In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings ("ignore")
```

In [2]:
```
churn_df=pd.read_csv(r"C:\Users\s323\Desktop\Gatherings\Data Science\ML\Amit Mishra
churn_df_test=pd.read_csv(r"C:\Users\s323\Desktop\Gatherings\Data Science\ML\Amit M
```

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

Out[3]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122

In [4]:
```
churn_df_test.head()
```

Out[4]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls
0	LA	117	408	No	No	0	184.5	97	31.37	351.6	80
1	IN	65	415	No	No	0	129.1	137	21.95	228.5	83
2	NY	161	415	No	No	0	332.9	67	56.59	317.8	97
3	SC	111	415	No	No	0	110.4	103	18.77	137.3	102
4	HI	49	510	No	No	0	119.3	117	20.28	215.1	109

In [5]:
```
churn_df.shape
```

Out[5]:
```
(2666, 20)
```

In [6]:
```
churn_df_test.shape
```

Out[6]:
```
(667, 20)
```

Data Wrangling/Manipulation is the first step in ML

In [7]:
```
churn_df.isnull().sum()
```

Out[7]:
```
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
Customer service calls    0
Churn                     0
dtype: int64
```

No missing values are there

In [8]:
```
# For test data
churn_df_test.isnull().sum()
```

Out[8]:
```
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
Customer service calls    0
Churn                     0
dtype: int64
```

2nd step- Data Preprocessing - Cateogrical data to numerical data

- International Plan
- Voice mail plan

In [9]:
```
churn_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2666 entries, 0 to 2665
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   State                   2666 non-null   object 
 1   Account length          2666 non-null   int64  
 2   Area code               2666 non-null   int64  
 3   International plan      2666 non-null   object 
 4   Voice mail plan         2666 non-null   object 
 5   Number vmail messages   2666 non-null   int64  
 6   Total day minutes       2666 non-null   float64
 7   Total day calls         2666 non-null   int64  
 8   Total day charge        2666 non-null   float64
 9   Total eve minutes       2666 non-null   float64
 10  Total eve calls         2666 non-null   int64  
 11  Total eve charge        2666 non-null   float64
 12  Total night minutes     2666 non-null   float64
 13  Total night calls       2666 non-null   int64  
 14  Total night charge      2666 non-null   float64
 15  Total intl minutes      2666 non-null   float64
 16  Total intl calls        2666 non-null   int64  
 17  Total intl charge       2666 non-null   float64
 18  Customer service calls  2666 non-null   int64  
 19  Churn                   2666 non-null   bool   
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 398.5+ KB
```

In [10]:
```
# Label Encoder is used for mapping categorical data into 0, 1, 2, 3, 4...
from sklearn.preprocessing import LabelEncoder
le_encoder=LabelEncoder()
```

In [11]:
```
le_encoder.fit(churn_df["International plan"])
```

Out[11]:
```
LabelEncoder()
```

Since it is fitted now transform

In [12]:
```
churn_df["International plan"]=le_encoder.transform(churn_df["International plan"]
```

It is mandatory to replace into the existing data frame- It will result yes no value into 0/1

- other ways are pd.get_numeric, One hot encoding

In [13]:
```
# we have to fit data once to make label encoder to understand the data
# Test data- we had to labeling now
```

In [14]:
```
churn_df_test["International plan"]=le_encoder.transform(churn_df_test["Internatio
```

Now we will fit our model on voice mail plan and then transform- first train and then apply change

In [15]:
```
le_encoder.fit(churn_df["Voice mail plan"])
```

Out[15]:
```
LabelEncoder()
```

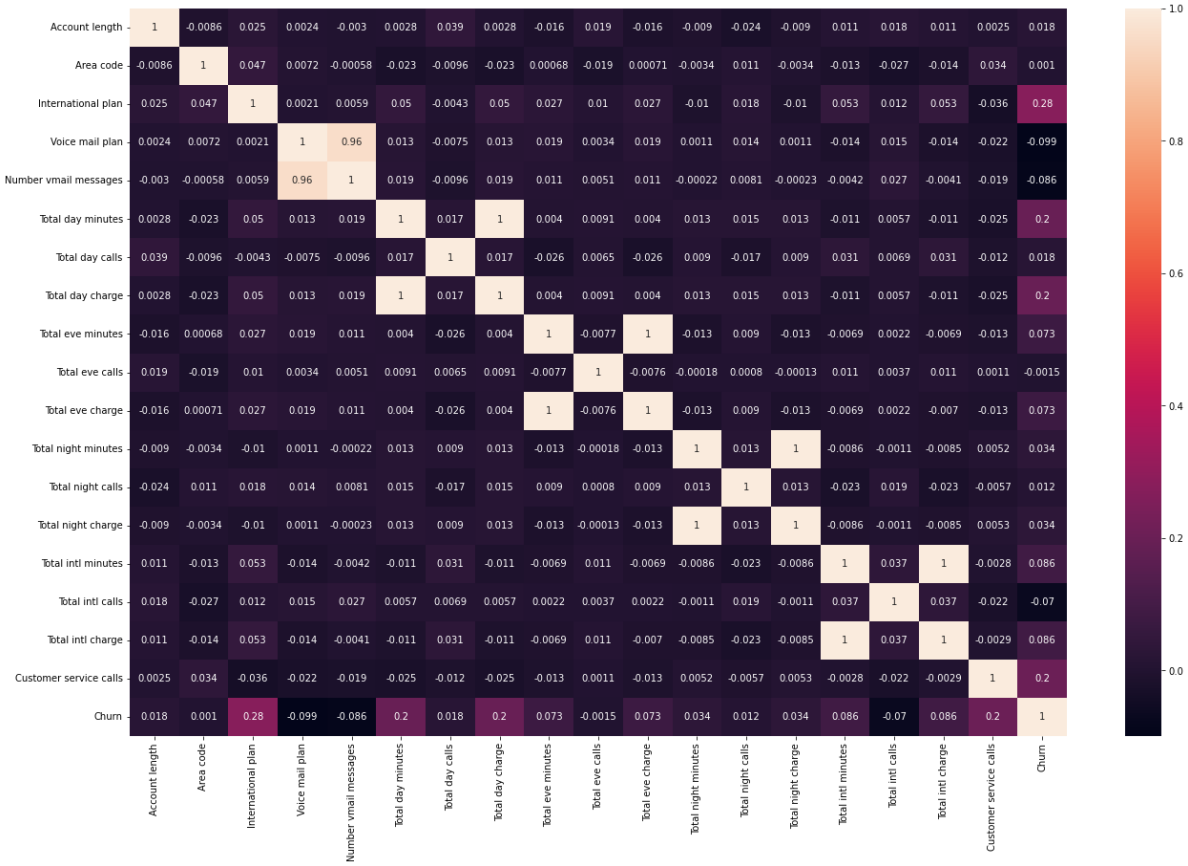
In [16]:
```
churn_df["Voice mail plan"]=le_encoder.transform(churn_df["Voice mail plan"])
```

In [17]:
```
churn_df_test["Voice mail plan"]=le_encoder.transform(churn_df_test["Voice mail pla
```

Co-relation: Heatmap - Feature selection

In [18]:
```
plt.figure(figsize=(22,14))
sns.heatmap(churn_df.corr("pearson"),annot=True)
```

Out[18]:
```
<AxesSubplot:>
```

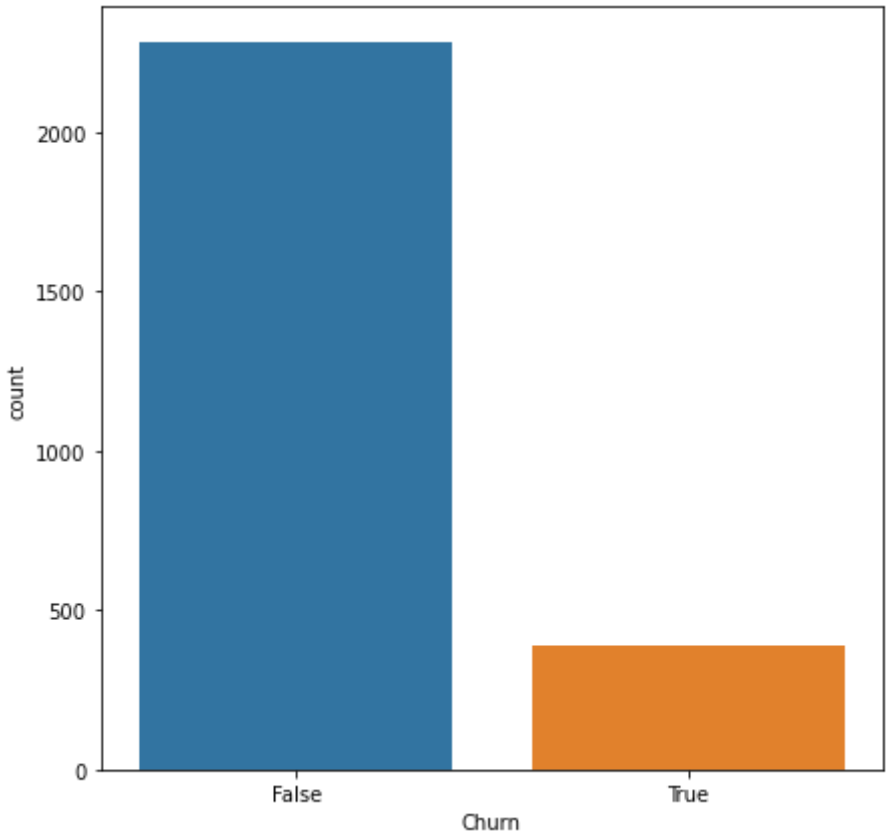


In [19]:
```
# ( Total International charge,mins)(total int mins,total international chr)(night
```

Data Augementation - Issues is sometimes in case of classification data- Look at the distribution of data

In [20]:
```
plt.figure(figsize=(7,7))
sns.countplot(x= "Churn", data=churn_df)
```

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



Imbalanced class- Data is imbalanced, majority of data is of class 0, very few of data is class:1

- We had to increase the no of samples for true datasets

In [21]:
```
churn_df["Churn"].value_counts()
```

Out[21]:
```
False    2278
True      388
Name: Churn, dtype: int64
```

As per above no's obtained - we will do resampling of data

- It is done by Bootstraping method to regenrate random samples for each class

In [22]:
```
from sklearn.utils import resample
# df_0 denotes class 0, how to filter the data as per the value

df_0 = churn_df[churn_df['Churn'] == False]
df_1 = churn_df[churn_df['Churn'] == True]
```

In [23]:
```
#Now apply resampling to our existing dataset, which had less class, as per above v
#Increased =  1000 more samples
df_1_unsample=resample(df_1,n_samples=1388,replace= True, random_state=123)
```

In [24]:
```
#now, let's merge.

churn_df1=pd.concat([df_0,df_1_unsample])
```

In [25]:
```
churn_df1["Churn"].value_counts()
```

Out[25]:
```
False    2278
True     1388
Name: Churn, dtype: int64
```

In [26]:
```
# To convert into percentage
churn_df1["Churn"].value_counts()/churn_df1.shape[0]
```

Out[26]:
```
False    0.621386
True     0.378614
Name: Churn, dtype: float64
```

In [27]:
```
churn_df1.columns
```

Out[27]:
```
Index(['State', 'Account length', 'Area code', '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')
```

Features

In [28]:
```
X = churn_df1.drop(['State','Number vmail messages','Total day charge','Total eve
                    'Total intl minutes','Churn'], axis = 1)
```

In [29]:
```
Y = churn_df1["Churn"]
```

Train and Test Split

In [30]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test= train_test_split(X,Y,test_size=0.2, random_state=8)
```

Decision Tree Classifier

In [31]:
```
from sklearn.tree import DecisionTreeClassifier
# maxinum depth of trees is 19,hypermaters we had to add
# class weight not will be always used, where there will be inbalance will do it
# class_weight = balance the class distribution {class 1 - 62%, class 0 - 38% }
# pre-pruining is implemented using max_depth = 12
clf_tree=DecisionTreeClassifier(criterion = "entropy",random_state=0, max_depth=12,
```

In [32]:
```
clf_tree.fit(x_train,y_train)
```

Out[32]:
```
DecisionTreeClassifier(class_weight={False: 0.38, True: 0.62},
                       criterion='entropy', max_depth=12, random_state=0)
```

In [33]:
```
clf_tree.get_depth()
```

Out[33]:
```
12
```

In [34]:
```
# score function is used just to understand r_2 of the prediction or basis idea mec
# it will calculate how much exact is the model correct now, again will check how l
clf_tree.score(x_test,y_test)
```

Out[34]:
```
0.9495912806539509
```

In [35]:
```
clf_tree.score(x_train, y_train)
```

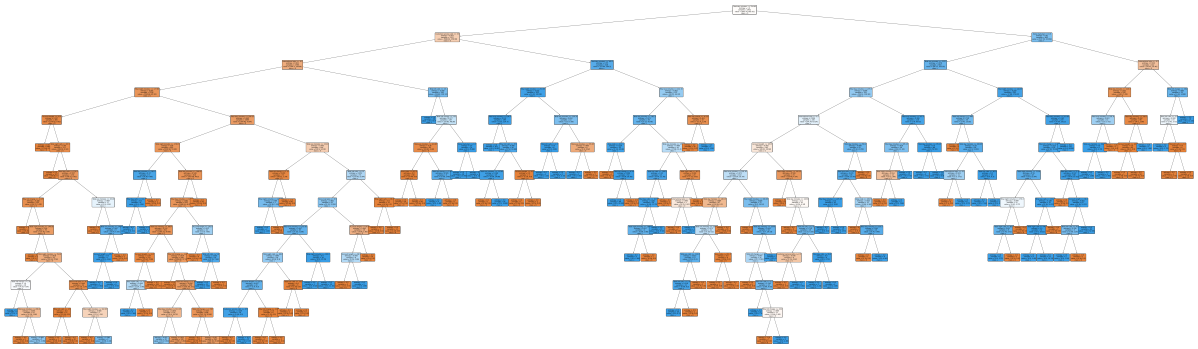
Out[35]:
```
0.9805593451568895
```

Plot Decision Tree

In [36]:
```
clf_tree
```

Out[36]:
```
DecisionTreeClassifier(class_weight={False: 0.38, True: 0.62},
                       criterion='entropy', max_depth=12, random_state=0)
```

In [37]:
```
from sklearn import tree
fig = plt.figure(figsize=(130,40))
# we can also use- tree.plot_tree
tree.plot_tree(clf_tree, feature_names=X.columns, class_names= str(churn_df1.Churn
                   fontsize = 9.5)
plt.show()
```



Classification Metrics

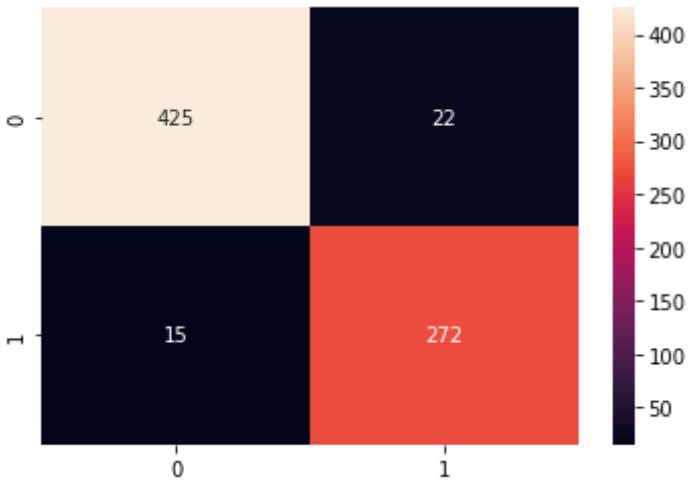
In [38]:
```
# predicted o/p will be 1 or 0
predictions = clf_tree.predict(x_test)
```

In [39]:
```
from sklearn.metrics import confusion_matrix, classification_report
confusion_matrix(y_test, predictions)
```

Out[39]:
```
array([[425,  22],
       [ 15, 272]], dtype=int64)
```

In [40]:
```
plt.figure(figsize = (6,4))
sns.heatmap(confusion_matrix(y_test, predictions), annot=True, fmt = '0.0f')
```

Out[40]:
```
<AxesSubplot:>
```



In [41]:
```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

       False       0.97      0.95      0.96       447
        True       0.93      0.95      0.94       287

    accuracy                           0.95       734
   macro avg       0.95      0.95      0.95       734
weighted avg       0.95      0.95      0.95       734
```

Ensemble Learning - Random Forest

In [42]:
```
from sklearn.ensemble import RandomForestClassifier
clf_rf=RandomForestClassifier(bootstrap=True, criterion='entropy', n_estimators=20(
```

In [43]:
```
clf_rf.fit(x_train,y_train)
```

Out[43]:
```
RandomForestClassifier(criterion='entropy', n_estimators=200)
```

In [44]:
```
clf_rf.score(x_train,y_train)
```

Out[44]:
```
1.0
```

In [45]:
```
clf_rf.score(x_test,y_test)
```

Out[45]:
```
0.9931880108991825
```

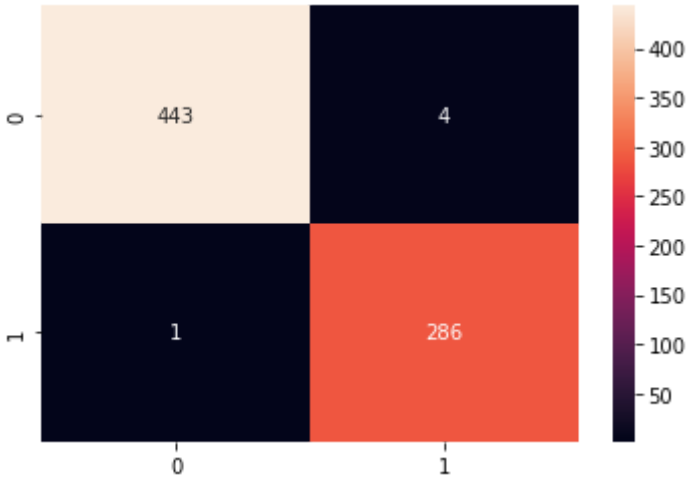
In [46]:
```
predictions=clf_rf.predict(x_test)
```

In [47]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, predictions)
```

Out[47]:
```
array([[443,   4],
       [  1, 286]], dtype=int64)
```

In [51]:
```
sns.heatmap(confusion_matrix(y_test, predictions),annot=True,fmt = '0.0f')
```

Out[51]:
```
<AxesSubplot:>
```



In [ ]: