Churn Prediction of Telco Provider

Cindy Felicia Turnip - Data Science Showcase

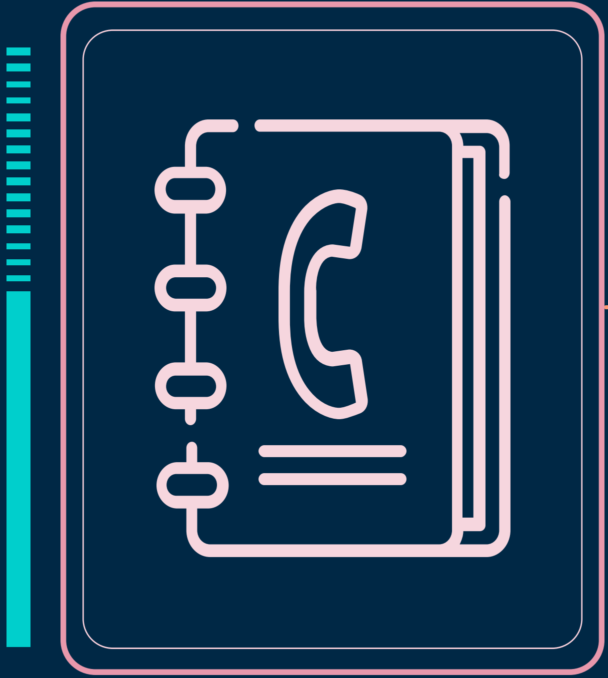
TABLE OF CONTENTS

Exploratory Data Analysis (EDA)

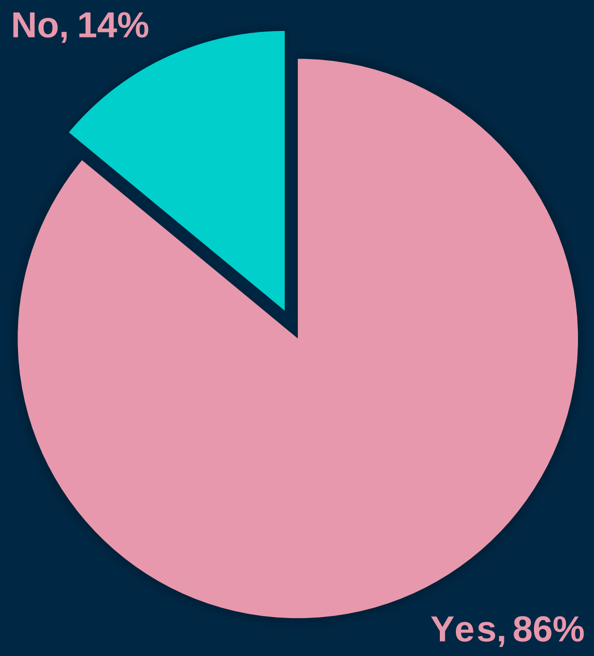
01

What is Churn?

The term “Churn” means leaving the company. In this case churn in telecommunication is the customer who has the right to decide to stay or leave the provider.



Customer Research



Yes

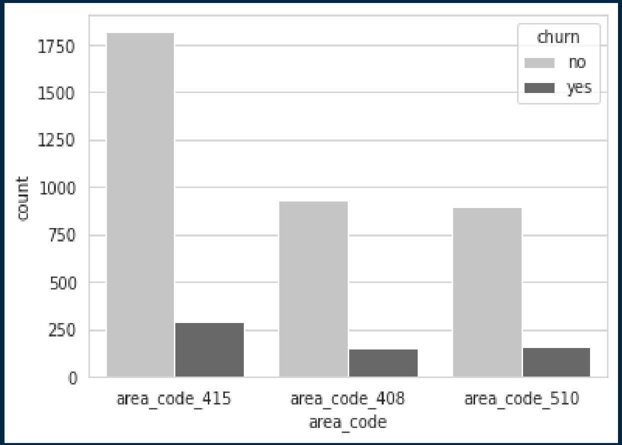
The customer who decided to leave about 86%

