UNIVERSITÀ DI PISA

Data Mining and Machine Learning

Recysis: Recipes Analysis

Application developed by

Cantini Irene, De Filomeno Elisa

Index

1. Introduction 3
2. Dataset 4
   2.1 Dataset description 4
   2.2 Dataset cleaning 5
   2.3 Data reduction 6
      2.3.1 Dimensionality reduction 6
      2.3.2 Data reduction 6
   2.4 Training set building 7
3. Classification 8
   3.1 Text elaboration using Python 8
   3.2 Multinomial Naïve Bayes 9
   3.3 Decision Tree Classifier 10
   3.4 Linear SVC (Support vector Classifier) 11
   3.5 K-Neighbors Classifier 13
   3.6 Random Forest Classifier 14
   3.7 Comparison 15
4. Streaming Analysis 16
   4.1 Event 1 - 13/04/2009 17
   4.2 Event 2 – 04/10/2009 18
   4.3 Event 3 – 03/01/2010 19
   4.4 Event 4 – 11/04/2010 20
   4.5 Event 5 – 06/06/2010 21
   4.6 Event 6 – 07/11/2010 22
   4.7 Conclusions 23
5. APPLICATION 25
   5.1 Preparing Data 25
   5.2 Develop application 26

1. Introduction

Recipes Suggester is an application where users can find many kinds of recipes posted by other users and leave comments about those recipes.

Italians' passion for cooking activities is known all over the world and, in the last two years, the global pandemic has further strengthened it. Many studies have demonstrated that more than 50% of Italian people have improved their culinary skills to cope with the restrictions imposed by the pandemic and at the same time to find a form of home entertainment. To improve these skills, people have started to follow cooking programs and recipe sites, therefore they felt the need to share their opinion and experience about it.

The purpose of our project is to develop a "sentiment analysis" able to label as positive/negative or neutral all the comments related to a recipe. This process will provide instant feedback about the community's opinion.

For each recipe the application shows the number of positive, negative and neutral opinions and for each comment it shows the associated sentiment.

All the codes and files are available at https://github.com/IreneCantini/Recysis/tree/main

2. Dataset

2.1 Dataset description

The dataset we used in our application is obtained by the following URL: https://www.kaggle.com/irkaal/foodcom-recipes-and-reviews .

The dataset contains 1.401.982 comments related to 522.517 recipes and it is composed of scraped data from Food.com which is a recipe website where users can upload their dishes and leave comments.

We focused our attention on the dataset containing the information about the reviews that users left on recipes.

Reviews.csv is composed of 8 attributes:

- ReviewId
- RecipeId
- AuthorID
- AuthorName
- Rating (from 0 to 5)
- Review (contains the text of the comment)
- DateSubmitted
- DateModified

2.2 Dataset cleaning

In this phase we clean the dataset using some python scripts.

- We remove the rows in which the comment's text isn't present because useless for our purpose.

Code used:

```
import pandas as pd


df = pd.read_csv("../dataset/reviews.csv")  # read original dataset
filtered_df = df[df['Review'].notnull()]    # remove rows with null review's value
filtered_df.to_csv(r'../dataset/review_notNull.csv', index=False)    # save new dataset
```

- We remove characters that represent the end of the line and multiple spaces into the comments' text.

Code used:

```
import pandas as pd


def modifyStr(x):
    x = x.replace('\r', '')
    x = x.replace('\n', '')
    x = x.replace('  ', '')
    x = x.replace('&', 'and')
    return x


df = pd.read_csv("../dataset/review_notNull.csv")  # read original dataset
df['Review'] = df['Review'].apply(modifyStr)
df.to_csv(r'../dataset/modified_comments.csv', index=False)    # save new
dataset
```

2.3 Data reduction

2.3.1 Dimensionality reduction

In this phase we remove not important attributes to allow easier visualization of the data. To make our sentiment analysis on the recipes' comments we consider only two attributes:

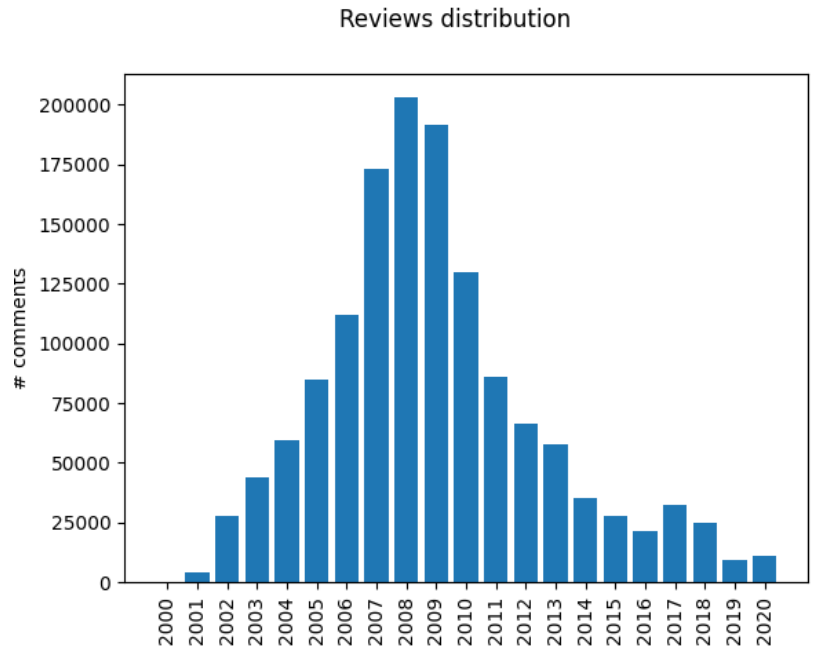
- **Rating**: we will use this attribute to divide comments into three classes establishing a ground truth.
- **Review**: this attribute maintains the text of the comment that we will use to build some classifiers.

For this reason, all the other dataset's attributes will not be used in the phase of classification.

2.3.2 Data reduction

In this phase we start building the training set starting from the entire dataset reviews.csv. The number of comments is very high and for this reason we decided to keep just a representation of them.

We analysed the distribution of comments over time, extracting this graph:



The comments date from 25 January 2000 to 28 December 2020. We decided to use only the comments from 2008 because it's the year which had the most of them.

Initially we had 1.401.752 comments but after this step the number got reduced to 202.979.

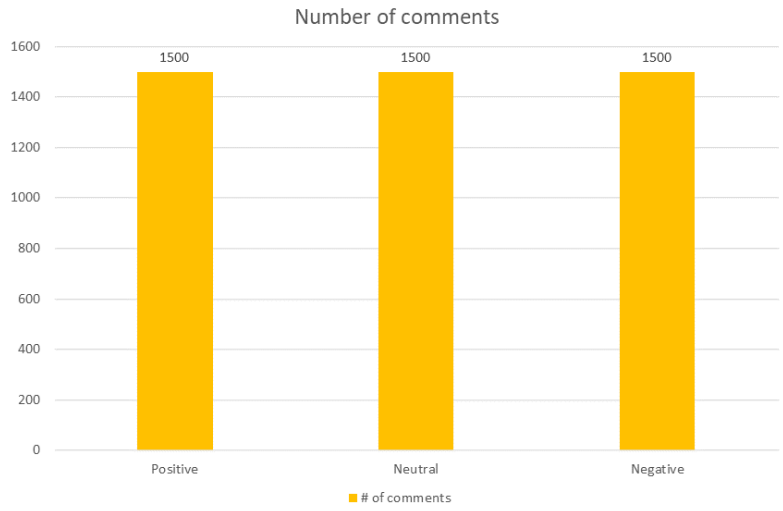
2.4 Training set building

For the training set we built a dataset composed of 4500 comments labelled with three different classes using a ground truth:

- Positive comments: their rating is 5.
- Neutral comments: their rating is 3.
- Negative comments: their rating is 1.

We decided to have a balanced training set composed of 1500 comments from each class. We removed all the comments rated 0 (not rated comments) and comments rated 2 or 4 (they don't have a well-defined class).

