

1. 3333 records are present in the dataset.
2. 21 features are there in the dataset.

```

In [6]: print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   state                  3333 non-null   object  
1   account length         3333 non-null   int64   
2   area code              3333 non-null   int64   
3   phone number           3333 non-null   object  
4   international plan      3333 non-null   object  
5   voice mail plan        3333 non-null   object  
6   number vmail messages  3333 non-null   int64   
7   total day minutes       3333 non-null   float64  
8   total day calls         3333 non-null   int64   
9   total day charge        3333 non-null   float64  
10  total eve minutes       3333 non-null   float64  
11  total eve calls         3333 non-null   int64   
12  total eve charge        3333 non-null   float64  
13  total night minutes     3333 non-null   float64  
14  total night calls       3333 non-null   int64   
15  total night charge      3333 non-null   float64  
16  total intl minutes      3333 non-null   float64  
17  total intl calls        3333 non-null   int64   
18  total intl charge       3333 non-null   float64  
19  customer service calls  3333 non-null   int64   
20  churn                   3333 non-null   bool    
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 524.2+ KB
None

```

In the above picture, the datatypes with object and int64 are discrete; columns with Dtype float 64 are continuous, churn has bool Dtype which is binary.

3. Features such as State, Account length, area code, phone number are irrelevant because the churn status is not linked with any kind of pattern with these features.
4. There are no missing values in the data.
5. The continuous numeric features are total day minutes, total day charge, total eve minutes, total eve charge, total night minutes, total night charge, total int minutes and total int charge.

```

In [7]: df.describe()

Out[7]:

```

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	customer service calls	churn
count	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00	3333.00
mean	101.06	437.18	8.10	179.78	100.44	30.56	200.88	100.11	17.08	200.87	100.11	9.04	10.24	4.48	2.76	1.56	
std	39.82	42.37	13.69	54.47	20.07	9.26	50.71	19.92	4.31	50.57	19.57	2.28	2.79	2.46	0.75	1.32	
min	1.00	408.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	23.20	33.00	1.04	0.00	0.00	0.00	0.00	
25%	74.00	408.00	0.00	143.70	87.00	24.43	166.60	87.00	14.16	167.00	87.00	7.52	8.50	3.00	2.30	1.00	
50%	101.00	415.00	0.00	179.40	101.00	30.50	201.40	100.00	17.12	201.20	100.00	9.05	10.30	4.00	2.78	1.00	
75%	127.00	510.00	20.00	216.40	114.00	36.79	235.30	114.00	20.00	235.30	113.00	10.59	12.10	6.00	3.27	2.00	
max	243.00	510.00	51.00	350.00	165.00	59.64	363.70	170.00	30.91	305.00	175.00	17.77	20.00	20.00	5.40	9.00	

```

In [8]: df.describe(include=['object', 'bool'])

Out[8]:

```

	state	phone number	international plan	voice mail plan	churn
count	3333	3333	3333	3333	3333
unique	51	3333	2	2	2
top	WV	382-4657	no	no	False
freq	106	1	3010	2411	2850

Feature	average	median	maximum	minimum	Std dev
total day minutes	179.78	179.40	350.80	0.0	54.47
total day charge	30.56	30.5	59.64	0.0	9.26
total eve minutes	200.98	201.4	363.70	0.0	50.71
total eve charge	17.08	17.12	30.91	0.0	4.31
total night minutes	200.87	201.20	395.00	23.20	50.57
total night charge	9.04	9.05	17.77	1.04	2.28
total int minutes	10.24	10.30	20.00	0.0	2.79
total int charge	2.76	2.78	5.40	0.0	0.75

6. A customer makes 1.56 calls on an average to the company
7. There are total 51 states in the data

The screenshot shows a Jupyter Notebook titled "Exploratory data analysis". The interface includes a top bar with the Jupyter logo, the title, and a "Last Checkpoint: 11 hours ago (unsaved changes)" message. Below the top bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar with icons for file operations and execution is also present. The notebook content consists of two cells. The first cell is a text cell containing a recap of data analysis concepts, mentioning a 85.5% share of loyal clients and a formula for churn prediction. The second cell is a code cell with the following Python code:

```
In [44]: lis=pd.unique(df['state'])
n=len(lis)
print(n)
print(lis)

51
['KS' 'OH' 'NJ' 'OK' 'AL' 'MA' 'MO' 'LA' 'WV' 'IN' 'RI' 'IA' 'MT' 'NY'
 'ID' 'VT' 'VA' 'TX' 'FL' 'CO' 'AZ' 'SC' 'NE' 'WY' 'HI' 'IL' 'NH' 'GA'
 'AK' 'MD' 'AR' 'WI' 'OR' 'MI' 'DE' 'UT' 'CA' 'MN' 'SD' 'NC' 'WA' 'NM'
 'NV' 'DC' 'KY' 'ME' 'MS' 'TN' 'PA' 'CT' 'ND']
```

8. The following is the distribution of the “Churn” and it is skewed because True count is very less compared to False count.

localhost:8888/notebooks/Data%20Mining%20Project-1/Experiment-1/Exploratory%20data%20analysis.ipynb

jupyter Exploratory data analysis Last Checkpoint: 4 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

In order to see statistics on non-numerical features, one has to explicitly indicate data types of interest in the `include` parameter.

```
In [8]: df.describe(include=['object', 'bool'])
```

```
Out[8]:
```

	state	phone number	international plan	voice mail plan	churn
count	3333	3333	3333	3333	3333
unique	51	3333	2	2	2
top	WV	382-4657	no	no	False
freq	106	1	3010	2411	2850

For categorical (type `object`) and boolean (type `bool`) features we can use the `value_counts` method. Let's have a look at the distribution of `churn`:

```
In [9]: df['churn'].value_counts()
```

```
Out[9]: False    2850
        True     483
        Name: churn, dtype: int64
```

2850 users out of 3333 are *loyal*; their `Churn` value is `0`. To calculate fractions, pass `normalize=True` to the `value_counts` function.

```
In [10]: df['churn'].value_counts(normalize=True)
```

```
Out[10]: False    0.86
         True     0.14
         Name: churn, dtype: float64
```

Type here to search

localhost:8888/notebooks/Data%20Mining%20Project-1/Experiment-1/Exploratory%20data%20analysis.ipynb

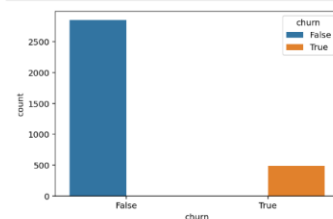
jupyter Exploratory data analysis Last Checkpoint: 4 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

```
print("precision=",precision); print("recall=",recall); print("accuracy=",accuracy)
```

```
churn      False  True  All
customer service calls
False      2245   391  2636
True        605    92   697
All        2850   483  3333
precision= 0.06599713055954089
recall= 0.09523809523809523
accuracy= 0.1752925292529253
```

```
In [41]: sns.countplot(x='churn', hue='churn', data=df);
```

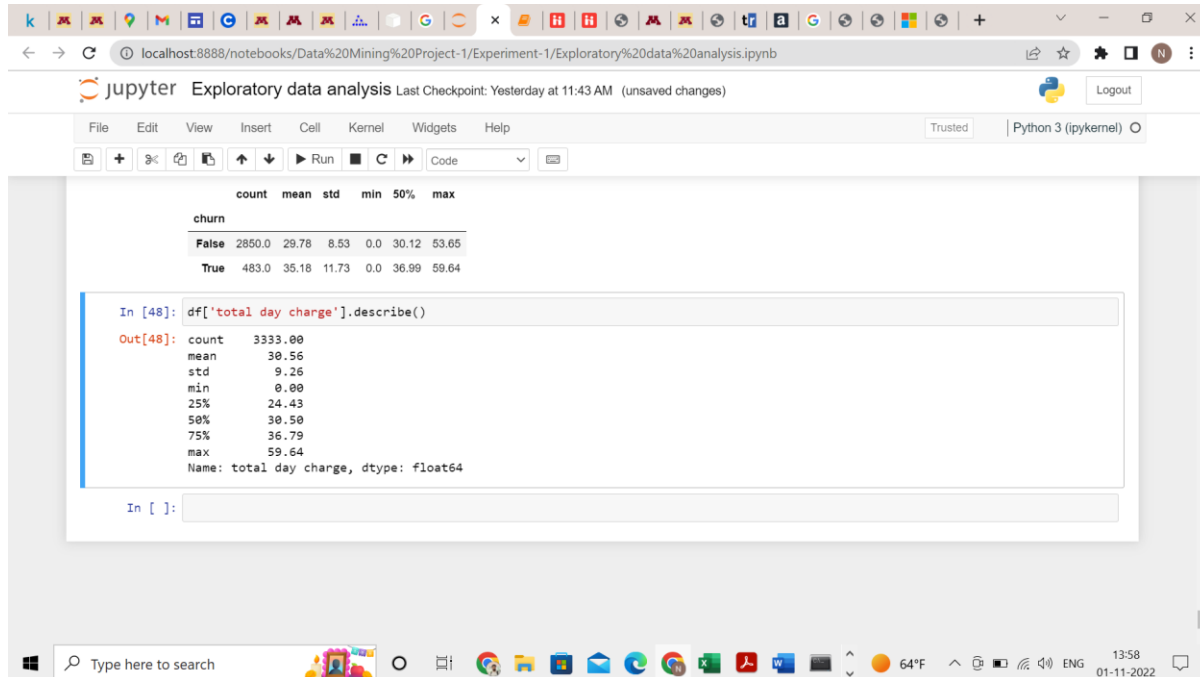


```
In [ ]:
```

Type here to search

18:28 02-11-2022

9. Min is 0 and max is 59.64



The screenshot shows a Jupyter Notebook interface. At the top, the browser address bar displays 'localhost:8888/notebooks/Data%20Mining%20Project-1/Experiment-1/Exploratory%20data%20analysis.ipynb'. The notebook title is 'Exploratory data analysis'. Below the menu bar, a summary table for the 'churn' variable is displayed:

	count	mean	std	min	50%	max
churn						
False	2850.0	29.78	8.53	0.0	30.12	53.65
True	483.0	35.18	11.73	0.0	36.99	59.64

Below the table, a code cell contains the following code:

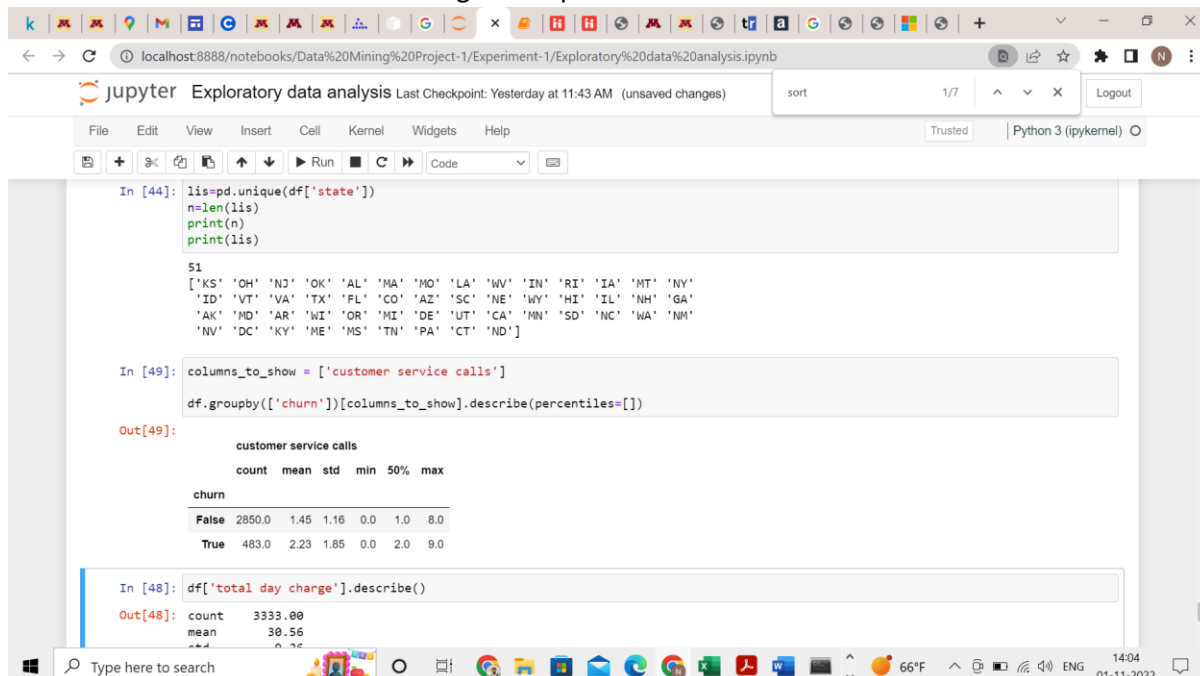
```
In [48]: df['total day charge'].describe()
```

The output of the code cell is:

```
Out[48]: count    3333.00
mean       30.56
std        9.26
min         0.00
25%        24.43
50%        30.50
75%        36.79
max        59.64
Name: total day charge, dtype: float64
```

For higher total day charge the churn is True and lesser the total day charge the churn is False.

10. The average customer service calls made by the user who has churned out of the company is 2.23 and average customer service calls made by the users who did not churn out is 1.45. This shows that the users who have churn out have more customer service calls and the users who didn't churn out have lesser customer service calls. This shows that there might be a problem with the customer service.



The screenshot shows a Jupyter Notebook interface. The code cell contains the following code:

```
In [44]: lis=pd.unique(df['state'])
n=len(lis)
print(n)
print(lis)

51
['KS' 'OH' 'NJ' 'OK' 'AL' 'MA' 'MO' 'LA' 'WV' 'IN' 'RI' 'IA' 'MT' 'NY'
 'ID' 'VT' 'VA' 'TX' 'FL' 'CO' 'AZ' 'SC' 'NE' 'WY' 'HI' 'IL' 'NH' 'GA'
 'AK' 'MD' 'AR' 'WI' 'OR' 'MI' 'DE' 'UT' 'CA' 'MN' 'SD' 'NC' 'WA' 'NM'
 'NV' 'DC' 'KY' 'ME' 'MS' 'TN' 'PA' 'CT' 'ND']
```

Below the code cell, a code cell contains the following code:

```
In [49]: columns_to_show = ['customer service calls']
df.groupby(['churn'])[columns_to_show].describe(percentiles=[])
```

The output of the code cell is a grouped summary table for 'customer service calls':

	count	mean	std	min	50%	max
churn						
False	2850.0	1.45	1.16	0.0	1.0	8.0
True	483.0	2.23	1.65	0.0	2.0	9.0

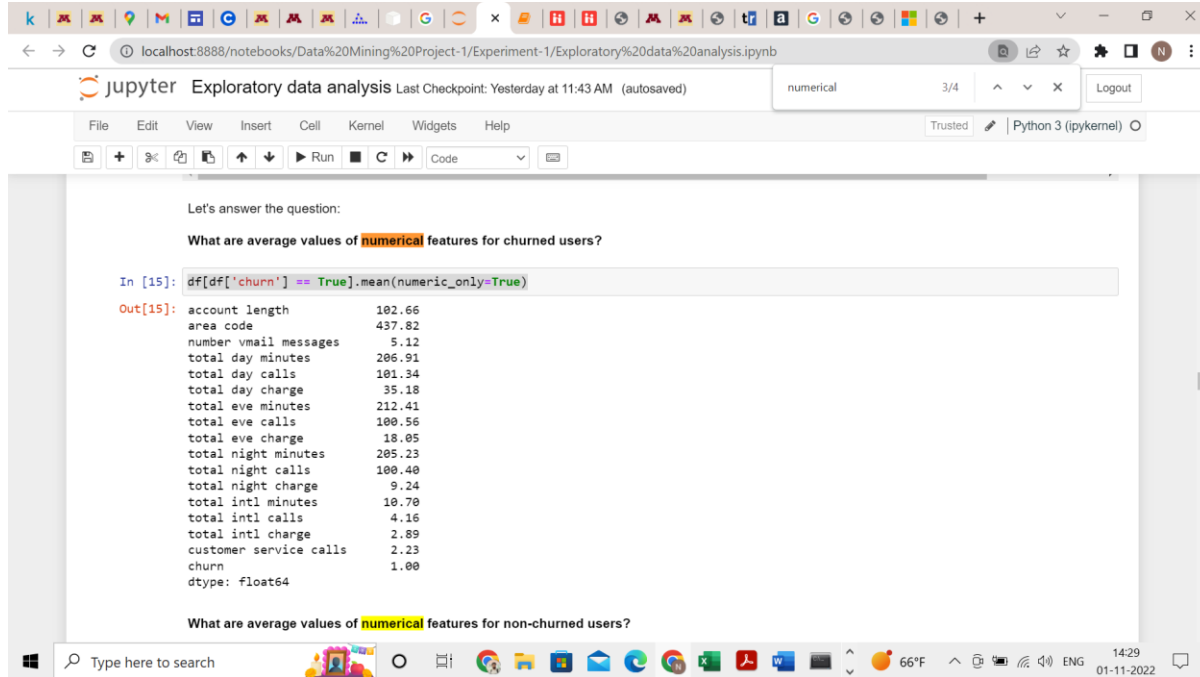
Below the table, a code cell contains the following code:

```
In [48]: df['total day charge'].describe()
```

The output of the code cell is:

```
Out[48]: count    3333.00
mean       30.56
std        9.26
```

11. For Churn=True



This screenshot shows a Jupyter Notebook interface with the title "Exploratory data analysis". The code cell contains a Python command to calculate the mean of numerical features for churned users. The output displays a list of features and their corresponding mean values.

Let's answer the question:

What are average values of **numerical** features for churned users?

```
In [15]: df[df['churn'] == True].mean(numeric_only=True)
```

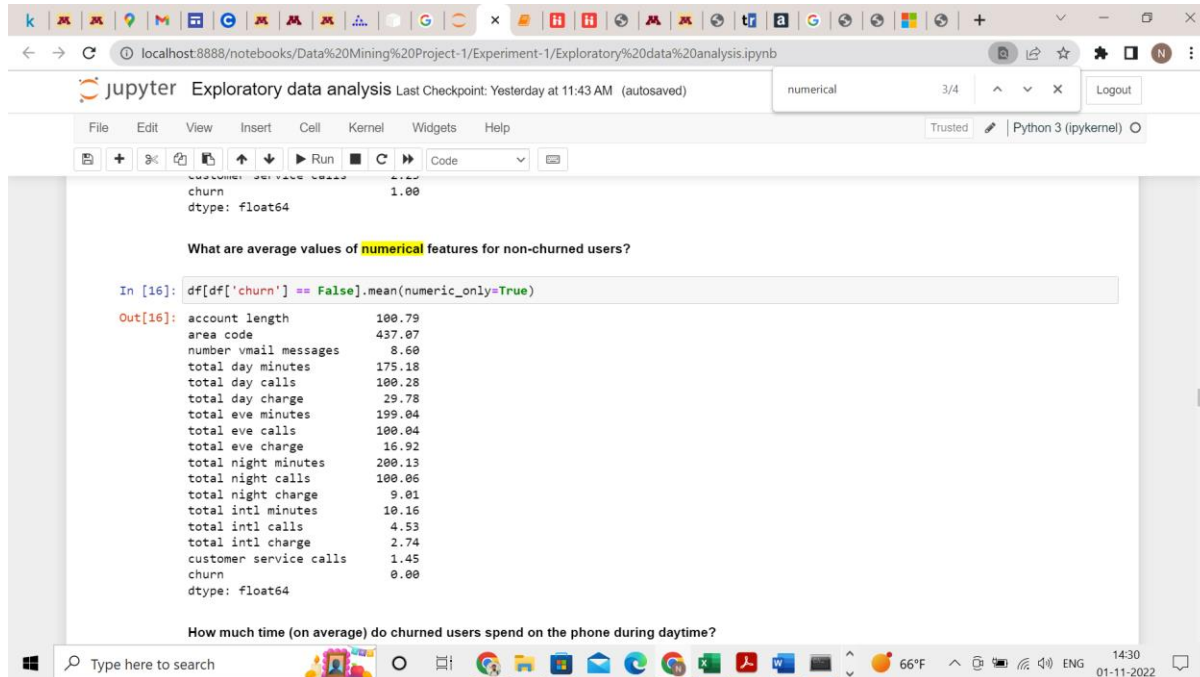
Out[15]:

Feature	Mean Value
account length	102.66
area code	437.82
number vmail messages	5.12
total day minutes	206.91
total day calls	101.34
total day charge	35.18
total eve minutes	212.41
total eve calls	100.56
total eve charge	18.05
total night minutes	205.23
total night calls	100.40
total night charge	9.24
total intl minutes	10.70
total intl calls	4.16
total intl charge	2.89
customer service calls	2.23
churn	1.00

dtype: float64

What are average values of **numerical** features for non-churned users?

For Churn=False



This screenshot shows a Jupyter Notebook interface with the title "Exploratory data analysis". The code cell contains a Python command to calculate the mean of numerical features for non-churned users. The output displays a list of features and their corresponding mean values.

What are average values of **numerical** features for non-churned users?

```
In [16]: df[df['churn'] == False].mean(numeric_only=True)
```

Out[16]:

Feature	Mean Value
account length	100.79
area code	437.07
number vmail messages	8.60
total day minutes	175.18
total day calls	100.28
total day charge	29.78
total eve minutes	199.04
total eve calls	100.04
total eve charge	16.92
total night minutes	200.13
total night calls	100.06
total night charge	9.01
total intl minutes	10.16
total intl calls	4.53
total intl charge	2.74
customer service calls	1.45
churn	0.00

dtype: float64

How much time (on average) do churned users spend on the phone during daytime?

Comparing

```

In [50]: a=df[df['churn'] == True].mean(numeric_only=True)
         b=df[df['churn'] == False].mean(numeric_only=True)
         print(a-b)

account length      1.87
area code           0.74
number vmail messages -3.49
total day minutes   31.74
total day calls      1.05
total day charge     5.40
total eve minutes   13.37
total eve calls      0.52
total eve charge     1.14
total night minutes  5.10
total night calls    0.34
total night charge   0.23
total intl minutes   0.54
total intl calls     -0.37
total intl charge    0.15
customer service calls 0.78
churn               1.00
many_service_calls  0.24
dtype: float64

```

From the above images, it should be noted the customer service calls are almost 1.5 times more from the churned-out users than the users who stayed with the company, so there may be an issue with the customer care like delays, no resolution, not able to understand the customer's problems etc. It should also be noted that the churned-out users have lesser number of voice mail messages and lesser number of international calls but more calls especially the day time and evening compared to the users who didn't churn out. So, the company could come up with a subscription focusing on local calls (not international plan) with more talk time and reasonable price to retain the users who are churning out.

12. When international call = no and churn = False

```

total intl minutes  0.54
total intl calls    -0.37
total intl charge   0.15
customer service calls 0.78
churn               1.00
many_service_calls  0.24
dtype: float64

In [42]: crss_arr=pd.crosstab(df['international plan'] == 'no', df['churn'], margins=True)

print(crss_arr); crss_arr = crss_arr.values
precision = crss_arr[1,0]/np.sum(crss_arr[:,0])
recall = crss_arr[1,0]/np.sum(crss_arr[:,1])
accuracy = np.sum(crss_arr[1,0]+crss_arr[0,1])/np.sum(crss_arr)

print("precision=",precision); print("recall=",recall); print("accuracy=",accuracy)

churn      False  True  All
international plan
False      186   137   323
True      2664  346  3010
All       2850  483  3333
precision= 0.4673684210526316
recall= 0.4425249169435216
accuracy= 0.2100960096009601

In [39]: crss_arr=pd.crosstab(df['international plan'] == 'no', df['churn'], margins=True)

print(crss_arr); crss_arr = crss_arr.values
precision = crss_arr[1,1]/np.sum(crss_arr[1,:])
recall = crss_arr[1,1]/np.sum(crss_arr[:,1])
accuracy = np.sum(crss_arr[0,0]+crss_arr[1,1])/np.sum(crss_arr)

```

Precision = 0.467

recall = 0.443

accuracy = 0.21

13. $P(\text{churn} = \text{yes} \mid \text{international plan} = \text{no}) = 346/3333 = 0.104$
 $P(\text{churn} = \text{no} \mid \text{international plan} = \text{no}) = 2664/3333 = 0.799$
 $P(\text{churn} = \text{yes} \mid \text{international plan} = \text{yes}) = 137/3333 = 0.041$
 $P(\text{churn} = \text{no} \mid \text{international plan} = \text{yes}) = 186/3333 = 0.056$
 $P(\text{churn}=\text{yes}) = 483/3333 = 0.145$
 $P(\text{churn}=\text{no}) = 2850/3333 = 0.855$
 $P(\text{international plan} = \text{yes}) = 323/3333 = 0.097$
 $P(\text{international plan} = \text{no}) = 3010/3333 = 0.903$
 $P(\text{international plan} = \text{yes} \mid \text{churn} = \text{yes}) =$

$$\frac{P(\text{churn}=\text{yes} \mid \text{international plan}=\text{yes}).P(\text{international plan}=\text{yes})}{P(\text{churn}=\text{yes})} = \frac{(0.041).(0.097)}{0.145} = 0.027$$

 $P(\text{international plan} = \text{no} \mid \text{churn} = \text{yes}) =$

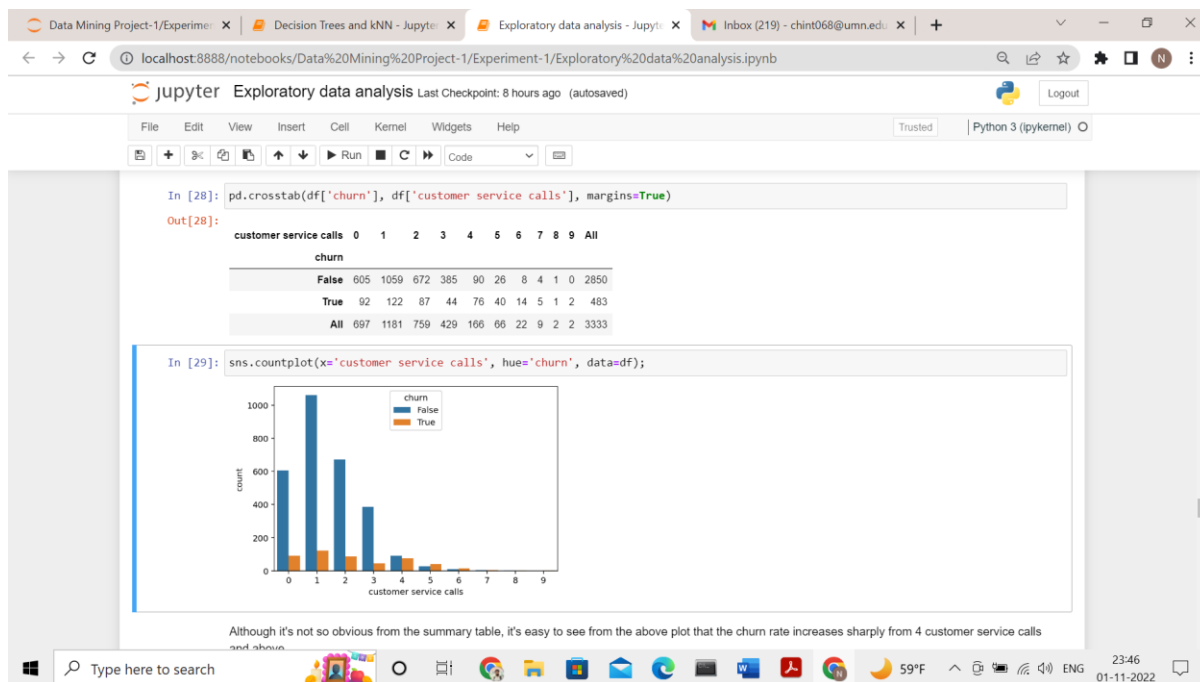
$$\frac{P(\text{churn}=\text{yes} \mid \text{international plan}=\text{no}).P(\text{international plan}=\text{no})}{P(\text{churn}=\text{yes})} = \frac{(0.104).(0.903)}{0.145} = 0.647$$

 $P(\text{international plan} = \text{yes} \mid \text{churn} = \text{no}) =$

$$\frac{P(\text{churn}=\text{no} \mid \text{international plan}=\text{yes}).P(\text{international plan}=\text{yes})}{P(\text{churn}=\text{no})} = \frac{(0.056).(0.097)}{0.855} = 0.006$$

 $P(\text{international plan} = \text{no} \mid \text{churn} = \text{no}) =$

$$\frac{P(\text{churn}=\text{no} \mid \text{international plan}=\text{no}).P(\text{international plan}=\text{no})}{P(\text{churn}=\text{no})} = \frac{(0.799).(0.903)}{0.855} = 0.844$$
14. $P(\text{Churn}=\text{True} \mid \text{customer service call}=0)$



Requesting x Google x Data Mining x Exploratory x Machine Learning x Decision Tree x Construct x CSCI 5523 x

localhost:8888/notebooks/Data%20Mining%20Project-1/Experiment-1/Exploratory%20data%20analysis.ipynb

jupyter Exploratory data analysis Last Checkpoint: a minute ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```

False      2664   346  3010
True        186   137   323
All         2850   483  3333
precision= 0.4425249169435216
recall= 0.4673684210526316
accuracy= 0.2100960096009601

In [40]: crss_arr=pd.crosstab(df['customer service calls'] == 0 , df['churn'], margins=True)

print(crss_arr); crss_arr = crss_arr.values
precision = crss_arr[1,1]/np.sum(crss_arr[1,:])
recall = crss_arr[1,1]/np.sum(crss_arr[:,1])
accuracy = np.sum(crss_arr[0,0]+crss_arr[1,1])/np.sum(crss_arr)

print("precision=",precision); print("recall=",recall); print("accuracy=",accuracy)

churn
customer service calls    False    True    All
False                    2245    391  2636
True                     605     92   697
All                      2850    483  3333
precision= 0.06599713055954089
recall= 0.09523809523809523
accuracy= 0.1752925292529253

In [ ]:

```

Type here to search 70°F 13:22 02-11-2022

$$\begin{aligned}
 & P(\text{Churn}=\text{True} \mid \text{Customer service calls}=0) \\
 &= \frac{P(\text{Customer service calls}=0 \mid \text{Churn}=\text{True}) \times P(\text{Churn}=\text{True})}{P(\text{Customer service calls}=0)} \\
 &= \frac{\left(\frac{92}{3333}\right) \times \left(\frac{483}{3333}\right)}{\left(\frac{697}{3333}\right)} = \frac{92 \times 483}{697 \times 3333} \\
 &= 0.019 \approx \boxed{0.02}
 \end{aligned}$$

15.

The screenshot shows a Jupyter Notebook titled "Exploratory data analysis" with the following content:

Let's construct another contingency table that relates *Churn* with both *International plan* and freshly created *Many_service_calls*.

```
In [41]: crss_arr=pd.crosstab(df['many_service_calls'] & (df['international plan'] == 'yes'), df['churn'], margins=True)

print(crss_arr); crss_arr = crss_arr.values
precision = crss_arr[1,1]/np.sum(crss_arr[1,:])
recall = crss_arr[1,1]/np.sum(crss_arr[:,1])
accuracy = np.sum(crss_arr[0,0]+crss_arr[1,1])/np.sum(crss_arr)

print("precision=",precision); print("recall=",recall); print("accuracy=",accuracy)
```

churn	False	True	All
row_0			
False	2841	464	3305
True	9	19	28
All	2850	483	3333

```
precision= 0.3392857142857143
recall= 0.01966873706041408
accuracy= 0.2145214521452145
```

Therefore, predicting that a customer is not loyal (*Churn*=1) in the case when the number of calls to the service center is greater than 3 and the *International Plan* is added (and predicting *Churn*=0 otherwise), we might expect an accuracy of 85.8% (we are mistaken only 464 + 9 times). This number, 85.8%, that we got through this very simple reasoning serves as a good starting point (*baseline*) for the further machine learning models that we will build.

Recall that, before the advent of machine learning, the data analysis process looked something like this. Let's recap what we've covered:

- The share of loyal clients in the sample is 85.5%. The most naive model that always predicts a "loyal customer" on such data will guess right in about

precision = 0.339

recall = 0.019

accuracy = 0.214