L'objectif de ce projet est d'étudier, en utilisant des méthodes d'apprentissage automatique, l'impact de différents critères (notes des critiques, appelation) sur le prix d'un vin.

Pour ce faire, on s'appuiera sur le site Millesima (`https://www.millesima.fr/`), qui a l'avantage de ne pas posséder de protection contre les bots. Par respect pour l'hébergeur du site, on veillera à limiter au maximum le nombre de requêtes. En particulier, on s'assurera d'avoir un code fonctionnel avant de scraper l'intégralité du site, pour éviter les répétitions.

Instructions

Ce projet est à réaliser impérativement en binôme : les projets individuels seront pénalisés. Seules les bibliothèques spécifiquement mentionnées dans le sujet sont autorisées.
Le jury donnera zéro s'il s'aperçoit lors de la soutenance que le code n'est pas maîtrisé. Ce sera en particulier le cas si le code a été produit via une IA générative.

Premier jalon : récupération des données en `python`

Tout projet de machine learning s'appuie sur un ensemble massif de données. L'objectif de cette première partie est la récupération de ces données, et leur stockage dans le format CSV.
Cette partie est à coder en `python`.

Étudier la page d'un vin sur le site `https://www.millesima.fr/`.

On pourra prendre comme exemple la page du millésime 2016 de Château Gloria : `https://www.millesima.fr/chateau-gloria-2016.html`, reprise dans les Figures 1, 2 et 3.

Question 1 En utilisant les bibliothèques `selenium` (dont on lira la documentation) et `Beautiful soup`, écrivez une fonction `getsoup()` qui prend en entrée l'URL de la page d'un vin, et renvoie la soupe correspondant à cette page HTML.

Ici, nous utilisons `selenium` plutôt que `requests` dans la mesure où une partie du contenu des pages de ce site est généré via un script JS. Contrairement à `requests`, `selenium` permet de lancer un navigateur et son JavaScript engine, nécessaire à l'exécution du code JavaScript.

Question 2 En haut de chaque page (cf. l'encadré rouge sur la Figure 1) est indiqué le prix du vin. Attention, on sera intéressé par le prix unitaire, et pas par le prix d'une caisse : ici, 60,33€ pour une bouteille, et pas 724,00€ pour une caisse de 12 bouteilles.

Écrivez une fonction `prix()` qui prend en entrée la soupe correspondant à la page d'un vin, et renvoie son prix unitaire. Dans l'exemple, il faudra renvoyer `60.33`.

L'*appellation* d'un vin reflète sa provenance géographique : il peut s'agir d'une région entière (par exemple, *Languedoc*), d'une ville ou d'un village (par exemple, *Pauillac*), ou même d'une seule parcelle dans les cas les plus précis (par exemple, *Chambolle-Musigny Les Amoureuses*, qui correspond à une parcelle bien délimitée du village de Chambolle-Musigny).

Question 3 Écrivez une fonction `appellation()` qui attend en entrée la soupe correspondant à la page d'un vin et renvoie, sous forme de chaîne de caractères, l'appellation du vin. Cette information est disponible plus bas dans la page dans le tableau "Détails", comme indiqué en rouge sur la Figure 2. Ici, on renverra `"Saint-Julien"`.

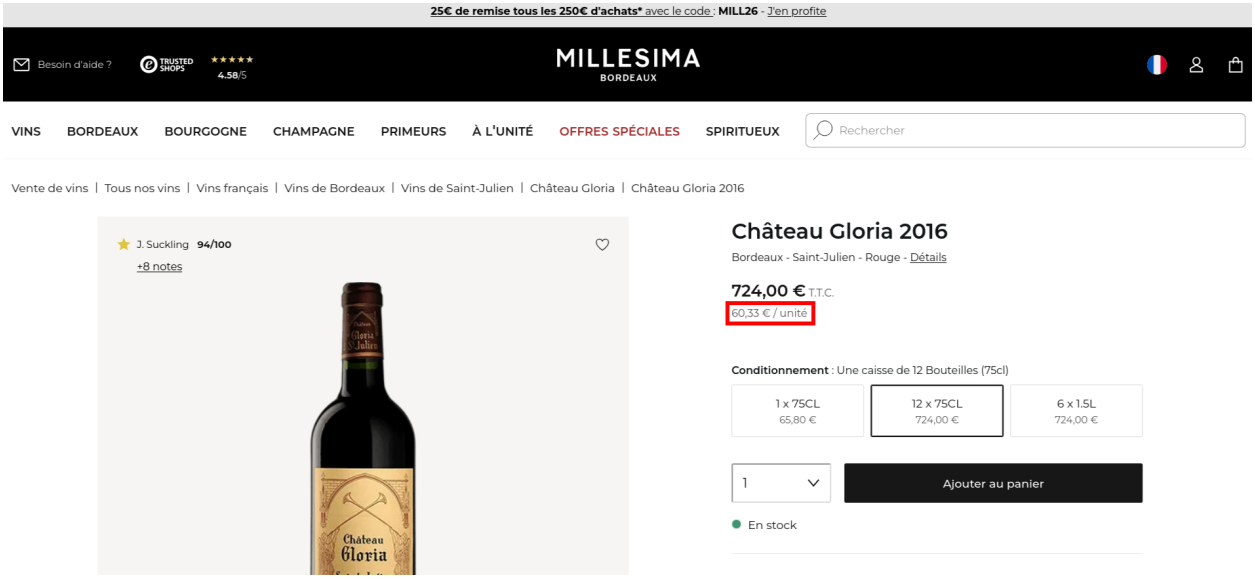

FIGURE 1 – En rouge, le prix (à l'unité) du vin.

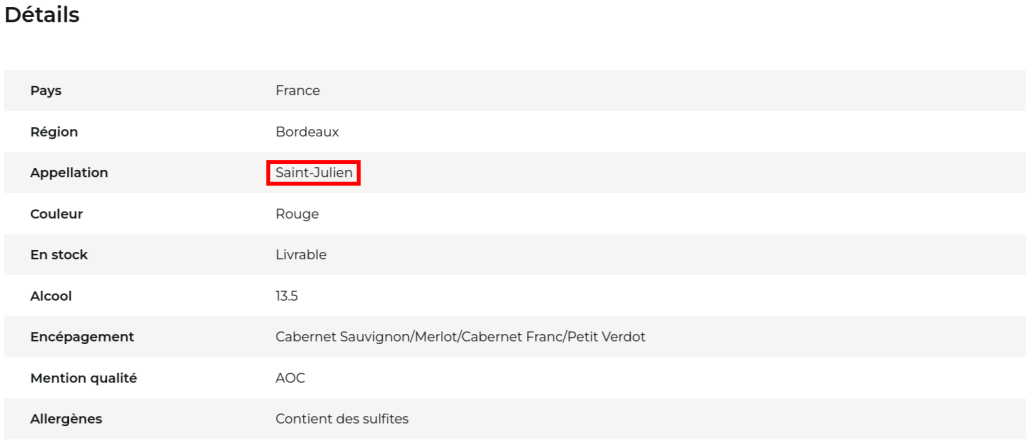

FIGURE 2 – Les détails du vin. On s'intéressera à son appellation, en rouge.

Dans le monde du vin, un certain nombre de critiques donnent des notes aux différentes cuvées. Il existe un grand nombre de ces critiques, mais on s'intéressera seulement dans ce sujet à trois des plus connus : Parker, J. Robinson et J. Suckling.

Les notes attribuées par ces critiques à la cuvée en question se retrouvent dans la section "Évaluation et notation" de la page du vin. Les trois notes qui nous intéressent apparaissent en rouge dans la Figure 3.

Question 4 Écrivez une fonction `parker()`, qui prend une soupe en argument et renvoie la note

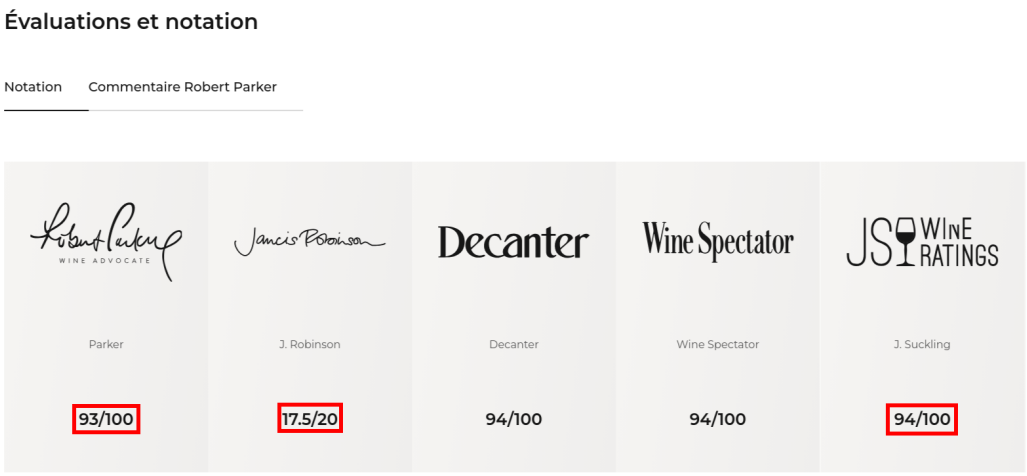

FIGURE 3 – Les notes attribuées par les critiques. En rouge, celles de Parker, J. Robinson et J. Suckling.

attribuée au vin par Parker. La réponse devra respecter les instructions suivantes :

- on omettra la base de la note (ici, `"/100"`). Dans ce cas, on renverra donc `93`.
- tous les critiques ne notent pas tous les vins : en cas d'absence de note de Parker, on renverra la constante `None`.
- la note est parfois une plage (par exemple, `"90-93/100"`). Dans ce cas, on renverra la moyenne (dans cet exemple, `91.5`).
- la note est parfois agrémentée d'un `+` (par exemple, `"96+/100"`). On ne tiendra alors pas compte du `+` (dans ce cas de figure, on renverra simplement `96`).

On testera cette fonction sur différentes pages du site pour s'assurer qu'elle couvre tous les cas de figure évoqués.

Question 5 Implémentez les fonctions `robinson()` et `suckling()` qui reprennent la question précédente, mais pour les critiques J. Robinson et J. Suckling. On fera évidemment en sorte d'éviter toute duplication de code : il faudra factoriser un maximum de code entre ces trois fonctions.

Question 6 Écrivez une fonction `informations()` qui attend la soupe de la page d'un vin en argument, et renvoie une chaîne de caractères contenant toutes les informations de l'annonce, séparées par des virgules, dans l'ordre :

`"Appelation,Parker,J.Robinson,J.Suckling,Prix"`

Sur l'annonce de Château Gloria 2016, cette fonction devra donc renvoyer :

`"Saint-Julien,93,17.5,94,60.33"`

Question 7 Nous allons limiter nos recherches aux vins de Bordeaux, détaillés sur la page `https://www.millesima.fr/bordeaux.html?page=1` et les suivantes.

En étudiant l'URL et la structure des différentes pages de résultats (il devrait y en avoir entre 50 et 100), écrivez un script qui parcourt toutes les annonces proposées par le site et appelle `informations()` sur les soupes correspondantes.

Les résultats obtenus devront être enregistrés, à raison d'une ligne par annonce, dans un fichier CSV (comma-separated values) dont la première ligne indiquera les étiquettes (sans guillemets) des différents champs :

```
Appellation,Robert,Robinson,Suckling,Prix
```

Pour la suite du projet, on travaillera à partir de ce fichier CSV local : il ne faudra surtout pas scraper le site à chaque nouvelle session de travail !

Deuxième jalon : Nettoyage des données

Avant de pouvoir utiliser notre jeu de données dans le cadre d'un apprentissage automatique, il va nous falloir les analyser et les nettoyer. Pour cela, on utilisera la bibliothèque `pandas`.

Question 8 Commencez par importer le contenu du CSV de la partie précédente dans un `DataFrame` nommé `vins`, et vérifiez la conformité des données.

Question 9 Supprimez de `vins`, s'il y en a, toutes les lignes n'ayant pas d'appellation.

La bibliothèque d'apprentissage automatique `scikit-learn`, que nous utiliserons dans la partie suivante, ne peut manipuler que des données numériques. Il va falloir procéder à un certain nombre de conversions et de complétions.

Unicode

En fonction de la manière dont vous avez traité la première partie, il est probable que la colonne `"Prix"` contienne un caractère unicode "espace insécable" (`'\u202f'`) pour les vins coûtant plus de 1000€. Cela posera problème au moment de la conversion de cette chaîne vers un entier, il faut donc retirer ce caractère.

Question 10 En vous aidant de la documentation python concernant l'unicode, écrivez une fonction qui attend une chaîne de caractères en unicode et renvoie la sous-chaîne retreinte aux caractères ASCII : elle supprimera donc tous les caractères unicode non-ASCII. Appliquez cette fonction à chaque élément de la colonne `vins["Prix"]`, et vérifiez que le caractère `'\u202f'` n'apparaît plus.

Notes manquantes

Certains champs des colonnes `"Robert"`, `"Robinson"` et `"Suckling"` de `vins` ne contiennent pas de valeur numérique, puisque chaque critique ne note pas chaque vin. Nous allons remplir ces champs d'une manière adaptée au contexte.

Question 11 Pour chaque appellation, calculez la moyenne des notes `"Robert"` des vins de cette appellation – on écartera temporairement les vins n'ayant pas de note par `"Robert"` au moment de calculer la moyenne – et stockez le résultat dans un `DataFrame`. Dans le cas où aucune note n'existerait pour une appellation, on fera en sorte d'attribuer la note `0`.

Indication : Vous aurez besoin des fonctions `groupby()` et `mean()`.

Question 12 Répétez l'opération de la Question 11 (en factorisant le code au maximum) pour `"Robinson"` et `"Suckling"`.

Question 13 Modifiez `vins` en remplaçant les valeurs manquantes de chacune des colonnes `"Robert"`, `"Robinson"` et `"Suckling"` par la note moyenne des vins de la même appellation, en faisant une jointure (via `merge()`) avec le(s) `DataFrame` obtenu(s) aux questions précédentes.

Colonne `"Appellation"`

On va maintenant se concentrer sur la colonne `"Appellation"`, qui est la seule contenant des valeurs non-numériques. Pour cette colonne, on va utiliser la méthode des variables indicatrices. Cela consiste à créer une nouvelle colonne pour chaque appellation `a` possible. Une ligne possèdera un `1` dans la colonne `a` si et seulement si son `Appelation` vaut `a`, et un `0` dans toutes les autres colonnes. Un exemple minimal est donné en Figure 4.

	Appellation	···
0	Pauillac	···
1	Margaux	···
2	Margaux	···
3	Pomerol	···

(a) Avant

	App_Pauillac	**App_Pomerol**	**App_Margaux**	···
0	1	0	0	···
1	0	0	1	···
2	0	0	1	···
3	0	1	0	···

(b) Après

FIGURE 4 – Illustration de l'introduction de variables indicatrices

Question 14 En utilisant la fonction `get_dummies()` de `pandas`, créez des variables indicatrices pour remplacer la colonne `"Appellation"`.

À ce stade, on devrait obtenir un `DataFrame` ayant une trentaine de colonnes et entre 2000 et 3000 lignes n'ayant que des valeurs numériques. Nos données sont prêtes pour la phase d'apprentissage.

Troisième jalon : Apprentissage

Nous sommes prêts à passer à l'apprentissage à proprement parler. Pour ce faire, nous allons utiliser la bibliothèque `scikit-learn`.

Préparation de données d'apprentissage et de test

Le but est d'estimer le prix d'un produit à partir des attributs. La colonne `"Prix"` correspond ainsi aux étiquettes ("targets"), tandis que les autres colonnes sont les attributs ("features").

Question 15 Préparez la matrice de données et le vecteur des étiquettes pour votre jeu de données. Séparez les données en un jeu d'entraînement et un jeu de test, sur un base 75%/25%, en utilisant le paramètre `random_state = 49`.

Nous utiliserons les notations suivantes :

- `X_train` (resp. `y_train`) fait référence aux données d'apprentissage (resp. à leur étiquette),
- `X_test` (resp. `y_test`) fait référence aux données de test (resp. à leur étiquette).

Premier modèle : Régression Linéaire

Dans un premier temps, nous allons appliquer une *régression linéaire* `LinearRegression` sur nos données.

Question 16 Évaluez ce modèle sur le jeu de test par r^2`_score`. Rappelons qu'il s'agit de la mesure par défaut pour la fonction `score()` des méthodes de régression.

Question 17 Nous allons Faire une comparaison entre les estimations (`estim_LR`) faites par le modèle pour les données de test et les prix `y_test`. Dessinez une `figure` où l'axe x représente `estim_LR` et l'axe y est `y_test`. Pensez à utiliser la fonction `scatter` pour représenter les points. Nous appellerons dans la suite *figure de visualisation* cette représentation. Rappelons que si votre modèle fonctionne bien pour ce jeu de données, vous devriez avoir les points autour de la diagonale où `estim_LR` est égal à `y_test`. Que constatez-vous ?

Question 18 Nous allons maintenant évaluer l'impact du pré-traitement sur nos données.

En utilisant la commande `make_pipeline`, normalisez d'abord les données avant de lancer l'apprentissage, et évaluez le score sur le jeu de test. Répétez l'expérience, cette fois-ci en standardisant les données avant l'apprentissage.

Dessinez les figures de visualisation pour les deux cas. Observez-vous les mêmes choses qu'à la question précédente ?

Question 19 Trouvez les valeurs minimales et maximales pour les prix dans votre dataset. Que peut-on dire de la dispersion des prix dans votre dataset ?

Dans le cas où les étiquettes sont trop dispersées, une solution pourrait être de modifier l'échelle des étiquettes. Pour ce faire, on pourrait transformer les prix de sorte que les prix utilisés dans l'apprentissage soient $ln(y[i])$ à l'aide des fonctions `ln et exp` de la bibliothèque numphy.

Quelles sont les nouvelles valeurs pour le prix minimum et maximum de votre dataset ?

Question 20 Après avoir éventuellement transformé les prix, remplissez le tableau suivant avec les résultats obtenus grâce à la régression linéaire, avec ou sans pré-traitement. Constatez-vous une amélioration des prédictions due au pré-traitement pour ce jeu de données et cette méthode ? Dessinez la figure de visualisation pour la méthode qui donne le meilleur score. Que constatez-vous ?

Méthode	r^2
LR	
Normalisation + LR	
Standardisation + LR	

Deuxième modèle : Arbre de Décision

Dans la suite, vous utiliserez les prix transformés ou non en fonction de votre dataset.

Le deuxième modèle d'apprentissage que nous allons considérer sont les *arbres de décision* `DecisionTreeRegressor`.

Question 21 En utilisant la méthode de `validation croisée (cv)` avec $cv = 5$, déterminez la valeur de la *meilleure* profondeur maximale (h) parmi `max_depth`$= \{3, 4, 5\}$ sur les données `X_train` (sans pré-traitement).

Remplissez à nouveau un tableau de trois lignes (une pour la méthode "AD", une pour "Normalisation + AD" et une pour "Standardisation + AD") avec les résultats que vous avez obtenus grâce aux arbres de décision avec la profondeur maximale, h, avec ou sans pré-traitement pour les données de test.

Constatez-vous une amélioration significative suite au pré-traitement ? Les scores obtenus sont-ils satisfaisants ?

Troisième modèle : N plus proches voisins

En guise de troisième méthode d'apprentissage, nous allons nous pencher sur la méthode des *plus proches voisins* `KNeighborsRegressor`.

Question 22 Évaluez ce modèle sur le jeu de test, en employant dans une premier temps le paramètre `n_neighbors = 4`. Testez ensuite l'impact du pré-traitement, comme pour les modèles précédents. Constatez-vous une amélioration significative suite au pré-traitement ? Les scores obtenus sont-ils satisfaisants ?

Refaites l'expérience avec un paramètre `n_neigbors=5`. L'augmentation du nombre de voisins considérés permet-elle de meilleurs résultats ? Si cette amélioration n'est pas significative, nous conserverons `n_neighbors = 4` pour la suite, par souci de simplicité.

Question 23 Remplissez à nouveau un tableau de trois lignes (une pour la méthode "KNN", une pour "Normalisation + KNN" et une pour "Standardisation + KNN") avec les résultats que vous avez obtenus grâce à la méthode des plus proches voisins, avec ou sans pré-traitement.

Discussions sur le jeu de données

Nous avons considéré à ce stade trois méthodes d'apprentissage, à savoir la régression linéaire (LR), les arbres de décision (AD) et la méthode des plus proches voisins (KNN). En gardant uniquement le meilleur score obtenu pour chacune de ces trois méthodes (avec pré-traitement ou non), nous obtenons un tableau de 3 lignes :

Méthode	r^2
LR	
AD	
KNN	

Question 24 Replissez le tableau, et comparez les scores obtenus pour ce jeu de données grâce à ces différentes méthodes. Que constatez-vous ? Dans la suite, nous noterons $\mathcal{M}$ la méthode qui donne le meilleur score pour les données de test.

Réduction le nombre d'attributs

Nous souhaitons maintenant réduire le nombre d'attributs de notre jeu de données en utilisant la méthode `PCA`.

Question 25 Réduisez le nombre de composantes à 5. La modélisation avec 5 composantes principales est-elle satisfaisante pour ce jeu de données ? Justifiez.

Question 26 Lancez la méthode $\mathcal{M}$ sur les données obtenues après réduction des composantes, et comparez les nouveaux scores aux scores précédents. Que constatez-vous ?

Matrice de corrélation

Nous allons tenter de comprendre dans quelle mesure chaque attribut permet de prédire l'étiquette.

Question 27 Établissez la matrice de corrélation `DataFrame.corr()` du jeu de données. On pourra trouver un exemple de création et d'affichage de matrice de corrélation à cette adresse.

Question 28 Quels sont les attributs dont la connaissance nous renseigne le plus sur le prix ?

Pour déterminer cela, on regardera la colonne (ou la ligne) "Prix" de la matrice, qui indique la corrélation entre le prix et chacun des autres attributs, considérés individuellement. Plus le résultat est grand, plus la corrélation entre le prix et cet attribut est grande. À l'inverse, plus ce résultat est grand dans les négatifs, plus la corrélation inverse est grande. Enfin, un résultat proche de zéro signifie qu'il y a une faible corrélation entre le prix et cet attribut.

Question 29 Modifiez le jeu de données en ne conservant que les cinq attributs qui sont le plus corrélés (positivement ou négativement) au prix. Appliquez la méthode $\mathcal{M}$ à ce nouveau jeu de données, et comparez le score de prédiction aux résultats obtenus précédemment. Que constatez-vous ?

Quatrième modèle : une méthode ensembliste

Pour terminer, vous allez explorer une méthode qui n'a pas été étudiée dans le cours.

Question 30 On se penche sur l'apprentissage via une méthode ensembliste, comme le *Gradient Boosting* ou les *Random Forests*.

Choisissez une de ces méthodes, et refaites l'apprentissage en l'utilisant. Remplissez à nouveau un tableau de trois lignes (une pour la méthode "ens", une pour "Normalisation + ens" et une pour "Standardisation + ens") avec les résultats que vous avez obtenus grâce à la méthode ensembliste que vous avez choisie, avec ou sans pré-traitement. Que constatez-vous ?