REPORT

LIBRARIES USED:

- Numpy: Used for working on arrays. Useful in domains of algebra and matrices.
- Pandas: Used for analysis of data and manipulation of dataframes.
- Seaborn: Used to plot graphs and visualize data.
- Matplotlib: Used for plotting various types of visualizations.
- Textblob: Used to perform analysis and operations of text data and provides to divide text into NLP.
- Nltk: Used for processing text data, tokenization of data, classification, stemming etc.
- Wordclouds: Used to visualize frequent words in an unstructured data in a visual and expansive format.
- Sklearn: Used to perform regression, classification, clustering and dimensionality reduction of dataset.

DATASET USED:

I used the database of tweets about Virat Kohli.

CODE EXPLAINED:

1. Firstly, I checked for the consistency of the database by using .head(), .info(). I got a generic understanding of the data from these initial steps.
2. Then, I checked for null % in all columns to check which columns are significant to the result and which aren't.
3. I made sure the excess columns are removed and kept only the column that we are working on (tweets). I stored this data into a new dataframe.
4. I renamed tweets column to text for better understanding.
5. Defined a function named Data_Cleaning(). This function converts text to uniform lowercase and replaces all unwanted characters which contribute nothing to the sentiment calculation. The function also tokenizes the text (breaks up the entire chunk of text into small parts) and then adds the

tokens together into a string. Re is used here, it's used to work with regular expressions and was imported at the start of the code.

6. This function is applied to all rows in the dataframe using .apply() function over the entire Text column.
7. Duplicates are checked for and the similar rows are dropped using drop_duplicates().
8. Stemming function is declared. The function makes sure that reduces a word to it's stem data. This helps in data normalization and removes redundancy.
9. This stemming function is once again applied to all rows of dataframe. This time lambda is used inside apply function and it's called for each row.
10. Polarity function is declared and it is used to evaluate the sentiment of the text and provide a -ve score for negative sentiment, +ve score for positive sentiment and 0 for neutral statements.
11. Again, the polarity function is applied to the entire text column and a new column Polarity_value is created to store the evaluated polarity value for each corresponding row text.
12. Nature of data checked once again using .head() and .info().
13. An analysis_sentiment function is declared that provides output of "positive", "negative" or "neutral" based on polarity value.
14. A new column named Sentiment_value is added and the outputs obtained by applying sentiment function on each row in polarity column is stored in it.
15. Pie chart and bar chart are drawn to show the distribution of various entries in Sentiment_value column. These help to easily visualize the sentiment distribution of the data.
16. The neutral, positive and negative sentiment values data are filtered separately and sorted. The obtained 3 dataframes are used to obtain wordclouds for each particular activity.
17. Countvectorizer function is used to convert the text to vector form based on frequency of each data.
18. The data for regression analysis is prepared. X is the text column and Y is the Sentiment_value column.
19. The train and test data is separated from the dataframe and the ratio of train to test dataframe is 7:3 and randomized segregation is used for better results and avoid overfitting.
20. Regression analysis is performed and the predicted Y, actual Y values are compared using an accuracy score. The accuracy of this model turns out to be 86.85%.

21. Confusion matrix is plotted to evaluate the model. Confusion matrix is a N x N table (where N is the number of classes) that contains the number of correct and incorrect predictions of the classification model. The rows of the matrix represent the real classes, while the columns represent the predicted classes. Classification report is also evaluated which provides other constants which help for better understanding of accuracy of model.
22. Up next, we implement a grid search approach to improve our model's accuracy. GridSearchCV uses a fit and search method. It is a cross validation method that finds the optimal values for a given set of parameters.
23. Again, the confusion matrix and classification report is provided for the improved model. The accuracy is now 88.13%.
24. We apply linearSVC class on the model to improve it's accuracy. LinearSVC applies a linear kernel function to do classification and tries to find the best fit model by returning the best fit hyperplane.
25. After applying linearSVC, accuracy of model is 89.78%. The confusion matrix and classification report are also documented.
26. Finally, we use GridSearchCV to again fine tune the results. We use degree and gamma in addition to provide hyper-parameter tuning to the model. The confusion matrix and classification report are computed.
27. Finally, the accuracy of the model is 89.94%.