The distribution of the training set is shown below:



3. Classification

3.1 Text elaboration using Python

We performed four pre-processing steps to transform our comments into BOW (bag of words) 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. We set *strip_accents='ascii'* which removes accents and performs other character normalization during the preprocessing step working on characters that have a direct ASCII mapping. We set *lowercase=True* which converts all characters to lowercase before tokenizing. We set *token_pattern=r"(?u)\b[^\d\W][^\d\W]+\b"* to define what constitutes a "token". Using this regular expression, we denote as token all words containing at least two characters and without number inside.
- **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'*.
- **Stemming:** in this third step we reduced each token to its stem or root form, removing its suffix, to group similar words. We performed this step passing to the *analyzer* parameter a function called *stemming* where we used the EnglishStemmer function from nltk.stem.snowball.
- **Stem Filtering:** in this fourth step we reduced the number of stems maintaining only the most relevant ones. We set *max_features=3000* as CountVectorizer's parameter which considers the top max_features ordered by term frequency across the corpus.

At the end of the pre-processing steps, we performed:

- a supervised learning stage to assign at each feature of each text the Tf-IDF value using the TfidfTransformer function.
- some tests with different classifiers and for each one we made a "5 fold Cross-Validation".

The results of the tests are shown in the sections below.

3.2 Multinomial Naïve Bayes

We decided to maintain the default parameter after performing a different change of it because in this way we obtained the highest accuracy. Using them the classifier uses the Laplace smoothing to handle the problem of zero probability (each conditional probability has not to be zero).

Correctly Classified Instances: 3545 (78.777%)

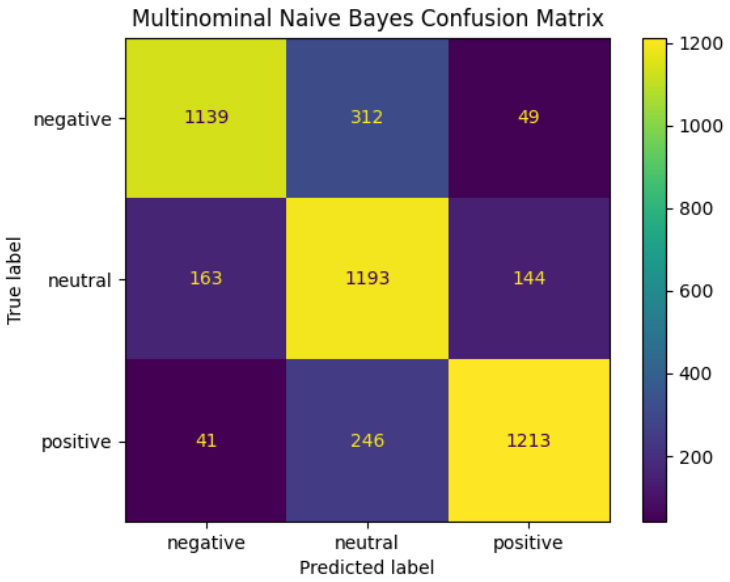
Incorrectly Classified Instances: 955 (21.222%)

Total Number of Instances: 4500

Accuracy Multinomial Naïve Bayes: 0.79 (+/- 0.04)

	Precision	Recall	F1-score	Support
Negative	0.85	0.76	0.80	1500
Neutral	0.68	0.80	0.73	1500
Positive	0.86	0.81	0.83	1500
Average	0.80	0.79	0.79	

Confusion Matrix



3.3 Decision Tree Classifier

We decided to set the *max_leaf_nodes* equal to 30 because we realised that, thanks to it, we obtain the highest accuracy. Without setting this parameter an unlimited number of leaf nodes are generated. For the other parameters we maintained the default value (for instance, the Gini Index as a function to measure the quality of a split and the "best split" as a strategy used to choose the split at each node)

Correctly Classified Instances: 2915 (64.777%)

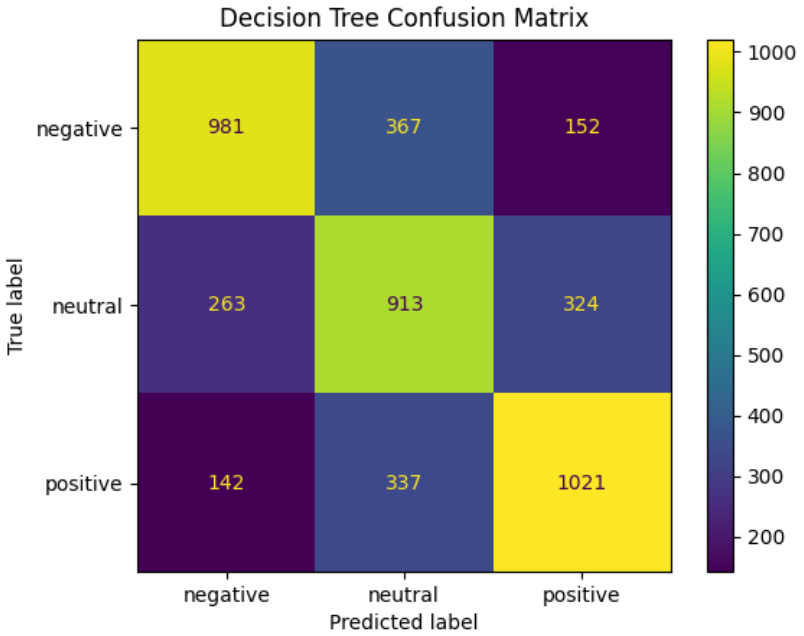
Incorrectly Classified Instances: 1585 (35.222%)

Total Number of Instances: 4500

Accuracy decision tree classifier: 0.65 (+/- 0.04)

	Precision	Recall	F1-score	Support
Negative	0.71	0.65	0.68	1500
Neutral	0.56	0.61	0.59	1500
Positive	0.68	0.68	0.68	1500
Average	0.65	0.65	0.65	

Confusion Matrix



3.4 Linear SVC (Support vector Classifier)

We decided to set the C parameter equal to 0.1 which is a regularization parameter. In this way we obtained the highest accuracy.

Correctly Classified Instances: 3675 (81.666%)

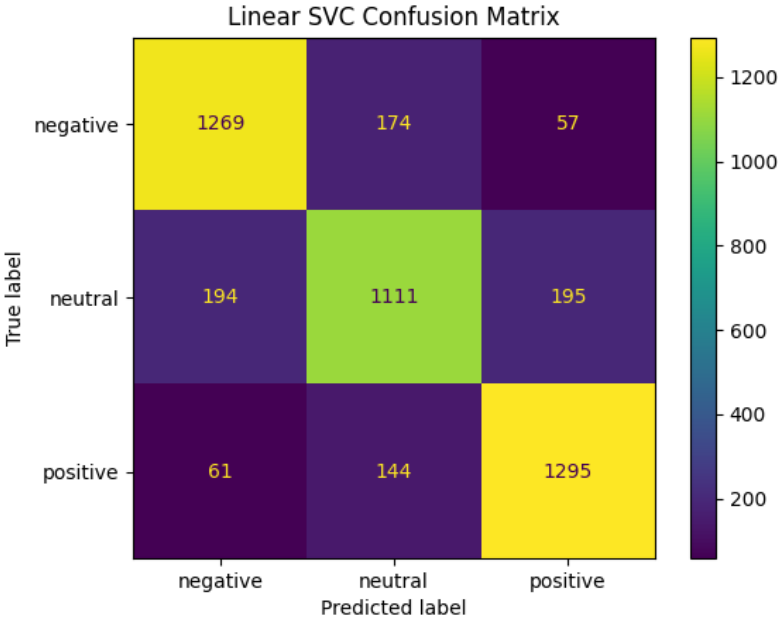
Incorrectly Classified Instances: 825 (18.333%)

Total Number of Instances: 4500

Accuracy linear svc: 0.82 (+/- 0.03)

	Precision	Recall	F1-score	Support
Negative	0.83	0.85	0.84	1500
Neutral	0.78	0.74	0.76	1500
Positive	0.84	0.86	0.85	1500
Average	0.82	0.82	0.82	

Confusion Matrix


Linear SVC Confusion Matrix
negative
neutral
positive
True label
1269
174
57
194
1111
195
61
144
1295
negative
neutral
positive
Predicted label
1200
1000
800
600
400
200

3.5 K-Neighbors Classifier

We decided to set the *n_neighbors* parameter equal to *40*. We kept this decision after testing many values of K: we used an increasing value of k and chose k with the maximum accuracy. We also decided to set weights parameter equal to 'distance' which weights points by the inverse of their distance. In this case, the classifier gives greater weight to closer neighbours.

Correctly Classified Instances: 3312 (73.6%)

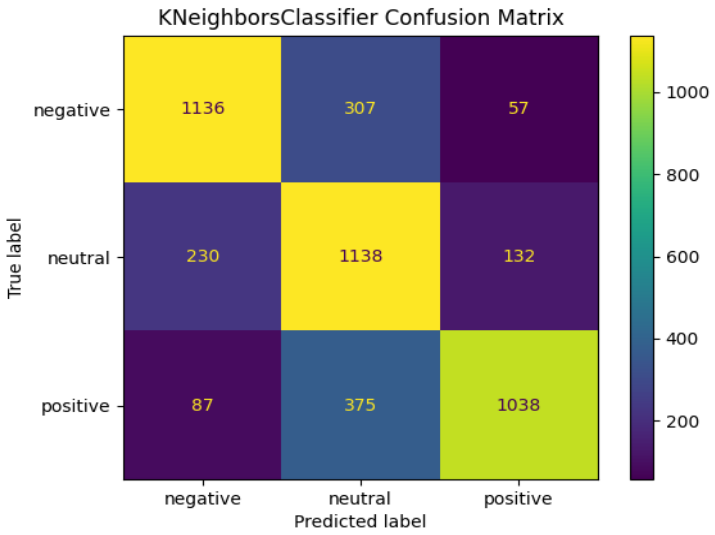
Incorrectly Classified Instances: 1188 (26.4%)

Total Number of Instances: 4500

Accuracy k-neighbors classifier: 0.74 (+/- 0.04)

	Precision	Recall	F1-score	Support
Negative	0.78	0.76	0.77	1500
Neutral	0.63	0.76	0.69	1500
Positive	0.85	0.69	0.76	1500
Average	0.75	0.74	0.74	

Confusion Matrix



3.6 Random Forest Classifier

We decided to maintain the default parameters.

Correctly Classified Instances: 3507 (77.933%)

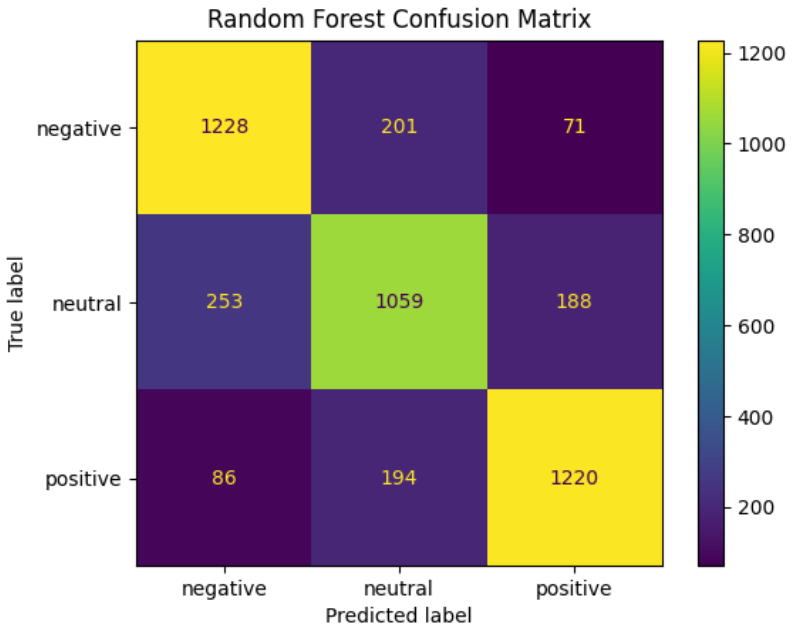
Incorrectly Classified Instances: 993 (22.066%)

Total Number of Instances: 4500

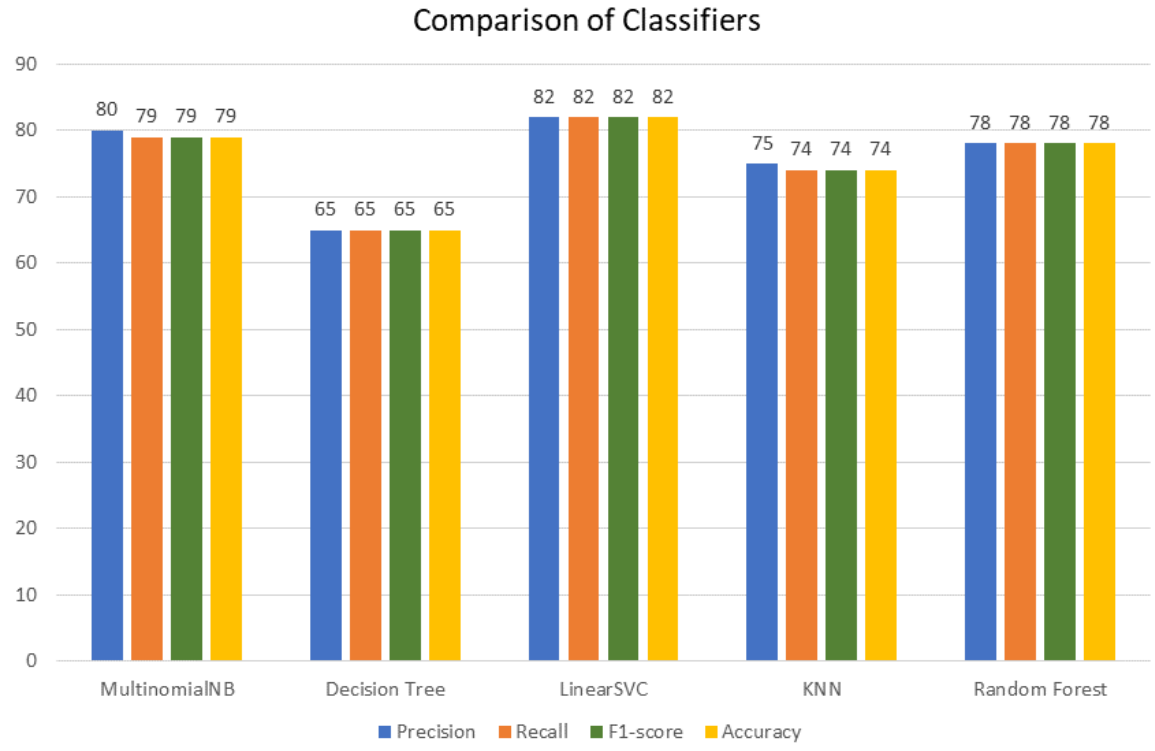
Accuracy random forest classifier: 0.78 (+/- 0.03)

	Precision	Recall	F1-score	Support
Negative	0.78	0.82	0.80	1500
Neutral	0.73	0.71	0.72	1500
Positive	0.82	0.81	0.82	1500
Average	0.78	0.78	0.78	

Confusion Matrix



3.7 Comparison



After we obtained these results we performed a paired t-test to compare the best two classifier: Multinomial Naïve Bayes and Linear SVC. We assumed a significance threshold of α=0.05 for rejecting the null hypothesis that both algorithms perform equally well on the dataset and conducted the 5-fold cross-validated t-test. To make this test we used a python script, the code is shown below:

```
1.  t, p = paired_ttest_kfold_cv(estimator1=text_clf,
2.                               estimator2=text_clf3,
3.                               X=X, y=y, cv=5, random_seed=1)
4.
5.  print('p value: %.3f' % p)
```

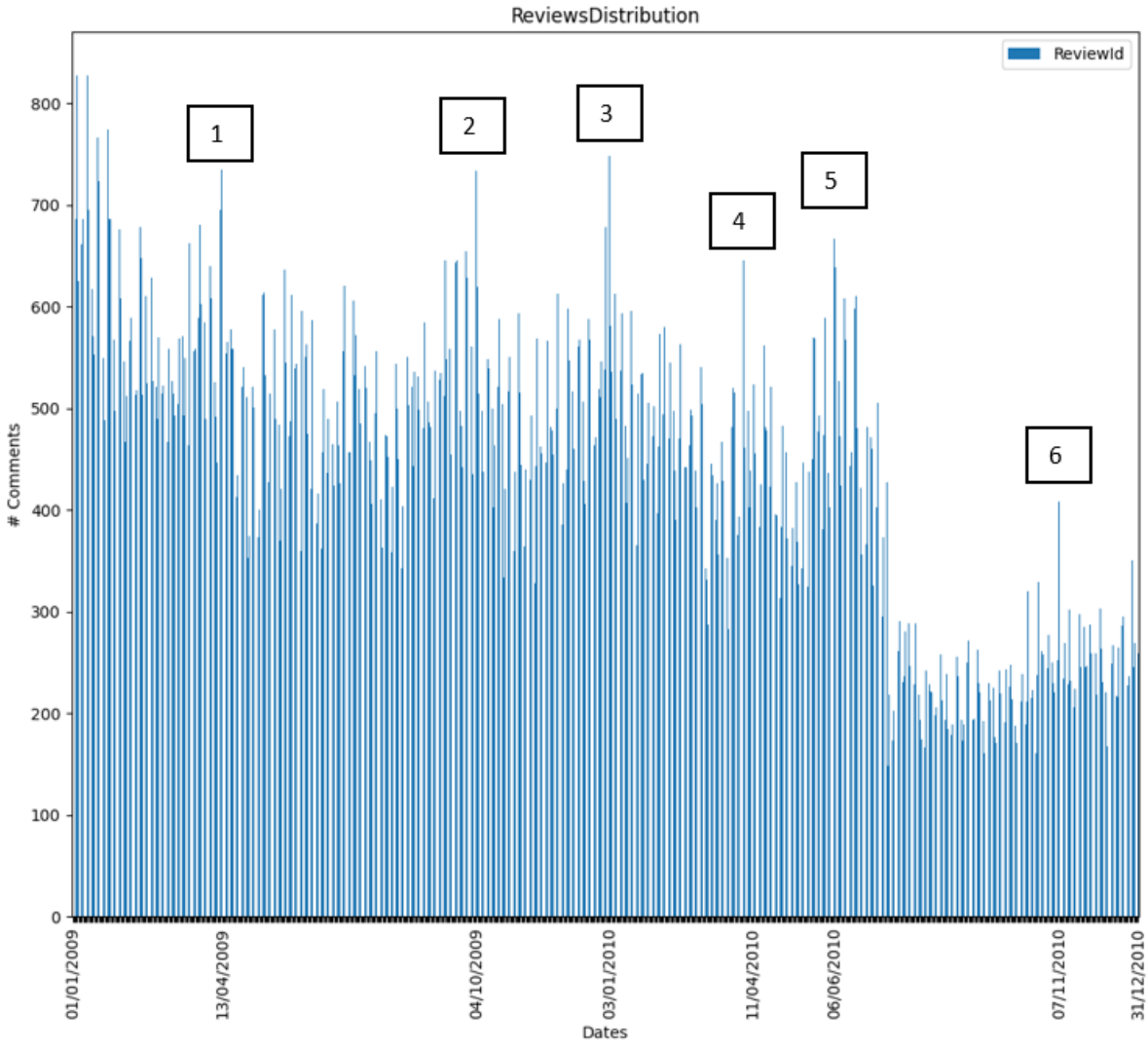
We obtained a p value equal to is 0.003. Since $p < \alpha$ we rejected the null hypothesis and for this reason we decided to use for our application the Linear SVC because it is the model with a lower error rate.

4. Streaming Analysis

We decided to start this analysis to understand if during the following years the users of the recipe site changed their writing style. The more their writing style changes the more the predictive accuracy of our classifier deteriorates over time. In fact, this change, can make the model built on old data inconsistent with new data.

We extracted a graph monitoring the number of comments written in 2009 and 2010 because these are the two years after 2008 (the year of the comments extracted for our previous analysis).

The comments distribution is shown below:



In this subsequent time window (2009-2010) 321.327 comments were collected, we identified 6 events corresponding to the days in which comment peaks were detected.

For each event 60 new comments were labelled, and they were separately used as a test set for the following learning setting:

- Static model: trained with initial training set composed of 4500 comments
- Sliding model: trained each time with the most recent 4500 comments, we removed each time the oldest 60 comments and we introduced the newest 60 comments.
- Incremental model: trained with the initial training set adding the labelled data of all previous events.

To develop this analysis, we wrote three scripts in python, one for each model.

In the next chapters the obtained results are shown for each event.

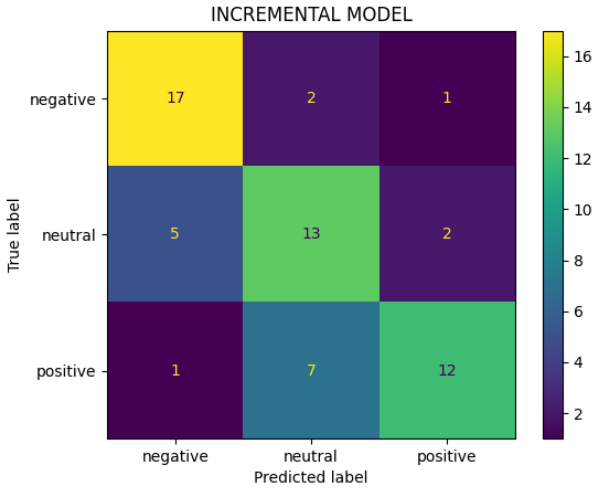
4.1 Event 1 - 13/04/2009

This is the first event and for this reason we obtained the same result for all three models because they worked only with the initial dataset.

In the table below are shown the macro average related to all three methods.

	ACCURACY	PRECISION	RECALL	F1-MEASURE
STATIC	0.70	0.71	0.70	0.70
SLIDING	0.70	0.71	0.70	0.70
INCREMENTAL	0.70	0.71	0.70	0.70

Confusion matrix for all the models:

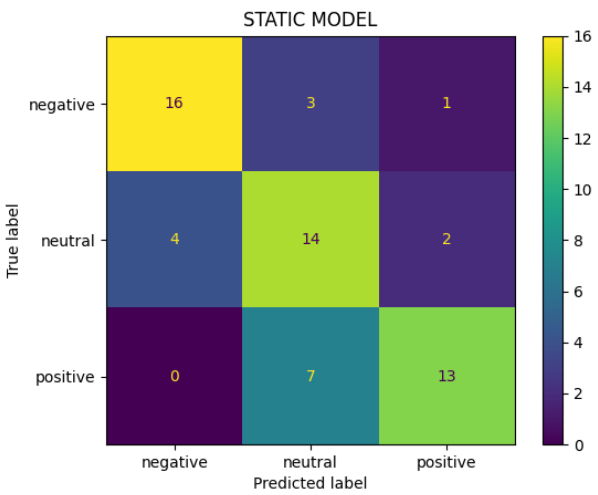


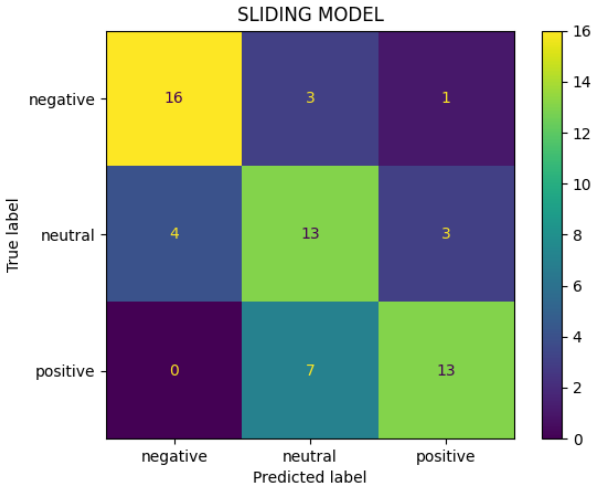
4.2 Event 2 – 04/10/2009

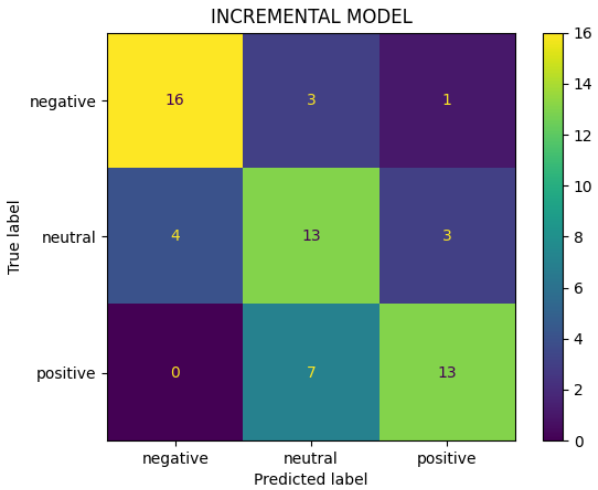
In the table below are shown the macro average related to all three methods.

	ACCURACY	PRECISION	RECALL	F1-MEASURE
STATIC	0.72	0.73	0.72	0.72
SLIDING	0.70	0.71	0.70	0.70
INCREMENTAL	0.70	0.71	0.70	0.70

Confusion matrices:





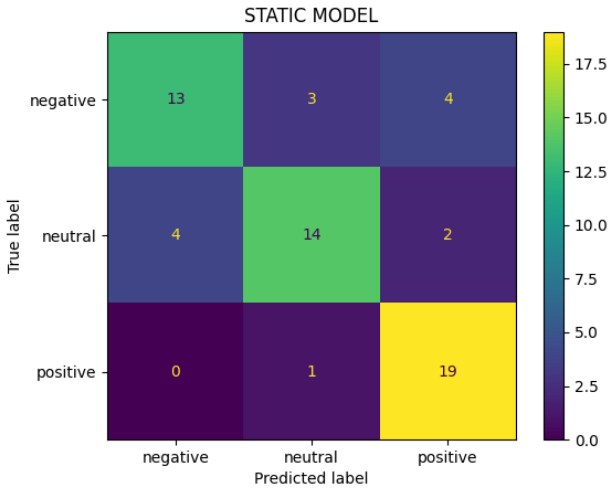


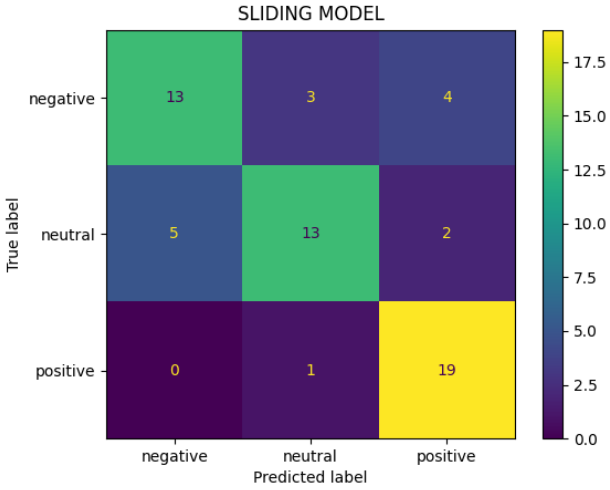
4.3 Event 3 – 03/01/2010

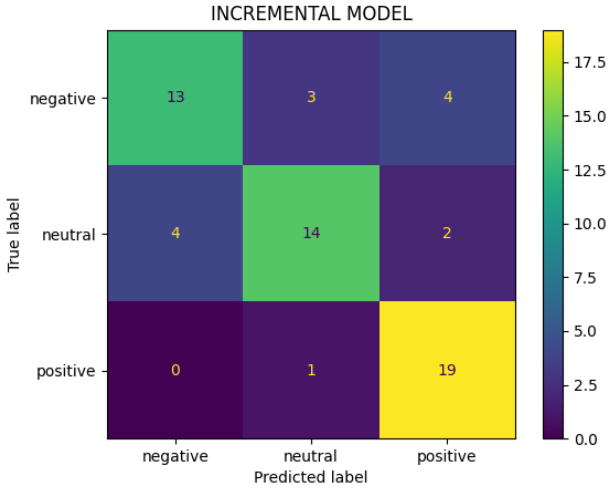
In the table below are shown the macro average related to all three methods.

	ACCURACY	PRECISION	RECALL	F1-MEASURE
STATIC	0.77	0.77	0.77	0.76
SLIDING	0.75	0.75	0.75	0.74
INCREMENTAL	0.77	0.77	0.77	0.76

Confusion matrices:





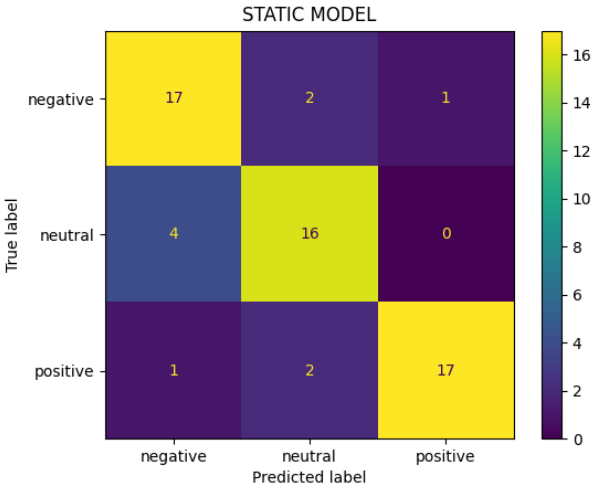


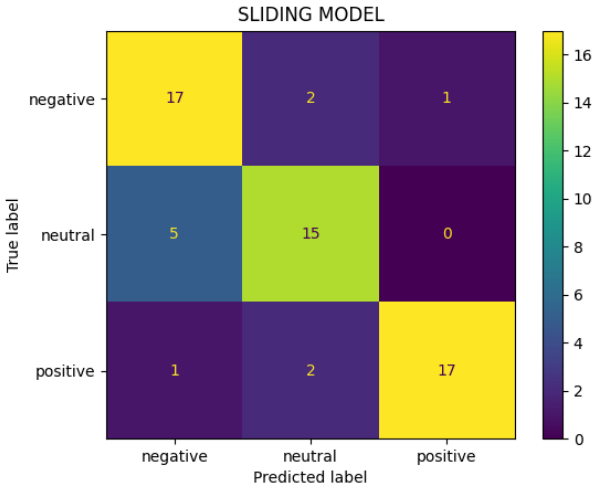
4.4 Event 4 – 11/04/2010

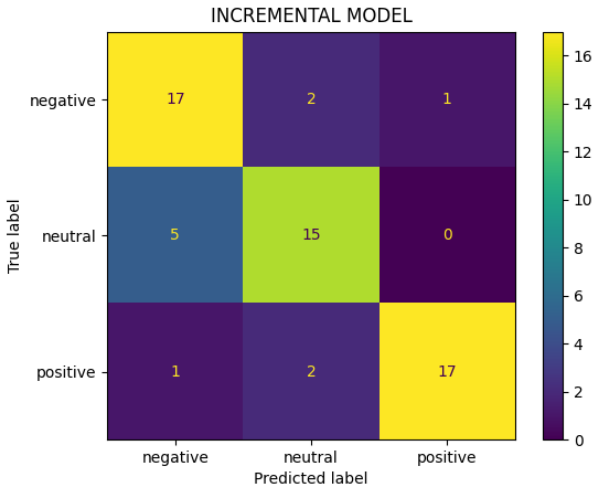
In the table below are shown the macro average related to all three methods.

	ACCURACY	PRECISION	RECALL	F-MEASURE
STATIC	0.83	0.84	0.83	0.83
SLIDING	0.82	0.82	0.82	0.82
INCREMENTAL	0.82	0.82	0.82	0.82

Confusion matrices:





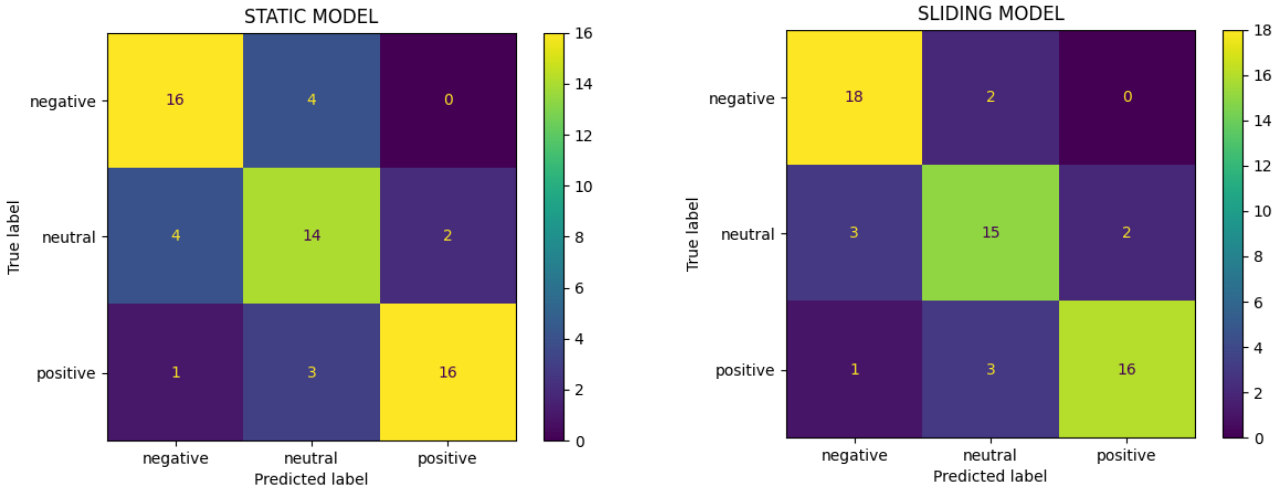


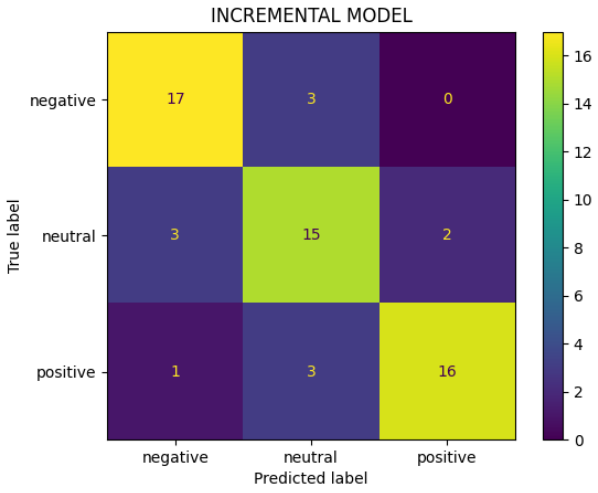
4.5 Event 5 – 06/06/2010

In the table below are shown the macro average related to all three methods.

	ACCURACY	PRECISION	RECALL	F-MEASURE
STATIC	0.77	0.77	0.77	0.77
SLIDING	0.82	0.82	0.82	0.82
INCREMENTAL	0.80	0.80	0.80	0.80

Confusion matrices:



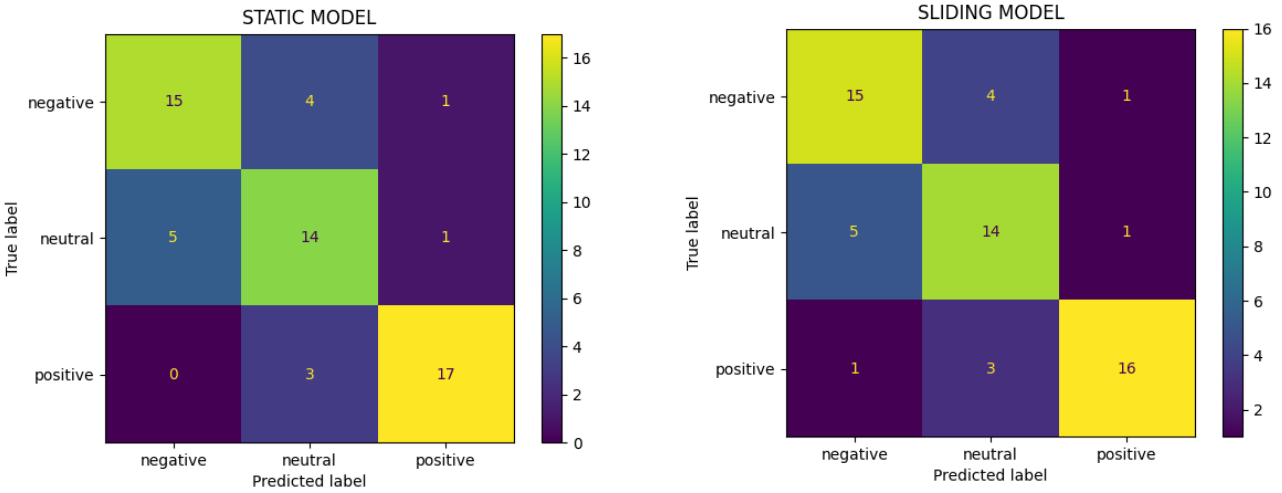


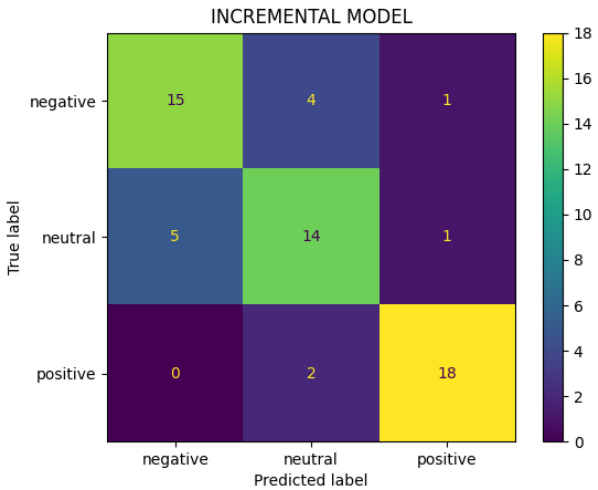
4.6 Event 6 – 07/11/2010

In the table below are shown the macro average related to all three methods.

	ACCURACY	PRECISION	RECALL	F-MEASURE
STATIC	0.77	0.77	0.77	0.77
SLIDING	0.75	0.76	0.75	0.75
INCREMENTAL	0.78	0.78	0.78	0.78

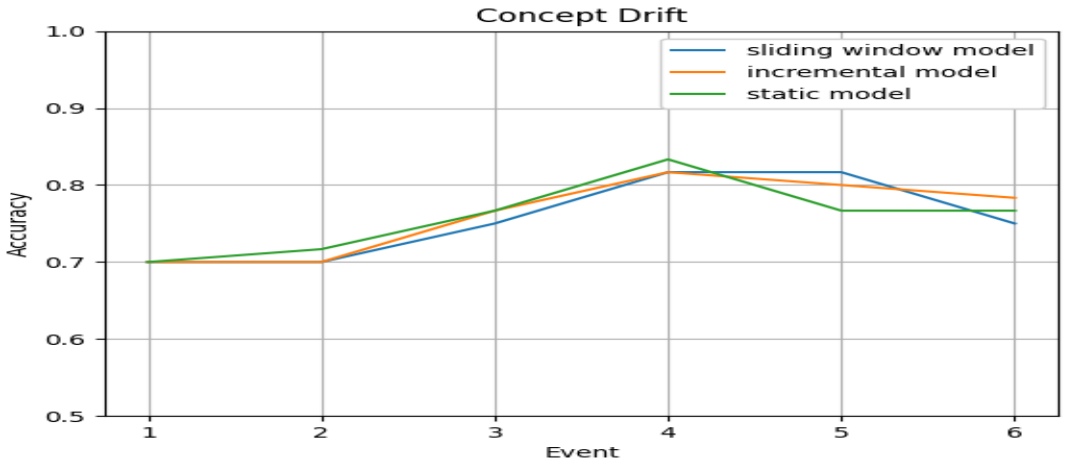
Confusion matrices:





4.7 Conclusions

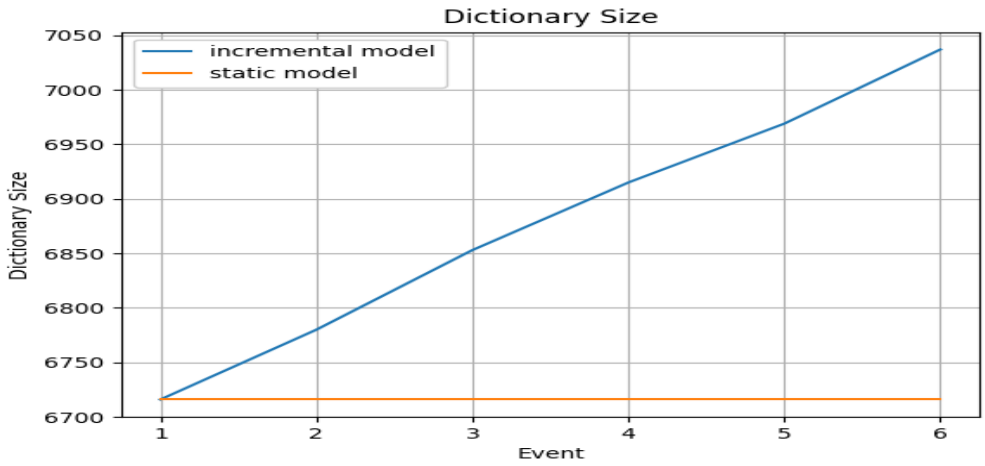
We collected all the results obtained from the previous analysis in the graph below:



We can see that the accuracies of predictions of the three models don't seem to become deteriorated as time passes. In our analysis we haven't explicitly detected much Concept Drift, this means that users didn't change their way to express opinions during the time.

In the first four events the accuracies of the three models are similar but the static model has higher accuracies in comparison to the other two. The situation changes in the last two events. Here we can see that the incremental model performs better than the static model, this is probably due to the increase of the training set. In all events, apart from event 5, the sliding window model performs worse than the static model and the incremental model.

Another consideration must be done on the dictionary size, although the static model uses the same dictionary every time, the sliding and incremental models update their dictionaries during the time. The incremental model adds always new comments increasing the size of the dictionary as shown in the plot below:

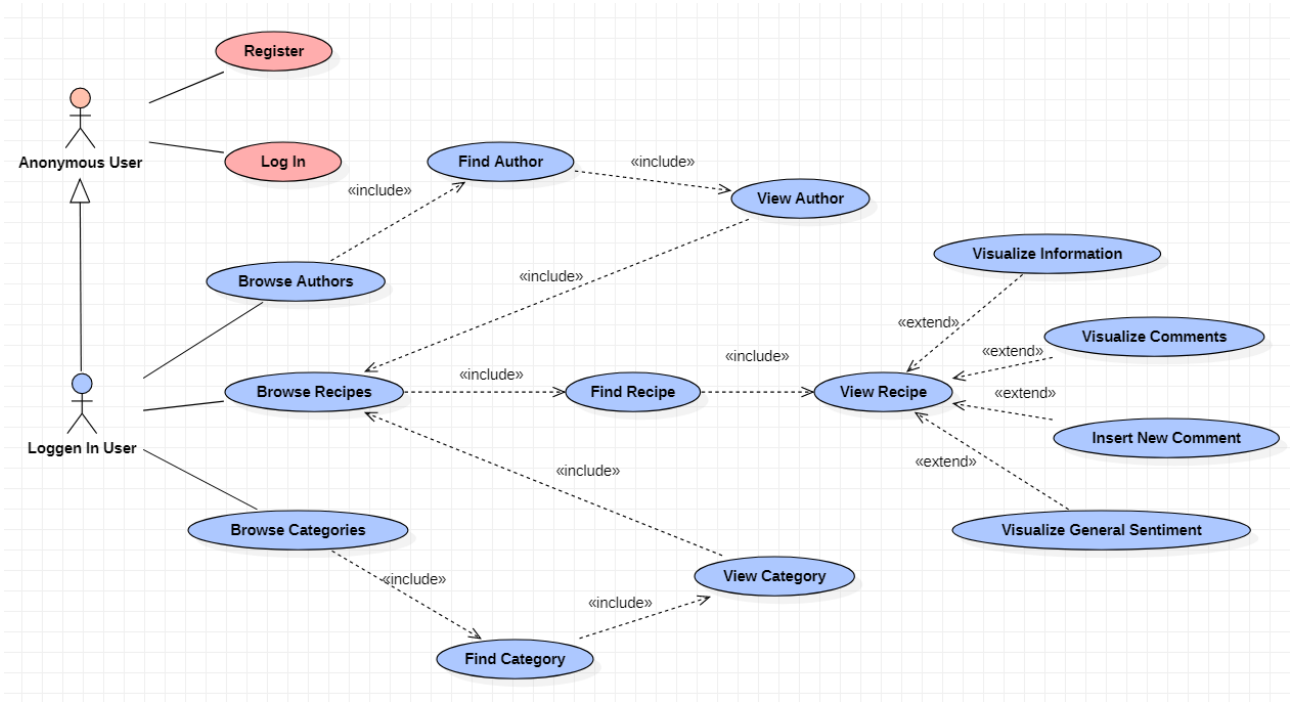


However, in choosing the incremental model we must take into consideration the fact that new input data is continuously used to further train the model. We couldn't handle the out-of-core learning with the increase of the training set in memory. The classifier LinearSVC doesn't support incremental learning. For this reason, we have decided to use the static model.

5. APPLICATION

The application we made works as a normal recipe site where a user can log in and find information about many different recipes. Within this application a user can also see comments related to a specific recipe and the sentiments associated with them. The sentiments are calculated using the static model according to the analysis made in the fourth chapter. When the user leaves a comment, under a specific recipe, it's analysed to find the sentiment associated. Thanks to different sentiments a user can have a general view about the opinion of other users.

Below the use case diagram is reported:



5.1 Preparing Data

First we saved our classification model into three different files, and we used them to classify all the comments present in the review.csv dataset. We performed on them the pre-processing steps (tokenization, stop-word-filtering and stemming).

```
comments_df = pd.read_csv("../dataset/modified_comment.csv")
count_vect_model = pickle.load(open('../model/count_vect.sav', 'rb'))
tfidf_model = pickle.load(open('../model/tfidf_model.sav', 'rb'))
loaded_model = pickle.load(open("../model/static_model.sav", 'rb'))

X_test = comments_df['Review'].values
```

```
def static_model(loaded_model, X_test):

    X_test_counts = count_vect_model.transform(X_test)
    X_test_tfidf = tfidf_model.transform(X_test_counts)

    predicted = loaded_model.predict(X_test_tfidf)
    comments_df['Rating']=predicted
    comments_df.to_csv(r'../dataset/predicted_all_comments.csv', index=False)


if __name__ == '__main__':

    static_model(loaded_model, X_test)
```

5.2 Develop application

We developed our application using python.

When a user starts the application, they must pass a login phase where they must insert an existing user ID and password or register to create a new account.

```
WELCOME TO RECYLIS
**********************************
LOGIN MENU'
What do you want to do?
Select:
1-> Log In
2-> Register
**********************************
**********************************
Write command:
>?
```

Figure 1

After this the main menu is showed. It's possible to browse all the recipes, browse the recipes filtered by category or browse the authors who uploaded at least 1 recipe.

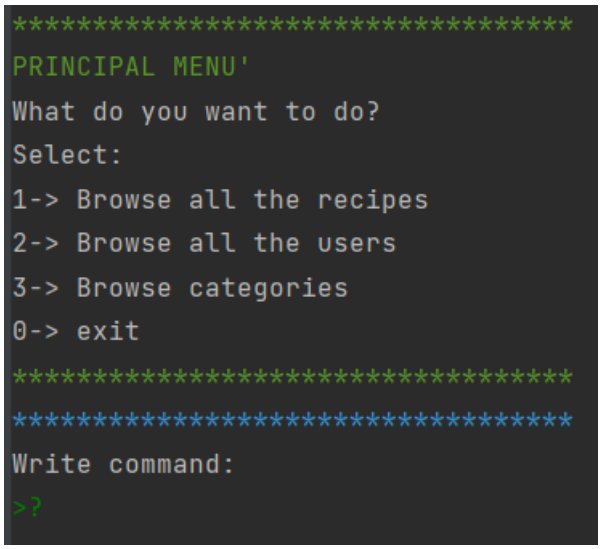


Figure 2

Now we analyse all the three possible commands:

- *Browse all the recipes* → the recipes' previews will be displayed 10 at a time. Each of them will be identified by a number, their ID, which will be required in order to view it.

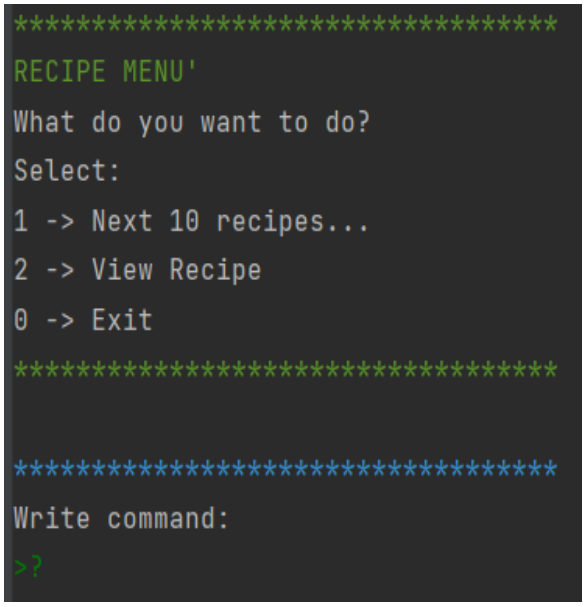


Figure 3

After selecting "View Recipe" it will be asked to insert the id of the desired recipe, then it will be possible to visualize the recipe itself, the general sentiment, the comments or to leave one.

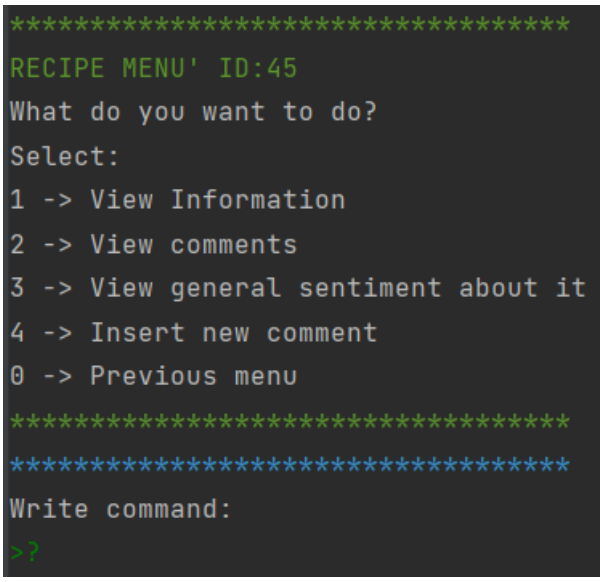
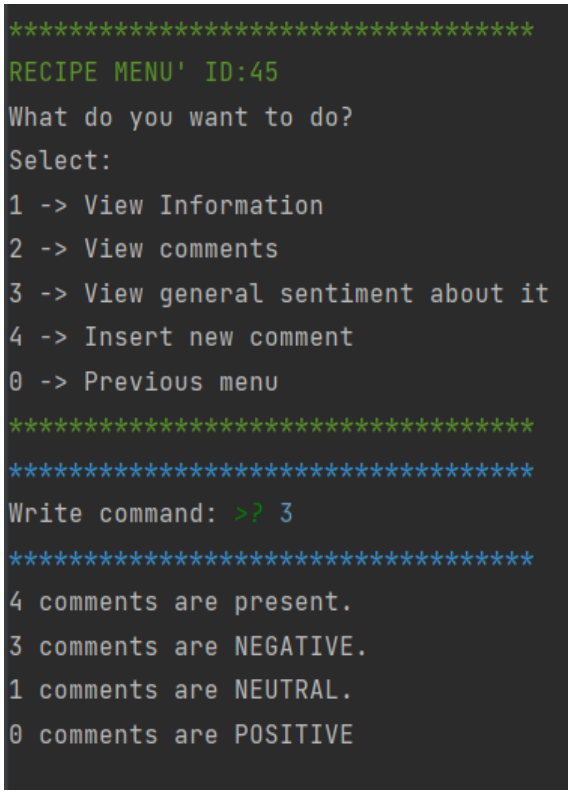


Figure 4



- *Browse all the users* → the authors' names will be displayed 10 at a time. Each of them will be identified by a number, their ID, which will be required in order to view their recipes.

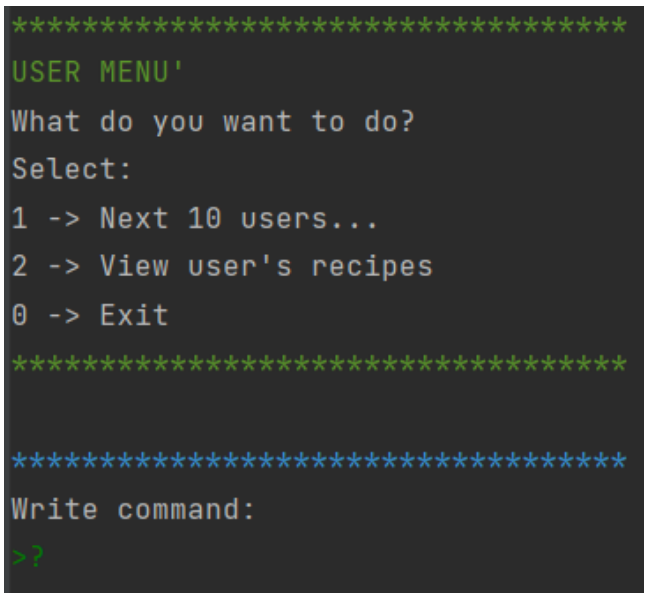


Figure 5

- *Browse categories* → the categories will be displayed 10 at a time. Each of them will be identified by their own name which will be required in order to view it.

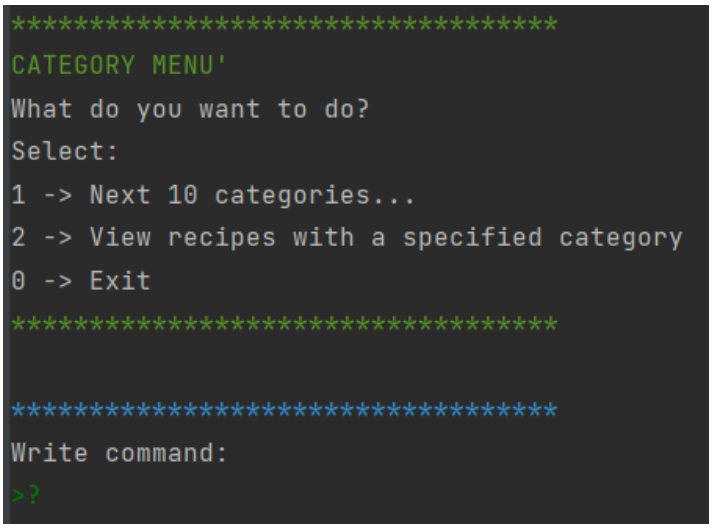


Figure 6

Finally press 0 to Exit.