15/09/2023

TP Projet :

Rapport Intermédiaire

Catil Gillian, Eyili Fatih, Le Devehat Mikael

Les données fournis pour ce projet proviennent de Kraggle et permet de caractériser 75 000 bières brassés avec 176 styles différents. Ces données ont divers attributs que nous l'on devoir utiliser afin de créer des prédictions sur l'amertume de la bière brassé ainsi que les totaux d'alcools de celle-ci. Pour cela, nous avons :

- BeerID : l'id correspond à la bière
- Name : le nom de la bière
- URL : le lien de la recette de bière
- Style : Type de la bière
- StyleID : Id du style de la bière
- Size(L) : Quantité brassée
- OG : Densité spécifique du moût avant fermentation
- FG : Densité spécifique du moût après fermentation
- ABV : alcool par volume
- IBU : L'unité d'amertume Internationale
- Color : référencement de la couleur de la bière
- BoilSize : le liquide au début de l'ébullition
- BoilTime : Le moment ou le moût bouilli
- BoilGravity : Densité du moût avant ébullition
- Efficiency : Efficacité de l'extraction des sucres du grain pendant le moût
- MashThickness : quantité d'eau par livre de céréales
- SugarScale : Détermine la concentration de solides dissous dans le moût
- BrewMethod : La technique de brassage
- PitchRate : Levure ajoutée fermenteur par unité de gravité
- PrimaryTemp : Température du stade de fermentation
- PrimingMethod :
- PrimingAmount : Quantité de sucre de préparation utilisée
- UserId : l'id correspond à l'utilisateur

Ces données possèdent différente valeur dont des valeurs Null (ou NaN) ou des valeurs qui ne sont pas juste. Pour cela, nous devons nettoyer les données en fonction du nombre de valeur Null, du

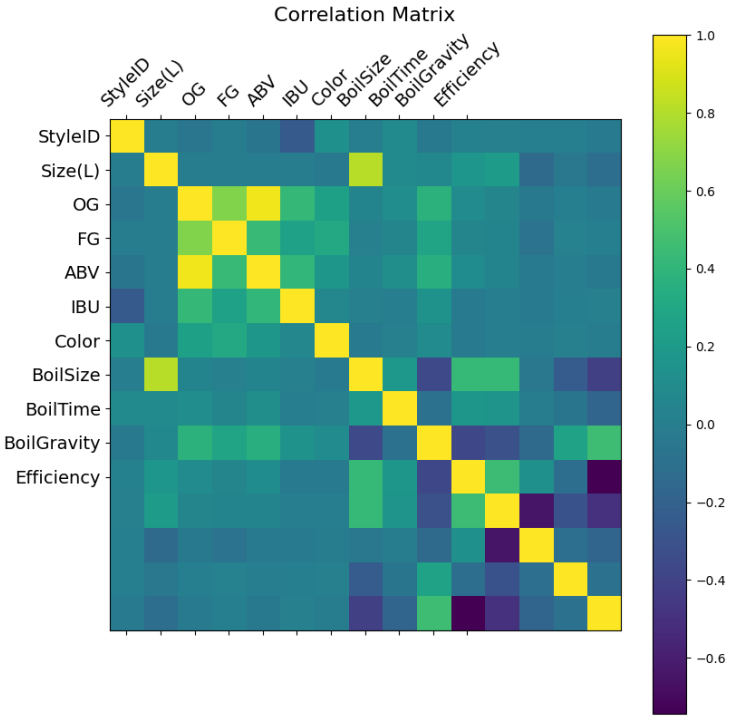
nombre de valeur aberrante, en fonction des colonnes utilisables. Ainsi, les colonnes qui ne serons pas utilisées sont :

- BeerID :
- UserID
- URL
- NAME
- Style
- PrimingMethod
- PrimingAmount
- PitchRate
- MashThickness
- PrymaryTemp
- SugarScale
- BrewMethod

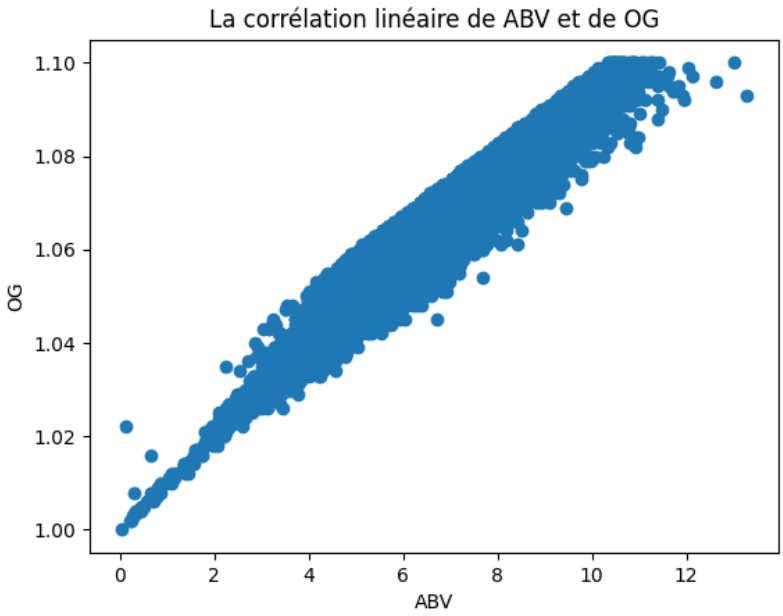
Les raisons sur le non-utilisation de ces colonnes sont dû au fait que :

- Nous n'avons pas besoin de l'id
- L'utilisation de string ne permet pas d'effectuée de calcule.
- Il y a trop de valeur Null (ou NaN)
- Il y a trop de valeur aberrante
- Des problèmes de corrélations
- L'utilisation de la colonne ne nous permet de créer des prédictions

Maintenant, que les données sont nettoyées de tous types de problèmes, nous pouvons utiliser les données restantes afin de créer notre prédiction. Nous devons prédire le taux d'amertume (IBU) et d'alcool (AVB)d'une potentiel bière. Pour cela nous avons faire un tableau de corrélation.



Comme nous pouvons le voir via le tableau de corrélation, il y a une corrélation linéaire entre OG et ABV, ce qui nous permettra d'avoir une prédiction selon une courbe linéaire.



Pour IBU, La feature adéquat n'ai pas encore bien définit à travers le tableau de corrélation. Ainsi, le choix d'utilisation de bin pourrait aider à prédire l'IBU via des catégories. Pour commencer, afin d'effectuer des tests nous avons défini les bins de tel sort que nous avons : [0,20), [20,40), [40,60), [60,80), [80,100), [100,150). Grâce à un one hot, nous avons les 6 classes que ce dessine via une représentation booléenne.

	[0, 20)	[20, 40)	[40, 60)	[60, 80)	[80, 100)	[100, 150)
0	True	False	False	False	False	False
1	False	False	False	True	False	False
5	False	False	True	False	False	False
7	True	False	False	False	False	False
8	False	True	False	False	False	False
...	...	...	...	...	...	...
73856	False	True	False	False	False	False
73857	False	False	True	False	False	False
73858	False	True	False	False	False	False
73859	False	True	False	False	False	False
73860	False	False	True	False	False	False

Les nouvelles données d'IBU, sera ensuite séparer en 2 parties : une partie entrainement et une partie teste. La partie entrainement servira à entrainer un RandomForest Classifier que l'on va ensuite tester via les données test. Il nous suffit de comparer la prédiction de celui avec les classes réelles de nos données ainsi nous arrivons à une précision de :

```
Accuracy: 0.21020148847340714
```

Malheureusement, les tests ont une précision de 21,02% de réussite. Pour continuer de tester les bins, nous sommes repartis avec les bins égales à [0,40) et [40,150), ainsi pour voir l'efficacité sur 2 classes au lieu 6 classes, et la précision monte jusqu'à 70%

Annexes :

```python
pd.set_option('display.max_columns', None) # a check c'est quoi

df = pd.read_csv('./recipeData.csv', encoding="latin1")

orig_leng = len(df)

df = df.drop(
 columns=["BeerID", "UserId", "URL", "Name", "Style", "PrimingMethod", "PrimingAmount", "PitchRate", "MashThickness",
          "PrimaryTemp", "SugarScale"])  # PrimaryTemp
df.dropna(inplace=True)
ser = df.isna().mean() * 100

# aze = df["Size(L)"].quantile(0.95)
df = df[df["Size(L)"] <= df["Size(L)"].quantile(0.95)]
df = df[df["OG"] <= df["OG"].quantile(0.95)]
df = df[df["FG"] <= df["FG"].quantile(0.95)]
df = df[(df["IBU"] <= 150) & (df["IBU"] > 0)]  #IBU max == 150 selon wikipedia & supérieur a zero
df = df[df["BoilSize"] <= df["BoilSize"].quantile(0.95)]

hist = df.hist(bins=50, log=True)


new_len = len(df)
# print(new_len / orig_leng)

one_hot = pd.get_dummies(df["BrewMethod"])
print(one_hot)
df = df.drop(columns=["BrewMethod"])
df = df.join(one_hot)

print(df)
```

Code #1 : Code qui permet de nettoyer les données