SAE S5 BUT3

Aide à la décision Modélisation Mathématiques

Romain GOURAUD

Mathys MEUNIER

18/12/2023

Table des matières

Vue d'ensemble du sujet

Le but de cette SAE est de mettre en place des modèles pour réaliser des prédictions (classification).
Vous allez travailler en binôme (vous êtes 29 étudiant(e)s donc 13 binômes et 1 trinôme).
Vous devez choisir un jeu de données parmi ceux disponibles ici : datasets.html#amazon_reviews et qui respecte les contraintes suivantes :

- Il ne faut pas que 2 binômes différents travaillent sur le même jeu de données
- Il faut au moins qu'une variable caractéristique corresponde à des avis d'utilisateur qui sont rédigés en langage naturel
- La variable cible correspondra à une évaluation (une note) qui souvent sera un nombre (mais pas nécessairement). Il faudra que cette évaluation ne soit pas binaire, mais concerne au moins 3 valeurs différentes. C'est cette variable cible qui sera prédite.
- Les jeux de données peuvent être volumineux (de l'ordre du giga ou plus pour certains).
- Vos traitements doivent pouvoir tourner sur les machines de l'IUT. Ce qui signifie que vous pouvez tout à fait récupérer des données volumineuses hors de l'IUT mais il faudra en choisir un extrait pour que les traitements puissent s'effectuer dans des temps convenables.
- Vous utiliserez exclusivement scikit-learn comme bibliothèque pour réaliser de l'apprentissage automatique et spacy pour réaliser des traitements linguistiques. Une classe peut être utile pour représenter le texte qui est en langage naturel sous une forme exploitable par les systèmes de classification (tableau de nombre) : CountVectorizer de la bibliothèque scikit-learn. Cette classe permet de représenter un texte sous forme d'une matrice de nombre.

Attention ! Si vous voulez tester le projet merci de faire attention de bien avoir le json dans le répertoire data et de pas changer son nom merci.

Projet id: 21105
Project link : https://gitlab.univ-nantes.fr/E217817H/saeadmm

Développement

I. Introduction

Comme expliqué dans la vue d'ensemble du sujet, le principe de cette SAE est de prédire si un avis est positif ou négatif, puis de prédire la note de cet avis. Pour cela, nous avons utilisé scikit-learn et spacy.

II. Présentation du jeu de données

Pour ce projet, durant le choix de dataset, nous avions deux grands critères pour gagner du temps. Le premier que le dataset soit d'une grande entreprise du digital et notre deuxième qu'elle fasse de l'ia si possible. En effet, nous sommes partis du constat que si l'entreprise est de ce domaine, il y aurait une norme bien respectée dans les données et surtout que celle-ci serait dite "propre".

Pour cela, nous nous sommes orientés vers les datasets de Google, plus particulièrement celui de Google pour les restaurants. Ce dataset se présente de la façon suivante :


```
"name":"The Fish Spot",
"address":"5101 W Pico Blvd, Los Angeles, CA  90019",
"Description":null,
"Latitude":34.0481627,
"Longitude":-118.3494339,
"category":["Seafood restaurant"],
"gmap_url":"https://www.google.com/maps/place/The+Fish+Spot/",
"Avg_rating":4.3,
"Num_of_reviews":80,
"price":"$$",

"Reviews":
[
  {"user_id":"111210125124533240892",
  "time":"3 years ago",
  "Rating":5,
  "text":"Absolutely love this place.",
  "pics":[
    {"id":"AF1QipO1ejvRhkVBlg-v52UczxYMD7uebcZIhKC9uGud",
     "url":["https://lh5.googleusercontent.com/p/"]},
    ],
  "link":"https://www.google.com/maps/reviews/"},
...,]
```


	subset	full
Restaurants:	30K	65K
Users:	37K	1.01M
Reviews:	108K	1.77M
Images:	203K	4.43M

Le dataset est composé de plus d'un million d'utilisateurs et un utilisateur est présent s'il a un ou plusieurs avis, nous disposons donc d'un dataset de plus d'un million d'avis ce qui va nous permettre de faire exactement ce que nous voulons sans problème lors des étapes suivantes. Les notes quant à elles sont de 1 à 5.

