**SENTIMENTAL ANALYSIS FOR MARKETING**

**Dataset :**

Twitter data was scraped from February of 2015 and contributors were asked to first classify positive, negative, and neutral tweets, followed by categorizing negative reasons (such as “late flight” or “rude service”).

**Program**:

{

"cells": [

{

"cell\_type": "markdown",

"id": "65d1b629",

"metadata": {

"papermill": {

"duration": 0.005039,

"end\_time": "2023-08-16T14:15:41.969489",

"exception": false,

"start\_time": "2023-08-16T14:15:41.964450",

"status": "completed"

},

"tags": []

},

"source": [

"# Importing Libraries"

]

},

{

"cell\_type": "code",

"execution\_count": 1,

"id": "562d3cb8",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:15:41.980120Z",

"iopub.status.busy": "2023-08-16T14:15:41.979729Z",

"iopub.status.idle": "2023-08-16T14:16:04.435508Z",

"shell.execute\_reply": "2023-08-16T14:16:04.433961Z"

},

"papermill": {

"duration": 22.463896,

"end\_time": "2023-08-16T14:16:04.437812",

"exception": false,

"start\_time": "2023-08-16T14:15:41.973916",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"/opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5\n",

" warnings.warn(f\"A NumPy version >={np\_minversion} and <{np\_maxversion}\"\n"

]

},

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[nltk\_data] Error loading stopwords: <urlopen error [Errno -3]\n",

"[nltk\_data] Temporary failure in name resolution>\n"

]

}

],

"source": [

"# Basic libraries\n",

"import numpy as np\n",

"import pandas as pd\n",

"import matplotlib.pyplot as plt\n",

"import seaborn as sns\n",

"%matplotlib inline\n",

"import pickle\n",

"import warnings\n",

"warnings.filterwarnings(action='ignore')\n",

"\n",

"# nltk\n",

"import nltk\n",

"nltk.download('stopwords')\n",

"\n",

"## Preprocessing libraries\n",

"import re\n",

"from nltk.corpus import stopwords\n",

"from nltk.stem.porter import PorterStemmer\n",

"from sklearn.feature\_extraction.text import TfidfVectorizer\n",

"\n",

"# For Model training\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.linear\_model import LogisticRegression\n",

"from sklearn.naive\_bayes import BernoulliNB\n",

"from sklearn.svm import LinearSVC # a variant of SVC optimized for large datasets\n",

"\n",

"# Metrics for accuracy\n",

"from sklearn.metrics import accuracy\_score,confusion\_matrix, classification\_report"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"id": "30c38b25",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:04.448914Z",

"iopub.status.busy": "2023-08-16T14:16:04.448563Z",

"iopub.status.idle": "2023-08-16T14:16:04.598104Z",

"shell.execute\_reply": "2023-08-16T14:16:04.596834Z"

},

"papermill": {

"duration": 0.157462,

"end\_time": "2023-08-16T14:16:04.600149",

"exception": false,

"start\_time": "2023-08-16T14:16:04.442687",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>tweet\_id</th>\n",

" <th>airline\_sentiment</th>\n",

" <th>airline\_sentiment\_confidence</th>\n",

" <th>negativereason</th>\n",

" <th>negativereason\_confidence</th>\n",

" <th>airline</th>\n",

" <th>airline\_sentiment\_gold</th>\n",

" <th>name</th>\n",

" <th>negativereason\_gold</th>\n",

" <th>retweet\_count</th>\n",

" <th>text</th>\n",

" <th>tweet\_coord</th>\n",

" <th>tweet\_created</th>\n",

" <th>tweet\_location</th>\n",

" <th>user\_timezone</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>570306133677760513</td>\n",

" <td>neutral</td>\n",

" <td>1.0000</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>Virgin America</td>\n",

" <td>NaN</td>\n",

" <td>cairdin</td>\n",

" <td>NaN</td>\n",

" <td>0</td>\n",

" <td>@VirginAmerica What @dhepburn said.</td>\n",

" <td>NaN</td>\n",

" <td>2015-02-24 11:35:52 -0800</td>\n",

" <td>NaN</td>\n",

" <td>Eastern Time (US &amp; Canada)</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>570301130888122368</td>\n",

" <td>positive</td>\n",

" <td>0.3486</td>\n",

" <td>NaN</td>\n",

" <td>0.0000</td>\n",

" <td>Virgin America</td>\n",

" <td>NaN</td>\n",

" <td>jnardino</td>\n",

" <td>NaN</td>\n",

" <td>0</td>\n",

" <td>@VirginAmerica plus you've added commercials t...</td>\n",

" <td>NaN</td>\n",

" <td>2015-02-24 11:15:59 -0800</td>\n",

" <td>NaN</td>\n",

" <td>Pacific Time (US &amp; Canada)</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>570301083672813571</td>\n",

" <td>neutral</td>\n",

" <td>0.6837</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>Virgin America</td>\n",

" <td>NaN</td>\n",

" <td>yvonnalynn</td>\n",

" <td>NaN</td>\n",

" <td>0</td>\n",

" <td>@VirginAmerica I didn't today... Must mean I n...</td>\n",

" <td>NaN</td>\n",

" <td>2015-02-24 11:15:48 -0800</td>\n",

" <td>Lets Play</td>\n",

" <td>Central Time (US &amp; Canada)</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>570301031407624196</td>\n",

" <td>negative</td>\n",

" <td>1.0000</td>\n",

" <td>Bad Flight</td>\n",

" <td>0.7033</td>\n",

" <td>Virgin America</td>\n",

" <td>NaN</td>\n",

" <td>jnardino</td>\n",

" <td>NaN</td>\n",

" <td>0</td>\n",

" <td>@VirginAmerica it's really aggressive to blast...</td>\n",

" <td>NaN</td>\n",

" <td>2015-02-24 11:15:36 -0800</td>\n",

" <td>NaN</td>\n",

" <td>Pacific Time (US &amp; Canada)</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>570300817074462722</td>\n",

" <td>negative</td>\n",

" <td>1.0000</td>\n",

" <td>Can't Tell</td>\n",

" <td>1.0000</td>\n",

" <td>Virgin America</td>\n",

" <td>NaN</td>\n",

" <td>jnardino</td>\n",

" <td>NaN</td>\n",

" <td>0</td>\n",

" <td>@VirginAmerica and it's a really big bad thing...</td>\n",

" <td>NaN</td>\n",

" <td>2015-02-24 11:14:45 -0800</td>\n",

" <td>NaN</td>\n",

" <td>Pacific Time (US &amp; Canada)</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" tweet\_id airline\_sentiment airline\_sentiment\_confidence \\\n",

"0 570306133677760513 neutral 1.0000 \n",

"1 570301130888122368 positive 0.3486 \n",

"2 570301083672813571 neutral 0.6837 \n",

"3 570301031407624196 negative 1.0000 \n",

"4 570300817074462722 negative 1.0000 \n",

"\n",

" negativereason negativereason\_confidence airline \\\n",

"0 NaN NaN Virgin America \n",

"1 NaN 0.0000 Virgin America \n",

"2 NaN NaN Virgin America \n",

"3 Bad Flight 0.7033 Virgin America \n",

"4 Can't Tell 1.0000 Virgin America \n",

"\n",

" airline\_sentiment\_gold name negativereason\_gold retweet\_count \\\n",

"0 NaN cairdin NaN 0 \n",

"1 NaN jnardino NaN 0 \n",

"2 NaN yvonnalynn NaN 0 \n",

"3 NaN jnardino NaN 0 \n",

"4 NaN jnardino NaN 0 \n",

"\n",

" text tweet\_coord \\\n",

"0 @VirginAmerica What @dhepburn said. NaN \n",

"1 @VirginAmerica plus you've added commercials t... NaN \n",

"2 @VirginAmerica I didn't today... Must mean I n... NaN \n",

"3 @VirginAmerica it's really aggressive to blast... NaN \n",

"4 @VirginAmerica and it's a really big bad thing... NaN \n",

"\n",

" tweet\_created tweet\_location user\_timezone \n",

"0 2015-02-24 11:35:52 -0800 NaN Eastern Time (US & Canada) \n",

"1 2015-02-24 11:15:59 -0800 NaN Pacific Time (US & Canada) \n",

"2 2015-02-24 11:15:48 -0800 Lets Play Central Time (US & Canada) \n",

"3 2015-02-24 11:15:36 -0800 NaN Pacific Time (US & Canada) \n",

"4 2015-02-24 11:14:45 -0800 NaN Pacific Time (US & Canada) "

]

},

"execution\_count": 2,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Reading our dataset\n",

"df = pd.read\_csv('/kaggle/input/twitter-airline-sentiment/Tweets.csv')\n",

"df.head()"

]

},

{

"cell\_type": "markdown",

"id": "8ca1f7ce",

"metadata": {

"papermill": {

"duration": 0.004696,

"end\_time": "2023-08-16T14:16:04.610061",

"exception": false,

"start\_time": "2023-08-16T14:16:04.605365",

"status": "completed"

},

"tags": []

},

"source": [

"# EDA"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"id": "e3526faf",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:04.621668Z",

"iopub.status.busy": "2023-08-16T14:16:04.620987Z",

"iopub.status.idle": "2023-08-16T14:16:04.654437Z",

"shell.execute\_reply": "2023-08-16T14:16:04.653076Z"

},

"papermill": {

"duration": 0.041952,

"end\_time": "2023-08-16T14:16:04.656834",

"exception": false,

"start\_time": "2023-08-16T14:16:04.614882",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"data": {

"text/plain": [

"tweet\_id 0\n",

"airline\_sentiment 0\n",

"airline\_sentiment\_confidence 0\n",

"negativereason 5462\n",

"negativereason\_confidence 4118\n",

"airline 0\n",

"airline\_sentiment\_gold 14600\n",

"name 0\n",

"negativereason\_gold 14608\n",

"retweet\_count 0\n",

"text 0\n",

"tweet\_coord 13621\n",

"tweet\_created 0\n",

"tweet\_location 4733\n",

"user\_timezone 4820\n",

"dtype: int64"

]

},

"execution\_count": 3,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"df.isnull().sum()"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"id": "7b152a79",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:04.669876Z",

"iopub.status.busy": "2023-08-16T14:16:04.669476Z",

"iopub.status.idle": "2023-08-16T14:16:04.867707Z",

"shell.execute\_reply": "2023-08-16T14:16:04.866829Z"

},

"papermill": {

"duration": 0.207536,

"end\_time": "2023-08-16T14:16:04.869769",

"exception": false,

"start\_time": "2023-08-16T14:16:04.662233",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"data": {

"image/png":

"text/plain": [

"<Figure size 700x300 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"# Checking the distribution of airlines\n",

"plt.figure(figsize=(7,3))\n",

"sns.countplot(data=df,x='airline', palette=['#1f78b4', '#33a02c', '#e31a1c', '#ff7f00', '#6a3d9a', '#a6cee3'])\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"id": "a7fa1354",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:04.884219Z",

"iopub.status.busy": "2023-08-16T14:16:04.883838Z",

"iopub.status.idle": "2023-08-16T14:16:05.018558Z",

"shell.execute\_reply": "2023-08-16T14:16:05.017629Z"

},

"papermill": {

"duration": 0.143801,

"end\_time": "2023-08-16T14:16:05.020630",

"exception": false,

"start\_time": "2023-08-16T14:16:04.876829",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"data": {

"image/png":

"text/plain": [

"<Figure size 700x300 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"# Seeing the distribution of positive and negative tweet reviews in target column\n",

"plt.figure(figsize=(7,3))\n",

"sns.countplot(data=df,x='airline\_sentiment',palette=['yellow', 'green','red'])\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"id": "f4daa7f5",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:05.040098Z",

"iopub.status.busy": "2023-08-16T14:16:05.038024Z",

"iopub.status.idle": "2023-08-16T14:16:05.287821Z",

"shell.execute\_reply": "2023-08-16T14:16:05.286965Z"

},

"papermill": {

"duration": 0.262129,

"end\_time": "2023-08-16T14:16:05.290275",

"exception": false,

"start\_time": "2023-08-16T14:16:05.028146",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"data": {

"image/png":

"text/plain": [

"<Figure size 800x800 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"# Calculate the value counts for each negative reason\n",

