train several classification algorithms to predict churn rate of telecommunication customers

- 1- logistic Regression
- 2- support vector machine
- 3- Knn
- 4- Random forest classifer

IMPORT LIBRARIES/DATASETS AND PERFORM EXPLORATORY DATA ANALYSIS

In [2]:

```
!pip install cufflinks
# Cufflinks is a third-party wrapper library around Plotly
```

```
Collecting cufflinks
  Downloading cufflinks-0.17.3.tar.gz (81 kB)
     |████████████████████████████████| 81 kB 179 kB/s eta 0:00:01
Requirement already satisfied: numpy>=1.9.2 in /home/rahma/anaconda3/lib/python3.8/site-packages (from cufflink
s) (1.20.1)
Requirement already satisfied: pandas>=0.19.2 in /home/rahma/.local/lib/python3.8/site-packages (from cufflink
s) (1.3.5)
Requirement already satisfied: plotly>=4.1.1 in /home/rahma/anaconda3/lib/python3.8/site-packages (from cufflin
ks) (5.6.0)
Requirement already satisfied: six>=1.9.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (from cufflinks)
(1.15.0)
Collecting colorlover>=0.2.1
  Downloading colorlover-0.3.0-py3-none-any.whl (8.9 kB)
Requirement already satisfied: setuptools>=34.4.1 in /home/rahma/anaconda3/lib/python3.8/site-packages (from cu
fflinks) (52.0.0.post20210125)
Requirement already satisfied: ipython>=5.3.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (from cuffli
nks) (7.22.0)
Requirement already satisfied: ipywidgets>=7.0.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (from cuf
flinks) (7.6.3)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /home/rahma/anaconda3/lib/python
3.8/site-packages (from ipython>=5.3.0->cufflinks) (3.0.17)
Requirement already satisfied: decorator in /home/rahma/.local/lib/python3.8/site-packages (from ipython>=5.3.0
->cufflinks) (4.3.0)
Requirement already satisfied: pygments in /home/rahma/anaconda3/lib/python3.8/site-packages (from ipython>=5.
3.0->cufflinks) (2.8.1)
Requirement already satisfied: backcall in /home/rahma/anaconda3/lib/python3.8/site-packages (from ipython>=5.
3.0->cufflinks) (0.2.0)
Requirement already satisfied: pexpect>4.3 in /home/rahma/anaconda3/lib/python3.8/site-packages (from ipython>=
5.3.0->cufflinks) (4.8.0)
Requirement already satisfied: jedi>=0.16 in /home/rahma/anaconda3/lib/python3.8/site-packages (from ipython>=
5.3.0->cufflinks) (0.17.2)
Requirement already satisfied: traitlets>=4.2 in /home/rahma/anaconda3/lib/python3.8/site-packages (from ipytho
n>=5.3.0->cufflinks) (5.0.5)
Requirement already satisfied: pickleshare in /home/rahma/anaconda3/lib/python3.8/site-packages (from ipython>=
5.3.0->cufflinks) (0.7.5)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /home/rahma/anaconda3/lib/python3.8/site-packages
(from ipywidgets>=7.0.0->cufflinks) (1.0.0)
Requirement already satisfied: nbformat>=4.2.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (from ipywi
dgets>=7.0.0->cufflinks) (5.1.3)
Requirement already satisfied: widgetsnbextension~=3.5.0 in /home/rahma/anaconda3/lib/python3.8/site-packages
(from ipywidgets>=7.0.0->cufflinks) (3.5.1)
Requirement already satisfied: ipykernel>=4.5.1 in /home/rahma/anaconda3/lib/python3.8/site-packages (from ipyw
idgets>=7.0.0->cufflinks) (5.3.4)
Requirement already satisfied: jupyter-client in /home/rahma/anaconda3/lib/python3.8/site-packages (from ipyker
nel>=4.5.1->ipywidgets>=7.0.0->cufflinks) (6.1.12)
Requirement already satisfied: tornado>=4.2 in /home/rahma/anaconda3/lib/python3.8/site-packages (from ipykerne
l>=4.5.1->ipywidgets>=7.0.0->cufflinks) (6.1)
Requirement already satisfied: parso<0.8.0,>=0.7.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (from j
edi>=0.16->ipython>=5.3.0->cufflinks) (0.7.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /home/rahma/anaconda3/lib/python3.8/site-packages (fr
om nbformat>=4.2.0->ipywidgets>=7.0.0->cufflinks) (3.2.0)
Requirement already satisfied: ipython-genutils in /home/rahma/anaconda3/lib/python3.8/site-packages (from nbfo
rmat>=4.2.0->ipywidgets>=7.0.0->cufflinks) (0.2.0)
Requirement already satisfied: jupyter-core in /home/rahma/anaconda3/lib/python3.8/site-packages (from nbformat
>=4.2.0->ipywidgets>=7.0.0->cufflinks) (4.7.1)
Requirement already satisfied: attrs>=17.4.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (from jsonsch
ema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidgets>=7.0.0->cufflinks) (20.3.0)
Requirement already satisfied: pyrsistent>=0.14.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (from js
onschema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidgets>=7.0.0->cufflinks) (0.17.3)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/rahma/anaconda3/lib/python3.8/site-packages (fro
m pandas>=0.19.2->cufflinks) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in /home/rahma/anaconda3/lib/python3.8/site-packages (from pandas>=
0.19.2->cufflinks) (2021.1)
Requirement already satisfied: ptyprocess>=0.5 in /home/rahma/anaconda3/lib/python3.8/site-packages (from pexpe
ct>4.3->ipython>=5.3.0->cufflinks) (0.7.0)
Requirement already satisfied: tenacity>=6.2.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (from plotl
y>=4.1.1->cufflinks) (8.0.1)
Requirement already satisfied: wcwidth in /home/rahma/anaconda3/lib/python3.8/site-packages (from prompt-toolki
t!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=5.3.0->cufflinks) (0.2.5)
Requirement already satisfied: notebook>=4.4.1 in /home/rahma/anaconda3/lib/python3.8/site-packages (from widge
tsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (6.3.0)
Requirement already satisfied: jinja2 in /home/rahma/anaconda3/lib/python3.8/site-packages (from notebook>=4.4.
1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (2.11.3)
Requirement already satisfied: terminado>=0.8.3 in /home/rahma/anaconda3/lib/python3.8/site-packages (from note
book>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.9.4)
Requirement already satisfied: argon2-cffi in /home/rahma/anaconda3/lib/python3.8/site-packages (from notebook>
=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (20.1.0)
Requirement already satisfied: nbconvert in /home/rahma/anaconda3/lib/python3.8/site-packages (from notebook>=
4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (6.0.7)
Requirement already satisfied: Send2Trash>=1.5.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (from not
ebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (1.5.0)
Requirement already satisfied: prometheus-client in /home/rahma/anaconda3/lib/python3.8/site-packages (from not
ebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.10.1)
Requirement already satisfied: pyzmq>=17 in /home/rahma/anaconda3/lib/python3.8/site-packages (from notebook>=
4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (20.0.0)
Requirement already satisfied: cffi>=1.0.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (from argon2-cf
fi->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (1.14.5)
Requirement already satisfied: pycparser in /home/rahma/anaconda3/lib/python3.8/site-packages (from cffi>=1.0.0
->argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (2.20)
Requirement already satisfied: MarkupSafe>=0.23 in /home/rahma/anaconda3/lib/python3.8/site-packages (from jinj
a2->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (1.1.1)
Requirement already satisfied: mistune<2,>=0.8.1 in /home/rahma/anaconda3/lib/python3.8/site-packages (from nbc
onvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.8.4)
Requirement already satisfied: defusedxml in /home/rahma/anaconda3/lib/python3.8/site-packages (from nbconvert-
>notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.7.1)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in /home/rahma/anaconda3/lib/python3.8/site-packages (fro
m nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.5.3)
Requirement already satisfied: jupyterlab-pygments in /home/rahma/anaconda3/lib/python3.8/site-packages (from n
bconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.1.2)
Requirement already satisfied: testpath in /home/rahma/anaconda3/lib/python3.8/site-packages (from nbconvert->n
otebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.4.4)
Requirement already satisfied: bleach in /home/rahma/anaconda3/lib/python3.8/site-packages (from nbconvert->not
ebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (3.3.0)
Requirement already satisfied: entrypoints>=0.2.2 in /home/rahma/anaconda3/lib/python3.8/site-packages (from nb
convert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.3)
Requirement already satisfied: pandocfilters>=1.4.1 in /home/rahma/anaconda3/lib/python3.8/site-packages (from
nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (1.4.3)
Requirement already satisfied: nest-asyncio in /home/rahma/anaconda3/lib/python3.8/site-packages (from nbclient
<0.6.0,>=0.5.0->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (1.5.1)
Requirement already satisfied: async-generator in /home/rahma/anaconda3/lib/python3.8/site-packages (from nbcli
ent<0.6.0,>=0.5.0->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (1.10)
Requirement already satisfied: webencodings in /home/rahma/anaconda3/lib/python3.8/site-packages (from bleach->
nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.5.1)
Requirement already satisfied: packaging in /home/rahma/anaconda3/lib/python3.8/site-packages (from bleach->nbc
onvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (20.9)
Requirement already satisfied: pyparsing>=2.0.2 in /home/rahma/anaconda3/lib/python3.8/site-packages (from pack
aging->bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (2.4.7)
Building wheels for collected packages: cufflinks
  Building wheel for cufflinks (setup.py) ... done
  Created wheel for cufflinks: filename=cufflinks-0.17.3-py3-none-any.whl size=67922 sha256=b095eb40d0e061a6f49
61cbd32bfba20099cebfa58057e4eceac6b8ea8c9e30b
  Stored in directory: /home/rahma/.cache/pip/wheels/6b/76/62/6da97734911ffcbdd559fd1a3f28526321f0ae699182a2386
6
Successfully built cufflinks
Installing collected packages: colorlover, cufflinks
Successfully installed colorlover-0.3.0 cufflinks-0.17.3
```

In [3]:

```
import numpy as np # Multi-dimensional array object
import pandas as pd # Data Manipulation
import matplotlib.pyplot as plt # Data Visualization
import seaborn as sns # Data Visualization
import plotly.express as px # Interactive Data Visualization
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot # Offline version of the Plotly m
import cufflinks as cf # Works as a connector between the pandas library and plotly
cf.go_offline()
init_notebook_mode(connected=True) # To connect Jupyter notebook with JavaScript
```

In [4]:

```
# Read the CSV file
df = pd.read_csv('telecom_churn.csv')
```

In [5]:

```
# Load the top 5 instances
df.head()
```

Out[5]:

	state	account_length	area_code	phone_number	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_d
0	16	128	415	2845	0	1	25	265.1	
1	35	107	415	2301	0	1	26	161.6	
2	31	137	415	1616	0	0	0	243.4	
3	35	84	408	2510	1	0	0	299.4	
4	36	75	415	155	1	0	0	166.7	

5 rows × 21 columns

In [6]:

```
# Load the bottom 5 instances
df.tail()
```

Out[6]:

	state	account_length	area_code	phone_number	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	tota
4995	11	50	408	2000	0	1	40	235.7	
4996	49	152	415	394	0	0	0	184.2	
4997	7	61	415	313	0	0	0	140.6	
4998	7	109	510	3471	0	0	0	188.8	
4999	46	86	415	2412	0	1	34	129.4	

5 rows × 21 columns

In [7]:

```
# Check the shape of the dataframe
df.shape
```

Out[7]: (5000, 21)

In [8]:

```
# Display the feature columns
df.columns
```

Out[8]:

```
Index(['state', 'account_length', 'area_code', 'phone_number',
       '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',
       'number_customer_service_calls', 'class'],
      dtype='object')
```

In [9]:

```
# Obtain the summary of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 21 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   state                          5000 non-null   int64  
 1   account_length                 5000 non-null   int64  
 2   area_code                      5000 non-null   int64  
 3   phone_number                   5000 non-null   int64  
 4   international_plan             5000 non-null   int64  
 5   voice_mail_plan                5000 non-null   int64  
 6   number_vmail_messages          5000 non-null   int64  
 7   total_day_minutes              5000 non-null   float64
 8   total_day_calls                5000 non-null   int64  
 9   total_day_charge               5000 non-null   float64
 10  total_eve_minutes              5000 non-null   float64
 11  total_eve_calls                5000 non-null   int64  
 12  total_eve_charge               5000 non-null   float64
 13  total_night_minutes            5000 non-null   float64
 14  total_night_calls              5000 non-null   int64  
 15  total_night_charge             5000 non-null   float64
 16  total_intl_minutes             5000 non-null   float64
 17  total_intl_calls               5000 non-null   int64  
 18  total_intl_charge              5000 non-null   float64
 19  number_customer_service_calls  5000 non-null   int64  
 20  class                          5000 non-null   int64  
dtypes: float64(8), int64(13)
memory usage: 820.4 KB
```

- What is the maximum and average daily minutes?

In [11]:

```
df.agg({'total_day_minutes':['max', 'mean']})
```

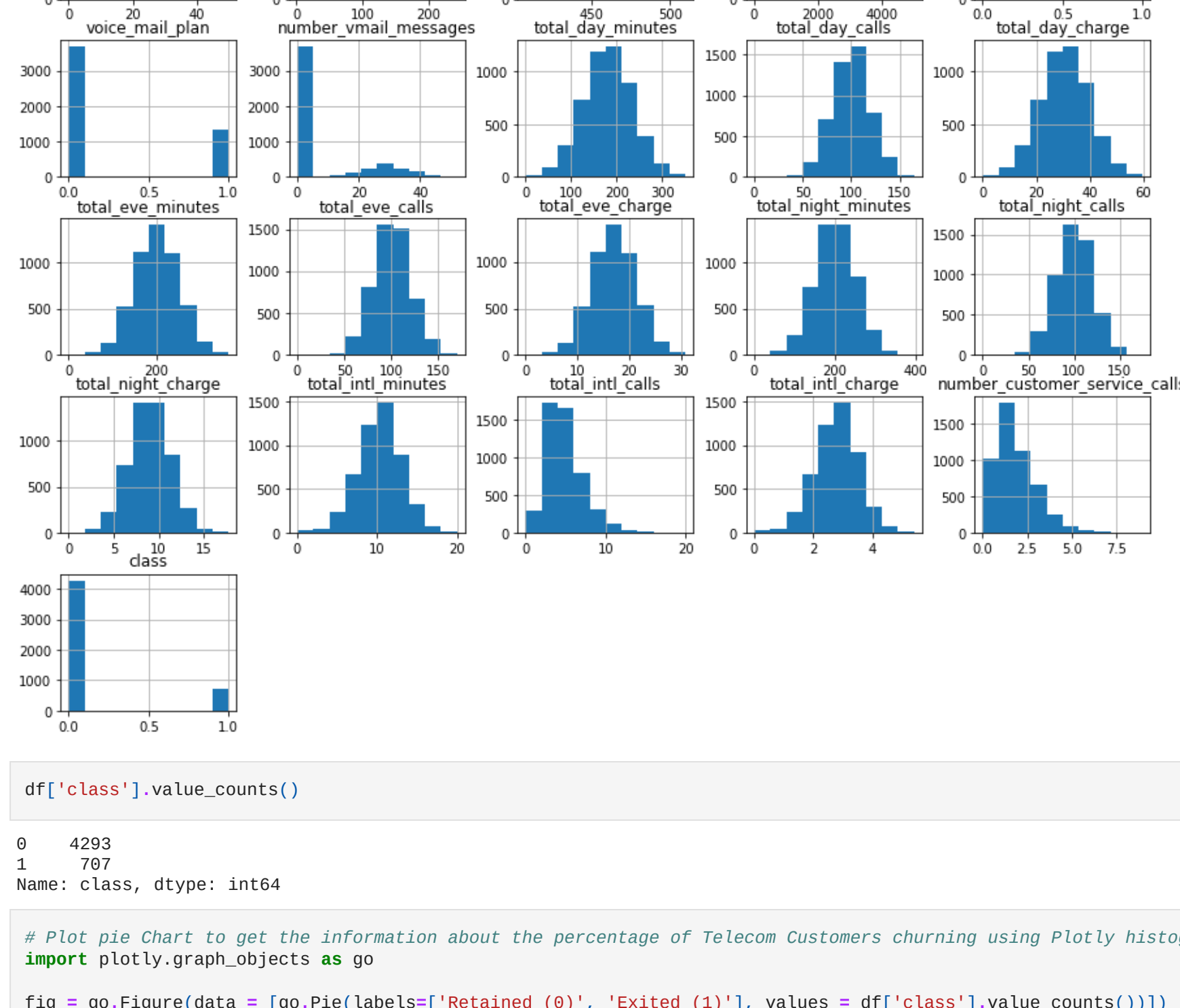
Out[11]:

	total_day_minutes
max	351.5000
mean	180.2889

PERFORM DATA VISUALIZATION

In [12]:

```
df.hist(figsize=(15,12));
```



In [13]:

```
df['class'].value_counts()
```

Out[13]:

```
0    4293
1     707
Name: class, dtype: int64
```

In [14]:

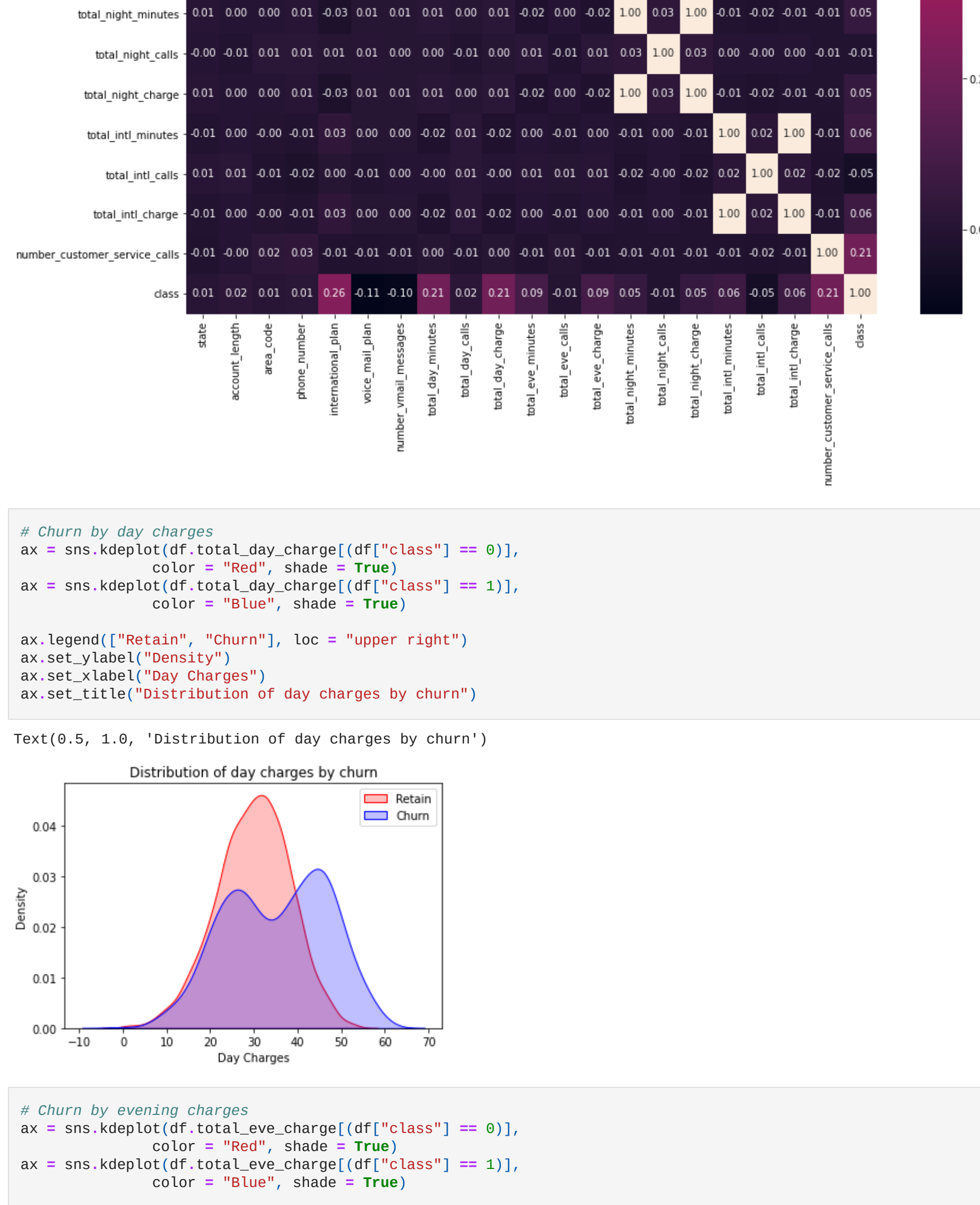
```
# Plot pie Chart to get the information about the percentage of Telecom Customers churning using Plotly histogr
import plotly.graph_objects as go

fig = go.Figure(data = [go.Pie(labels=['Retained (0)', 'Exited (1)'], values = df['class'].value_counts())])
fig.show()
```

In [15]:

```
# Plot histogram graph for the international plan service used by the Telecom customers with respect to churned
fig = px.histogram(df, x='international_plan', color='class')
fig.show()
```

In [16]:

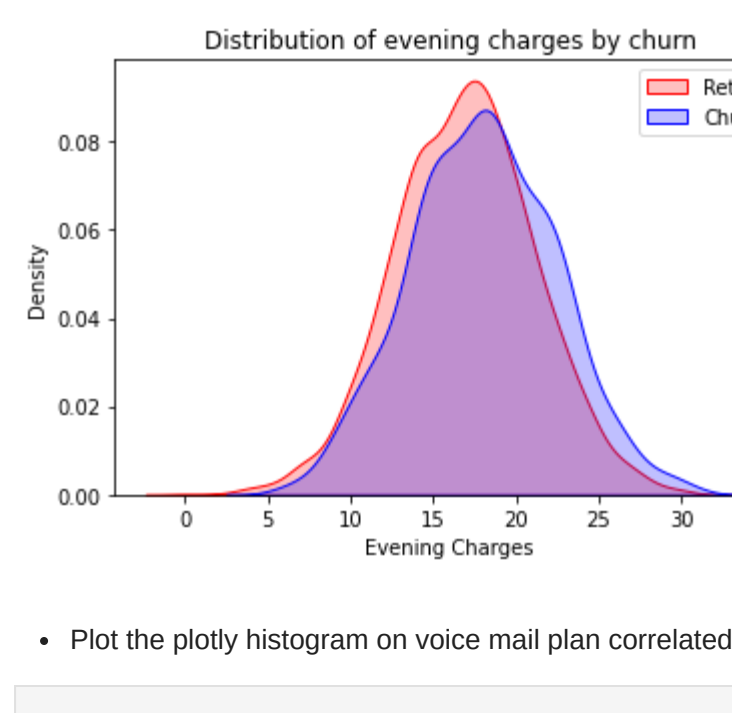
```
# Correlation Matrix

corr_matrix = df.corr()
plt.figure(figsize = (15, 15))
sns.heatmap(corr_matrix, annot=True, fmt='0.2f')
plt.title("Correlation Matrix of Telecom Customers", fontsize = 20)
plt.show()

# It is clearly shown that "voice_mail_plan" and "number_vmail_messages" are highly correlated.
# It is clearly shown that "total day charge" and "total daily minutes" are highly correlated.
```



In [17]:

```
# Churn by day charges
ax = sns.kdeplot(df.total_day_charge[(df["class"] == 0)],
                color = "Red", shade = True)
ax = sns.kdeplot(df.total_day_charge[(df["class"] == 1)],
                color = "Blue", shade = True)

ax.legend(["Retain", "Churn"], loc = "upper right")
ax.set_ylabel("Density")
ax.set_xlabel("Day Charges")
ax.set_title("Distribution of day charges by churn")
```

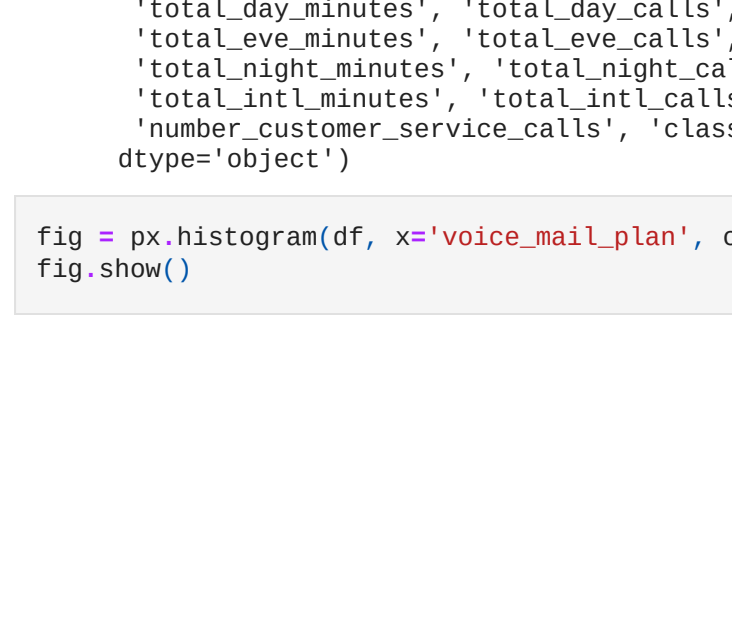
Out[17]: Text(0.5, 1.0, 'Distribution of day charges by churn')



In [18]:

```
# Churn by evening charges
ax = sns.kdeplot(df.total_eve_charge[(df["class"] == 0)],
                color = "Red", shade = True)
ax = sns.kdeplot(df.total_eve_charge[(df["class"] == 1)],
                color = "Blue", shade = True)

ax.legend(["Retain", "Churn"], loc = "upper right")
ax.set_ylabel("Density")
ax.set_xlabel("Evening Charges")
ax.set_title("Distribution of evening charges by churn")
```

Out[18]: Text(0.5, 1.0, 'Distribution of evening charges by churn')



- Plot the plotly histogram on voice mail plan correlated with Churn feature

In [19]:

```
df.columns
```

Out[19]:

```
Index(['state', 'account_length', 'area_code', 'phone_number',
       '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',
       'number_customer_service_calls', 'class'],
      dtype='object')
```

In [20]:

```
fig = px.histogram(df, x='voice_mail_plan', color='class')
fig.show()
```

IDENTIFY FEATURE IMPORTANCE & PREPARE THE DATA BEFORE MODEL TRAINING

In [21]:

```
# Unnecessary features would decrease the training speed, the model interpretability and the generalization per
# Therefore, finding and selecting the most useful features in the dataset is crucial.
# Assigning input features to X and output (Churn) to y

X = df.drop(["class", "area_code", "phone_number"], axis = "columns") # area_code and phone_number features are
y = df["class"]
```

In [22]:

```
X.shape
```

Out[22]: (5000, 18)

In [23]:

```
y.shape
```

Out[23]: (5000,)

In [24]:

```
# Perform train/test split
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=42)
```

In [25]:

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(4000, 18)
(4000,)
(1000, 18)
(1000,)
```

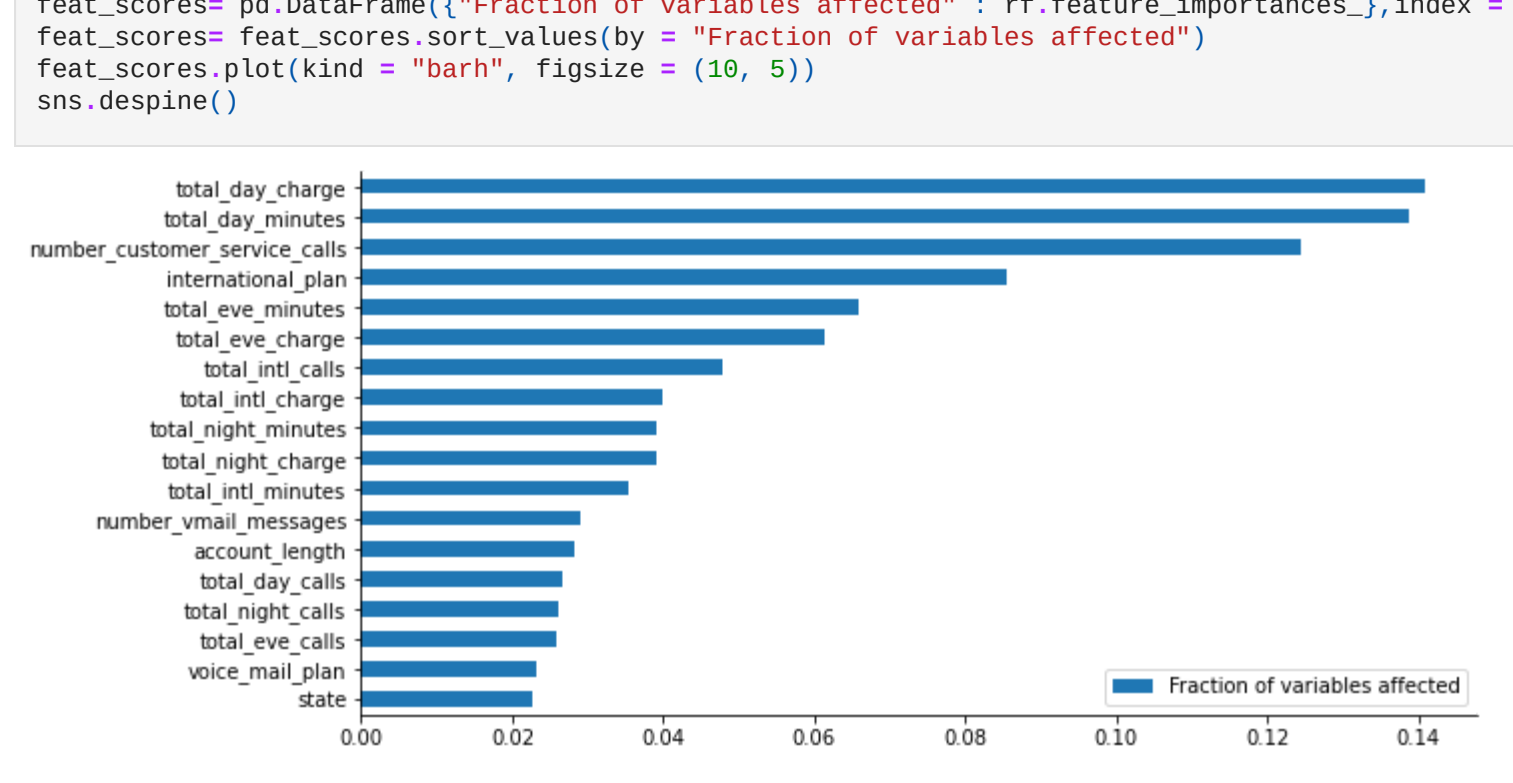
In [26]:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train.values.ravel())
```

Out[26]: RandomForestClassifier()

In [27]:

```
# Plot the feature importance

feat_scores= pd.DataFrame({"Fraction of variables affected" : rf.feature_importances_},index = X.columns)
feat_scores= feat_scores.sort_values(by = "Fraction of variables affected")
feat_scores.plot(kind = "barh", figsize = (10, 5))
sns.despine()
```



In [28]:

```
# The above graph is generated by Random Forest algorithm
# The graph indicates that "total_day_minutes" tops the list of important features followed by "total_day_minut
```

TRAIN AND EVALUATE A LOGISTIC REGRESSION CLASSIFIER

In [29]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

model_LR = LogisticRegression(max_iter=2000, class_weight='balanced')
model_LR.fit(X_train, y_train)
```

Out[29]: LogisticRegression(class_weight='balanced', max_iter=2000)

In [30]:

```
y_pred = model_LR.predict(X_test)
```

In [31]:

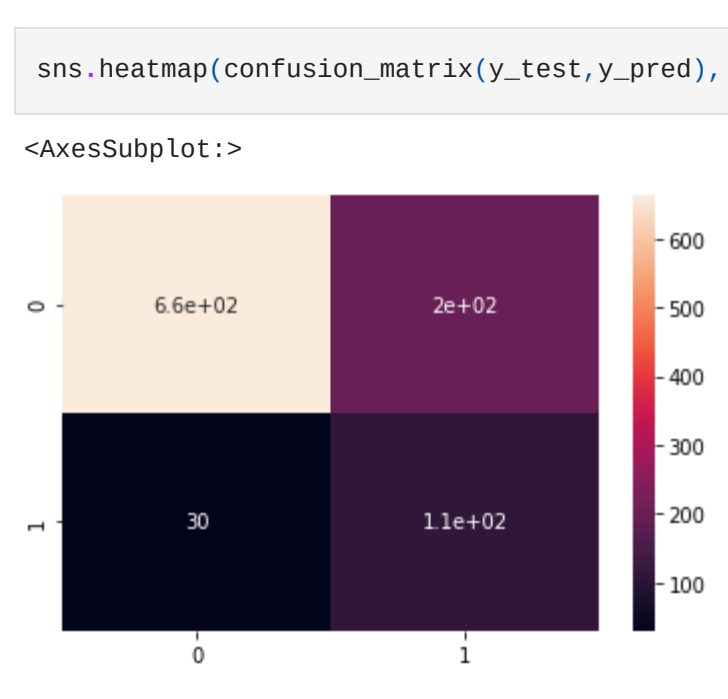
```
# precision is the ratio of TP/(TP+FP)
# recall is the ratio of TP/(TP+FN)
# F-beta score can be interpreted as a weighted harmonic mean of the precision and recall
# where an F-beta score reaches its best value at 1 and worst score at 0.
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.77      0.85       861
           1       0.35      0.78      0.49       139

    accuracy                           0.77      1000
   macro avg       0.66      0.78      0.67      1000
weighted avg       0.87      0.77      0.80      1000
```

In [35]:

```
sns.heatmap(confusion_matrix(y_test,y_pred), annot=True)
```

Out[35]: <AxesSubplot:>



TRAIN AND EVALUATE A SUPPORT VECTOR MACHINE CLASSIFIER

In [38]:

```
from sklearn.calibration import CalibratedClassifierCV # For probability score output
from sklearn.svm import LinearSVC

model_svm = LinearSVC(max_iter=10000, class_weight='balanced')
model_svm = CalibratedClassifierCV(model_svm)
model_svm.fit(X_train, y_train)
```

```
/home/rahma/anaconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:985: ConvergenceWarning:

Liblinear failed to converge, increase the number of iterations.

/home/rahma/anaconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:985: ConvergenceWarning:

Liblinear failed to converge, increase the number of iterations.

/home/rahma/anaconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:985: ConvergenceWarning:

Liblinear failed to converge, increase the number of iterations.

/home/rahma/anaconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:985: ConvergenceWarning:

Liblinear failed to converge, increase the number of iterations.

/home/rahma/anaconda3/lib/python3.8/site-packages/sklearn/svm/_base.py:985: ConvergenceWarning:

Liblinear failed to converge, increase the number of iterations.
```

Out[38]:

```
CalibratedClassifierCV(base_estimator=LinearSVC(class_weight='balanced',
                                                max_iter=10000))
```

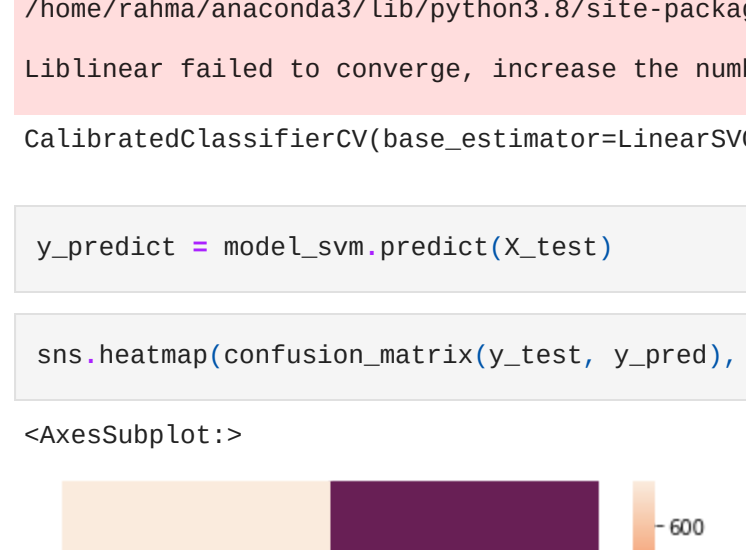
In [39]:

```
y_predict = model_svm.predict(X_test)
```

In [40]:

```
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True)
```

Out[40]: <AxesSubplot:>



In [41]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.77      0.85       861
           1       0.35      0.78      0.49       139

    accuracy                           0.77      1000
   macro avg       0.66      0.78      0.67      1000
weighted avg       0.87      0.77      0.80      1000
```

TRAIN AND EVALUATE A RANDOM FOREST CLASSIFIER

In [42]:

```
from sklearn.ensemble import RandomForestClassifier

model_rf = RandomForestClassifier()
model_rf.fit(X_train, y_train)
```

Out[42]: RandomForestClassifier()

In [43]:

```
y_pred = model_rf.predict(X_test)
```

In [44]:

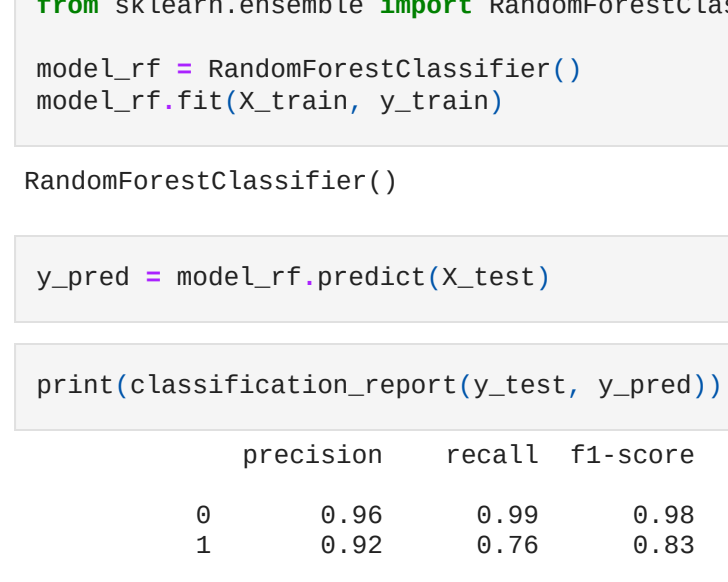
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       861
           1       0.92      0.76      0.83       139

    accuracy                           0.96      1000
   macro avg       0.94      0.88      0.91      1000
weighted avg       0.96      0.96      0.96      1000
```

In [45]:

```
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True)
```

Out[45]: <AxesSubplot:>



TRAIN AND EVALUATE A K-NEAREST NEIGHBOUR (KNN)

In [46]:

```
from sklearn.neighbors import KNeighborsClassifier

model_knn= KNeighborsClassifier()
model_knn.fit(X_train, y_train)
```

Out[46]: KNeighborsClassifier()

In [47]:

```
y_pred= model_knn.predict(X_test)
```

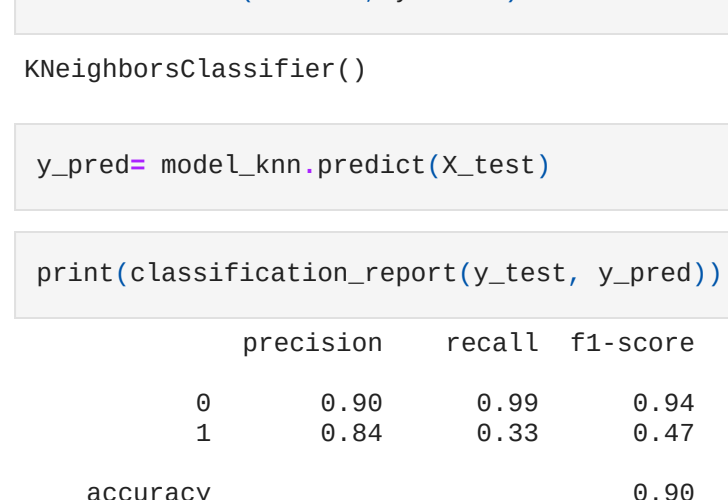
In [48]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       861
           1       0.84      0.33      0.47       139

    accuracy                           0.90      1000
   macro avg       0.87      0.66      0.71      1000
weighted avg       0.89      0.90      0.88      1000
```

In [50]:

```
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True);
```



TRAIN AND EVALUATE A NAIVE BAYES CLASSIFIER

In [51]:

```
from sklearn.naive_bayes import GaussianNB
```

In [52]:

```
model_NB = GaussianNB()
model_NB.fit(X_train, y_train)
```

Out[52]: GaussianNB()

In [53]:

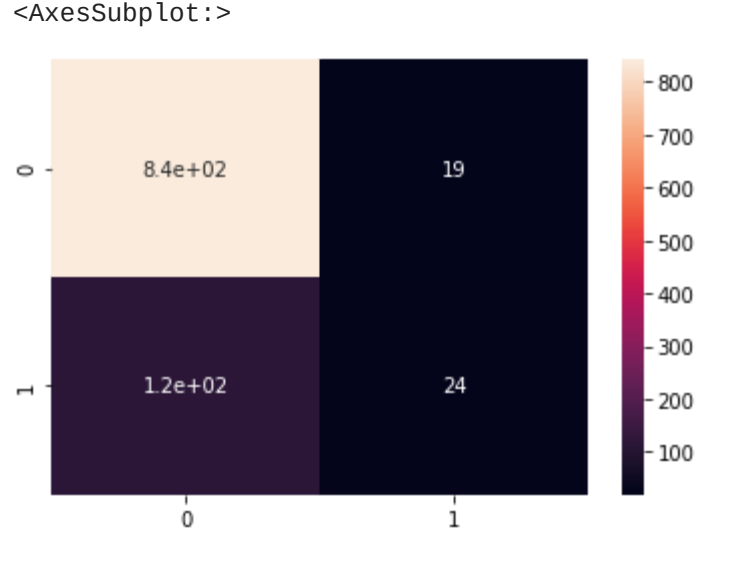
```
y_pred = model_NB.predict(X_test)
```

In [54]:
```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.88      0.98      0.93       861
           1       0.56      0.17      0.26       139

    accuracy                           0.87      1000
   macro avg       0.72      0.58      0.60      1000
weighted avg       0.84      0.87      0.83      1000
```

In [55]:
```
cm = confusion_matrix(y_test, y_predict)
sns.heatmap(cm, annot = True)
```

Out[55]: <AxesSubplot:>



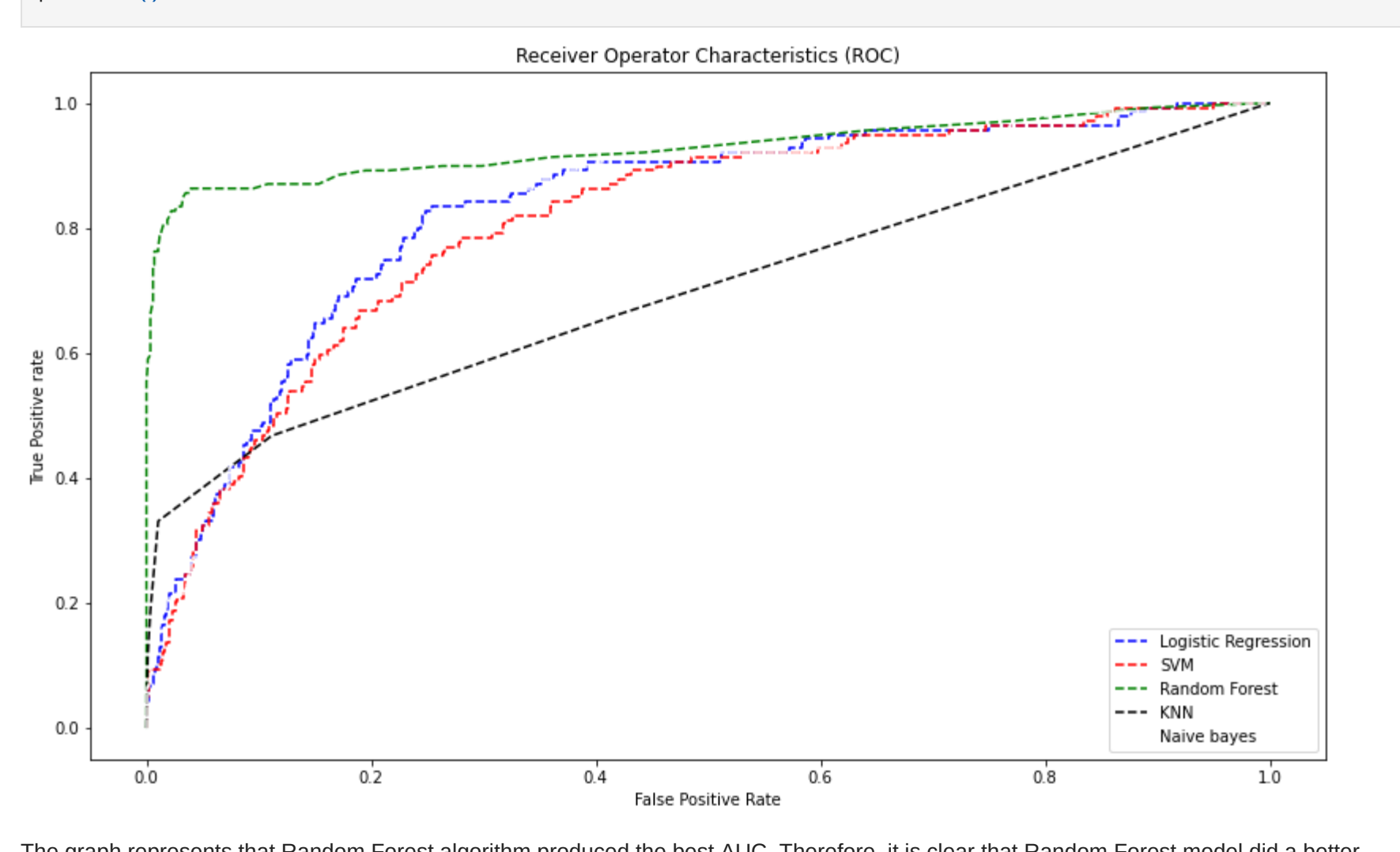
PLOT ROC CURVES FOR THE 5 MODELS AND FIND AUC SCORES

In [65]:
```
# ROC curve
from sklearn.metrics import roc_curve

fpr1, tpr1, thresh1 = roc_curve(y_test, model_LR.predict_proba(X_test)[:, 1], pos_label = 1)
fpr2, tpr2, thresh2 = roc_curve(y_test, model_svm.predict_proba(X_test)[:, 1], pos_label = 1)
fpr3, tpr3, thresh3 = roc_curve(y_test, model_rf.predict_proba(X_test)[:, 1], pos_label = 1)
fpr4, tpr4, thresh4 = roc_curve(y_test, model_knn.predict_proba(X_test)[:, 1], pos_label = 1)
fpr5, tpr5, thresh5 = roc_curve(y_test, model_NB.predict_proba(X_test)[:, 1], pos_label = 1)
```

In [66]:
```
# AUC score

from sklearn.metrics import roc_auc_score

auc_score1 = roc_auc_score(y_test, model_LR.predict_proba(X_test)[:, 1])
auc_score2 = roc_auc_score(y_test, model_svm.predict_proba(X_test)[:, 1])
auc_score3 = roc_auc_score(y_test, model_rf.predict_proba(X_test)[:, 1])
auc_score4 = roc_auc_score(y_test, model_knn.predict_proba(X_test)[:, 1])
auc_score5 = roc_auc_score(y_test, model_NB.predict_proba(X_test)[:, 1])

print("Logistic Regression: ", auc_score1) # Logistic Regression
print("Support Vector Machine: ", auc_score2) # Support Vector Machine
print("Random Forest: ", auc_score3) # Random Forest
print("K-Nearest Neighbors: ", auc_score4) # K-Nearest Neighbors
print("Naive Bayes: ", auc_score5) # Naive Bayes
```

```
Logistic Regression:  0.8306804034124616
Support Vector Machine:  0.811144812373098
Random Forest:  0.9287343644248365
K-Nearest Neighbors:  0.6986188053041886
Naive Bayes:  0.8481270732542885
```

In [70]:
```
plt.figure(figsize=(14,8))
plt.plot(fpr1, tpr1, linestyle = "--", color = "Blue", label = "Logistic Regression")
plt.plot(fpr2, tpr2, linestyle = "--", color = "red", label = "SVM")
plt.plot(fpr3, tpr3, linestyle = "--", color = "green", label = "Random Forest")
plt.plot(fpr4, tpr4, linestyle = "--", color = "black", label = "KNN")
plt.plot(fpr5, tpr5, linestyle = "--", color = "white", label = "Naive bayes")

plt.title('Receiver Operator Characteristics (ROC)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')

plt.legend(loc = 'best')
plt.savefig('ROC', dpi = 300)
plt.show()
```



The graph represents that Random Forest algorithm produced the best AUC. Therefore, it is clear that Random Forest model did a better job of classifying the churned/retained telecom customers.

TASK #11: CONCLUSION & PROJECT RECAP

In [71]:
```
y_predict = model_rf.predict(X_test)
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       861
           1       0.92      0.76      0.83       139

    accuracy                           0.96      1000
   macro avg       0.94      0.88      0.91      1000
weighted avg       0.96      0.96      0.96      1000
```

Amongst all the trained models, Random Forest Classifier algorithm produced the highest Area under the ROC curve (AUC).

The following scores are the results of the Random Forest Classifier model

1. Accuracy: ~96% label accuracy
2. Precision: ~96% labeled as Retained customers and ~92% labeled as churned customers
3. Recall: ~99% labeled as Retained customers and ~76% labeled as churned customers

- We can improve this model even more better by using "Grid Search" method.