III. Les prétraitements réalisés

Dans un premier temps, nous avons choisi de ne pas garder tous les champs du dataset. Nous avons ciblé les éléments importants pour le sujet comme le "rating" autrement dit la note, le "review_text" qui contient le texte de l'avis. À des fins de possible statistique, nous avons gardé d'autre élément comme l'id du notant, l'id du restaurant et avons créé un id unique à l'avis composé de idDuNotant_idDuRestaurant. En plus de tout ça, nous avons ajouté un champ label pour définir si l'avis est positif ou négatif.

```python
rawData = []
with open("./data/image_review_all.json") as fichier:
    for line in fichier:
        dataTempo = json.loads(line)
        rawData.append({
            "id": dataTempo.get('user_id') + "_" + dataTempo.get('business_id'),
            "business_id": dataTempo.get('business_id'),
            "user_id": dataTempo.get('user_id'),
            "rating": dataTempo.get('rating'),
            "review_text": dataTempo.get('review_text'),
            "label": 'positif' if dataTempo.get('rating') > 2 else 'negatif'
        })
rawDF = pd.DataFrame(rawData)
```

Récupération des champs à garder et création du label.

IV. Différentes expérimentations réalisées

A. Équilibrage du dataset

Dans un premier temps, nous avions utilisé nos données de façon brute, mais nous n'avions des résultats que trop peu convenables. Après avoir analysé nos données, nous nous sommes rendu compte que nous ne faisions pas un apprentissage sur

assez d'avis négatif. Pour répondre à ce problème, nous avons donc créé un sous dataset pour avoir autant d'avis positif que négatif.

```
positiveReviewDF = rawDF[rawDF['label'] == 'positif']
negativeReviewDF = rawDF[rawDF['label'] == 'negatif']
positiveReviewDF = positiveReviewDF.sample(frac=1, random_state=42)

Data = pd.concat([negativeReviewDF, positiveReviewDF.head(min(len(negativeReviewDF), len(positiveReviewDF)))])
Data = Data.dropna()
Data = Data.sample(frac=1, random_state=42)

print("Taille du DataFrame = " ,Data.shape)
Data.head()
```

Séparation des avis positifs des négatifs et équilibrage

Pour cela, nous séparons les avis positifs des négatifs, mélangeons les avis positifs, car ceux-ci sont plus nombreux. Suite à cette séparation, nous créons un dataset composé de la concaténation des avis négatifs et du même nombre d'avis positifs. Pour rendre le dataset exploitable, nous supprimons les colonnes vides si existantes et mélangeons le dataset pour masquer la concaténation.

B. Optimisation des hyperparamètres de knn

Comme nous le verrons, le résultat du modèle KNN n'est pas très bon, pour cela, nous avons utilisé GridSearchCV pour optimiser les hyperparamètres du modèle. Nous parlerons de cet ajout et son effet sur les résultats dans la partie suivante.

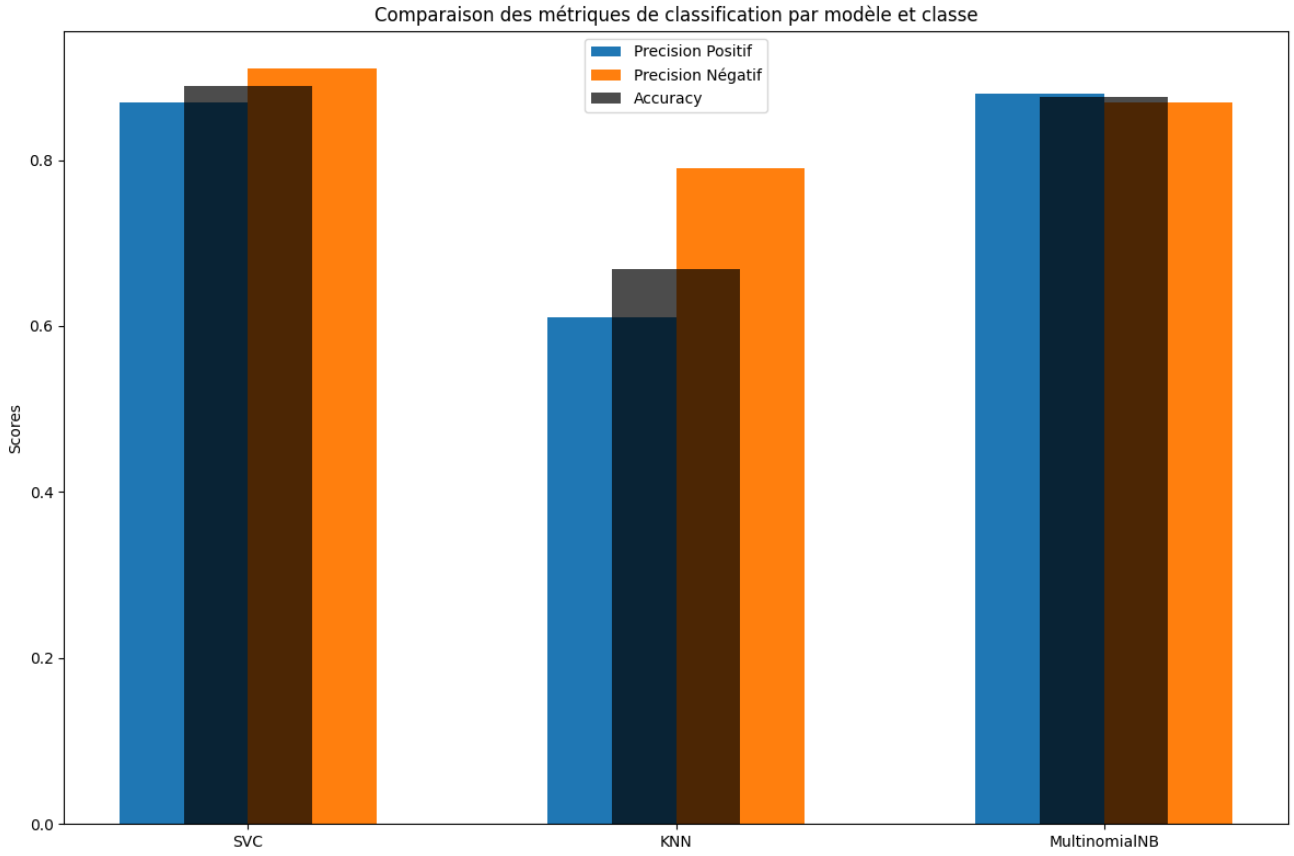
```
vectorizer = CountVectorizer()
reviewVectorised = vectorizer.fit_transform(DataForLearning['review_text'])
trainX, testX, trainY, testY = train_test_split(reviewVectorised,DataForLearning["rating"] ,test_size=0.20)


param_grid = {
    'n_neighbors': [1,3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}

knn = KNeighborsClassifier()


grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy')
grid_search.fit(trainX, trainY)


print("Meilleurs hyperparamètres:", grid_search.best_params_)

knn_classifier3 = grid_search.best_estimator_
```

Optimisation des hyperparamètres du modèle KNN

V. Analyse des résultats globaux obtenus

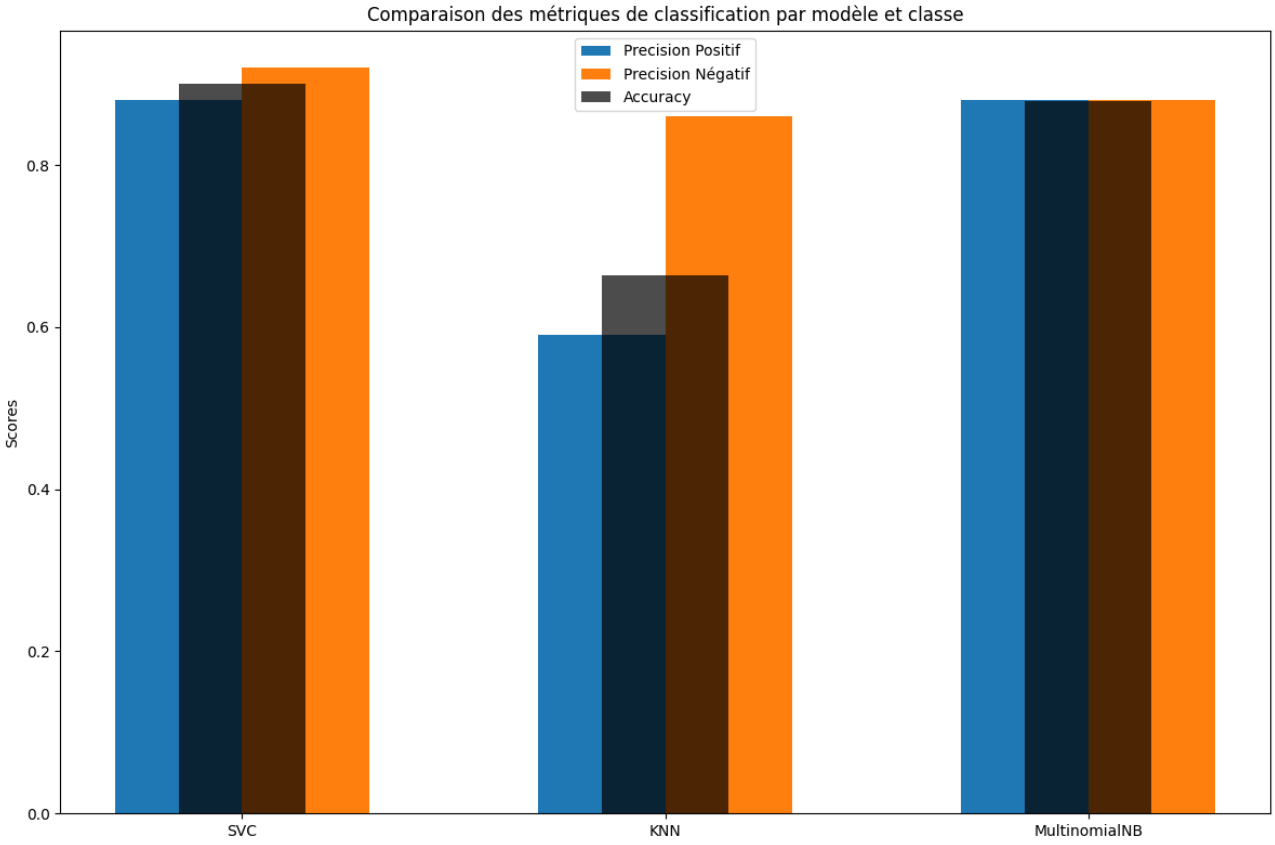
A. Analyse des résultats de la partie deux.

Tout d'abord, pour réaliser notre apprentissage, nous avons sélectionné plusieurs classifier. C'est suite à nos recherches que nous avons choisi SVC et MultinomialNB mais pour knn c'est uniquement car il était utilisé en cours. Les résultats obtenus ne sont vraiment pas hétérogènes comme peut le montrer ce graphique pour la partie deux :



Résultats constatés avant optimisation de knn

Ce graphique représente nos résultats pour la deuxième partie, avant l'ajout de l'optimisation des hyperparamètres du modèle knn la prédiction de si l'avis est positif ou négatif. Comme nous pouvons l'observer que le modèle knn est bon dernier malgré pour tenter de pallier ça nous avons mis en place une optimisation des hyperparamètres du modèle comme montré précédemment. Voici le résultat de l'optimisation mise en place.



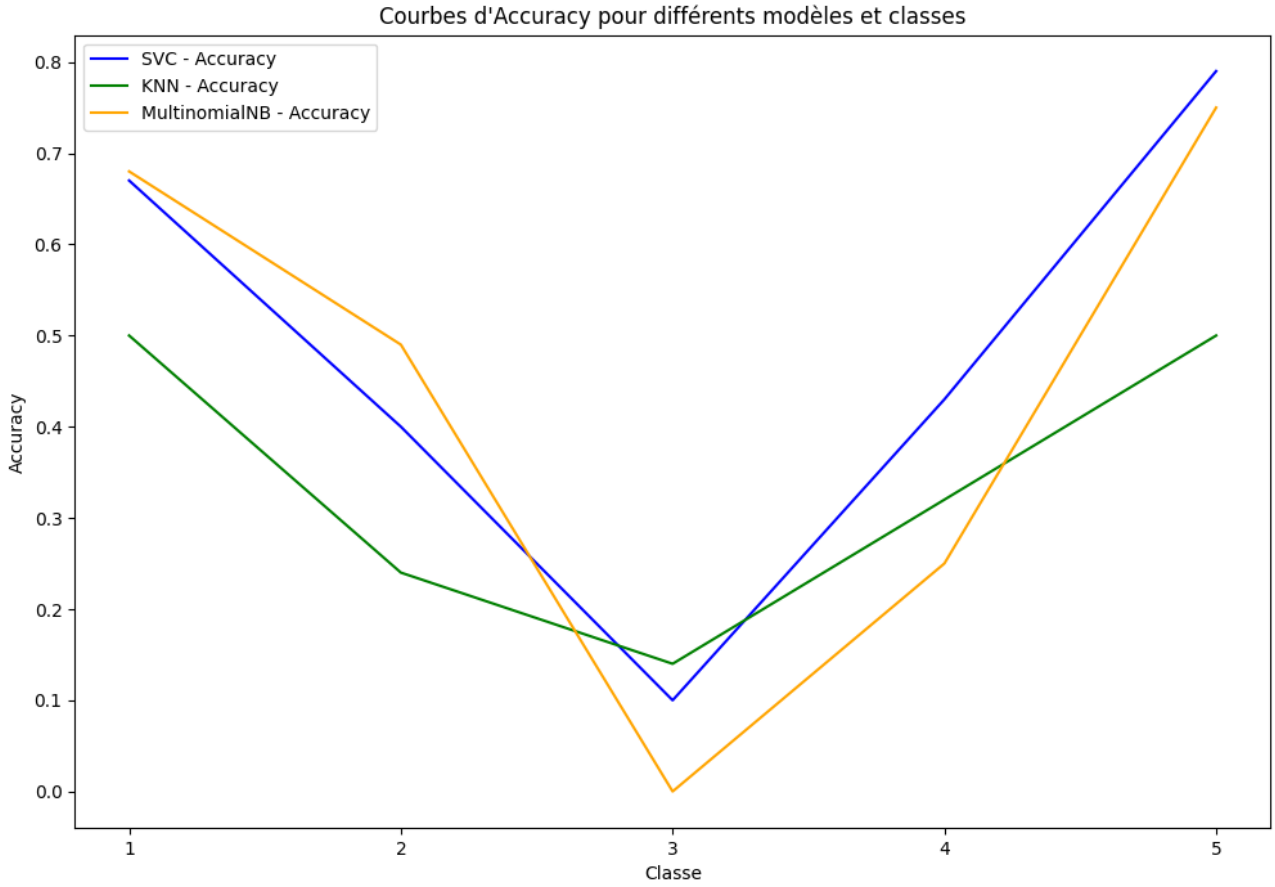
Après optimisation des hyperparamètres.

Après optimisation, les précisions de prédictions du modèle KNN pour les avis négatifs rattrapent les autres modèles, cependant les avis positifs restent très durs à prédire pour le modèle.

Notre graphique ne permet pas de montrer quel modèle est le meilleur, mais les modèles MultinomialNB et SVC sont dans les mêmes tranches de résultats avec toujours plus de 85 % de prédiction juste. Ces modèles montres aussi un résultat de prédiction homogène au contrario du modèle KNN et prédit plus difficilement les avis positifs.

B. Analyse des résultats de la partie trois.

Ensuite, comme pour les prédictions de la partie, tous les modèles n'ont pas su répondre aux attentes, du moins pas dans leurs ensembles. Nous allons donc analyser les résultats de ces modèles.



Voici ainsi le graphique du taux de précision de prédiction des classes par les différents modèles. Comme pour la partie précédente, le modèle KNN est moins performant, étant bon dernier dans toutes les classes. Le meilleur modèle dans notre cas est SVC, il reste complet dans la prédiction des différentes classes surtout dans la classe de note 3 cette classe s'est révélé la plus dure à prédire pour tous les modèles, en effet en son caractère de classe neutre ou classe du milieu couplé à son manque de représentation total dans le dataset la rend plus difficile que les autres à prédire. Quant à lui, le modèle MultinomialNB est bon deuxième avec des résultats catastrophiques dans la prédiction de la classe 3.

C. Tableau récapitulatif des résultats

Prediction\classifier	MultinomialNB	KNN	SVC/SVM
Précision prédiction binaire	0.884	0.643	0.868
Précision prédiction multiclasse	0.656	0.456	0.621

Tableau récapitulatif par type de prédiction

Classes \ Classifier	MultinomialNB	KNN	SVC/SVM
Positif	0.90	0.62	0.9
Négatif	0.88	0.83	0.86

Détails de la prédiction binaire par classe des modèles

Classes\Classifier	MultinomialNB	KNN	SVC/SVM
1	0.70	0.54	0.62
2	0.42	0.32	0.40
3	0.00	0.07	0.09
4	0.15	0.22	0.48
5	0.78	0.47	0.8

Détails de la prédiction multi classes des modèles

Comme vu précédemment les résultats sont meilleurs avec MultinomialNB et SVC. Ce score supérieur nous a permis dans le NoteBook de tester avec des phrases non apprises par les modèles et d'avoir des résultats concluants.

En effet, les tests réalisés sur le modèle KNN n'étaient pas concluants.

VI. Explication de nos choix pour la partie bonus

A. Pourquoi avoir choisi ces classifier

Pour choisir nos classifier non vu dans le cours, nous nous sommes référés à plusieurs articles comme la documentation de SKLEARN[1] plus particulièrement un article de la documentation qui expliquait comment travailler avec du texte comme data (lien[2]), en effet le classifier MultinomialNB est utilisé dans les exemples. Pour le classifier SVC (svm lien[3]) il semblait juste intéressant à utiliser. Enfin, le classifieur KNN a été vu en cours, il était donc logique de le choisir.

B. Comment fonctionnent nos classifier

Le classifier MultinomialNB fonctionne sur le théorème de Bayes, c'est une règle de probabilité conditionnelle qui inverse les probabilités. Il permet de calculer la probabilité d'un événement A sachant qu'un événement B s'est produit.

Le Support Vector Classifier (SVC) trouve le meilleur plan pour séparer les groupes de données. Il s'assure que cet espace entre les groupes est aussi grand que possible. Cela garantit une bonne classification, même avec de nouvelles données. Les SVC sont flexibles et peuvent gérer des situations plus compliquées grâce à l'utilisation de noyaux. En résumé, le SVC est un outil puissant pour organiser des données en groupes distincts.

Le KNN fonctionne en prédisant la classe d'une nouvelle observation en fonction des classes des voisins les plus proches. Il mesure la distance entre l'observation à prédire et les observations d'apprentissage existantes dans un espace multidimensionnel. Le "k" représente le nombre de voisins à considérer. La classe la plus fréquente parmi les k voisins détermine la prédiction du modèle. C'est un algorithme simple et intuitif pour la classification.

VII. Conclusion

En conclusion, durant cette SAE, nous avons mis en place des modèles de prédiction d'avis utilisateur basés sur le dataset de Google pour les restaurants. Les prétraitements, l'équilibrage du dataset et l'optimisation des hyperparamètres KNN ont été essentiels. Les modèles SVC et MultinomialNB ont montré une précision constante de plus de 85 %, tandis que le KNN a nécessité une optimisation. En résumé, ce projet nous permit de créer des modèles pour prédire les évaluations d'avis utilisateurs, mettant en lumière des insights sur les forces et faiblesses de chaque méthode.

VIII. Glossaire

1. Documentation des classifiers sklearn

   https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

2. Documentation “Working with text data”

   https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

3. Documentation SVM/SVC

   https://scikit-learn.org/stable/modules/svm.html