UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET
INFORMATIQUE APPLIQUÉES

PAR
YANN AMOUSSOU

CLASSIFICATION DES SIGNAUX ELECTROENCEPHALOGRAMME (EEG) POUR DÉTECTER L'ETAT DE FATIGUE : CAS DE LA CONDUITE AUTOMOBILE

AVRIL 2022

SOMMAIRE

L'encéphalographie a été inventé en 1875 par Richard Caton. Dès lors les travaux ont été effectué pour approfondir les connaissances sur le fonctionnement du cerveau. Les avancés majeures sont arrivées en 1920 grâce au neurologue allemand Hans Berger qui réussit à amplifier les signaux électriques de l'activité cérébrale, et à les décrire sous forme de graphe. En 1950 la technique de l'EEG va se développer au niveau que nous l'utilisons actuellement.

Aujourd'hui grâce aux avancées majeures de la science et des technologies de l'information, on peut maintenant faire la capture des imageries cérébrales, et des signaux EEG de plus en plus faible. Ces avancées ont aussi permis d'utiliser l'électroencéphalographie, en médecine, pour détecter de nombreuses maladies comme l'épilepsie, détecter un état de mort cérébrale, les troubles de la conscience, de la vigilance et les troubles du sommeil, etc.

Avec l'avènement de l'intelligence artificielle, le Machine Learning (ML) a permis grâce à ses algorithmes de faire la classification de données diverse. Cette technique fait partie des techniques les plus modernes et efficace, qui utilise les données pour soit prendre une décision ou pour retourner les statistiques et bilan de certaines études. Le Machine Learning est notamment utilisé en électroencéphalographie en prenant comme données; les signaux de l'activité électrique du cerveau (EEG) obtenu numériquement. Grâce à un algorithme spécifique il devient possible de faire la classification et retourner un résultat pouvant permettre de déterminer l'état physiologique d'un individu (ex : état de fatigue, état de faculté affaiblie).

Des recherches sont toujours en cours dans le domaine de l'électroencéphalographie pour pouvoir détecter notamment des états physiologiques chez les êtres vivants. C'est dans ce contexte que s'inscrit l'étude présentée dans ce mémoire. Nous allons faire la classification des données électroencéphalogramme en vue de détecter la fatigue chez un individu qui conduit un véhicule automobile.

ABSTRACT

In this Master thesis we are interested in EEG signals capture, in the classification of these EEG signals in order to detect the state of fatigue in a driving subject. We looked at how the brain works, and how electrical activities of a neuron send messages and control, for example, muscles. The activity of neurons is measurable using the techniques and methods that we have presented. These measurements will be collected in the form of EEG signals, these digitally collected EEG signals will be the basis of the data used. Preprocessing allows us to reframe our signals to remove superfluous information, for example, among the treatments we have frequency framing, to have the frequencies that interest us. The data is then filtered by high pass filters and low pass filters. Phases then follow to remove as much noise as possible from the data, because by obtaining the signals, the movement of the eyelids, and muscles could be captured. The use of the wavelet transforms to extract features and ICA to correct artefacts has been well known and were implemented in our research. The data taken in a state of fatigue and the data taken in a normal state are therefore put together with the creation of a state variable to differentiate them. The data thus prepared, the decision tree was used to make the classification considering the two target variables: the condition of the subject and the position of the vehicle. The results first show that the chosen algorithm has a good prediction rate because of the large amount of data. But with smaller amount of data, other algorithms such as Neural Networks and Support Vector Machine (SVM) shown better results. Our approach showed that regardless of the patient's condition, the position of the vehicle was well predicted. Better results are obtained for fixed vehicle positions than when the subject performs actions. The subject condition classification results obtained in this study are very satisfactory with an efficiency of 94.8 %.

REMERCIEMENT

Mes remerciements vont tout d'abord à DIEU tout puissant qui nous donne chaque jour la force et le souffle de vie de tous les jours, puis à :

M. François Meunier, professeur à l'Université du Québec à Trois-Rivières pour sa confiance et les efforts consentis dans la réalisation de ce mémoire, pour le temps qu'il m'a accordé, ses apports, et corrections.

Tous les professeurs du département de mathématiques et d'informatique, pour les temps d'apprentissage passé ensemble, le partage de connaissance m'a permis d'évoluer.

Tous les professeurs et personnels de l'Université du Québec à Trois-Rivières, pour le support.

Mes camarades de promotion, avec qui j'ai su cultiver un esprit de partage et de famille.

Mes remerciements vont finalement à tous ceux qui ont cru en moi et en mes capacités pour leurs encouragements mais aussi à ceux qui ont douté de moi car ils m'ont permis de me surpasser.

DEDICACES

A ma mère Béline Akpé Kafui AMOUSSOU, aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement que j'ai toujours eu pour toi.

Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1 : INTRODUCTION

1.1 Introduction

La fatigue constitue la principale cause des accidents de la circulation routière. Selon la Société de l'assurance automobile du Québec (SAAQ), une moyenne de 78 décès et 8532 blessés est constaté chaque année à cause de la fatigue au volant. Ces accidents surviennent généralement en milieu d'après-midi ou la nuit, à cause de notre horloge interne, toujours selon la SAAQ. Les études menées aux États-Unis ont montré que la fatigue est la principale cause sur 200 accidents qui ont entrainé la mort de conducteurs de véhicules lourds. Ces chiffres restent énormes, malgré les efforts des autorités pour pousser les conducteurs à ne pas prendre la route en cas de fatigue.

Pour essayer de pallier ces problèmes, nous allons dans ce présent mémoire effectuer des analyses sur des données d'encéphalogrammes (EEG). Ces analyses vont d'abord nous permettre de comprendre comment fonctionne un cerveau humain normal par rapport à un cerveau en situation de fatigue. Et ainsi à travers un modèle de prédiction essayer de détecter la fatigue chez un ind4idu effectuant des tâches comme la conduite d'un véhicule.

Des efforts dans le sens de diminuer la conduite d'un véhicule sous l'effet de la fatigue ont été entrepris. Ainsi on distingue deux catégories de méthodes pour faire face à cette situation. La première reste la sensibilisation, avec des projets publicitaires de plus en plus dissuasifs, la Québec reste parmi les provinces qui investissent le plus au Canada dans les publicités sur la sécurité routière. La seconde catégorie de méthode reste technologique et commence à prendre de l'ampleur. Tous les constructeurs automobiles se lancent dans ce dernier. Ainsi nous disposons des systèmes qui se basent de plus en plus sur les données physiologiques (le mouvement des paupières, le temps de clignement les yeux, etc.). Il existe aussi des systèmes qui se basent sur le comportement de conduite (la tenue de voie du conducteur, les coups de volant, etc.). Comme exemple des constructeurs tels que Mercedes avec son système Attention Assist et Volkswagen ont eu recours selon l'article cité en [3] à un détecteur de fatigue analysant en particulier les mouvements imprimés par le conducteur au volant.

François Meunier et al. [29] ont mis en place aussi un prototype pour automatiser la détection de facultés affaiblies chez les conducteurs qui est basé sur le mouvement des pupilles

Notre étude quant à elle prendra en compte les signaux produits par le cerveau et son comportement en temps de fatigue. Nous allons au cours notre étude faire une revue de littérature en vue de comparer, discuter, des études faites et en cours. Ensuite nous allons recueillir nos données pour implémenter notre méthode expérimentale. Et pour finir nous présenterons nos résultats que nous discuterons.

1.2 Objectif et hypothèses

Dans notre étude nous allons faire l'analyse des signaux électroencéphalogramme (EEG) afin de détecter la fatigue grâce à des algorithmes de classification. Les individus pris en compte sont des individus en situation de conduite automobile. Ainsi il y aura des données sur des individus suffisamment reposé, puis les individus en état de fatigue au volant.

Les données découlent de tests effectués sur 65 sujets, qui sont les étudiants de premier cycle et des cycles supérieurs du NCTU, qui ont été recrutés pour participer à l'expérience. Nous allons détailler, les outils, les méthodes utilisées ainsi que le déroulement de l'expérience.

1.3 Contenu du mémoire

Dans cette partie nous allons exposer les différentes parties du mémoire. La première partie présentée est le chapitre 1. Le chapitre 1 est dédié à l'introduction générale du mémoire. Cette partie est d'abord composée de la description sommaire de la problématique à résoudre dans le mémoire. Les objectifs et les hypothèses posées sont ensuite suxinctement abordés. Pour finir dans ce chapitre nous allons présenter les différentes parties du mémoire. Le chapitre 2 est quant à lui dédié à la revue de littérature qui fait une description sommaire des structures et du fonctionnement du cerveau, des effets de la fatigue sur le cerveau, des effets de la fatigue sur la conduite. Cette revue de littérature permet aussi d'introduire l'électroencéphalographie. Pour clore le chapitre 2, nous allons présenter les articles, œuvres et mémoires qui traitent du sujet du présent mémoire. Dans le chapitre 3 nous allons d'abord présenter les données utilisées pour effectuer la détection de l'état de fatigue d'un conducteur de véhicules. Ensuite, avant de décrire l'algorithme de classification utilisé pour la détection de l'état de fatigue d'un conducteur, nous allons d'abord, présenter les traitements préalables effectués sur les données. Le chapitre 4 sera

consacré à la présentation et à la discussion des résultats, puis pour finir nous présenterons les perspectives de cette recherche dans le chapitre de la conclusion

CHAPITRE 2 : REVUE DE LITTÉRATURE

2.1 Introduction

Nous nous intéresserons dans ce chapitre dans un premier temps, à mettre en place le contexte général du mémoire en présentant le cerveau et les fonctionnements du système nerveux, Ensuite nous allons présenter quelques effets de la fatigue sur le cerveau. Pour terminer ce chapitre, nous parlerons de quelques travaux qui traitent des effets de la fatigue sur les activités cérébrales. De plus, quelques travaux portant sur la détection de l'état de fatigue de conducteurs seront abordés.

2.2 Le cerveau humain

Le cerveau fait partie des principaux organes de l'homme. Selon l'Institut du Cerveau, d'un point de vue philosophique, le cerveau est l'organe qui perçoit, qui pense et qui agit. Scientifiquement, le cerveau est l'organe où se construisent, nos sentiments, nos émotions, notre capacité de réflexion. Le cerveau joue aussi un grand rôle dans nos facultés motrices. Le cerveau pèse environ 1,4 kg, il baigne dans le liquide céphalo-rachidien. Le cerveau est recouvert par les méninges, ce qui lui confère une protection optimum. Le cerveau consomme une bonne partie de l'énergie produite par l'organisme.

Le système nerveux est l'ensemble que forment le cerveau et la moelle épinière. Ce système est capable selon l'institut du cerveau d'intégrer les informations, de contrôler la motricité et d'assurer les fonctions cognitives.

Le cerveau est séparé en deux parties gauche et droite qui sont rattachées par le corps calleux. Chaque partie est composée du lobe frontal, du lobe pariétal, du lobe occipital et du lobe temporal comme le montre la Figure 1.

Le lobe frontal intervient dans le contrôle musculaire, dans le langage, la pensée, la résolution des problèmes, la pensée, la planification.

Le lobe pariétal intervient dans la conscience du corps et de l'espace environnant.

Le lobe occipital intervient dans la vue, dans la reconnaissance des formes et des couleurs.

Le lobe temporal intervient dans l'encodage des informations auditives et dans l'intégration des informations.

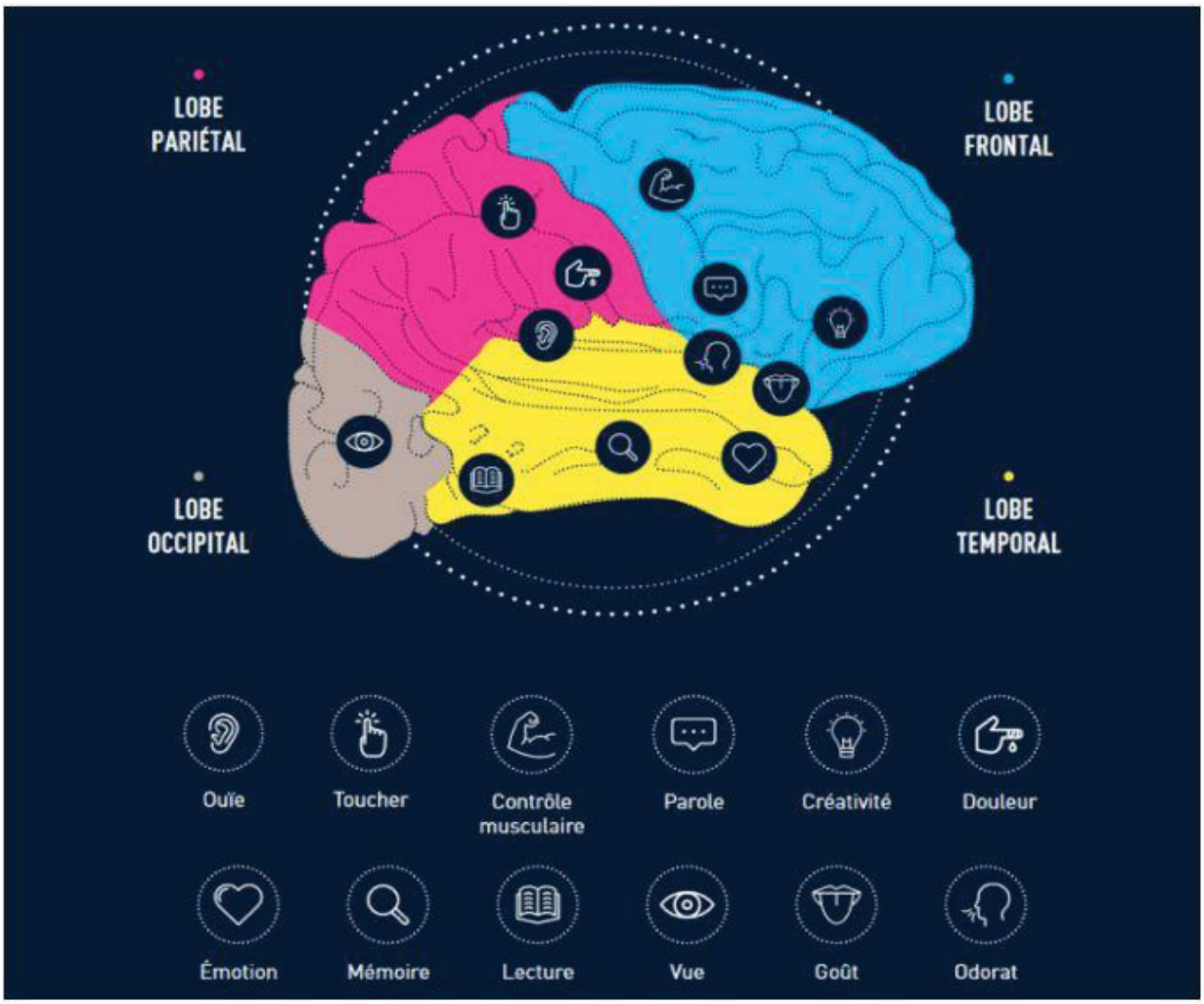


Figure 1 : Représentation des différentes parties du cerveau selon l'institut du cerveau

2.3 Les effets de la fatigue sur le cerveau

Le cerveau a différentes réactions en fonction de l'état du corps. La fatigue du corps aussi fait réagir différemment le corps. Nous allons voir dans cette partie, les effets de la fatigue sur le corps humain et les systèmes qui existent pour détecter la fatigue sur le cerveau. D'après l'article de passeport santé sur le sujet [26], le cerveau étant l'organe principal de commande de notre

corps les symptômes que l'on peut voir sur la conduite ressemble à ces derniers : la maladresse et le manque de coordination. Le sujet fatigué est moins productif, il peut ressentir des sautes d'humeur, de la nervosité. Cela peut conduire à un défaut des sens aussi. Pour éviter les effets de la fatigue, le repos est de mise.

2.4 Les effets de la fatigue sur la conduite

De manière générale la fatigue physique se caractérise par une perte de concentration. En situation de conduite automobile, la fatigue peut mener à la somnolence, et à l'endormissement.

Selon la SAAQ [1] les effets d'une longue période d'éveil sont semblables à la consommation d'alcool. Un conducteur fatigué a du mal à garder les yeux ouverts. Il ressent un sentiment d'inconfort dans le siège du conducteur dû au fait que son corps cherche une position confortable pour se reposer. Il prend plus de temps pour exécuter certaines actions comme freiner tardivement. Il a du mal à maintenir une trajectoire et une vitesse constante. Le conducteur cesse de manière générale de regarder dans les rétroviseurs, ou d'assurer certains gestes de sécurité.

Des systèmes existent aujourd'hui dans les voitures pour pouvoir lutter contre la fatigue au volant, parmi ces systèmes, nous pouvons citer : le SDF (Driver Monitor System) de Toyota qui fonctionne grâce à une caméra à infrarouge qui détecte les mouvements et certains signes de fatigue dans les yeux des conducteurs. Des systèmes des compagnies indépendantes aussi peuvent se retrouver sur le marché : le DADS (Driver Alterness Detection System) développé par SRG International [27]. Au Québec la SAAQ préconise de s'arrêter et se reposer lorsque la fatigue au volant survient.

2.5 L'électroencéphalogramme (EEG)

L'électroencéphalogramme (EEG) est l'activité électrique du cerveau, généralement sous forme de graphe, il est mesuré grâce à un examen qui est l'encéphalographie. Ce graphe transcrit les signaux émis par le cerveau sous forme d'ondes. Il est obtenu grâce à des électrodes placées sur le cuir chevelu. L'EEG renseigne sur l'activité neurophysiologique et permet de donner des diagnostiques neurologiques.

2.5.1 Principe de fonctionnement

Le cerveau fonctionne grâce aux neurones. Les neurones sont des cellules qui transmettent de l'information à d'autres cellules nerveuses. Le cerveau des mammifères contient entre cent millions et cent milliards de neurones en fonction de l'espèce. Un neurone est composé d'un corps cellulaire (le noyau et le cytoplasme), de dendrites et d'un axone (voir Figure 2).

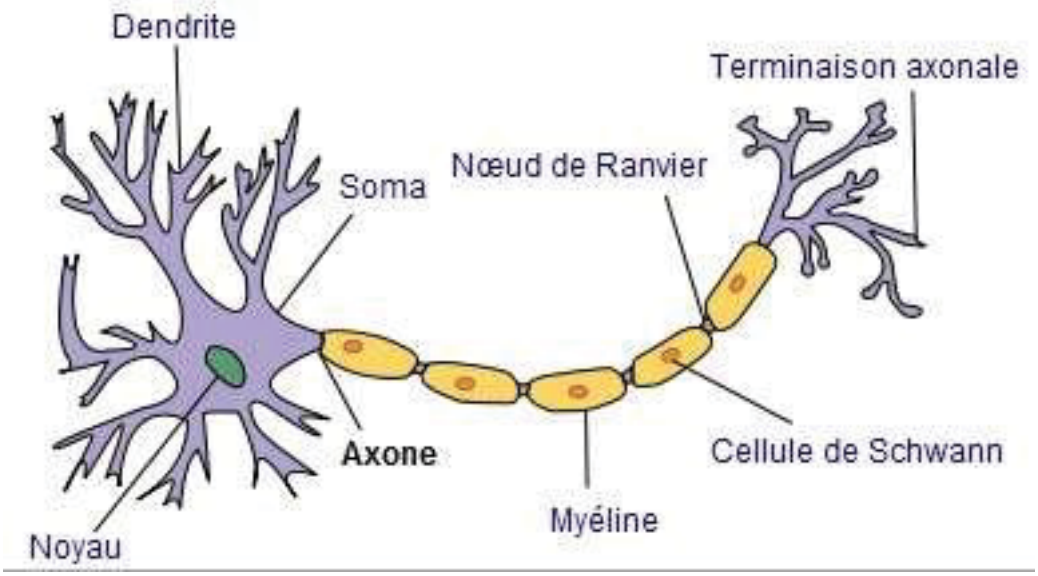


Figure 2 : Structure d'un neurone.

En transmettant les informations (recevoir ou envoyé un message), les neurones transmettent des impulsions électriques, qui sont situé au niveau de l'axone. C'est cette impulsion qui est recueillie lors de l'EEG.

2.5.2 Les signaux EEG

Les signaux EEG sont les signaux résultants de l'électroencéphalographie. L'activité électrique enregistrée par EEG provient en majorité des neurones pyramidaux situés dans les couches 2I, V et IV du cerveau (voir Figure 3).

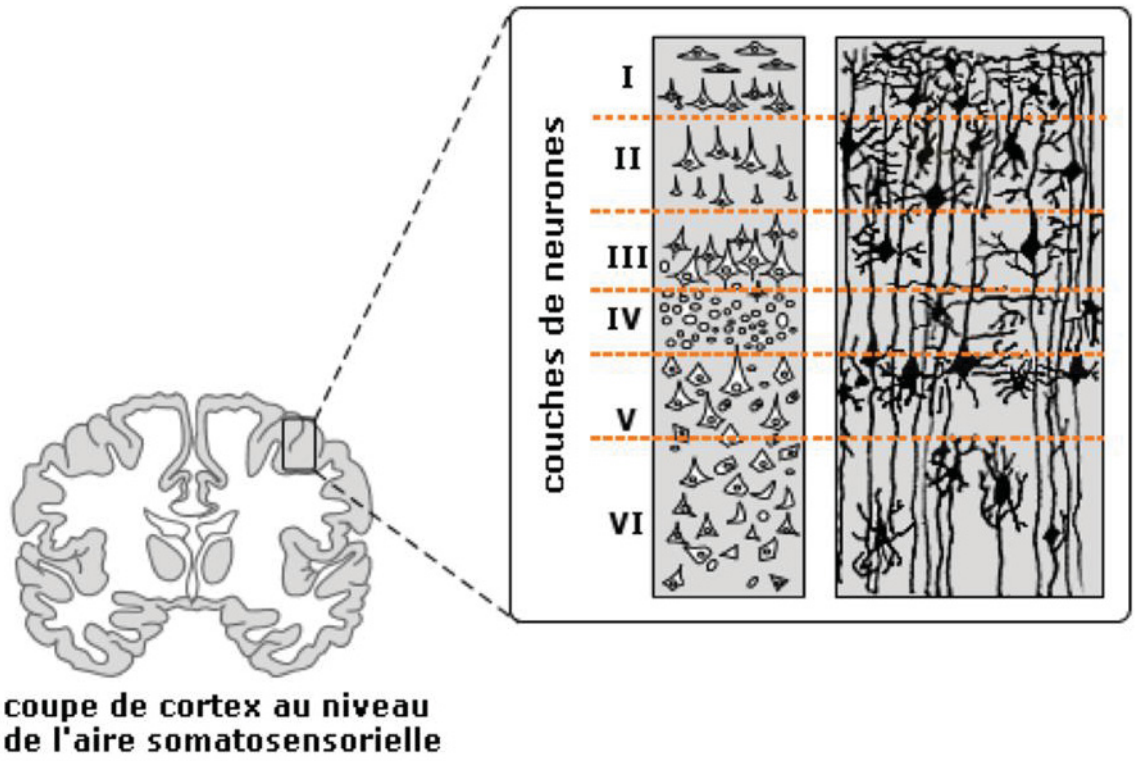


Figure 3 : Les six couches du cortex du cerveau (http://www.maxicours.com/se/fiche/0/2/142702.html).

La synchronisation de l'activité de cellules nerveuses est à l'origine des signaux captés. Le signal est généralement capté grâce à des électrodes placées sur le cuir chevelu. Les électrodes sont généralement incorporées dans un système destiné à capter les signaux électriques du cerveau. Les appareils pour capter les signaux EEG sont généralement de 32 ou 64 électrodes, mais on peut trouver des systèmes de 16 jusqu'à 256 électrodes.

D'ailleurs lors de certaines de nos expériences nous avons testé l'appareil EMOTIV EPOC de la société EMOTIV qui compte 16 électrodes, dont 14 sont destinés à la capture de signaux et 2 électrodes sont des électrodes témoins. Le système est connecté à l'ordinateur grâce à un dispositif Bluetooth et à une application sur l'ordinateur permettant la capture et certaines analyses des signaux EEG. Avant la pose, les électrodes doivent d'abord être imbibées de solution saline, pour ainsi faciliter la conductivité entre les électrodes et le cuir chevelu.

Les données utilisées dans notre recherche proviennent d'une étude menée à l'université National Chiao Tung University [32], ont été capturées à l'aide d'un système à 32 électrodes. Sur les 32 électrodes, il y a 30 électrodes pour capter les signaux et 2 électrodes témoins.

Les systèmes de captures de signaux EEG ne captent pas uniquement les activités électriques EEG, mais ils captent aussi les activités électriques résultantes par exemple des clignements des yeux, du mouvement des muscles. Ces derniers signaux étant considérés comme des artéfacts qu'il faut filtrer avant de faire l'analyse proprement dite des signaux EEG. De par leur grande sensibilité, ces systèmes de capture arrivent à capter différents types de signaux EEG.

2.5.3 Les types de signaux EEG

L'EEG peut mesurer l'activité du cerveau, lorsque le corps humain se trouve dans différentes positions. Ainsi en fonction de ce que l'on veut mesurer, on peut capturer les signaux EEG lorsque le cerveau est au repos, lorsque le cerveau est en activité, ou pour déterminer certaines anomalies comme l'épilepsie et les troubles du sommeil. Il est cependant fortement utilisé pour déterminer les épisodes épileptiques. Dans le cadre du mémoire de Aymen BEN MESSAOUD [19], portant sur l'utilisation des signaux du cerveau (EEG) et vocaux pour la détection et le monitoring des facultés d'une personne, un test de mémoire de type N-Back a été effectué en même temps que la prise des prises des signaux EEG mesurant de l'activité du cerveau. On peut aussi mesurer l'activité électrique du cerveau lors du processus langagier, Tremblay et coll., 2008 [30].

Ainsi en fonction de l'activité du cerveau on peut déterminer différents rythmes cérébraux. Les rythmes peuvent être mesurés même si le cerveau n'est pas stimulé. Les rythmes interviennent notamment dans le cadre de la classification de l'état d'éveil ou de sommeil, de l'état de vigilance ou de somnolence. Ainsi les rythmes sont caractérisés par les ondes du signal EEG. La fréquence de ces ondes est mesurée en Hertz (Hz). De plus une notion de temps intervient, car les signaux sont pris généralement en fonction du temps qui est exprimé en milliseconde.

Nous présentons ici des exemples de rythme cérébraux :

Les rythmes delta sont des ondes lentes, qui interviennent dans les états de méditation profonde ou de sommeil sans rêves.

Les rythmes thêta interviennent dans les états de sommeil profond et dans l'apprentissage et la consolidation en mémoire.

Les rythmes alpha interviennent dans les états d'éveil, de vigilance, de méditation légère. Ils interviennent aussi dans la coordination d'activités mentales et à l'apprentissage.

Les rythmes bêta interviennent dans un état d'éveil surtout lorsque nous réalisons des tâches comme prendre une décision ou résoudre des problèmes.

Les rythmes gamma sont les ondes les plus rapides, qui interviennent dans un état d'attention élevé ou de concentration.

La figure 4 permet de présenter les intervalles de fréquences associés à chacun des différents rythmes. Cette figure (voir Figure 4) permet aussi d'exposer une onde typique pour chaque intervalle de fréquences ainsi que l'état associé.

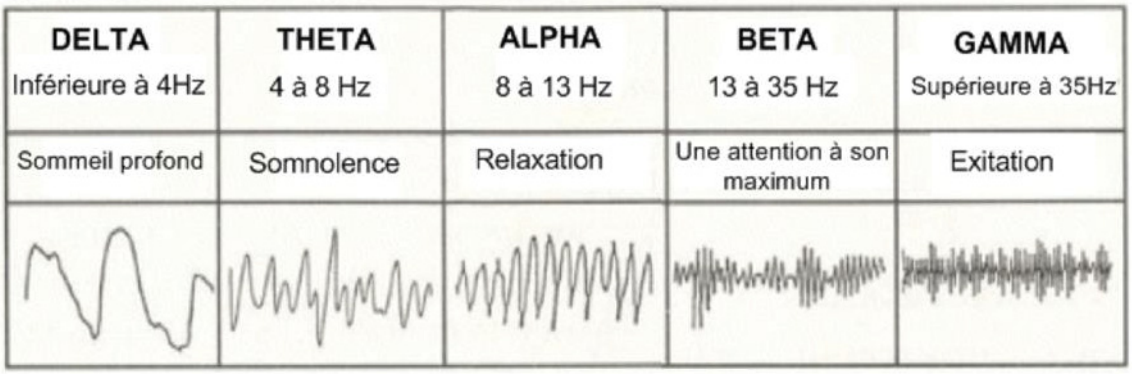


Figure 4 : Les différentes ondes (http://tpe-batement-binauraux.webnode.fr/quest-ce-que-les-battements-binauraux-/)

2.6 La revue de littérature

Cette partie sera consacrée à la présentation des articles, mémoires et œuvres qui traitent du sujet de la détection de somnolence au volant. Le premier article concerné est : Automatic recognition of alertness level by using wavelet transform and artificial neural network qui a été rédigé par Kemal Kiymik, Mehmet Akin, Abdulhamit Subasi [20]. Cet article traite de la reconnaissance automatique du niveau de vigilance au volant à l'aide de la transformée en ondelette et de réseau de neurones artificiels. Le but de cet article est de proposer une méthode utilisant les réseaux de neurones artificiels (voir Figure 5), avec comme données d'entrée un spectre complet d'enregistrements (EEG), en vue de détecter les états de vigilance et de somnolence chez les sujets.

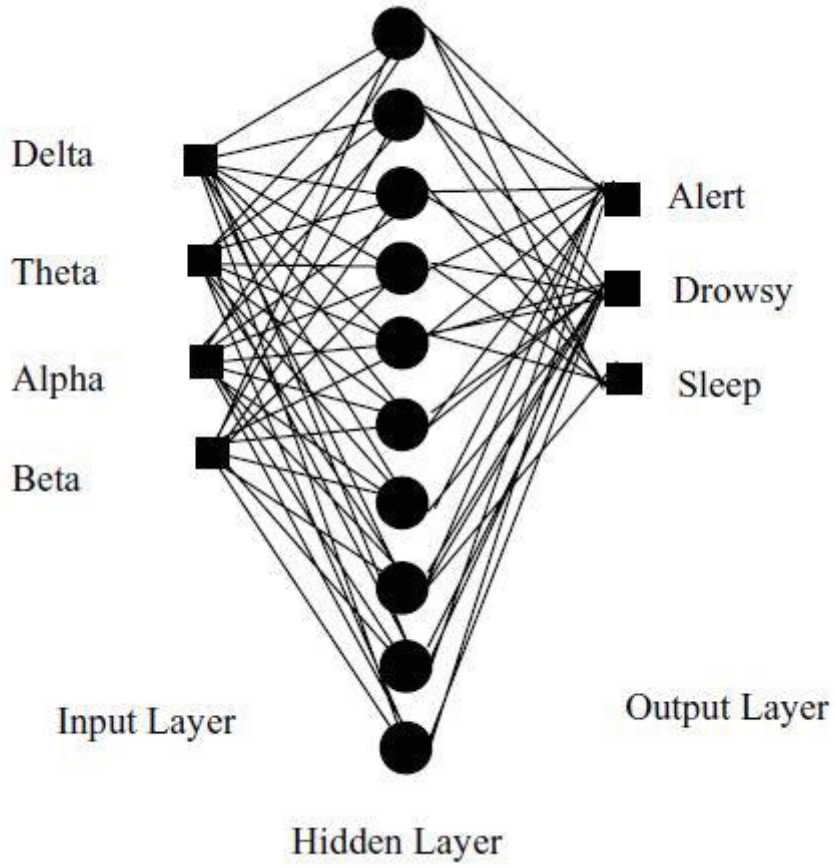


Figure 5 : Réseau de neurones article [20]

Ce système permet d'obtenir trois sorties discrètes : alerte, somnolence et sommeil. La procédure utilise la densité spectrale de puissance (PSD) de la transformée en ondelettes discrète de l'EEG comme entrées du réseau de neurones, chacune des entrées étant le niveau de puissance mesurée pour chaque bande de fréquences. L'étude a été réalisée sur 30 sujets sains, dont 14 femmes et 16 hommes âgés de 18 à 65 ans. L'algorithme de reconnaissance permettant de déduire le niveau de vigilance a été testé à l'aide des signaux EEG enregistrés de 12 sujets sains. Les résultats obtenus sont les suivants pour la précision du réseau de neurones artificiels est de 96% pour d'état d'alerte, 95% d'état de somnolence et 94% soit d'état de sommeil. La classification nous donne 3% d'état d'alerte, 4% d'état de somnolence et 5% d'état de sommeil sur le total. La conclusion générale tirée de cette étude est que les résultats suggèrent que cet algorithme de reconnaissance automatique est applicable pour distinguer les états : alerte, somnolence et sommeil.

Le second article étudié, est un article de la conférence internationale sur les réseaux de neurones qui s'est tenu du 6 au 11 Juillet 2014 à Pekin en Chine. Le titre de l'article est : EEG-based

Driving Fatigue Prediction System Using Functionnal-link-based Fuzzy Neural Network. Cet article est rédigé par Yu-Ting Liu, Yang-Yin Lin, Shang-Lin Wu des étudiants de National Chiao-Tung Univesity à Hsinchu en Taiwan et Chun-Hsiang Chuang, Mukesh Prasad, Chin-Teng Lin [21]. Cet article expose de façon générale un système de prédiction de la fatigue au volant basé sur l'EEG utilisant un réseau de neurones flous basé sur les liens fonctionnels (voir Figure 6).

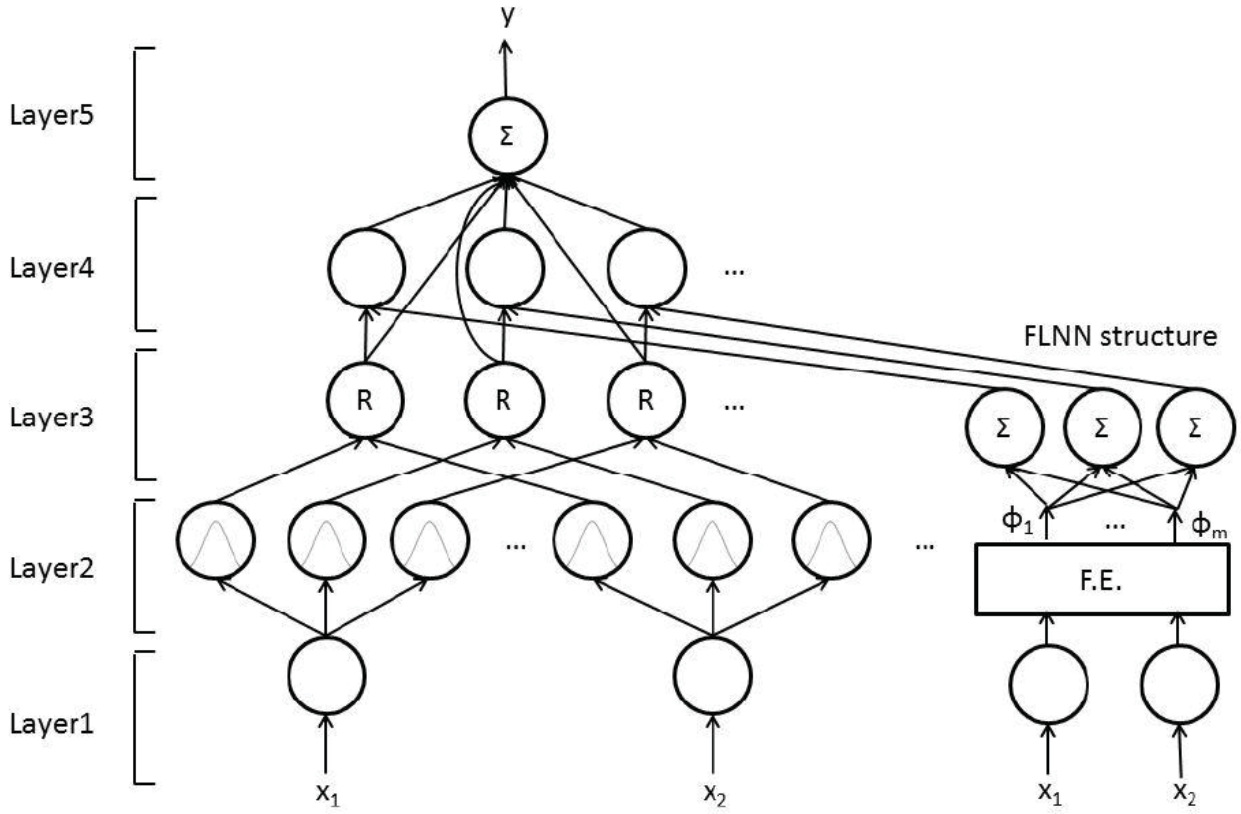


Figure 6 : Algorithme utilisé dans l'article [21].

Avant de faire la classification une scène tridimensionnelle (3D) a d'abord été développée dans un environnement de réalité virtuelle pour représenter des situations de conduite les plus réels possibles. L'étude a été faite sur dix jeunes adultes droitiers en bonne santé avec un âge moyen de 24,2 ans. Les signaux EEG obtenus ont été séparés en différents rythmes cérébraux. Ces différents rythmes cérébraux ont été utilisés comme données d'entrées dans le réseau de neurones flous basé sur les liens fonctionnels. L'algorithme a été subdivisé en deux parties. Une première partie (Fuzzy Logic Neural Network) qui fait l'apprentissage de la structure et qui a pour données

d'entrée les variables qui sont introduites pour construire les règles floues. La deuxième (partie gauche dans la Figure 6) est consacrée à l'apprentissage des paramètres qui est basé sur l'apprentissage par l'algorithme de rétropropagation. Tous les paramètres libres ont été mis à jour par les modèles itérativement. Le lien fonctionnel vers la couche de sortie a été utilisé pour augmenter la flexibilité. Le système des performances FLFNN est comparé à la régression linéaire. Les résultats de cette étude montrent en général l'efficacité du modèle FLFNN proposé.

Pour traiter du sujet, l'article A cognitive and neurophysiological test of change from an individual's baseline publié en janvier 2011 dans Clin Neurophysiol nous parle d'un test cognitif et neurophysiologique du changement par rapport à la ligne de base d'un individu. Les auteurs sont : Alan Gevins, Michael E. Smith, Linda K. McEvoy, Aaron B. Ilan, Cynthia S. Chan, An Jiang, Lita Sam-Vargas, et Gordon Abraham, tous de San Francisco Brain Research Institute & SAM Technology à San Francisco en Californie [22]. L'objectif de cet article est de présenter un test neurophysiologique cognitif automatisé qui caractérise comment un individu a été affecté par un médicament ou un traitement. Le test calcule des sous-scores découlant des performances des tâches de mémoire de travail, l'activation corticale et la vigilance, et combine ces sous-scores en un score global. Le test a été effectué sur 16 individus en bonne santé. Les signaux EEG ont été enregistrés pendant l'exécution de la tâche et dans les conditions de repos. Les signaux ont été échantillonnés à 256 Hz et sont passés dans un filtre passe-bande numérique de 0,1 à 35 Hz. Un test de n-back à différent niveau de difficulté a été proposé sur les sujets. Les artéfacts obtenus ont été décontaminés grâce à des algorithmes de quatrième génération. Des analyses combinées des performances EEG et des tâches cognitives ont été effectuées et le score global a été calculé comme la moyenne de trois sous-scores. Le test a été appliqué dans une étude à double insu contrôlée par placebo portant sur l'alcool, la caféine, et la privation de sommeil. Les résultats indiquent que tous les traitements ont affecté le score global tandis que les effets différentiels sur les sous-scores ont mis en évidence la valeur ajoutée des mesures EEG.

L'article dont le titre est Monitoring Driver's Alertness Based on the Driving Performance Estimation and the EEG Power Spectrum Analysis et les auteurs sont : S. F. Liang, C. T. Lin, R. C. Wu, Y. C. Chen, T. Y. Huang, et T. P. Jung publié en 2005 parle de la surveillance de la vigilance d'un conducteur en fonction de l'estimation des performances de conduite et de l'analyse du spectre de puissance EEG [23]. Le but de cet article est de proposé des méthodes,

pour des estimations de la somnolence qui combine l'électroencéphalogramme (EEG) par l'analyse de spectre de puissance de sous-bande spectrale. Pour ce faire une étude a été effectuée sur 16 sujets âgés de 20 à 40 ans. Les études ont été réalisées dans un environnement de réalité virtuelle basées sur des expériences de conduite sur autoroute. Les données collectées ont subi une suppression de bruit, en vue d'enlever les interférences. Ensuite une analyse spectrale à moyenne mobile est effectuée pour obtenir le spectre de puissance des signaux EEG. Une analyse de corrélation a ensuite été effectuée.

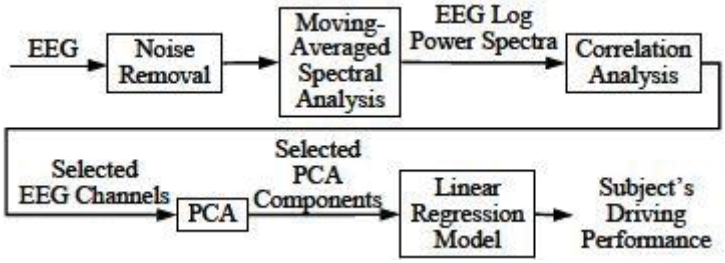


Figure 7 : Méthodologie implémentée dans l'article [22].

Les canaux EEG significatifs sont par la suite sélectionnés. Une analyse en composantes principales est effectuée sur les canaux EEG non corrélés et ce en vue de sélectionner les composants des signaux comportant un maximum d'information. Les composants sélectionnés sont passés au modèle de régression linéaire pour déterminer les performances du sujet (figure 7). Les résultats montrent qu'il est possible de surveiller quantitativement la vigilance du conducteur avec des changements simultanés dans les performances de conduite dans un simulateur de conduite réaliste.

Les auteurs Shyh-Yueh Cheng et Hong-Te Hsu ont parlé de la mesure de la fatigue mentale en utilisant l'EEG, le titre de l'article est Mental Fatigue Measurement Using EEG. Les objectifs de cette étude sont entre autres d'explorer le niveau d'excitation et la fonction cognitive lors de l'exécution de diverses tâches dans un état de fatigue à l'aide des mesures électroencéphalographiques [24]. La figure 8 montre la forme d'onde moyenne de l'ERP. Colonne la plus à gauche : 16 ERP à essai unique. Deuxième colonne à partir de la gauche : ERP moyens calculés sur 4 essais (forme d'onde supérieure de chaque paire) et une estimation du bruit résiduel (forme d'onde inférieure de chaque paire). Deuxième colonne à partir de la droite : ERP moyen calculé sur 16 essais (forme d'onde supérieure) et bruit résiduel (faible forme d'onde). Colonne la plus à droite : ERP moyen calculé sur 64 essais et bruit résiduel.

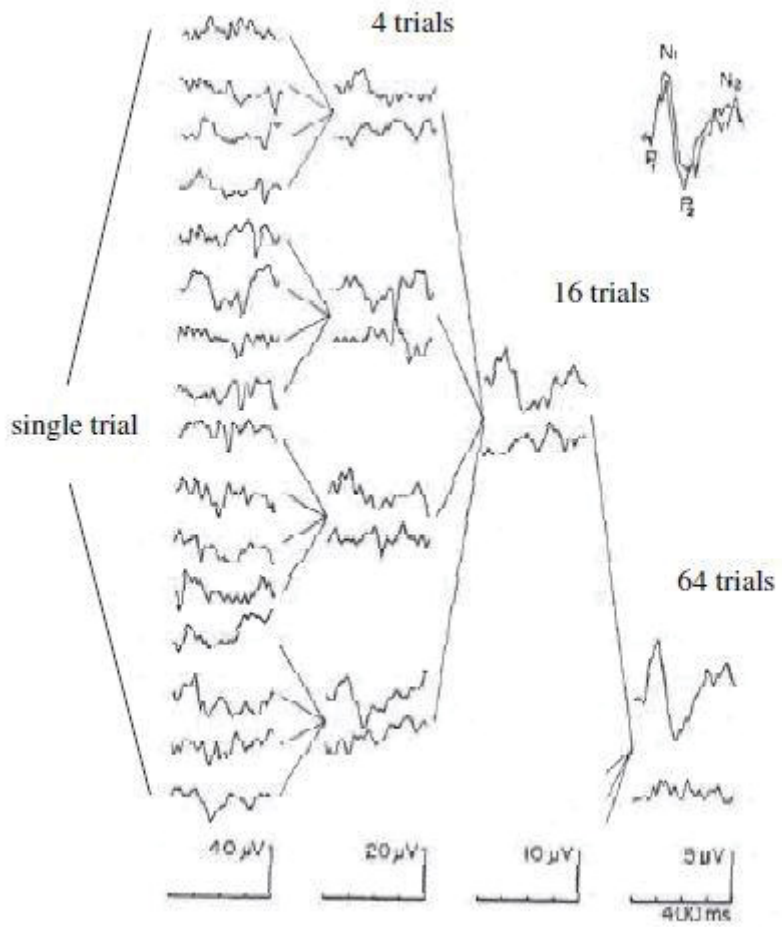


Figure 8 : La moyenne de l'ERP en forme d'onde [24].

Ensuite comparer la réponse comportementale et la réponse physiologique (signaux EEG) à la fatigue mentale lors de l'exécution de certaines tâches sur un écran. Et pour finir, examiner l'état de récupération de la fatigue mentale après 180 minutes de tâches expérimentales avec 60 minutes de repos. L'étude a été réalisée sur 23 étudiants de sexe masculin d'un âge moyen de 22 ans. Les participants sont soumis à des tâches de programmation, pendant ce temps des captures des signaux EEG ont été effectuées. Les données ont ensuite été filtrées et après segmentées en des intervalles de 200 à 800 millisecondes, un autre filtrage de type bande passante est ensuite effectué sur les segments puis une analyse de variance séparé. La méthode utilisée est décrite comme une succession d'étape où on prend des enregistrements à laquelle on applique la moyenne de l'ERP comme sur la figure 8 et les échelle de notation NASA-TLX pour évaluer. En conclusion la méthode proposée est potentiellement applicable à l'évaluation de l'état de la

fatigue des travailleurs et à la gestion de la fatigue sous l'angle de la gestion des risques professionnels.

2.7 Les algorithmes de Machine Learning

Le Machine Learning (ML) est une composante importante de l'intelligence artificielle. Le Machine Learning permet de mettre en place des systèmes qui apprennent ou améliorent les performances en fonction des données (exemples) qu'ils traitent [25]. Il existe deux approches d'apprentissage en machine learning. Nous distinguons donc l'apprentissage supervisé dont les algorithmes sont les plus utilisés. Dans ce genre d'apprentissage, on se base sur des données qui existent déjà pour enseigner à l'algorithme les conclusions (étiquettes) qu'il doit tirer. Pour ce type d'apprentissage nous pouvons donner les exemples d'algorithme comme : la régression linéaire et logistique, les machines à vecteurs de support.

L'autre type d'apprentissage est l'apprentissage non supervisé, les algorithmes dans ce cas-ci fonctionnent de manière indépendante, l'ordinateur construit des schémas complexes et des processus sans guide. Il est basé sur des données sans étiquette, ni résultat spécifique défini [25]. Comme exemples de ce type d'algorithme nous pouvons citer : la mise en cluster de k-moyennes, l'analyse de composantes principales et indépendantes, et les règles d'associations.

Nous présentons ici un autre algorithme de machine learning qui est l'arbre de décision. Un arbre de décision est un algorithme d'apprentissage supervisé. Un arbre de décision est un schéma représentant les résultats possibles d'une série de choix interconnecté [15]. Le début est un nœud qui va se séparer au fur et à mesure qu'on se déplace dans l'arbre de nœud en nœud. Il existe trois types de nœuds : les nœuds de hasard représenté par un cercle, des nœuds de décision représenté par un carré et les nœuds terminaux. Les cercles montrent les probabilités de certains résultats. Les carrés illustrent des décisions, et le résultat final, un schéma de décision typique est présenté à la figure 9 [15].

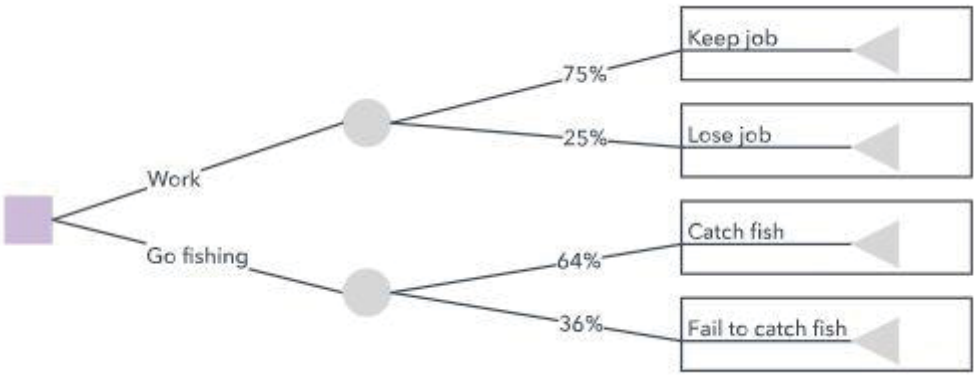


Figure 9 : Exemple d'un arbre de décision [15]

2.8 Conclusion

Dans ce chapitre, nous avons exposé les effets de la fatigue sur le corps humain, et sur le cerveau. Nous avons analysé le fonctionnement du cerveau humain et les principales cellules qui permettent au cerveau de fonctionner. Cette analyse nous amène à comprendre comment capturer l'activité du cerveau ainsi que les outils permettant de faire la capture des signaux EEG. A travers des articles scientifiques nous avons vu comment les études ont déjà traitées les signaux capturés. Et pour terminer nous avons introduit le Machine Learning et d'un de ces algorithmes qui est l'arbre de décision. Dans le prochain chapitre, la méthodologie utilisée dans notre étude est décrite

CHAPITRE 3 : MÉTHODE EXPÉRIMENTALE

3.1 Introduction

Dans ce chapitre nous allons parler de la méthode utilisée pour faire la détection automatique du niveau de vigilance d'un conducteur, mais d'abord nous allons présenter les données capturées. Ensuite, les dispositifs matériels et outils logiciels utilisés pour faire l'analyse des signaux capturées (données EEG) seront aussi présentés. Les étapes de traitement des données extraites du processus de capture de signaux, dont le prétraitement des données effectués dans Matlab ainsi que l'analyse en composantes indépendantes (ICA) seront introduites. Une description des étapes d'extraction des caractéristiques et de l'algorithme de classification pour la détection du niveau de vigilance d'un conducteur sera ensuite abordée.

3.2 Base de données

3.1.1 Expérience et évènements

L'étude menée [31] est effectuée auprès de 65 sujets, qui sont les étudiants de premier cycle et des cycles supérieurs du NCTU. Ces étudiants ont été recrutés pour participer à l'expérience (tâche de conduite avec une attention soutenue de 90 minutes). Tous les sujets devaient avoir un permis de conduire. Aucun des participants n'avait des antécédents de troubles psychologiques. Avant l'expérience, tous les participants ont rempli un formulaire de consentement indiquant leur compréhension claire du protocole expérimental qui avait été approuvé par l'Institutional Review Broad de l'hôpital général des vétérans de Taipei, Taiwan.

Le paradigme expérimental développé autour d'un environnement virtuel, simulait une conduite nocturne sur une autoroute à quatre voies, et le sujet a été invité à maintenir la voiture en marche au centre de la voie. Les événements de sortie de voie ont été induits au hasard pour faire dériver la voiture de la voie de croisière d'origine vers les côtés gauche ou droit (début de la déviation). Chaque participant a reçu pour instruction de compenser rapidement cette perturbation en braquant le volant (déclenchement de la réponse) de façon à faire revenir la voiture dans la voie de croisière d'origine (décalage de la réponse). Pour éviter les impacts d'autres facteurs pendant la

tâche, les participants n'ont réagi à l'événement de perturbation de voie qu'en tournant le volant, et ils n'ont pas eu à contrôler les pédales d'accélérateur et de frein dans cette expérience. Un essai complet, comprenant la période de référence, le début de la déviation, le début de la réponse et le décalage de la réponse, est illustré sur la figure 10 (A). Les signaux EEG ont été enregistrés simultanément (figure 10 (B)). L'essai illustré dans la figure 10, a lieu dans un intervalle de 5 ~ 10 s après la fin de l'essai au cours duquel le sujet doit retourner sur la ligne médiane de la troisième voie de circulation. Si un sujet s'endormait pendant l'expérience, il n'y avait aucun retour pour l'alerter.

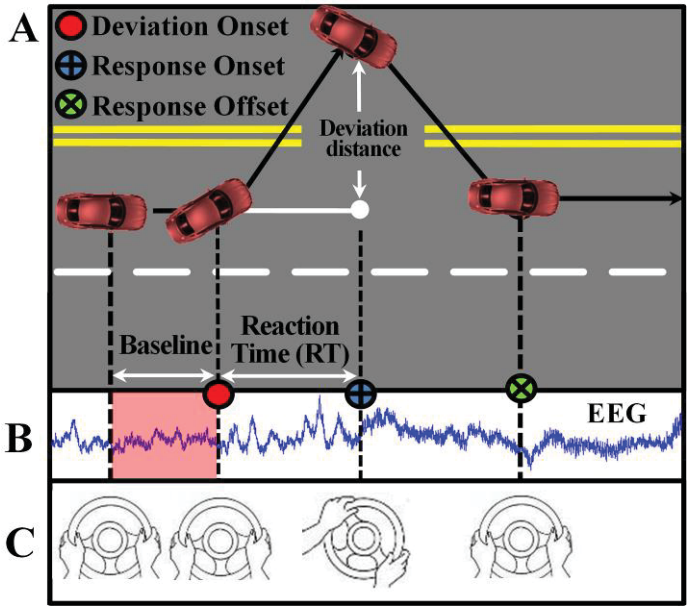


Figure 10 : (A) Paradigme de déviation de voie lié à l'événement. (B-C) EEG et le comportement ont été enregistrés simultanément.

3.1.2 Les signaux

Les signaux EEG ont été obtenus en utilisant le système Scan SynAmps2 Express (Compumedics Ltd., VIC, Australie). Les signaux EEG enregistrés ont été collectés avec un capuchon EEG câblé avec 32 électrodes Ag / AgCl, y compris 30 électrodes EEG et 2 électrodes de référence (mastoïdes latéraux opposés). Les électrodes EEG ont été placées selon un système international 10-20 modifié. L'impédance de contact entre toutes les électrodes et la peau a été maintenue <5 kΩ.

Les fichiers nommés avec les suffixes .set contiennent tous les signaux. Les enregistrements EEG amplifiés par le système Scan SynAmps2 Express (Compumedics Ltd., VIC, Australie) ont été numérisés à 500 Hz (résolution: 16 bits).

Dans l'ensemble, les données EEG comprennent 32 données de canal EEG et une donnée de position latérale du véhicule.

Les 32 premières données sont FP1, FP2, F7, F3, FZ, F4, F8, FT7, FC3, FCZ, FC4, FT8, T3, C3, CZ, C4, T4, TP7, CP3, CPZ, CP4, TP8, A1, T5, P3, PZ, P4, T6, A2, O1, OZ, O2.

La 33ième donnée est la position du véhicule, qui est utilisée pour décrire la position du véhicule simulé.

3.1.3 Annotations sur données

Les événements de l'expérience sont: Début de la déviation, Début de la réponse, Décalage de la réponse (voir EEG.event.type du Tableau 1).

Tableau 1 : Glossaire de la base de données

EEG.event.type	Explications
251	Début de la déviation (à gauche)
252	Début de la déviation (à droite)
253	Début de la réponse
254	Fin de la réponse

3.3 Traitement des données

3.2.1 Pré-traitement des données

À partir des données découlant de l'étude [32], nous avons dans notre étude effectuée des traitements et analyses sur ces données grâce aux à l'outil EEGLAB incorporé dans MATLAB. Ces divers traitements vont nous permettre de tout d'abord de faire un prétraitement des données électrophysiologiques, l'analyse des potentiels évoqués, l'analyse de l'activité oscillatoire avec des analyses Temps-Fréquence et pour finir une première classification utilisant les composantes

ICA (analyse en composantes indépendantes). Mais tout d'abord nous allons nous intéresser à MATLAB et sa fonctionnalité EEGLAB.

3.2.1.1 MATLAB et EEGLAB

MATLAB est un langage de script émulé par un environnement de développement qui s'appelle aussi MATLAB [5]. MATLAB est essentiellement utilisé pour effectuer des calculs et les numériser. Ainsi il permet d'une manière globale de faire des calculs, de développer des algorithmes, de modéliser, de simuler, d'analyser, et d'explorer les données. MATLAB permet aussi de produire des graphes scientifiques et même de développer des applications [6]. MATLAB permet notamment d'incorporer des composants qui vont permettre d'élargir son champ d'application. EEGLAB fait partie des composants que MATLAB intègre.

EEGLAB est un composant de MATLAB pour traiter les données électroencéphalogramme (EEG), magnéto encéphalogramme (MEG) et d'autres signaux électrophysiologiques [7]. EEGLAB incorpore une analyse en composantes indépendantes (ICA) pour la classification. EEGLAB permet de prendre en entrée des données physiologiques sous différentes formes en vue d'un prétraitement et d'une analyse subséquente ICA. Après l'application des prétraitements, les composants ICA déduits par l'analyse en composantes indépendantes peuvent être récupérés afin de leur appliquer à leur tour, d'autres traitements. Les différentes fonctions d'EEGLAB utilisées dans notre étude pour traiter nos données vont être introduites dans la prochaine section de ce mémoire.

3.2.1.2 Pré-traitement avec EEGLAB

a- Visualisation des données

Les traitements appliqués aux données commencent par l'importation des données dans EEGLAB. De plus, ces données étant déjà dans un format compatible, elles peuvent alors être importées directement, nous permettant d'avoir accès aux informations de bases sur les données.

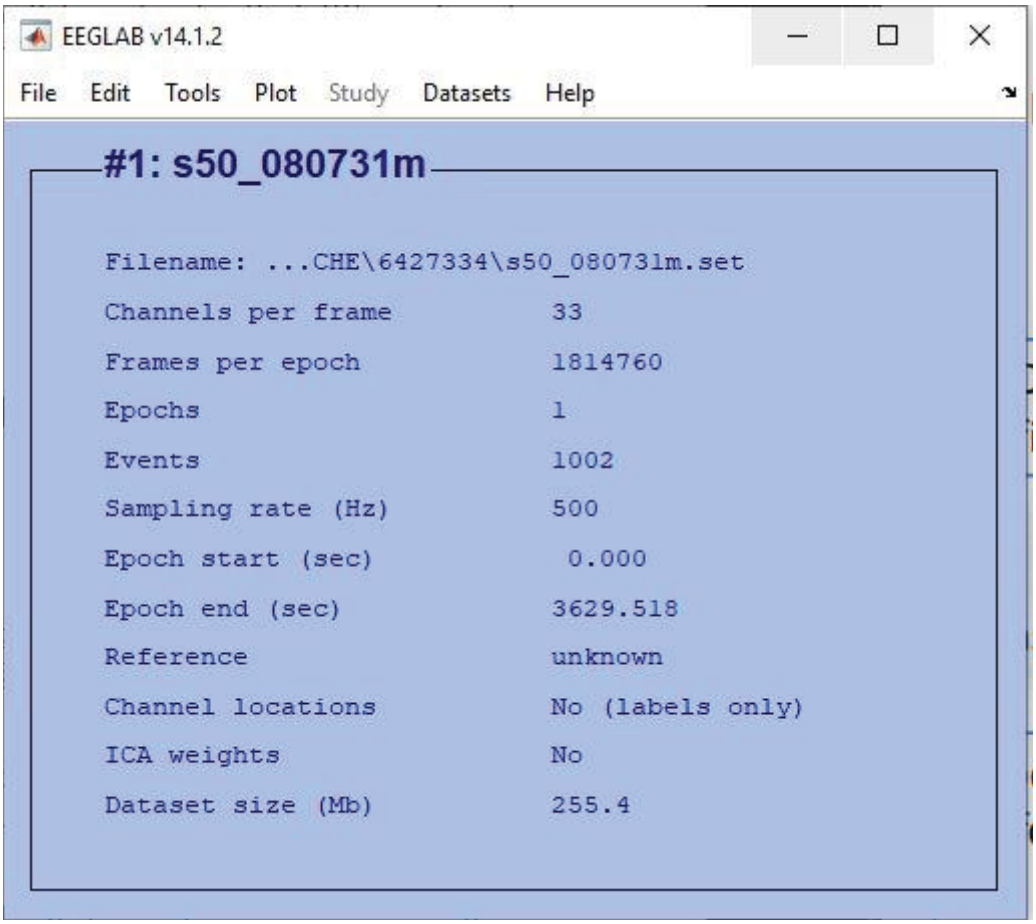


Figure 11 : Fenêtre de chargement de jeux de données dans EEGLAB.

Le nom du jeu de données chargé est s50_080731m (voir Figure 11), nous avons alors des informations comme le lien vers le jeu de données, le nombre de canaux qui représentent le nombre d'électrodes qui font la capture des signaux EEG, dans notre cas il s'agit de 32 (canaux 1 à 32) et le dernier canal (33) est utilisé pour enregistrer la position latérale du véhicule sur la route. Ce fichier de données comporte aussi le nombre d'événements (ex : sorties de route) lors de l'enregistrement. Le fichier emmagasine aussi le temps de l'enregistrement en seconde qui sera très utile dans notre étude étant alors un de nos principaux facteurs. Le taux d'échantillonnage correspond à la fréquence d'échantillonnage des divers signaux.

La visualisation des données s'effectue dans un graphe qui prend en paramètre le temps et les valeurs des divers canaux (voir Figure 12). L'enregistrement pour chaque canal est fait pendant une durée de temps donné. Les signaux affichés sont centrés afin que ce signal soit symétrique et centré par rapport à 0. Pour ce faire, les signaux sont modifiés de façon à enlever la composante

stationnaire, aussi appelé décalage CC. Nous pouvons aussi observer les évènements dans la visualisation. Dans le cas de la figure 12, nous pouvons observer la ligne verte qui indique l'évènement 252 (Début de la déviation (à droite)), puis après l'évènement 253 (Début de la réponse) suivi par l'événement 254 (Fin de la réponse).

Le décalage CC est un déplacement d'amplitude moyen à partir de zéro. Si nous prenons par exemple un signal à 10 Hz si nous enlevons le décalage CC le centre revient à 0 donc : Signal = $\sin(2\pi ft)$ avec $f = 10$ et $0 \leq t \leq 1$.

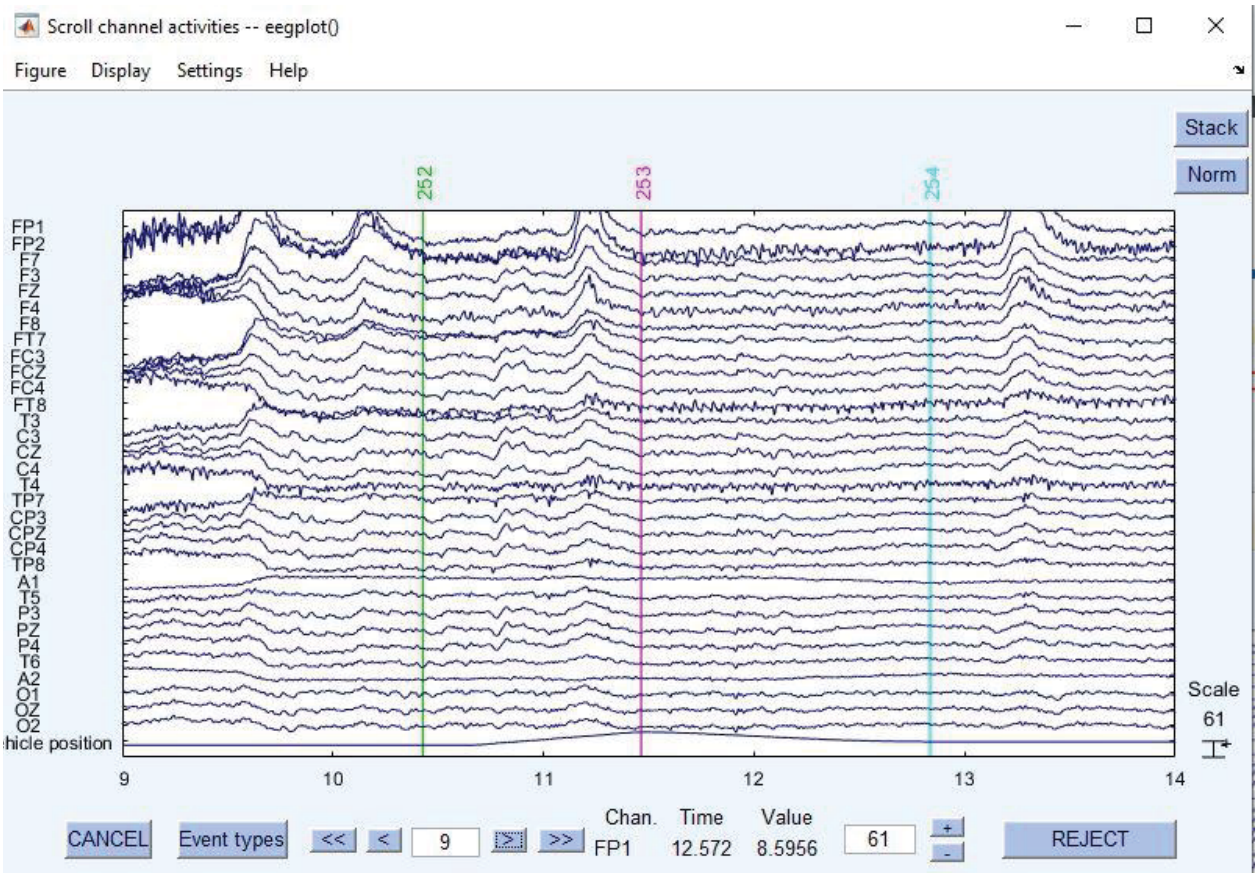


Figure 12 : Visualisation du jeu de données.

À partir des signaux bruts, une détection des artéfacts est ensuite effectuée afin de les éliminer. Selon le dictionnaire, un artéfact ou un bruit est un phénomène d'origine humaine, artificielle, intervenant dans l'étude des faits naturels. On distingue donc deux types d'artéfacts : les artéfacts physiologiques et les artéfacts non-physiologiques. Les artéfacts physiologiques proviennent du muscle (EMG), du cœur (ECG), des mouvements oculaires (VEOG et HEOG). Dans certaines situations les ondes EEG de type alpha sont considérées comme des bruits. Les artéfacts non

physiologiques sont généralement de plus grandes amplitudes pouvant dépasser des fréquences de 50 Hz. Les mouvements au niveau des poses d'électrodes causent généralement ce genre d'artéfacts.

Pour aller plus loin dans l'élimination des bruits, nous pouvons analyser le spectre en fréquence des signaux capturés par les électrodes pour explorer les données et enlever les mauvaises données [9]. L'analyse se fait en donnant un intervalle de temps à analyser en millisecondes, en définissant les fréquences à montrer dans la carte topologique (la carte représentant le cerveau) et enfin définir des fréquences à analyser (dans notre étude il s'agit des ondes Bêta qui caractérisent l'éveil et l'activité cognitive et qui oscillent entre 15 et 30 Hz). Cette analyse peut s'effectuer dans EEGLAB avec la fonction pop_spectopo(). Dans la figure 13 nous pouvons observer la densité spectrale de la puissance des signaux EEG capturés par les électrodes durant un intervalle de temps donné pour une bande spectrale de 15 à 30 Hz.

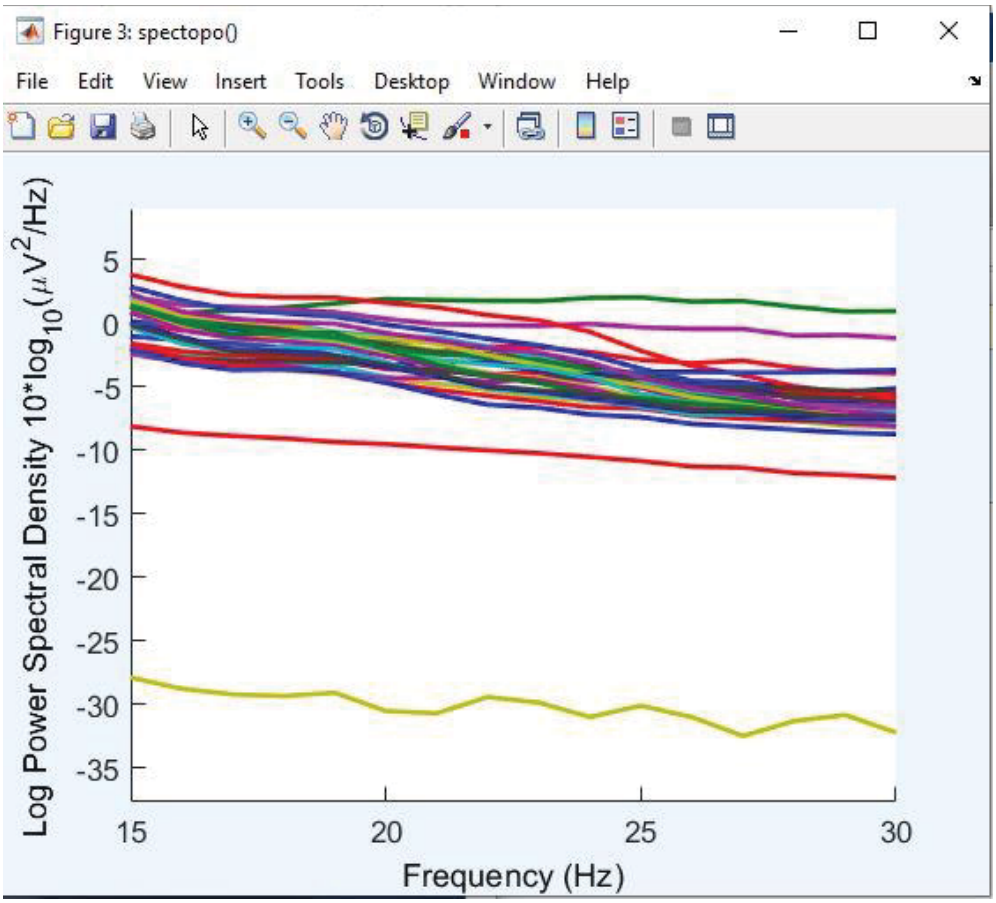


Figure 13 : Spectopo() sur les données avec une fréquence de 15 à 30 HZ

b- Échantillonnage

Selon l'aide de Google Analytics [10] dans le domaine de l'analyse de données l'échantillonnage consiste à s'intéresser à un sous-ensemble des données afin d'identifier des informations significatives qui concernent globalement l'ensemble de données. Dans notre étude le choix adéquat du taux d'échantillonnage va nous permettre de bien reproduire les signaux EEG. Ce taux d'échantillonnage est basé sur le concept de fréquence d'échantillonnage des signaux EEG. C'est le nombre de fois par seconde qu'un signal analogique est échantillonné [9]. Le choix de l'échantillonnage se fait donc selon le théorème de Nyquist-Shannon.

Le théorême d'échantillonnage, dit aussi théorême de Shannon ou de Nyquist-Shannon énonce que l'échantillonnage d'un signal exige un nombre d'échantillons par unité de temps supérieur au double de l'écart entre les fréquences minimale et maximale qu'il contient [11]. Une séquence (X_n) avec n qui est un entier provient d'un échantillonnage d'un signal continu $X(t)$:

$$X_n = X(nT_e)$$

T_e est la période d'échantillonnage, $F_e = \frac{1}{T_e}$ est la fréquence d'échantillonnage. Le signal échantillonné est :

$$X_d(t) = \sum_{n=-\infty}^{+\infty} X(nT_e)\delta(t - nT_e)$$

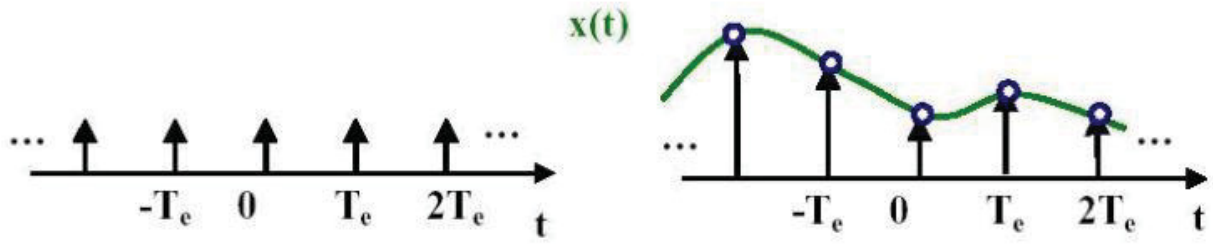


Figure 14 : Représentation de l'échantillonnage [12].

Le théorême de Nyquist-Shannon stipule alors qu'un signal pourra être reconstruit totalement si et seulement si $F_e/2 \geq F_{max}$, F_{max} étant la fréquence maximale.

c- Filtrage des données

Le filtrage est le procédé qui va nous permettre d'enlever les composantes superflues des signaux capturés nous permettant de conserver que l'information dont nous avons besoin. Dans notre étude le filtrage va nous permettre d'enlever les ondes qui ne sont pas pertinentes au processus de détection de la vigilance au volant. Le filtrage va nous permettre d'enlever aussi une partie du bruit. La phase de filtrage nous permettra donc de juste laisser passer les ondes Bêta qui sont d'intérêt dans notre étude (15 – 30 Hz).

Il existe plusieurs types de filtrage selon le type de filtres choisis, ainsi nous avons : le filtre Passe-bas qui laisse passer les basses fréquences, le filtre Passe-haut qui laisse passer les hautes fréquences, le filtre coupe-bande (ou passe-bande) qui est une combinaison du filtre Passe-bas et du filtre passe-haut (voir Figure 15). Nous allons utiliser le filtre passe-bande qui laisse passer la bande de fréquences voulues dans l'intervalle 15 – 30 Hz.

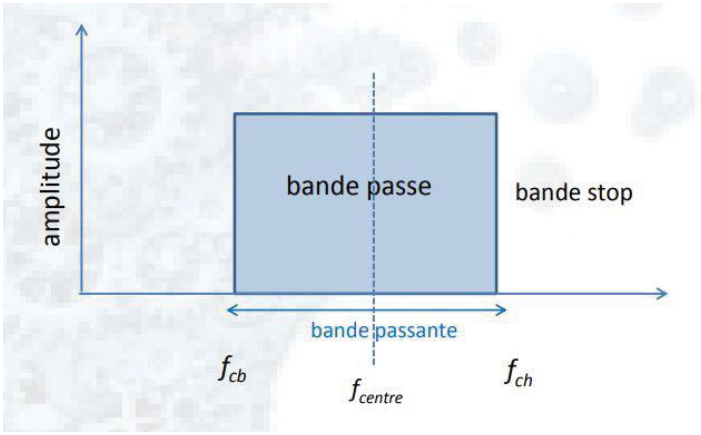


Figure 15 : Représentation fréquentielle d'une passe-bande [9].

d- Prétraitement des données

Le prétraitement des données va notamment consister à repérer les mauvaises électrodes (ex : mal fixées sur le cuir chevelu). Celles-ci étant rejetées, une interpolation des signaux EEG capturés des électrodes les plus proches est effectuée pour reconstruire les signaux de ces électrodes fautives. Ensuite, la bande de fréquences 15-30 HZ est segmentée des signaux bruts de chaque électrode et une analyse en composantes indépendantes (ICA) est effectuée pour éliminer les artéfacts.

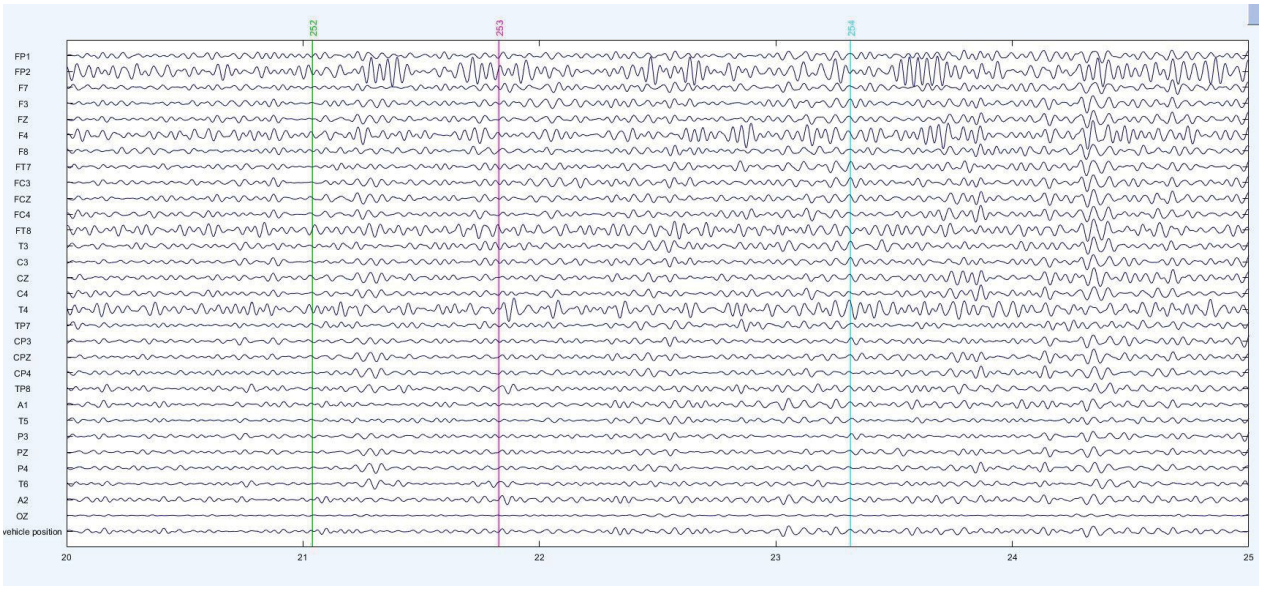


Figure 16 : Électroencéphalogramme après filtrage.

Un parcours visuel sur l'électroencéphalogramme après le filtrage (voir Figure 16) nous montre que le signal de l'électrode Oz est presque constant, ce que le spectre de fréquences de cette électrode vient confirmer. Cette électrode sera alors considérée comme une mauvaise électrode, et sera enlevée.

La segmentation des données va se faire lors de l'occurrence des évènements. Lorsque nous prenons l'évènement 251 (Début de la déviation à gauche) ou 252 (Début de la déviation à droite), qui prend en compte la déviation du véhicule, nous allons segmenter les signaux EEG de telle sorte que la segmentation comprenne le début de la réponse ainsi que la fin de la réponse. Donc, une partie des signaux seront extraits avant l'occurrence de l'évènement appelé pré-stimulus, et une seconde partie après la réponse appelé post-stimulus. L'intervalle pré-stimulus sert comme référence (base line) [9]. Les signaux découlant de cette activité post-stimulus sont normalisés par rapport à cette référence (base line) [9]. La partie post-stimulus commence à partir de l'événement 251 ou 252 jusqu'à la fin de la réponse. La partie pré-stimulus correspond à 500 millisecondes avant l'événement 251 ou 252. La figure 17 schématise la segmentation des données chaque partie jaune et blanche représente un segment. Après la segmentation nous avons sélectionné les meilleurs segments, puis nous avons rejeté les segments bruités.

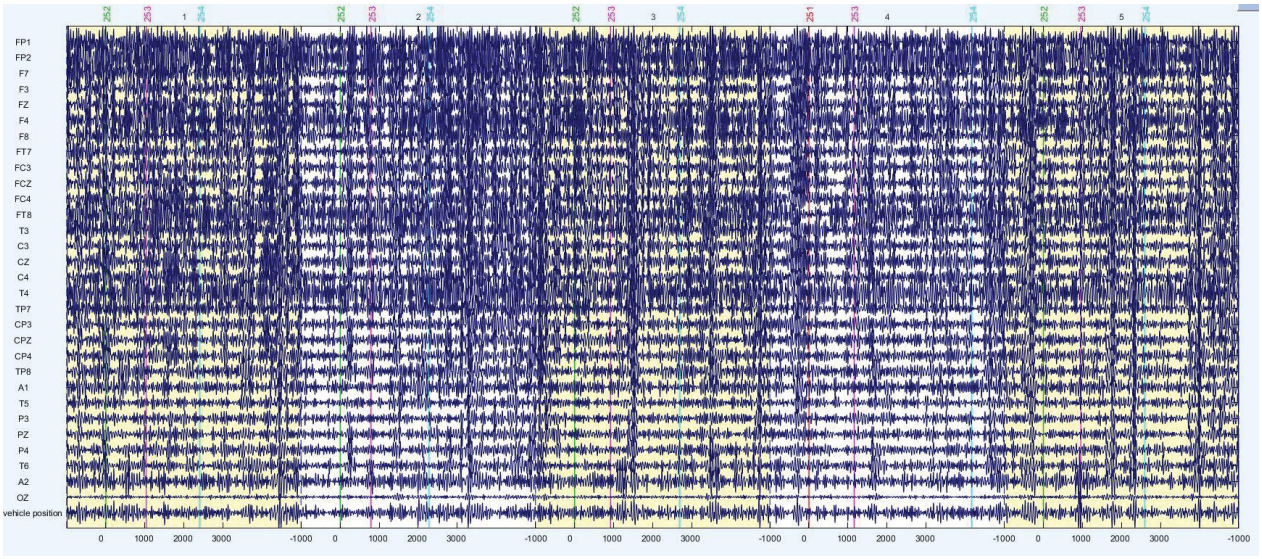


Figure 17 : Segmentation des données.

Les signaux segmentés sont ensuite corrigés pour en éliminer les artéfacts. Cette phase est basée sur une ICA.

e- Correction des artéfacts avec l'Independant Component Analysis (ICA)

Selon [13], l'ICA permet de séparer et de supprimer une grande variété d'artéfacts des données EEG en utilisant la méthode de la décomposition linéaire. L'ICA est basée sur l'hypothèse selon laquelle chaque série enregistrée avec des mélanges spatialement stables des activités de sources cérébrales et artéfactuelles sont temporellement indépendantes. Le délai entre la source et les électrodes sont considérés comme négligeables.

3.4 Extraction des caractéristiques

Dans notre étude, nous allons utiliser la méthode de la transformation en ondelettes (morlet wavelet transform) : cette méthode a été utilisé dans le mémoire de Berdaa Souhail, classification des signaux EEG pour la détection de facultés affaiblies dû à la consommation d'alcool et de drogue en 2019 [6]. Cette méthode permet d'extraire les caractéristiques dans le domaine des fréquence temps. Cette méthode nous permet de voir en fonction du temps, les rythmes cérébraux utilisés. Parmi les méthodes d'extraction des caractéristiques, la transformation en ondelette est efficace car elle subit moins de perte d'informations que les autres méthodes. Entre les divers avantages de la transformée en ondelette nous pouvons citer qu'elle a une taille de fenêtrage

variable. Le fenêtrage est réduit pour les grandes fréquences et est grand pour les petites fréquences. Il existe différents types de transformation en ondelette, dans notre étude la transformée en ondelette de Gabor a été utilisée.

3.5 Classification

Avant de pouvoir effectuer la classification automatique de l'état de vigilance d'un conducteur, nous devons d'abord exporter les données extraites de chaque événement répertorié dans un fichier et afin d'y ajouter une étiquette correspondant à la caractéristique identifiant l'état de fatigue d'un individu. Cette caractéristique (étiquette) sera l'une des caractéristiques cibles de la classification. Outre cette caractéristique nous devons aussi ajouter la position latérale du véhicule. La classification va donc se faire en fonction des deux caractéristiques. Nous utiliserons le langage de programmation python pour réaliser notre classification automatique.

L'algorithme de classification utilisé est l'arbre de décision, les paramètres utilisés pour définir un arbre de décision sont les suivants [28]:

- Critères : indice de diversité Gini et l'entropie, par défaut le critère gini est sélectionné. L'indice Gini et l'entropie sont des critères qui calculent le gain d'information associés à la division d'un nœud dans un arbre de décision. Les algorithmes d'arbre de décision utilisent le gain d'information pour orienter le processus de division des nœuds. Les indices gini et l'entropie sont considérées comme des mesures de l'impureté d'un nœud. Un nœud associé à plusieurs classes est considéré impure alors qu'un nœud associé à une seule classe, donc non divisé, sera alors considéré comme pure. L'entropie en statistiques est l'analogue de l'entropie en thermodynamique et exprime la notion de désordre. Dans ce contexte, un nœud associé à plusieurs classes sera considéré en désordre.
- Diviseur : caractéristique pouvant être fixée soit meilleur ou aléatoire. Par défaut, un diviseur dit meilleur est sélectionné. Ce paramètre permet de choisir une stratégie pour déterminer la division de chaque nœud. La sélection Meilleur permet de choisir la meilleure répartition. La sélection Aléatoire permet quant à elle de choisir la meilleure répartition aléatoire.

- Max_depth est un paramètre entier qui par défaut est sans valeur fixée à priori, et exprime la profondeur maximale de l'arbre de décision.
- Min_samples_split qui est un entier dont la valeur par défaut est fixée à 2. Ce paramètre exprime le nombre minimum d'échantillons requis pour diviser un nœud interne.
- Min_samples_leaf qui est un entier dont la valeur par défaut est 1 et qui exprime le minimum d'échantillons requis pour être considéré comme un nœud feuille.
- Min_weight_fraction_leaf qui permet de donner une valeur à la fraction pondérée minimale de la somme totale des poids. Par défaut ce paramètre est fixé à 0.0 et les échantillons auront donc le même poids.
- Max_features représente le nombre de caractéristiques à prendre en compte lors de la recherche du meilleur partage (split).
- Random_state contrôle le caractère aléatoire de l'estimateur.
- Max_leaf_nodes permet de définir le nombre maximal de nœuds feuilles.
- Min_impurity_decrease : Scinde un nœud si l'impureté peut être diminuée.
- Class_weight : Permet de définir le poids des classes.
- Ccp_alpha est un paramètre de complexité.

Dans nos travaux nous avons utilisé une méthode tirée de la librairie Python sklearn dont l'entête est la suivante : *sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0).* Ce classificateur paramétré comme suit a été utilisé pour les trois séries d'entrainements (en prenant pour cible la position de la voiture, l'état du sujet et la combinaison des deux).

Le schéma représentatif de l'arbre de décision développé dans le contexte de notre recherche est présenté dans la Figure 18.

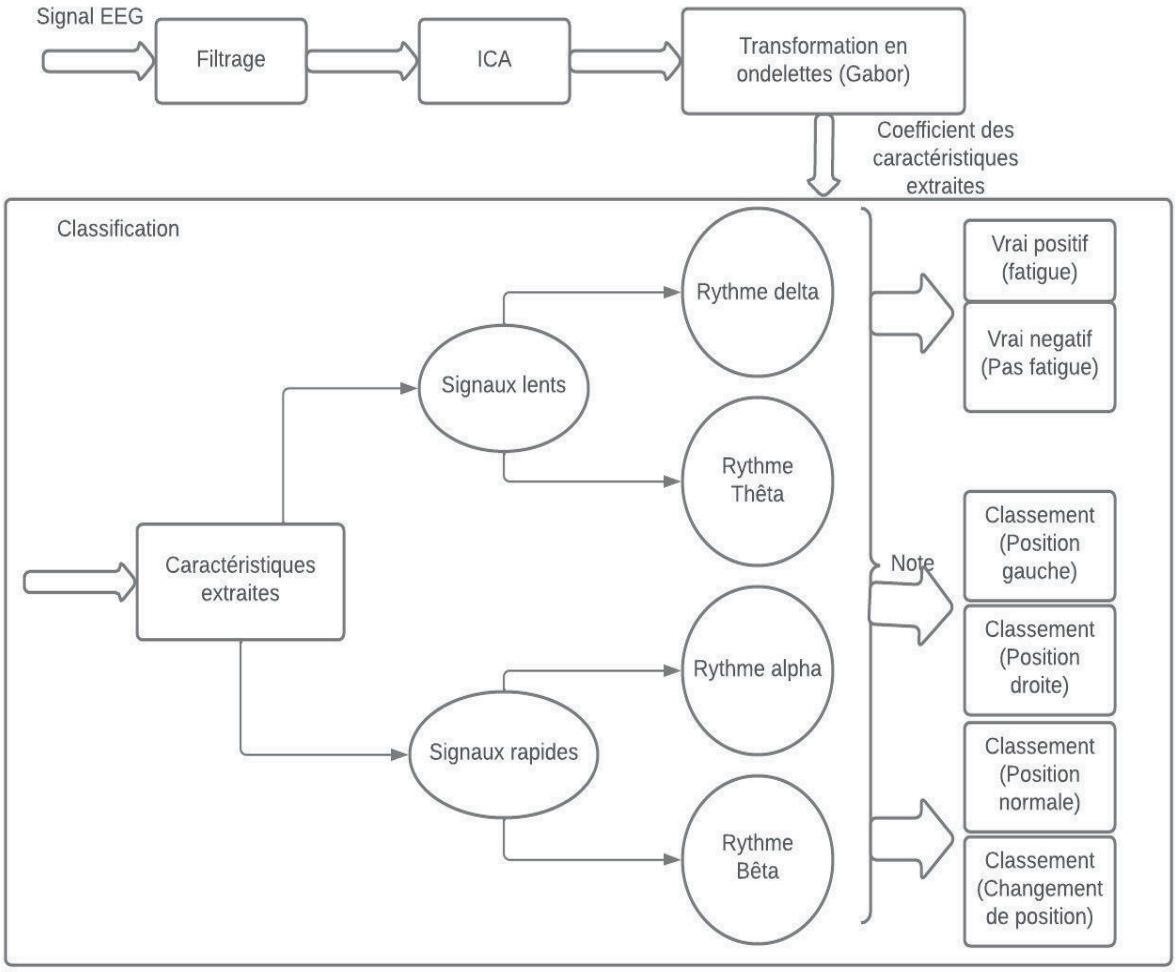


Figure 18 : Illustration de l'arbre de décision utilisé.

Avant de commencer tout processus de classification, nous devons d'abord charger les bibliothèques nécessaires à la classification. Ces bibliothèques sont essentiellement des bibliothèques de machine Learning. Les données sont chargées sous une forme telle que représentée dans la figure 19. Le tableau contient le poids issu de l'ICA pour chaque signal correspondant à une électrode donnée. Les données de l'état sont de 0, si l'état est normal et 1 si le sujet est fatigué. Quant à la position du véhicule 157 est la position normale du véhicule, au-delà de 181 la voiture est extrêmement à droite et en dessous de 100 la voiture est extrêmement à gauche.

```
In [2]: df = pd.read_csv('donnee1t.csv', sep=';')

In [3]: df.head()

Out[3]:
```

FP1	FP2	F7	F3	FZ	F4	F8	FT7	FC3	...	P3	PZ	P4	T6	A2	O1	OZ	O2	vehicle position	etat
2.8861	8.4701	-14.3051	-6.0232	-1.4431	-6.3997	5.5840	-14.9325	-9.3485	...	-9.4740	-2.8234	1.7568	-15.6227	4.4547	3.8272	0.6902	11.1680	157.0	0
2.2587	7.8427	-14.8070	-5.8350	-0.7529	-6.3369	6.0859	-14.7443	-9.0348	...	-8.9721	-1.5058	2.3214	-14.4306	4.0155	5.9605	2.5097	13.8032	157.0	0
2.1960	7.6545	-14.9953	-5.3330	-0.1882	-6.0232	6.3997	-14.3679	-8.9093	...	-8.4701	-0.8784	2.9489	-13.2385	3.0743	7.1526	3.5763	15.6227	157.0	0
2.3842	7.9055	-15.0580	-4.0782	0.3137	-5.1448	7.0271	-14.1796	-8.0937	...	-7.7172	-0.3137	3.9527	-11.9837	2.4469	7.8427	4.4547	16.6893	157.0	0
2.1960	8.4074	-15.2462	-1.5685	0.8156	-4.0782	8.0309	-14.6188	-5.9605	...	-6.9643	0.3137	5.0193	-11.1053	2.1960	8.2819	5.1448	16.8148	157.0	0

columns

Figure 19 : Présentation des données après chargement

3.5.1 Classification en prenant comme variable cible la position du véhicule

Tout au long des enregistrements des données, le véhicule peut occuper plusieurs positions latérales sur la route et ce dans le but de pouvoir aussi considérer la position latérale du véhicule comme variable de sortie. Dans ce cas, il devient possible de pouvoir prédire la position du véhicule en se basant sur les signaux EEG traités. Les enregistrements de la position du véhicule sont des nombres entiers, il convient donc de les regrouper pour une précision accrue. Le découpage a été fait selon l'algorithme présenté à la Figure 20.

```
In [15]: vehicle_pos = df['vehicle position'].values
vehicle_pos_discretized = []
new_vehicle_pos = []
vehicle_pos_counts = [0] * 10
ranges = ['≤ 100', '101-110', '111-120', '121-130', '131-140', '141-150', '151-160', '161-170', '171-180', '≥ 181']
for val in vehicle_pos:
    if val <= 100:
        new_vehicle_pos.append('≤ 100')
        vehicle_pos_counts[0] += 1
        vehicle_pos_discretized.append('≤ 100')
    elif 101 <= val <= 110:
        new_vehicle_pos.append('101-110')
        vehicle_pos_counts[1] += 1
        vehicle_pos_discretized.append('101-110')
    elif 111 <= val <= 120:
        new_vehicle_pos.append('111-120')
        vehicle_pos_counts[2] += 1
        vehicle_pos_discretized.append('111-120')
    elif 121 <= val <= 130:
        new_vehicle_pos.append('121-130')
        vehicle_pos_counts[3] += 1
        vehicle_pos_discretized.append('121-130')
    elif 131 <= val <= 140:
        new_vehicle_pos.append('131-140')
        vehicle_pos_counts[4] += 1
        vehicle_pos_discretized.append('131-140')
    elif 141 <= val <= 150:
        new_vehicle_pos.append('141-150')
        vehicle_pos_counts[5] += 1
        vehicle_pos_discretized.append('141-150')
    elif 151 <= val <= 160:
        new_vehicle_pos.append('151-160')
        vehicle_pos_counts[6] += 1
        vehicle_pos_discretized.append('151-160')
    elif 161 <= val <= 170:
        new_vehicle_pos.append('161-170')
        vehicle_pos_counts[7] += 1
        vehicle_pos_discretized.append('161-170')
    elif 171 <= val <= 180:
        new_vehicle_pos.append('171-180')
        vehicle_pos_counts[8] += 1
        vehicle_pos_discretized.append('171-180')
    elif val >= 181:
        new_vehicle_pos.append('≥ 181')
        vehicle_pos_counts[9] += 1
        vehicle_pos_discretized.append('≥ 181')
```

Figure 20 : Découpage de la position du véhicule

Après avoir subdivisé les positions latérales du véhicule un traitement des données est ensuite effectué tout d'abord pour subdiviser les jeux de données en deux groupes, un groupe pour effectuer les tests de validation du processus de classification et un second groupe pour l'entrainement du modèle de classification. Par la suite, une mise à l'échelle des données a été effectuée à l'aide de fonctionnalités de normalisation. La Figure 21 permet de présenter les fonctionnalités sklearn utilisées pour faire le découpage des données ainsi que leur normalisation.

Splitting Dataset Into Training and Test Sets

```
In [22]: y = df['vehicle position'].values
X = df.drop('vehicle position', axis=1).values
```

```
In [23]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Feature Scaling (Normalization)

```
In [24]: from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Figure 21 : Subdivision du jeu de données

Ainsi l'algorithme de création de l'arbre de décision est basé sur la bibliothèque sklearn.tree. Cette fonctionnalité sklearn a été appliquée sur le jeu de données en vue de faire l'apprentissage de l'arbre de décision (voir Figure 18) utilisé dans le contexte du processus de détection automatique de la vigilance d'un conducteur.

3.5.2 Classification en prenant comme variable cible l'état du sujet

La variable état est une variable pouvant prendre juste deux valeurs soit 0 pour un sujet peu ou pas fatigué et 1 pour un sujet fatigué donc ayant un niveau de vigilance amoindri. Les données ont été séparées en deux parties, une partie pour l'entrainement et 3% pour les tests. De même une normalisation a été effectué. L'arbre de décision a été aussi utilisée dans ce contexte.

Pour finir, un apprentissage a été effectué en prenant les deux variables simultanément.

3.6 Conclusion

Dans ce chapitre, nous avons présenté les traitements préalables appliqués sur les données. Entre autres, nous avons présentés les techniques de suppression de bruits les données. Après avoir effectué l'épuration des données, nous avons ensuite utilisé un arbre de décision pour faire l'entrainement du classificateur servant à déterminer si une personne est vigilante au volant. Nous avons aussi fait une brève présentation des outils et fonctionnalités utilisés. La méthodologie étant présentée, nous exposons dans le prochain chapitre les résultats qui découlent de cette méthodologie introduite.

CHAPITRE 4 : PRÉSENTATION DES RÉSULTATS ET DISCUSSION

4.1 Introduction

Dans ce présent chapitre et après avoir présenté dans le précédent chapitre une méthodologie qui prenait en compte le traitement distinct des données EEG. Deux types de traitements ont notamment été effectués en prenant d'abord comme variable cible la position latérale du véhicule, ensuite la variable état (vigilance) a été utilisé pour faire la classification. Mieux encore nous avons utilisé les deux variables comme cible pour voir le rapport entre la position latérale du véhicule et l'état du conducteur. Dans ce chapitre nous présentons les résultats de classification obtenus et ensuite nous analysons ces résultats.

4.2 Présentation des résultats et interprétations

Nous allons d'abord présenter l'évaluation des modèles, ici dans notre cas nous avons évaluer nos modèles grâce aux critères d'évaluation. Les critères d'évaluation de ces modèles sont la précision, le rappel et le f1-score.

Le premier critère d'évaluation de modèle utilisé dans notre étude est : la précision. La précision tente de répondre à la question suivante : Quelle proportion d'identification positives était réellement correcte [16] ? Selon un principe énoncé en exemple, quand un utilisateur souhaite avoir des informations quelconques d'une base de données en interrogeant cette dernière. Il souhaite avoir des informations qui correspondent à ses attentes. Toutes les informations supplémentaires ou non nécessaires obtenues peuvent être considérées comme des bruits. La précision a pour but d'éviter ce genre de bruit. Plus la précision est grande, plus les informations proposées sont pertinentes. Alors voici une formule qui permet de calculer la précision :

$$precision\ d'une\ recheche = \frac{nombre\ d'informations\ pertinantes\ pour\ la\ recherche}{nombre\ d'informations\ pour\ la\ recherche}$$

La précision peut être la valeur prédictive positive. La formule qui définit mieux la précision est la suivante :

$$precision = \frac{VP}{VP + FP}$$

Où VP est le nombre de valeurs positives bien prédites, FP est le nombre de valeurs positives qui ont été mal prédites. Comme le montre la Figure 22.

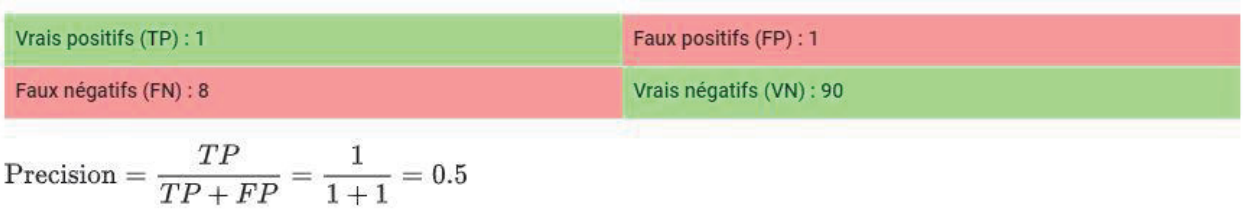


Figure 22 : Exemple du calcul de la précision [16]

Le rappel ou recall tente de répondre à la question quelle proportion de positifs réels a été identifiés correctement [16] ? Selon un principe énoncé en exemple quand un utilisateur souhaite avoir des informations quelconques d'une base de données en interrogeant cette dernière. Il souhaite avoir des informations qui correspondent à ses attentes. Si la quantité d'informations correspond à ce qu'il s'attend on parle de rappel. Dans le cas contraire on parle de silence. La formule du rappel est la suivante :

$$rappel = \frac{VP}{VP + FN}$$

Où FN représente les faux négatifs. La figure 23 montre l'exemple d'un rappel :

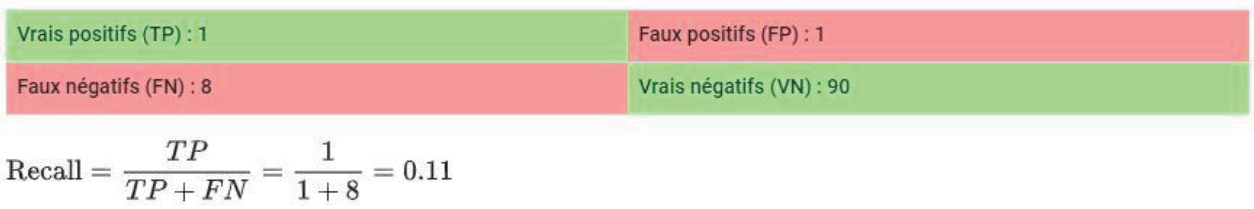


Figure 23 : Exemple de calcul du rappel [16]

Le F-mesure ou F-score est une mesure qui combine la précision et le rappel. Ainsi on obtient leur moyenne harmonique dont la formule est :

$$F = 2 \,.\, \frac{précision \,.\, rappel}{précision + rappel}$$

Dans certaines circonstances on peut rencontrer F1 au lieu de F parce que la précision et le rappel sont pondérés et F1 lorsque la pondération est égale à 1. β est la pondération de F mésure. La formule des F-mesure pondérées devient :

$$F_\beta = \frac{(1 + \beta^2) . (précision . rappel)}{(\beta^2 . précision + rappel)}$$

Dans notre étude, le premier modèle testé est le modèle où la variable cible est la position latérale du véhicule. Alors nous obtenons les résultats présentés à la Figure 24. La position latérale normale de la voiture est 157 il survient un décalage à chaque fois que la voiture tend vers la gauche, la position de la voiture subit une décrémentation et chaque fois que la voiture tend vers la droite il y'a une incrémentation. Ainsi on voit que le modèle a de façon générale bien prédit lorsque la voiture était complètement à gauche avec des score proche de 1 et au point 0 tous les scores étaient à 1.

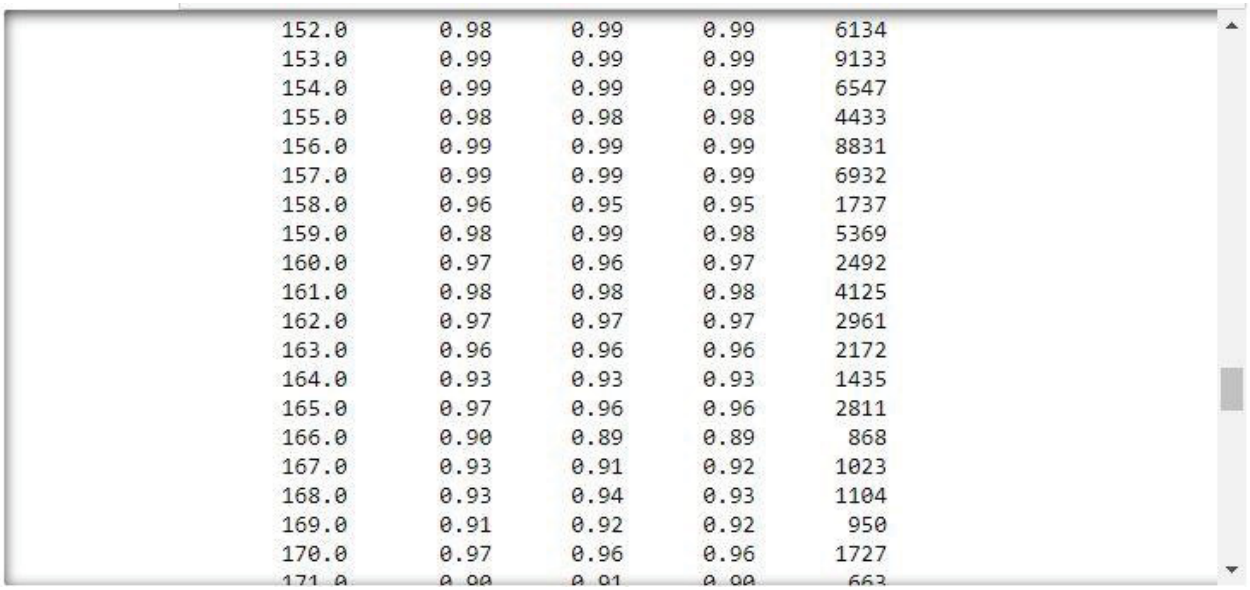
```
In [51]: print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	35550
1.0	0.99	0.99	0.99	2620
2.0	0.90	0.86	0.88	268
3.0	0.86	0.92	0.89	249
4.0	0.87	0.85	0.86	194
5.0	0.86	0.87	0.86	189
6.0	0.84	0.88	0.86	175
7.0	0.84	0.80	0.82	164
8.0	0.85	0.81	0.83	187
9.0	0.81	0.83	0.82	155
10.0	0.80	0.79	0.79	166
11.0	0.78	0.76	0.77	141
12.0	0.83	0.76	0.79	183
13.0	0.72	0.82	0.77	141
14.0	0.78	0.76	0.77	160
15.0	0.78	0.81	0.80	168
16.0	0.79	0.80	0.80	157

```
In [52]: print('Accuracy: {:.3f}%'.format(accuracy_score(y_test, predictions)*100))
Accuracy: 95.360%
```

Figure 24 : Test du modèle pour la variable position latérale de la voiture (1)

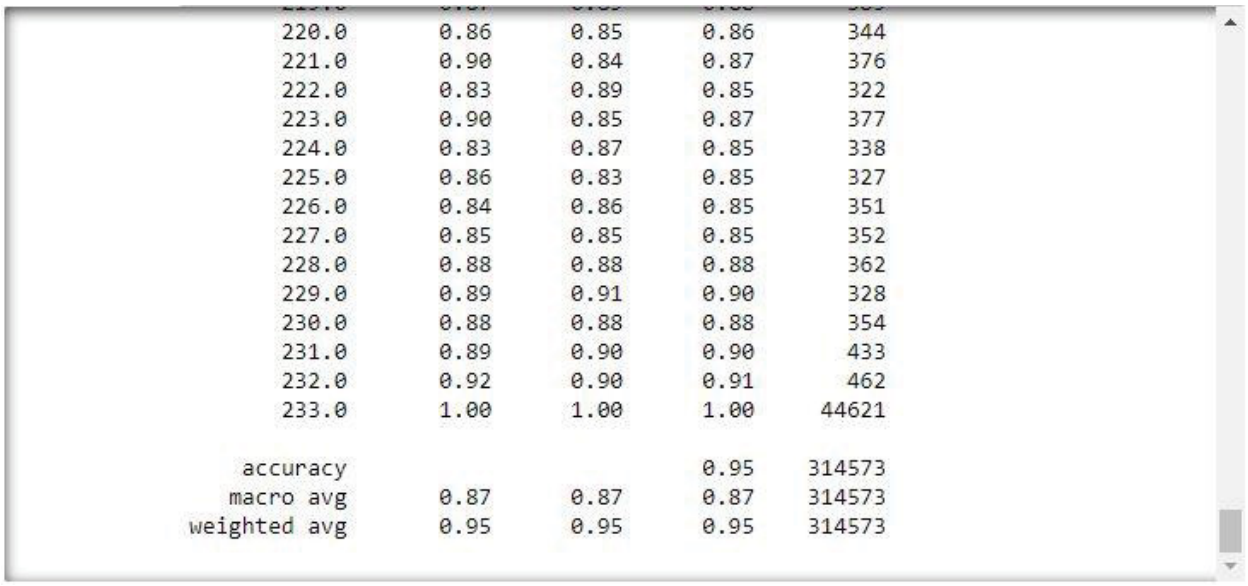
Le modèle lorsque la voiture est proche de la position latérale normale est très proche de 1 aussi dans la majorité des cas comme le montre la Figure 25.



152.0	0.98	0.99	0.99	6134
153.0	0.99	0.99	0.99	9133
154.0	0.99	0.99	0.99	6547
155.0	0.98	0.98	0.98	4433
156.0	0.99	0.99	0.99	8831
157.0	0.99	0.99	0.99	6932
158.0	0.96	0.95	0.95	1737
159.0	0.98	0.99	0.98	5369
160.0	0.97	0.96	0.97	2492
161.0	0.98	0.98	0.98	4125
162.0	0.97	0.97	0.97	2961
163.0	0.96	0.96	0.96	2172
164.0	0.93	0.93	0.93	1435
165.0	0.97	0.96	0.96	2811
166.0	0.90	0.89	0.89	868
167.0	0.93	0.91	0.92	1023
168.0	0.93	0.94	0.93	1104
169.0	0.91	0.92	0.92	950
170.0	0.97	0.96	0.96	1727
171.0	0.90	0.91	0.90	663

Figure 25 : Test du modèle pour la variable position latérale de la voiture (2)

De même pour toutes les mesures, les tests sont proches de 1 pour le modèle basé sur l'arbre de décision et ayant pour variable cible la position latérale de la voiture. La Figure 26 propose les résultats.



220.0	0.86	0.85	0.86	344
221.0	0.90	0.84	0.87	376
222.0	0.83	0.89	0.85	322
223.0	0.90	0.85	0.87	377
224.0	0.83	0.87	0.85	338
225.0	0.86	0.83	0.85	327
226.0	0.84	0.86	0.85	351
227.0	0.85	0.85	0.85	352
228.0	0.88	0.88	0.88	362
229.0	0.89	0.91	0.90	328
230.0	0.88	0.88	0.88	354
231.0	0.89	0.90	0.90	433
232.0	0.92	0.90	0.91	462
233.0	1.00	1.00	1.00	44621
accuracy			0.95	314573
macro avg	0.87	0.87	0.87	314573
weighted avg	0.95	0.95	0.95	314573

Figure 26 : Test du modèle pour la variable position latérale de la voiture (3)

La Figure 27 présente pour les mêmes tests mais avec des regroupements de la position latérale du véhicule ainsi on a dix regroupements où 0 correspond aux valeurs de la position latérale du véhicule inférieures à 101, 1 correspond à [101-110], 2 correspond à [111-120], 3 correspond à

[121-130], 4 correspond à [131-140], 5 correspond à [141-150], 6 correspond à [151-160], 7 correspond à [161-170], 8 correspond à toutes les valeurs supérieures à 180. La Figure 27 montre que tous les scores sont proches de 1.

```
In [61]: print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
0	0.98	0.98	0.98	6525
1	0.99	0.99	0.99	7007
2	0.99	0.99	0.99	8445
3	0.99	0.99	0.99	16898
4	1.00	1.00	1.00	55928
5	1.00	1.00	1.00	57628
6	0.99	0.99	0.99	19220
7	0.98	0.98	0.98	6256
8	1.00	1.00	1.00	68567
9	1.00	1.00	1.00	68099
accuracy			1.00	314573
macro avg	0.99	0.99	0.99	314573
weighted avg	1.00	1.00	1.00	314573

Figure 27 : Test du modèle pour le regroupement des valeurs de la variable position latérale de la voiture

Si nous prenons la variable état qui vérifie l'état de vigilance du sujet au moment de la conduite, 0 est obtenu si le sujet n'est pas fatigué et 1 si le sujet est fatigué, nous obtenons les résultats dans la Figure 28 où tous les scores sont à un.

```
In [94]: print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	191118
1	1.00	1.00	1.00	123455
accuracy			1.00	314573
macro avg	1.00	1.00	1.00	314573
weighted avg	1.00	1.00	1.00	314573

Figure 28 : Test du modèle pour la variable état

Une évaluation en prenant en compte les deux variables nous donne le résultat exposé dans la Figure 29 où le score est à près de 95% avec une perte de Hamming de 0.26, soit la fraction de données mal prédites.

Model Testing & Evaluation

In [103]:
```
# Concatenating 2 predicted labels into 1 numpy array
y_pred = np.concatenate((np.expand_dims(classifier_1.predict(X_test), axis=1),
                         np.expand_dims(classifier_2.predict(X_test), axis=1)), axis=1)

# Converting true labels into a numpy array
y_true = np.concatenate((np.expand_dims(np.asarray(vehicle_y_test), axis=1),
                         np.expand_dims(np.asarray(state_y_test), axis=1)), axis=1)

# Calculating Exact Match score and the Hamming Loss
EMscore = 0
count_Ham = 0

for i in range(0, y_true.shape[0]):
    count_EM = 0
    for j in range(0, y_true.shape[1]):
        if y_true[i][j] == y_pred[i][j]:
            count_EM += 1
        else:
            count_Ham += 1
    if count_EM == 2:
        EMscore += 1

EMscore = EMscore / y_true.shape[0]
Hamming_Loss = count_Ham / (y_true.shape[0] * y_true.shape[1])

print("Using Binary Relevance With Decision Tree Classifier:", "\n")
print("Exact Match score: {:.3f}%".format(EMscore*100))
print("Hamming Loss: {:.3f}".format(Hamming_Loss))
```

```
Using Binary Relevance With Decision Tree Classifier:

Exact Match score: 94.803%
Hamming Loss: 0.026
```

Figure 29 : Évaluation du modèle prenant en compte les deux variables (position latérale et état)

Après avoir effectué différents tests sur le modèle de classification des états de vigilance d'un conducteur et/ou sur la position latérale du véhicule, nous devons ensuite évaluer les performances de ces modèles de classification implémentés dans chaque contexte. Les performances sont alors évaluées à partir des résultats que chacun des modèles nous donne. Le principal outil utilisé pour évaluer l'efficacité du processus de classification des états de vigilance de conducteur (et/ou sur la position latérale du véhicule) déduite des résultats de tests est la matrice de confusion. La matrice de confusion, aussi appelée matrice d'erreur est un tableau qui présente différentes prévisions et résultats de test, en les comparant avec les valeurs réelles [17]. Elles sont utilisées en intelligence artificielle, en apprentissage automatique, en data mining et

même en statistiques. La figure 30 présente un exemple de matrice de confusion, pour évaluer l'efficacité des résultats de tests de grossesse.

	Pas enceinte	Enceinte
Résultat du test positif	Faux positif	Vrai positif
Résultat du test négatif	Vrai négatif	Faux négatif

Figure 30 : Exemple de matrice de confusion [17]

Les Figures 31 et 32, présentent dans notre cas les matrices de confusion permettant d'évaluer les résultats obtenus dans notre étude. De façon générale, ces matrices nous montrent de très bon pourcentage de bonnes classifications car une bonne partie des données se trouvent dans le vrai positif et le vrai négatif, pour les deux variables. Et les valeurs mal classées sont très petites voir même négligeables.

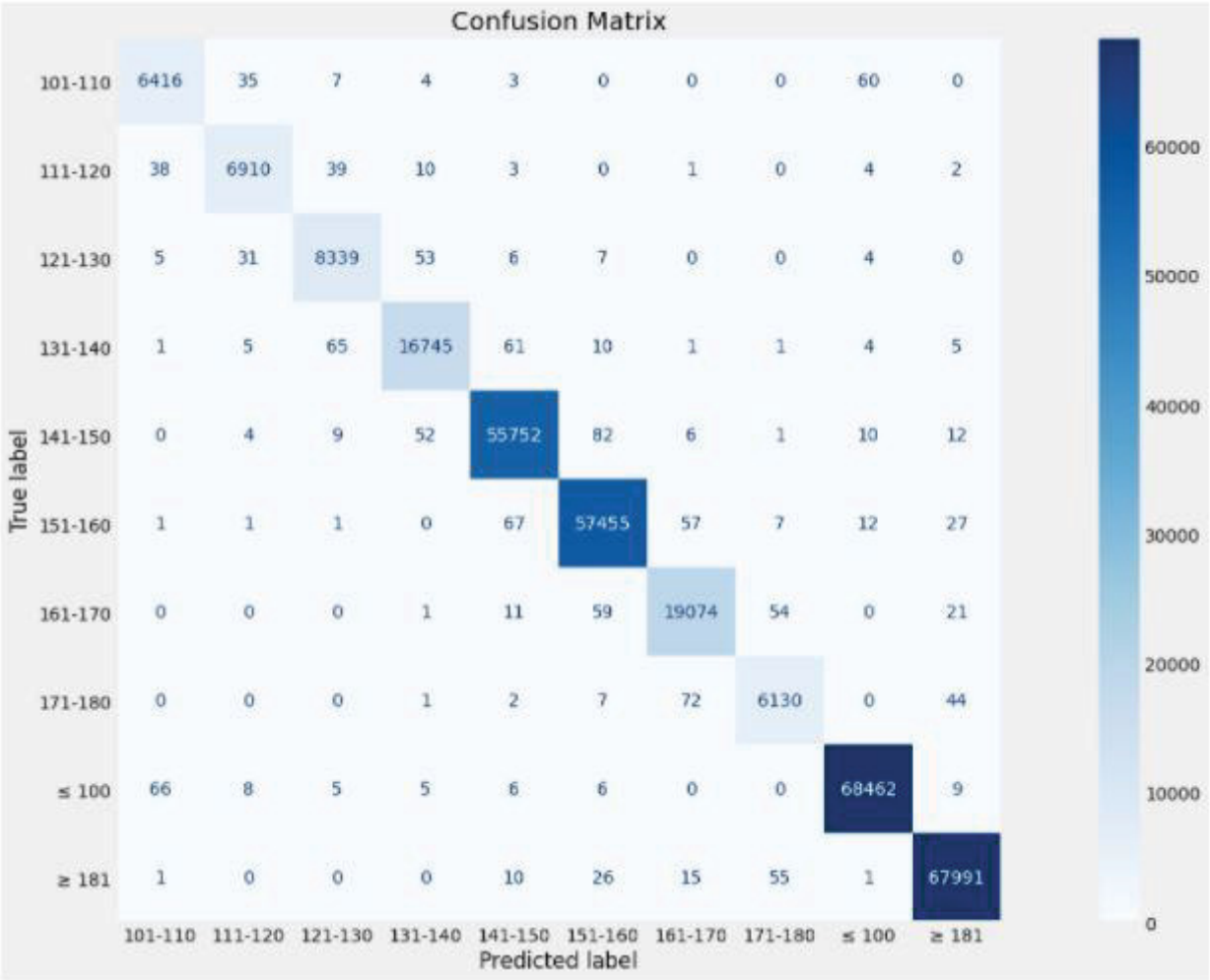


Figure 31 : Matrice de confusion utilisant la variable position latérale de la voiture

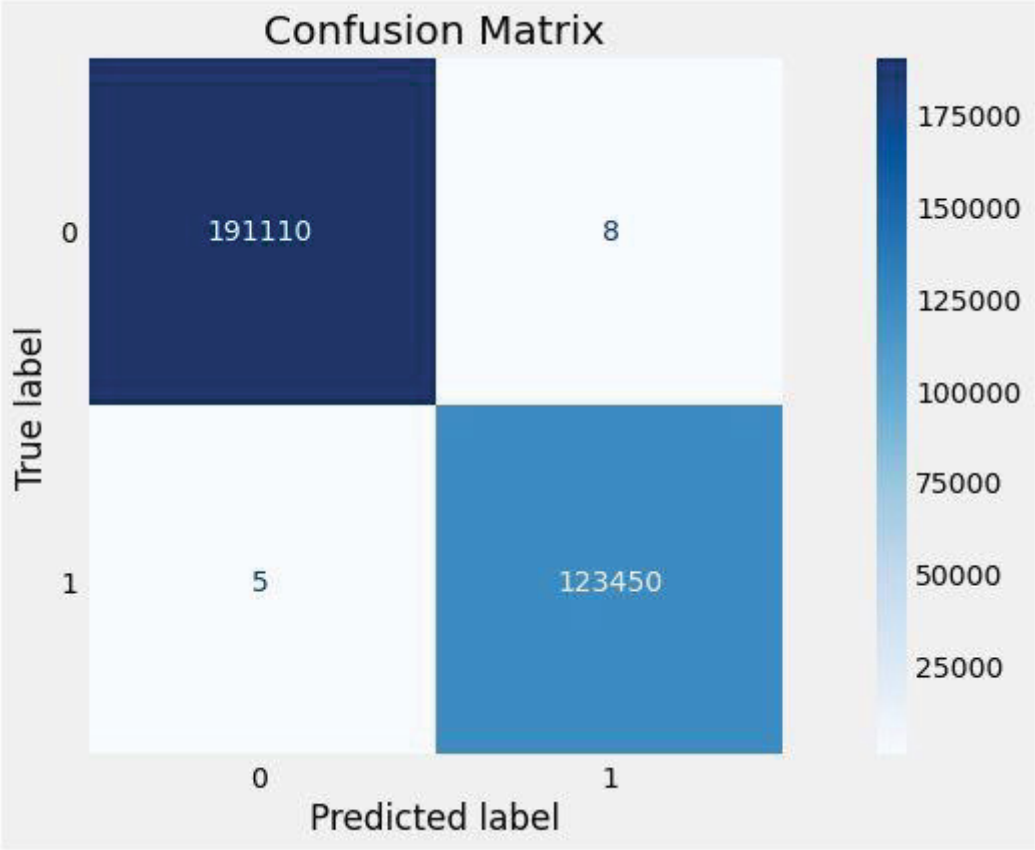


Figure 32 : La matrice de confusion utilisant l'état de vigilance du sujet

4.3 Discussions

Les tests ayant été effectués, nous voyons bien que l'algorithme de classification basée sur un arbre de décision a un très bon niveau de prédiction pour tous les critères d'évaluation étudiés. Qu'il s'agisse de faire une classification en prenant pour variable cible l'état de vigilance d'un conducteur et la position latérale du véhicule produisant des précisions et des rappels très élevés et proche de 1 (Figure 27 et 28). Quand nous combinons les deux critères de d'évaluation, l'état de vigilance du sujet et la position latérale du véhicule, l'algorithme reste toujours performant mais dans une moindre mesure que lorsque les variables cibles sont prises individuellement. Pour ce qui est de la variable cible État, l'algorithme de classification est comme pour ainsi dire parfait. Cela s'explique notamment par la grande quantité de données fournies. Une simulation en abaissant la quantité de données montre que l'algorithme est moins précis. Une simulation avec d'autres algorithmes comme l'annexe nous le démontre, avec moins de données d'entraînement les scores d'autres algorithmes tels que Random Forest, XGBoost, Naïves Bayes donnent une

plus grande précision que celle obtenue avec l'arbre de décision employé dans la présente recherche. Une fois la quantité de données commence à être importante l'arbre de décision s'avère être le meilleur algorithme pour notre étude avec des scores qui frôlent la perfection Tableau 2.

Tableau 2 : Tableau comparatif de l'efficacité des algorithmes prenant en compte la variable cible état

Critères d'évaluations / Algorithmes	Précision	Recall	F1-Score
Random Forest	0.855	0.98	0.456
XGBoost	0.92	0.96	0.306
Naïves Bayes	0.98	1	0.492
Decision Tree	1	1	1

Donc l'algorithme de classification basé sur l'arbre de décision produit une bonne détection de l'état de vigilance d'un conducteur et de la position latérale du véhicule. De même lorsqu'on combine les deux critères nous obtenons aussi des bons résultats de classification de l'algorithme. Cependant, la classification lorsque nous utilisons les deux variables cibles (l'état de vigilance d'un conducteur et la position latérale du véhicule) fonctionne moins bien que lorsqu'on les prend séparément.

Tableau 3 : Tableau récapitulatif des résultats des différents algorithmes

Résultats / Algorithmes	Exact match score	Hamming Loss
Random Forest	98,624%	0.9%
XGBoost	88,956%	4,1%
Naïves Bayes	95,180%	1,9%
Decision Tree	94,803%	2,6%

Au niveau de la détection automatique basé sur l'algorithme d'arbre de décision, celui-ci prédit bien l'emplacement latéral du véhicule lorsqu'on considère des intervalles de position latérale.

Ainsi on remarque que les positions latérales qui sont proches de la normale soit l'intervalle [141-150] et [151-160] sont très bien prédites. Les prédictions sont aussi bonnes lorsque la voiture est extrêmement à gauche et extrêmement à droite. Quant aux déplacements latéraux transitoires de la voiture, du centre vers les extrêmes à gauche ou à droite, ceux-ci sont assez bien prédits par le modèle de classification basé sur la variable cible position latérale. Cependant, ces résultats de prédiction ne sont pas aussi bons que ceux obtenus pour les intervalles de distances proches de la normale et ceux proches des intervalles extrêmes. Cela s'explique par le fait que le sujet a une conduite normale en ligne droite que ce soit à gauche ou à droite. Mais la conduite est moins précise dans les opérations pour se rendre à gauche ou pour se rendre à droite. La figure 31 montre un très bon résultat de classification de l'état de fatigue avec des erreurs très faibles ce qui s'explique par la grande quantité de données fournie à l'algorithme lors du processus d'entraînement.

D'autre part l'utilisation d'une limite de profondeur rend de mauvais résultats. La mise à défaut permet de créer plus de nœud et de lutter contre la perte d'information.

4.4 Conclusion

Dans ce chapitre nous avons présenté les résultats de notre méthodologie de classification de l'état de vigilance d'un conducteur et/ou sur la position latérale du véhicule qui sont de bons résultats en considérant les deux facteurs utilisés pour la classification : l'état du sujet et la position latérale du véhicule. L'analyse des résultats nous montre que de façon général l'algorithme prédit bien la position du véhicule sauf dans les situations où le conducteur fait des manœuvres transitoires vers l'extrême gauche ou vers l'extrême droite. Mais la classification des données du poids des signaux EEG des sujets est très bonne pour prédire l'état du sujet à la conduite.

CHAPITRE 5 : CONCLUSION

Dans ce mémoire nous nous sommes intéressés aux signaux EEG, à la classification des signaux EEG pour déceler l'état de fatigue pour un sujet effectuant des opérations de conduite automobile en simulateur. Nous nous sommes intéressés au fonctionnement du cerveau, et de comment les activités électriques d'un neurone permettent de faire passer les messages et de commander, par exemple, les muscles. L'activités des neurones est mesurable grâce des technique et procédé que nous avons présenté. Ces mesures seront recueillies sous forme de signaux EEG, ces signaux EEG recueillis numériquement seront les bases de nos données utilisées. Un traitement préalable nous permet de recadrer nous signaux en vue d'enlever le superflu d'information comme par exemple, parmi les traitements on a le cadrage de fréquence, en vue d'avoir les fréquences qui nous intéressent. Les données sont ensuite filtrées par des filtres passe-haut et des filtres passe-bas. Suit alors des phases pour enlever le plus grand nombre de bruits dans les données, car en obtenant les signaux, le mouvement des paupières, et des muscles ont pu aussi être capturé. L'utilisation de la transformée en ondelette pour extraire les caractéristiques et l'ICA pour corriger les artéfacts ont été notoire. Les données prises en état de fatigue et les données prises en état normal sont donc mises ensemble avec la création d'une variable état pour les différencier. Les données ainsi prêtes, l'arbre de décision est d'abord entraîné et ensuite a été utilisé pour faire la classification automatique en prenant en compte les deux variable cibles : l'état du sujet et la position latérale du véhicule. Les résultats montrent tout d'abord que l'algorithme choisi a une bonne prédiction à cause de la grande quantité de données mais plus la quantité de données est faible d'autres algorithmes donnent de meilleurs résultats. Pour ce qui est de la position latérale du véhicule, indépendamment de l'état du patient la position latérale du véhicule est bien prédite. On obtient de meilleurs résultats pour les positions fixes du véhicule que lorsque le sujet effectue des manœuvres transitoires faisant bouger le véhicule vers la gauche ou vers la droite. La classification de l'état du sujet est très satisfaisante. La question qui va se poser sera de savoir comment automatiser un tel système pour l'incorporer dans un automobile et accommodé le conducteur. Outre le coté recherche, le défi sera de savoir si avec les données EEG on peut faire déplacer une voiture dans un espace donné, des études se sont déjà penché sur le sujet

RÉFÉRENCES

[1] - https://saaq.gouv.qc.ca/securite-routiere/comportements/fatigue/saviez-vous/#:~:text=Chaque%20ann%C3%A9e%2C%20la%20fatigue%20au,cause%20de%20notre%20horloge%20interne. Consulté le 20-02-2022.

[2] - https://ia.ca/zone-conseils/vehicule/quoi-faire-pour-eviter-la-fatigue-au-volant#:~:text=La%20seule%20m%C3%A9thode%20pour%20contrer,un%20repos%20de%2015%20minutes. Consulté le 20-02-2022.

[3] - https://www.guillaumedarding.fr/technique-le-detecteur-de-fatigue-3898654.html#:~:text=Le%20d%C3%A9tecteur%20de%20fatigue%20analyse,pr%C3%A9venir%20la%20fatigue%20sur%20autoroute. Consulté le 20-02-2022.

[4] - https://fr.wikipedia.org/wiki/Analyse_en_composantes_ind%C3%A9pendantes Consulté le 20-02-2022.

[5] - https://fr.wikipedia.org/wiki/MATLAB Consulté le 20-02-2022.

[6] - CLASSIFICATION DES SIGNAUX EEG POUR LA DETECTION DE FACULTES AFFAIBLIES DU A LA CONSOMATION D'ALCOOL ET DE DROGUES, de SOUHAIL BERDAA, Septembre 2018, Université du Québec à Trois-Rivières

[7] - https://en.wikipedia.org/wiki/EEGLAB Consulté le 20-02-2022.

[8] – EEGLAB Tutorial, Arnaud Delorme, Toby Fernsler, Hilit Serby, Scott Makeig, 12 Avril 2006, University of San Diego California

[9] – Introduction à EEGLAB CREx jeudi 17 mars 2016 https://blricrex.hypotheses.org/files/2015/03/FormationEEGLAB_Debutants_cor.pdf Consulté le 20-02-2022.

[10] - https://support.google.com/analytics/answer/2637192?hl=fr Consulté le 20-02-2022.

[11] - https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_d%27%C3%A9chantillonnage#:~:text

=Le%20th%C3%A9or%C3%A8me%20d'%C3%A9chantillonnage%2C%20dit,spectrale%20et%20d'amplitude%20limit%C3%A9es. Consulté le 20-02-2022.

[12] - http://www-verimag.imag.fr/~tdang/DocumentsCours/echan-Shannon-b.pdf Consulté le 20-02-2022.

[13] - https://cnl.salk.edu/~jung/artifact.html Consulté le 20-02-2022.

[14] - https://institutducerveau-icm.org/fr/actualite/comprendre-le-cerveau-et-son-fonctionnement/ Consulté le 20-02-2022.

[15] - https://www.lucidchart.com/pages/fr/arbre-de-decision/#section_0 Consulté le 20-02-2022.

[16] - https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall?hl=fr Consulté le 20-02-2022.

[17] - https://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel Consulté le 20-02-2022.

[18] - https://www.lemagit.fr/definition/Matrice-de-confusion Consulté le 20-02-2022.

[19] – UTILISATION DES SIGNAUX DU CERVEAU (EEG) ET VOCAUX POUR LA DÉTECTION ET LE MONITORING DES FACULTÉS D'UNE PERSONNE, de Aymen Ben Messaoud, 2020, Université du Québec à Trois-Rivières

[20] – Automatic recognition of alertness level by using wavelet transform and artificial neural network, Kemal Kiymik, Mehmet Akin, Abdulhamit Subas, Journal of Neuroscience Methods 139 (2004) 231-240

[21] – EEG-based Driving Fatigue Prediction System Using Functional-link-based Fuzzy Neural Network, Yu-Ting Liu, Yang-Yin Lin, Shang-Lin Wu, Chun-Hsiang Chuang, Mukesh Prasad, Chin-Teng Lin, 2014 International Joint Conference on Neural network (IJCNN), 6-11 Juillet 2014, Pekin, Chine

[22] – A cognitive and neurophysiological test of change from an individual's baseline, Alan Gevins, Michael E. Smith, Linda K. McEvoy, Aaron B. Ilan, Cynthia S. Chan, An Jiang, Lita Sam-Vargas, et Gordon Abraham, Clin Neurophysiol, Janvier 2011.

[23] – Monitoring Driver's Altertness Based on the Driving Performance Estimation and the EEG Power Spectrum Analysis, S. F. Liang, C. T. Lin, R. C. Wu, Y. C. Chen, T. Y. Huang, et T. P. Jung, Engineering in Medecine and Biology 27th Annual Conference, Shanghai, Chine, 1-4 Septembre 2005.

[24] – Mental Fatigue Measurement Using EEG, Shyh-Yueh Cheng et Hong-Te Hsu, Institute of Engineering Science and Technology, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan.

[25] - https://www.oracle.com/ca-fr/data-science/machine-learning/what-is-machine-learning/ Consulté le 20-02-2022.

[26] - https://www.passeportsante.net/fr/Actualites/Dossiers/DossierComplexe.aspx?doc=7-signes-que-vous-etes-beaucoup-trop-fatigue#:~:text=Baisse%20de%20la%20vision%2C%20baisse,Le%20repos%20est%20alors%20n%C3%A9cessaire. Consulté le 20-02-2022.

[27] - https://www.guideautoweb.com/articles/48532/systeme-de-detection-de-fatigue-ce-que-vous-devez-savoir/#:~:text=Si%20la%20voiture%20venait%20%C3%A0,sonore%20peut%20aussi%20%C3%AAtre%20%C3%A9mise. Consulté le 20-02-2022.

[28] - https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html Consulté le 20-02-2022.

[29] – DÉTECTION DE FACULTÉS AFFAIBLIES PAR L'ANALYSE DU COMPORTEMENT OCULAIRE SUR DES SÉQUENCES VIDÉO, Pier-Olivier HOUDE, Avril 2008.

[30] – On the time-course and frequency selectivity of the EEG for different modes of response selection: Evidence from speech production and keyboard pressing, Pascale Tremblay, Douglas M. Shiller, Vincent L.Gracco 2008.

[31] - Data Analysis for Multi-channel EEG Recordings during A Sustained-attention Driving Task, Zehong Cao, Chun-Hsiang Chuang, Jung-Kai King, et Chin-Teng Lin, Avril 2019.

[32] - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6472414/bin/41597_2019_27_MOESM1_ESM.zip consulté le 01-02-2022.

ANNEXES

Importing the Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, accuracy_score, confusion_m
atrix
import matplotlib
matplotlib.style.use('fivethirtyeight')
```

Loading the Dataset

In [2]:
```
df = pd.read_csv('donnee1t.csv', sep=';')
```

In [3]:
```
df.head()
```

Out[3]:

	epoch	FP1	FP2	F7	F3	FZ	F4	F8	FT7	FC3	...	F
0	0.0	2.8861	8.4701	-14.3051	-6.0232	-1.4431	-6.3997	5.5840	-14.9325	-9.3485	...	-9.47
1	2.0	2.2587	7.8427	-14.8070	-5.8350	-0.7529	-6.3369	6.0859	-14.7443	-9.0348	...	-8.97
2	4.0	2.1960	7.6545	-14.9953	-5.3330	-0.1882	-6.0232	6.3997	-14.3679	-8.9093	...	-8.47
3	6.0	2.3842	7.9055	-15.0580	-4.0782	0.3137	-5.1448	7.0271	-14.1796	-8.0937	...	-7.71
4	8.0	2.1960	8.4074	-15.2462	-1.5685	0.8156	-4.0782	8.0309	-14.6188	-5.9605	...	-6.96

5 rows × 35 columns

Exploratory Data Analysis

Data Cleansing

In [4]:
```
df.dropna(axis=0, inplace=True)
df.dropna(axis=1, inplace=True)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1048575 entries, 0 to 1048574
Data columns (total 35 columns):
 #   Column            Non-Null Count    Dtype  
---  ------            --------------    -----  
 0   epoch             1048575 non-null  float64
 1   FP1               1048575 non-null  float64
 2   FP2               1048575 non-null  float64
 3   F7                1048575 non-null  float64
 4   F3                1048575 non-null  float64
 5   FZ                1048575 non-null  float64
 6   F4                1048575 non-null  float64
 7   F8                1048575 non-null  float64
 8   FT7               1048575 non-null  float64
 9   FC3               1048575 non-null  float64
 10  FCZ               1048575 non-null  float64
 11  FC4               1048575 non-null  float64
 12  FT8               1048575 non-null  float64
 13  T3                1048575 non-null  float64
 14  C3                1048575 non-null  float64
 15  CZ                1048575 non-null  float64
 16  C4                1048575 non-null  float64
 17  T4                1048575 non-null  float64
 18  TP7               1048575 non-null  float64
 19  CP3               1048575 non-null  float64
 20  CPZ               1048575 non-null  float64
 21  CP4               1048575 non-null  float64
 22  TP8               1048575 non-null  float64
 23  A1                1048575 non-null  float64
 24  T5                1048575 non-null  float64
 25  P3                1048575 non-null  float64
 26  PZ                1048575 non-null  float64
 27  P4                1048575 non-null  float64
 28  T6                1048575 non-null  float64
 29  A2                1048575 non-null  float64
 30  O1                1048575 non-null  float64
 31  OZ                1048575 non-null  float64
 32  O2                1048575 non-null  float64
 33  vehicle position  1048575 non-null  float64
 34  etat              1048575 non-null  int64  
dtypes: float64(34), int64(1)
memory usage: 288.0 MB
```

Statistics

In [6]:
```
df.describe()
```

Out[6]:

	epoch	FP1	FP2	F7	F3	FZ
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
mean	1.048574e+06	1.552225e-02	4.685246e-02	3.230125e-02	3.766735e-02	-1.132329e-02
std	6.053953e+05	1.702444e+02	1.478655e+02	4.061290e+01	9.556586e+01	2.435387e+01
min	0.000000e+00	-2.055921e+03	-2.055921e+03	-2.055419e+03	-2.055921e+03	-1.985776e+03
25%	5.242870e+05	-2.566140e+01	-2.434380e+01	-1.731670e+01	-1.769320e+01	-1.355220e+01
50%	1.048574e+06	-5.647000e-01	6.274000e-01	1.003900e+00	2.510000e-01	1.255000e-01
75%	1.572861e+06	2.773180e+01	2.628880e+01	2.095570e+01	1.838330e+01	1.399140e+01
max	2.097148e+06	2.055858e+03	2.055858e+03	1.833815e+03	2.055858e+03	2.025742e+03

8 rows × 35 columns

Count

In [7]:
```
df.describe(include='all').loc['count'].round(3)
```

Out[7]:
```
epoch              1048575.0
FP1                1048575.0
FP2                1048575.0
F7                 1048575.0
F3                 1048575.0
FZ                 1048575.0
F4                 1048575.0
F8                 1048575.0
FT7                1048575.0
FC3                1048575.0
FCZ                1048575.0
FC4                1048575.0
FT8                1048575.0
T3                 1048575.0
C3                 1048575.0
CZ                 1048575.0
C4                 1048575.0
T4                 1048575.0
TP7                1048575.0
CP3                1048575.0
CPZ                1048575.0
CP4                1048575.0
TP8                1048575.0
A1                 1048575.0
T5                 1048575.0
P3                 1048575.0
PZ                 1048575.0
P4                 1048575.0
T6                 1048575.0
A2                 1048575.0
O1                 1048575.0
OZ                 1048575.0
O2                 1048575.0
vehicle position   1048575.0
etat               1048575.0
Name: count, dtype: float64
```

Mean

In [8]:
```
df.describe(include='all').loc['mean'].round(3)
```

Out[8]:
```
epoch              1048574.000
FP1                      0.016
FP2                      0.047
F7                       0.032
F3                       0.038
FZ                      -0.011
F4                       0.006
F8                       0.014
FT7                     -0.001
FC3                      0.009
FCZ                     -0.003
FC4                     -0.191
FT8                      0.010
T3                       0.024
C3                       0.016
CZ                      -0.004
C4                      -0.182
T4                       0.014
TP7                      0.022
CP3                      0.008
CPZ                     -0.021
CP4                     -0.010
TP8                      0.008
A1                      -0.001
T5                       0.047
P3                      -0.003
PZ                       0.004
P4                      -0.032
T6                       0.002
A2                      -0.012
O1                       0.052
OZ                       0.032
O2                       0.047
vehicle position       137.950
etat                     0.392
Name: mean, dtype: float64
```

Standard Deviation

In [9]:
```
df.describe(include='all').loc['std'].round(3)
```

Out[9]:
```
epoch              605395.347
FP1                   170.244
FP2                   147.866
F7                     40.613
F3                     95.566
FZ                     24.354
F4                    158.731
F8                     27.625
FT7                    24.862
FC3                    23.042
FCZ                    24.420
FC4                    71.347
FT8                    27.373
T3                     22.123
C3                     19.856
CZ                     24.349
C4                     70.378
T4                     27.330
TP7                    17.050
CP3                    18.245
CPZ                    20.245
CP4                    19.160
TP8                    19.945
A1                     10.759
T5                     25.347
P3                     16.219
PZ                     19.471
P4                     18.554
T6                     16.799
A2                     10.121
O1                     22.878
OZ                     24.319
O2                     22.061
vehicle position       68.630
etat                    0.488
Name: std, dtype: float64
```

Minimum

In [10]:
```
df.describe(include='all').loc['min'].round(3)
```

Out[10]:
```
epoch                0.000
FP1              -2055.921
FP2              -2055.921
F7               -2055.419
F3               -2055.921
FZ               -1985.776
F4               -2055.921
F8               -2006.355
FT7              -2039.483
FC3              -1521.550
FCZ              -2037.789
FC4              -2005.853
FT8              -2043.122
T3               -1818.381
C3               -1997.571
CZ               -2053.913
C4               -1918.642
T4               -2018.402
TP7              -1998.826
CP3              -1900.447
CPZ              -1808.907
CP4              -1877.860
TP8              -1976.490
A1               -2013.131
T5               -1921.152
P3               -1766.556
PZ               -1977.745
P4               -1966.012
T6               -2054.980
A2               -1927.112
O1               -1919.521
OZ               -1707.956
O2               -2047.639
vehicle position     0.000
etat                 0.000
Name: min, dtype: float64
```

0.25 Quantile

In [11]:
```
df.describe(include='all').quantile(q=0.25).round(3)
```

Out[11]:
```
epoch              585118.260
FP1                    -6.839
FP2                    -6.051
F7                     -4.305
F3                     -4.395
FZ                     -3.397
F4                     -3.838
F8                     -3.424
FT7                    -2.934
FC3                    -3.184
FCZ                    -3.390
FC4                    -3.547
FT8                    -3.412
T3                     -2.539
C3                     -2.670
CZ                     -3.391
C4                     -3.540
T4                     -3.393
TP7                    -2.054
CP3                    -2.588
CPZ                    -2.917
CP4                    -2.572
TP8                    -2.243
A1                     -1.098
T5                     -4.278
P3                     -2.183
PZ                     -2.695
P4                     -2.643
T6                     -2.588
A2                     -1.122
O1                     -3.968
OZ                     -4.235
O2                     -3.717
vehicle position      104.158
etat                    0.000
Name: 0.25, dtype: float64
```

0.50 Quantile (Median)

In [12]:
```
df.describe(include='all').quantile(q=0.50).round(3)
```

Out[12]:
```
epoch              1048574.000
FP1                     13.874
FP2                     13.458
F7                      10.980
F3                       9.317
FZ                       7.058
F4                       8.125
F8                       7.435
FT7                      7.058
FC3                      6.530
FCZ                      7.027
FC4                      7.121
FT8                      7.372
T3                       5.710
C3                       5.521
CZ                       7.027
C4                       7.121
T4                       7.341
TP7                      4.517
CP3                      5.086
CPZ                      5.730
CP4                      5.014
TP8                      4.490
A1                       2.227
T5                       8.470
P3                       4.361
PZ                       5.521
P4                       5.239
T6                       4.707
A2                       2.196
O1                       7.053
OZ                       7.482
O2                       6.831
vehicle position       143.975
etat                     0.440
Name: 0.5, dtype: float64
```

0.75 Quantile

In [13]:
```
df.describe(include='all').quantile(q=0.75).round(3)
```

Out[13]:
```
epoch              1179646.500
FP1                    641.648
FP2                    624.864
F7                     488.914
F3                     585.639
FZ                     524.701
F4                     633.013
F8                     499.658
FT7                    498.181
FC3                    511.546
FCZ                    524.923
FC4                    567.475
FT8                    515.390
T3                     529.977
C3                     522.285
CZ                     490.127
C4                     566.748
T4                     503.186
TP7                    523.960
CP3                    478.569
CPZ                    518.247
CP4                    519.802
TP8                    494.023
A1                     469.754
T5                     488.177
P3                     469.630
PZ                     460.760
P4                     518.469
T6                     439.556
A2                     413.435
O1                     515.061
OZ                     519.593
O2                     527.734
vehicle position       183.500
etat                     1.000
Name: 0.75, dtype: float64
```

Maximum

In [14]:
```
df.describe(include='all').loc['max'].round(3)
```

Out[14]:
```
epoch              2097148.000
FP1                   2055.858
FP2                   2055.858
F7                    1833.815
F3                    2055.858
FZ                    2025.742
F4                    2055.858
F8                    1915.756
FT7                   1918.140
FC3                   1977.055
FCZ                   2026.433
FC4                   2055.858
FT8                   1979.439
T3                    2053.537
C3                    2029.570
CZ                    1887.460
C4                    2055.858
T4                    1930.752
TP7                   2044.690
CP3                   1859.540
CPZ                   2012.253
CP4                   2021.727
TP8                   1916.258
A1                    1846.740
T5                    1876.668
P3                    1829.863
PZ                    1784.626
P4                    2018.213
T6                    1707.830
A2                    1623.380
O1                    1991.611
OZ                    2005.414
O2                    2044.753
vehicle position       233.000
etat                     1.000
Name: max, dtype: float64
```

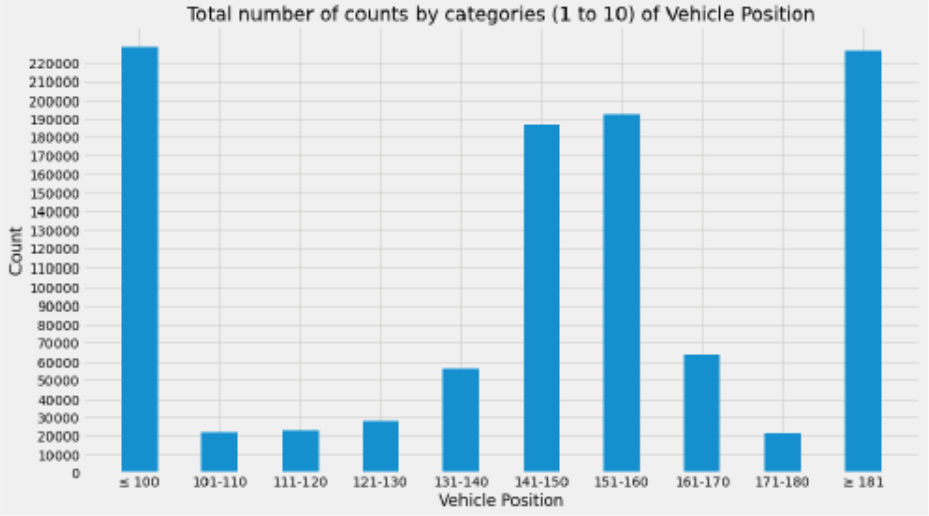
Applying Discretization To Vehicle Position

In [15]:
```
vehicle_pos = df['vehicle position'].values
vehicle_pos_discretized = []
new_vehicle_pos = []
vehicle_pos_counts = [0] * 10
ranges = ['≤ 100', '101-110', '111-120', '121-130', '131-140', '141-150', '151
-160', '161-170', '171-180', '≥ 181']
for val in vehicle_pos:
    if val <= 100:
        new_vehicle_pos.append('≤ 100')
        vehicle_pos_counts[0] += 1
        vehicle_pos_discretized.append('≤ 100')
    elif 101 <= val <= 110:
        new_vehicle_pos.append('101-110')
        vehicle_pos_counts[1] += 1
        vehicle_pos_discretized.append('101-110')
    elif 111 <= val <= 120:
        new_vehicle_pos.append('111-120')
        vehicle_pos_counts[2] += 1
        vehicle_pos_discretized.append('111-120')
    elif 121 <= val <= 130:
        new_vehicle_pos.append('121-130')
        vehicle_pos_counts[3] += 1
        vehicle_pos_discretized.append('121-130')
    elif 131 <= val <= 140:
        new_vehicle_pos.append('131-140')
        vehicle_pos_counts[4] += 1
        vehicle_pos_discretized.append('131-140')
    elif 141 <= val <= 150:
        new_vehicle_pos.append('141-150')
        vehicle_pos_counts[5] += 1
        vehicle_pos_discretized.append('141-150')
    elif 151 <= val <= 160:
        new_vehicle_pos.append('151-160')
        vehicle_pos_counts[6] += 1
        vehicle_pos_discretized.append('151-160')
    elif 161 <= val <= 170:
        new_vehicle_pos.append('161-170')
        vehicle_pos_counts[7] += 1
        vehicle_pos_discretized.append('161-170')
    elif 171 <= val <= 180:
        new_vehicle_pos.append('171-180')
        vehicle_pos_counts[8] += 1
        vehicle_pos_discretized.append('171-180')
    elif val >= 181:
        new_vehicle_pos.append('≥ 181')
        vehicle_pos_counts[9] += 1
        vehicle_pos_discretized.append('≥ 181')
```

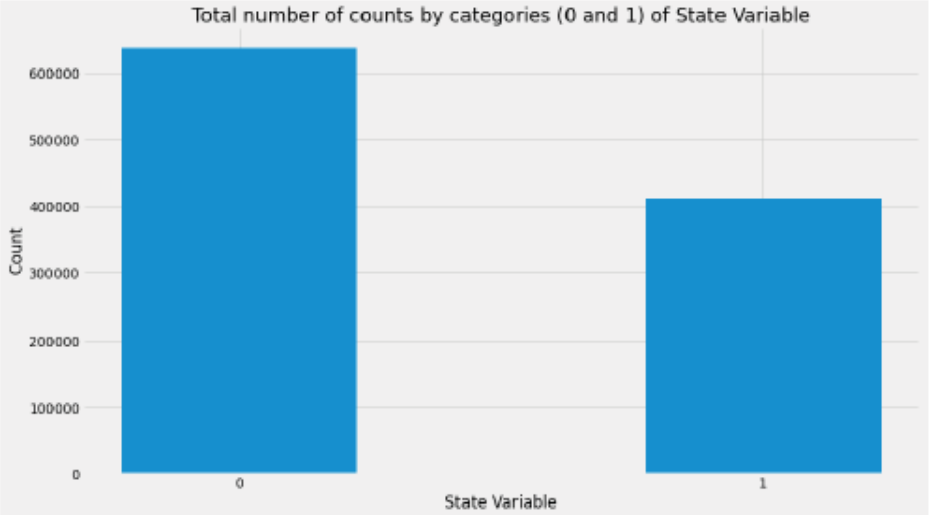
Visualization

Frequency Diagram

In [16]:
```
x_range = range(len(ranges))
y_range = range(0, max(vehicle_pos_counts)+50, 10000)
plt.figure(figsize=(14, 8))
plt.bar(x_range, vehicle_pos_counts, 0.45, align="center")
plt.title('Total number of counts by categories (1 to 10) of Vehicle Position'
)
plt.xlabel('Vehicle Position')
plt.ylabel('Count')
plt.xticks(x_range, ranges)
plt.yticks(y_range)
plt.show()
```

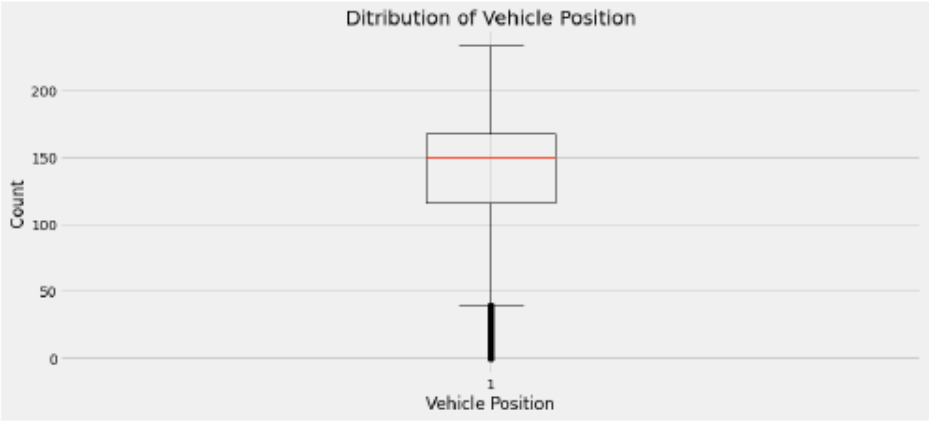


In [17]:
```
x_range = range(len(df['etat'].unique()))
y_range = range(0, max(df['etat'].value_counts())+50, 100000)
plt.figure(figsize=(14, 8))
plt.bar(x_range, df['etat'].value_counts(), 0.45, align="center")
plt.title('Total number of counts by categories (0 and 1) of State Variable')
plt.xlabel('State Variable')
plt.ylabel('Count')
plt.xticks(x_range, x_range)
plt.yticks(y_range)
plt.show()
```

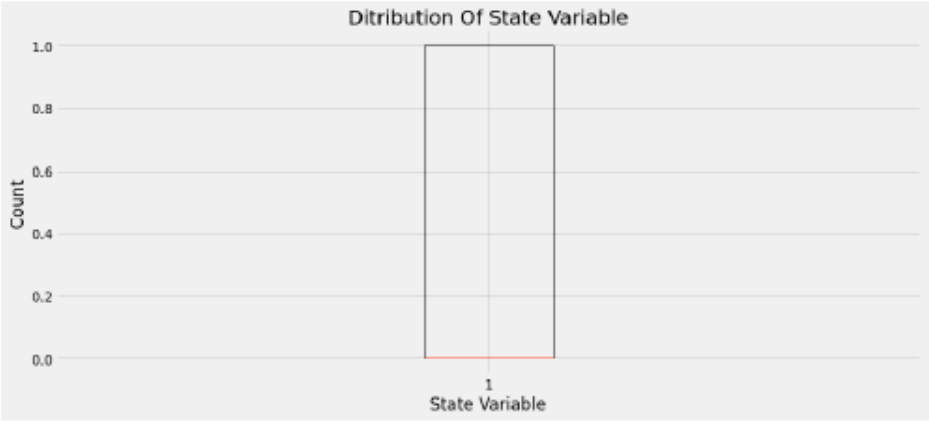


Box Plot

In [18]:
```
plt.figure(figsize=(14, 6))
plt.boxplot(df['vehicle position'])
plt.title('Ditribution of Vehicle Position')
plt.xlabel('Vehicle Position')
plt.ylabel('Count')
plt.show()
```

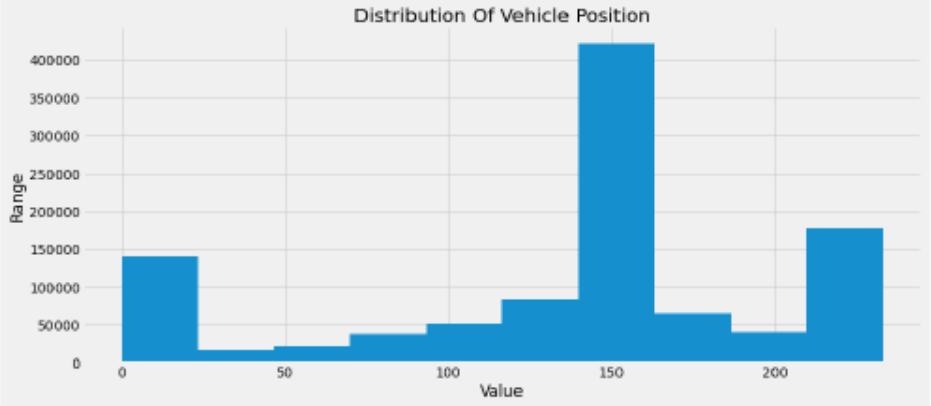


In [19]:
```
plt.figure(figsize=(14, 6))
plt.boxplot(df['etat'])
plt.title('Ditribution Of State Variable')
plt.xlabel('State Variable')
plt.ylabel('Count')
plt.show()
```

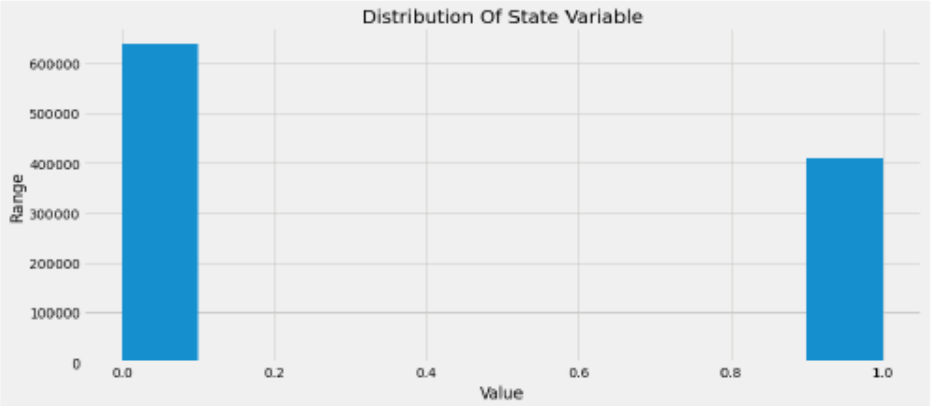


Histograms

In [20]:
```
plt.figure(figsize=(14, 6))
plt.hist(df['vehicle position'], 10)
plt.title('Distribution Of Vehicle Position')
plt.xlabel('Value')
plt.ylabel('Range')
plt.show()
```



In [21]:
```
plt.figure(figsize=(14, 6))
plt.hist(df['etat'], 10)
plt.title('Distribution Of State Variable')
plt.xlabel('Value')
plt.ylabel('Range')
plt.show()
```



Single Label Classification of Vehicle Position

Data Processing

Splitting Dataset Into Training and Test Sets

In [22]:
```
y = df['vehicle position'].values
X = df.drop('vehicle position', axis=1).values
```

In [23]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Feature Scaling (Normalization)

In [24]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Support Vector Machines

In [25]:
```
# from sklearn.svm import SVC
# classifier = SVC()
# classifier.fit(X_train, y_train)
```

In [26]:
```
# predictions = classifier.predict(X_test)
```

In [27]:
```
# print(classification_report(y_test, predictions))
```

In [28]:
```
# print(accuracy_score(y_test, predictions))
```

Naive Bayes

In [29]:
```
# from sklearn.naive_bayes import GaussianNB
# classifier = GaussianNB()
# classifier.fit(X_train, y_train)
```

In [30]:
```
# predictions = classifier.predict(X_test)
```

In [31]:
```
# print(classification_report(y_test, predictions))
```

In [32]:
```
# print(accuracy_score(y_test, predictions))
```

In [33]:
```
# print(confusion_matrix(y_test, predictions))
```

K Nearest Neighbors

In [34]:
```
# from sklearn.neighbors import KNeighborsClassifier
# classifier = KNeighborsClassifier()
# classifier.fit(X_train, y_train)
```

In [35]:
```
# predictions = classifier.predict(X_test)
```

In [36]:
```
# print(classification_report(y_test, predictions))
```

In [37]:
```
# print(accuracy_score(y_test, predictions))
```

In [38]:
```
# print(confusion_matrix(y_test, predictions))
```

Random Forest

In [39]:
```
# from sklearn.ensemble import RandomForestClassifier
# classifier = RandomForestClassifier()
# classifier.fit(X_train, y_train)
```

In [40]:
```
# predictions = classifier.predict(X_test)
```

In [41]:
```
# print(classification_report(y_test, predictions))
```

In [42]:
```
# print(accuracy_score(y_test, predictions))
```

In [43]:
```
# print(confusion_matrix(y_test, predictions))
```

XGBoost

In [44]:
```
# from xgboost import XGBClassifier
# classifier = XGBClassifier()
# classifier.fit(X_train, y_train)
```

