MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Crash Pulse prediction applied to Lateral Crash Configurations

Autor: Miguel Rodríguez González

Director: Francisco José López Valdés

Göteborg, Sweden

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Crash Pulse prediction applied to lateral Crash Configurations

en la ETS de Ingeniería - ICML de la Universidad Pontificia Comillas en el

curso académico 2020/2021 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Miguel Rodríguez González Fecha: 12/07/2021

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

LOPEZ VALDES FRANCISCO JOSE - 09437440B

Firmado digitalmente por LOPEZ VALDES FRANCISCO JOSE - 09437440B
Fecha: 2021.07.12 09:44:53 -04'00'

Fdo.: Francisco José López Valdés Fecha: 12/07/2021

MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Crash Pulse prediction applied to Lateral Crash Configurations

Autor: Miguel Rodríguez González

Director: Francisco José López Valdés

Göteborg, Sweden

CRASH PULSE PREDICTION APPLIED TO LATERAL CRASH CONFIGURATIONS

Autor: Rodríguez González, Miguel.
Director: Francisco José López Valdés.
Entidad Colaboradora: Volvo Cars

RESUMEN DEL PROYECTO

Proyecto centrado en la predicción de los pulsos de choque producidos en la colisión lateral. Utilizando metodologías de extracción y procesamiento de datos para obtener una configuración inicial que describa el posicionamiento de ambos vehículos, se ha desarrollado un método de Machine Learning para realizar la predicción del "Crash Pulse" y una posterior evaluación de la predicción realizada comparándola con los resultados que se obtuvieron en las simulaciones de estas colisiones.

Palabras clave: ML, Análisis de datos, Pulso de choque, Seguridad del vehículo.

1. Introducción

La seguridad de los ocupantes ha sido y será un atributo clave de todos los vehículos. Las metodologías actuales para evaluar y mejorar la protección de los ocupantes se basan en pruebas de choque físicas y virtuales. La continua evolución del software de simulación y las metodologías de elementos finitos (EF) han contribuido a una gran reducción de los costes económicos. Sin embargo, estas simulaciones son costosas desde el punto de vista computacional y, por lo tanto, aparecen algunas limitaciones en cuanto a la cantidad de pruebas de choque tanto físicas como virtuales que se pueden realizar.

2. Definición del proyecto

Esta necesidad se traduce en este proyecto, donde, pretendiendo utilizar soluciones de Big Data y ML, tendrá como objetivo final asistir a los ensayos de choque tanto físicos como virtuales. Este proyecto, por tanto, se establece como una progresión de estudios anteriores ([1] y [2]) y pretende desarrollar una herramienta virtual capaz de asistir a las metodologías actuales de evaluación de los efectos de un ensayo de choque de vehículos en diferentes configuraciones.

Por lo tanto, el objetivo de este proyecto es investigar métodos numéricos para predecir el pulso de choque de un vehículo en el impacto lateral, utilizando datos de simulación. Para tener una idea clara de las diferentes tareas a realizar, se ha definido una lista:

- Identificar la viabilidad y las limitaciones de los métodos existentes para la aplicación de la seguridad de los vehículos.
- Crear un método que analice automáticamente los datos de prueba y simulación y clasifique las condiciones de impacto, según las configuraciones de choque detalladas en [1] y [2].
- Implementar y entrenar un método que prediga los pulsos de choque dada una configuración de choque.
- Evaluar la precisión de los pulsos de choque predichos.

3. Descripción del modelo/sistema/herramienta

El siguiente proyecto pretende desarrollar un método que sea capaz de analizar los datos de simulación (pruebas de choque virtuales) para utilizarlos como entrada de un algoritmo de Machine Learning para predecir el pulso de choque sufrido en cada configuración específica de choque lateral dada. Para crear dicha metodología, el enfoque se dividirá en dos partes principales: La obtención de los datos necesarios y la creación del algoritmo mediante ML para la predicción.

3.1. Obtención de datos.

En esta sección del proyecto, el método a seguir consiste en un análisis profundo de los datos disponibles en Volvo Cars, basado en pruebas de choque virtuales. Los siguientes pasos para desarrollar esta fase se inspiran en la metodología seguida en estudios anteriores [1], que se pueden ver en la siguiente figura:

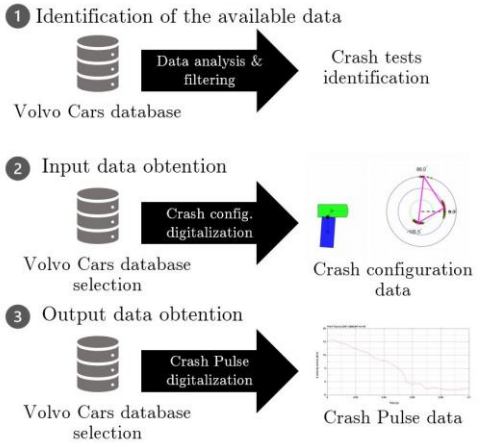

Figura 1: *Descripción general de los tres pasos del método de recopilación de datos. Paso 1: Identificación de los datos disponibles; Paso 2: Extracción de datos de entrada. [2]; Paso 3: Extracción de datos de salida.*

3.1.1. Paso 1.

El punto de partida de este método consiste en filtrar y analizar los datos disponibles. El objetivo principal de este paso es recopilar los datos que corresponden a lo que el algoritmo necesita para trabajar o en lo que el proyecto completo quiere basarse. No sólo en términos de configuraciones de colisiones, sino también en términos de datos de colisiones virtuales. Es necesario establecer un método para obtener los parámetros necesarios y evaluar qué tipo de configuraciones de colisión se utilizarán en el algoritmo.

3.1.2. Paso 2.

Una vez elegidas las pruebas de choque, el método extraerá los parámetros necesarios de los archivos de las pruebas de choque. De este modo, el objetivo principal es analizar automáticamente todos los datos necesarios que pueden definir la configuración de choque de la colisión. Además, a la hora de analizar estos datos, es importante señalar que cada colisión puede clasificarse en dos configuraciones de colisión, dependiendo del vehículo que se considere como vehículo anfitrión. El vehículo anfitrión es el vehículo en el que se analiza el ocupante en riesgo.

3.1.3. Paso 3.

Siguiendo la metodología anterior, el algoritmo también analizará los datos para obtener el pulso de choque (rotaciones y traslaciones de 3 ejes) que se requiere para entrenar el algoritmo predictivo. Una vez más, es importante tener en cuenta que ambos vehículos serán considerados como el vehículo anfitrión para diferentes configuraciones de choque, y, por lo tanto, se extraerán y recogerán las series temporales de pulsos de choque. Siguiendo este método de recogida de datos, el algoritmo ML dispondrá de todos los datos necesarios para el siguiente método.

3.2. Desarrollo del algoritmo.

En este punto del proceso, las principales dificultades estarán relacionadas con la selección del modelo ML, cómo entrenarlo, qué datos se van a utilizar en la fase de entrenamiento y en la de validación y, finalmente, cómo evaluar las predicciones del pulso de choque. Siguiendo el método de recopilación de datos, esta fase podría dividirse en los siguientes pasos:

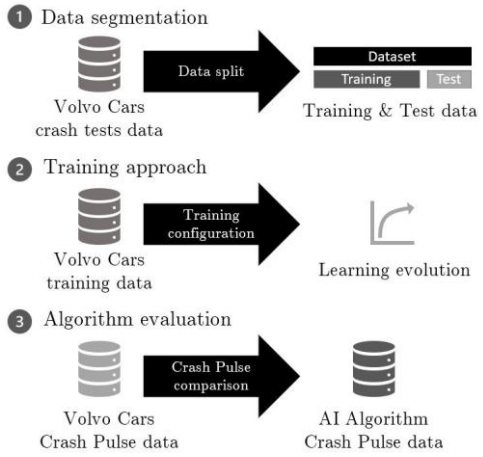

Figura 2: *Descripción general de los tres pasos del método de ML. Paso 1: Segmentación de datos; Paso 2: Enfoque de capacitación; Paso 3: Evaluación del algoritmo.*

Antes de poder aplicar una metodología de este tipo, es importante centrar la decisión en qué modelo de ML se utilizará y qué enfoque se adoptará para el proceso de aprendizaje. En términos generales, este enfoque de ML permite tener un proceso de aprendizaje capaz de tener en cuenta no sólo la configuración inicial del choque, sino también los pulsos de choque ya extraídos para desarrollar futuras predicciones.

3.2.1. Paso 1.

Una vez elegido el modelo, se puede desarrollar el método del algoritmo ML. Como ya se ha dicho, es de vital importancia la forma de dividir y gestionar los datos. Dependiendo de la decisión sobre cómo dividir los datos, el rendimiento del algoritmo al realizar las pruebas puede variar drásticamente.

3.2.2. Paso 2.

Una vez divididos los datos, puede comenzar el proceso de entrenamiento. El algoritmo de ML trabaja con los datos de entrada y salida para aprender cómo están estructurados los datos y cómo están conectados. A continuación, empieza a predecir las posibles salidas para los datos de entrada dados. Además, el método compara sus predicciones con los datos de salida reales y aprende durante el proceso iterando y

haciendo algunas modificaciones en diferentes parámetros como los pesos y los parámetros de sesgo. Estos parámetros son los responsables del proceso de aprendizaje del algoritmo, modificándose en función de la precisión que alcance el modelo. De este modo, el modelo puede aprender y predecir el impulso de choque de las diferentes configuraciones de choque. Sin embargo, hay que tener algunas consideraciones cuidadosas a la hora de entrenar el modelo, ya que un modelo sobre o infra entrenado puede ser problemático. Estas consideraciones incluyen asegurarse de que los datos son variados y representan lo máximo posible las diferentes simulaciones que se quieren predecir.

3.2.3. Paso 3.

La fase de evaluación puede ser, en ciertas fases, realizada en paralelo a la fase de entrenamiento. El objetivo principal de esta fase es probar el modelo entrenado con los datos de prueba designados, analizando cómo el modelo es capaz de predecir las curvas de pulsos de choque basadas en las configuraciones de choque dadas. Como se ha explicado anteriormente, esto tendrá una fuerte relación con la forma en que se segmentan y dividen los datos.

4. Resultados

Durante el procedimiento para obtener los conjuntos de datos y las predicciones, se encontraron y se enfrentaron algunos problemas. Estos se explicarán en las siguientes secciones.

4.1. Método de recopilación de datos

Para obtener los conjuntos de datos deseados, fue necesario adoptar diferentes enfoques. En primer lugar, los datos disponibles podían dividirse en diferentes escenarios de simulación: simulaciones de coche a coche y simulaciones de sled deformable de coche a movimiento. Debido al pequeño conjunto de datos recogidos en las simulaciones coche a coche, se necesitaban datos más variados, por lo que se optó por obtener datos de simulación de sled deformable coche a coche, que pueden asemejarse a la colisión de dos coches

El algoritmo para las simulaciones coche a coche se desarrolló para obtener las diferentes variables a partir de los archivos que corresponden a la salida de una simulación realizada con LsDyna (llamados archivos *d3plot* y *binout*). Para ello, se obtuvieron las diferentes posiciones de ambos vehículos mediante la agrupación de los nodos de ambos vehículos. Estos nodos son los diferentes puntos que componen los elementos y partes de cada vehículo. Mediante la agrupación de los mismos, el algoritmo puede identificar los dos vehículos y extraer las variables. Por lo tanto, el siguiente paso fue obtener las velocidades de cada vehículo, medir los ángulos de la configuración del choque y obtener las masas de los vehículos (para más información, ver *Anexo II*).

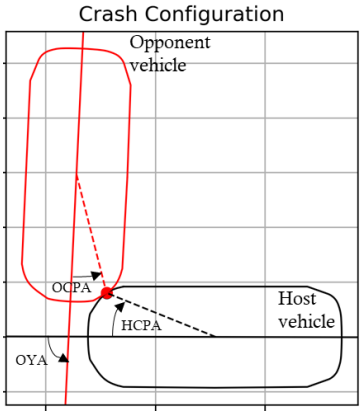

Figura 3: Obtención de la configuración de choque a través de los archivos d3plot y binout.

El enfoque para las simulaciones de sled deformable con coche fue muy similar, aunque se introdujeron algunas variaciones para resolver la falta de nodos específicos vinculados a los acelerómetros de los vehículos reales. Estos acelerómetros se utilizaron para obtener las diferentes velocidades del coche. Como éstas no eran las mismas para el sled deformable en movimiento, el enfoque se desarrolló detectando la velocidad de los nodos y extrayendo el valor para todo el conjunto. Esto se llevó a cabo después del método de agrupación que ayudó a diferenciar entre el coche y el sled deformable en movimiento o la barrera en movimiento.

Además, se introdujo un último paso para extraer el impulso de choque de las simulaciones (aceleraciones de traslación y rotación de 3 ejes). La metodología fue similar tanto para las simulaciones coche a coche como para las simulaciones coche a sled deformable en movimiento. En el caso de las simulaciones coche a coche, las series temporales que contienen los pulsos de choque se obtuvieron de la simulación, extrayendo las traslaciones y rotaciones de 3 ejes en forma de velocidades. En cuanto a las simulaciones de coche a sled deformable, en este caso, en lugar de extraer las aceleraciones de ambos vehículos, sólo se obtuvieron las aceleraciones del vehículo anfitrión.

