In [6]:
```
import pandas as pd
#Loading the case1churn.csv dataset
datainput = pd.read_csv('Case1Churn.csv')
datainput.head()
```

Out[6]:

	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	Frequency of SMS	Distinct Called Numbers	Age Group	Tariff Plan
0	8	0	38	0	4370	71	5	17	3	1
1	0	0	39	0	318	5	7	4	2	1
2	10	0	37	0	2453	60	359	24	3	1
3	10	0	38	0	4198	66	1	35	1	1
4	3	0	38	0	2393	58	2	33	1	1

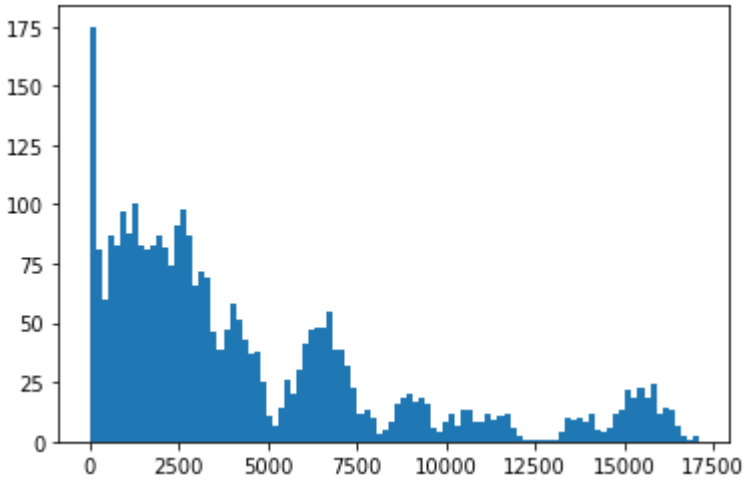
In [7]:
```
#checking total values of the case1churn.csv dataset
datainput['Churn'].value_counts()
```

Out[7]:
```
0    2655
1     495
Name: Churn, dtype: int64
```

In [8]:
```
#Exploring dataset/checking customer churn by age
print(datainput.groupby('Age')['Churn'].value_counts())
```

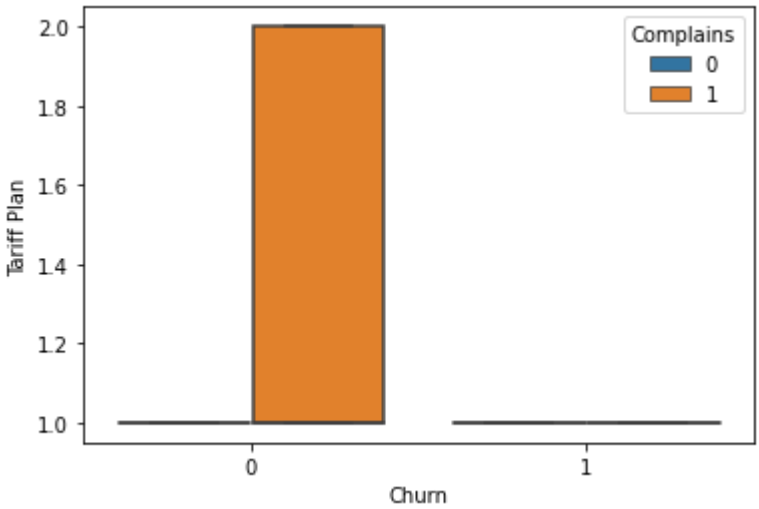
```
Age  Churn
15   0         123
25   0         853
     1         184
30   0        1195
     1         230
45   0         316
     1          79
55   0         168
     1           2
Name: Churn, dtype: int64
```

In [9]:
```
# Import matplotlib and seaborn
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize the distribution of 'Total day minutes'
plt.hist(datainput['Seconds of Use'], bins = 100)

# Display the plot
plt.show()
```



In [10]:

```
# Creating a box plot to understand the binary variables
sns.boxplot(x = 'Churn',
            y = 'Tariff Plan',
            data = datainput,
            sym = "",
            hue = "Complains")
# Display the plot
plt.show()
```



In [11]:

```
# Displaying information about the dataset
datainput.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3150 entries, 0 to 3149
Data columns (total 16 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Call  Failure             3150 non-null   int64
 1   Complains                 3150 non-null   int64
 2   Subscription  Length      3150 non-null   int64
 3   Charge  Amount            3150 non-null   int64
 4   Seconds of Use            3150 non-null   int64
 5   Frequency of use          3150 non-null   int64
 6   Frequency of SMS          3150 non-null   int64
 7   Distinct Called Numbers   3150 non-null   int64
 8   Age Group                 3150 non-null   int64
```

```
 9   Tariff Plan              3150 non-null   int64
 10  Status                   3150 non-null   int64
 11  Age                      3150 non-null   int64
 12  Customer Value           3150 non-null   float64
 13  FN                       3150 non-null   float64
 14  FP                       3150 non-null   float64
 15  Churn                    3150 non-null   int64
dtypes: float64(3), int64(13)
memory usage: 393.9 KB
```

In [12]:

```
# Diplaying summary of the dataset
datainput.describe()
```

Out[12]:

	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	Frequency of SMS	
count	3150.000000	3150.000000	3150.000000	3150.000000	3150.000000	3150.000000	3150.000000	31
mean	7.627937	0.076508	32.541905	0.942857	4472.459683	69.460635	73.174921	
std	7.263886	0.265851	8.573482	1.521072	4197.908687	57.413308	112.237560	
min	0.000000	0.000000	3.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	30.000000	0.000000	1391.250000	27.000000	6.000000	
50%	6.000000	0.000000	35.000000	0.000000	2990.000000	54.000000	21.000000	
75%	12.000000	0.000000	38.000000	1.000000	6478.250000	95.000000	87.000000	
max	36.000000	1.000000	47.000000	10.000000	17090.000000	255.000000	522.000000	

In [13]:

```
# Preprocessing the dataset/dropping unwanted variables

datainput.drop(['FN','FP'], axis=1, inplace=True)
datainput
```

Out[13]:

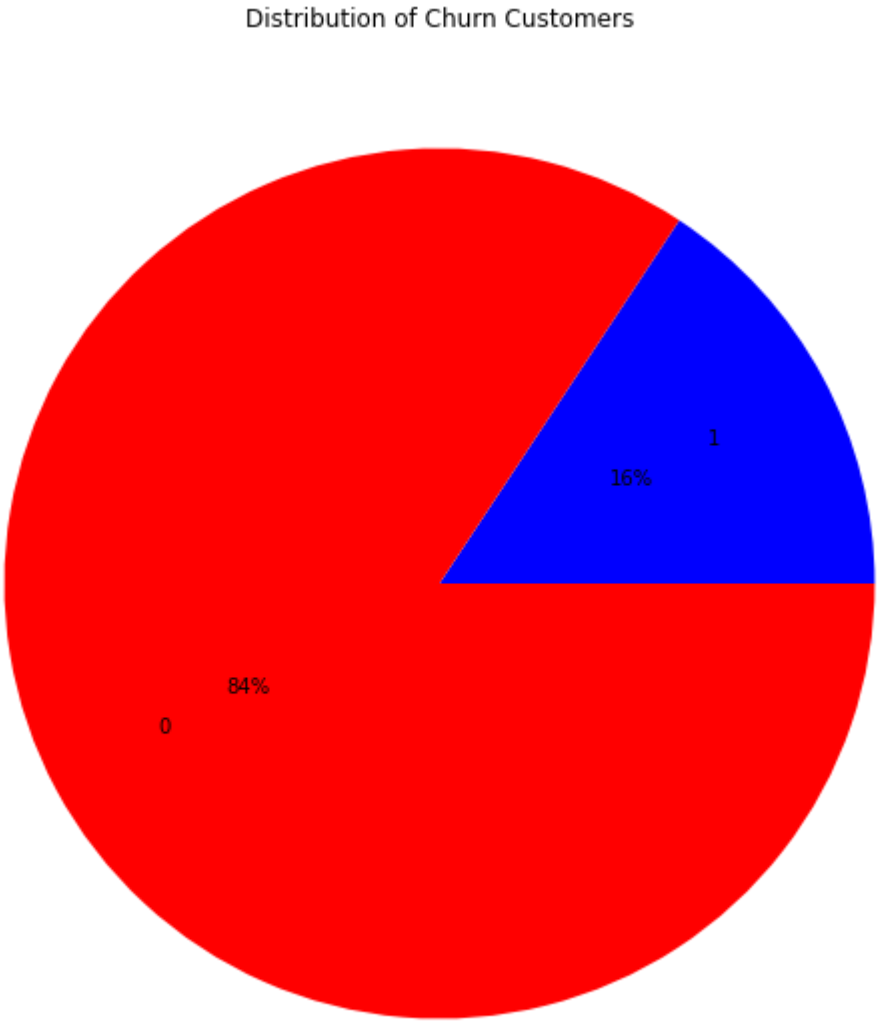
	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	Frequency of SMS	Distinct Called Numbers	Age Group	Ta P
0	8	0	38	0	4370	71	5	17	3	
1	0	0	39	0	318	5	7	4	2	
2	10	0	37	0	2453	60	359	24	3	
3	10	0	38	0	4198	66	1	35	1	
4	3	0	38	0	2393	58	2	33	1	
...	...	...	...	...	...	...	...	...	...	
3145	21	0	19	2	6697	147	92	44	2	
3146	17	0	17	1	9237	177	80	42	5	
3147	13	0	18	4	3157	51	38	21	3	

	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	Frequency of SMS	Distinct Called Numbers	Age Group	Ta P
3148	7	0	11	2	4695	46	222	12	3	
3149	8	1	11	2	1792	25	7	9	3	

3150 rows × 14 columns

In [14]:
```
# Visualizing summary of churn in the dataset

plt.figure(figsize=(10,10))
plt.pie(x=[495, 2655], labels=['1','0'], autopct='%1.0f%%', pctdistance=0.5,labeldistan
plt.title('Distribution of Churn Customers')
```

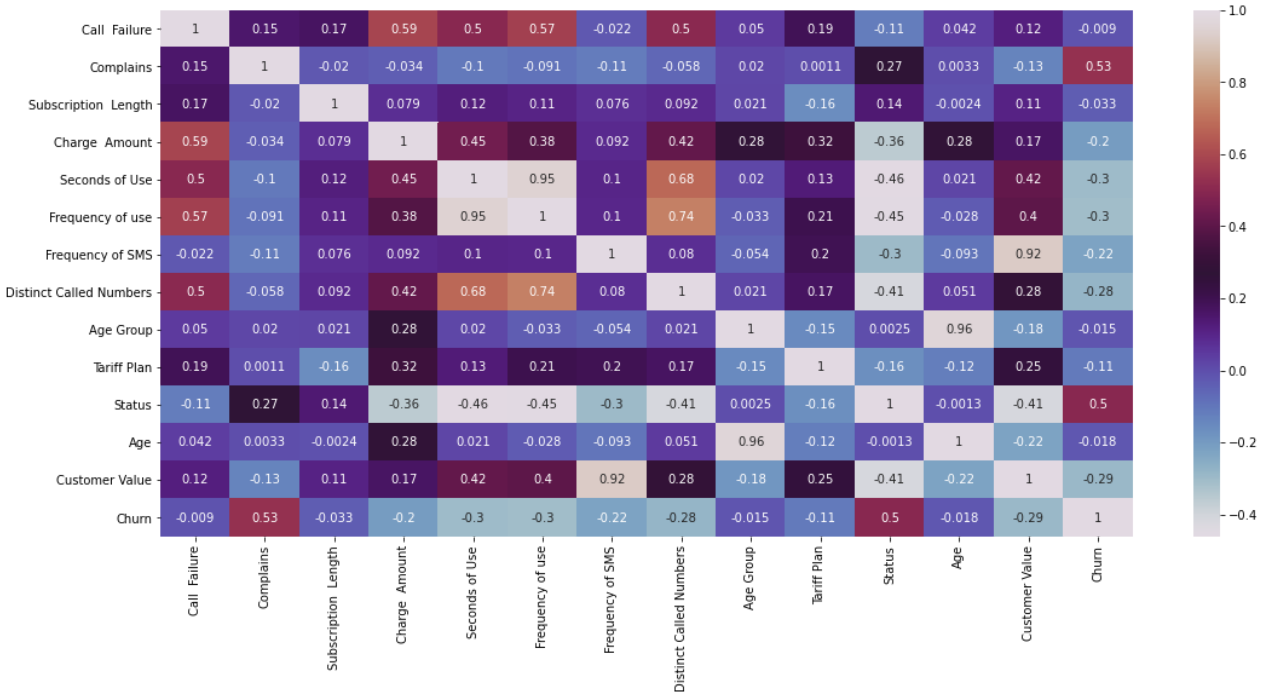
Out[14]: Text(0.5, 1.0, 'Distribution of Churn Customers')



In [15]:
```
# Creating a correlation matrix plot
corr = datainput.corr()
```

```
plt.figure(figsize=(18,8))
sns.heatmap(corr, annot = True, cmap='twilight')
```

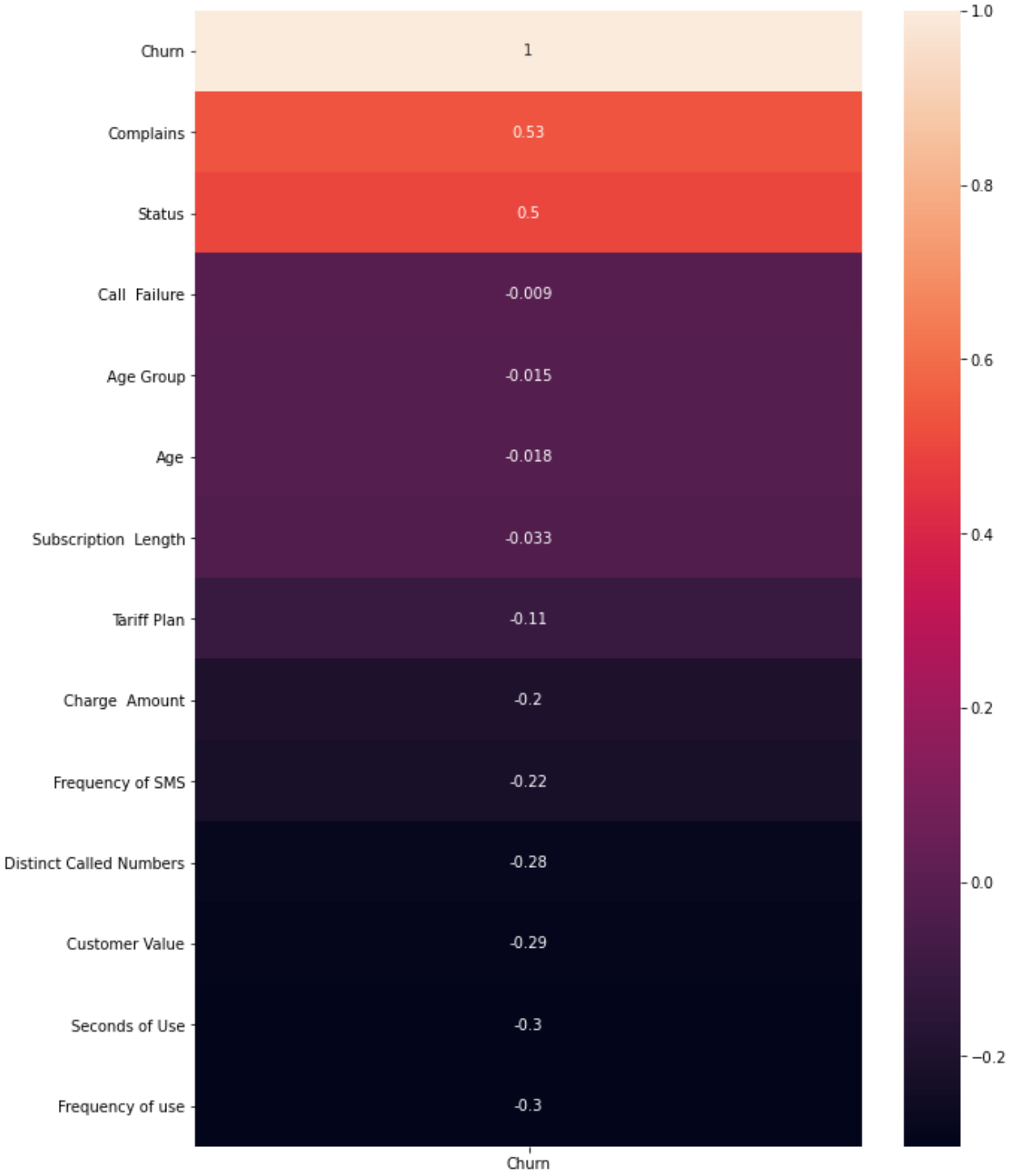
Out[15]: <AxesSubplot:>



In [16]:

```
# Creating a correlation matrix plot in order

fig, ax = plt.subplots(figsize=(10,14))
churn_data_corr = datainput.corr()[['Churn']].sort_values(
  by='Churn', ascending=False)
sns.heatmap(churn_data_corr, annot=True, ax=ax)
```

Out[16]: <AxesSubplot:>



In [17]:
```
#Identifying response variable:

response = datainput['Churn']
dataset = datainput.drop(columns = 'Churn')
```

In [18]:
```
# Creating a training and testing data

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(datainput, response,stratify = resp
```

In [19]:

```
print("Number transactions X_train datainput: ", X_train.shape)
print("Number transactions y_train datainput: ", y_train.shape)
print("Number transactions X_test datainput: ", X_test.shape)
print("Number transactions y_test datainput: ", y_test.shape)
```

```
Number transactions X_train datainput:  (2520, 14)
Number transactions y_train datainput:  (2520,)
Number transactions X_test datainput:  (630, 14)
Number transactions y_test datainput:  (630,)
```

In [62]:

```
# Creating Support Vector Machine and Predicting

from sklearn.svm import SVC
svclassifier = SVC(kernel='linear', probability= True)
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)
```

Out[62]:

```
SVC(kernel='linear', probability=True)
```

In [ ]:

```
# Creating Evaluation Metrics of the Support Vector Machine
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

In [ ]:

```

```