UNIVERSITÀ DI PISA

DATA MINING AND MACHINE LEARNING

< Hotel reviews Application REPORT >

Project made by**:**

- **Domenico Armillotta -** badge #: - email: d.armillotta@studenti.unipi.it
- **Leonardo Bellizzi -** badge #: 643019 email: l.bellizzi@studenti.unipi.it

Summary

INTRODUCTION

Our application aims to classify hotel reviews, to plot a time graph and view the trend over time.

The plot is given by the analysis of user comments made on the various online booking sites.

The hotel could therefore understand based on the trend which were the worst days and understand why.

DATASET DESCRIPTION

Our dataset comes from kaggle. The dataset collects all the reviews of various hotels from various online sites, therefore from different sources.

Link:
https://www.kaggle.com/datafiniti/hotel-reviews?select=Datafiniti_Hotel_Reviews.csv

and these are the characteristics:

Attribute : 25
Dimension : 30 000 reviews

Attributes
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

DATASET SPLITTING

Dataset was split in 75% of training and 25% of test data

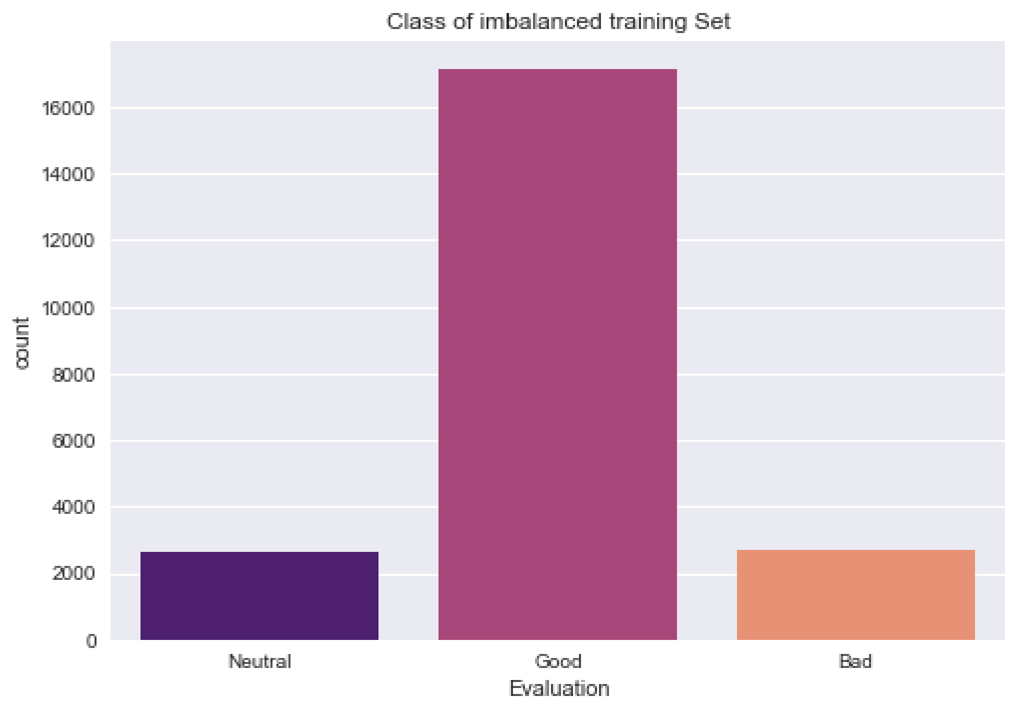
LABEL PHASE

We have labeled our comments following this pattern:

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

TRAINING SET COMPOSITION

After the label phase we were able to see how was the distribution of the data for the training set.



We noticed that our training set was very unbalanced.

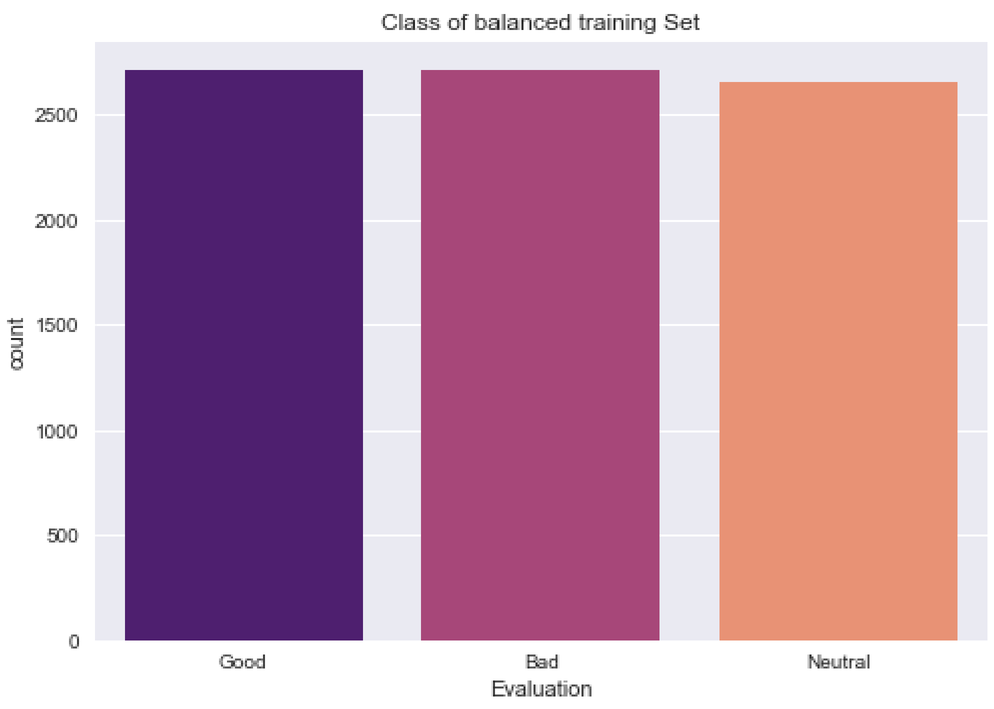
In particular we had:

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

Since that our training set was very unbalanced we did an **undersampling** on training set. In particular:

8072 comment labeled:

- 2711 Rating Good
- 2711 Rating Neutral
- 2653 Rating Bad



CLEANING AND PREPROCESSING PHASE

Dataset Cleaning

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]

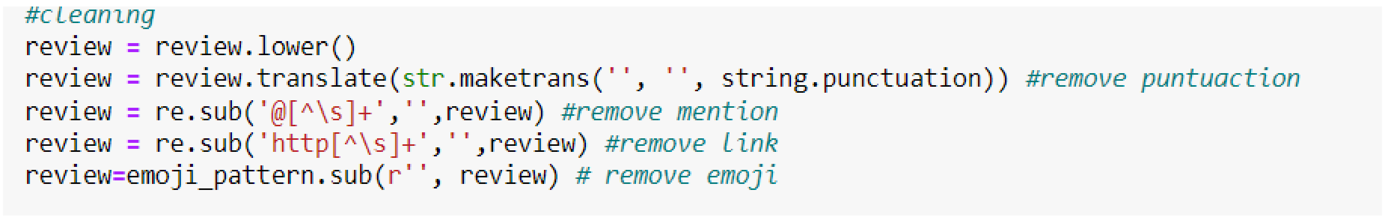
2. Remove punctuation from sentences

   [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



```
#cleaning
review = review.lower()
review = review.translate(str.maketrans('', '', string.punctuation)) #remove puntuaction
review = re.sub('@[^\s]+','',review) #remove mention
review = re.sub('http[^\s]+','',review) #remove link
review=emoji_pattern.sub(r'', review) # remove emoji
```

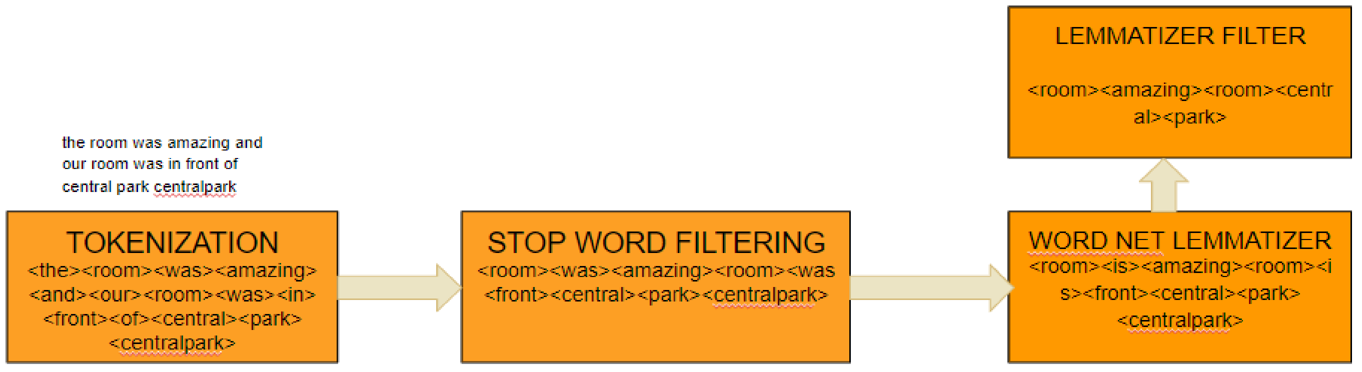
Text Elaboration

We performed pre-processing steps to transform our comments into TF IDF representation in Python. To carry out them, we used the CountVectorizer function from sklearn.

- **Tokenization**: in this first step we transformed a stream of characters into a stream of processing units called tokens. In this way each text is represented as a set of words. To do this we set some parameters of the CountVectorizer function
- **Stop Words Filtering**: in this second step we removed the stop words (which provide little or no useful information to the text analysis) setting the stop_words parameter equal to 'english
- **Lemmatization**: is the process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form.Unlike stemming, lemmatization depends on correctly identifying the intended part of speech and meaning of a word in a sentence. To do this task we use the library nltk.

-> better : good

- **Lemmatization Filtering**: in this fourth step we reduced the number of stems maintaining only the most relevant ones. We set max_features=5000 as CountVectorizer's parameter which considers the top max_features ordered by term frequency across the corpus.



the room was amazing and our room was in front of central park centralpark
TOKENIZATION
<the><room><was><amazing><and><our><room><was><in><front><of><central><park><centralpark>
STOP WORD FILTERING
<room><was><amazing><room><was<front><central><park><centralpark>
WORD NET LEMMATIZER
<room><is><amazing><room><is><front><central><park><centralpark>
LEMMATIZER FILTER
<room><amazing><room><central><park>

TF-IDF & N-GRAMS

TF-IDF was used as a feature extraction technique. Was used to measure the importance of a term with respect to a document or collection of documents. This function increases proportionally to the number of times the term is contained in the document, but increases inversely with the frequency of the term in the collection. The idea behind this behavior is to give more importance to the terms that appear in the document, but which in general are infrequent

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
count_vect=CountVectorizer( max_features=5000,lowercase=True, analyzer='word', stop_words= 'english',ngram_range=(1,1))
X_train_counts=count_vect.fit_transform(raw_documents=train_X)

tfidf_transformer=TfidfTransformer()
X_train_tf=tfidf_transformer.fit_transform(X_train_counts)

X_test_counts=count_vect.fit_transform(raw_documents=test_X)
X_test_tf=tfidf_transformer.fit_transform(X_test_counts)
```

With TF-IDF the N-GRAM was used. N-grams are continuous sequences of words or symbols or tokens in a document. In technical terms, they can be defined as the neighboring sequences of items in a document.

CLASSIFIERS USED

Classifiers that have been used are:

- Multinomial Naive Bayes;
- Support Vector Machine;
- Random Forest;
- Decision Tree;
- KNN;
- Bagging;
- AdaBoost.

CROSS VALIDATION

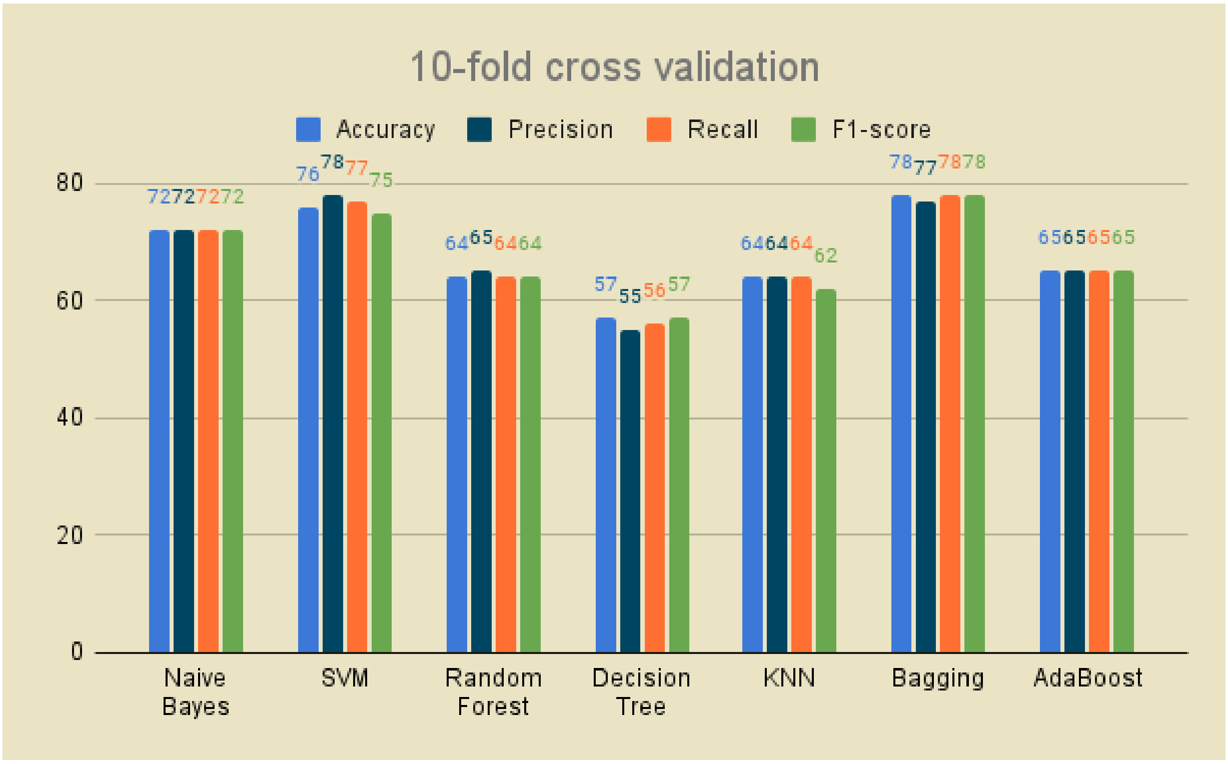
Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. In particular it was used to figure out the performance of the classifiers and understand if classifiers were overfitting and detect the best one.

We used the K-FOLD and not the STRATIFIED K-FOLD because our data are balanced.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. In our case the chosen K is equal to 10

Cross-validation was primarily used to estimate the skill of a machine learning model on unseen data.

Here the results of our classifiers under the 10-Fold Cross Validation.

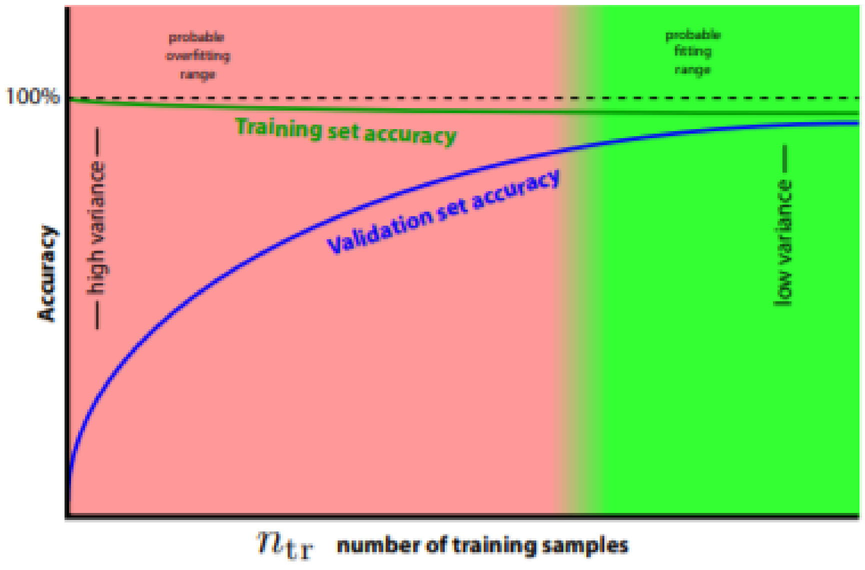


Overfitting classifiers

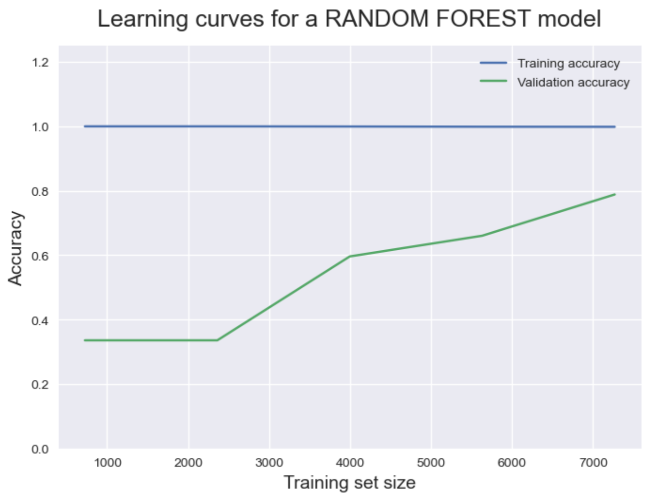
We tried to modify the parameters to get better results, but plotting the learning curve, we got suspicious seeing the training accuracy so far from the validation accuracy, and doing some research we realized that two of our classifiers were overfitting. So we changed the parameters to try to avoid overfitting.

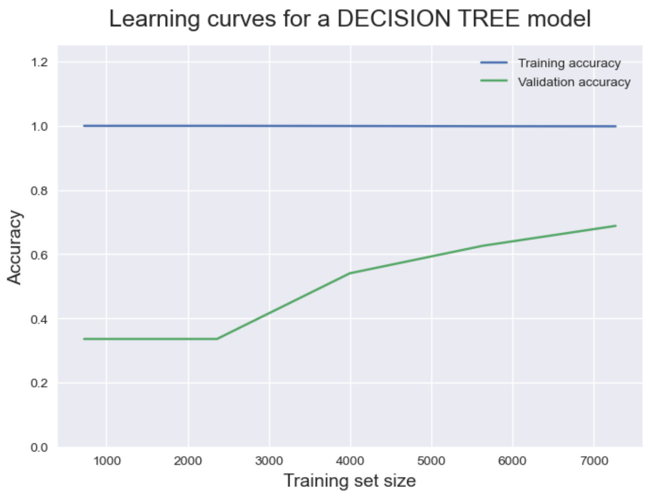
The tuning work for RANDOM FOREST even if we lost some accuracy score but DECISION TREE was still in an overfitting situation.

According to **Tom Mitchell, Machine Learning, McGraw-Hill Science/Engineering/Math; 1997**, the Overfitting situation is caught in the following scheme:



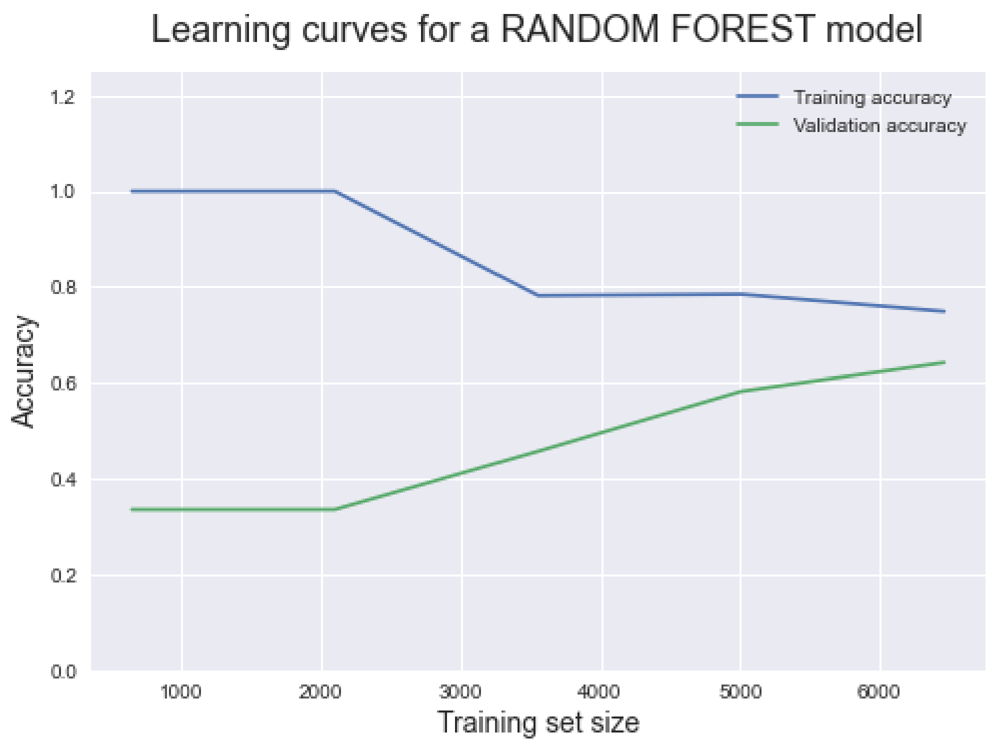
Random Forest and Decision Tree are in this situation:





These two classifiers are in the red window, this means that are in an overfitting range.

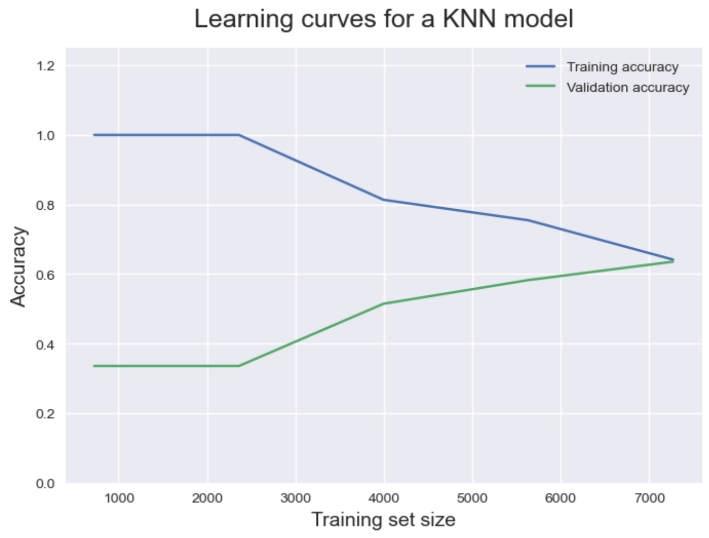
https://people.ece.uw.edu/bilmes/classes/ee511/ee511_spring_2020/overfitting_underfitting.pdf

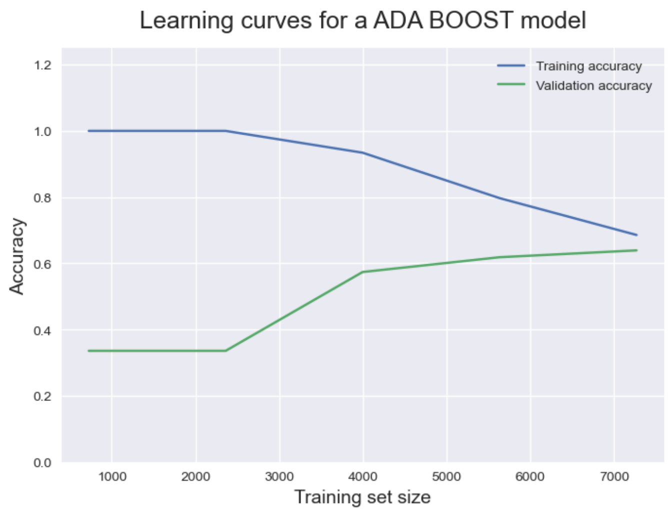


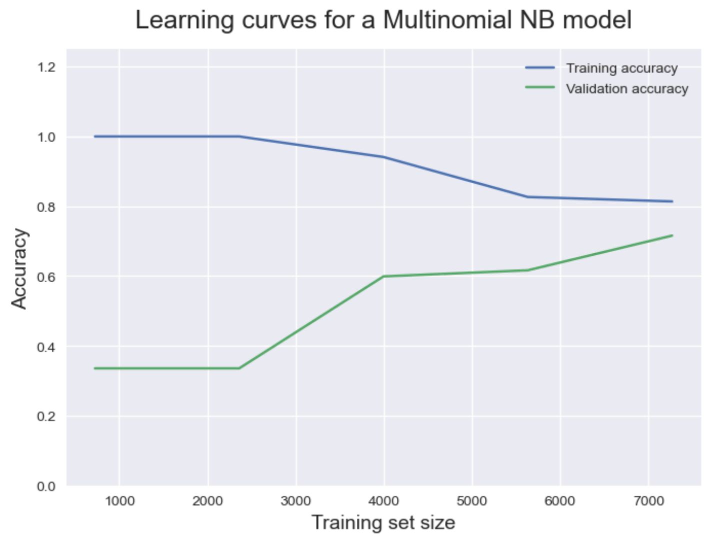
Random Forest now is not anymore overfitting but how predictable its accuracy decreased.

Classifiers not in overfitting

For the other classifiers the accuracies show us these results:





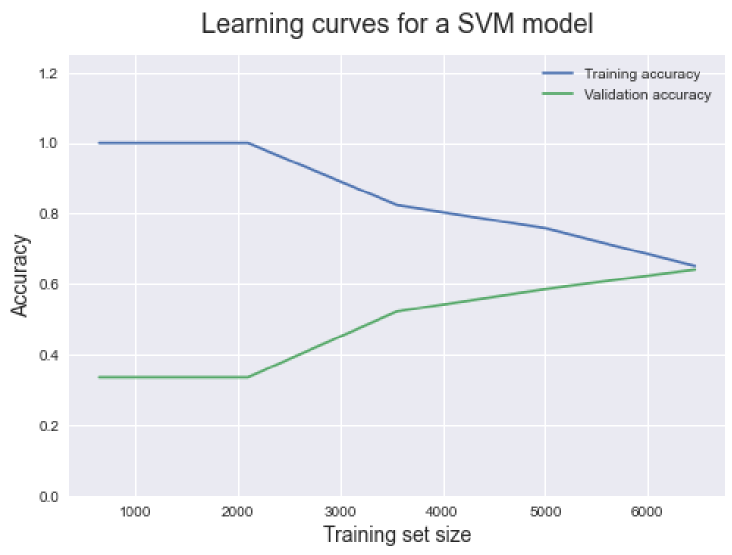


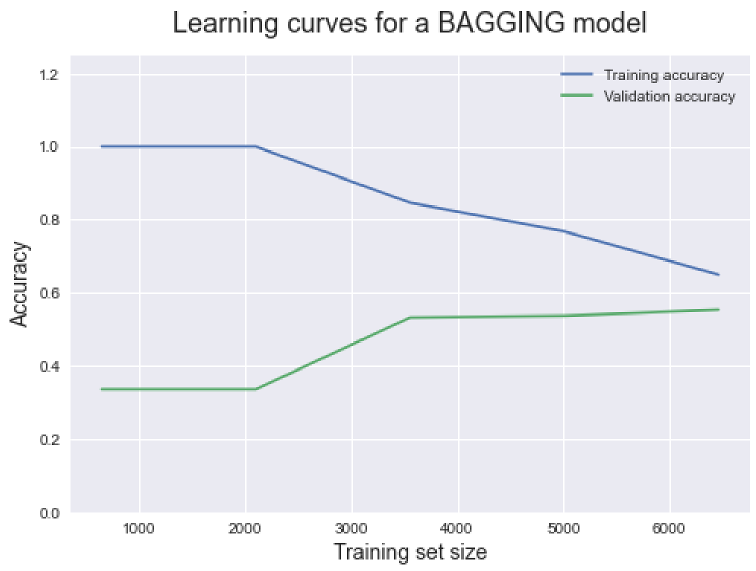
Best classifiers from K-Fold cross validation

Basing only on accuracy score isn’t always a good practice. Accuracy give us only a general idea on how the classifiers worked.

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

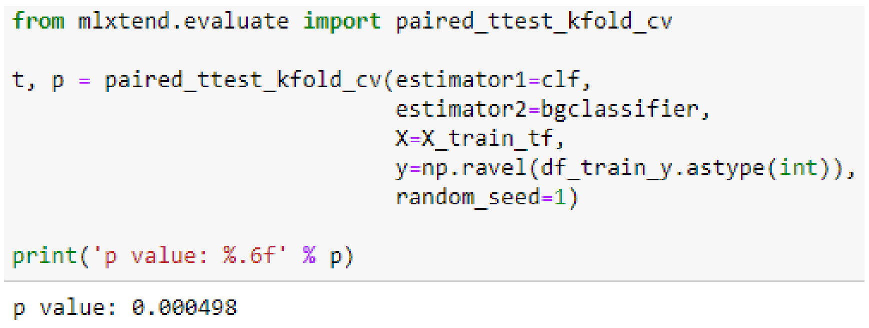
Here the learning curves of the two best classifiers:





T-TEST TWO PICK UP THE BEST

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



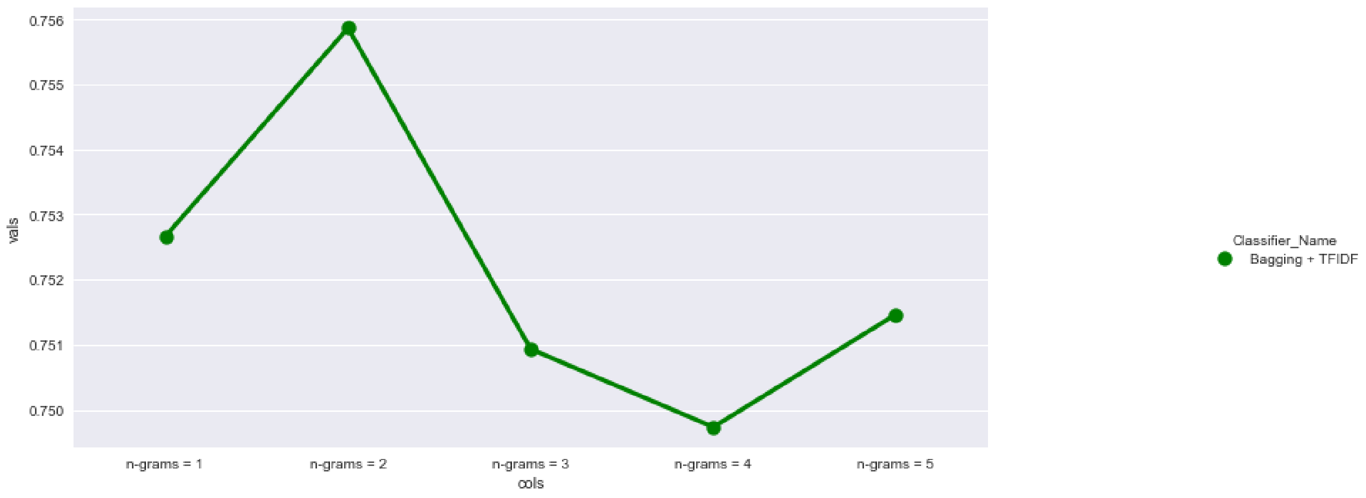
```
from mlxtend.evaluate import paired_ttest_kfold_cv

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<α (=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**

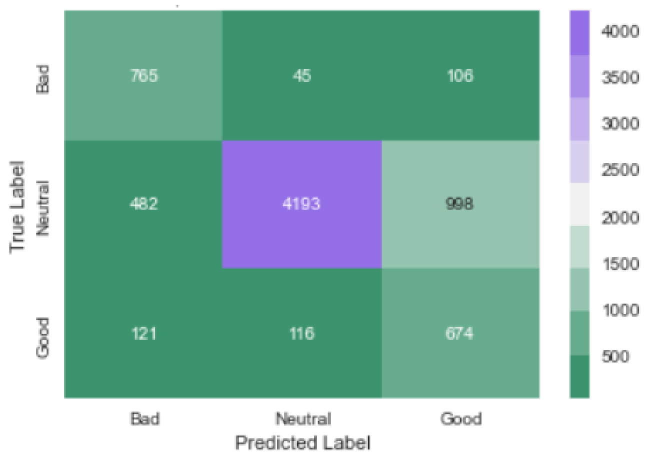
BAGGING CLASSIFIER INSIGHT

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.

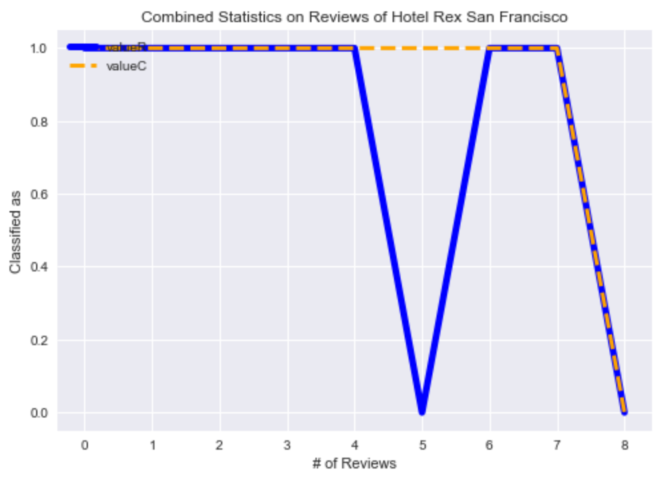
Here is the confusion matrix of the Bagging Classifier.

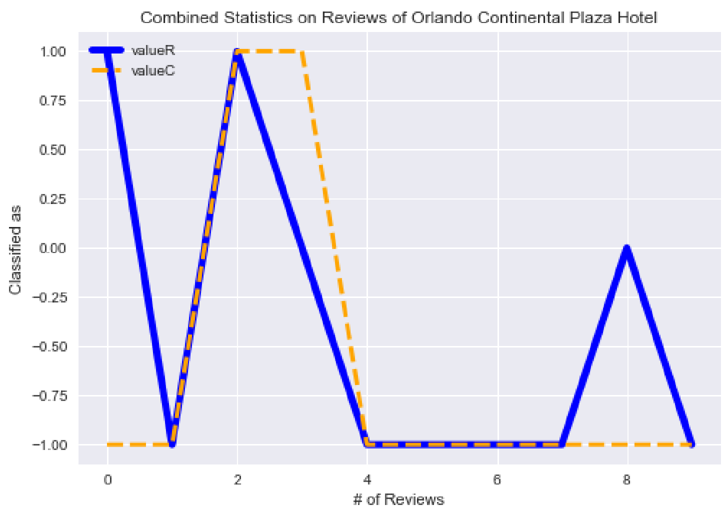


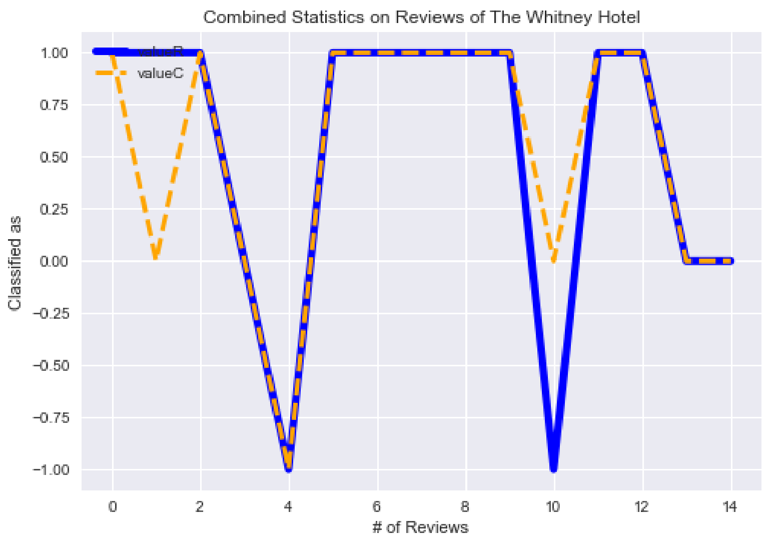
```
The Classification report:
              precision    recall  f1-score   support

         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
```

TRENDING DIFFERENCES BETWEEN REAL AND CLASSIFIED

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







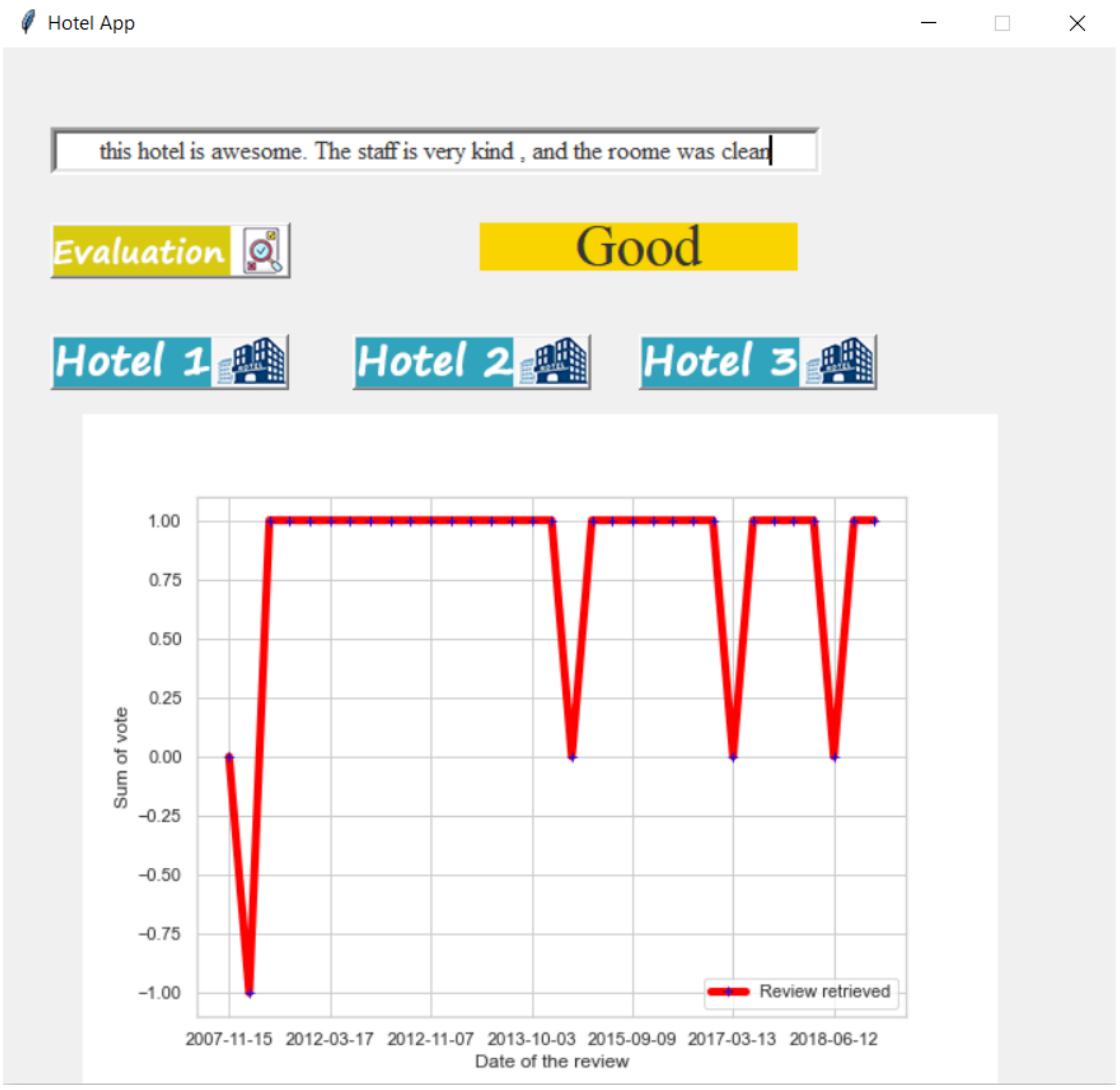
APPLICATION

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

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

Hotel 2 (Hotel Rex San Francisco): Shows the hotel's trend over time, based on the classification of related comments

Hotel 3 (Orlando Continental Plaza Hotel): Shows the hotel's trend over time, based on the classification of related comments



this hotel is awesome. The staff is very kind , and the roome was clean