El enfoque final para la recopilación de estos conjuntos de datos de pulsos de choque se orientó al método ML (para más información, ver *Anexo II*).. En primer lugar, estas series temporales se convirtieron en aceleraciones, ya que son más fáciles de entender y dan una mejor sensación de lo que ocurre durante el choque. Como la complejidad del modelo aumentaría drásticamente al introducir las diferentes series temporales de los pulsos de choque, se desarrolló un método de reducción de datos. Para simplificar estos conjuntos de datos, se aplicó una discretización de las series temporales. Así, las series temporales a analizar se cortaron en el tiempo a 80 ms para tener el mismo formato y cantidad de puntos de cada pulso de choque de salida. Las series temporales resultantes se dividieron en varios intervalos. De cada intervalo, se calculó la media y se extrajo como valor discreto, reduciendo cada serie temporal a un conjunto de datos de 16 valores medios. Esta cantidad de puntos consistía en un valor por cada 5 ms, que describía el movimiento y la tendencia del mencionado pulso de choque. A continuación, se puede ver un ejemplo de los diferentes valores medios:

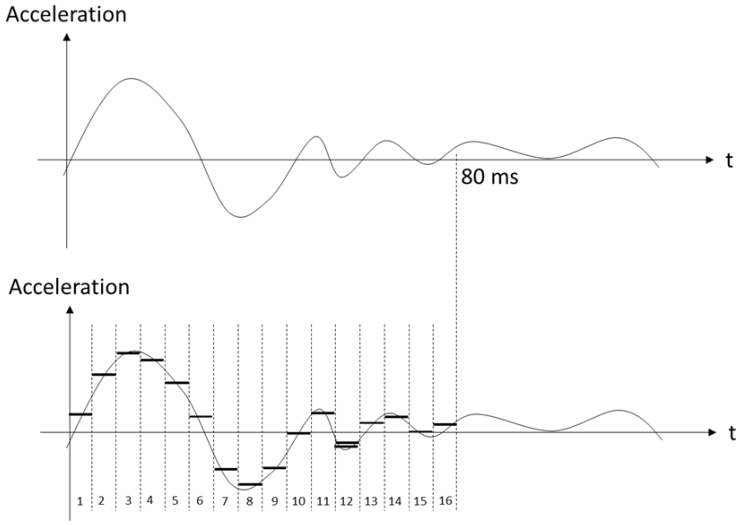

Figura 4: Transformación de la serie temporal.

4.2. Método de ML

Cuando se recogieron los datos, fue necesaria una fase de preprocesamiento para transformar los datos de entrada en algo que el algoritmo de ML pudiera entender.

En primer lugar, los ángulos se convirtieron en coordenadas de módulo 1, para ayudar al algoritmo a entender conceptos como que 180 grados y -180 grados son, de hecho, lo mismo.

Se creó un conjunto de datos de casi 500 simulaciones diferentes, con el siguiente formato:

$HCPA_X$	**$HCPA_Y$**	**$OCPA_X$**	**$HCPA_Y$**	**OYA_X**	**OYA_Y**	**HS**	**OS**	**MASS H**	**MASS O**
-0.88	-0.39	0.99	0.53	0.92	-0.02	-0.57	-0.54	-1.06	-0.69
0.99	0.53	-0.88	-0.39	0.92	-0.02	-0.54	-0.57	1.43	-0.69
0.18	1.93	-1.76	1.51	-0.72	1.53	-0.57	0.68	-0.85	-0.69

Tabla 1 *Datos normalizados de entrada para el algoritmo de ML.*

4.2.1. Training-Test Data split

El siguiente paso fue seleccionar los datos de validación que ayudarían a evaluar el algoritmo y que serían significativos para tener en cuenta los diferentes tipos de simulaciones recogidas. Para ello, se trazó un gráfico que incluía las 5 variables principales que describen la configuración del choque:

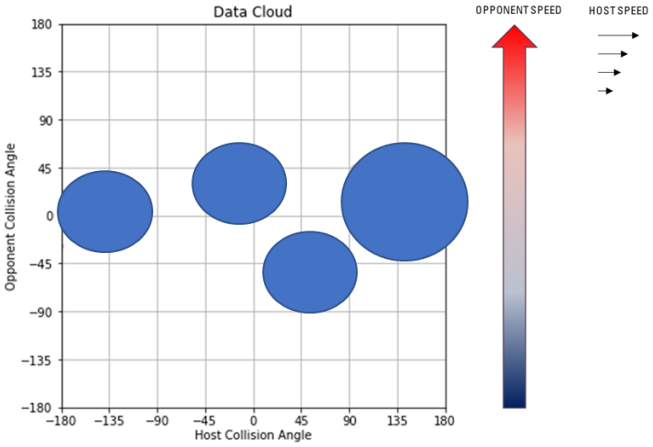

Figura 5: *Datos de las diferentes simulaciones recogidas (HCPA vs OCPA con OYA, HS y OS incluidos). Debido a la confidencialidad, no se pueden mostrar flechas.*

De esta nube de datos, es necesario extraer los atribuidos como choques laterales. Para ello, el HCPA (Host Collision Point Angle) será la variable correspondiente que indique qué colisiones corresponden a una configuración lateral o no. Como se ha explicado anteriormente, el HCPA corresponde al ángulo entre la dirección del coche y el FPOC (First Point of Contact). En cuanto a las dimensiones normalizadas del coche, se puede ver que los ángulos correspondientes a un choque lateral se corresponderán con los intervalos de (45º, 135º) y (-135º, -45º) como se puede ver a continuación:

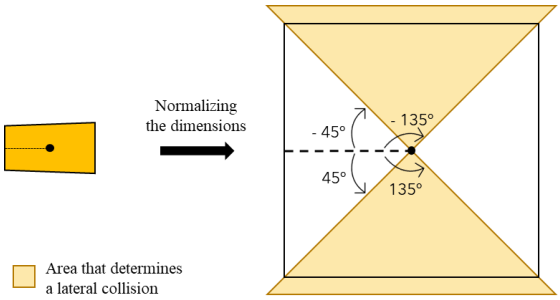

Figura 6: *Criterio para determinar las colisiones laterales.*

Esta figura representa el vehículo normalizado, lo que significa que las dimensiones frontales y laterales se normalizan a un valor de 1. Con este paso, se midieron el HCPA y el OCPA.

Al aplicar esta selección, la nube de datos resultante con las colisiones laterales se reduce a una cantidad de 100 simulaciones. A partir de esta nube de datos, los conjuntos de datos de entrenamiento y de prueba se dividieron en una proporción del 80% y del 20% respectivamente (para más información, ver *Anexo II*). La segmentación se realizó dividiendo aleatoriamente los datos y analizando la división aplicada. El objetivo principal era obtener un conjunto de datos de prueba bien distribuido que fuera lo suficientemente representativo de las simulaciones recogidas, con el fin de validar las diferentes configuraciones extraídas. Al guardar los conjuntos de datos de entrenamiento y de prueba, se pueden comparar y evaluar fácilmente los distintos modelos utilizados. Tras algunas pruebas y comprobaciones del rendimiento de los diferentes algoritmos de ML, se eligió el conjunto de datos final

Al elegir el enfoque de ML, se desarrollaron múltiples algoritmos. Por lo tanto, los modelos finales consistieron en cuatro métodos: Perceptrón multicapa (MLP) utilizando redes neuronales (NN), Regresión de Random Forest, Regresión de Support Vector Machine y Regresión de Gradient Boost (para más información, ver *Anexo II*).. Debido a los resultados obtenidos (mostrados en el Apéndice I), se realizaron dos enfoques diferentes. Inicialmente, como ya se ha explicado, el conjunto de datos de entrenamiento se creó a partir de la nube de datos correspondiente únicamente a las colisiones laterales. Por lo tanto, el modelo fue entrenado y probado con colisiones laterales.

En segundo lugar, tras analizar los resultados, se creó un segundo conjunto de entrenamiento. Este conjunto se eligió a partir de la nube de datos original antes de diferenciar entre colisiones laterales y no laterales. Esto significa que el segundo conjunto de datos de entrenamiento tenía colisiones laterales y no laterales, incluyendo colisiones frontales, traseras y oblicuas. Por lo tanto, el modelo se entrenó con cualquier tipo de colisión, pero se sólo validó con colisiones laterales.

Estos dos enfoques tenían como objetivo verificar que, en el primer enfoque, el modelo no se sobreajuste en exceso a los datos, ya que la cantidad de datos introducidos era limitada, y tal vez, sesgada. Para comprender mejor y comprobar la fiabilidad de estas predicciones, se aplicó el segundo enfoque

Es importante señalar que el enfoque del modelo se desarrolló en función del tipo de aceleración. Se entrenó un modelo con todo el conjunto de datos de aceleraciones traslacionales y otro modelo con rotaciones. Esta división se eligió para evitar tener datos erróneos en los modelos. No sólo eso, sino que también se utilizaron los tres ejes juntos, ya que estas aceleraciones también pueden estar conectadas. Por lo tanto, la configuración final del enfoque ML consistía en un modelo que predecía las aceleraciones traslacionales y otro modelo para las aceleraciones rotacionales.

Después de utilizar estos cuatro métodos, los resultados de las predicciones fueron mejores cuando se utilizó el modelo de regresión Random Forest. Este modelo es un algoritmo de aprendizaje supervisado que utiliza el método de aprendizaje por conjuntos para la regresión. Como breve descripción, el aprendizaje de conjuntos es una técnica que combina predicciones de múltiples algoritmos de aprendizaje automático para hacer una predicción más precisa que un solo modelo. Por lo tanto, este enfoque funciona creando múltiples árboles de decisión durante la fase de entrenamiento, prediciendo tantos valores de salida como árboles de decisión tenga el modelo. El resultado final consiste en la media de los diferentes valores predichos de todos los árboles de decisión incluidos. Estos árboles de decisión se desarrollan en paralelo, evitando cualquier tipo de influencia entre ellos y utilizando diferentes partes del conjunto de datos de entrenamiento para cada árbol. Este modelo tiene la ventaja de que el rendimiento no varía dependiendo de si los datos están normalizados o no. Por esta razón y para simplificar la metodología, este modelo no requiere un paso de normalización.

Utilizando el conjunto de entrenamiento del primer enfoque, con sólo simulaciones laterales para entrenar el modelo, los resultados fueron los siguientes:

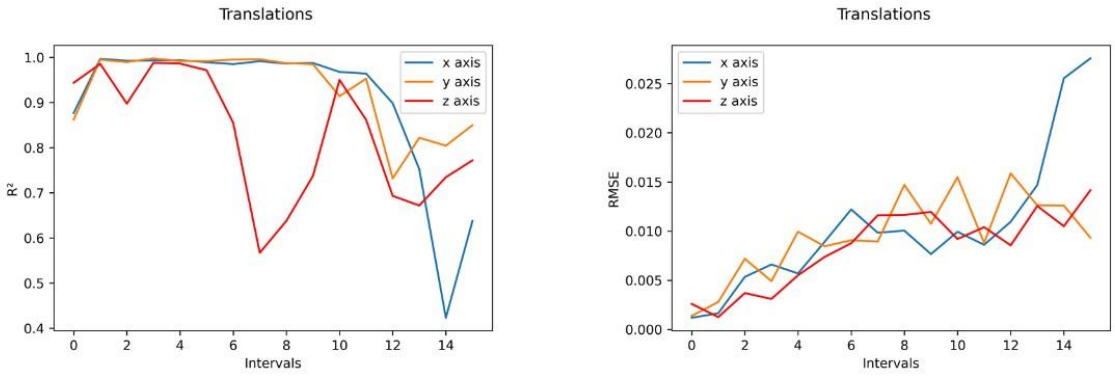

Figure 7: *R^2 and RMSE for Translations using Random Forest.*

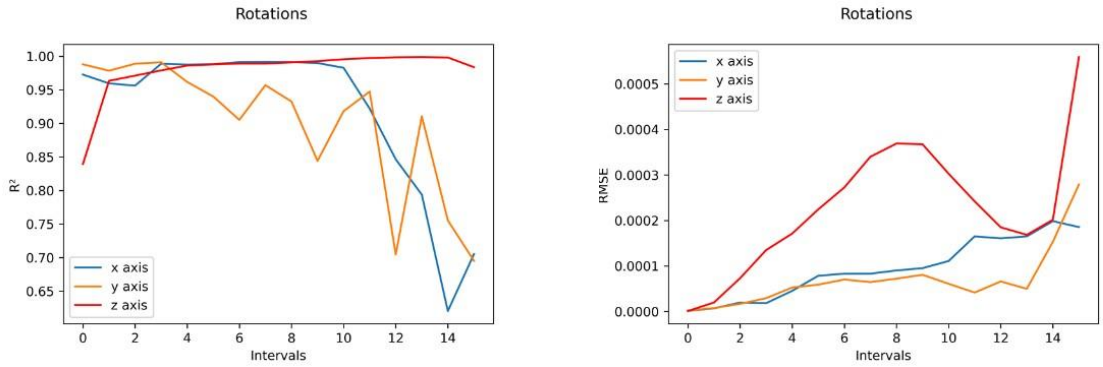

Figure 8: *R^2 and RMSE for Rotations using Random Forest Regression.*

Utilizando el conjunto de entrenamiento del segundo enfoque, que contiene cualquier tipo de colisión, los resultados fueron los siguientes:

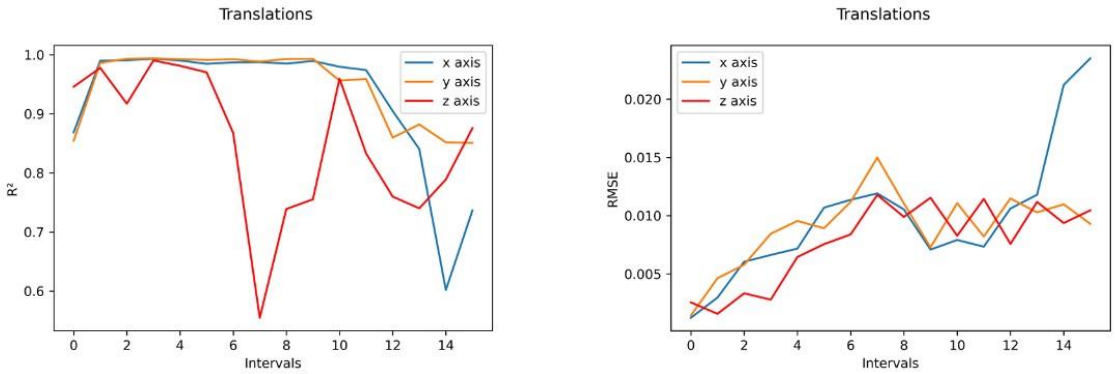

Figure 9: *R^2 and RMSE for Translations using Random Forest Regression.*

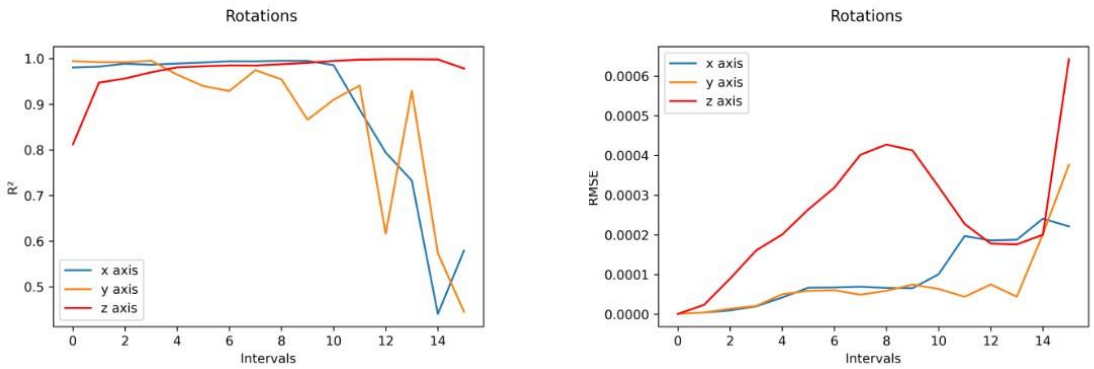

Figure 10: *R^2 and RMSE for Rotations using Random Forest Regression.*

Como se ha mencionado anteriormente, las métricas finales utilizadas para evaluar las predicciones y la precisión del modelo fueron el RMSE (error cuadrático medio) y el R2. En estas figuras, los tres ejes se representan a través de los diferentes intervalos. Estos intervalos representan cada uno de los puntos del pulso de choque a predecir (por tanto, hay 16 intervalos para cada pulso de choque y cada rotación o traslación del eje).

5. Conclusiones

Tras el análisis de los resultados y la comparación entre los distintos modelos del primer enfoque, se sacaron algunas conclusiones. En primer lugar, los modelos obtuvieron resultados diferentes y cada uno de ellos fue mejor en algunas partes que en otras. Sin embargo, fue el modelo de Regresión de Random Forest el que pareció predecir valores más cercanos a los verdaderos en todos los intervalos. Una posible razón de estos aparentes buenos resultados podría ser que el conjunto de datos utilizado para el entrenamiento era muy similar al utilizado en las pruebas. Por ello, como ya se ha explicado, se realizó una segunda aproximación, utilizando el conjunto de datos de entrenamiento general recogido (incluyendo todos los tipos de simulaciones) para determinar si los resultados eran fiables o, en caso contrario, qué tipo de resultados podían esperarse en condiciones no sesgadas.

Después de este nuevo enfoque, aparecieron algunas ideas nuevas:

- En cuanto a las traducciones, se observó que los valores de RMSE se redujeron ligeramente. Este hecho fue sorprendente ya que, tras introducir nuevos y más variados conjuntos de datos para el proceso de entrenamiento, cabía esperar un mayor RMSE. Sin embargo, esto significó que todos los modelos pudieron adaptarse a este nuevo conjunto de entrenamiento, a pesar de tener más variaciones de las que aprender. En cuanto al R^2, en términos generales, los valores eran ligeramente mayores, lo que significaba una mejora con respecto al primer enfoque. Sin embargo, el comportamiento general siguió siendo muy similar, con valores cercanos a 1 para los ejes “x” e “y”, y con mayores variaciones para el eje z.

- En cuanto a las rotaciones, los resultados no fueron tan prometedores como en las traslaciones. En este caso, el RMSE aumentó ligeramente, a pesar de ver que la tendencia general de estas curvas no era tan ruidosa (entendiendo por ruidosa las curvas con muchos valores de pico y sin tendencia clara apreciada). En cuanto al R2, los modelos obtuvieron resultados drásticamente diferentes entre sí. Mientras

que la Regresión del Random Forest y la Regresión del Gradient Boost siguieron obteniendo buenos resultados, la Red Neural y el SVR vieron afectado su rendimiento. Estos dos últimos modelos parecían tener problemas para obtener buenos valores de R^2 en algunos intervalos a pesar de mantener resultados relativamente buenos en otros. Este nuevo comportamiento podría ser el resultado de la introducción de estos nuevos conjuntos de datos para el proceso de entrenamiento que, en las rotaciones, dio lugar a un peor rendimiento en algunos modelos que en la primera aproximación. Estos valores de R2 fueron en algunas ocasiones inferiores a 0, lo que significa que el modelo de predicción no es capaz de hacer mejores predicciones que establecer el valor predicho como la media de los valores objetivo.

Después de todas estas aproximaciones, la decisión final fue elegir el Random Forest como el algoritmo final a utilizar, ya que este modelo obtuvo buenos resultados en ambas aproximaciones. Una de las principales preocupaciones de este modelo es que fácilmente sobreajuste sus predicciones, lo que hace que sea complejo de entrenar. Sin embargo, tras la segunda aproximación, se vio que, a pesar de introducir una variedad diferente de simulaciones, el modelo seguía siendo capaz de hacer buenas predicciones para estas configuraciones específicas de choque lateral. Además, se vio que este modelo también era capaz de tener los mismos valores de rendimiento al introducir los datos de entrada sin una fase previa de normalización, lo que puede ser muy útil a la hora de utilizar este modelo a posteriori.

Otra decisión que había que tomar era qué enfoque era el más adecuado. La decisión final fue mantener el segundo enfoque, ya que ayudaba al modelo a aprender de diferentes configuraciones de choque, lo que podría ser útil para futuros desarrollos. No sólo eso, sino que también se eligió porque el primer enfoque tenía algunas limitaciones, que se explicarán en la siguiente sección.

Por lo tanto, el modelo elegido fue capaz de hacer buenas predicciones, teniendo unas puntuaciones de R^2 y RMSE relativamente buenas.

5.1. **Limitaciones**

En cuanto a las limitaciones encontradas a lo largo del proyecto, a continuación, se enumeran las principales:

- La cantidad de datos recogidos, como se ve en algunas figuras, podría no ser lo suficientemente variada, lo que hace difícil entrenar y probar el modelo con la certeza de que este modelo no estará sobreajustando los datos. Por lo tanto, para los próximos desarrollos, sería necesario no sólo recoger más datos, sino también más variados.
- El enfoque del modelo se basó en el aprendizaje supervisado. Sin embargo, hoy en día existen varios enfoques con aprendizaje no supervisado que podrían ser adecuados para este proyecto. Con ello, los pulsos de choque podrían no necesitar ser recortados y discretizados en diferentes intervalos, aumentando el uso directo de este modelo para ser utilizado posteriormente.
- Los datos de entrada tendrían que ser más variados al introducir más modelos de coches. El modelo funciona con 7 parámetros, pero si este modelo se utilizara con diferentes marcas de coches, tipos de coches, propiedades mecánicas como la rigidez, u otros parámetros, podría resultar un algoritmo más viable y robusto.

6. Referencias

[1] Leledakis, A., 2020. “A method for predicting crash configurations using counterfactual simulations and real-world data".

[2] Wågström, L., 2019. “Integrated Safety: Establishing Links for a Comprehensive Virtual Tool Chain”.

CRASH PULSE PREDICTION APPLIED TO LATERAL CRASH CONFIGURATIONS

Author: Rodríguez González, Miguel.
Director: Francisco José López Valdés.
Collaborating Entity: Volvo Cars

ABSTRACT

Project focused on the prediction of crash pulses produced in the in-side collision. Using methodologies to extract and process data to obtain an initial configuration that describes the positioning of both vehicles, a Machine Learning method has been developed to make the prediction of the "Crash Pulse" and a subsequent evaluation of the prediction made comparing with the results that were obtained in the simulations of these collisions.

Keywords: AI, Data Analysis, Crash Pulse, Safety.

1. Introduction

Occupant safety has been and will be a key attribute of all vehicles. Current methodologies to evaluate and improve occupant protection are based on physical and virtual crash tests. The continuous evolution of simulation software and Finite Elements (FE) methodologies have contributed to a great economic cost reduction. However, these simulations are computationally expensive and, therefore, some limitations appear in terms of the amount of both physical and virtual crash tests that can be performed.

2. Project definition

This need is translated into this project, where, aiming to use Big Data and ML solutions, it will have as the final objective to assist to both physical and virtual crash tests. This project, then, is established as a progression of previous studies ([1] and [2]) and aims to develop a virtual tool capable of assisting the current methodologies of assessing the effects of a vehicle crash test in different configurations.

Therefore, the objective of this project is to investigate numerical methods to predict the crash pulse of a vehicle's lateral-impact, using simulation data. To get a clear idea of the different tasks to be performed, a list has been defined:

- Identify the feasibility and limitations of existing methods for the application of vehicle safety.
- Create a method that automatically analyzes the test and simulation data and classifies the impact conditions, according to the crash configurations detailed in [1] and [2].
- Implement and train a method that predicts crash pulses given a crash configuration.
- Assess the accuracy of predicted crash pulses.

3. Description of the model/system/tool

The following project aims to develop a method that will be capable of analyzing simulation data (virtual crash tests) to be used as input of a Machine Learning algorithm to predict the crash pulse suffered in each specific lateral crash configuration given. To create such a methodology, the approach will be divided into two main parts: Obtaining the necessary data and creating the algorithm using ML for prediction.

3.1. Data gathering.

In this section of the project, the method to follow consists of a deep analysis of the data available at Volvo Cars particularly applied to car-to-car simulations. The next steps to develop this phase are inspired by the methodology followed in previous studies [1], which can be seen in the following figure:

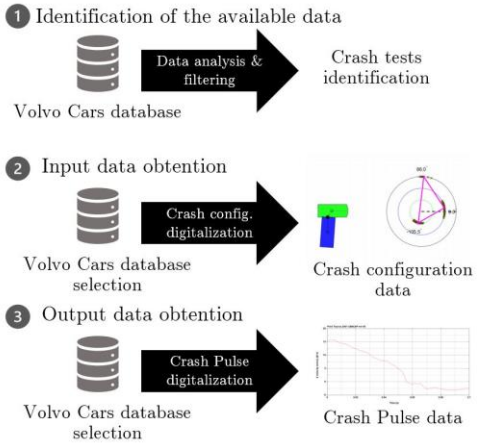

Figure 1 *Overview of the three steps of the data collection method. Step 1: Identification of available data; Step 2: Extract input data. [2]; Step 3: Extract output data.*

3.1.1. Step 1.

The initial starting point of this method consists of filtering and analyzing the data available. The main purpose of this step is to gather the data that corresponds to what the algorithm needs to work with or what the whole project would like to be based on. Not only in terms of crash configurations but also terms of virtual crash data. It is necessary to establish a method to obtain the required parameters and evaluate what kind of crash configurations will be used in the algorithm.

3.1.2. Step 2.

Once the crash tests are chosen, the method will extract the required parameters from the crash tests files. By doing so, the main goal is to automatically parse through all the required data that can define the crash configuration of the collision. Moreover, when analyzing these data, it is important to point out that each crash can be categorized into two crash configurations, depending on which vehicle is considered to be the host vehicle. The host vehicle is the vehicle where the occupant at risk is analyzed.

3.1.3. Step 3.

Following the previous methodology, the algorithm will also parse through the data to obtain the crash pulse (3 axis rotations and translations) that is required to train the predictive algorithm. Once again, it is important to consider that both vehicles will be

considered as the host vehicle for different crash configurations, and, therefore, the crash pulse time series will be extracted and gathered. By following this data gathering method, the ML algorithm will be provided with all the required data for the next method.

3.2. ML method.

At this point of the process, the main difficulties will be related to the selection of the ML model, how to train it, which data are to be used at the training and the test phase, and finally, how to evaluate the crash pulse predictions. Following the Data Gathering method, this phase could be divided into the following steps:

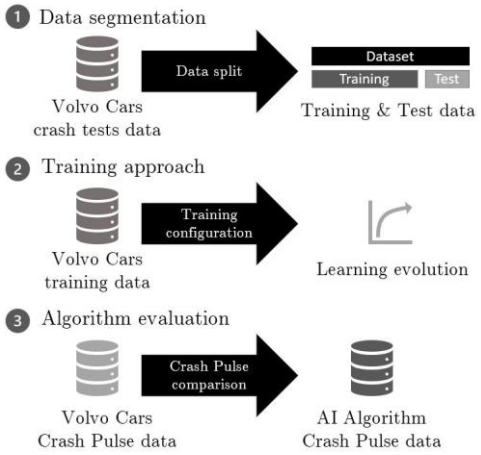

Figure 2 *Overview of the three steps of the ML method. Step 1: Data segmentation; Step 2: Training approach; Step 3: Algorithm evaluation.*

Before being able to apply such a methodology, it is important to focus the decision on what ML model will be used and what approach will be taken to the learning process. In general terms, this ML approach allows having a learning process capable of taking into account not only the initial crash configuration but also the crash pulses already extracted to develop future predictions.

3.2.1. Step 1.

Once the model is chosen, the ML algorithm method can be developed. As stated before, it is of crucial importance the way the data is split and managed. Depending on the decision on how to split the data, the performance of the algorithm when testing can vary drastically.

3.2.2. Step 2.

After the data is split, the training process can begin. The ML algorithm works with the input and output data to learn how the data is structured and how it is connected. Then it starts predicting possible outputs for the given input data. Furthermore, the method compares its predictions to the actual output data and learns during the process by iterating and making some modifications on different parameters such as the weights and bias parameters. These parameters are responsible for the learning process of the algorithm, being changed depending on the accuracy that the model achieves. By doing so, the model can learn and predict the crash pulse of the different crash configurations. However, there must be some careful considerations when training the model, as an over or under-trained model can be problematic. These considerations include making sure

the data is varied and it represents as much as possible the different simulations that would like to be predicted.

3.2.3. Step 3.

The evaluation phase can be, at certain phases, performed in parallel to the training phase. The main aim of this step is to test the trained model with the designated test data, analyzing how the model is capable of predicting the crash pulse curves based on the crash configurations given. As explained before, this will have a strong connection with how the data is segmented and split.

4. Results

During the procedure of obtaining the datasets and the predictions, some issues were encountered and faced. These will be explained in the next sections.

4.1. Data gathering method.

To obtain the desired datasets, different approaches were needed. First of all, the available data could be divided into different simulation scenarios: car-to-car simulations and car-to-moving deformable sled simulations. Due to the small dataset gathered from car-to-car simulations, more varied data were needed, and therefore, the approach was to obtained car-to-moving deformable sled simulation data, which can resemble the collision of two cars

The algorithm for car-to-car simulations was developed to obtain the different variables from the files that correspond to the output of a simulation performed using LsDyna (called *d3plot* and *binout* files). To do so, the different positions of both vehicles were obtained through clustering the nodes of both vehicles. These nodes are the different points that the elements and parts of each vehicle are made of. By clustering these, the two vehicles can be identified by the algorithm and the variables can be extracted. Therefore the next step was to obtain the velocities of each vehicle, to measure the angles of the crash configuration, and to obtain the masses of the vehicles (For more information, see *Appendix II*).

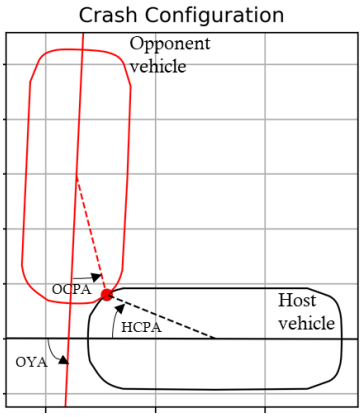

Figure 3: Getting the lock settings through the output files from the simulation.

The approach for the car-to-moving deformable sled simulations was very similar, although some variations were introduced to solve the lack of specific nodes linked to

the accelerometers of the actual vehicles. These accelerometers were used to obtain the different velocities of the car. As these were not the same for the moving deformable sled, the approach was developed by detecting the velocity of the nodes and extracting the value for the whole set. This was performed after the clustering method which helped differentiate between the car and the moving deformable sled or moving barrier (For more information, see *Appendix II*).

Furthermore, a final step was introduced to extract the crash pulse of the simulations (accelerations of 3-axis translation and rotation). The methodology was similar for both car-to-car simulations and car-to-moving deformable sled simulations. As for the car-to-car simulations, the time series containing the crash pulses were obtained from the simulation, extracting the 3-axis translations and rotations in the shape of velocities. As for the car-to-moving deformable sled simulations, in this case, instead of extracting the accelerations of both vehicles, only the accelerations for the host vehicle were obtained.

The final approach to gathering these crash pulses datasets was oriented to the ML method (For more information, see *Appendix II*). First of all, these time series were converted to accelerations, as they are easier to understand and have a better feeling of what is happening during the crash. As the complexity of the model would increase drastically by introducing the different time series of the crash pulses, a data-reduction method was developed. To simplify these datasets, a discretization of the time series was applied. Therefore, the time series to analyze were cut at time 80 ms to have the same format and number of points of every output crash pulse. The resulting time series were split into several intervals. From each interval, the mean was computed and extracted as the discrete value, reducing each time series to a dataset of 16 mean values. This number of points consisted of one value per 5 ms, which described the motion and the trend of the mentioned crash pulse. An example of the different mean values can be seen below:

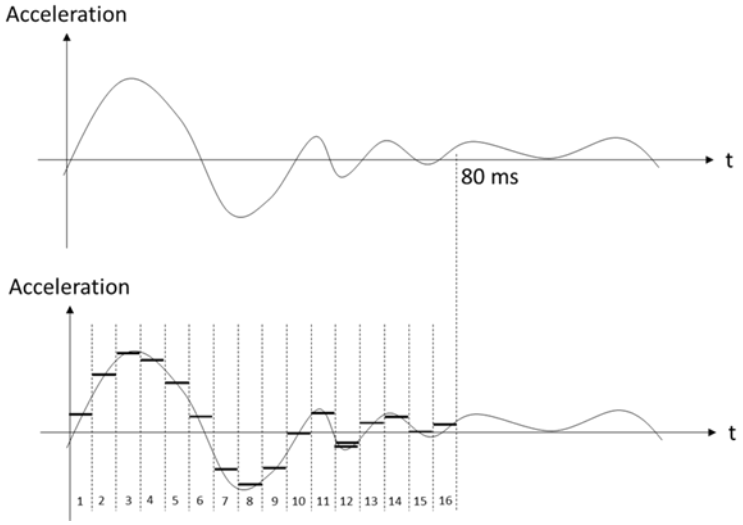

Figure 4: Time series transformation.

4.2. ML method

When the data was gathered, a pre-processing phase to transform the input data into something that the ML algorithm could understand was needed.

First of all, the angles were converted to coordinates of modulus 1, to help the algorithm understand concepts such as 180 degrees and -180 degrees are, in fact, the same.

A dataset of almost 500 different simulations was created, with the following format:

$HCPA_X$	$HCPA_Y$	$OCPA_X$	$HCPA_Y$	OYA_X	OYA_Y	HS	OS	MASS H	MASS O
-0.88	-0.39	0.99	0.53	0.92	-0.02	-0.57	-0.54	-1.06	-0.69
0.99	0.53	-0.88	-0.39	0.92	-0.02	-0.54	-0.57	1.43	-0.69
0.18	1.93	-1.76	1.51	-0.72	1.53	-0.57	0.68	-0.85	-0.69

Table 1: *Sample of a normalized input data frame for the ML algorithm.*

4.2.1. Training-Test Data split

The next step was to select the validation data that would help evaluate the algorithm and that would be significant to take into account the different kinds of simulations gathered. For this purpose, a graph including the 5 main variables that describe the crash configuration was plotted:

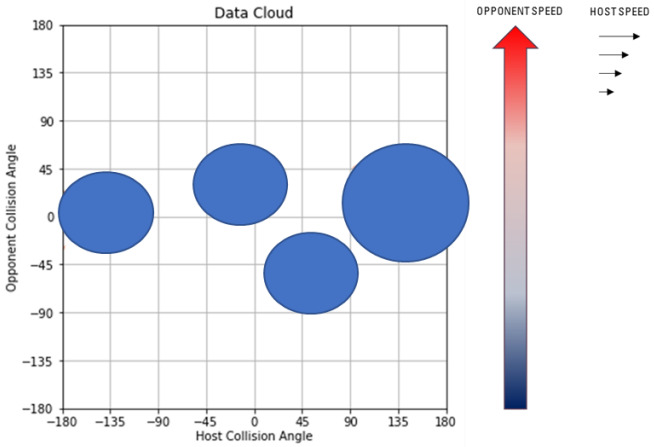

Figure 5: Data of the *different simulations collected (HCPA vs OCPA with OYA, HS, and OS included). Due to confidentiality, no arrows could be shown.*

From this data cloud, it is necessary to extract those attributed as lateral crashes. To do so, the HCPA (Host Collision Point Angle) will be the corresponding variable indicating which crashes correspond to a lateral configuration or not. As it has been explained before, the HCPA corresponds to the angle between the direction of the car and the FPOC (First Point of Contact). Regarding normalized dimensions of the car, it can be seen that the angles corresponding to a lateral crash will correspond with the intervals of (45º, 135º) and (-135º, -45º) as it can be seen below:

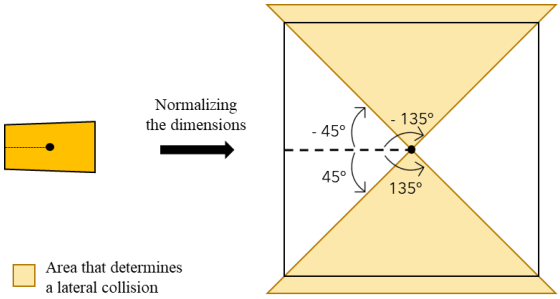

Figure 6: *Criterion for determining lateral collisions.*

This figure represents the vehicle normalized, meaning that the frontal and lateral dimensions are normalized to a value of 1. With this step, the HCPA and the OCPA were measured.

By applying this selection, the resulting data cloud with the lateral collisions is reduced to an amount of 100 simulations. From this data cloud, the training and test datasets were divided in a proportion of 80% and 20% respectively (For more information, see *Appendix II*). The segmentation was performed by randomly splitting the data and analyzing the split applied. The main objective was to obtain a well-distributed test dataset that could be representative enough of the simulations gathered, aiming to validate the different configurations extracted. By saving the training and test datasets, the different models used can be easily compared and evaluated. After some trials and checks on the performance of the different ML algorithms, the final dataset was chosen

When choosing the ML approach, multiple algorithms were developed. Therefore, the final models consisted of four methods: Multilayer Perceptron (MLP) using Neural Networks (NN), Random Forest Regression, Support Vector Machine Regression, and Gradient Boost Regression (For more information, see *Appendix II*). Due to the results obtained (shown in *Appendix* I), two different approaches were done. Initially, as explained before, the training data set was created from the data cloud corresponding to only lateral collisions. Therefore, the model was both trained and tested with lateral collisions.

Secondly, after analyzing the results, a second training set was created. This set was chosen from the original data cloud before differentiating between lateral and non-side collisions. This means that the second training data set had lateral and non-side collisions including frontal, rear, and oblique collisions. Therefore, the model was trained with any type of collision but tested only with lateral collisions.

These two approaches aimed to verify that in the first approach, the models were not overfitting the data, as the amount of data introduced was limited, and maybe, biased. To have a better understanding and check how reliable these predictions were, the second approach was applied

It is important to note that the model approach was developed in terms of the type of acceleration. One model was trained with the entire data set of translational accelerations and another model with rotations. This division was chosen to avoid having misleading

data in the models. Not only that but the three axes were also used together, as these accelerations may be also connected. Therefore, the final configuration of the ML approach consisted of a model that predicted translational accelerations and another model for rotational accelerations.

After using these four methods, the results of the predictions were better when using the Random Forest Regression model. This model is a supervised learning algorithm that uses the set learning method for regression. As a brief description, set learning is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. Therefore, this approach works by creating multiple decision trees during the training phase, predicting as many output values as the model has decision trees. The final output consists of the average of the different predicted values of all the included decision trees. These decision trees are developed in parallel, avoiding any kind of influence between them and using different parts of the training dataset for each tree. This model has the advantage that the performance does not vary depending on whether the data is normalized or not. For this reason and to simplify the methodology, this model did not require a normalization step.

Using the training set of the first approach, with only lateral simulations to train the model, the results were the following:

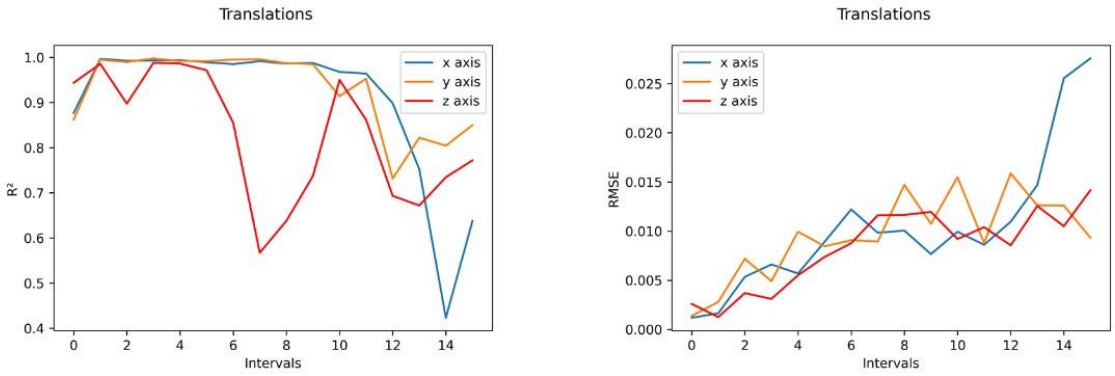

Figure 7: *R^2 and RMSE for Translations using Random Forest.*

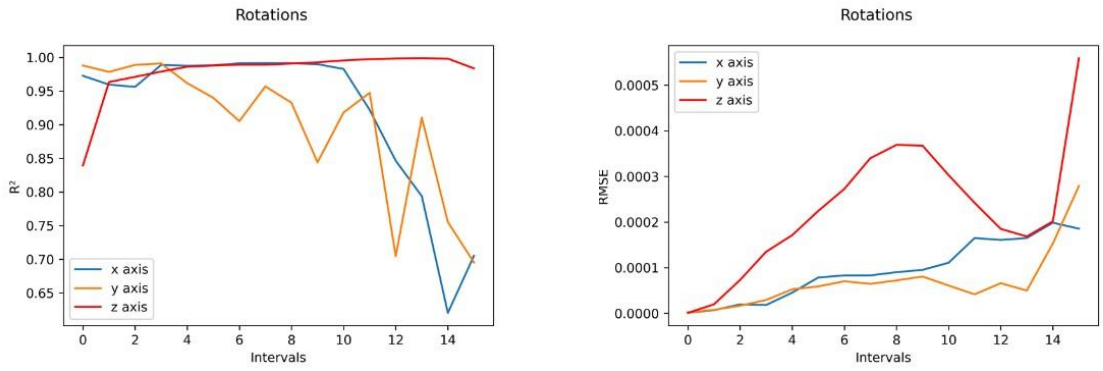

Figure 8: *R^2 and RMSE for Rotations using Random Forest Regression.*

Using the training set of the second approach, containing any type of collision, the results were the following:

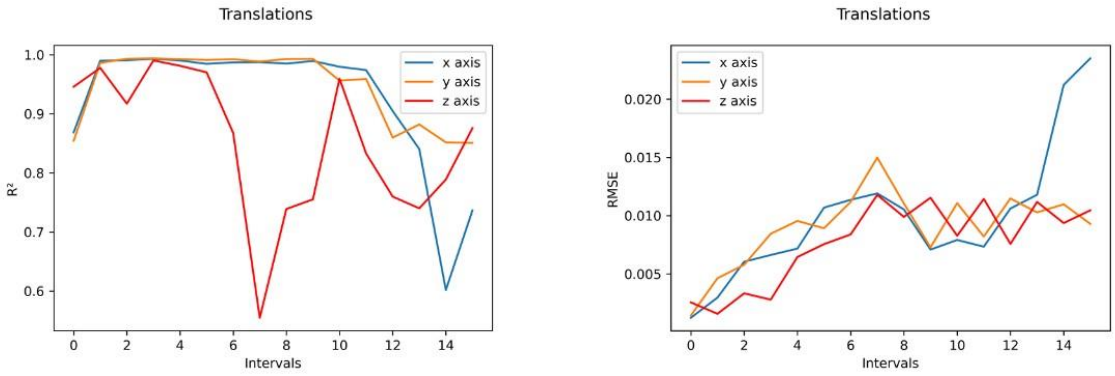

Figure 9: *R^2 and RMSE for Translations using Random Forest Regression.*

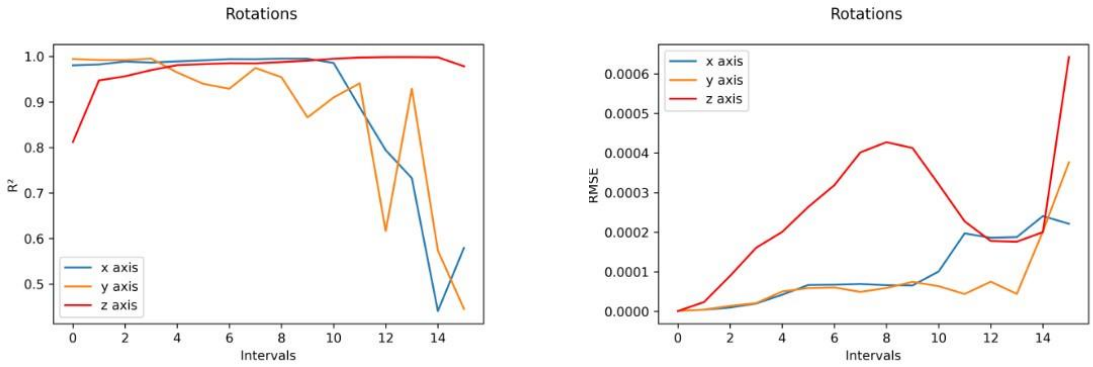

Figure 10: *R^2 and RMSE for Rotations using Random Forest Regression.*

As mentioned above, the final metrics used to evaluate predictions and model accuracy were the RMSE (Root Mean Square Error) and $R^{2.}$ In these figures, the three axes are represented through the different intervals. These intervals represent each of the points of the crash pulse to be predicted (therefore, there are 16 intervals for each crash pulse and each rotation or translation of the axis).

5. Conclusions

After the analysis of results and comparison between the different models of the first approach, some conclusions were made. First of all, the models had different results and each of these was better at some parts than others. However, it was the Random Forest Regression model that seemed to be predicting closer values to the true values throughout all the intervals. One possible reason for these apparent good results could be that the data set used for the training was very similar to the one used in testing. Because of this, as explained before, a second approach was performed, using the general training dataset gathered (including all types of simulations) to determine if the results could be trusted or if not, what kind of results could be expected under unbiased conditions.

Some new insights appeared after this new approach was performed:

- In terms of the translations, it was seen that the RMSE values were reduced slightly. This fact was surprising as, after introducing new and more varied sets of data for the training process, a larger RMSE could be expected. However, this meant that all the models could adapt to this new training set, despite having more variations to learn from. As for the R2, in general terms, the values were slightly

larger, meaning an improvement from the first approach. However, the general behavior remained very similar, having values close to 1 for the x and y axes and seeing larger variations for the z-axis.

- In terms of the rotations, the results were not as promising as in the translations. In this case, the RMSE increased slightly, despite seeing that the general trend of these curves was not as noisy (noisy understood as curves with a lot of peaks values and not clear trend appreciated). As for the R2, the models obtained drastically different results between them. While the Random Forest Regression and the Gradient Boost Regression remained to obtain good results, the Neural Network and SVR saw their performance affected. These two last models seemed to have problems obtaining good R2 values at some intervals despite keeping relatively good results at some others. This new behavior could be a result of the introduction of these new datasets for the training process which, in rotations, resulted in worse performance in some models than in the first approach. These R2 values were on some occasions lower than 0, which means that the prediction model is not capable of making better predictions than establishing the predicted value as the mean of the target values.

After all these approaches, the final decision was to choose the Random Forest as the final algorithm to be used, as this model performed good results in both approaches. One of the main concerns of this model is that it easily overfits its predictions, making it complex to train. However, after the second approach, it was seen that despite introducing a different variety of simulations, the model was still capable of making good predictions for these specific lateral crash configurations. Furthermore, it was seen that this model was also capable of having the same performance values when introducing the input data without a previous phase of normalization, which can be very helpful when using this model afterward.

Another decision to make was what approach was the most suitable. The final decision was to keep the second approach, as it helped the model learn from different crash configurations, which could be helpful for future developments. Not only that but also it was chosen because the first approach had some limitations, which will be explained in the next section.

Therefore, the chosen model was capable of making good predictions, having relatively good R2 and RMSE scores.

5.1. **Limitations**

In terms of the limitations found throughout the project, the main ones will be listed below:

- The amount of data gathered, as seen in some figures, could be not varied enough, making it difficult to train and test the model with certainty that this model will not be overfitting the data. Therefore, for the next developments, not only more data but more varied data would be needed to be gathered.
- The model approach was based on supervised learning. However, nowadays exist several approaches with unsupervised learning that could be suitable for this project. With so, the crash pulses might not need to be cropped and discretize in different intervals, increasing the direct use of this model to be used afterward.

- The input data would need to be more varied when introducing more car models. The model is working with 7 parameters but if this model were to be used with different car brands, car types, mechanical properties such as stiffness, or other parameters, it could result in a more viable and robust algorithm.

6. References

[1] Leledakis, A., 2020. “A method for predicting crash configurations using counterfactual simulations and real-world data".

[2] Wågström, L., 2019. “Integrated Safety: Establishing Links for a Comprehensive Virtual Tool Chain”.

Index

Index of figures

Index of tables

Chapter 1. INTRODUCTION

Occupant safety has been and will be a key attribute of all vehicles. Current methodologies to evaluate and improve occupant protection are based on physical and virtual crash tests. The continuous evolution of simulation software and Finite Elements (FE) methodologies have contributed to a great economic cost reduction. However, these simulations are computationally expensive and, therefore, some limitations appear in terms of the amount of both physical and virtual crash tests that can be performed.

Furthermore, this study will be focused on lateral collisions. These collisions are defined as those where the lateral of a vehicle is impacted. These typically occur at intersections or parking lots, where changes in direction are necessary. To contextualize these collisions, for fatalities in the US, it is estimated that around 22% of the number of people killed in vehicles were struck on the lateral ([1] Wikipedia, 2009).

Chapter 2. STATE OF THE ART

Traffic safety technologies are based on two principal ideas: crash avoidance and injury mitigation for inevitable crashes. Following this trend, there is research on ways to facilitate the assessment of future traffic safety. One aspect in particular that has been developed is a method for predicting crash configurations when introducing crash-avoiding countermeasures. This approach facilitates the correct prioritization of the evaluation and development of future occupant and Vulnerable Road User (VRU) protection systems ([2] Leledakis, 2020). Due to the development of such a method, which clustered and grouped different types of crashes into several crash configurations, the results demonstrated a feasible predictive way for in-crash testing of injury prevention capabilities.

On the same page, there have been several studies that apply this new configuration system to different crash scenarios. In combination with other new approaches, this has created new ways of analyzing and predicting the combined safety performance. Therefore, it can be used for transferring output from pre-crash simulations to input for crash simulations ([3] Wågström, 2019).

2.1 *OBJECTIVES*

This project aims to investigate numerical methods of predicting the acceleration pulse of a vehicle impact, using test and simulation data. To have a clear understanding of the different tasks to be completed, a list has been defined:

- Identifying feasibility and limitations of existing methods for the application of vehicle safety.

- Create a method that automatically parses through test and simulation data and classifies the impact conditions, according to the crash configurations detailed in ([3] Wågström, 2019) and ([2] Leledakis, 2020).

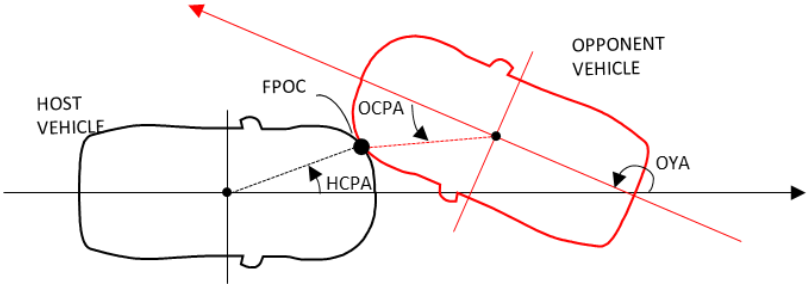

Figure 1: *Definition of VPARCC (Volvo parametric crash configuration) angles in host and opponent vehicle.*

- Implement and train a method that predicts crash pulses given a crash configuration.
- Evaluate the accuracy of the predicted crash pulses.
- Documenting the results in a thesis report.

2.2 *MOTIVATION*

In parallel to the previous studies, new research surges as this Master Thesis to develop an algorithm that is capable of not only classifying every crash into a crash configuration but also predicting the crash pulse of such crash test from a crash configuration given.

As stated before, one of the main disadvantages of using the current methodologies is that they contribute to a great economic cost in terms of the physical crash tests and they are computationally expensive in terms of virtual crash simulations. Therefore, it would be beneficial to reduce, both the economic and the computational cost of these crash tests, as these are fundamental to understand how the vehicles behave in these situations and how they affect the occupants.

This need is translated into this project, where, aiming to use Big Data and ML solutions, it will have as the final objective to assist to both physical and virtual crash tests. This project, then, is established as a progression of previous studies (([2] Leledakis, 2020) & ([3]

Wågström, 2019)) and aims to develop a virtual tool capable of assisting the current methodologies of assessing the effects of a vehicle crash test in different configurations.

2.3 *SCHEDULE*

The schedule for the work is presented in the form of a Gantt chart (flow chart). The main phases and requirements have been divided into different sub-phases to be performed during the completion of this project.

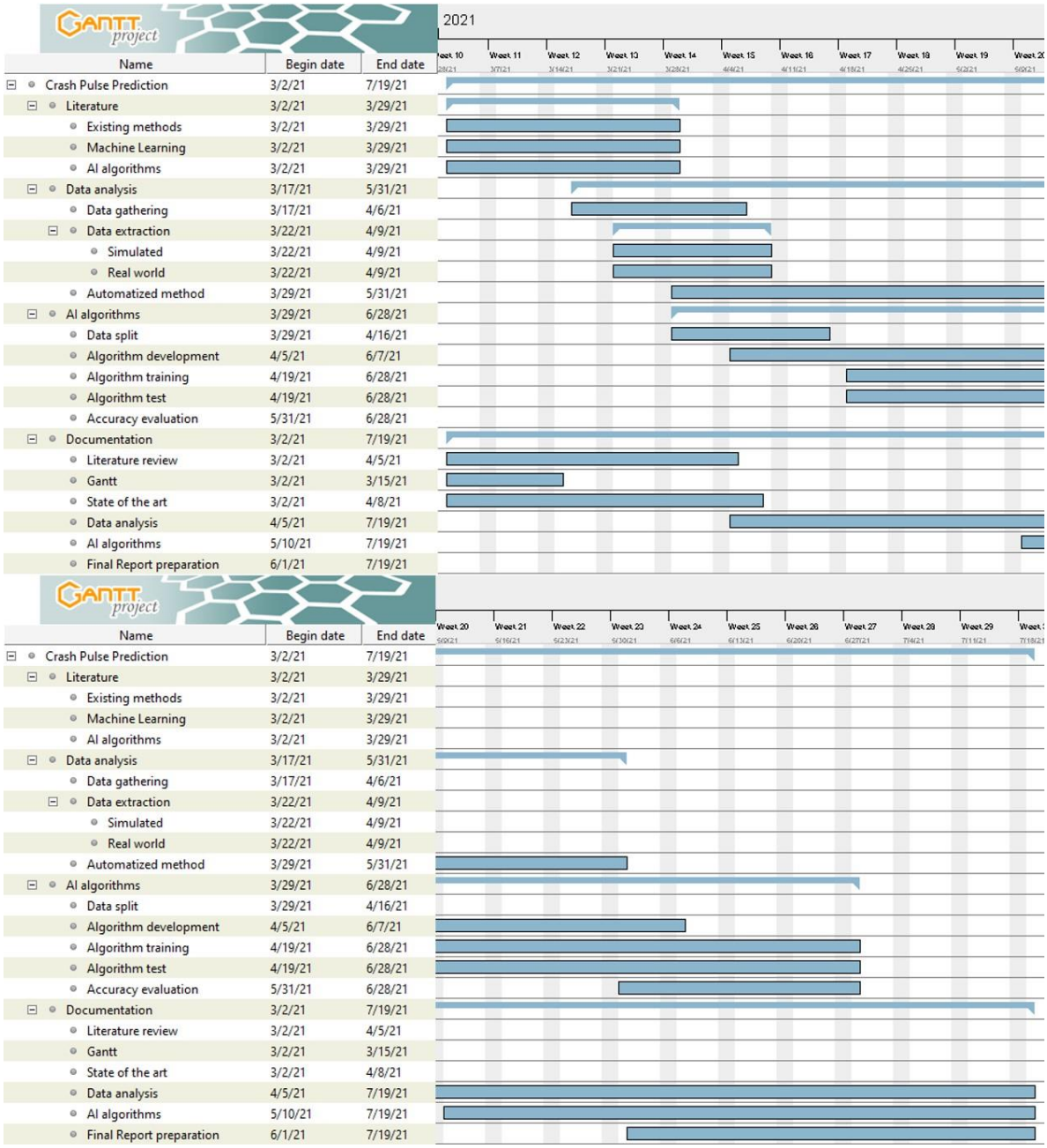

Figure 2: *Gantt chart of the project.*

Chapter 3. METHODOLOGY

The following project aims to develop a method that will be capable of analyzing simulation data (virtual crash tests) to be used as input of a Machine Learning algorithm to predict the crash pulse suffered in each specific lateral crash configuration given. To create such methodology, the approach will be divided into two main parts:

1. To have data to be used as the input for the mentioned algorithm, developing a specific method to gather it is needed. The data will require an analysis to develop the method that automatically parses through simulation data and classifies the impact conditions. Not only the desired input data but also the crash pulses resulting from these tests will be needed- These crash pulses will be used in the training algorithm and to evaluate the final results.
2. Once the required data is gathered, it will be introduced to an algorithm that will be trained with certain simulation tests. Then, it will be tested to measure its actual performance in predicting actual virtual tests results.

Therefore, the method could be distinguished into two main parts, defined as Data Gathering Phase and ML Phase.

3.1 *DATA GATHERING METHOD*

In this section of the project, the method to follow consists of a deep analysis of the data available at Volvo Cars particularly applied to car-to-car simulations. The next steps to develop this phase are inspired by the methodology followed in previous studies ([2] Leledakis, 2020), which can be seen in the next figure:

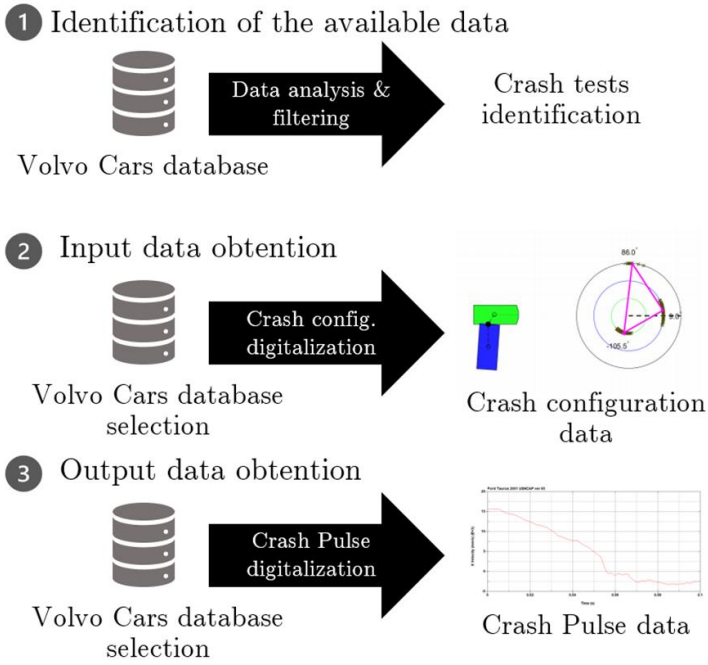

Figure 3: *Overview of the three steps of the Data Gathering method. Step 1: Identification of the available data; Step 2: Input data extraction* ***([3] Wågström, 2019)****; Step 3: Output data extraction.*

3.1.1 STEP 1

The initial starting point of this method consists of filtering and analyzing the data available. The main purpose of this step is to gather the data that corresponds to what the algorithm needs to work with or what the whole project would like to be based on. Not only in terms of crash configurations but also terms of virtual crash data. It is necessary to establish a method to obtain the required parameters and evaluate what kind of crash configurations will be used in the algorithm.

3.1.2 STEP 2

Once the crash tests are chosen, the method will extract the required parameters from the crash tests files. By doing so, the main goal is to automatically parse through all the required data that can define the crash configuration of the collision. Moreover, when analyzing these data, it is important to point out that each crash can be categorized into two crash configurations, depending on which vehicle is considered to be the host vehicle. The host vehicle is the vehicle where the occupant protection is assessed.

3.1.3 STEP 3

Following the previous methodology, the algorithm will also parse through the data to obtain the crash pulse (3 axis rotations and translations) that is required to train the predictive algorithm. Once again, it is important to consider that both vehicles will be considered as the host vehicle for different crash configurations, and, therefore, the crash pulse time series will be extracted and gathered. By following this data gathering method, the ML algorithm will be provided with all the required data for the next steps (training & accuracy testing).

3.2 *MACHINE LEARNING METHOD*

At this point of the process, the main difficulties will be related to the selection of the ML model, how to train it, which data are to be used at the training and the test phase, and finally, how to evaluate the crash pulse predictions. Following the Data Gathering method, this phase could be divided into the following steps:

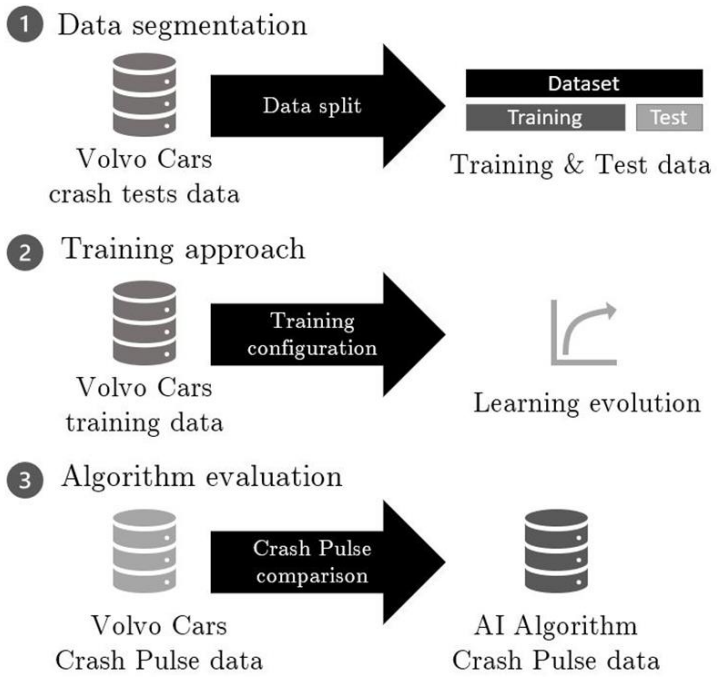

Figure 4: *Overview of the three steps of the ML method. Step 1: Data segmentation; Step 2: Training approach; Step 3: Algorithm evaluation.*

Before being able to apply such a methodology, it is important to focus the decision on what ML model will be used and what approach will be taken to the learning process. In general terms, this ML approach allows having a learning process capable of taking into account not only the initial crash configuration but also the crash pulses already extracted to develop future predictions.

3.2.1 STEP 1

Once the model is chosen, the ML algorithm method can be developed. As stated before, it is of crucial importance the way the data is split and managed. Depending on the decision on how to split the data, the performance of the algorithm when testing can vary drastically.

3.2.2 STEP 2

After the data is split, the training process can begin. The ML algorithm works with the input and output data to learn how the data is structured and how it is connected. Then it starts predicting possible outputs for the given input data. Furthermore, the method compares its predictions to the actual output data and learns during the process by iterating and making some modifications on different parameters such as the weights and bias parameters. These parameters are responsible for the learning process of the algorithm, being changed depending on the accuracy that the model achieves. By doing so, the model can learn and predict the crash pulse of the different crash configurations. However, there must be some careful considerations when training the model, as an over- or under-trained model can be problematic. These considerations include making sure the data is varied and it represents as much as possible the different simulations that would like to be predicted.

3.2.3 STEP 3

The evaluation phase can be, at certain phases, performed in parallel to the training phase. The main aim of this step is to test the trained model with the designated test data, analyzing how the model is capable of predicting the crash pulse curves based on the crash configurations given. As explained before, this will have a strong connection with how the data is segmented and split.

Chapter 4. IMPLEMENTATION

During the procedure of obtaining the datasets and the predictions, some issues were encountered and faced. These will be explained in the next sections.

4.1 *DATA GATHERING METHOD*

To obtain the desired datasets, different approaches were needed. First of all, the available data, that satisfied the filtering criteria, could be divided into different simulation scenarios: car-to-car simulations and car-to-moving deformable sled simulations.

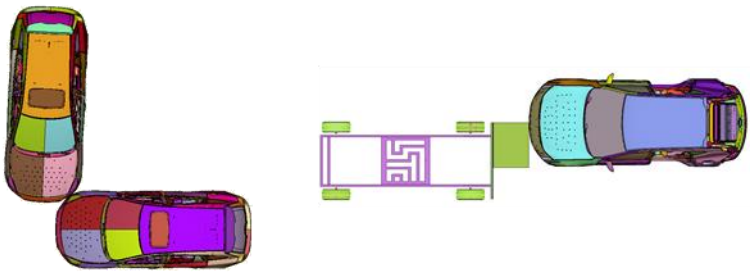

Figure 5: *Examples of a car-to-car crash (left) and car-to-moving barrier (right).*

Due to the small dataset gathered from car-to-car simulations, more varied data were needed, and therefore, the approach was to obtain car-to-moving deformable sled simulation data, which can resemble the collision of two cars.

The algorithm for car-to-car simulations was developed to obtain the different variables from the files that correspond to the output of a simulation performed using LsDyna (called *d3plot* and *binout* files). To do so, the different positions of both vehicles were obtained through clustering the nodes of both vehicles (For more information, see *Appendix II*). These nodes are the different points that the elements and parts of each vehicle are made of. By clustering these, the two vehicles can be identified by the algorithm and the variables can be extracted. Therefore the next step was to obtain the velocities of each vehicle, to measure the angles of the crash configuration, and to obtain the masses of the vehicles.

A representation of how the algorithm worked throughout the data can be seen below:

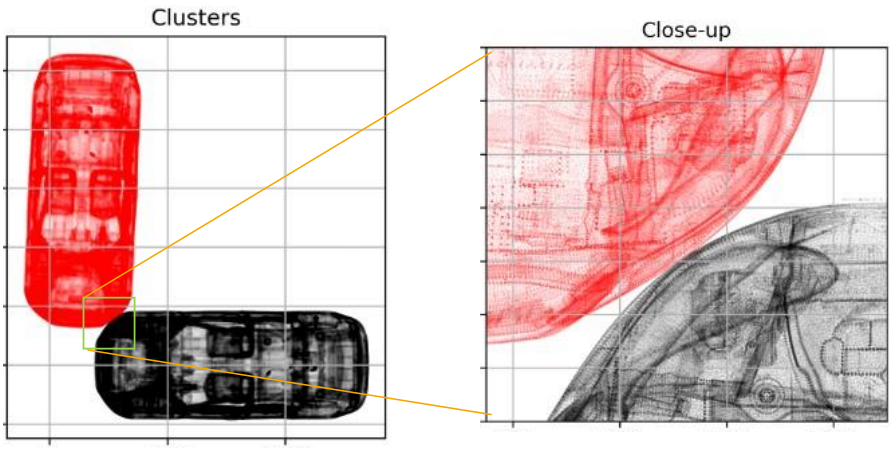

Figure 6: *Clustered nodes of different vehicles.*

To achieve such a resolution of the two clusters, several parameters were taken into account. First of all, the algorithm analyzed the different nodes that composed the file with the two vehicles, and after first filtering out some elements such as the ground, the clustering began. To do so, the different variables implicated to split the data cloud of nodes were the x and y velocities of all the nodes and the ids of the nodes These ids represent the name given to each node. They are numbered in different sequences, usually establishing a large offset between the nodes of different vehicles

After that, the following functions consisted of extracting the different data needed for the positional angles, the velocities, and the masses. To reduce the number of points used by the algorithm, the data cloud of the nodes of the vehicles was simplified to only those corresponding to the boundary of the vehicle, making it simpler to obtain the axes of each vehicle and therefore, the different angles This is represented in the figure below:

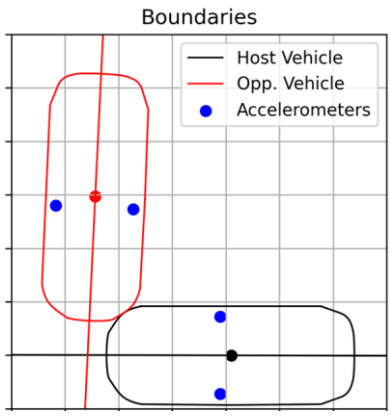

Figure 7: *Boundaries of the two vehicles along with the x-axis of each vehicle.*

Once this step was reached, the only parameter left to know was the FPOC (First Point Of Contact). By measuring the closest distance between the boundaries in the different timesteps, an accurate estimation could be extracted. The calculation then was estimated by measuring the minimum distance between the boundaries of the vehicles in different timesteps. Once the algorithm detected intrusion between the vehicles, the points with the closest distances in the previous timestep were chosen as the FPOC. An example can be seen below:

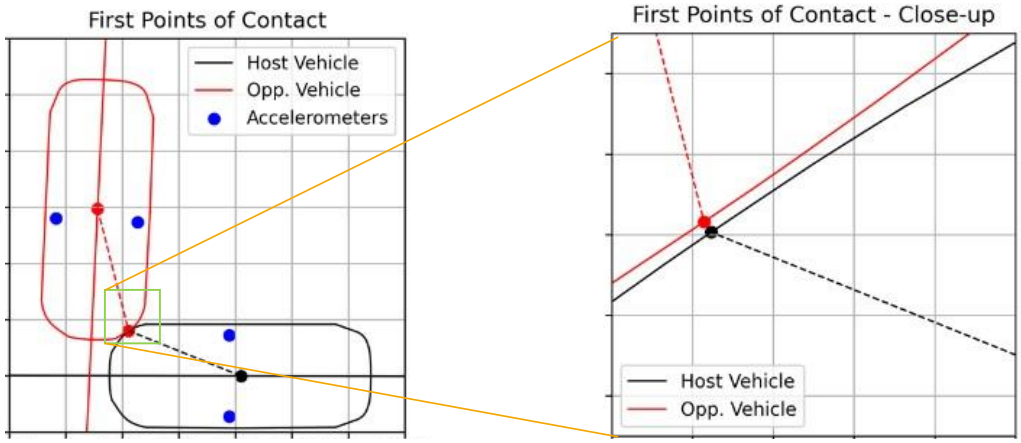

Figure 8: *Example of the extraction of FPOC.*

In addition to finding the FPOC, a normalization of the distances between the geometrical center of the vehicle and the FPOC was required studies ([2] Leledakis, 2020) : By doing so,

the different vehicle dimensions will not influence the values of the angles. Therefore, the FPOC and the angles were finally obtained based on this geometry:

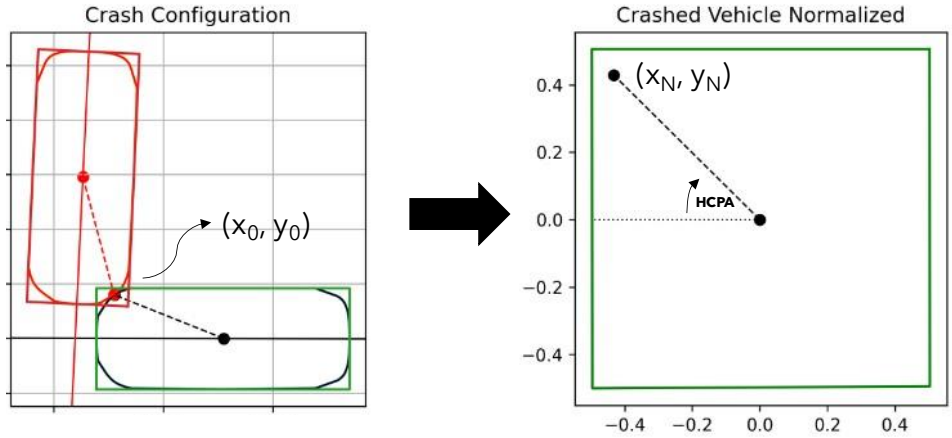

Figure 9: *Normalized FPOC coordinates*

This figure represents the vehicle normalized, meaning that the frontal and lateral dimensions are normalized to a value of 1. With this step, the HCPA and the OCPA were measured.

Another variable to take into account was the masses of the two vehicles involved. To get these two parameters, a similar process was developed. The output files of the simulations allow the user to have access to the masses of the different parts. Therefore, the process required to find which parts correspond to each vehicle. To do so, another cluster was developed (For more information, see *Appendix II*), this time taking as clustering parameters the velocities of the different parts (as those parts corresponding to a vehicle should move with the same velocity) and the parts ids (as happened before, usually these parts have an offset to differentiate between parts of different vehicles or models). By doing so, the masses corresponding to each clustered vehicle were extracted.

An example of the resulting crash configuration can be seen in the next figure:

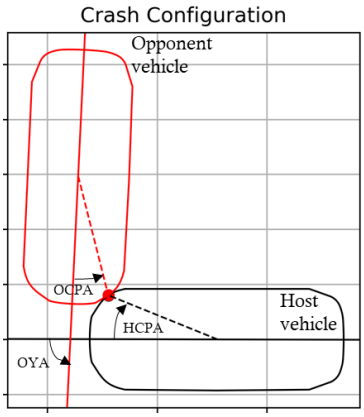

Figure 10: *Crash Configuration obtention through the output files of the simulation.*

The approach for the car-to-moving deformable sled simulations was very similar, although some variations were introduced to solve the lack of specific nodes linked to the accelerometers of the actual vehicles. These accelerometers were used to obtain the different velocities of the car. As these were not the same for the moving deformable sled, the approach was developed by detecting the velocity of the nodes and extracting the value for the whole set. This was performed after the clustering method which helped differentiate between the car and the moving deformable sled or moving barrier.

Furthermore, a final step was introduced to extract the crash pulse of the simulations (accelerations of 3-axis translation and rotation). The methodology was similar for both car-to-car simulations and car-to-moving deformable sled simulations. As for the car-to-car simulations, the time series containing the crash pulses were obtained from the simulation, extracting the 3-axis translations and rotations in the shape of velocities. As for the car-to-moving deformable sled simulations, in this case, instead of extracting the accelerations of both vehicles, only the accelerations for the host vehicle were obtained.

The final approach to gathering these crash pulses datasets was oriented to the ML method (For more information, see *Appendix II*). First of all, these time series were converted to accelerations, as they are easier to understand and have a better feeling of what is happening during the crash. As the complexity of the model would increase drastically by introducing the different time series of the crash pulses, a data-reduction method was developed. To simplify these datasets, a discretization of the time series was applied. Therefore, the time

series to analyze were cut at time 80 ms to have the same format and number of points of every output crash pulse. The resulting time series were split into several intervals. From each interval, the mean was computed and extracted as the discrete value, reducing each time series to a dataset of 16 mean values. This number of points consisted of one value per 5 ms, which described the motion and the trend of the mentioned crash pulse. An example of the different mean values can be seen below:

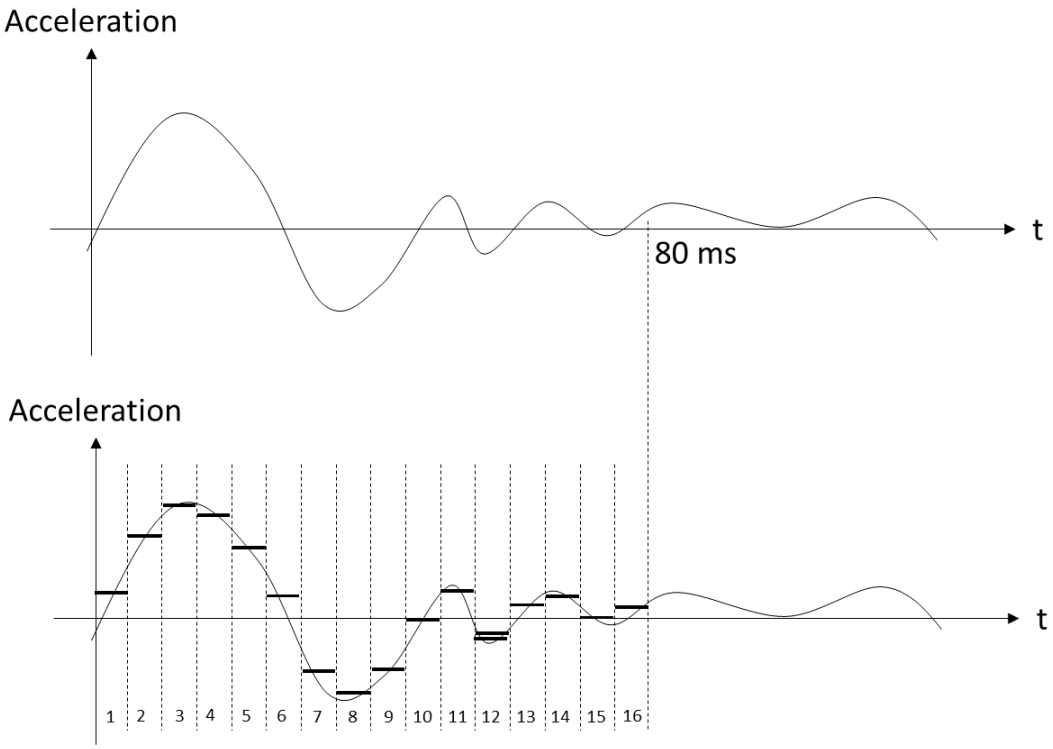

Figure 11: *Time series transformation.*

4.2 *MACHINE LEARNING METHOD*

When the data was gathered, a pre-processing phase to transform the input data into something that the ML algorithm could understand was needed.

First of all, the angles were converted to coordinates of modulus 1, to help the algorithm understand concepts such as 180 degrees and -180 degrees are, in fact, the same.

