Documentation

Notebook

Our data headline does not exist, so we gonna label by our own

```
# We could see that some columns are not totally useful for us, So I gonna drop them to clean
```

We could conclude, that we have 4 labels

1. Neutral
2. Positive
3. Negative
4. Irrelevant

and encoded them into

Neutral'= '0'

'Positive' = '1'

'Negative' ='-1'

'Irrelevant' = '2'. Then,

Checking if there’s any missing null value.

Train data has missing null values. Fixed it

We have no missing null values in val_data. So, we could

leave them just the way it is.

Raw Data Cleaning

1. Labeling the data headline, dropping unuseful columns,
2. dropping duplicate data
3. One-hot encoding, standardization
4. Null missing value

Data preprocessing for twitter text

Manipulate our text feature in 7 methods for twitter analysis in order to remove noise, clean, and ready to train the model as a helpful feature, we have done, these 7 data cleaning methods for this data.

1. Replace space
2. Tokenization
3. stop words
4. remove links
5. remove the Twitter user (acc username)
6. remove punctuation
7. remove emoji

As I fancy the best result, I'm also tried to implement a Named_Entity_Recognition_(NER) model that I trained last week(Please check here). That detects Names which is useful for this Twitter sentiment analysis. We could detect names and remove them from our data.

Exploring data nature.

Let's dive into Data Exploration

- *Pie charts of the 4 label frequency*
- Word Frequency

And then, Prepare for Modeling/training. Later on,

I did 4 algorithms as below:

Evaluation

1. Multinimional Naive Bayes

```
MultinomialNB()
Accuracy score:  0.6667867921978468
F1 score:  0.6570664405036415
recall 0.6667867921978468
Precision:  0.6364687639517542
Confusion Matrix:
 [[1599   86  180  162]
 [ 944 5522 1174  955]
 [ 409  436 3015  444]
 [ 957  600 1050 4666]]
```

2. Random Forest

```
RandomForestClassifier(n_jobs=-1, random_state=300)
Accuracy score:  0.881616289022028
F1 score:  0.8817911883486355
recall 0.881616289022028
Precision:  0.873596216232276
Confusion Matrix:
 [[3156   60   77   89]
 [ 206 6019  211  268]
 [ 124  143 4660  134]
 [ 423  422  471 5736]]
```

3. Decision Tree

```
DecisionTreeClassifier(random_state=10)
Accuracy score:  0.7913869994143881
F1 score:  0.7912600288746594
recall 0.7913869994143881
Precision:  0.7826887723987173
Confusion Matrix:
 [[2779  180  206  286]
 [ 325 5440  390  438]
 [ 247  365 4163  317]
 [ 558  659  660 5186]]
```

4. Logistic Regression

```
LogisticRegression(max_iter=600, random_state=10)
Accuracy score:  0.7247173296094419
F1 score:  0.7214644012217224
recall 0.7247173296094419
Precision:  0.7064360570442305
Confusion Matrix:
 [[2227  209  264  294]
 [ 620 5575  859  746]
 [ 434  425 3603  504]
 [ 628  435  693 4683]]
```

[Model Notebook]()