In [45]:
```
# predictions = classifier.predict(X_test)
```

In [46]:
```
# print(classification_report(y_test, predictions))
```

In [47]:
```
# print(accuracy_score(y_test, predictions))
```

In [48]:
```
# print(confusion_matrix(y_test, predictions))
```

In [51]:
```
print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	35550
1.0	0.99	0.99	0.99	2620
2.0	0.90	0.86	0.88	268
3.0	0.86	0.92	0.89	249
4.0	0.87	0.85	0.86	194
5.0	0.86	0.87	0.86	189
6.0	0.84	0.88	0.86	175
7.0	0.84	0.80	0.82	164
8.0	0.85	0.81	0.83	187
9.0	0.81	0.83	0.82	155
10.0	0.80	0.79	0.79	166
11.0	0.78	0.76	0.77	141
12.0	0.83	0.76	0.79	183
13.0	0.72	0.82	0.77	141
14.0	0.78	0.76	0.77	160
15.0	0.78	0.81	0.80	168
16.0	0.79	0.80	0.80	157
17.0	0.80	0.81	0.81	197
18.0	0.75	0.78	0.77	171
19.0	0.77	0.73	0.75	165
20.0	0.77	0.77	0.77	176
21.0	0.83	0.80	0.81	157
22.0	0.78	0.84	0.81	158
23.0	0.82	0.78	0.80	157
24.0	0.75	0.77	0.76	159
25.0	0.73	0.78	0.75	155
26.0	0.79	0.85	0.82	180
27.0	0.83	0.83	0.83	203
28.0	0.80	0.84	0.82	173
29.0	0.81	0.75	0.78	214
30.0	0.78	0.79	0.79	185
31.0	0.82	0.75	0.78	181
32.0	0.87	0.86	0.86	196
33.0	0.82	0.83	0.83	193
34.0	0.80	0.79	0.80	218
35.0	0.83	0.83	0.83	218
36.0	0.77	0.84	0.81	192
37.0	0.84	0.85	0.84	203
38.0	0.81	0.78	0.79	201
39.0	0.78	0.75	0.76	208
40.0	0.82	0.80	0.81	200
41.0	0.85	0.86	0.85	217
42.0	0.79	0.85	0.82	201
43.0	0.83	0.77	0.80	216
44.0	0.82	0.84	0.83	254
45.0	0.83	0.87	0.85	253
46.0	0.84	0.80	0.82	242
47.0	0.84	0.86	0.85	228
48.0	0.88	0.87	0.87	254
49.0	0.85	0.87	0.86	248
50.0	0.87	0.87	0.87	252
51.0	0.79	0.81	0.80	226
52.0	0.83	0.85	0.84	233
53.0	0.86	0.83	0.84	265
54.0	0.85	0.85	0.85	220

55.0	0.82	0.83	0.83	252
56.0	0.87	0.83	0.85	249
57.0	0.84	0.87	0.86	296
58.0	0.84	0.85	0.84	285
59.0	0.84	0.86	0.85	264
60.0	0.88	0.83	0.85	296
61.0	0.86	0.87	0.86	260
62.0	0.86	0.86	0.86	285
63.0	0.85	0.85	0.85	256
64.0	0.84	0.85	0.85	245
65.0	0.92	0.89	0.91	290
66.0	0.82	0.81	0.82	259
67.0	0.82	0.89	0.86	246
68.0	0.88	0.85	0.87	320
69.0	0.88	0.86	0.87	297
70.0	0.88	0.90	0.89	372
71.0	0.85	0.84	0.84	292
72.0	0.83	0.86	0.85	322
73.0	0.88	0.84	0.86	322
74.0	0.87	0.92	0.89	418
75.0	0.88	0.86	0.87	393
76.0	0.85	0.86	0.85	355
77.0	0.86	0.86	0.86	416
78.0	0.88	0.86	0.87	444
79.0	0.88	0.90	0.89	465
80.0	0.89	0.88	0.88	484
81.0	0.90	0.87	0.88	479
82.0	0.87	0.89	0.88	431
83.0	0.86	0.85	0.86	460
84.0	0.85	0.91	0.88	541
85.0	0.89	0.84	0.86	521
86.0	0.86	0.85	0.85	521
87.0	0.87	0.88	0.87	528
88.0	0.88	0.85	0.86	515
89.0	0.86	0.87	0.86	559
90.0	0.87	0.87	0.87	524
91.0	0.88	0.87	0.87	556
92.0	0.83	0.87	0.85	602
93.0	0.84	0.83	0.83	609
94.0	0.89	0.86	0.87	661
95.0	0.89	0.88	0.88	622
96.0	0.89	0.90	0.89	635
97.0	0.89	0.91	0.90	761
98.0	0.90	0.91	0.91	803
99.0	0.86	0.89	0.87	597
100.0	0.88	0.86	0.87	644
101.0	0.88	0.84	0.86	682
102.0	0.86	0.86	0.86	611
103.0	0.86	0.88	0.87	628
104.0	0.87	0.86	0.87	635
105.0	0.85	0.85	0.85	610
106.0	0.86	0.84	0.85	675
107.0	0.84	0.86	0.85	607
108.0	0.87	0.88	0.88	677
109.0	0.89	0.86	0.88	715
110.0	0.86	0.89	0.87	644
111.0	0.88	0.85	0.87	736

112.0	0.87	0.87	0.87	664
113.0	0.88	0.87	0.87	673
114.0	0.90	0.89	0.89	724
115.0	0.85	0.88	0.87	628
116.0	0.87	0.91	0.89	704
117.0	0.86	0.88	0.87	614
118.0	0.88	0.88	0.88	685
119.0	0.91	0.89	0.90	723
120.0	0.86	0.87	0.87	705
121.0	0.87	0.88	0.88	702
122.0	0.90	0.86	0.88	745
123.0	0.86	0.88	0.87	740
124.0	0.88	0.87	0.88	725
125.0	0.91	0.90	0.90	924
126.0	0.92	0.91	0.91	947
127.0	0.91	0.91	0.91	800
128.0	0.89	0.91	0.90	830
129.0	0.91	0.90	0.90	883
130.0	0.92	0.92	0.92	1117
131.0	0.88	0.92	0.90	699
132.0	0.92	0.89	0.90	924
133.0	0.90	0.92	0.91	936
134.0	0.92	0.89	0.90	933
135.0	0.95	0.96	0.96	1969
136.0	0.92	0.91	0.92	1033
137.0	0.96	0.96	0.96	2061
138.0	0.92	0.95	0.93	1311
139.0	0.98	0.98	0.98	3629
140.0	0.97	0.97	0.97	3236
141.0	0.96	0.95	0.96	2191
142.0	0.98	0.98	0.98	4712
143.0	0.98	0.98	0.98	5834
144.0	0.98	0.98	0.98	4668
145.0	0.93	0.92	0.92	1346
146.0	0.97	0.98	0.98	3841
147.0	0.99	0.99	0.99	6616
148.0	0.99	0.98	0.99	6338
149.0	0.99	0.99	0.99	12493
150.0	0.99	0.99	0.99	8221
151.0	0.99	0.99	0.99	6032
152.0	0.98	0.99	0.99	6134
153.0	0.99	0.99	0.99	9133
154.0	0.99	0.99	0.99	6547
155.0	0.98	0.98	0.98	4433
156.0	0.99	0.99	0.99	8831
157.0	0.99	0.99	0.99	6932
158.0	0.96	0.95	0.95	1737
159.0	0.98	0.99	0.98	5369
160.0	0.97	0.96	0.97	2492
161.0	0.98	0.98	0.98	4125
162.0	0.97	0.97	0.97	2961
163.0	0.96	0.96	0.96	2172
164.0	0.93	0.93	0.93	1435
165.0	0.97	0.96	0.96	2811
166.0	0.90	0.89	0.89	868
167.0	0.93	0.91	0.92	1023
168.0	0.93	0.94	0.93	1104

169.0	0.91	0.92	0.92	950
170.0	0.97	0.96	0.96	1727
171.0	0.90	0.91	0.90	663
172.0	0.88	0.87	0.88	686
173.0	0.88	0.89	0.88	793
174.0	0.89	0.87	0.88	666
175.0	0.85	0.87	0.86	628
176.0	0.87	0.88	0.88	665
177.0	0.89	0.87	0.88	575
178.0	0.88	0.86	0.87	574
179.0	0.85	0.89	0.87	524
180.0	0.84	0.85	0.84	518
181.0	0.86	0.85	0.85	530
182.0	0.86	0.87	0.87	507
183.0	0.89	0.90	0.90	517
184.0	0.86	0.83	0.85	478
185.0	0.88	0.87	0.88	523
186.0	0.85	0.89	0.87	530
187.0	0.87	0.83	0.85	488
188.0	0.83	0.86	0.85	510
189.0	0.88	0.86	0.87	521
190.0	0.87	0.87	0.87	554
191.0	0.85	0.85	0.85	526
192.0	0.85	0.89	0.87	507
193.0	0.89	0.82	0.86	527
194.0	0.83	0.88	0.85	504
195.0	0.82	0.85	0.83	499
196.0	0.87	0.82	0.84	509
197.0	0.83	0.86	0.84	526
198.0	0.86	0.86	0.86	547
199.0	0.88	0.86	0.87	505
200.0	0.89	0.84	0.86	503
201.0	0.83	0.85	0.84	496
202.0	0.87	0.88	0.87	544
203.0	0.85	0.86	0.85	490
204.0	0.88	0.89	0.89	518
205.0	0.88	0.83	0.86	497
206.0	0.85	0.86	0.86	516
207.0	0.83	0.86	0.84	496
208.0	0.82	0.86	0.84	484
209.0	0.86	0.87	0.86	482
210.0	0.88	0.88	0.88	461
211.0	0.86	0.88	0.87	399
212.0	0.89	0.86	0.88	434
213.0	0.84	0.83	0.83	379
214.0	0.84	0.91	0.87	404
215.0	0.87	0.84	0.85	406
216.0	0.85	0.89	0.87	391
217.0	0.89	0.85	0.87	354
218.0	0.88	0.85	0.86	335
219.0	0.87	0.89	0.88	389
220.0	0.86	0.85	0.86	344
221.0	0.90	0.84	0.87	376
222.0	0.83	0.89	0.85	322
223.0	0.90	0.85	0.87	377
224.0	0.83	0.87	0.85	338
225.0	0.86	0.83	0.85	327

226.0	0.84	0.86	0.85	351
227.0	0.85	0.85	0.85	352
228.0	0.88	0.88	0.88	362
229.0	0.89	0.91	0.90	328
230.0	0.88	0.88	0.88	354
231.0	0.89	0.90	0.90	433
232.0	0.92	0.90	0.91	462
233.0	1.00	1.00	1.00	44621
accuracy			0.95	314573
macro avg	0.87	0.87	0.87	314573
weighted avg	0.95	0.95	0.95	314573

In [52]:
```
print('Accuracy: {:.3f}%'.format(accuracy_score(y_test, predictions)*100))
```

```
Accuracy: 95.360%
```

In [53]:
```
cm = confusion_matrix(y_test, predictions)
```

In [54]:
```
# data = confusion_matrix(y_test, predictions)
# df_cm = pd.DataFrame(data, columns=np.unique(y_test), index=np.unique(y_tes
t))
# df_cm.index.name = 'Actual'
# df_cm.columns.name = 'Predicted'
# plt.figure(figsize=(20, 10))
# sns.set(font_scale=0.5) # for label size
# sns.heatmap(df_cm, cmap="Blues", annot=True, annot_kws={"size": 5}) # font s
ize
```

Single Label Classification of Vehicle Position Discretized

Data Processing

Splitting Dataset Into Training and Test Sets

In [55]:
```
X = df.drop('vehicle position', axis=1).values
```

In [56]:
```
''' vehicle_pos_discretized was created above in discretization step '''
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, vehicle_pos_discretized
, test_size=0.3)
```

Feature Scaling (Normalization)

In [57]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Label Encoding

In [58]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
```

Decision Tree Classifier

In [59]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Out[59]:
```
DecisionTreeClassifier()
```

In [60]:
```
predictions = classifier.predict(X_test)
```

In [61]:
```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      6525
           1       0.99      0.99      0.99      7007
           2       0.99      0.99      0.99      8445
           3       0.99      0.99      0.99     16898
           4       1.00      1.00      1.00     55928
           5       1.00      1.00      1.00     57628
           6       0.99      0.99      0.99     19220
           7       0.98      0.98      0.98      6256
           8       1.00      1.00      1.00     68567
           9       1.00      1.00      1.00     68099

    accuracy                           1.00    314573
   macro avg       0.99      0.99      0.99    314573
weighted avg       1.00      1.00      1.00    314573
```

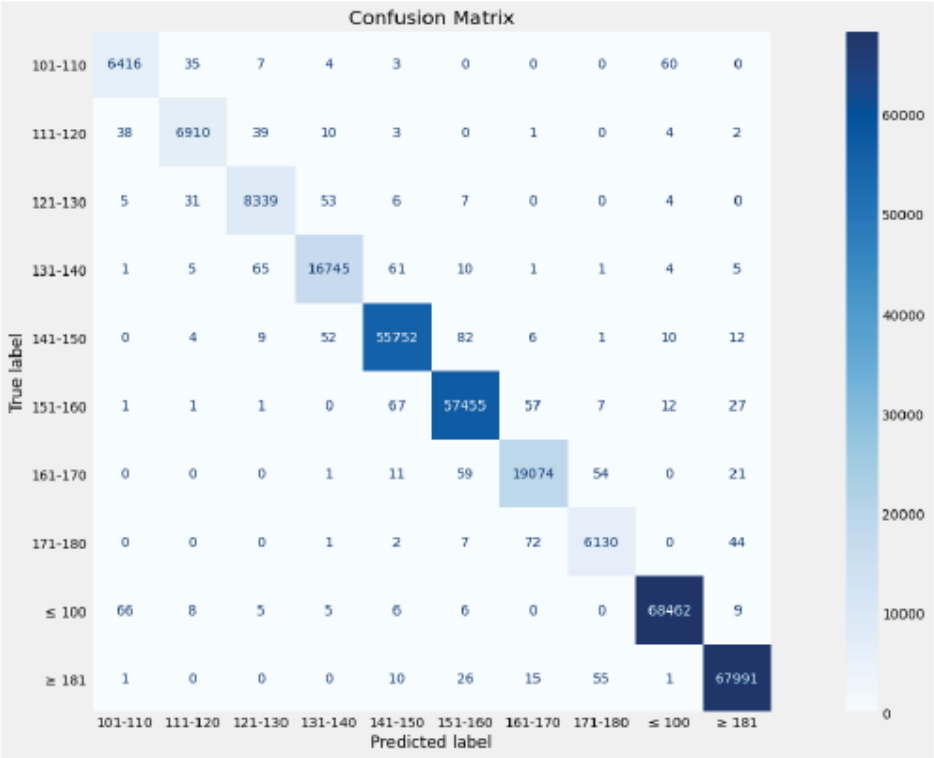
In [62]:
```
print('Accuracy: {:.3f}%'.format(accuracy_score(y_test, predictions)*100))
```

```
Accuracy: 99.587%
```

In [63]:
```
print(confusion_matrix(y_test, predictions))
```

```
[[ 6416    35     7     4     3     0     0     0    60     0]
 [   38  6910    39    10     3     0     1     0     4     2]
 [    5    31  8339    53     6     7     0     0     4     0]
 [    1     5    65 16745    61    10     1     1     4     5]
 [    0     4     9    52 55752    82     6     1    10    12]
 [    1     1     1     0    67 57455    57     7    12    27]
 [    0     0     0     1    11    59 19074    54     0    21]
 [    0     0     0     1     2     7    72  6130     0    44]
 [   66     8     5     5     6     6     0     0 68462     9]
 [    1     0     0     0    10    26    15    55     1 67991]]
```

In [64]:
```
from sklearn.metrics import plot_confusion_matrix
fig, ax = plt.subplots(figsize=(24, 12))
plot_confusion_matrix(classifier, X_test, y_test,
                                  display_labels=np.unique(le.inverse_transform
(predictions)),
                                  cmap=plt.cm.Blues,
                                  normalize=None,
                                    ax=ax)
ax.set_title('Confusion Matrix')
plt.grid(None)
plt.show()
```



Single Label Classification of State Variable

Data Processing

Splitting Dataset Into Training and Test Sets

In [65]:
```
y = df['etat'].values
X = df.drop('etat', axis=1).values
```

In [66]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Feature Scaling (Normalization)

In [67]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Support Vector Machines

In [68]:
```
# from sklearn.svm import SVC
# classifier = SVC()
# classifier.fit(X_train, y_train)
```

In [69]:
```
# predictions = classifier.predict(X_test)
```

In [70]:
```
# print(classification_report(y_test, predictions))
```

In [71]:
```
# print(accuracy_score(y_test, predictions))
```

Naive Bayes

In [72]:
```
# from sklearn.naive_bayes import GaussianNB
# classifier = GaussianNB()
# classifier.fit(X_train, y_train)
```

In [73]:
```
# predictions = classifier.predict(X_test)
```

In [74]:
```
# print(classification_report(y_test, predictions))
```

In [75]:
```
# print(accuracy_score(y_test, predictions))
```

In [76]:
```
# print(confusion_matrix(y_test, predictions))
```

K Nearest Neighbors

In [77]:
```
# from sklearn.neighbors import KNeighborsClassifier
# classifier = KNeighborsClassifier()
# classifier.fit(X_train, y_train)
```

In [78]:
```
# predictions = classifier.predict(X_test)
```

In [79]:
```
# print(classification_report(y_test, predictions))
```

In [80]:
```
# print(accuracy_score(y_test, predictions))
```

In [81]:
```
# print(confusion_matrix(y_test, predictions))
```

Random Forest

In [82]:
```
# from sklearn.ensemble import RandomForestClassifier
# classifier = RandomForestClassifier()
# classifier.fit(X_train, y_train)
```

In [83]:
```
# predictions = classifier.predict(X_test)
```

In [84]:
```
# print(classification_report(y_test, predictions))
```

In [85]:
```
# print(accuracy_score(y_test, predictions))
```

In [86]:
```
# print(confusion_matrix(y_test, predictions))
```

XGBoost

In [87]:
```
# from xgboost import XGBClassifier
# classifier = XGBClassifier()
# classifier.fit(X_train, y_train)
```

In [88]:
```
# predictions = classifier.predict(X_test)
```

In [89]:
```
# print(classification_report(y_test, predictions))
```

In [90]:
```
# print(accuracy_score(y_test, predictions))
```

In [91]:
```
# print(confusion_matrix(y_test, predictions))
```

Decision Tree

In [92]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Out[92]:
```
DecisionTreeClassifier()
```

In [93]:
```
predictions = classifier.predict(X_test)
```

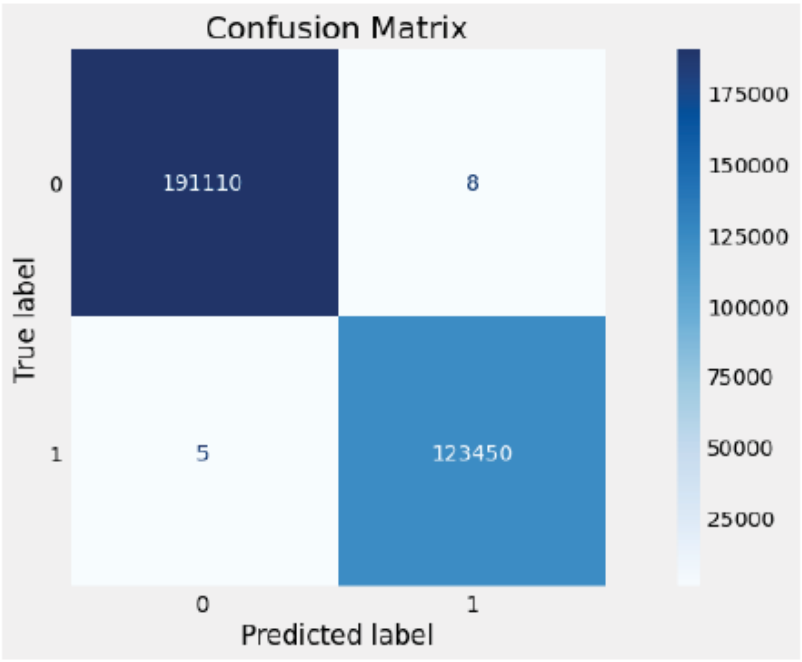
In [94]:
```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    191118
           1       1.00      1.00      1.00    123455

    accuracy                           1.00    314573
   macro avg       1.00      1.00      1.00    314573
weighted avg       1.00      1.00      1.00    314573
```

In [95]:
```
print('Accuracy: {:.3f}%'.format(accuracy_score(y_test, predictions)*100))
```

```
Accuracy: 99.996%
```

In [96]:
```
from sklearn.metrics import plot_confusion_matrix
fig, ax = plt.subplots(figsize=(14, 6))
plot_confusion_matrix(classifier, X_test, y_test,
                                  display_labels=df['etat'].unique(),
                                  cmap=plt.cm.Blues,
                                  normalize=None,
                                    ax=ax)
ax.set_title('Confusion Matrix')
plt.grid(None)
plt.show()
```



Multi Label Classification

Data Preprocessing

Splitting Dataset Into Training and Test Sets

In [97]:
```
X = df.iloc[:, :-2].values
y = df.loc[:, ['vehicle position', 'etat']].values
```

In [98]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [99]:
```
vehicle_y_train, state_y_train = y_train[:, 0], y_train[:, 1]
vehicle_y_test, state_y_test = y_test[:, 0], y_test[:, 1]
```

Feature Scaling (Normalization)

In [100]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Binary Relevance

Model Training

In [101]:
```
classifier_1 = DecisionTreeClassifier()
classifier_1.fit(X_train, vehicle_y_train)
```

Out[101]:
```
DecisionTreeClassifier()
```

In [102]:
```
classifier_2 = DecisionTreeClassifier()
classifier_2.fit(X_train, state_y_train)
```

Out[102]:
```
DecisionTreeClassifier()
```

Model Testing & Evaluation

In [103]:
```
# Concatenating 2 predicted labels into 1 numpy array
y_pred = np.concatenate((np.expand_dims(classifier_1.predict(X_test), axis=1),
                         np.expand_dims(classifier_2.predict(X_test), axis=1
)), axis=1)

# Converting true labels into a numpy array
y_true = np.concatenate((np.expand_dims(np.asarray(vehicle_y_test), axis=1),
                         np.expand_dims(np.asarray(state_y_test), axis=1)), ax
is=1)

# Calculating Exact Match score and the Hamming Loss
EMscore = 0
count_Ham = 0

for i in range(0, y_true.shape[0]):
    count_EM = 0
    for j in range(0, y_true.shape[1]):
        if y_true[i][j] == y_pred[i][j]:
            count_EM += 1
        else:
            count_Ham += 1
    if count_EM == 2:
        EMscore += 1

EMscore = EMscore / y_true.shape[0]
Hamming_Loss = count_Ham / (y_true.shape[0] * y_true.shape[1])

print("Using Binary Relevance With Decision Tree Classifier:", "\n")
print("Exact Match score: {:.3f}%".format(EMscore*100))
print("Hamming Loss: {:.3f}".format(Hamming_Loss))
```

```
Using Binary Relevance With Decision Tree Classifier:

Exact Match score: 94.803%
Hamming Loss: 0.026
```

Classifier Chains

Model Training

In [104]:
```
classifier = DecisionTreeClassifier()
classifier.fit(X_train, vehicle_y_train)
```

Out[104]:
```
DecisionTreeClassifier()
```

In [105]:
```
print("For the label 'Vehicle Position', we have:","\n")
print("The accuracy of the model on the training set is: {:.2f}%".format(class
ifier.score(X_train, vehicle_y_train)*100))
print("The accuracy of the model on the testing set is: {:.2f}%".format(classi
fier.score(X_test, vehicle_y_test)*100))
```

```
For the label 'Vehicle Position', we have:

The accuracy of the model on the training set is: 100.00%
The accuracy of the model on the testing set is: 94.85%
```

In [106]:
```
X_train_1 = np.concatenate((X_train, np.expand_dims(vehicle_y_train, axis=1)),
axis=1)
X_train_augment_1 = pd.DataFrame(data=X_train_1, index=None, columns=None)
```

In [107]:
```
classifier = DecisionTreeClassifier()
classifier.fit(X_train_augment_1, state_y_train)
```

Out[107]:
```
DecisionTreeClassifier()
```

In [108]:
```
print("For the label 'State Variable', we have:","\n")
print("The accuracy of the model on the training set is: {:.3f}%".format(class
ifier.score(X_train_augment_1, state_y_train)*100))
```

```
For the label 'State Variable', we have:

The accuracy of the model on the training set is: 100.000%
```

In [109]:
```
classifier_1 = DecisionTreeClassifier()
classifier_1.fit(X_train, vehicle_y_train)
y1_pred = classifier_1.predict(X_test)
```

In [110]:
```
X_test_1 = np.concatenate((X_test, np.expand_dims(y1_pred, axis=1)), axis=1)
X_test_augment_1 = pd.DataFrame(data=X_test_1,index=None, columns=None)
```

In [111]:
```
classifier_2 = DecisionTreeClassifier()
classifier_2.fit(X_train_augment_1, state_y_train)
y2_pred = classifier_2.predict(X_test_augment_1)
```

Model Testing & Evaluation

In [112]:
```
# Concatenating 2 predicted labels into 1 numpy array
y_pred_3 = np.concatenate((np.expand_dims(classifier_1.predict(X_test), axis=1
),
                           np.expand_dims(classifier_2.predict(X_test_augment_1
), axis=1)), axis=1)

# Converting true labels into a numpy array
y_true = np.concatenate((np.expand_dims(np.asarray(vehicle_y_test), axis=1),
                           np.expand_dims(np.asarray(state_y_test), axis=1)), ax
is=1)

# Calculating Exact Match score and Hamming Loss
EMscore = 0
count_Ham = 0
for i in range(0,y_true.shape[0]):
    count_EM = 0
    for j in range(0,y_true.shape[1]):
        if y_true[i][j]==y_pred_3[i][j]:
            count_EM += 1
        else:
            count_Ham += 1
    if count_EM == 2:
        EMscore += 1
EMscore = EMscore / y_true.shape[0]
Hamming_Loss = count_Ham / (y_true.shape[0] * y_true.shape[1])

print("Using Classifier Chain Method With DecisionTreeClassifier:", "\n")
print("Exact Match Score: {:.3f}%".format(EMscore*100))
print("Hamming Loss: {:.3f}".format(Hamming_Loss))
```

```
Using Classifier Chain Method With DecisionTreeClassifier:

Exact Match Score: 94.842%
Hamming Loss: 0.026
```