MAJOR PROJECT REPORT
CUSTOMER CHURN PREDICTION

Name: Avinash R C

Group: 7

- **Objective:**

  The objective of this project is to create a ML model for predicting the customer churn by taking use of the given 'train' and 'test' datasets.

- **Tools Used:**
  1. **Platform:**
     a. Jupyter Notebook
  2. **Language Used:**
     a. Python
  3. **Libraries:**
     a. Sci-kit learn
     b. Matplotlib
     c. Seaborn
     d. Pandas
     e. NumPy

- **Dataset used:**
  - Train.csv
  - Test.csv

- **Data Description:**
  - The given dataset contains various attributes, which on the basis of those attributes and their relationships, a model has been created.
  - The attributes are:
    - State
    - Account_length
    - Area_code
    - International_plan
    - Voice_mail_plan

        - Number_vmail_messages
        - Total_data_minutes
        - Total_eve_minutes
        - Total_eve_charge
        - Total_night_minutes
        - Total_night_calls
        - Total_night_charge
        - Total_intl_minutes
        - Total_intl_calls
        - Number_customer_service_calls
        - Churn
    - From the above-named attributes, we can say their description i.e., these attributes represent a service provider data set. [e.g., Airtel, Jio].
    - This represents a customer database and information related to their usage of the service like number of calls made in night, day, voicemail usage etc.,
    - Based on these data, a model must be created to predict the customer churn i.e., whether the customer wants to change the service/plans or not or in other words – 'The number of paying customers who fail to become repeat customers.

- **Approach:**
    - My approach for this project is on the basis of the following steps implemented to develop a model.
- **Steps Involved**
    - **STEP 1:** Importing the needful libraries
    - **STEP 2:** Uploading the 'train' dataset
    - **STEP 3:** Null-value check
    - **STEP 4:** EDA and Visualisation
        - Pie charts, bar plot
        - Plotting the state wise customers according to churn outcome.
    - **STEP 5:** Correlation matrix and heatmap
    - **STEP 6:** Feature Engineering
    - **STEP 7:** Developing a Model
        - Random-Forest Classifier
        - Accuracy detection using **F-Score**
    - **STEP 8:** Dimensionality Reduction
    - **STEP 9:** Uploading the 'test' dataset
    - **STEP 10:** Testing the 'test' dataset using the model developed
    - **STEP 11:** Conclusion

- **Explanation of the Steps Used:**

- **<u>STEP 1:</u> Importing the needful libraries**
- This is the first step of any ML problems.
- Importing the necessary libraries for imparting EDA, pre-processing, regression …. etc., to develop a Model.

