Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
*** * ***
Université de Carthage
*** * ***
Institut National des Sciences Appliquées et de Technologie

Rapport du projet Data Mining

Filière : Génie Logiciel
Niveau : 4ième Année

Sujet :

Le Data Mining pour la prévision de succès des films

Réalisé par :

Amal MTIBAA

Yasmine MHIRI

Mohamed Wassim RIAHI

Année Universitaire : 2018/2019

Table des matières

Table des figures

1 Introduction

Notre société évolue quotidiennement et ses attentes changent. Même chose pour le Cinéma. Les gens apprécient le cinéma depuis les années 50. Depuis l'arrivée de la télévision et puis l'internet, la production des films dans le monde est augmentée de plus en plus.

Mais cette augmentation n'implique pas la réussite de tous ces films. C'est-à-dire, des millions de dollars perdus lorsque les films ne rentabilisent pas leurs budgets.

Alors que le financement du cinéma se complique, il devient primordial de convaincre les investisseurs de la rentabilité de leurs dépenses. Des outils mathématiques permettent de prévoir avec une marge d'erreur acceptable les recettes d'un film, et donc son équilibre économique. Pour faire ceci, nous allons utiliser le data mining pour la prédiction des succès des films.

Ce rapport vise à présenter dans un premier temps le contexte et la spécification des besoins. Puis il présente le résultat du travail que nous avons réalisé, ainsi que la démarche suivie basant sur la méthode CRISP.

2 Cahier des charges

2.1 Contexte

Dans le cadre d'un projet dans la matière « Apprentissage et fouilles des données », nous avons choisis le sujet « Le Data Mining pour la prévision de succès des films ».

Ce sujet permet de prédire le succès d'un film au box-office, accompagné d'une interface simple qui facilite l'utilisation des algorithmes utilisés et la visualisation des résultats.

2.2 Etude de l'existant

Plusieurs start-up assurent avoir mis au point un programme qui permet de prédire le succès d'un film en analysant son scénario.

Google prétend connaître à l'avance la popularité d'un film au box-office, d'après une étude. Une enquête réalisée par des employés du géant américain considère que le moteur de recherche est capable de savoir, avant même leur sortie, si un film fera un carton ou non auprès des spectateurs.

Mais, il n'y a pas une application ou un site web qui offre au public, y compris les directeurs et les producteurs, des outils qui les aident à prédire le succès d'un film avec des caractéristiques spécifiques.

2.3 Analyse des besoins

2.3.1 Les besoins fonctionnels

- ✓ Notre application permet aux investisseurs de prévoir le succès des films selon plusieurs critère tels que le budget, le caste, les directeurs, les genres du film, etc...
- ✓ Notre application permet aux utilisateurs de tester des exemples avec des films déjà sortis lorsqu'ils n'ont pas des nouveaux films à saisir.
- ✓ Notre application permet aux utilisateurs d'évaluer les résultats des algorithmes de classification en donnant le taux d'erreurs de chacun.
- ✓ Notre application fournit un outil d'évaluation du caste d'un film avec un algorithme qui mesure la compatibilité des acteurs en utilisant leurs expériences précédentes.
- ✓ Notre application fournit un outil d'évaluation des directeurs et des scénaristes du film en mesurant leurs rendements dans les films dont ils ont fait partie.

2.3.2 Les besoins non fonctionnels

Par conjonction aux besoins fonctionnels précédemment cités, notre solution devra respecter un ensemble de critères qui contribuent à une meilleure qualité de la solution obtenue. Parmi ces critères, nous citons :

- ✓ **La performance :** Le temps de réponse des algorithmes et la façon de parcourir le « dataset » doivent être optimaux. Ainsi que le temps de chargement de l'interface doit être rapide.
- ✓ **L'intégrité :** Le traitement des tous les données du « dataset » en prenant en considération les mauvaises données pour éviter l'arrêt de la politique d'importation.

3 Travail réalisé

3.1 La méthode utilisée

Pour réaliser notre projet, nous avons utilisé la méthode « CRISP » qui se décompose en 6 étapes allant de la compréhension du problème métier au déploiement et la mise en production.

