In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
#load the data
filePath = 'C:/Users/HP/Documents/python_one_campus/'
fileName = 'Churn_Train.csv'
```

In [3]:
```
data = pd.read_csv(filePath + fileName)
```

In [ ]:
```
'''Steps to take in cleaning the data
load the data
inspect the data- using description and visualization
replace missing values
transform the data- rescaling, standardization, normalization
visualise results with graphs
'''
```

In [4]:
```
#inspecting the data
data.head(10)
```

Out[4]:

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messag
0	NV	125.0	area_code_510	no	no	0
1	HI	108.0	area_code_415	no	no	0
2	DC	82.0	area_code_415	no	no	0
3	HI	NaN	area_code_408	no	yes	30
4	OH	83.0	area_code_415	no	no	0
5	MO	89.0	area_code_415	no	no	0
6	NC	135.0	area_code_415	no	no	0
7	PA	28.0	area_code_415	no	no	0
8	IA	86.0	area_code_408	no	no	0
9	IN	65.0	area_code_415	no	no	0

In [5]:
```
1 data.describe()
```

Out[5]:

minutes	total_night_calls	total_night_charge	total_intl_minutes	total_intl_calls	total_intl_charge	nun
3.000000	3333.000000	3133.000000	3133.000000	3032.000000	3133.000000	
.198883	100.107711	9.054028	10.226843	4.470317	2.761752	
).430664	19.568609	2.269421	2.805291	2.455364	0.757396	
3.200000	33.000000	1.040000	0.000000	0.000000	0.000000	
7.300000	87.000000	7.530000	8.500000	3.000000	2.300000	
.400000	100.000000	9.060000	10.300000	4.000000	2.780000	
5.300000	113.000000	10.590000	12.100000	6.000000	3.270000	
5.000000	175.000000	17.770000	20.000000	20.000000	5.400000	

In [6]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   state                           3333 non-null   object
 1   account_length                  2832 non-null   float64
 2   area_code                       3333 non-null   object
 3   international_plan              3333 non-null   object
 4   voice_mail_plan                 3333 non-null   object
 5   number_vmail_messages           3133 non-null   float64
 6   total_day_minutes               3133 non-null   float64
 7   total_day_calls                 3133 non-null   float64
 8   total_day_charge                3133 non-null   float64
 9   total_eve_minutes               3032 non-null   float64
 10  total_eve_calls                 3133 non-null   float64
 11  total_eve_charge                3133 non-null   float64
 12  total_night_minutes             3133 non-null   float64
 13  total_night_calls               3333 non-null   int64
 14  total_night_charge              3133 non-null   float64
 15  total_intl_minutes              3133 non-null   float64
 16  total_intl_calls                3032 non-null   float64
 17  total_intl_charge               3133 non-null   float64
 18  number_customer_service_calls   3133 non-null   float64
 19  churn                           3333 non-null   object
dtypes: float64(14), int64(1), object(5)
memory usage: 520.9+ KB
```

In [7]:
```
#identify missing values
data.isna().sum()
```

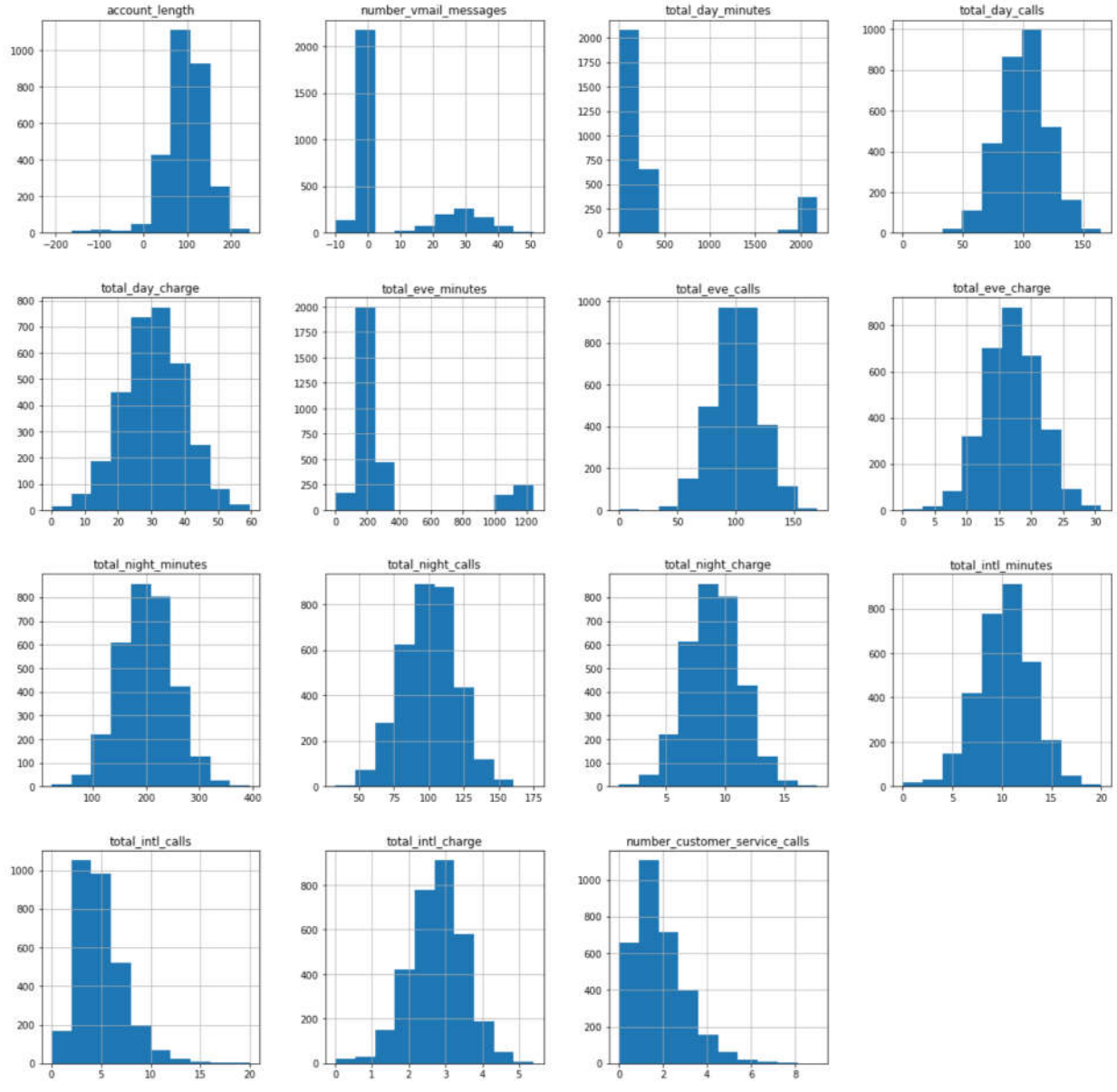
Out[7]:
```
state                           0
account_length                501
area_code                       0
international_plan              0
voice_mail_plan                 0
number_vmail_messages         200
total_day_minutes             200
total_day_calls               200
total_day_charge              200
total_eve_minutes             301
total_eve_calls               200
total_eve_charge              200
total_night_minutes           200
total_night_calls               0
total_night_charge            200
total_intl_minutes            200
total_intl_calls              301
total_intl_charge             200
number_customer_service_calls 200
churn                           0
dtype: int64
```

