`

Department Of Robotics and Mechatronics Engineering

University Of Dhaka

Lab Report – 01

Subject Code – 3211

Lab Topic - Analyzing a dataset on the churn rate of telecom operator clients

Submitted To	Submitted By
Dr. Md Mehedi Hasan Assistant professor Department Of Robotics and Mechatronics Engineering University Of Dhaka	Mirza Afnan Islam Roll – AE-172-018 Session – 2020-21 Department Of Robotics and Mechatronics Engineering University Of Dhaka

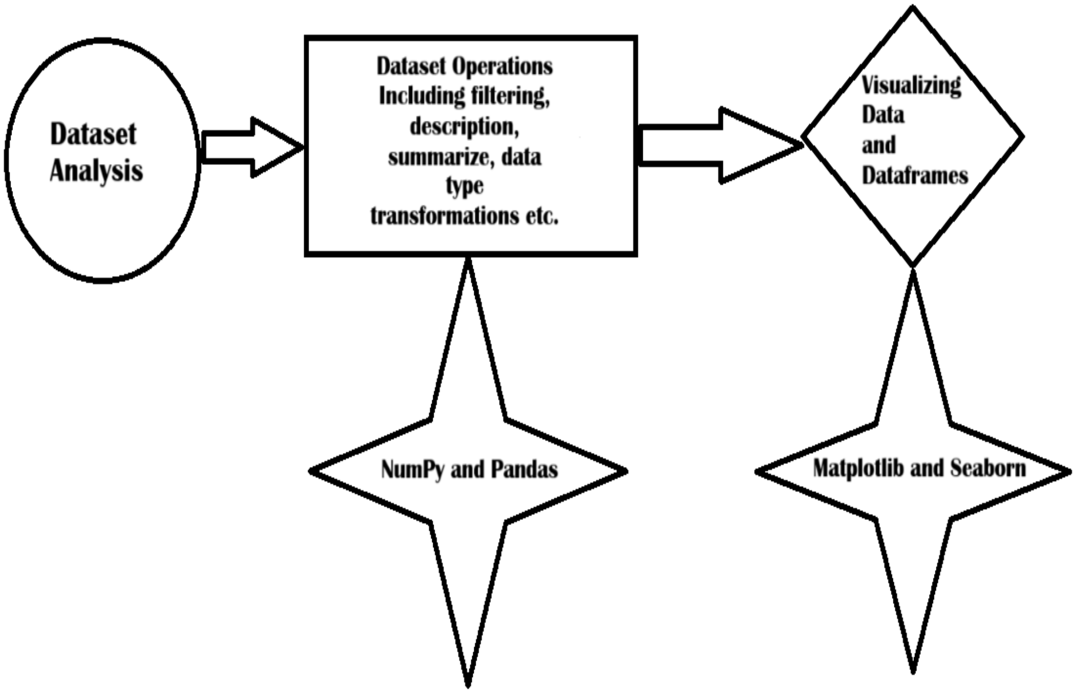
`

Objectives:

We are provided with a large dataset on a certain telecom company and churn. A dataset may contain three different types of data. They are –

a) Duplicate values
b) Outlier values
c) Validated and customized values

Duplicate value means repetition of same data of same quantity. Again, Outlier value means garbage value which is totally incomprehensive in terms of data quality. Validated and customized values are such values of data quality on which we may make operations using python libraries like Pandas or Numpy and gain productive outputs through visual representations with graph, bar-chart or pie-chart using matplotlib libraries. It helps to make uncovering the factors that are responsible for the customer churn. As a telecommunication service, understanding what causes the churn, will help develop strategies that will reduce the churn. In this lab task, we should focus on making some specific operations on provided datasets. The workflow or major operations that's followed by me on our lab task is shown below:


Dataset Analysis
Dataset Operations Including filtering, description, summarize, data type transformations etc.
Visualizing Data and Dataframes
NumPy and Pandas
Matplotlib and Seaborn

`

Dataset Analysis

Hereby, we are provided with such a dataset of Telecom company where there are 21 columns regarding state, account, length, area code, phone number, international plan, voice mail plan, 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 international minutes, total international calls, total international charge, customer service call and churn. There are 3 bool data variables which have 'YES/NO' responses. Rest of the data variables are numeric and one is char data variable. However, on char data variable, we cannot make operations without searching or plotting. Therefore, the 'State' variable is not so much keeping in operating considerations. We can make operations on numeric values –

- Operating or Identifying garbage values.
- Making simple overview of datasets.
- Data cleaning or deleting repetitive values.
- Making visualizations of dataset.
- Composing correlation matrix.

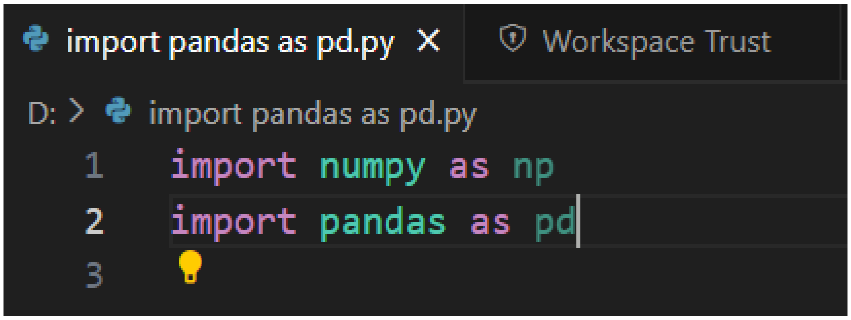
All of these operations that I made on Visual Studio Platform by Microsoft and I used the following python libraries –

Python Library Name	Logo
NumPy	
Pandas	

`

Matplotlib	matplotlib

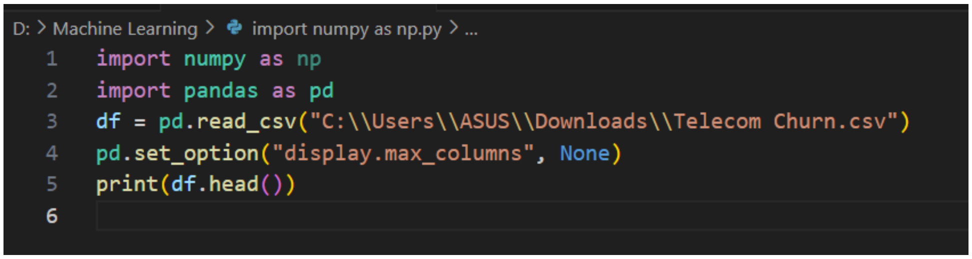
Making simple overview of datasets is one of the major parts of this dataset analysis part. In this portion, initially I import libraries Numpy and Pandas.



```python
import numpy as np
import pandas as pd
```

Figure- Importing Python Libraries

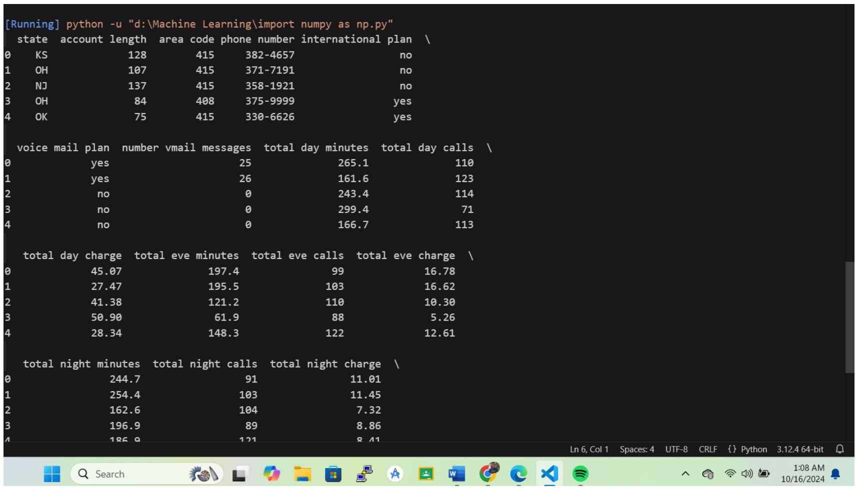
Then, I should just read the datafile that’s provided into Google Classroom. In order to get that, initially I should just download it and Load dataset (replacing DATA_URL with the actual file path.) using command. After that, to read csv file, I declare a variable named datafile in short ‘df’ where it is assigned to read downloaded csv file. In order to check it can read all data or not, I made to print a datafile read function named df.head(). This function can print only first five rows and all columns. However, the print dimension will be $5 \times n$ where n defines number of columns. This number of columns can be predefined by code with another panda (pd) display function named **pd.set_option(“display.max_columns”,None)**.Here, attribute **display.max_columns** is used to show all columns in code terminal. It configures pandas to show all columns of the DataFrame when displaying it, without truncation of columns; useful for viewing all of the data.



```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
print(df.head())
```

`

Output Console following:



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   total day charge  total eve minutes  total eve calls  total eve charge  \
0             45.07              197.4               99             16.78
1             27.47              195.5              103             16.62
2             41.38              121.2              110             10.30
3             50.90               61.9               88              5.26
4             28.34              148.3              122             12.61

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41
```

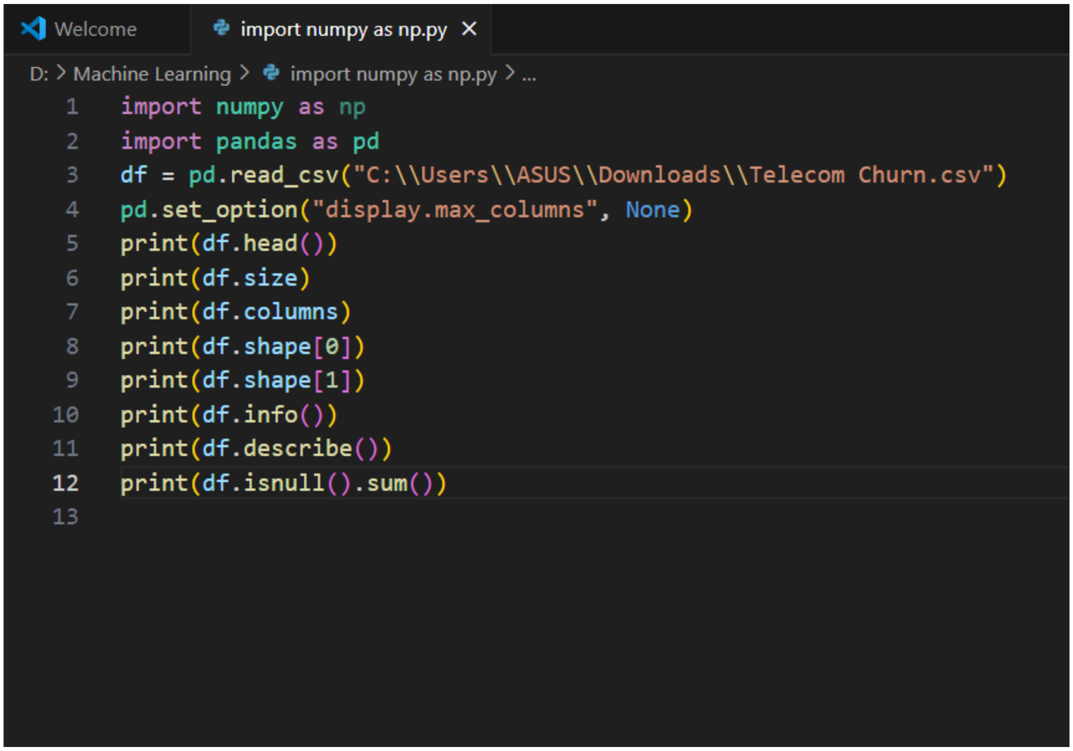
Ln 6, Col 1   Spaces: 4   UTF-8   CRLF   {} Python   3.12.4 64-bit
Search
1:08 AM
10/16/2024

After that I dived into more features of data specially data dimensionality, feature names, and feature types. Data dimension means number of datafiles row and columns. Again, in order to know the name of rows or columns we can define seven different methods.

- ✓ **print(df.size):** It defines the number of elements of data matrix.
- ✓ **print(df.columns):** It defines the name or title of columns.
- ✓ **print(df.shape[0]):** It defines the number of rows that our datafile contains. Meanwhile, each row predefines each user's info.
- ✓ **print(df.shape[1]):** It defines total number of columns.
- ✓ **print(df.info()):** It gives an overview of the dataset's structure and data types. Data types are either Boolean or integer or charcter.
- ✓ **print(df.describe()):** It gives an overview of datasets descriptive statistics for numerical columns to better understand the distribution of data.
- ✓ **print(df.isnull().sum()):** It returns a DataFrame of the same shape as df, where each entry is True if the corresponding value is null (missing) and False otherwise.

The code snippets are shown below:

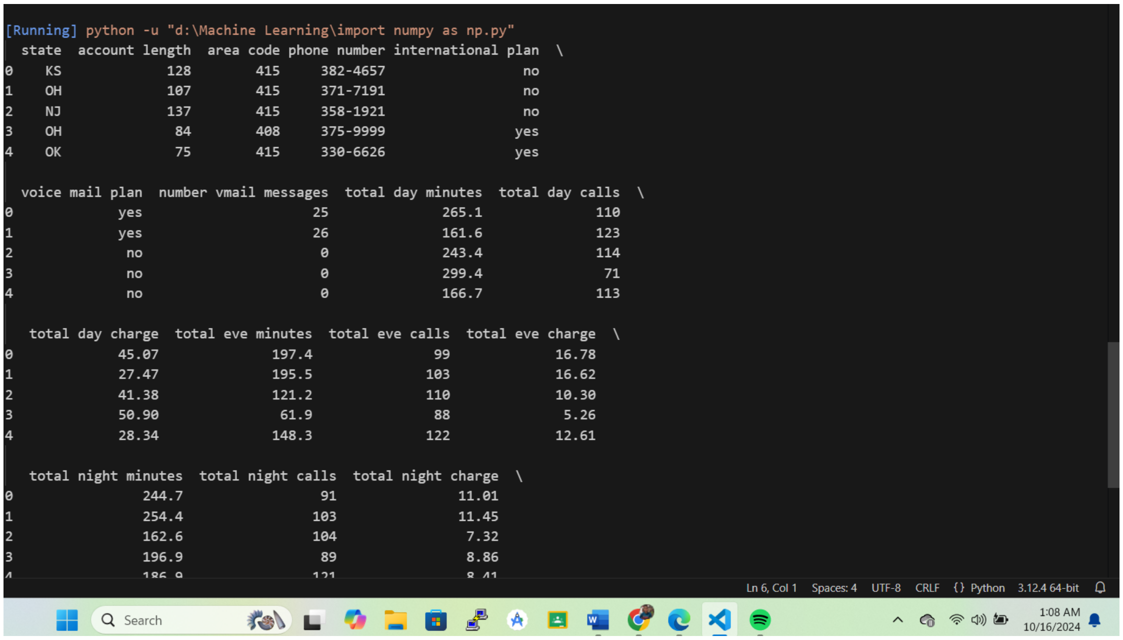
`



```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
print(df.head())
print(df.size)
print(df.columns)
print(df.shape[0])
print(df.shape[1])
print(df.info())
print(df.describe())
print(df.isnull().sum())
```

Output console following

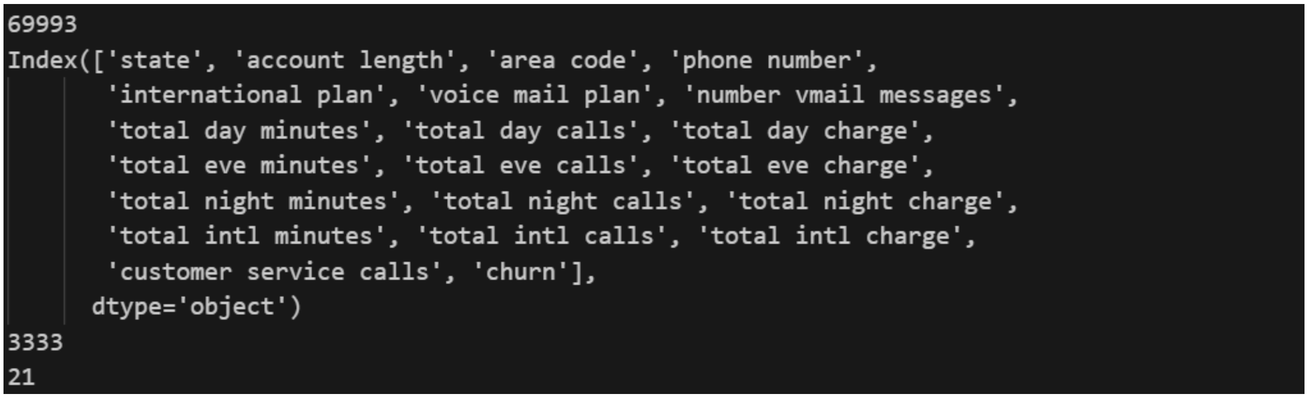
1. Dataset Reading Output



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes

  voice mail plan  number vmail messages  total day minutes  total day calls  \
0             yes                     25              265.1              110
1             yes                     26              161.6              123
2              no                      0              243.4              114
3              no                      0              299.4               71
4              no                      0              166.7              113

   total day charge  total eve minutes  total eve calls  total eve charge  \
0             45.07              197.4               99             16.78
1             27.47              195.5              103             16.62
2             41.38              121.2              110             10.30
3             50.90               61.9               88              5.26
4             28.34              148.3              122             12.61

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41
```

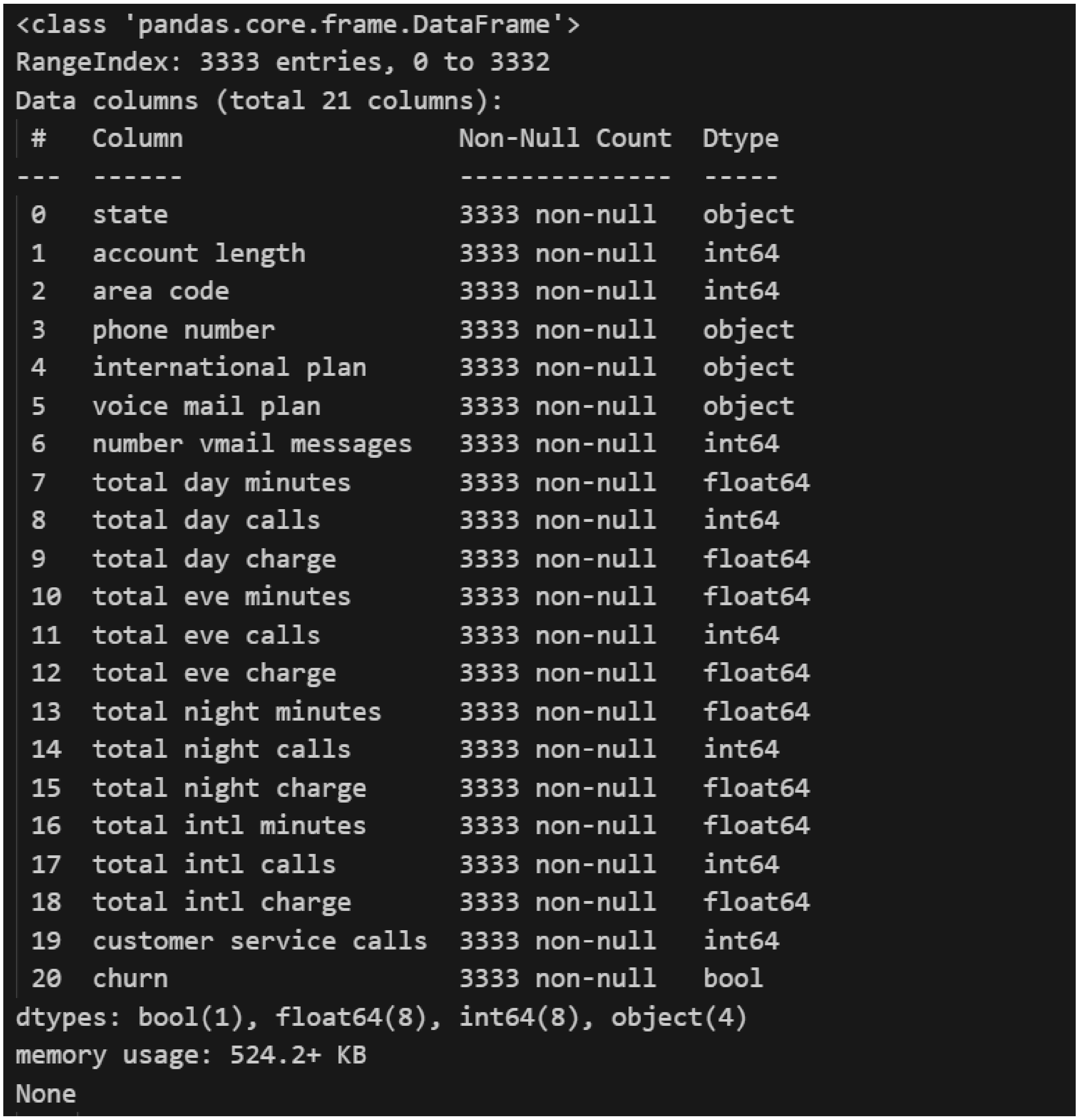
`

2. Data Features Output



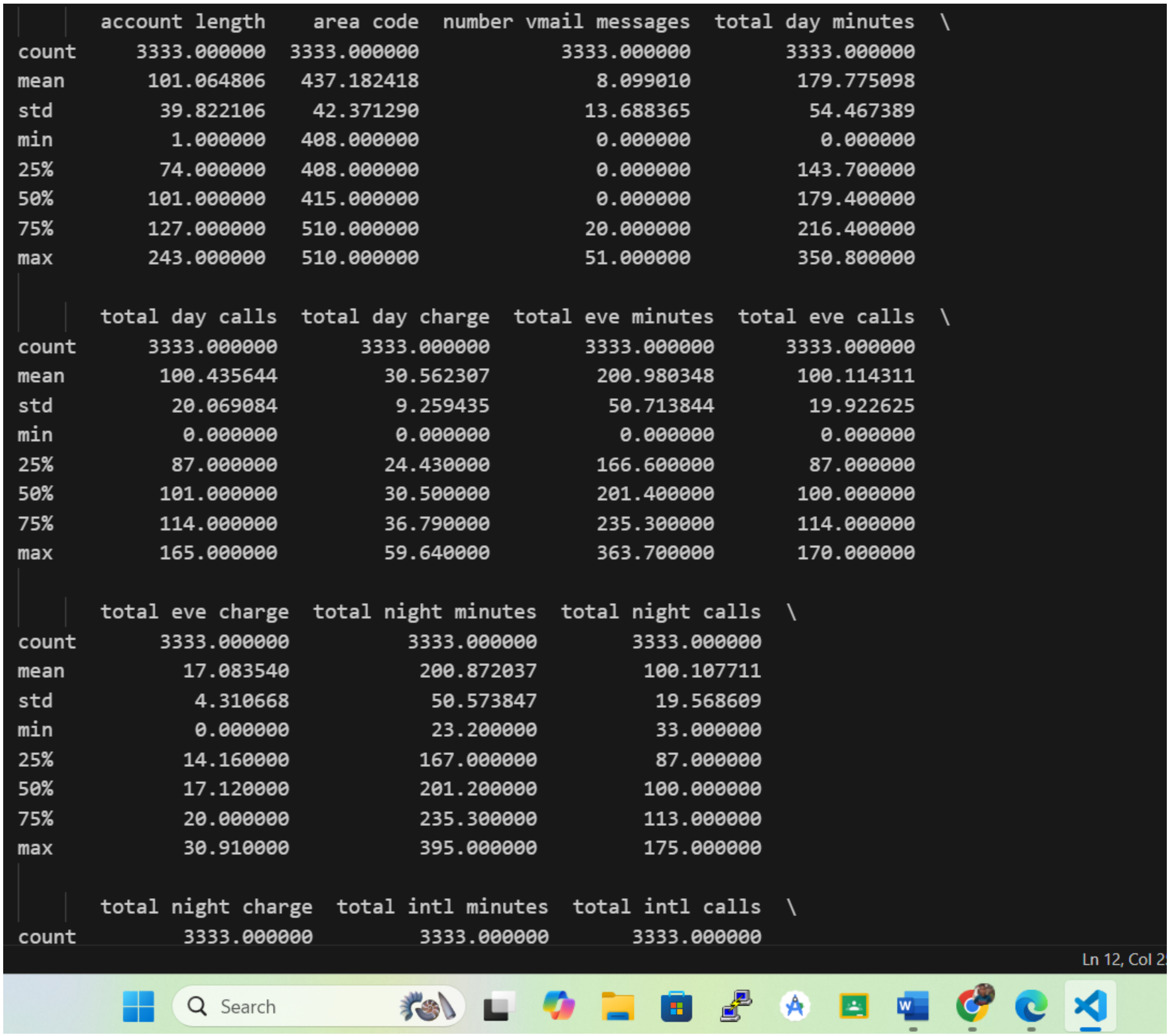
```
69993
Index(['state', 'account length', 'area code', 'phone number',
       'international plan', 'voice mail plan', 'number vmail messages',
       'total day minutes', 'total day calls', 'total day charge',
       'total eve minutes', 'total eve calls', 'total eve charge',
       'total night minutes', 'total night calls', 'total night charge',
       'total intl minutes', 'total intl calls', 'total intl charge',
       'customer service calls', 'churn'],
      dtype='object')
3333
21
```

3. Data info



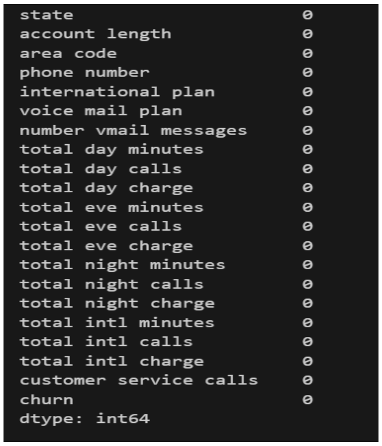
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   3333 non-null   object
 1   account length          3333 non-null   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
```

4. Data description:



```
       account length   area code  number vmail messages  total day minutes  \
count     3333.000000  3333.000000           3333.000000        3333.000000
mean       101.064806   437.182418              8.099010         179.775098
std         39.822106    42.371290             13.688365          54.467389
min          1.000000   408.000000              0.000000           0.000000
25%         74.000000   408.000000              0.000000         143.700000
50%        101.000000   415.000000              0.000000         179.400000
75%        127.000000   510.000000             20.000000         216.400000
max        243.000000   510.000000             51.000000         350.800000

       total day calls  total day charge  total eve minutes  total eve calls  \
count      3333.000000       3333.000000        3333.000000      3333.000000
mean        100.435644         30.562307         200.980348       100.114311
std          20.069084          9.259435          50.713844        19.922625
min           0.000000          0.000000           0.000000         0.000000
25%          87.000000         24.430000         166.600000        87.000000
50%         101.000000         30.500000         201.400000       100.000000
75%         114.000000         36.790000         235.300000       114.000000
max         165.000000         59.640000         363.700000       170.000000

       total eve charge  total night minutes  total night calls  \
count       3333.000000          3333.000000        3333.000000
mean          17.083540           200.872037         100.107711
std            4.310668            50.573847          19.568609
min            0.000000            23.200000          33.000000
25%           14.160000           167.000000          87.000000
50%           17.120000           201.200000         100.000000
75%           20.000000           235.300000         113.000000
max           30.910000           395.000000         175.000000

       total night charge  total intl minutes  total intl calls  \
count         3333.000000         3333.000000       3333.000000
```

5. Null checking:



```
state                     0
account length            0
area code                 0
phone number              0
international plan        0
voice mail plan           0
number vmail messages     0
total day minutes         0
total day calls           0
total day charge          0
total eve minutes         0
total eve calls           0
total eve charge          0
total night minutes       0
total night calls         0
total night charge        0
total intl minutes        0
total intl calls          0
total intl charge         0
customer service calls    0
churn                     0
dtype: int64
```

`

Data operations

In machine learning, most of the cases we make data manipulation or data handling or handling data errors by making operations on a certain dataset. Dataset operation set is a subset of Data manipulating mechanism. According to the definition, dataset operation defines such a required process or useful criterion which will make the dataset more useful, valid and rearranged in a certain order. For example, we know Boolean data variable only contains two keywords. Those are, “YES or NO”. But there is a operation called data_type change or data_type tranfer using a certain method. Hence, the Boolean data variable might change into integer type data variable which should be bounded in between 0 to 1 [According to Fuzzy Logic]. Similarly, there are many data manipulating operations. Among them, I will do the followings [According to the provided lecture sheet] –

- Data type conversion
- Specific data type descripting
- Counting the data responses
- Sorting
- Data indexing and retrieving
- Applying functions to cell, columns and rows
- Grouping and Summarizing the tabulation
- DataFrame transformations

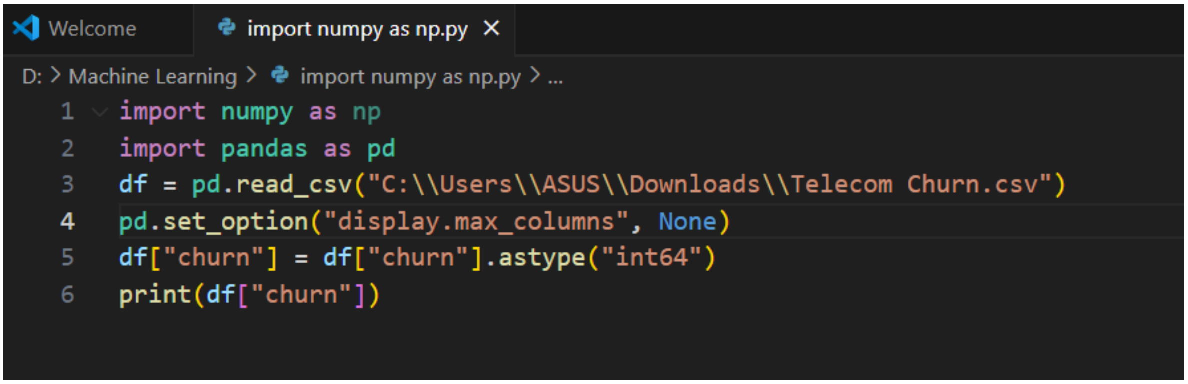
These operations making on telecom.csv file is shown extendedly below with examples:

- **Data type conversion**: Data type conversion means converting data type into one form to another. For example, in our given dataset there are a Boolean data variable named “Churn” which includes only “YES/NO” response. To transform “YES/NO” response into “1/0” we use a pandas method named-

```
df [“variable_name”] = df[“variable_name”].astype(“datatype”)
```

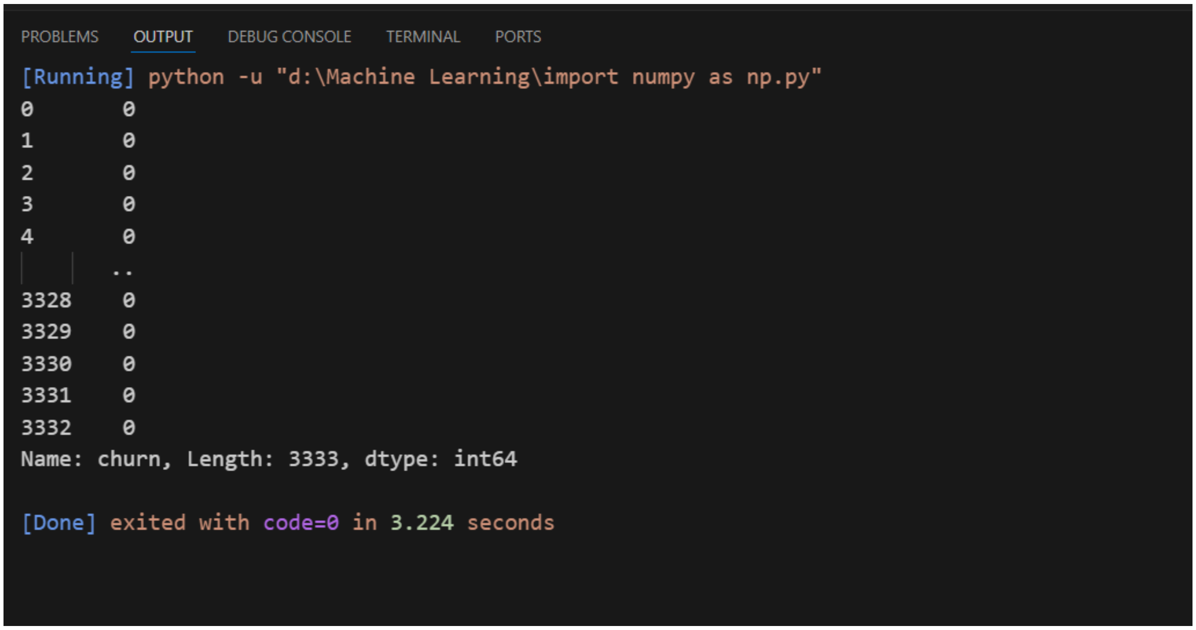
`

Code Snippets:



```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
df["churn"] = df["churn"].astype("int64")
print(df["churn"])
```

Output Console:



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
0       0
1       0
2       0
3       0
4       0
        ..
3328    0
3329    0
3330    0
3331    0
3332    0
Name: churn, Length: 3333, dtype: int64

[Done] exited with code=0 in 3.224 seconds
```

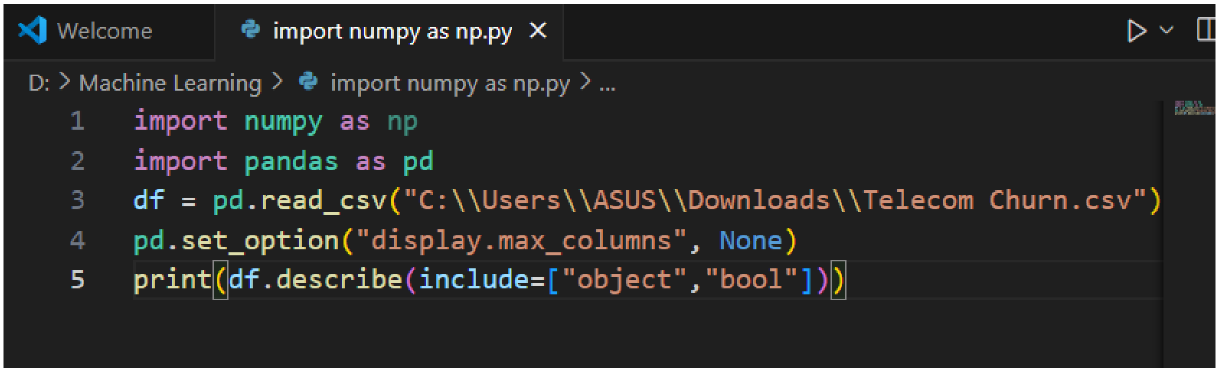
❖ Specific data type descripting :

Sometimes we don’t need to describe all data or we don’t need to know all columns data types. We need some specific type data. For example, let I assume that, I need to know whether any survey is happened about the loyality or disloyality of customer’s behaviour. Ofcourse, to know that, we need to know whether Boolean type data variable is enlisted or not. Hence, we need a summarization of categorical or non categorical datatypes. To do this, we need a command that -

df.describe(include=["data_type1","data_type2"])

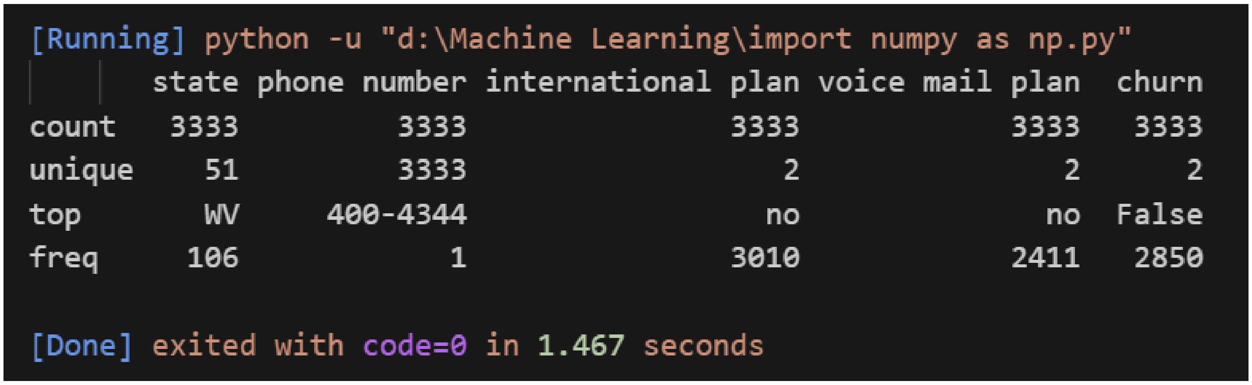
This command in pandas generates a summary of the statistics for the columns in the **DataFrame(df)** that are of data type object **(usually representing strings or categorical data)** and bool **(Boolean_values).**

Code Snippets:



```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
print(df.describe(include=["object","bool"]))
```

Output Console:



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
       state phone number international plan voice mail plan  churn
count   3333         3333               3333            3333   3333
unique    51         3333                  2               2      2
top       WV     400-4344                 no              no  False
freq     106            1               3010            2411   2850

[Done] exited with code=0 in 1.467 seconds
```

❖ Counting the data responses:

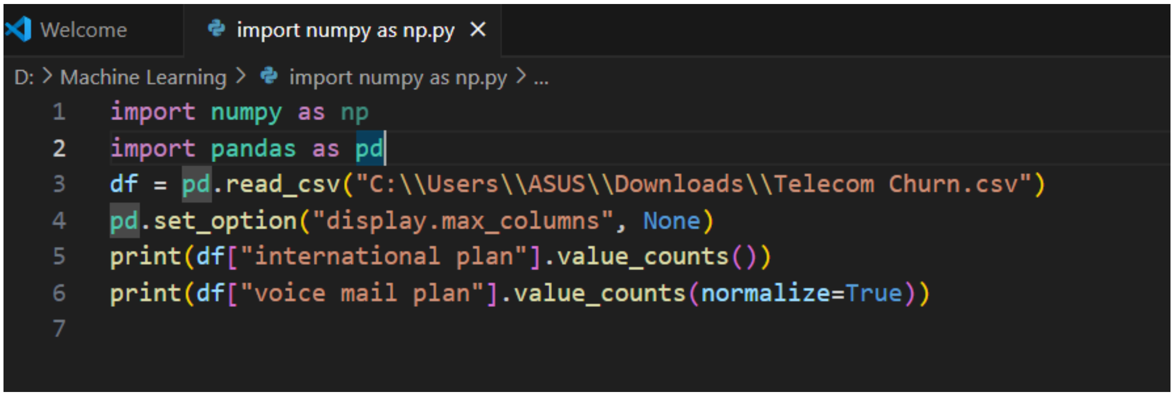
Counting helps in understanding the distribution of categorical variables regarding showing the frequency of each unique value. For instance, in data from a survey or poll, if the options were "Yes", "No", and "Maybe", knowing how many participants chose each option is beneficial in analyzing the emerging pattern or stated preference. In our dataset, we may count the number of users who are agree or disagree to the churn of service. For this, we may use two methods. They are-

1. *df["Variable_name"].value_counts()*
2. *df["Variable_name"].value_counts(normalize=true)*

To count data in a DataFrame, the count() function or other methods like value_counts() for specific columns can be used. Here, variable name should be Boolean type data or numeric. For example, in our dataset there are "international plan" and "voice mail plan" which are boolean type data. Now we, can count in numbers or percentage formation to count the user's response.

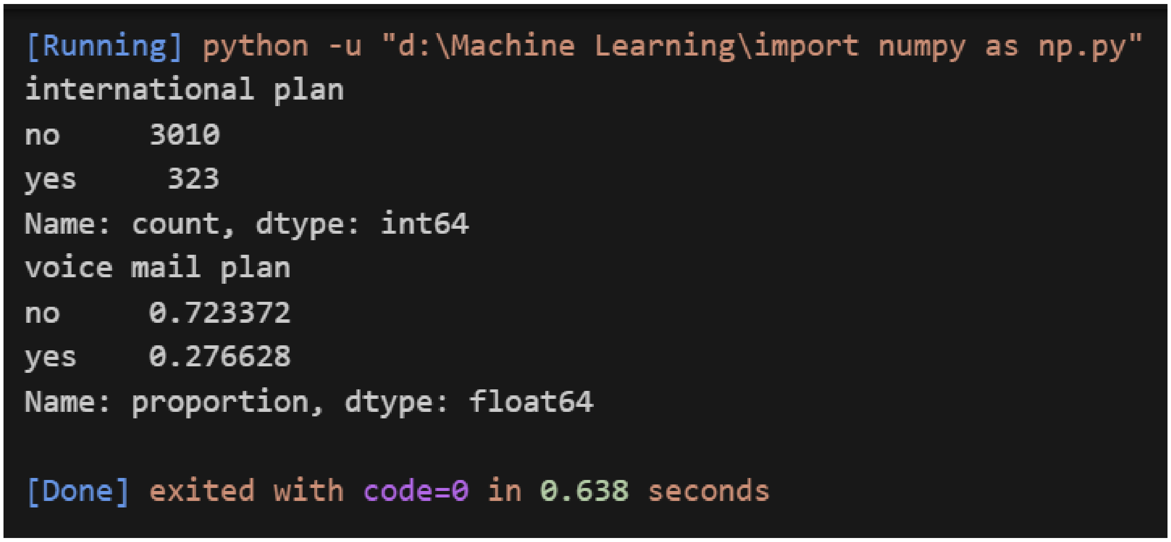
Code Snippets:

`



