La prédiction de notes de piano

Moignet Gabriel
Leroux Thibaut

Contexte

Le sujet sélectionné a été celui des notes de pianos, car ce sont des données temporelles prenant de l'importance dans leur ordonnancement chronologique. L'idée a donc été à partir de morceaux de pianos classiques savoir s'il était possible de prédire la suite d'une musique ou du moins pouvoir en reconstituer une partie cohérente, et s'il était possible d'apprendre des motifs d'un compositeur et pouvoir cerner son style.

Ainsi pour répondre à cette question si pour certaines méthodes statistiques et naïves l'utilisation d'un seul morceau peut suffire afin de ne pas biaiser la prédiction avec plusieurs morceaux, pour l'utilisation de réseaux de neurones un seul morceau demeure trop peu. Ainsi pour allier efficacité et temps de calcul nous sommes partis sur un dataset de 50 morceaux classiques de piano répartis sur le moins d'artistes possibles afin de conserver une cohérence dans les données.

Si dans une séries temporelles plus conventionnelles les features utilisées peuvent être une moyenne, un maximum, un minimum, médiane, saisonnalité, période…Dans la musique de telles informations sont plus sensibles voire inexistantes, ainsi nous sommes partis sur l'utilisation des notes, leur durée, leur volume et leur écart temporel par rapport à la précédente note.

Les travaux existants

En l'état, la prédiction de piano est un sujet qui a déjà été fait par plusieurs personnes. On retrouve beaucoup de travaux faits par des personnes dans l'objectif de prédire les notes de piano dans un cadre de loisir, mais il y a aussi un certain nombre d'articles visant à attaquer le problème avec une compréhension du sujet approfondie. De notre côté, pour ne pas faire de redondance avec le travail déjà fait, nous avons d'abord étudié ce qui a été fait, et nous avons ensuite déterminé les techniques employées dans la majorité des cas.

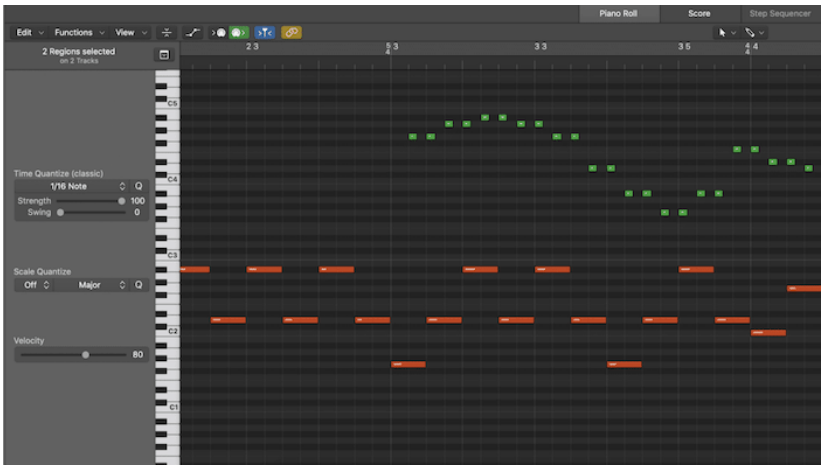
Le sujet de la prédiction de piano est à la fois vaste et restreint, il y a certes beaucoup de travaux fait dessus mais beaucoup se base sur celui de Sigorour Skuli (https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5), qui en 2017 publie un article expliquant comment il utilise des réseaux de neurones pour prédire des notes de piano. Ainsi, beaucoup de travaux présentés sur des sites découle de cet article, en gardant l'implémentation exacte à celle d'origine.

Dans ce contexte, nous sommes tomber sur le site de Tensorflow qui propose lui aussi une implémentation de deep learning pour prédire des notes de piano (https://www.tensorflow.org/tutorials/audio/music_generation?hl=fr). D'un côté le travail de Tensorflow propose un traitement des données très bon mais un réseau assez faible, alors que de l'autre Skuli proposait un traitement des données faible mais un modèle plus robuste. Nous avons donc décidé de prendre le positif de ces 2 articles et de les mélanger ensemble dans notre travail pour avoir un résultat peu similaire à ce qu'on peut voir sur internet.

Le processus préparatoire

Récupération des données

Notre sujet étant la musique, il nous faut donc obtenir des musiques pour nos données d'entraînement. Ainsi, si on veut pouvoir traiter les musiques et s'en servir avec des modèles de prédiction il faut qu'on puisse passer d'un fichier audio à un fichier avec des chiffres, le seul type de fichier qui existe permettant une telle exploitation est l'extension ".MIDI". Les fichiers vont nous permettre d'extraire chacune des notes jouées dans une musique, et lorsque celui-ci est bien fait, on peut même extraire les notes pour chaque instruments. Pour des raisons de temps et de complexité on ne peut pas prendre plusieurs instruments et afin de garder une cohérence dans les données, on ne va choisir qu'un seul instrument dont on prendra les notes : le piano. Aussi, pour ne pas trop complexifier la tâche de constitution d'un dataset, on ne va prendre que des musiques classique de piano uniquement.



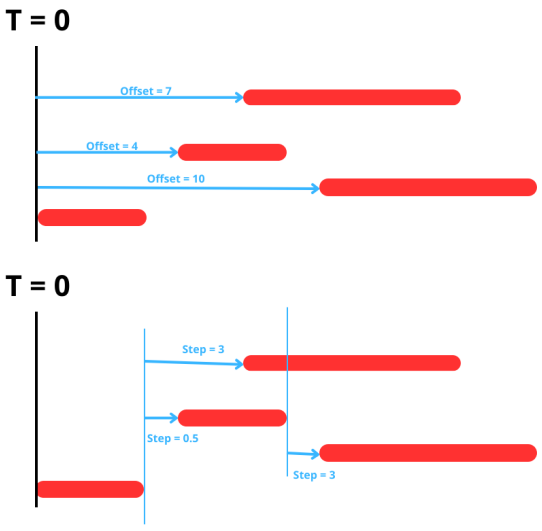
Représentation d'un fichier MIDI
https://latouchemusicale.com/wp-content/uploads/2022/08/fichiers-midi-piano.png

Une fois tous les fichiers "MIDI" de musique classique réunis avec un nombre minimum de compositeur afin de garder un minimum de cohérence lors de l'apprentissage en deep-learning, on peut commencer le traitement des fichiers.

Comme mentionné, un fichier "MIDI" contient les notes d'un instrument avec sa durée, la note en question (pitch), le moment à laquelle elle est jouée. A l'aide de la librairie 'midi21' on est en mesure de prendre chacun des fichiers et extraire chacune des notes ainsi que ses informations associées.

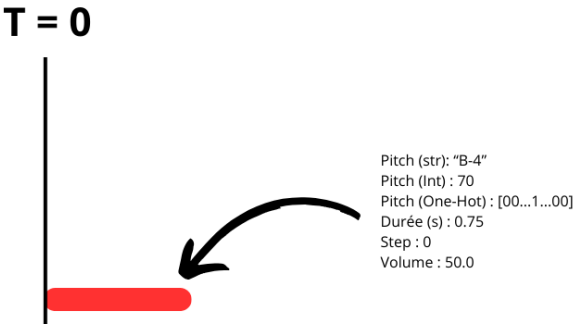
Traitement des données

Il y a d'abord le problème des "chord", une "chord" est une association d'au moins 2 notes sans aucune contrainte. Comme le nombre de combinaisons possible pour une "chord" est d'au moins 88^2, on a décidé pour les futures prédictions de décomposer les "chord" par chacune des notes qui la compose et prendre chacune de ces informations associées, ce qui est permis par "midi21".
Après l'extraction il y a différents traitements à appliquer en prévision des différents types de prédiction qui vont être effectués. Tout d'abord lors de l'extraction des notes, l'information du pitch est une "string", ainsi pour pouvoir traiter cette information plus tard à l'aide de 'midi21' on peut facilement récupérer le pitch sous forme d'entier associée à la note.
Ensuite il y a l'information de quand démarre la note (l'offset), en l'état cette donnée dépend fortement du morceau puisque plus un morceau sera long plus l'offset le sera aussi. Lors de l'entraînement en deep-learning cette donnée sera plus aléatoire dans le sens qu'elle ne permet pas une chronologie parfaite. En prédisant l'offset on se risque à obtenir énormément de notes au même endroit, même si elle est normalisée, cela demandera un entraînement bien plus long que pour les autres variables. Pour des raisons de complexités, nous avons préféré transformer l'"offset" en "step". Si l'"offset" représente le moment où la note est jouée, le "step" représente la durée à laquelle est jouée par rapport à la fin de la note précédente, de cette manière cette nouvelle feature permettra une prédiction plus facile et qualitative. Cette nouvelle feature tient mieux compte de l'espacement entre chaque note, ce qui est implicitement contenu dans l'"offset". Pour calculer le "step" de chaque note, il suffit juste de se servir de "l'offset" et de le soustraire au précédent à condition qu'il soit trié dans l'ordre chronologique.



Différence entre l'offset et le step

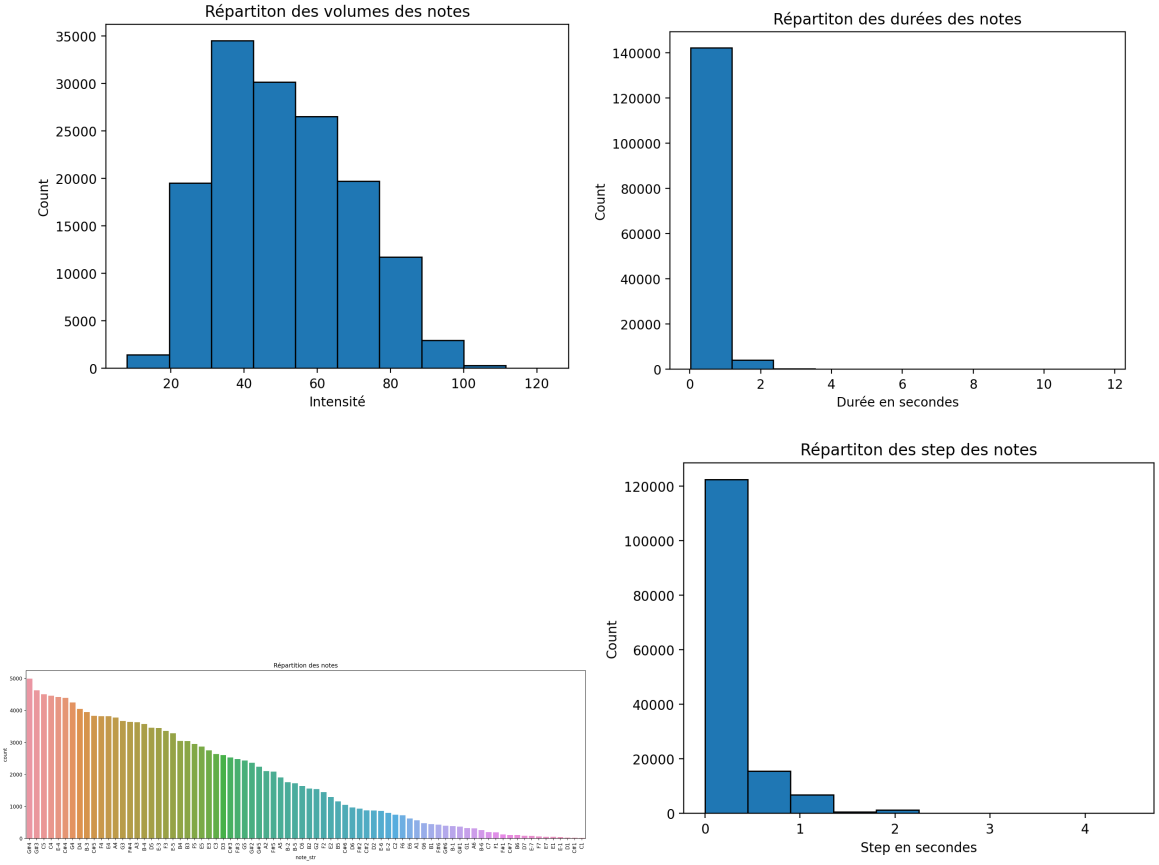
Enfin, pour faire des tests selon l'encodage de données, on se propose d'aussi transformer les notes en one-hot. Même si les possèdent sous forme d'entier, on va garder leur valeur en chaîne de caractère et les one-hot encoder.



Caractéristique que l’on prendra sur chaque note ainsi que son format

Analyse exploratoire

Avec les 50 morceaux que l’on possède, également répartis entre Chopin et Schubert. On va maintenant essayer d’explorer ce que les données nous proposent et la manière dont les notes sont réparties. Après avoir récupéré la note sur chacun des morceaux, en les rassemblant tous dans un dataframe, on peut mener une première étude exploratoire.



Différents graphiques de répartition des données. (Haut gauche) Répartition du volume des notes. (Haut droit) Répartition de la durée des notes. (Bas gauche) Répartition du pitch des notes. (Bas droit) Répartition du step des notes

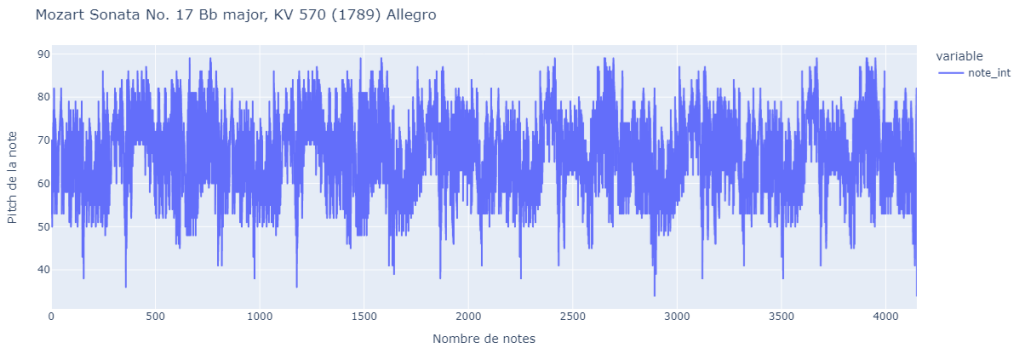
Chaque composante d’une note à des répartitions déséquilibrées à l’exception du volume. Comme on peut le voir, le pitch n’est pas également réparti mais il y a une progression constante de l’utilisation de certaines notes, dans le sens où la note la plus utilisée n’a pas un écart significatif avec la deuxième note utilisée. Par contre, du côté du step et de la durée

d’une note, la répartition est totalement déséquilibrée puisque la classe dominante constitue plus de 80% de la répartition. Ce déséquilibre dans les données va rendre la prédiction sur ces variables beaucoup plus compliqué, et peut-être demander des traitements supplémentaires afin de rendre ces prédictions moins concentrées, et c’est pour ces raisons que l’on utilisera uniquement le pitch des notes pour la prédiction statistique.

Différentes méthodes de prédiction

Avant d'entamer l'essai de différentes méthodes de prédictions pour voir laquelle/lesquelles marchent le mieux, il est nécessaire de préciser que les données utilisées n'auront pas la même forme. Ainsi pour les méthodes purement statistiques (naïves, statistiques) on ne pourra utiliser qu'un seul morceau puisque les librairies permettant de ne que faire des prédictions sur des séries temporelles uniques. On pourrait joindre les partitions de différents morceaux et les mettre à la suite pour agrandir le jeu de données, le problème étant qu'en les concaténant les notes perdent du sens d'un morceau à l'autre, et on fournit des prédictions faussées pour l'entraînement.
Donc, il n'y aura que pour les réseaux de neurones que le dataset de 50 morceaux pourra être réellement utile.

Ainsi, pour les prédictions statistiques nous utiliserons un morceau de Mozart présentant des motifs récurrents lors de son morceau, ce qui facilitera l'utilisation des modèles ARIMA.

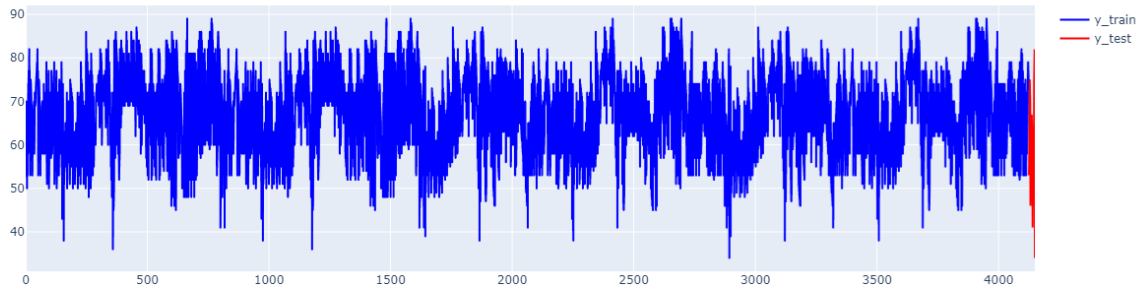


Représentation graphique des notes du morceau (le notes ont été converties en entier)

Dans tous les cas, que la méthode soit naïve, statistique ou un réseau de neurones on va principalement axé la prédiction sur les notes, même si elle pourrait se faire sur la durée de la note, son volume ou même le step.

Méthodes simplistes

Dans un premier temps nous allons décrire l'utilisation de méthode dite "simpliste" en comparaison avec celle des réseaux de neurones qui est beaucoup plus complexe. Toutes les méthodes décrites dans cette partie ne demandent pas de traitement particulier des données si ce n'est la transformation du pitch des notes en entier. L'entraînement et la prédiction se feront à chaque fois sur une musique, la librairie 'sktime' étant adaptée des séries temporelles, nous n'allons donc pas créer des classes ou de labels.

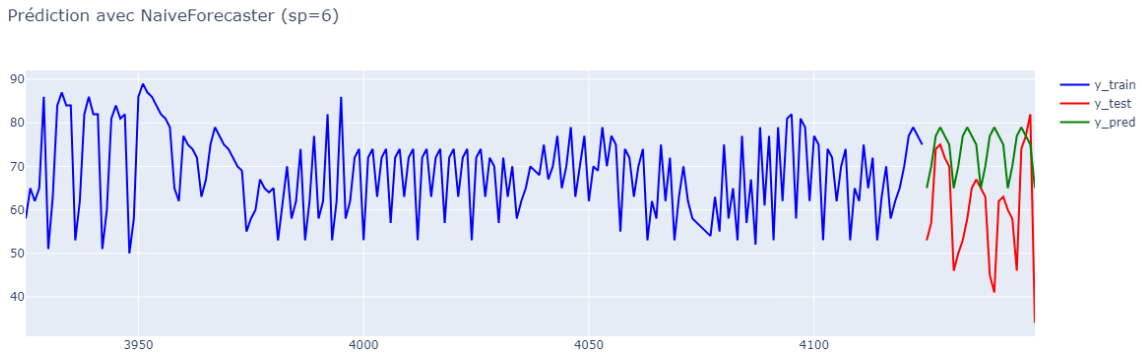


Répartition des données qui seront à utiliser pour l'entraînement et pour comparaison des futures prédictions.

La répartition des données est telle que l'on prendra notre morceau, dont on garde les 25 dernières notes comme test et le reste comme entraînement.

Méthode naïve

En se basant sur le travail effectué en TP, on va utiliser la librairie sktime pour effectuer les prédictions. On commence par essayer une méthode naïve, c'est-à -dire qu'il n'y a pas d'entraînement puisque le modèle va prédire en fonction de conditions applicables à tout instant. Ainsi, dans le contexte de la musique il serait pertinent d'appliquer une prédiction basée sur les dernières notes puisque souvent un rythme se reproduit, donc en choisissant un certain décalage on pourrait mettre en place une prédiction naïve suffisante.
Cette première approche permettra de créer une base de référence pour les futures méthodes de prédiction.



Prédiction de la méthode naïve face aux notes réelles

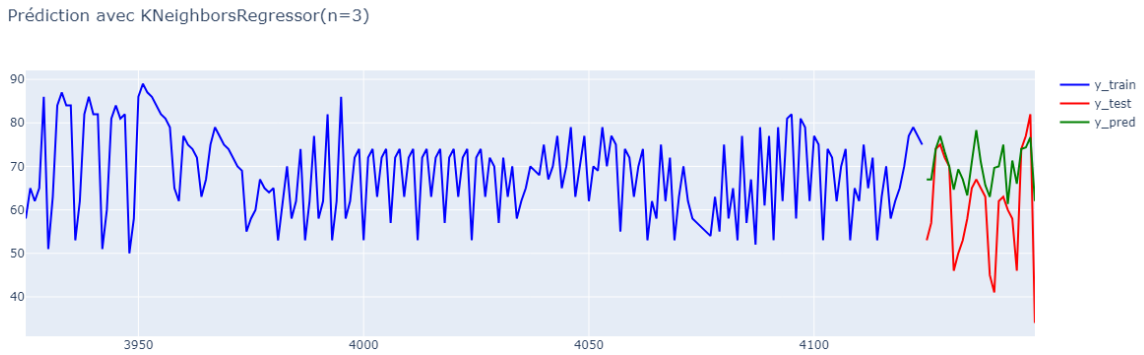
Le résultat est qu'il n'y a que 0.27% d'erreur moyenne, ce qui est plutôt faible et bon signe. Même si cela marche car c'est une prédiction à court terme, en augmentant l'horizon l'erreur augmente car la musique n'est pas assez répétitive, et surtout parce que le modèle n'est pas assez évolué pour comprendre ces changements à long terme.

Méthode statistique

Maintenant que l'on a essayé une méthode naïve, on est tenté de voir si une méthode qui puisse apprendre à une meilleure efficacité au niveau des prédictions. Pour cela, nous allons mettre en place différents types de méthodes statistiques et voir laquelle fonctionne le mieux.

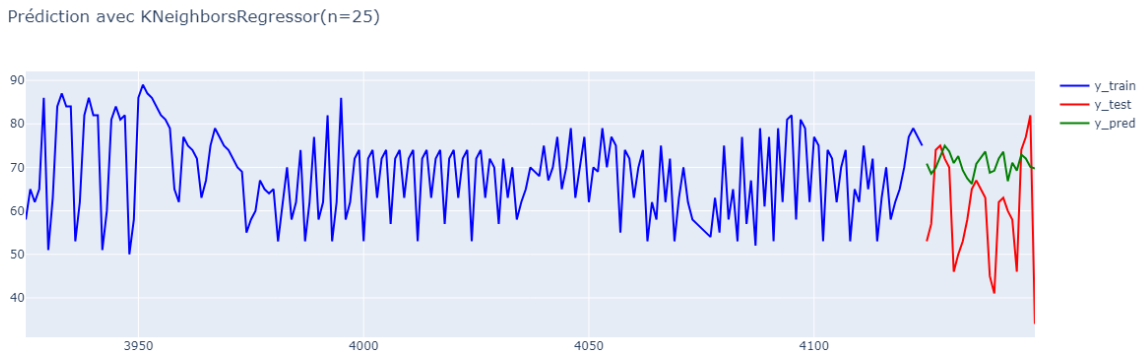
Les K-Neighbors

On commence par essayer une méthode des k-neighbors. En partant du principe qu'une musique ait une mélodie ou un refrain, cela signifie que plusieurs fois dans un morceau on verra des motifs se répéter et donc avoir une similarité. Car un modèle des k-neighbors se base sur la similarité, cela pourrait être utile pour prédire des notes similaires à des données passées. Cela peut permettre de prendre en compte les dépendances temporelles et la structure musicale dans le morceau entré, on s'attend tout de même à une efficacité limitée, le modèle ne pourra capture que des dépendances peu sophistiquées.



Prédiction de la méthode des plus proches voisins (3 voisins) face aux notes réelles

Avec le nombre de voisins fixés à 3 on a une perte de 0.202%. Mais comme mentionné , avec seulement 3 voisins on prend des motifs assez peu généraux. En augmentant, on pourrait soit augmenter la généralisation puisqu' il y aura plus de voisins pour cerner certaine tendance, mais cela peut aussi desservir dans le cas où la musique à très peu de similarités.

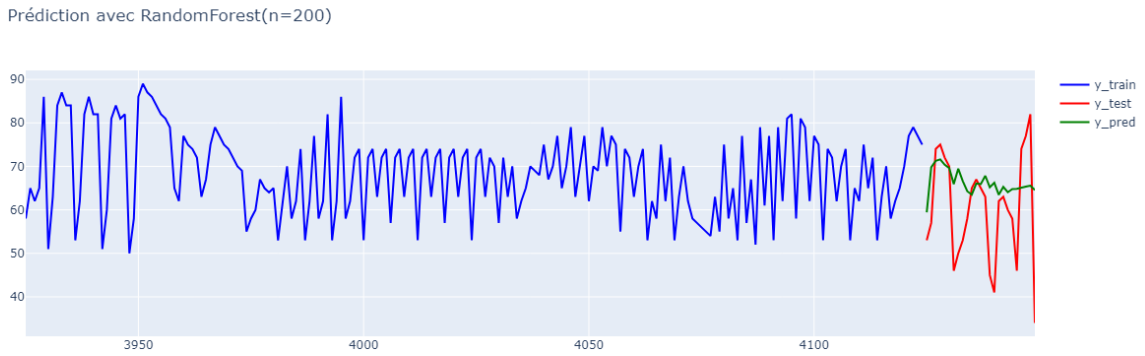


Prédiction de la méthode des plus proches voisins (25 voisins) face aux notes réelles

Avec un nombre de voisins fixés à 25, on a une perte de 0.248%.
Donc ici en augmentant le nombre de voisins pris en compte, on rentre dans le deuxième cas, on apprend des motifs récurrents ce qui n'est pas le cas durant tout un morceau. Même si on est passé de 0.202 à 0.248, l'erreur est assez faible en général. Cette diminution par rapport à la méthode naïve montre qu'une musique a des motifs réguliers et confirme que l'utilisation d'une méthode des plus proches voisins restent cohérente avec de telles données.

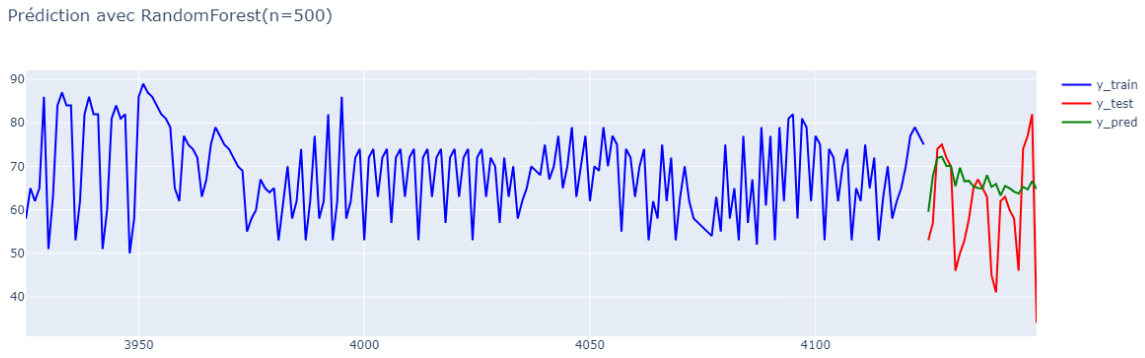
La RandomForest

Ensuite, on peut essayer une méthode de forêt aléatoire. Une telle méthode permet de modéliser des relations complexes entre des variables, et pour la musique cela pourrait s'adapter. Chaque morceau à des structures différentes, certains auront beaucoup d'harmonies à l'ouïe et de similarités qui se produisent lors du morceau, d'autres non. Si la méthode des k-neighbors aurait une faiblesse sur des morceaux ayant peu de similarités, la forêt aléatoire pourrait être efficace sur tout type de morceau et spécialement ceux étant non linéaire et ayant peu de similarités répétées. La forêt aléatoire pourrait capturer des relations non linéaires entre différentes notes, ce qui rend ce modèle le plus adapté de tous en théorie pour l'instant.



Prédiction de la méthode de forêt aléatoire (200 estimateurs) face aux notes réelles

Avec un nombre d'estimateurs de 200, on a une perte de 0.188%, ce qui est pourtant la méthode présentant la perte la plus basse. Même en regardant la prédiction elle semble suivre la tendance générale, même si cela manque d'amplitude.



Prédiction de la méthode de forêt aléatoire (500 estimateurs) face aux notes réelles

Avec un nombre d'estimateurs de 500, on a une perte de 0.146%. Donc en augmentant le nombre d'estimateurs, le modèle cerne les motifs plus précisément et donc apporte de meilleure prédiction. On peut même discerner de meilleures amplitudes qu'avec 200 estimateurs. Cette approche avec une forêt aléatoire est donc cohérente avec ce type de données, puisque les résultats semblent corrects.

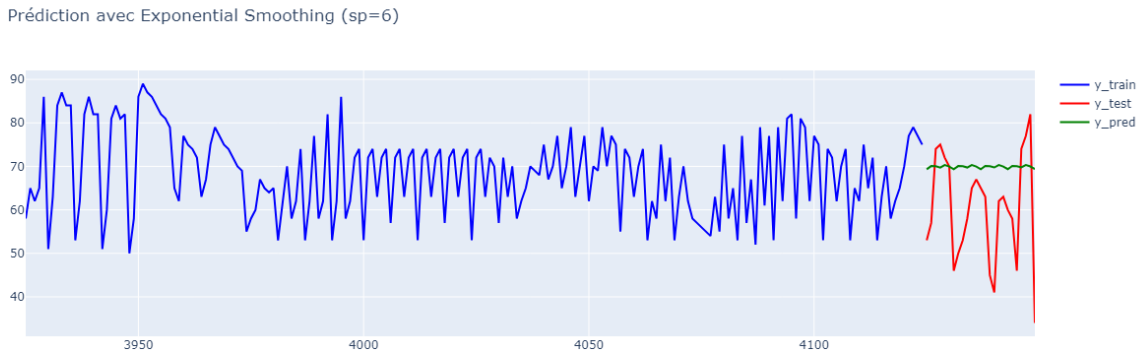
La musique comme série temporelle (tendance et saisonnalité)

Les 2 méthodes précédentes partent du principe qu'une série temporelle puisse être quelconque, à l'origine ce sont même des modèles applicables à n'importe quels domaines. Les suivants que l'on souhaite essayer sont propres aux séries temporelles, les séries qui possèdent une tendance et des saisonnalités. En pratique, le terme de tendance et de saisonnalités n'ont aucun sens dans le monde de la musique, par contre en voyant plus loin et comme on souhaite essayer ces méthodes, on peut rapprocher la saisonnalité d'une série temporelle à une mélodie d'une musique qui se répètera plusieurs fois, et la tendance serait le motif récurrent derrière ces mélodies.
Ainsi, pour toutes les méthodes décrites en dessous on part du principe que la musique dont on souhaite prédire des notes possède une mélodie qui se répète afin de remplir ce critère de saisonnalité.

Exponential Smoothing

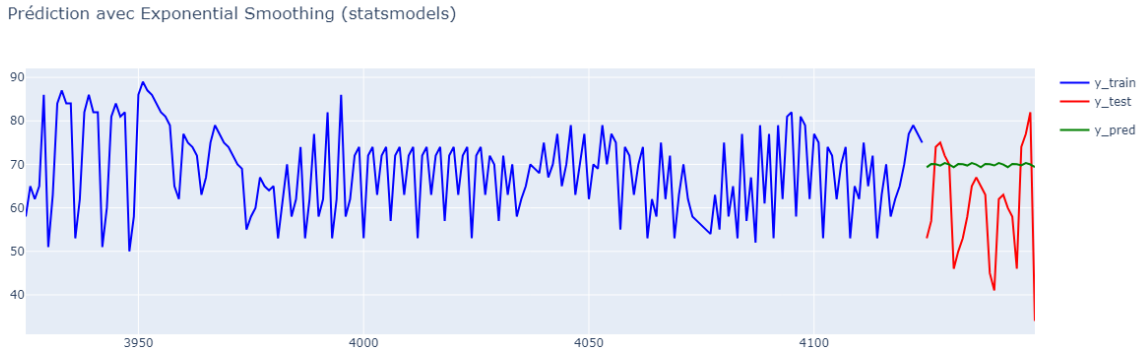
La technique de lissage exponentiel est très souvent utilisée pour des séries temporelles qui varient peu et qui sont assez lisses. Cette description est le total inverse des notes de musiques que l'on peut voir sur des graphiques, c'est ample, il y a énormément de variations. On s'attend alors à avoir de très mauvais résultats, et dans un deuxième temps cela permettra aussi de confirmer le caractère du lissage exponentiel d'être adapté aux séries peu variées et lisses. De plus, il y a aussi des raisons de saisonnalité et de tendances

qui ont été expliquées ci-dessus.



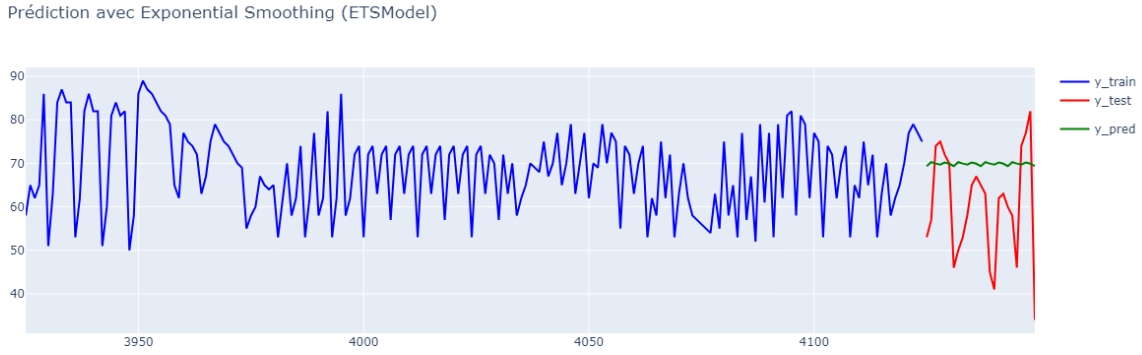
Prédiction de la méthode de lissage exponentiel (librairie : sktime) face aux notes réelles

Avec ‘sktime’ on obtient une loss de 0.245%.



Prédiction de la méthode de lissage exponentiel (librairie : statsmodels) face aux notes réelles

Avec ‘Statsmodels’ on a une loss de 0.245%



Prédiction de la méthode de lissage exponentiel (ETSModel) face aux notes réelles

Et finalement, avec ‘ETSModel’ on a une loss de 0.246%.

Malgré différentes implémentations du lissage exponentiel les résultats et observations sont les mêmes avec une loss avoisinant les : 0.25%. Le chiffre à beau l'être faible, mais en l'état c'est une des pires prédictions qui ait été faite, moins bonnes que celle naïve. Comme décrit plus haut, le lissage exponentiel est adapté aux séries peu variées et assez lisses, l'inverse même de notre signal de musique. Cela se retranscrit à travers la prédiction qui ressemble presque à une ligne droite et ne prend en compte aucune amplitude et aucune tendance. Ça confirme donc les aprioris que l'on avait avant d'essayer cette méthode, et que le lissage exponentiel est peu adapté pour une telle prédiction.

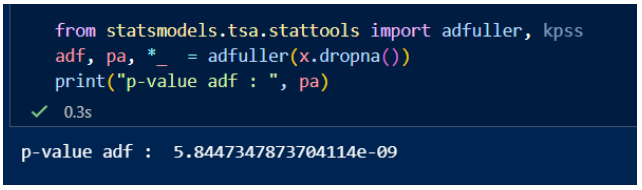
ARIMA

La méthode ARIMA est un peu moins restrictive que celle du lissage exponentiel, puisque ici il faut que la série temporelle à laquelle on applique cette méthode n'ait pas de tendances et de saisons : elles doivent être stationnaires. Quand bien même la musique que l'on souhaite prédire n'est pas stationnaire, on peut lui soustraire sa tendance et ses saisonnalités afin de la rendre hypothétiquement stationnaire. Si déjà la notion de saisonnalité était très peu adaptée aux musiques, en effectuant ces calculs on prend le risque d'aggraver le résultat des prédictions. On ne s'attend pas à avoir de très bons résultats avec l'ARIMA.

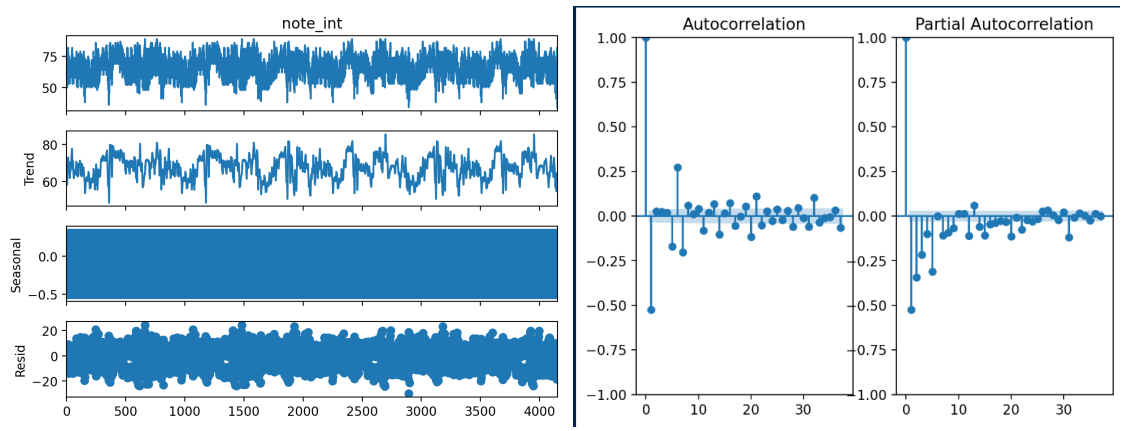
Mise en place de l'ARIMA

Comme décrit plus haut, la mise en place d'un ARIMA nécessite une série temporelle sans saisonnalité. De plus, pour concrètement mettre en place un ARIMA on doit déterminer les ordres du modèles avec partir de l'autocorrélation du signal.

Avant de calculer les ordres directement, il faut savoir si la série temporelle est stationnaire, dans notre cas la valeur du test de dickey-fuller est en 10^-9 ce qui est assez faible, on se propose néanmoins de différer une fois la série pour être sûr.



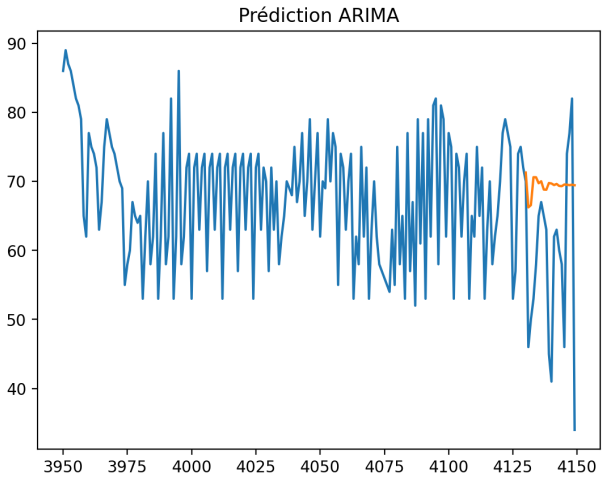
Une fois cela fait, on va décomposer notre série temporelle en saisonnalité et tendance pour soustraire au signal original ses saisonnalités. En partant du principe qu'une période est équivalente à un rythme récurrent, on déconstruit la série temporelle avec une période de 6 pour obtenir les graphs suivants :



Grâce aux graphiques d’autocorrélation on peut déterminer que notre modèle ARIMA sera tel que : ARIMA(5,1,0).

Prédiction avec l'ARIMA

Possédant les ordres du modèle, on peut effectuer la prédiction.

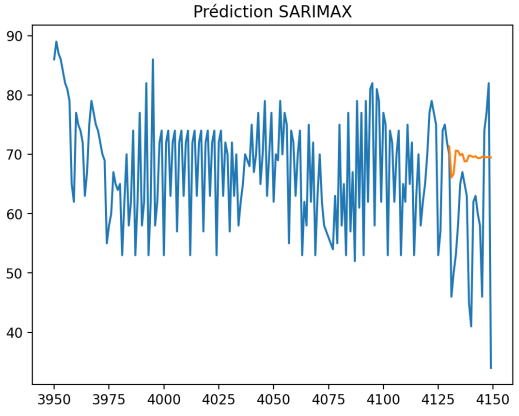


Prédiction de la méthode ARIMA mis à la suite du morceau

La prédiction semble très loin de la réalité, la loss est d'environ 0.26, mais il semble tout de suite logique de dire que la prédiction est très loin des valeurs réelles.
Comme attendu, le modèle n'est pas non plus assez adapté pour de telles séries étant données qu'on part initialement sur des séries très variées.

SARIMAX (optionnel)

Le SARIMAX est une sorte d'extension du modèle ARIMA à qui on donne la possibilité de prendre en compte des variables exogènes pour améliorer les prédictions. Cela peut s'avérer utile lorsqu'il y a une relation entre les 2 variables sélectionnées, ce qui n'est pas forcément le cas des notes et d'une de ses composantes.
En effet, dans certains morceaux, on peut voir une relation entre les notes et les volumes de celles-ci, dans ces cas là le SARIMAX pourrait être applicable. Encore est-il que comme pour l'ARIMA et lissage exponentiel, cette méthode est adaptée aux séries temporelles avec des tendances et des saisonnalités, ce qui n'est pas forcément le cas de la musique qui représente un signal variant énormément. Une fois de plus, on ne s'attend pas à avoir de résultats convaincants.

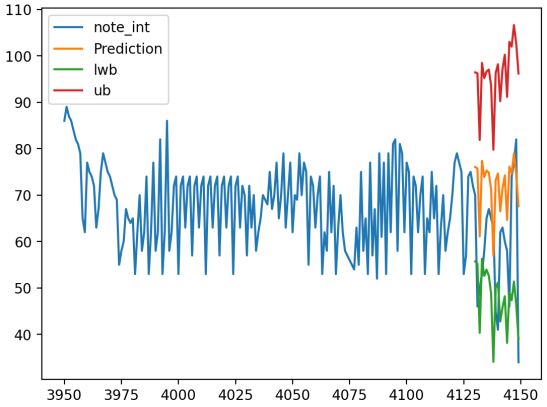


Prédiction de la méthode SARIMAX mis en parallèle des notes réelles

La prédiction du modèle SARIMAX, même si tenant compte des saisonnalités, reste très décalée par rapport à ce qui est attendu. Une fois de plus la loss est d'environ 0.26, mais à vue d'œil on sait que ca ne sera pas efficace en comparaison avec les précédents modèles statistiques.

Auto ARIMA

Les musiques considérées comme séries temporelles sont assez complexes, puisque peuvent être considérées telles seules les morceaux possédant une saisonnalité et une tendance, soit certains morceaux. Ainsi, certaines fois déterminer les ordres des modèles ARIMA est assez compliqué, c'est pourquoi l'usage d'un auto ARIMA pourrait aider dans la mise en place d'un tel modèle.



Prédiction du modèle en utilisant un auto-arima

Si précédemment on avait utilisé un modèle ARIMA en déterminant nous mêmes les ordres, le résultat était moyen. En utilisant un auto-arima pour avoir le meilleur modèle selon son AIC on peut voir que la prédiction est très correcte et a cerné les différentes nuances de la musique. Même si la prédiction est un peu haute, dans son amplitude et sa variation elle est presque parfaite.

Cette observation nous montre que malgré tout un modèle ARIMA peut être adapté à condition d'avoir les bons ordres du modèle. Et c'est cette même condition qui va faire qu'on va garder le modèle ARIMA comme moyen, puisque ces ordres sont compliqué à déterminer; si dans le cas présents on utilisait une musique avec des "saisonnalités" marquées, ce n'est pas le cas de tous, ce qui implique une détermination des ordres plus sommaires. Si trouver les ordres exacts est la condition pour avoir un modèle ARIMA parfait cela reste compliqué, étant donné qu'un ordre différent changerait totalement la prédiction, ce qui rend cette condition d'obtention encore plus parfaite mais plus difficile à atteindre.

Méthode réseaux de neurones

Après avoir essayé les différentes méthodes statistiques on est tenté de voir si les réseaux de neurones peuvent faire mieux. L'intérêt de telles méthodes va être de pouvoir faire des choses plus complexes surtout dans notre cas.
Si jusqu'à maintenant on ne faisait que prédire les notes et qu'on ne tenait pas compte du contexte (à l'exception de SARIMAX) ou des composantes de la note comme le volume, la durée ou le step, si on voulait intégrer ce contexte cela aurait demandé la mise en place d'au moins 4 SARIMAX pour prédire les 4 composantes en tenant compte du contexte des autres composantes. A cela s'ajoute le fait qu'une telle méthode qui aurait pu être fonctionnelle nécessite que les données présentent des tendances et de saisons, ce qui n'est pas forcément le cas des musiques en général.

Les réseaux de neurones vont permettre de pouvoir prédire avec du contexte, le tout en permettant de prédire les 4 composantes de la note de musique simultanément à partir d'entrées sans tenir compte d'une potentielle saisonnalité ou tendance, puisque tout ça sera appris. En plus de tout ça, si jusqu'à maintenant on était bridé à la prédiction et entraînement sur un seul, cette nouvelle méthode va permettre l'apprentissage sur de multiples morceaux et donc considérer plus de grands nombre de motifs et récurrences.

Le développement utilisant un modèle de réseaux de neurones est donc l'endroit où l'on a consacré le plus de temps.

Changement dans les données

Si jusqu'à maintenant pour les méthodes simplistes on n'utilisait qu'un seul morceau de musique pour l'entraînement et la prédiction, et que les notes étaient sous formes d'entiers. Avec le deep learning on a opéré à quelques changements puisque pour des raisons d'efficacité, on va transformer les notes en classes one-hot encodées et créer manuellement des séquences d'une longueur donnée : les données d'entraînement (X), et la note suivante correspondante : les données à prédire (Y).

Dans un premier temps, on prend l'intégralité des morceaux auxquels on extrait les notes ainsi que chacune de ses composantes (durée, volume, step), puis on les rassemble toutes dans un dataframe. A partir de ce dataframe, on connaît toutes les notes jouées dans les données d'entraînement, on peut donc aisément entraîner le one-hot encodeur sur les notes ; et il en va de même pour les volumes ou durée si on veut.

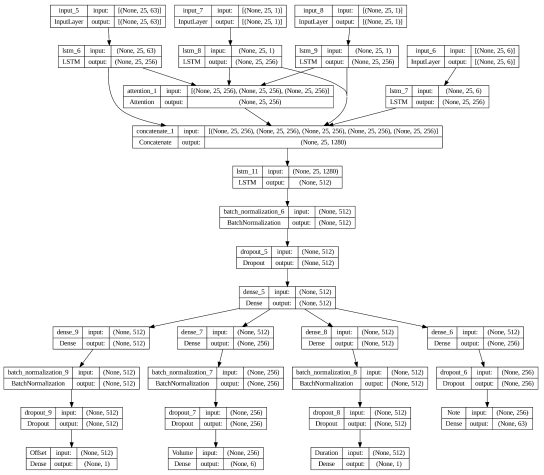
Une fois les one-hot encodeur entraîné, on va commencer à constituer les dataset d'entraînement. On va prendre les notes de chacun des morceaux, encoder les notes, et former des séquences d'une longueur donnée (entrée) ainsi que la note suivante (sortie). De cette manière nous aurons des entrées et sorties différentes pour : les notes, le step, le volume et la durée.

Le modèle

Les données sont prêtes à l'emploi. Comme mentionné, il y a 4 entrées et 4 sorties, cela est dû à la structure du modèle qui est tel.
A ce moment plusieurs options s'ouvrent :

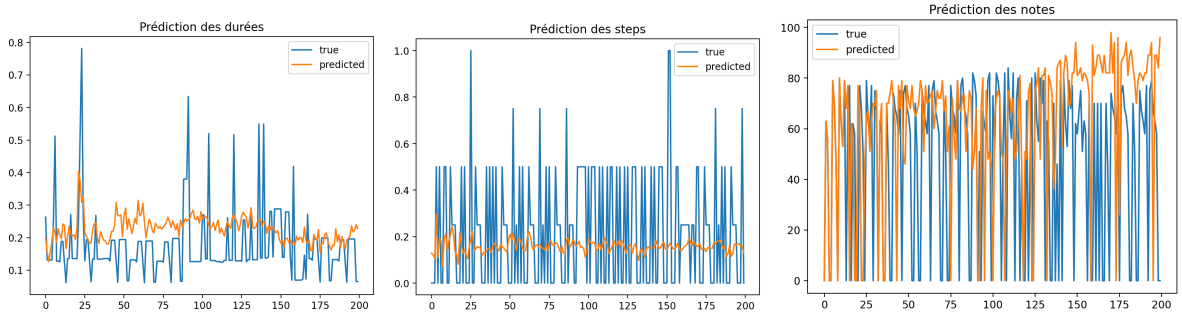
- Prédire les notes comme un bloc de composantes indivisible (note, step, volume et durée). Car la note dans son block a un caractère temporel que l'on souhaite prédire
- Ou bien on prend chacune des composantes de la note, car chacune des ces composantes a un caractère temporel, de cette manière on aura plus de liberté et de nuances ainsi qu'un meilleur apprentissage. Cela implique d'avoir 4 entrées et 4 sorties.

Notre architecture de modèle s'est donc arrêtée sur celle-ci :



En prenant chaque composante comme entité à part entière, chacune a son entrée accueillie par un LSTM. Ensuite on concatène ces 4 entrées, car même si chacune des composantes à un caractère temporel, il n'en demeure pas moins que la note (assemblage des composantes) a aussi un caractère temporel. On soumet aussi cette concaténation a des LSTM, que l'on divise en 4 sorties : une pour chaque composante.
De cette manière nous prédisons à la fois la note, mais aussi sa durée, son step et son volume.

Résultats



Après avoir entraîné le modèle sur les 50 musiques, en essayant de faire une prédiction sur un morceau de Chopin (**Grande Valse Brillante, Opus 18 (1831)**), on peut voir que les prédictions, peu importe la composantes, sont assez peu précises par rapport aux données originales. Respectivement, les SMAPES loss sont de : 0.39, 2.09E16 et 1.36 pour les durées, les notes et les step. Si en soit les chiffres ne sont pas marquants car les plages de valeurs de données sont assez faibles pour certaines, il n'en demeurent pas moins que graphiquement c'est en total décalage. Au-delà du plan graphique, auditivement le résultat n'est pas forcément extrêmement mauvais, mais il est loin d'être bon, la mélodie initiale se perd dans un embouteillage de notes. Au niveau des durées et des step, on peut voir sur les données originales que cela fonctionne sous forme de pallier se répétant tous les X temps, la prédiction est elle bien plus mobile puisqu'elle prendra une multitude valeurs à l'instar des donnés originales qui n'en prennent que quelques unes. Pour les notes, l'observation est que les premières notes prédites sont correctes, mais plus que le temps avance, plus les données prédites servent pour la prédiction, les notes prédites sont à ce moment en totale liberté étant très loin de l'original. Une chose qu'on peut noter néanmoins, même si le modèle prédit à côté des notes réelles, il y a la présence de récurrence dans les prédictions avec des schémas se répétant : il y a donc une mélodie.

Les points d'améliorations

Notre problématique initiale avait pour principal but d'être résolu par l'utilisation du deep-learning, au vue des résultats ce n'est finalement pas la meilleure méthode pour bien des raisons. Ce n'est pas tant du côté technique, c'est que notre implémentation n'est pas parfaite et peut être grandement améliorée pour la rendre plus robuste.

- La quantité de données d'entraînement : en l'ayant entraîné sur seulement 50 musiques (Chopin et Schubert) on a permis au modèle de connaître les caractères récurrents de ces 2 artistes : une musique rythmée et rapide. Ce qui est à double tranchant, d'un côté le modèle est spécialisé et à cerner le style de musique de ces 2 artistes, mais cela contraint énormément les prédictions. Le modèle ne prédira presque jamais des notes longues ou des temps de pauses, puisqu'ils sont eux mêmes non présents dans les musiques des ces 2 compositeurs. En ayant plus de données d'entraînement, avec des musiques aux tempos variés et différents, le modèle serait en capacité d'avoir une plus grande liberté de prédiction, et ne serait pas contraint de ne pas prédire un certain type de note car il n'en a pas vu d'autres.

- L’encodage des données : pour la prédiction des notes nous avons choisis le one-hot, pour la durée et les step nous avons choisis une régression. L’analyse exploratoire nous montrait à quels points les classes peuvent être déséquilibrées ce qui influencerait la prédiction. Ce problème est donc intimement lié à celui précédent, peut-être qu’au lieu de faire de la régression sur les durée, en les encodant sous forme de classes et en appliquant des poids de classes on aurait pu résoudre ce problème où les notes courtes sont prédites en majorité.
- Pour finir, le dernier point d’amélioration serait l’entraînement lui-même. Si d’un côté on augmente les données d’entraînement et qu’on entraîne le modèle plus longtemps, le modèle serait en mesure d’être plus robuste. Dans notre cas, avec 50 morceaux nous avons entraîné le modèle presque 30 heures afin d’avoir une loss générale de 0.8. Avec plus de données et plus de temps, les résultats seraient drastiquement différent et bien meilleur.

Comparaisons des résultats/analyses

En ayant différentes techniques, certaines pour montrer que l’approche était cohérente, d’autres pour montrer leur impuissance face à de telles données, on établit une conclusion face à tout ce qui a été fait.

Il est déjà clair que toutes les méthodes statistiques liées de près ou de loin à l’ARIMA ne sont pas assez avancées pour prédire de la musique. Ces méthodes ne prenant en compte que des séries univariées et lisses, la musique n’est pas un bon candidat pour de telles prédictions. On en profite par la même occasion pour confirmer ce caractère propre aux ARIMA et ses dérivées.
Par rapport aux chiffres purs, les méthodes des proches voisins et de la forêt aléatoire semblent celles qui ont les meilleures prédictions, non seulement parce que leurs loss sont assez faibles, mais aussi parce que les prédictions sont en concordances avec le reste de données et garde une certaine cohérence par rapport aux motifs récurrents.

En dehors de ces méthode qui semblent efficaces pour la prédiction des notes, il n’en demeurent pas moins que le réseau de neurone reste une méthode extrêmement puissante, et que même si la prédiction au niveau des notes est assez sommaires par moment, cela garde une cohérence auditive ce qui n’aurait pas forcément été le cas pour les autres méthodes. Si la plupart des méthodes n’utilisaient qu’un morceau pour l‘entraînement, le réseau de neurones en utilisait près de 50 ce qui lui donne une meilleure connaissance générale des morceaux et de la polyphonie. Le réseau de neurones peut généraliser alors que les autres méthodes non.

Conclusion

Pour conclure le travail mené sur l'étude de la prédiction des notes de piano, cela peut se formuler par une certaine supériorité écrasante du réseau de neurones.

Après avoir essayé différentes méthodes statistiques, toutes sans exception ont montré une certaine limite dans leur prédiction et l'entraînement. Le point récurrent est un apprentissage limité des motifs liés à la mélodie, au rythme et la polyphonie. Le réseau de neurones a permis de supprimer ces barrières avec des données d'entraînement plus variées, et une meilleure compréhension des motifs récurrents et des liens entre chaque notes, en plus de nous donner la possibilité de donner du contexte dans l'apprentissage avec chacune des composantes d'une note de piano. Le réseau de neurones nous a aussi permis de prédire plusieurs sorties, plus seulement le pitch d'une note, donnant des prédictions plus variées en fonction du texte, augmentant ainsi ses degrés de liberté.

Bibliographie

https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5
https://arxiv.org/ftp/arxiv/papers/1908/1908.01080.pdf
https://arxiv.org/ftp/arxiv/papers/2203/2203.12105.pdf
https://www.tensorflow.org/tutorials/audio/music_generation?hl=fr
https://blog.floydhub.com/generating-classical-music-with-neural-networks/
https://david-exiga.medium.com/music-generation-using-lstm-neural-networks-44f6780a4c5
https://data-flair.training/blogs/automatic-music-generation-lstm-deep-learning/
https://github.com/jordan-bird/Keras-LSTM-Music-Generator
https://blog.paperspace.com/music-generation-with-lstms/
https://mct-master.github.io/machine-learning/2022/05/20/kriswent-generating-piano-accompaniments-using-fourier-transforms.html