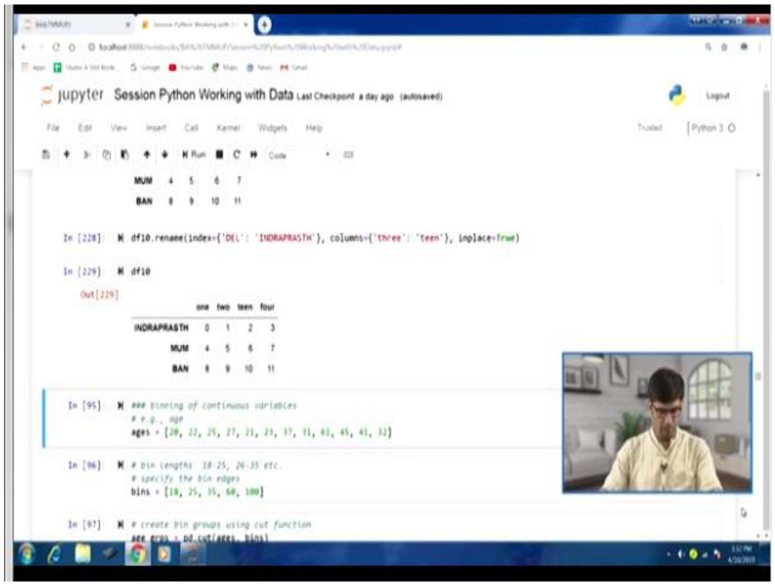
Business Analytics and Text Mining Modeling Using python

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

Lecture-32
Python Working with Data – Part III

Welcome to the course business analytics and text mining modeling using python. So, in previous few lectures we have been discussing the aspects of working with data. Specifically we stop at this point we want to discuss the meaning of continuous variables.

(Refer Slide Time: 00:37)



```
     MUM    4  5  6  7
     BAN    8  9 10 11

In [228]: df10.rename(index={'DEL': 'INDRAPRASTH'}, columns={'three': 'teen'}, inplace=True)

In [229]: df10
Out[229]:
             one  two  teen  four
INDRAPRASTH    0    1     2     3
        MUM    4    5     6     7
        BAN    8    9    10    11

In [95]:  ### binning of continuous variables
          # e.g., age
          ages = [20, 22, 25, 27, 21, 23, 37, 31, 61, 45, 41, 32]

In [96]:  # bin lengths  18-25, 26-35 etc.
          # specify the bin edges
          bins = [18, 25, 35, 60, 100]

In [97]:  # create bin groups using cut function
          age_grps = pd.cut(ages, bins)
```

So, with pickup on this point itself, so as you might understand that you know 2 types of variables continuous and categorical variables like we have discussed in our previous courses business analytics and data mining modeling using R. So, many times if we are require to perform binning of continuous variable that means we are require to transform continuous variable as a into a categorical variable.

So, for that we will have to create categories will have to create bins that means those category and those groups we also refer those groups as bins. So, we will have to you know perform all that. So, how do we you know complete that process and automate that process using in python. So, for that we have you know few examples here.

So, let us talk about this variable ages, so let us say we have these ages of you know certain points certain individuals, so this is list of you know these values.

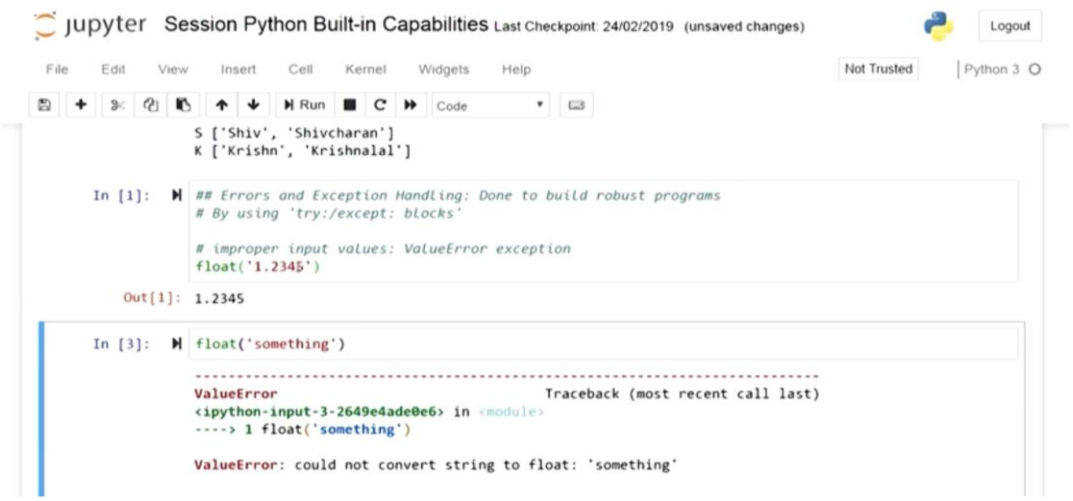
(Video Starts: 01:47)

So, let me create this variable here ages and let us say we want binning bin lengths of these bins of this interval let us say 18 to 25 then 26, 35 etc. so on. So, we can specify the bin s is here, so for example you know first bin is starting 18 and ending 25. So, we can note down these end points here and specify that in a variable let us say bins.

And these edges are going to be use to create you know these you know groups or categories or bins. So, let us define bins as well, so if you focus on the bins that we just refine 18, 25 then you know 35, so depending on the bin lengths that we actually wanted. So, based on those bin lengths we can define these bin edges and then these bin edges are going to be use to cut the variable, the continuous variable into appropriate bins.

So, once we do this ages variable can we call age, groups variable . So, for this to perform this process we use cut function, so this is a pandas is a function pd.cut. And first argument we are passing the variable ages the continuous variable which we want to cut where we want to perform binning. And then we also specify our bins that means bin edges which we would like to you know follow to actually perform the binning on the first argument you know the variable ages here.

(Refer Slide Time: 02:28)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
S ['Shiv', 'Shivcharan']
K ['Krishn', 'Krishnalal']
```

In [1]:
```
## Errors and Exception Handling: Done to build robust programs
# By using 'try:/except: blocks'

# improper input values: ValueError exception
float('1.2345')
```

Out[1]: 1.2345

In [3]:
```
float('something')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-2649e4ade0e6> in <module>
----> 1 float('something')

ValueError: could not convert string to float: 'something'
```

So, if I run this will end up with will have the binning here, so if you look at this the ages variable that we had you know 12 values there and in the output 2, 3, 2. If you look at the first value which was actually 20 has now been replaced with it is a bin which is starting from 18 to 25, so that group. And similarly 22 is also falling in the same groups, so the same you know bin there similarly for 25.

For 27 this will change, so you can see the change in the category 25 to 35. Then again 21 in the first group 18 to 25, so in this fashion all the 12 values they have been put into their appropriate bins as per our specified bin edges as per our you know bin lengths. So, finally we are end up with 4 intervals, so when we specified 5 bin edges, so essentially will have you know you know 4 groups there.

So, you can see those categories are also specified and 18, 25 and 25 to 35, 35 to 60, 60 to 100, so if you look at this is underscore groups. So, this has become a categorical variable rather and ordinal variable because as you can see in the category the order is important 1 group is lower than the you know other 1 category can be considered as lower in comparison to other categories.

So, this is the ordinal variable the ages has been we have been convert age into an ordinal age_groups variable. So, in the analytics sometimes we are require to perform this kind of data transformation. Now if you want to have a look at a nicer presentation of you know age_groups. So, for that we have course attribute, so in this course attribute instead of mentioning the bin interval we can have the numeric course to represent those you know those bins.

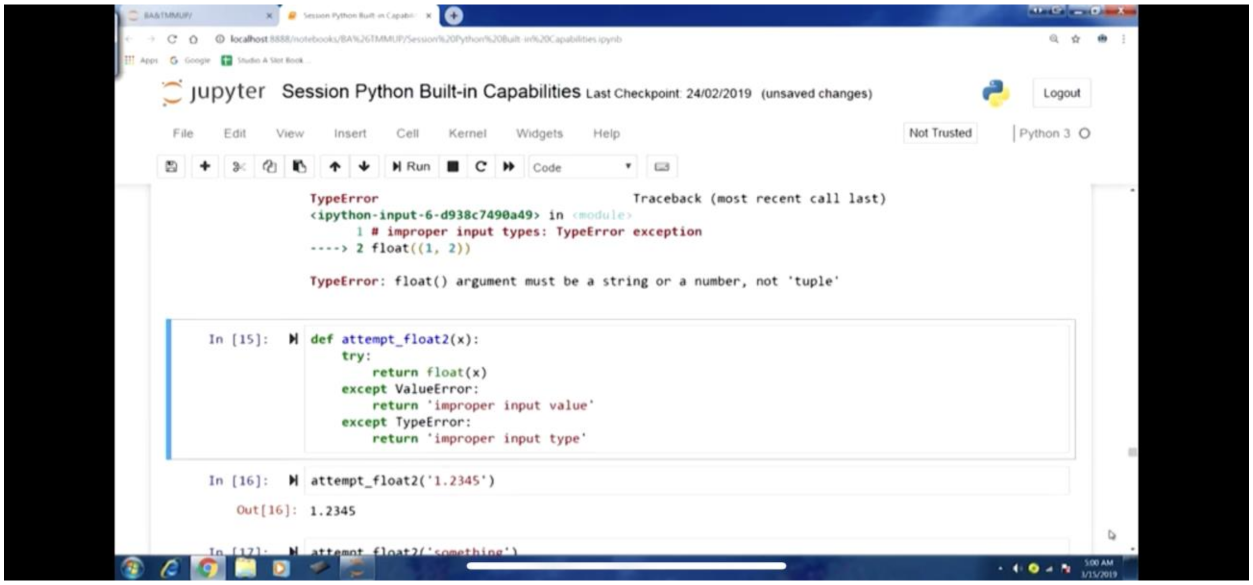
So, instead of 18, 25, 25 35, 35 60, 60 100, we can have 0, 1, 2, 3 to represent those intervals there, those categories. So, if I run age_groups.codes will have you know 12 values with these numeric codes in this form. Similarly we also have another attribute categories which will again just inform us about the intervals that we have. So, if I run this we will have all the categories of the data there in age_groups as you can see.

Now we can also compute frequencies for different categories, different bins of age groups. So, for that we can call this pandas function value_counts and we can pass on this variable age_groups. So, if I run this we will have in the as you can see in output 2, 3, 5 for each of the interval for each of the you know bins or categories that we have created we have the frequency numbers.

So, 18 to 25 5 instances of this particular category, second one 35 to 60 you know 3 instances 25 to 35 3 instances 6 to 100 1 instances. So, in this fashion we can compute the frequencies , so this is similar you know this data is similar to what we typically observed in a histogram , so this is how we can actually perform this. Now if we look at the intervals the way they were created the bin intervals the open close sides.

If you want to change that we would like to you know if you look at the default thing the starting side is the you know open you have parenthesis that means open side and the closing side is brackets that means you know close side. So, that means inclusive side also, so if you want to change this you know interval you know sides uh.

(Refer Slide Time: 07:28)



```
TypeError                                 Traceback (most recent call last)
<ipython-input-6-d938c7490a49> in <module>
      1 # improper input types: TypeError exception
----> 2 float((1, 2))

TypeError: float() argument must be a string or a number, not 'tuple'
```

```
In [15]:  def attempt_float2(x):
              try:
                  return float(x)
              except ValueError:
                  return 'improper input value'
              except TypeError:
                  return 'improper input type'

In [16]:  attempt_float2('1.2345')

Out[16]:  1.2345

In [17]:  attempt_float2('something')
```

And how whether they are going to be open or close, so that can also be done, so by default left side is open and right side is closed. So, we can use right argument in this cut function to specify this, so the next line of code you can see that pd.cut we are going to do cut. Now in this case we are using ages and those bin bins also been edges also and right argument is indicated as false.

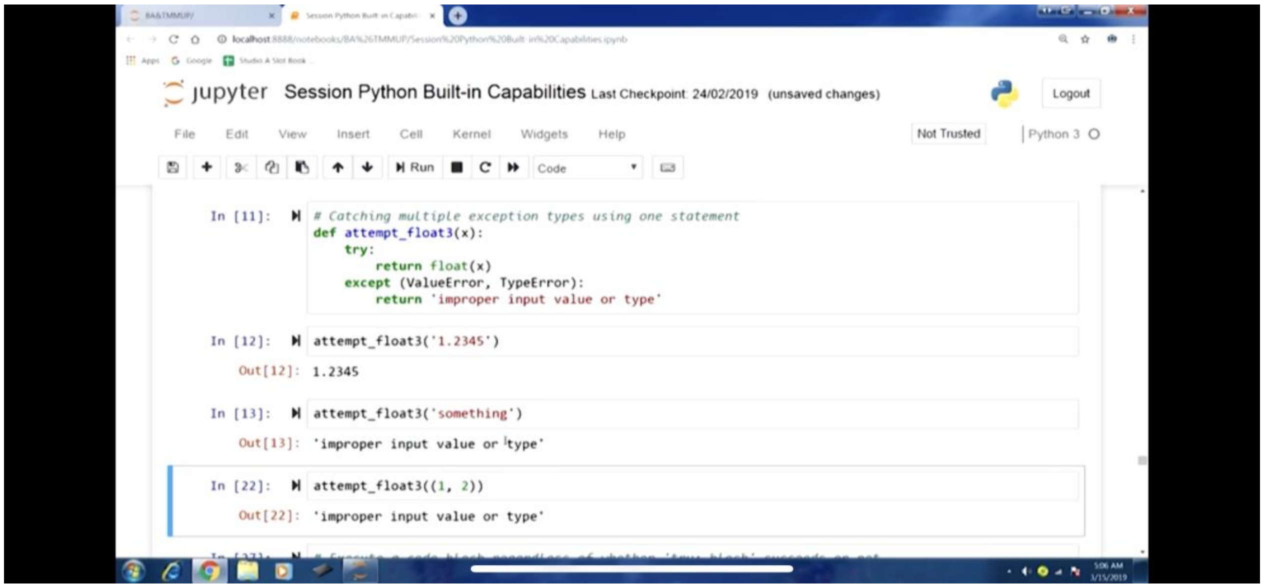
So, therefore now right will be close site here and the other side would be reversed. So, if I run this you can see the bin intervals the way they have been defines that is actually has been changed. We can also change bin names, so instead of using these you know values to indicate the interval we can have slightly qualitative terms to actually you know define these intervals.

So, for example 18 to 26 we can call this age group as youth 26 to 36 we can call this age group as young adult 36 to 61 we can call this age group as middle aged and then 61 to 100 we can call this senior group. So, this is the qualitative term for defining these you know these intervals which are right now using numeric values. So, for this we can use the labels argument in the cut function, so whenever we are creating these bins using pd.cut we can use this labels argument and specify this.

So, if I run this one, so you can see the all the 12 values that we havd in the ages variable . Now it has been converted into let us say age_groups kind of variable and the values now the qualitative terms the linguistics terms are being used here youth youth youth for those values young adults and in this fashion. Now if you look at the categories also, so these labels are being used, now instead of the intervals with the numeric values.

So, youth less than young adult less than middle age, less than senior, so in this fashion we would be able to create the binning and rename them using linguistic or qualitative terms as well. Now we can also create if we just if we do not want to specify the bin length or bin edges we can just specify the number of bins that we want. So, those if we just specify the number of bin, so equal length bins are going to be created.

(Refer Slide Time: 13:00)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
In [11]:  # Catching multiple exception types using one statement
          def attempt_float3(x):
              try:
                  return float(x)
              except (ValueError, TypeError):
                  return 'improper input value or type'

In [12]:  attempt_float3('1.2345')
Out[12]:  1.2345

In [13]:  attempt_float3('something')
Out[13]:  'improper input value or type'

In [22]:  attempt_float3((1, 2))
Out[22]:  'improper input value or type'
```

So, this can also be achieved so using same cut function so let us take a example here. So, now we will use var2 will take random values following uniform distribution, so var2 and what we will do we will create 4 bins. So, in the pd.cut function will pass that value 4 to indicate that we want 4 bin, so it will create you know equal length bins for us. And that argument if you look at precision that is just to indicate the decimal points that we want in our output.

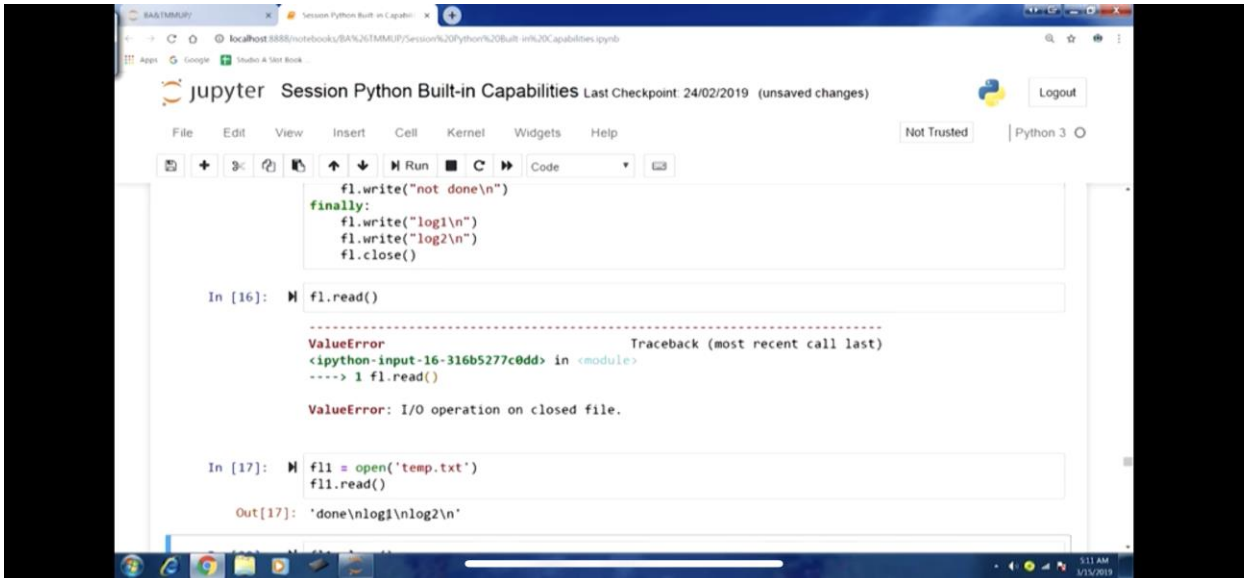
So, if I run this will have you know different bins and these are equal sized bins, so you can have a look at this 0.18 to 0.38. If you look at the categories, so we have 4 you know categories here then 0.38 to 0.58, 0.58 to 0.7, 0.78 to 0.98. So, you can look at the length of these bin intervals that is equal here and appropriately the age values of the age variable they have been you know classified with one of these categories.

Now the bin lengths can also be based on sample quantiles so there also even if we apply sample contrast we are going to you know typically will or roughly we get equal length bins. So, for that we have another function qcut function which is going to use this quantas. So, let us for take this example for this variable that we had just created var2. So, var_groups so this is the you know categorical variable we can create using pd.qcut function.

First argument we are specifying the variable and then second one the number of groupings that we want, number of bins that we want. So, let me run this and you can see that if you look at the categories and these specific values side change from the previous output 39, you can compare the categories into 39 and 240 there we are 0.18 to 0.38. Now we have 0.182 to 0.331 and similarly for others also. So, you can see here, so slight of course you know slight changes in terms of you know bin length.

However if you know look at the frequencies for bins here so that is the advantage here using the sample quantiles here. So, if I run pd.value_counts here, so you can look at the frequencies. So, you can see same innings of these groups that are based on sample quantiles you can see 5 values in each of these groups. Now we can also specify our own quantile, so we can use the same function pd.qcut.

(Refer Slide Time: 17:47)



jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)

```
        fl.write("not done\n")
    finally:
        fl.write("log1\n")
        fl.write("log2\n")
        fl.close()

In [16]:  fl.read()

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-316b5277c0dd> in <module>
----> 1 fl.read()

ValueError: I/O operation on closed file.

In [17]:  fl1 = open('temp.txt')
          fl1.read()

Out[17]: 'done\nlog1\nlog2\n'
```

And in the second argument instead of asking for a number of you know bins. For example in 2, 4, 0 base specified we wanted 4 bins using sample quantile. We can specify the quantiles itself using a list, so you can see 0.25, 0.5, 0.75 and 1, so 5 values we have been specified. So, essentially we are looking for 4 bins using these sample quantiles, so if I run this and you can have a look at the output you can also compare the output 2, 4, 0 and 2, 4, 1.