No

The customer who decided to stay about 14%

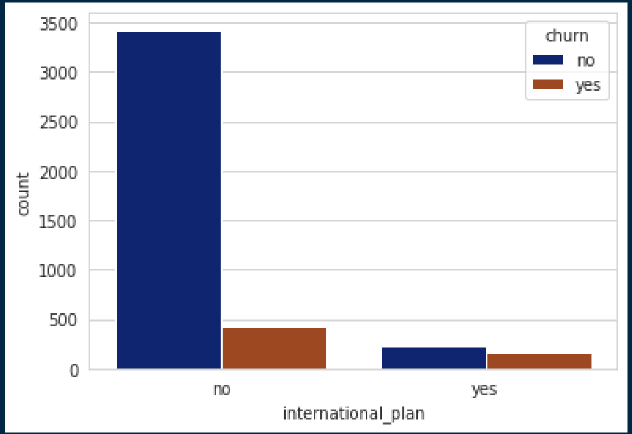
Customer Area Distribution

- Most of customer who decided to leave are in area code 415
- Most of customer who decided to stay are also in area code 415



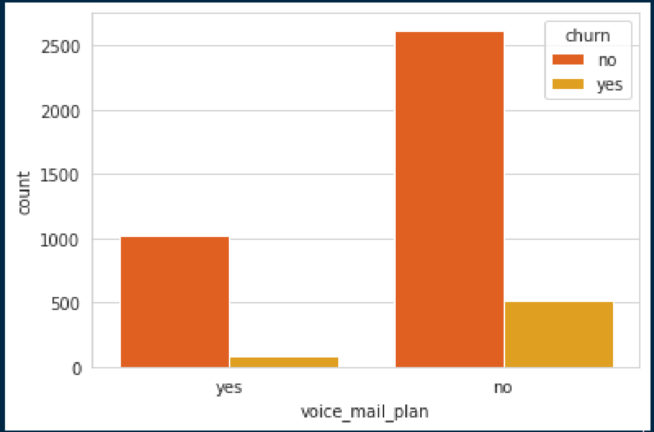
Customer International Plan

- More customers who continuing theirs service has no plan for theirs international calls.
- Maybe the provider can re-evaluate theirs international plan to make their customers more interested in.

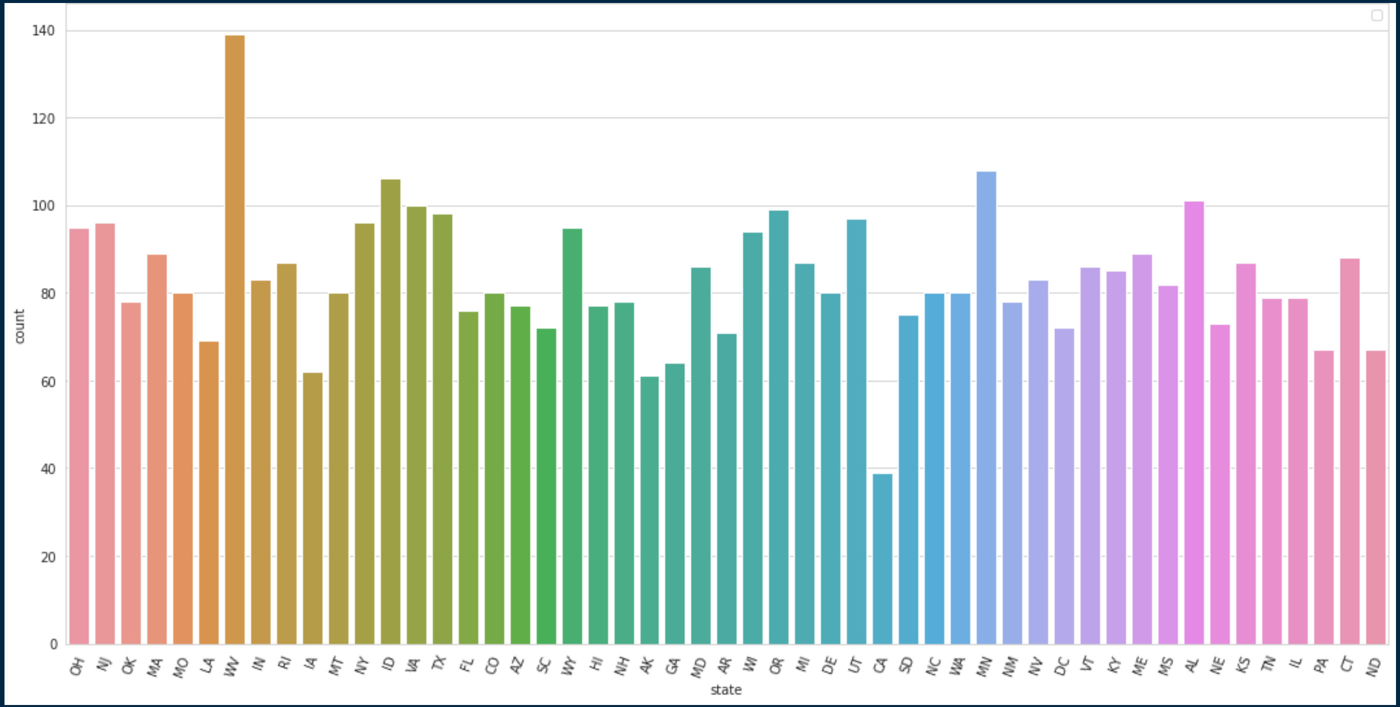


Customer Voice Mail Plan

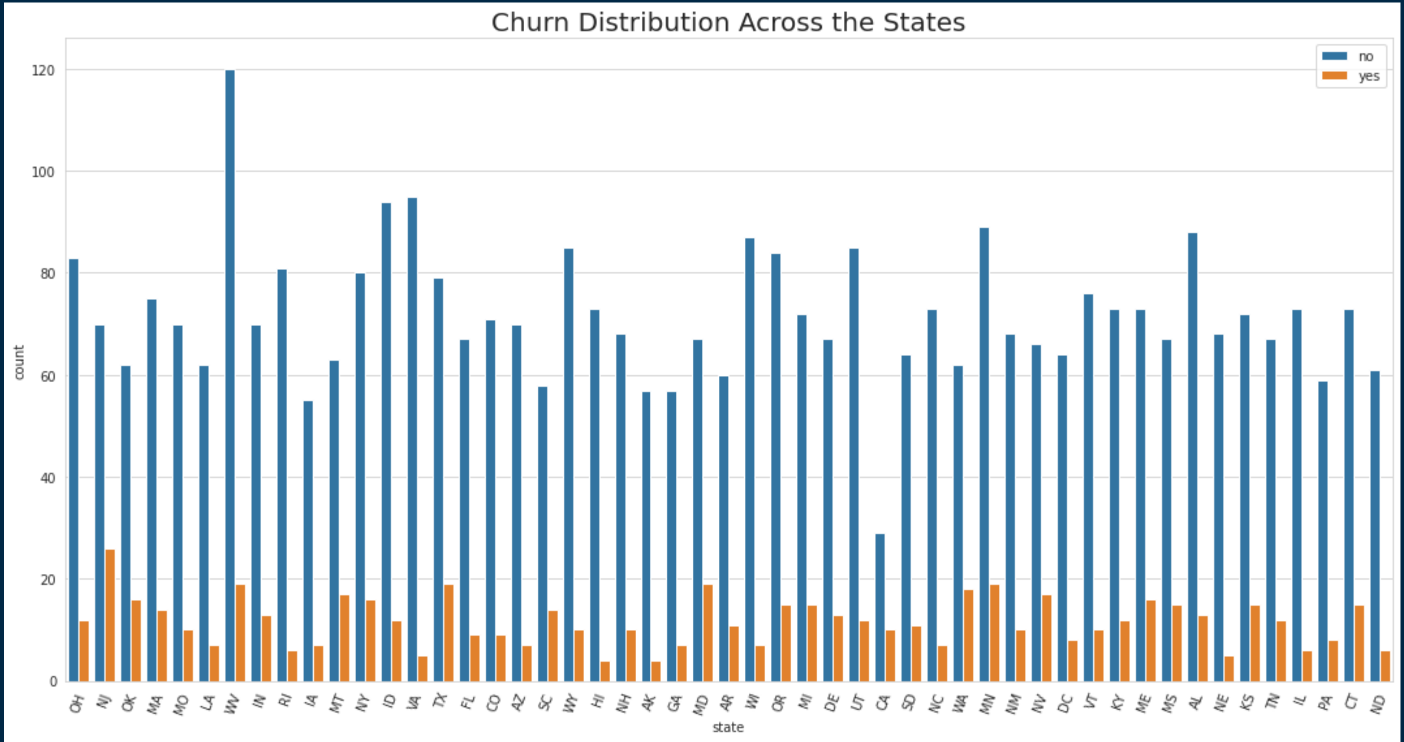
- More customers who continuing theirs service also has no plan for theirs voice mail.
- Maybe the provider can re-evaluate theirs voice mail plan to make their customers more interested in



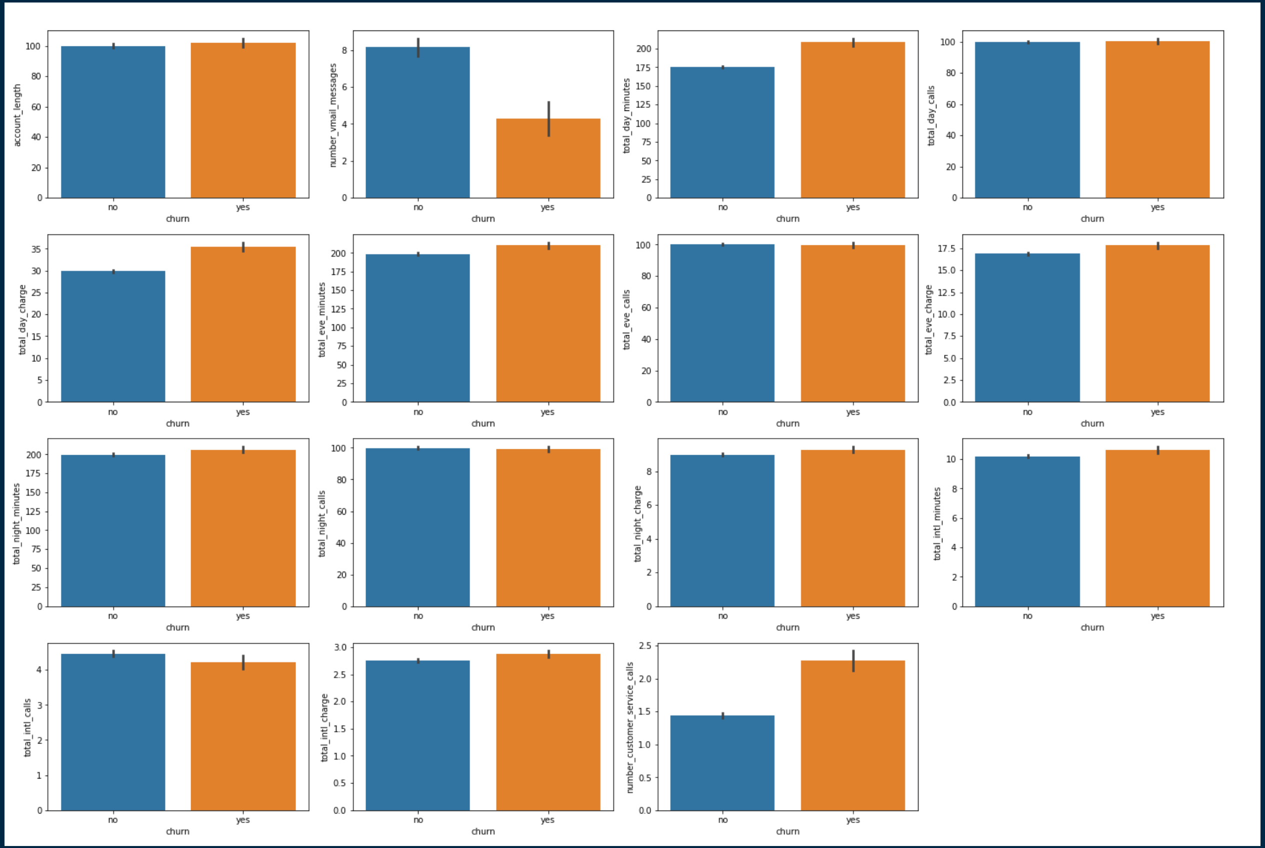
Customer Distribution Across the State



Churn Distribution Across the State



Bar Chart Vizualitation

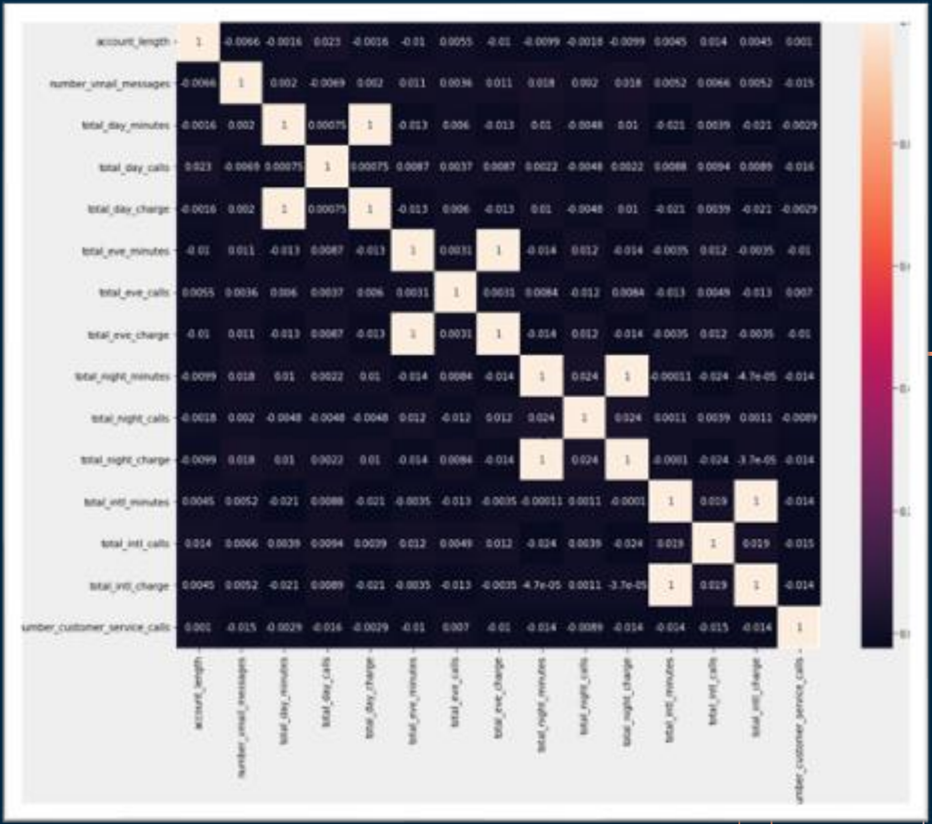


Insight

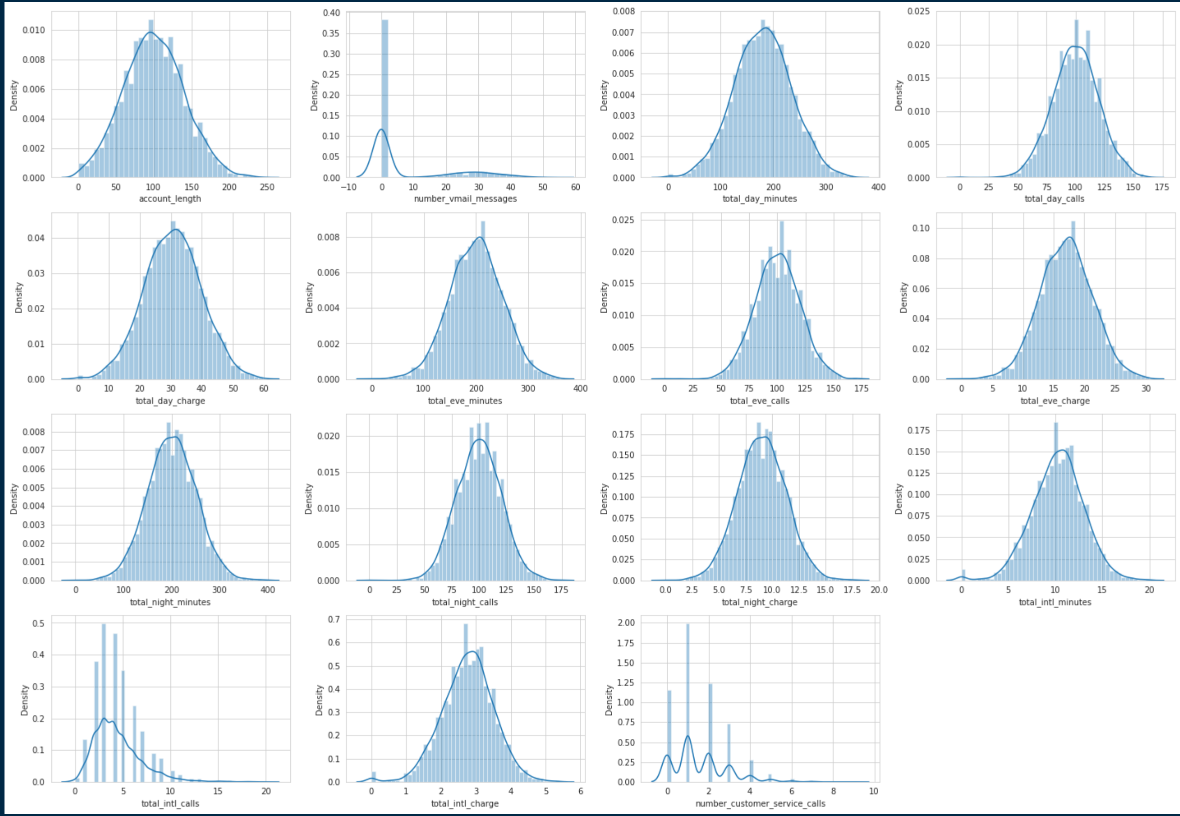
- West Virginia (WV) has the highest customer in telco provider
- New Jersey (NJ) has the highest customer who continuing theirs service, meanwhile West Virginia has the highest customer who quit theirs service
- The customer who continuing their service has more calling the call service than the customers who quit. Maybe the telco provider can more training the call service how to retain their customers
- The customer who continuing their service has more charge for calls than the remaining customers. Maybe telco provider has to works a effective plans to facilitate late payments.

Heat Map Correlation

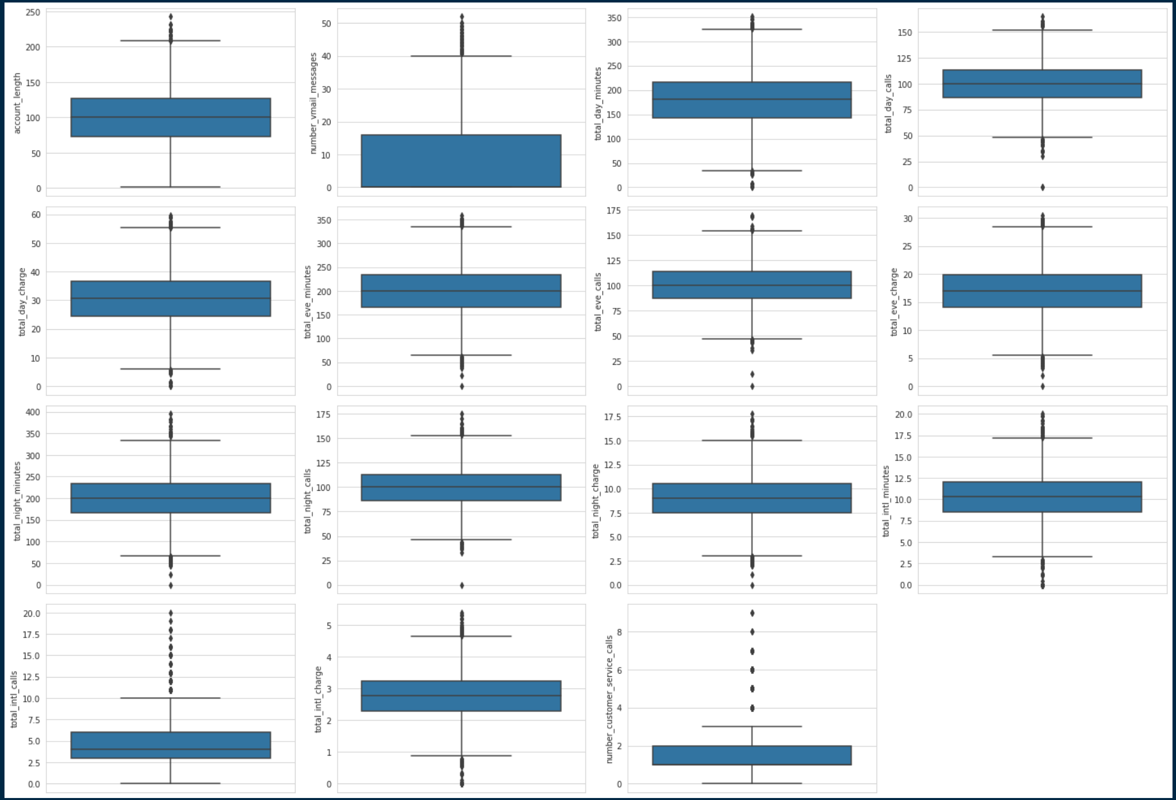
- From the heatmap, there are strong correlation (1):
  - total_day_minutes & total_day_charge
  - total_eve_minutes & total_eve_charge
  - total_night_minutes & total_night_charge
  - total_intl_minutes & total_intl_charge
- So we'll clean up "charge" column



Distribution Vizualitation



Box Plot Vizualitation



Insight

- account_length, total_day_minutes, total_eve_minutes, total_night_minutes, total_intl_minutes, total_day_calls, total_eve_calls, total_night_calls has distributed normally. Meanwhile total_intl_calls and number_customer_service_calls has distributed poisson.
- There's a lot outliner when we see in the boxplot graph. We will cap the outliers to avoid any impact on the prediction of the models.
- We can see in the 'churn' columns that its data is imbalanced with imbalanced with 85.4% Non-Churn cases and only 14.6% Churn cases. We'll have to balanced the data so the model will not bias in the non-churn case.

Data Pre-Processing

02

Data Cleaning

- There’s no missing value and duplicated value data train

```
Missing Value: 0

List of Missing Value:
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
number_customer_service_calls    0
churn                            0
dtype: int64

Duplicated Value: 0
```

Data Cleaning

- There's no missing value and duplicated value in data test

```
Missing Value: 0

List of Missing Value:
id                               0
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
number_customer_service_calls    0
dtype: int64

Duplicated Value: 0
```

Cut the Outliers

- Cut the outliers using Inter Quartile

```
The total of rows before filtering outliers: 4250
The total of rows after filtering outliers: 3516
```

Data Normalizing

- Normalize data train and test using standard scalar since standard scalar is good for normal distribution data

_vmail_messages	total_day_minutes	total_day_calls	total_eve_minutes	total_eve_calls
1.521141	-0.351617	1.200994	-0.101193	0.152926
-0.553619	1.201207	0.734564	-1.619139	0.513136
-0.553619	-0.254803	0.682738	-1.065488	1.130637
-0.553619	-0.438940	-1.079330	-1.988922	-0.310200
2.398925	1.489752	-0.820203	0.440201	0.564594
...	...	...	...	...
-0.553619	1.225886	0.786389	1.187937	0.050010
-0.553619	0.155234	-1.545760	0.885574	-0.618950
-0.553619	-0.042191	-0.561075	-1.414840	-0.927701
-0.553619	-0.178870	0.060832	-0.150225	1.336471
2.638320	1.055037	1.408296	0.460631	1.336471

Feature Enconding

- For encoding, we'll use one-hot encoding for both data train & test.

```
[ ] df_train4=pd.get_dummies(df_train3)
```

```
[ ] df_train4
```

	account_length	number_vmail_messages	total_day_minutes	total_day_calls	total_eve_minutes	total_eve_calls
0	0.190232	1.521141	-0.351617	1.200994	-0.101193	0.152926
1	0.957062	-0.553619	1.201207	0.734564	-1.619139	0.513136
3	-0.627721	-0.553619	-0.254803	0.682738	-1.065488	1.130637
5	1.212673	-0.553619	-0.438940	-1.079330	-1.988922	-0.310200
7	1.059307	2.398925	1.489752	-0.820203	0.440201	0.564594
...	...	...	...	...	...	...
4243	1.033746	-0.553619	1.225886	0.786389	1.187937	0.050010
4245	-0.423233	-0.553619	0.155234	-1.545760	0.885574	-0.618950
4246	-0.678843	-0.553619	-0.042191	-0.561075	-1.414840	-0.927701
4247	-0.627721	-0.553619	-0.178870	0.060832	-0.150225	1.336471
4248	-1.266746	2.638320	1.055037	1.408296	0.460631	1.336471

3516 rows × 70 columns

Machine Learning

03

Train the Model

- Split the features and label
- The features are columns on the right
- The label is churn

```
0   state
1   account_length
2   area_code
3   international_plan
4   voice_mail_plan
5   number_vmail_messages
6   total_day_minutes
7   total_day_calls
8   total_eve_minutes
9   total_eve_calls
10  total_night_minutes
11  total_night_calls
12  total_intl_minutes
13  total_intl_calls
14  number_customer_service_calls
```

Result of the Model

- Train the model into these machine learning:
  - Logistic Regression
  - K-Nearest Neighbors
  - Decision Tree
  - Random Forest Classifier
  - Naïve Bayes
  - Linear Discriminant Analysis
  - Support vector machines
- The best accuracy score is Random Forest so we'll hyperparameter tuning this model.

```
The score of accuracy is:
LR 0.9110862926225991
KNN 0.9014752277832463
CART 0.9288685293152621
RF 0.949497741096893
NB 0.5753792180914161
LDA 0.8940246334014791
SVM 0.9288672673582191
```

Hyperparameter Tuning the Model

- Tuning the Random Forest model to search for the best parameters.
- List of the parameters can see on the right

```
chosen_model = RandomForestClassifier()
n_estimators = list(range(100, 2000, 100)) #the number of forest in the tree
max_features = ['auto', 'sqrt', 'log2']
max_depth = list(range(5, 50, 5))
criterion = ['gini', 'entropy']
min_samples_split = list(range(1,10))
min_samples_leaf = list(range(1,10))
bootstrap = [True, False]

param = {'n_estimators': n_estimators,
                  'max_features' : max_features,
                  'max_depth' : max_depth,
                  'criterion' : criterion,
                  'min_samples_split' : min_samples_split,
                  'min_samples_leaf' : min_samples_leaf,
                  'bootstrap': bootstrap}
```

```
random = RandomizedSearchCV(estimator=chosen_model, param_distributions=param, cv=3,
                            n_jobs=-1, verbose=2, n_iter=100, random_state=42)
random.fit(X_train,y_train)
print(random.best_params_)
```

```
best_model = RandomForestClassifier(n_estimators=1500, max_features ='auto',
                                    max_depth=45, criterion='gini',
                                    min_samples_split=6, min_samples_leaf=1,
                                    bootstrap=False, random_state = 42)
best_model.fit(X_train,y_train)
y_pred = best_model.predict(X_test)
```

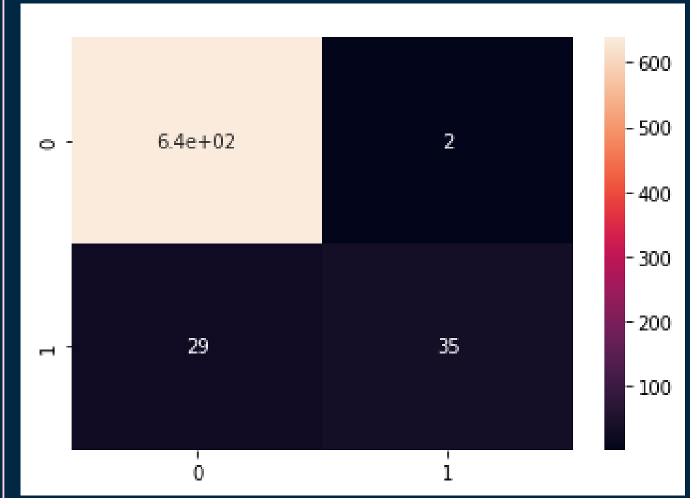
Hyperparameter Tuning the Model

- This is the new score after hyperparameter tuning the model

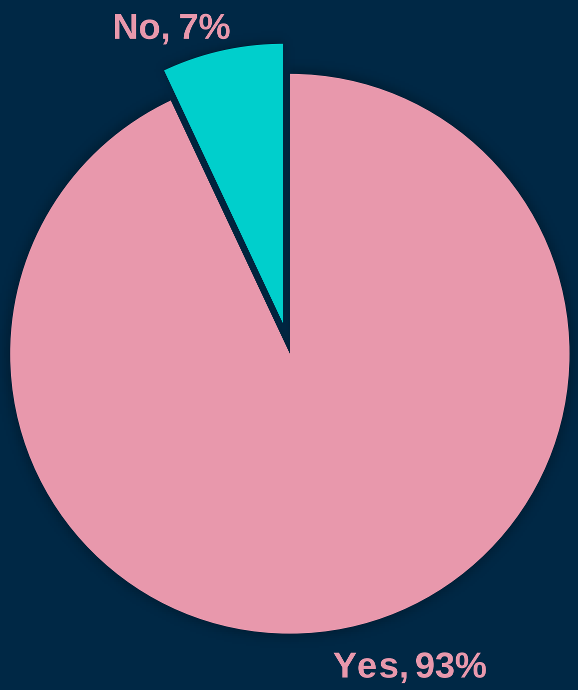
```
Accuracy Score is:
0.9559659090909091

Improvement of 0.68%.

Precision: 0.946

Recall: 0.547

F1 Score: 0.693
```

Hyperparameter Tuning the Model

- The accuracy of the model is 95% with the correct churn prediction being 35. Not the churn but really not the churn prediction is 640, churn but actually not churn prediction is 29, not churn but actually churn prediction is 2



Data Test Prediction



Yes

The customer who decided to leave about 86%

No

The customer who decided to stay about 14%

Conclusion

Given the accuracy score, the model we'll use is Random Forest

From the data train, there are a lot of imbalance data so we have to balanced data to get a better model result

You can check the notebook link

Thank You