الجامعة السعودية الالكترونية
Saudi Electronic University
2011 1432
College of Computing and Informatics

2020/2021 First Semester

Course Code	DS560
Course Name	advanced data mining
CRN	24539
Assignment type	Project
Module	All
Assignment Points	10

Student ID	G200002614	G200007615
Student Name	Enad S. Alotaibi	Abdulaziz M. Alqumayzi

Introduction

In this we have data about calls of different states with area code like total minutes call, total charger and people are calling in day or night. This competition is about predicting whether a customer will change telecommunications provider, something known as "churning". The dataset contains 333 samples. Each sample contains 19 features and 1 Boolean variable "churn" which indicates the class of the sample.

Import All Necessary Libraries

```
# Basic Libraries
import pandas as pd
import numpy as np

#Visualziation
from matplotlib import pyplot as plt
from matplotlib.gridspec import GridSpec
import seaborn as sns

# machine learning
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split

#classification models
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

#ensemble classification models
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier

# ignore warnings
import warnings
warnings.filterwarnings("ignore")
```

Import the Dataset

```
# import dataset and store in variable and check the top 5 rows of the dataset
df = pd.read_csv("telecom_churn.csv")
df.head()
```

	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	Cu
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	

College of Computing and Informatics

Check the shape of the Dataset

```
#checking of size (rows and columns) of the dataframe
print(f"There are {df.shape[0]} rows and {df.shape[1]} columns in the dataframe ")
```

```
There are 3333 rows and 20 columns in the dataframe
```

Information about the Dataset

```
# check all information of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   3333 non-null   object
 1   Account length          3333 non-null   int64
 2   Area code               3333 non-null   int64
 3   International plan      3333 non-null   object
 4   Voice mail plan         3333 non-null   object
 5   Number vmail messages   3333 non-null   int64
 6   Total day minutes       3333 non-null   float64
 7   Total day calls         3333 non-null   int64
 8   Total day charge        3333 non-null   float64
 9   Total eve minutes       3333 non-null   float64
 10  Total eve calls         3333 non-null   int64
 11  Total eve charge        3333 non-null   float64
 12  Total night minutes     3333 non-null   float64
 13  Total night calls       3333 non-null   int64
 14  Total night charge      3333 non-null   float64
 15  Total intl minutes      3333 non-null   float64
 16  Total intl calls        3333 non-null   int64
 17  Total intl charge       3333 non-null   float64
 18  Customer service calls  3333 non-null   int64
 19  Churn                   3333 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB
```

Dataset Summary (Numerical Variables)

```
# check all numerical values summary of the dataset
df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
Account length	3333.0	101.064806	39.822106	1.00	74.00	101.00	127.00	243.00
Area code	3333.0	437.182418	42.371290	408.00	408.00	415.00	510.00	510.00
Number vmail messages	3333.0	8.099010	13.688365	0.00	0.00	0.00	20.00	51.00
Total day minutes	3333.0	179.775098	54.467389	0.00	143.70	179.40	216.40	350.80
Total day calls	3333.0	100.435644	20.069084	0.00	87.00	101.00	114.00	165.00
Total day charge	3333.0	30.562307	9.259435	0.00	24.43	30.50	36.79	59.64
Total eve minutes	3333.0	200.980348	50.713844	0.00	166.60	201.40	235.30	363.70
Total eve calls	3333.0	100.114311	19.922625	0.00	87.00	100.00	114.00	170.00
Total eve charge	3333.0	17.083540	4.310668	0.00	14.16	17.12	20.00	30.91
Total night minutes	3333.0	200.872037	50.573847	23.20	167.00	201.20	235.30	395.00
Total night calls	3333.0	100.107711	19.568609	33.00	87.00	100.00	113.00	175.00
Total night charge	3333.0	9.039325	2.275873	1.04	7.52	9.05	10.59	17.77
Total intl minutes	3333.0	10.237294	2.791840	0.00	8.50	10.30	12.10	20.00
Total intl calls	3333.0	4.479448	2.461214	0.00	3.00	4.00	6.00	20.00
Total intl charge	3333.0	2.764581	0.753773	0.00	2.30	2.78	3.27	5.40
Customer service calls	3333.0	1.562856	1.315491	0.00	1.00	1.00	2.00	9.00

Convert Churn Column Boolean values into string

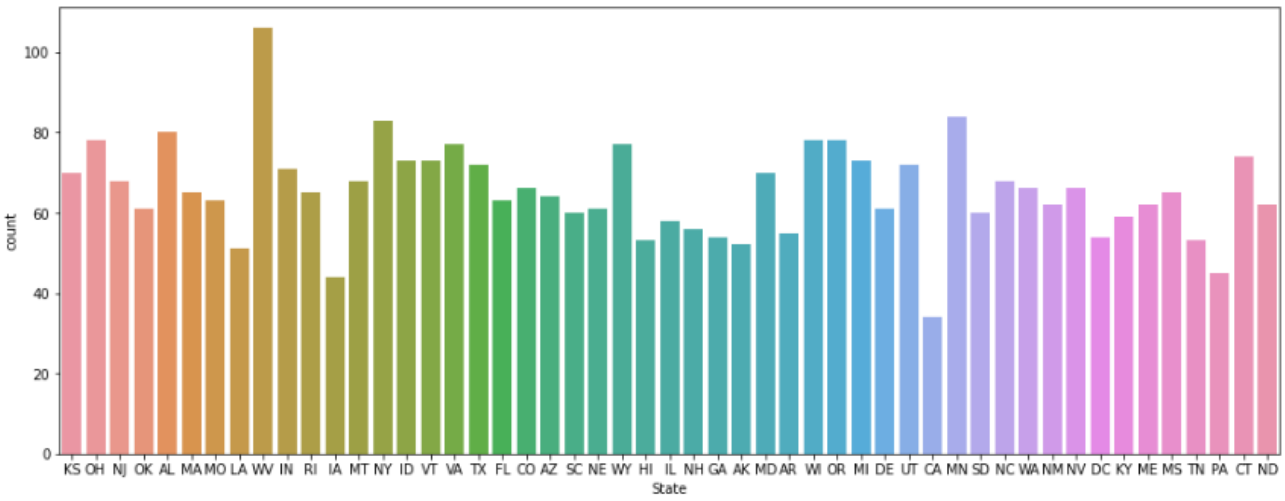
```
# churn is boolean so first we will convert into string
df = df.replace({"Churn":{False :"No",True:"Yes"}})
```

Dataset Summary (Categorical Variables)

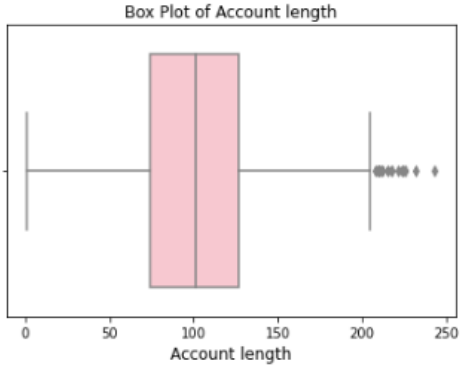
```
# check all object values summary of the dataset
df.describe(include = object).transpose()
```

	count	unique	top	freq
State	3333	51	WV	106
International plan	3333	2	No	3010
Voice mail plan	3333	2	No	2411
Churn	3333	2	No	2850

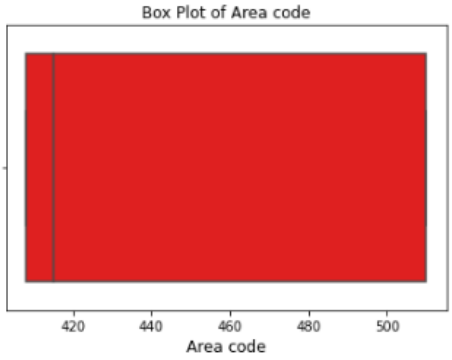
Handle the Outliers of All Variables



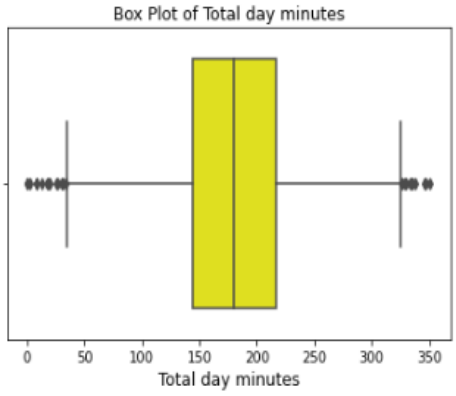
As we can see **State** column has no Outlier.



As you can see in column **Account length** there are outliers, to handle this we will replace all values which is greater than 200 by null values so later we can fill by our own technique.

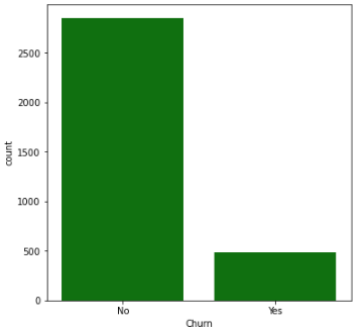


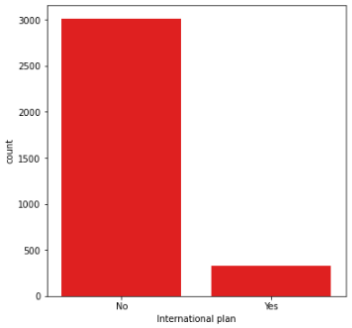
As you can see in column, **Area Code** there is no outliers.

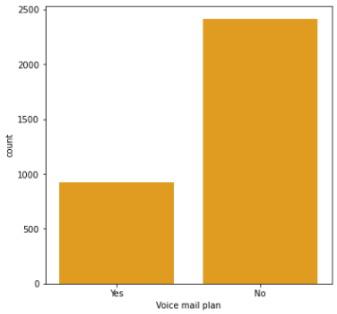


As you can see in column **Total day minutes** there are outliers, to handle this we will replace all values which is greater than 325 and less than 40 by null values so later we can fill by our own technique.

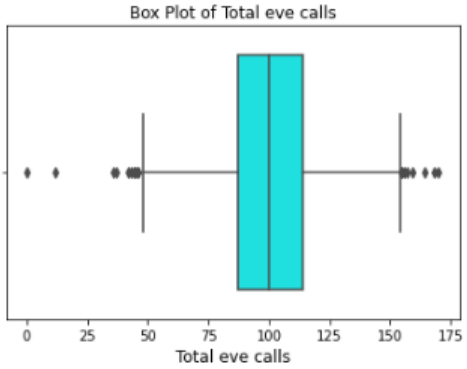
As you can see in column **Number vmail messages** there are outliers, to handle this we will replace all values which is greater than 50 by null values so later we can fill by our own technique.



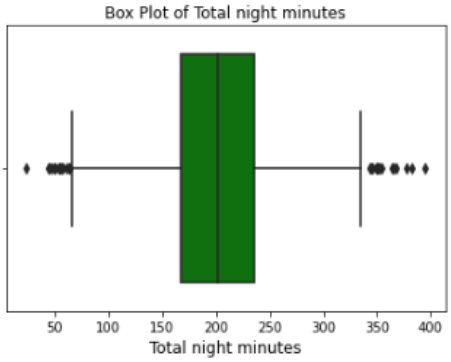




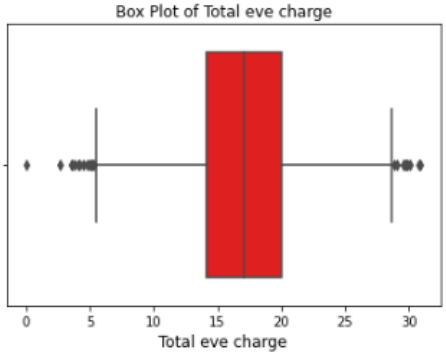
As you can see in column **International Plans , churn and Voice mail plan** there is outlier



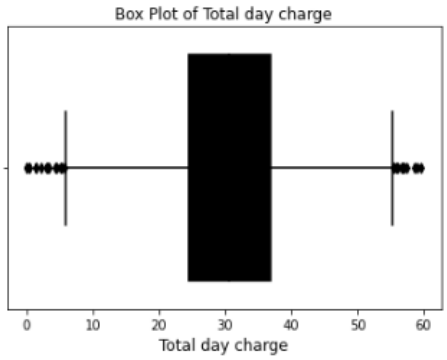
As you can see in column **Total eve Calls** there are outliers, to handle this we will replace all values which is greater than 160 and less than 50 by null values so later we can fill by our own technique



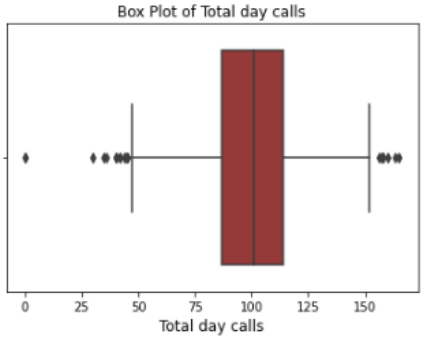
As you can see in column **Total night minutes** there are outliers, to handle this we will replace all values which is greater than 340 and less than 70 by null values so later we can fill by our own technique



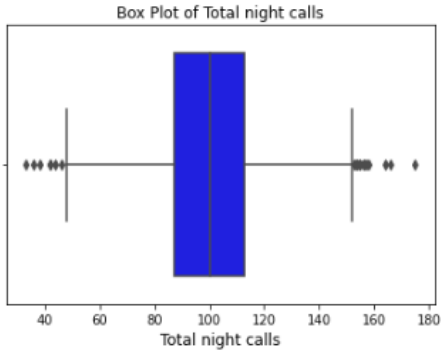
As you can see in column **Total eve charges** there are outliers, to handle this we will replace all values which is greater than 28 and less than 6 by null values so later we can fill by our own technique



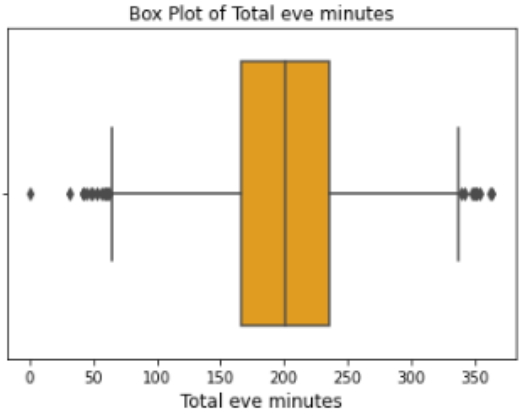
As you can see in column **Total Day Charges** there are outliers, to handle this we will replace all values which is greater than 55 and less than 8 by null values so later we can fill by our own technique



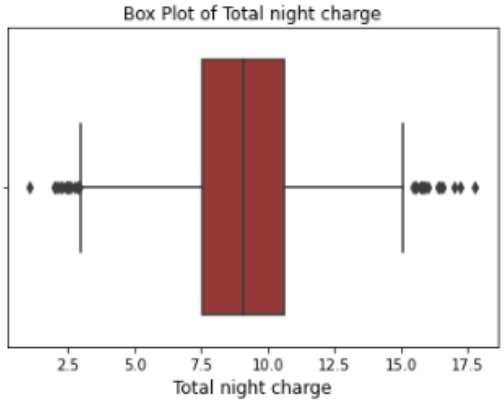
As you can see in column **Total day calls** there are outliers, to handle this we will replace all values which is greater than 150 and less than 50 by null values so later we can fill by our own technique



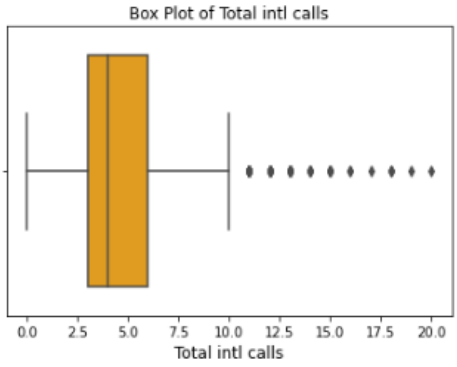
As you can see in column **Total night calls** there are outliers, to handle this we will replace all values which is greater than 155 and less than 50 by null values so later we can fill by our own technique



As you can see in column **Total eve minutes** there are outliers, to handle this we will replace all values which is greater than 340 and less than 60 by null values so later we can fill by our own technique

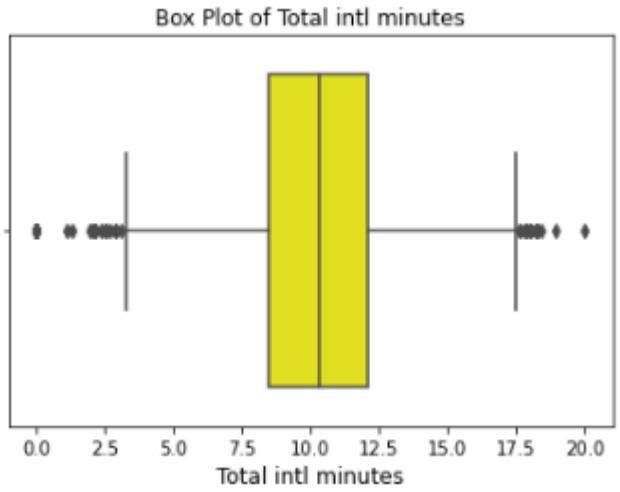


As you can see in column **Total Night Charge** there are outliers, to handle this we will replace all values which is greater than 15 and less than 2.7 by null values so later we can fill by our own technique

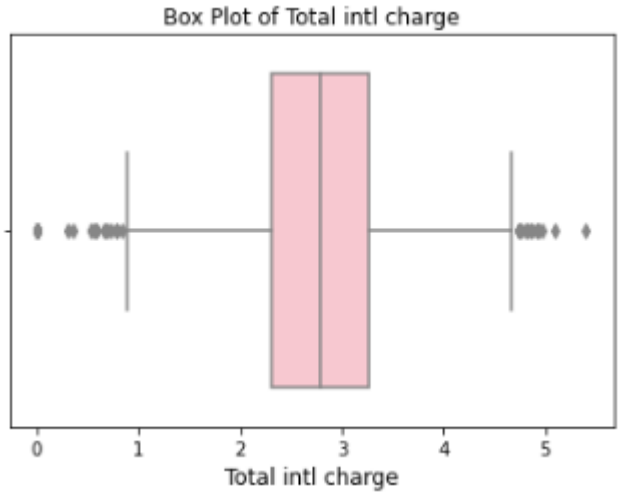


As you can see in column **Total init calls** there are outliers, to handle this we will replace all values which is greater than 10 by null values so later we can fill by our own technique

As you can see in column **Customer service calls** there are outliers, to handle this we will replace all values which is greater than 2.5 by null values so later we can fill by our own technique

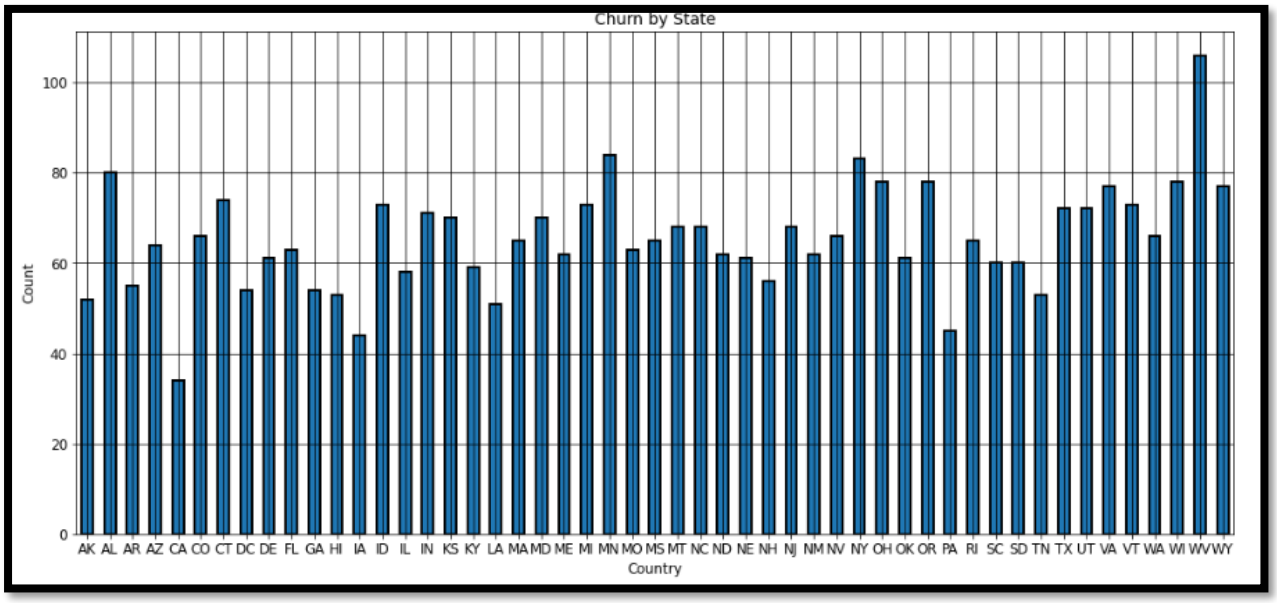


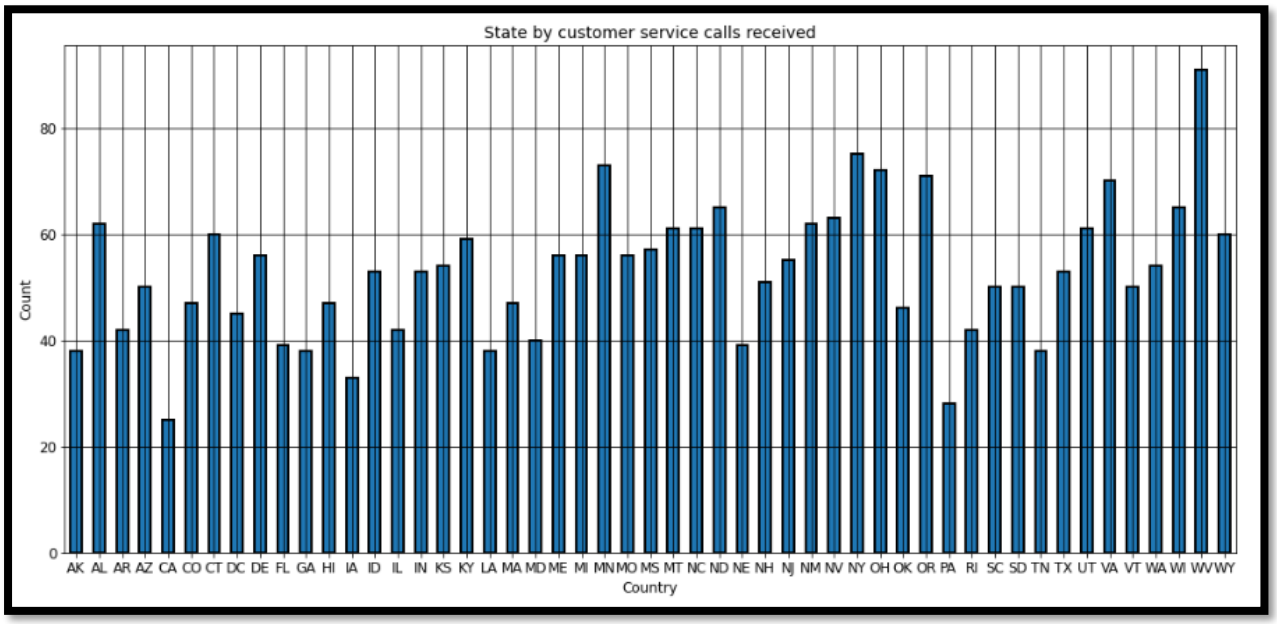
As you can see in column **Total init minutes** there are outliers, to handle this we will replace all values which is greater than 17.5 and less than 3 by null values so later we can fill by our own technique



As you can see in column **Total init charge** there are outliers, to handle this we will replace all values which is greater than 4.7 and less than 1 by null values so later we can fill by our own technique

What is the state where most churns occur? Is there statistical difference between the numbers of customer service calls received in this state compared to the remaining states? What do you observe?





The WV State has most churns and Yes We can easily see from above graph that curn has direct relationship with Customer service calls. Because we have more churn in WV and more calls in WV.

Data Cleaning:

As you see, we have null values in these columns

```
1 df.isnull().sum()
```

```
State                     0
Account length           27
Area code                 0
International plan         0
Voice mail plan            0
Number vmail messages      3
Total day minutes         29
Total day calls           42
Total day charge          36
Total eve minutes         21
Total eve calls           25
Total eve charge          39
Total night minutes       33
Total night calls         22
Total night charge        34
Total intl minutes         0
Total intl calls         128
Total intl charge         62
Customer service calls   696
Churn                     0
dtype: int64
```

Handling the Missing values:

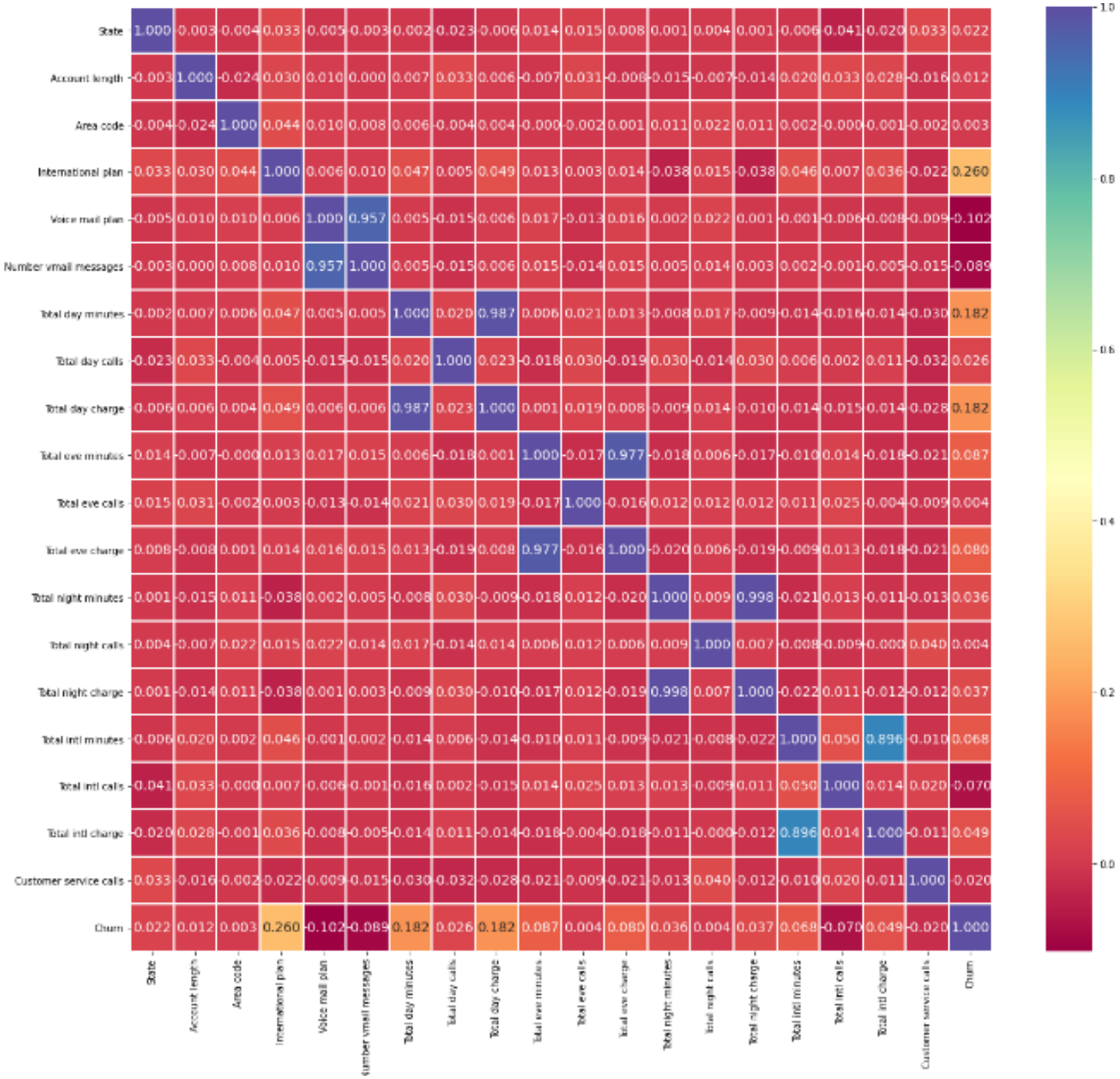
Remove the missing values by forward fill and fillna by mode in Account length column

```
df["Account length"] = df["Account length"].fillna(df['Account length'].mode()[0])
df["Number vmail messages"] = df["Number vmail messages"].ffill()
df["Total day minutes"] = df["Total day minutes"].ffill()
df["Total day calls"] = df["Total day calls"].ffill()
df["Total day charge"] = df["Total day charge"].ffill()
df["Total eve minutes"] = df["Total eve minutes"].ffill()
df["Total eve calls"] = df["Total eve calls"].ffill()
df["Total eve charge"] = df["Total eve charge"].ffill()
df["Total night minutes"] = df["Total night minutes"].ffill()
df["Total night calls"] = df["Total night calls"].ffill()
df["Total night charge"] = df["Total night charge"].ffill()
df["Total intl calls"] = df["Total intl calls"].ffill()
df["Total intl charge"] = df["Total intl charge"].ffill()
df["Customer service calls"] = df["Customer service calls"].ffill()
```

What external data sources would be useful to enrich this dataset and why?

If we know that, the person who is calling is boy or girl so we can analyze more and the proper time so we can easily see which time most people call.

Heat Map



From Heat Map we can see that these three columns "International plan", "Total day charge", "Total day minutes" is correlating more with our target variable so these will become our feature variable for classification model.

Classification model development

So first, we will split our data into features and labels. Our label is churn column which is stored In y and our features which we selected by seeing out heatmap which is correlating more with our target variables ("International plan", "Total day charge", "Total day minutes") store in X.

```
y = df['Churn']
X = df[["International plan", "Total day charge", "Total day minutes"]].values
```

Develop and compare at least three classification models that predict customer churn. You are expected to perform hyperparameter tuning and choose the best combination.

Now we will use these classifiers:

1. **Gaussian Naive Bayes**
2. **Decision Tree Classifier**
3. **SVC Classifier**

```
# hyperparameters
param_gb = {'var_smoothing': [0.00000001, 0.000000001, 0.00000001]}

# Gaussian classifier
gb = GaussianNB()

# GridsearchCV with crossvalidation = 5
gb_cv = GridSearchCV(gb, param_gb, cv = 5)

gb_cv.fit(X, y)

# Print the tuned parameters and score
print("Tuned GaussianNB  Best Parameters: {}".format(gb_cv.best_params_))
print("Best score is {}".format(gb_cv.best_score_))
```

```
Tuned GaussianNB  Best Parameters: {'var_smoothing': 1e-08}
Best score is 0.8457888173030603
```

Above we use **Gaussian Naive Bayes** Classifier and use different hyperparameters like var_smoothing with different with 5 times cross validation so we can see the best accuracy we can achieve from this classifier is 84.57%. by var_smoothing 1 exp -8.

```
# hyperparameters
param_dist = {"max_depth": [3,5],
              "criterion": ["gini", "entropy"]}

# Decision Tree classifier
tree = DecisionTreeClassifier()

# GridsearchCV with crossvalidation = 5
tree_cv = GridSearchCV(tree, param_dist, cv = 5)

tree_cv.fit(X, y)

# Print the tuned parameters and score
print("Tuned Decision Tree Best Parameters: {}".format(tree_cv.best_params_))
print("Best score is {}".format(tree_cv.best_score_))
```

```
Tuned Decision Tree Best Parameters: {'criterion': 'entropy', 'max_depth': 3}
Best score is 0.8652903278090684
```

Above we use Decision tree classifier and use different hyperparameters like max_depth with value 3 and 5 and criterion with value gini and entropy with 5 times cross validation so we can see the best accuracy we can achieve from this classifier is 86.52% by entropy criterion with max_depth 3.

```
# defining parameter range
param_grid = {'C': [0.1, 1, 100],
              'gamma': [1, 0.1, 0.01]}

# SVC Classifier
svc = SVC()

# GridsearchCV with crossvalidation = 5
grid_svc = GridSearchCV(svc , param_grid, cv = 5)

# fitting the model for grid search
grid_svc.fit(X, y)

# Print the tuned parameters and score
print("Tuned SVC Best Parameters: {}".format(grid_svc.best_params_))
print("Best score is {}".format(grid_svc.best_score_))
```

```
Tuned SVC Best Parameters: {'C': 1, 'gamma': 0.1}
Best score is 0.8625867246556901
```

Above we use SVC classifier and use different hyperparameters like gamma with values 1,0.1, 0.01 and C with 0.1, 1, 100 with 5 times cross validation so we can see the best accuracy we can achieve from this classifier is 86.25% by 0.01 gamma and 1 with C.

As you can see we use Gaussian , Decision tree and SVC Classifier, and we achieve best accuracy from Decision Tree Classifier which is 86.52%

Ensemble Classification Models

```
# Create the parameter grid
param_rfc = {'min_samples_leaf': [4, 5],
    'min_samples_split': [10, 12],
    'n_estimators': [300, 1000]}


# creating a rfc classifier
rfc = RandomForestClassifier()

# GridsearchCV with crossvalidation = 5
rfc_cv = GridSearchCV(rfc, param_rfc, cv = 5)

# fitting the model for grid search
rfc_cv.fit(X, y)

# Print the tuned parameters and score
print("Hyperparameter tunning best Parameters: {}".format(rfc_cv.best_params_))
print("Best score is {}".format(rfc_cv.best_score_))
```

```
Hyperparameter tunning best Parameters: {'min_samples_leaf': 5, 'min_samples_split': 12, 'n_estimators': 1000}
Best score is 0.8508898703801252
```

Above we use Random Forest classifier and use different hyperparameters like min_sample leaf with value 4 and 5 and min samples split with value 10, 12 and n_estimator with values 300 and 1000 with 5 times cross validation so we can see the best accuracy we can achieve from this classifier is 85.08% by min_samples_leas with 5 and n_estimator with 1000 and min_samples_split with 12.

```
# Create the parameter grid
param_abc = {'learning_rate': [0.001, 0.00001, 0.000001],
    'n_estimators': [300, 500]}


# creating a rfc classifier
ad  = AdaBoostClassifier()

# GridsearchCV with crossvalidation = 5
ad_cv = GridSearchCV(ad, param_abc, cv = 5)


# fitting the model for grid search
ad_cv.fit(X, y)

# Print the tuned parameters and score
print("Hyperparameter tunning best Parameters: {}".format(ad_cv.best_params_))
print("Best score is {}".format(ad_cv.best_score_))
```

```
Hyperparameter tunning best Parameters: {'learning_rate': 1e-05, 'n_estimators': 300}
Best score is 0.8631904768336552
```

Above we use Ada Boost classifier and use different hyperparameters like learning rate with values 0.001, 0.00001 and 0.000001 and n_estimator with values 300 and 500 with 5 times

cross validation so we can see the best accuracy we can achieve from this classifier is 86.31% with learning rate 1 exp -5 and n estimator with 300.

```
# Create the parameter grid
param_gbc = {'learning_rate': [0.001, 0.00001, 0.000001],
    'n_estimators': [300, 500]}


# creating a rfc classifier
gbc  = GradientBoostingClassifier()

# GridsearchCV with crossvalidation = 5
gbc_cv = GridSearchCV(gbc, param_gbc, cv = 5)


# fitting the model for grid search
gbc_cv.fit(X, y)

# Print the tuned parameters and score
print("Hyperparameter tunning best Parameters: {}".format(gbc_cv.best_params_))
print("Best score is {}".format(gbc_cv.best_score_))
```

```
Hyperparameter tunning best Parameters: {'learning_rate': 0.001, 'n_estimators': 300}
Best score is 0.8550859705282494
```

Above we use Gradient Boosting classifier and use different hyperparameters like learning rate with values 0.001, 0.00001 and 0.000001 and n_estimator with values 300 and 500 with 5 times cross validation so we can see the best accuracy we can achieve from this classifier is 85.50% with learning rate 0.001 and n estimator with 300.

As you can see we use Random Forest Classifier,AdaBoost Classifier and GradientBoosting Classifier, and we achieve best accuracy from AdaBoost Classifier which is 86.31%

Analyze the results of your best performing classifier. Is it statistically different than the least performing classifier?

As we can see the accuracy of all the classifier, best performing classifier is Decision Tree classifier which accuracy is 86.52% if we see other classifier also there is no such big difference we see. so can say that all classifier are performing good in this dataset.

Novelty and Innovation

- If we know which state are making too much call then we can reduce the price of that state.
- If we know which time people are, most calls then we can reduce the price of that time.
- If we know if most of the user is using international calls then we can give special package to them.
- If company will give special package to the people whose account, length is larger so they will not churn.
- If we know if most of the user is using voice mail plan then we can give special package to them.

References:

- Kane, F. (2017). Hands-On Data Science and Python Machine Learning. Birmingham, UK : Packt Publishing Ltd.
- ingh, A. (2018). A comprehensive guide to ensemble learning (with Python codes). Retrieved from: https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/
- Fatima, A., Nazir, N., & Khan, M. G. (2017). Data cleaning in a data warehouse: A survey of data pre-processing techniques and tools. IJ Information Technology and Computer Science, 3, 50-61.
- Kwak, S. K., & Kim, J. H. (2017). Statistical data preparation: management of missing values and outliers. Korean Journal of Anesthesiology, 70(4), 407.