2022S AML 2304 Natural Language Processing

Final Project

Group Members:

Bhanu Prakash Mahadevuni C0850515

Deeksha Naikap C0835440

Pramod Reddy Gurrala C0850493

Sai Varun Kollipara C0828403

Instructor: Harriet Huang

Submission Date: 16-08-2022

Problem Statement:

- The task is to perform Machine Learning models on a news dataset provided. The task includes to perform multiple word vectorization methods.
- For the Solution proposed, we have considered 20 newsgroup data. We will be performing all the basic Data Analysis, Data Visualization, Machine Learning Model Building and Word Vectorization Methods. We will compare the results from multiple models with the vectorization methods and try identifying the best model for the text-based applications.

Steps Involved:

1. Importing Packages
2. Importing Dataset
3. EDA
    a. Data Acquisition
    b. Data Understanding
    c. Data Visualization
    d. Data Splitting
4. Word Vectorization
    a. Count Vectorizer
    b. TF-IDF Vectorizer
5. TSNE Visualizer
6. Model Building
    a. Evaluation Function
    b. Machine Learning Algorithms
    c. Comparing the ML Algorithms with multiple Variables

Prerequisites:

- Dataset – 20 Newsgroup Data
- Libraries – All the libraries for ML models, WV methods, basic analysis, data visualization, evaluation, and others.
- Word Vectorization Methods –
    - Count Vectorizer
    - TF-IDF Vectorizer
- Machine Learning Models –
    - Logistic Regression
    - SVC
    - Naïve Bayes

About Dataset:

- This dataset consists of news about 20 different groups, each of the record has a header, footer, data, target, target name and quote.

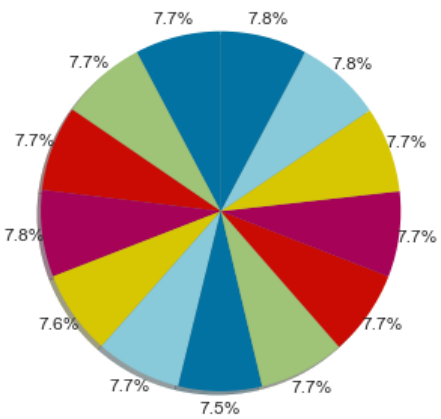
- And the dataset can be used as full, train or test subsets. Some functions are available for making it easy to work, like limiting the categories, removing some information about the record and compatible with ML models and functions.

EDA - Exploratory Data Analysis:

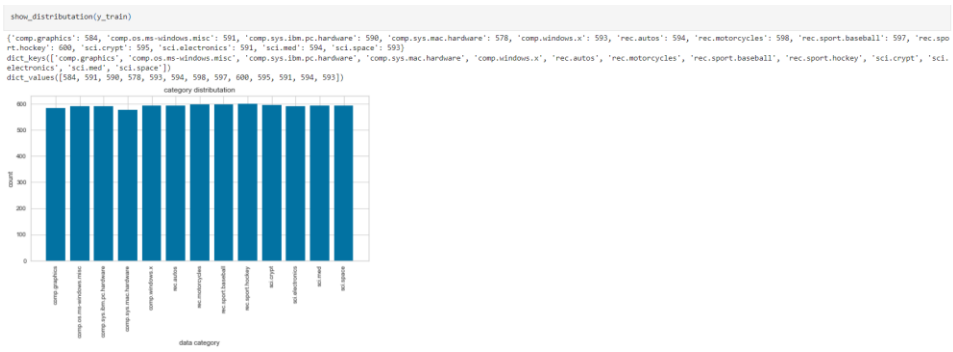
- **Data Acquisition** - We are collecting the data from the fetch_20newsgroups function. We have created the variables according to the subsets used. We used the categories attribute to limit the usage of the categories.
- **Data Understanding** - After cleaning the data, we have our finalized data we need to perform the necessary data analysis. In this step we are checking the details on the variables from test and train. Some of the details are listed below:
    - Target Name
    - Target
    - Length
    - Type of variable
    - Displaying the contents of variables
- **Data Visualization** - Visualizing the data helps in identifying the relation between the variables. There are multiple types of visualization types that can be implemented.
- **Data Splitting** - As all the basic pre-processing steps are done, now we will split the data into testing and training variables. Those will be used for the following:
    - Training the Model
    - Testing the Model
    - Validating the Model
    - Predicting the Model Performance
    - For TSNE Visualizing

```
news_groups_train = fetch_20newsgroups(subset='train', categories=category, shuffle=True, download_if_missing=False, remove=('headers', 'footers'))
news_groups_test = fetch_20newsgroups(subset='test', categories=category, shuffle=True, download_if_missing=False, remove=('headers', 'footers'))
```

Variables with the Dataset



Pie Chart on the Target Categories and Percentage of them



Distribution Function that displays all the information about the variable provided

Word Vectorization:

- A technique used in natural language processing (NLP) called word embeddings or word vectorization maps words or phrases from a lexicon to a corresponding vector of real numbers that can be used to identify word predictions and word similarity/semantics.
- The term "vectorization" refers to a traditional technique for transforming input data from its original text-based format into real-number vectors, which is the format that is supported by ML models. Some of the methods:
    - Count Vectorizer
    - Hashing Vectorizer
    - Word2Vec
    - Term frequency-inverse document frequency(TF-IDF)
    - Pointwise mutual information (PMI)
- There are many variables that are important for using vectorization functions. Some of them are listed here
    - max_df
    - min_df
    - stop_words

Count Vectorization:

- Count Vectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector based on the frequency (count) of each word that occurs in the entire text.
- We have built a variable using count vectorizer function. This is used for creating test, train, validate variables that are used in TSNE visualizer and Model Training and Comparing the Model Performance.

```
vectorizer = CountVectorizer(max_df=0.97, min_df=3,
                                max_features=None,
                                stop_words='english')
t0 = time()

vec_x_sp_train = vectorizer.fit_transform(x_sp_train)
vec_x_sp_val = vectorizer.fit_transform(x_sp_val)

vec_x_train = vectorizer.fit_transform(x_train)
vec_x_test = vectorizer.fit_transform(x_test)


print("done in %0.3fs." % (time() - t0))
```

```
done in 3.325s.
```

Count Vectorizer Function

```
print(vec_x_train)
```

```
  (0, 3830)     1
  (0, 22119)    1
  (0, 12947)    1
  (0, 15837)    2
  (0, 6350)     2
  (0, 22118)    1
  (0, 22474)    1
  (0, 24535)    1
  (0, 24502)    2
  (0, 7286)     1
```

Count Vectorizer Variables Representation

TF-IDF Vectorization:

- TFIDF works by proportionally increasing the number of times a word appears in the document but is counterbalanced by the number of documents in which it is present.
- We have built a variable using TF-IDF vectorizer function. This is used for creating test, train, validate variables that are used in TSNE visualizer and Model Training and Comparing the Model Performance.

```
tfidf_vectorizer = TfidfVectorizer(max_df=0.97, min_df=2,
                                max_features=None,
                                stop_words='english')
t0 = time()

tfidf_x_sp_train = tfidf_vectorizer.fit_transform(x_sp_train)
tfidf_x_sp_val = tfidf_vectorizer.transform(x_sp_val)

tfidf_x_train = tfidf_vectorizer.transform(x_train)
tfidf_x_test = tfidf_vectorizer.transform(x_test)

print("done in %0.3fs." % (time() - t0))
```

```
done in 3.013s.
```

TF-IDF Vectorizer

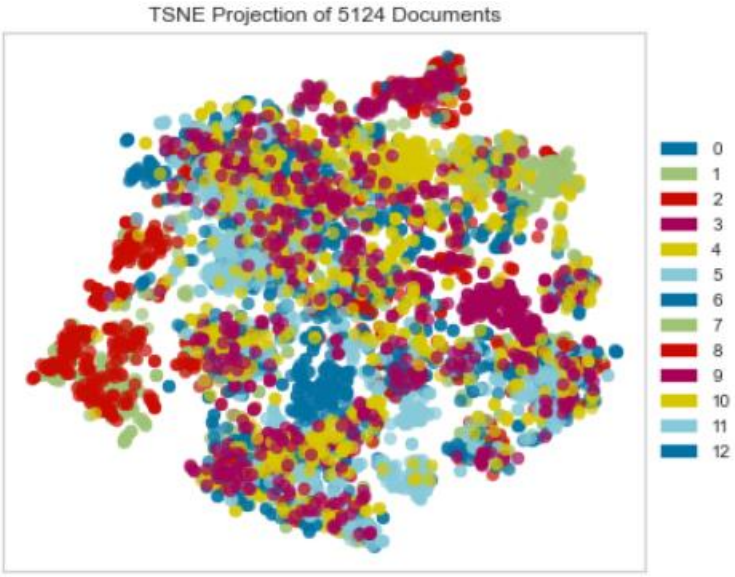
```
print(tfidf_x_train)
```

```
  (0, 30710)    0.04001673180719335
  (0, 30665)    0.22976010083140172
  (0, 29340)    0.09694991470444973
  (0, 29328)    0.05922738134750787
  (0, 29208)    0.17785183992820947
  (0, 28973)    0.17210658094624462
  (0, 28596)    0.14458275107538915
  (0, 28163)    0.1496840001410517
  (0, 28027)    0.060752630341837095
  (0, 27871)    0.3495984002025758
  (0, 27736)    0.17210658094624462
```

TF-IDF Variable Representation

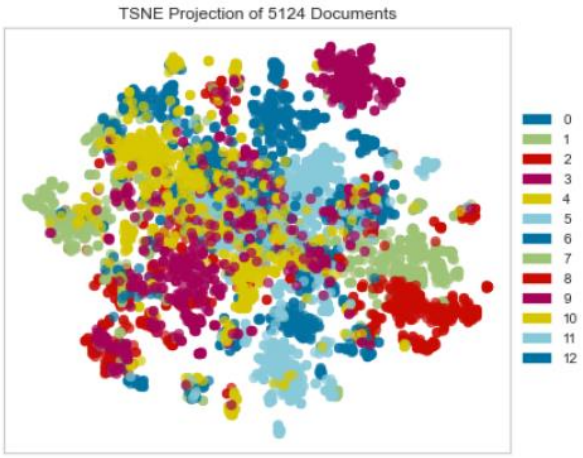
TSNE:

- t-SNE which stands for t distribution-Stochastic neighborhood embedding
- t-SNE is a Dimensionality Reduction algorithm
- Generally, t-SNE is a technique to turn a list high-dimensional vectors into a list of lower dimensional ones while keeping the relative similarity of things as close to the original as possible.
- It uses ML to try to create lower dimensional points, such that the pairs have a similar probability distribution across the pairs.




```
3]: <AxesSubplot:title={'center':'TSNE Projection of 5124 Documents'}>
```

TSNE Visualizer with Count Vectorizer Variables



7]: <AxesSubplot:title={'center':'TSNE Projection of 5124 Documents'}>

TSNE Visualizer with TF-IDF Variables

Model Building:

- In this project, we are working on multiple machine learning models. Some of them are listed here:
  - SVC – Support Vector Classifier
  - Logistic Regression
  - Naïve Bayes

```
params = {'C': [0.01, 1, 3]}
rs_lr = show_performance_with_gscv('LogisticRegression', LogisticRegression(penalty='l2'), tfidf_x_train, y_train, params)
print(rs_lr)
```

```
{'model_name': 'LogisticRegression', 'params': {'C': 3}, 'train_time': 11.449575848049589, 'val_time': 0.009639183680216 47, 'train_score': 0.9921408157963109, 'val_score': 0.8810080540400104, 'best_model': LogisticRegr
ession(C=3)}
C:\Users\gruha\miniconda3\envs\ash_off\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
params = {'C': [0.1, 1, 3], 'gamma':[0.5, 0.9]}
rs_svc = show_performance_with_gscv('SVC', SVC(kernel='linear'), tfidf_x_train, y_train, params)
print(rs_svc)
```

```
{'model_name': 'SVC', 'params': {'C': 3, 'gamma': 0.5}, 'train_time': 35.6962198416392, 'val_time': 9.355389833450317, 'train_score': 0.9966874512860483, 'val_score': 0.8781501688750325, 'best_model': SVC(C=3, gamma=0.
5, kernel='linear')}
```

```
params = {'alpha': [0.0001, 0.01, 0.5, 0.95]}
rs_nb = show_performance_with_gscv('NaiveBayes', MultinomialNB(), tfidf_x_train, y_train, params)
print(rs_nb)
```

```
{'model_name': 'NaiveBayes', 'params': {'alpha': 0.01}, 'train_time': 0.06782331069310506, 'val_time': 0.01878337065378825, 'train_score': 0.9911665367627956, 'val_score': 0.8893219017926736, 'best_model': MultinomialN
B(alpha=0.01)}
```

ML Models for TF-TDF Variables

```
params = {'C': [0.01, 1, 3]}
rs_lr_vec = show_performance_with_gscv('LogisticRegression', LogisticRegression(penalty='l2'), vec_x_train, y_train, params)
print(rs_lr_vec)
```

```
{'model_name': 'LogisticRegression', 'params': {'C': 1}, 'train_time': 19.06526494026184, 'val_time': 0.008976830376519097, 'train_score': 0.9962977396726421, 'val_score': 0.8307352559106261, 'best_model': LogisticRegr
ession(C=1)}
C:\Users\gruha\miniconda3\envs\ash_off\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
params = {'C': [0.1, 1, 3], 'gamma':[0.5, 0.9]}
rs_svc_vec = show_performance_with_gscv('SVC', SVC(kernel='linear'), vec_x_train, y_train, params)
print(rs_lr_vec)
```

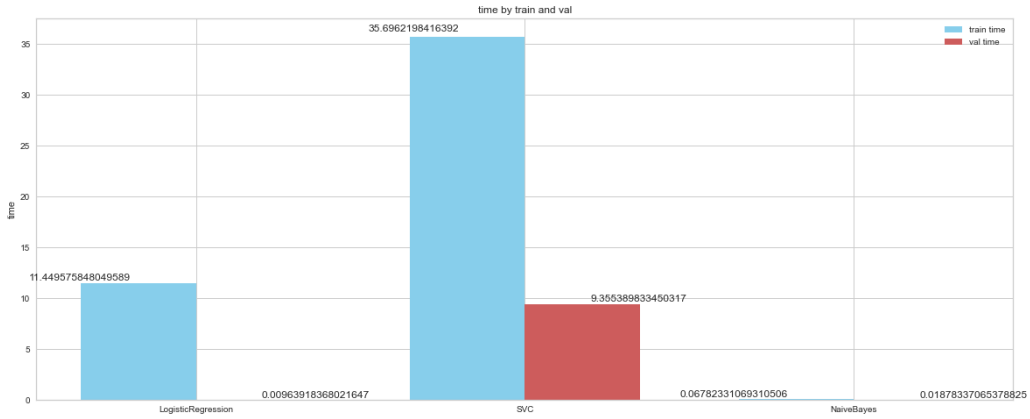
```
{'model_name': 'LogisticRegression', 'params': {'C': 1}, 'train_time': 19.06526494026184, 'val_time': 0.008976830376519097, 'train_score': 0.9962977396726421, 'val_score': 0.8307352559106261, 'best_model': LogisticRegr
ession(C=1)}
```

```
params = {'alpha': [0.0001, 0.01, 0.5, 0.95]}
rs_nb_vec = show_performance_with_gscv('NaiveBayes', MultinomialNB(), vec_x_train, y_train, params)
print(rs_nb_vec)
```

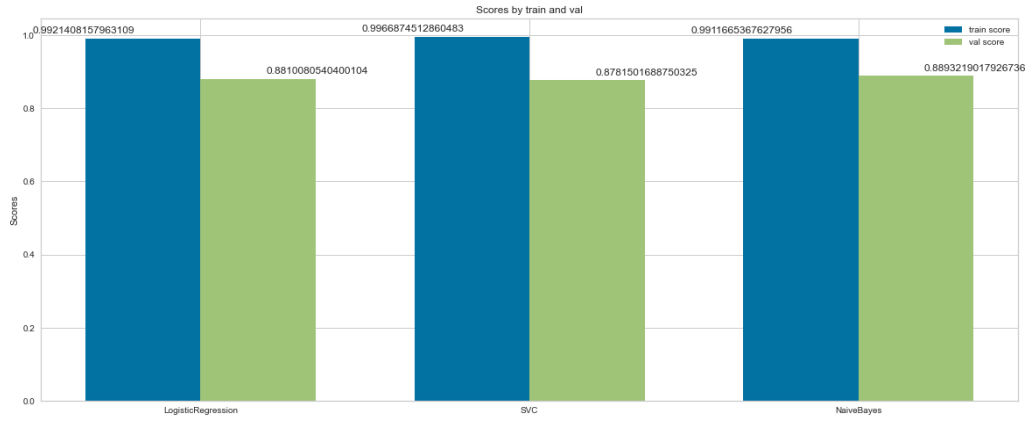
```
{'model_name': 'NaiveBayes', 'params': {'alpha': 0.0001}, 'train_time': 0.06167074044545492, 'val_time': 0.01567753156026204 3, 'train_score': 0.9562223954273837, 'val_score': 0.8437256430241621, 'best_model': Multinomi
alNB(alpha=0.0001)}
```

ML Models for Count Vectorizer Variables

Model Comparison:



Visualization For comparing the Time taken for the Models to Train



Visualization for comparing the mean-test-score for the Models

Conclusion:

- From the model building results, we can conclude that Naive Bayes Algorithm performs best for any type of Word Vectorization Methods.
- We can also understand that TF-IDF data variables were able to provide more validation score than Count Vectorization variables.
- We can also conclude that scores of the models should be high, and time taken for the models to train should be less.

Future Developments:

- We can perform additional tasks like
    - Removing Punctuation
    - Removing numbers
    - Removing Characters / Symbols
    - Performing Word2Vec Model
    - Performing Pointwise mutual information (PMI)

References:

1. https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html
2. https://neptune.ai/blog/vectorization-techniques-in-nlp-guide
3. https://thepoints.medium.com/feature-extraction-from-text-using-countvectorizer-tfidfvectorizer-9f74f38f86cc
4. https://www.quora.com/What-is-the-difference-between-TfidfVectorizer-and-CountVectorizer-1
5. https://www.scikit-yb.org/en/latest/api/text/tsne.html
6. https://www.tutorialspoint.com/scikit_learn/index.htm
7. https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/
8. https://medium.com/@siyao_sui/nlp-with-the-20-newsgroups-dataset-ab35cd0ea902
9. https://www.researchgate.net/figure/A-text-snippet-from-the-20-Newsgroup-dataset-The-transparent-part-represents-the-local_fig1_339857996