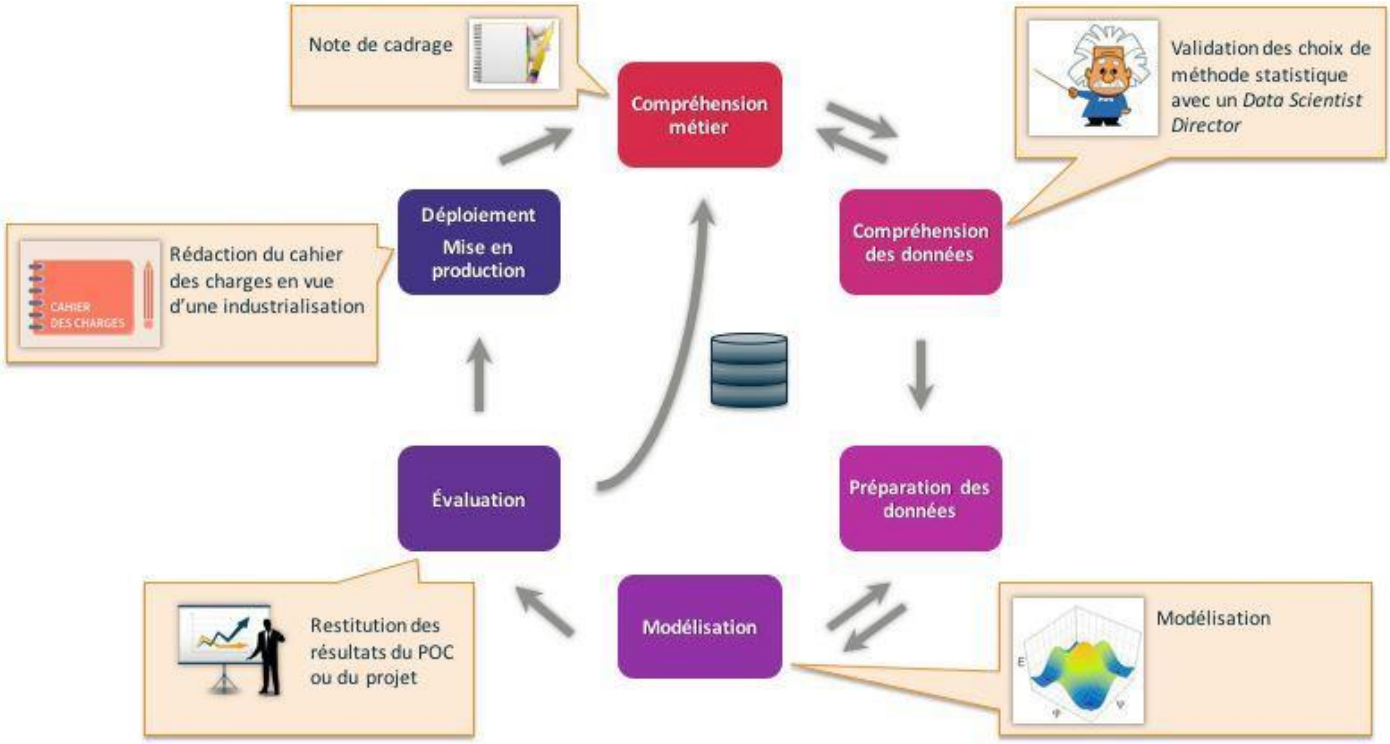


Figure 1. La méthode CRISP illustrée

3.2 Les étapes suivies

3.2.1 La compréhension du problème métier

Cette étape consiste à bien comprendre les éléments métiers et problématiques que la Data Science vise à résoudre ou à améliorer.

Après l'analyse de tous les sujets proposés par notre enseignant et l'utilisation de l'internet pour faire une petite recherche de chacun, nous nous sommes mis d'accord sur le sujet « Le Data Mining pour la prévision de succès des films » vu la disponibilité d'un « dataset » gratuit, et notre motivation et inspiration dans le domaine de cinéma et des films bien évidemment.

Le choix du sujet est fait, ainsi nous avons essayé de comprendre l'aspect et les éléments métiers de notre application pour préciser nos besoins fonctionnels rédigés ci-dessus.

La prévision du succès des films nécessite l'utilisation du domaine de « data mining » pour l'analyse des données, ainsi que l'utilisation des algorithmes de classification supervisés qui permet de classer les films selon leurs succès. Le choix des classes doit être bien étudié.

Le succès ou l'échec du film dépend de son rang dans le box-office. On parle également d'« échelle de succès », de « classement » calculé d'après le montant des recettes. Il s'agit d'une donnée initialement destinée aux professionnels, notamment pour permettre la répartition des recettes et aider à la programmation. C'est devenu un argument de promotion avec une communication axée sur les records et l'idée sous-jacente que l'œuvre mérite d'être consommée car elle a rencontré un important succès public.

Pour classer un film sorti, il faut calculer le rapport du budget avec le revenu. On a besoin de cette étape pour construire le dataset d'entraînement. On se base sur cette dernière pour prédire la classe d'un nouveau film.

