

CREDA PA11 - PJE09

Analyse et traitement des données - Prédiction du rendement des actions - Qube RT



Nathan B. - M. Ronan Cardin - 02/2021



Sommaire

I - Contexte et définitions

II - Data cleaning

III - Data exploration

IV - Features engineering

V - Classification, test, et optimisation

VI - Prédiction finale

VII - Conclusions

I - Contexte, définitions, et intégration des données

Contexte : data science et apprentissage automatique pour stratégies d'investissement quantitatives (Challenge ENS/Collège de France 2020)

Objectif : prédire le rendement des actions sur le marché américain par analyse de données historiques (20 jours précédents)

Fichiers fournis :

- données d'entraînement (x_train) => **variables catégorielles**/**variables quantitatives continues**
- fichier labels (y_train)
- données test (x_test)

	ID	DATE	STOCK	INDUSTRY	INDUSTRY_GROUP	SECTOR	SUB_INDUSTRY	RET_1	\
0	0	0	2	18	5	3	44	-0.015748	
1	1	0	3	43	15	6	104	0.003984	
2	2	0	4	57	20	8	142	0.000440	
3	3	0	8	1	1	1	2	0.031298	
4	4	0	14	36	12	5	92	0.027273	
	VOLUME_1	RET_2	...	RET_16	VOLUME_16	RET_17	VOLUME_17		\
0	0.147931	-0.015504	...	0.059459	0.630899	0.003254	-0.379412		
1	NaN	-0.090580	...	0.015413	NaN	0.003774	NaN		

II - Intégration des données et data cleaning

Librairie pandas : permet la manipulation des données sous la forme de dataframe 2D

Librairie seaborn : modélisation graphique/statistique avancée et manipulation dataframe

Création du dataframe :

Importation des données

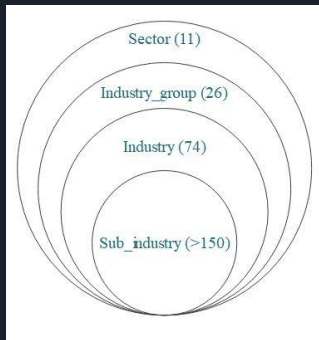
```
X_train = pd.read_csv('./Données projet Qube RT/x_train_Lafd4AH.csv', sep = ',')
X_test = pd.read_csv('./Données projet Qube RT/x_test_c7ETL4q.csv', sep = ',')
y_train = pd.read_csv('./Données projet Qube RT/y_train_JQU4vbI.csv', sep = ',')
y_test = pd.read_csv('./Données projet Qube RT/test_rand.csv', sep = ',')
```

Data cleaning :

- Suppression des NaN (Not a Number) : 418 595 lignes ➡ 314 160 lignes (*pas de remplacement ou de prédiction pour conserver moy/med*)
- Réindexation

Présentation rapide des données (418 495 actions - 20 derniers jours, début à $t_0 = t-1$ jour)

- 7 variables catégorielles : ID, date, stock, industry, industry_group, sector, sub-industry



- 40 variables quantitatives continues : couples (rendement résiduel, volume relatif)

Rendement résiduel :

$$R_j^t = \frac{P_j^t}{P_j^{t-1}} - 1$$

Volume relatif :

$$\bar{V}_j^t = \frac{V_j^t}{\text{median}(\{V_j^{t-1}, \dots, V_j^{t-20}\})}$$
$$\mathcal{V}_j^t = \bar{V}_j^t - \frac{1}{n} \sum_{i=1}^n \bar{V}_i^t$$

Typologie du problème :

✓ Apprentissage supervisé (*labels fournis*)

✗ Apprentissage non-supervisé

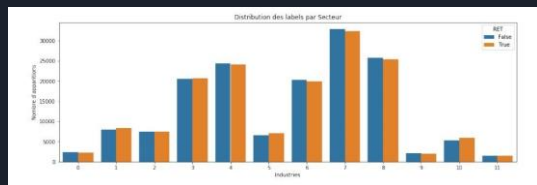
✓ Classification ($y=1$ si $y > y_{\text{moy}}(j)$, 0 sinon, pour $y=R_t(j)$)

✗ Régression

III - Data exploration

Analyse de l'équilibre des données et recherche d'un fitting approprié

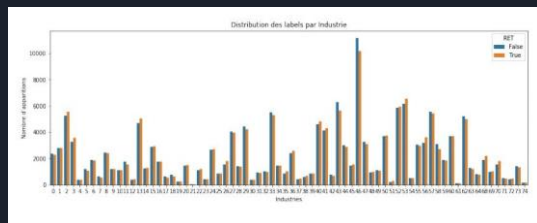
Le jeu de données est parfaitement **équilibré**. Cependant, en termes de répartition :



Feature Sector : peu d'individus mais
risque d'underfitting ✓



Feature Industry_group : équilibré,
nombre convenable ✓

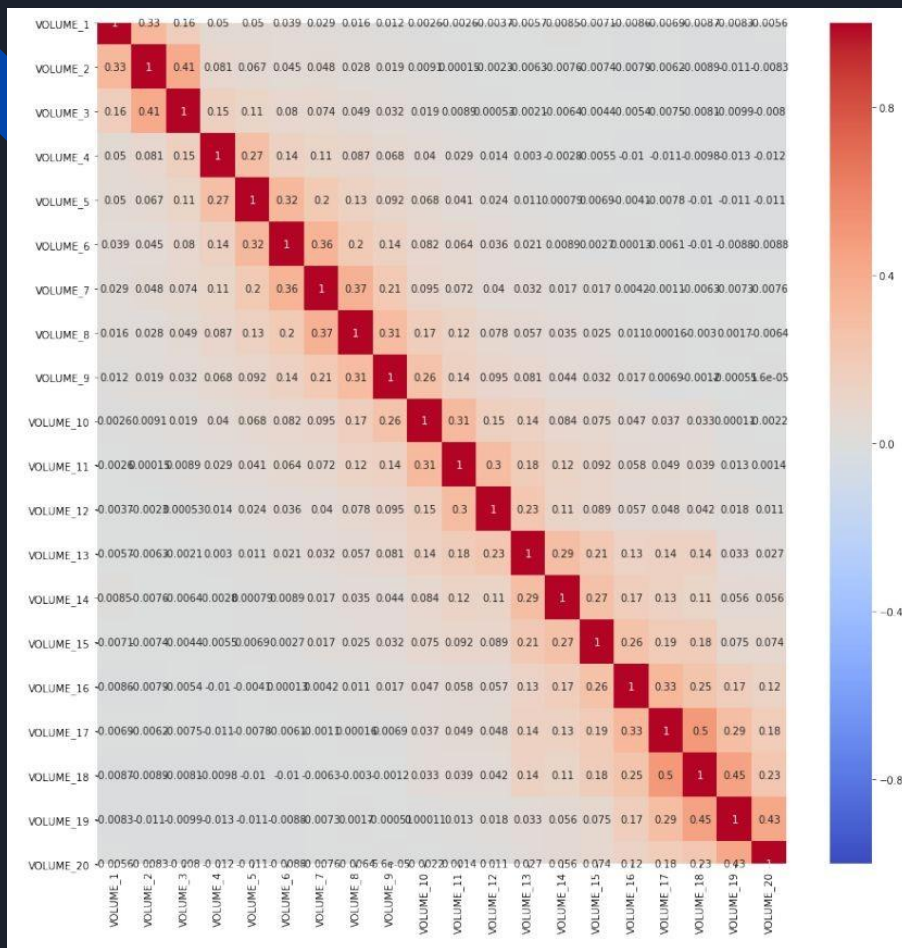


Feature Industry : idem ✓



Feature Sub Industry : risque d'overfitting ✗

Analyse des corrélations



Modèle de heatmap pour analyse de corrélation entre volumes

Méthodologie machine :

- Agrégation des colonnes features pertinentes (RET, Volume, ou les 2 ensemble)
- Calcul de la matrice des corrélations du tableau ainsi obtenu (au sens de Bravais-Pearson)
- Représentation sous forme de heatmap via Seaborn

Tableau récapitulatif :

Features	Corrélation	Commentaire
Volumes	30% entre jours i et i-1	5% entre i et i-20, inutile d'aller au delà
Return	Aléatoire entre le jour i et les précédents	Trop chaotique pour être significative
Volumes & Returns	De 10 à 22% entre Volume_i et RET_i	Non négligeable, à conserver pour la prédiction

IV - Features engineering

Cherchant à enrichir notre modèle de prédiction, on procède à la création des features renseignées dans le tableau suivant, puis à l'analyse des corrélations entre grandeurs de la même façon qu'auparavant :

Couche

Features nouvellement créées

1

Moy RET_1 par secteur et industrie

Moy RET_1 par date et industrie

Moy Vol_1 par secteur et industrie

Moy Vol_1 par date et industrie



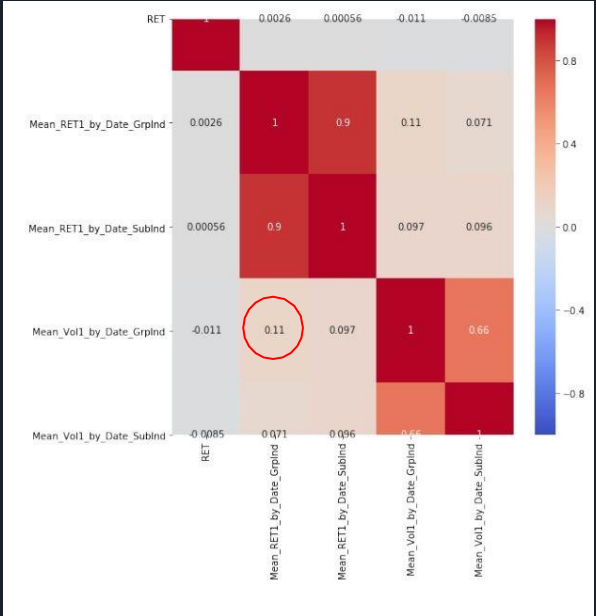
Commentaire

Corrélation de 0.12 entre moyennes RET_1 et Vol_1 cond. à date et secteur, pertinente pour prédiction future *

*Date anonymisée depuis le début : au mieux, ajout d'info au pire, baisse précision négligeable

IV - Features engineering

=> *Une erreur s'est glissée dans le rapport : c'est le nombre entouré ci-dessous dans la matrice qui doit être considéré, et non "Mean_RET1_by_Date_SubInd". De ce fait, une mauvaise " nouvelle feature " a été acceptée en tant que pertinente dans les modèles de prédiction ; cependant, l'écart entre le coefficient de corrélation entouré (0.11 - celui qui doit normalement être retenu) et celui liant "Mean_RET1_by_Date_SubInd" (la mauvaise feature) et "Mean_Vol1_Date_GrpInd" (0.097) est assez peu élevé (0.013). Par conséquent, même si c'est une erreur, le fait d'avoir retenu cette " mauvaise feature " n'est certainement que peu significatif dans les résultats d'accuracy auxquels on aboutit ultérieurement.

Couche	Features nouvellement créées		
2	Moy RET_1 par date et groupe d'industries		<p>Commentaire*</p> <p>Corrélation de +1% (0.11) entre les moy RET_1 et Vol_1 cond. à date et groupe indus : conservons-les pour la suite.</p>
	Moy RET_1 par date et sous-industrie		
	Moy Vol_1 par date et groupe d'industries		
	Moy Vol_1 par date et sous-industrie		

Préparation des dataset - Bilan provisoire

Retenu :

- volumes et retours des 5 derniers jours (seuil de corrélation $\geq 5\%$)
- 3 nouvelles features : Mean_RET1_Date_Sector, Mean_VOL1_Date_Sector, Mean_RET1_Date_SubInd

Rejeté :

- données antérieures à 5 jours
- variables catégorielles (date, stock, industry...), déjà prises en compte dans nouvelles features

Allure du dataset :

```
df_train = data_train[['RET_1', 'VOLUME_1', 'RET_2', 'VOLUME_2', 'RET_3',  
                        'VOLUME_3', 'RET_4', 'VOLUME_4', 'RET_5', 'VOLUME_5',  
                        'RET',  
                        'Mean_RET1_by_Date_Sector', 'Mean_Vol1_by_Date_Sector',  
                        'Mean_RET1_by_Date_SubInd']]
```

V - Classification, test, et optimisation

A) Modèle général - Entraîné sur toutes les données

Modèles testés (importation via scikit-learn) :

- Régression logistique
- Arbre de décision
- Random forest

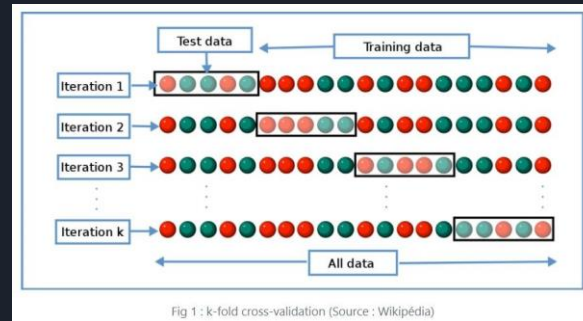
Normalisation/scaling des données :

MinMaxScaler()

Cette technique transforme les caractéristiques (x_i) en adaptant chacune sur une plage donnée (par défaut [-1 .. 1]). Il est possible de changer cette plage via les paramètres `feature_range=(min, max)`. Pour faire simple voici la formule de transformation de chaque caractéristique :

$$\frac{x_i - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$$

Cross-validation : dans notre cas, $k=7$.



Performance des classifieurs avec CV ($k=7$) :

	Performance en %
Logistic Regression	50.531258
Decision Tree Classifier	49.815059
Random Forest Classifier	50.416347

V - Classification, test, et optimisation

Optimisation des hyperparamètres :

Pour le modèle Random Forest avec utilisation de GridSearch

Paramétrage :

```
[43]: {'max_depth': 12, 'n_estimators': 200}
```

Résultat :

```
param_grid={'max_depth' : [8,12], 'n_estimators' : [100,200]}
```

Après optimisation, on privilégie le modèle RF sur la RL bien que la RL soit a priori meilleure :

Accuracy RL avant GS (t=0) : 50,53%	Accuracy RL après GS (t+45min) : 50,74%
Accuracy RF avant GS (t=0) : 50,41%	Accuracy RF après GS (t+22min) : 51,53%

B) Modèle par industrie

- Chaque industrie isolément se voit associer une accuracy relativement aux classifieurs étudiés.
- Le classifieur à la meilleure moyenne est ainsi défini comme celui convenant le mieux pour cette industrie précise.

5.2 V.II) Modèle par Industrie

Choix du modèle par Industrie

```
[5]: def get_data_per_industrie(ind):
    data = data_train[data_train.INDUSTRY == ind]
    X = data.loc[:,['RET_1', 'VOLUME_1', 'RET_2', 'VOLUME_2', 'RET_3',
                    'VOLUME_3', 'RET_4', 'VOLUME_4', 'RET_5', 'VOLUME_5',
                    'Mean_RET1_by_Date_Sector', 'Mean_Vol1_by_Date_Sector',
                    'Mean_RET1_by_Date_SubInd']]

    X = (X-X.max())/(X.max() - X.min())
    y = data.RET
    return X, y

[5]: models = []
models.append(('Logistic Regression',LogisticRegression()))
models.append(('Decision Tree Classifier',DecisionTreeClassifier()))
models.append(('Random Forest Classifier',RandomForestClassifier()))

[5]: accuracy_results = []
ind_list = []
mod = []
index = [m[0] for m in models]

for ind in data_train.INDUSTRY.unique():
    for nom_model,model in models:
        print(ind, nom_model)
        X, y = get_data_per_industrie(ind)
        cv_results = cross_val_score(model, X, y, cv=7, scoring='accuracy')
        accuracy_results.append(cv_results.mean()*100)
        mod.append(nom_model)
        ind_list.append(ind)

18 Logistic Regression
18 Decision Tree Classifier
18 Random Forest Classifier
57 Logistic Regression
57 Decision Tree Classifier
57 Random Forest Classifier
```

B) Modèle mixte

- En-deçà d'un certain seuil d'accuracy : recours au modèle général
- Au-delà : conservation du modèle ayant le mieux scoré pour l'industrie en question

```
[69]: Best_model_Mixte = Best_model.copy()
```

Nous avons obtenu 50.2% d'accuracy avec le modèle général. Nous utilisons ce résultat comme seuil. Toutes les industries ayant une meilleure accuracy seule garde leur model, pour les autres, nous prenons le modèle général.

```
[70]: Best_model_Mixte.loc[Best_model_Mixte['Performance en %'] < 50.2, 'Model'] =  
      ↳ 'Model General'
```

```
[71]: Best_model_Mixte.loc[Best_model_Mixte.Model == 'Model General', 'Model_code'] =  
      ↳ 'model_général'
```

```
[72]: Best_model_Mixte.head()
```

```
[72]:
```

	Industrie	Performance en %	Model	\
0	0	51.154297	Logistic Regression	
1	1	46.143883	Model General	
2	2	51.294754	Logistic Regression	
3	3	52.608616	Logistic Regression	
4	4	52.854446	Logistic Regression	

VI - Prédiction finale

Démarche adoptée :

- 1) Import des données
- 2) Mise en forme du dataset de test `x_test` (retrait des NaN, ajout des nouvelles features, scaling...)
- 3) Prédiction du `x_test` via modèle général après entraînement
 - Obtention d'une accuracy
- 4) Renouvellement du processus avec le modèle par industrie après entraînement sur le `x_train`
- 5) Prédiction du `x_test` via modèle par industrie
 - Obtention d'une accuracy
- 6) Prédiction via modèle mixte
 - Obtention d'une accuracy
- 7) Comparaison des accuracy

VII - Résultats et conclusions

Dans un premier temps :

- filtrage des returns et des volumes sur 5 derniers jours

Création de 3 nouvelles features :

- 1) 'Mean_RET1_by_Date_Sector' : moyenne du Retour à j-1 conditionnellement à la Date et au Secteur
- 2) 'Mean_Vol1_by_Date_Sector' : moyenne du Volume à j-1 conditionnellement à la Date et au Secteur
- 3) 'Mean_RET1_by_Date_SubInd' : moyenne du Retour à j-1 conditionnellement à la Date et à la Sous Industrie

Approches d'apprentissage envisagées :

- a) Modèle général entraîné sur toutes données optimisé légèrement. Accuracy : 50,79%
- b) Modèle par industrie non optimisé. Accuracy : 49,81%
- c) Modèle mixte sur industries avec modèle propre insatisfaisant. Accuracy : 49.85%

Bilan :

- Modèle a) assez satisfaisant mais pas exceptionnel (score public : 51,31%)
- Nécessité de recherches ultérieures pour optimisation avancée (meilleure puissance de calcul, etc...)
- Très bon approfondissement du domaine de la data science et de l'apprentissage automatique néanmoins