BARRY Alpha
Master 2

H3 HITEMA

Rapport Python DATA

Accident de voitures aux Etats-Unis

H3 Hitema
ISSY

1. Introduction

Nombreux sont les accidents routiers qui arrivent chaque jour dans le monde. Les Etats-Unis n'échappent pas à la règle avec environ 38000 morts et plus de 4 millions de blessé par ces mêmes accidents. Aujourd'hui, l'objectif est de baisser le nombre d'accident de la route. Nous avons déjà pu faire une étude sur la prédiction de la gravité d'un accident et la localisation des zones les plus accidents et maintenant nous souhaiterons avoir de meilleurs résultats. Nous avons désormais 2 objectifs :

- Les méthodes pour analyser les données.
- Quelle est la meilleure méthode pour avoir une précision très accurate.

Répondre à ces objectifs nous permettra d'être beaucoup plus précis et donc efficace dans la prise de décision des mesures nécessaires à prendre afin de diminuer le nombre d'accident.

Nous allons nous intéresser à plusieurs choses :

1. La possibilité de requêter sur notre dataset
2. La manière d'obtenir une meilleure accuracy
3. Une meilleure précision pour le nombre d'accident dans une ville
4. La mise en place d'un cloud.

2. Adaptation du Dataset

Dans un premier temps, on a dû adapter notre Dataset car il contenait beaucoup trop de données (quelques millions de lignes). Le problème était que, faute de la quantité de données, le temps d'exécution était augmenté et donc, les performances en été impacté. Il a donc été décidé de créer une version mini de note Dataset avec une diminution des données à 100000 permettant tout de mêmes d'avoir des résultats satisfaisant malgré la quantité de données réduite.

Dans un second temps, afin de pouvoirs appliquer nos différents algorithmes, une phase de cleaning a dû être réalisée sur nos données. Il y a également eu une réflexion autour du faite de garder certaines colonnes ou non (voir Liste_des_colonnes.docx).

Dans un dernier temps, un traitement des champs vide soit NaN ont été effectué. C'est-à-dire qu'il a été décidé d'enlever toutes les lignes contenant des valeurs NaN afin d'avoir une cohérence entre chaque ligne.

3. Requêter sur notre dataset

Dans un premier temps, nous avons chercher à importer notre dataset (fichier .csv) dans une base de données. En l'occurrence dans notre cas il s'agit de Postgres.

PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence de type BSD :

```
db=# SELECT ID, Severity, description FROM Public."US_Accidents_June20_mini";
    id     | severity |                                  description
-----------+----------+------------------------------------------------------------
--------------------
 ID        | Severity | Description
 A-1901773 | 3        | Accident on I-10 Westbound at Exits 296A 296B I-75.
 A-2923261 | 3        | At Imperial Main St/Exit 186 - Accident. Three lanes blocked.
 A-323310  | 2        | Accident on Brown Rd at Old Peachtree Rd.
 A-820738  | 2        | Accident on Falls Of Neuse Rd Northbound at Rainwood Ln.
 A-2084835 | 2        | Accident on Dockery Rd at VA-903 Goodes Ferry Rd.
 A-1750350 | 3        | Right lane blocked due to accident on I-95 Northbound before Exit 23 FL-818 Griffin Rd.
 A-388961  | 2        | Accident on TX-249 Southbound at FM-1960 Cypress Creek Pkwy.
 A-1273837 | 2        | Accident on 156th Ave at 132nd St.
 A-254833  | 3        | Very slow traffic and left hand shoulder closed due to accident on I-66 Eastbound at Exit 44 VA-234 Byp Prince William Pkwy.
 A-53061   | 2        | Accident on Vernon Ave near Grand Ave.
 A-591016  | 3        | 3 left lane blocked due to accident on I-15 Southbound at CA-56 Ted Williams Pkwy.
 A-3218076 | 2        | At Front St - Accident.
```

Figure 1 : Affichage de features de notre dataset sous forme de table dans Postgres

L'avantage d'avoir notre dataset sous la forme de table dans une base de données et que l'on peut requêter dessus et donc :

- **C**reate/Ajouter des données facilement et rapidement dans notre dataset
- **R**ead/Afficher facilement des informations dans ce même dataset
- **U**pdate/Modifier des informations sans avoir à rechercher les lignes précise qui nous intéressent
- **D**elete/Supprimer une/des lignes qui nous gênes

Vous savez désormais ce qu'est un CRUD.

A noter que l'utilisateur peut retransformer la table et fichier .csv :

```
db=# exit
root@8b0e87e5850e:/usr/src/postgres# cd ..
root@8b0e87e5850e:/usr/src# chmod 777 postgres/
root@8b0e87e5850e:/usr/src# psql -U abarry -d db
psql (13.0 (Debian 13.0-1.pgdg100+1))
Type "help" for help.

db=# COPY Public."US_Accidents_June20_mini" TO '/usr/src/postgres/US_Psql.csv' DELIMITER ',' CSV HEADER;
COPY 200002
db=# exit
root@8b0e87e5850e:/usr/src# ls postgres/
database.env  Dockerfile  README.txt  US_Accidents_June20_mini.csv  US_Psql.csv
root@8b0e87e5850e:/usr/src#
```

Figure 2 : Exportation de la table sous forme de fichier .csv

4. Avoir une meilleure précision

Nous souhaitons savoir dans n'importe quel contexte quelle est le meilleur algorithme et selon l'algorithme la courbe de ROC que cela généré pour le model d'entrainement et de test. Nous avons pour cela utilisé streamlit.

Streamlit est un outil qui nous permet de construire, visualiser et partager efficement des applications composées de données :

US Car Accident

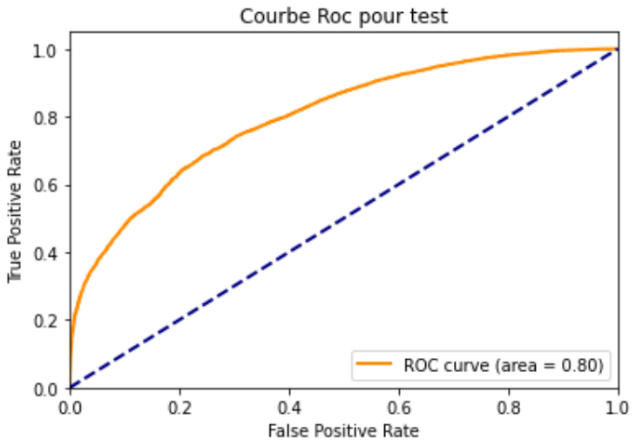
Algortihme de prédiction

Nous voulons savoir quelle algorithme à la meilleurs accuracy:

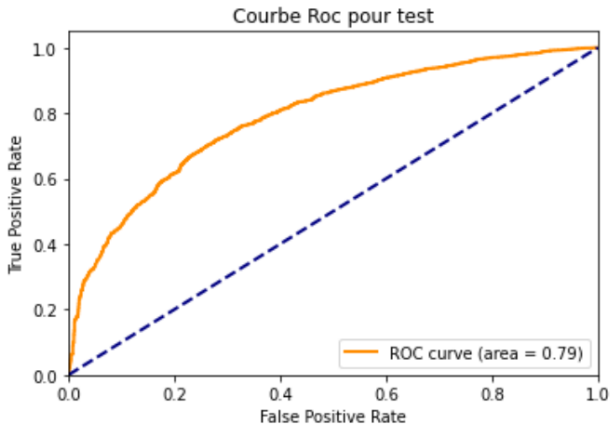
Algorithme:

- KNN
- RandomForest
- GradientBoosting

L'accuracy est de : 92.514104 % pour le train et de 76.927604 % pour le test.



Courbe de roc train



Courbe de roc test

L'avantage de cet outil est qu'il nous permet de voir simplement et rapidement les informations qui nous intéressent.

Comme nous pouvons voir, la figure, nous avons chercher à comparer dans cet exemple les résultats obtenus depuis l'algorithme GradientBoosting.

Comme nous pouvons le voir d'ailleurs, la courbe de ROC pour les données de test et sensiblement proche de la courbe de ROC pour les données d'entrainement pour ce même algorithme.

Avec un résultat de 0.79 et 0.80, nous la courbe de roc est assez éloigné de la courbe aléatoire pour que le nombre de faux positive n'affecte pas trop notre précision.

En revanche, l'accurracy obtenu pour le model d'entrainement et beaucoup plus élevé que celui de test (92% contre 76%). Nous pouvons alors faire 2 hypothèses :

- Nous avons fait de l'overfitting
- Les répartitions des 100000 données sélectionnées n'est pas bonne et donc pas assez varié pour être assez pertinente.

Il serait alors intéressant de tester, soit avec plus de données en test, soit en essayant d'avoir un meilleur repartions de nos 100000 données sur tout notre dataset d'origine.

Figure 3 : Affichage des données via Streamlit

Pour chaque algorithme, nous nous sommes également intéressant au l'accurracy, selon 2 critères :

- L'algorithme
- La répartition des données entre le model d'entrainement et de tests(train size).

L'objectif était de savoir si en fonction du nombre de données pour un algorithme, l'accuracy aller changer de manière significative ou non. Pour réaliser cela, nous avons utilisé FastApi, outil nous permettant de réaliser rapidement une API en python. Nous avons pu dès lors obtenir les résultats suivants (*voir annexe1*) :

Algorithme	Précision à 0.1 de train size(train-test)	Précision à 0.9 de train size (train-test)
k-nearest neighbors	69%-69%	69%-67%
RandomForestClassifier	74%-72%	79%-71%
GradientBoostingClassifier	91%-77%	100%-71%

Figure 4 : Tableau de la précision des prédictions de la gravité (Severity)

Comme nous pouvons voir, plus l'on a de données, plus l'accuracy de notre algorithme en entrainement est grand et inversement. Cependant, l'accuracy du model d'entrainement restera toujours supérieurs jusqu'à atteindre 100% d'accuracy. Renforçant l'idée qu'il y a surement un problème au niveau de l'échantillon du jeu de données choisi.

5. Nombre d'accident selon la ville

Nous nous sommes également aux nombres de d'accident dans une ville :

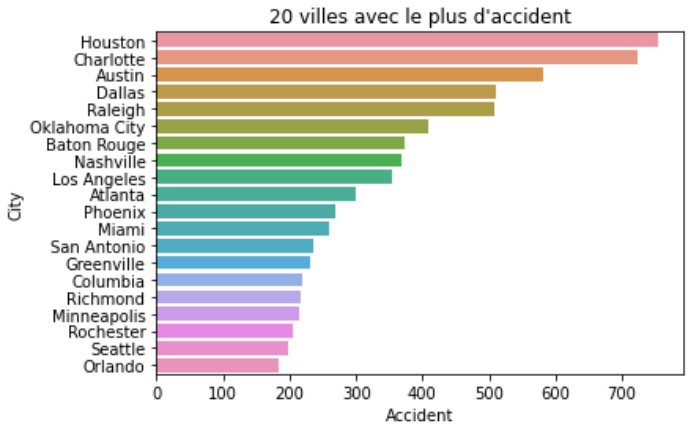


Figure 5 : Histogramme des 20 villes avec le plus d'accidents

Nous pouvons voir dans le diagramme ci-dessus que certaines villes sont beaucoup plus d'accidents que d'autres. En l'occurrence :

- La Houston avec 755 accidents
- Le Charlotte avec 2557 accidents

Ont beaucoup plus d'accidents que la plupart des autres villes.

Nous avons mis en place un système permettant de connaitre le nombre d'accident selon la ville avec le même outil soit **l'API Streamlit**. Cela nous permet d'avoir rapidement et de façon précise ce même nombre plutôt qu'un graphique approximatif :

Figure 6 : Affichage du nombre d'accident dans une ville via l'API Streamlit

6. Mise en place d'un cloud

Afin de permettre à tous les utilisateurs d'accéder à nos résultats, il a été décidé d'ouvrir un cloud via l'outil Heroku. Heroku est une entreprise créant des logiciels pour serveur qui permettent le déploiement d'applications web. Ainsi nous pouvons accéder directement au différents API. Actuellement, il a été mis en place les API suivant :

- FastApi
- Streamlit

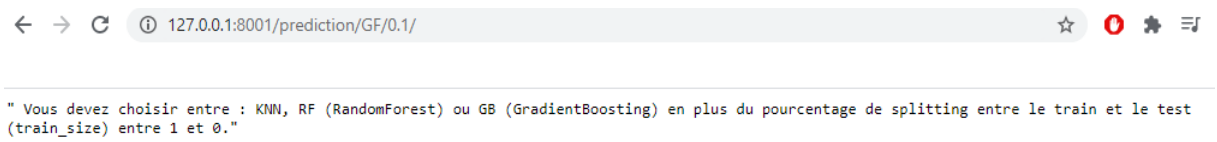
7. Conclusion

En conclusion, nous avons pu voir diverses méthodes pour avoir des informations plus précises sur le nombre d'accident dans une ville. De plus, nous avons pu voir des solutions afin d'obtenir une meilleurs accuracy. Il ne faudra cependant faire attention au jeu de données utilisée et tester nos méthodes sur divers échantillons de notre dataset afin de voir selon l'échantillon si notre accuracy changera ou non.

8. Glossaire

Mot	Definition
k-nearest neighbors	k-nearest neighbors est un algorithme servant à estimer la sortie associée à une nouvelle entrée x, la méthode des k plus proches voisins consiste à prendre en compte (de façon identique) les k échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée x, selon une distance à définir.
RandomForestClassifier	RandomForestClassifier est une méthode d'apprentissage d'ensemble pour la classification, la régression et d'autres tâches qui fonctionnent en construisant une multitude d' arbres de décision au moment de l'apprentissage et en sortant la classe qui est le mode des classes (classification) ou la moyenne / moyenne de la prédiction (régression) des arbres individuels.
GradientBoostingClassifier	Gradient Boosting est un algorithme qui va assembler, unir en un tous des modèles élaborés séquentiellement sur un échantillon d'apprentissage dont les poids des individus sont corrigés au fur et à mesure. En bref, il va construire plusieurs modèles et à chaque nouveau modèle, il va, en cas d'erreur donner un poids supplémentaire à ces erreurs et se corriger.
FastAPI	FastAPI est un framework Web moderne et rapide (hautes performances) pour la création d'API avec Python 3.6+ basées sur des indices de type Python standard.
BSD	La licence BSD (Berkeley Software Distribution License) est une licence libre utilisée pour la distribution de logiciels. Elle permet de réutiliser tout ou une partie du logiciel sans restriction, qu'il soit intégré dans un logiciel libre ou propriétaire.

9. Annexes



Annexe 1 : Résultat de GradientBoosting avec 0.1 en train size sur FastApi

https://www.kaggle.com/sobhanmoosavi/us-accidents

https://www.nsc.org/road-safety/safety-topics/fatality-estimates

https://fr.wikipedia.org/wiki/Licence_BSD

https://fr.wikipedia.org/wiki/Heroku