In [8]:
```
# Skewness for each attribute

skew = data.skew()
print(skew)
```

```
account_length                -1.168594
number_vmail_messages          1.285841
total_day_minutes              2.203279
total_day_calls               -0.125103
total_day_charge              -0.027607
total_eve_minutes              2.089711
total_eve_calls               -0.062598
total_eve_charge               0.002546
total_night_minutes            0.006069
total_night_calls              0.032500
total_night_charge             0.006017
total_intl_minutes            -0.249428
total_intl_calls               1.323692
total_intl_charge             -0.249691
number_customer_service_calls  1.068860
dtype: float64
```

In [9]:

```
#histogram to check the distributuion of the data
import matplotlib.pyplot as plt


data.hist()
plt.gcf().set_size_inches(20,20)
plt.show()
```



In [10]:
```
'''to replace missing values, the distributuion, and description of the data
mean is usually used for a data that is well distributed and progressing pre
skewed when visialised and the mean is close to the median value median is u
if the standard deviation is too high in relation to the mean, if the scale
is changing so much and if it looks skewed when visualised or not well distr
using these conditions, missing values for the numerical variables will be r
'account_length'- median, 'number_vmail_messages'- median, 'total_day_minute
'total_day_calls'- mean, 'total_day_charge'- mean, 'total_eve_minutes'- medi
'total_eve_calls'- mean, 'total_eve_charge'- mean, 'total_night_minutes'-mea
'total_night_charge'- mean, 'total_intl_minutes'- mean, 'total_intl_calls'-
'total_intl_charge- mean', 'number_customer_service_calls'- median'''
```

Out[10]:
```
"to replace missing values, the distributuion, and description of the data will
be considered\nmean is usually used for a data that is well distributed and pro
gressing predictable, if it is not skewed \nwhen visialised and the mean is clo
se to the median value\nmedian is usually used if the standard deviation is too
high in relation to the mean, if the scale of the data\nis changing so much and
if it looks skewed when visualised or not well distributed...\nusing these cond
itions, missing values for the numerical variables will be replaced as follows
\n'account_length'- median, 'number_vmail_messages'- median, 'total_day_minute
s'- median,\n'total_day_calls'- mean, 'total_day_charge'- mean, 'total_eve_minu
tes'- median, \n'total_eve_calls'- mean, 'total_eve_charge'- mean, 'total_night
_minutes'-mean,\n'total_night_calls'- mean, 'total_night_charge'- mean, 'total_
intl_minutes'- mean, 'total_intl_calls'- median,\n'total_intl_charge- mean', 'n
umber_customer_service_calls'- median"
```

In [11]:
```
#replace missing values
data['account_length'] = data['account_length'].fillna(data['account_length'
data['number_vmail_messages'] = data['number_vmail_messages'].fillna(data['n
data['total_day_minutes'] = data['total_day_minutes'].fillna(data['total_day
data['total_day_calls'] = data['total_day_calls'].fillna(data['total_day_cal
data['total_day_charge'] = data['total_day_charge'].fillna(data['total_day_c
data['total_eve_minutes'] = data['total_eve_minutes'].fillna(data['total_eve
data['total_eve_calls'] = data['total_eve_calls'].fillna(data['total_eve_cal
data['total_eve_charge'] = data['total_eve_charge'].fillna(data['total_eve_c
data['total_night_minutes'] = data['total_night_minutes'].fillna(data['total
data['total_night_charge'] = data['total_night_charge'].fillna(data['total_n
data['total_intl_minutes'] = data['total_intl_minutes'].fillna(data['total_i
data['total_intl_calls'] = data['total_intl_calls'].fillna(data['total_intl_
data['total_intl_charge'] = data['total_intl_charge'].fillna(data['total_int
data['number_customer_service_calls'] = data['number_customer_service_calls'
```

In [12]:
```
#new data with the missing values replaced
data.head(10)
```

Out[12]:

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messag
0	NV	125.0	area_code_510	no	no	0
1	HI	108.0	area_code_415	no	no	0
2	DC	82.0	area_code_415	no	no	0
3	HI	100.0	area_code_408	no	yes	30
4	OH	83.0	area_code_415	no	no	0
5	MO	89.0	area_code_415	no	no	0
6	NC	135.0	area_code_415	no	no	0
7	PA	28.0	area_code_415	no	no	0
8	IA	86.0	area_code_408	no	no	0
9	IN	65.0	area_code_415	no	no	0

In [13]:
```
data.isna().sum()
```

Out[13]:
```
state                          0
account_length                 0
area_code                      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
number_customer_service_calls  0
churn                          0
dtype: int64
```

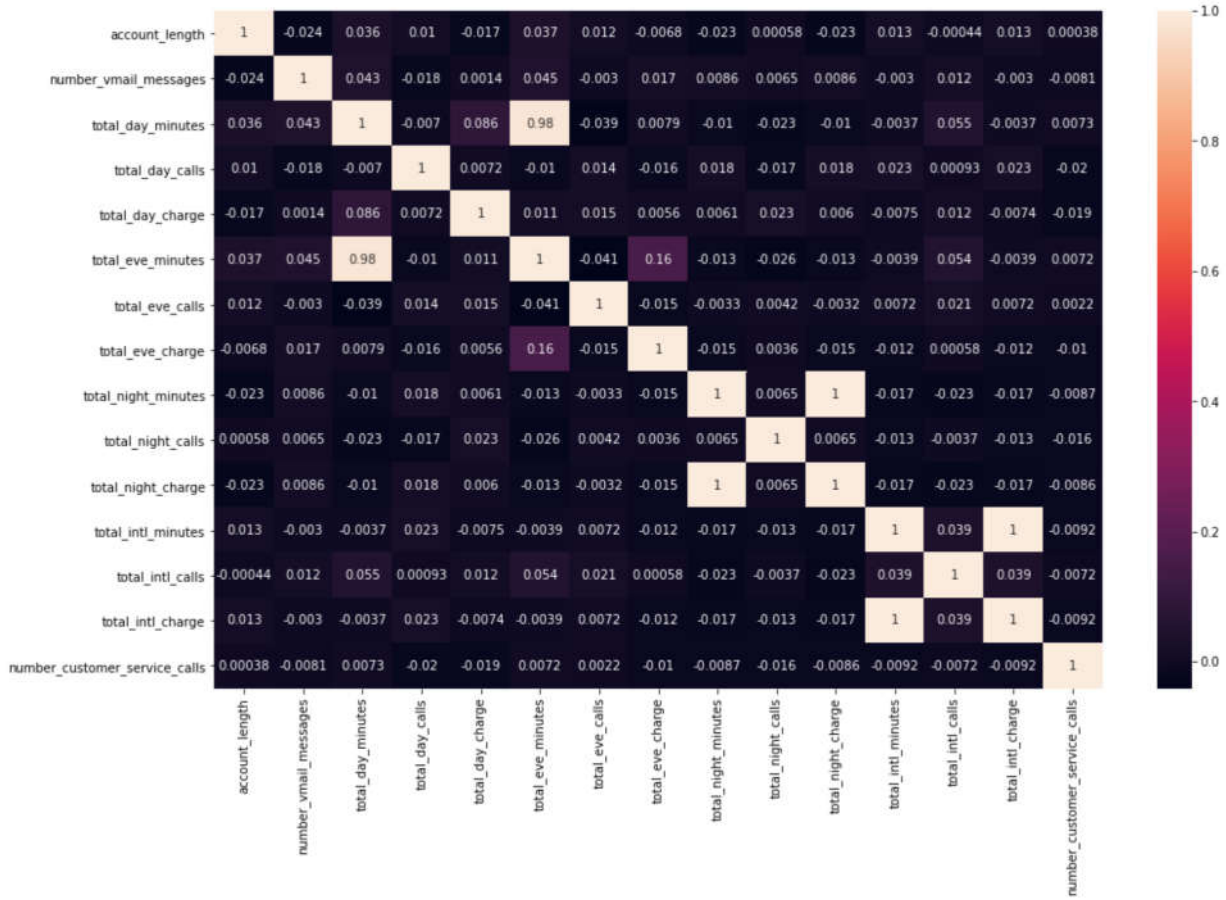
In [14]:
```
1 data.describe()
```

Out[14]:

	account_length	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charg
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.00000
mean	97.723972	6.892589	405.238854	100.331631	30.62845
std	44.139070	13.450072	609.646965	19.428637	8.99306
min	-209.000000	-10.000000	0.000000	0.000000	0.00000
25%	77.000000	0.000000	151.800000	88.000000	24.89000
50%	100.000000	0.000000	190.500000	100.331631	30.62845
75%	122.000000	0.000000	234.200000	113.000000	36.43000
max	243.000000	51.000000	2185.100000	165.000000	59.64000

In [15]:
```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import set_option
data = data
set_option('display.width', 100)

plt.figure(figsize=(16,10))

sns.heatmap(data.corr(), annot = True)
```

Out[15]: <AxesSubplot:>



In [16]:
```
'''After replacing missing values, data tranformation is done. But some coul
data and it has to be changed to numerical data before it can be transformed
So encoding has to be done'''

#categorical data: 'state', 'area_code', 'international_plan', 'voice_mail_p
#to encode to numerical data

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
# Assigning numerical values and storing in another column


data['state'] = le.fit_transform(data['state'])
data['area_code'] =le.fit_transform(data['area_code'])
data['international_plan'] =le.fit_transform(data['international_plan'])
data['voice_mail_plan'] = le.fit_transform(data['voice_mail_plan'])
data['churn'] = le.fit_transform(data['churn'])
```

In [17]:
```
#encoded data
data.head(10)
```

Out[17]:

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messages
0	33	125.0	2	0	0	0.0
1	11	108.0	1	0	0	0.0
2	7	82.0	1	0	0	0.0
3	11	100.0	0	0	1	30.0
4	35	83.0	1	0	0	0.0
5	24	89.0	1	0	0	0.0
6	27	135.0	1	0	0	0.0
7	38	28.0	1	0	0	0.0
8	12	86.0	0	0	0	0.0
9	15	65.0	1	0	0	0.0

In [18]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   state                           3333 non-null   int32  
 1   account_length                  3333 non-null   float64
 2   area_code                       3333 non-null   int32  
 3   international_plan              3333 non-null   int32  
 4   voice_mail_plan                 3333 non-null   int32  
 5   number_vmail_messages           3333 non-null   float64
 6   total_day_minutes               3333 non-null   float64
 7   total_day_calls                 3333 non-null   float64
 8   total_day_charge                3333 non-null   float64
 9   total_eve_minutes               3333 non-null   float64
 10  total_eve_calls                 3333 non-null   float64
 11  total_eve_charge                3333 non-null   float64
 12  total_night_minutes             3333 non-null   float64
 13  total_night_calls               3333 non-null   int64  
 14  total_night_charge              3333 non-null   float64
 15  total_intl_minutes              3333 non-null   float64
 16  total_intl_calls                3333 non-null   float64
 17  total_intl_charge               3333 non-null   float64
 18  number_customer_service_calls   3333 non-null   float64
 19  churn                           3333 non-null   int32  
dtypes: float64(14), int32(5), int64(1)
memory usage: 455.8 KB
```

In [19]:
```
1 len(data.columns)
```

Out[19]:
```
20
```

In [20]:

```
'''The data is comprised of attributes with varying scales and this can impa
or models it will be used for. Rescaling needs to be done for this data to f

# Rescale data (between -1 and 1)
from pandas import read_csv
from numpy import set_printoptions
from sklearn.preprocessing import MinMaxScaler

dataframe = data
array = data.values
# separate array into input and output components
#X is the predictor variable
#Y is the outcome variable
X = array[:,0:19]
Y = array[:,19]
demo = MinMaxScaler(feature_range=(-1, 1))
rescaledX = demo.fit_transform(X)
# summarize transformed data
set_printoptions(precision=3)

print(rescaledX[0:5,:])
```

```
[[ 0.32   0.478  1.    -1.    -1.    -0.672  0.843  0.2   -0.039  0.78
   0.259 -0.034  0.184 -0.169  0.185  0.09   -0.3    0.089 -1.    ]
 [-0.56   0.403  0.    -1.    -1.    -0.672 -0.733  0.2    0.662 -0.645
   0.094  0.216  0.108  0.085  0.108  0.4    -0.1    0.4   -0.556]
 [-0.72   0.288  0.    -1.    -1.    -0.672 -0.725  0.321  0.712 -0.709
   0.176 -0.004  0.328 -0.437  0.328  0.17   -0.6    0.17  -1.    ]
 [-0.56   0.367 -1.    -1.     1.     0.311 -0.899 -0.139 -0.371 -0.707
   0.271  0.003 -0.136 -0.225 -0.136  0.1    -0.2    0.1   -0.556]
 [ 0.4    0.292  0.    -1.    -1.    -0.672 -0.691  0.455  0.924 -0.634
   0.365  0.251 -0.297  0.141 -0.296  0.58   -0.3    0.581 -1.    ]]
```

In [21]:
```
rescaledXdf = pd.DataFrame(rescaledX)
rescaledXdf.columns = ['state', 'account_length', 'area_code', 'internationa
       'number_vmail_messages', 'total_day_minutes', 'total_day_calls', 'tot
       'total_eve_minutes', 'total_eve_calls', 'total_eve_charge', 'total_ni
       'total_night_calls', 'total_night_charge', 'total_intl_minutes', 'tot
       'total_intl_charge', 'number_customer_service_calls']
rescaledXdf['churn'] = data['churn']
rescaledXdf
```

Out[21]:

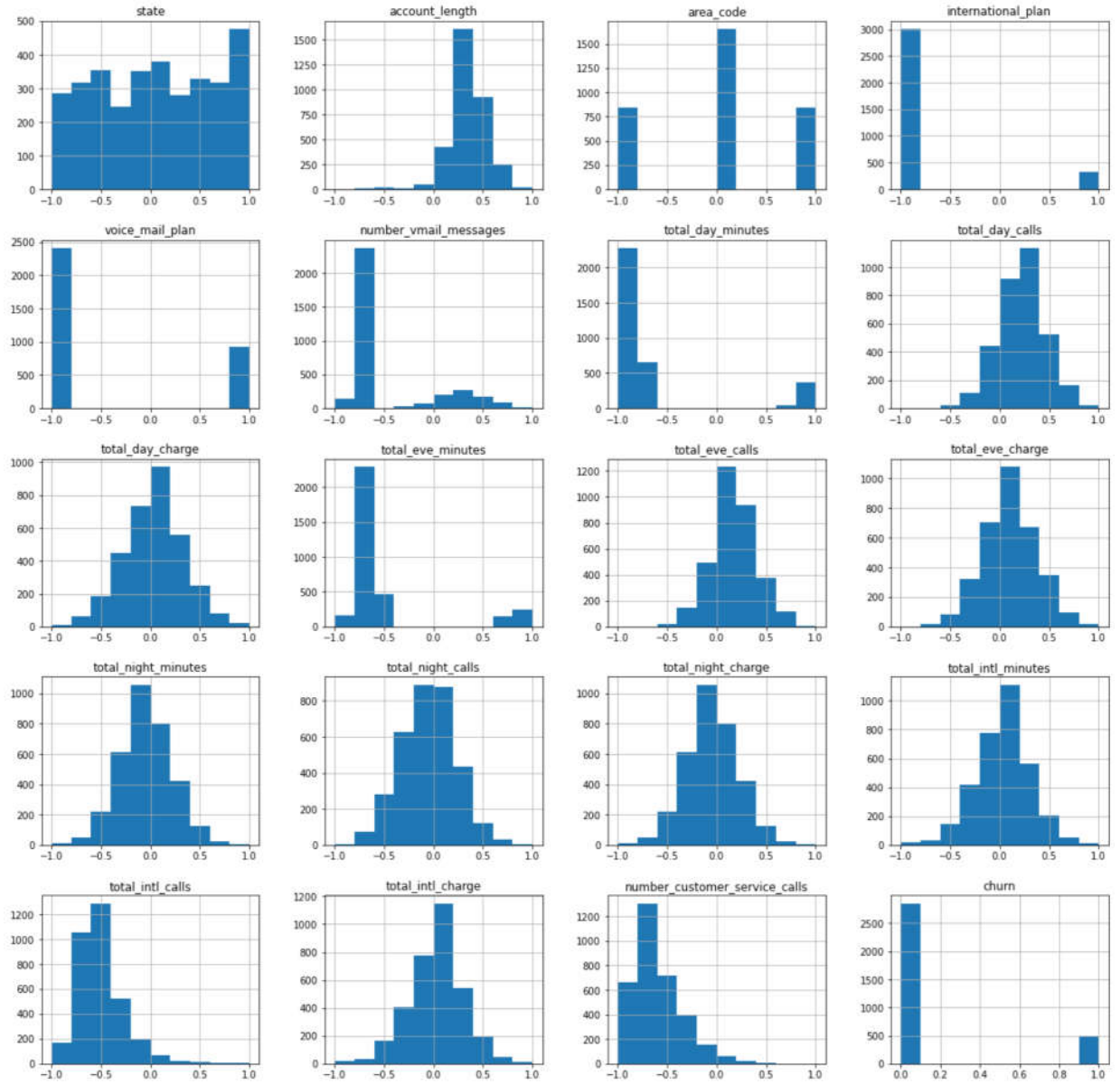
alls	total_night_charge	total_intl_minutes	total_intl_calls	total_intl_charge	number_customer_service_
014	0.184698	0.09	-0.3	0.088889	-1.00
507	0.108189	0.40	-0.1	0.400000	-0.55
520	0.328153	0.17	-0.6	0.170370	-1.00
352	-0.135684	0.10	-0.2	0.100000	-0.55
345	-0.295876	0.58	-0.3	0.581481	-1.00
...	...	...	...	...	
592	-0.002989	0.01	-0.5	0.011111	-0.77
099	-0.350867	-0.41	-0.6	-0.411111	-0.77
268	-0.160789	0.07	-0.8	0.070370	-0.55
169	0.141662	-0.09	-0.5	-0.088889	-0.55
014	-0.513449	-0.10	-0.7	-0.100000	-0.33

In [22]:
```
rescaledXdf.describe()
```

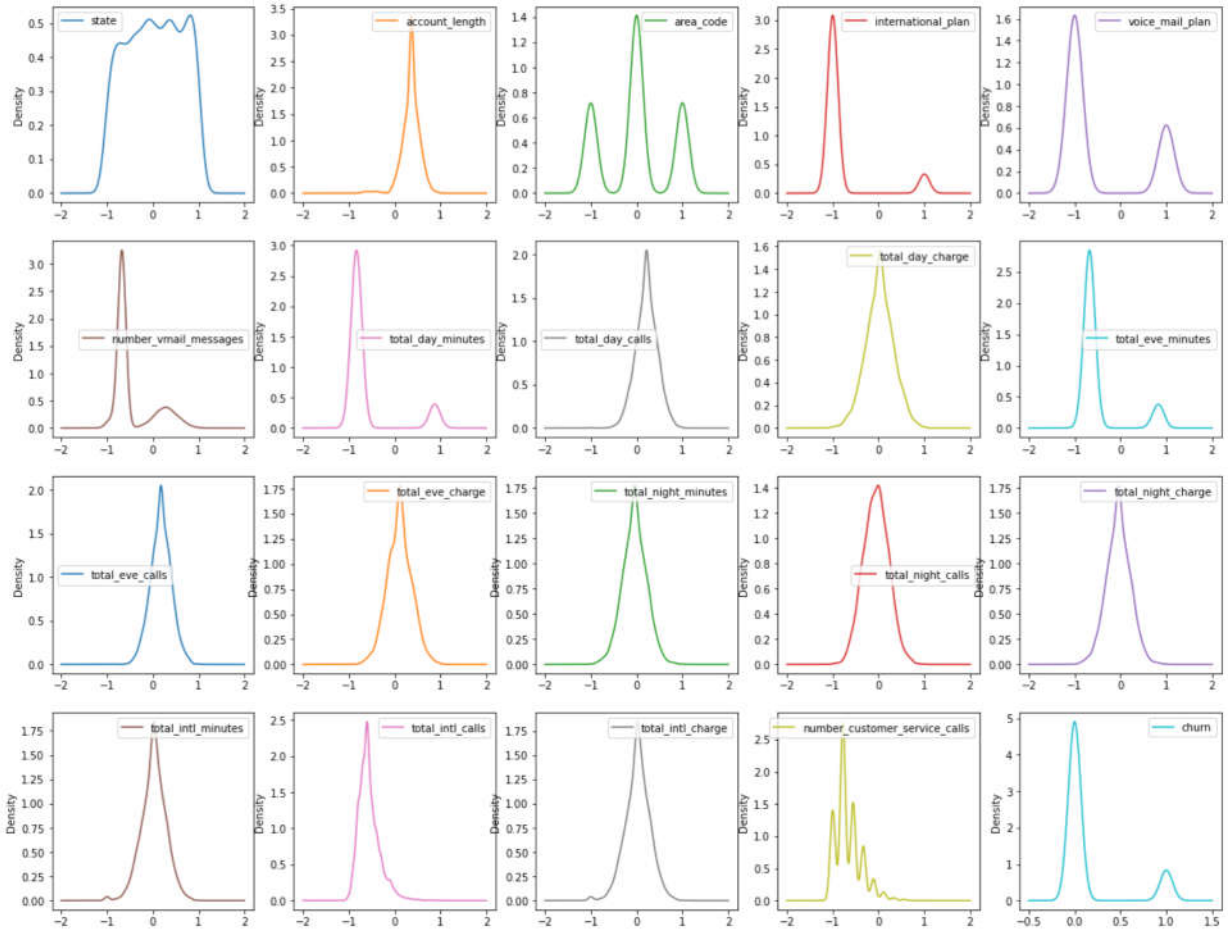
Out[22]:

_night_charge	total_intl_minutes	total_intl_calls	total_intl_charge	number_customer_service_calls	
3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	333
-0.041957	0.022684	-0.557216	0.022871	-0.660499	
0.263031	0.271980	0.234571	0.271968	0.285005	
-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
-0.213389	-0.140000	-0.700000	-0.140741	-0.777778	
-0.041957	0.022684	-0.600000	0.022871	-0.777778	
0.127316	0.190000	-0.500000	0.188889	-0.555556	
1.000000	1.000000	1.000000	1.000000	1.000000	

In [25]:
```
#Univariate Histograms to show the data distribution
import matplotlib.pyplot as plt


rescaledXdf.hist()
plt.gcf().set_size_inches(20,20)
plt.show()
```



In [26]:
```
#density plots
from matplotlib import pyplot
from pandas import read_csv



rescaledXdf.plot(kind='density', subplots=True, layout=(5,5), sharex=False)
plt.gcf().set_size_inches(20,20)
pyplot.show()
```



In [27]:
```
data = rescaledXdf
```

In [28]:
```
'''Standardization is another scaling technique that needs to be done so tha
centered around the mean with a unit (1) standard deviation'''
#standardization
# Standardize data (0 mean, 1 stdev)
from sklearn.preprocessing import StandardScaler
from pandas import read_csv
from numpy import set_printoptions

array = data.values
# separate array into input and output components
X = array[:,0:19]
Y = array[:,19]
scaler = StandardScaler().fit(X)
standardX = scaler.transform(X)
# summarize transformed data
set_printoptions(precision=3)
print(standardX[0:5,:])
```

```
[[ 4.682e-01  6.180e-01  1.409e+00 -3.276e-01 -6.184e-01 -5.125e-01
   2.638e+00 -6.855e-02 -2.189e-01  2.585e+00  3.562e-01 -5.173e-01
   8.612e-01 -4.144e-01  8.618e-01  2.475e-01  1.097e+00  2.428e-01
  -1.191e+00]
 [-1.016e+00  2.328e-01 -8.457e-04 -3.276e-01 -6.184e-01 -5.125e-01
  -1.864e-01 -6.855e-02  2.107e+00 -3.023e-01 -3.695e-01  4.097e-01
   5.728e-01  5.056e-01  5.709e-01  1.388e+00  1.949e+00  1.387e+00
   3.683e-01]
 [-1.286e+00 -3.563e-01 -8.457e-04 -3.276e-01 -6.184e-01 -5.125e-01
  -1.722e-01  4.462e-01  2.271e+00 -4.329e-01 -6.668e-03 -4.068e-01
   1.409e+00 -1.385e+00  1.407e+00  5.417e-01 -1.824e-01  5.424e-01
  -1.191e+00]
 [-1.016e+00  5.157e-02 -1.410e+00 -3.276e-01  1.617e+00  1.718e+00
  -4.839e-01 -1.510e+00 -1.321e+00 -4.284e-01  4.080e-01 -3.804e-01
  -3.559e-01 -6.188e-01 -3.564e-01  2.843e-01  1.523e+00  2.836e-01
   3.683e-01]
 [ 6.032e-01 -3.336e-01 -8.457e-04 -3.276e-01 -6.184e-01 -5.125e-01
  -1.113e-01  1.012e+00  2.973e+00 -2.818e-01  8.228e-01  5.394e-01
  -9.675e-01  7.100e-01 -9.655e-01  2.049e+00  1.097e+00  2.054e+00
  -1.191e+00]]
```

In [29]:
```
standardXdf = pd.DataFrame(standardX)
standardXdf.columns = ['state', 'account_length', 'area_code', 'internationa
       'number_vmail_messages', 'total_day_minutes', 'total_day_calls', 'tot
       'total_eve_minutes', 'total_eve_calls', 'total_eve_charge', 'total_ni
       'total_night_calls', 'total_night_charge', 'total_intl_minutes', 'tot
       'total_intl_charge', 'number_customer_service_calls']
standardXdf['churn'] = data['churn']
standardXdf
```

Out[29]:

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_mes
0	0.468241	0.618049	1.408514	-0.32758	-0.618396	-0.5
1	-1.015970	0.232845	-0.000846	-0.32758	-0.618396	-0.5
2	-1.285827	-0.356291	-0.000846	-0.32758	-0.618396	-0.5
3	-1.015970	0.051573	-1.410205	-0.32758	1.617086	1.7
4	0.603170	-0.333631	-0.000846	-0.32758	-0.618396	-0.5
...	...	...	...	...	...	
3328	0.603170	1.048572	-0.000846	-0.32758	1.617086	0.8
3329	-0.543721	-0.650858	-0.000846	-0.32758	1.617086	2.2
3330	1.007955	0.051573	-0.000846	-0.32758	-0.618396	-0.5
3331	0.535705	-1.330630	-1.410205	-0.32758	-0.618396	-0.5
3332	1.480204	-1.285312	-1.410205	-0.32758	-0.618396	-0.5

3333 rows × 20 columns

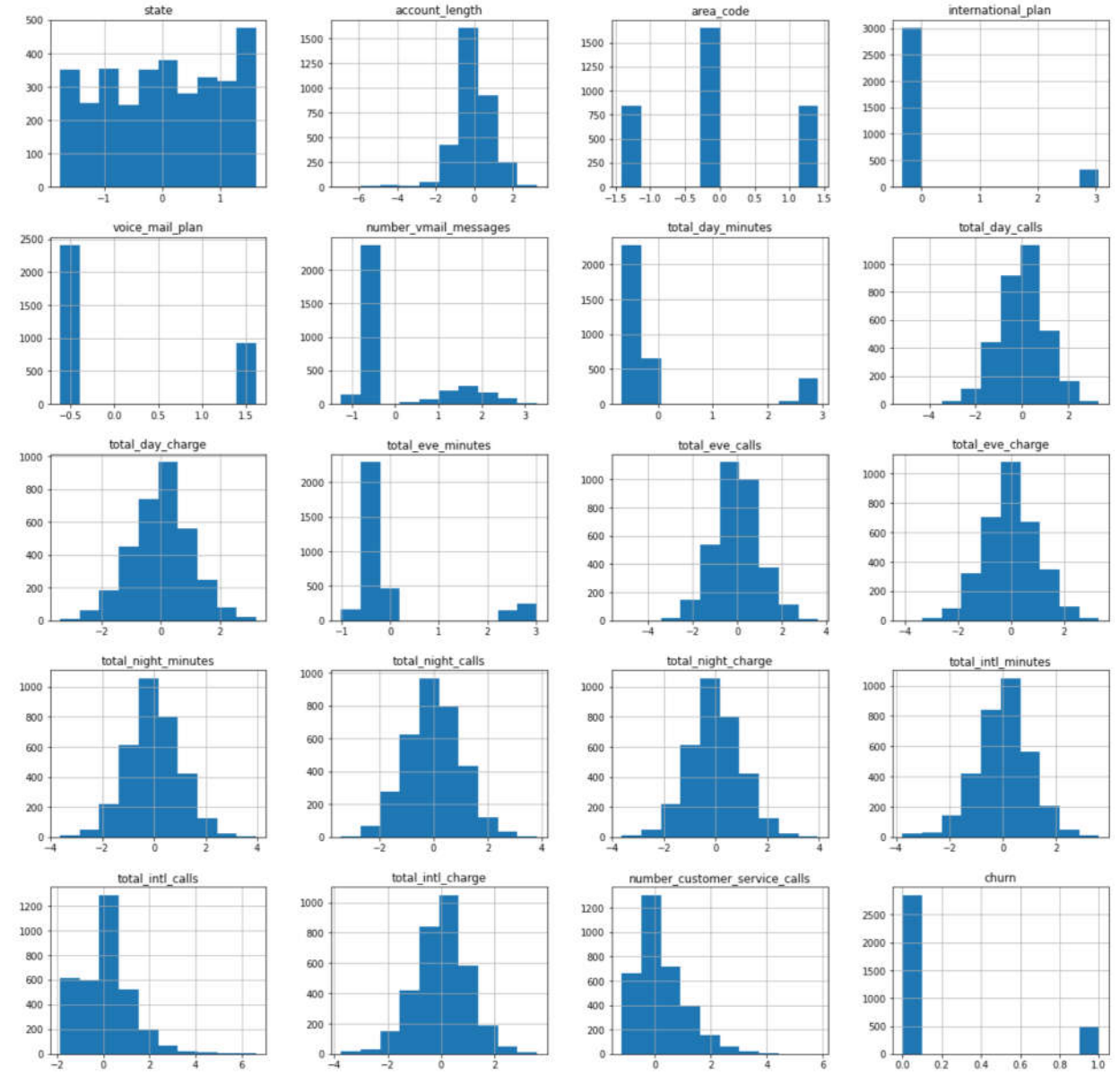
In [30]:
```
standardXdf.describe()
```

Out[30]:

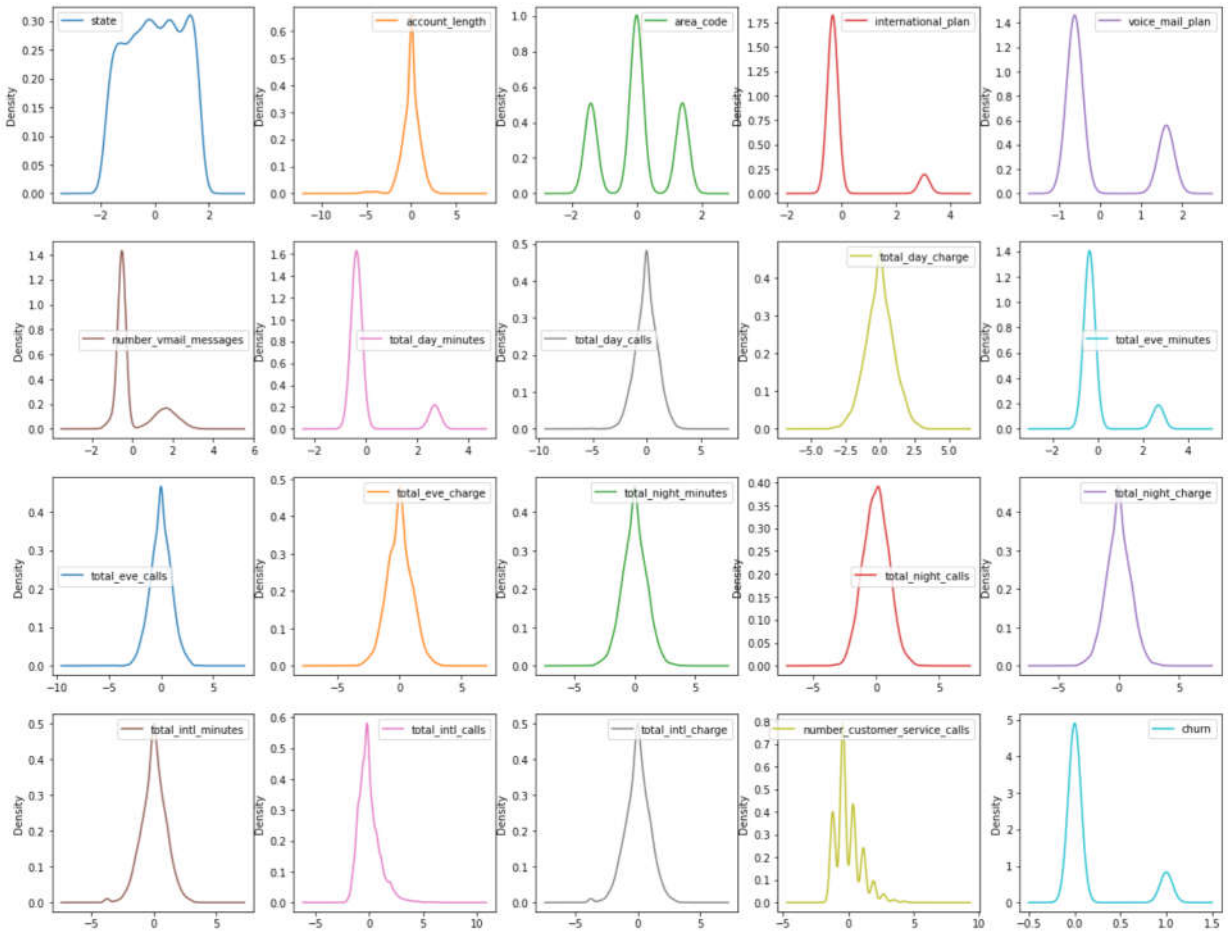
al_night_charge	total_intl_minutes	total_intl_calls	total_intl_charge	number_customer_service_calls
3.333000e+03	3.333000e+03	3.333000e+03	3.333000e+03	3.333000e+03
3.291030e-17	-4.613438e-17	-1.183505e-16	1.597549e-16	-3.650778e-17
1.000150e+00	1.000150e+00	1.000150e+00	1.000150e+00	1.000150e+00
-3.642861e+00	-3.760715e+00	-1.887915e+00	-3.761562e+00	-1.191388e+00
-6.518525e-01	-5.982387e-01	-6.087944e-01	-6.016755e-01	-4.115578e-01
-6.437803e-15	-4.452608e-15	-1.824209e-01	-1.772188e-14	-4.115578e-01
6.436449e-01	6.152696e-01	2.439527e-01	6.105223e-01	3.682729e-01
3.961937e+00	3.593881e+00	6.639556e+00	3.593346e+00	5.827088e+00

In [31]:

```
#Univariate Histograms to show the data distribution after standardization
import matplotlib.pyplot as plt


standardXdf.hist()
plt.gcf().set_size_inches(20,20)
plt.show()
```



In [32]:
```
#density plots
from matplotlib import pyplot
from pandas import read_csv



standardXdf.plot(kind='density', subplots=True, layout=(5,5), sharex=False)
plt.gcf().set_size_inches(20,20)
pyplot.show()
```



In [33]:
```
data = standardXdf
```

In [34]:

```
'''Normalization is another scaling technique that needs to be done so that
same range of Gaussian distribution and to rescale each row to have a length
# Normalize data (length of 1)
from sklearn.preprocessing import Normalizer
from pandas import read_csv
from numpy import set_printoptions

# separate array into input and output components
array = data.values
X = array[:,0:19]
Y = array[:,19]
scaler = Normalizer().fit(X)
normalizedX = scaler.transform(X)
# summarize transformed data
set_printoptions(precision=5)
print(normalizedX[0:5,:])
```

```
[[ 1.00228e-01  1.32294e-01  3.01495e-01 -7.01192e-02 -1.32369e-01
  -1.09709e-01  5.64722e-01 -1.46732e-02 -4.68599e-02  5.53309e-01
   7.62463e-02 -1.10725e-01  1.84342e-01 -8.86997e-02  1.84477e-01
   5.29863e-02  2.34750e-01  5.19668e-02 -2.55019e-01]
 [-2.58844e-01  5.93231e-02 -2.15463e-04 -8.34592e-02 -1.57552e-01
  -1.30581e-01 -4.74974e-02 -1.74648e-02  5.36697e-01 -7.70301e-02
  -9.41498e-02  1.04387e-01  1.45930e-01  1.28813e-01  1.45455e-01
   3.53501e-01  4.96670e-01  3.53340e-01  9.38266e-02]
 [-3.19472e-01 -8.85226e-02 -2.10120e-04 -8.13894e-02 -1.53644e-01
  -1.27342e-01 -4.27733e-02  1.10869e-01  5.64282e-01 -1.07569e-01
  -1.65673e-03 -1.01075e-01  3.50178e-01 -3.44229e-01  3.49654e-01
   1.34595e-01 -4.53236e-02  1.34768e-01 -2.96008e-01]
 [-2.48572e-01  1.26180e-02 -3.45027e-01 -8.01473e-02  3.95643e-01
   4.20400e-01 -1.18383e-01 -3.69428e-01 -3.23213e-01 -1.04812e-01
   9.98339e-02 -9.30688e-02 -8.70772e-02 -1.51404e-01 -8.71955e-02
   6.95612e-02  3.72642e-01  6.93960e-02  9.01033e-02]
 [ 1.19293e-01 -6.59845e-02 -1.67260e-04 -6.47877e-02 -1.22304e-01
  -1.01367e-01 -2.20110e-02  2.00247e-01  5.87972e-01 -5.57388e-02
   1.62722e-01  1.06682e-01 -1.91354e-01  1.40428e-01 -1.90953e-01
   4.05326e-01  2.16901e-01  4.06285e-01 -2.35629e-01]]
```

In [35]:
```
normalizedXdf = pd.DataFrame(normalizedX)
normalizedXdf.columns = ['state', 'account_length', 'area_code', 'internatio
       'number_vmail_messages', 'total_day_minutes', 'total_day_calls', 'tot
       'total_eve_minutes', 'total_eve_calls', 'total_eve_charge', 'total_ni
       'total_night_calls', 'total_night_charge', 'total_intl_minutes', 'tot
       'total_intl_charge', 'number_customer_service_calls']
normalizedXdf['churn'] = data['churn']
normalizedXdf
```

Out[35]:

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_mes
0	0.100228	0.132294	0.301495	-0.070119	-0.132369	-0.1
1	-0.258844	0.059323	-0.000215	-0.083459	-0.157552	-0.1
2	-0.319472	-0.088523	-0.000210	-0.081389	-0.153644	-0.1
3	-0.248572	0.012618	-0.345027	-0.080147	0.395643	0.4
4	0.119293	-0.065984	-0.000167	-0.064788	-0.122304	-0.1
...	...	...	...	...	...	
3328	0.171796	0.298656	-0.000241	-0.093302	0.460582	0.2
3329	-0.123880	-0.148290	-0.000193	-0.074635	0.368433	0.5
3330	0.484276	0.024778	-0.000406	-0.157387	-0.297111	-0.2
3331	0.104932	-0.260638	-0.276225	-0.064165	-0.121129	-0.1
3332	0.360594	-0.313117	-0.343542	-0.079802	-0.150648	-0.1

3333 rows × 20 columns

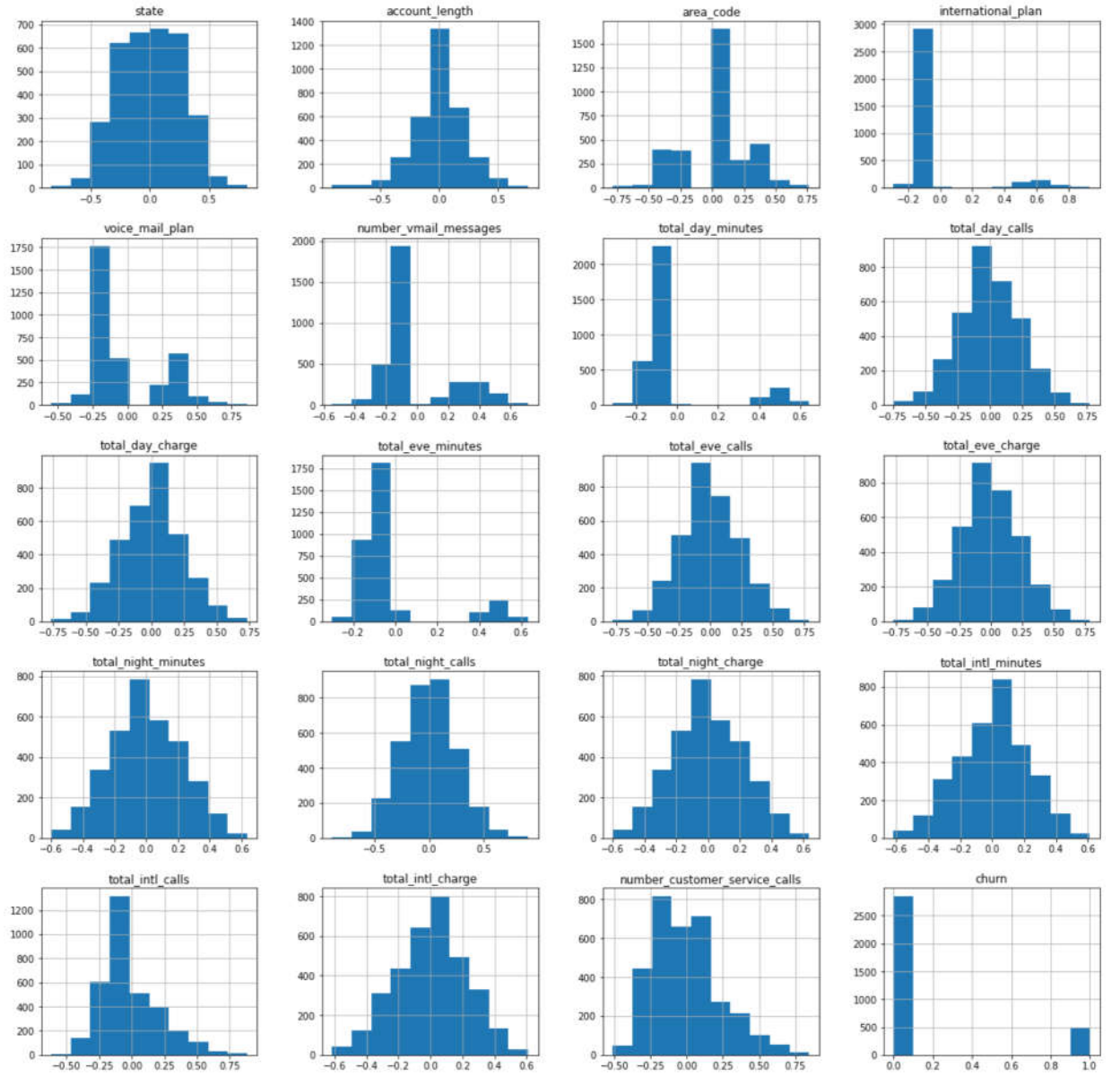
In [36]:
```
normalizedXdf.describe()
```

Out[36]:

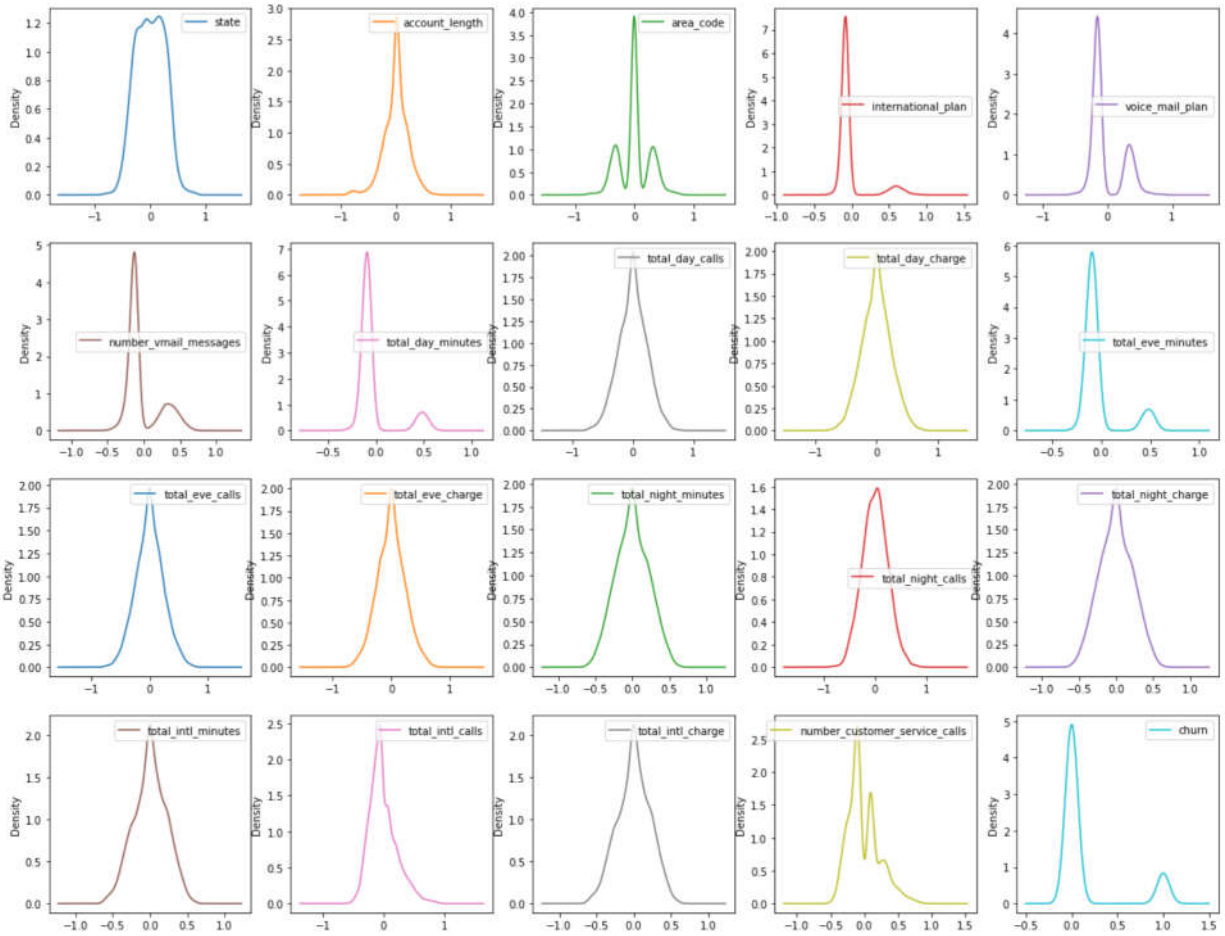
	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3
mean	0.000974	0.005289	0.001114	-0.019526	-0.019806	
std	0.259511	0.218280	0.257155	0.209493	0.248771	
min	-0.836284	-0.909214	-0.784659	-0.291277	-0.549864	
25%	-0.206435	-0.110933	-0.197151	-0.094794	-0.174418	
50%	-0.000911	0.014462	-0.000209	-0.078104	-0.136073	
75%	0.206247	0.129142	0.192624	-0.063376	0.270647	
max	0.823215	0.757165	0.763397	0.931544	0.863848	

In [37]:

```
#Univariate Histograms to show distribution of normalized data
import matplotlib.pyplot as plt


normalizedXdf.hist()
plt.gcf().set_size_inches(20,20)
plt.show()
```



In [38]:
```
#density plots
from matplotlib import pyplot
from pandas import read_csv



normalizedXdf.plot(kind='density', subplots=True, layout=(5,5), sharex=False
plt.gcf().set_size_inches(20,20)
pyplot.show()
```

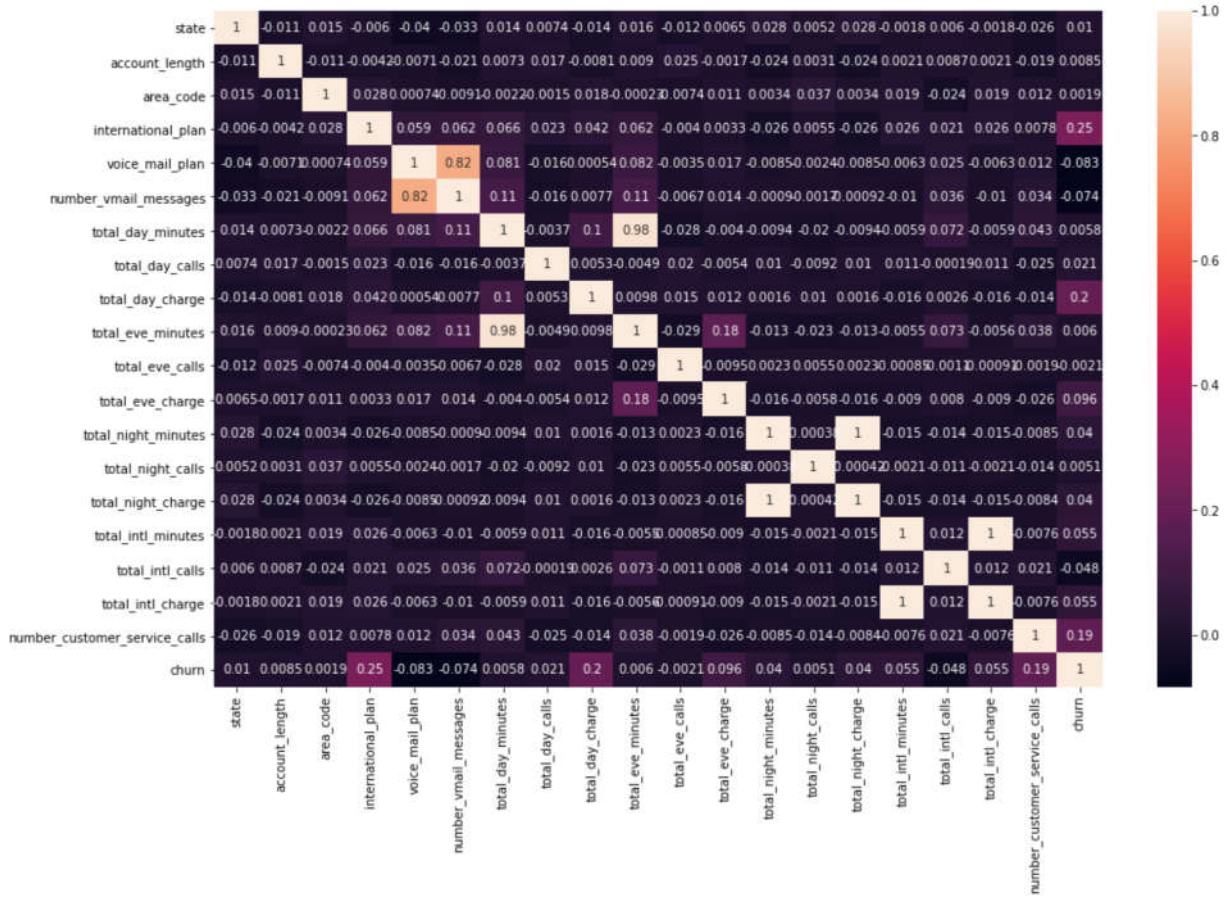


In [39]:
```
'''Comparing the visialization of the transformed data using the differnt te
the data showed a normal expected Gaussian distribution after normalization'
```

Out[39]:
```
'Comparing the visialization of the transformed data using the differnt techniq
ues,\nthe data showed a normal expected Gaussian distribution after normalizati
on'
```

In [40]:
```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import set_option
data = normalizedXdf
set_option('display.width', 100)

plt.figure(figsize=(16,10))

sns.heatmap(data.corr(), annot = True)
```

Out[40]: <AxesSubplot:>



In [ ]:
```
1
```