Springboard Capstone 3: Twitter Sentiment Analysis for US Airlines.

Twitter Sentiment Analysis for US Airlines involves analyzing tweets related to airlines in the United States to determine the sentiment expressed by users. It provides insights into customer opinions and feelings about different aspects of the airline industry, and has been used to inform business decisions and improve customer satisfaction.

This analysis was based on the Twitter US airline sentiment dataset (Crowdflower. (2016). "Airline Twitter Sentiment". Retrieved from <https://data.world/crowdflower/airline-twitter-sentiment>) and built upon an empirical study of machine learning classifiers, highlighting the usefulness of sentiment analysis in uncovering insights from large datasets (Rai, Sawan. (2021). "An Empirical study of Machine Learning Classifiers with Tweet Sentiment Classification". <https://www.analyticsvidhya.com/blog/2021/11/an-empirical-study-of-machine-learning-classifier-with-tweet-sentiment-classification/>).

Background:

Twitter is a popular platform for customers to share their experiences with airlines, and sentiment analysis can be used to analyze these tweets and categorize them into positive, negative, or neutral sentiment categories. This type of analysis involves using natural language processing techniques and machine learning algorithms to extract meaning from the text data and classify it into different sentiment categories. Researchers and industry professionals have used Twitter sentiment analysis to gain insights into customer opinions and feelings about different aspects of the airline industry. For example, sentiment analysis can be used to identify common customer complaints, determine areas for improvement, and monitor brand reputation.

1. Data  
   Data citation: Crowdflower. (2016). "Airline Twitter Sentiment". Retrieved from <https://data.world/crowdflower/airline-twitter-sentiment>  
     
   This data contained 'Airline-Sentiment-2-w-AA.csv', which had 14640 rows of data with 19 columns. Columns: \_unit\_id, \_golden, \_unit\_state, \_trusted\_judgments, \_last\_judgment\_at, airline\_sentiment, airline\_sentiment:confidence, negativereason, negativereason:confidence, airline airline\_sentiment\_gold, name, negativereason\_gold, retweet\_count, text, tweet\_coord, tweet\_created, tweet\_id, tweet\_location and user\_timezone.
2. Method  
   The first step in the analysis was to import the necessary libraries, load the data, and preprocess it to prepare it for classification. Preprocessing involved several steps, including cleaning the text data, removing stop words, and converting the text into numerical form using CountVectorizer from scikit-learn. The data was then split into training and testing sets, with the training set used to train the classification models and the testing set used to evaluate their performance. Five different classification models were used in this study: Naive Bayes, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest, and Gradient Boosting Classifier (GBC). Each model was trained on the training set and used to predict the sentiment of the test set. The performance of each model was then evaluated using precision, recall, and F1-score, with Seaborn used to plot the evaluation results. To gain a more nuanced understanding of the performance of each model, precision, recall, and F1-score were also plotted for each model with respect to each sentiment class. This analysis helped to identify any biases or limitations in the models and provided a more comprehensive picture of their overall performance. Finally, a bar plot was used to compare the overall weighted scores of each model and determine which one performed the best in predicting the sentiment of the Tweets.
3. Data Wrangling & Cleaning  
   The Pandas DataFrame contained 14640 rows of data containing 19 columns. The columns had information pertaining to various US airlines twitter accounts, much of which was not necessary for the Sentiment Analysis. The DataFrame was summarized into 2 columns: The text of each tweet & the labeled sentiment of each tweet. The ‘text’ was normalized using Natural Language Processing techniques, such as ‘re’ for regular expressions, ‘emoji’ for handling emojis, and ‘TweetTokenizer’ from ‘nltk.tokenize’ for tokenization. A function named ‘preprocess\_text\_data()’ was created to take a string of text as input and performs several pre-processing steps, such as replacing Twitter usernames and numbers with <USER> and <NUMBER> respectively, replacing emojis with their text description using the ‘demojize()’ function, and tokenizing the text using ‘TweetTokenizer’. This function is useful for preparing text data for machine learning models in NLP tasks like sentiment analysis, text classification, or text generation. Overall, the provided code demonstrates how to perform common text data cleaning tasks in NLP using Python modules.
4. Exploratory Data Analysis  
   Visualization of the data was done by creating a countplot of sentiment labels in an airline dataset and a stacked bar chart of sentiment labels broken down by airline. Both visualizations use the Matplotlib and Seaborn libraries to help compare the sentiment distribution of different airlines. The data needed to be formatted using the scikit-learn library's preprocessing module to perform label encoding on a categorical variable in the dataset, which converts it into a numerical format for use in machine learning models; it is useful when the categorical variable has no inherent order. The encoded numerical values can then be used as input features in machine learning models.
5. Modeling  
   In order to prepare the data for modeling, the text data was further preprocessed and train-test splitting was performed. Two empty lists were initialized to store the preprocessed text data and their corresponding categorical labels. A train-test split was done on the preprocessed text data and categorical labels using the train\_test\_split function from the sklearn.model\_selection module. It splits the data into training and testing sets with a test size of 0.3. Finally, it initializes a list ‘labels’ that contains the names of the categorical labels for further use. It all starts with a basic data preprocessing step that is necessary for training a machine learning model on text data.  
     
   In preparation for the comparison of various models, a function ‘show\_performance\_data’ was created that takes three parameters: Y\_test, Y\_pred, and model\_name. The function first uses the classification\_report() function to print a classification report containing the precision, recall, f1-score, and support for each class. The target\_names parameter is set to "labels", which presumably is a list of the class labels. The function then generates a confusion matrix using the confusion\_matrix() function and saves it as a heatmap using the seaborn library. The confusion matrix shows the number of true positive, false positive, true negative, and false negative predictions for each class. Finally, the function returns the classification report as a dictionary object using the output\_dict parameter in the classification\_report() function. This dictionary object can be used to retrieve the precision, recall, and f1-score values for each class.  
     
   A method for converting text data into numerical feature vectors called ‘CountVectorizer’ was fit to the training data. NLP models are incapable of comprehending textual data; they solely operate with numerical inputs. Therefore, it is necessary to transform the textual data into vectors. Additionally, the test data was transformed to ensure that both training and testing, and the resulting feature vectors have the same dimensions.   
     
   Finally various classifiers are trained on the data. Naïve Bayes, Support Vector Machines, K-Nearest Neighbors, Random Forest Classifiers, and Gradient Boosting Classifiers were compared. The performance of each model is compared via the ‘show\_performance\_data’ function previously prepared. The results were plotted against each other using different metrics, such as ‘precision’, ‘recall’ and ‘F1-score’.
6. Predictions & Conclusions  
   Although all the models performed well in Sentiment Classification, SVM emerged as the best in terms of 'Overall Weighted Scores'. The precision and F1-score of Random Forest and SVM were better than the other classifiers, while Naive Bayes had the highest recall score but by a small margin. The sentiment analysis task was accomplished successfully by all the classifiers. Cross-validation was also performed to verify the results using Scikit-learn’s ‘cross\_val\_score’. The cross-validation was performed 5-fold with F1-score as the metric of interest. The results of this test confirmed the results of the prior comparison, SVM has indeed performed the best for the data input. Improvements were made to the SVM model using Hyperparameter Tuning. The best Hyperparameters were {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}. This produced an F1-score of 0.80.
7. Future Improvements  
   I forgot about ‘Linear Regression’, so an improvement that could be made would be to test more models to see if they can achieve better scores for Sentiment Analysis.
8. Credits  
   Crowdflower. (2016). "Airline Twitter Sentiment". Retrieved from <https://data.world/crowdflower/airline-twitter-sentiment>  
   Rai, Sawan. (2021). "An Empirical study of Machine Learning Classifiers with Tweet Sentiment Classification". https://www.analyticsvidhya.com/blog/2021/11/an-empirical-study-of-machine-learning-classifier-with-tweet-sentiment-classification/