```
# importing the necesarry python libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import zipfile
import plotly.express as px
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
path = "/content/drive/MyDrive/iranian+churn+dataset.zip"
```

```
df = zipfile.ZipFile(path)    # unarchiving the dataset
```

```
df.namelist()    # This tells us the name of the file in the dataset
```

```
['Customer Churn.csv']
```

```
# Reading the file
df = pd.read_csv(df.open("Customer Churn.csv"))
```

```
df.head(6)
```

	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	Frequency of SMS	Distinct Called Numbers	Age Group	Tariff Plan	Status	Age	Customer Value	Churn
0	8	0	38	0	4370	71	5	17	3	1	1	30	197.640	0
1	0	0	39	0	318	5	7	4	2	1	2	25	46.035	0
2	10	0	37	0	2453	60	359	24	3	1	1	30	1536.520	0
3	10	0	38	0	4198	66	1	35	1	1	1	15	240.020	0
4	3	0	38	0	2393	58	2	33	1	1	1	15	145.805	0

```
# checking for any missing values
df.isnull().sum()
```

```
Call  Failure             0
Complains                 0
Subscription  Length      0
Charge  Amount            0
Seconds of Use            0
Frequency of use          0
Frequency of SMS          0
Distinct Called Numbers   0
Age Group                 0
Tariff Plan               0
Status                    0
Age                       0
Customer Value            0
Churn                     0
dtype: int64
```

```
df["Age Group"].unique()
```

```
array([3, 2, 1, 4, 5])
```

```
df.Status.unique()
```

```
array([1, 2])
```

```
df.Status = df.Status.replace({2: 0})
```

```
df.Status.unique()
```

```
array([1, 0])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3150 entries, 0 to 3149
Data columns (total 14 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  Churn                   3150 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 344.7 KB
```

```
# Gives us a statistical summary of the dataset
df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
Call Failure	3150.0	7.627937	7.263886	0.0	1.00000	6.00	12.00000	36.00
Complains	3150.0	0.076508	0.265851	0.0	0.00000	0.00	0.00000	1.00
Subscription Length	3150.0	32.541905	8.573482	3.0	30.00000	35.00	38.00000	47.00
Charge Amount	3150.0	0.942857	1.521072	0.0	0.00000	0.00	1.00000	10.00
Seconds of Use	3150.0	4472.459683	4197.908687	0.0	1391.25000	2990.00	6478.25000	17090.00
Frequency of use	3150.0	69.460635	57.413308	0.0	27.00000	54.00	95.00000	255.00
Frequency of SMS	3150.0	73.174921	112.237560	0.0	6.00000	21.00	87.00000	522.00
Distinct Called Numbers	3150.0	23.509841	17.217337	0.0	10.00000	21.00	34.00000	97.00
Age Group	3150.0	2.826032	0.892555	1.0	2.00000	3.00	3.00000	5.00
Tariff Plan	3150.0	1.077778	0.267864	1.0	1.00000	1.00	1.00000	2.00
Status	3150.0	0.751746	0.432069	0.0	1.00000	1.00	1.00000	1.00
Age	3150.0	30.998413	8.831095	15.0	25.00000	30.00	30.00000	55.00
Customer Value	3150.0	470.972916	517.015433	0.0	113.80125	228.48	788.38875	2165.28
Churn	3150.0	0.157143	0.363993	0.0	0.00000	0.00	0.00000	1.00

```
# Gives us the rows and columns of the dataset
df.shape
```

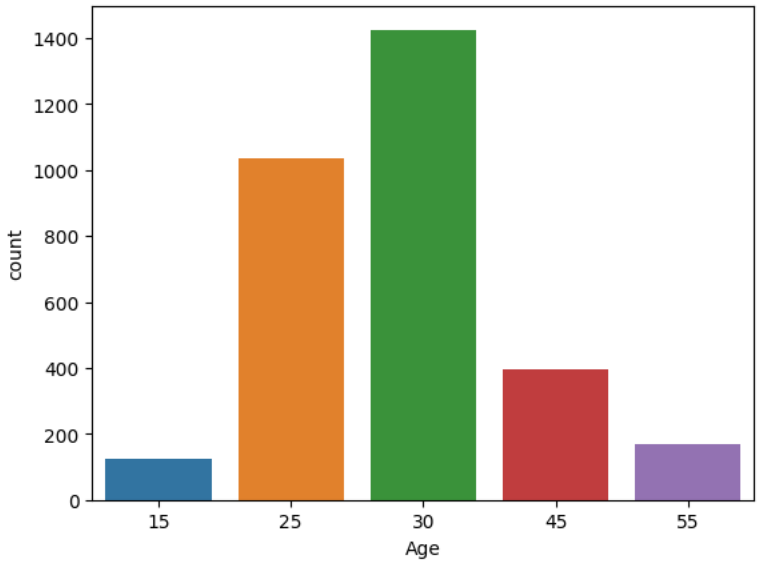
```
(3150, 14)
```

```
df.columns
```

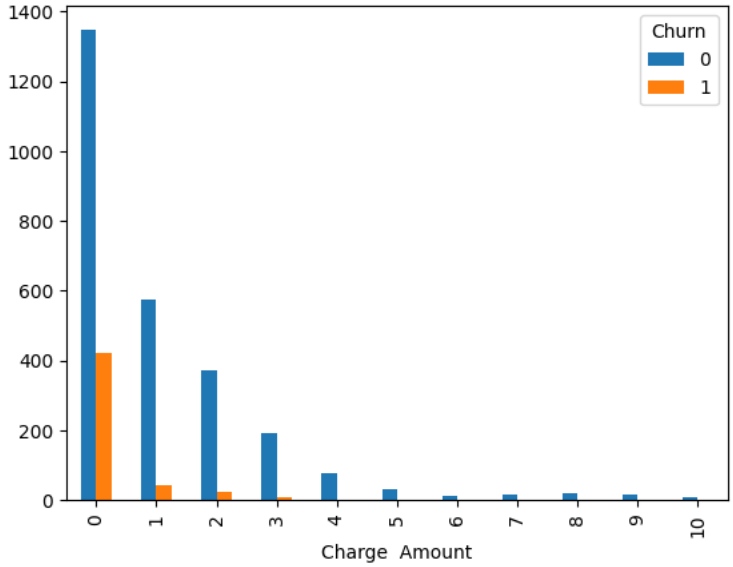
```
Index(['Call  Failure', 'Complains', 'Subscription  Length', 'Charge  Amount',
       'Seconds of Use', 'Frequency of use', 'Frequency of SMS',
       'Distinct Called Numbers', 'Age Group', 'Tariff Plan', 'Status', 'Age',
       'Customer Value', 'Churn'],
      dtype='object')
```