```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
print(df["international plan"].value_counts())
print(df["voice mail plan"].value_counts(normalize=True))
```

Output Console:



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
international plan
no      3010
yes      323
Name: count, dtype: int64
voice mail plan
no      0.723372
yes     0.276628
Name: proportion, dtype: float64

[Done] exited with code=0 in 0.638 seconds
```

❖ **Sorting:** It's not obligatory that we should find an arranged set of data. We may comprehensively customize the order of data(single columns or multiple columns) according to our need and condition. In order to implement this, we should use a panda library inclusive method called

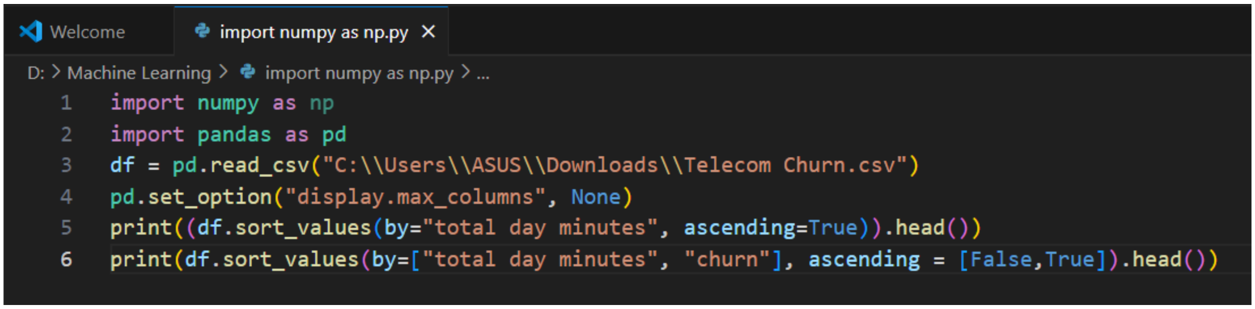
1. *df.sort_values(by="column name", ascending = (True or False)).head() // For operating in a single line*

2. *df.sort_values(by=["column name", "column name",………………], ascending = [(True or False),(True or False), ……….]).head() // For operating in multiple lines*

Here, **df.sort_values** is a method where according to a certain column name we can arrange values in ascending or descending orders and then print the top five data rows in dimension matrix of 5×21.

`

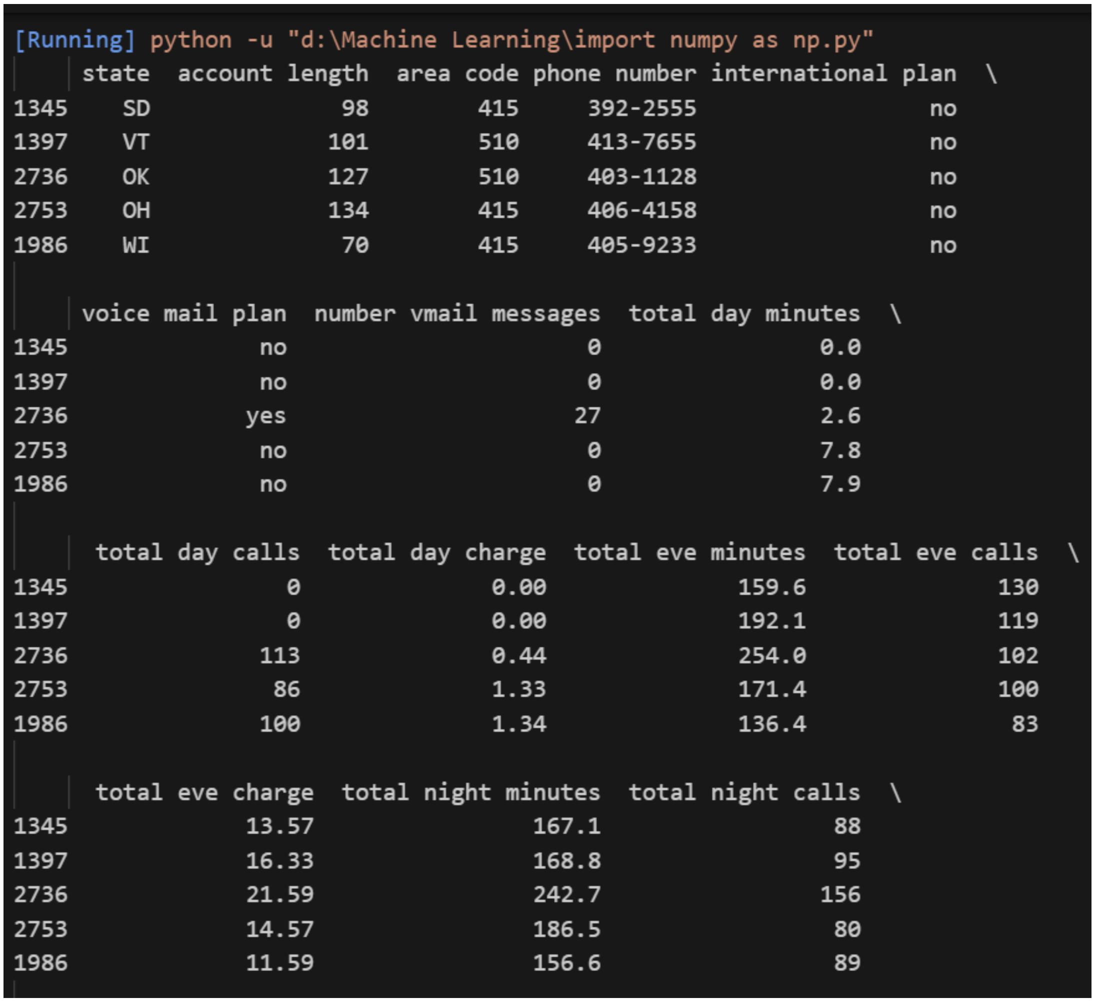
Code snippets:



```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
print((df.sort_values(by="total day minutes", ascending=True)).head())
print(df.sort_values(by=["total day minutes", "churn"], ascending = [False,True]).head())
```

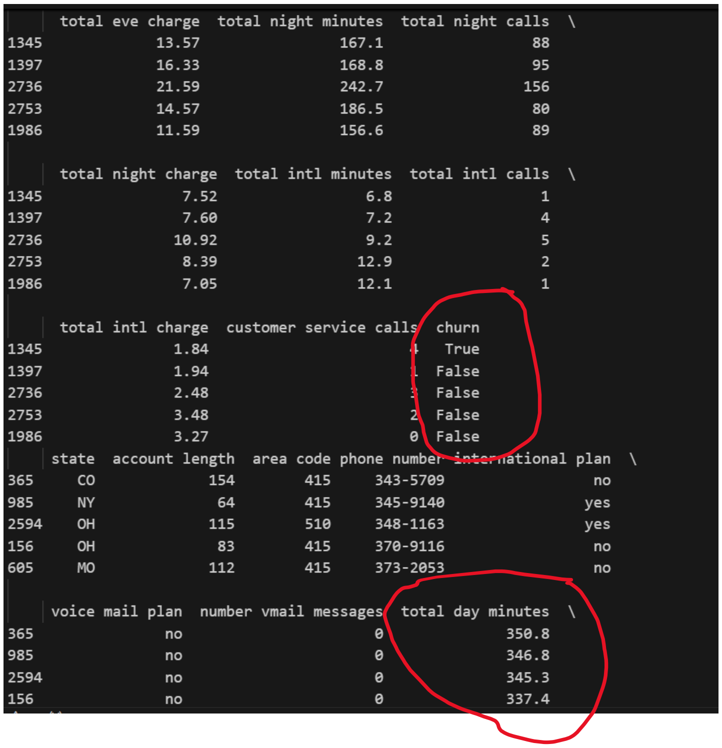
Output Console:

Single conditional output-



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
     state  account length  area code phone number international plan  \
1345    SD              98        415     392-2555                 no   
1397    VT             101        510     413-7655                 no   
2736    OK             127        510     403-1128                 no   
2753    OH             134        415     406-4158                 no   
1986    WI              70        415     405-9233                 no   

     voice mail plan  number vmail messages  total day minutes  \
1345              no                      0                0.0   
1397              no                      0                0.0   
2736             yes                     27                2.6   
2753              no                      0                7.8   
1986              no                      0                7.9   

      total day calls  total day charge  total eve minutes  total eve calls  \
1345                0              0.00              159.6              130   
1397                0              0.00              192.1              119   
2736              113              0.44              254.0              102   
2753               86              1.33              171.4              100   
1986              100              1.34              136.4               83   

      total eve charge  total night minutes  total night calls  \
1345             13.57                167.1                 88   
1397             16.33                168.8                 95   
2736             21.59                242.7                156   
2753             14.57                186.5                 80   
1986             11.59                156.6                 89   
```

Multiple conditional outputs-



```
      total eve charge  total night minutes  total night calls  \
1345             13.57                167.1                 88
1397             16.33                168.8                 95
2736             21.59                242.7                156
2753             14.57                186.5                 80
1986             11.59                156.6                 89

      total night charge  total intl minutes  total intl calls  \
1345                7.52                 6.8                 1
1397                7.60                 7.2                 4
2736               10.92                 9.2                 5
2753                8.39                12.9                 2
1986                7.05                12.1                 1

      total intl charge  customer service calls  churn
1345               1.84                       4   True
1397               1.94                       1  False
2736               2.48                       3  False
2753               3.48                       2  False
1986               3.27                       0  False
     state  account length  area code phone number international plan  \
365     CO             154        415     343-5709                 no
985     NY              64        415     345-9140                yes
2594    OH             115        510     348-1163                yes
156     OH              83        415     370-9116                 no
605     MO             112        415     373-2053                 no

     voice mail plan  number vmail messages  total day minutes  \
365               no                      0              350.8
985               no                      0              346.8
2594              no                      0              345.3
156               no                      0              337.4
```

Here the outputs are shown according to ascending order of churn and descending order of total day minutes.

❖ ***Indexing or retrieving data:***

Indexing or data retrieving means setting any particular column as the index of the DataFrame. These indexes are having improved performance for some operation like data retrieval and lookups. Data retrieving has allowed the selection of rows or columns either as Label-based or integer-based. Retrieving all data with respect to a particular customer with his account number is an example of data retrieve. In data indexing and retrieve we should lookup on following customizations

1. Certain variables mean
2. Certain tabular split using loc and iloc method
3. Determining maxima or minima values of a certain dataset.

Mean defines average of all inclusive data including outliers on non-outliers. Here, we can make means by using panda method called

df[“column name”].mean()

It shows mean of that certain columns data. This operation helps summarize the central tendency of numerical variables like call minutes or charges.

`

loc method is mainly a label-based indexing. It's used for selecting rows and columns by labels. This gives a subset of rows where the State is 'KS', and only specific columns are shown. **iloc** method is position-based indexing. It's used to select rows and columns by their integer positions.

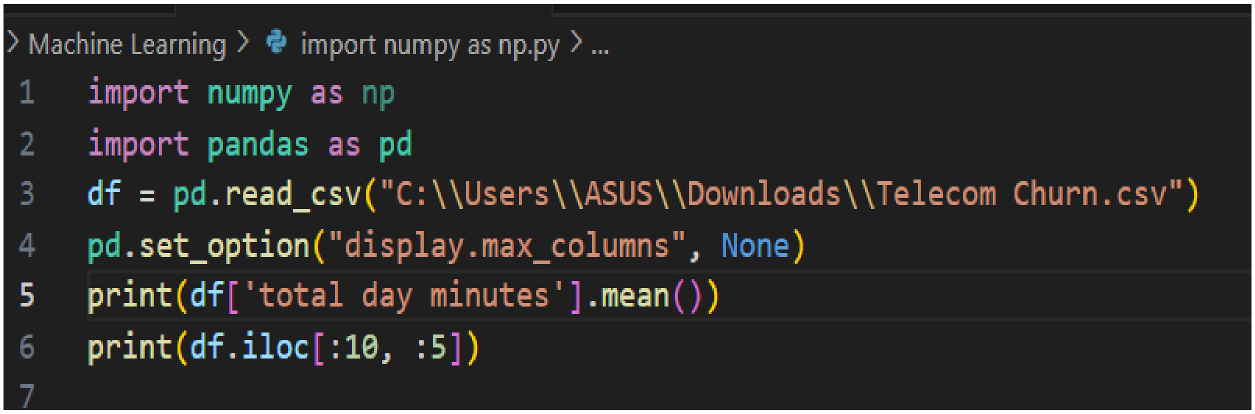
df.loc[df['Column name_1'] == 'Element name', ['Column name_2', 'Column name_3']]

max() and **min()** are essential for finding extreme values, such as the highest and lowest charges or usage times across customers. These methods return the maximum and minimum values in a specified column.

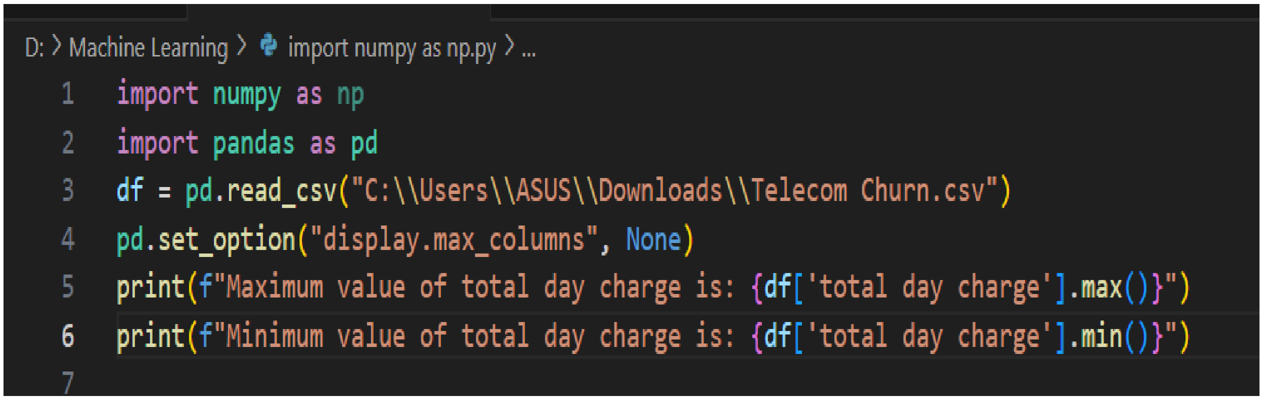
df['column_name'].max() // for maxima finding

df['column_name'].min() // for minima finding

Code snippets:



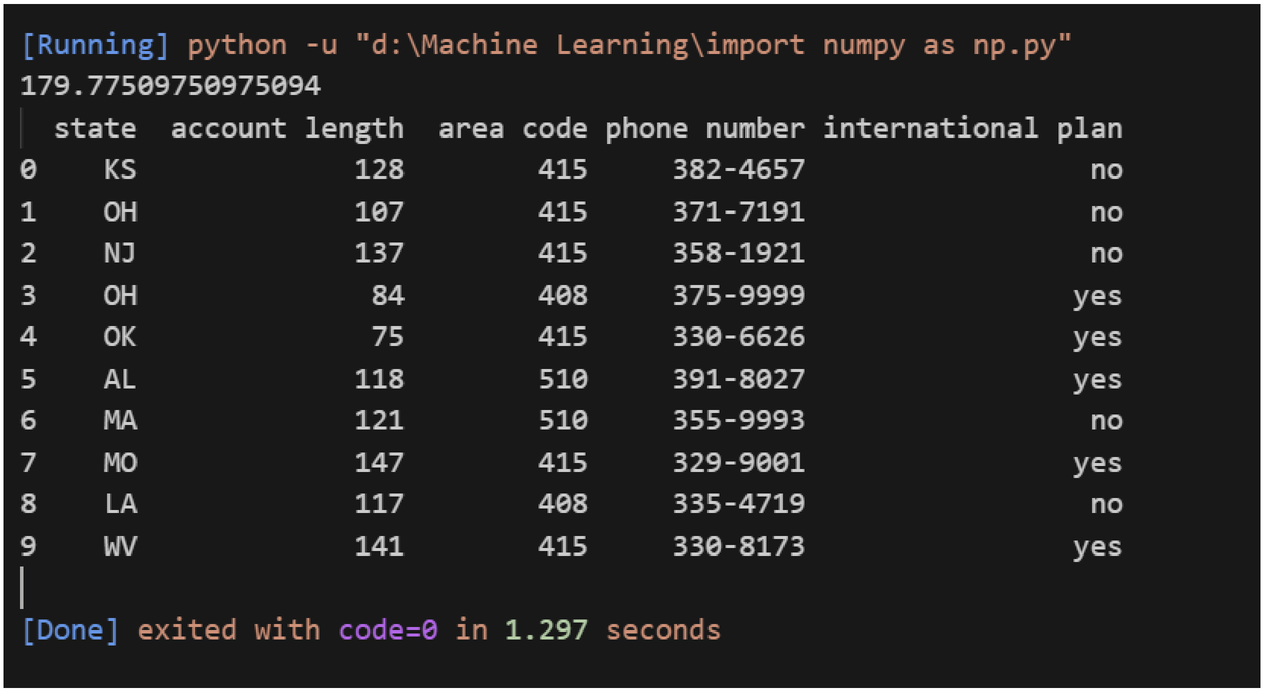
```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
print(df['total day minutes'].mean())
print(df.iloc[:10, :5])

```



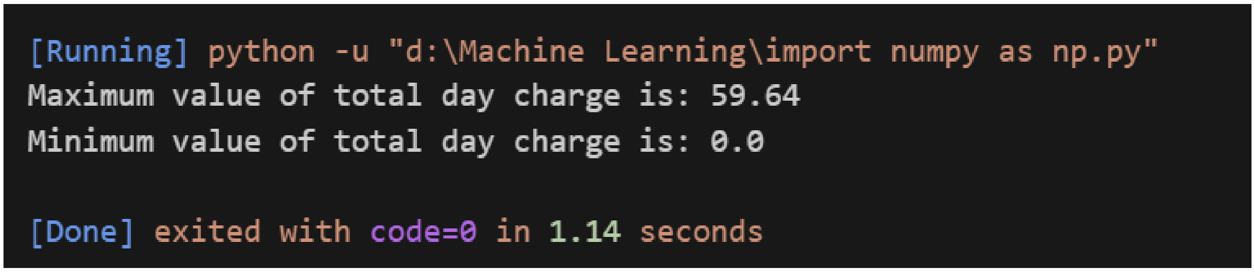
```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
print(f"Maximum value of total day charge is: {df['total day charge'].max()}")
print(f"Minimum value of total day charge is: {df['total day charge'].min()}")

```

`

Console outputs:



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
179.77509750975094
  state  account length  area code phone number international plan
0    KS             128        415     382-4657                 no
1    OH             107        415     371-7191                 no
2    NJ             137        415     358-1921                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes
5    AL             118        510     391-8027                yes
6    MA             121        510     355-9993                 no
7    MO             147        415     329-9001                yes
8    LA             117        408     335-4719                 no
9    WV             141        415     330-8173                yes

[Done] exited with code=0 in 1.297 seconds
```



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
Maximum value of total day charge is: 59.64
Minimum value of total day charge is: 0.0

[Done] exited with code=0 in 1.14 seconds
```

- **Applying functions to cell, columns and rows:**

  We can apply functions to specific cells, entire columns, or rows by using two different methods. They are:

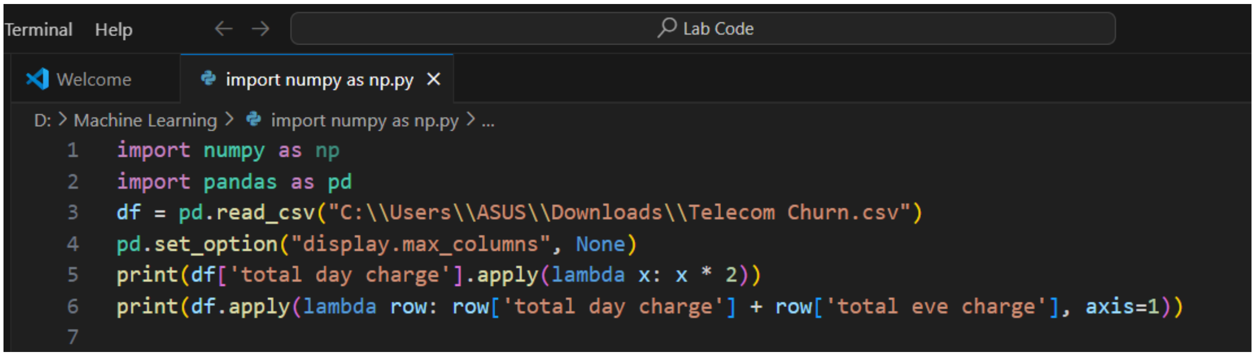
  1. ***apply():*** It's highly preferable for a column, which applies a function to each element in that column. Using this we can centrally calculate and handle the data elements individually.
  2. ***applymap():*** It's used to apply a function to each element of the DataFrame. Using this we can centrally calculate and handle the whole dataset elements individually.

  Apply() method is a subset method of applymap() because applymap() works with whole element of data framework but apply() method works on each element of a certain column.

`

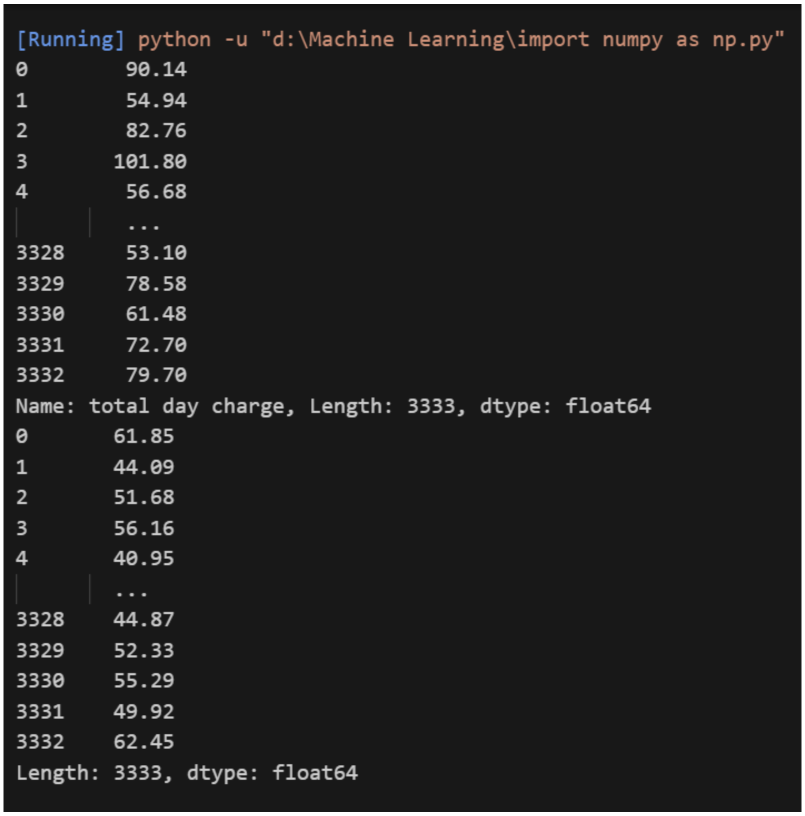
Code snippets:

Example of applying a lambda function to multiply every value in 'total day charge' by 2 and summing values from multiple columns for each row



```
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
print(df['total day charge'].apply(lambda x: x * 2))
print(df.apply(lambda row: row['total day charge'] + row['total eve charge'], axis=1))
```

Output console:



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
0        90.14
1        54.94
2        82.76
3       101.80
4        56.68
         ...
3328     53.10
3329     78.58
3330     61.48
3331     72.70
3332     79.70
Name: total day charge, Length: 3333, dtype: float64
0       61.85
1       44.09
2       51.68
3       56.16
4       40.95
        ...
3328    44.87
3329    52.33
3330    55.29
3331    49.92
3332    62.45
Length: 3333, dtype: float64
```

- ***Grouping and Summarizing the tabulation:*** Grouping and summarizing are important activities in data manipulation, with reasons such as analysis of big data. These operations provide insight into the underlying patterns, an overview of data quality, and a basis on which data-driven decisions can be effected. We can use groupby() to group data by one or more columns. Furthermore to make data aggregation, we may use sum(), mean(), count(), etc. Initially,

`

the groupby method divides the grouping_columns by their values. Then, columns are selected. If columns_to_show is not included, all non groupby clauses will be included. Finally, one or several functions are applied to the obtained groups per selected columns.

```
columns_to_show = ["column_name", "column_name",……………….]
df.groupby(["column_name"])[columns_to_show].agg([np.mean/ np.std
/  np.min / np.max])
```

If we want to see how the observations in our sample are distributed in the context of multiple variables, we can use a contingency table. To make a contingency table we use crosstab method.

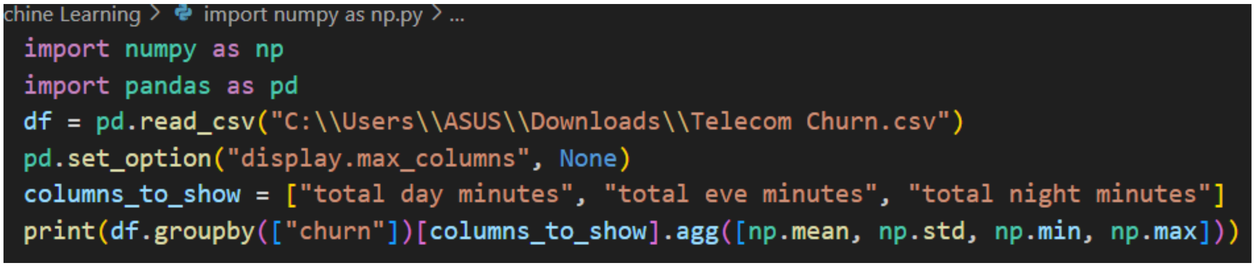
```
    pd.crosstab(df["Column_name"], df["Column_name"])
```

Pivot tables are known to anyone who has worked with Excel. And of course, in Pandas, there is also the implementation of pivot tables: the method “pivot_table” takes the following parameters:

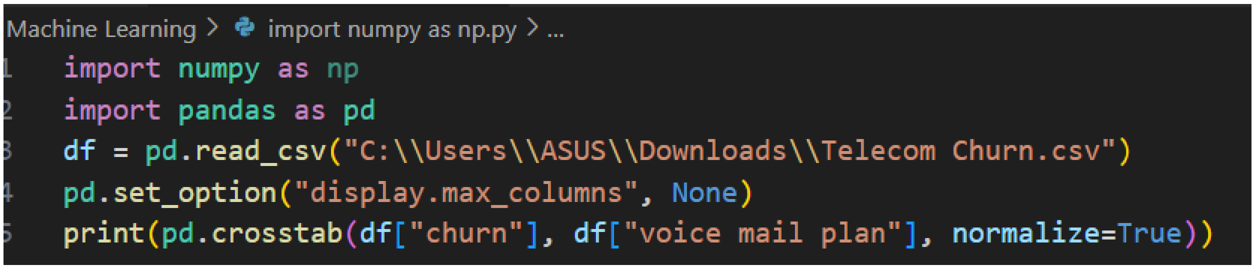
1. **values** - a list of variables for which statistics will be calculated,
2. **index** - the list of variables to group data over
3. **aggfunc** - what statistics we need to calculate for groups, ex. sum, mean, maximum, minimum or something else.

```
df.pivot_table(["Variable_name"],["Variable_name"],aggfunc="mean/max
/min/median/mode")
```

Code snippets:

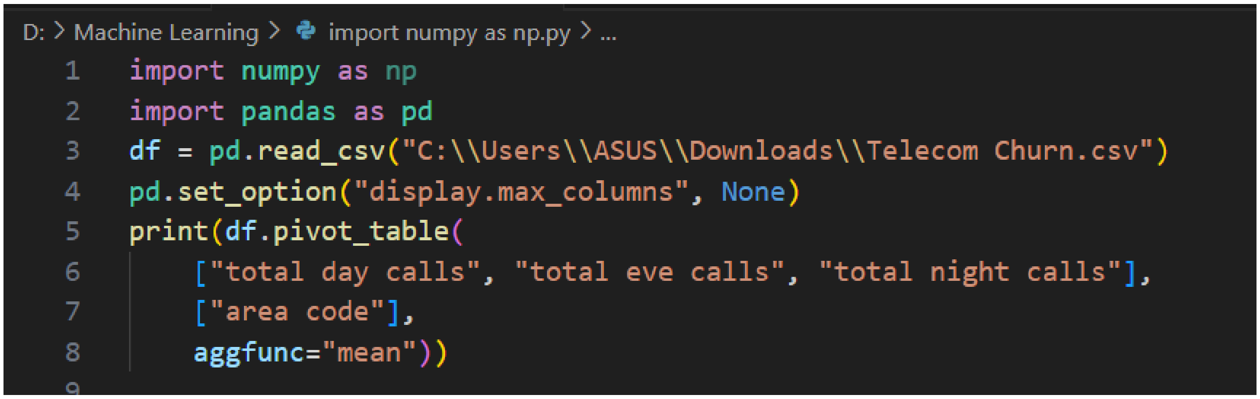


```
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
columns_to_show = ["total day minutes", "total eve minutes", "total night minutes"]
print(df.groupby(["churn"])[columns_to_show].agg([np.mean, np.std, np.min, np.max]))
```



```
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
print(pd.crosstab(df["churn"], df["voice mail plan"], normalize=True))
```

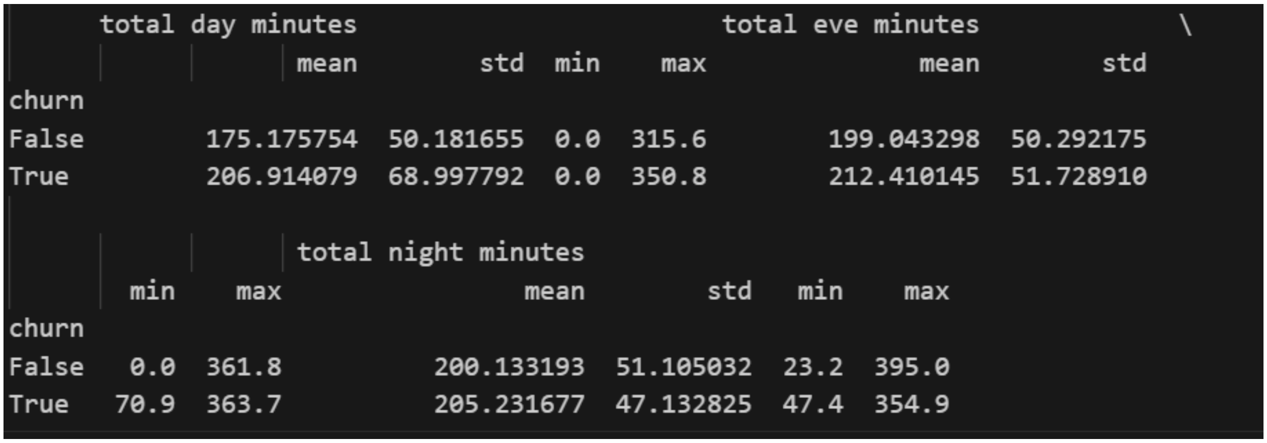
`



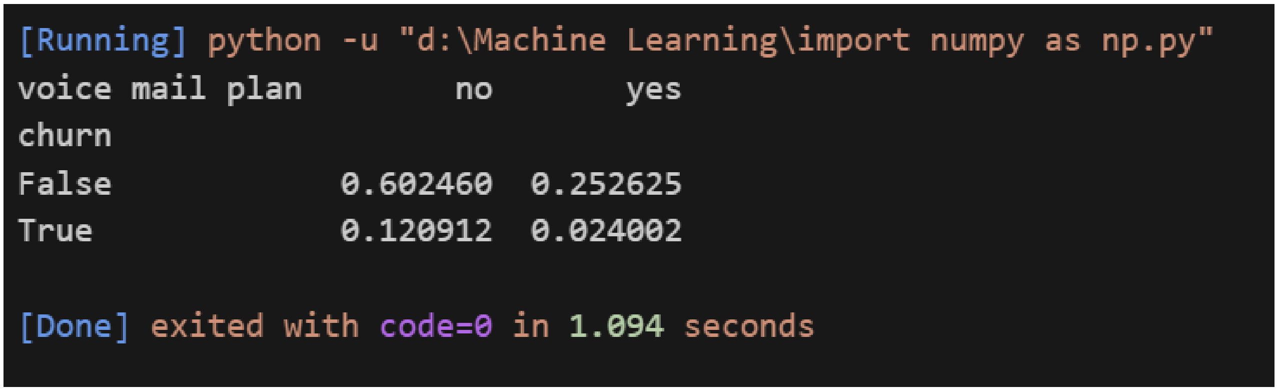
D: > Machine Learning > import numpy as np.py > ...

```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
print(df.pivot_table(
    ["total day calls", "total eve calls", "total night calls"],
    ["area code"],
    aggfunc="mean"))
```

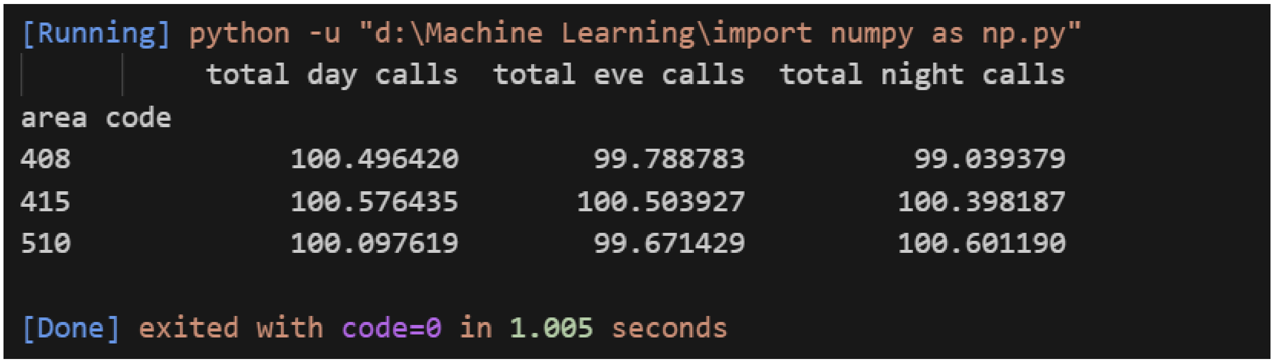
Console outputs:



```
      total day minutes                               total eve minutes             \
                   mean        std  min    max                   mean        std
churn
False        175.175754  50.181655  0.0  315.6             199.043298  50.292175
True         206.914079  68.997792  0.0  350.8             212.410145  51.728910

                   total night minutes
        min    max                 mean        std   min    max
churn
False   0.0  361.8          200.133193  51.105032  23.2  395.0
True   70.9  363.7          205.231677  47.132825  47.4  354.9
```



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
voice mail plan        no       yes
churn
False            0.602460  0.252625
True             0.120912  0.024002

[Done] exited with code=0 in 1.094 seconds
```



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
           total day calls  total eve calls  total night calls
area code
408             100.496420        99.788783          99.039379
415             100.576435       100.503927         100.398187
510             100.097619        99.671429         100.601190

[Done] exited with code=0 in 1.005 seconds
```

`

❖ *DataFrame transformations:*

DataFrame transformations are core to the manipulation of data with Pandas because one is able to reshape, clean, or even modify data in a way that it becomes more applicable to analysis or further processing. Incase of DataFrame transformation we may use to insert a new column by manually using a method. Also, we can add a column more easily without creating an intermediate Series. Incase of deleting columns like unnecessary which may emerge the dataset, we can use drop method. For drop method,

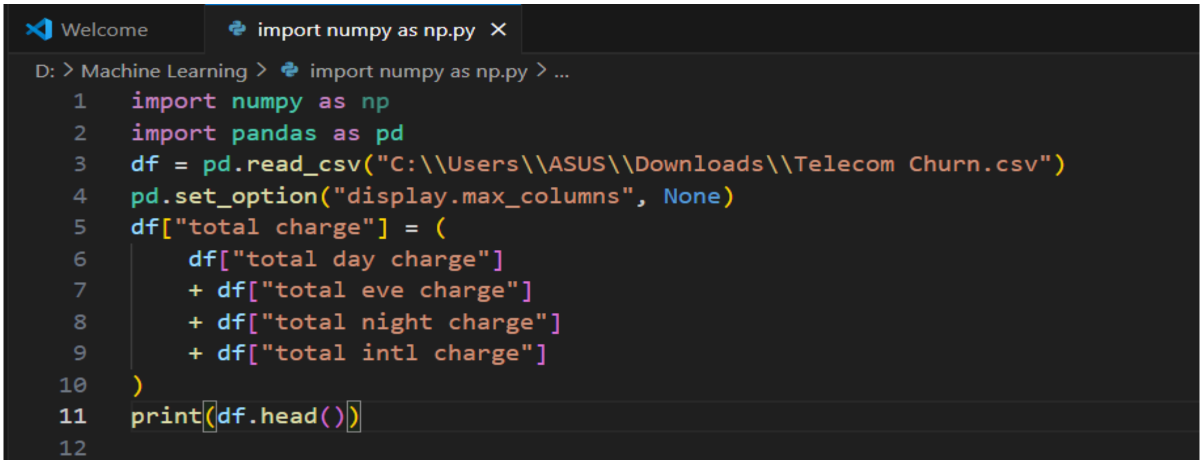
df.drop(["Column_name", "Column_name",...],axis=int, inplace=True)

df.drop([matrix dimension]).head()

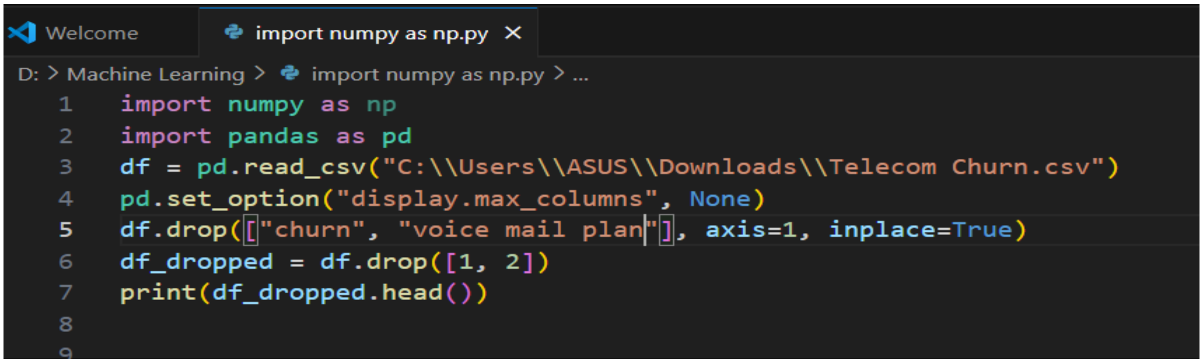
In case of adding a new column in datasheet which is sum of another column's data, we may just write the algorithm using pd library.

df["New variable"] = (df["variable_1"] + df["variable_2"] + df["variable_3"]+ df["variable_4"])

Code snippets:



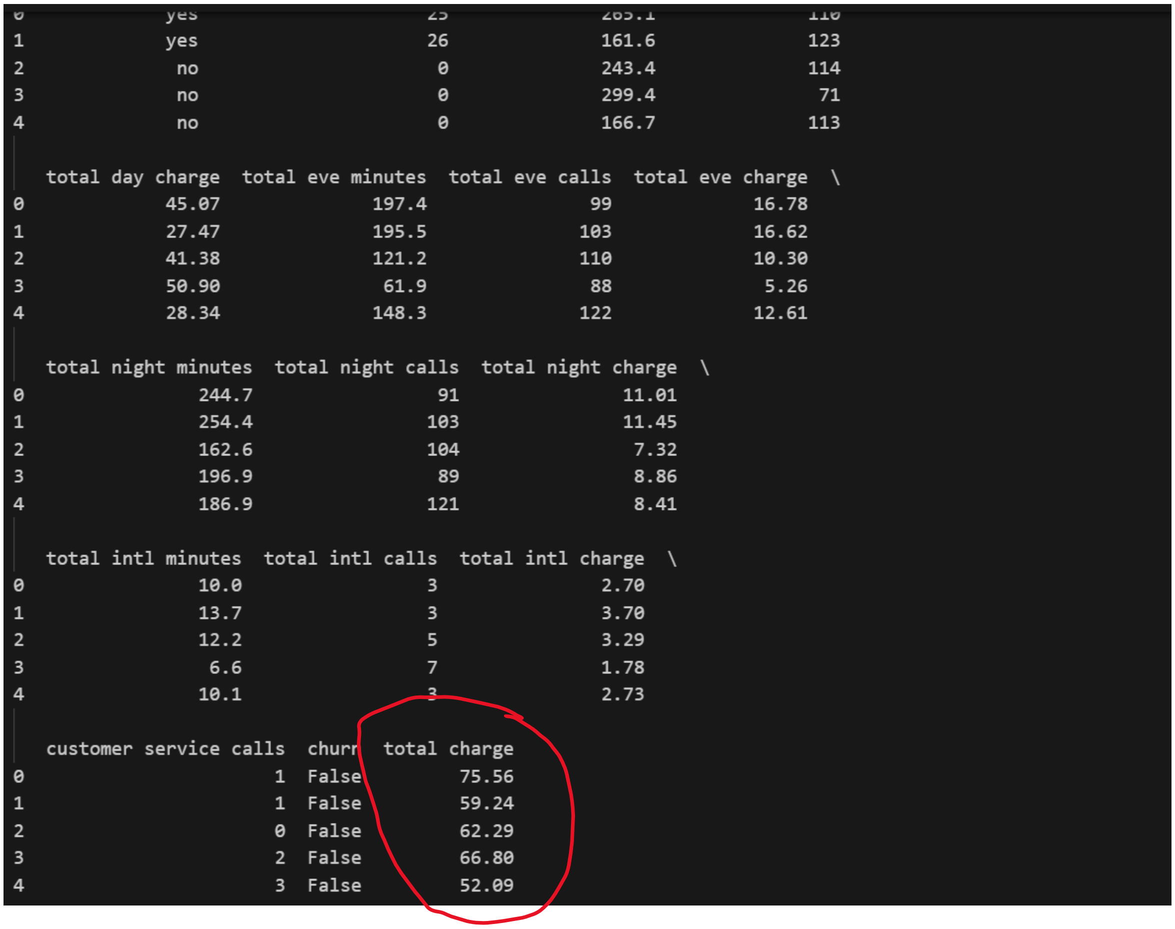
```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
df["total charge"] = (
    df["total day charge"]
    + df["total eve charge"]
    + df["total night charge"]
    + df["total intl charge"]
)
print(df.head())
```