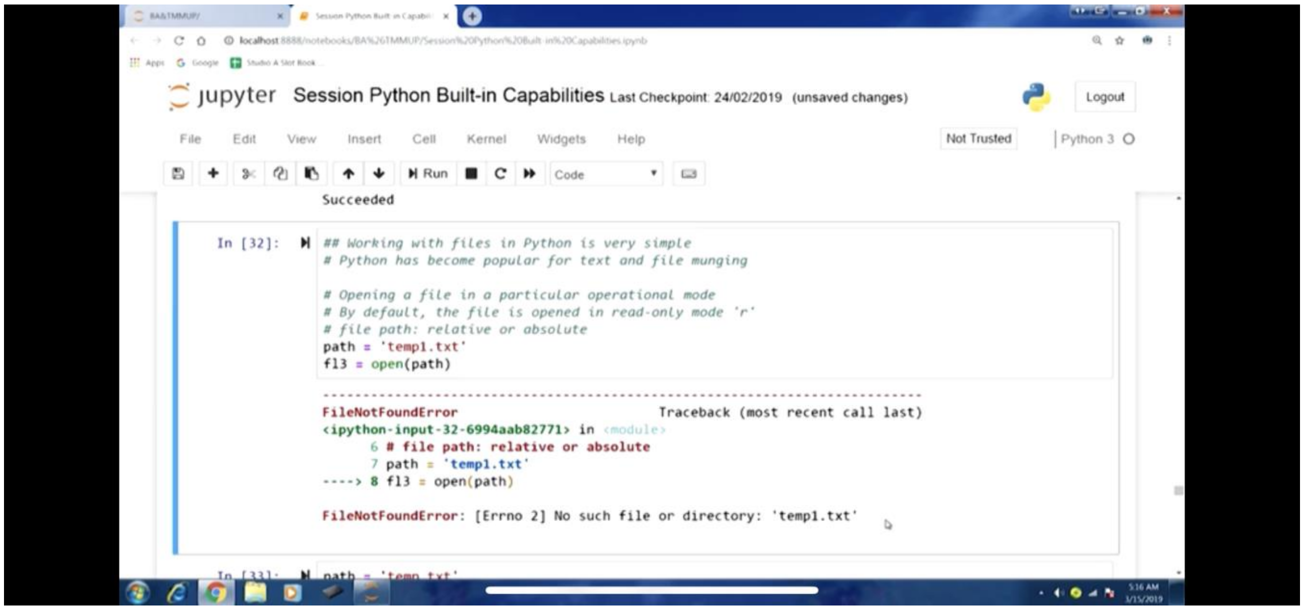
If you look at both the outputs are same, so essentially when we called when we ask for 4 bins using the pd.qcut function and did not specify the actual quantile it was taking these quantiles itself these 0 to 0.25, 0.5, 0.75. So, these quantiles are by default taken if we do not specify the quantiles in the qcut function, so if you want different quantiles you can always go ahead and specify the same.

So, this was about binning of continuous variables and different variations different approaches that we can take to perform these binnings and transformation that we talked about. Now let us move to the next aspect and now this about finding outliers, so outlier you know detection and replacement is an important task in the analytics domain. So, to demonstrate this let us take this example here in this case we have 1000 cross 4 you know array with these dimensions. So, let me define this I can so typically when we are looking when we are analyzing outliers.

Typically we will like to have a look at the descriptive statistics, so in this case for this data frame df11 will like to call this with this method describe and we will have a number of you know multiple statistics here. So, you can have look here count, mean, standard deviation mean 25 percentile, 50%, 75% in max. So, you can have a look at these multiple statistics and these statistics in a way can indicate the presence of any outliers there.

So, higher you know higher max value and a lower mean might indicate that there are outliers. So, for example if we have a look at the mean values at different columns, so they are in the range of let us say 0.49, 0.003 and if you look at the you know maximum value in these columns, so those values are going in the range of you know 2.98, 2.74, 3.05 and 3.45. Similarly mean values also they are going in the negative direction -3.57, -3 all of them are you know in the absolute sense they are above 3.

(Refer Slide Time: 22:44)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
Succeeded

In [32]:  ## Working with files in Python is very simple
          # Python has become popular for text and file munging

          # Opening a file in a particular operational mode
          # By default, the file is opened in read-only mode 'r'
          # file path: relative or absolute
          path = 'temp1.txt'
          f13 = open(path)

          ---------------------------------------------------------------------------
          FileNotFoundError                         Traceback (most recent call last)
          <ipython-input-32-6994aab82771> in <module>
                6 # file path: relative or absolute
                7 path = 'temp1.txt'
          ----> 8 f13 = open(path)

          FileNotFoundError: [Errno 2] No such file or directory: 'temp1.txt'

In [33]:  path = 'temp.txt'
```

So, you can see that mean is quite low and mean and maximum value are on the higher side, so there are chances that outliers are present in this particular data set. So, how do we find out you know these values which are exceeding let us say 3 or -3. So, let us find out values which are exceeding 3 or -3 for a particular column, so as you can look at the mean statistics for all the columns there are values which are exceeding you know which are actually exceeding 3 in the absolute sense.

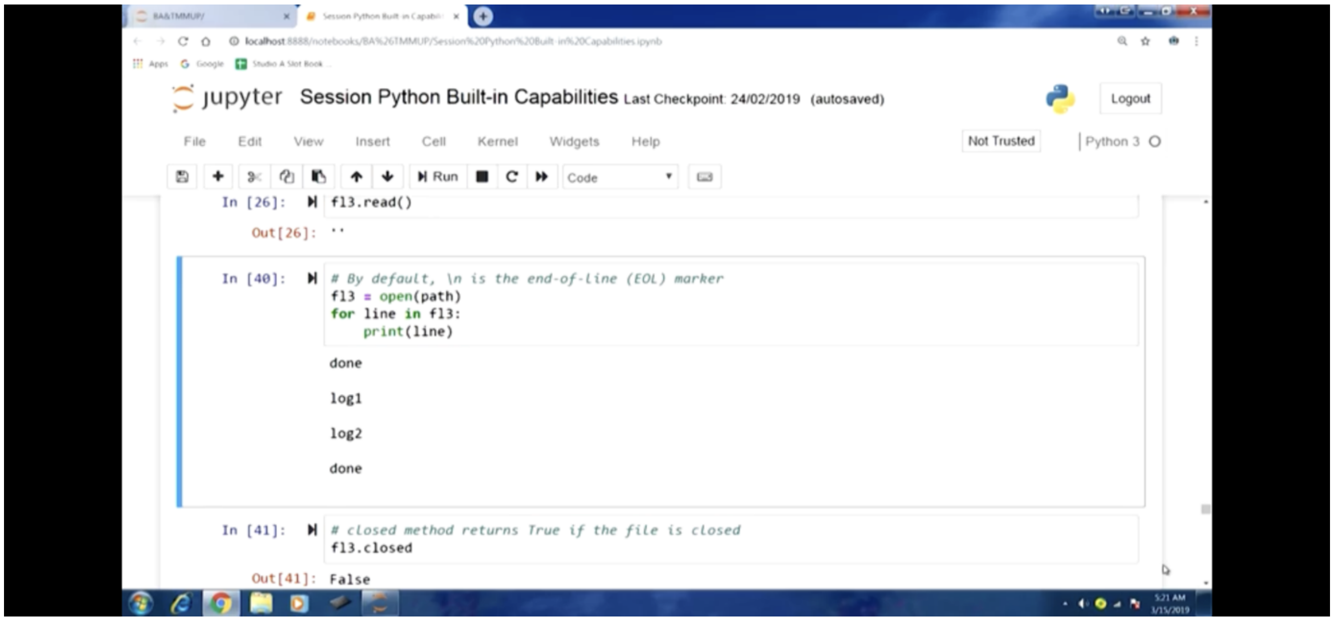
So, let us pick the column with the index 2 column 2 that is third column, so what we will do df11 and will specify the index to pick that column. And within the brackets now we are taking an absolute of this particular columns values and then comparing it with 3. So, for any values where the values greater than 3, so those rows are going to be defined as true and those would be selected.

So, if I run this you can have a look at the output 2, 4, 5, so we have 2 rows, 2 9 8 and 7 51 with row index 2 9 8 and 7 5 1 and both of them are form the column 2 and having values you know more than their absolute value is more than 3 you can see -3.06 and 3.05. So, in this fashion we can filter out these outliers, now we can apply the same thing for all the columns. So, you can see here df11 absolute value.any one.

So whereas we are specifying the x is and we can done it here and you can see that. So, you can find out here again from the output, so all the rows where we have at least some value with greater than 3. So, exceeding 3 or -3, so you can see so all these row indices are there in the output as you can see here. So, total we have 13s as rows where this is happening in 1 or more columns. Now let us move to you know another aspect here.

So, one thing is to find out these outlier and then sometimes you would like to change the values also depending on the our you know modeling requirement. So, for a given range we can modify value, so let us say if there are any values which are outside the range of -3 to 3. So, we would like to replace them with you know all those values as either -3 or 3 depending on the sign. So, we can record the sign using the sign function and then we can replace the value with either 3 or as we can replace the value with 3./

(Refer Slide Time: 27:44)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
In [26]:  f13.read()
Out[26]:  ''

In [40]:  # By default, \n is the end-of-line (EOL) marker
          f13 = open(path)
          for line in f13:
              print(line)

          done

          log1

          log2

          done

In [41]:  # closed method returns True if the file is closed
          f13.closed
Out[41]:  False
```

And sign will record using np.sign function here. So, the next line here we are actually performing that thing. So, we have first in the you know identifying in the left hand side all the rows like the previous output where the values are greater than 3 . And then you can look at right hand side we are recording the sign for those and then multiplying it with 3 value. So, if I run this and let us have a look at the describe method output here again.

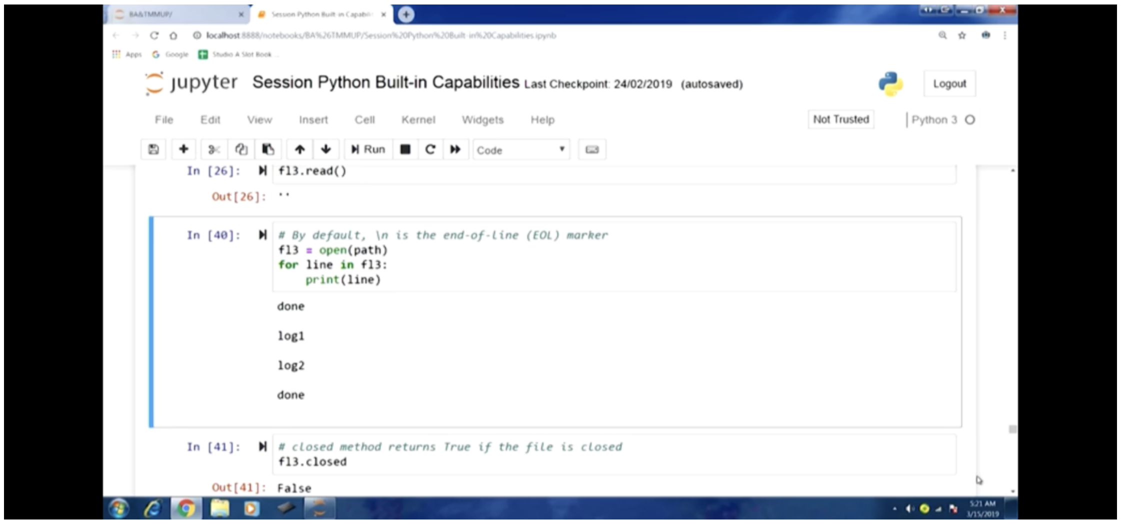
So, you can see that minimum value, minimum statistics for all the columns and maximum statistics for all the column, now it is within -3 and 3. So, in this fashion what we have actually done we have replace all the outlier values , values which were exceeding 3 or -3 with actually those numbers -3 and 3 itself. Now let us move to another aspect which is about random sampling or random reordering of rows, so this is also very important.

Because we typically will be dealing with larger data sets and therefore we would be drawing a sample points from there. So, typically we do random sampling so how do we perform this in the python platform. So, let us take the example of this data frame, so this data frame we have 5000 rows and 4 columns. So, let me first create this one, so you can have a look 5000 rows, 4 columns here.

Now first thing that we will like to demonstrate is randomly sampling and rows without replacement. So, this is the more common exercise more typical exercise in modeling context where we randomly draw sample points, randomly draw cases so let us say n rows of without replacement. So, for this we can use sample method and let us say we would like to draw 500 rows out of 5000 rows that we have in our data set.

So, for this we can apply this we can call this method sample df12.sample and we can specify in the keyword argument and the number of points that you would like to sample 500 in this case. So, if I run and you can have a look at the output 250, so if you focus on the row indices here these values have been randomly indices have been randomly drawn from 5000 rows that we have.

(Refer Slide Time: 34:23)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
In [26]:  f13.read()

Out[26]:  ''

In [40]:  # By default, \n is the end-of-line (EOL) marker
          f13 = open(path)
          for line in f13:
              print(line)

          done

          log1

          log2

          done

In [41]:  # closed method returns True if the file is closed
          f13.closed

Out[41]:  False
```

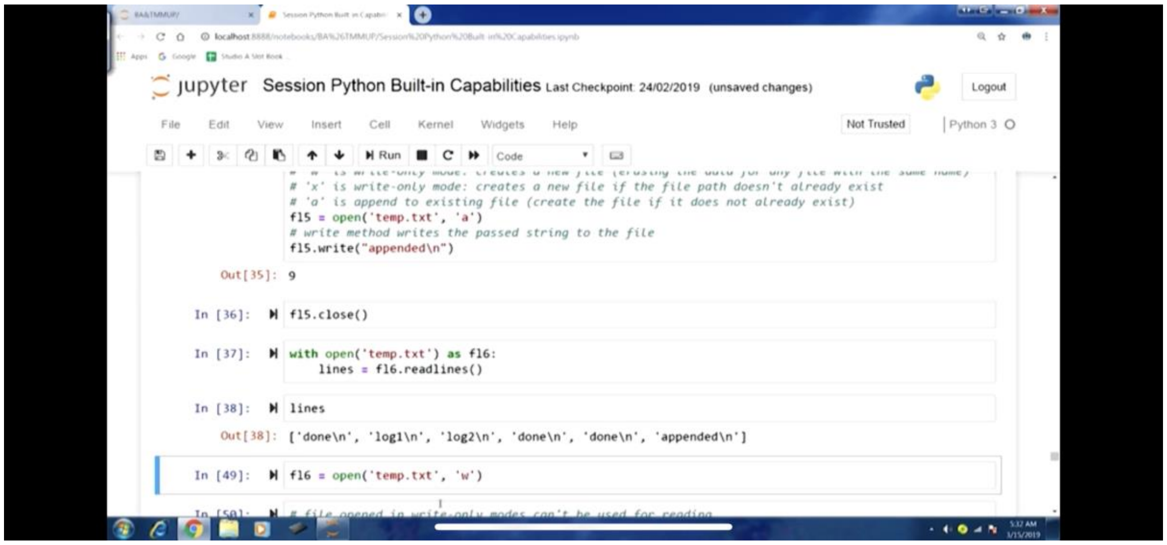
So, you can see the random index values are starting like 1180, 3329, 2319, 3968, so all these seem to be quite random. And therefore the sample method is able to provide us you know those you know rows. Now sometimes we would like to you know we would like to draw these cases with replacement. So, for that we have this replacement argument, so we can specify replace as true, so again same method df12.sample n=500 and replace true.

So, in this fashion if I call this method you would see the output is going to be based on that process. So, again you can see because this is also a random sampling here different row index values we can see in the output as you can see in 2, 5, 1. Now let us move forward to the next aspect which is about random reordering of row indicer, so if you want to certain situation you would like to perform this also.

So, for that we have this permutation function , so to demonstrate this we will take this define this data frame df13 we have 5 rows and 4 columns in this data frame. And what we can do is we can if you want to perform a random permutation we can call this np.random.permutation and we can specify the number of rows there. So, we have 5 rows, so if I run this will have the randomly drawn reorder for these row indicers.

So, if you look at the output 2 5 3, now 3 is coming first then 4 then 0 then 1 and 2 . Now I can use the reindex method here to actually use this random permutation here, so I can call df13.reindex and I can pass on this random permutation that we have just drawn and I can run this and you can see. Now you can focus on the output 2 5 4 and specifically the row indices, so you can see 3, 4, 0, 1, 2.
So, those random permutation of re random reordering of rows have actually been executed. Instead of reindex method we can also use the take method here, so df13.take we can pass the permute and will get the same output. So, let move forward now next thing is about dummy variables. So, as you would understand that in the analytics context we would be dealing with categorical variables.

(Refer Slide Time: 37:27)


jupyter Session Python Built-in Capabilities Last Checkpoint: 24/02/2019 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

```
# 'x' is write-only mode: creates a new file if the file path doesn't already exist
# 'a' is append to existing file (create the file if it does not already exist)
f15 = open('temp.txt', 'a')
# write method writes the passed string to the file
f15.write("appended\n")
Out[35]: 9

In [36]:  f15.close()

In [37]:  with open('temp.txt') as f16:
              lines = f16.readlines()

In [38]:  lines
Out[38]: ['done\n', 'log1\n', 'log2\n', 'done\n', 'done\n', 'appended\n']

In [49]:  f16 = open('temp.txt', 'w')
```

And typically in most of the software, most of the implementations of algorithm whether statistical or data mining. Typically, categorical variable they are converting into dummy variable and so how do we perform this process. So, this so dummy variable this is about converting as you can see in the comments converting categorical variables into dummy variables. So, what are dummy variables they are going to be filled with 1s and 0s.

So once we indicate the presence of that category in a row observation or case. And zeros will indicate the absence of that category in a row or case in a particular observation. So, given categorical variable and it might be having values like you know if there are you know 4 categories. So, those categories we might have numeric codes like 0, 1, 2, 3. So therefore these at particular column for that categorical variable might be filled with these values 0, 1, 2, 3 you know for different observation different value.

Now we would like to convert such a column into dummy variables where for each category will have a you know separate different dummy variable. And in each of those dummy variable the values would be either 0 or 1 one were indicate the presence of that particular category for that row and 0 will indicate the absence. So, that is the kind of transformation that we are require to perform in dummy coding.

So, let us take a categorical variable with k distinct values that means category corresponding to categories. So, we will take this data frame here, so if you look at here we have this 1 column with variable and it has you know 3 distinct values a, b, c and we have 6 rows. So, we can use get_dummies function which will actually perform this dummy coding for us. So 3 distinct value as I talked about a, b, c.

So, I can call pd.get_dummy I can pass on df14 and the column name var here. So, for this column will have be dummy variables here, so you can see now in the output 2, 5, 7 3 columns with the name a, b, c and you can see only zeros and ones. You can compare this particular output with the previous output with the variable column. So, you can see for b for column a in the row when we have b so that means for that row b category is there, so that means a is absence, so that is why 0 and 0.

So, or in other way you can map it like this for column a wherever 1 is present that means in the column variable a would be indicated. So, you can check it back with the var column, so you will get the idea about what we are trying to do here. Now while we are creating these dummy variables we would like to sometimes we would like to rename them as well. And in this renaming you know most times more often the typical practices that we would like to prefix these dummy variables with the variables name.

So, for that we have this prefix argument in the get_dummies function, so we can use that, so pd.get_dummies df14 this variable and we can prefix var here. So, if I run this you can see in the output 2, 5, 8 and you can focus on the column names here. So, they have been changed from a to var_ab2 var_b and c has been change to var_c. now we can combine dummy creation and binning process.

So binning is about you know converting a continuous variable into categorical variable. But eventually when you convert it into a categorical variable you will also be doing dummy coding. So, why not combine these 2 processes, so we can use of combination of get_dummies and cut function and perform this. So, let us take this variable 3 and these are the values and let us take this binning for this you know variable these bin edges.

And if we perform the binning on this variable 3 using this bins let us have a look at the output. So, you can see pd.cut upto 262 you can see how this binning has work, now let us combine. So, if you look at how we are combining this, so we are calling pd.get_dummies function. Because this is the later process, so within this we are passing the pd.cut because this will give us the categorical variable.

So, essentially categorical variable is input for the get_dummies you know variable to produce the dummies. So, the pd.cut output of that particular function is going to be the argument for get_dummies in this case. So, if I run in this fashion you can see the column that we have just got in the output that you know you can see focus on the column names 0000.2, 0.2 to 4 0.4. So, you can see all those you know bin edges that we have they have bin intervals have been created.

And those are being used as the column names. And dummy coding has also happen for each of these column names we have we are filling the zeros and ones. So, indicating the presence sort of sense, so in this fashion we can in one go we would be able to convert the categorical variable into continuous variable to categorical variable and also transform the categorical variable into multiple dummy variables. **(Video starts: 31:22)**

So, we will stop here and in the next lecture will start about discussion on another important aspects especially in the context of this particular course that is string and text processing, so let us top here, thank you.

Keywords: Permutations, Relative Error, quantitative methods, exceptional error, numerical and categorical values.