3.2.2 La compréhension des données

Cette phase vise à déterminer précisément les données à analyser, à identifier la qualité des données disponibles et à faire le lien entre les données et leur significations d'un point de vue métier.

Les individus de notre projet sont les films et chaque film possède plusieurs données.
Les données que nous avons trouvées dans notre « dataset » sont :

- **Budget** : Outil pratique, il regroupe les informations nécessaires à l'élaboration des budgets de production : établissement du budget prévisionnel, recherche des moyens nécessaires au financement du projet, détermination du coût plateau, amortissement du déficit de production.
- **Genres** du film : Les genres cinématographiques se définissent par les thèmes principaux du film et par la manière de le traiter.
- **Keywords** : les mots clés du film. Ils sont liés au thème et à l'histoire du film, ainsi les mots les plus répétés dans le film par les acteurs.
- **Origine** : La langue parlée au sien du film.
- **Production_companies** : Les entreprises ou les studios de production du film.
- Le revenu : Le revenu générés par le film.
- **Runtime** : La durée du film.
- **Title** : Le titre du film.
- **Id** : L'identifiant du film associé à la base de données de la propriétaire du « dataset ».
- **Homepage** : Le site officiel de l'entreprise productrice.
- **Overview** : Un résumé du film.
- **Popularity** : Une approximation sur la renommée du film par le public. Cad La popularité du film dans le monde.
- **Production_countries** : Les pays de production.
- **Release_date** : La date de sortie du film.
- **Spoken_languages** : Les dialogues du film traduits en plusieurs langues.
- **Tagline** : des slogans liés au film.
- **Vote_average** : L'évaluation du film par un nombre précis de personnes.
- **Vote_count** : Le nombre de personnes qui votent pour le film.
- **Cast** : Le cast du film, cad les acteurs principaux et secondaires du film.

- **Crew** : L'équipe du film, y compris le directeur et le scénariste.
- Nous avons remarqué que les données sont riches, mais dans notre cas il y a ceux qui n'ont pas une valeur ajoutée qu'il faut les éliminer pour optimiser la taille de « dataset »

3.2.3 La construction du Data Hub

Cette phase de préparation des données regroupe les activités liées à la construction de l'ensemble précis des données à analyser, faite à partir des données brutes. Elle inclut ainsi le classement des données en fonction des critères choisis, le nettoyage des données.

Notre « dataset » est de format csv, et divisée en deux fichiers, donc nous avons fusionné ces fichiers et nous avons ajouté, supprimé et modifié des colonnes avec les bibliothèques « pandas » et « csv » du python.

Nous avons fixé d'abord les données qui nous aident à obtenir un résultat efficace en ayant une influence sur la classification, et puis nous avons élevé les autres qui sont :

- **Id** (il n'a aucune information sur le film)
- **Homepage** (le site de l'entreprise productrice est inutile puisque le champ « Production_companies » posséde les noms de ces entreprises)
- **Overview** (le résumé est inutile puisque nous avons les mots-clés)
- Production_countries (le champ « Production_companies » posséde les noms de ces entreprises)
- **Spoken_languages** (Le champ Origine possède la langue officielle du film)
- **Tagline** (le slogan est inutile puisque nous avons les mots-clés du film)
- Ensuite, nous avons utilisé le budget et le revenu pour savoir si le film a rentabilisé son budget ou pas en calculant le pourcentage pour lui classer comme suit :

Le rapport revenu/budget	L'état du film	Equivalent dans le « dataset »
100% <	FLOP	0
100% < < 125%	AVERAGE	1
125% < < 200%	HIT	2
200% < < 300%	SUPER-HIT	3
> 300%	BLOCKBUSTER	4

Figure 2. Tableau des états d'un film dans le Box-office

Voici le code pour la création des classes

```
#Cleaning the data
for s in lines :
    if o == 0 :
        y.append(s)
        o = o+1
    else:
        if s[0] != '' and s[5] != '':
            if int(s[0]) != 0 and int(s[5]) != 0:
                y.append(s)
            else :  c=c+1
#Calculating the ratio
for line in y:
    if line_count == 0:
        line_count += 1
    else :
        print(line[0],"and",line[5])
        d=float(line[5])/float(line[0])
        #print(d)
        if d < 1:
            #out="flop"
            out=0
        elif d > 1 and d <= 1.25 :
            #out="average"
            out=1
        elif d > 1.25 and d <= 2:
            #out="hit"
            out=2
        elif d > 2 and d <= 3:
            #out="super-hit"
            out=3
        else :
            #out="blockbuster"
            out=4
        line[12]=out
```

Figure 3. Un aperçu de code de création de classe

Après la construction de notre classe et la suppression de l'attribut revenu, Nous avons modélisé le cast et le crew d'une manière simple puisqu'ils sont compliqués et de format json. Nous avons modifié aussi les attributs "genres" et "production_compagnies" en enlevant les parties non nécessaires.

Pour l'attribut cast : Nous avons parcouru toute notre dataset et pour chaque acteur différent, nous avons compté le nombre total de film auxquels il a participé ainsi que le nombre de film auxquels il a participé et qui ont fait au moin un "hit". Puis le ratio (Nombre de film réussi auquel il a participé/ Nombre de film total auquel il a participé) est calculé pour chaque acteur. Finalement pour chaque film, on calcule la moyenne des ratios des acteurs de ce film est en fonction du résultat une note est attribué à l'ensemble du cast.

```
for i in range(0, 4803):
    for x in df2['cast'][i]:
        moviesOfActor[x['name']] = moviesOfActor[x['name']] + 1
        if df2['vote_average'][i] > 6:
            successfullMoviesOfActor[x['name']] = successfullMoviesOfActor[x['name']] + 1
    for crew in df2['crew'][i]:
        if crew['job'] == 'Director':
            moviesOfDirectors[crew['name']] = moviesOfDirectors[crew['name']] + 1
            if df2['vote_average'][i] > 6:
                successfullMoviesOfDirectors[crew['name']] = successfullMoviesOfDirectors[crew['name']] + 1
            # a = True
        elif crew['job'] == 'Writer':
            moviesOfWriters[crew['name']] = moviesOfWriters[crew['name']] + 1
            if df2['vote_average'][i] > 6:
                successfullMoviesOfWriters[crew['name']] = successfullMoviesOfWriters[crew['name']] + 1
```

Figure 4. Aperçu du code pour le calcul du nombre de film total et réussi de chaque acteur

```
a = 0
for x in df2['cast'][i]:
    a = a + successfullMoviesOfActor[x['name']] / moviesOfActor[x['name']]
if len(df2['cast'][i]) > 0:
    a = a / len(df2['cast'][i])
    if a < 0.4:
        df2['castgrade'].iat[i] = '0'
    elif a < 0.65:
        df2['castgrade'].iat[i] = '1'
    else:
        df2['castgrade'].iat[i] = '2'
```

Figure 5. Aperçu du code pour l'attribution de la note à chaque cast d'un film

Pour l'attribut crew : De même que pour l'attribut cast, pour l'attribut crew on s'est intéressé principalement au "producer" et au "writer". La même méthode précédente a été utilisé, c'est à dire, pour chaque producer et writer on compte le nombre total de film dans lesquels il a participé et le nombre de ces films qui ont fait au moins un "hit". Ensuite le ratio (Nombre de film réussi auquel il a participé/ Nombre de film total auquel il a participé) et un note est attribué à chaque chaque producer et à chaque writer.

```
for crew in df2['crew'][i]:
    if crew['job'] == 'Director':
        a = successfullMoviesOfDirectors[crew['name']] / moviesOfDirectors[crew['name']]
        if a < 0.4:
            df2['directorGrade'].iat[i] = '0'
        elif a < 0.65:
            df2['directorGrade'].iat[i] = '1'
        else:
            df2['directorGrade'].iat[i] = '2'
        # tmp1 = True
    elif crew['job'] == 'Writer':
        a = successfullMoviesOfWriters[crew['name']] / moviesOfWriters[crew['name']]
        if a < 0.4:
            df2['writerGrade'].iat[i] = '0'
        elif a < 0.65:
            df2['writerGrade'].iat[i] = '1'
        else:
            df2['writerGrade'].iat[i] = '2'
```

Figure 6: Aperçu du code pour l'attribution de la note à chaque writer et producer d'un film

Pour les attributs "genres" et "production_compagnies" : Le format initial contenait un "id" pour chaque genre et pour chaque compagnie de production différente, nous avons donc supprimer ces id et toutes informations non necessaire et n'avons gardé que les différents nom des compagnies et des genres de chaque film.

```
df['production_companies'] = df['production_companies'].apply(json.loads)

for i in range(0, 4803):
    tab = []
    for prod in df['production_companies'][i]:
        tab.append(prod['name'])
    df['production_companies'].iat[i] = tab

df['genres'] = df['genres'].apply(json.loads)

for i in range(0, 4803):
    tab = []
    for prod in df['genres'][i]:
        tab.append(prod['name'])
    df['genres'].iat[i] = tab
```

Figure 7. Aperçu du code de cleaning des attributs "genres" et "production_compagnies"

Après tous ces étapes, nous avons obtenu une « dataset » bien structurée :

budget	genres	keywords	original_lan	popularity	production_companies	release_date	runtime	title	vote_average	vote_count	castgrade	directorGrade	writerGrade	class
237000000	['Action', 'Adventure', 'Fanta	[{"id": 1463	en	150.437577	['Ingenious Film Partners	10/12/2009	162.0	Avatar	7.2	11800	2	2	2	4
300000000	['Adventure', 'Fantasy', 'Acti	[{"id": 270,	en	139.082615	['Walt Disney Pictures', 'J	19/05/2007	169.0	Pirates of t	6.9	4500	2	1	1	4
245000000	['Action', 'Adventure', 'Crime	[{"id": 470,	en	107.376788	['Columbia Pictures', 'Dar	26/10/2015	148.0	Spectre	6.3	4466	2	2	1	4
250000000	['Action', 'Crime', 'Drama', 'Th	[{"id": 849,	en	112.31295	['Legendary Pictures', 'Wa	16/07/2012	165.0	The Dark Kr	7.6	9106	2	2	1	4
260000000	['Action', 'Adventure', 'Scien	[{"id": 818,	en	43.926995	['Walt Disney Pictures']	07/03/2012	132.0	John Carter	6.1	2124	2	2	1	1
258000000	['Fantasy', 'Action', 'Adventu	[{"id": 851,	en	115.699814	['Columbia Pictures', 'Lau	01/05/2007	139.0	Spider-Mar	5.9	3576	0	2	1	4
260000000	['Animation', 'Family']	[{"id": 1562	en	48.681969	['Walt Disney Pictures', 'V	24/11/2010	100.0	Tangled	7.4	3330	2	2	1	3
280000000	['Action', 'Adventure', 'Scien	[{"id": 8828	en	134.2792290000	['Marvel Studios', 'Prime I	22/04/2015	141.0	Avengers: A	7.3	6767	2	2	2	4
250000000	['Adventure', 'Fantasy', 'Fam	[{"id": 616,	en	98.885637	['Warner Bros.', 'Heyday F	07/07/2009	153.0	Harry Potte	7.4	5293	2	2	1	4
250000000	['Action', 'Adventure', 'Fanta	[{"id": 849,	en	155.790452	['DC Comics', 'Atlas Entert	23/03/2016	151.0	Batman v Si	5.7	7004	0	2	1	4
270000000	['Adventure', 'Fantasy', 'Acti	[{"id": 83, "	en	57.925623	['DC Comics', 'Legendary E	28/06/2006	154.0	Superman I	5.4	1400	0	2	1	2
200000000	['Adventure', 'Action', 'Thrill	[{"id": 627,	en	107.928811	['Eon Productions']	30/10/2008	106.0	Quantum o	6.1	2965	2	2	1	3
200000000	['Adventure', 'Fantasy', 'Acti	[{"id": 616,	en	145.8473790000	['Walt Disney Pictures', 'J	20/06/2006	151.0	Pirates of t	7.0	5246	2	1	1	4
255000000	['Action', 'Adventure', 'West	[{"id": 1556	en	49.046956	['Walt Disney Pictures', 'J	03/07/2013	149.0	The Lone R	5.9	2311	0	1	1	0

Figure 8. Aperçu de la base de données après nettoyage

3.2.4 La modélisation

C'est la phase de Data Science proprement dite. La modélisation comprend le choix, le paramétrage et le test de différents algorithmes ainsi que leur enchaînement, qui constitue un modèle. Ce processus est d'abord descriptif pour générer de la connaissance, en expliquant pourquoi les choses se sont passées. Il devient ensuite prédictif en expliquant ce qu'il va se passer, puis prescriptif en permettant d'optimiser une situation future.

Nous avons utilisé les algorithmes supervisés de classification suivants :

- **Decision Tree** : utilise un représentation arborescente. Chaque nœud interne de l'arbre correspond à un attribut et chaque nœud feuille correspond à une étiquette de classe.
- **Random Forest** : combine des prédicteurs d'arbres où chaque arbre dépend des valeurs d'un vecteur aléatoire échantillonné indépendamment avec la même distribution pour tous les arbres de la forêt.
- **Naive Bayes** : est un classificateur simples et probabilistes qui utilisent le théorème de Bayes.
- **k_Neighbors** : l'entrée comprend les k exemples d'apprentissage les plus proches dans l'espace descriptif.

En utilisant la bibliothèque sklean, on a utilisé le classificateurs **DecisionTreeClassifier**, **KNeighborsClassifier**, **RandomForestClassifier** et **GaussianNB**. Ces classificateurs ne supportent pas les données catégoriques. C'est pourquoi, on a besoin premièrement d'encoder ces données afin d'obtenir des données numériques. Ensuite, on doit enlever la colonne « class » du dataset et la stocker dans une autre variable pour l'apprentissage.

Pour cela, on a implémenté la fonction suivante :

```
mapper={}
def PrepareData(df):
    # replace the Nan with "NA" which acts as a unique category
    df.fillna("NA", inplace=True)
    # make list of all column headers
    categorical_list = list(df.columns.values)
    #exclude the class column
    categorical_list.remove('class')
    newdf = pd.DataFrame(columns=categorical_list)
    #Converting Categorical Data to integer labels
    for x in categorical_list:
        mapper[x]=preprocessing.LabelEncoder()
    for x in categorical_list:
        if (df.dtypes[x]==object):
            newdf[x]= mapper[x].fit_transform(df.__getattr__(x).astype(str))
        else:
            newdf[x]=df.__getattr__(x)
    print(newdf)
    # make a class series encoded :
    le = preprocessing.LabelEncoder()
    myclass = le.fit_transform(df.__getattr__('class'))
   #newdf is the dataframe with all columns except class column and myclass is the class column
    return newdf, myclass , categorical_list
```

Figure 9. Aperçu du code de préparation et d'encodage de la base de données

On a choisi d'utiliser la classe LabelEncoder pour l'encodage. Chaque colonne a son propre LabelEncoder stocké dans mapper.
Voici la nouvelle DataFrame obtenu :

budget	genres	keywords	original_language	popularity	production_companies	production_countries	release_date	runtime	title	vote_average	vote_count	cast	crew
237000000	61	534	7	150.437577	1394	440	2313	64	379	7.2	11800	3229	2601
300000000	338	1608	7	139.082615	3535	465	1943	70	2648	6.9	4500	3819	109
245000000	41	2617	7	107.376788	532	299	3183	50	3181	6.3	4466	867	3418
250000000	131	3892	7	112.312950	1561	465	2686	67	3612	7.6	9106	2650	2498
260000000	75	3778	7	43.926995	3572	465	2633	33	1901	6.1	2124	4194	2525
258000000	812	3894	7	115.699814	570	465	1938	40	3193	5.9	3576	3730	202

Figure 10. Extrais de la base de données après encodage

On a besoin maintenant de découper cette database en une partie pour l'apprentissage et une autre pour le test, en utilisant la méthode « train_test_split »
On teste chaque algorithme et on calcule le taux d'erreurs avec les deux méthodes suivantes :

```
def test(nb):
    nb.fit(X_train,y_train)
    P=nb.predict(X_test)
    return erreur2(P,y_test)
def erreur2(test_data,test_target):
    return (1-accuracy_score(test_data,test_target))
```

Figure 11. Méthodes de test des algorithmes

3.2.4.1 L'évaluation

L'évaluation vise à vérifier le modèle ou les connaissances obtenues afin de s'assurer qu'ils répondent aux objectifs formulés au début du processus. Elle contribue aussi à la décision de déploiement du modèle ou, si besoin est, à son amélioration. A ce stade, on teste notamment la robustesse et la précision des modèles obtenus.

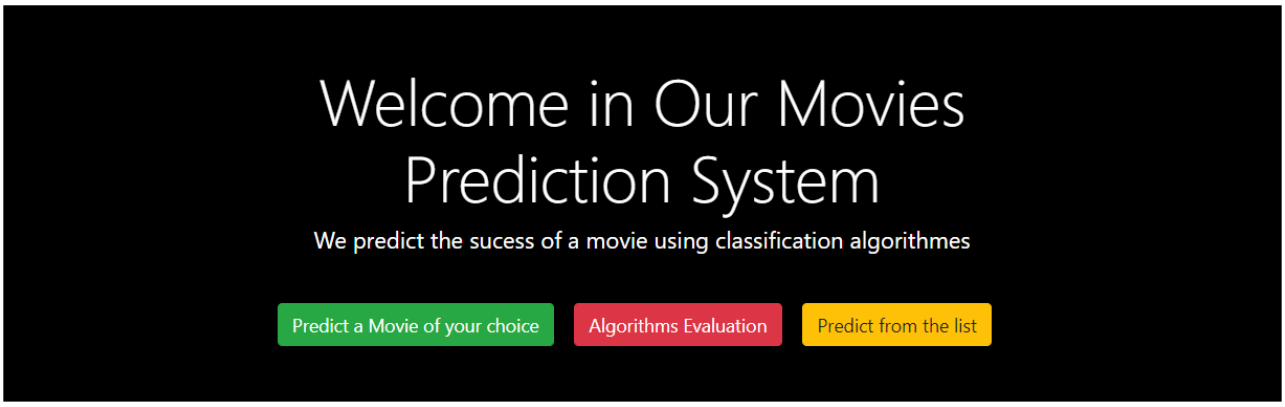
- ✓ Taux d'erreur Decision Tree= 0.242
- ✓ Taux d'erreur Random Forest=0.18
- ✓ Taux d'erreur Naive Bayes=0.26
- ✓ Taux d'erreur k_neighbors=0.26

3.2.5 Le déploiement

Il s'agit de l'étape finale du processus. Elle consiste en une mise en production pour les utilisateurs finaux des modèles obtenus. Son objectif est de mettre la connaissance obtenue par la modélisation, dans une forme adaptée, et l'intégrer au processus de prise de décision.
Le déploiement peut ainsi aller, selon les objectifs, de la simple génération d'un rapport décrivant les connaissances obtenues jusqu'à la mise en place d'une application, permettant l'utilisation du modèle obtenu, pour la prédiction de valeurs inconnues d'un élément d'intérêt.

Pour la création de note application, nous avons créé une API python qui utilise les méthodes déjà implémentés et une application Angular comme front end.
Notre application comporte 3 parties :

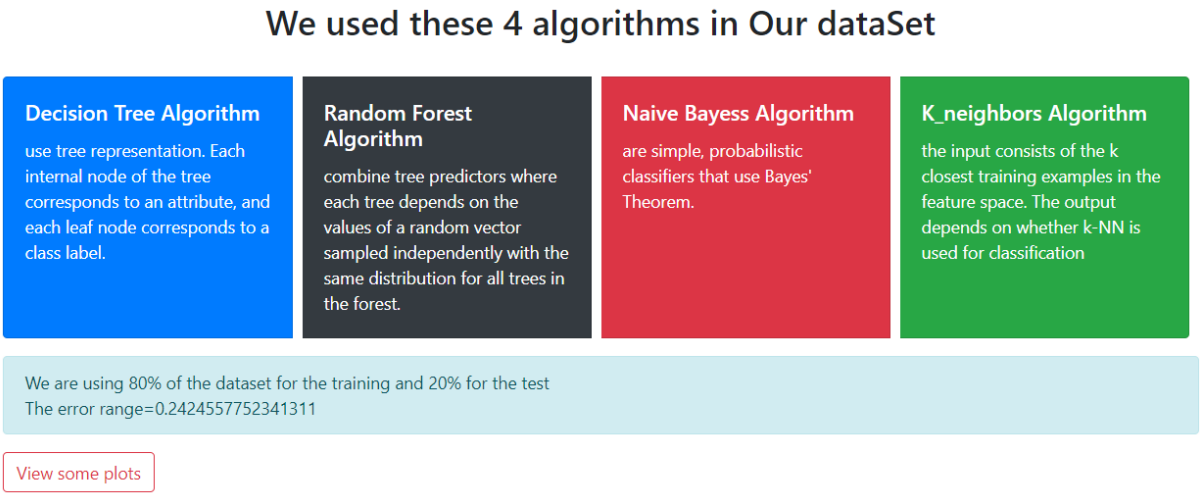
1. Test Des algorithmes et observation graphes :
2. Prédiction à l'aide d'un formulaire
3. Prédiction à partir des films existants.



Figue 12. Page d'accueil

➢ <u>Test Des algorithmes et observation graphes</u>

On présente ici, le taux d'erreurs de l'utilisation de chaque algorithme sur notre base de données en utilisant 80% pour l'apprentissage et 20% pour le test.



Figue 13. Page d'évaluation des algorithmes

➢ <u>Prédiction à l'aide d'un formulaire</u>

En remplissant le formulaire suivant, l'utilisateur peut prédire le succès d'un film

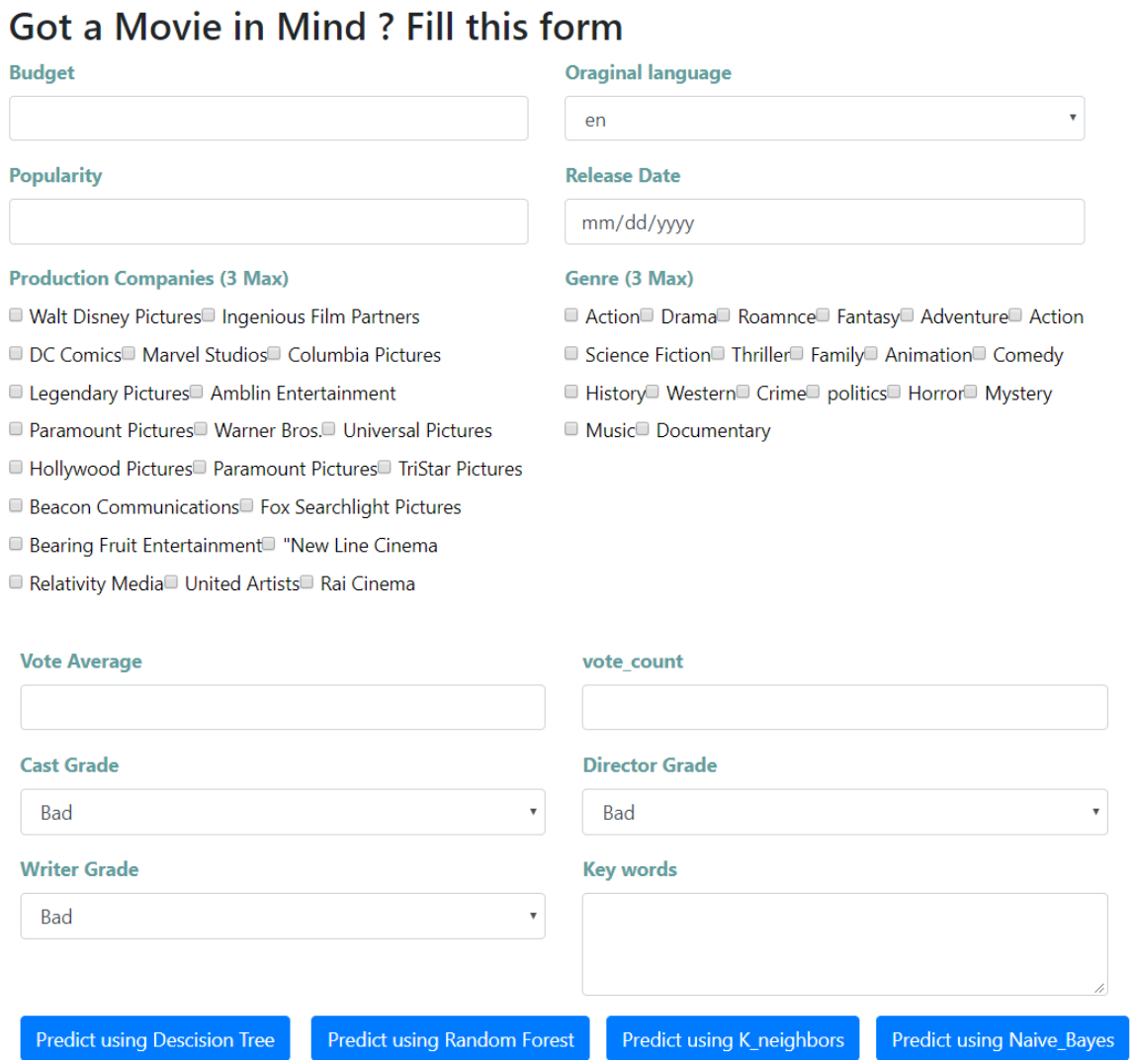


Figure 14. Formulaire film à prédire

➢ <u>Prédiction à partir des films existants :</u>

Ici, on propose des films avec des valeurs réelles qui sont exclus de la base de données dans l'apprentissage et on demande au système de prédire la classe du film sélectionné.

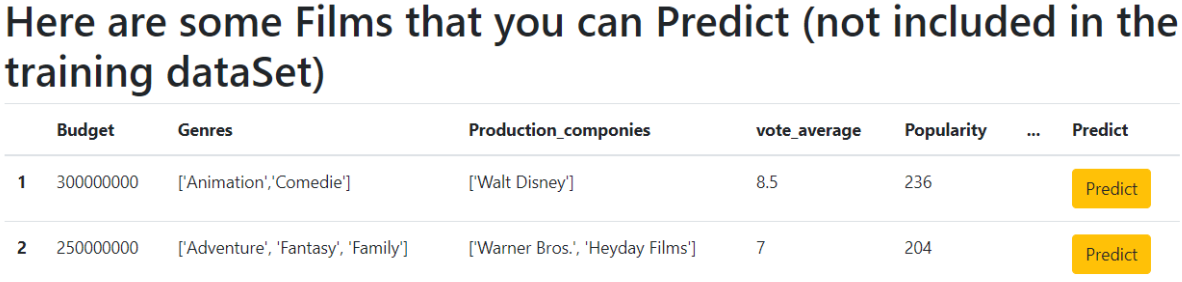


Figure 15. Page prédiction films existants

4 Conclusion :

En développant cette application, nous avons pu se familiariser avec le métier de « Data Scientist » et nous avons pu expérimenter la bibliothèque sklean . Notre application est en cours de développement et peut être de plus en plus amélioré.