Business Analytics and Text Mining Modeling Using python

Prof. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology Roorkee

Lecture-29
Database Using Python-Pandas-Part IV

Welcome to the course business analytics and text mining modeling using python. So, in the previous few lectures we have been discussing pandas, so let us start from the point where we stopped in the previous lecture. So, we are talking about arithmetic between objects with unmatch indices. So, what happens in those scenarios, so let us focus on some of those things. So, for matched indices wherever you know we are dealing with 2 objects and you know certain indices are going to be matching overlapping.

So, for operation would be applied but there are would be you know certain indices which are present in 1 object but are changing and absent in another object or present in the another object, absent in the first object. So, in those kind of unmatch indices a union of unmatch indices is created and an nan values are you know filled in those places. So, let us take an example, so we will create this series12 defined the series 12 object here, we will use the pd series function and can see the values and indexes here.

(Video Starts: 01:26)

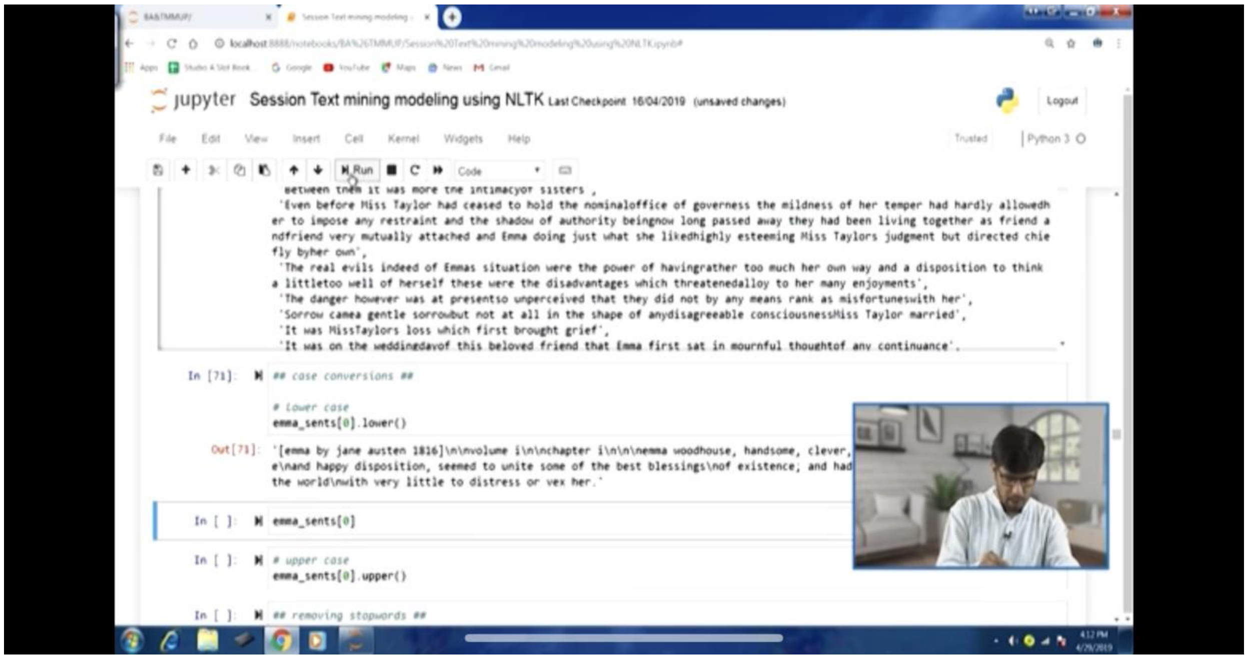
So, if I run this, this is the series12 and let us take another one, so series13 let us create another one series13 here and this is series13. So, you can see the in you can compare output 331 and 333 and you can see that the series 12 has you know 4 row indices a, c, d, e and series13, 5 indices a, c, e, f, g. So, few indices are common few are not common and of course the values you know, so if I apply an operation like series 12+3 is then wherever the indices are matching will have the you know sum total of those 2 elements values.

If I run this, so you can see for a, you can compare, a was present in both, so therefore 7.3-2.1. So, we got 5.2 but the places where the that particular element is missing in one of the series object there will get any nan. So, in this fashion you know whenever we are encountering unmatched and we are performing certain operations this is how it is going to be dealt with.

So, df5 data frame and let us take the df4 data frame and you can see row and I indices are matching acd acd here. So, if I reindex df4 and I insert one more row here one more index here, so df4 would be something like this. Now if I now dear df4 and df5 you know they will have 1 row be which would not match, so the row b. So, if I apply df4+df5 and look at the output b is there and nan is there.

So, in this fashion whenever the indices are not matching the output is going to be generated. Now we can also fill values whenever unmatch indices are involved we can fill those values, so we can fill nans let us say with zeros. So, for that we can use this fill_value argument that is available in add method because essentially till now we talked about the you know we were adding these you know objects.

(Refer Slide Time: 04:05)


jupyter Session Text mining modeling using NLTK Last Checkpoint 16/04/2019 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Logout
Run
Code

```
 between them it was more the intimacyor sisters',
 'Even before Miss Taylor had ceased to hold the nominaloffice of governess the mildness of her temper had hardly allowedh
er to impose any restraint and the shadow of authority beingnow long passed away they had been living together as friend a
ndfriend very mutually attached and Emma doing just what she likedhighly esteeming Miss Taylors judgment but directed chie
fly byher own',
 'The real evils indeed of Emmas situation were the power of havingrather too much her own way and a disposition to think
a littletoo well of herself these were the disadvantages which threatenedalloy to her many enjoyments',
 'The danger however was at presentso unperceived that they did not by any means rank as misfortuneswith her',
 'Sorrow camea gentle sorrowbut not at all in the shape of anydisagreeable consciousnessMiss Taylor married',
 'It was MissTaylors loss which first brought grief',
 'It was on the weddingdayof this beloved friend that Emma first sat in mournful thoughtof any continuance',

In [71]: ## case conversions ##

         # lower case
         emma_sents[0].lower()

Out[71]: '[emma by jane austen 1816]\n\nvolume i\n\nchapter i\n\n\nemma woodhouse, handsome, clever,
         e\nand happy disposition, seemed to unite some of the best blessings\nof existence; and had
         the world\nwith very little to distress or vex her.'

In [ ]:  emma_sents[0]

In [ ]:  # upper case
         emma_sents[0].upper()

In [ ]:  ## removing stopwords ##
```

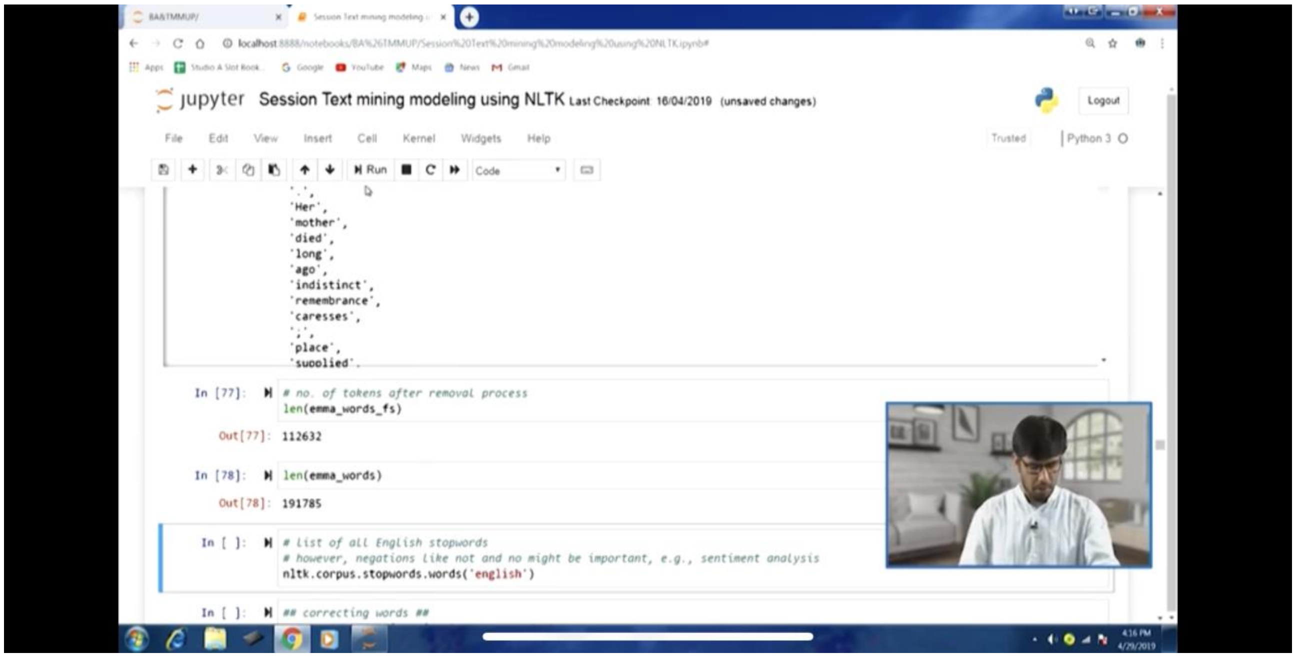
So, let us say df4.add and we are adding this with df5 and we can say that in the r1 fill_value at 0, so in this case will you know have that kind of output. If I run this you can see here in the output but if you look at the row number b, all the values are still in a nan. So, it did not work, so why did not work because the df4 it had any end values. So, the original data frames that we are using this should not be having nan values for it to work.

It is only because of the operation that the nan values which are being produced they are going to be filled with this particular approach. So, let us take another example, so in this what will do we will fill the df4 values 1 to 2 that means that particular second row and fill it with value 2. So, you can see df4 now df4 has changed, now I apply again the same thing df4.add df5, fill_value 0.
(Refer Slide Time: 00:43)
Then even though I have an unmatch indices, now you know in this case you know those values are going to be filled. Let us move forward let us take another operation 2/df4 here, so in this case we will have these kind of values. We also have a div method here also we can call this df4.div and pass on the value 2, so another output here in this fashion, we have rdiv method which is div with arguments flipped.

So, when we use the division operator this / operator there, so instead of that we can also use rdiv which will achieve the same thing because the arguments are going to be flipped, so this is another way to achieve the same thing. Now let us move forward, now then we will talk about another aspect which is about arithmetic between data frame and series. So, there are certain similarities as you might have noticed in data frame and series objects and so how the arithmetic thing will work out.

So, data frame and see series they can be considered similar to 2d array and 1d array , so that is the scenario here. So, we will pick you know 2d array and 1d array to demonstrate this, so let us initialize this let us see first how this thing works out in array objects. So, let us initialize array a double r object and you can see this is a 2d array here that we have you know defined, save also you can see 3*4.

(Refer Slide Time:08:04)


jupyter Session Text mining modeling using NLTK Last Checkpoint 16/04/2019 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
Logout

```
 '.',
 'Her',
 'mother',
 'died',
 'long',
 'ago',
 'indistinct',
 'remembrance',
 'caresses',
 ';',
 'place',
 'supplied',
```

```
In [77]:  # no. of tokens after removal process
          len(emma_words_fs)

Out[77]:  112632

In [78]:  len(emma_words)

Out[78]:  191785

In [ ]:   # list of all English stopwords
          # however, negations like not and no might be important, e.g., sentiment analysis
          nltk.corpus.stopwords.words('english')

In [ ]:   ## correcting words ##
```

And if I look at one of the you know array 0 if I look at array bracket 0, then this becomes a 1d array, you can look at the save just 1 dimension with 4 elements. So, if I perform a difference between 2d array in 1d array, so let us see what is the output here, so array-array(0), so we are differencing it to you know one day are 1d array from a 2d array. So, you can see the output and you can see all the you know elements that were there all the rows that were there in 2d array you know difference has been taken and subtraction is performed once for each row.

So, you see this is a kind of broadcasting has happened, so where for each of the row in the 2d array, 1d array has been subtracted in each of those rows. So, you can see kind of broadcasting has happened, so if you look at how the different thing happens in 2d array and 1d array probably something similar will happen in case of data frame and series. So, now let us take an example df4 and let us take the you know first row here India4 as a series.

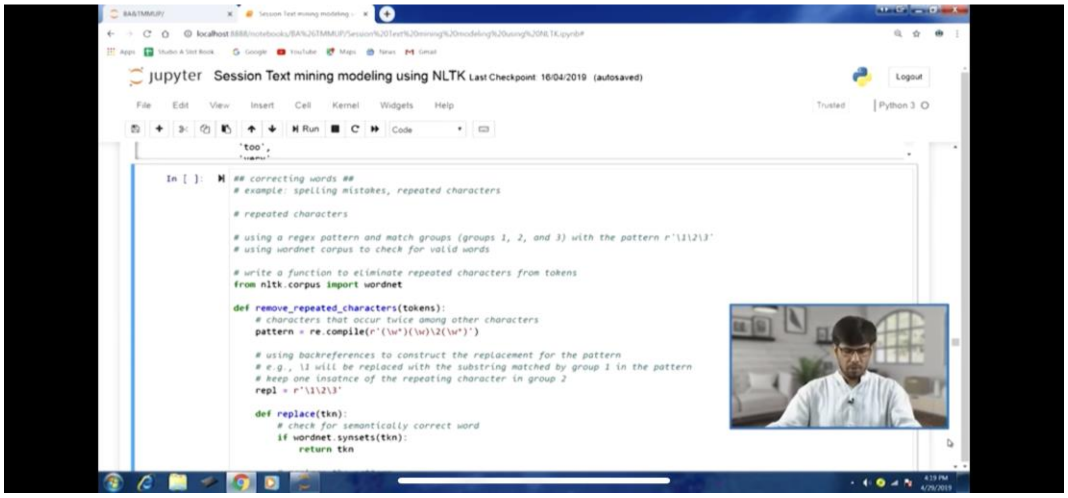
So, this is our row here you can see index corresponding to row index a, now if I you know perform this subtraction df4-series0_df4. So, a broadcasting will happen here again if I run this and a broadcasting here as also happen. So, in data frame each of the row they have been you know subtracted with the series. Now let us move forward again coming to unmatched indices here as we said a union of an imagined unmatch indices with any nan values kept.

