Amazon Sentimental Analysis

Classification Problem:

Daily Analysis of a product such as emotions, sentiment etc. using Amazon data

Text_ID | Product_Description | Product_Type | Sentiment

Team : Pankti Rashmin Satra

EDA & Visualization

Sentimental Analysis

Model Building

Finalize Model based on Accuracy

Deployment

EDA & VISUALIZATION

Product_Type	0	1	2	3	4	5	6	7	8	9	All
Sentiment											
0	0	0	2	1	0	0	2	0	1	105	111
1	4	5	69	49	0	36	84	43	65	44	399
2	1	1	15	10	0	6	16	8	6	3702	3765
3	47	53	379	240	19	171	563	276	122	219	2089
All	52	59	465	300	19	213	665	327	194	4070	6364

```
def review_cleaning(text):
    stop_words = stopwords.words('english')
    stop_words.extend(["sxsw","@","rt","re","w","u","m","s","sxswi","mention","link","amp","sx","sw","wi","sxs",
                       "google","app","phone","pad","apple","austin","quot","android","ipad","marissa","mayer",
                       "social","network","store","via","popup","called","zlf","zms","quotmajorquot"])

    text = normalize("NFKD", text).encode("ascii", "ignore").decode("utf-8", "ignore") # Encoding & Decoding Data
    text = contractions.fix(text)                                                  #Contraction Replacement
    text = re.sub("\[.*?\]","", text)                                              #brackets
    text = re.sub("https?://\S+|www\.\S+", "", text)                               #links
    text = re.sub("<.*?>+", "", text)                                              #characters
    text = re.sub("[%s]" % re.escape(string.punctuation), "", text)                #punctuations
    text = re.sub("\n","", text)                                                   #new line
    text = re.sub("\w*\d\w*","", text)                                             #numbers
    text = " ".join([s for s in re.split("([A-Z][a-z]+[^A-Z]*)",text) if s])       #Split attached Uppercase words
    text = "".join("".join(s)[:2] for _, s in itertools.groupby(text))          #remove letter repeating twice in continuation
    text = str(text).lower()                                                       #Normalization
    text = " ".join(s for s in str(text).split() if s not in stop_words)           #stopwords
    text = " ". join([w.lemmatize() for w in TextBlob(text).words])                #Lemmatizaion
    return text
```

SENTIMENTAL ANALYSIS

	mean	median
	Polarity_score	Polarity_score
Sentiment		
0	0.078659	0.000000
1	0.022575	0.000000
2	0.101766	0.000000
3	0.211351	0.136364

since sentiment 0 & 1 have very few reviews and polarity scores to a similar range, we combine both of them to balance the data

```
amazon["Sentiment"] = amazon["Sentiment"].replace(0,1)
labelencoder = LabelEncoder()
amazon["Sentiment"] = labelencoder.fit_transform(amazon["Sentiment"])
```

		mean	median
		Polarity_score	Polarity_score
	Sentiment		
Negative	0	0.034782	0.000000
Neutral	1	0.101766	0.000000
Positive	2	0.211351	0.136364

MODEL BUILDING

- X = TfidfVectors, Product_Type,

  Polarity_score

  Shape = (6364, 7592)
- Y = Sentiment

  Shape = (6364, 1)
- Trained Model using Various Classifiers like :

  SVM, XG Boost, Ada Boost, KNN, Random Forest, Decision Tree,

  Gradient Boosting

Accuracy	Sentiment	Classifier						
		SVM	XGBM	AB	KNN	RF	CART	GB
Precision	0	1.00	0.61	0.00	0.72	0.00	0.00	0.73
	1	0.92	0.92	0.92	0.91	0.60	0.60	0.92
	2	0.83	0.83	0.80	0.84	1.00	0.50	0.83
Recall	0	0.16	0.18	0.00	0.22	0.00	0.00	0.21
	1	0.98	0.98	0.98	0.98	1.00	1.00	0.98
	2	0.91	0.88	0.91	0.88	0.01	0.01	0.89
F1	0	0.28	0.28	0.00	0.33	0.00	0.00	0.32
	1	0.95	0.94	0.95	0.94	0.75	0.75	0.95
	2	0.86	0.85	0.85	0.86	0.01	0.01	0.86
Model Score		88.61%	87.82%	87.27%	88.37%	60.41%	60.33%	88.53%

	0	1	2	3	4	5	6	7	8	9	...	7582	7583	7584	7585	7586	7587	7588	7589	Product_Type	Polarity_score
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9	0.3
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9	-0.6

FINAL MODEL BASED ON ACCURACY

Gradient Boosting Model returns the Highest Accuracy

```
model = GradientBoostingClassifier()
model.fit(X,Y)
dump(model, open("Amazon.sav", "wb"))
loaded_model = load(open('Amazon.sav', 'rb'))
result = loaded_model.score(X, Y)
```

89.3935%

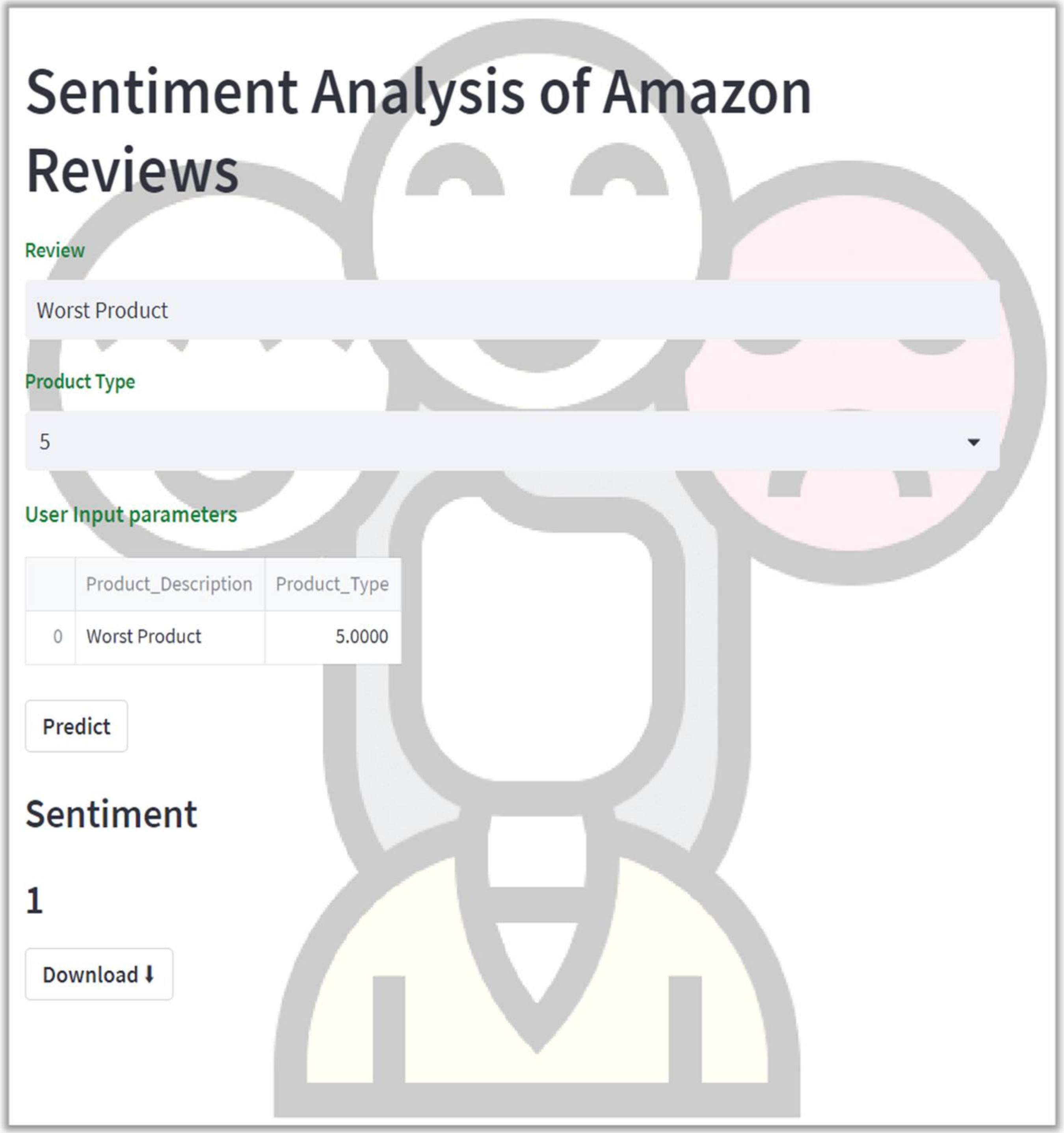
<u>Accuracy</u>

Classification Report -

	precision	recall	f1-score	support
0	0.73	0.21	0.32	106
1	0.92	0.98	0.95	767
2	0.83	0.89	0.86	400
accuracy			0.89	1273
macro avg	0.83	0.69	0.71	1273
weighted avg	0.88	0.89	0.87	1273

DEPLOYMENT

- Deployed model on Streamlit using command prompt
- Predict Button to Predict the Sentiment
- Download Button to save prediction as .csv file.



THANK YOU