Estimation Du Temps De Transcodage D'une Vidéo Youtube

Python For Data Analysis

FOURNIER Gaétan - GRATZMULLER Emma

Contexte

Notre dataset propose diverses informations concernant les caractéristiques de vidéos youtube et leur temps de transcodage. Dans ce projet nous chercherons à analyser ce jeu de données afin d'établir un modèle de machine learning permettant d'estimer le temps de transcodage pour une vidéo youtube.

Informations sur le dataset

Le dataset étudié comporte 2 fichiers – il s'agit de caractéristiques de vidéos youtubes aléatoires et de leur temps de transcodage.

- Le premier dataset présente les informations principales concernant les vidéos. On y retrouve les éléments suivants:

id	duration	bitrate	bitrate (video)	height	width	framerate	famerate (est)	codec	category	url
Id de la vidéo	Durée de la vidéo	Bitrate de la vidéo	Bitrate de la vidéo	Hauteur de la vidéo (pixels)	Largeur de la vidéo (pixels)	Framerate de la vidéo	Estimation du framerate	Standard d'encodage choisi	Catégorie de la vidéo	Url de la vidéo

- Le second dataset propose plus de détails quant aux différents éléments - il contient les informations les plus pertinentes à notre étude.

id	duration	codec	width	height	bitrate	framerate	i	p	b	frames
Id de la vidéo	Durée de la vidéo	standard d'encodage choisi	Largeur de la vidéo (pixels)	Hauteur de la vidéo (pixels)	Bitrate de la vidéo	Framerate de la vidéo	nombre d'images i	nombre d'images p	nombre d'images b	Nombre total d'images

i_size	p_size	b_size	size	o_codec	o_bitrate	o_framerate	o_width	o_height	umem	utime
taille totale en bits de i	taille totale en bits de p	taille totale en bits de b	taille totale de la video	Output bitrate	Output bitrate	Output framerate	Largeur output (pixels)	hauteur output (pixels)	codec allocated memory	temps de transcodage

La première observation du datasets et les informations à disposition le concernant nous ont donné une première direction pour notre étude - c'est le second dataset qui contiendra les informations les plus pertinentes pour notre estimation. Le premier restera cependant intéressant à analyser afin de se familiariser avec les différents éléments.

Exploration et traitement

	duration	codec	width	height	bitrate	framerate	i	p	b	frames	...	p_size	b_size	size	o_codec	o_bitrate	o_
0	130.35667	mpeg4	176	144	54590	12.000000	27	1537	0	1564	...	825054	0	889537	mpeg4	56000	
1	130.35667	mpeg4	176	144	54590	12.000000	27	1537	0	1564	...	825054	0	889537	mpeg4	56000	
2	130.35667	mpeg4	176	144	54590	12.000000	27	1537	0	1564	...	825054	0	889537	mpeg4	56000	
3	130.35667	mpeg4	176	144	54590	12.000000	27	1537	0	1564	...	825054	0	889537	mpeg4	56000	
4	130.35667	mpeg4	176	144	54590	12.000000	27	1537	0	1564	...	825054	0	889537	mpeg4	56000	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
779	972.27100	h264	480	360	278822	29.000000	560	28580	0	29140	...	26561730	0	33886358	flv	242000	
780	129.88100	vp8	640	480	639331	30.162790	36	3855	0	3891	...	9503846	0	10379630	mpeg4	539000	
781	249.68000	vp8	320	240	359345	25.068274	129	6113	0	6242	...	9456514	0	11215178	flv	539000	
782	183.62334	h264	1280	720	2847539	29.000000	98	5405	0	5503	...	60113035	0	65359329	mpeg4	539000	
783	294.61334	mpeg4	176	144	55242	12.000000	61	3474	0	3535	...	1950409	0	2034411	h264	820000	

	duration	bitrate	bitrate(video)	height	width	frame rate	codec	category
0	267	373	274	568	320	29.97	h264	Music
1	267	512	396	480	270	29.97	h264	Music
2	267	324	263	400	226	29.97	flv1	Music
3	267	85	55	176	144	12.00	mpeg4	Music
4	31	1261	1183	640	480	24.00	h264	People & Blogs
...	...	...	...	...	...	...	...	...
168281	68	816	560	480	360	29.97	vp8	Music
168282	68	340	273	320	240	29.97	flv1	Music
168283	68	81	55	176	144	12.00	mpeg4	Music
168284	285	1290	1181	854	480	29.97	h264	Sports

Nous avons ensuite décidé d'explorer et traiter notre dataset afin de posséder une meilleure connaissance de ce dernier et en extraire l'information utile. Pour cela, les étapes suivantes ont été réalisées:

- Importation des 2 datasets dans les variables *youtube_videos* et *transcoding_mesurment* avec pandas + aperçu des dataframe
- Suppression des colonnes jugées inutiles: id, url, framerate(est)
- Vérification et correction des types

```
duration             int64
bitrate              int64
bitrate(video)       int64
height               int64
width                int64
frame rate         float64
codec               object
category            object
dtype: object
```

```
duration        float64
codec            object
width             int64
height            int64
bitrate           int64
framerate       float64
i                 int64
p                 int64
b                 int64
frames            int64
i_size            int64
p_size            int64
b_size            int64
size              int64
o_codec          object
o_bitrate         int64
o_framerate     float64
o_width           int64
o_height          int64
umem              int64
utime           float64
dtype: object
```

```
0         mpeg4
1         mpeg4
2         mpeg4
3         mpeg4
4         mpeg4
          ...
68779      h264
68780       vp8
68781       vp8
68782      h264
68783     mpeg4
Name: codec, Length: 68784, dtype: category
Categories (4, object): ['flv', 'h264', 'mpeg4', 'vp8']
```

```
0                 Music
1                 Music
2                 Music
3                 Music
4         People & Blogs
              ...
168281            Music
168282            Music
168283            Music
168284           Sports
168285           Sports
Name: category, Length: 168286, dtype: category
Categories (16, object): ['Autos & Vehicles', 'Comedy', 'Education', 'Entertainment', ..., 'Science & Technology', 'Shows', 'Sp
orts', 'Travel & Events']
```

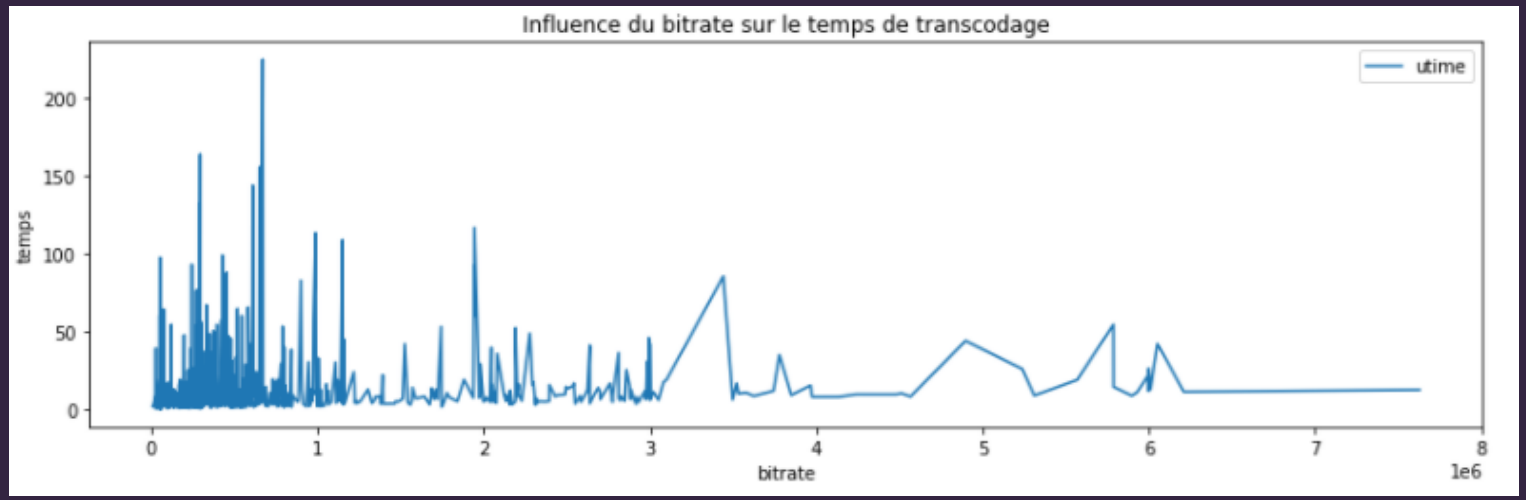
Cette étape nous a permis de connaître plus en profondeur notre dataset et ses attributs. Nous avons également pu nous assurer que les types n'entreront pas en conflit avec notre étude. Nos dataframes sont maintenant prêts à être analysés.

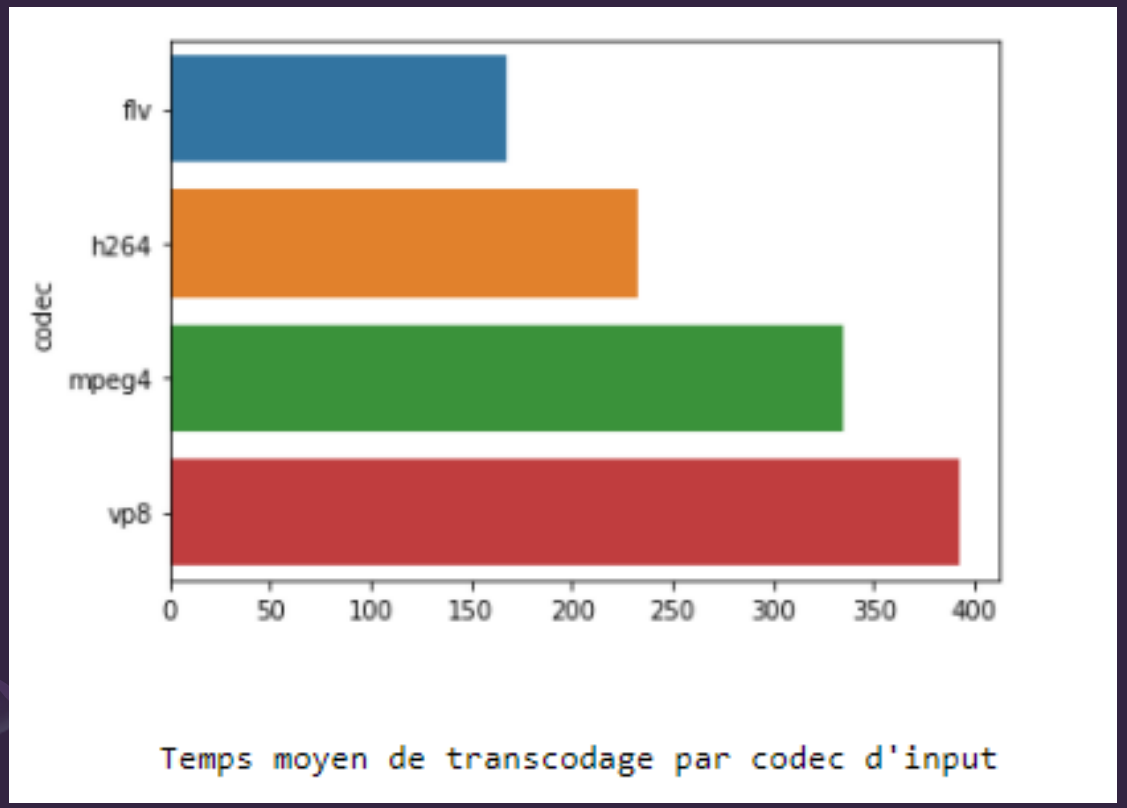
Analyse & Plots

Nous avons ensuite décidé d'analyser nos jeux de données et plus particulièrement les impacts que peuvent avoir les différents paramètres sur le temps de transcodage. Le but est de faire ressortir les corrélations existantes en amont de notre modèle de machine learning.

Pour chaque dataset, nous avons réalisés plusieurs plots afin de faire ressortir les relations entre certains éléments du dataset. L'approche était différente pour chaque dataset, les informations n'étant pas les mêmes. Le dataframe youtube_videos nous a permis une première analyse du dataset où nous avons principalement analysé les influences du choix de codec sur les différents paramètres, par catégorie de vidéos. Le dataframe transcription_mesurment nous a lui permis d'identifier les éléments pouvant impacter de façon non-négligeable le temps de transcodage. Nous incluons ici les plots les plus significatifs de notre étude.

Concernant le premier dataframe, nous avons pu faire quelques observations concernant le codec qui semble impacter le bitrate et le framerate. On peut également constater que une certaine corrélation entre le temps de vidéo et sa catégorie. Bien que cette analyse ne soit pas forcément pertinente à notre étude, elle nous permet une première approche du contexte ainsi qu'une piste – le codec sera un élément clé à étudier dans le second dataframe.



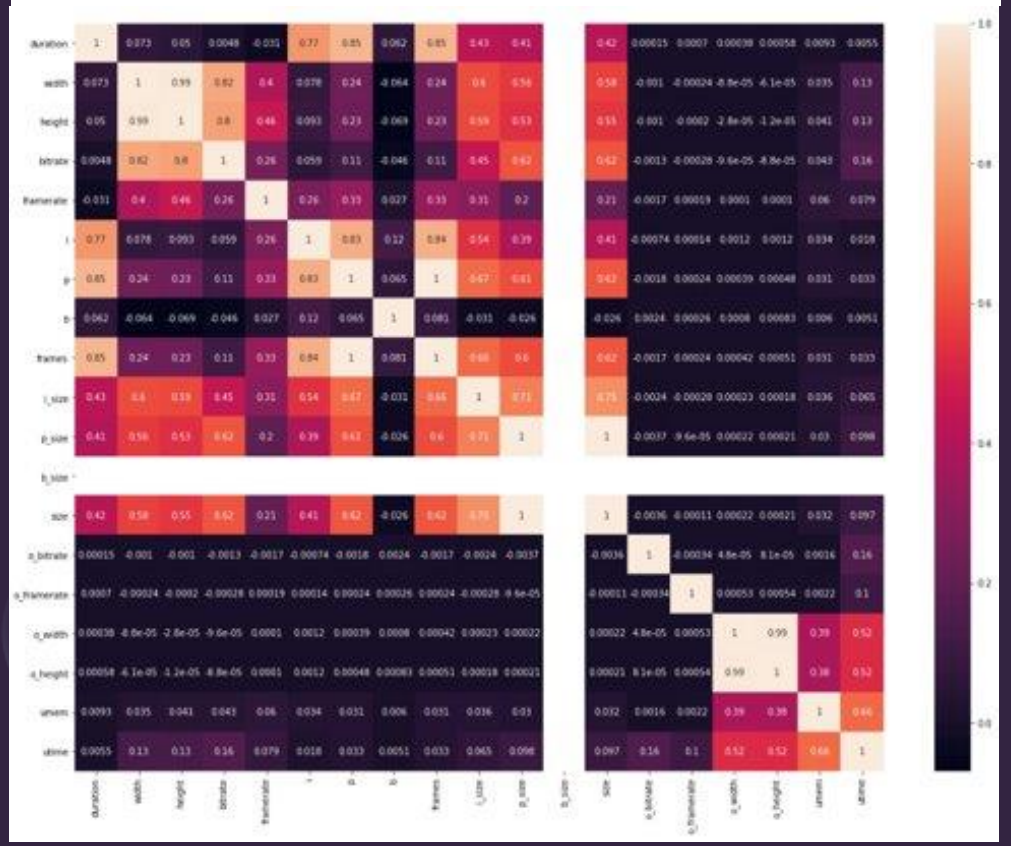


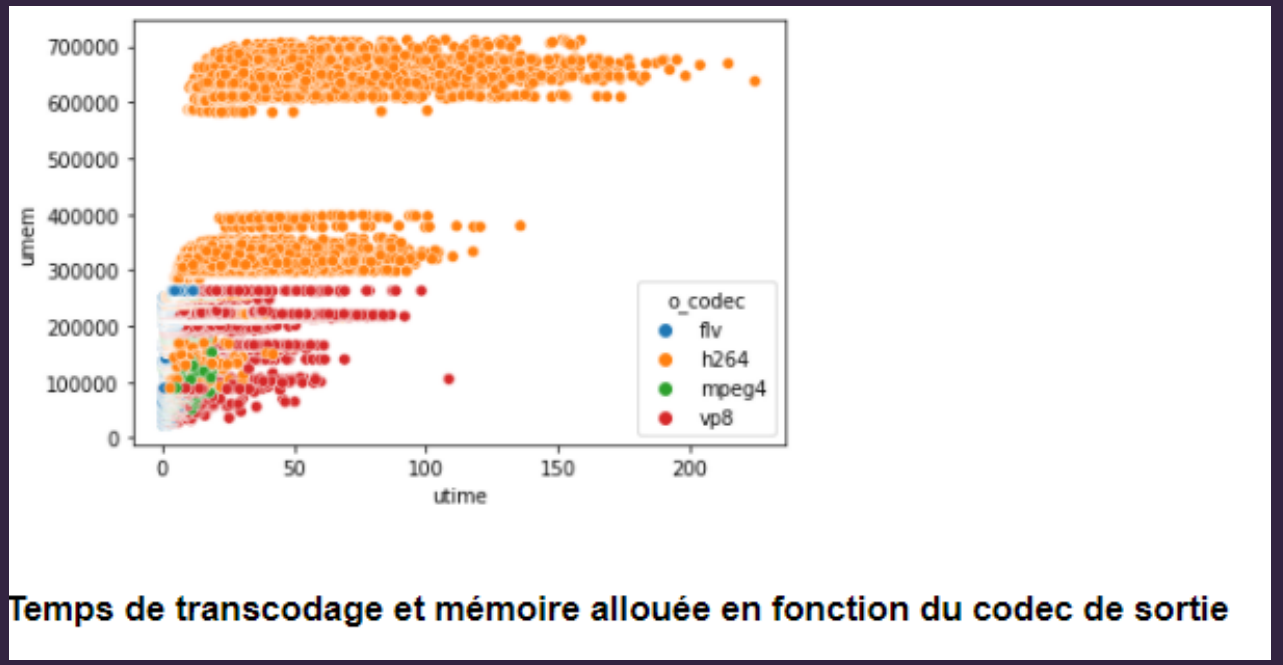
Pour le second dataframe nous avons dans un premier temps analysé les impacts des paramètres seuls sur le temps de transcodage. Notre souhait était de dégager les paramètres principaux qui peuvent avoir une influence sur le temps de transcodage, aussi bien pour l'input que l'output. Nous avons utilisé matplotlib et seaborn afin d'observer nos résultats à travers différents types de visualisations.

Nous avons également étudié le temps moyen de transcodage par codec afin d'observer s'il existe une réelle différence de performance, ce qui est effectivement le cas pour le codec d'entrée. Ces premières visualisations nous ont permis d'évaluer l'influence des différents paramètres.

Nous avons ensuite poursuivi cette analyse en incluant le codec en plus des différents paramètres. En plus de nouvelles visualisations, nous avons réutilisées les précédentes en rajoutant le codec comme paramètre afin de vérifier nos observations et y inclure l'influence du codec d'entrée ou de sortie.

Matrice de correlation du dataframe transcoding_mesurment





Temps de transcodage et mémoire allouée en fonction du codec de sortie

Enfin, nous avons réalisé une matrice de corrélation afin de pouvoir observer les différentes corrélations non seulement pour le temps de transcodage mais pour tous les éléments.
Cette analyse nous a donc offert une meilleure compréhension du contexte de l'étude ainsi qu'une visualisation de l'influence des différents éléments du dataset et des performances des différents codecs.

IV – Machine Learning

- Afin de prédire correctement la valeur de transcodage d'une vidéo youtube, il faut choisir un modèle adéquat.
- Pour cela, il faut tout d'abord regarder notre problème. Nous voulons retourner une variable quantitative en réponse à un certain nombre de paramètres définis. Ce problème s'identifie comme un problème de régression car le résultat est continu.
- Plusieurs modèles nous permettent de traiter ce type de problème : régression linéaire, support vector regression, Stochatic Gradient Descent Regression, Decision trees, Random Forrest, etc...
- Nous allons donc en tester quelques-uns afin de savoir s'ils peuvent nous aider à résoudre notre problème.

IV – Machine Learning

- Avant de tester un modèle, nous devons mettre toutes nos variables à l'échelle. En effet, elles n'ont pas toutes la même unité. Il y en a même qui désigne le codec et sont donc des strings, pour cela, nous allons utiliser "one hot encoder" afin d'en retourner une valeur.
- On va donc passer de ça (à gauche) à ça (à droite) :

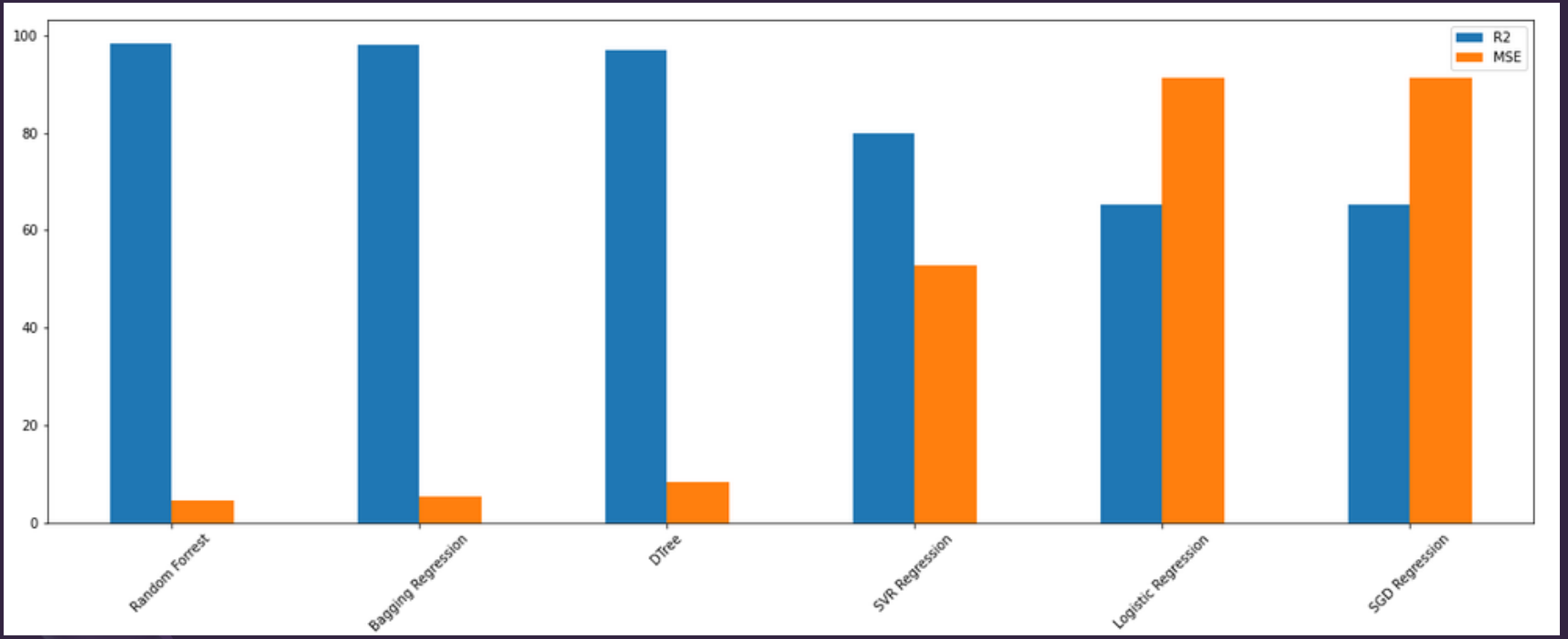
	duration	codec	width	height	bitrate	framerate	i	p	b	frames	i_size	p_size	b_size	size	o_codec	o_bitrate	o_framerate
0	130.35667	mpeg4	176	144	54590	12.000000	27	1537	0	1564	64483	825054	0	889537	mpeg4	56000	12.00
1	130.35667	mpeg4	176	144	54590	12.000000	27	1537	0	1564	64483	825054	0	889537	mpeg4	56000	12.00
2	130.35667	mpeg4	176	144	54590	12.000000	27	1537	0	1564	64483	825054	0	889537	mpeg4	56000	12.00
3	130.35667	mpeg4	176	144	54590	12.000000	27	1537	0	1564	64483	825054	0	889537	mpeg4	56000	12.00
4	130.35667	mpeg4	176	144	54590	12.000000	27	1537	0	1564	64483	825054	0	889537	mpeg4	56000	12.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
68779	972.27100	h264	480	360	278822	29.000000	560	28580	0	29140	7324628	26561730	0	33886358	flv	242000	24.00
68780	129.88100	vp8	640	480	639331	30.162790	36	3855	0	3891	875784	9503846	0	10379630	mpeg4	539000	29.97
68781	249.68000	vp8	320	240	359345	25.068274	129	6113	0	6242	1758664	9456514	0	11215178	flv	539000	12.00
68782	183.62334	h264	1280	720	2847539	29.000000	98	5405	0	5503	5246294	60113035	0	65359329	mpeg4	539000	12.00
68783	294.61334	mpeg4	176	144	55242	12.000000	61	3474	0	3535	84002	1950409	0	2034411	h264	820000	24.00

68784 rows × 20 columns

```
array([-0.77947143, -0.9698459 , -1.11709934, -0.58130467, -1.55777082,
       -1.04221202, -0.94164847, -0.10835422, -0.94558437, -0.65192934,
       -0.42430907,  0.        , -0.45132645, -0.73537489,  0.57770197,
       -0.27025258, -0.079886  , -0.09168945,  0.        ,  0.        ,
        1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
        0.        ])
```

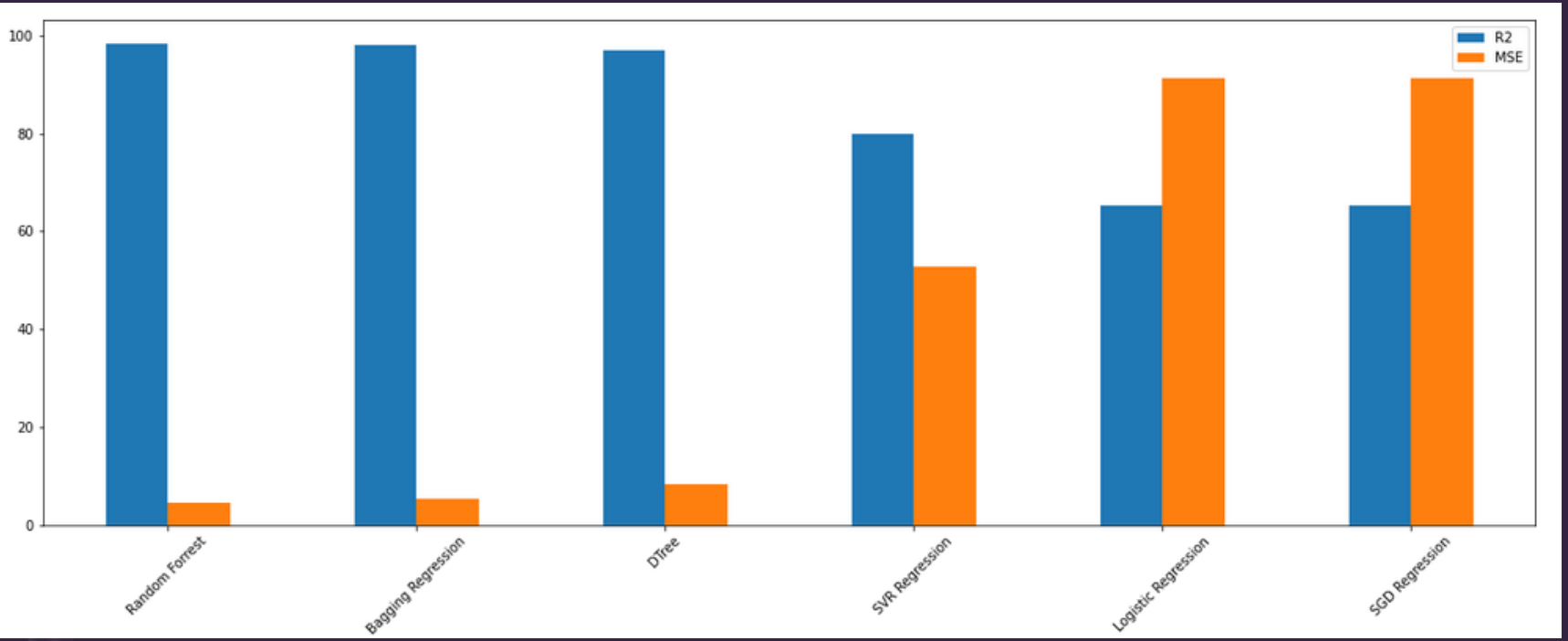
IV – Machine Learning

- On va donc pouvoir ensuite tester les différents modèles et analyser les résultats ci-dessous.



On peut voir les différents coefficients de détermination, ils se situent entre 0 et 1. Plus ils se rapprochent de 1, plus la distribution des points peut être expliquée par le modèle.
On voit ici que les modèles Random Forrest, Bagging et Dtecision Tree se rapprochent le plus de cette distribution. Les trois autres ne dépassent pas 0.8.

IV – Machine Learning

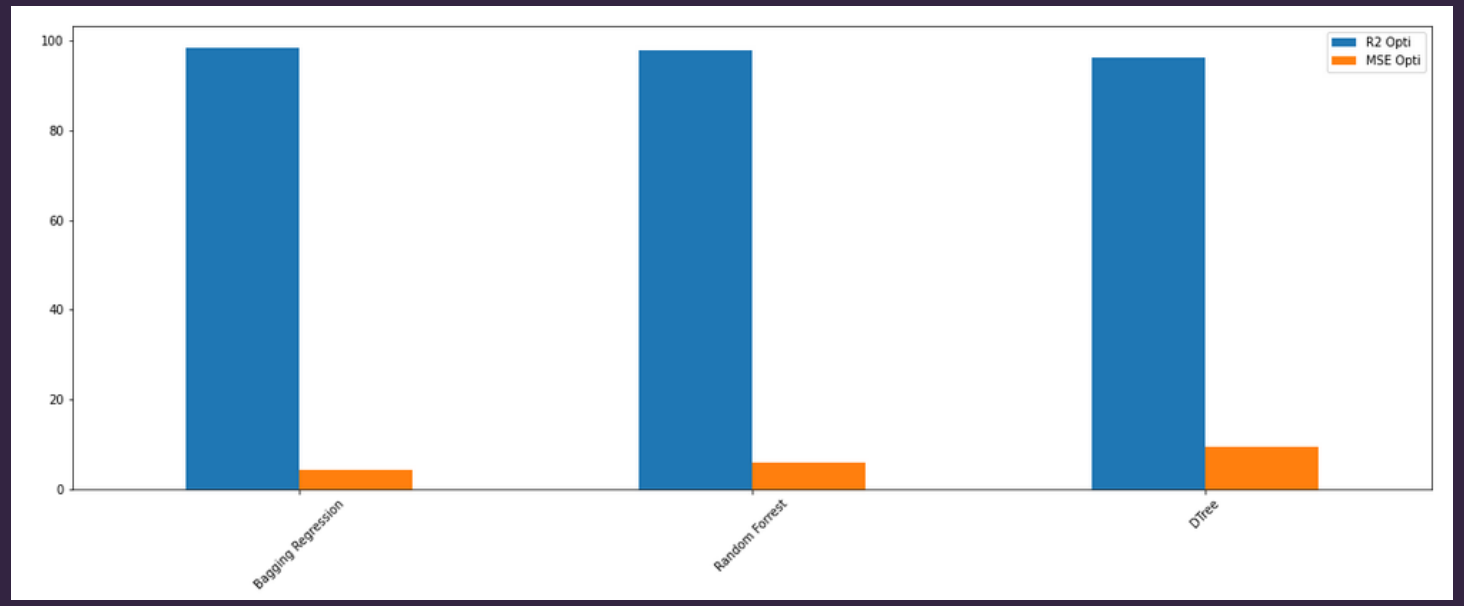


La moyenne des écart entre les points et le modèle est aussi visIble sur ce graphe.
Plus ce nombre est proche de 0, plus le modèle est proche de ces points. C'est pourquoi les trois premiers modèles sont les plus précis et permettent une prédiction plus vraie des données.

IV – Machine Learning

- Nous avons pu choisir trois modèles. Parmis ces trois modèles, il nous encore à déterminer tous les paramètres de ces derniers. Pour cela, nous avons utilisé GridSearchCV qui va tester le modèle avec différents paramètres et nous ressortir les meilleurs paramètres afin de pouvoir trancher sur le bon modèle à utiliser ici.

- Suite à tous ces tests, nous obtenons le graphique suivant :



Le modèle qui se rapproche le plus de la perfection pour notre dataset est donc bien le Bagging Regression suivant certains paramètres. Nous allons donc utiliser ce dernier pour notre API.

V - API

- Afin de pouvoir envoyer des requêtes et de recevoir la prédiction du temps de transcodage d'une vidéo youtube, nous allons créer une API contentant notre modèle.
- Le téléchargement de notre modèle se fait grâce à pickle, une librairie python. En effet, nous pouvons ensuite l'utiliser directement dans notre API.
- Nous avons ensuite développé un serveur qui est capable de traiter les requêtes envoyées vers ce dernier. La requête doit comporter plusieurs éléments mis dans un certain ordre : "duration","codec","width","height","bitrate","framerate","i","p","b","frames","i_size","p_size","b_size","size","o_codec","o_bitrate","o_framerate","o_width","o_height","umem"
- La réponse à cette requête sera le temps de transcodage de la vidéo ayant les paramètres envoyés au préalable.
- Example d'une requête envoyée vers notre API :

```
Le temps de transcodage de votre video devrait etre de : 16.187 secondes.
```