Searches the tuple for a specified value and returns the position of where it was found**

**Dictionary**

Another type of built-in data structure in Python is the dictionary. It stores data in the form of key-value pairs. The keys defined for a dictionary need to be unique. Though values in a dictionary can be mutable or immutable objects, only immutable objects are allowed for keys.

D.keys() : List of keys

• D.values() : List of values

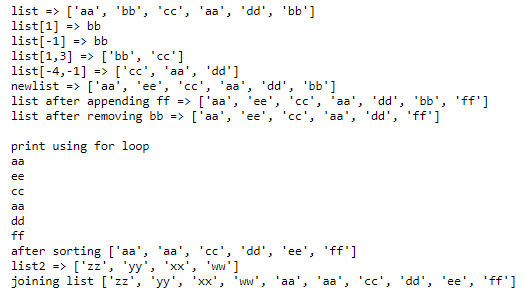
• D.clear() : remove all items

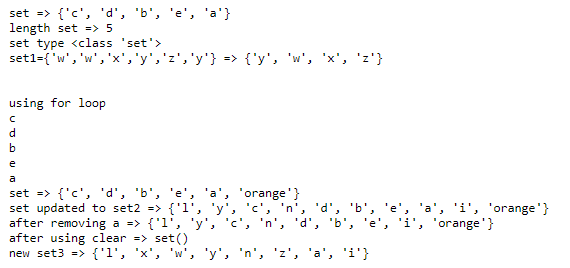
• D.update(D2) : Merge key values from D2 into D. NOTE: Overwrites any matching keys in D.

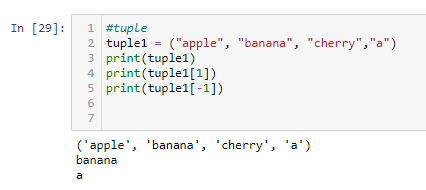
• D.pop(key) : returns the value of given key and removes this key:value pair from dictionary.

**Code & Output:**









**Task 3 [28/05/2021]**

**Task :**

1. **random module function with explanation**
2. **build password generator program containing number, alphabet and special characters**
3. **write note about NLP, NLU, NLG with example**
4. **perform text to speech example using gtts**

**1.**

|  |  |
| --- | --- |
| **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 |
| [setstate()](https://www.w3schools.com/python/ref_random_setstate.asp) | Restores the internal state of the random number generator |
| [getrandbits()](https://www.w3schools.com/python/ref_random_getrandbits.asp) | Returns a number representing the random bits |
| [randrange()](https://www.w3schools.com/python/ref_random_randrange.asp) | Returns a random number between the given range |
| [randint()](https://www.w3schools.com/python/ref_random_randint.asp) | Returns a random number between the given range |
| [choice()](https://www.w3schools.com/python/ref_random_choice.asp) | Returns a random element from the given sequence |
| [choices()](https://www.w3schools.com/python/ref_random_choices.asp) | Returns a list with a random selection from the given sequence |
| [shuffle()](https://www.w3schools.com/python/ref_random_shuffle.asp) | Takes a sequence and returns the sequence in a random order |
| [sample()](https://www.w3schools.com/python/ref_random_sample.asp) | Returns a given sample of a sequence |
| [random()](https://www.w3schools.com/python/ref_random_random.asp) | Returns a random float number between 0 and 1 |
| [uniform()](https://www.w3schools.com/python/ref_random_uniform.asp) | Returns a random float number between two given parameters |
| [triangular()](https://www.w3schools.com/python/ref_random_triangular.asp) | Returns a random float number between two given parameters, you can also set a mode parameter to specify the midpoint between the two other parameters |
| betavariate() | Returns a random float number between 0 and 1 based on the Beta distribution (used in statistics) |
| expovariate() | Returns a random float number based on the Exponential distribution (used in statistics) |
| gammavariate() | Returns a random float number based on the Gamma distribution (used in statistics) |

**2.**

**Code & Output:**



3.

* **NLU:**

Neuro-linguistic programming (NLP) is a psychological approach that involves analyzing strategies used by successful individuals and applying them to reach a personal goal. It relates thoughts, language, and patterns of [behavior](https://www.goodtherapy.org/blog/psychpedia/behavior) learned through experience to specific outcomes.

Proponents of NLP assume all human action is positive. Therefore, if a plan fails or the unexpected happens, the experience is neither good nor bad—it simply presents more useful information.

## HOW NEURO-LINGUISTIC PROGRAMMING WORKS

Modeling, action, and effective communication are key elements of neuro-linguistic programming. The belief is that if an individual can understand how another person accomplishes a task, the process may be copied and communicated to others so they too can accomplish the task.

Proponents of neuro-linguistic programming propose that everyone has a personal map of reality. Those who practice NLP analyze their own and other perspectives to create a systematic overview of one situation. By understanding a range of perspectives, the NLP user gains information. Advocates of this school of thought believe the senses are vital for processing available information and that the body and mind influence each other. Neuro-linguistic programming is an experiential approach. Therefore, if a person wants to understand an action, they must perform that same action to learn from the experience.

NLP practitioners believe there are natural hierarchies of learning, communication, and change. The six logical levels of change are:

* **Purpose and**[**spirituality**](https://www.goodtherapy.org/learn-about-therapy/issues/spirituality)**:**  This can be involvement in something larger than oneself, such as religion, ethics, or another system. This is the highest level of change.
* **Identity:** Identity is the person you perceive yourself to be and includes your responsibilities and the [roles](https://www.goodtherapy.org/blog/psychpedia/role) you play in life.
* **Beliefs and values:** These are your personal belief system and the issues that matter to you.
* **Capabilities and skills:** These are your abilities and what you can do.
* **Behaviors:** Behaviors are the specific actions you perform.
* **Environment:** Your environment is your context or setting, including any other people around you. This is the lowest level of change.

The purpose of each logical level is to organize and direct the information below it. As a result, making a change in a lower level may cause changes in a higher level. However, making a change in a higher level will also result in changes in the lower levels, according to NLP theory.

* **NLU**

NLU is branch of[natural language processing (NLP)](https://www.twilio.com/docs/glossary/what-natural-language-processing-nlp), which helps computers understand and interpret human language by breaking down the elemental pieces of speech. While[speech recognition](https://www.twilio.com/speech-recognition) captures spoken language in real-time, transcribes it, and returns text, NLU goes beyond recognition to determine a user’s intent. Speech recognition is powered by statistical machine learning methods which add numeric structure to large datasets. In NLU, machine learning models improve over time as they learn to recognize syntax, context, language patterns, unique definitions, sentiment, and intent.

Regardless of the approach used, most natural-language-understanding systems share some common components. The system needs a [lexicon](https://en.wikipedia.org/wiki/Lexicon) of the language and a [parser](https://en.wikipedia.org/wiki/Parser) and [grammar](https://en.wikipedia.org/wiki/Grammar) rules to break sentences into an internal representation. The construction of a rich lexicon with a suitable [ontology](https://en.wikipedia.org/wiki/Ontology_(information_science)) requires significant effort, *e.g.*, the [Wordnet](https://en.wikipedia.org/wiki/Wordnet) lexicon required many person-years of effort.

The system also needs theory from [*semantics*](https://en.wikipedia.org/wiki/Semantics) to guide the comprehension. The interpretation capabilities of a language-understanding system depend on the semantic theory it uses. Competing semantic theories of language have specific trade-offs in their suitability as the basis of computer-automated semantic interpretation. These range from [*naive semantics*](https://en.wikipedia.org/wiki/Naive_semantics) or [*stochastic semantic analysis*](https://en.wikipedia.org/wiki/Stochastic_semantic_analysis) to the use of [*pragmatics*](https://en.wikipedia.org/wiki/Pragmatics) to derive meaning from context. [Semantic parsers](https://en.wikipedia.org/wiki/Semantic_parser) convert natural-language texts into formal meaning representations.

Advanced applications of natural-language understanding also attempt to incorporate logical [inference](https://en.wikipedia.org/wiki/Inference) within their framework. This is generally achieved by mapping the derived meaning into a set of assertions in [predicate logic](https://en.wikipedia.org/wiki/Predicate_logic), then using [logical deduction](https://en.wikipedia.org/wiki/Logical_deduction) to arrive at conclusions. Therefore, systems based on functional languages such as [Lisp](https://en.wikipedia.org/wiki/Lisp_(programming_language)) need to include a subsystem to represent logical assertions, while logic-oriented systems such as those using the language [Prolog](https://en.wikipedia.org/wiki/Prolog) generally rely on an extension of the built-in logical representation framework.

The management of [context](https://en.wikipedia.org/wiki/Context_(language_use)) in natural-language understanding can present special challenges. A large variety of examples and counter examples have resulted in multiple approaches to the formal modeling of context, each with specific strengths and weaknesses

* **NLG**

NLG is a type of AI that generates natural language from structured data.

With the right data in the right format, an NLG system can automatically turn numbers in a spreadsheet into data-driven narratives or even use associations between words to create partially or fully machine-written text. NLG systems use machine learning, deep learning, and neural networks (all forms of AI) to generate natural language.

These systems can generate natural language in a variety of formats. They can turn numbers into narratives based on pre-set templates. They can predict which words need to be generated next (in, say, an email you're actively typing). Or, the most sophisticated systems can formulate entire summaries, articles, or responses.

NLG is used across a range of contexts. Totask, NLG systems can write email subject lines, snippets of blog posts, summaries of in-depth reports, short-form advertising copy, chatbot responses, and much, much more.

NLG has been a field of AI research for decades, but the most impressive, and useful, advances in the technology have only happened recently.

As far back as 1986, research was published on [**possible NLG use cases**](https://apps.dtic.mil/sti/pdfs/ADA175135.pdf). Ten years later, researchers at the University of Aberdeen were publishing about how to use the technology for [**text and sentence planning**](https://arxiv.org/pdf/cmp-lg/9605002.pdf). As late as 2006, [**obstacles to NLG adoption**](https://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199276349.001.0001/oxfordhb-9780199276349-e-15) were still being defined and discussed among leaders in the field.

Yet totask you use NLG every task whether you know it or not.

A system like [**Gmail's Smart Compose**](https://www.blog.google/products/gmail/subject-write-emails-faster-smart-compose-gmail/) now offers suggestions on what you should type next in an email. Then, it learns each time you pick one of its suggestions. It's using advanced NLG and NLP to read your emails while you write them, then predict what you are trying to say next—in real-time.

The Associated Press [**uses NLG**](https://www.marketingaiinstitute.com/blog/how-the-associated-press-and-the-orlando-magic-write-thousands-of-content-pieces-in-seconds?_ga=2.45476836.1302922534.1578923674-849186137.1573136992) to automatically create machine-written corporate earnings reports. In the past, a human reporter reviewed earnings data when it was released, then wrote a report on it. Totask, the AP [**uses NLG**](https://www.marketingaiinstitute.com/blog/how-the-associated-press-and-the-orlando-magic-write-thousands-of-content-pieces-in-seconds?_ga=2.45476836.1302922534.1578923674-849186137.1573136992) to ingest that data, then produce a narrative in seconds, freeing up reporters to pursue higher-value tasks within the organization. All the AP earnings reports follow the same exact format. So, once NLG is set up, their system can produce thousands of stories at scale.

Even weirder, in 2019 OpenAI, a non-profit AI research company, announced they built an AI model that essentially [**writes coherent paragraphs of text at scale**](https://www.marketingaiinstitute.com/blog/what-happens-to-marketing-when-ai-can-write-like-humans?_ga=2.51322473.1574380102.1601390374-367201747.1601390374) . The model is called GPT-2 and it learned how to write this well by analyzing eight million web pages. In 2020, OpenAI updated the model and released GPT-3.

In early experiments, GPT-3 has been used to produce everything from coherent blog posts to press releases to technical manuals, often with a high degree of accuracy. To do that, GPT-3 uses 175 billion parameters in its language model, compared to GPT-2's 1.5 billion.

It's still early tasks for GPT-3, and the validity of the model hasn't been fully explored. But one thing should give marketers pause:

The speed of improvement in OpenAI's language models.

The first GPT model came out in 2018. GPT-2 was released with greatly expanded capabilities in 2019. Just a year later, GPT-3 uses 100x as much data as its predecessor and is beginning to display incredible content creation capabilities, including turning text into code and evaluating investment memos.

This type of machine-powered language generation has serious implications for marketers.

With properly structured data, marketers can use NLG to automate some narratives that are based on data. That might include analytics reports, data-driven sections of blog posts, product descriptions with standardized formats and lots of specifications, or even partial or entire articles.

With the right data and a properly configured NLG template, you can leave some types of content creation to the machines while humans create other types of content and double down on promotion.

4.

from gtts import gTTS

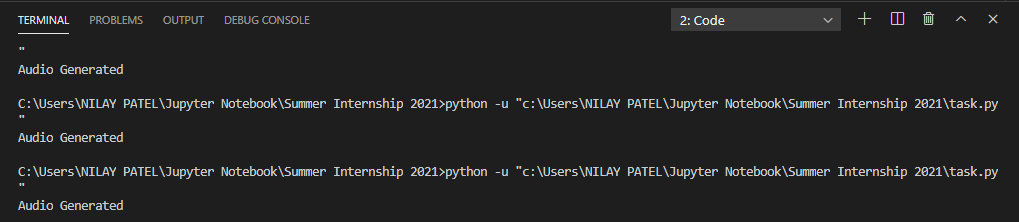
mytext1 = 'I am Christan Grey'

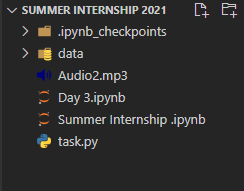
language = 'en'

obj = gTTS(text=mytext1, lang=language, slow=False)

obj.save("Audio2.mp3")

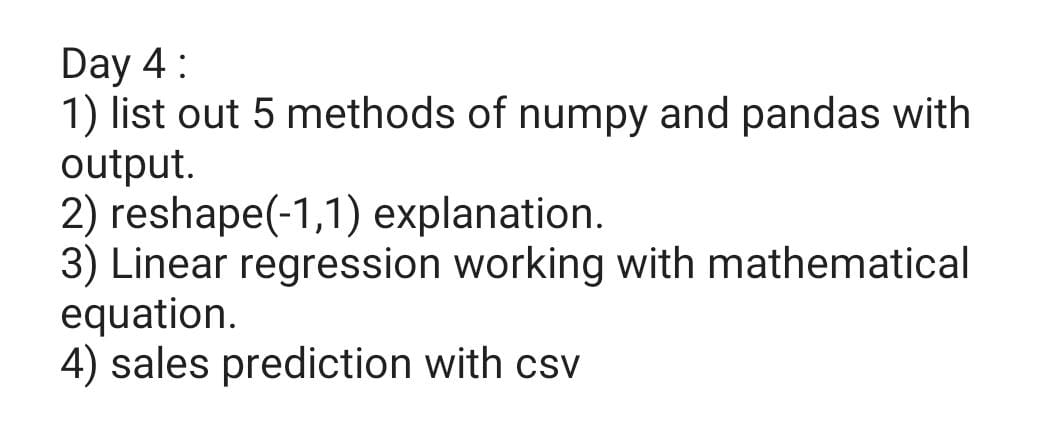
print("Audio Generated")





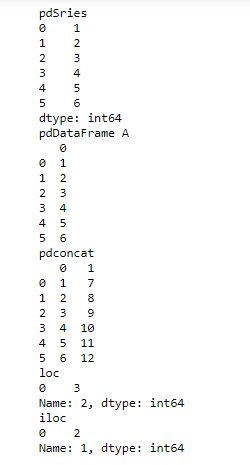
**Task 4 [31/05/2021]**

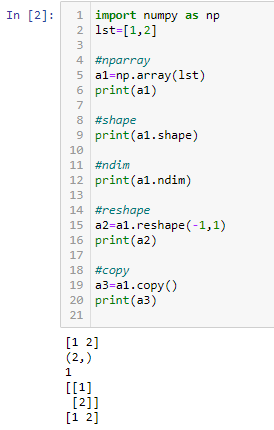
**Task :**



1)



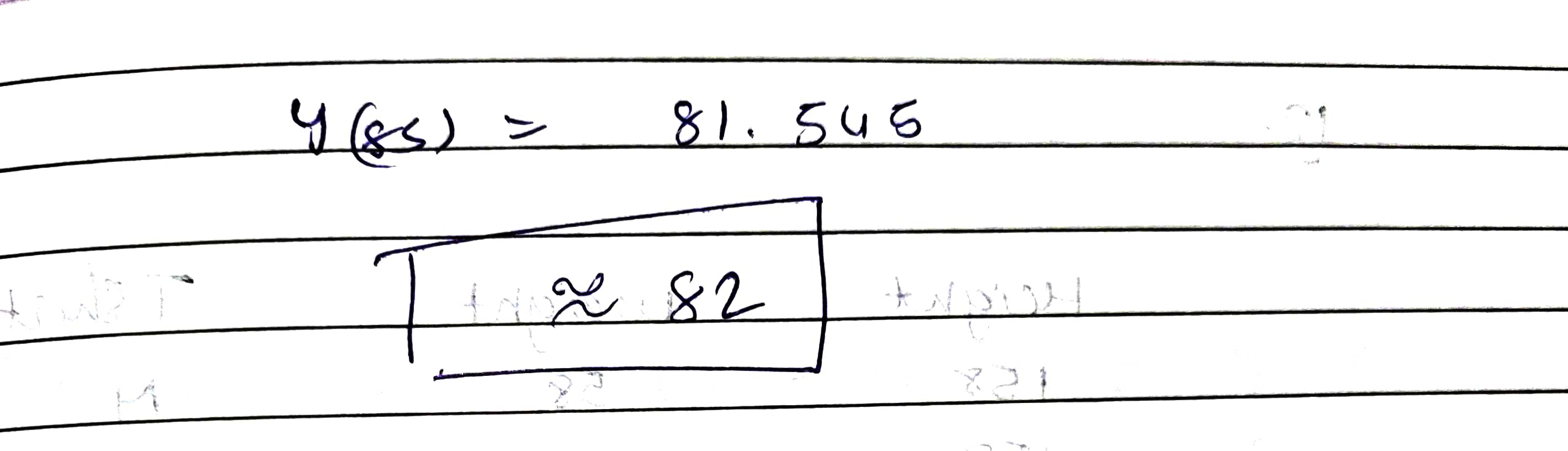
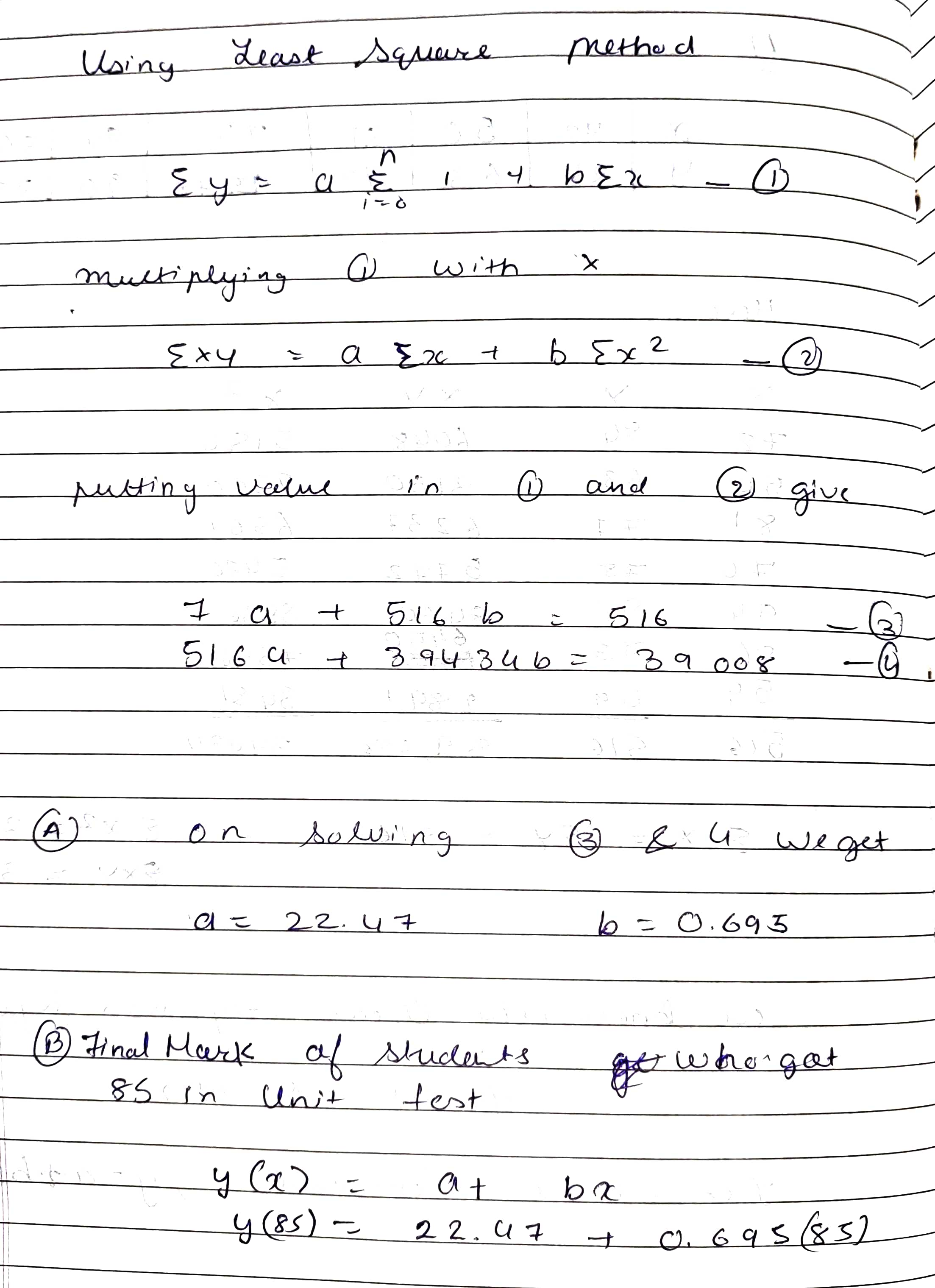
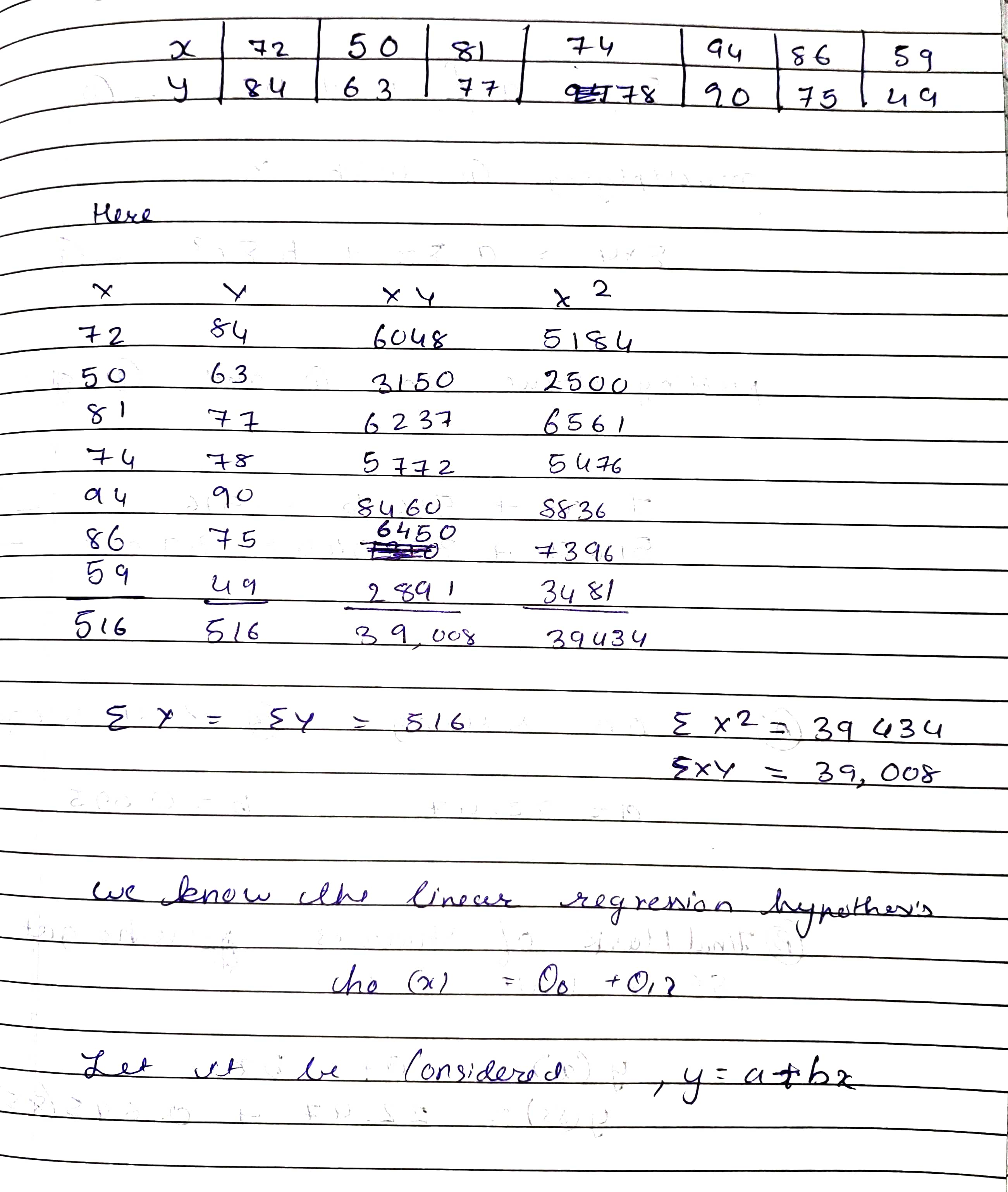




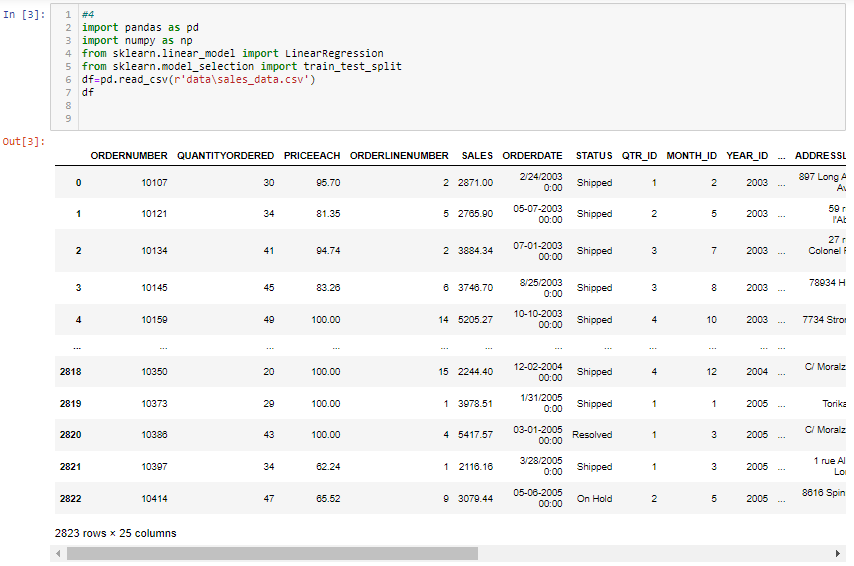
2)

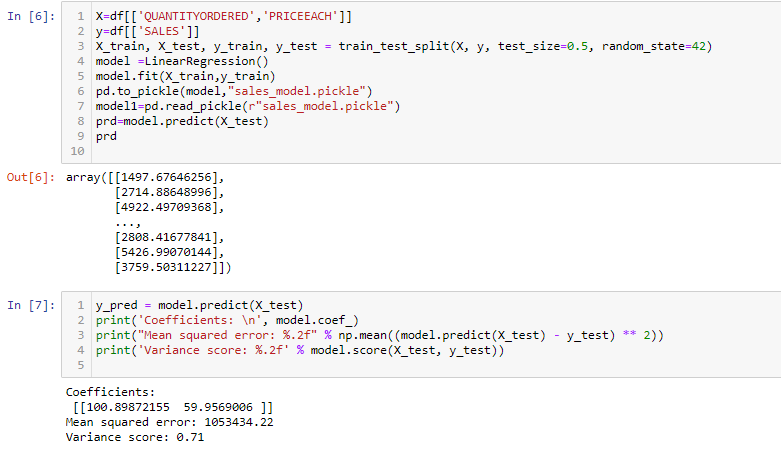
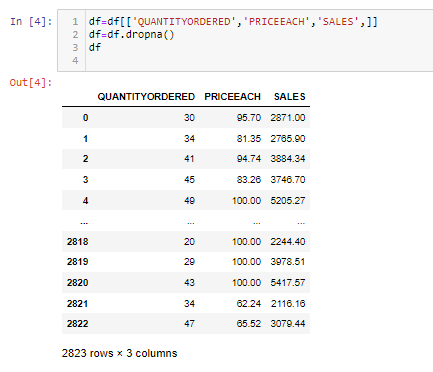
-1 in reshape function is used when you dont know or want to explicitly tell the dimension of that axis. E.g,  
If you have an array of shape (2,4) then reshaping it with (-1, 1), then the array will get reshaped in such a way that the resulting array has only 1 column and this is only possible by having 8 rows, hence, (8,1).

3)



4)





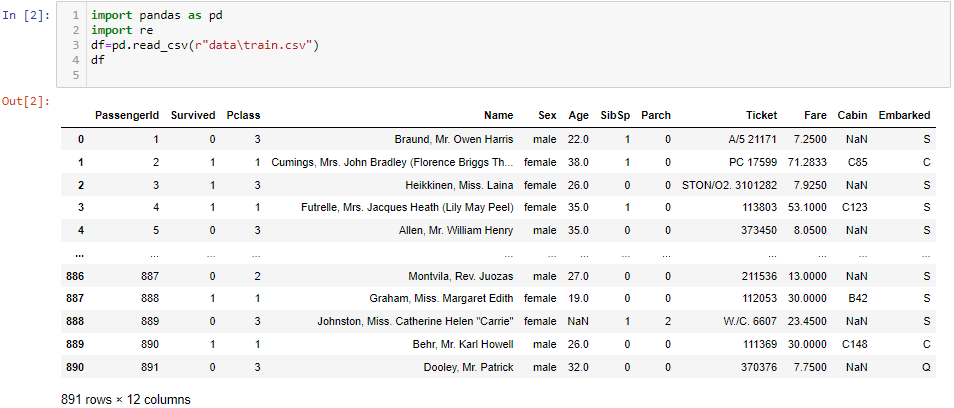
**Task 5 [01/06/2021]**

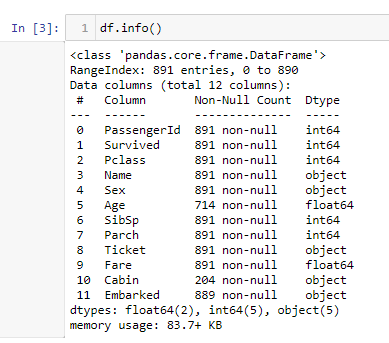
**Task:**

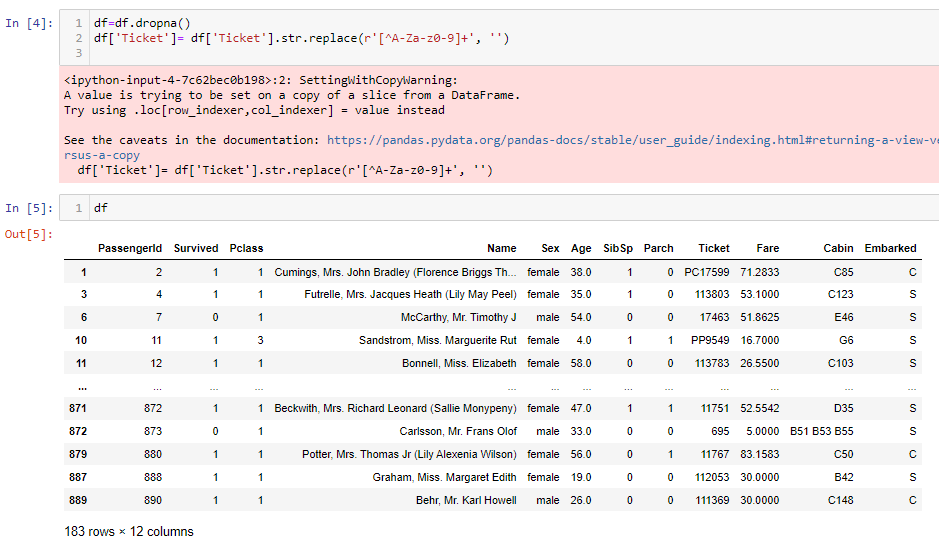
1) Try to clean this data 2) decision tree explanation 3) random state working

**Code & Output:**

**1)**







2)

Decision Trees (DTs) are a non-parametric supervised learning method used for [classification](https://scikit-learn.org/stable/modules/tree.html#tree-classification) and [regression](https://scikit-learn.org/stable/modules/tree.html#tree-regression). The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

* For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.
* As with other classifiers, [DecisionTreeClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html" \l "sklearn.tree.DecisionTreeClassifier" \o "sklearn.tree.DecisionTreeClassifier) takes as input two arrays: an array X, sparse or dense, of shape (n\_samples, n\_features) holding the training samples, and an array Y of integer values, shape (n\_samples,), holding the class labels for the training samples:

>>>

**>>> from** **sklearn** **import** tree

**>>>** X = [[0, 0], [1, 1]]

**>>>** Y = [0, 1]

**>>>** clf = tree.DecisionTreeClassifier()

**>>>** clf = clf.fit(X, Y)

After being fitted, the model can then be used to predict the class of samples:

>>>

**>>>** clf.predict([[2., 2.]])

array([1])

**Some advantages of decision trees are:**

* Simple to understand and to interpret. Trees can be visualised.
* Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
* The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
* Able to handle both numerical and categorical data. However scikit-learn implementation does not support categorical variables for now. Other techniques are usually specialised in analysing datasets that have only one type of variable. See [algorithms](https://scikit-learn.org/stable/modules/tree.html#tree-algorithms) for more information.
* Able to handle multi-output problems.
* Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
* Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
* Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

**The disadvantages of decision trees include:**

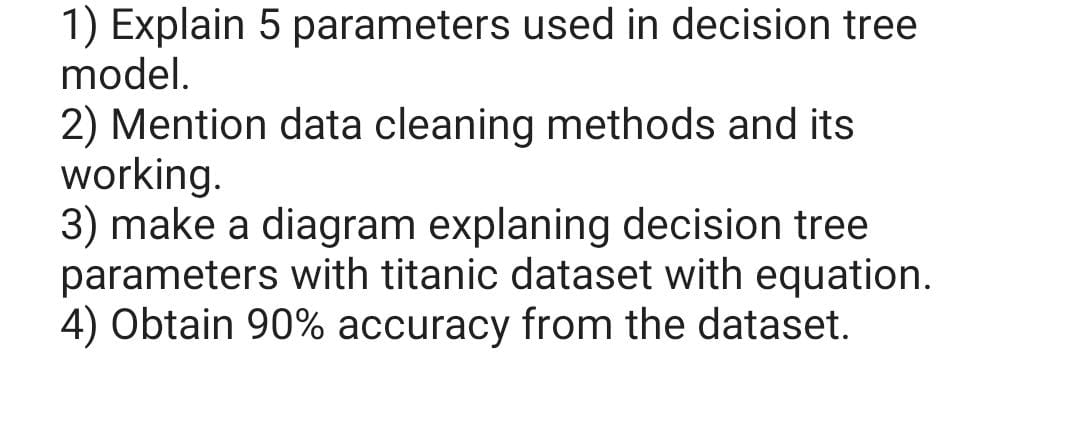
* Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
* Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
* Predictions of decision trees are neither smooth nor continuous, but piecewise constant approximations as seen in the above figure. Therefore, they are not good at extrapolation.
* The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
* There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
* Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

**3)** Random state ensures that the splits that you generate are reproducible. Scikit-learn uses random permutations to generate the splits. The random state that you provide is used as a seed to the random number generator. This ensures that the random numbers are generated in the same order.