Importing the Libraries

In [139]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import f1_score,confusion_matrix
from sklearn.metrics import accuracy_score
```

- **<u>STEP 2:</u> Uploading the 'train' dataset**
- Here, we upload the 'train' dataset that we are going to use for the further process i.e., using this dataset a model is created.

Uploading the train-dataset

In [3]:
```python
ds = pd.read_csv('C:/Users/rcavi/Desktop/major project/train.csv')
```

In [4]:
```python
ds
```

Out[4]:

l_eve_charge	total_night_minutes	total_night_calls	total_night_charge	total_intl_minutes	total_intl_calls	total_intl_charge	number_customer_service_calls	churn
16.62	254.4	103	11.45	13.7	3	3.70	1	no
10.30	162.6	104	7.32	12.2	5	3.29	0	no
5.26	196.9	89	8.86	6.6	7	1.78	2	no
12.61	186.9	121	8.41	10.1	3	2.73	3	no
29.62	212.6	118	9.57	7.5	7	2.03	3	no
...	...	...	...	...	...	...	...	...
20.72	213.7	79	9.62	10.3	6	2.78	0	no
11.15	186.2	89	8.38	11.5	6	3.11	3	no
16.41	129.1	104	5.81	6.9	7	1.86	1	no
18.96	297.5	116	13.39	9.9	5	2.67	2	no
22.70	154.8	100	6.97	9.3	16	2.51	0	no

- **<u>STEP 3:</u> Null-value check**
- This step is to be done before proceeding to pre-processing and EDA
- It is done to check for any null-values or empty cells in the given dataset which are not needed for processing.
- Their presence may lead to some hindrance while proceeding with visualisation.

Checking for null-values

In [8]:
```
#Confirm the number of missing values in each column.
ds.info()
```

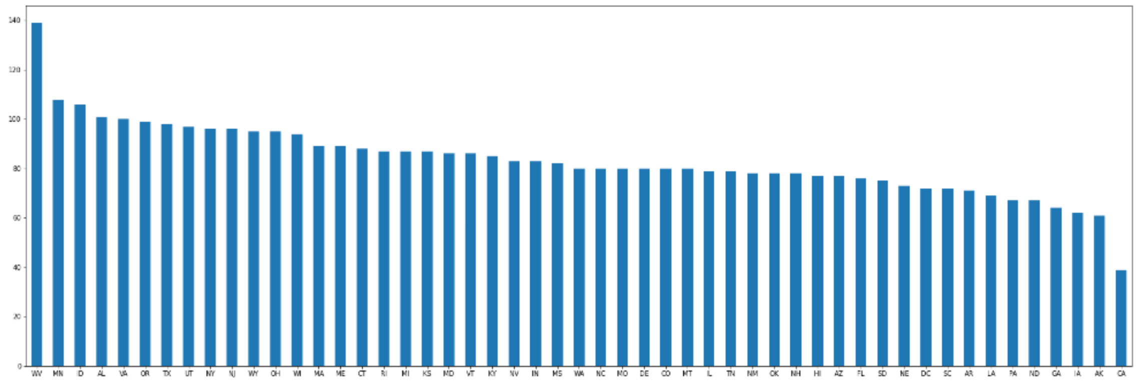
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4250 entries, 0 to 4249
Data columns (total 20 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   state                          4250 non-null   object 
 1   account_length                 4250 non-null   int64  
 2   area_code                      4250 non-null   object 
 3   international_plan             4250 non-null   object 
 4   voice_mail_plan                4250 non-null   object 
 5   number_vmail_messages          4250 non-null   int64  
 6   total_day_minutes              4250 non-null   float64
 7   total_day_calls                4250 non-null   int64  
 8   total_day_charge               4250 non-null   float64
 9   total_eve_minutes              4250 non-null   float64
 10  total_eve_calls                4250 non-null   int64  
 11  total_eve_charge               4250 non-null   float64
 12  total_night_minutes            4250 non-null   float64
 13  total_night_calls              4250 non-null   int64  
 14  total_night_charge             4250 non-null   float64
 15  total_intl_minutes             4250 non-null   float64
 16  total_intl_calls               4250 non-null   int64  
 17  total_intl_charge              4250 non-null   float64
 18  number_customer_service_calls  4250 non-null   int64  
 19  churn                          4250 non-null   object 
dtypes: float64(8), int64(7), object(5)
memory usage: 664.2+ KB
```

- **<u>STEP 4:</u> EDA and Visualisation**

- EDA is the process of analysing the given data for finding some useful relationships between the dataset, useful patterns, statistical representation, mean, median, finding any outliers etc.,

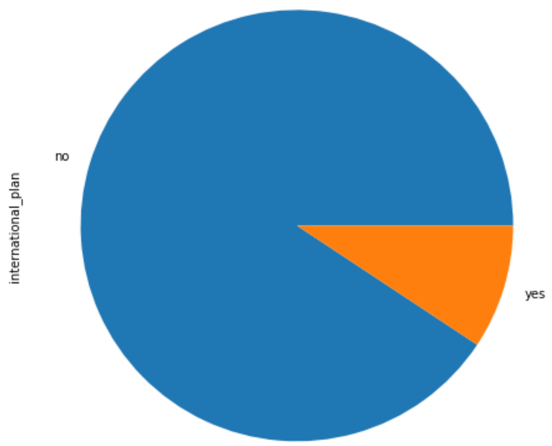
- In this step, we used Bar-plot representation using *'seaborn'* library for analysing the data and pie-chart for representing % of data to check its majority

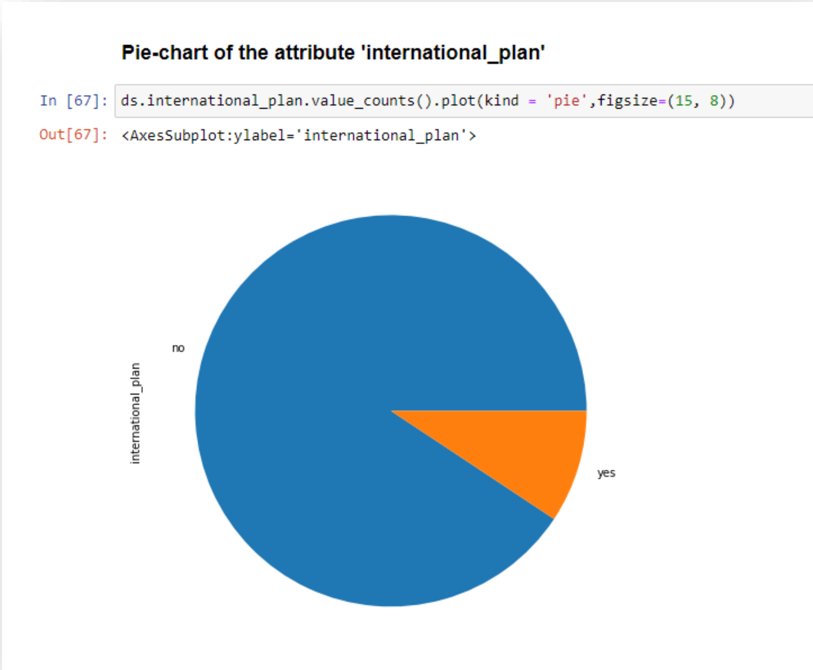
```
In [53]: ds.state.value_counts().plot(kind='bar',figsize=(30, 10), rot=0)
Out[53]: <AxesSubplot:>
```



Pie-chart of the attribute 'international_plan'

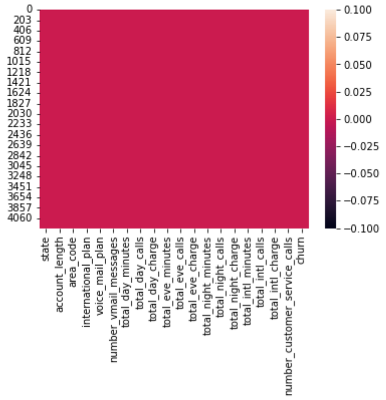
```
In [67]: ds.international_plan.value_counts().plot(kind = 'pie',figsize=(15, 8))
Out[67]: <AxesSubplot:ylabel='international_plan'>
```





- **<u>STEP 5:</u> Correlation matrix and heatmap**

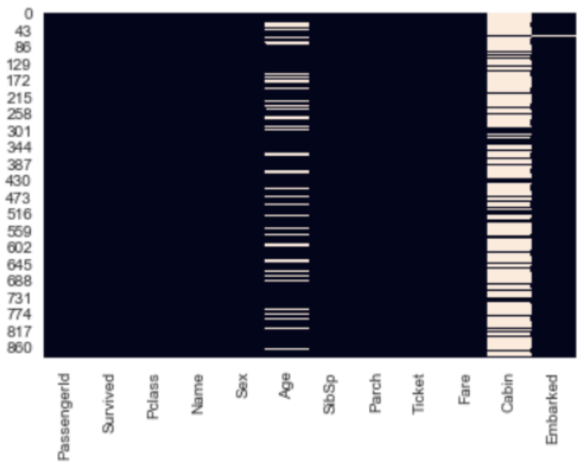
- Step 5 is a sub-part of step 4 - A <u>correlation matrix</u> is simply a table which displays the correlation coefficients for different variables.
- It is a powerful tool to summarize a large dataset and to identify and visualize patterns in the given data.
- A <u>heatmap</u> contains values representing various shades of the same colour for each value to be plotted. Usually, the darker shades of the chart represent higher values than the lighter shade.
- The observation of the visuals is mentioned in the below mentioned diagrams.
- The below 'red' heatmap shows that there are no missing values present in the dataset.
- Else, they will be missing part depicting missing values in the heat map.



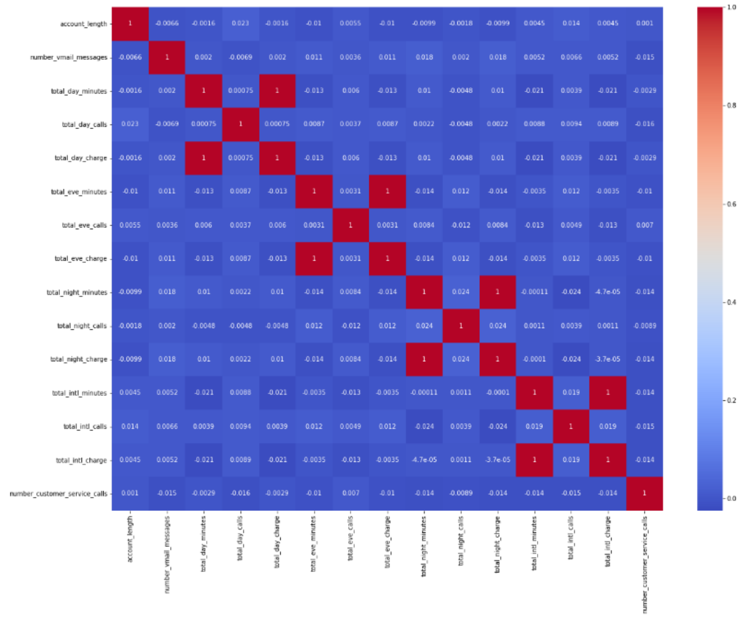
Observation : From the heat map , we can say there are no missing values in the 'train data set', if any present empty cells will be there.

Example of 'heat-map with missing values'

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x15cedc60470>
```



```
]: <AxesSubplot:>
```



plotting statewise customers

```
In [108]: fig, ax=plt.subplots(figsize=(20,5))
sns.countplot(data = ds, x='state', order=ds['state'].value_counts().index, palette='tab10', hue='churn')
plt.xticks(rotation=90)
plt.xlabel('State', fontsize=10, fontweight='bold')
plt.ylabel('Customers', fontsize=10, fontweight='bold')
plt.title('State wise Customers', fontsize=12, fontweight='bold')
plt.show()
```

- **<u>STEP 6:</u> Feature Engineering**
- <u>Feature engineering</u> is the pre-processing step of machine learning, which is used to transform raw data into features that can be used for creating a predictive model using Machine learning or statistical Modelling.
- It aims to improve the performance of models.
- Here, we have created columns for rate of calls, so call charge columns could be dropped as they are corelated.

```
ds1['day_rate'] = ds1['total_day_charge']/ds1['total_day_minutes']
ds1['eve_rate'] = ds1['total_eve_charge']/ds1['total_eve_minutes']
ds1['night_rate'] = ds1['total_night_charge']/ds1['total_night_minutes']
ds1['intl_rate'] = ds1['total_intl_charge']/ds1['total_intl_minutes']

In [125]: ds1.drop({'total_day_charge','total_eve_charge','total_night_charge','total_intl_charge'},axis=1,inplace= True)

In [126]: ds1.dropna(inplace=True)

In [127]: X0=ds1.drop('churn', axis=1)
          y= ds1.churn

In [128]: X0.columns

Out[128]: Index(['state', 'account_length', 'area_code', 'international_plan',
                 'voice_mail_plan', 'number_vmail_messages', 'total_day_minutes',
                 'total_day_calls', 'total_eve_minutes', 'total_eve_calls',
                 'total_night_minutes', 'total_night_calls', 'total_intl_minutes',
                 'total_intl_calls', 'number_customer_service_calls', 'day_rate',
                 'eve_rate', 'night_rate', 'intl_rate'],
                dtype='object')
```

```
In [129]: X0.drop({'voice_mail_plan'},axis=1,inplace= True)

In [130]: X0.isnull().sum().any()

Out[130]: False

In [131]: X0.isnull().sum().any()

Out[131]: False
```

- **<u>STEP 7:</u> Developing a Model and Accuracy using F-Score**

- This part of the step gives us the desired output.
- It is the most important of any ML/ Data science problems
- It is a process in which a machine learning (ML) algorithm is fed with sufficient training data to learn from.
- Here we used 'Random-Forest' classifier for developing a model

Developing a Model

In [137]:
```
X_train, X_valid, y_train, y_valid = train_test_split(X1, y, train_size=0.80,test_size=0.20,random_state=1)
```

In [138]:
```
print(X_train.isnull().sum().any())
print(X_valid.isnull().sum().any())
print(y_train.isnull().sum().any())
print(y_valid.isnull().sum().any())
```

```
False
False
False
False
```

In [140]:
```
X_train.head(3)
```

Out[140]:

	account_length	number_vmail_messages	total_day_minutes	total_day_calls	total_eve_minutes	total_eve_calls	total_night_minutes	total_night_calls	total
1170	36	0	202.4	115	230.7	115	202.0	127	
2689	41	0	223.8	67	244.8	74	223.8	156	
2578	95	0	190.2	119	157.1	70	181.5	120	

3 rows × 68 columns

- **Random Forest**
- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
- Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

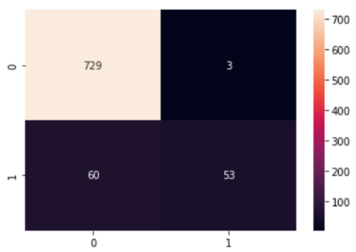
Random_forest classifier

In [143]:
```
from sklearn.ensemble import RandomForestClassifier
clf_rf = RandomForestClassifier(n_estimators=20 ,random_state=43)
clr_rf = clf_rf.fit(X_train,y_train)

ac = accuracy_score(y_valid,clf_rf.predict(X_valid))
print('Accuracy is: ',int(ac*100)," %")
cm = confusion_matrix(y_valid,clf_rf.predict(X_valid))
sns.heatmap(cm,annot=True,fmt="d")
```

```
Accuracy is:  92  %
```

Out[143]: <AxesSubplot:>



Accuracy of the model is 92%

F-Score

In [104]:
```
fscore=f1_score(y_valid,clf_rf.predict(X_valid),average='micro')
fscore=fscore*100
print(f"Accuracy of the model using F-Score is {round(fscore,2)}")

#'micro':
#Calculate metrics globally by counting the total true positives, false negatives and false positives.
```

```
Accuracy of the model using F-Score is 92.54
```

Accuracy of the model is 92%

- **<u>STEP 8:</u> Dimensionality Reduction**
- Dimensionality reduction simply refers to the process of reducing the number of attributes in a dataset while keeping as much of the variation in the original dataset as possible
- This process can be classified into two ways:
    - Feature Selection ($\chi 2-$ ***test***)
    - Feature Extraction
- **Advantages:**
    - A lower number of dimensions in data means less training time and less computational resources and increases the overall performance of machine learning algorithms
    - Dimensionality reduction is extremely useful for data visualization
- Here, we have used $\chi 2-$ ***test*** for dimensionality reduction.
- A $\chi 2-$ ***test*** is basically a data analysis on the basis of observations of a random set of variables. Usually, it is a comparison of two statistical data sets.

In [142]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
# find best scored 5 features
select_feature = SelectKBest(chi2, k=5).fit(X_valid, y_valid)
a = select_feature.scores_
b = X_train.columns
df = pd.DataFrame(list(zip(b, a)),
               columns =['Column', 'Score'])
df.dtypes
```

Out[142]:
```
Column     object
Score     float64
dtype: object
```

In [144]:
```
df['Score'] = df['Score'].replace(np.nan, 0)
df['Score'] = df['Score'].astype(int)
df.sort_values(by='Score',ascending=False)
```

Out[144]:

	Column	Score
2	total_day_minutes	398
1	number_vmail_messages	313
4	total_eve_minutes	78
67	international_plan_yes	60
10	number_customer_service_calls	52
...	...	...
22	state_DE	0
21	state_DC	0
19	state_CO	0
47	state_NV	0
34	state_MD	0

68 rows × 2 columns

- **<u>STEP 9:</u> Uploading the 'test' dataset**

  &

  <u>STEP 10:</u> Testing the 'test' dataset using the model developed

- Now, we have to test the model we developed using the 'test' dataset
- For that, we'll make predictions for the first few rows of the test data to see how the predict function works.

Uploading the Test dataset

In [148]:
```
ts=pd.read_csv('C:/Users/rcavi/Desktop/major project/test.csv')
```

In [149]:
```
ts.head()
```

Out[149]:

	id	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge
0	1	KS	128	area_code_415	no	yes	25	265.1	110	45.07
1	2	AL	118	area_code_510	yes	no	0	223.4	98	37.98
2	3	IA	62	area_code_415	no	no	0	120.7	70	20.52
3	4	VT	93	area_code_510	no	no	0	190.7	114	32.42
4	5	NE	174	area_code_415	no	no	0	124.3	76	21.13

In [150]:
```
ts1=ts.copy()
```

In [151]:
```
ts1['day_rate'] = ts1['total_day_charge']/ts1['total_day_minutes']
ts1['eve_rate'] = ts1['total_eve_charge']/ts1['total_eve_minutes']
ts1['night_rate'] = ts1['total_night_charge']/ts1['total_night_minutes']
ts1['intl_rate'] = ts1['total_intl_charge']/ts1['total_intl_minutes']
```

In [152]:
```
ts1.drop({'id','total_day_charge','total_eve_charge','total_night_charge','total_intl_charge','voice_mail_plan'},axis=1,inplace=
ts2= pd.get_dummies(ts1, drop_first = True)
```

In [70]:
```
submission = pd.DataFrame({
        "id": ts["id"],
        "churn": results
    })

submission.to_csv('churn_output.csv', index=False)
```

In [76]:
```
churn_output.shape
```

Out[76]: (750, 2)

- **STEP 11: Conclusion**
- The used **Random Forest Classifier** for our model gives the accuracy of **92 %** when detected using **F-Score** which is a great result for a ML model.
- So, after pre-processing and EDA techniques, the model developed using **Random Forest** is apt for the given Boston House Prediction
- After testing the model using the 'test.csv' dataset, the **'churn_output.csv'** is the final output of whether the customer churn is positive or negative.
- Sample – **'churn_output.csv'**

	A	B	C	D	E
1	id	churn			
2	1	no			
3	2	no			
4	3	no			
5	4	no			
6	5	no			
7	6	no			
8	7	no			
9	8	no			
10	9	no			
11	10	yes			
12	11	yes			
13	12	no			
14	13	no			
15	14	no			
16	15	no			
17	16	no			
18	17	no			
19	18	no			
20	19	no			
21	20	no			
22	21	no			

- \