AN INTELLIGENT CUSTOMER SUPPORT AND FEEDBACK SYSTEM WITH SENTIMENTAL ANALYSIS

OBIE UDEMEZUE #100764160

INTRODUCTION

- Our projects goal is to reduce process time within the business and increase the customer retention rate for an electronics online retailer.
- Highest satisfaction rate to ensure the potential for repeat customers.
- Quick solutions or help outside of the standard working day hours, getting questions answered 24/7.
- Presenting sentimental analysis data and fitting it into the right algorithm I can predict a potential increase in satisfied customers.
- Online service would create a more personalized experience, increase customer loyalty and lower a companies' customer support costs.

RATIONALE

STATEMENT

E-COMMERCE INDUSTRY IS A FAST PACED, HIGHLY COMPETITIVE AND CONTINUOUSLY GROWING SPACE REQUIRING BUSINESSES OPERATING WITHIN THE INDUSTRY TO MAINTAIN A HIGH CUSTOMER ENGAGEMENT AND SATISFACTION LEVEL.

COMPANIES ARE FACED MAJOR PROBLEM THEY ARE FACED WITH, WHICH IS KEEPING UP WITH CUSTOMER ENGAGEMENT ESPECIALLY IN THE AREA OF PROVIDING CUSTOMER SUPPORT, ENQUIRY TICKETS RESOLUTION AND FEEDBACK

THE GOAL OF THIS PROJECT IS TO DEVELOP AN INTELLIGENT CUSTOMER SUPPORT AND FEEDBACK SYSTEM TO HELP THE SELECTED COMPANY IMPROVE ITS CURRENT BUSINESS PROCESSES.

PROBLEM STATEMENT

- TRADITIONAL FEEDBACK SURVEYS OVER EMAIL HAVE A 10% OPEN RATE AND LESS THAN 5% OF COMPLETION RATE WHICH LEAVES BEHIND BUSINESSES ON GETTING PROPER CUSTOMER FEEDBACK. IMPLEMENTING A CHATBOT WHICH CAN DIRECTLY COLLECT THE FEEDBACK FROM THE CUSTOMERS CAN HELP THE BUSINESS TO UNDERSTAND THE EFFECTIVENESS OF THE SERVICES THEY ARE OFFERING.

- INABILITY TO TAKE REAL TIME INSTANT ACTION – IF THE COMPANIES TAKE A LONG TIME TO RESPOND, THERE IS A CHANCE THAT THE CUSTOMER MAY LEAVE AND NOT RETURN. ON AVERAGE CHATBOTS TAKE APPROXIMATELY 2 MINUTES TO ANSWER CUSTOMER QUERIES AND CAN HELP REDUCE CUSTOMER SERVICE REPRESENTATIVES' INTERVENTION.

- LACK OF UNDERSTANDING BEST TIME TO REACH OUT TO CUSTOMER – TRADITIONAL FEEDBACK COLLECTION SYSTEMS DON'T COLLECT ENOUGH INFORMATION ON THE BEST TIME TO REACH OUT TO CUSTOMERS. IN OUR PROJECT, I WILL USE MACHINE LEARNING TECHNIQUES TO UNDERSTAND USER BEHAVIOR AND FIND THE BEST TIME AND CORRECT MEDIUM TO REACH OUT TO THE CUSTOMERS.

- LACK OF SENTIMENTAL ANALYSIS – THE CONVENTIONAL FEEDBACK/SUPPORT METHODS CANNOT EFFECTIVELY DETERMINE THE SENTIMENT ANALYSIS USER EXHIBIT. BY IMPLEMENTING SENTIMENTAL ANALYSIS, I CAN DETERMINE IF THE CUSTOMER IS SATISFIED OR NOT, AND IF IT IS AN AGGRIEVED CUSTOMER, THEN IT WILL TAKE NECESSARY ACTION TO REDIRECT THE CONVERSATION TO A LIVE AGENT.

DATA AND DATA REQUIREMENTS

Dialogue Data

- The data I used was Amazon reviews on electronic products and was retrieved from Kaggle
- This dataset used to train the chatbot model in order to solve the problem of low engagement and slow decision.
- Tags and labels of sentiments associated with available text observations are used for developing the sentiment analysis model
- To ensure sustained engagement, details of the customer's purchase history will be provided. This will enable the system to better understand customer preferences.
- Customer review and feedback will be necessary as it is the text data for which I are trying to deduce the sentiments.

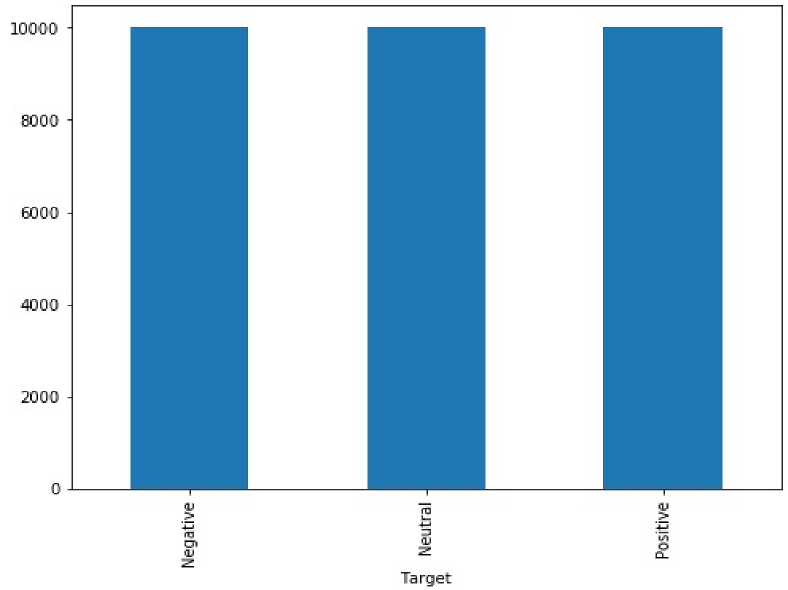
	Rating	Reviews	Target
0	3	It's battery life is great. It's very responsi...	Neutral
1	3	My fiance had this phone previously, but cause...	Neutral
2	3	unfortunately Sprint could not activate the ph...	Neutral
3	3	the reasons for the 3 star rating was it was i...	Neutral
4	3	I love the phone, but one problem and one prob...	Neutral

DATA PREPROCESSING

Feature Engineering

Rating	Sentiment Label
1- 2	Negative
3	Neutral
4-5	Positive

Feature Selection



DATA PROCESSING

Label Encoding

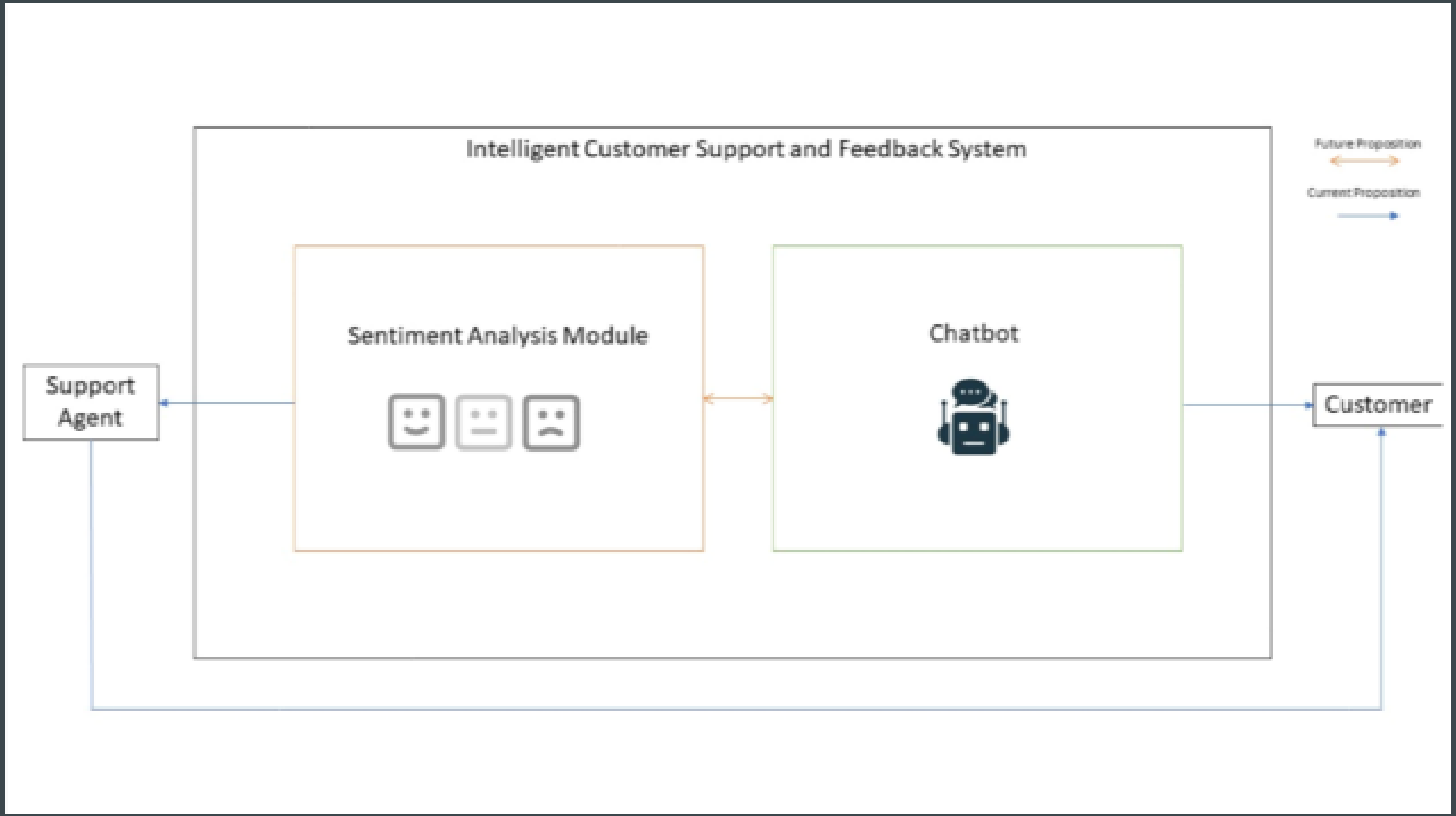
```
df1['category_id'] = df1['Target'].factorize()[0]
```

```
Out[2]: {'Neutral': 0, 'Negative': 1, 'Positive': 2}
```

TF ID Vectorization

```
In [30]: from sklearn.feature_extraction.text import TfidfVectorizer
         tfidf1 = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2', encoding='latin-1', ngram_range=(1, 2), stop_words='english')
         features1 = tfidf1.fit_transform(df1.Reviews).toarray()
         labels = df1.category_id
```

MODEL AND ARCHITECTURE APPROACH

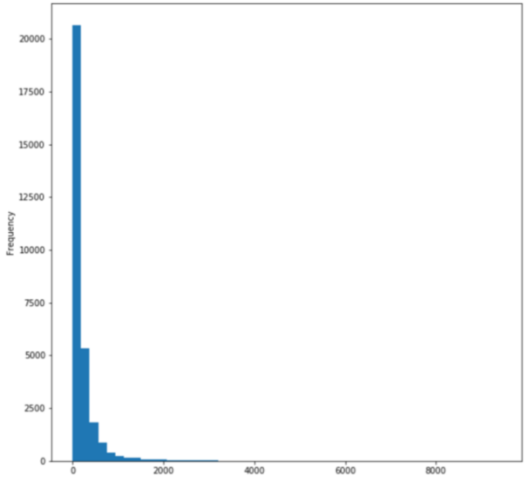


ARCHITECTURE OF INTELLIGENT CUSTOMER SUPPORT AND FEEDBACK SYSTEM

PYTHON DATA ANALYSIS AND MODEL FITTING

```python
fd['lenght'].plot(figsize=(10,10),bins=50,kind='hist')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0e90d44550>
```




```python
from io import StringIO
col = ['Target', 'Reviews']
df = df[col]
df = df[pd.notnull(df['Reviews'])]
df.columns = ['Target', 'Reviews']
df['category_id'] = df['Target'].factorize()[0]
category_id_df = df[['Target', 'category_id']].drop_duplicates().sort_values('category_id')
category_to_id = dict(category_id_df.values)
id_to_category = dict(category_id_df[['category_id', 'Target']].values)
#df.head(100)
df.info
```

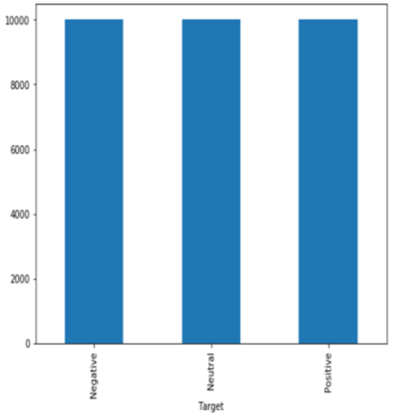
```
<bound method DataFrame.info of          Target                                            Reviews  \
0       Neutral  It's battery life is great. It's very responsi...
1       Neutral  My fiance had this phone previously, but cause...
2       Neutral  unfortunately Sprint could not activate the ph...
3       Neutral  the reasons for the 3 star rating was it was i...
4       Neutral  I love the phone, but one problem and one prob...
...         ...                                                ...
29996  Positive  I bought this product for a friend in Ghana an...
29997  Positive  The box came selaed, yet I saw some fingerprin...
29998  Positive  "excelente, coloque mi SIM CARD, segui as inst...
29999  Positive  I am from QATAR, and the new Iphone 4s factory...
30000  Positive                    I was very happy with this phone

       category_id
0                0
1                0
2                0
3                0
4                0
...            ...
29996            2
29997            2
29998            2
29999            2
30000            2

[29990 rows x 3 columns]>
```

Distribution for Reviews

```python
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,6))
df.groupby('Target').Reviews.count().plot.bar(ylim=0)
plt.show()
```




```python
from sklearn.feature_selection import chi2
import numpy as np
N = 2
for Target, category_id in sorted(category_to_id.items()):
    features_chi2 = chi2(features, labels == category_id)
    indices = np.argsort(features_chi2[0])
    feature_names = np.array(tfidf.get_feature_names())[indices]
    unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
    bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
    print("# '{}':".format(Target))
    print("  . Most correlated unigrams:\n. {}".format('\n. '.join(unigrams[-N:])))
    print("  . Most correlated bigrams:\n. {}".format('\n. '.join(bigrams[-N:])))
```

```
# 'Negative':
  . Most correlated unigrams:
. great
. good
  . Most correlated bigrams:
. great phone
. didn work
# 'Neutral':
  . Most correlated unigrams:
. blu
. ok
  . Most correlated bigrams:
. blu phone
. ok phone
# 'Positive':
  . Most correlated unigrams:
. excellent
. great
  . Most correlated bigrams:
. great phone
. works great
```

Distribution of Labels

ALGORITHM FITTING

Random Forest Classifier

Random Forest Classifier

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=200, max_depth=3, random_state=0)
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, df.index, test_size=0.33, ran
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
#Testing Prediction

print("-----CONFUSTION MATRIX-----")

print(confusion_matrix(y_test, y_pred))

print("-----CLASSIFICATION REPORT-----")

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

```
-----CONFUSTION MATRIX-----
[[2015  775  500]
 [ 459 2623  205]
 [ 811  299 2210]]
-----CLASSIFICATION REPORT-----
              precision    recall  f1-score   support

           0       0.61      0.61      0.61      3290
           1       0.71      0.80      0.75      3287
           2       0.76      0.67      0.71      3320

    accuracy                           0.69      9897
   macro avg       0.69      0.69      0.69      9897
weighted avg       0.69      0.69      0.69      9897
```

Logistic Regression

Logistic Regression

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state=0)
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, df.index, test_size=0.33, ran
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
#Testing Prediction

print("-----CONFUSTION MATRIX-----")

print(confusion_matrix(y_test, y_pred))

print("-----CLASSIFICATION REPORT-----")

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

```
C:\Users\rosha\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\rosha\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:469: FutureWarning: Default multi_class will be cha
nged to 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

```
-----CONFUSTION MATRIX-----
[[2619  343  328]
 [ 222 2964  101]
 [ 329  158 2833]]
-----CLASSIFICATION REPORT-----
              precision    recall  f1-score   support

           0       0.83      0.80      0.81      3290
           1       0.86      0.90      0.88      3287
           2       0.87      0.85      0.86      3320

    accuracy                           0.85      9897
   macro avg       0.85      0.85      0.85      9897
weighted avg       0.85      0.85      0.85      9897
```

ALGORITHM FITTING

Decision Tree Classifier

Decistion Tree Classifier

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, df.index, test_size=0.33, ran
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
#Testing Prediction

print("-----CONFUSTION MATRIX-----")

print(confusion_matrix(y_test, y_pred))

print("-----CLASSIFICATION REPORT-----")

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

```
-----CONFUSTION MATRIX-----
[[2712  230  348]
 [ 256 2847  184]
 [ 310  177 2833]]
-----CLASSIFICATION REPORT-----
              precision    recall  f1-score   support

           0       0.83      0.82      0.83      3290
           1       0.87      0.87      0.87      3287
           2       0.84      0.85      0.85      3320

    accuracy                           0.85      9897
   macro avg       0.85      0.85      0.85      9897
weighted avg       0.85      0.85      0.85      9897
```

KNN

K Nearest Neighbour

```
In [13]: from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, df.index, test_size=0.33, ran
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
#Testing Prediction

print("-----CONFUSTION MATRIX-----")

print(confusion_matrix(y_test, y_pred))

print("-----CLASSIFICATION REPORT-----")

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

```
-----CONFUSTION MATRIX-----
[[2363   88  839]
 [ 550 1932  805]
 [ 607   91 2622]]
-----CLASSIFICATION REPORT-----
              precision    recall  f1-score   support

           0       0.67      0.72      0.69      3290
           1       0.92      0.59      0.72      3287
           2       0.61      0.79      0.69      3320

    accuracy                           0.70      9897
   macro avg       0.73      0.70      0.70      9897
weighted avg       0.73      0.70      0.70      9897
```

```
In [ ]:
```

ALGORITHM FITTING

Naïve Bayes Classifier

```
from sklearn.metrics import classification_report, confusion_matrix
y_pred=clf.predict(count_vect.transform(X_test))
#y_pred=final.predict("An obvious vanity press for Julie in her first movie with Blake. Let's see. Where do we begin. She is a t

print("-----CONFUSTION MATRIX-----")

print(confusion_matrix(y_test, y_pred))

print("-----CLASSIFICATION REPORT-----")

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

```
-----CONFUSTION MATRIX-----
[[2175  253   48]
 [ 475 1762  253]
 [ 200  447 1885]]
-----CLASSIFICATION REPORT-----
              precision    recall  f1-score   support

    Negative       0.76      0.88      0.82      2476
     Neutral       0.72      0.71      0.71      2490
    Positive       0.86      0.74      0.80      2532

    accuracy                           0.78      7498
   macro avg       0.78      0.78      0.78      7498
weighted avg       0.78      0.78      0.78      7498
```

SVM

Support Vector Machines

```
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
model = LinearSVC()
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, df.index, test_size=0.33, ran
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
#Testing Prediction

print("-----CONFUSTION MATRIX-----")

print(confusion_matrix(y_test, y_pred))

print("-----CLASSIFICATION REPORT-----")

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

```
-----CONFUSTION MATRIX-----
[[2746  258  286]
 [ 179 3000  108]
 [ 261  101 2958]]
-----CLASSIFICATION REPORT-----
              precision    recall  f1-score   support

           0       0.86      0.83      0.85      3290
           1       0.89      0.91      0.90      3287
           2       0.88      0.89      0.89      3320

    accuracy                           0.88      9897
   macro avg       0.88      0.88      0.88      9897
weighted avg       0.88      0.88      0.88      9897
```

ALGORITHM SELECTION AND RATIONALE

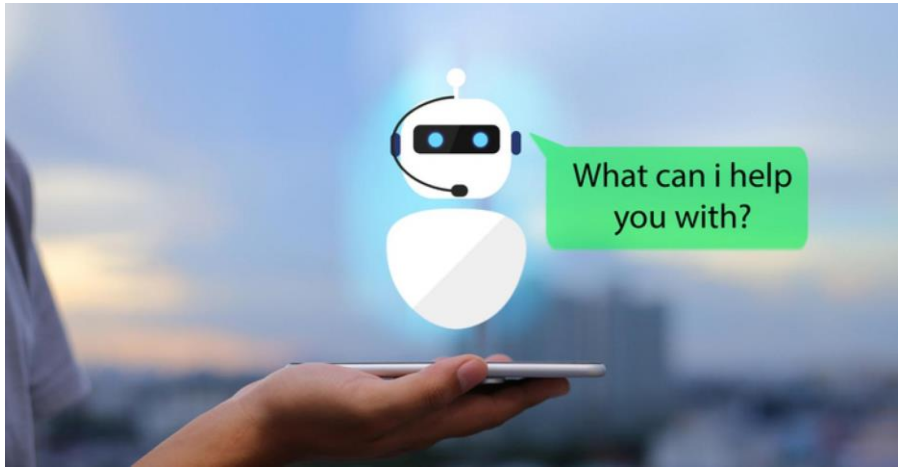
SUPPORT VECTOR MACHINES GAVE ME OUR HIGHEST ACCURACY SCORE OF 88% CORRECT PREDICTIONS. AFTER ATTAINING THE METRICS, OUR MAIN FOCUS WAS ON THE RECALL PORTION OF THE SCORE. THIS IS DUE TO THE FACT THAT FALSE NEGATIVES ARE NOT PERMITTED. I WOULD RATHER CLASSIFY A HAPPY CUSTOMER AS SAD THAN A SAD CUSTOMER AS HAPPY. THIS COULD BE DISTRACTING REGARDING WHICH CUSTOMERS TO REACH OUT TO AS I WOULD WANT TO JUSTIFY AND COMPENSATE ANY CUSTOMERS WHO WERE NOT SATISFIED. OVERALL THE RECALL SCORES FOR OUR SVM MODEL WERE QUITE HIGH THROUGHOUT ALL THREE CATEGORIES OF NEGATIVE, NEUTRAL AND POSITIVE.

MODEL ARCHITECTURE AND APPROACH

The system comprises of two modules the sentiment analysis and chatbot modules. Currently, our system will provide support agent with data about the customers sentiments this would make it easier for the agent to quickly identify high priority cases where the customer needs to me contacted

- 3 phases for building the model:
  - Phase 1- Natural Language Processing (NLP) to build a classifier using Python Programming Language which will classify the customers feedback into Positive, Negative and Neutral. Cleaning and preprocessing data is also necessary to format in a way the program can easily understand and in order to fit an appropriate model.
  - Phase 2 - improving the accuracy of the existing model by introducing sophisticated ways to enhance the accuracy. Text processing, N Grams, TF-IDF.
  - Phase 3- Implement Deep Learning Model

CHATBOT MODULE



FOR THIS MODULE, I WILL BE USING THE FAQ TEXT AS PROVIDED BY THE CLIENT IN A JSON FORMAT WHICH CONTAINS THE INTENTS, POSSIBLE QUESTIONS AND RESPONSES TO TRAIN A SIMPLE NEURAL NETWORK

THANK YOU FOR YOUR ATTENTION