haute école
neuchâtel berne jura

ingénierie
www.he-arc.ch

Bachelor of Science HE-ARC in Engineering

Orientation : Ingénierie des données (ID)

P3 204 CC

Doppler : exploration, analyse et prédiction de séries temporelles médicales

Réalisé par :
Firas Dridi

Sous la direction de :
Fabrizio Albertetti
Haute École Arc, HES-SO

26 janvier 2024

Table des matières

Introduction

Le domaine de la santé numérique, en pleine expansion, bénéficie directement des progrès technologiques pour affiner les diagnostics et améliorer les traitements. Dans ce contexte, l'analyse des séries temporelles médicales se distingue par sa capacité à éclairer les processus biologiques et les mécanismes des maladies.

Ce projet, intégré au programme de Bachelor en Ingénierie des Données de la Haute École Arc, vise à explorer, analyser et prédire ces séries temporelles, avec une attention particulière aux données du Doppler transcrânien, sous la supervision de Fabrizio Albertetti, Ph.D.

Le projet se fixe deux objectifs principaux : d'abord, il entend démontrer les possibilités et les limites de l'analyse des séries temporelles médicales en s'appuyant sur des méthodes d'exploration et de modélisation pointues. Ensuite, il cherche à illustrer la manière dont ces techniques peuvent être adaptées à d'autres contextes médicaux, élargissant ainsi leur champ d'application dans la santé numérique.

Cette initiative représente une occasion précieuse d'appliquer et d'approfondir mes compétences en ingénierie des données, marquant la synthèse de trois années d'études et la transition vers le secteur professionnel.

1 État de l'Art et Analyse

1.1 Revue de l'Art

La recherche actuelle met en lumière l'efficacité des méthodes de machine learning pour traiter les données médicales, notamment celles issues du Doppler transcrânien. Ces études révèlent le potentiel de ces outils pour détecter précocement les troubles neurovasculaires.

Cependant, l'application pratique de ces découvertes reste complexe, notamment à cause de la grande variabilité des données et des particularités de chaque cas clinique.

1.2 Analyse Critique

Bien que prometteuse, l'analyse du Doppler transcrânien rencontre plusieurs défis. La première difficulté réside dans la variabilité des données, influencée par les conditions d'examen, l'expérience de l'opérateur et les particularités physiologiques des patients. Par ailleurs, l'analyse de séries temporelles médicales, par leur complexité et multidimensionnalité, nécessite des méthodes avancées et flexibles. Enfin, l'exploitation clinique des modèles prédictifs exige une collaboration étroite entre experts en données et professionnels de santé pour garantir une interprétation adéquate des résultats.

Ce projet s'attache à surmonter ces obstacles grâce à une démarche méthodique, de l'exploration des données à la validation des modèles prédictifs. En mobilisant des stratégies d'analyse de données sophistiquées et en considérant les spécificités du milieu médical, ce travail ambitionne de renforcer les outils de support à la décision en neurologie et au-delà.

2 Exploration et Analyse des Données

2.1 Identification des Datasets

La quête d'un dataset approprié pour un projet dans le cadre de la Data Science, en particulier dans le domaine médical, présente des défis considérables en termes de pertinence, de densité des données et de conformité éthique.

Cette recherche s'est principalement orientée vers des plateformes réputées telles que Kaggle et IEEE, reconnues pour leur richesse en datasets variés et leur fiabilité. L'idée de solliciter directement des hôpitaux a été envisagée, mais cette voie s'est rapidement heurtée à des obstacles réglementaires et éthiques non négligeables, notamment en ce qui concerne la protection des données patients et l'obtention des consentements nécessaires.

Dans ce contexte complexe, la découverte d'un dataset de Doppler transcrânien sur le site de l'IEEE a été une occasion de manipuler des données réalistes et médicales.

Ce dataset se distinguait par plusieurs aspects : il était prêt à l'emploi, structuré de manière intuitive avec des fichiers textes contenant les colonnes t, I et Q (représentant respectivement le temps et les composantes du signal Doppler en coordonnées polaires), et accompagné de données nettoyées et organisées.

De plus, le dataset offrait la possibilité de visualiser les spectrogrammes des patients à l'aide de scripts MATLAB, offrant ainsi une approche directe et visuelle de l'analyse préliminaire des données.

Enfin, les données de Doppler transcrânien ont été recueillies auprès de volontaires sains du MIT et de patients en soins neurocritiques du BMC entre 2016 et 2020, avec l'approbation des comités d'éthique et le consentement éclairé des participants. Le dataset comprend 16 enregistrements de sujets sains et 29 de patients en soins neurocritiques.

2.2 Exploration de Données

L'exploration des données du Doppler transcrânien a constitué une phase fondamentale de ce projet, marquée par une série de défis techniques et conceptuels. D'emblée, la nature des données, encapsulant des informations complexes sur le flux sanguin cérébral, a exigé une compréhension approfondie des principes sous-jacents de la mesure Doppler et de sa pertinence clinique.

L'un des premiers obstacles a été la structuration des données elles-mêmes, organisées en séries temporelles avec des variables I et Q représentant des composants de signaux Doppler en coordonnées polaires. La transformation de ces signaux en une forme exploitable pour l'analyse a nécessité une familiarisation avec des concepts spécifiques au domaine de l'ultrasonographie et une adaptation des méthodes d'analyse de signal conventionnelles.

Table 1 – Exemple de données issues du Doppler transcrânien (.txt).

t	I	Q
881753582	-831302	732068
881753726	-830206	734614
881753870	-830424	734636
881754014	-828761	735278
881754158	-828064	737322
881754302	-825612	739111
881754446	-828383	743804

2.2.1 Temps

Comme on peut le voir ci-dessus, la colonne "t" qui représente le temps, a posé un défi particulier en raison de son format initial en microsecondes, nécessitant une conversion précise en secondes pour une interprétation correcte. Les premières tentatives d'interprétation directe des données temporelles ont conduit à des problèmes de gestion de la mémoire et à des difficultés dans la représentation graphique des données. D'après les scripts Matlab, ce traitement s'effectue selon l'équation suivante :

$$tEcho\_seconds = \frac{tEcho}{1e6}$$

où $tEcho$ représente le temps brut. Par la suite, la fréquence d'échantillonnage $fsEcho$ est déduite comme l'inverse de la médiane des différences temporelles successives, facilitant l'analyse spectrale ultérieure.

2.2.2 I/Q Signal

Les colonnes I et Q représentent les composantes réelles et imaginaires des signaux Doppler, formant ensemble un signal complexe (IQ). Ce signal complexe est essentiel pour capturer à la fois l'amplitude et la phase des ondes ultrasonores réfléchies par le flux sanguin cérébral. Les signaux en quadrature, sont un concept fondamental en traitement du signal, particulièrement dans les domaines de la communication et de la mesure.

Les deux étant orthogonaux l'un par rapport à l'autre, permettant ainsi une représentation en deux dimensions du signal. Cette représentation bidimensionnelle est essentielle pour la modulation et la démodulation des signaux dans les systèmes de communication, car elle permet de transporter plus d'informations et d'améliorer l'efficacité spectrale.

$$\cos(2\pi ft + \varphi(t)) = \cos(2\pi ft)\cos(\varphi(t)) + \cos\left(2\pi ft + \frac{\pi}{2}\right)\sin(\varphi(t)) \tag{1}$$

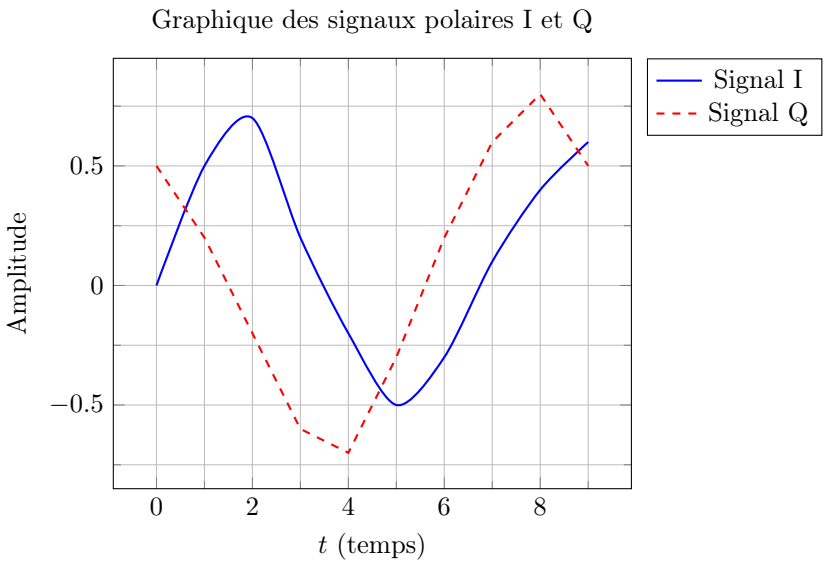

FIGURE 1 – Exemple de graphique des signaux polaires I et Q.

Le calcul de la résultante des signaux I et Q est crucial car il fournit l'amplitude et la phase du signal original, ce qui est souvent nécessaire dans l'analyse des signaux.

En combinant ces deux composants, on obtient une vue complète du signal, incluant à la fois sa magnitude et sa phase. Cette approche permet d'extraire des caractéristiques significatives des signaux, telles que la vitesse du flux sanguin dans le cerveau, qui sont essentielles pour le diagnostic et le suivi des pathologies neurovasculaires.

Dans notre cas, d'autres paramètres ont été pris en considération afin d'avoir une génération de spectrogramme qui suit fidèlement les scripts originaux :

1. Filtrage du signal : Un filtre passe-haut est appliqué au signal complexe pour éliminer les fréquences indésirables et conserver les informations pertinentes. Le choix de la fréquence de coupure (fwc_normalized) est adapté à la nature des données Doppler.
2. Génération du spectrogramme : Le signal filtré est ensuite utilisé pour générer un spectrogramme, qui décompose le signal en ses composantes fréquentielles au fil du temps. Cette représentation fournit une vue d'ensemble riche des variations de vitesse du flux sanguin dans le cerveau.

Les premières tentatives de génération n'incluaient pas le filtrage du signal, ce qui induisait inévitablement du bruit au sein des spectrogrammes :

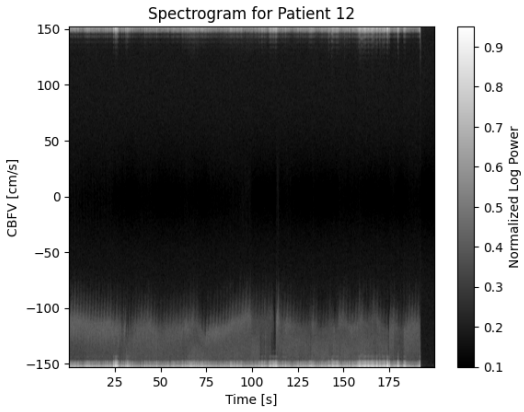

FIGURE 2 – Sans filtrage

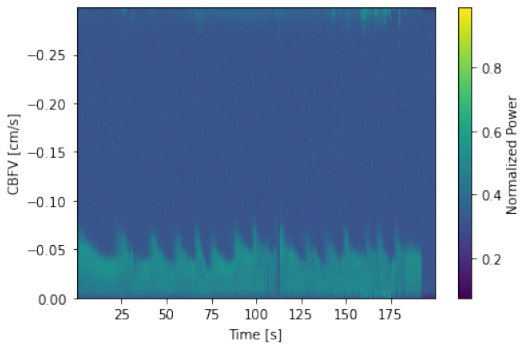

FIGURE 3 – Avec filtrage

2.2.3 Extraction et lissage de l'enveloppe supérieure

L'enveloppe supérieure du spectrogramme est extraite en identifiant les valeurs maximales de puissance spectrale (SP_filt) pour chaque instant temporel. Cette enveloppe est ensuite lissée pour obtenir une courbe continue et représentative de la vitesse maximale du flux sanguin, un indicateur clé de l'état neurovasculaire du sujet.

```
# Extraction de l'enveloppe superieure
        envelope = np.max(SP_filt, axis=0)

        # Lissage de l'enveloppe
        envelope_smoothed = median_filter(envelope, size=5)

        # Gestion des valeurs vides
        envelope_smoothed[np.isnan(envelope_smoothed)] = 0

        # Creation du DataFrame
        df = pd.DataFrame({
            'ID_State': id_state,
            'time': tSpectrogram,
            'envelope_value': envelope_smoothed,
            'numero_patient': patient_number,
            'State': state
        })
```

Listing 1 – Calcul de l'enveloppe supérieure

L'enveloppe supérieure, que nous avons calculée à partir des spectrogrammes, joue un rôle crucial dans notre analyse, car elle est traitée comme une série temporelle distincte. Cette approche ne se contente pas de capturer les attributs clés des spectrogrammes, mais elle facilite également la manipulation des données au sein d'un dataframe, optimisant de ce fait l'efficacité de notre code.

Il convient de noter que, bien que notre méthode actuelle de calcul de l'enveloppe offre déjà des résultats significatifs, expliqués prochainement, elle présente un potentiel d'amélioration. Avec plus de temps et de ressources, nous pourrions affiner cette technique pour obtenir une représentation encore plus fidèle des données du spectrogramme, ce qui pourrait à son tour enrichir davantage notre analyse.

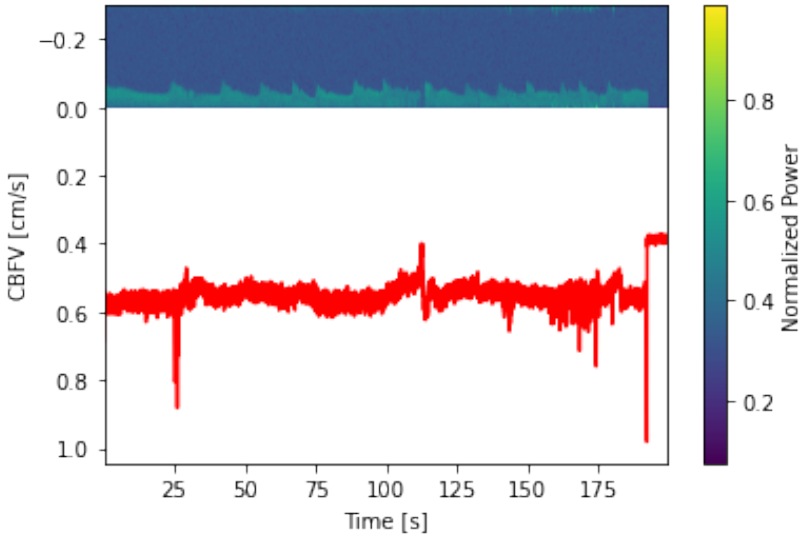

FIGURE 4 – Spectrogramme & enveloppe

2.3 Analyse de Données

Le cœur de ce projet réside dans l'automatisation de l'analyse des données à travers l'utilisation de bibliothèques spécialisées. L'objectif est d'extraire de manière efficiente des caractéristiques pertinentes des séries temporelles, qui sont cruciales pour la construction de modèles prédictifs fiables.

Pour ce faire, trois outils principaux ont été exploités : TSFresh, Featuretools et TSFEL. Chaque bibliothèque offre une approche unique pour l'ingénierie des caractéristiques, permettant ainsi une exploration approfondie et une comparaison de leurs performances respectives ainsi qu'un gain de temps considérable.

2.3.1 TSFresh

TSFresh (Time Series Feature Extraction based on Scalable Hypothesis tests) est une bibliothèque Python influente dans le domaine de l'analyse de données de séries temporelles. Elle automatise l'extraction de caractéristiques, facilitant ainsi la tâche d'ingénierie des caractéristiques. L'un des atouts majeurs de TSFresh est sa capacité à générer un grand nombre de caractéristiques et à évaluer leur pertinence statistique.

Dans le cadre de ce projet, TSFresh a été employée pour extraire des caractéristiques des données Doppler transcrâniennes, en utilisant spécifiquement le jeu de paramètres MinimalFCParameters. Cette configuration a été choisie pour sa simplicité et son efficacité, offrant un compromis optimal entre la quantité de caractéristiques générées et la performance de calcul. L'objectif était de minimiser le risque d'overlifting tout en conservant les informations essentielles contenues dans les données.

```
def extract_features_ts_fresh(input_df, id_col, time_col, value_col,
    status_col, output_folder):
    """
    Extrait les caracteristiques avec TS-Fresh, ajoute la colonne
        Status, et enregistre le resultat dans un fichier CSV.

    :param input_df: DataFrame contenant les donnees pour l'extraction
        .
    :param id_col: Nom de la colonne d'identifiant dans le DataFrame.
    :param time_col: Nom de la colonne de temps dans le DataFrame.
    :param value_col: Nom de la colonne de valeur dans le DataFrame.
    :param status_col: Nom de la colonne de statut dans le DataFrame.
    :param output_folder: Dossier ou le fichier CSV resultant sera
        enregistre.
    :return: DataFrame des caracteristiques extraites avec la colonne
        Status.
    """
    # Parametres d'extraction
    extraction_settings = MinimalFCParameters()

    # Preparation des donnees pour TS-Fresh
    df_for_tsfresh = input_df[[id_col, time_col, value_col]]

    # Extraction des caracteristiques
    extracted_features = extract_features(df_for_tsfresh,
                                          column_id=id_col,
                                          column_sort=time_col,
```

```
                                                default_fc_parameters=
                                                    extraction_settings,
                                                n_jobs=0)

    # Ajout de la colonne Status au DataFrame extrait
    status_df = input_df[[id_col, status_col]].drop_duplicates().
        set_index(id_col)
    extracted_features = extracted_features.join(status_df, how='left'
        )

    # Enregistrement des resultats
    output_file = os.path.join(output_folder, "ts_fresh_features.csv")
    extracted_features.to_csv(output_file)

    return extracted_features
```

Listing 2 – Fonction d'extraction - TSFresh

Le code présenté ci-dessus illustre le processus d'extraction des caractéristiques avec TSFresh. La fonction extract_features_ts_fresh prend en entrée un DataFrame contenant les données Doppler (identifiant, temps, valeur du signal et statut de santé), ainsi que le dossier de destination pour les résultats. Le cœur de cette fonction réside dans l'appel à la méthode extract_features de TSFresh, qui traite le DataFrame préparé (df_for_tsfresh) en utilisant les MinimalFCParameters. Ces paramètres définissent un ensemble restreint mais significatif de calculs pour extraire des caractéristiques, réduisant ainsi la dimensionnalité des données tout en préservant les aspects cruciaux pour la prédiction.

Après l'extraction, la colonne Status est rajoutée au DataFrame des caractéristiques extraites, permettant ainsi de conserver l'information relative à l'état de santé associé à chaque enregistrement. Ce processus sera répété pour toutes les librairies similaires.

Finalement, les caractéristiques extraites, enrichies de la colonne Status, sont sauvegardées dans un fichier CSV pour une utilisation ultérieure, comme l'entraînement de modèles prédictifs.

2.3.2 Featuretools

Featuretools se distingue par sa capacité à générer automatiquement de nouvelles caractéristiques à travers des opérations d'agrégation et de transformation, en exploitant les relations hiérarchiques entre les données. Elle repose sur le concept d'ensemble d'entités (EntitySet) pour modéliser les relations complexes entre les données.

```
def extract_features_featuretools(input_df, id_col, time_col,
    value_col, status_col, output_folder):
    es = ft.EntitySet(id='Time_Series')
    es = es.add_dataframe(dataframe_name='data', dataframe=input_df,
                          index='index', time_index=time_col,
                          logical_types={id_col: 'Categorical',
                              time_col: 'Datetime'})

    es.normalize_dataframe(base_dataframe_name='data',
        new_dataframe_name=id_col, index=id_col)

```

```
    # Mise a jour des primitives d'agr gation en fonction de celles disponibles
    agg_primitives = ["count", "max", "min", "mean", "sum", "std", "median", "skew", "variance"]
    trans_primitives = ["month", "weekday", "day", "year", "is_weekend"]

    feature_matrix, feature_defs = ft.dfs(entityset=es, target_dataframe_name=id_col,
                                          agg_primitives=agg_primitives,
                                          trans_primitives=trans_primitives,
                                          max_depth=2, verbose=True)

    # Joindre la colonne Status
    status_df = input_df[[id_col, status_col]].drop_duplicates().set_index(id_col)
    feature_matrix = feature_matrix.join(status_df, how='left')

    output_file = os.path.join(output_folder, "featuretools_features.csv")
    feature_matrix.to_csv(output_file)

    return feature_matrix
```

Listing 3 – Fonction d'extraction - Featuretools

La fonction extract_features_featuretools initialise un ensemble d'entités avec les données d'entrée, en définissant une entité principale nommée 'data' avec un index et un index temporel. Le système d'entités est ensuite normalisé pour créer une nouvelle entité basée sur l'identifiant du patient, permettant ainsi d'appliquer des primitives d'agrégation et de transformation.

Les primitives d'agrégation (agg_primitives) et de transformation (trans_primitives) sélectionnées sont spécifiées pour extraire des caractéristiques statistiques et temporelles pertinentes. Les primitives d'agrégation telles que "count", "max", "min", "mean" permettent de résumer des informations clés à partir des séries temporelles.
La fonction ft.dfs (Deep Feature Synthesis) est ensuite appelée pour générer automatiquement le jeu de caractéristiques à partir de l'ensemble d'entités configuré, ciblant l'entité définie par l'identifiant du patient.

2.3.3 TSFEL

TSFEL offre une large gamme de fonctionnalités statistiques, spectrales et temporelles prêtes à l'emploi, permettant une analyse approfondie et une préparation efficace des données pour des modèles d'apprentissage automatique.

```
def extract_features_tsfel(input_df, id_col, time_col, value_col,
    status_col, output_folder):
    cfg = tsfel.get_features_by_domain()

    # Initialiser un DataFrame pour stocker les resultats
    extracted_features = pd.DataFrame()

    for state in input_df[id_col].unique():
        # Selectionner les donnees pour l'ID_State actuel
        temp_df = input_df[input_df[id_col] == state]

        # Extrait les caracteristiques TSFEL pour les valeurs
            numeriques
        temp_features = tsfel.time_series_features_extractor(cfg,
            temp_df[value_col].values, fs=1)

        # Convertir les caracteristiques en DataFrame et ajouter les
            colonnes ID_State et Status
        temp_features_df = pd.DataFrame(temp_features, index=[0])
        temp_features_df[id_col] = state
        temp_features_df[status_col] = temp_df[status_col].iloc[0]

        # Concatener avec le DataFrame global
        extracted_features = pd.concat([extracted_features,
            temp_features_df])

    # Enregistrement des resultats
    output_file = os.path.join(output_folder, "tsfel_features.csv")
    extracted_features.to_csv(output_file, index=False)

    return extracted_features
```

Listing 4 – Fonction d'extraction - TSFEL

La fonction extract_features_tsfel commence par récupérer la configuration par défaut des fonctionnalités fournies par TSFEL, en utilisant tsfel.get_features_by_domain(). Cette configuration inclut un ensemble diversifié de descripteurs calculés dans différents domaines (statistique, spectral, temporel), offrant une vision complète des propriétés des séries temporelles.

Pour chaque état unique dans les données (id_col), la fonction sélectionne les observations correspondantes et applique TSFEL pour extraire les caractéristiques de la colonne spécifiée (value_col), qui représente les valeurs de l'enveloppe supérieure dans ce contexte. TSFEL traite ces valeurs comme une série temporelle et calcule un ensemble de caractéristiques définies par la configuration préalablement chargée.

2.3.4 features extraction

La fonction prepare_data_and_extract_features joue un rôle central dans le processus d'analyse des données Doppler, orchestrant la préparation des données et leur transformation à l'aide des bibliothèques d'extraction de caractéristiques comme TSFresh, Featuretools, et TSFEL. Son objectif est de simplifier et d'automatiser l'extraction de caractéristiques à partir des données d'enveloppe supérieure calculées précédemment, facilitant ainsi la comparaison des performances des différentes bibliothèques et la sélection des caractéristiques les plus pertinentes pour la modélisation.

```
def prepare_data_and_extract_features(processed_folder, interim_folder
    , csv_file_name, extract_functions):
    """
    Prepare les donnees et extrait les caracteristiques en utilisant
       les fonctions fournies.

    :param processed_folder: Chemin du dossier contenant le fichier
       CSV de depart.
    :param interim_folder: Chemin du dossier pour sauvegarder les
       resultats des extractions.
    :param csv_file_name: Nom du fichier CSV a lire.
    :param extract_functions: Dictionnaire des fonctions d'extraction
       a appliquer.
    """

    # Lecture du fichier CSV
    df_train = pd.read_csv(os.path.join(processed_folder,
       csv_file_name))

    # Ajouter la colonne 'Status' si necessaire
    if 'Status' not in df_train.columns:
        df_train['Status'] = df_train['ID_State'].apply(lambda x: "
           Unhealthy" if "unhealthy" in x.lower() else "Healthy")

    # Appel des fonctions d'extraction et affichage des premieres
       lignes des resultats
    for func_name, func in extract_functions.items():
        extracted_df = func(df_train, 'ID_State', 'time', '
           envelope_value', 'Status', interim_folder)
        print(f"\nResultat pour {func_name}:\n", extracted_df.head())
```

Listing 5 – Fonction d'extraction

La fonction commence par charger les données d'entraînement à partir d'un fichier CSV spécifié, situé dans le dossier processed_folder. Ces données contiennent les mesures d'enveloppe supérieure pour chaque patient, identifiées par ID_State, ainsi que les instants de temps (time) et les valeurs d'enveloppe (envelope_value).

Un point clé de cette fonction est l'ajout de la colonne 'Status', qui n'est pas toujours présente dans les données initiales. Cette colonne est dérivée de ID_State et indique l'état de santé du patient (sain ou pathologique), en fonction de la présence du terme "unhealthy" dans l'identifiant. Cette étape est essentielle pour étiqueter les données, permettant ainsi de réaliser des analyses supervisées par la suite.

La fonction itère ensuite sur un dictionnaire extract_functions, qui mappe le nom de chaque bibliothèque d'extraction de caractéristiques à la fonction d'extraction correspondante. Pour chaque bibliothèque, la fonction d'extraction appropriée est appelée avec le DataFrame des données d'entraînement, les noms des colonnes pertinentes (ID_State, time, envelope_value, Status), et le chemin vers le dossier interim_folder où les résultats seront sauvegardés.

3 Développement de Modèles Prédictifs

3.1 Construction des Modèles

Pour ce projet, nous avons privilégié une approche pragmatique en sélectionnant des modèles de machine learning bien établis et adaptés à la nature de nos données, en plus d'avoir eu l'expérience de les utiliser, notamment lors du projet P2.

Les modèles choisis sont :

1. **SVM (Support Vector Machine) :** Un modèle puissant pour la classification, connu pour sa capacité à gérer des espaces de caractéristiques de grande dimension et à effectuer une séparation linéaire efficace des classes dans cet espace.
2. **Perceptron :** Un des plus anciens algorithmes de classification, utilisé ici pour sa simplicité et son efficacité dans les tâches de classification binaire.
3. **Random Forest :** Un modèle ensembliste qui construit une multitude d'arbres de décision lors de l'entraînement et produit la classe qui est le mode des classes des arbres individuels. Il est réputé pour sa robustesse et sa capacité à éviter le surajustement.

Afin de systématiser le processus d'entraînement et d'évaluation des modèles, un module Python dédié nommé model_training.py a été développé. Ce module encapsule la logique nécessaire pour former les modèles sur l'ensemble de données d'entraînement et les sauvegarder pour une utilisation ultérieure sur les données de test. Les modèles sont sauvegardés au format pickle, ce qui facilite leur réutilisation sans nécessiter de reformation. La construction est la suivante :

1. **Prétraitement :** Les données sont standardisées à l'aide de StandardScaler pour garantir que les caractéristiques sont sur une échelle comparable. Cela est particulièrement important pour des modèles comme le SVM qui sont sensibles à l'échelle des caractéristiques d'entrée.
2. **Formation du modèle :** Formation des modèles cités précédemment.

3.2 Optimisation des Modèles et Évaluation

3.2.1 Optimisation

Une pipeline a été mise en place afin de déterminer le meilleur modèle pour une combinaison donnée, comme en témoigne le code ci-dessous :

```
1 def train_evaluate_pipeline(data_path, pipeline, params, target_col='
    Status', exclude_cols=None):
2     data = pd.read_csv(data_path)
3 
4     if exclude_cols is not None:
5         X = data.drop(exclude_cols + [target_col], axis=1)
6     else:
7         X = data.drop(target_col, axis=1)
```

```
8
9       y = data[target_col]
10
11      X_train, X_test, y_train, y_test = train_test_split(X, y,
            test_size=0.3, random_state=42)
12
13      grid_search = GridSearchCV(pipeline, params, cv=2, scoring='
            f1_weighted')
14      grid_search.fit(X_train, y_train)
15
16      best_model = grid_search.best_estimator_
17      y_pred = best_model.predict(X_test)
18
19      f1 = f1_score(y_test, y_pred, average='weighted')
20      conf_matrix = confusion_matrix(y_test, y_pred)
21      class_report = classification_report(y_test, y_pred)
22
23      return f1, conf_matrix, class_report, best_model
24
25  def train_and_evaluate_all_models(lib_timeseries_csv_folder,
        model_folder, exclude_columns):
26      # Configuration des mod les
27      models_params = {
28          'SVC': {
29              'pipeline': Pipeline([('scaler', StandardScaler()), ('svc'
                    , SVC())]),
30              'params': {'svc__C': [0.1, 1, 10], 'svc__kernel': ['rbf',
                    'linear']}
31          },
32          'RandomForest': {
33              'pipeline': Pipeline([('rf', RandomForestClassifier())]),
34              'params': {'rf__n_estimators': [10, 50, 100], '
                    rf__max_depth': [None, 5, 10]}
35          },
36          'Perceptron': {
37              'pipeline': Pipeline([('scaler', StandardScaler()), ('
                    perceptron', Perceptron())]),
38              'params': {'perceptron__penalty': [None, 'l2', 'l1'], '
                    perceptron__alpha': [0.0001, 0.001, 0.01]}
39          }
40      }
41
42      # Chemins des fichiers CSV
43      csv_files = {
44          'TS-Fresh': os.path.join(lib_timeseries_csv_folder, '
                ts_fresh_features.csv'),
45          'Featuretools': os.path.join(lib_timeseries_csv_folder, '
                featuretools_features.csv'),
46          'TSFEL': os.path.join(lib_timeseries_csv_folder, '
                tsfel_features.csv')
47      }
48
```

```
    scores = []
    confusion_matrices = {}
    class_reports = {}

    for model_name, model_info in models_params.items():
        for lib_name, file_path in csv_files.items():
            f1, conf_matrix, class_report, best_model =
                train_evaluate_pipeline(
                file_path,
                model_info['pipeline'],
                model_info['params'],
                exclude_cols=exclude_columns[lib_name]
            )
            scores.append({'Librairie': lib_name, 'Modele': model_name
                , 'F1-Score': f1})
            confusion_matrices[f"{model_name}_{lib_name}"] =
                conf_matrix
            class_reports[f"{model_name}_{lib_name}"] = class_report

            # Enregistrement du meilleur modele
            model_filename = os.path.join(model_folder, f"{model_name}
                _{lib_name}.pickle")
            with open(model_filename, 'wb') as file:
                pickle.dump(best_model, file)
```

Listing 6 – Pipeline

Une recherche exhaustive (GridSearchCV) est employée pour chaque modèle afin de déterminer la combinaison d'hyperparamètres qui maximise la performance, mesurée par le F1-score. Cette étape est cruciale pour ajuster les modèles aux spécificités des données.

3.2.2 Évaluation

Après la construction des modèles prédictifs, une phase d'évaluation est entreprise sur les données d'entraînement pour mesurer leur efficacité et leur précision. Cette évaluation se concentre sur diverses métriques, notamment le F1-score, la matrice de confusion et le rapport de classification, permettant une analyse approfondie de la performance de chaque modèle.

Le F1-score, une métrique qui harmonise la précision et le rappel, est particulièrement pertinent dans le contexte médical où l'équilibre entre la détection des vrais positifs (maladies identifiées correctement) et la réduction des faux positifs (diagnostics incorrects) est crucial. La matrice de confusion offre une vue d'ensemble de la classification, distinguant les vrais positifs, les faux positifs, les vrais négatifs et les faux négatifs. Enfin, le rapport de classification fournit une analyse détaillée de la précision, du rappel (sensibilité) et du score F1 pour chaque classe (sain ou malade).

Comme on peut le noter ci-dessous, le Perceptron en association aux caractéristiques extraites par TSFEL, a atteint une performance parfaite avec un score F1 de 1.00. Cela suggère un surapprentissage, phénomène où le modèle est tellement bien ajusté aux données d'entraînement qu'il peut perdre en généralisation sur de nouvelles données. Dans notre cas, les paires les plus intéressantes semblent être le Random Forest associé à Featuretools et TSFEL ainsi que le Perceptron associé à TSFEL pour un f1-Score respectif de 0.73, ce qui forme une baseline plutôt solide.

Librairie	Modèle	F1-Score
TS-Fresh	SVC	0.33
Featuretools	SVC	0.33
TSFEL	SVC	0.33
TS-Fresh	RandomForest	0.50
Featuretools	RandomForest	0.73
TSFEL	RandomForest	0.73
TS-Fresh	Perceptron	0.50
Featuretools	Perceptron	0.73
TSFEL	Perceptron	1.00

Table 2 – Scores F1 des modèles sur les données d'entraînement

Concernant la sensibilité (recall), elle est particulièrement scrutée dans notre contexte médical. Un rappel élevé signifie que le modèle est capable d'identifier correctement un grand nombre de patients réellement malades, ce qui est vital pour éviter de passer à côté de diagnostics cruciaux.

Pour plus de clarté, il est plus judicieux de démontrer nos performances par des tableaux incluant les métriques nécessaires. Il est primordial de les replacer dans un contexte médical en prenant compte de la répartition des classes et des échantillons peu nombreux. Ci-dessous, nous retrouvons l'ensemble des évaluations détaillées pour toutes les combinaisons :

Modèle	Classe	Précision	Rappel	F1-Score	Support
2*SVC_TS-Fresh	Healthy	0.00	0.00	0.00	2
	Unhealthy	0.50	1.00	0.67	2
2*SVC_Featuretools	Healthy	0.00	0.00	0.00	2
	Unhealthy	0.50	1.00	0.67	2
2*SVC_TSFEL	Healthy	0.00	0.00	0.00	2
	Unhealthy	0.50	1.00	0.67	2
2*RandomForest_TS-Fresh	Healthy	0.50	0.50	0.50	2
	Unhealthy	0.50	0.50	0.50	2
2*RandomForest_Featuretools	Healthy	1.00	0.50	0.67	2
	Unhealthy	0.67	1.00	0.80	2
2*RandomForest_TSFEL	Healthy	1.00	0.50	0.67	2
	Unhealthy	0.67	1.00	0.80	2
2*Perceptron_TS-Fresh	Healthy	0.50	0.50	0.50	2
	Unhealthy	0.50	0.50	0.50	2
2*Perceptron_Featuretools	Healthy	1.00	0.50	0.67	2
	Unhealthy	0.67	1.00	0.80	2
2*Perceptron_TSFEL	Healthy	1.00	1.00	1.00	2
	Unhealthy	1.00	1.00	1.00	2

Table 3 – Rapports de classification exhaustifs pour les modèles sur les données d'entraînement

Suite à cela, les modèles seront évalués sur un ensemble de données de test séparé, permettant d'apprécier leur capacité à généraliser à des données non vues. Les métriques clés incluent le F1-score, la matrice de confusion et un rapport de classification détaillé.

3.3 Documentation et Rapport d'Analyse

Pour résumer, cette section se focalise essentiellement sur la présentation des métriques de performance et l'analyse comparative des différents modèles de machine learning mis en œuvre. Lors de l'exécution des notebooks associés au projet, une attention particulière a été accordée à l'affichage des résultats de manière claire et informative, permettant ainsi une évaluation objective et une comparaison rigoureuse des approches testées.

Enfin, cette documentation en temps réel des performances facilite la comparaison entre les différentes librairies utilisées, telles que TSFresh, Featuretools et TSFEL. En exposant les résultats de manière normalisée, elle permet d'identifier rapidement les outils les plus efficaces pour le traitement et l'analyse des séries temporelles médicales en fonction des critères de succès spécifiés.

4 Méthodologie

Cette section décrit l'approche structurée adoptée dans le cadre de ce projet, mettant l'accent sur l'utilisation de pratiques standardisées pour la gestion du code, des données et des résultats. En s'appuyant sur des principes éprouvés comme l'approche Cookie Cutter pour la structuration du projet et une gestion rigoureuse des artefacts, ce travail vise à la reproductibilité, l'efficacité et la clarté tout au long du processus de développement.

Trois Jupyter Notebooks principaux orchestrent le flux de travail, chacun ayant un rôle spécifique dans le processus d'analyse et de modélisation :

— **1_Environnement** : Ce notebook sert de point de départ pour le projet. Il prépare l'environnement en configurant les dossiers nécessaires et en répartissant les données brutes en ensembles d'entraînement et de test. L'objectif est de mettre en place une structure de base pour les étapes d'analyse et de modélisation qui suivront.

— **2_Trainset** : Après la préparation initiale, ce notebook se concentre sur l'entraînement des modèles de machine learning. Il utilise l'ensemble d'entraînement pour entraîner différents modèles et évaluer leurs performances. Les modèles sont ajustés et optimisés en fonction des résultats obtenus, avec pour objectif de maximiser l'efficacité de la prédiction sur les données médicales.

— **3_Testset** : Le dernier notebook du trio utilise l'ensemble de test pour évaluer la capacité des modèles entraînés à généraliser sur de nouvelles données. Les performances des modèles sont mesurées, fournissant une estimation de leur efficacité en conditions réelles.

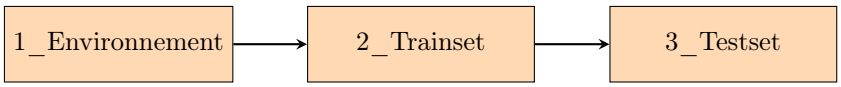

Figure 5 – Flux de travail des notebooks dans le projet

4.1 Cookie Cutter

L'approche Cookie Cutter a servi de fondement pour structurer le projet Doppler, offrant un cadre initial robuste pour organiser les différents composants d'un projet en Data Science. Cette méthode propose une arborescence de dossiers standardisée qui facilite la séparation claire entre le code source, les données, la documentation et les résultats. Pour ce projet, bien que s'inspirant de Cookie Cutter, une personnalisation a été nécessaire pour s'adapter aux spécificités de l'analyse de séries temporelles médicales et le contexte du cadre académique.

La structure résultante se présente comme suit :

- **src** : Contient les modules Python pour les différentes étapes de l'analyse.
  - `data_concatenation.py` : Concatène les fichiers CSV.
  - `data_processing.py` : Traite les données brutes.
  - `feature_extraction.py` : Extrait les caractéristiques des données.
  - `model_testing.py` : Teste les modèles prédictifs.
  - `model_training.py` : Entraîne les modèles de machine learning.
- **notebooks** : Regroupe les Jupyter Notebooks pour guider l'utilisateur à travers les phases du projet. L'ordre d'exécution dépend de la nomenclature utilisée (1_Environnement, 2_Trainset, 3_Testset).
- **models** : Contient les modèles de machine learning entraînés, sauvegardés au format pickle.
- **docs** : Dossier pour la documentation, incluant la planification, le cahier des charges, et diverses notes.
- **data** : Contient les dossiers pour les données brutes, traitées et intermédiaires.
  - **raw** : Pour les fichiers de données brutes.
  - **processed** : Regroupe les données traitées.
  - **interim** : Pour les données intermédiaires, incluant les ensembles de test et d'entraînement.

4.2 Gestion des Artefacts

Les artefacts dans ce projet comprennent tous les fichiers générés durant le processus d'analyse, allant des données traitées aux modèles de machine learning. Une gestion rigoureuse de ces artefacts est cruciale pour assurer l'intégrité des résultats et faciliter les phases de test et de validation.

- Les données traitées sont sauvegardées dans le dossier **processed** pour référence et utilisations ultérieures.
- Le dossier **interim** joue un rôle clé dans la gestion des ensembles de données de test et d'entraînement, ainsi que pour le stockage des résultats intermédiaires de l'extraction des caractéristiques.
- Les modèles entraînés sont archivés dans le dossier **models**, permettant leur réutilisation pour de futures analyses ou prédictions.

Cette structure soigneusement conçue assure non seulement une organisation logique des différents éléments du projet, mais facilite également la collaboration, la révision et l'extension du travail réalisé.

5 Généricité et Adaptabilité

Dans le cadre de ce projet, l'objectif était de concevoir une solution capable d'analyser et de prédire des séries temporelles médicales, avec une attention particulière portée aux données issues du Doppler transcrânien. Bien que l'ambition initiale incluait la généralisation et l'adaptabilité du système pour accueillir une variété de datasets et conditions médicales, plusieurs facteurs ont influencé le développement de cette généricité.

5.1 Intégration de Datasets Divers

La capacité d'un système à intégrer divers datasets est cruciale pour sa flexibilité et son utilité à long terme, notamment dans un domaine aussi complexe et varié que celui de la santé. Cependant, chaque dataset médical vient avec son propre ensemble de défis, y compris des structures de données distinctes, des mesures spécifiques et des contraintes d'éthique et de confidentialité.

Une approche entièrement générique aurait nécessité un cadre extrêmement flexible et complexe, susceptible d'augmenter la difficulté de mise en œuvre et de maintenance, ainsi que de potentiellement impacter la précision des modèles pour des cas spécifiques.

5.2 Optimisation des Modèles pour Diverses Conditions

L'objectif d'optimiser les modèles prédictifs pour une gamme étendue de conditions médicales est louable mais se heurte à la réalité pratique des contraintes de recherche. Chaque condition médicale a ses propres indicateurs et biomarqueurs, rendant nécessaire une personnalisation poussée des modèles pour chaque cas. Cette personnalisation requiert non seulement une expertise médicale approfondie mais aussi des ensembles de données vastes et bien annotés pour chaque condition.

Dans le contexte de ce projet, réalisé dans un cadre académique avec des ressources et un temps limités, atteindre un niveau élevé d'optimisation pour une gamme étendue de conditions était un objectif ambitieux. Ainsi, une décision stratégique a été prise de concentrer les efforts sur l'approfondissement de l'analyse pour des cas spécifiques, notamment les données du Doppler transcrânien, pour assurer une qualité et une précision maximales.

5.3 Approche pragmatique

Face aux défis mentionnés, une stratégie de généricité ciblée a été adoptée. Cette stratégie implique la construction d'un système modulaire et évolutif, où des composants spécifiques peuvent être adaptés ou ajoutés pour répondre aux besoins de différents datasets ou conditions médicales sans réinventer la roue. Cette approche permet une certaine flexibilité tout en maintenant une base solide et cohérente. Les éléments clés de cette stratégie comprennent :

- Une architecture logicielle modulaire, où chaque composant (traitement des données, extraction des caractéristiques, entraînement des modèles, etc.) est conçu comme une unité indépendante, facilitant ainsi les modifications ou les extensions.
- L'adoption de pratiques de développement logiciel rigoureuses, incluant la documentation détaillée et l'intégration continue, pour assurer la fiabilité et la maintenabilité du système.
- Une démarche itérative de développement, permettant d'ajuster continuellement le système en fonction des résultats.

6 Résultats et Délivrables

La phase d'exploration et d'analyse des données Doppler transcrâniennes a révélé des insights significatifs sur les caractéristiques distinctives des séries temporelles médicales. Grâce à une approche méthodique, combinant des techniques avancées de traitement du signal et d'analyse statistique, le projet a permis de délimiter avec précision les paramètres clés des signaux I et Q, ainsi que de déterminer l'importance de l'enveloppe supérieure des spectrogrammes pour l'évaluation du flux sanguin cérébral.

Ces résultats ont non seulement enrichi notre compréhension des données Doppler, mais ont également établi une baseline pour la phase de modélisation prédictive.

6.1 Module des Modèles Prédictifs

Le cœur de la phase de modélisation réside dans le module model_testing.py, conçu pour appliquer les modèles prédictifs sauvegardés au format pickle à de nouvelles données de test. Ce module illustre l'aboutissement du projet, transformant la théorie et l'analyse en applications pratiques capables de prédire l'état de santé neurologique à partir des données Doppler non connues.

Le module fonctionne en chargeant les modèles prédictifs entraînés – SVM, Perceptron et Random Forest – et en les appliquant aux données extraites du dossier de test. Cette démarche assure une évaluation rigoureuse de la performance des modèles dans des conditions réelles, simulant ainsi leur utilisation potentielle dans un contexte clinique.

TABLE 4 – Résultats des modèles sur les données de test

Modèle	Librairie	Precision (Healthy)	Recall (Healthy)	Precision (Unhealthy)	Recall (Unhealthy)	F1-Score (Average)	Accuracy
Perceptron	Featuretools	0.50	0.33	0.71	0.83	0.58	0.67
RandomForest	Featuretools	0.50	0.33	0.71	0.83	0.58	0.67
SVC	Featuretools	0.00	0.00	0.67	1.00	0.40	0.67
Perceptron	TSFEL	0.50	0.67	0.80	0.67	0.65	0.67
RandomForest	TSFEL	0.67	0.67	0.83	0.83	0.75	0.78
SVC	TSFEL	0.00	0.00	0.67	1.00	0.40	0.67
Perceptron	TS-Fresh	0.40	0.67	0.75	0.50	0.55	0.56
RandomForest	TS-Fresh	0.67	0.67	0.83	0.83	0.75	0.78
SVC	TS-Fresh	0.00	0.00	0.67	1.00	0.40	0.67

Dans ce tableau, les performances des différents modèles sont résumées, offrant une vue d'ensemble de leur efficacité sur les données de test. Le recall est particulièrement mis en exergue, étant un indicateur crucial dans le diagnostic médical, où le coût d'un faux négatif (ne pas identifier une condition pathologique) peut être considérablement élevé.

Les résultats montrent que les modèles basés sur la librairie TSFEL, notamment le RandomForest, présentent une performance prometteuse avec un équilibre entre le recall pour les classes 'Healthy' et 'Unhealthy' ainsi qu'une précision élevée. Cela indique que cette combinaison de modèle et de librairie pourrait être particulièrement adaptée pour une application réelle, offrant un bon compromis entre la capacité à détecter correctement les cas pathologiques et à minimiser les faux positifs.

Le Perceptron et le RandomForest, associés à la librairie Featuretools, montrent également des résultats intéressants, bien que le recall pour la classe 'Healthy' soit inférieur à celui observé avec TSFEL. En revanche, les modèles SVC, malgré une précision élevée pour la classe 'Unhealthy', affichent un recall nul pour la classe 'Healthy', ce qui pourrait limiter leur utilisation dans un contexte où la détection précoce de conditions pathologiques est critique.

Ces analyses mettent en lumière l'importance d'une évaluation multidimensionnelle des modèles prédictifs, prenant en compte à la fois la précision globale et les performances spécifiques à chaque classe, afin d'assurer l'adéquation des outils de diagnostic avec les exigences du domaine médical.

7 Organisation et Planification

La gestion et la planification d'un projet académique, tel que celui-ci réalisé en troisième année de bachelor en Ingénierie des Données, nécessitent une approche structurée et méthodique. Le projet s'est déroulé dans un cadre rigoureux, organisé à l'aide d'un calendrier comportant les jalons nécessaires et des objectifs clairement définis dès le départ

7.1 Planning du Projet

Le planning du projet a été élaboré selon la méthode Gantt, un outil de planification efficace permettant de visualiser l'ensemble des tâches à réaliser et leur répartition dans le temps. Cette approche a facilité une gestion du temps optimale, chaque phase du projet étant clairement identifiée, des premières étapes d'exploration des données jusqu'à la finalisation des modèles prédictifs et la rédaction du rapport final.

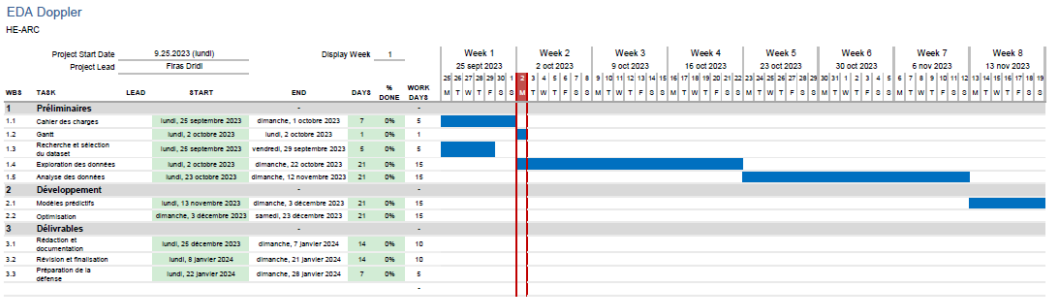

FIGURE 6 – Planning Gantt

Le tableau Gantt, a servi de référence tout au long du projet, permettant de suivre l'avancement des travaux et d'ajuster le planning en fonction des impondérables et des découvertes faites en cours de route. Cette planification dynamique a été cruciale pour la réussite du projet, offrant une flexibilité nécessaire face aux défis inhérents à la recherche appliquée.

Néanmoins, le déroulement du projet a été marqué par des retards, principalement attribuables aux défis rencontrés lors de la phase d'exploration des données. La complexité inhérente à l'analyse des séries temporelles médicales, en particulier les données Doppler transcrâniennes, a exigé un ajustement technique considérable.

Les retards ont été accentués par la nécessité de maîtriser la génération des spectrogrammes à partir des signaux I et Q, un processus essentiel mais exigeant une certaine compréhension des outils de traitement du signal. Cette phase a impliqué de multiples itérations de tests et d'ajustements, prolongeant la période d'exploration au-delà des estimations initiales.

Toutefois, une fois cette étape critique surmontée, le rythme de développement du projet a connu une amélioration. La compréhension acquise durant la phase exploratoire a facilité les étapes subséquentes, permettant une mise en œuvre plus fluide et efficace des analyses et des modèles prédictifs.

Face aux retards initiaux, une révision du planning Gantt s'est avérée indispensable. Les ajustements apportés ont permis de réallouer les ressources de manière à compenser le temps perdu, assurant ainsi le respect des objectifs à long terme. Cette flexibilité dans la gestion du planning a été déterminante pour le maintien de la dynamique de travail.

7.2 Collaboration

Bien que le projet ait été mené individuellement, la collaboration avec le superviseur du projet, le Dr. Albertetti, a joué un rôle central dans son développement. Des réunions hebdomadaires étaient organisées chaque lundi, permettant non seulement de faire le point sur l'avancement des travaux, mais aussi de bénéficier de conseils avisés, de discuter des obstacles rencontrés et de redéfinir les objectifs si nécessaire.

Ces échanges réguliers ont contribué à maintenir une dynamique de travail et à orienter efficacement les efforts de recherche. De plus, la mise à disposition de quatre périodes le lundi pour se consacrer pleinement au projet a permis de dédier un temps conséquent à la réalisation des différentes tâches, complété par un travail personnel soutenu en dehors des heures de cours.

La gestion du projet a également été appuyée par l'utilisation de Git, un outil de versionning qui a facilité l'organisation du code source, la documentation et le partage des avancées. Un journal de travail hebdomadaire était tenu dans les wikis du repository Git, offrant une transparence sur le déroulement du projet et permettant de consigner méthodiquement les progrès réalisés, les décisions prises et les connaissances acquises.

En somme, l'organisation et la planification du projet se sont appuyées sur des outils de gestion de projet éprouvés et une collaboration étroite avec le superviseur, assurant ainsi une progression réfléchie vers la réalisation des objectifs fixés.

8 Évolution du Cahier des Charges

Dans le cadre de ce projet de bachelor, l'évolution du cahier des charges a joué un rôle clé dans l'alignement des objectifs et des méthodologies employées. Au début du projet, une révision substantielle du cahier des charges a été nécessaire pour incorporer une approche méthodologique plus détaillée, ainsi que pour définir clairement le découpage du processus en modules Python distincts.

Ces modifications ont permis de structurer le projet de manière plus efficace, en mettant en place des étapes clairement définies pour l'exploration des données, l'analyse, la construction des modèles, et l'évaluation. L'intégration de modules Python a offert une flexibilité et une modularité accrues, facilitant ainsi l'adaptation du projet à de nouvelles exigences ou données.

8.1 Modifications et Mises à Jour

Les mises à jour et modifications du cahier des charges ont été effectuées en étroite collaboration avec le superviseur, garantissant que les ajustements étaient en accord avec les objectifs pédagogiques et les attentes du projet. Chaque modification était soigneusement discutée et validée lors des réunions hebdomadaires, permettant ainsi un suivi rigoureux de l'avancement et des orientations du projet.

8.2 Communication et Documentation des Changements

Le cahier des charges révisé était stocké sur un disque sécurisé de l'école, accessible à la fois par l'étudiant et le superviseur. Ce document servait de référence tout au long du projet, assurant une source de vérité unique pour les objectifs, les exigences, et les spécifications méthodologiques. La documentation et la communication des changements étaient essentielles pour maintenir la transparence et l'alignement entre toutes les parties prenantes impliquées dans le projet.

9 Travaux futurs

Les travaux futurs de ce projet offrent un large éventail de possibilités pour améliorer et étendre les recherches actuelles. Une amélioration immédiate consisterait à perfectionner le calcul de l'enveloppe supérieure du spectrogramme.

Actuellement, bien que l'enveloppe fournisse une représentation simplifiée et utile des données spectrogrammiques, elle ne reflète pas avec précision les amplitudes maximales à chaque instant. Un recalibrage minutieux de cette enveloppe, pour qu'elle épouse fidèlement les pics d'amplitude du spectrogramme, pourrait améliorer significativement la qualité des données extraites et, par conséquent, la précision des modèles prédictifs développés. Au-delà de cet aspect technique, les travaux futurs pourraient explorer de nouvelles méthodologies d'extraction de caractéristiques, en s'appuyant sur des techniques avancées d'apprentissage profond ou de traitement d'images, pour capturer des aspects plus subtils des données Doppler transcrâniennes.

9.1 Domaines de recherche ouverts

Dans le domaine de la recherche, plusieurs pistes restent ouvertes. L'application de ces modèles prédictifs à d'autres conditions médicales, où les données temporelles jouent un rôle crucial, pourrait étendre leur applicabilité et leur utilité clinique. De plus, l'exploration de l'interaction entre les variables physiologiques capturées par le Doppler transcrânien et d'autres biomarqueurs pourrait révéler des corrélations inattendues et des insights diagnostiques précieux.
Enfin, le développement d'une plateforme intégrée, combinant l'analyse de données, la modélisation prédictive et la visualisation interactive, pourrait transformer la manière dont les cliniciens accèdent et interprètent les données Doppler transcrâniennes, facilitant ainsi la prise de décision en temps réel et la personnalisation des soins aux patients.

Conclusion

La conclusion de ce projet marque la fin d'un riche apprentissage et la découverte des séries temporelles médicales. Ce travail, réalisé dans le cadre de mon cursus de Bachelor en Ingénierie des Données à la Haute École Arc, sous la supervision experte de Fabrizio Albertetti, Ph.D., a non seulement permis d'approfondir ma compréhension des défis inhérents à l'analyse de données médicales complexes mais a également ouvert des perspectives pour l'amélioration des diagnostics et des traitements dans le domaine neurovasculaire.

En abordant des problématiques complexes et en appliquant des méthodes d'analyse de données avancées au domaine médical, j'ai pu développer une compréhension profonde des enjeux et des techniques à la pointe de l'ingénierie des données. Cette expérience m'a non seulement permis d'acquérir des compétences techniques précises, mais a également affiné ma capacité à gérer des projets d'envergure, à travailler de manière autonome et à collaborer efficacement avec des experts de divers domaines.

L'approche méthodologique adoptée, l'utilisation des librairies d'analyse de séries temporelles et la construction de modèles prédictifs sont autant d'éléments qui constituent une base solide pour mon travail de bachelor. La gestion des artefacts, l'organisation du code en modules réutilisables et l'attention portée à la documentation et à la présentation des résultats sont des pratiques professionnelles que je compte intégrer dans mes futurs projets académiques et professionnels.

En conclusion, ce projet a été une étape cruciale dans ma formation, me dotant des outils, des connaissances et de la confiance nécessaires pour aborder avec succès mon travail de bachelor. L'expérience acquise et les leçons apprises seront des atouts précieux dans la poursuite de mes études et de ma carrière professionnelle dans le domaine dynamique et en constante évolution de l'ingénierie des données.

Table des figures