```
sb.countplot(data = df , x ="Age")
```

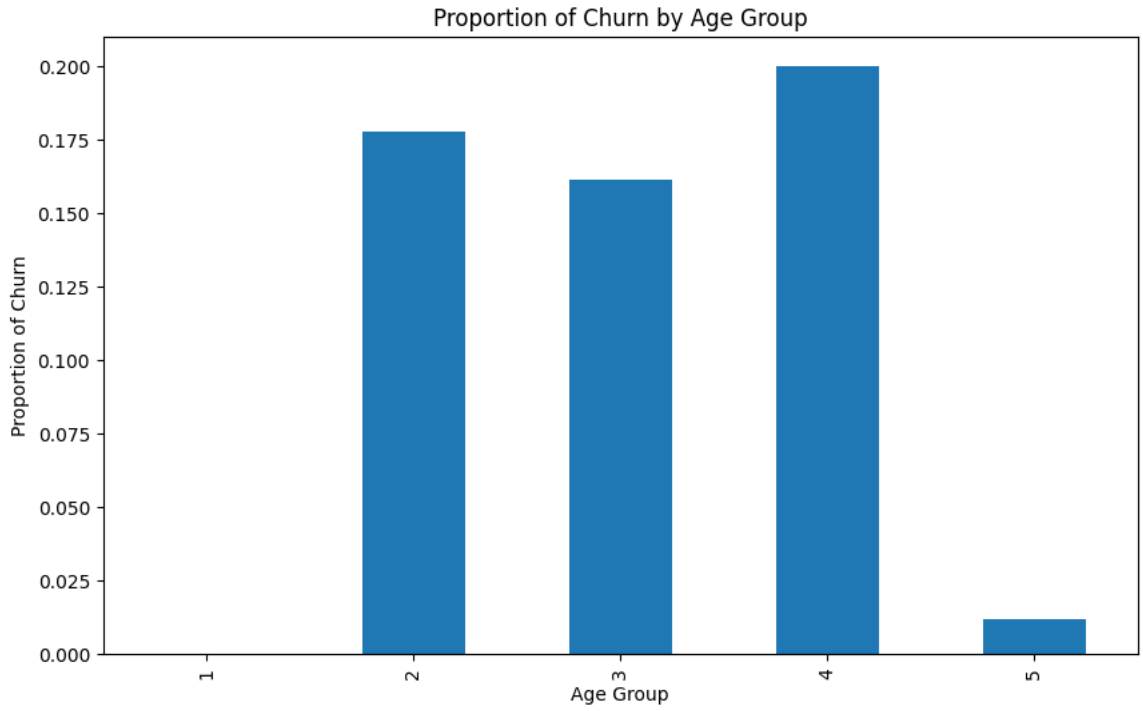
```
<Axes: xlabel='Age', ylabel='count'>
```




```
pd.crosstab(df["Charge  Amount"] , df.Churn).plot(kind = "bar");
```

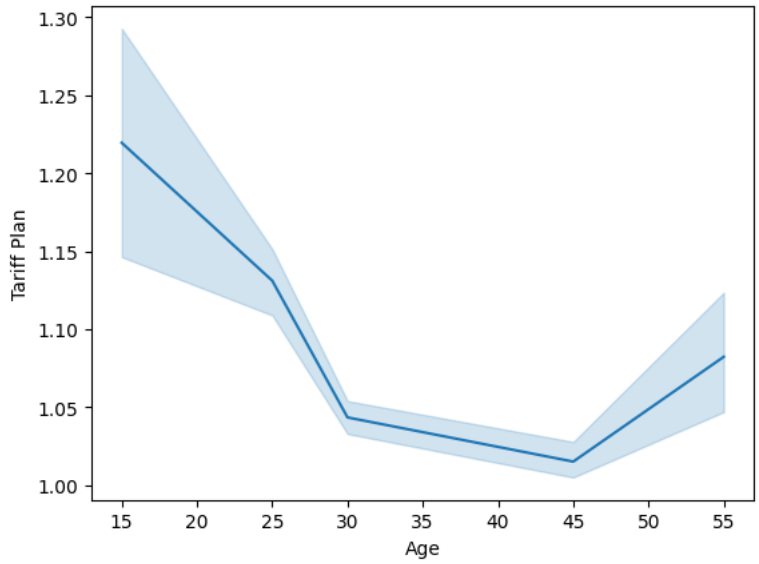



```
# Calculate the proportion of churned customers within each age group
age_group_churn_proportions = df.groupby('Age Group')['Churn'].mean()

# Create the bar plot
plt.figure(figsize=(10, 6))
age_group_churn_proportions.plot(kind='bar')

plt.title('Proportion of Churn by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Proportion of Churn')

plt.show()
```



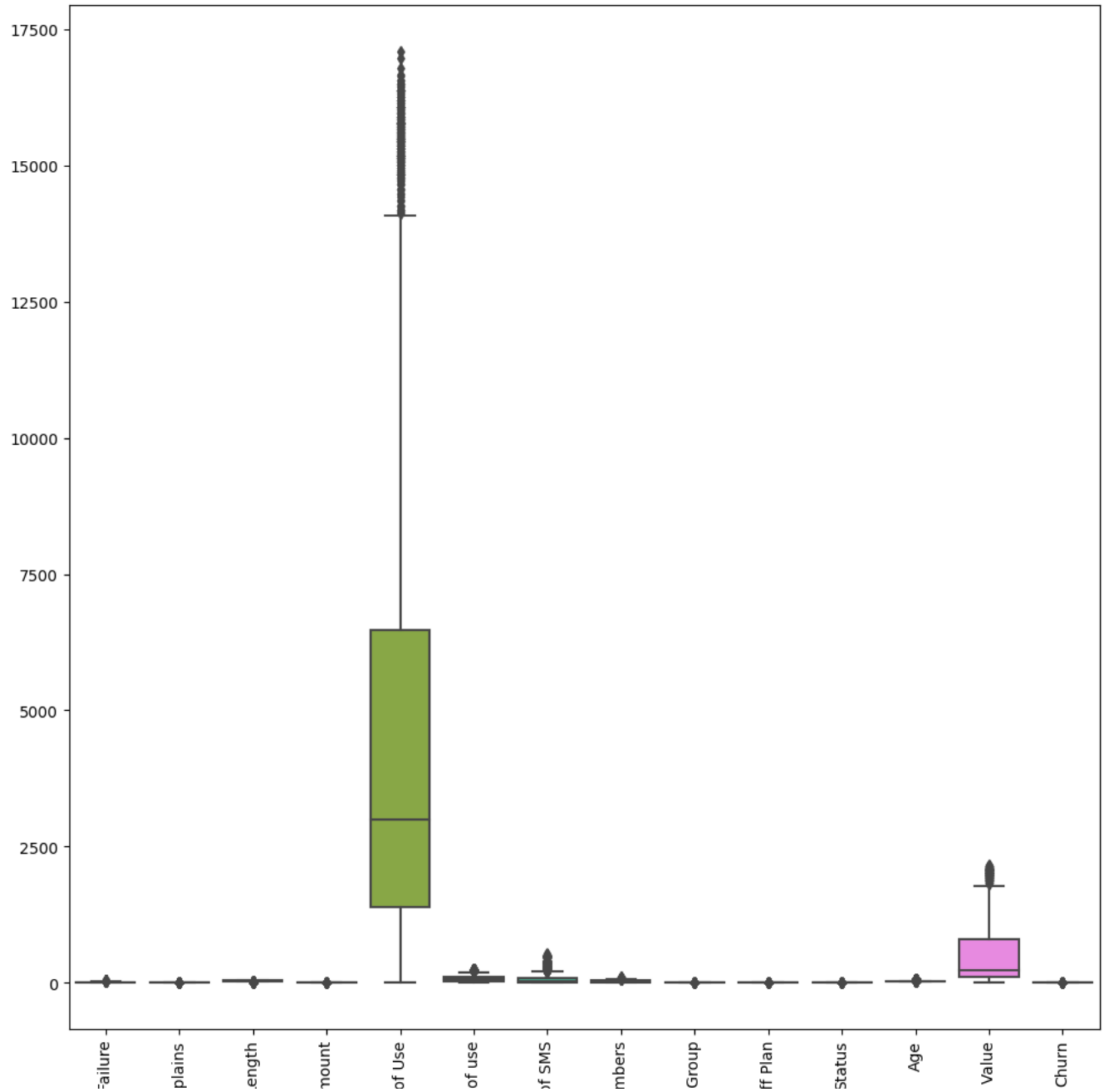

```
sb.lineplot(data = df , y = "Tariff Plan" , x = "Age" )
```

```
<Axes: xlabel='Age', ylabel='Tariff Plan'>
```




```
plt.figure(figsize = (12,12))
plt.xticks(rotation = 90);
sb.boxplot(df)
```

```
<Axes: >
```




```
df["Age Group"].unique()
```

```
array([3, 2, 1, 4, 5])
```

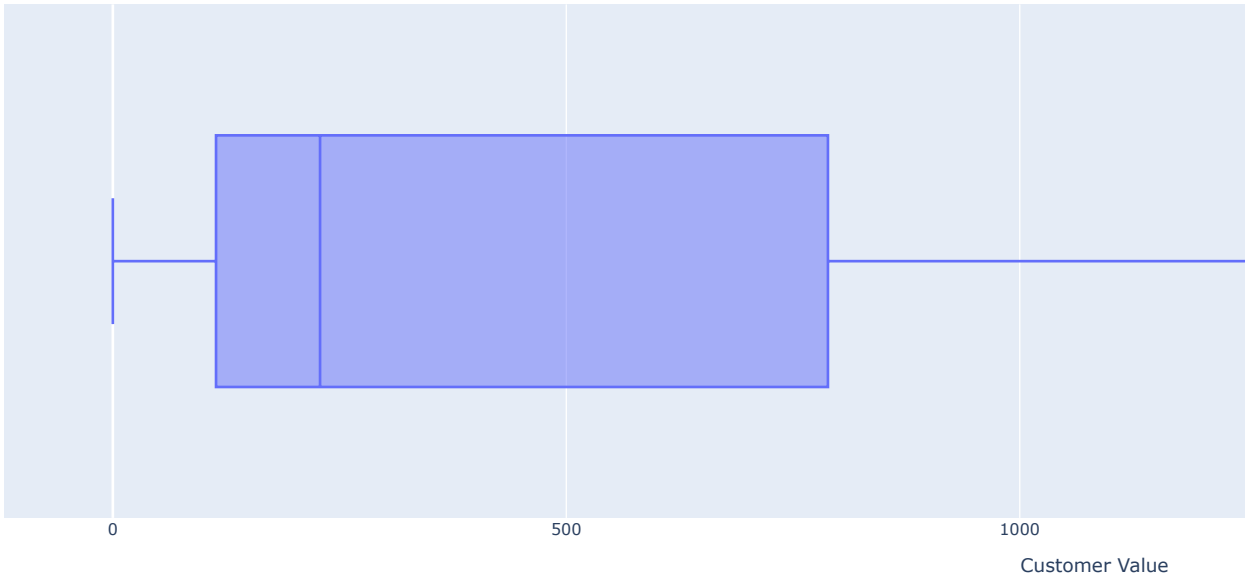
```
df["Tariff Plan"].unique()
```

```
array([1, 2])
```

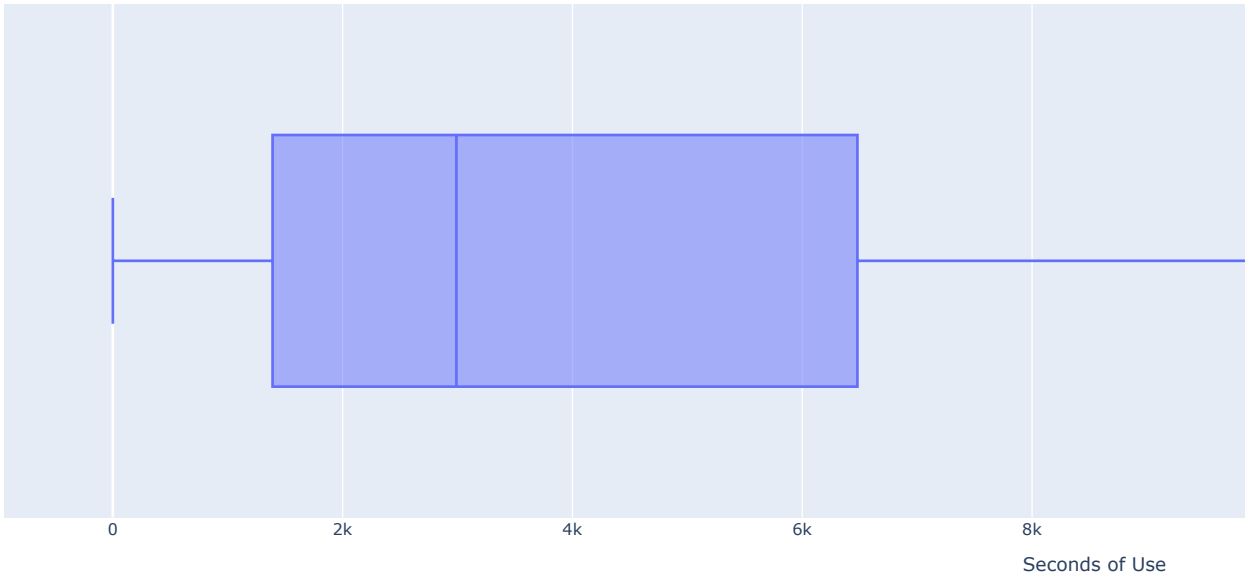
```
df.columns
```

```
Index(['Call  Failure', 'Complains', 'Subscription  Length', 'Charge  Amount',
       'Seconds of Use', 'Frequency of use', 'Frequency of SMS',
       'Distinct Called Numbers', 'Age Group', 'Tariff Plan', 'Status', 'Age',
       'Customer Value', 'Churn'],
      dtype='object')
```

```
plot = px.box(df , x = "Customer Value")
plot.show()
```




```
plot = px.box(df , x = "Seconds of Use")
plot.show()
```

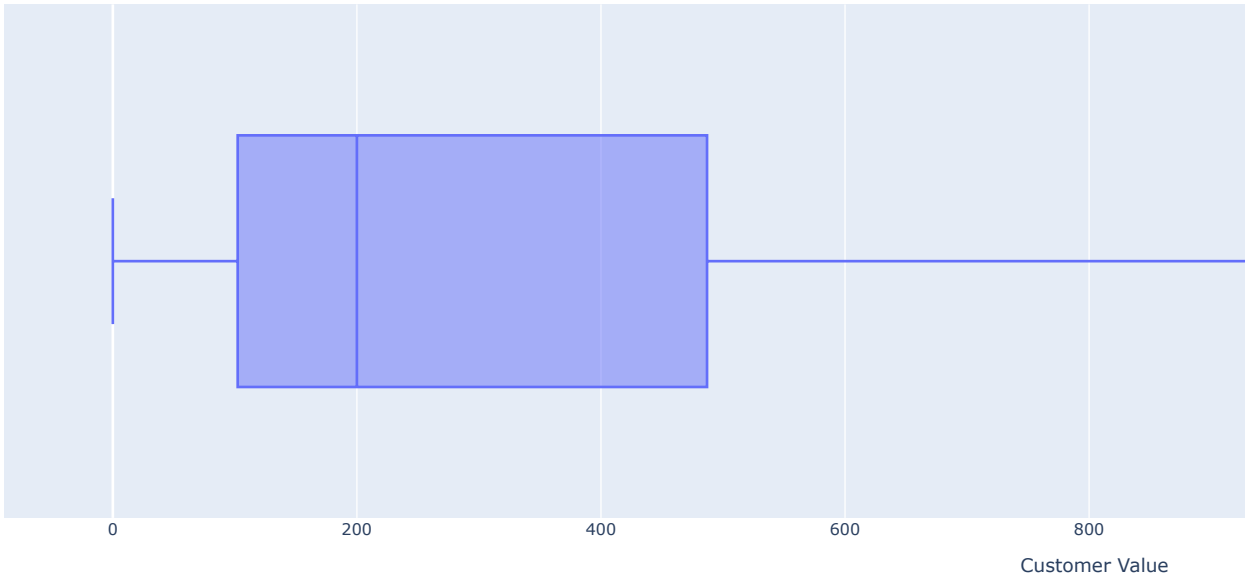



```
df_1 = df[(df["Customer Value"] < 1610)  & (df["Seconds of Use"] < 15000)]
```

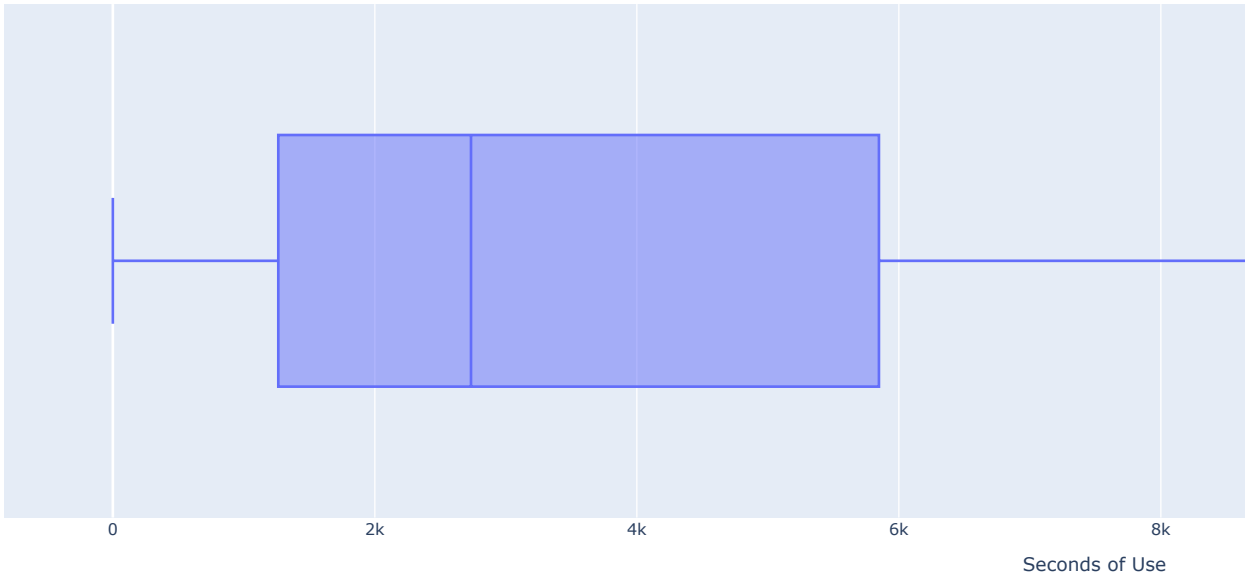
```
df_1.shape
```

```
(2840, 14)
```

```
plot = px.box(df_1 , x = "Customer Value")    # We can see that our outlier has been dealt with
plot.show()
```




```
plot = px.box(df_1 , x = "Seconds of Use")    # we can see that have dealt a little bit with the outlier in the Second of use
plot.show()                                   # i did not remove further due to less data set
```




```
percent_outlier = ((df.shape[0]- df_1.shape[0])/df.shape[0]) * 100
print(round(percent_outlier , 2),"%")


#From this we can see that this is only 10 percent of the whole data so we can drop it     -

# df_1 will be our new data frame
```

```
     9.84 %
```

Traing the Dataset

```
df_1.columns
```

```
Index(['Call  Failure', 'Complains', 'Subscription  Length', 'Charge  Amount',
       'Seconds of Use', 'Frequency of use', 'Frequency of SMS',
       'Distinct Called Numbers', 'Age Group', 'Tariff Plan', 'Status', 'Age',
       'Customer Value', 'Churn'],
      dtype='object')
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```