"value\_counts = df['negativereason'].value\_counts()\n",

"\n",

"# Create a donut-like pie chart using matplotlib and seaborn\n",

"plt.figure(figsize=(8, 8))\n",

"labels = value\_counts.index\n",

"values = value\_counts.values\n",

"colors = sns.color\_palette('pastel')[0:len(labels)] # Use pastel colors for the chart\n",

"plt.pie(values, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140, wedgeprops=dict(width=0.3))\n",

"plt.title('Overall distribution for negative reasons')\n",

"plt.axis('equal') # Equal aspect ratio ensures the pie chart is drawn as a circle.\n",

"plt.show()"

]

},

{

"cell\_type": "markdown",

"id": "8b5b22c3",

"metadata": {

"papermill": {

"duration": 0.0083,

"end\_time": "2023-08-16T14:16:05.307411",

"exception": false,

"start\_time": "2023-08-16T14:16:05.299111",

"status": "completed"

},

"tags": []

},

"source": [

"## Data clearing and preprocessing of Text"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"id": "fd6bf500",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:05.322523Z",

"iopub.status.busy": "2023-08-16T14:16:05.321899Z",

"iopub.status.idle": "2023-08-16T14:16:37.607161Z",

"shell.execute\_reply": "2023-08-16T14:16:37.605444Z"

},

"papermill": {

"duration": 32.295432,

"end\_time": "2023-08-16T14:16:37.609494",

"exception": false,

"start\_time": "2023-08-16T14:16:05.314062",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"corpus = []\n",

"ps=PorterStemmer()\n",

"for i in range(len(df)):\n",

" # Removing special characters from text(message)\n",

" review = re.sub('[^a-zA-Z]', ' ', df['text'][i])\n",

" \n",

" # Converting entire text into lower case\n",

" review = review.lower()\n",

" \n",

" # Splitting our text into words\n",

" review = review.split()\n",

" \n",

" # Stemming and removing stopwords\n",

" review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]\n",

" \n",

" # Joining all the words into a comple text\n",

" review = ' '.join(review)\n",

" \n",

" # Appending each text into the list corpus\n",

" corpus.append(review) "

]

},

{

"cell\_type": "code",

"execution\_count": 8,

"id": "e65effa0",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:37.624097Z",

"iopub.status.busy": "2023-08-16T14:16:37.623745Z",

"iopub.status.idle": "2023-08-16T14:16:37.629458Z",

"shell.execute\_reply": "2023-08-16T14:16:37.628045Z"

},

"papermill": {

"duration": 0.01555,

"end\_time": "2023-08-16T14:16:37.631758",

"exception": false,

"start\_time": "2023-08-16T14:16:37.616208",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# Creating the Bag of Words model\n",

"cv = TfidfVectorizer(ngram\_range=(1,2), max\_features=500000)"

]

},

{

"cell\_type": "code",

"execution\_count": 9,

"id": "0c5f5e86",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:37.646135Z",

"iopub.status.busy": "2023-08-16T14:16:37.645724Z",

"iopub.status.idle": "2023-08-16T14:16:38.212149Z",

"shell.execute\_reply": "2023-08-16T14:16:38.211211Z"

},

"papermill": {

"duration": 0.576022,

"end\_time": "2023-08-16T14:16:38.214308",

"exception": false,

"start\_time": "2023-08-16T14:16:37.638286",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# We will use X as independent feature section\n",

"X = cv.fit\_transform(corpus)\n",

"# We will use y as dependent feature section\n",

"y=df['airline\_sentiment']"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"id": "5caa204f",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:38.229410Z",

"iopub.status.busy": "2023-08-16T14:16:38.228581Z",

"iopub.status.idle": "2023-08-16T14:16:38.343176Z",

"shell.execute\_reply": "2023-08-16T14:16:38.341777Z"

},

"papermill": {

"duration": 0.124498,

"end\_time": "2023-08-16T14:16:38.345618",

"exception": false,

"start\_time": "2023-08-16T14:16:38.221120",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"No. of feature\_words: 91436\n"

]

}

],

"source": [

"print('No. of feature\_words: ', len(cv.get\_feature\_names\_out()))"

]

},

{

"cell\_type": "code",

"execution\_count": 11,

"id": "7b6dd5ff",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:38.360704Z",

"iopub.status.busy": "2023-08-16T14:16:38.359761Z",

"iopub.status.idle": "2023-08-16T14:16:38.613891Z",

"shell.execute\_reply": "2023-08-16T14:16:38.613192Z"

},

"papermill": {

"duration": 0.263732,

"end\_time": "2023-08-16T14:16:38.615977",

"exception": false,

"start\_time": "2023-08-16T14:16:38.352245",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# Creating a pickle file for the TfidfVectorizer\n",

"with open('cv-transform.pkl', 'wb') as f:\n",

" pickle.dump(cv, f)"

]

},

{

"cell\_type": "markdown",

"id": "323315dc",

"metadata": {

"papermill": {

"duration": 0.006117,

"end\_time": "2023-08-16T14:16:38.628510",

"exception": false,

"start\_time": "2023-08-16T14:16:38.622393",

"status": "completed"

},

"tags": []

},

"source": [

"# Model Training"

]

},

{

"cell\_type": "code",

"execution\_count": 12,

"id": "4ccb8d92",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:38.643751Z",

"iopub.status.busy": "2023-08-16T14:16:38.642446Z",

"iopub.status.idle": "2023-08-16T14:16:38.653480Z",

"shell.execute\_reply": "2023-08-16T14:16:38.652214Z"

},

"papermill": {

"duration": 0.020821,

"end\_time": "2023-08-16T14:16:38.655725",

"exception": false,

"start\_time": "2023-08-16T14:16:38.634904",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# Train Test Split\n",

"X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, random\_state = 0)"

]

},

{

"cell\_type": "code",

"execution\_count": 13,

"id": "3975b93f",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:38.671110Z",

"iopub.status.busy": "2023-08-16T14:16:38.669912Z",

"iopub.status.idle": "2023-08-16T14:16:38.675684Z",

"shell.execute\_reply": "2023-08-16T14:16:38.674475Z"

},

"papermill": {

"duration": 0.015633,

"end\_time": "2023-08-16T14:16:38.677952",

"exception": false,

"start\_time": "2023-08-16T14:16:38.662319",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# Training using three algorithms, let's see which will give us better result\n",

"model1=LogisticRegression()\n",

"model2=BernoulliNB()\n",

"model3=LinearSVC()\n",

"model=[model1, model2, model3]"

]

},

{

"cell\_type": "code",

"execution\_count": 14,

"id": "884e8464",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:38.693160Z",

"iopub.status.busy": "2023-08-16T14:16:38.692774Z",

"iopub.status.idle": "2023-08-16T14:16:44.368158Z",

"shell.execute\_reply": "2023-08-16T14:16:44.367314Z"

},

"papermill": {

"duration": 5.685399,

"end\_time": "2023-08-16T14:16:44.370070",

"exception": false,

"start\_time": "2023-08-16T14:16:38.684671",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"M-O-D-E-L : 1\n",

"Confusion matrix : \n",

" [[2694 532 285]\n",

" [ 77 351 81]\n",

" [ 17 36 319]]\n",

"Accuracy score : 0.7659380692167578\n",

"Classification Report : \n",

" precision recall f1-score support\n",

"\n",

" negative 0.97 0.77 0.86 3511\n",

" neutral 0.38 0.69 0.49 509\n",

" positive 0.47 0.86 0.60 372\n",

"\n",

" accuracy 0.77 4392\n",

" macro avg 0.60 0.77 0.65 4392\n",

"weighted avg 0.86 0.77 0.79 4392\n",

"\n",

"-----------------------------------------------------------\n",

"\n",

"M-O-D-E-L : 2\n",

"Confusion matrix : \n",

" [[2780 850 670]\n",

" [ 8 69 13]\n",

" [ 0 0 2]]\n",

"Accuracy score : 0.6491347905282332\n",

"Classification Report : \n",

" precision recall f1-score support\n",

"\n",

" negative 1.00 0.65 0.78 4300\n",

" neutral 0.08 0.77 0.14 90\n",

" positive 0.00 1.00 0.01 2\n",

"\n",

" accuracy 0.65 4392\n",

" macro avg 0.36 0.80 0.31 4392\n",

"weighted avg 0.98 0.65 0.77 4392\n",

"\n",

"-----------------------------------------------------------\n",

"\n",

"M-O-D-E-L : 3\n",

"Confusion matrix : \n",

" [[2620 428 197]\n",

" [ 135 426 100]\n",

" [ 33 65 388]]\n",

"Accuracy score : 0.7818761384335154\n",

"Classification Report : \n",

" precision recall f1-score support\n",

"\n",

" negative 0.94 0.81 0.87 3245\n",

" neutral 0.46 0.64 0.54 661\n",

" positive 0.57 0.80 0.66 486\n",

"\n",

" accuracy 0.78 4392\n",

" macro avg 0.66 0.75 0.69 4392\n",

"weighted avg 0.83 0.78 0.80 4392\n",

"\n",

"-----------------------------------------------------------\n",

"\n"

]

}

],

"source": [

"i = 0\n",

"for algo in model:\n",

" i += 1\n",

" print(\"M-O-D-E-L :\",i)\n",

" algo.fit(X\_train, y\_train)\n",

" y\_pred=algo.predict(X\_test)\n",

" # Checking the accuracy\n",

" print(\"Confusion matrix : \\n\",confusion\_matrix(y\_pred,y\_test))\n",

" print(\"Accuracy score : \",accuracy\_score(y\_pred,y\_test))\n",

" print(\"Classification Report : \\n\",classification\_report(y\_pred,y\_test))\n",

" print(\"-----------------------------------------------------------\\n\")"

]

},

{

"cell\_type": "markdown",

"id": "22a8f1f8",

"metadata": {

"papermill": {

"duration": 0.006703,

"end\_time": "2023-08-16T14:16:44.383796",

"exception": false,

"start\_time": "2023-08-16T14:16:44.377093",

"status": "completed"

},

"tags": []

},

"source": [

"#### Based on the metrics, Model 3 appears to be the best performer among the three models. It has the highest accuracy score (0.782) and generally higher precision, recall, and F1-scores for all three classes compared to Model 1 and Model 2. Model 1 also performs reasonably well with a good accuracy score (0.766) and balanced precision and recall for each class.\n",

"\n",

"#### On the other hand, Model 2 shows relatively low accuracy (0.649) and poor precision and F1-scores for all classes, except for \"negative\" where it has a relatively higher recall. This suggests that Model 2 may have difficulties in correctly classifying the data points for most classes.\n",

"\n",

"#### Overall, Model 3 is the recommended choice for its better overall performance across various metrics."

]

},

{

"cell\_type": "code",

"execution\_count": 15,

"id": "57fb6f30",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:44.401042Z",

"iopub.status.busy": "2023-08-16T14:16:44.400322Z",

"iopub.status.idle": "2023-08-16T14:16:44.406763Z",

"shell.execute\_reply": "2023-08-16T14:16:44.405992Z"

},

"papermill": {

"duration": 0.017646,

"end\_time": "2023-08-16T14:16:44.408413",

"exception": false,

"start\_time": "2023-08-16T14:16:44.390767",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# Creating a pickle file for our model 3 i.e. LinearSVC\n",

"with open(\"tweetmodel.pkl\",\"wb\") as file:\n",

" pickle.dump(model3,file)"

]

},

{

"cell\_type": "markdown",

"id": "99b9ab10",

"metadata": {

"papermill": {

"duration": 0.006628,

"end\_time": "2023-08-16T14:16:44.422075",

"exception": false,

"start\_time": "2023-08-16T14:16:44.415447",

"status": "completed"

},

"tags": []

},

"source": [

"## Thank You 😊"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "2520b534",

"metadata": {

"papermill": {

"duration": 0.007105,

"end\_time": "2023-08-16T14:16:44.436164",

"exception": false,

"start\_time": "2023-08-16T14:16:44.429059",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.10.12"

},

"papermill": {

"default\_parameters": {},

"duration": 71.961864,

"end\_time": "2023-08-16T14:16:45.466002",

"environment\_variables": {},

"exception": null,

"input\_path": "\_\_notebook\_\_.ipynb",

"output\_path": "\_\_notebook\_\_.ipynb",

"parameters": {},

"start\_time": "2023-08-16T14:15:33.504138",

"version": "2.4.0"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}