```
## Comcast Telecom Complaints Dataset
```

Context

Estimation of Comcast Customer Top Complaints.

Acknowledgements

Kaggle Datasets

```
import warnings
warnings.filterwarnings(action='ignore')

import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go

import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
```

```
import numpy as np
import pandas as pd
```

```
###!mkdir ~/.kaggle
```

```
###!cp /kaggle.json ~/.kaggle/
```

Saved successfully!

```
###!pip install keras-tuner
```

```
###!pip install kaggle
```

```
####! kaggle datasets download -d yasserh/comcast-telecom-complaints
```

```
###! unzip /content/comcast-telecom-complaints.zip
```

```
comcast = pd.read_csv("/content/Comcast.csv")
```

```
training_data = comcast.sample(frac=0.7, random_state=25)
testing_data = comcast.drop(training_data.index)
```

```
print(training_data.shape, testing_data.shape)
```

```
(1557, 11) (667, 11)
```

```
training_data.to_csv("train.csv")
testing_data.to_csv("test.csv")
```

```
training_data.columns
```

```
Index(['Ticket #', 'Customer Complaint', 'Date', 'Date_month_year', 'Time',
       'Received Via', 'City', 'State', 'Zip code', 'Status',
       'Filing on Behalf of Someone'],
      dtype='object')
```

```
training_data.Status.value_counts()
```

```
Solved     658
Closed     521
Open       266
Pending    112
Name: Status, dtype: int64
```

```
####! pip install unidecode
```

```
####! pip install nltk
```

```
import re, unidecode
from bs4 import BeautifulSoup
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Needed only once
# import nltk
# nltk.download('stopwords')
# nltk.download('punkt')
# nltk.download('wordnet')

def remove_html_tags(text):
    soup = BeautifulSoup(text, "html.parser")
    stripped_text = soup.get_text(separator=" ")
    return stripped_text
def remove_accented_chars(text):
    text = unidecode.unidecode(text)
```

Saved successfully!

```
    result = re.sub(r'\d+', '', text)
    return result
def remove_slash_with_space(text):
    return text.replace('\\', " ")
def remove_punctuation(text):
    translator = str.maketrans('', '', string.punctuation)
    return text.translate(translator)
def text_lowercase(text):
    return text.lower()
def remove_whitespace(text):
    return  " ".join(text.split())
def remove_stopwords(text):
    stop_words = set(stopwords.words("english"))
    word_tokens = word_tokenize(text)
    filtered_text = [word for word in word_tokens if word not in stop_words]
    return ' '.join(filtered_text)
def stem_words(text):
    stemmer = PorterStemmer()
    word_tokens = word_tokenize(text)
    stems = [stemmer.stem(word) for word in word_tokens]
    return ' '.join(stems)
def lemmatize_words(text):
    lemmatizer = WordNetLemmatizer()
    word_tokens = word_tokenize(text)
    # provide context i.e. part-of-speech
    lemmas = [lemmatizer.lemmatize(word, pos ='v') for word in word_tokens]
    return ' '.join(lemmas)
```

```
# Perform preprocessing
def perform_preprocessing(text):
    text = remove_html_tags(text)
    text = remove_accented_chars(text)
    text = remove_numbers(text)
    text = remove_stopwords(text)
    text = text_lowercase(text)
    text = remove_slash_with_space(text)
```

```
#    text = remove_punctuation(text)
    text = stem_words(text)
    text = lemmatize_words(text)
    text = remove_whitespace(text)
    return text
```

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
True
```

```
training_data.columns
```

```
Index(['Ticket #', 'Customer Complaint', 'Date', 'Date_month_year', 'Time',
       'Received Via', 'City', 'State', 'Zip code', 'Status',
       'Filing on Behalf of Someone'],
      dtype='object')
```

```
training_data['Customer_Complaint_corpus'] = training_data['Customer Complaint'].apply(perform_preprocessing)
```

Saved successfully!

```
rpus'] = testing_data['Customer Complaint'].apply(perform_preprocessing)
```

```
####! pip install pycaret==2.3.4
```

```
###! pip install jinja2
```

```
###! pip install markupsafe==2.0.1
```

```
###! pip install evalml
```

```
from pycaret.classification import *
```

```
exp_mclf101 = setup(data = training_data, target = 'Status', session_id=123)
```

	Description	Value
0	Session id	123
1	Target	Status
2	Target type	Multiclass
3	Target mapping	Closed: 0, Open: 1, Pending: 2, Solved: 3
4	Original data shape	(1557, 12)
5	Transformed data shape	(1557, 12)
6	Transformed train set shape	(1089, 12)
7	Transformed test set shape	(468, 12)
8	Ordinal features	2
9	Numeric features	1
10	Categorical features	10
11	Preprocess	True
12	Imputation type	simple

```
compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
nb	Naive Bayes	0.6685	0.7877	0.6685	0.5478	0.5850	0.4502	0.5157	0.3430
ridge	Ridge Classifier	0.6538	0.0000	0.6538	0.5648	0.5770	0.4310	0.4779	0.2820
	ier	0.6262	0.7892	0.6262	0.5918	0.5988	0.4173	0.4293	0.8870
	g	0.6235	0.7892	0.6235	0.6091	0.6087	0.4259	0.4318	0.5870
rf	Random Forest Classifier	0.6226	0.7911	0.6226	0.6006	0.6031	0.4175	0.4239	0.6490
catboost	CatBoost Classifier	0.6180	0.7838	0.6180	0.5971	0.6004	0.4143	0.4212	0.3600
et	Extra Trees Classifier	0.6143	0.7870	0.6143	0.6041	0.6036	0.4125	0.4158	0.5160
lightgbm	Light Gradient Boosting Machine	0.6125	0.7904	0.6125	0.6035	0.6039	0.4142	0.4174	0.5380
lda	Linear Discriminant Analysis	0.5758	0.7762	0.5758	0.5966	0.5737	0.3766	0.3888	0.3530
dt	Decision Tree Classifier	0.5583	0.6827	0.5583	0.5663	0.5577	0.3504	0.3531	0.6330
ada	Ada Boost Classifier	0.5226	0.5438	0.5226	0.5075	0.5013	0.2767	0.2826	0.4160
lr	Logistic Regression	0.4224	0.4794	0.4224	0.1784	0.2509	0.0000	0.0000	1.7460
dummy	Dummy Classifier	0.4224	0.5000	0.4224	0.1784	0.2509	0.0000	0.0000	0.6970
knn	K Neighbors Classifier	0.3554	0.5264	0.3554	0.3465	0.3472	0.0238	0.0240	0.5850
svm	SVM - Linear Kernel	0.2542	0.0000	0.2542	0.0806	0.1190	0.0000	0.0000	0.3010
qda	Quadratic Discriminant Analysis	0.2432	0.4573	0.2432	0.1532	0.1308	0.0160	0.0350	0.4560

Saved successfully!

```
▾ GaussianNB
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
nb2 = create_model('nb')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6422	0.7610	0.6422	0.5282	0.5590	0.4060	0.4757
1	0.6789	0.7976	0.6789	0.5288	0.5881	0.4728	0.5227
2	0.6606	0.7963	0.6606	0.5433	0.5773	0.4373	0.5075
3	0.6606	0.7784	0.6606	0.5214	0.5756	0.4440	0.4887
4	0.7156	0.8421	0.7156	0.5846	0.6286	0.5283	0.5972
5	0.7064	0.8117	0.7064	0.5883	0.6226	0.5119	0.5878
6	0.6422	0.7483	0.6422	0.5552	0.5638	0.4014	0.4848

```
tune_model(nb2)
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6422	0.7600	0.6422	0.5282	0.5590	0.4060	0.4757
1	0.6881	0.7872	0.6881	0.5487	0.5995	0.4858	0.5466
2	0.6606	0.7972	0.6606	0.5433	0.5773	0.4373	0.5075
3	0.6789	0.7762	0.6789	0.5401	0.5889	0.4699	0.5324
4	0.7156	0.8339	0.7156	0.5846	0.6286	0.5283	0.5972
5	0.7064	0.8103	0.7064	0.5883	0.6226	0.5119	0.5878
[illegible]	[illegible]	[illegible]	[illegible]	0.5706	0.5724	0.4160	0.5087
[illegible]	[illegible]	[illegible]	[illegible]	0.5610	0.5958	0.4661	0.5353
8	0.6606	0.7600	0.6606	0.5519	0.5808	0.4349	0.5045
9	0.6574	0.7816	0.6574	0.5476	0.5768	0.4280	0.4988
Mean	0.6740	0.7851	0.6740	0.5564	0.5902	0.4584	0.5295
Std	0.0228	0.0240	0.0228	0.0185	0.0209	0.0389	0.0370

Saved successfully!

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
▾                    GaussianNB
GaussianNB(priors=None, var_smoothing=2e-09)
```

```
prediction = predict_model(nb2, data = testing_data)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Naive Bayes	0.7031	0	0.7031	0.5859	0.6253	0.4783	0.5378

```
prediction.head(2)
```

	Ticket #	Customer Complaint	Date	Date_month_year	Time	Received Via	City	State	Zip code	Filing on Behalf of	Customer_Complaint_corpus	Status

```
prediction.reset_index(inplace=True)
```

```
prediction.head(3)
```

	index	Ticket #	Customer Complaint	Date	Date_month_year	Time	Received Via	City	State	Zip code	Filing on Behalf of Someone	Customer_Complaint_corpus
0	2	242732	Speed and Service	18-04-15	18-Apr-15	9:55:47 AM	Internet	Acworth	Georgia	30101	Yes	speed servic
1	4	307175	Comcast not working and no service to boot	26-05-15	26-May-15	1:25:26 PM	Internet	Acworth	Georgia	30101	No	comcast work servic boot
2	5	338519	ISP Charging for arbitrary data limits with ov...	06-12-15	06-Dec-15	9:59:40 PM	Internet	Acworth	Georgia	30101	No	isp charg arbitrari data limit overag fee

Saved successfully!

```
prediction["prediction_label"].value_counts()
```

```
Solved    487
Closed    180
Name: prediction_label, dtype: int64
```

✓ 1m 6s completed at 8:58 PM