* Random\_state can be 0 or 1 or any other integer. It should be the same value if you want to validate your processing over multiple runs of the code. By the way, I have seen random\_state=42 used in many official examples of scikit.
* the random\_state parameter is used for initializing the internal random number generator, which will decide the splitting of data into train and test indices in your case.
* If random\_state is None or np.random, then a randomly-initialized RandomState object is returned.
* If random\_state is an integer, then it is used to seed a new RandomState object.
* This is to check and validate the data when running the code multiple times. Setting random\_state a fixed value will guarantee that the same sequence of random numbers is generated each time you run the code.

**Task 6 [02/06/2021]**

**Task:**



**1) class sklearn.tree.DecisionTreeClassifier(\*, criterion='gini', splitter='best', max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features=None, random\_state=None, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, class\_weight=None, ccp\_alpha=0.0)**

**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.

**min\_samples\_leaf*int or float, default=1***

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min\_samples\_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

* If int, then consider min\_samples\_leaf as the minimum number.
* If float, then min\_samples\_leaf is a fraction and ceil(min\_samples\_leaf \* n\_samples) are the minimum number of samples for each node.

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

**2)**

## **Why Data Cleaning is Necessary**

* Data cleaning might seem dull and uninteresting, but it’s one of the most important tasks you would have to do as a data science professional. Having wrong or bad quality data can be detrimental to your processes and analysis. Poor data can cause a stellar algorithm to fail.
* On the other hand, high-quality data can cause a simple algorithm to give you outstanding results. There are many data cleaning techniques, and you should get familiar with them to improve your data quality. Not all data is useful. So that’s another major factor which affects your data quality.
* **Removal of Unwanted Observations**

Since one of the main goals of data cleansing is to make sure that the dataset is free of unwanted observations, this is classified as the first step to data cleaning. Unwanted observations in a dataset are of 2 types, namely; the duplicates and irrelevances.

* **Duplicate Observations**

A data is said to be a duplicate if it is repeated in a dataset, with it having more than one occurrence. This usually arises when the dataset is created as a result of combining data from two or more sources.

This can also occur in some other cases, including when a respondent makes more than one submission to a survey or error during data entry.

* **Irrelevant Observations**

Irrelevant observations are those that don’t actually fit the specific problem that you’re trying to solve. Like having the price when you are only dealing with quantity.

For example, if you were building a model for prices of apartments in an estate, you don’t need data showing the number of occupants of each house. Irrelevant observations mostly occur when data is generated by scraping from another data source.

* **Fix Data Structure**

After removing unwanted observations, the next thing to do is to make sure that the wanted observations are well-structured. Structural errors may occur during data transfer due to a slight human mistake or incompetency of the data entry personnel.

Some of the things one should look out for when fixing data structure include; typographical errors, grammatical blunders, and so on. The data structure is mostly concerned with categorical data.

From the graph above we observe that

* “school” should be capitalized to give us “School”
* **Filter-out Outliers**

In order to improve the performance of your model, you should remove outliers. Outliers are data points that differ significantly from other observations in a data set.

Outliers are very tricky, in the sense that they are of the same type with other observations, making them look wanted but hugely different from the others. For example, a particular data point may be numerical like other observations in the data set but may turn out to be a big 1000 with the rest between the range 1-10.

Although problematic to some models, there should be a valid reason for removing an outlier. Outliers may arise from a measurement error that is unlikely to be real data, while it may also be as a result of scraping a bigger dataset.

Outliers may give more insight into your model the way the other observations can't. Hence, you should be careful when removing outliers from your data.

* **Handle Missing Data**

You may end up with missing values in your data due to errors during [data collection](https://www.formpl.us/blog/data-collection-method) or [non-response bias](https://www.formpl.us/blog/response-non-response-bias) from respondents. You can avoid this by adding data validation to your survey.

However, now that you already have missing data, how do you handle them?

There are 2 common ways of handling missing data, which are; entirely removing the observations from the data set and imputing a new value based on other observations.

* **Drop Missing Values**

By dropping missing values, you drop information that may assist you in making better conclusions on the subject of study. It may deny you the opportunity of benefiting from the possible insights that can be gotten from the fact that a particular value is missing.

For example, when collecting the scores of students in various exams, student A's score mag be missing in mathematics because he didn't sit for the exam. Assume that this happened because he was sick, and sick students are allowed to rest the exam at a later date.

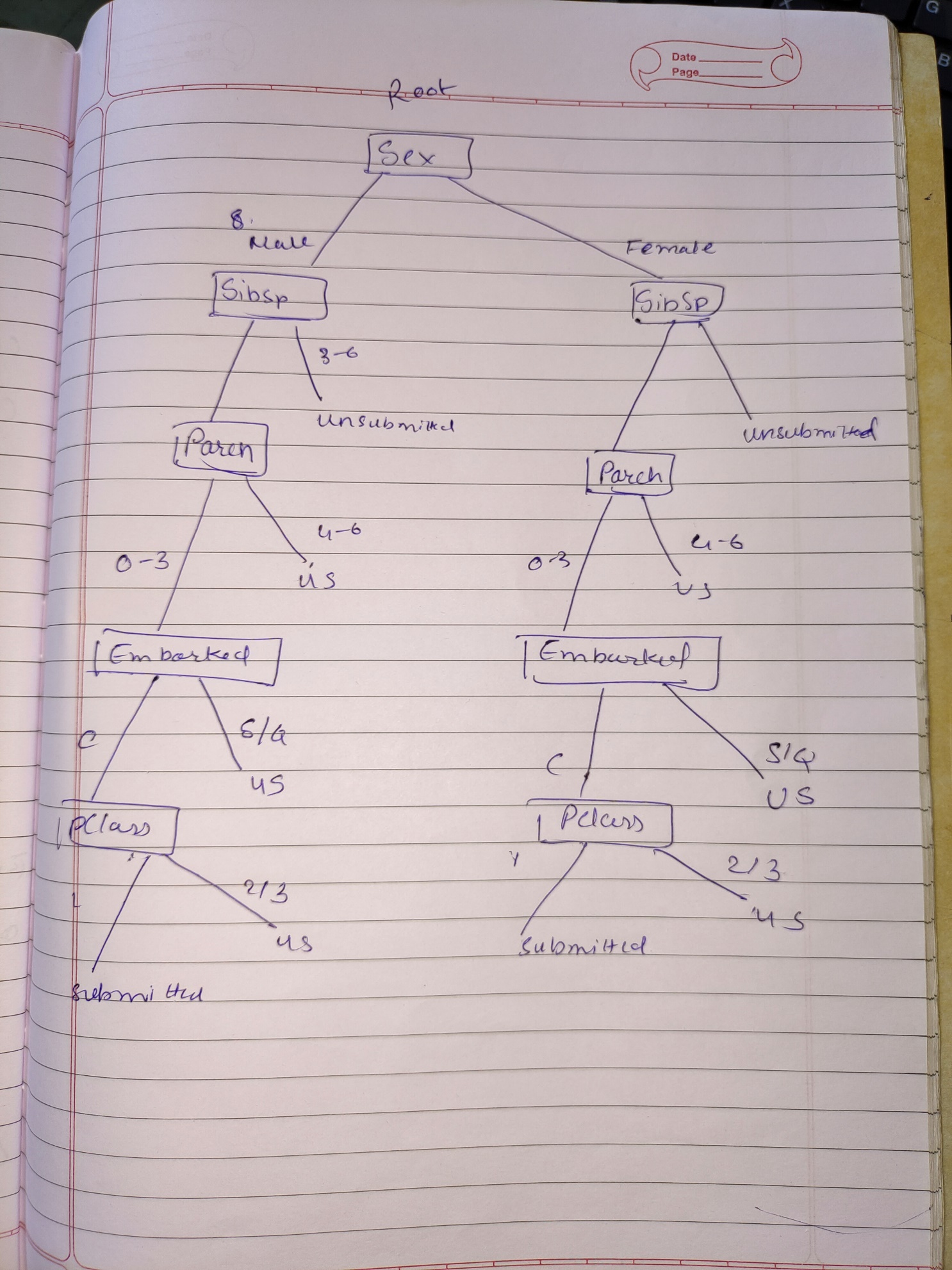
If the whole observation was deleted, we can not detect that he was sick.

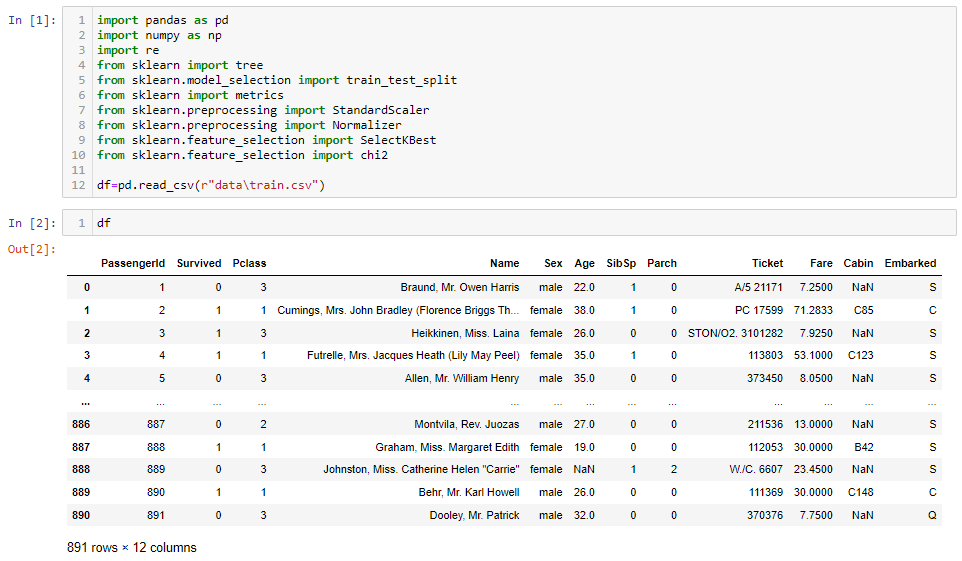
* **Inpute Missing Values**

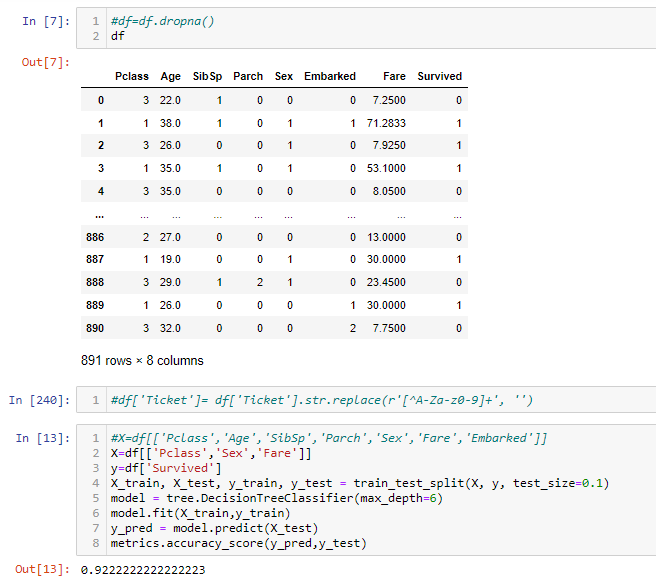
Consider another student B who wrote the exam but his score was missing because the teacher forgot to enter her score. If the teacher imputes a random score for her, it may end up rendering the data incorrect.

Student B may have scored higher or lower than the score the teacher randomly assigns to her.

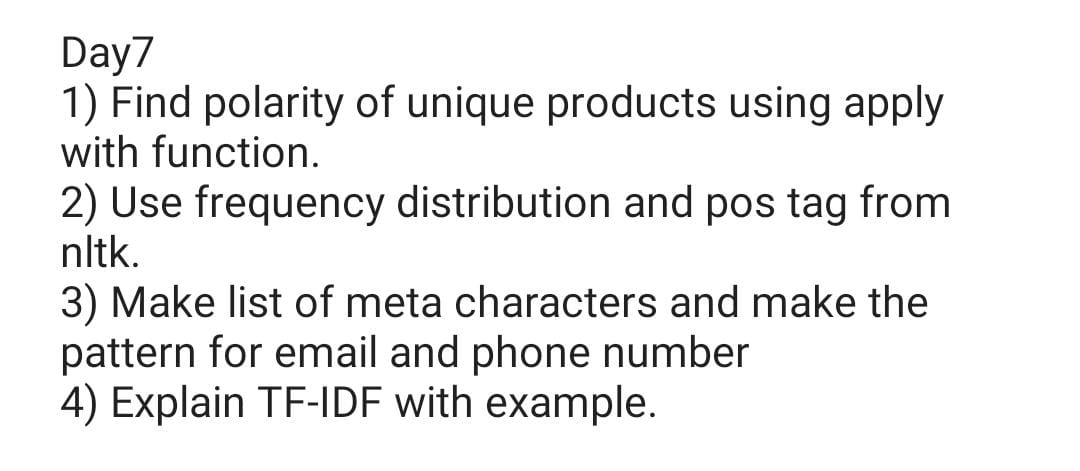
**3)**



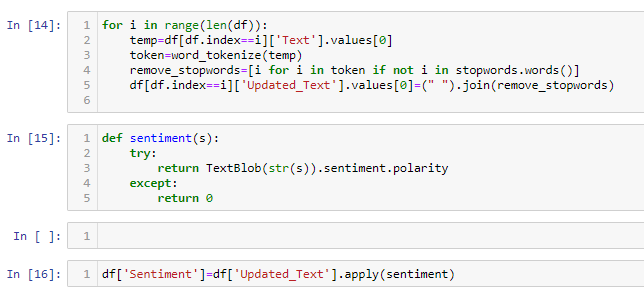
**4)** 

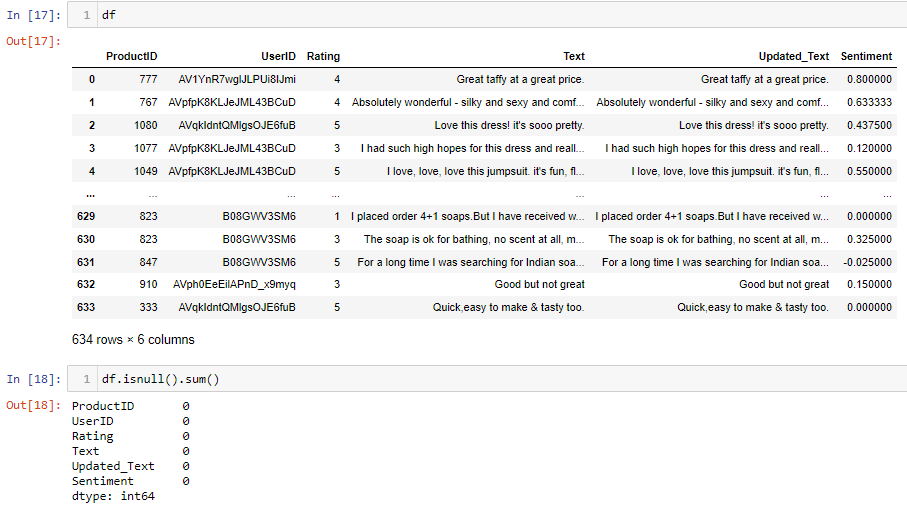


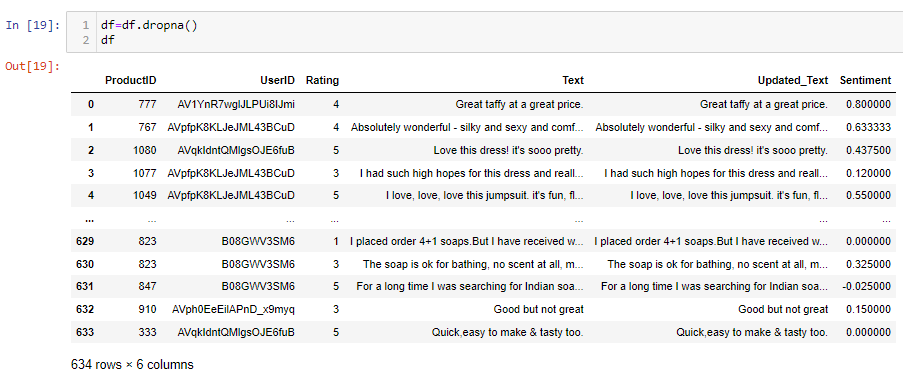
**Task 7 [03/06/2021]**

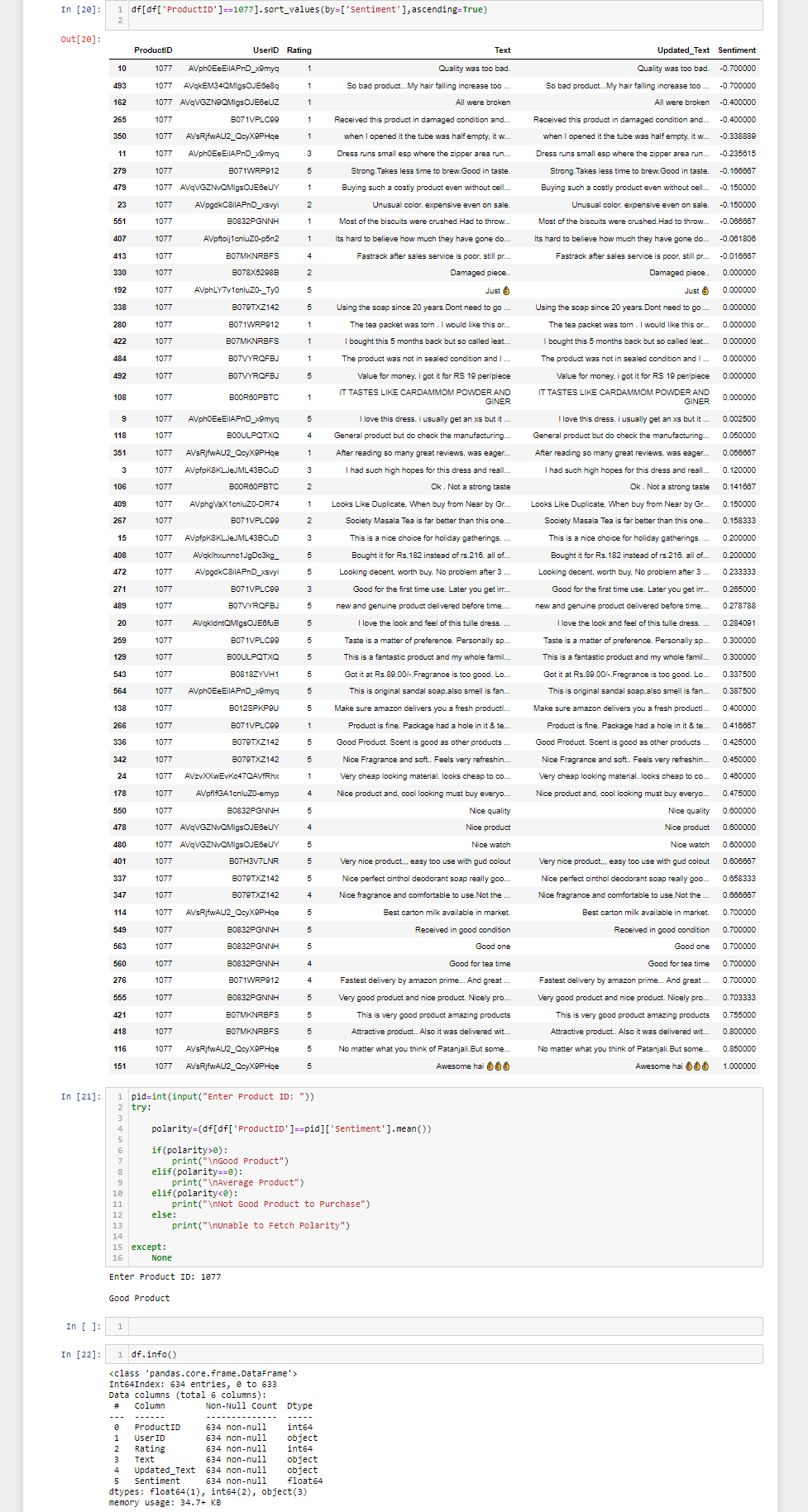
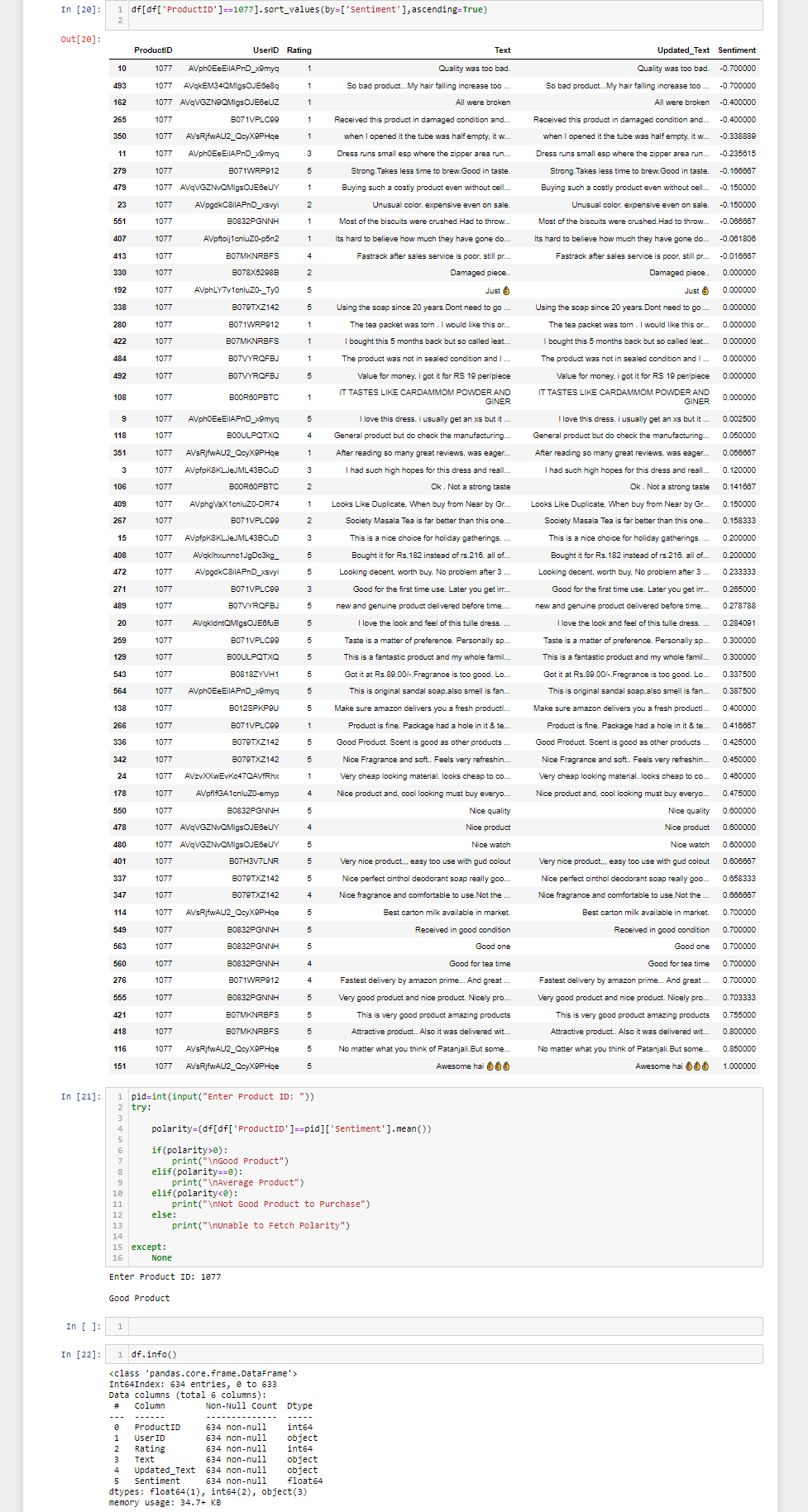
**Task: **

**1)** 

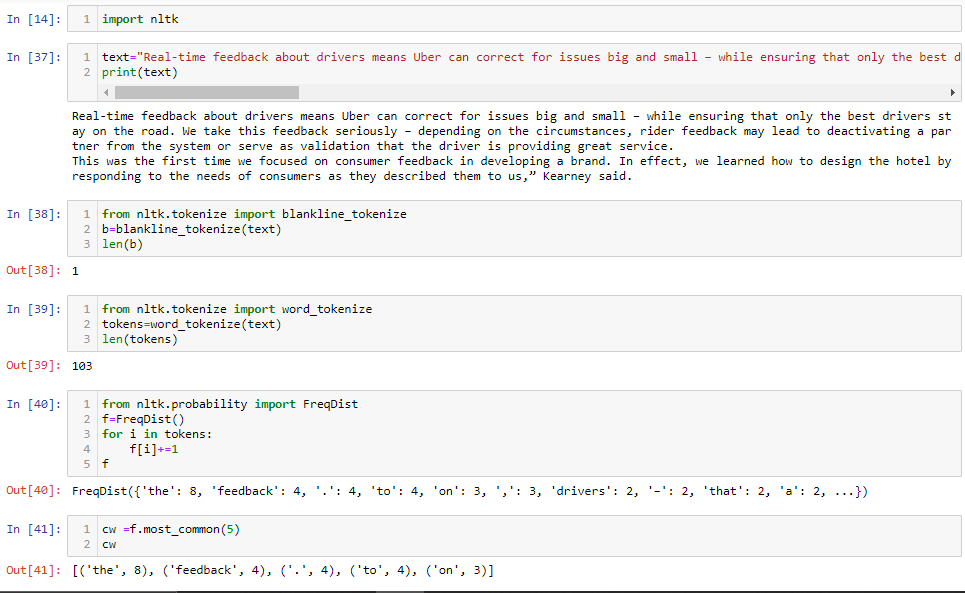


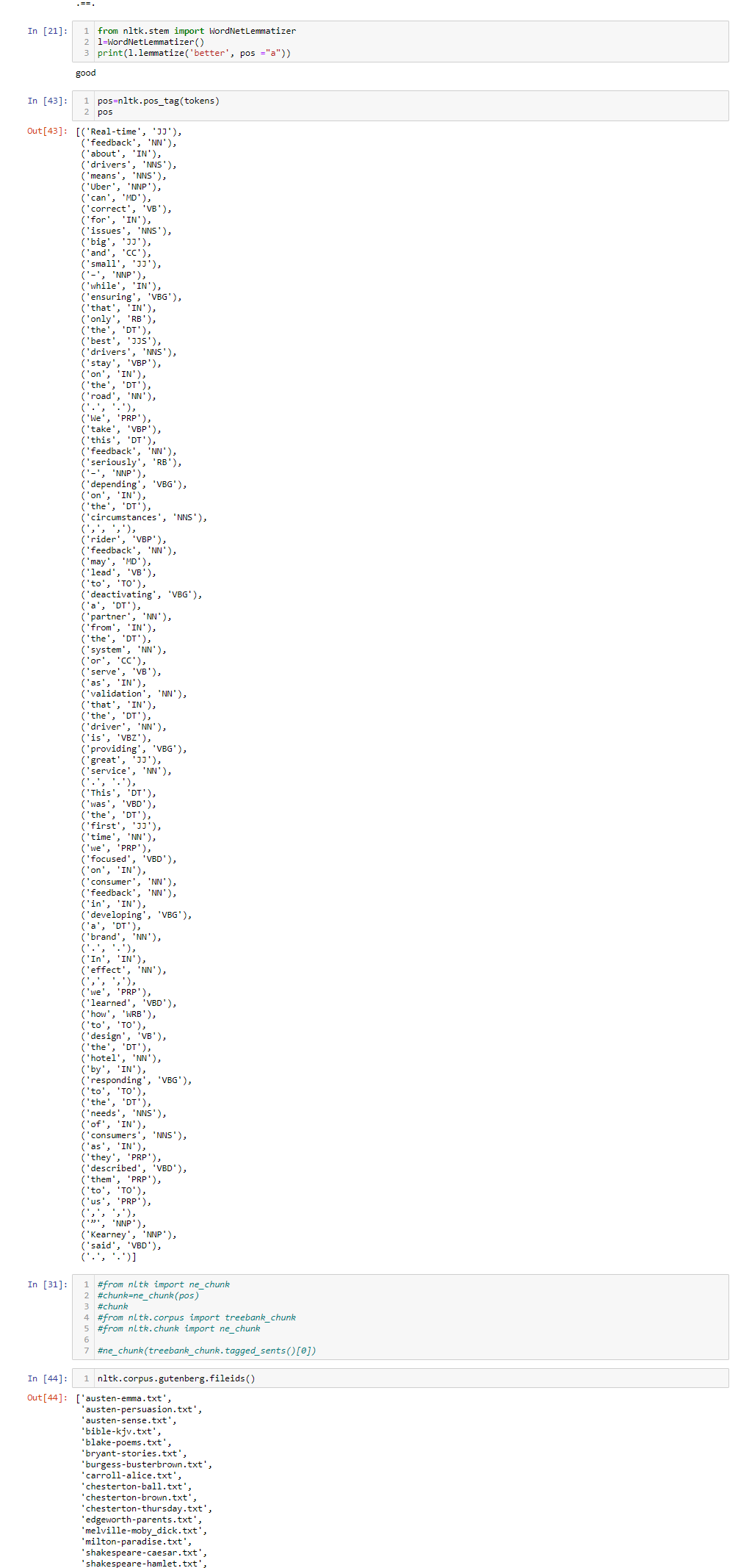






**2)**

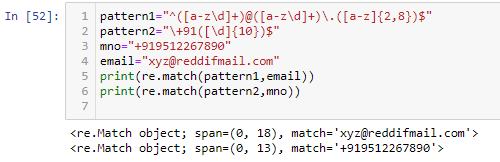




**3)**

|  |  |  |
| --- | --- | --- |
| **Character** | **Description** | **Example** |
| [] | A set of characters | "[a-m]" |
| \ | Signals a special sequence (can also be used to escape special characters) | "\d" |
| . | Any character (except newline character) | "he..o" |
| ^ | Starts with | "^hello" |
| $ | Ends with | "world$" |
| \* | Zero or more occurrences | "aix\*" |
| + | One or more occurrences | "aix+" |
| {} | Exactly the specified number of occurrences | "al{2}" |
| | | Either or | "falls|stays" |

**4)**

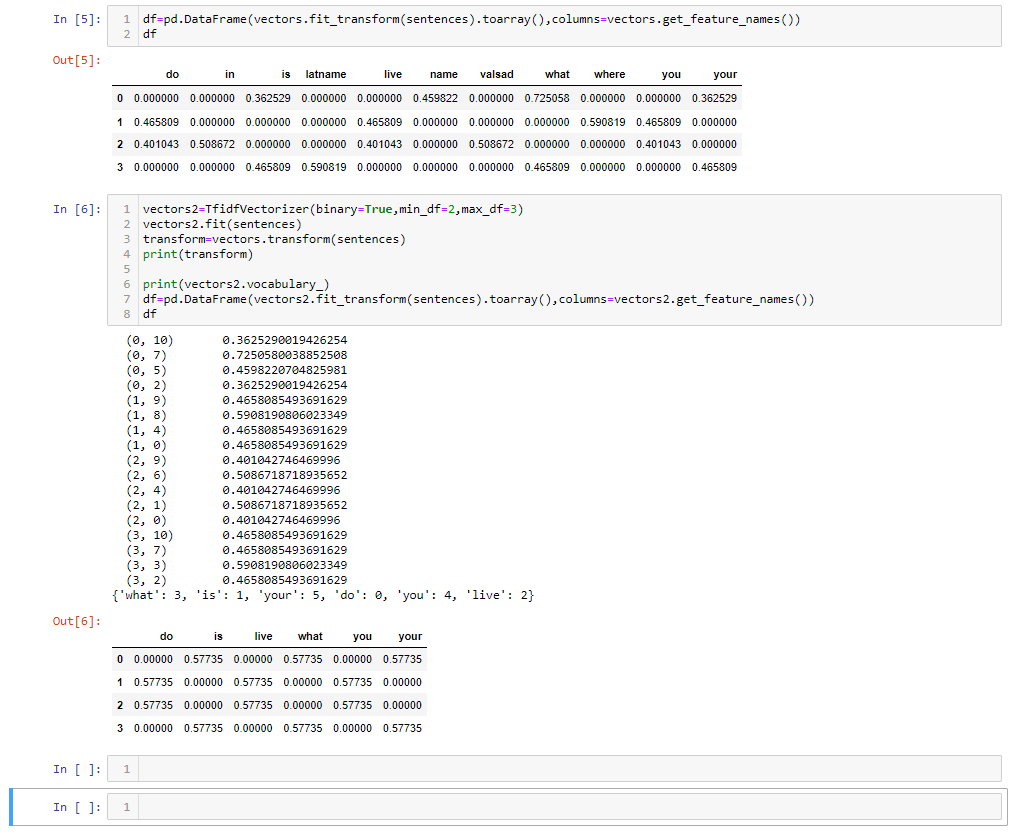


**Task 8 [04/06/2021]**

**Task: Implement TF-IDF**

**Code:**





**Task 9 [07/06/2021]**

**Task: Explain three techniques of Streaming.**

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”. Stemming is an important part of the pipelining process in Natural language processing. The input to the stemmer is tokenized words.

* For example: “**Flying**” is a word and its suffix is “**ing**”, if we remove “**ing**” from “**Flying**” then we will get base word or root word which is “**Fly**”.
* We uses these suffix to create a new word from original stem word.

The stem of the [verb](https://en.wikipedia.org/wiki/Verb) **wait** is **wait**: it is the part that is common to all its inflected variants.

1. **wait** (infinitive)
2. **wait** (imperative)
3. **wait**s (present, 3rd person, singular)
4. **wait** (present, other persons and/or plural)
5. **wait**ed (simple past)
6. **wait**ed (past participle)
7. **wait**ing (progressive)

Sometime spelling may also change in order to make a new word.

1. beaut**y**, dut**y** + -ful → beaut**i**ful, dut**i**ful (-y changes to i)
2. heav**y**, read**y** + -ness → heav**i**ness, read**i**ness (-y changes to i)
3. able, possible + -ity → ab**ility**, possib**ility** (-le changes to il)
4. permi**t**, omi**t** + -ion → permi**ss**ion, omi**ss**ion (-t changes to ss)

The main aim is to reduce the **inflectional**forms of each word into a common base word or root word or stem word.

**Inflection**is a process of word formation, in which a word is modified to express different grammatical categories such as tense, case, voice, aspect, person, number, gender, mood, animacy, and definiteness

There are majorly 2 errors in Stemming Algorithms which are as follows.

# OverStemming

Over-stemming is when two words with different stems are stemmed to the same root. This is also known as a false positive.

* universal
* university
* universe

All the above 3 words are stemmed to univers which is wrong behavior.

Though these three words are etymologically related, their modern meanings are in widely different domains, so treating them as synonyms in NLP/NLU will likely reduce the relevance of the search results

# UnderStemming

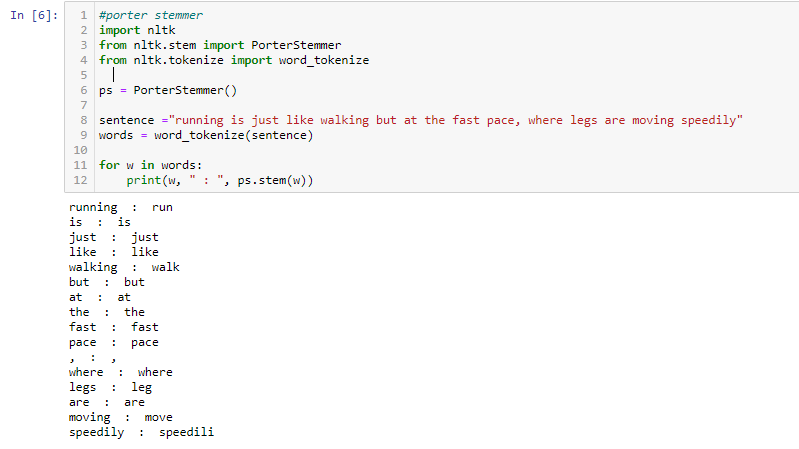
Under-stemming is when two words that should be stemmed to the same root are not. This is also known as a false negative. Below is the example for the same.

* alumnus
* alumni
* alumnae

As of now there are lot of ways by which we can stem a word and in this article we will be focusing on 3 stemming techniques which are part of truncating Stemming Algorithm, We wont be discussing about statistical or mixed Stemming Algorithm here.

# Porter Stemmer

* This is one of the most common and gentle stemmer, Its fast but not very precise.

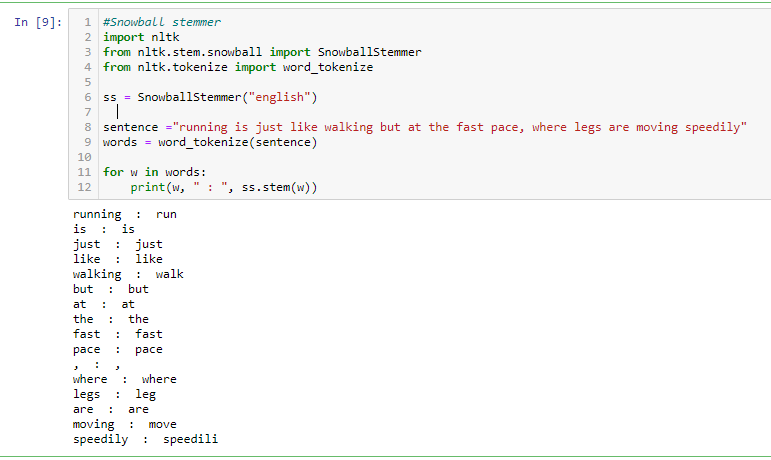


In output we can see how the suffixes were removed.

Look at the input and you can see we are passing “**was**” and getting “**wa**” as output. This is something which should be considered under less precise algorithm. To increase the precision another algorithm came which was SnowBall Stemmer.

# Snowball Stemmer

* The actual name of this stemmer is [English Stemmer](http://snowball.tartarus.org/algorithms/english/stemmer.html) or Porter2 Stemmer
* There were some improvements done on Porter Stemmer which made it more precise over large data-sets

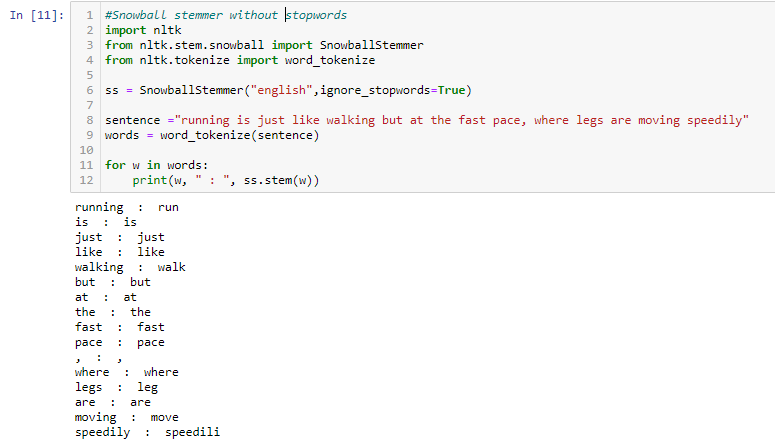


As this was an improvement over Porter Stemmer hence we can see in the results how gracefully it handled “was” input. There was lots of improvement done in this algorithm. Hence currently it is one of my favorite algorithm to work with.

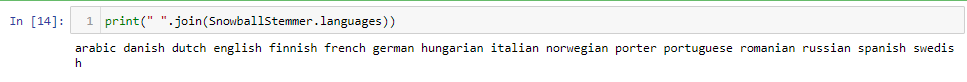
There is one more impotant feature added to this algorithm which was excluding Stop Word Stemming.

Due to this feature we observed difference in “was” input.

Below is the implementation for the same.



You can also check the languages supported in SnowBall Stemmer.



# Lancaster Stemmmer

* It is very aggressive algorithm
* It will hugely trim down your working set, this statement itself has pros and cons, sometime you many want this in your datasets but maximum time you will be avoiding it.
* Aggression can be observed by “Caring” input, It was converted to “car” which is altogether a different word in English dictionary.

# Conclusion

Snowball Stemmer cames out to be one of the best suited algorithm for my needs, but this totally depends on use case and dataset.

One more important thought, We should Do Stemming first or remove Stop Words first?????

So as per my thinking this depends,

* If your dataset is having stopwords with stems then you should go for stemming first and then stopword removal.
* If your dataset is not having stopword without stems then you should go for Stopwords removal first and then Stemming second.

**Task 10 [08/06/2021]**

**Task:**

* 1. FtaExplain Collaborative and content based filtering with example.
  2. Explain Cosine similarity with Equation.
  3. Explain RMSE and MSE with mathematical equation.
  4. Explain Matrix Factorization.

**1) Collaborative filtering**

Collaborative Filtering (CF) may be a mean of recommendation(advice) based on users’ past behavior. There are 2 classes of CF:

• User-based: check the similarity between target users and different users

• Item-based: check the similarity between the things that focus on users rates/ interacts with and different items

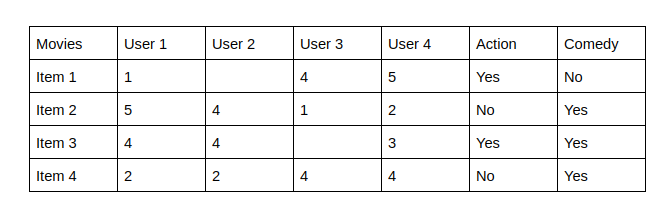
The key plan behind CF is that similar users share an equivalent interest which similar things area unit liked by a user.

Assume there are m users and n things, we tend to use a matrix with size m\*n to denote the past behavior of users. every cell within the matrix represents the associated opinion that a user holds. as an example, M\_ denotes how user i likes item j. Such matrix is named utility matrix. CF is like filling the blank (cell) within the utility matrix that a user has not seen/rated before supported the similarity between users or things. There are 2 kinds of opinions, explicit opinion and implicit opinion. the previous one directly shows how a user rates that item (think of it as rating an app or a movie), whereas the latter one only is a proxy that provides us heuristics regarding how an user likes a item (e.g. no. of likes, clicks, visits). explicit opinion is more straight-forward than the implicit one as we have a tendency to don't got to guess what will that number implies. as an example, there may be a song that user likes substantially, however he listens to that one time as a result of he was busy whereas he was taking note of it. while not explicit opinion, we can not be sure whether or not the user dislikes that item or not.

However, most of the feedback that we tend to collect from users are implicit. Thus, handling implicit feedback properly is extremely vital, however that's out of the scope of this journal post. I’ll move and discuss however CF works. Content based Recommender System:- It’s mainly classified as an outgrowth and continuation of information filtering research. In this system, the objects are mainly defined by their associated features. A content-based recommender learns a profile of the new user’s interests based on the features present, in objects the user has rated. It’s basically a keyword specific recommender system here keywords are used to describe the items. Thus, in a content-based recommender system the algorithms used are such that it recommends users similar items that the user has liked in the past or is examining currently.

**Content Based Filtering**

**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) Cosine Similarity**

* Cosine similarity is a measure used to determine how similar the documents are irrespective of their size.
* Mathematically, it measures the cosine function of the angle between two vectors projected inside a multi-dimensional area.

****

****

****

* As a similarity metric, how does cosine similarity differ from the number of common words?

When plotted on a multi-dimensional space, where every dimension corresponds to a word within the document, the cosine similarity captures the orientation (the angle) of the documents and not the magnitude. If you wish the magnitude, cypher the geometer distance instead.



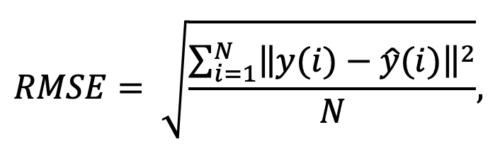
* The cosine similarity is advantageous as a result of although the 2 similar documents ar way apart by the euclidean distance as a result of the scale (like, the word ‘cricket’ appeared fifty times in one document and ten times in another) they may still have a smaller angle between them. Smaller the angle, higher the similarity.

**3) Root Mean Square Error**

Root mean square error or root mean square deviation is one of the most commonly used measures for evaluating the quality of predictions. It shows how far predictions fall from measured true values using Euclidean distance.

To compute RMSE, calculate the residual (difference between prediction and truth) for each data point, compute the norm of residual for each data point, compute the mean of residuals and take the square root of that mean. RMSE is commonly used in supervised learning applications, as RMSE uses and needs true measurements at each predicted data point.

Root mean square error can be expressed as



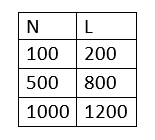
where N is the number of data points, y(i) is the i-th measurement, and y ̂(i) is its corresponding prediction.

Note: RMSE is NOT scale invariant and hence comparison of models using this measure is affected by the scale of the data. For this reason, RMSE is commonly used over standardized data.

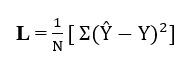
In machine learning, it is extremely helpful to have a single number to judge a model’s performance, whether it be during training, cross-validation, or monitoring after deployment. Root mean square error is one of the most widely used measures for this. It is a [proper scoring rule](https://sites.stat.washington.edu/raftery/Research/PDF/Gneiting2007jasa.pdf) that is intuitive to understand and compatible with some of the most common statistical assumptions.

**Mean Squared Error**

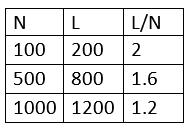
* Now consider we are using SSE as our loss function. So if we have a dataset of say 100 points, our SSE is, say, 200. If we increased data points to 500, our SSE would increase as the squared errors will add up for 500 data points now. So let’s say it becomes 800. If we increase the number of data points again, our SSE will further increase. Fair enough? Absolutely not!



* The error should decrease as we increase our sample data as the distribution of our data becomes more and more narrower (referring to normal distribution). The more data we have, the less is the error. But in the case of SSE, the complete opposite is happening. Here, finally, comes in our warrior — Mean Squared Error. Its expression is:

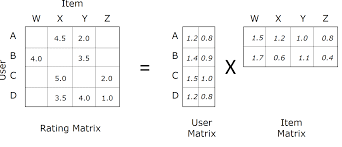


We take the average or mean of SSE. So more the data, lesser will be the aggregated error, MSE.



**4) Matrix Factorization**

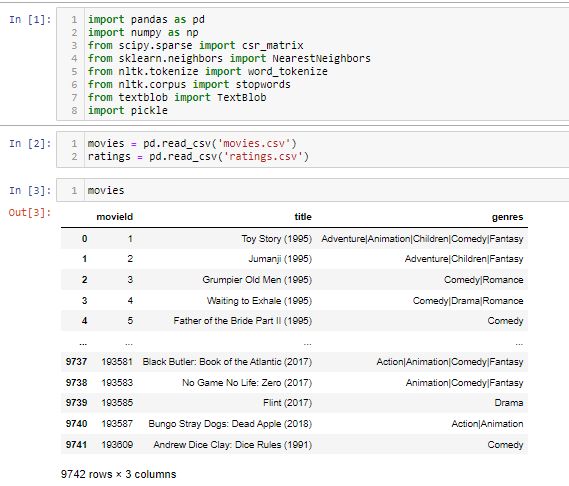
* Matrix factorization comes in limelight after Netflix competition (2006) when Netflix announced a prize money of $1 million to those who will improve its root mean square performance by 10%. Netflix provided a training data set of 100,480,507 ratings that 480,189 users gave to 17,770 movies.
* Matrix factorization is the collaborative based filtering method where matrix m\*n is decomposed into m\*k and k\*n . It is basically used for calculation of complex matrix operation. It is used for discovering latent features between two entities (can be used for more than two entities but this will come under tensor factorization)

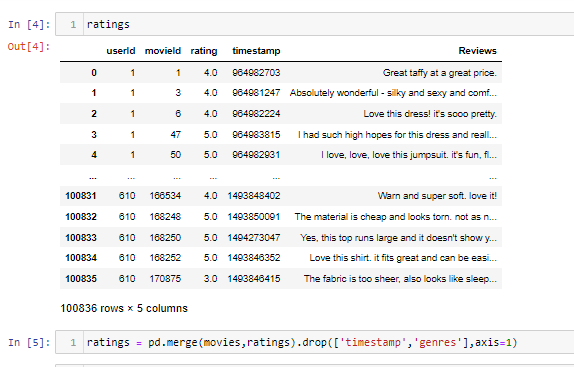
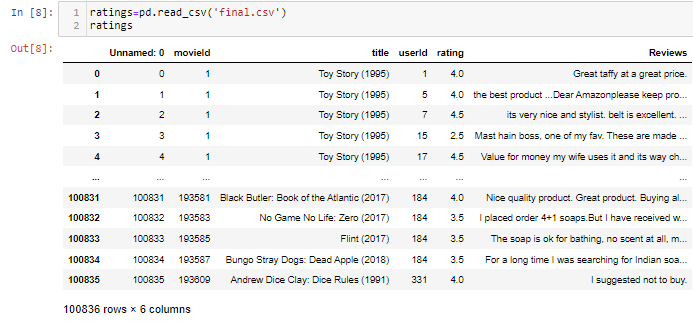


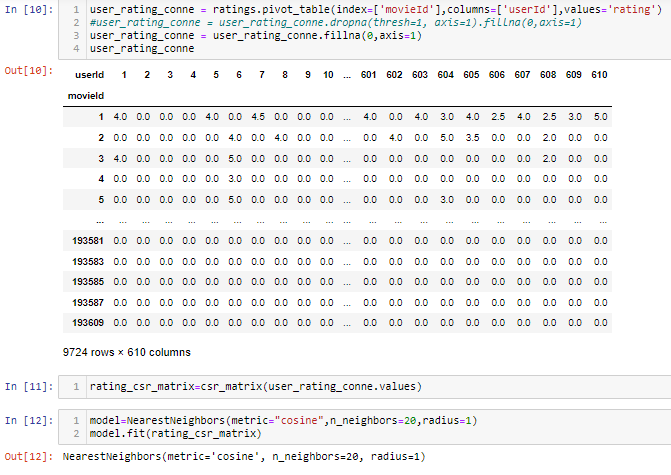
**Task 11 [09/06/2021]**

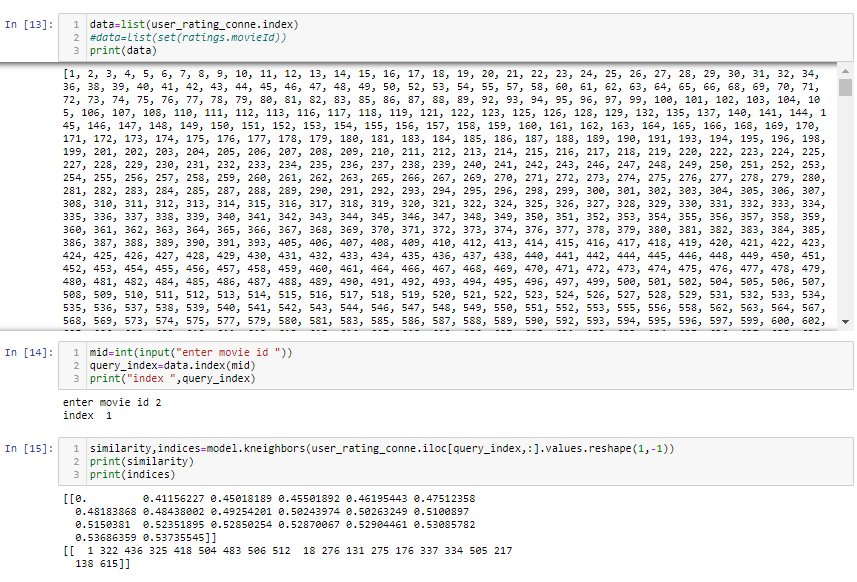
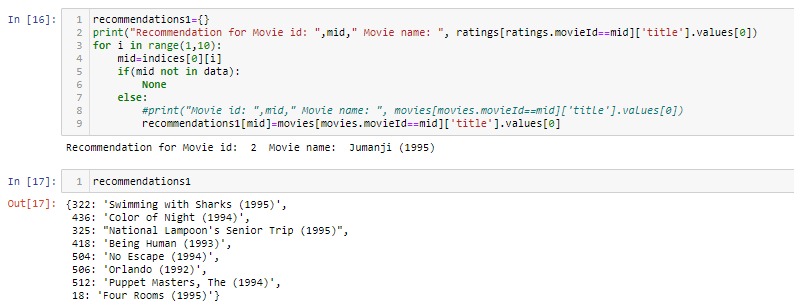
**Task: Perform Recommendation with any dataset**

**Code:**



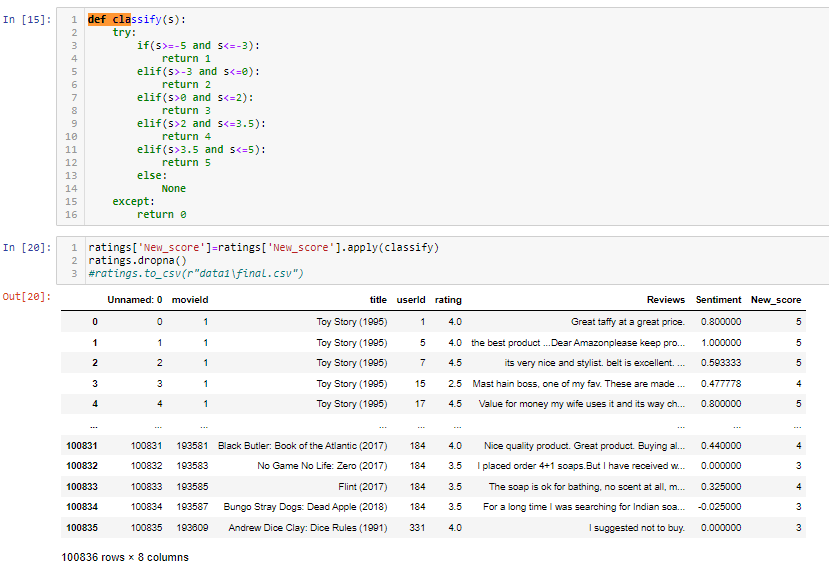
  



**Task 12 [11/06/2021]**

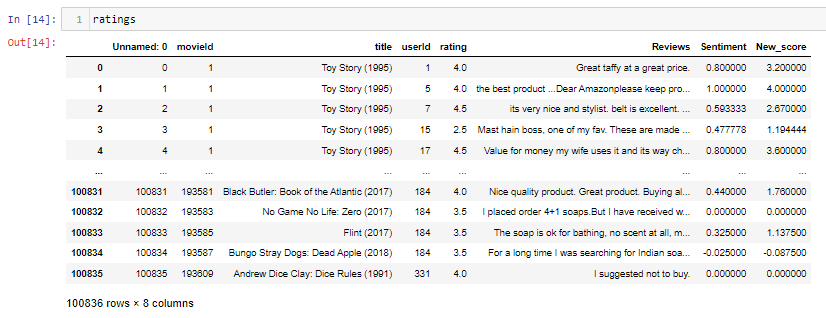
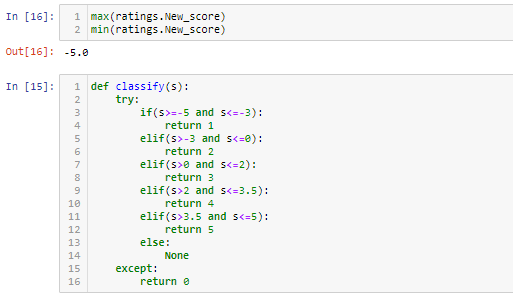
**Task: find the key from the dictionary containing 1-5 ratings as keys and 40 values**.

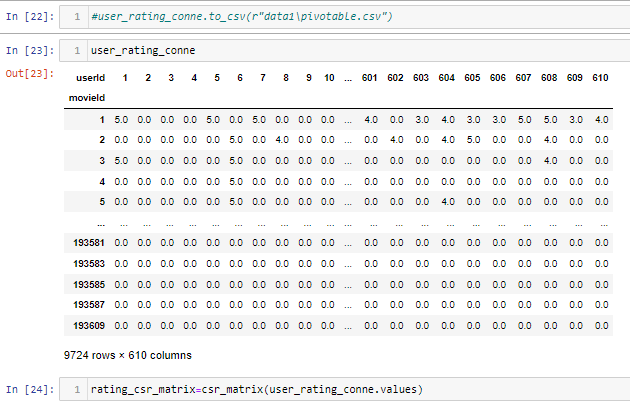
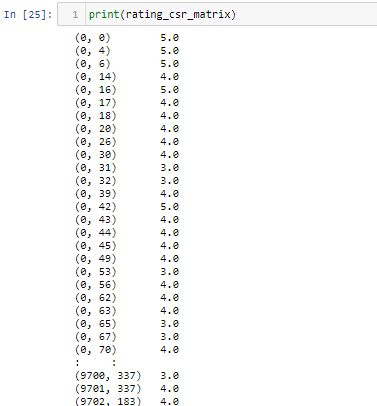


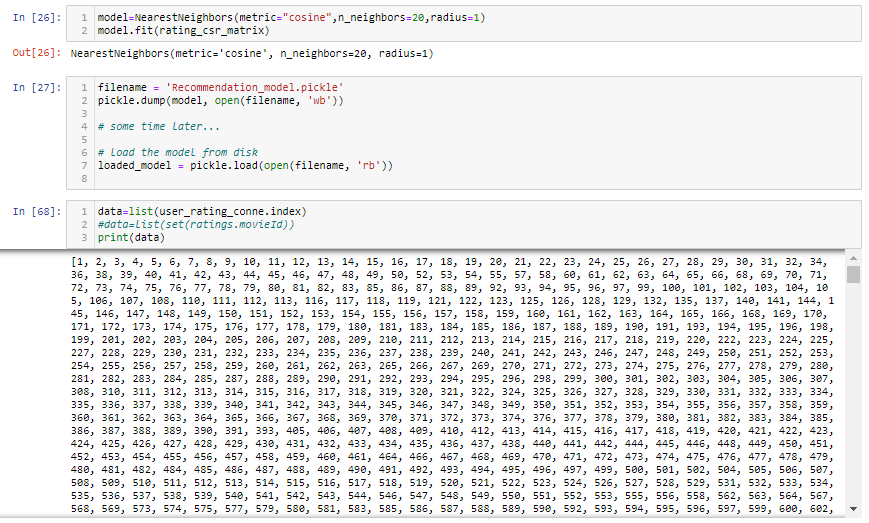
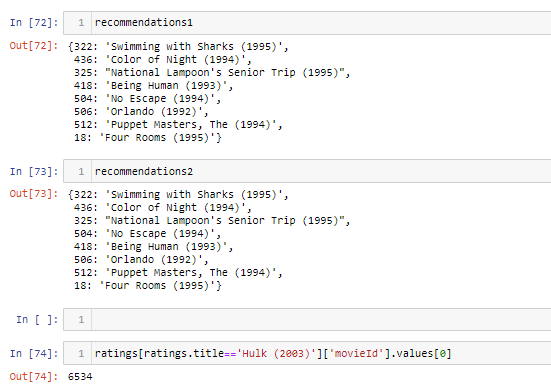
**Task 13 [14/06/2021]**

**Task: Compare the Recommendation with Rating and new Rating consists of Sentiment Score.**

**Code:**

**Task 14 [15/06/2021]**

**Task: Explain Surprise Package and its Working.**

* Surprise is a Python scikit for building and analyzing recommender systems that deal with explicit rating data.

Surprise 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, 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. Users can use both built-in datasets (Movielens, Jester), and their own custom datasets.
* Provide various ready-to-use prediction algorithms such as baseline algorithms, neighborhood methods, matrix factorization-based ( SVD, PMF, SVD++, NMF), and many others. Also, various similarity measures (cosine, MSD, pearson…) are built-in.
* Make it easy to implement new algorithm ideas.
* Provide tools to evaluate, analyse and compare the algorithms’ performance. Cross-validation procedures can be run very easily using powerful CV iterators (inspired by scikit-learn excellent tools), as well as exhaustive search over a set of parameters.

Code:

