Customer Churn Prediction using Machine Learning

I. Introduction

a. Background and Purpose of the project

There is said “Customer is the King,” they are important because they make your business drive revenue, your business does not continue to exist without them. Business can save cost 5 times than acquire new customers (Mage,2022), and the increasing of customer retention 5% could help business increase profits more than 25% (Fred Reichard of Bain and Company).

Customer churn refer to the action of customer chooses to drop their service provider (D. L Gardini, A.Nebot and A. Vellido,2017)

Churn prediction could help business to identify their customers who are likely to churn, give them reliable information about current customers, so they could build effective customer retention and marketing strategy. Moreover, it helps management team to make better decision using data-driven, not assumption anymore to shape their strategy and could step ahead of competitors.

The goal of this studyiny:

- Explore Data and Analyze what happend
- Using four of machine learning methods: XGBoost, Random Forest, Logistics Regression and Gradient Boosting Classifier to build churn prediction models
- Investigate important features on churn model

b. Objectives of the analysis

- Assessment of the measures of frequency
- How much strong relationship exists among the variables
- Making inferences and taking decisions
- To acquire insights regarding the relationships which are being observed

c. Overview of the data source and variables

This dataset is randomly collected from an Iranian telecom company database over a period of 12 months. A total of 3150 rows of data, each representing a customer, bear information for 13 columns. The attributes that are in this dataset are call failures, frequency of SMS, number of complaints, number of distinct calls, subscription length, age group, the charge amount, type of service, seconds of use, status, frequency of use, and Customer Value.

All of the attributes except for attribute churn is the aggregated data of the first 9 months. The churn labels are the state of the customers at the end of 12 months. The three months is the designated planning gap.

II. Data Acquisition and Cleaning

In [1]:

```
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import balanced_accuracy_score,roc_auc_score,m
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix,classification_report
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingClassifier,RandomFores
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold, StratifiedGrou
%matplotlib inline
```

```
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/co
mpat.py:36: FutureWarning: pandas.Int64Index is deprecated and wil
l be removed from pandas in a future version. Use pandas.Index wit
h the appropriate dtype instead.
  from pandas import MultiIndex, Int64Index
```

In [2]:

```
#Import data

df=pd.read_csv('/Users/roatny/Desktop/ICT Master Application/6.Mach
```

In [3]:
```
df.head()
```

Out[3]:

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

- Anonymous Customer ID
- Call Failures: number of call failures
- Complains: binary (0: No complaint, 1: complaint)
- Subscription Length: total months of subscription
- Charge Amount: Ordinal attribute (0: lowest amount, 9: highest amount)
- Seconds of Use: total seconds of calls
- Frequency of use: total number of calls
- Frequency of SMS: total number of text messages
- Distinct Called Numbers: total number of distinct phone calls
- Age Group: ordinal attribute (1: younger age, 5: older age)
- Tariff Plan: binary (1: Pay as you go, 2: contractual)
- Status: binary (1: active, 2: non-active)
- Churn: binary (1: churn, 0: non-churn) - Class label
- Customer Value: The calculated value of customer

Data Cleaning and preprocessing

- Checking missing value, NO missing value

In [4]:
```
df.isna().sum()
```

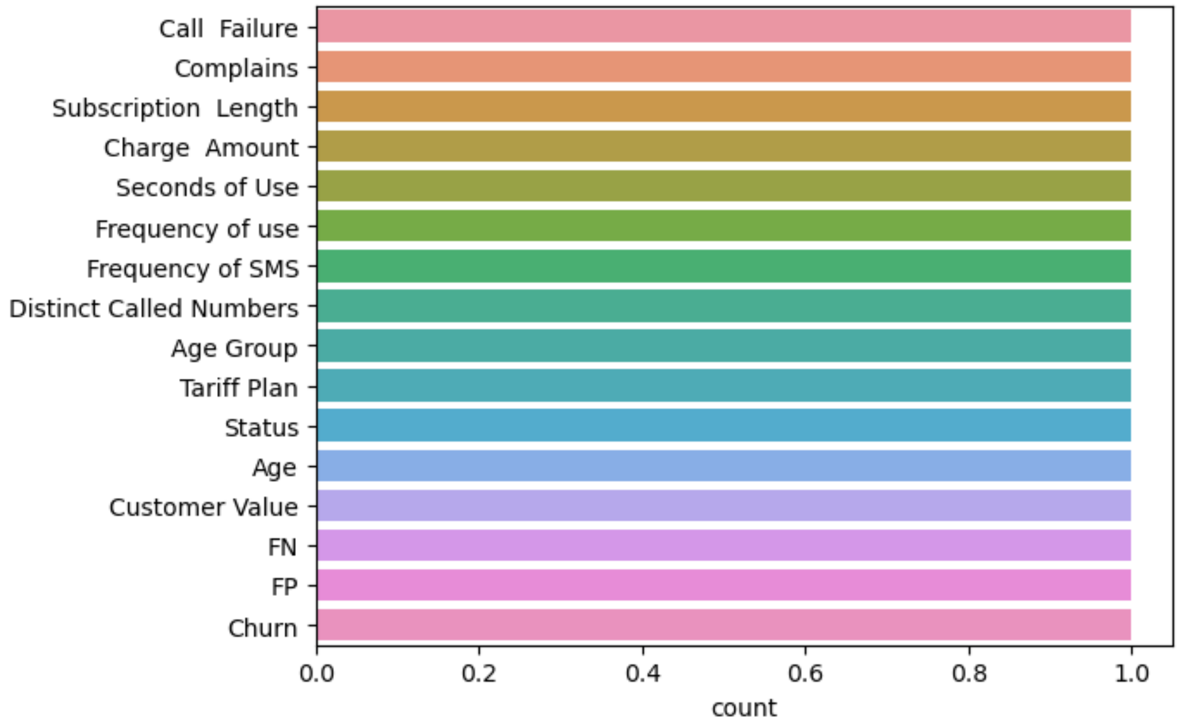
Out[4]:
```
Call  Failure              0
Complains                  0
Subscription  Length       0
Charge  Amount             0
Seconds of Use             0
Frequency of use           0
Frequency of SMS           0
Distinct Called Numbers    0
Age Group                  0
Tariff Plan                0
Status                     0
Age                        0
Customer Value             0
FN                         0
FP                         0
Churn                      0
dtype: int64
```

In [5]:
```
#Check our all columns
all_cl=df.columns
```

In [6]:
```
all_cl
```

Out[6]:
```
Index(['Call  Failure', 'Complains', 'Subscription  Length', 'Char
ge  Amount',
       'Seconds of Use', 'Frequency of use', 'Frequency of SMS',
       'Distinct Called Numbers', 'Age Group', 'Tariff Plan', 'Sta
tus', 'Age',
       'Customer Value', 'FN', 'FP', 'Churn'],
      dtype='object')
```

In [7]:
```
sns.countplot(y=all_cl)

# Show plot
plt.show()
```



In [8]:
```
df.shape
```

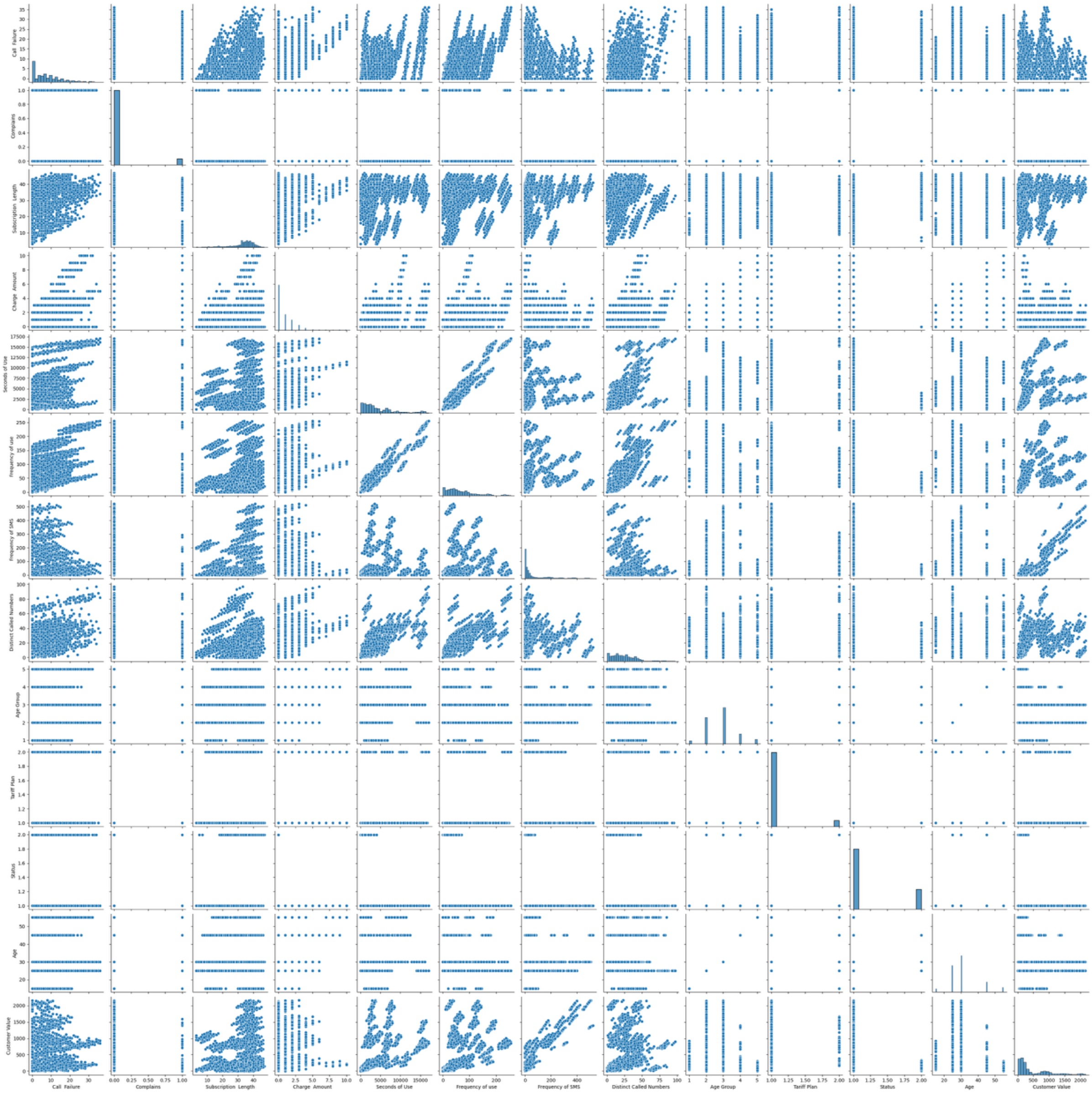
Out[8]:
```
(3150, 16)
```

Correlations

- Correlations can tell us about the direction of the relationship, the form (shape) of the relationship, and the degree (strength) of the relationship between two variables.

In [9]:
```
columns=['Call  Failure', 'Complains', 'Subscription  Length', 'Cha
        'Seconds of Use', 'Frequency of use', 'Frequency of SMS',
        'Distinct Called Numbers', 'Age Group', 'Tariff Plan', 'Stat
        'Customer Value']
sns.pairplot(df[columns])
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x7f9080407070>
```



Type *Markdown* and LaTeX: α^2

In [10]:
```
df.dtypes
```

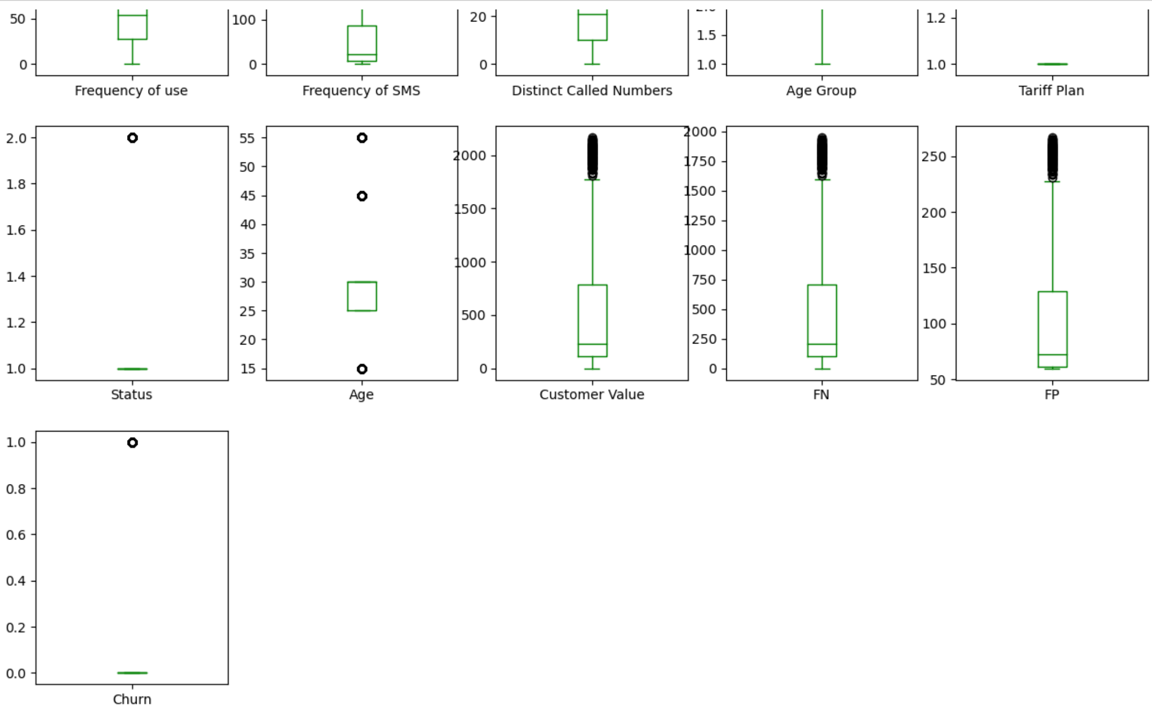
Out[10]:
```
Call  Failure                int64
Complains                    int64
Subscription  Length         int64
Charge  Amount               int64
Seconds of Use               int64
Frequency of use             int64
Frequency of SMS             int64
Distinct Called Numbers      int64
Age Group                    int64
Tariff Plan                  int64
Status                       int64
Age                          int64
Customer Value             float64
FN                         float64
FP                         float64
Churn                        int64
dtype: object
```

In [11]:
```
df.describe()
```

Out[11]:

	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	F
count	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	
std	7.263886	0.265851	8.573482	1.521072	4197.908687	57.413308	1
min	0.000000	0.000000	3.000000	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	30.000000	0.000000	1391.250000	27.000000	
50%	6.000000	0.000000	35.000000	0.000000	2990.000000	54.000000	
75%	12.000000	0.000000	38.000000	1.000000	6478.250000	95.000000	
max	36.000000	1.000000	47.000000	10.000000	17090.000000	255.000000	5

In [12]:
```
df.plot(kind="box",subplots=True,layout=(5,5),figsize=(15,20),color
```



In [ ]:

II. Data Exploration and Visualization

a. Univariate analysis

Analyze data of just one variable

In [13]:
```
#Check churn number
churn=df['Churn']
```

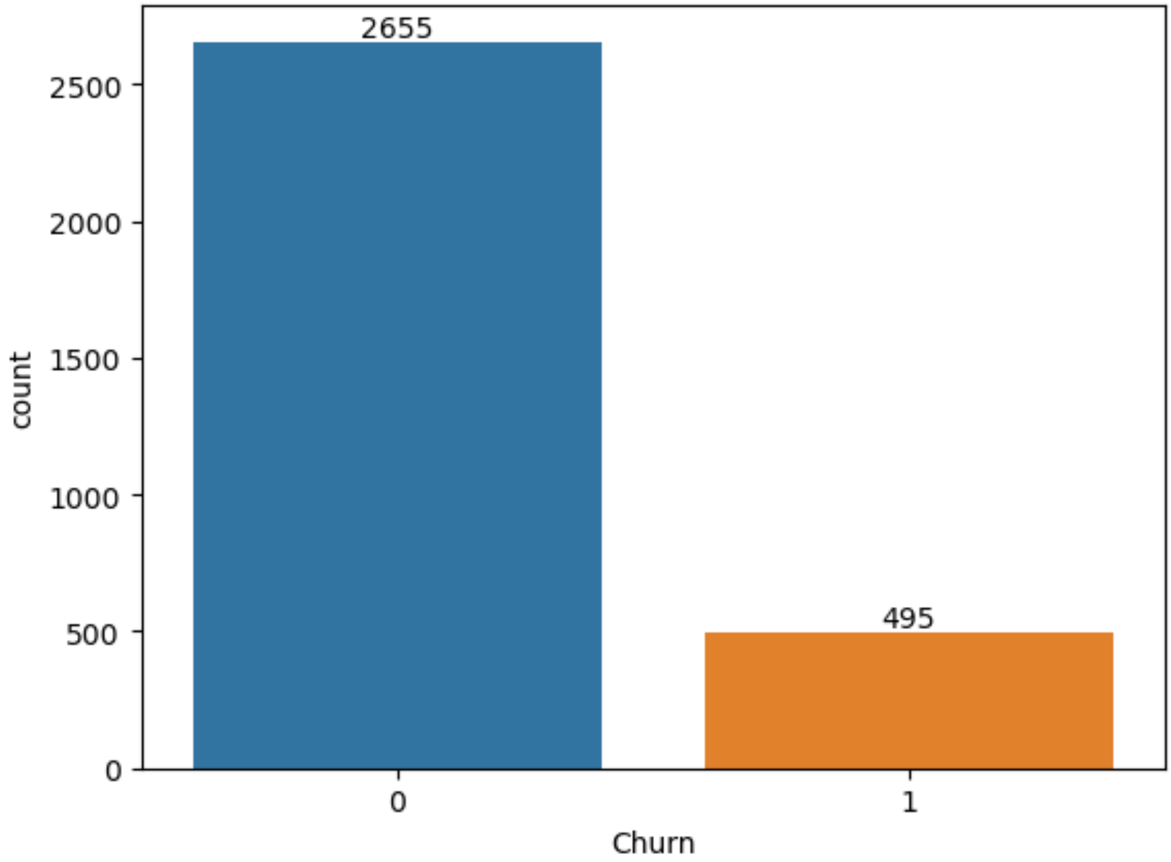
In [14]:
```
churn.unique()
```

Out[14]:
```
array([0, 1])
```

In [15]:
```
sum(churn)/len(churn)
```

Out[15]:
```
0.15714285714285714
```

In [16]:
```
churn_plot=sns.countplot(x='Churn',data=df)
for i in churn_plot.containers:
    churn_plot.bar_label(i,)
```

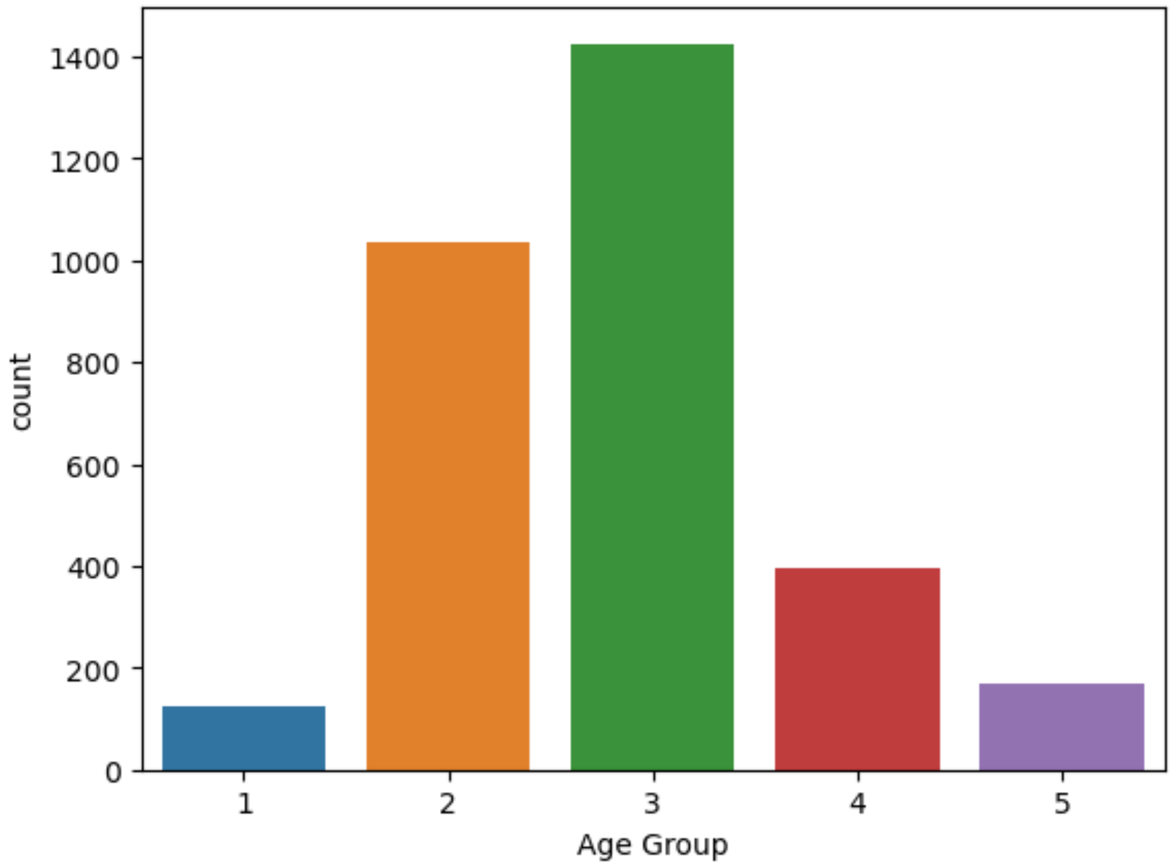


In [17]:
```
df['Age Group'].value_counts()
```

Out[17]:
```
3    1425
2    1037
4     395
5     170
1     123
Name: Age Group, dtype: int64
```

In [18]:
```
sns.countplot(x='Age Group',data=df)
```

Out[18]: <AxesSubplot:xlabel='Age Group', ylabel='count'>

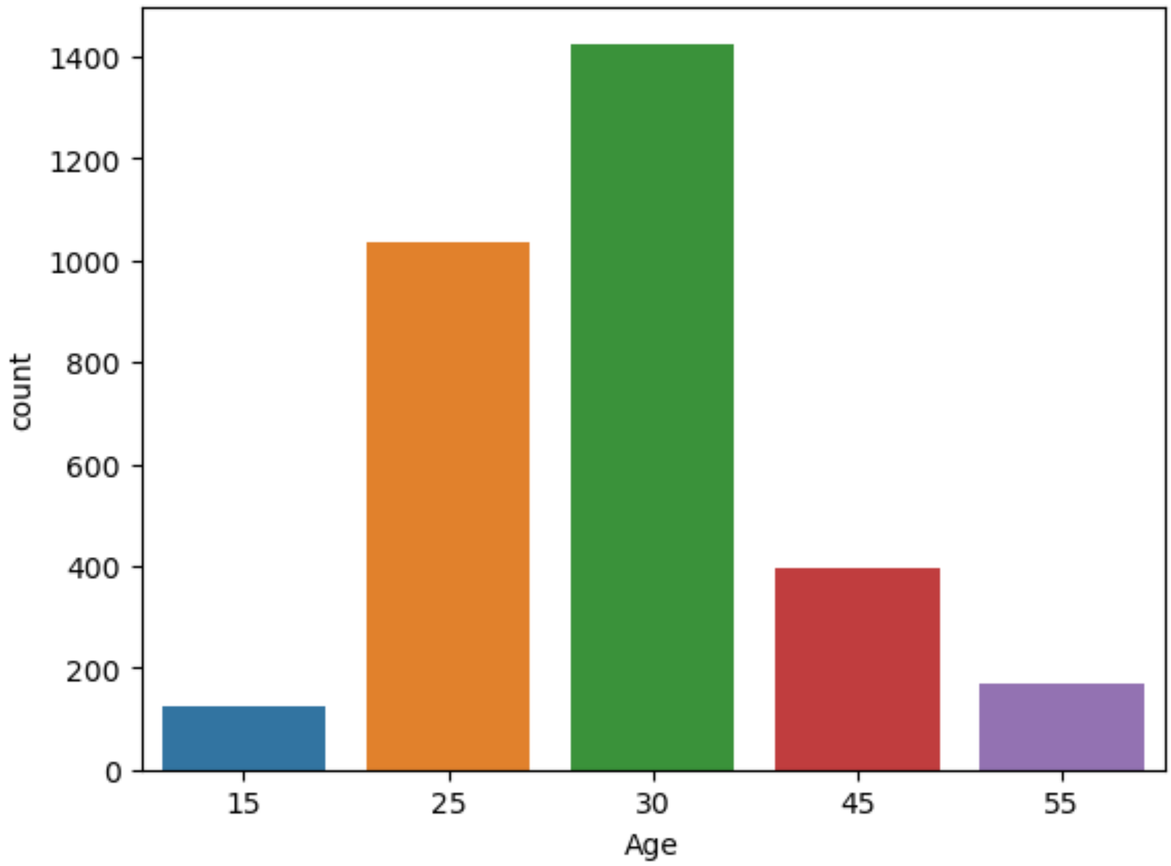


In [19]:
```
df['Age'].value_counts()
```

Out[19]:
```
30    1425
25    1037
45     395
55     170
15     123
Name: Age, dtype: int64
```

In [20]:
```
sns.countplot(x='Age',data=df)
```

Out[20]:
```
<AxesSubplot:xlabel='Age', ylabel='count'>
```



b. Bivariate analysis

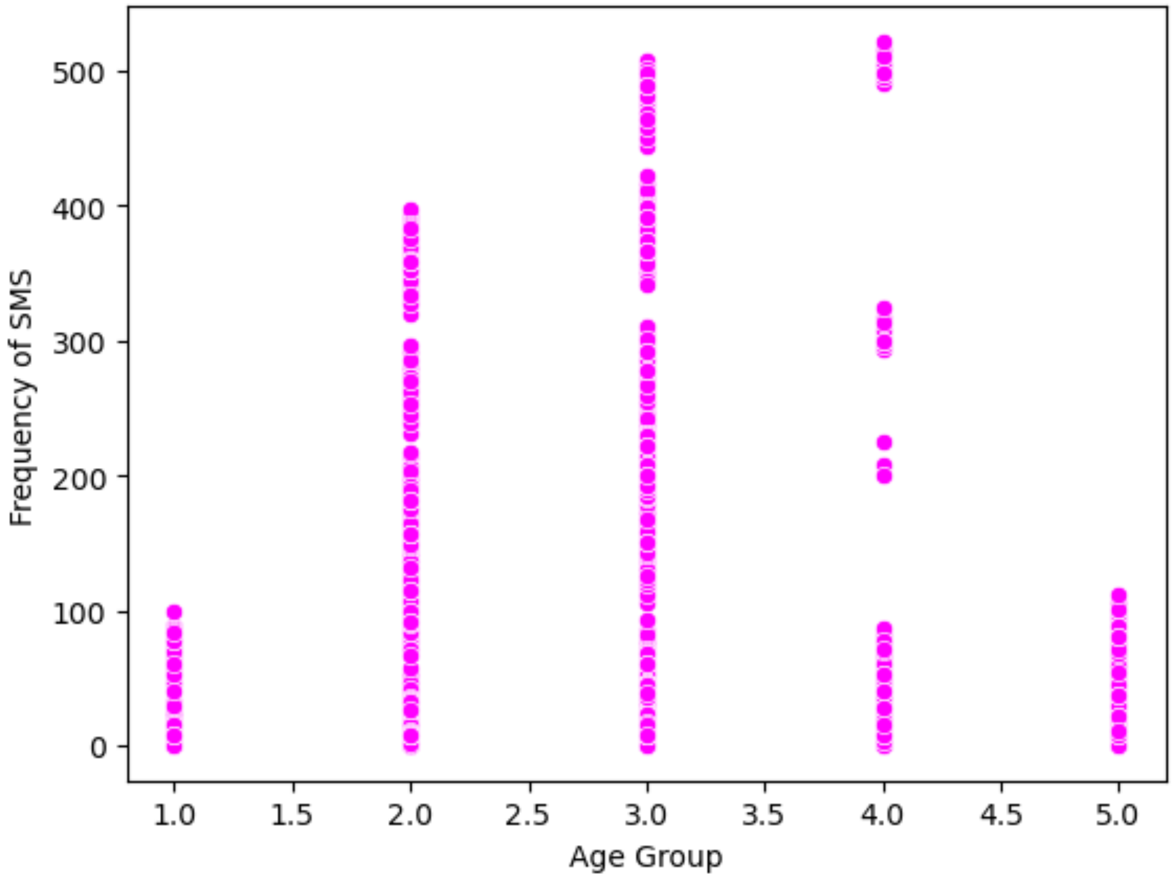
We use two variables and compare them. This way, you can find how one feature affects the other.

In [21]:
```
#Age group that sent SMS the most often

age_group_sms=df.groupby(['Age Group'])['Frequency of SMS'].max()
print(age_group_sms)
```

```
Age Group
1     99
2    398
3    508
4    522
5    112
Name: Frequency of SMS, dtype: int64
```

In [22]:
```
sns.scatterplot(data=df,x='Age Group',y='Frequency of SMS',color='m
```

Out[22]: <AxesSubplot:xlabel='Age Group', ylabel='Frequency of SMS'>

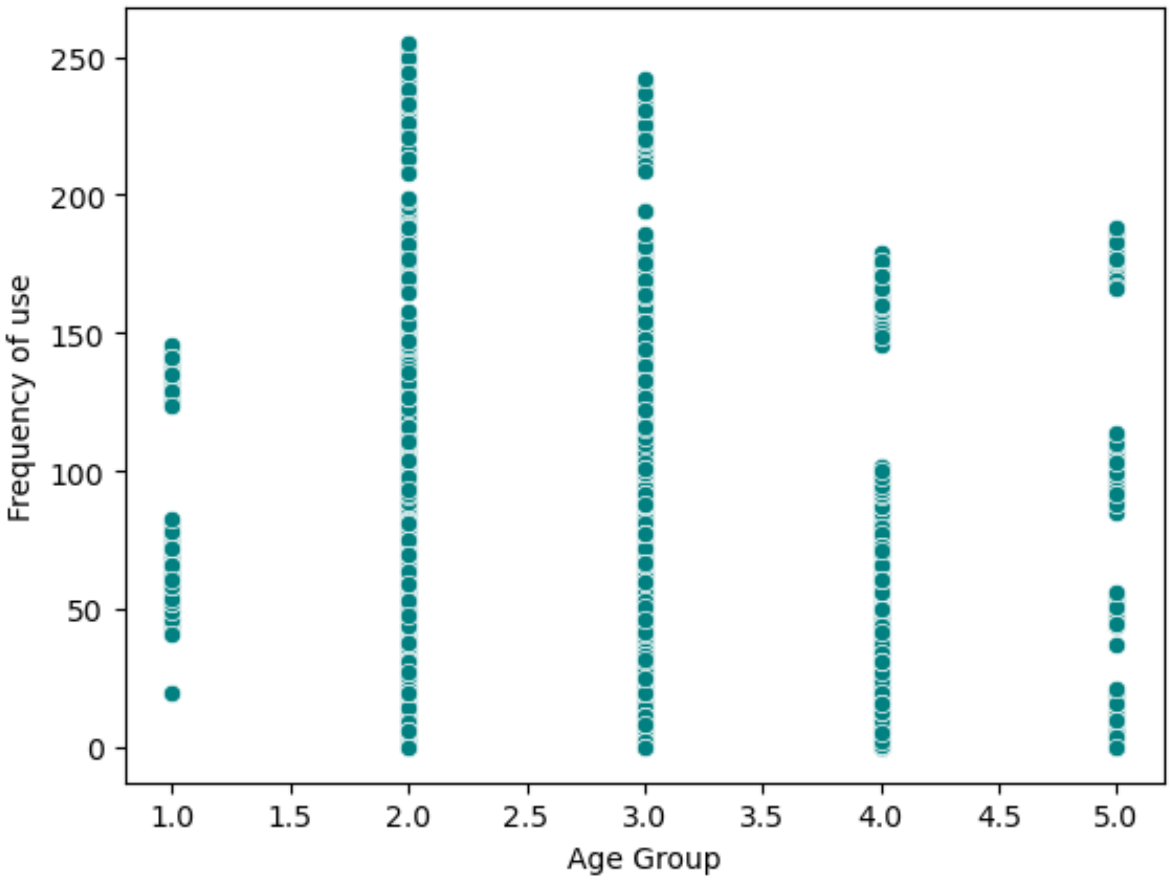


In [23]:
```
#Age group that Call the most often

age_group_call=df.groupby(['Age Group'])['Frequency of use'].max()
print(age_group_call)
```

```
Age Group
1    146
2    255
3    242
4    179
5    188
Name: Frequency of use, dtype: int64
```

In [24]:
```
sns.scatterplot(data=df,x='Age Group',y='Frequency of use',color='t
```

Out[24]:
```
<AxesSubplot:xlabel='Age Group', ylabel='Frequency of use'>
```



In [25]:
```
#Age group that spend the most

age_group_spend=df.groupby(['Age Group'])['Customer Value'].max()
print(age_group_spend)
```

```
Age Group
1     920.315
2    2148.030
3    2165.280
4    1393.850
5     311.040
Name: Customer Value, dtype: float64
```

- Age Group 4 with age 45 did send SMS the most requency
- Age Group 2 with age 25 did Call the most requency
- Age Group 3 with age 30 did spend the most

In [26]:
```
churn_plan=print(len(df[(df["Churn"] == 1) & (df["Tariff Plan"]==1)
```

```
489
```

- Customer mostly use plan 1 (Pay as you go) with 489 customers of 495 or 99% of churn
- Customer use plan 2 (Contractual): 6 customers of 495 or round 1% of Churn

In [27]:
```
churn_plan=(len(df[(df["Churn"] == 0) & (df["Tariff Plan"]==1)]))
#churn_plan=(len(df[(df["Churn"] == 0) & (df["Tariff Plan"]==2)]))
```

In [28]:
```
churn_plan
```

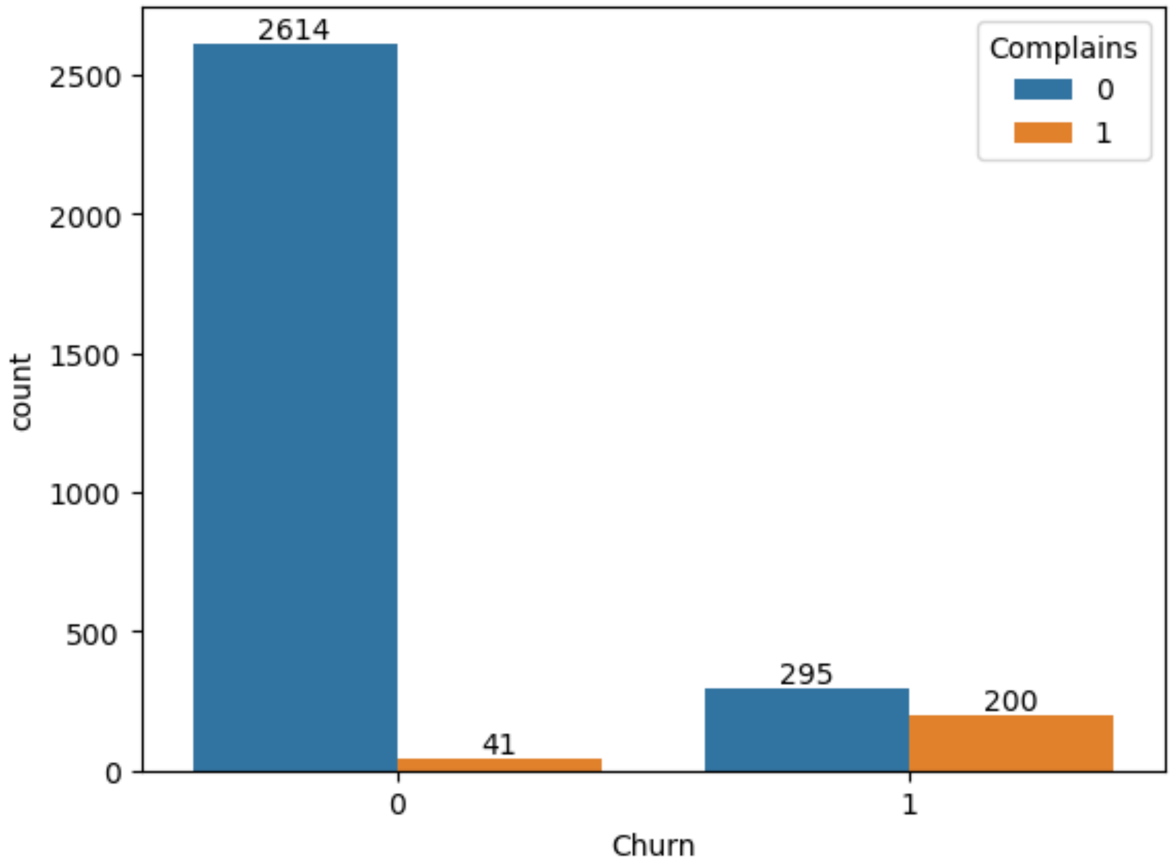
Out[28]: 2416

- 2416 or 77% of all customers that was non-churn using plan 1

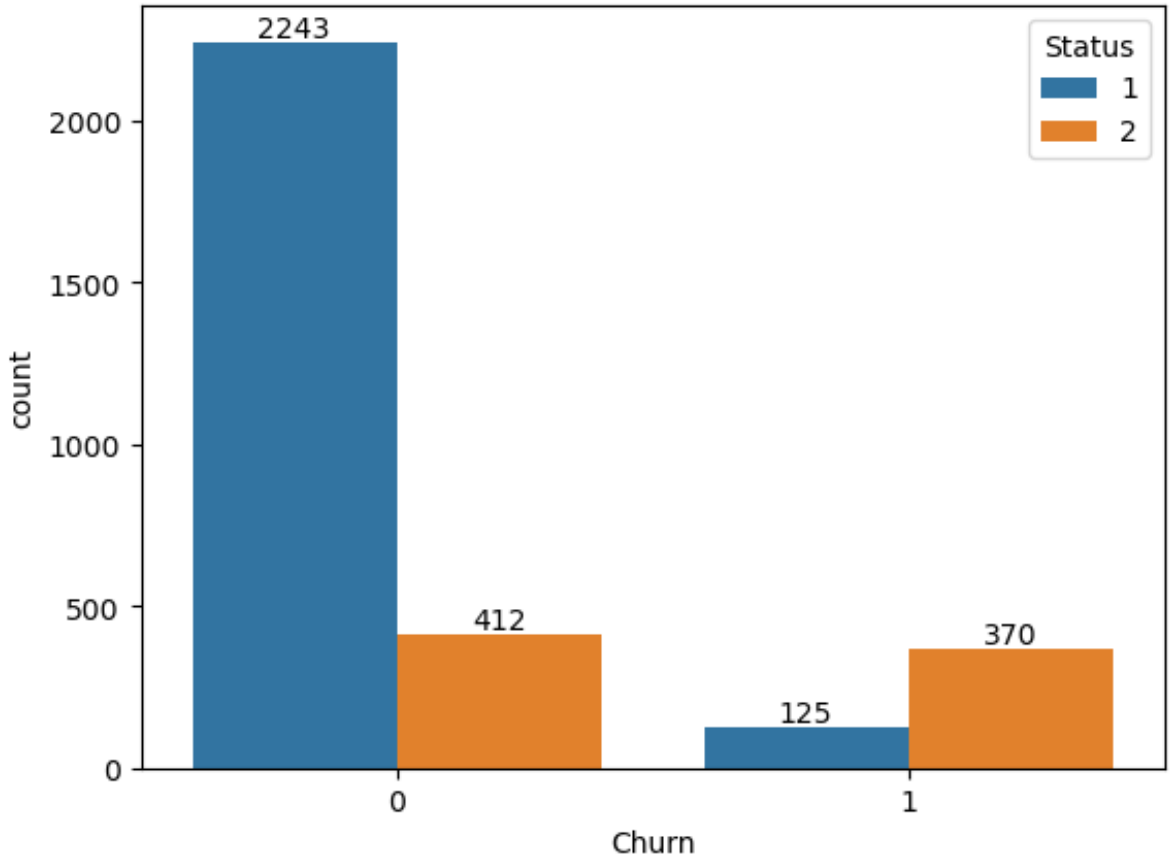
In [29]:
```
churn_complains=(len(df[(df["Churn"] == 1) & (df["Complains"]==1)]))
churn_complains
```

Out[29]: 200

In [30]:
```

complains_plot=sns.countplot(data=df, x="Churn", hue="Complains")
for i in complains_plot.containers:
   complains_plot.bar_label(i,)
```



In [31]:
```
status_churn=sns.countplot(data=df, x="Churn", hue="Status")
for i in status_churn.containers:
   status_churn.bar_label(i,)
```



Distinct Called Numbers

In [32]:
```
df['Distinct Called Numbers'].value_counts()
```

Out[32]:
```
0      154
2       88
10      78
15      77
6       76
      ...
95       1
93       1
88       1
87       1
97       1
Name: Distinct Called Numbers, Length: 92, dtype: int64
```

In [33]:

```
dist_call_min=df.groupby(['Age Group'])['Distinct Called Numbers'].
dist_call_max=df.groupby(['Age Group'])['Distinct Called Numbers'].
dist_call_mean=df.groupby(['Age Group'])['Distinct Called Numbers']
dis_call_class={'dist_call_min':dist_call_min,'dist_call_mean':dist
```

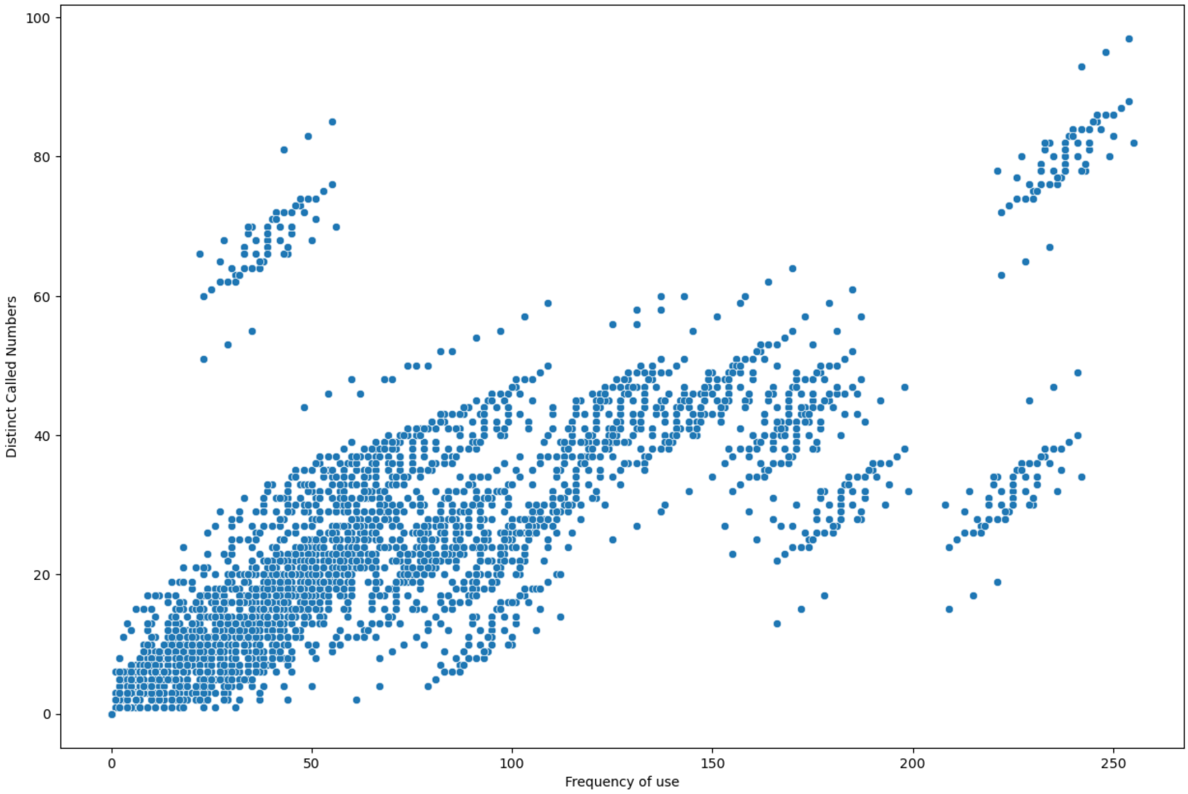
In [34]:

```
table=pd.DataFrame(dis_call_class)
table
```

Out[34]:

Age Group	dist_call_min	dist_call_mean	dist_call_max
1	7	34.325203	55
2	0	22.985535	97
3	0	21.502456	61
4	0	26.086076	83
5	0	29.723529	85

In [35]:
```
#Distinct Called Numbers & Frequency of use
plt.figure(figsize = (15,10))
dis_frecall=sns.scatterplot(data=df, x="Frequency of use", y="Disti
```

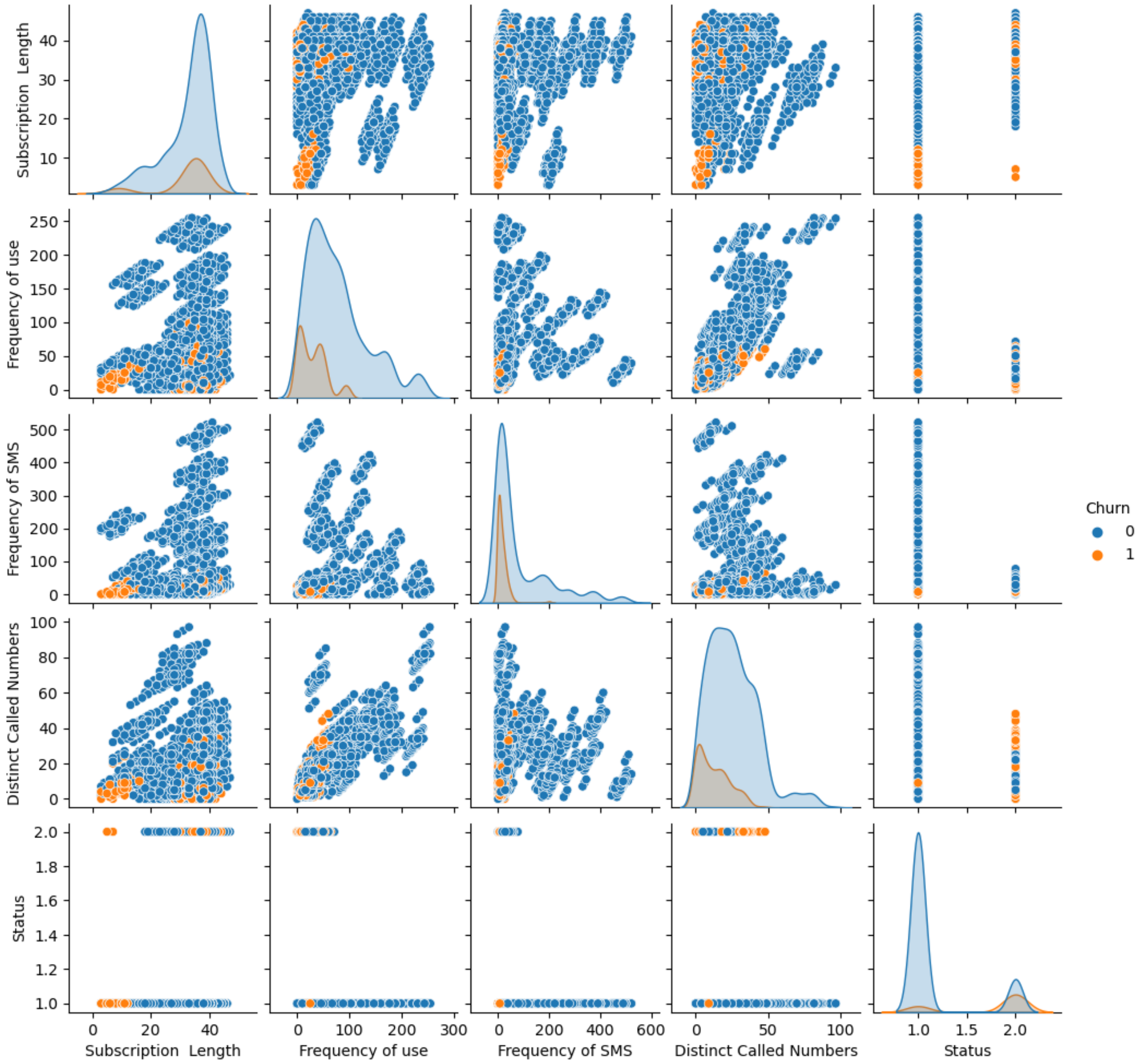


c. Multivariate analysis

In [36]:
```
df_new=df.drop(['Complains','Customer Value','Seconds of Use','Age'
```

In [37]:
```
sns.pairplot(df_new, hue="Churn", height=2)
```

Out[37]: <seaborn.axisgrid.PairGrid at 0x7f90950483a0>

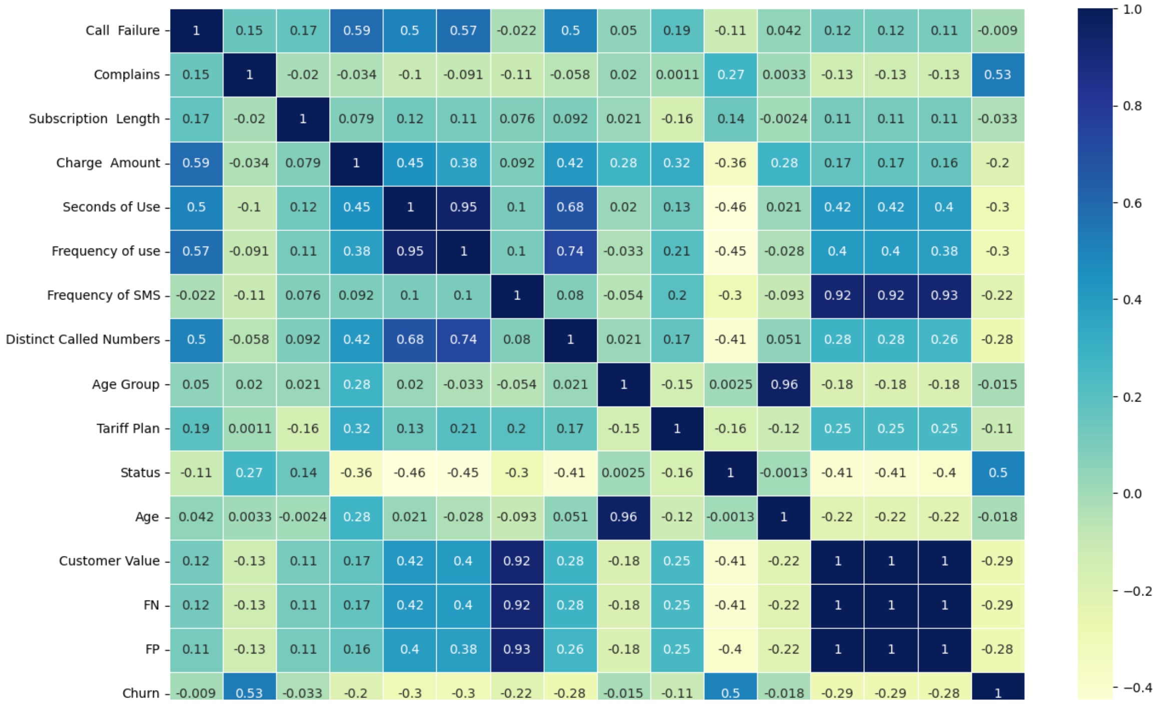


In [ ]:

IV. Feature Engineering

a. Feature selection

In [73]:
```
corr = df.corr()
plt.figure(figsize = (15,10))
dfplot = sns.heatmap(corr, cmap="YlGnBu", annot=True,linewidths=0.5
```



In [81]:
```
#This process calculates the correlations of all the features with
#a threshold of 0.2 was chosen

cor_target = abs(corr["Churn"])
relevant_features = cor_target[cor_target > 0.2]
relevant_features.index
```

Out[81]:
```
Index(['Complains', 'Charge  Amount', 'Seconds of Use', 'Frequency
of use',
       'Frequency of SMS', 'Distinct Called Numbers', 'Status',
       'Customer Value', 'FN', 'FP', 'Churn'],
      dtype='object')
```

V. Model Development

V.1 Model Development with all features

In [82]:
```
#Split Data into Dependent and Independent
#We are create X for data that we want to use to make prediction
#and y which has data that we want to predict

X=df.drop('Churn',axis=1)
```

In [83]:
```
y=df['Churn']
```

In [84]:
```
y.shape
```

Out[84]: (3150,)

In [85]:
```
sum(y)/len(y)
```

Out[85]: 0.15714285714285714

In [86]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,ra
```

In [87]:
```
sum(y_train)/len(y_train)
```

Out[87]: 0.15736961451247167

In [88]:
```
sum(y_test)/len(y_test)
```

Out[88]: 0.15661375661375662

In [89]:
```
y_test.shape
```

Out[89]: (945,)

In [90]:
```
models=[LogisticRegression(),
        RandomForestClassifier(),
        xgb.XGBClassifier(),
       GradientBoostingClassifier()]
```

In [91]:

```
model_names=['LogisticRegression',
            'RandomForestClassifier',
            'XGBClassifier',
            'GradientBoostingClassifier']

acc=[]
auc=[]
pre=[]
re=[]
f1=[]
eval_acc={}


for model in range(len(models)):
    classification_model=models[model]
    classification_model.fit(X_train,y_train)
    pred=classification_model.predict(X_test)
    pred_prob=classification_model.predict_proba(X_test)[:,1] #we w
    acc.append(balanced_accuracy_score(y_test,pred))
    auc.append(roc_auc_score(y_test, pred_prob))
    pre.append(precision_score(pred,y_test))
    re.append(recall_score(pred,y_test))
    f1.append(f1_score(pred,y_test))



eval_acc={'Modelling Algorithm':model_names,
          'Accuracy':acc,
          'AUC':auc,
          'Precision':pre,
          'Recall':re,
          'F1_Score':f1,
         }
eval_acc
```

```
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/sklearn/li
near_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to c
onverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
(https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver opti
ons:
    https://scikit-learn.org/stable/modules/linear_model.html#logi
stic-regression (https://scikit-learn.org/stable/modules/linear_mo
del.html#logistic-regression)
  n_iter_i = _check_optimize_result(
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/sk
learn.py:1224: UserWarning: The use of label encoder in XGBClassif
ier is deprecated and will be removed in a future release. To remo
ve this warning, do the following: 1) Pass option use_label_encode
```

```
r=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_cl
ass - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/da
ta.py:250: FutureWarning: pandas.Int64Index is deprecated and will
be removed from pandas in a future version. Use pandas.Index with
the appropriate dtype instead.
  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):

[20:36:12] WARNING: /var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp
/T/abs_44tbtwf8c1/croots/recipe/xgboost-split_1659548960882/work/s
rc/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like
to restore the old behavior.
```

Out[91]:
```
{'Modelling Algorithm': ['LogisticRegression',
  'RandomForestClassifier',
  'XGBClassifier',
  'GradientBoostingClassifier'],
 'Accuracy': [0.6552146563125233,
  0.8824816033097087,
  0.9316609583234426,
  0.8778485197870391],
 'AUC': [0.8795779782291703,
  0.9841720641595171,
  0.9916875784190715,
  0.9774153074027603],
 'Precision': [0.3581081081081081,
  0.7837837837837838,
  0.8783783783783784,
  0.777027027027027],
 'Recall': [0.5824175824175825,
  0.8854961832061069,
  0.9154929577464789,
  0.8712121212121212],
 'F1_Score': [0.4435146443514645,
  0.8315412186379928,
  0.896551724137931,
  0.8214285714285714]}
```

In [92]:
```
acc_table=pd.DataFrame(eval_acc)
acc_table = acc_table.sort_values(by='F1_Score', ascending=[False])
acc_table
```

Out[92]:

	Modelling Algorithm	Accuracy	AUC	Precision	Recall	F1_Score
2	XGBClassifier	0.931661	0.991688	0.878378	0.915493	0.896552
1	RandomForestClassifier	0.882482	0.984172	0.783784	0.885496	0.831541
3	GradientBoostingClassifier	0.877849	0.977415	0.777027	0.871212	0.821429
0	LogisticRegression	0.655215	0.879578	0.358108	0.582418	0.443515

Check if it is overfitting

In [93]:
```
model_names=['LogisticRegression',
             'RandomForestClassifier',
             'XGBClassifier',
             'GradientBoostingClassifier']

acc_train=[]
auc_train=[]


eval_acc_train={}


for model in range(len(models)):
    classification_model=models[model]
    classification_model.fit(X_train,y_train)
    pred=classification_model.predict(X_train)
    pred_prob=classification_model.predict_proba(X_train)[:,1] #we
    acc_train.append(balanced_accuracy_score(y_train,pred))
    auc_train.append(roc_auc_score(y_train, pred_prob))




eval_acc_train={'Modelling Algorithm':model_names,
          'Accuracy_Train':acc_train,
          'AUC_Train':auc_train,

         }
eval_acc_train
```

```
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/sklearn/li
near_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to c
onverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html
(https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver opti
ons:
    https://scikit-learn.org/stable/modules/linear_model.html#logi
stic-regression (https://scikit-learn.org/stable/modules/linear_mo
del.html#logistic-regression)
  n_iter_i = _check_optimize_result(
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/sk
learn.py:1224: UserWarning: The use of label encoder in XGBClassif
ier is deprecated and will be removed in a future release. To remo
ve this warning, do the following: 1) Pass option use_label_encode
r=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_cl
ass - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/da
ta.py:250: FutureWarning: pandas.Int64Index is deprecated and will
be removed from pandas in a future version. Use pandas.Index with
the appropriate dtype instead.
  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):

[20:36:56] WARNING: /var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp
/T/abs_44tbtwf8c1/croots/recipe/xgboost-split_1659548960882/work/s
rc/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like
to restore the old behavior.
```

Out[93]:

```
{'Modelling Algorithm': ['LogisticRegression',
  'RandomForestClassifier',
  'XGBClassifier',
  'GradientBoostingClassifier'],
 'Accuracy_Train': [0.6453322806897814,
  0.985147954324783,
  0.985147954324783,
  0.9258180994717136],
 'AUC_Train': [0.8755959896141927,
  0.9997277913408176,
  0.9997339955267819,
  0.9942494951343671]}
```

In [94]:
```
acc_table=pd.DataFrame(eval_acc_train)
acc_table = acc_table.sort_values(by='Accuracy_Train', ascending=[F
acc_table
```

Out[94]:

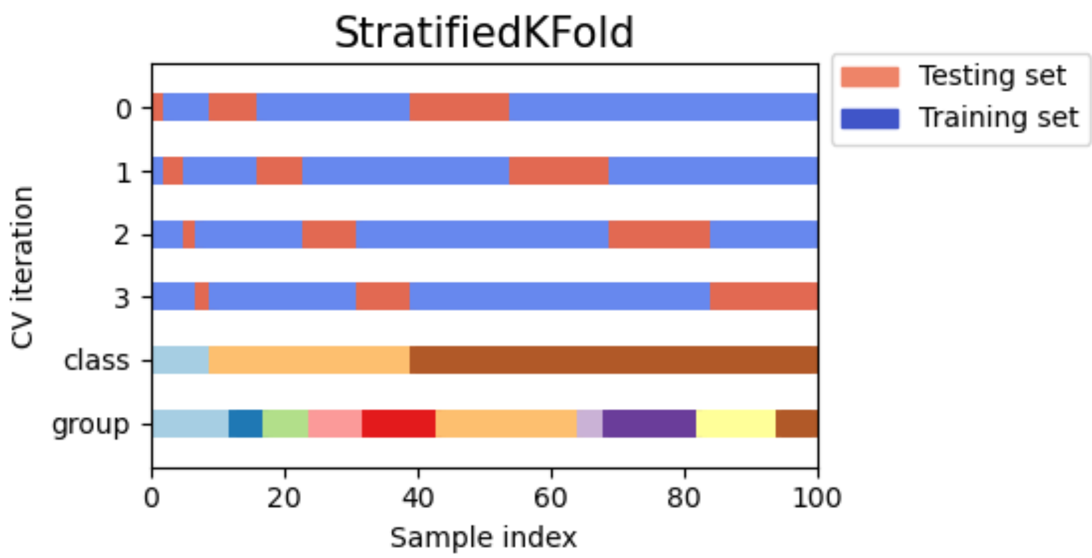
	Modelling Algorithm	Accuracy_Train	AUC_Train
1	RandomForestClassifier	0.985148	0.999728
2	XGBClassifier	0.985148	0.999734
3	GradientBoostingClassifier	0.925818	0.994249
0	LogisticRegression	0.645332	0.875596

Overfitting!!!

Use cross Validation!

1. Small/Imbalanced we use Stratified

Using Stratified k-fold¶



In [101]:
```
models=[LogisticRegression(),
        RandomForestClassifier(),
        xgb.XGBClassifier(),
       GradientBoostingClassifier()]
```

In [102]:
```
sk=StratifiedKFold(n_splits=5,shuffle=True,random_state=529)
```

In [103]:
```
sk.split(X,y)
```
Out[103]:
```
<generator object _BaseKFold.split at 0x7f9094849040>
```

In [104]:
```
for train_idx, val_idx in sk.split(X_train,y_train):
    X_train=X.loc[train_idx]
    y_train=y.loc[train_idx]

    X_val=X.loc[val_idx]
    y_val=y.loc[val_idx]
```

In [118]:
```
X_train.shape
```
Out[118]:
```
(1412, 15)
```

In [119]:
```
X_val.shape
```
Out[119]:
```
(352, 15)
```

In [105]:
```


model_names=['LogisticRegression',
             'RandomForestClassifier',
             'XGBClassifier',
             'GradientBoostingClassifier']

acc_val=[]
auc_val=[]
f1=[]
eval_acc_val={}

for model in range(len(models)):
    classification_model=models[model]
    classification_model.fit(X_train,y_train)
    pred=classification_model.predict(X_val)
    pred_prob=classification_model.predict_proba(X_val)[:,1] #we wa
    acc_val.append(balanced_accuracy_score(y_val,pred))
    auc_val.append(roc_auc_score(y_val, pred_prob))
    f1.append(f1_score(y_val,pred))


eval_acc_val={'Modelling Algorithm':model_names,
          'Accuracy_StrfKFold':acc_val,
          'AUC_StrfKFold':auc_val,
          'F1_Score':f1,

         }
eval_acc_val
```

```
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/sklearn/li
near_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to c
onverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
(https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver opti
ons:
    https://scikit-learn.org/stable/modules/linear_model.html#logi
stic-regression (https://scikit-learn.org/stable/modules/linear_mo
del.html#logistic-regression)
  n_iter_i = _check_optimize_result(
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/sk
learn.py:1224: UserWarning: The use of label encoder in XGBClassif
ier is deprecated and will be removed in a future release. To remo
ve this warning, do the following: 1) Pass option use_label_encode
r=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_cl
ass - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/da
ta.py:250: FutureWarning: pandas.Int64Index is deprecated and will
be removed from pandas in a future version. Use pandas.Index with
the appropriate dtype instead.
  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):

[20:50:46] WARNING: /var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp
/T/abs_44tbtwf8c1/croots/recipe/xgboost-split_1659548960882/work/s
rc/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like
to restore the old behavior.
```

Out[105]:

```
{'Modelling Algorithm': ['LogisticRegression',
  'RandomForestClassifier',
  'XGBClassifier',
  'GradientBoostingClassifier'],
 'Accuracy_StrfKFold': [0.5656853999405294,
  0.8336306868867083,
  0.887957181088314,
  0.8774903360095153],
 'AUC_StrfKFold': [0.8437109723461196,
  0.9751115075825156,
  0.9815640796907523,
  0.9758846268212906],
 'F1_Score': [0.24999999999999994,
  0.7722772277227723,
  0.8490566037735849,
  0.8301886792452831]}
```

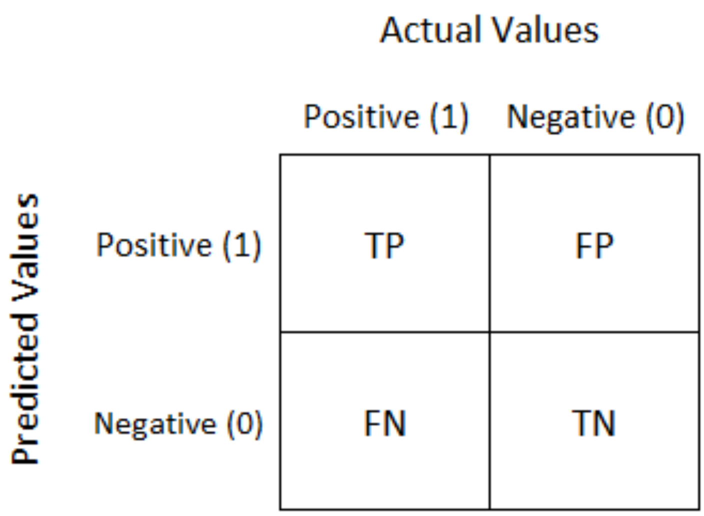
In [106]:
```
acc_table=pd.DataFrame(eval_acc_val)
acc_table = acc_table.sort_values(by='F1_Score', ascending=[False])
acc_table
```

Out[106]:

	Modelling Algorithm	Accuracy_StrfKFold	AUC_StrfKFold	F1_Score
2	XGBClassifier	0.887957	0.981564	0.849057
3	GradientBoostingClassifier	0.877490	0.975885	0.830189
1	RandomForestClassifier	0.833631	0.975112	0.772277
0	LogisticRegression	0.565685	0.843711	0.250000

We will choose XGBoost as out churn prediction model

plot_confusion_matrix() from Cross Validation



In [110]:
```
xgb_clf=xgb.XGBClassifier()
xgb_clf.fit(X_train,y_train)
```

```
[20:58:49] WARNING: /var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp
/T/abs_44tbtwf8c1/croots/recipe/xgboost-split_1659548960882/work/s
rc/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like
to restore the old behavior.
```

Out[110]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=
1,
              colsample_bynode=1, colsample_bytree=1, enable_categ
orical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.30000001
2,
              max_delta_step=0, max_depth=6, min_child_weight=1, m
issing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=
8,
              num_parallel_tree=1, predictor='auto', random_state=
0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsa
mple=1,
              tree_method='exact', validate_parameters=1, verbosit
y=None)
```

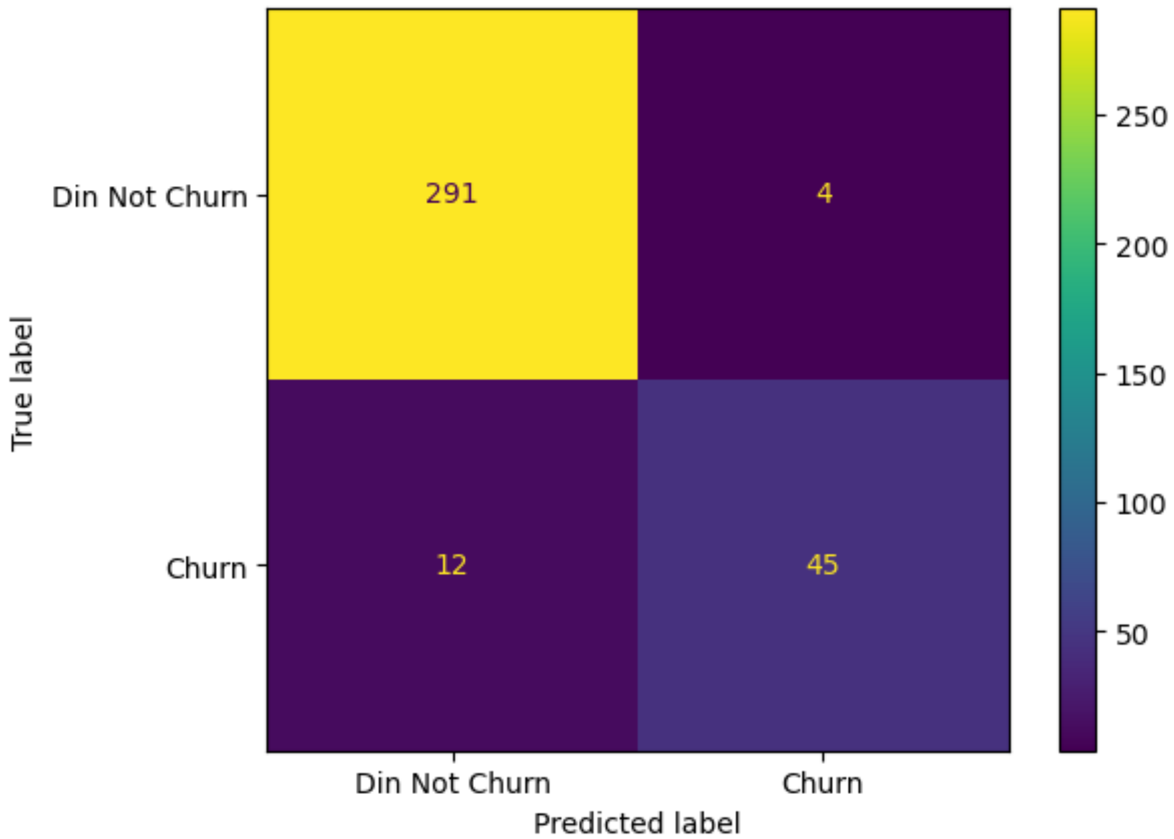
In [117]:
```
pred_xgb=xgb_clf.predict(X_val)
pred_xgb_prob=xgb_clf.predict_proba(X_val)[:,1] #we want predict on
print(balanced_accuracy_score(y_val,pred_xgb))
print(roc_auc_score(y_val,pred_xgb_prob))
print(classification_report(y_val,pred_xgb))
print(confusion_matrix(y_val,pred_xgb))
```

```
0.887957181088314
0.9815640796907523
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       295
           1       0.92      0.79      0.85        57

    accuracy                           0.95       352
   macro avg       0.94      0.89      0.91       352
weighted avg       0.95      0.95      0.95       352

[[291   4]
 [ 12  45]]
```

In [114]:
```
plot_confusion_matrix(xgb_clf,
                      X_val,
                      y_val,
                      values_format='d',
                      display_labels=['Din Not Churn','Churn'])
```

```
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/sklearn/ut
ils/deprecation.py:87: FutureWarning: Function plot_confusion_matr
ix is deprecated; Function `plot_confusion_matrix` is deprecated i
n 1.0 and will be removed in 1.2. Use one of the class methods: Co
nfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.fr
om_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[114]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f90942ffc10>
```