So, we will take another example here series14, so these are the values, now if I add df4 with this series14 here. Now this series14 here, so you can see that what will happen is because nothing is matching there. So, another is going to be added here, so df4+series, so wherever the values present that value is going to be added. Otherwise you can see here the indices that we have in the series daily Hyderabad and Bangalore.

So, Hyderabad is not present in df4, so that particular you know column is going to be created and Mumbai is also on my Mumbai is not present in the series14 and Hyderabad is not present in df4. So, those 2 columns become the unmatched, so in this case if I add this, so we will get this kind of response. Now let us talk about the broadcasting over the columns till now we have taken broadcasting over the rows.

So, we can use various arithmetic methods for to perform this, so for example submethod we have which we can be use for subtraction. So, match will match index axis and broadcast over columns, so let us take this series 15 this is nothing but we will define this with using Mumbai column in df4. And this is series15 as you can see and this is df4 and you call this method df4.sub and we can pass on this series15.

(Refer Slide Time: 10:26)


jupyter Session Text mining modeling using NLTK Last Checkpoint 16/04/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
'too',
```

```
In [ ]:  ## correcting words ##
         # example: spelling mistakes, repeated characters

         # repeated characters

         # using a regex pattern and match groups (groups 1, 2, and 3) with the pattern r'\1\2\3'
         # using wordnet corpus to check for valid words

         # write a function to eliminate repeated characters from tokens
         from nltk.corpus import wordnet

         def remove_repeated_characters(tokens):
             # characters that occur twice among other characters
             pattern = re.compile(r'(\w*)(\w)\2(\w*)')

             # using backreferences to construct the replacement for the pattern
             # e.g., \1 will be replaced with the substring matched by group 1 in the pattern
             # keep one insatnce of the repeating character in group 2
             repl = r'\1\2\3'

             def replace(tkn):
                 # check for semantically correct word
                 if wordnet.synsets(tkn):
                     return tkn
```

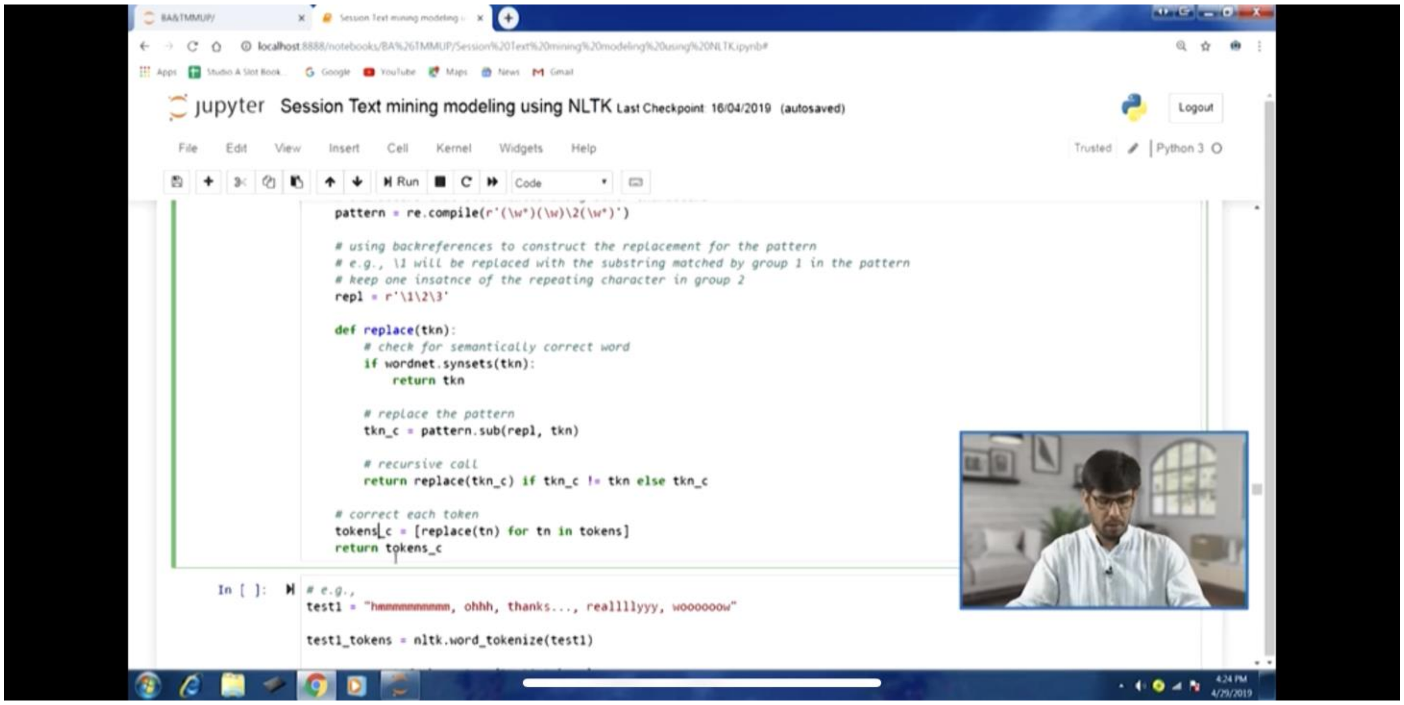
And we can specify axis in index axis=index and the broadcasting will happen over column, so if I run this you can see the values Delhi, Mumbai, Bangalore along the columns they have been changed because rod broadcasting this time happened over the column. Now let us talk about another important aspect of you know this pandas applying functions and method.

So, first we will start with NumPy element wise array functions that we have discussed before also. So, let us take df4 as an example here you can see the output, now we will call this function np.abs we can see that we are passing on just like in np.ab abs the numerical the NumPy function use to pass on 2d array and other arrays. Here we are passing df4 which is also you know 2 dimensional, so absolute value for each element is going to be computed.

So, NumPy functions are going to work well here, now there are certain data frame methods also available, so we can use them also. So, for example apply methods, so what is the role of apply method, so this particular method can be use to apply a function on all columns or rows. So, let us take an example, so let us define this function f using an lambda keyword, so this is nothing but difference between maximum value and you know minimum value for a given series.

Let us say, so difference between max and min value of a series, so let me define this function f. Now what we will do, we will call this function this method df4.apply and we will pass on this f is this particular lambda function here. And this would be applied in each of the series you know in each of the series along the you know column here. So, if I run this you can see that max and min values difference of that has been displayed in the outward you can see for the daily column 6-0 Mumbai 7-1 and Bangalore 8-2.

(Refer Slide Time:15:24)



jupyter Session Text mining modeling using NLTK Last Checkpoint 16/04/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
    pattern = re.compile(r'(\w*)(\w)\2(\w*)')

    # using backreferences to construct the replacement for the pattern
    # e.g., \1 will be replaced with the substring matched by group 1 in the pattern
    # keep one insatnce of the repeating character in group 2
    repl = r'\1\2\3'

    def replace(tkn):
        # check for semantically correct word
        if wordnet.synsets(tkn):
            return tkn

        # replace the pattern
        tkn_c = pattern.sub(repl, tkn)

        # recursive call
        return replace(tkn_c) if tkn_c != tkn else tkn_c

    # correct each token
    tokens_c = [replace(tn) for tn in tokens]
    return tokens_c
```

```
In [ ]: # e.g.,
        test1 = "hmmmmmmmmmm, ohhh, thanks..., reallllyyy, woooooow"

        test1_tokens = nltk.word_tokenize(test1)
```

So, the same output has been displayed here, similarly we can also broadcast over the rows for that we will have to specify this axis keyword argument here. So, axis will have to specify as columns and then we would be able to apply the function f you know over the rows. So, we can call the df4.apply and you can pass on the arguments and you can see the output here, now let us talk about another data frame sum method.

So, here if I just call df4.sum without any arguments then will be producing column sum, so column wise we will be generating sums. So, let me run this and for each of the columns Delhi, Mumbai, Bangalore we have got the summed up value. Similarly if I specify in the axis you know keyword argument here and column then we will obtain the row sums here, so along each row will be will be able to obtain sums, so if I run this you can see the output.

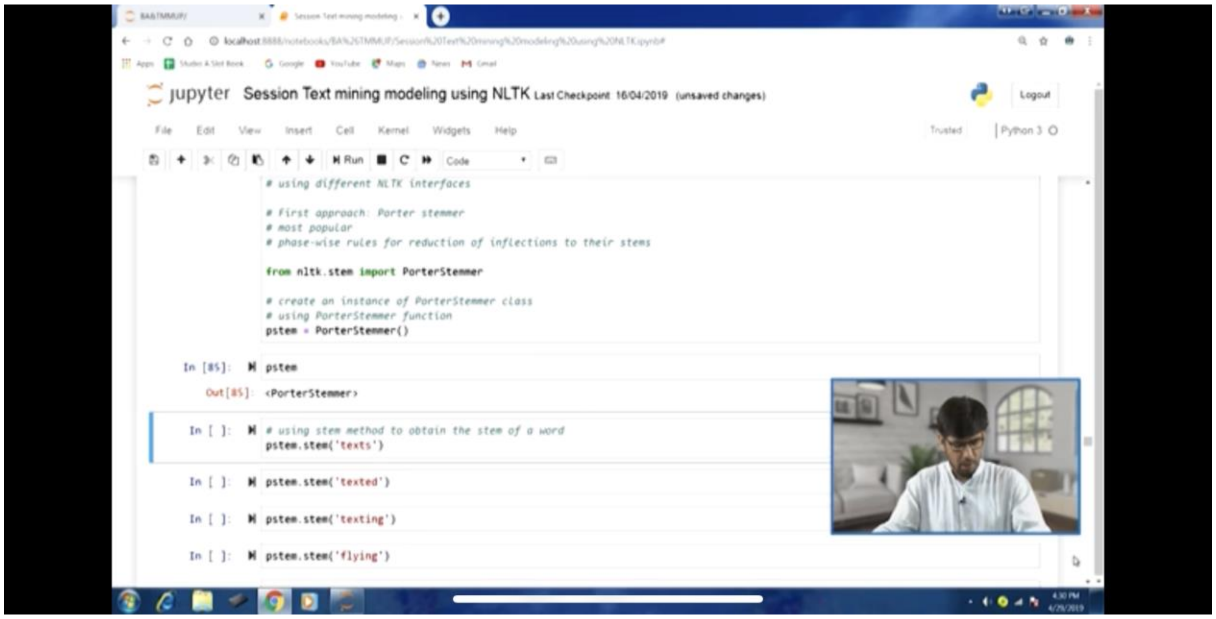
Similarly there is another important method min method, so this is also data frame method, so again for this one also if I call like df4.min without specifying the you know axis. Then we will take the column means, if I run this by default will take the column means and if I want to take row means I will have to specify axis=columns and we will get the row means, now returning multiple values using series, so that can also work.

So, let us take example of this function of f1 which using which will be returning such a series. So, you can see in the return statement what we have pd.series xmin, x.max. So, we are returning multiple series and with the index also min, max with the series object we are returning. So, let me define f1 and we can apply this f1 on this data frame df4 using the apply method, so if I run this you can see the output we have 2 row indices min and maximum.

And for each of those columns those particular values have been written, now let us talk about the some of the python methods, the element wise python methods which can also be used you know on these data frame and series objects. So, to demonstrate this will take this example where we want to compute a formatted string from a you know floating point value, so for this will be applying this you know apply map method.

And to demonstrate this will create this lambda function f2 where if you look at the expression here we are just formatting the value in a string formatted string you know %0.2 up to 2 decimal point. And then we will calling this apply map method and passing on this you know lambda function which is going to be applied you know. So, if I run this, this is again element wise, so all the elements present in the data frame and you can see that earlier the default output that we typically had, we had just 1 decimal point here, now we have 2 decimal point here.

(Refer Slide Time: 21:24)


jupyter Session Text mining modeling using NLTK Last Checkpoint 16/04/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

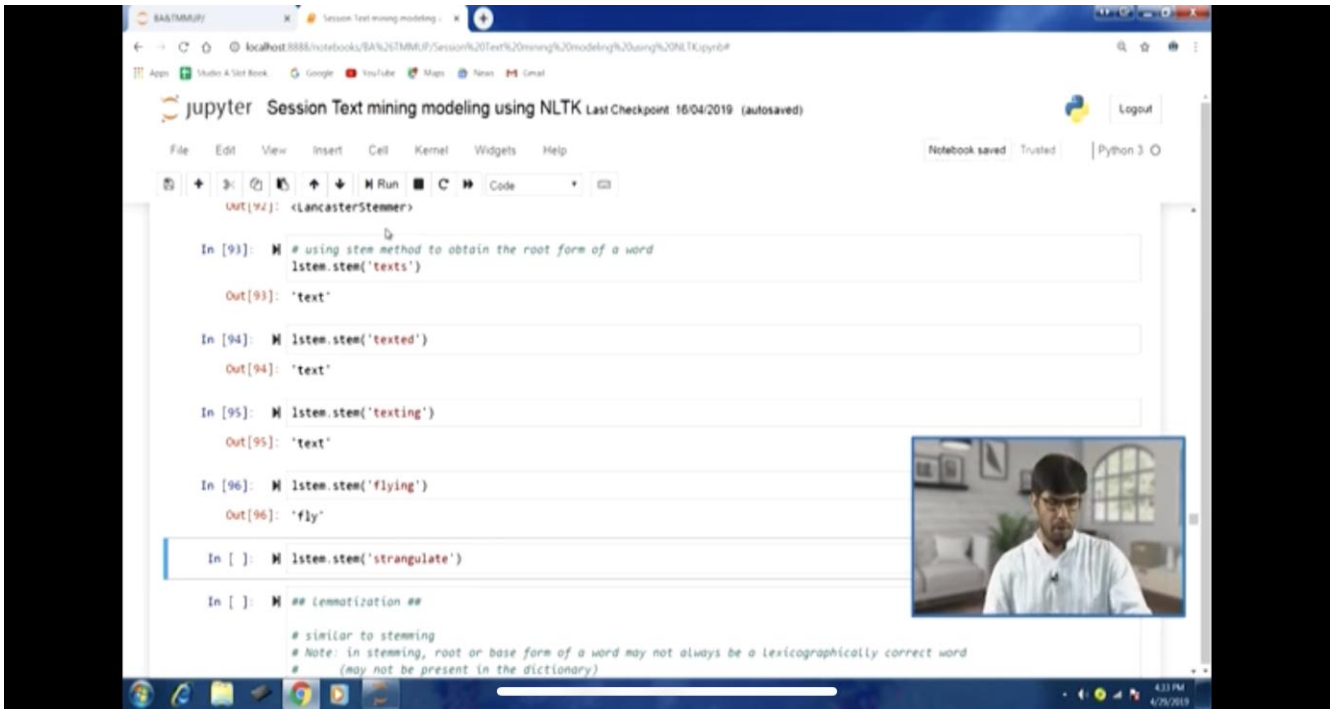
```
# using different NLTK interfaces

# First approach: Porter stemmer
# most popular
# phase-wise rules for reduction of inflections to their stems

from nltk.stem import PorterStemmer

# create an instance of PorterStemmer class
# using PorterStemmer function
pstem = PorterStemmer()

In [85]: pstem
Out[85]: <PorterStemmer>

In [ ]: # using stem method to obtain the stem of a word
        pstem.stem('texts')

In [ ]: pstem.stem('texted')

In [ ]: pstem.stem('texting')

In [ ]: pstem.stem('flying')
```

Similarly we have a map method for series which achieves the same functionality here just like apply map for data frame. So, we can call df4 and within brackets Delhi, so this is a series and we can call this method.map and pass on f2 and the same kind of operation would be applied. Now let us focus on another you know important operation sorting, so sorting by index, so for this we can perform this by using a sort_index method.

So, by default when we use this method we get the ascending ordering, so lower value comes first and then followed by you know comparatively higher values. So, let us take the example of the series16, here we have 4 values, 4, 7, -3, 2 and you can see the index also. So, let me define this and the series16 and we can call this method sort_index, so see series16.sort_index and let us have a look at the output.

So, you can see the indices they were d, a, b, c now these indices have been sorted and you can see in the output 378 abcd. So, the index have been you know the indices have been sorted in this output. Now let us take it example of a data frame here, so this is our data frame, so here also you can see that column indices and row indices they are not sorted. So, again we can call sort_index method to sort those indices.

(Refer Slide Time: 24:52)



jupyter Session Text mining modeling using NLTK Last Checkpoint 16/04/2019 (autosaved)

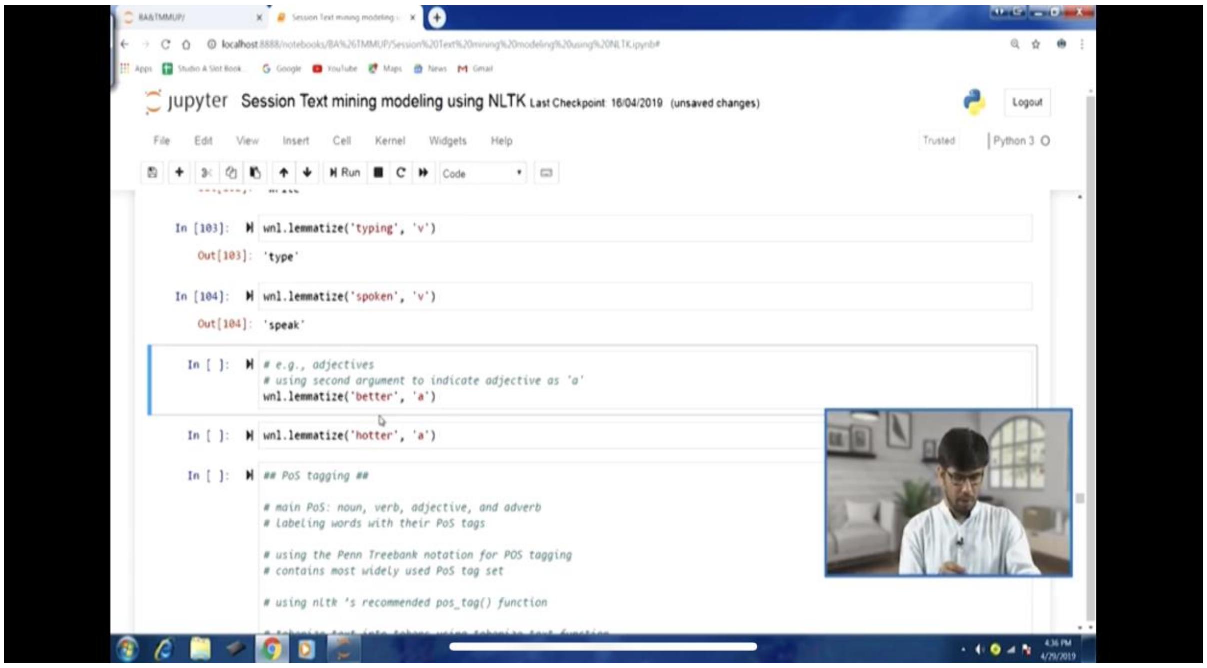
```
Out[92]: <LancasterStemmer>

In [93]:  # using stem method to obtain the root form of a word
          lstem.stem('texts')

Out[93]: 'text'

In [94]:  lstem.stem('texted')

Out[94]: 'text'

In [95]:  lstem.stem('texting')

Out[95]: 'text'

In [96]:  lstem.stem('flying')

Out[96]: 'fly'

In [ ]:   lstem.stem('strangulate')

In [ ]:   ## Lemmatization ##

          # similar to stemming
          # Note: in stemming, root or base form of a word may not always be a lexicographically correct word
          #      (may not be present in the dictionary)
```

So, df6 starts sort_index and you can see by default we do not specify the axis then it will sort the row indices if we specify the axis which is in the next example it will sort the you know column index you can see abcd. Now as we said that the default is you know ascending, so you know we have a keyword argument also to you know change that, so we can specify ascending=false, so we will get the descending order here.

So, if I apply again the df6.sort_index let us say 1 axis 1, column you know index ascending false. So, instead of abcd will get dcba, so if I run this, so descending ordering sorting has been performed here. Now that was about sorting in you know indexes we had 2 indexes row and column index, now let us talk about sorting by values.

So, with the values of these objects series and data frame how they can be sorted, so we have this matter sort_values method and typically if any nan values are present they are going to be you know kept at the end. So, let us again take the example of series16 will call.sort_values method and you can see values have been sorted -3, 2 then 4, 7. Similarly for the data frame let us create another define another data frame df7, this is the data frame and we can call this method sort_values.

(Refer Slide Time: 38:13)


jupyter Session Text mining modeling using NLTK Last Checkpoint 16/04/2019 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Logout
Run
Code

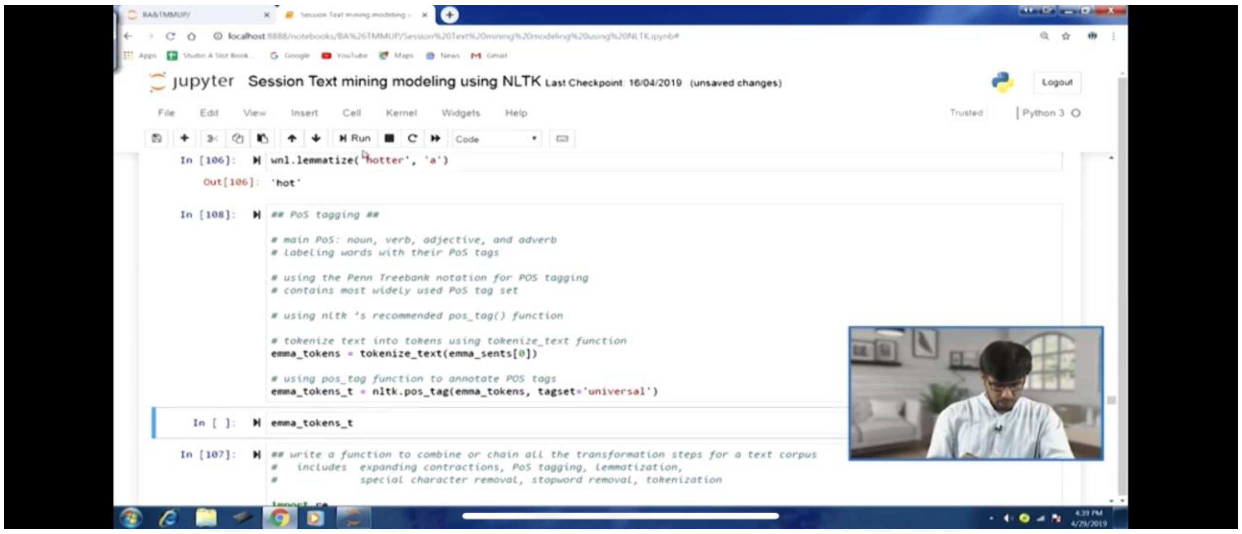
```
In [103]: wnl.lemmatize('typing', 'v')

Out[103]: 'type'

In [104]: wnl.lemmatize('spoken', 'v')

Out[104]: 'speak'

In [ ]:   # e.g., adjectives
          # using second argument to indicate adjective as 'a'
          wnl.lemmatize('better', 'a')

In [ ]:   wnl.lemmatize('hotter', 'a')

In [ ]:   ## PoS tagging ##

          # main PoS: noun, verb, adjective, and adverb
          # labeling words with their PoS tags

          # using the Penn Treebank notation for POS tagging
          # contains most widely used PoS tag set

          # using nltk 's recommended pos_tag() function
```

And there we can specify the column you know that we would like to use to perform this sorting by values. So, for this we can use this by keyword argument, so by=b, so b column we are picking here, so the values in that part of the column b are going to be you know sorted. So, sorting is based on that, so you can see have a look at the output, so other things they are automatically aligned.

Similarly we can do the sorting using multiple columns also, so in the this method sort_values and the by argument that we have we can pass on a list of you know column indices to you know perform this. So, in this case these 2 column a, b, so you can see first we had is specified a column, so first sorting is done by using values of column a, so you can see 0, 0, 1, 1 and then within that you know those smaller groups you 0, 0.

So, second column is used to perform so-3, 4 comes first, now you can see then for you know as per the you know column a we have 1, 1 and column b within that a small group 2, 7, 2 and 7. So, in this fashion this you know whenever we are using multiple columns this is how the sorting is going to be performed. Now sometimes we required to perform you know we required to evaluate ranking, so for that we have this rank method.

(Refer Slide Time: 30:34)


jupyter Session Text mining modeling using NLTK Last Checkpoint 16/04/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [106]:  wnl.lemmatize('hotter', 'a')
Out[106]:  'hot'

