Word Cloud:

Word Cloud is a technique of visualizing text data for different size depending upon the frequency or importance. Word cloud technique is widely used to analyze data from social media. To implement a word cloud, we have used a library.

```
from wordcloud import WordCloud, STOPWORDS
```

From the above library,

- We have converted “STOPWORDS” into a list to add “sil” (silence) and “uh” common during the planned speech.

```
stop_list = list(STOPWORDS) + ["sil", "uh"]
```

- We have created an empty list and named it as “vocab_list”.

```
vocab_list = []
```

- In the above empty list, we have inserted all the words except the stopwords from Jason file.

```
    for tok in json_text["tokens"]:
        Text = tok["text"].lower()
        if Text not in stop_list:
            text += " " + Text
            counter += 1
            doc_vocab.add(Text)
            VOCAB.add(Text)
            vocab_list.append(Text)
```

- To count the frequency of words, we have implemented counter “dicter” on “vocab_list”.

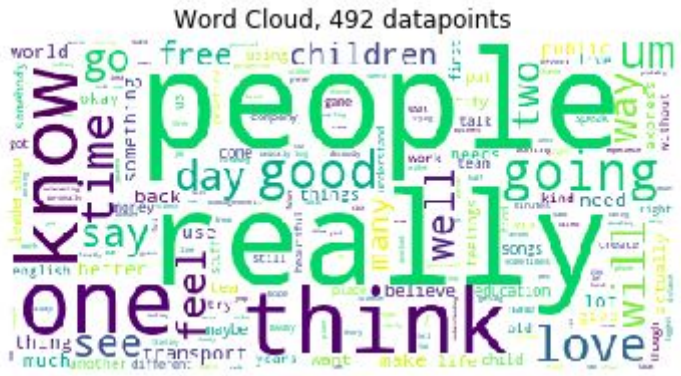
```
dicter = Counter(vocab_list)
```

- Now, we have set all the required parameter and have given “dicter” as input.

```
wordcloud = WordCloud(
    background_color='white',
    stopwords=stop_list,
    max_words=200,
    max_font_size=80,
    random_state=42
).generate_from_frequencies(dicter)

fig = plt.figure(1)
plt.imshow(wordcloud)
plt.title("Word Cloud, 492 datapoints")
plt.axis('off')
```

Result:



Random Forest Classifier: It is meta estimator that fits the numerous number of a decision tree on various sample size from a dataset. Average all of them to gradually improve and get the best accuracy with controlled over-fitting. We are using “Random Forest Classifier” for the classification because it uses group classifier instead of one to predict the target.

Implementation:
We have used the below-given library to split data in test and train along with the random seed. Data has been split into the 4 parts **(a_train, a_test, b_train, b_test). “a_train and a_test”** are containing train and test feature whereas “**b_train, b_test”** are cantaining selectors.
To apply Random Forest Classifier on the training and testing data, we have imported **“RandomForestClassifier”** from **“sklearn.ensemble”.** After importing classifier, we have fit the training data on it to further do the prediction and then find accuracy. To find the accuracy, we have used library **“metrics”** from **“sklearn”.**

Results:
After running algorithm, we got an accuracy of ____.

Naive Bayes Classifier: Niave Bayes algorithm is the simple and one of the effective algorithm used for the classification. It simplifies the metrics calculation by calculating the probabilities of the attribute for a given class independent of the values fro other attributes.

Implementation:
We have used the below-given library to split data in test and train along with the random seed. Data has been split into the 4 parts **(a_train, a_test, b_train, b_test). "a_train and a_test"** are containing train and test feature whereas "**b_train, b_test"** are containing selectors.
To apply Naive Bayes Classifier on the training and testing data, we have imported **"GaussianNB"** from **"sklearn.naive_bayes".** After importing classifier, we have fit the training data on it to further do the prediction and then find accuracy.To find the accuracy, we have used library **"metrics"** from **"sklearn".**

Results:
After running the algorithm, we got an accuracy of ____.

Comparison: Naive Bayes produce a better result as compared to the Random Forest classifier because the model size used by naive bayes is smaller and constant with respect to the data. So, if there is large dynamic data then there are no chances of occurrence of overfitting.Whereas in Random Forest model size is quite large and needs to be build carefully, otherwise it results in overfitting and low accuracy. Naive Bayes due to small model size easily adapts to the changes while random forest have to build forest every time.

Reference:
[1]J. Brownlee, "Naive Bayes Classifier From Scratch in Python", Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/. [Accessed: 07- May- 2019].

[2]"Naive Bayes Classification using Scikit-learn", DataCamp Community, 2019. [Online]. Available: https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn. [Accessed: 07- May- 2019].

[3]"How to decide when to use Naive Bayes for classification", Data Science, Analytics and Big Data discussions, 2019. [Online]. Available: https://discuss.analyticsvidhya.com/t/how-to-decide-when-to-use-naive-bayes-for-classification/5720. [Accessed: 07- May- 2019].

Gradient Boasting:

Gradient Boasting focuses consecutively to reduce error with each model until it gets the best model. Gradient Boasting uses weak learner to predict outcomes. This weak model we predict loss function which helps us to reduce error.

It uses function space for the optimization rather than in parameter space, which makes the use of custom loss function quite easier. It focuses on different examples step by step which helps it in learning, how to deal with unbalanced data. It helps us in

A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.