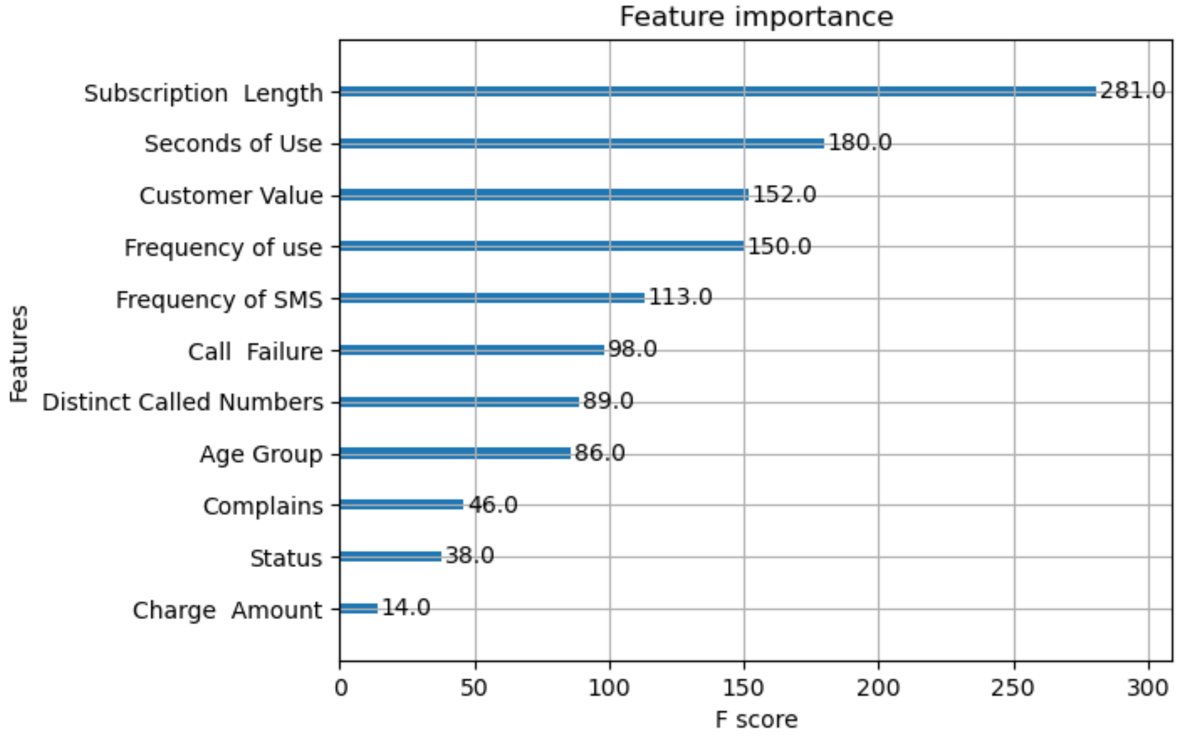


In [ ]:

Important Features

In [120]:
```
from xgboost import plot_importance
import matplotlib.pyplot as plt

model=xgb.XGBClassifier()
model.fit(X_train,y_train)
# plot feature importance
plot_importance(model)
plt.show()
```

```
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/sk
learn.py:1224: UserWarning: The use of label encoder in XGBClassif
ier is deprecated and will be removed in a future release. To remo
ve this warning, do the following: 1) Pass option use_label_encode
r=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_cl
ass - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/da
ta.py:250: FutureWarning: pandas.Int64Index is deprecated and will
be removed from pandas in a future version. Use pandas.Index with
the appropriate dtype instead.
  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):

[21:17:38] WARNING: /var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp
/T/abs_44tbtwf8c1/croots/recipe/xgboost-split_1659548960882/work/s
rc/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like
to restore the old behavior.
```



V.2 Reduce Features is effective to our models???

In [161]:
```
new_df=df[['Complains', 'Charge  Amount', 'Seconds of Use', 'Freque
       'Frequency of SMS', 'Distinct Called Numbers', 'Status',
       'Customer Value', 'FN', 'FP', 'Churn']]
```

In [162]:
```
new_df
```

Out[162]:

	Complains	Charge Amount	Seconds of Use	Frequency of use	Frequency of SMS	Distinct Called Numbers	Status	Customer Value
0	0	0	4370	71	5	17	1	197.640
1	0	0	318	5	7	4	2	46.035
2	0	0	2453	60	359	24	1	1536.520 1
3	0	0	4198	66	1	35	1	240.020
4	0	0	2393	58	2	33	1	145.805
...	...	...	...	...	...	...	...	...
3145	0	2	6697	147	92	44	1	721.980
3146	0	1	9237	177	80	42	1	261.210
3147	0	4	3157	51	38	21	1	280.320
3148	0	2	4695	46	222	12	1	1077.640
3149	1	2	1792	25	7	9	1	100.680

3150 rows × 11 columns

In [163]:
```
x=new_df.drop('Churn',axis=1)
y=new_df['Churn']
```

In [164]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,ra
```

In [165]:
```
for train_idx, val_idx in sk.split(x_train,y_train):
    x_train=x.loc[train_idx]
    y_train=y.loc[train_idx]

    x_val=x.loc[val_idx]
    y_val=y.loc[val_idx]
```

In [166]:

```
model_names=['LogisticRegression',
             'RandomForestClassifier',
             'XGBClassifier',
             'GradientBoostingClassifier']

acc_v=[]
auc_v=[]
f1=[]
eval_acc_v={}

for model in range(len(models)):
    classification_model=models[model]
    classification_model.fit(x_train,y_train)
    pred=classification_model.predict(x_val)
    pred_prob=classification_model.predict_proba(x_val)[:,1] #we wa
    acc_v.append(balanced_accuracy_score(y_val,pred))
    auc_v.append(roc_auc_score(y_val, pred_prob))
    f1.append(f1_score(y_val,pred))


eval_acc_v={'Modelling Algorithm':model_names,
           'Accuracy_StrfKFold':acc_v,
           'AUC_StrfKFold':auc_v,
           'F1 Score':f1,

         }
eval_acc_v
```

```
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/sklearn/li
near_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to c
onverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
(https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver opti
ons:
    https://scikit-learn.org/stable/modules/linear_model.html#logi
stic-regression (https://scikit-learn.org/stable/modules/linear_mo
del.html#logistic-regression)
  n_iter_i = _check_optimize_result(
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/sk
learn.py:1224: UserWarning: The use of label encoder in XGBClassif
ier is deprecated and will be removed in a future release. To remo
ve this warning, do the following: 1) Pass option use_label_encode
r=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_cl
ass - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
/Users/roatny/opt/anaconda3/lib/python3.9/site-packages/xgboost/da
ta.py:250: FutureWarning: pandas.Int64Index is deprecated and will
```

```
be removed from pandas in a future version. Use pandas.Index with
the appropriate dtype instead.
  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):

[21:59:31] WARNING: /var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp
/T/abs_44tbtwf8c1/croots/recipe/xgboost-split_1659548960882/work/s
rc/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed f
rom 'error' to 'logloss'. Explicitly set eval_metric if you'd like
to restore the old behavior.
```

Out[166]:

```
{'Modelling Algorithm': ['LogisticRegression',
  'RandomForestClassifier',
  'XGBClassifier',
  'GradientBoostingClassifier'],
 'Accuracy_StrfKFold': [0.687466307277628,
  0.8552560646900269,
  0.8641509433962264,
  0.8110512129380054],
 'AUC_StrfKFold': [0.9138814016172506,
  0.9708124759337697,
  0.9660184828648442,
  0.965979976896419],
 'F1 Score': [0.5346534653465347,
  0.7761194029850748,
  0.7714285714285715,
  0.7131782945736433]}
```

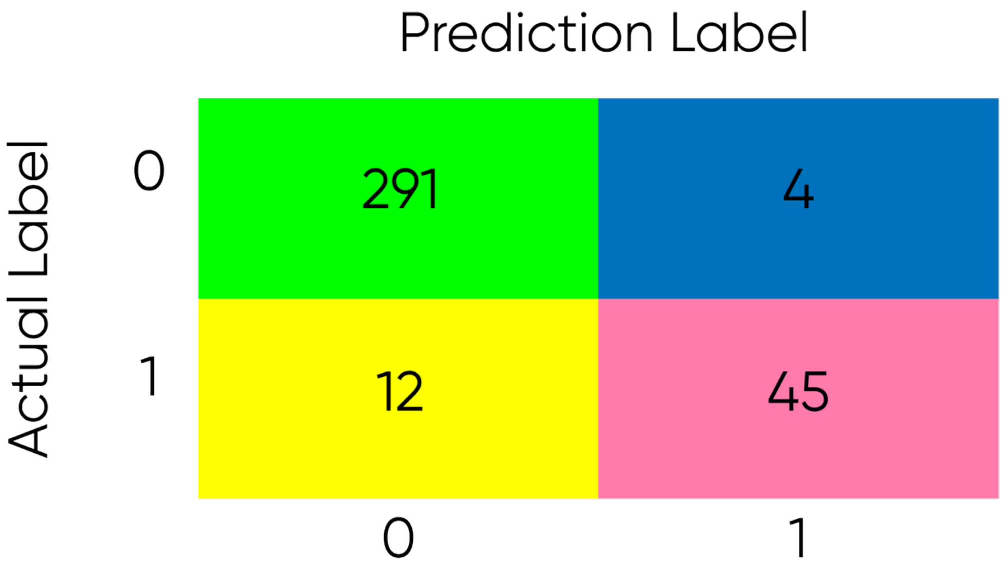
In [167]:

```
acc_table=pd.DataFrame(eval_acc_v)
acc_table = acc_table.sort_values(by='Accuracy_StrfKFold', ascendin
acc_table
```

Out[167]:

	Modelling Algorithm	Accuracy_StrfKFold	AUC_StrfKFold	F1 Score
2	XGBClassifier	0.864151	0.966018	0.771429
1	RandomForestClassifier	0.855256	0.970812	0.776119
3	GradientBoostingClassifier	0.811051	0.965980	0.713178
0	LogisticRegression	0.687466	0.913881	0.534653

VI. Conclusions and Recommendations



There are four machine learning algorithms in our study, and XGBoost is the most powerfull for build churn model based on F1 and AUC as well.

XGBoost had higtest accuracy 0.86 with AUC 0.96 and F1 score 0.77, then followed by Random Forest and GradientBoosting.

According confusion matrix, 45 people predict Churn correctly of 57 people churn. And total of people leave 295, we predict 291 left the company were correctly classified.

In this study, we do not confus on improving accuracy, we try to find out probability or how likely people are going to left, and investigate important features. As result, we see that Subscription Length is the most important features on our model, followed by Second of Use, Customer Value, and Frequency of Use.

Reducing features might not affect much on our model, so it is working well with defual features. In churn rate, imbalance data is normal, we want to keep them as the way it is, just try to find algorithms that can handle well with imbalance data like XGBoost and Stratified KFold metrix.

In brief, Churn is extremely difficult to fight, but we can use ML to provide actionable insight, current situation of customers to help business make better desicion on strategies.

In next study, we will look closely on customer behaviors, how they impact churn rate.