In [108]:  ## PoS tagging ##

           # main PoS: noun, verb, adjective, and adverb
           # labeling words with their PoS tags

           # using the Penn Treebank notation for POS tagging
           # contains most widely used PoS tag set

           # using nltk 's recommended pos_tag() function

           # tokenize text into tokens using tokenize_text function
           emma_tokens = tokenize_text(emma_sents[0])

           # using pos_tag function to annotate POS tags
           emma_tokens_t = nltk.pos_tag(emma_tokens, tagset='universal')

In [ ]:    emma_tokens_t

In [107]:  ## write a function to combine or chain all the transformation steps for a text corpus
           #   includes  expanding contractions, PoS tagging, lemmatization,
           #             special character removal, stopword removal, tokenization
```

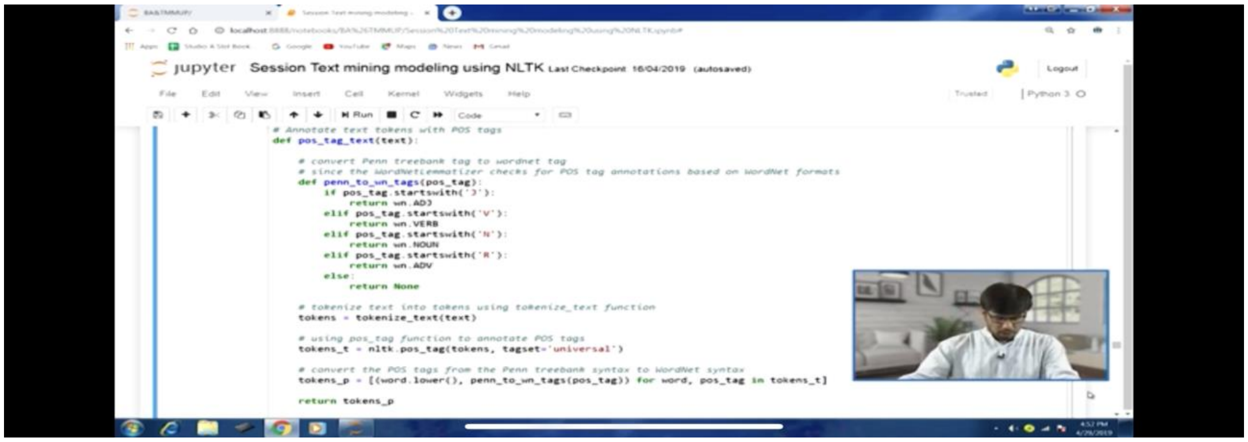
So, here also the default ordering is ascending ordering, so you know lower you know value it will be ranked 1. And so let us take an example series17 and you can have a look at the values, so 7, -5, 7 and all this, so if I call the rank method on this series, series17 do not rank and if I run this and you can compare the output 388 and output 389 here. And you can see that the lowest value was-5 for the index 1 and that has been given rank 1.

Because the default ranking is ascending, so the lowest value has been given the rank 1 here, so we can always change that using the ascending argument. Now another thing which is about you know breaking ties using order of listing, so if you look at the output 389 there are index you know row index0 and row index2, we have the value 6.5 as the rank. So, there a min of the same value is there, then the a min rank is provided.

So if we do not want to perform that we can implement another breaking ties. So, in that you know 1 example could be we can use order of listing, so whichever is listed first will rank that particular element first depending on whatever it is ascending or descending you know ranking. So, let us take this example series17.rank and we are changing this, we are calling this keyword argument method and we are indicating first.

So, first come first serve thing and ranking would be you know given as per that. So, if I run this now if you look at the output 390, then the row index0 it is now ranked 6 and row index2 it is now ranked 7 and both had the same value. So, because over 1row index0 came first, so it was ranked first, so it got 6 and row index2 came after that. So, despite the same value it got the it was ranked later got 7.

(Refer Slide Time: 40:12)



jupyter Session Text mining modeling using NLTK Last Checkpoint 16/04/2019 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

```
# Annotate text tokens with POS tags
def pos_tag_text(text):

    # convert Penn treebank tag to wordnet tag
    # since the WordNetLemmatizer checks for POS tag annotations based on WordNet formats
    def penn_to_wn_tags(pos_tag):
        if pos_tag.startswith('J'):
            return wn.ADJ
        elif pos_tag.startswith('V'):
            return wn.VERB
        elif pos_tag.startswith('N'):
            return wn.NOUN
        elif pos_tag.startswith('R'):
            return wn.ADV
        else:
            return None

    # tokenize text into tokens using tokenize_text function
    tokens = tokenize_text(text)

    # using pos_tag function to annotate POS tags
    tokens_t = nltk.pos_tag(tokens, tagset='universal')

    # convert the POS tags from the Penn treebank syntax to WordNet syntax
    tokens_p = [(word.lower(), penn_to_wn_tags(pos_tag)) for word, pos_tag in tokens_t]

    return tokens_p
```

So now we can as I said we can specify we can use the ascending keyword argument and specify the order of ranking and in this example we are making false. So, now we will have the you know descending you know ranking based on the you know values, so if I call this here, so you can see that. Now you can see that rank index0 it is given the rank1 because it had the highest value, so you can see how the output has changed, this is our df7.

So you can always compare. Now we can also perform the ranking you know, so ranking is df7 if we use a data frame like df7. And so how the ranking can be done on a data frame, so we do not specify anything in the arguments part, so we just call df7.rank. So, the ranking would be perform along the column and by comparing row elements. So, if I run df7.rank, so you would be able to see the output that the highest value was corresponding to row index1 and that has been ranked last 4th.

So, because the default order is ascending I can specify the change the axis, so if you want to compare along the row by comparing column elements, so we can change axis=columns. And you can see that along the row index0, rank 2 and 1 have been given based on the values, so you can see that. Now sometimes what might happen that our axes might have duplicate labels they might not have unique axes levels. So, many pandas function they actually require unique axes labels for you know for the processing but not all of them. So, we will talk about few examples, so let us take this series7 object here and how do we check uniqueness.

We can call this is_unique property here this is property of index, so we can call series7.index.is_unique and this will tell us whether the index that we have is unique or not, so that is how to check the uniqueness. Now if you want to axes elements with non unique indexes were non unique and indexes are there, then see how the you know output is displayed. So, let us say series7 let us axes alpha, so just 1 you know index is there with this value, alpha value.

So, if I run this, so a scalar value is written, just 1 value 1.3, now if I axis you know beta index and there are 2 values with beta index. So, therefore you can see that if I run the next line, for example a series has been written because there were 2 values. So, the written what is being returned if we are dealing with non unique indexes. So, the written value can become complex in a sense in 1 scenario we are getting a scalar value, now this we are getting serious object you know comprising all the instances.

So, it might create slight complexity in our you know code you know for the particular problem. So, now let us talk about the same thing for a data frame, so let us take this df8, this is the data frame and if we take df8.loc and try to axes this index b row index b, you can see row index is coming twice, row index the value b is coming twice. So, if I try to axes you know one of them, so you can see the return what is being return is a data frame.

Now if I try to axes a particular row which is unique and if I run this you can see a series return. So, in one case the data frame is being returned another case series is return, so the return you know dirt I data type is changing, so which will of course complicate the coding aspect. Now let us talk about the some of the descriptive statistics that we can perform on these you know data structures data frame and series. So, let us take the example of this df9 data frame here and this is df9, now if we want to perform column sum we can just call df9.sum like before.

So, this is df9 this is df9.sum. So, we will have column sum you can see that nan values have been excluded, so this is the default behavior of this method. Now we can also we can pass on the axes argument as 1 and we can compute the row sums here. If we want to change the default behavior where nan values are excluded, so for that we can use this skip na argument keyword argument and we can indicate it false. So, in that case nan you know values will would not be excluded and so you know the NaN operation is going to involve nan will get the output as nan.

So, if I call df9.sum axis column is skip na false, so you can see the output there are 2 nan values. Similarly let us take this column means df9.mean, so you can see for column 1 and 2 we have got the mean values. So, this is another statistics useful statistics, now sometime we might want column and row indexes of the maximum values that is there. So, for that we can call this idx max you know method, so df9.idxmax.

And we will get you can see for column 1 and 2 we will get this the map the you know index where the values are maximum. Now let us have a look at df9, so you can confirm the same that bindex along column 1 that is higher value 7.1. Similarly for column 2 the d index we have-1.3 which is higher value similarly we have the idxmin, so we will get the index where the value is minimum.

So, if I run this you can see for the column 1 and 2, the index where the value is minimum is return. We can also compute column cumulative sums, so for that we can call this cumsum method and df9.cumsum and we will get the output, you can see along the column side, the values are being summed up. Now there is another method called describe, so we can call df9.describe and multiple statistics are going to be generated using this particular method.

So, if I run this here you can see count, mean, standard deviation, min value 25 percentile, 50, 75. So, all these quantiles and max value all these you know is multiple statistics have been printed here. We can also use the you know describe method for non numeric data, so let us say we have the series of consists of non numeric data and we are also applying the sky method here, in the output we are getting count unique elements you know top and frequency, so all those things for non numeric data we are getting here.

So, for you know this series object we are just we can you know replicate sum of the repeat sum of the elements here using this kind of expression here, you can see this output. Now let us move to another important statistics correlation covariance, so typically for a pair of variables are involved. So, before we go and talk about these methods element methods to compute correlation covariance what we will do, we will try to obtain data from certain you know online sources.

So, for this we are taking this example of obtaining stock prices and volumes data from yahoo finance. So, for this we require this pandas_data reader module, so we you can install this is the additional model not available in the anaconda in installation. So, we will have to install it additionally, so you can either use anaconda navigator to find out the you know non installed you know modules and install them or you can use the anaconda prompt and type this you know this code conda install conda install pandas you know dash data reader.

So, you look at this and then you can import this module here you can see we have imported. Now what will do, we will download the ticker label data from yahoo finance data you know database, online database. And what we are doing is here we are calling this function web.get_data_yahoo and ticker label for these companies as you can see and if I run this and then further we can create a data frame from this dataset.

So, you can see just like in the R platform we use to import data from the excel files here we are demonstrating the same thing where we are connecting with the web resource and you know downloaded the data set and then creating a data frame. So, here you can see we are you know picking a particular you know particular data here and running this. So, we will get the price, so this is the data frame for those companies and this is a time series data, so index is time actually.

So, we can have a look at the first 5 values you can see here similarly for volume also similarly we can obtain this first 5 values here. We can compute the percentage changes, so returns and let us have a look at the last values, so you can see because we are looking at the last value, so the data is you know date is changed with the current date here. And then we can compute the correlation, so whatever data frame we have just now created we can use the corr method here.

So, let us say 2 column MSFT and IBM, so in this fashion we can call this data frame select column, then call this.corr method and pass on the argument, pass on the you know particular column against which we would like to compute the correlation. So, 2 columns are required, so it will get the correlation value similarly another example for covariance here again. So, in this fashion from a data frame you can easily compute correlation covariances.

We can also pass on column name as attribute, so instead of saying returns and within brackets and indicating the column index we can say use that column name as an attribute returns.MSFT and.corr. So, in this fashion also we can compute, you can also compute correlation matrix, so among all combinations, so we have to just call returns data frame and.corr the method and for all combinations will get the correlation value.

Similarly for we can also compute the covariance matrix, so returns.cov, so we will get the covariance matrix here. Now if I want to compute pair wise correlation with 1 particular column, so for this we have this another method corrwith, so returns.corrwith and we will have to specify. So, what will happen this column the correlation of this particular column which is being pass an argument the IBM column.

And it is correlation with other you know columns that are present in the data frame that will be computed and displayed in the output. So, if I run this you can see in the output, so it is quite similar to you know first 1 particular column of the you know matrix that we are generated, so the corrwith output is similar to that. Similarly we can also compute correlation between columns of different you know data frames having same column name.

So, 1 data frame having these same you know particular column name the another data frame also having the same column names. So, we want to compute the correlation between a particular column with particular name in a data frame and a particular column with the same name in another data frame. So, we can compute the correlation, so for that again we can use this corr with function where we are using returns.corrwith and passing on the second data frame as an argument here and we will get the output.

Let us move forward, now let us talk about the unique values, so in the series again like we have discuss before we can use unique method and we will be able to find out these unique values, we can sort them also. And for certain statistics we would be require to compute the you know value frequencies here, so for that we have this value_counts function counts method. So, for any series we can call this method and we will get the counts for you know each of those you know values.

Now let us move forward, now there is another you know pandas function value_counts that can also achieve the same thing. So, we can call pd.value_count and pass on the arguments here as you can see and this will also achieve the same output. Now there is some times we might be required to filter presence or absence of an element. So, for that we can use isin method, so we can pass on the argument a particular element and we can find out whether that is present or not.

So, let us say series18 and we will get a Boolean output here and you can see wherever that particular element is present it comes out that true is indicated and otherwise false is indicated. So, we can do the same thing for multiple elements, so we can pass on that a list of elements, so next example you can see wherever one of those elements are present to indicated. Now we can use this Boolean output and pass on into the series or here and only those wherever true is indicated only those elements would be selected.

So, let us take this filter and select those values, now another example where sometimes you know this is for the data frame. So, sometime we would be require to compute frequencies and the columns would be related, so that also we can perform. So, let us take this is example of df10 data frame, so you can see we have you know these values here in this data frame. Now what we will do we will use the apply method and within the apply method we are passing another you know function that python pandas function pd.value under_count.

So, it would be applied on you know each of you know on each of those columns, so row label are going to indicate the distinct values of the data frames that are there. So, if we look at the output number forty 439 distinct values are 1, 2, 3, 4, 5. So, those are going to become output in the next code that we are going to run df10.apply and then passing on this function pd.value_counts.

And the values they are going to indicate the count of those distinct values in the respective column. If I run this and you can see in the output number 440, the row indices are the distinct values of the data frame and along the column we are getting the counts. So, one is present one times, 3 is present 2 times, 4 is present 2 times, 2 and 5 are not present. So, that is how **we** it is like a you know frequency that we have got here.

Now, so there are some of the na values, so that is essentially means that those elements, those distinct values are not present. So, we can fill those na values with zeros, so for that again we can use fill na method and pass on zeros and output would be modified. So, with this we have been able to cover pandas package.
(Video Ends: 41:18)

And in the next lecture we will start another aspects of the text mining analytics where we will be focusing on how to work with data import, export and process and transform. So, now we will more focus on those aspects and how we can perform the same thing in the python platform. So, we will talk about that will start that discussion and let us stop here, thank you.

Keywords: Linear Algebra, Dataframe, Data Structure, Text Mining Analytics, Python.