ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

Réalisation d'un modèle de classification des plaintes des consommateurs

Réalisé par :

OURABAH Fatima Zahra

SAIDINE Meryem

Encadré par :

Prof Said Bahassine

ANNÉE SCOLAIRE 2022-2023

Table de matières :

I. Introduction :

Dans notre ère numérique actuelle, les données textuelles sont générées à un rythme exponentiel grâce aux avancées technologiques, à l'utilisation généralisée des médias sociaux et à la communication en ligne. Ces vastes quantités de données textuelles, telles que les commentaires, les avis, les tweets, les messages sur les forums, les e-mails, etc., renferment une mine d'informations précieuses. Cependant, extraire des connaissances significatives et exploitables à partir de ces données brutes est un défi complexe.

C'est là que l'importance du text mining (ou fouille de texte) entre en jeu. Le text mining est un domaine de l'intelligence artificielle qui se concentre sur l'analyse et l'extraction d'informations utiles à partir de textes non structurés. Il utilise des techniques d'apprentissage automatique, de traitement du langage naturel et de fouille de données pour comprendre, catégoriser et interpréter les données textuelles.

Dans cette étude, nous nous concentrerons sur l'application du text mining dans le domaine des plaintes des consommateurs pour les classifier. Nous nous appuyons sur le jeu de données "Consumer Complaints" accessible via le lien https://data.world/data-society/consumer-complaint-data, pour étudier les plaintes des consommateurs. L'analyse de ces données textuelles permet de découvrir les principaux problèmes, les termes ou expressions couramment utilisés par les consommateurs dans leurs plaintes, et les différents aspects qui ont un impact sur leur satisfaction et leur confiance.

Ce jeu de données offre une opportunité d'exploration et d'analyse approfondie des plaintes des consommateurs. Il fournit des informations précieuses pour comprendre les préoccupations des clients, améliorer les produits ou services, et renforcer la relation entre les entreprises et les consommateurs.

II. Description générale de cette étude :

L'objectif principal de cette étude est d'analyser et de classifier les plaintes des consommateurs en fonction des problèmes mentionnés dans les plaintes. En utilisant des techniques de text mining telles que l'apprentissage automatique et la sélection de fonctionnalités, nous chercherons à extraire des informations significatives des plaintes des consommateurs afin de mieux comprendre les problèmes auxquels les consommateurs sont confrontés et d'aider les entreprises à améliorer leurs produits, leurs services et leur relation client.

Dans le cadre de notre analyse approfondie des données, nous nous concentrerons sur les colonnes "Issue", "Consumer complaint narrative" et "Product" du jeu de données. Notre objectif sera d'explorer la distribution des plaintes, de détecter les tendances émergentes et d'établir des relations potentielles avec d'autres variables présentes dans le jeu de données. Pour atteindre cet objectif, nous mettrons en œuvre des techniques d'apprentissage automatique telles que le Support Vector Machine (SVM) ainsi que des méthodes de sélection de caractéristiques telles que l'information mutuelle (MI) et un algorithme génétique (GA). Nous détaillerons ces méthodes et discuterons de leur pertinence dans le contexte de notre tâche de classification.

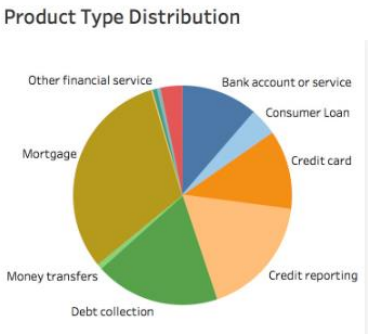
De plus, nous fournirons des explications détaillées sur la mise en œuvre du code utilisé pour la classification des plaintes des consommateurs, en décrivant les différentes étapes, les paramètres utilisés et les décisions prises pour aboutir à notre modèle final.

III. Etude du dataset :

1. Description du dataset

Le jeu de données "Consumer Complaints", accessible via le lien https://data.world/data-society/consumer-complaint-data, regroupe une vaste collection de plaintes de consommateurs déposées auprès de différentes entreprises entre 2013 et 2016. Ce jeu de données constitue une ressource précieuse pour comprendre les problèmes et les préoccupations rencontrés par les consommateurs dans divers secteurs, tels que les services financiers, les prêts, les cartes de crédit, les services bancaires, etc.

Avec plus de 600 000 lignes, ce jeu de données comprend plusieurs colonnes fournissant des informations clés. Par exemple, la colonne "Date reçue" indique la date d'enregistrement de la plainte, tandis que la colonne "Product" classe les plaintes en fonction des 12 catégories de produits distinctes. La colonne "Issue" décrit le problème ou la préoccupation soulevé(e) par le consommateur, et la colonne "Consumer complaint narrative" contient une description détaillée de la plainte.



Ce jeu de données s'avère particulièrement utile pour les entreprises et les chercheurs souhaitant analyser les plaintes des consommateurs et améliorer l'expérience client. En examinant les données, il est possible d'identifier les problèmes récurrents, les tendances émergentes et les domaines nécessitant des améliorations. Ainsi, les entreprises peuvent prendre des mesures correctives, adapter leurs produits ou services et renforcer la satisfaction client.

2. Choix des données utiles

Dans le cadre de cette étude, nous avons entrepris d'analyser un jeu de données initial comportant plus de 600 000 lignes, ce qui dépassait les capacités de traitement de nos ordinateurs. Afin de rendre le problème plus gérable, nous avons décidé de nous concentrer sur deux colonnes spécifiques : "Issue" (problème) et "Product" (produit) qui contient les classes des plaintes.

La colonne "Issue" contenait des problèmes spécifiques exprimés succinctement. Nous avons identifié les problèmes uniques dans cette colonne, qui se sont avérés au nombre de 95. Nous avons ensuite construit notre algorithme en nous basant sur ces problèmes distincts. L'algorithme a démontré une précision élevée pouvant atteindre 93% lors des tests utilisant les données sur lesquelles nous avions entraîné le modèle. Les résultats obtenus étaient précis dans ces conditions.

In [1]:
```python
import pandas as pd

# Load the dataset
data = pd.read_csv('Consumer_Complaints.csv')  # Update with the actual dataset file name

# Get unique classes in the "Issue" column
classes = data['Issue'].unique()
num_classes = len(classes)

# Print the unique classes and the number of classes
print("Unique Classes:")
for cls in classes:
    print(cls)
print("Number of Classes:", num_classes)
```

Cependant, lorsqu'il s'agissait de classifier de nouvelles plaintes, telles que "J'ai des difficultés à effectuer les paiements de mon prêt automobile", le modèle les classait systématiquement dans la catégorie "Carte de crédit", ce qui était incorrect. Nous avons effectué plusieurs tests et avons constaté que ce problème était lié aux données sur lesquelles nous avions initialement entraîné le modèle. C'était comme si nous avions entraîné le modèle uniquement sur l'objet d'un e-mail Gmail, par exemple, et que nous testions ensuite le modèle avec l'intégralité du message. Cette incohérence était donc attendue.

Dans le but de résoudre ce problème, nous avons entrepris de revoir notre approche en travaillant sur une autre colonne, à savoir "Consumer Complaint Narrative", qui contenait des descriptions textuelles plus détaillées des plaintes. L'objectif était de sélectionner les plaintes distinctes dans cette colonne, mais leur nombre dépassait les 100 000. Nous avons alors opté pour une sélection aléatoire d'un sous-ensemble de ces plaintes, tout en veillant à conserver les 12 classes de produits dans notre échantillon.

In [ ]:
```
import pandas as pd
import random
data = pd.read_csv('Consumer_Complaints.csv')
data = data.dropna(subset=['Consumer complaint narrative'])
num_samples = 1500
random_indices = random.sample(range(len(data)), num_samples)
random_data = data.iloc[random_indices]
new_data = random_data[['Consumer complaint narrative', 'Product']]
new_data.to_csv('Consumer complaint.csv', index=False)
```

Cette nouvelle approche a permis de résoudre le problème de classification erronée. En travaillant avec la colonne "Consumer Complaint Narrative", notre modèle a réussi à classifier avec précision les nouvelles plaintes en fonction du produit correspondant. Cette modification a donc permis d'améliorer la précision globale de notre modèle et d'obtenir des résultats plus fiables.

En conclusion, pour surmonter les problèmes de classification incorrecte, nous avons réorienté notre travail en nous appuyant sur la colonne "Consumer Complaint Narrative" et en sélectionnant un sous-ensemble aléatoire de plaintes distinctes tout en conservant l'ensemble des classes de produits. Cette approche a permis d'améliorer la précision de notre modèle et de garantir des résultats plus précis lors de la classification des nouvelles plaintes des consommateurs.

IV. Algorithmes utilisés

1. SVM (Support Vector Machine)

SVM est un algorithme d'apprentissage automatique supervisé servant aux taches de classification et régression.

Dans l'IA et l'apprentissage automatique, les systèmes d'apprentissage supervisé fournissent à la fois des données d'entrée et des données de sortie souhaitées, qui sont catégorisées en vue d'une classification. La classification fournit une base d'apprentissage pour le traitement futur des données. Les machines à vecteurs de support sont utilisées pour trier deux groupes de données en fonction d'une classification similaire.

SVM construit un modèle d'apprentissage qui affecte les nouveaux exemples à un groupe ou à un autre. Grâce à ces fonctions, les SVM sont appelés classificateurs linéaires binaires non probabilistes.

Comme d'autres machines d'apprentissage supervisé, un SVM nécessite des données classifiées pour être entraîné. Les groupes de matériaux sont étiquetés pour la classification. Les matériaux d'entraînement pour les SVM sont classés séparément en différents points de l'espace et organisés en groupes clairement séparés. Après avoir traité de nombreux exemples de formation, les SVM peuvent effectuer un apprentissage non supervisé. Les algorithmes tentent d'obtenir la meilleure séparation des données en maximisant la frontière autour de l'hyperplan et en assurant l'égalité entre les deux côtés.

Voici quelques-uns des principaux avantages des SVM :

- Efficace dans les cas à haute dimension.
- Sa mémoire est efficace car il utilise un sous-ensemble de points d'entraînement dans la fonction de décision, appelés vecteurs de support.
- Différentes fonctions de noyau peuvent être spécifiées pour les fonctions de décision et il est possible de spécifier des noyaux personnalisés.

SVC (Support Vector Classifier) :

SVC est une implémentation spécifique de l'algorithme Support Vector Machine (SVM) pour les tâches de classification. Dans scikit-learn, la bibliothèque Python d'apprentissage automatique, la classe SVC est utilisée pour créer un classificateur basé sur l'algorithme SVM.

La classe SVC de scikit-learn offre une interface permettant d'entraîner un modèle SVM sur des données d'entraînements labellisés et de l'utiliser pour faire des prédictions sur de nouvelles données inédites. Elle prend en charge la classification binaire et la classification multi classe.

2. Feature selection

Feature selection est le processus qui consiste à sélectionner un sous-ensemble de termes apparaissant dans l'ensemble d'apprentissage et à n'utiliser que ce sous-ensemble comme caractéristiques dans la classification des textes. Feature selection a deux objectifs principaux. Premièrement, elle rend la formation et l'application d'un classificateur plus efficaces en réduisant la taille du vocabulaire effectif. Ceci est particulièrement important pour les classificateurs qui, contrairement à NB, sont coûteux à former. Deuxièmement, Feature selection augmente souvent la précision de la classification en éliminant les caractéristiques parasites. Une caractéristique parasite est une caractéristique qui, lorsqu'elle est ajoutée à la représentation du document, augmente l'erreur de classification sur de nouvelles données. Dans cett etude feature selection sera implémenté avec un algorithme genetique et puis avec Mutual information.

a) Algorithme génétique :

Dans l'exploration de texte, un algorithme génétique peut nous aider à sélectionner les mots ou les phrases les plus importants et les plus pertinents du texte afin d'améliorer la précision de la classification. L'algorithme évalue différentes combinaisons de mots et sélectionne celles qui contribuent le plus à distinguer différentes catégories ou classes de texte. Ce faisant, nous pouvons identifier les caractéristiques clés qui ont le plus d'impact sur la tâche de classification et améliorer les performances globales de nos modèles de classification de texte.

b) Mutual information (MI) :

La sélection de caractéristiques par mutual information est une technique utilisée pour sélectionner les caractéristiques les plus informatives d'un ensemble de données sur la base de leur mutual information avec la variable cible. Elle est couramment employée dans les tâches d'apprentissage automatique et text mining afin d'identifier les caractéristiques les plus pertinentes pour la classification.

V. Construction du modèle de classification

1. SVM avec MI :

Explication du code et des résultats :

Entrée [ ]:
```
import pandas as pd
import random
data = pd.read_csv('Consumer_Complaints.csv')
data = data.dropna(subset=['Consumer complaint narrative'])
num_samples = 1500
random_indices = random.sample(range(len(data)), num_samples)
random_data = data.iloc[random_indices]
new_data = random_data[['Consumer complaint narrative', 'Product']]
new_data.to_csv('Consumer complaint.csv', index=False)
```

Nous commençons par lire notre fichier CSV qui contient les données en utilisant la bibliothèque pandas, puis nous supprimons toutes les lignes qui pourraient ne contenir aucun type de données (NaN), ensuite nous choisissons de sélectionner 1500 échantillons pour travailler mais aussi de conserver tous les nombres de classes, puis nous stockons "consumer complaint narrative" qui représente les textes avec lesquels nous voulons entraîner notre algorithme et "product" qui représente les classes dans un autre fichier CSV avec lequel nous allons travailler.

Entrée [1]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.feature_selection import SelectKBest, mutual_info_classif
```

Entrée [2]:
```
combined_table = pd.read_csv('Consumer complaint.csv')
```

Entrée [3]:
```
X = combined_table['Consumer complaint narrative']
y = combined_table['Product']
```

Entrée [4]:
```
vectorizer = TfidfVectorizer()
X_vectorized = vectorizer.fit_transform(X)
```

Entrée [5]:
```
mi_scores = mutual_info_classif(X_vectorized, y)
```

Ici, après avoir importé les bibliothèques nécessaires, nous importons également nos données, puis nous utilisons le TfidfVectorizer qui est utilisé pour transformer une collection de documents textuels en une matrice qui peut être utilisée comme entrée pour les algorithmes d'apprentissage automatique.

Pour ‘mutual_info_classif (X_vectorized, y)’, elle est utilisée pour calculer les scores d'information mutuelle entre les caractéristiques (X_vectorized) et la variable cible (y).

Entrée [6]:
```
sorted_features = np.array(vectorizer.get_feature_names_out())[np.argsort(mi_scores)[::-1]]
```

Entrée [7]:
```
k = 1000
selected_features = sorted_features[:k]
```

Entrée [8]:
```
X_selected = X_vectorized[:, [vectorizer.vocabulary_.get(feature) for feature in selected_features]]
```

Entrée [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2, random_state=42)
```

Entrée [10]:
```
svm_model = SVC()
svm_model.fit(X_train, y_train)
```

Ensuite, nous utilisons mutual information pour feature selection. Dans notre cas, nous ne voulons que 1000 caractéristiques et donc 'selected_features' sélectionnera les 1000 meilleures caractéristiques sur la base de leur score dans 'sorted_features'. Ensuite, après avoir indexé la matrice 'X_vectorized' avec les caractéristiques sélectionnées, nous commençons l'apprentissage.

```
Entrée [20]: new_complaint = "The loan servicer provided incorrect information about loan forgiveness options. "

Entrée [21]: new_complaint_vectorized = vectorizer.transform([new_complaint])
             new_complaint_selected = new_complaint_vectorized[:, [vectorizer.vocabulary_.get(feature) for feature i

Entrée [22]: predicted_class = svm_model.predict(new_complaint_selected)
             print(f"The new complaint belongs to the class: {predicted_class}")

             The new complaint belongs to the class: ['Mortgage']

Entrée [14]: y_pred = svm_model.predict(X_test)

Entrée [15]: accuracy = np.mean(y_pred == y_test)
             print(f"Accuracy: {accuracy}")

             Accuracy: 0.7233333333333334
```

Une fois training est terminé, notre modèle est prêt à être testé. Nous préparons donc nos données de test en suivant les mêmes étapes appliquées aux données de formation et nous commençons à tester.
Dans notre cas, nous avons testé différentes phrases et le modèle est censé prédire leurs classes. Nous pouvons voir que notre modèle fonctionne et qu'il a été capable de prédire la bonne classe avec une précision de 72 %.

```
Entrée [16]: print(classification_report(y_test, y_pred))

                              precision    recall  f1-score   support

          Bank account or service       0.86      0.55      0.67        22
                     Consumer Loan       1.00      0.13      0.23        23
                       Credit card       0.62      0.66      0.64        32
                 Credit reporting       0.74      0.82      0.78        67
                  Debt collection       0.67      0.87      0.76        70
                  Money transfers       0.00      0.00      0.00         2
                         Mortgage       0.75      0.87      0.81        67
                      Payday loan       0.00      0.00      0.00         2
                     Prepaid card       0.00      0.00      0.00         3
                     Student loan       1.00      0.58      0.74        12

                         accuracy                           0.72       300
                        macro avg       0.56      0.45      0.46       300
                     weighted avg       0.74      0.72      0.69       300
```

Les résultats de l'évaluation indiquent que le modèle de classification est relativement performant pour certaines catégories telles que "Debt collection", "Mortgage" et "Credit reporting". Ces classes ont une précision, un recall et un F1-scores plus élevés, ce qui suggère que le modèle peut identifier avec précision les instances appartenant à ces catégories. Cependant, pour des classes telles que "Money transfers", "Prepaid card" et "Payday loan", les performances du modèle sont nettement plus faibles. La précision, le recall et les F1-scores pour ces classes sont faibles, ce qui indique que le modèle a du mal à classer correctement les instances de ces catégories. Il est important de noter que l'absence d'étapes de prétraitement et de stemming dans les données peut avoir affecté les performances du modèle. Les techniques de prétraitement, telles que la suppression des stop words et l'application du stemming, peuvent améliorer la capacité du modèle à généraliser et à identifier des mo dèles significatifs dans les données textuelles. Compte tenu de ces facteurs, d'autres améliorations pourraient être apportées en incorporant des étapes de prétraitement et en explorant différentes techniques de sélection des caractéristiques ou différents algorithmes de classification.

2. SVM avec GA :

Lorsque nous avons travaillé avec l'algorithme génétique comme moyen de feature selection, nous avons essentiellement suivi les mêmes étapes que celles utilisées pour mutual information. La seule différence ici est le fait que nous avons d'abord utilisé la colonne 'Issue' pour nos données, l'entraînement et le test ont semblé fonctionner sans problème. Sachant que nous avons entraîné notre modèle seulement avec les 95 données uniques du fichier entier. Lorsque nous avons essayé de travailler avec le fichier de 1500 lignes de 'Consumer complaint narrative', le modèle a pris des heures seulement pendant le processus d'entraînement, c'est pourquoi nous avons choisi de garder la classification avec la colonne 'Issue'. Voici notre code :

Entrée [1]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from deap import creator, base, tools, algorithms
```

Entrée [2]:
```
train = pd.read_csv('combined_table.csv')
```

Entrée [3]:
```
X = train['Issue']
y = train['Products']
```

Entrée [4]:
```
vectorizer = TfidfVectorizer()
X_vectorized = vectorizer.fit_transform(X)
```

Entrée [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2, random_state=42)
```

Entrée [6]:
```
POPULATION_SIZE = 1000
CROSSOVER_PROBABILITY = 0.5
MUTATION_PROBABILITY = 0.2
N_GENERATIONS = 50
```

Entrée [7]:
```
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)
```

Entrée [8]:
```
toolbox = base.Toolbox()
```

Entrée [9]:
```
toolbox.register("attr_bool", np.random.choice, a=[True, False])
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_bool, len(vectorizer.
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("evaluate", lambda individual: evaluate(individual, X_train, y_train, X_test, y_test))
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.1)
toolbox.register("select", tools.selTournament, tournsize=3)
```

Entrée [10]:
```
def evaluate(individual, X_train, y_train, X_test, y_test):
    selected_features = [feature for feature, selected in zip(vectorizer.get_feature_names_out(), indiv
    X_train_selected = X_train[:, [vectorizer.vocabulary_.get(feature) for feature in selected_features
    X_test_selected = X_test[:, [vectorizer.vocabulary_.get(feature) for feature in selected_features]]
    svm_model = SVC()
    svm_model.fit(X_train_selected, y_train)
    y_pred = svm_model.predict(X_test_selected)
    fitness = np.mean(y_pred == y_test)

    return fitness,
```

Entrée [11]:
```
population = toolbox.population(n=POPULATION_SIZE)
final_population, logbook = algorithms.eaSimple(population, toolbox, cxpb=CROSSOVER_PROBABILITY, mutpb=
best_individual = tools.selBest(final_population, k=1)[0]
selected_features = [feature for feature, selected in zip(vectorizer.get_feature_names_out(), best_indi
X_selected = X_vectorized[:, [vectorizer.vocabulary_.get(feature) for feature in selected_features]]
svm_model = SVC()
svm_model.fit(X_selected, y)
```

Entrée [12]:
```
new_complaint = "Unauthorized transactions and fraudulent activities on my bank account!"
new_complaint_vectorized = vectorizer.transform([new_complaint])
new_complaint_selected = new_complaint_vectorized[:, [vectorizer.vocabulary_.get(feature) for feature i
predicted_class = svm_model.predict(new_complaint_selected)
print(f"The new complaint belongs to the class: {predicted_class}")
```

```
The new complaint belongs to the class: ['Credit card']
```

L'algorithme génétique utilisé dans cette étude comprend plusieurs paramètres clés tels que POPULATION_SIZE, CROSSOVER_PROBABILITY, MUTATION_PROBABILITY et N_GENERATIONS. Ces paramètres déterminent respectivement la taille de la population, la probabilité de crossover entre les individus c'est-à-dire la probabilité que deux individus échangent des informations génétiques, et puis la probabilité de mutation des individus et le nombre d'itérations de l'algorithme. De plus, ce code crée des classes personnalisées, FitnessMax et Individual, à l'aide du module "creator". La classe FitnessMax représente la valeur de fitness d'une solution individuelle, tandis que la classe Individual représente une solution individuelle, qui est une liste de valeurs binaires. Cette association entre les classes FitnessMax et Individual indique que la valeur de fitness doit être maximisée. Enfin, le module "toolbox" est utilisé pour définir les fonctions et opérateurs nécessaires à l'algorithme génétique. Il s'agit d'une bibliothèque appelée "deap" qui permet de définir et de manipuler les composants de l'algorithme génétique.

In [4]:
```
from sklearn.metrics import classification_report

# Predict the classes of the test dataset
predicted_classes = svm_model.predict(X_test)

# Compute the accuracy
accuracy = np.mean(predicted_classes == y_test)

# Generate a classification report
report = classification_report(y_test, predicted_classes)

# Print the accuracy and classification report
print(f"Accuracy: {accuracy}")
print(f"Classification Report:\n{report}")
```

```
Accuracy: 0.9473684210526315
Classification Report:
                           precision    recall  f1-score   support

             Consumer Loan       1.00      1.00      1.00         2
               Credit card       0.86      1.00      0.92         6
         Credit reporting        1.00      0.50      0.67         2
          Debt collection        1.00      1.00      1.00         2
                 Mortgage        1.00      1.00      1.00         1
   Other financial service       1.00      1.00      1.00         1
Payday loan, Consumer Loan       1.00      1.00      1.00         3
              Prepaid card       1.00      1.00      1.00         1
              Student loan       1.00      1.00      1.00         1

                  accuracy                           0.95        19
                 macro avg       0.98      0.94      0.95        19
              weighted avg       0.95      0.95      0.94        19
```

Les résultats montrent effectivement une précision élevée de 94,7%, ce qui semble être une bonne performance. Cependant, il est important de noter que les résultats peuvent être trompeurs.

Dans ce cas, il est possible que le modèle ait souffert d'un surapprentissage (overfitting). Le surapprentissage se produit lorsqu'un modèle s'ajuste trop précisément aux données d'entraînement, ce qui peut conduire à de très bonnes performances sur les données d'entraînement, mais à des performances médiocres sur de nouvelles données ou des données de test.

La précision de 86% pour la classe "Credit card" peut indiquer un surapprentissage, où le modèle a tendance à classer la plupart des plaintes dans cette classe spécifique, ce qui conduit à une précision élevée pour cette classe. Cela est produit à cause des données d'entraînement qui ne sont pas suffisamment représentatives de la variabilité réelle des données, et les caractéristiques utilisées pour l'entraînement ne capturent pas pleinement la complexité des différentes classes.

C'est pourquoi nous avons décidé de travailler avec la colonne "Consumer complaint narrative" au lieu de la colonne "Issue". Initialement, nous avons essayé d'utiliser la colonne "Consumer complaint narrative" , mais nous avons rencontré des problèmes d'exécution qui ont pris plus de 12 heures sans réponse. Nous avons également constaté que lorsque nous avons augmenté la taille de l'ensemble de données, l'exécution était trop lente en raison des limitations de performance de nos ordinateurs. Pour surmonter ces problèmes, nous avons choisi d'utiliser la colonne "Issue", qui nous permet d'analyser les plaintes des consommateurs en utilisant l'algorithme génétique et d'obtenir des résultats plus rapidement.

VI. Conclusion

En conclusion, cette étude démontre l'importance du text mining dans l'analyse des plaintes des consommateurs et la classification des problèmes associés. En utilisant des techniques de classification basées sur le SVM et la sélection de caractéristiques, nous avons pu analyser les plaintes des consommateurs et fournir des informations exploitables aux entreprises pour mieux comprendre les préoccupations de leurs clients.

Cependant, il est crucial de prendre en compte certaines limitations. Tout d'abord, l'utilisation de la colonne "Consumer Complaint Narrative" plutôt que la colonne "Issue" a permis d'améliorer la classification du modèle, cela pourrait avoir entraîné une perte d'informations plus détaillées contenues dans les descriptions narratives des plaintes. De plus, la raison pour laquelle le modèle n'a pas pu atteindre une précision très élevée est la similitude entre certaines classes. Cependant, ce problème peut être résolu en ajoutant des étapes de prétraitement et de stemming dans le processus.

Il est recommandé d'explorer d'avantage les techniques de prétraitement des données textuelles, telles que la suppression des stop words et l'application du stemming, pour améliorer la performance et la généralisation du modèle. Ces étapes aideront à mieux différencier les classes similaires, ce qui devrait améliorer la capacité du modèle à classer correctement les données. De plus, il serait intéressant d'explorer d'autres algorithmes de classification et de comparer leurs performances avec le SVM.

Malgré ces limitations, cette étude nous a permis de développer un modèle de classification des plaintes des consommateurs qui peut être utilisé pour fournir des informations précieuses aux entreprises. Il s'agit d'une étape importante vers l'amélioration de la compréhension des problèmes rencontrés par les consommateurs et la mise en place de mesures correctives pour offrir une meilleure expérience client.

VII. Bibliographie

https://datascientest.com/text-mining-definition

https://data.world/data-society/consumer-complaint-data

https://thecleverprogrammer.com/2022/11/28/consumer-complaint-classification-with-machine-learning

https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm