Hotel Reviews Analyzer

Application Goal

- Help categorize a received review into Good, Neutral or Bad
- Track the trend over time of a hotel based on the reviews received

Dataset source

Source: https://www.kaggle.com/datafiniti/hotel-reviews
Attribute : 25
Dimension : 30000 review

Attribute
Id
dateAdded
dateUpdated
Adress
Categories
primaryCategories
City
Country
Keys
Latitude
Longitude
Name
postalCode
Province
Review.date
Review.rating
Review.link
Review.title
review.userName
review.userCity

How we labeled our data

We based the label phase on ratings assign for each review.

- 4-5 stars→ Good
- 3 stars → Neutral
- 1-2 stars → Bad

Dataset situation in Training Set...

After the label phase we were able to see how was our distribution data.
We noticed that our training set was very unbalanced.
In particular we had:

- Almost 17.000 Good reviews;
- Over 2.000 Neutral reviews;
- Over 2.000 Bad reviews but more than the Neutrals

. . . till the Undersample phase

Since that our training set was very unbalanced we did an undersampling on training set

- In particular:

  8072 comment labelled:

  2711 Rating Good

  2711 Rating Neutral

  2653 Rating Bad

Cleaning phase

In order to obtain a cleaned text to pass to Preprocessing phase we did steps like:

[The room was amazing and our room was in front of Central Park!!! @centralpark]

1. Lower case words
   [the room was amazing and our room was in front of central park!!! @centralpark]
2. Remove punctuation from sentences
   [the room was amazing and our room was in front of central park @centralpark]
3. Remove mention @ or Hashtags from sentences
   [the room was amazing and our room was in front of central park centralpark]
4. Remove link http from sentences
5. Remove emoji from sentences

Preprocessing phase

In this phase we took the cleaned text coming from the previous one and we did:

1. Tokenization
2. Stop word filtering
3. WordNet Lemmatizer
4. Lemmatizer filtering

Feature extraction technique - TFIDF + NGRAM

For our proposal we used a **Term Frequency - Inverse Document Frequency** in order to get the relevance of the single token of the sentence.

We combined the TF-IDF with the N-GRAM technique that predicts the probability of a given N-gram within any sequence of words in the language.

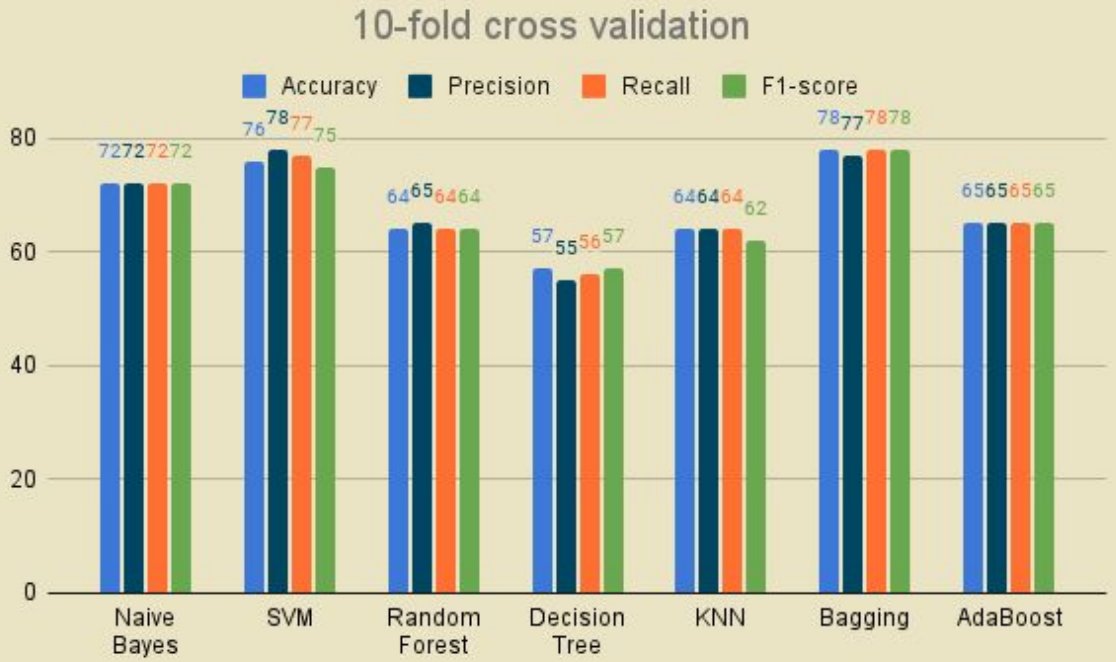
The following sparse matrix is a compressed view of the matrix that we obtained for NGRAM=1.

	zealand	zephyr	zero	zion	zip	zone	zoo	zorro	zu
0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
...	...	...	...	...	...	...	...	...	...
8070	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
8071	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
8072	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
8073	0.0	0.0	0.0	0.0	0.0	0.0	0.268154	0.0	0.0
8074	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0

Classifiers Evaluations with k-fold

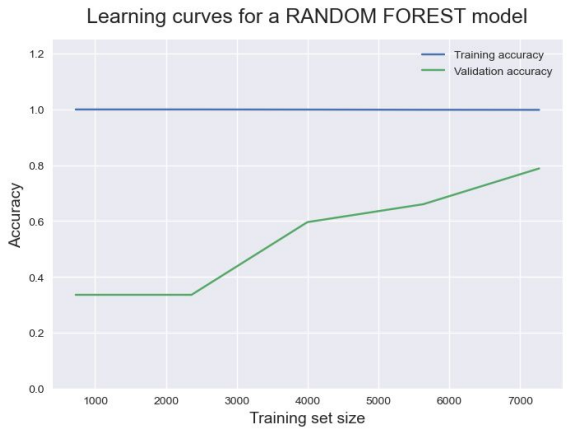
- AdaBoost
- Bagging
- Multinomial Naive Bayes
- Decision tree
- K-NN
- Random Forest
- SVM

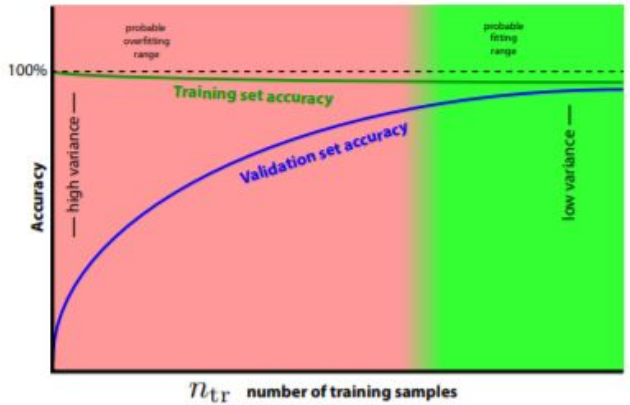
Classifiers comparison:



- T-test between two best classifiers

Classifiers overfitting

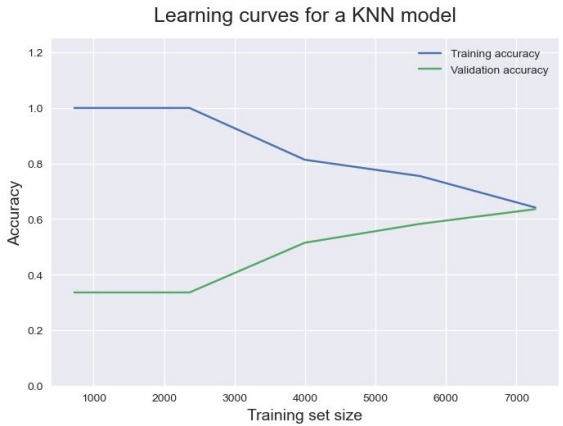


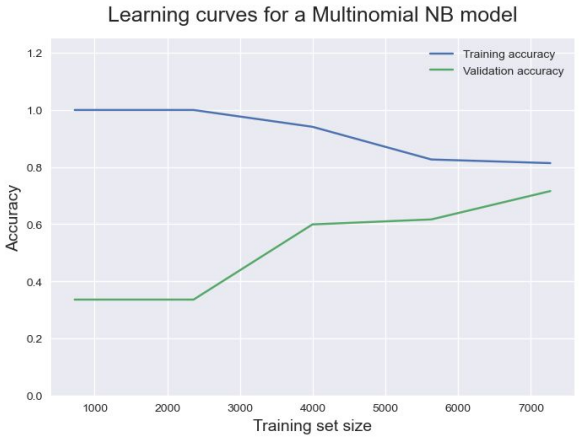


From a preliminary analysis on Accuracies' curves we observed that **Decision Tree** and **Random Forest** are in an overfitting situation. This because the training is constantly 1 and there is a big gap between the two curves.
Observing the third figure we are in the red window. This means that these twe are in overfitting range

https://people.ece.uw.edu/bilmes/classes/ee511/ee511_spring_2020/overfitting_underfitting.pdf
(Tom Mitchell, Machine Learning, McGraw-Hill Science/Engineering/Math; 1997)

Accuracies' curves of the other classifiers





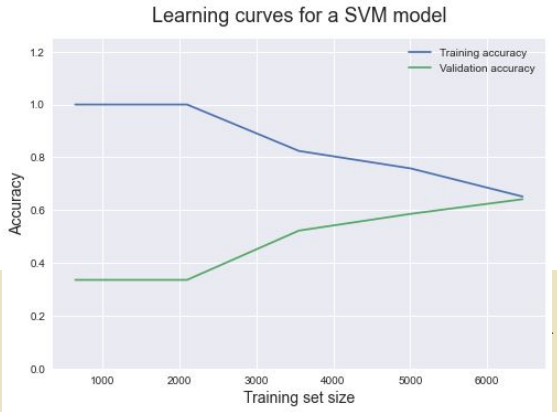
In this three classifiers the overfitting situation is not present due to the curves configuration

Best classifiers from K Fold

- Basing only on accuracy score isn't always a good practice.

- The classifiers that have best balance scores in terms of Accuracy, Precision, Recall and F1 are **Bagging** and **SVM**

- This means that models are somehow "balanced", that is, its ability to correctly classify positive samples is same as its ability to correctly classify negative samples.



T-test for comparison

We used the t test to compare the two best classifiers obtained from the k fold. And these are the results:

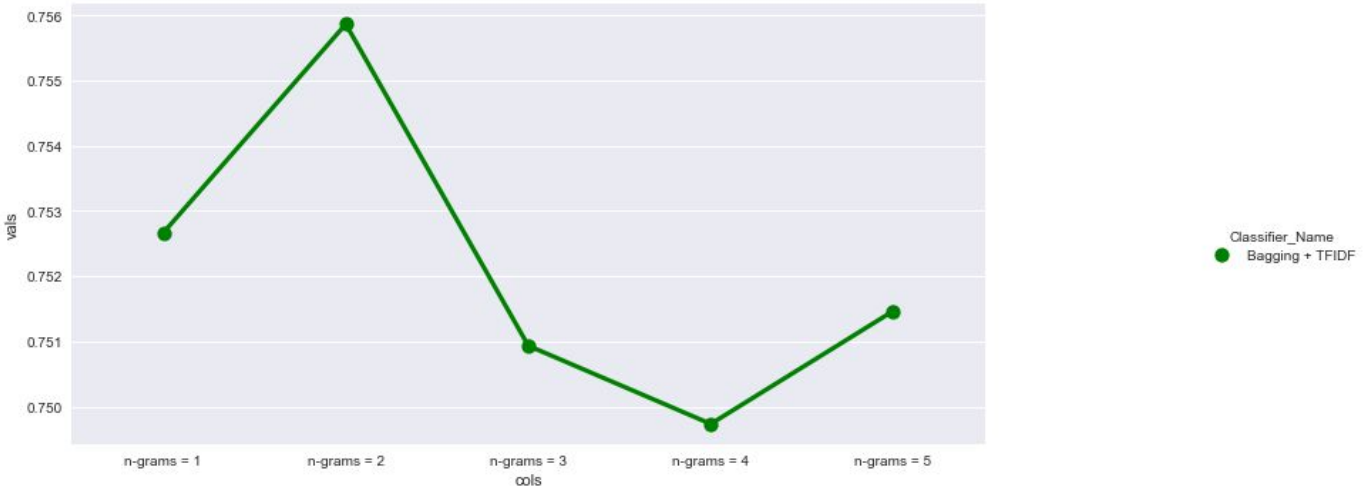
```
from mlxtend.evaluate import paired_ttest_kfold_cv

t, p = paired_ttest_kfold_cv(estimator1=clf,
                             estimator2=bgclassifier,
                             X=X_train_tf,
                             y=np.ravel(df_train_y.astype(int)),
                             random_seed=1)

print('p value: %.6f' % p)
```

```
p value: 0.000498
```

Since $p<\alpha$ (=0.05), we can accept the Alternative Hypothesis. This means that we take the best classifier between the two to build our application. **Bagging Classifier**

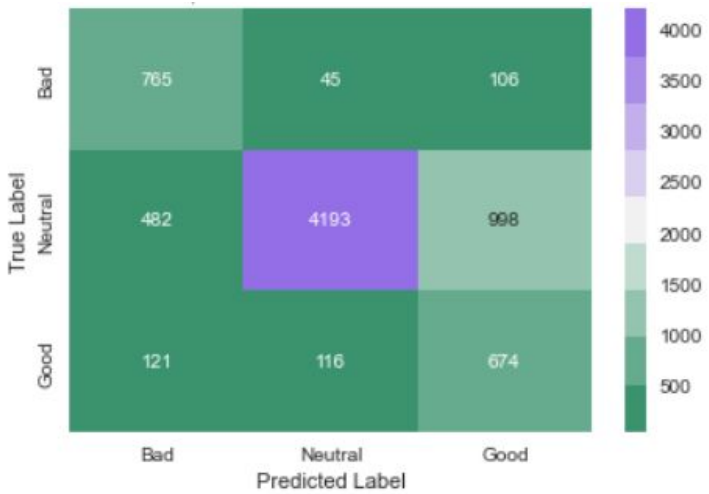
N-grams for Bagging

We tried to use different N-Grams to realize if our classifier accuracy could have been improved



With N-Gram = (1,2) the performance of **Bagging** increases.

Bagging for model construction



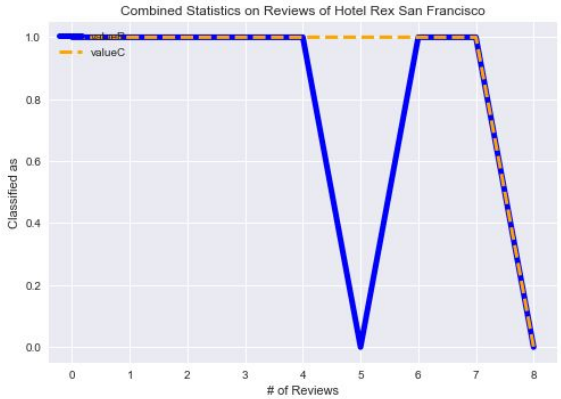
The Classification report:

	precision	recall	f1-score	support
Bad	0.56	0.84	0.67	916
Good	0.96	0.74	0.84	5673
Neutral	0.38	0.74	0.50	911
accuracy			0.75	7500
macro avg	0.63	0.77	0.67	7500
weighted avg	0.84	0.75	0.78	7500

Accuracy: 0.75

Hotel tranding differences with the rating

Assuming the graph given by the rating (BLUE) as a real trend, the trend calculated with our classifier (ORANGE) doesn't differs so much.



APPLICATION

Evaluation: Classify the posted review as good, bad or neutral

Hotel 1: Shows the hotel's trend over time, based on the classification of related comments

Hotel 2: Shows the hotel's trend over time, based on the classification of related comments

Hotel 3: Shows the hotel's trend over time, based on the classification of related comments