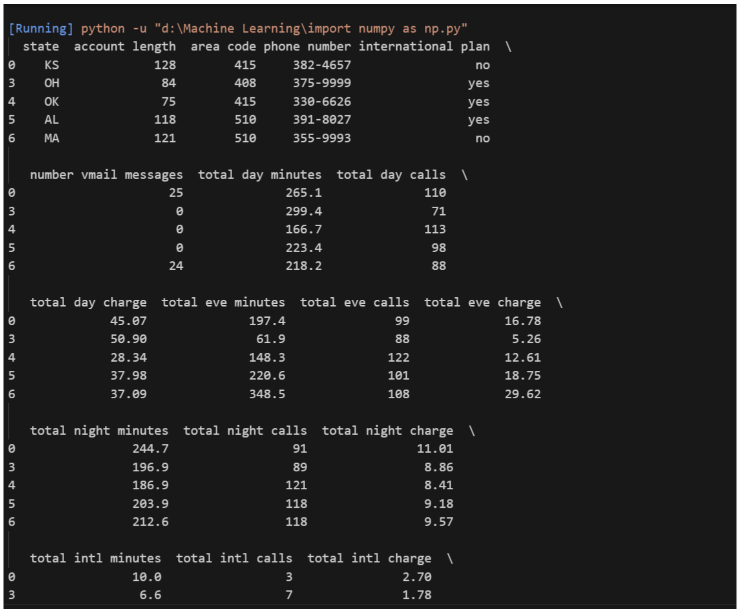
```python
import numpy as np
import pandas as pd
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
pd.set_option("display.max_columns", None)
df.drop(["churn", "voice mail plan"], axis=1, inplace=True)
df_dropped = df.drop([1, 2])
print(df_dropped.head())
```

`

Output console:



```
0           yes                   25           265.1             110
1           yes                   26           161.6             123
2            no                    0           243.4             114
3            no                    0           299.4              71
4            no                    0           166.7             113

   total day charge  total eve minutes  total eve calls  total eve charge  \
0             45.07              197.4               99             16.78
1             27.47              195.5              103             16.62
2             41.38              121.2              110             10.30
3             50.90               61.9               88              5.26
4             28.34              148.3              122             12.61

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
1                254.4                103               11.45
2                162.6                104                7.32
3                196.9                 89                8.86
4                186.9                121                8.41

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70
1                13.7                 3               3.70
2                12.2                 5               3.29
3                 6.6                 7               1.78
4                10.1                 3               2.73

   customer service calls  churn  total charge
0                       1  False         75.56
1                       1  False         59.24
2                       0  False         62.29
3                       2  False         66.80
4                       3  False         52.09
```



```
[Running] python -u "d:\Machine Learning\import numpy as np.py"
  state  account length  area code phone number international plan  \
0    KS             128        415     382-4657                 no
3    OH              84        408     375-9999                yes
4    OK              75        415     330-6626                yes
5    AL             118        510     391-8027                yes
6    MA             121        510     355-9993                 no

   number vmail messages  total day minutes  total day calls  \
0                     25              265.1              110
3                      0              299.4               71
4                      0              166.7              113
5                      0              223.4               98
6                     24              218.2               88

   total day charge  total eve minutes  total eve calls  total eve charge  \
0             45.07              197.4               99             16.78
3             50.90               61.9               88              5.26
4             28.34              148.3              122             12.61
5             37.98              220.6              101             18.75
6             37.09              348.5              108             29.62

   total night minutes  total night calls  total night charge  \
0                244.7                 91               11.01
3                196.9                 89                8.86
4                186.9                121                8.41
5                203.9                118                9.18
6                212.6                118                9.57

   total intl minutes  total intl calls  total intl charge  \
0                10.0                 3               2.70
3                 6.6                 7               1.78
```

Voice plan and churn is deleted

`

Data Visualization with extra Data Analysis Portion

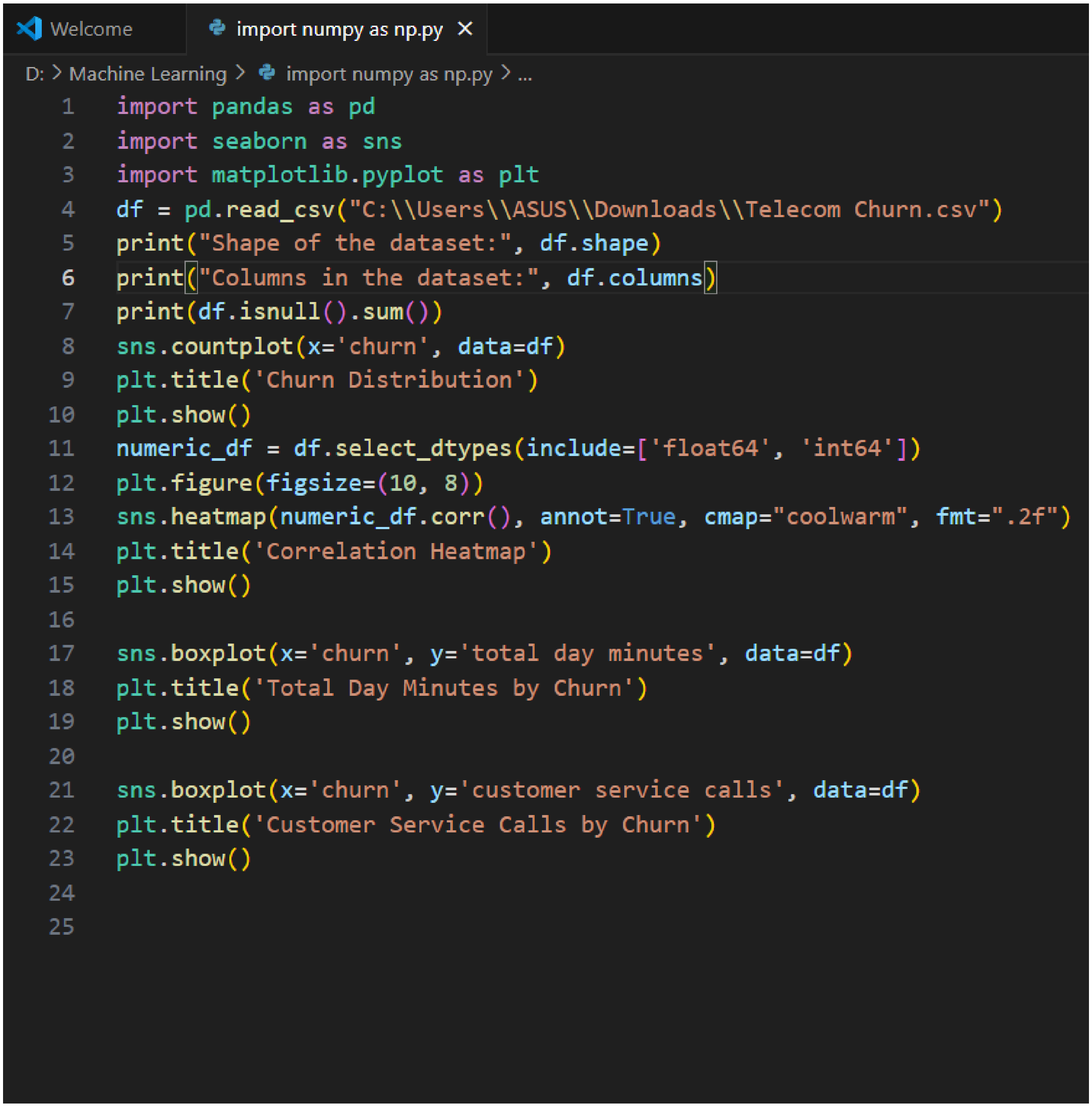
Data visualization is the core data manipulating mechanism of data science. In case of data visualization, we may use different visualizing graph charts. They are – pie-chart or bar chart or line graph or scatter plotting or correlation heatmaps etc. For data visualizing, we only use two python libraries. They are seaborn and matplotlib libraries. In order to gain a clear understanding of data's structure and content, I examined the churn distribution. Investigating possible connections between features, such as the differences in customer service calls or total day minutes between customers who have churned and those who have not. For this, I have made the use of a correlation heatmap which helps quickly to identify relationships between features, aiding feature selection for predictive modeling. Besides, it systematically covers basic information, summary statistics, missing data analysis, and visualizations, ensuring a thorough exploration of the data.

Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv") #Filepath
print("Shape of the dataset:", df.shape)
print("Columns in the dataset:", df.columns)
print(df.isnull().sum())
sns.countplot(x='churn', data=df)
plt.title('Churn Distribution')
plt.show()
numeric_df = df.select_dtypes(include=['float64', 'int64'])# Correlation heatmap is only for numeric data
plt.figure(figsize=(10, 8))
sns.heatmap(numeric_df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
```

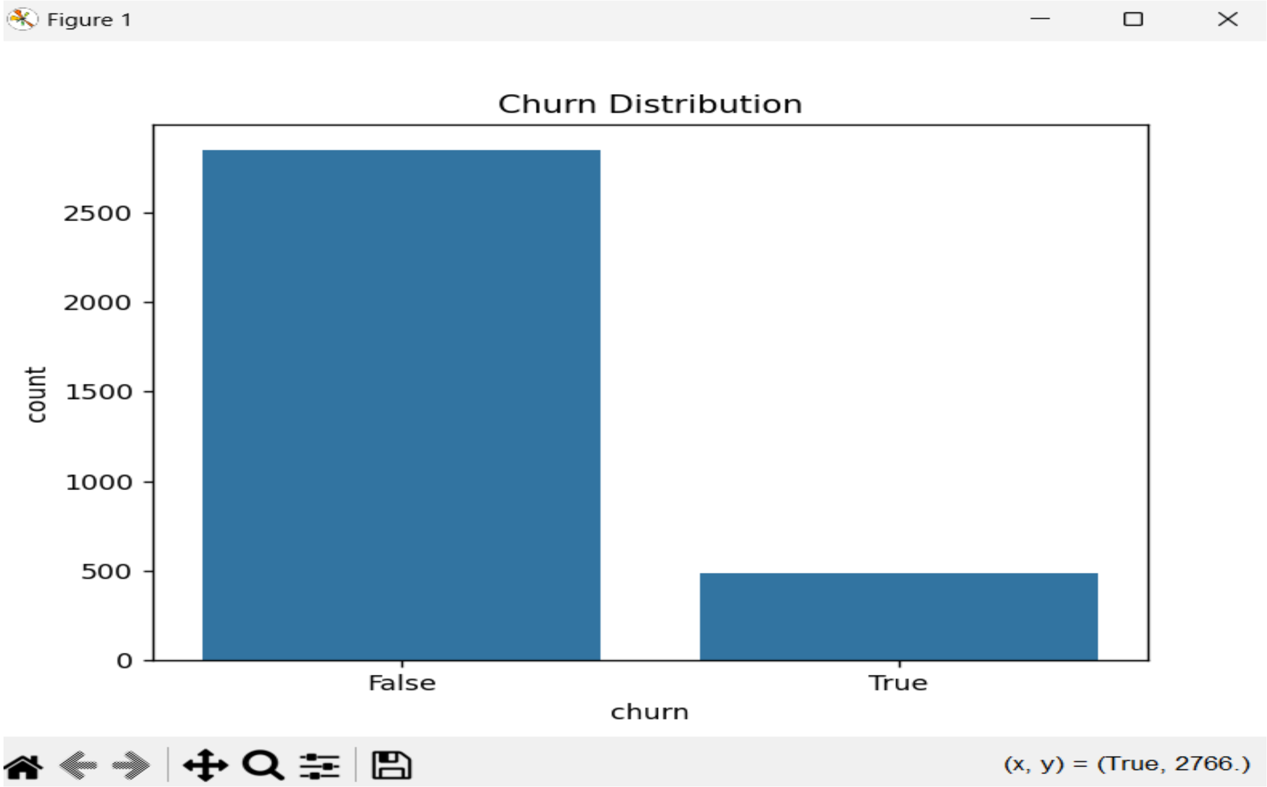
```
`

plt.title('Correlation Heatmap')
plt.show()

