FABYOLA LARASATI MASYITA

CHALLENGE – CHAPTER 2

CUSTOMER CHURN PREDICTION

GOOGLE COLABORATORY

https://colab.research.google.com/drive/1Q4Hids9HfFjl4x7Gr3z_WBTJu25Z65fe?usp=sharing

TO DO

IMPORT DATA

STATISTICAL ANALYSIS (EXPLORATORY DATA ANALYSIS)

DATA PREPROCESSING

MACHINE LEARNING ALGORITHM

USE CASE INTERPRETATION

PROBLEM

Perkembangan industri telekomunikasi sangatlah cepat, hal ini dapat dilihat dari perilaku masyarakat yang menggunakan internet dalam berkomunikasi. Perilaku ini menyebabkan banyaknya perusahaan telekomunikasi dan meningkatnya internet service provider yang dapat menimbulkan persaingan antar provider.

Pelanggan memiliki hak dalam memilih provider yang sesuai dan dapat beralih dari provider sebelumnya yang diartikan sebagai Customer Churn.

Peralihan ini dapat menyebabkan berkurangnya pendapatan bagi perusahaan telekomunikasi sehingga penting untuk ditangani.

IMPORT DATA

```
from google.colab import files
uploaded = files.upload()
```

Choose Files train.csv

- **train.csv**(text/csv) - 387621 bytes, last modified: 10/3/2022 - 100% done

```
Saving train.csv to train.csv
```

```
df_train = pd.read_csv(io.BytesIO(uploaded['train.csv']))
df_train.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
0	OH	107	area_code_415	no	yes	26	161.6	123	27.47	195.5	
1	NJ	137	area_code_415	no	no	0	243.4	114	41.38	121.2	
2	OH	84	area_code_408	yes	no	0	299.4	71	50.90	61.9	
3	OK	75	area_code_415	yes	no	0	166.7	113	28.34	148.3	
4	MA	121	area_code_510	no	yes	24	218.2	88	37.09	348.5	

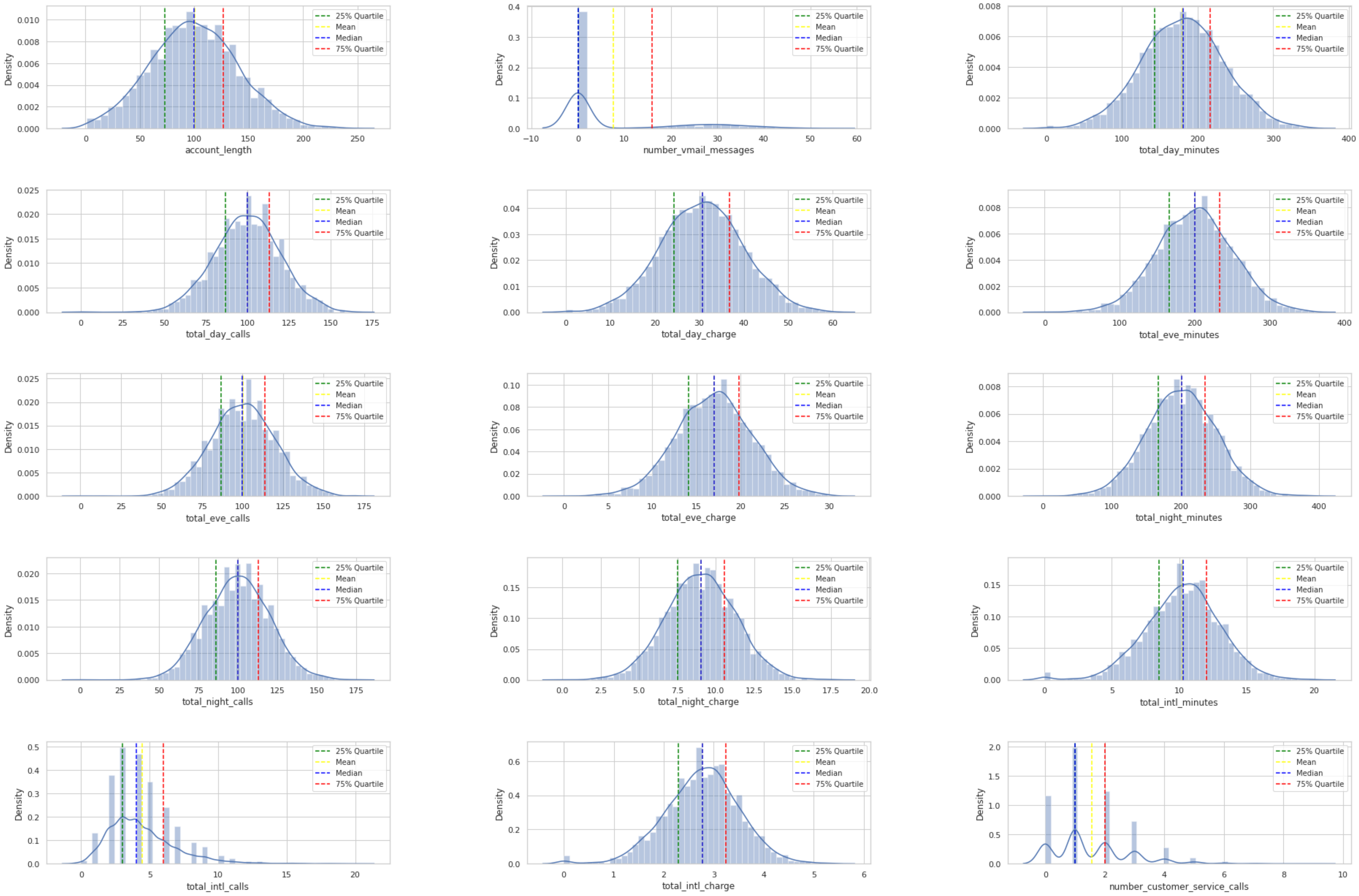
STATISTICAL ANALYSIS

(EXPLORATORY DATA ANALYSIS)

STATISTICAL DESCRIPTIVE

	account_length	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_n:
count	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4
mean	100.236235	7.631765	180.259600	99.907294	30.644682	200.173906	100.176471	17.015012	200.527882	
std	39.698401	13.439882	54.012373	19.850817	9.182096	50.249518	19.908591	4.271212	50.353548	
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	73.000000	0.000000	143.325000	87.000000	24.365000	165.925000	87.000000	14.102500	167.225000	
50%	100.000000	0.000000	180.450000	100.000000	30.680000	200.700000	100.000000	17.060000	200.450000	
75%	127.000000	16.000000	216.200000	113.000000	36.750000	233.775000	114.000000	19.867500	234.700000	
max	243.000000	52.000000	351.500000	165.000000	59.760000	359.300000	170.000000	30.540000	395.000000	

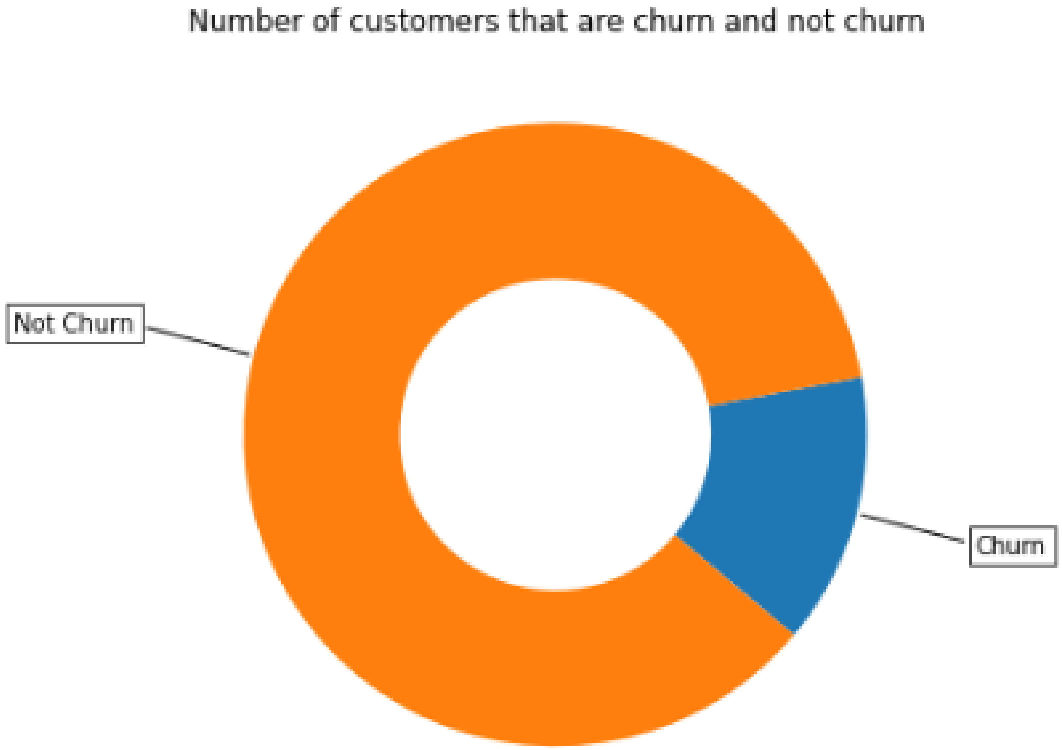
Descriptive Statistics



CUSTOMER CHURN PREDICTION

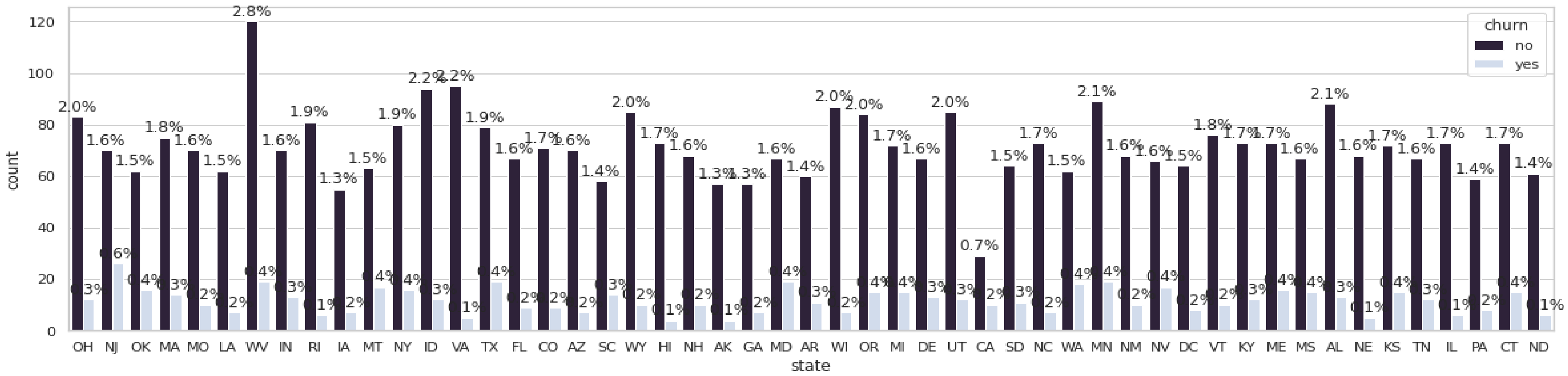
CHURN STATUS

```
Number of customers who churn: 598 , ( 14.070588235294117 % )
Number of customers who not churn: 3652 , ( 85.92941176470589 % )
```

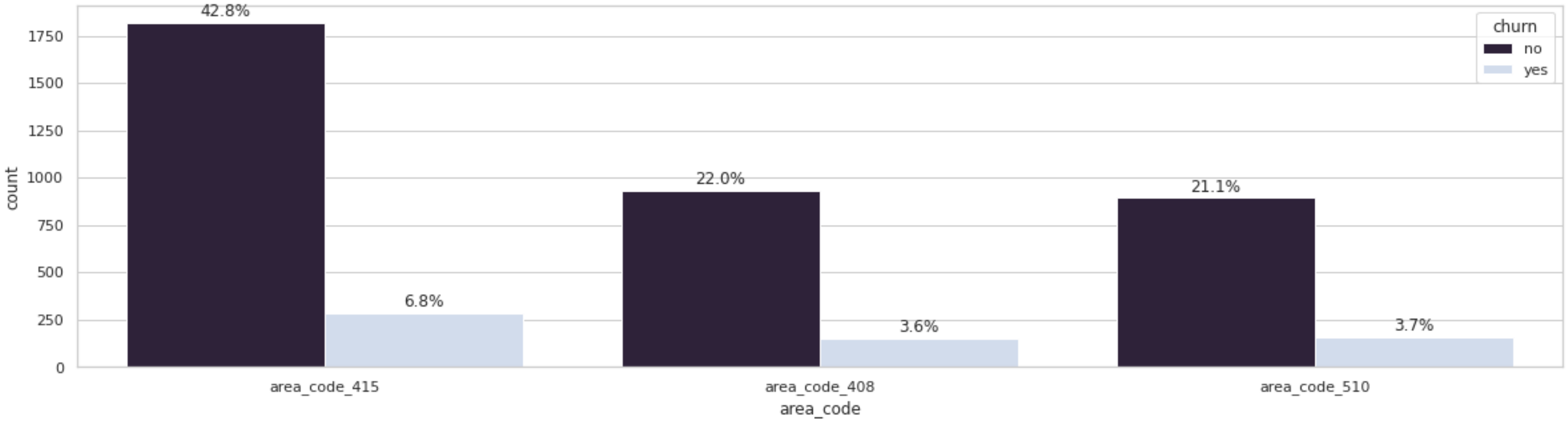


CATEGORICAL DATA ANALYSIS

Customer churn by state



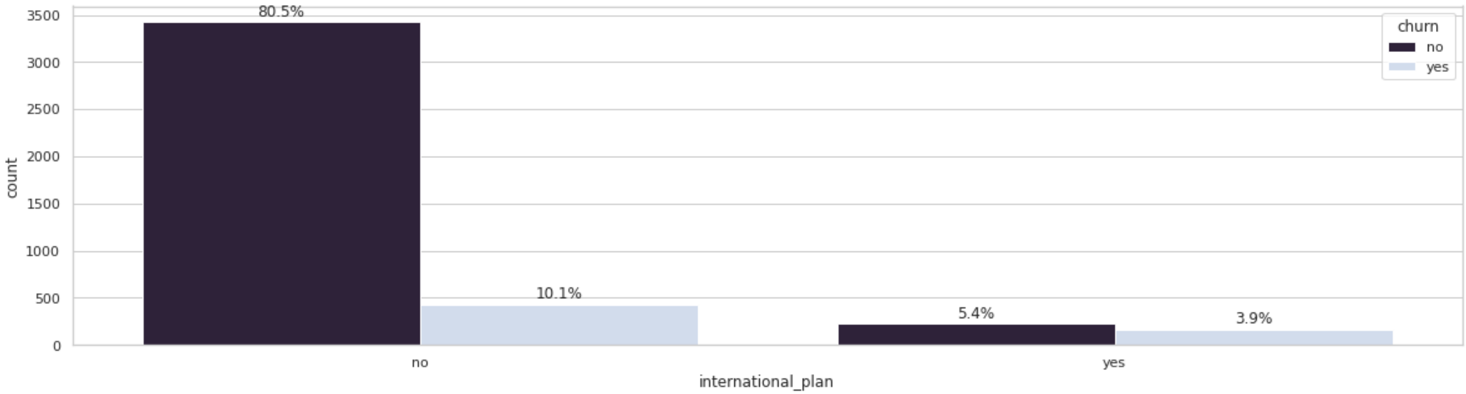
Customer churn by area code



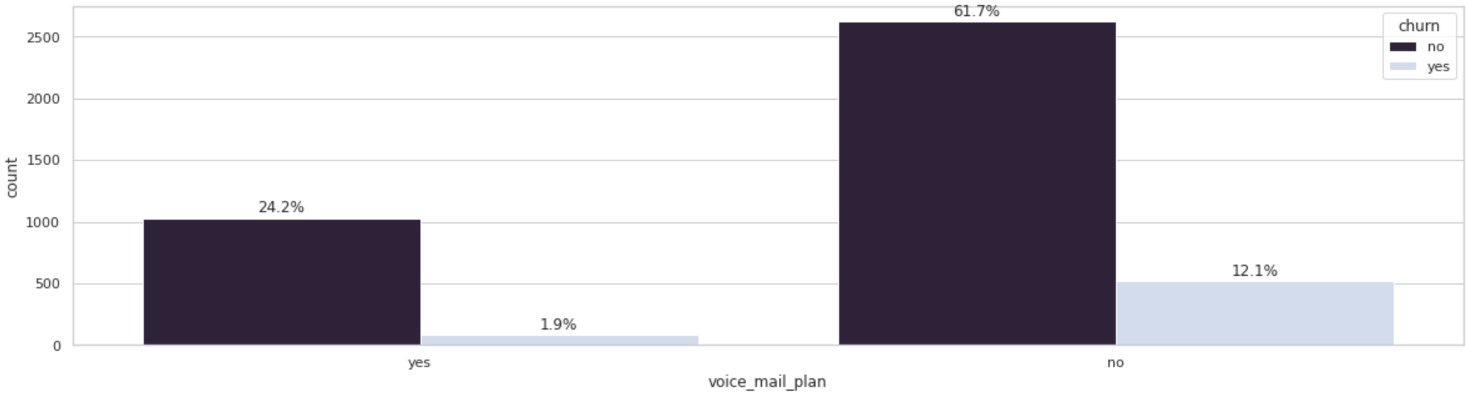
CUSTOMER CHURN PREDICTION

CATEGORICAL DATA ANALYSIS

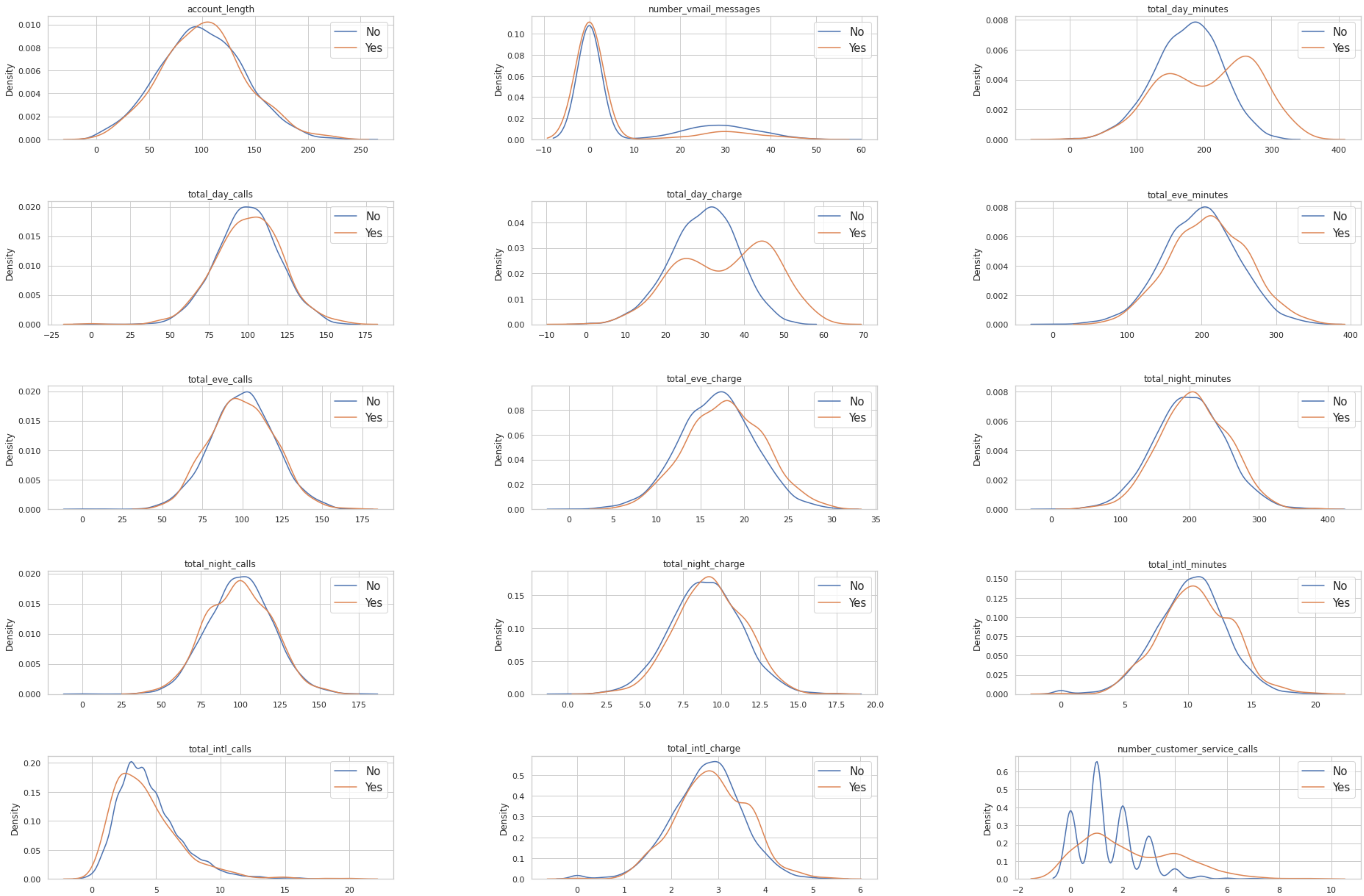
Customer churn by international plan



Customer churn by voice mail plan



Bivariate Analysis by Churn Status



CUSTOMER CHURN PREDICTION

DATA PREPROCESSING

Data Cleaning

Features Engineering

Data Splitting & Data Scaling

CUSTOMER CHURN PREDICTI

DATA CLEANING

Identify Missing Value

```
1 df_train.isna().any()
```

```
state                          False
account_length                 False
area_code                      False
international_plan             False
voice_mail_plan                False
number_vmail_messages          False
total_day_minutes              False
total_day_calls                False
total_day_charge               False
total_eve_minutes              False
total_eve_calls                False
total_eve_charge               False
total_night_minutes            False
total_night_calls              False
total_night_charge             False
total_intl_minutes             False
total_intl_calls               False
total_intl_charge              False
number_customer_service_calls  False
churn                          False
dtype: bool
```

Identify Duplicate Value

```
1 df_train.duplicated().any()
```

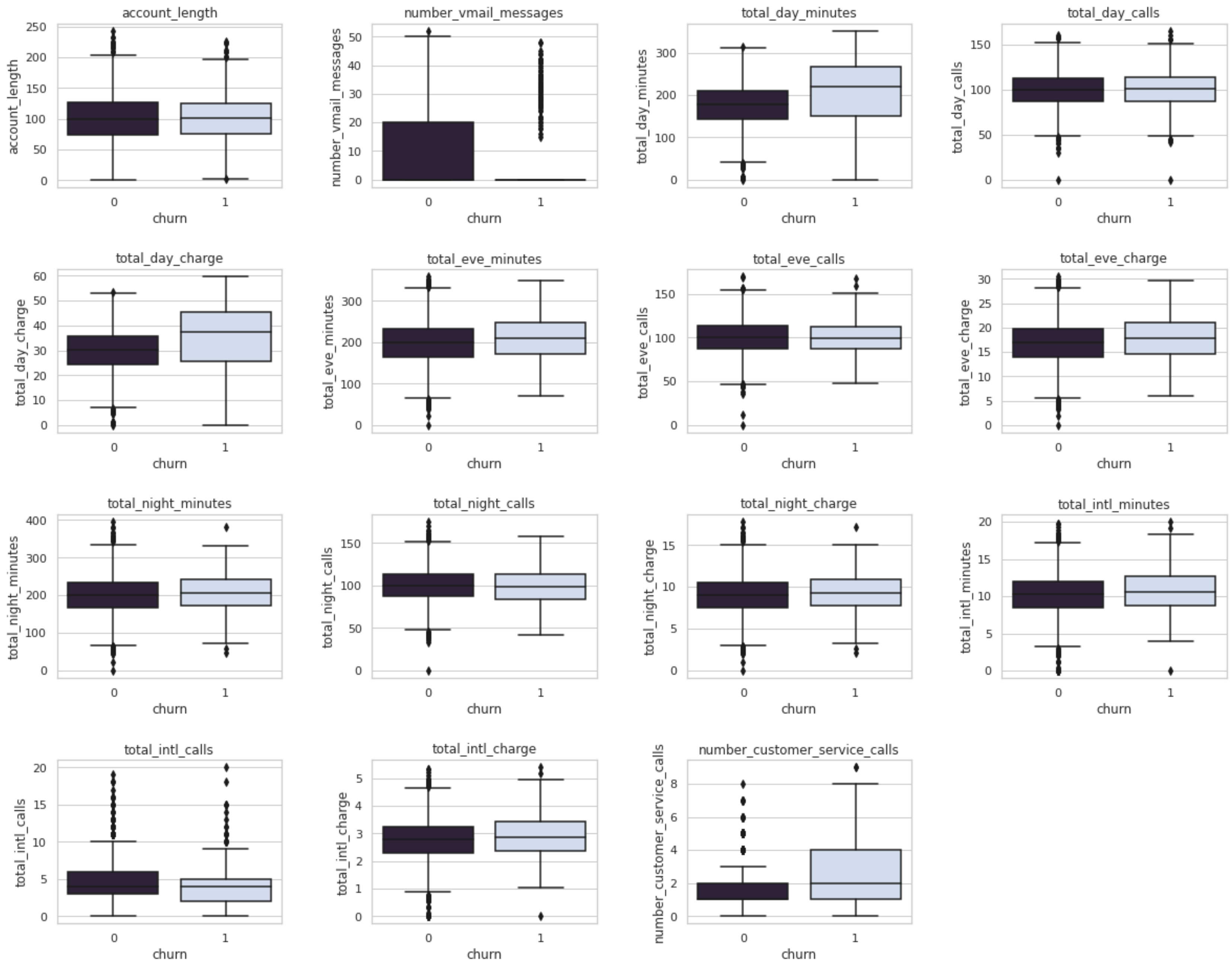
```
False
```

Identify Outliers

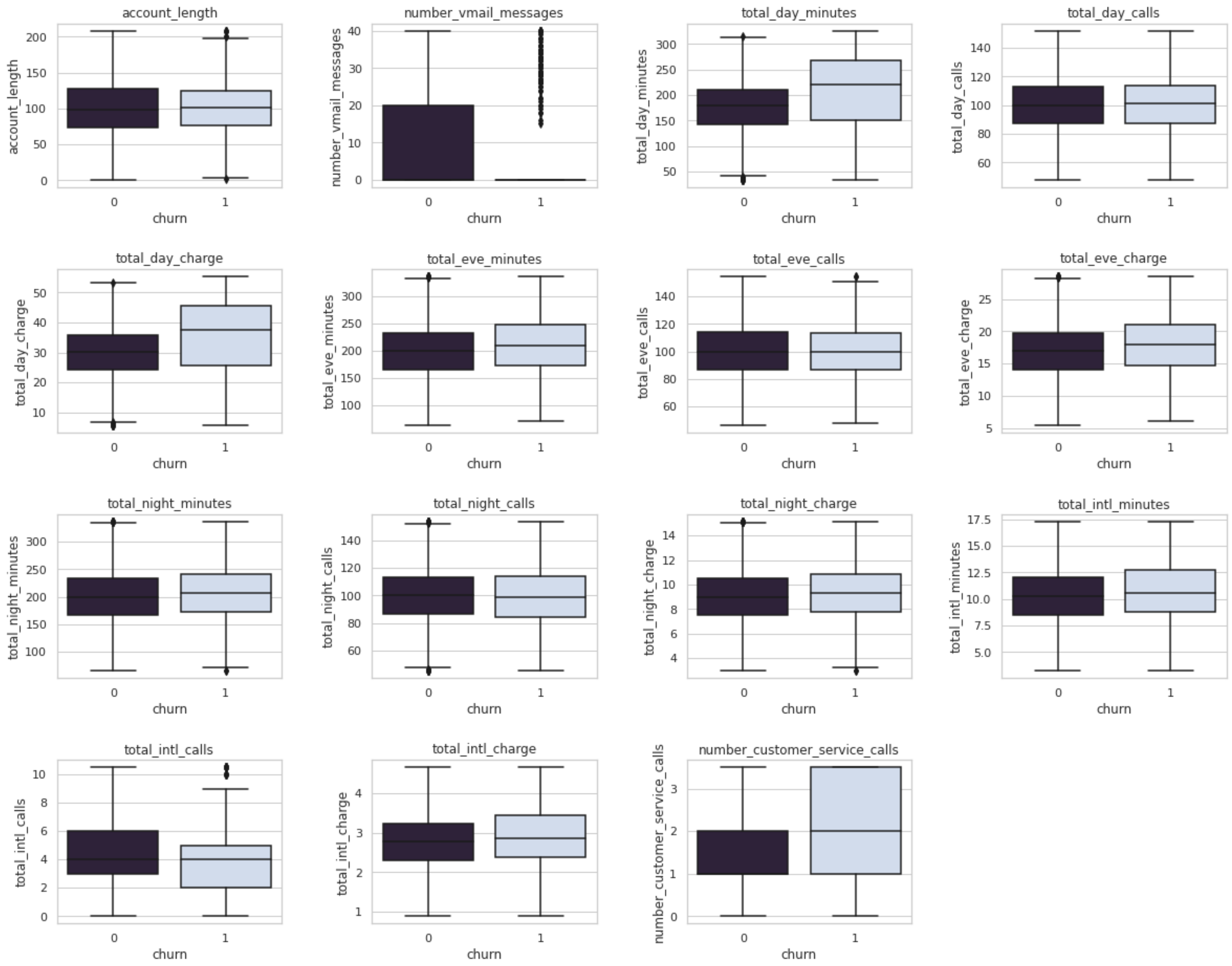
```
1 check_out(df_out[numerical]).sum()
```

```
account_length                    20
number_vmail_messages             86
total_day_minutes                 25
total_day_calls                   28
total_day_charge                  26
total_eve_minutes                 34
total_eve_calls                   24
total_eve_charge                  34
total_night_minutes               37
total_night_calls                 33
total_night_charge                37
total_intl_minutes                62
total_intl_calls                 100
total_intl_charge                 62
number_customer_service_calls    335
dtype: int64
```

Outlier Detection



Remove Outliers



FEATURES ENGINEERING

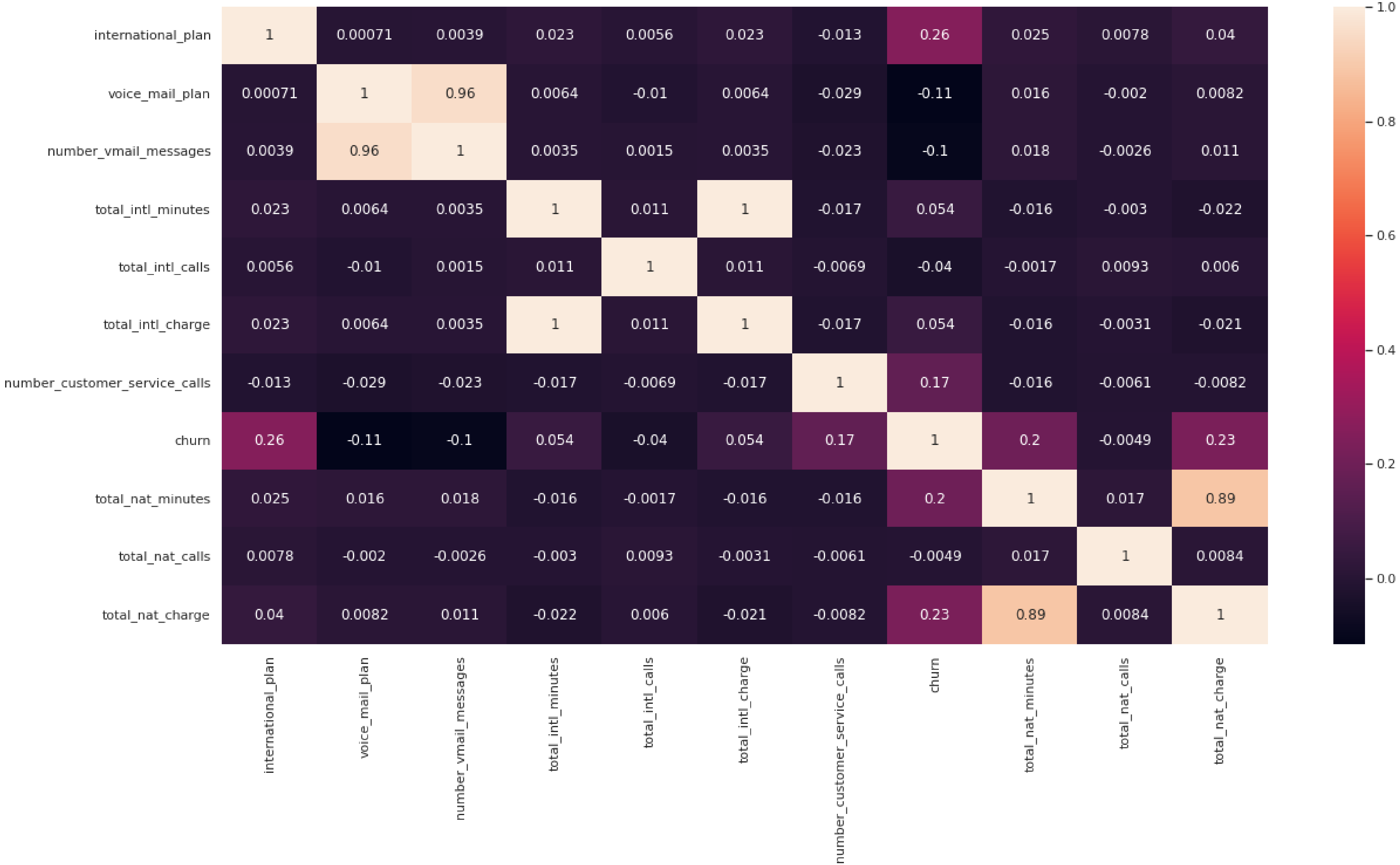
```
def preprocess(df):
    df['total_nat_minutes'] = df['total_day_minutes'] + df['total_eve_minutes'] + df['total_night_minutes']
    df['total_nat_calls'] = df['total_day_calls'] + df['total_eve_calls'] + df['total_night_calls']
    df['total_nat_charge'] = df['total_day_charge'] + df['total_eve_charge'] + df['total_night_charge']

    df['international_plan'].replace({'no': 0, 'yes': 1}, inplace=True)
    df['voice_mail_plan'].replace({'no': 0, 'yes': 1}, inplace=True)

    df.drop(columns=['state', 'area_code'], inplace=True)
    df.drop(columns=['account_length'] , axis=1 , inplace=True)
    df.drop(columns=['total_day_minutes', 'total_eve_minutes', 'total_night_minutes'], inplace=True)
    df.drop(columns=['total_day_calls', 'total_eve_calls', 'total_night_calls'], inplace=True)
    df.drop(columns=['total_day_charge', 'total_eve_charge', 'total_night_charge'], inplace=True)

    return df
```

	international_plan	voice_mail_plan	number_vmail_messages	total_intl_minutes	total_intl_calls	total_intl_charge	number_customer_service_calls	churn	total_nat_minutes	total_nat_calls	total_nat_charge
0	0	1	26.0	13.7	3.0	3.70	1.0	0	611.50	329.0	55.540
1	0	0	0.0	12.2	5.0	3.29	0.0	0	527.20	328.0	59.000
2	1	0	0.0	6.6	7.0	1.78	2.0	0	560.45	248.0	65.215
3	1	0	0.0	10.1	3.0	2.73	3.0	0	501.90	356.0	49.360
4	0	1	24.0	7.5	7.0	2.03	3.0	0	766.35	314.0	75.175


international_plan
voice_mail_plan
number_vmail_messages
total_intl_minutes
total_intl_calls
total_intl_charge
number_customer_service_calls
churn
total_nat_minutes
total_nat_calls
total_nat_charge
1 0.00071 0.0039 0.023 0.0056 0.023 -0.013 0.26 0.025 0.0078 0.04
0.00071 1 0.96 0.0064 -0.01 0.0064 -0.029 -0.11 0.016 -0.002 0.0082
0.0039 0.96 1 0.0035 0.0015 0.0035 -0.023 -0.1 0.018 -0.0026 0.011
0.023 0.0064 0.0035 1 0.011 1 -0.017 0.054 -0.016 -0.003 -0.022
0.0056 -0.01 0.0015 0.011 1 0.011 -0.0069 -0.04 -0.0017 0.0093 0.006
0.023 0.0064 0.0035 1 0.011 1 -0.017 0.054 -0.016 -0.0031 -0.021
-0.013 -0.029 -0.023 -0.017 -0.0069 -0.017 1 0.17 -0.016 -0.0061 -0.0082
0.26 -0.11 -0.1 0.054 -0.04 0.054 0.17 1 0.2 -0.0049 0.23
0.025 0.016 0.018 -0.016 -0.0017 -0.016 -0.016 0.2 1 0.017 0.89
0.0078 -0.002 -0.0026 -0.003 0.0093 -0.0031 -0.0061 -0.0049 0.017 1 0.0084
0.04 0.0082 0.011 -0.022 0.006 -0.021 -0.0082 0.23 0.89 0.0084 1
1.0
0.8
0.6
0.4
0.2
0.0

DATA SPLITTING AND DATA SCALING

Data Splitting

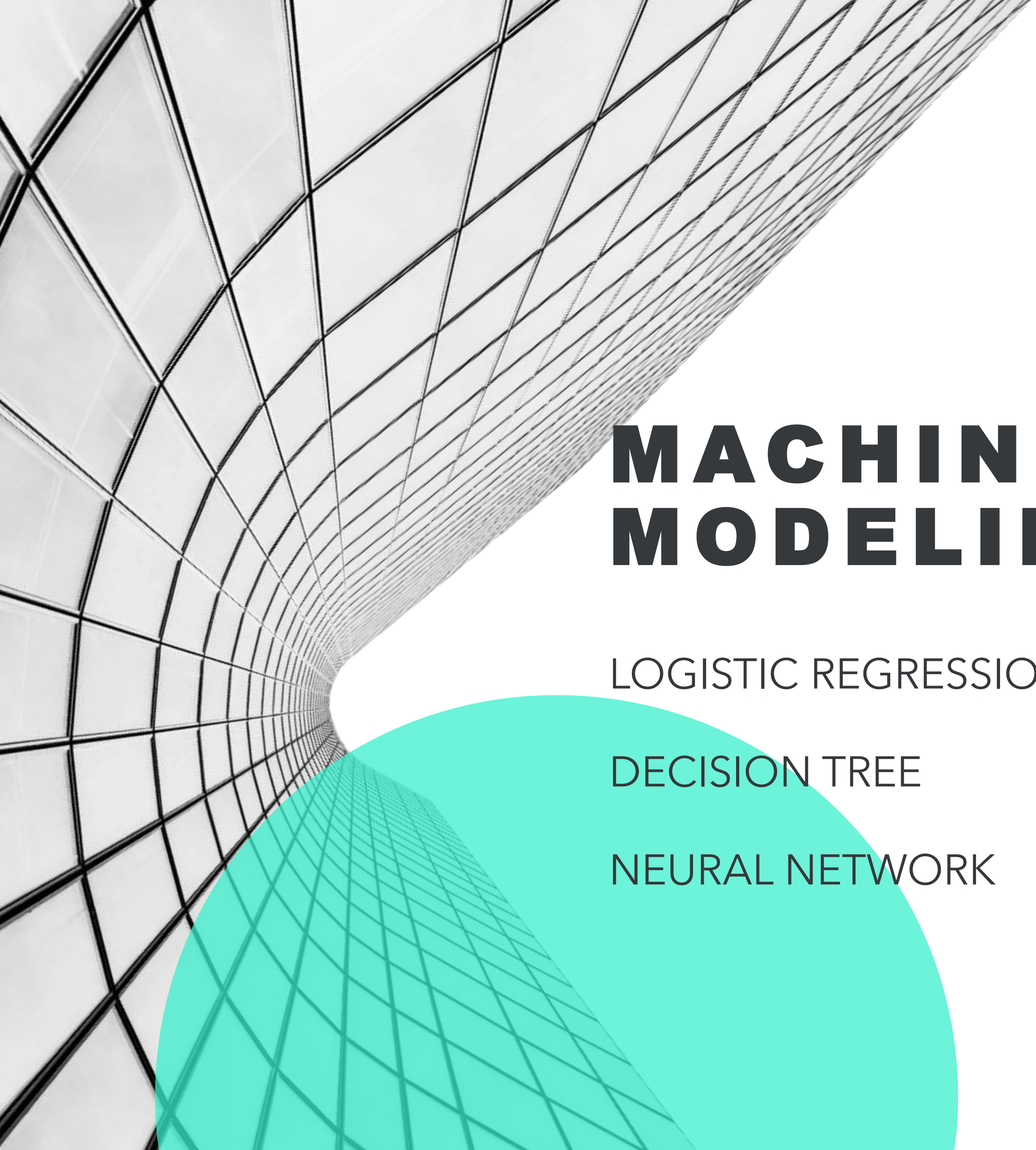
```
from sklearn.model_selection import train_test_split
x = df_train_clean.drop('churn',axis=1).values
y = df_train_clean.churn.values
# splitting tha data into test and train
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
print(x_train.shape, x_test.shape)
```

```
(3400, 10) (850, 10)
```

Data Scaling

```
from sklearn.preprocessing import MinMaxScaler

# creating the object of minmax scaler
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```



MACHINE LEARNING MODELING

LOGISTIC REGRESSION

GAUSSIAN NAIVE BAYES

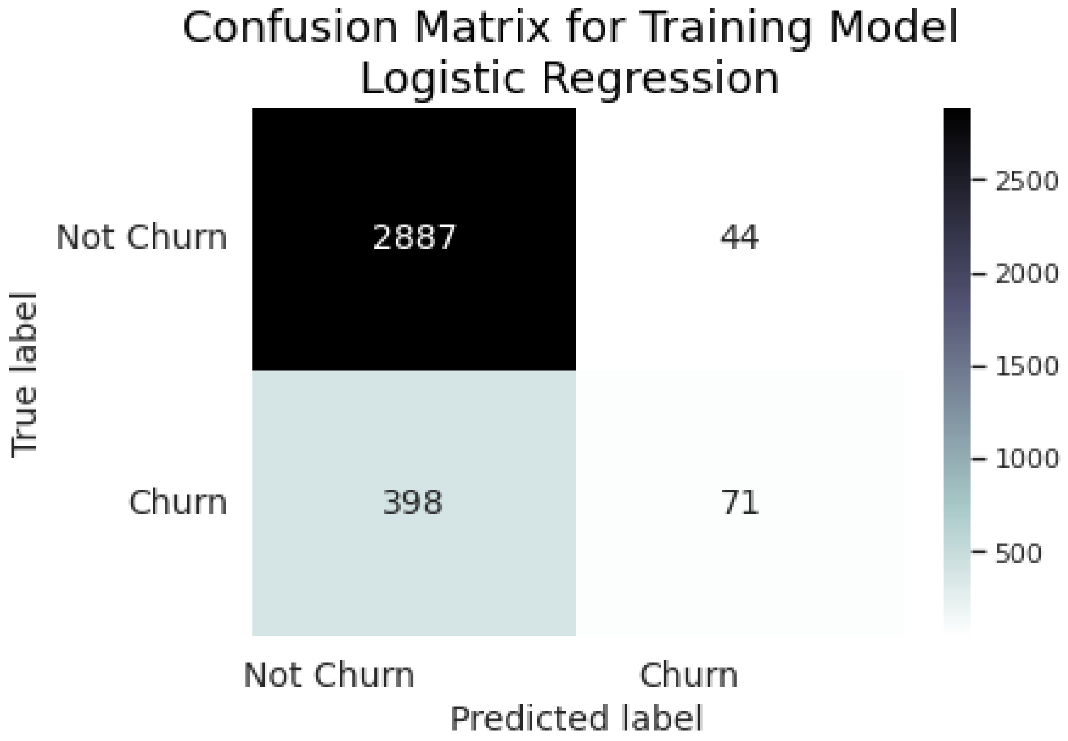
DECISION TREE

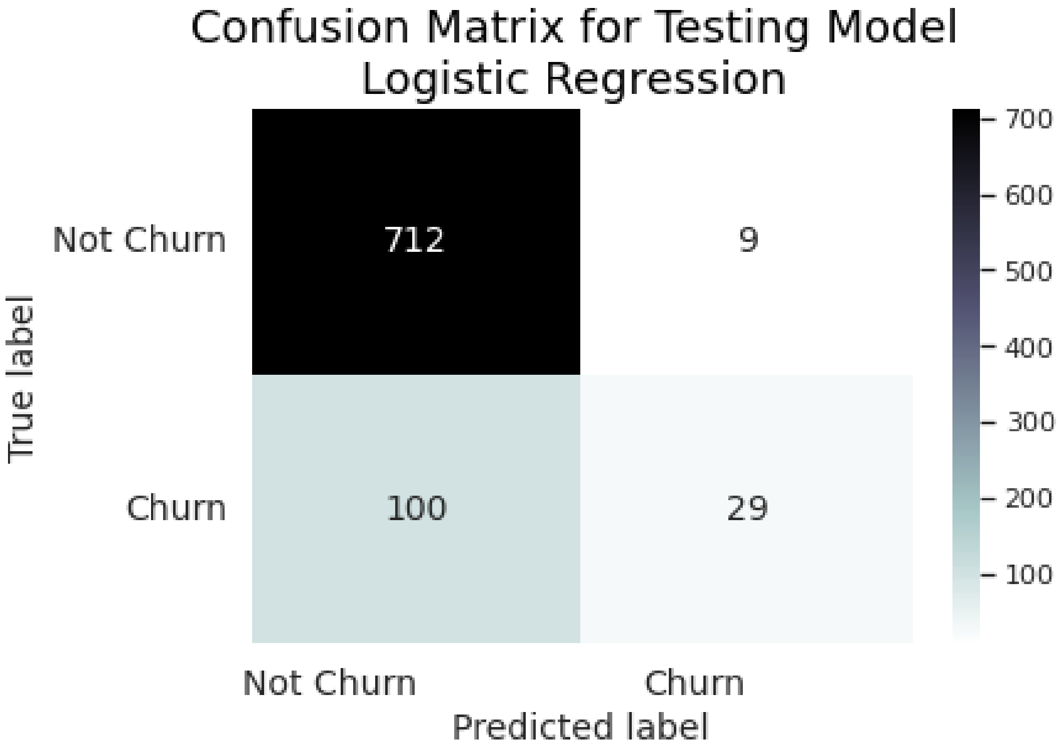
RANDOM FOREST

NEURAL NETWORK

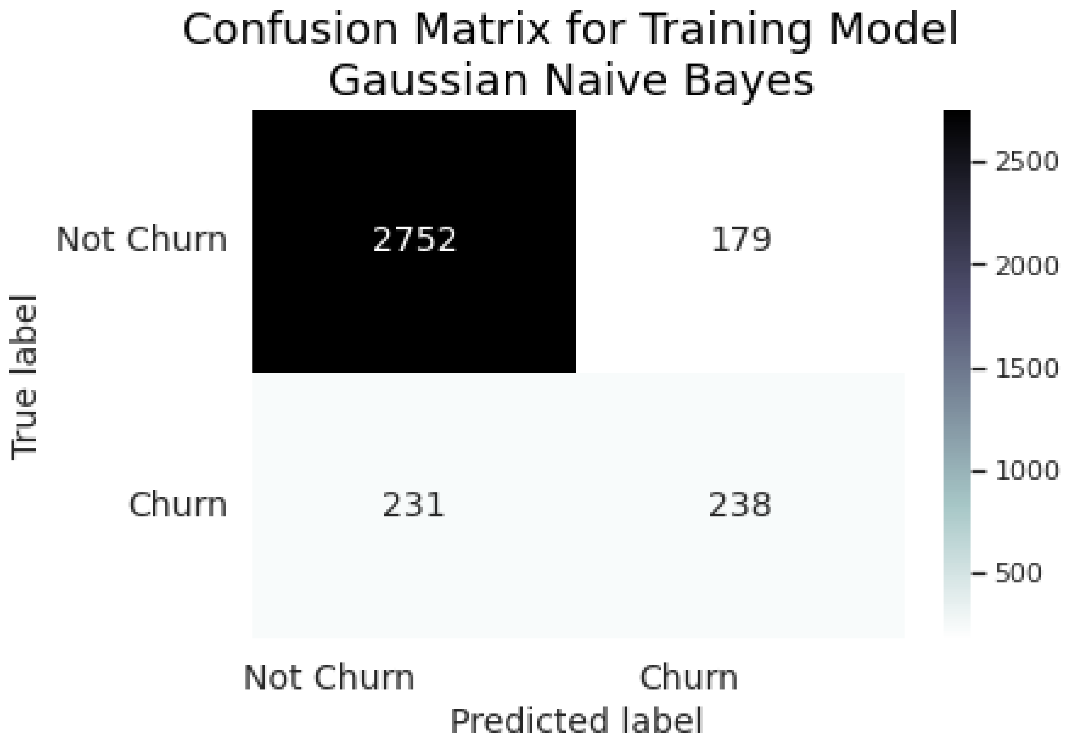
SUPPORT VECTOR MACHINE

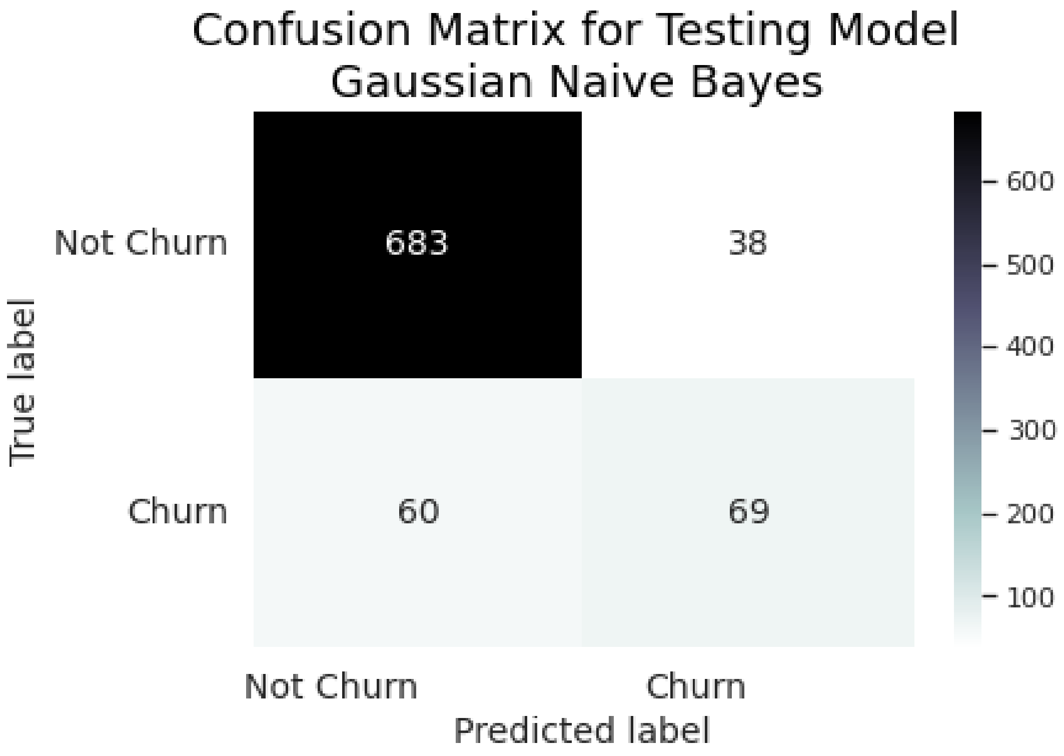
LOGISTIC REGRESSION



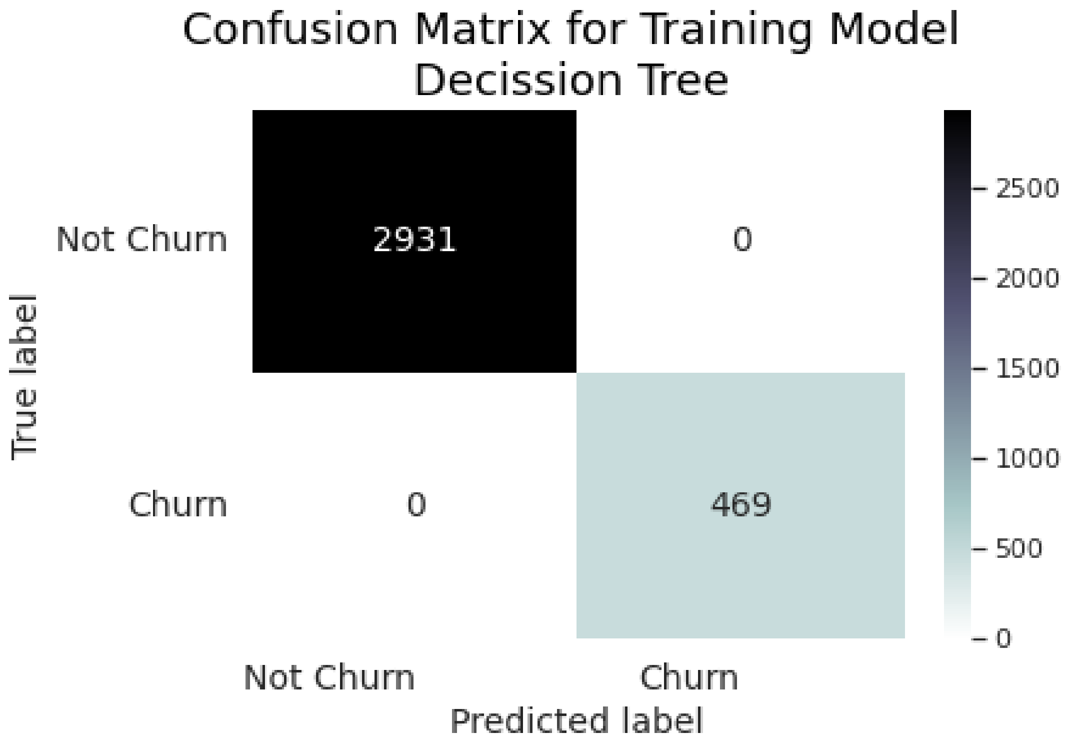


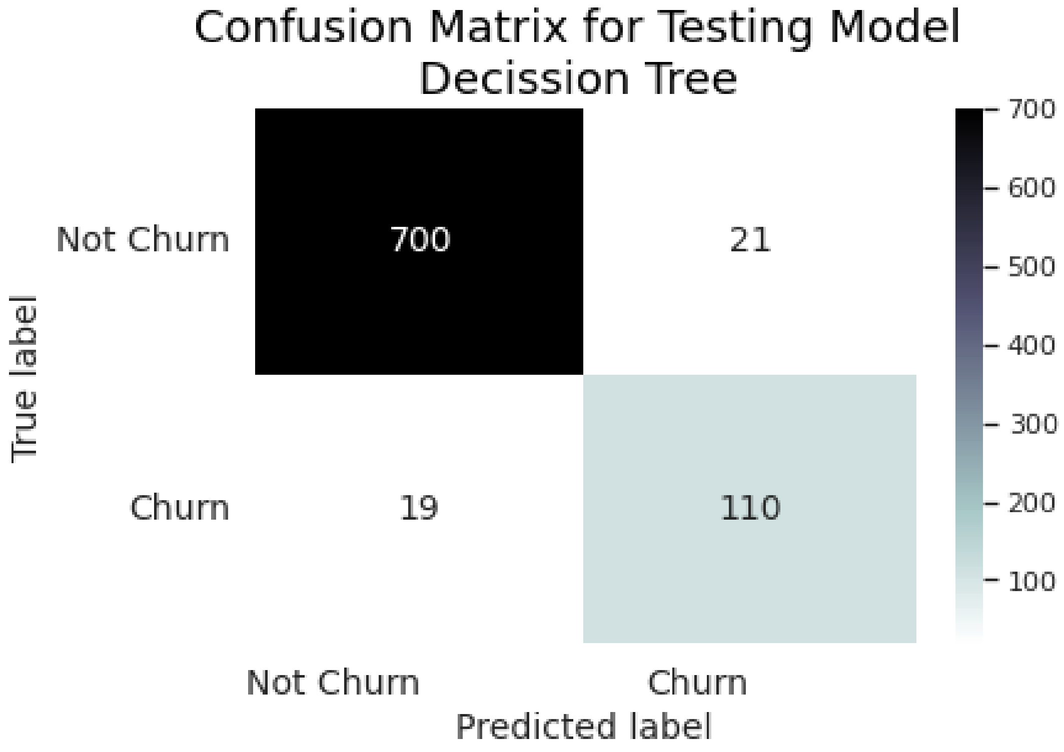
GAUSSIAN NAÏVE BAYES



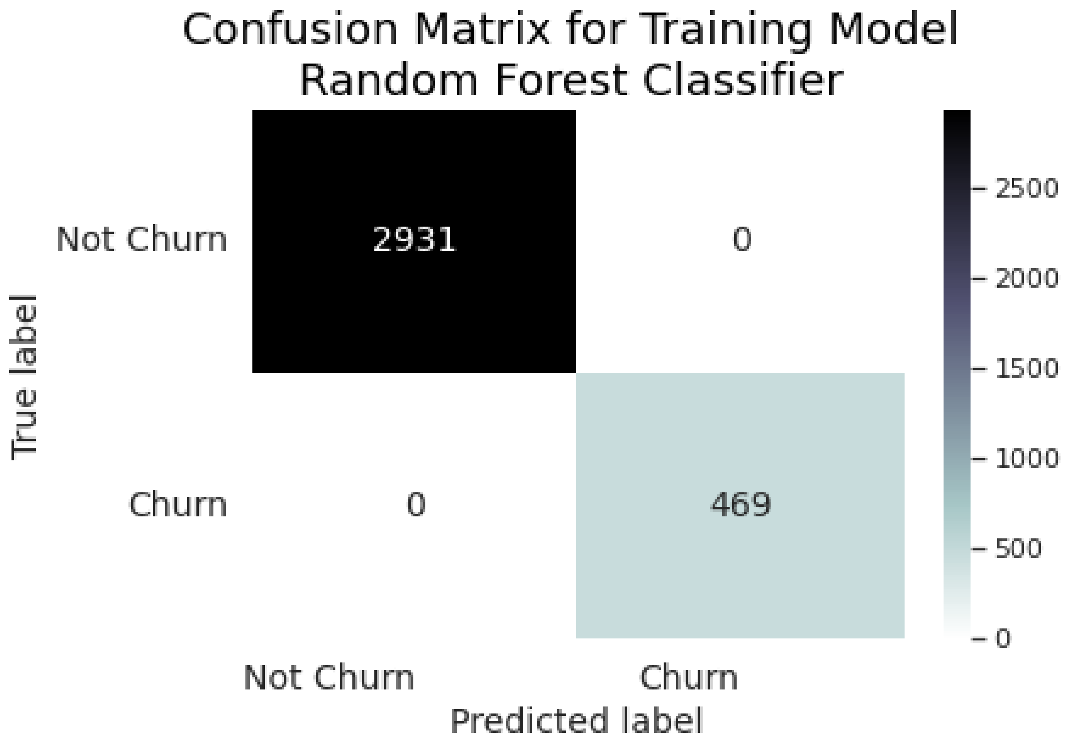


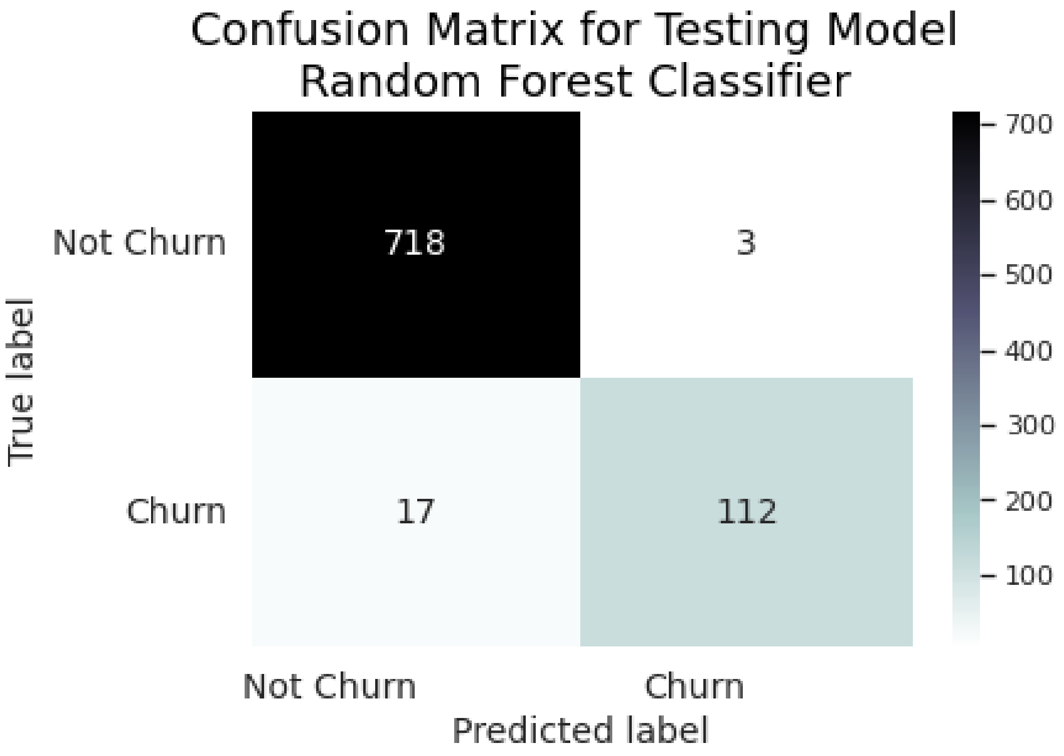
DECISION TREE



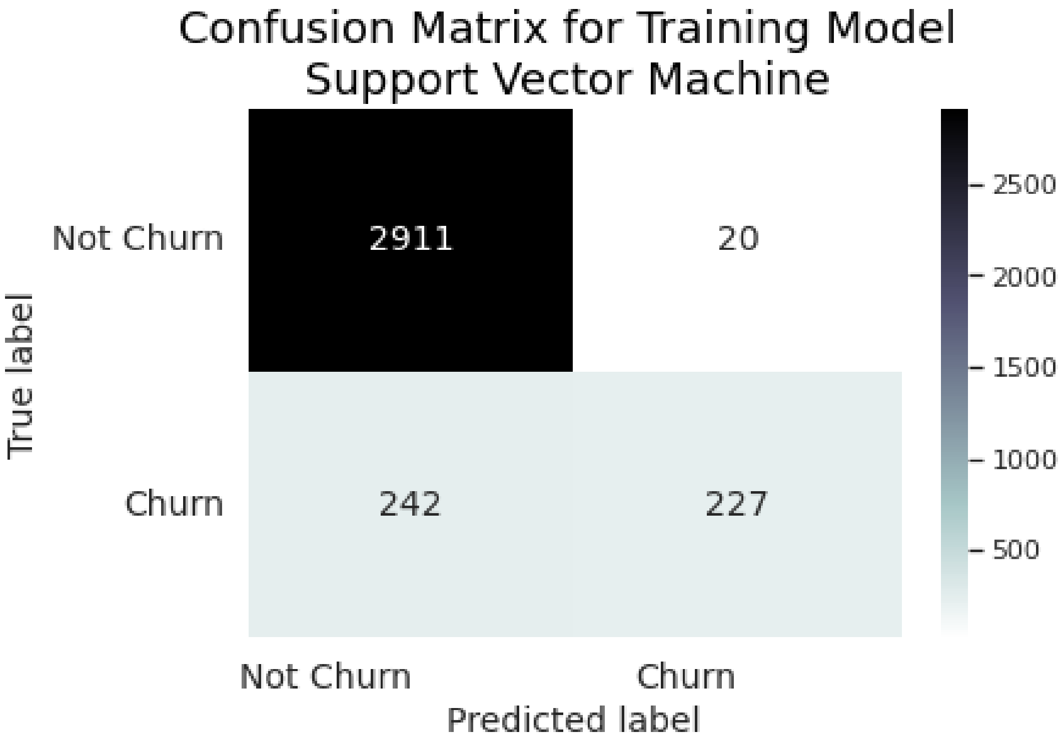


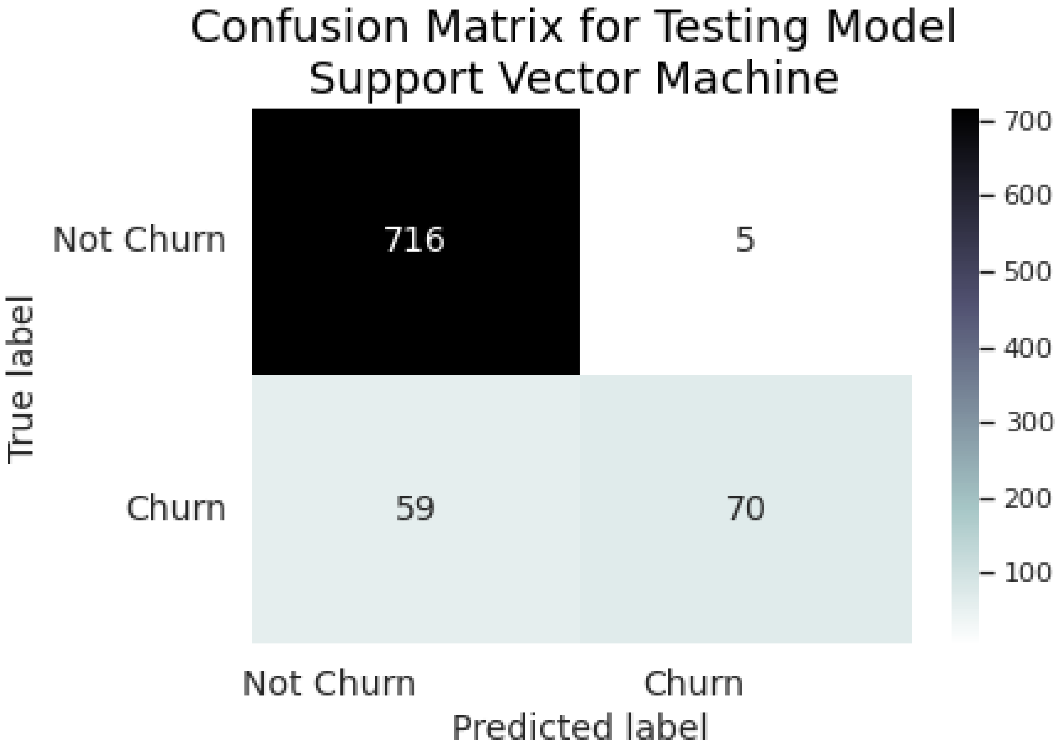
RANDOM FOREST CLASSIFIER



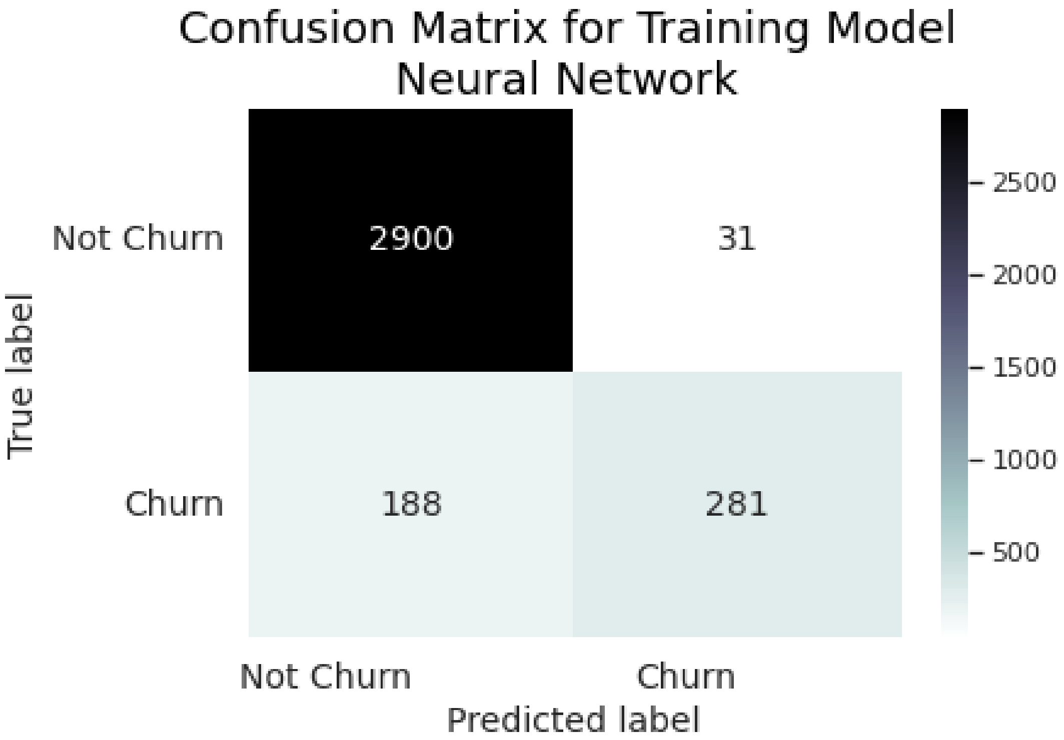


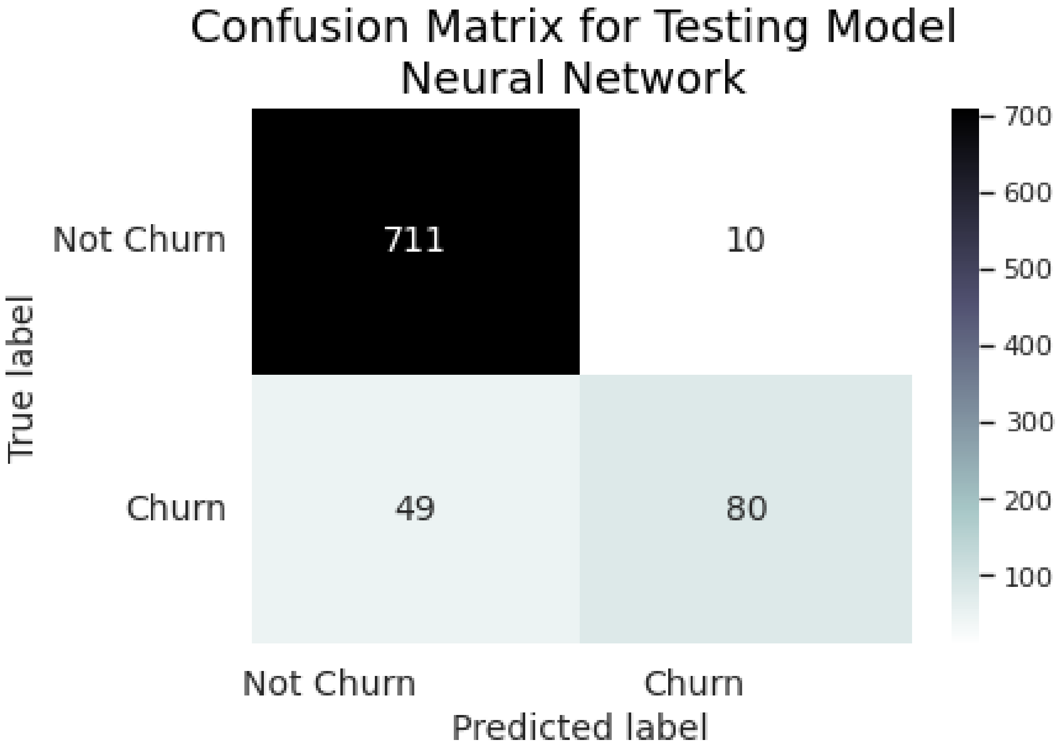
SUPPORT VECTOR MACHINE





NEURAL NETWORK





MODEL SUMMARY

	Model	Accuracy	Precision	f1_score
2	Random Forest Classifier	0.976	0.974	0.918
5	Decission Tree	0.953	0.840	0.846
0	Neural Network	0.932	0.882	0.739
1	Support Vector Machine	0.925	0.933	0.686
4	Gaussian Naive Bayes	0.885	0.645	0.585
3	Logistic Regression	0.872	0.763	0.347

USE CASE INTERPRETATION

IMPORT DATA

PREPROCESSING DATA

CHURN PREDICTION

DATA TEST

Import Data

```
1 uploaded = files.upload()
```

Choose Files test.csv

- **test.csv**(text/csv) - 69310 bytes, last modified: 10/3/2022 - 100% done

Saving test.csv to test.csv

```
1 df_test = pd.read_csv(io.BytesIO(uploaded['test.csv']))
2 df_test.head()
```

	id	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
0	1	KS	128	area_code_415	no	yes	25	265.1	110	45.07	197.4	
1	2	AL	118	area_code_510	yes	no	0	223.4	98	37.98	220.6	
2	3	IA	62	area_code_415	no	no	0	120.7	70	20.52	307.2	
3	4	VT	93	area_code_510	no	no	0	190.7	114	32.42	218.2	
4	5	NE	174	area_code_415	no	no	0	124.3	76	21.13	277.1	

Preprocessing Data

	id	international_plan	voice_mail_plan	number_vmail_messages	total_intl_minutes	total_intl_calls	total_intl_charge	number_customer_service_calls	total_nat_minutes	total_na
0	1	0	1	25	10.0	3	2.70	1	707.2	
1	2	1	0	0	6.3	6	1.70	0	647.9	
2	3	0	0	0	13.1	6	3.54	4	630.9	
3	4	0	0	0	8.1	3	2.19	3	538.5	
4	5	0	0	0	15.5	5	4.19	3	652.1	

Info Data

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 750 entries, 0 to 749
Data columns (total 20 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   id                             750 non-null    int64
 1   state                          750 non-null    object
 2   account_length                 750 non-null    int64
 3   area_code                      750 non-null    object
 4   international_plan             750 non-null    object
 5   voice_mail_plan                750 non-null    object
 6   number_vmail_messages          750 non-null    int64
 7   total_day_minutes              750 non-null    float64
 8   total_day_calls                750 non-null    int64
 9   total_day_charge               750 non-null    float64
 10  total_eve_minutes              750 non-null    float64
 11  total_eve_calls                750 non-null    int64
 12  total_eve_charge               750 non-null    float64
 13  total_night_minutes            750 non-null    float64
 14  total_night_calls              750 non-null    int64
 15  total_night_charge             750 non-null    float64
 16  total_intl_minutes             750 non-null    float64
 17  total_intl_calls               750 non-null    int64
 18  total_intl_charge              750 non-null    float64
 19  number_customer_service_calls  750 non-null    int64
dtypes: float64(8), int64(8), object(4)
memory usage: 117.3+ KB
```

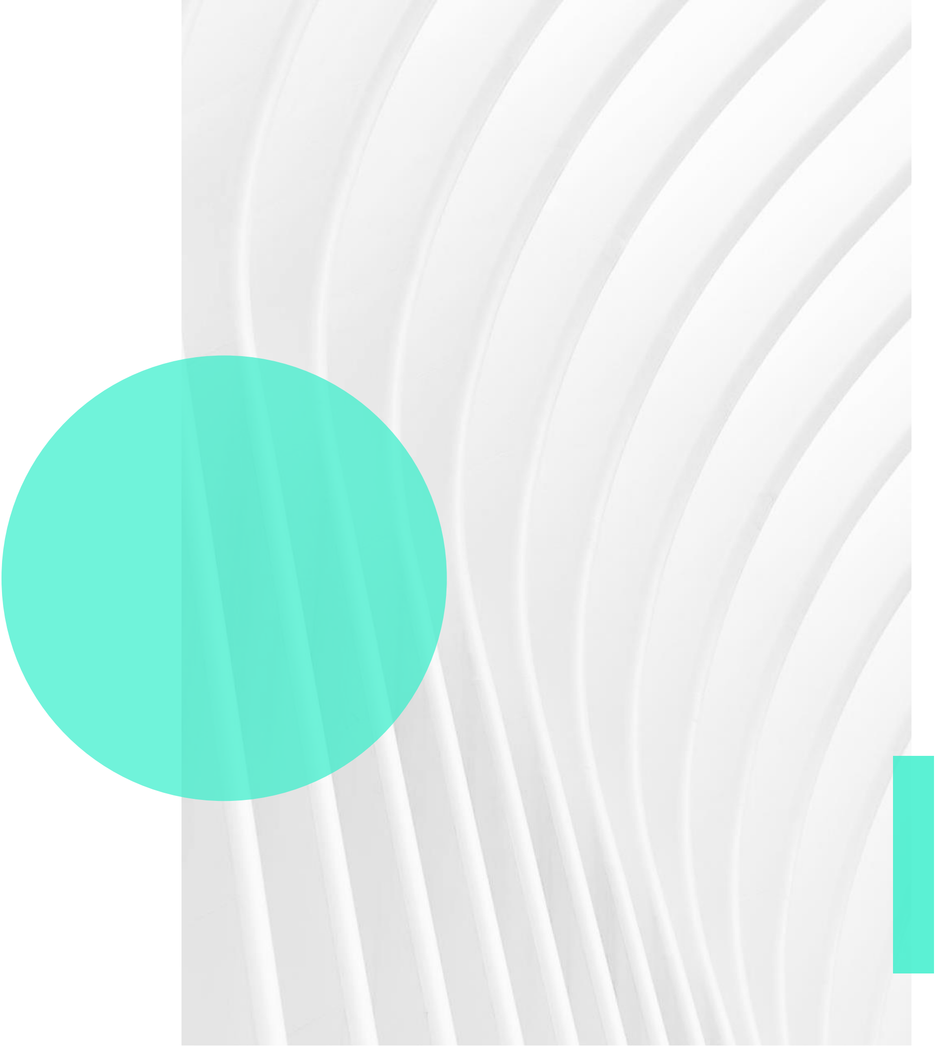
CHURN PREDICTION

```
predict = pd.Series(model_rf.predict(testing), name = 'churn').astype(int)
results = pd.concat([df_test_clean['id'], predict],axis = 1)
results['churn'].replace({0: 'no', 1: 'yes'}, inplace=True)
results.to_csv("predict application.csv", index = False)
results.head()
```

	id	churn
0	1	no
1	2	yes
2	3	yes
3	4	yes
4	5	yes

SUMMARY

Berdasarkan hasil machine learning yang telah diperoleh pada penggunaan data train, maka saat melakukan churn prediction, digunakan algortitma random forest classifier. Karena pada random forest classifier, diperoleh nilai akurasi, presisi, dan f1 yang tertinggi.



THANK YOU