sns.boxplot(x='churn', y='total day minutes', data=df)
plt.title('Total Day Minutes by Churn')
plt.show()
sns.boxplot(x='churn', y='customer service calls', data=df)
plt.title('Customer Service Calls by Churn')
plt.show()
```

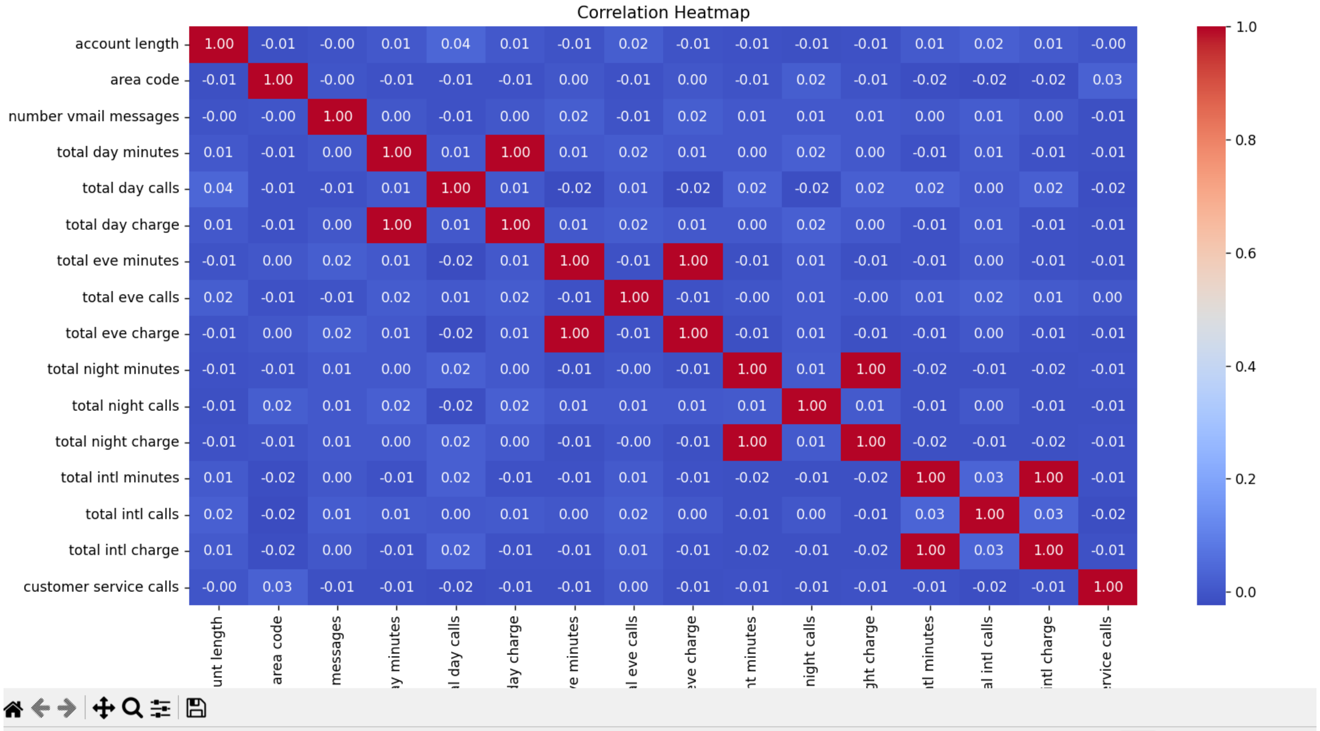
`



```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv("C:\\Users\\ASUS\\Downloads\\Telecom Churn.csv")
print("Shape of the dataset:", df.shape)
print("Columns in the dataset:", df.columns)
print(df.isnull().sum())
sns.countplot(x='churn', data=df)
plt.title('Churn Distribution')
plt.show()
numeric_df = df.select_dtypes(include=['float64', 'int64'])
plt.figure(figsize=(10, 8))
sns.heatmap(numeric_df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

sns.boxplot(x='churn', y='total day minutes', data=df)
plt.title('Total Day Minutes by Churn')
plt.show()

sns.boxplot(x='churn', y='customer service calls', data=df)
plt.title('Customer Service Calls by Churn')
plt.show()
```

`





	unt length	area code	messages	y minutes	l day calls	day charge	e minutes	l eve calls	ve charge	nt minutes	night calls	ght charge	tl minutes	al intl calls	intl charge	rvice calls
account length	1.00	-0.01	-0.00	0.01	0.04	0.01	-0.01	0.02	-0.01	-0.01	-0.01	-0.01	0.01	0.02	0.01	-0.00
area code	-0.01	1.00	-0.00	-0.01	-0.01	-0.01	0.00	-0.01	0.00	-0.01	0.02	-0.01	-0.02	-0.02	-0.02	0.03
number vmail messages	-0.00	-0.00	1.00	0.00	-0.01	0.00	0.02	-0.01	0.02	0.01	0.01	0.01	0.00	0.01	0.00	-0.01
total day minutes	0.01	-0.01	0.00	1.00	0.01	1.00	0.01	0.02	0.01	0.00	0.02	0.00	-0.01	0.01	-0.01	-0.01
total day calls	0.04	-0.01	-0.01	0.01	1.00	0.01	-0.02	0.01	-0.02	0.02	-0.02	0.02	0.02	0.00	0.02	-0.02
total day charge	0.01	-0.01	0.00	1.00	0.01	1.00	0.01	0.02	0.01	0.00	0.02	0.00	-0.01	0.01	-0.01	-0.01
total eve minutes	-0.01	0.00	0.02	0.01	-0.02	0.01	1.00	-0.01	1.00	-0.01	0.01	-0.01	-0.01	0.00	-0.01	-0.01
total eve calls	0.02	-0.01	-0.01	0.02	0.01	0.02	-0.01	1.00	-0.01	-0.00	0.01	-0.00	0.01	0.02	0.01	0.00
total eve charge	-0.01	0.00	0.02	0.01	-0.02	0.01	1.00	-0.01	1.00	-0.01	0.01	-0.01	-0.01	0.00	-0.01	-0.01
total night minutes	-0.01	-0.01	0.01	0.00	0.02	0.00	-0.01	-0.00	-0.01	1.00	0.01	1.00	-0.02	-0.01	-0.02	-0.01
total night calls	-0.01	0.02	0.01	0.02	-0.02	0.02	0.01	0.01	0.01	0.01	1.00	0.01	-0.01	0.00	-0.01	-0.01
total night charge	-0.01	-0.01	0.01	0.00	0.02	0.00	-0.01	-0.00	-0.01	1.00	0.01	1.00	-0.02	-0.01	-0.02	-0.01
total intl minutes	0.01	-0.02	0.00	-0.01	0.02	-0.01	-0.01	0.01	-0.01	-0.02	-0.01	-0.02	1.00	0.03	1.00	-0.01
total intl calls	0.02	-0.02	0.01	0.01	0.00	0.01	0.00	0.02	0.00	-0.01	0.00	-0.01	0.03	1.00	0.03	-0.02
total intl charge	0.01	-0.02	0.00	-0.01	0.02	-0.01	-0.01	0.01	-0.01	-0.02	-0.01	-0.02	1.00	0.03	1.00	-0.01
customer service calls	-0.00	0.03	-0.01	-0.01	-0.02	-0.01	-0.01	0.00	-0.01	-0.01	-0.01	-0.01	-0.01	-0.02	-0.01	1.00

`

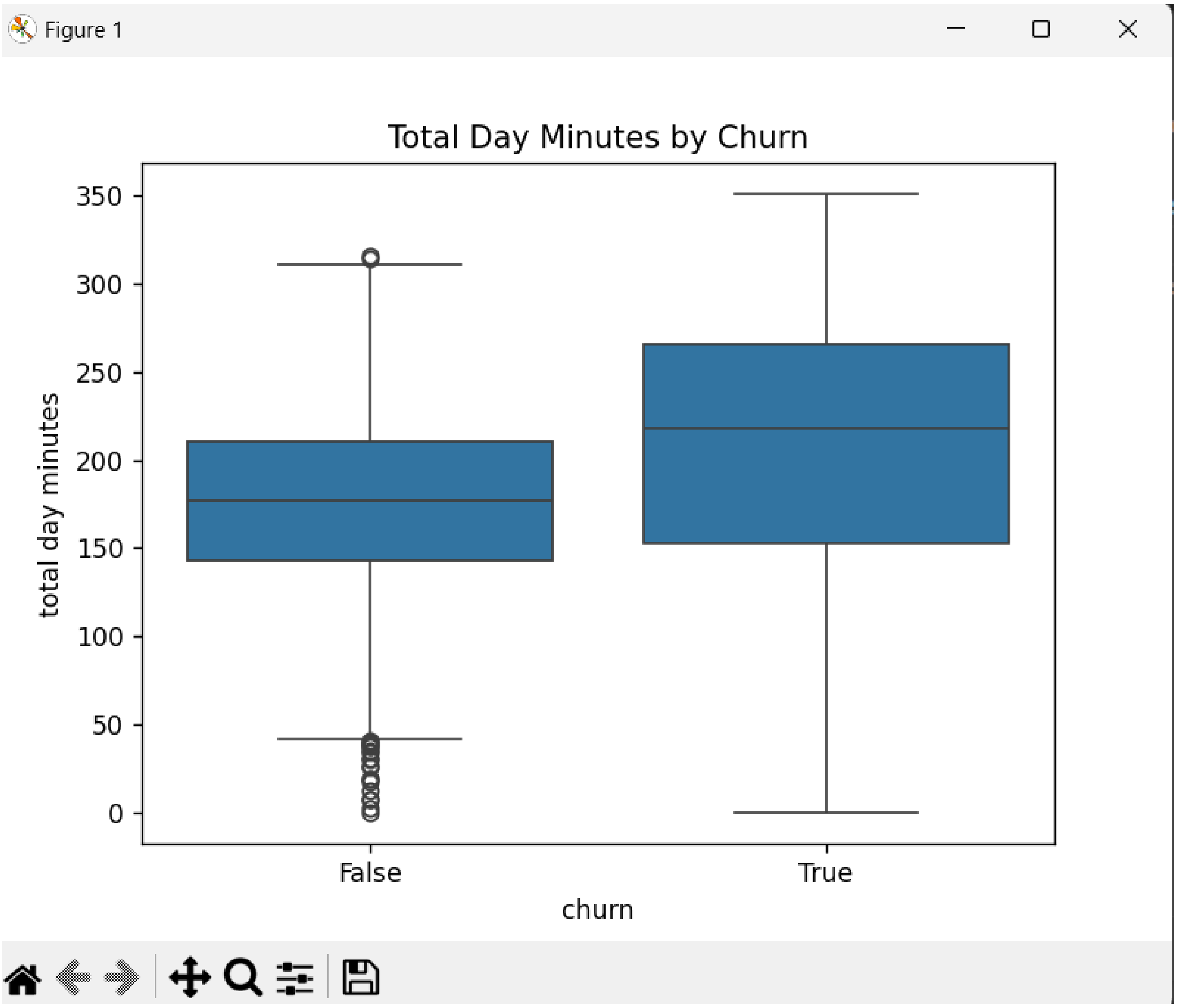


Figure 1
Total Day Minutes by Churn
total day minutes
350
300
250
200
150
100
50
0
False
True
churn

`

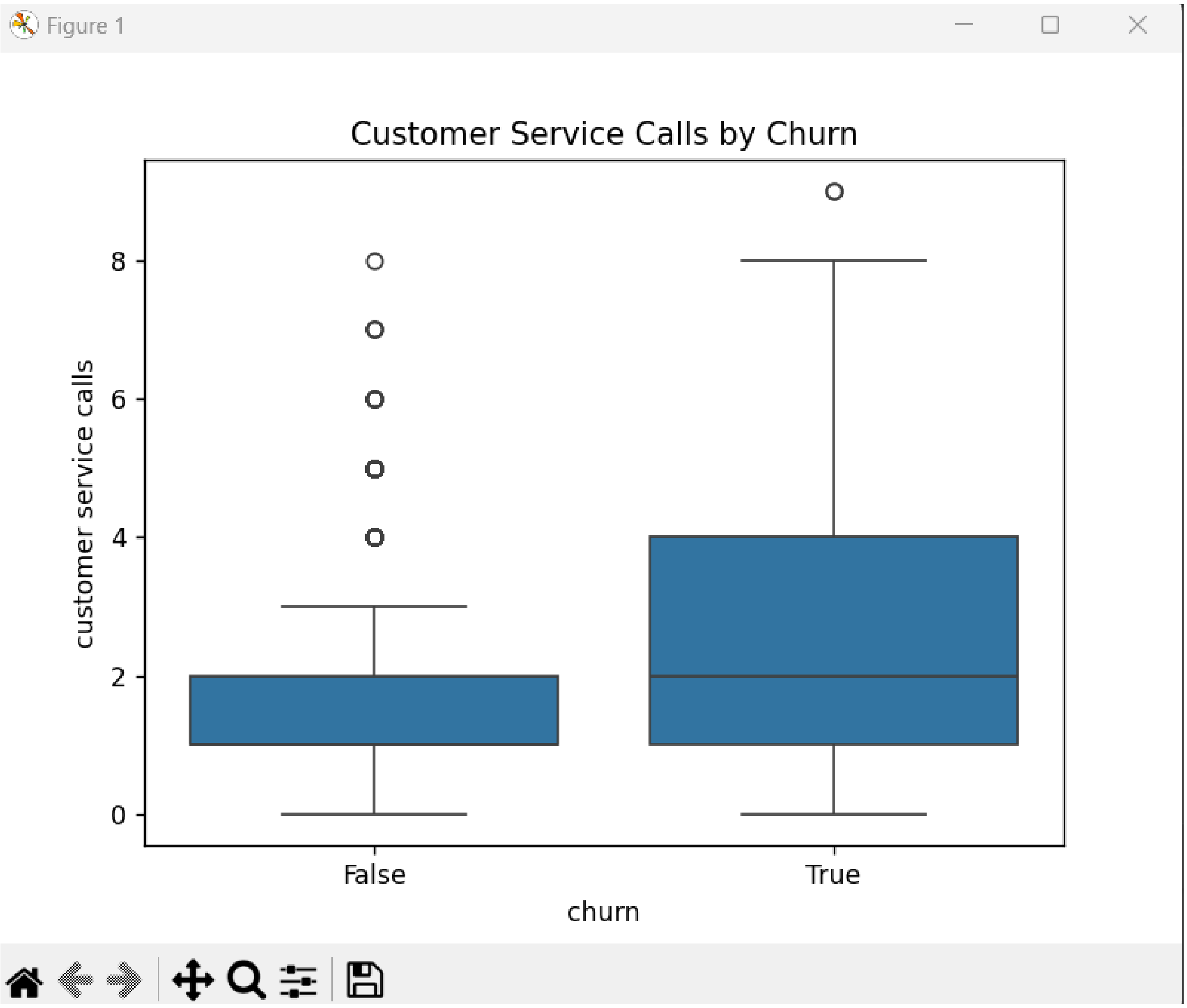


Figure 1
Customer Service Calls by Churn
customer service calls
0
2
4
6
8
False
True
churn