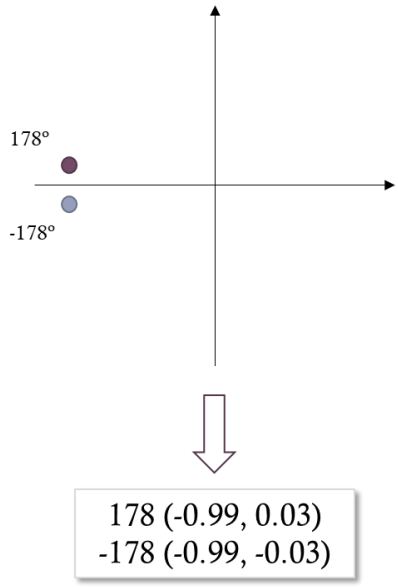

Figure 12: *Angle decomposition example.*

The second step was to normalize the input data when beneficial for the algorithm using the absolute maximum values of each interval as the scalar values. A dataset of almost 500 different simulations was created, with the following format:

$HCPA_X$	$HCPA_Y$	$OCPA_X$	$HCPA_Y$	OYA_X	OYA_Y	HS	OS	MASS H	MASS O
-0.88	-0.39	0.99	0.53	0.92	-0.02	-0.57	-0.54	-1.06	-0.69
0.99	0.53	-0.88	-0.39	0.92	-0.02	-0.54	-0.57	1.43	-0.69
0.18	1.93	-1.76	1.51	-0.72	1.53	-0.57	0.68	-0.85	-0.69

Table 1: *Sample of a normalized input data frame for the ML algorithm.*

4.2.1 TRAINING-TEST DATA SPLIT

The next step was to select the validation data that would help evaluate the algorithm and that would be important to take into account the different kinds of simulations gathered. For this purpose, a graph including the 5 main variables that describe the crash configuration was plotted. In this plot, the direction of the arrows represents the OYA angle, and the two axes represent the HCPA and the OCPA. With the color, the opponent speed is represented, and the size of the arrows represents the magnitude of the Host Speed. An example can be seen below:

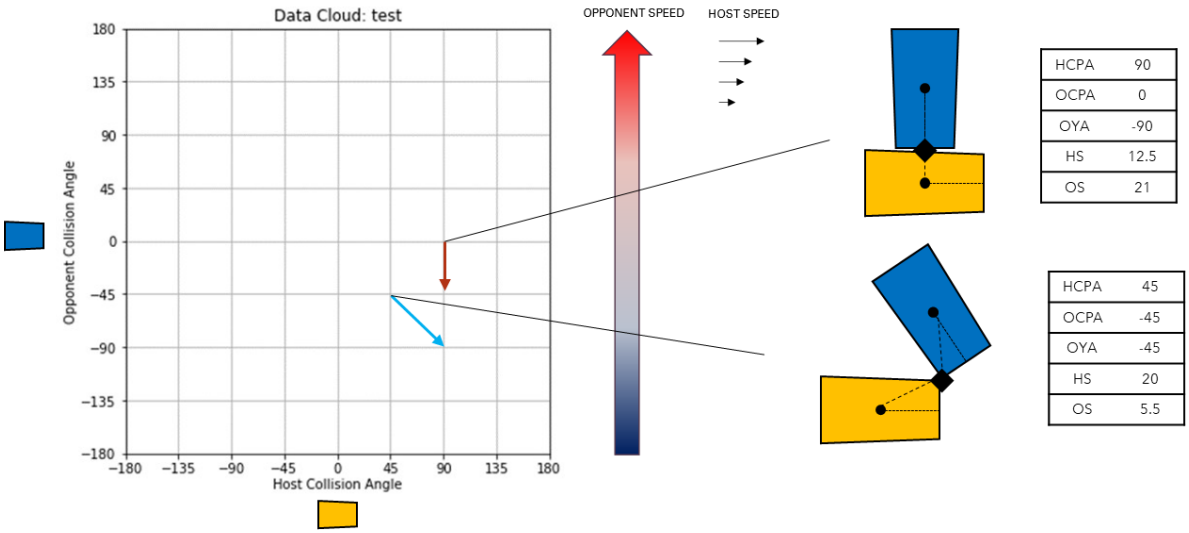

Figure 13: *Example of a Crash Configuration map with some clarification examples*

However, due to confidentiality issues, the arrows are not shown:

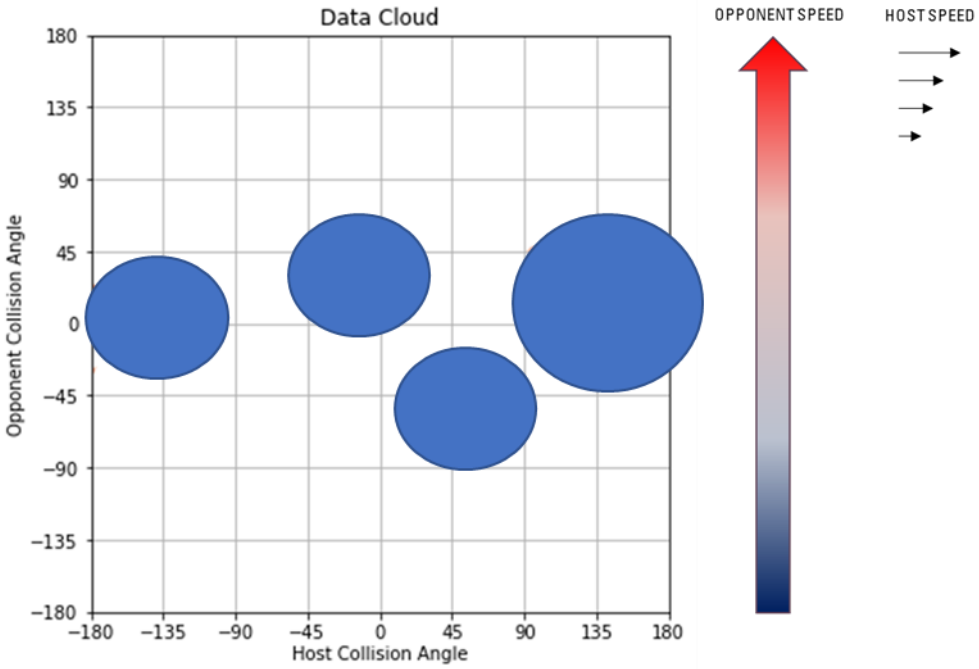

Figure 14: *Data cloud of the different simulations gathered (HCPA vs OCPA with OYA, HS, and OS included).*

From this data cloud, it is necessary to extract those attributed as lateral crashes. To do so, the HCPA (Host Collision Point Angle) will be the corresponding variable indicating which

crashes correspond to a lateral configuration or not. As it has been explained before, the HCPA corresponds to the angle between the direction of the car and the FPOC (First Point of Contact). Regarding normalized dimensions of the car, it can be seen that the angles corresponding to a lateral crash will correspond with the intervals of (45º, 135º) and (-135º, -45º) as it can be seen below:

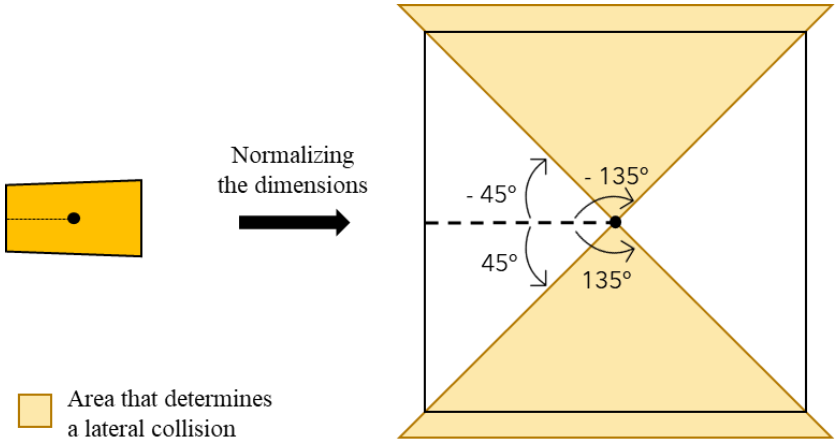

Figure 15: *Criterion to determine the lateral collisions.*

By applying this selection, the resulting data cloud with the lateral collisions is reduced to an amount of 100 simulations. The resulting plot is the following:

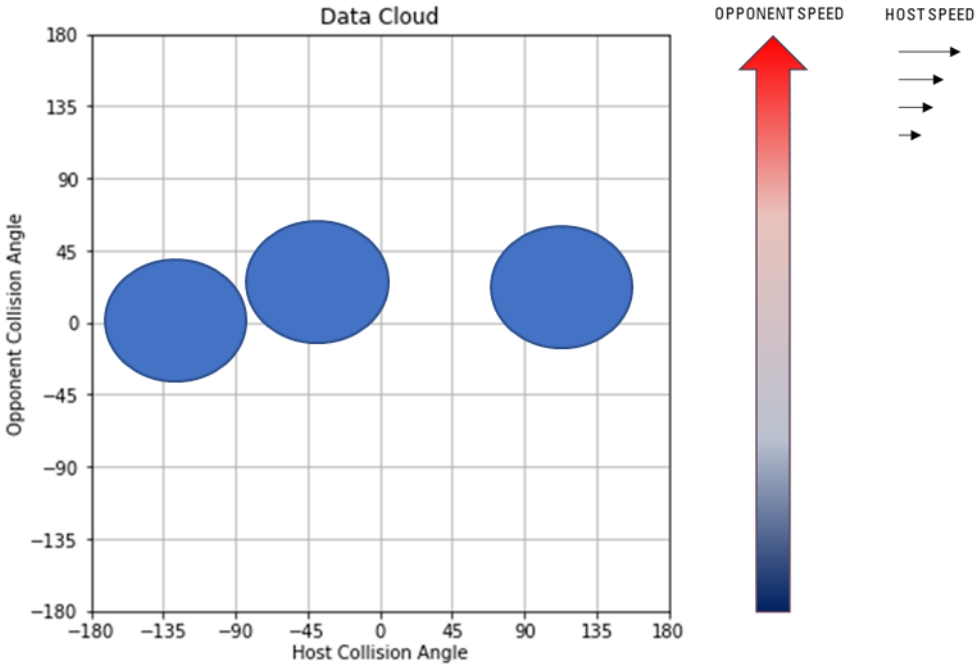

Figure 16: *Data cloud of the lateral collision simulations gathered (HCPA vs OCPA with OYA, HS, and OS included).*

From this data cloud, the training and test datasets were divided in a proportion of 80% and 20% respectively (For more information, see *Appendix II*). The segmentation was performed by randomly splitting the data and analyzing the split applied. The main objective was to obtain a well-distributed test dataset that could be representative enough of the simulations gathered, aiming to validate the different configurations extracted. By saving the training and test datasets, the different models used can be easily compared and evaluated.

4.2.2 ALGORITHM DEVELOPMENT

When choosing the ML approach, multiple algorithms were developed. Therefore, the final models consisted of four methods: Multilayer Perceptron (MLP) using Neural Networks (NN), Random Forest Regression, Support Vector Machine Regression, and Gradient Boost Regression (For more information, see *Appendix II*).

It is important to note that the model approach was developed in terms of the type of acceleration. One model was trained with the entire data set of translational accelerations and another model with rotations. This division was chosen to avoid having misleading data in the models. Not only that but the three axes were also introduced in each model, as these accelerations may also be connected. Therefore, the final configuration of the ML approach consisted of a model that predicted translational accelerations in the x-, y- and z-axis and another model for rotational accelerations also in the x-, y- and z-axis.

4.2.2.1 Neural Networks with MLP

This model allows to introduce multiple input and outputs and create different connections between the variables. One of the preprocessing requirements was to normalize the data before introducing it to the model. To normalize the input data, all the values were scaled using the absolute maximum values of each variable as the scalar values.

This model was introduced with two hidden layers, MSE (Mean Square Error) as the loss function and RMSE (Root Mean Square Error) and R^2 as additional metrics to evaluate the predictions. More details about the configuration of the layers and the hyperparameters can be seen in the table below:

Layer (type)	**Output Shape**	**Param #**
input (Input Layer)	[(None, 10)]	0
dense_01 (Dense)	(None, 256)	2816
dense_02 (Dense)	(None, 128)	32896
output (Dense)	(None, 48)	6192

Total params: 41,904
Trainable params: 41,904
Non-trainable params: 0

Table 2: *Model summary of the configuration of the Neural Network.*

This model represents the configuration of the algorithm used to learn and predict the different output data. It is composed of one input layer with the same number of neurons as input variables have the model; two hidden layers where the learning process takes place, and the different weights of the variables are readjusted to reduce the error of the predictions, and the output layer with as many neurons as output variables to return.

As mentioned before, the process will consist of two steps: first, create one model and train it to predict the translational accelerations; and secondly to train another model to predict the rotational accelerations. The configuration of the model remained the same, as the amount of data managed is not excessively complex to require a different configuration (For more information, see *Appendix II*).

4.2.2.2 Random Forest Regression

This model is a supervised learning algorithm that uses the ensemble learning method for regression. As a brief description, ensemble learning is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. Therefore, this approach operates by creating several decision trees during the training phase, predicting as many output values as decision trees have in the model. The final output consists of the mean of the different predicted values of all the decision trees included. These decision trees are developed in parallel, avoiding any kind of influence between them and using different parts of the training dataset for each tree ([4] Bakshi, 2019) (For more information, see *Appendix II*).

For the model used, the choice of the parameters was the following:

RandomForestRegressor(n_estimators=100, max_depth=15)

Where *n_stimators* corresponds to the number of decision trees in the model and *max_depth* corresponds to the number of divisions or depth levels of each decision tree. Once again, the model configuration remains the same for both Translations and Rotations, as it can adapt itself to both scenarios with the configuration given. An example of a random forest regression tree can be seen below:

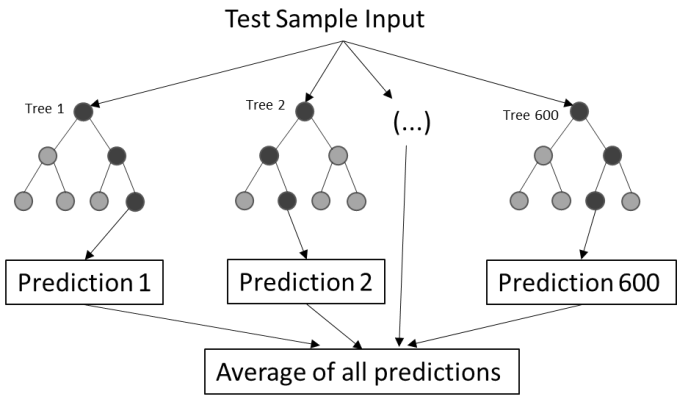

Figure 17: *Random Forest Regression structure with 600 decision trees and 2 depth levels.*

4.2.2.3 Support Vector Machine Regression

Commonly known as Support Vector Regression (SVR), it is a method that allows having a different approach than most linear regression algorithms. In these models, the main objective is to reduce the error understood as the sum of squared errors. However, the SVR gives the flexibility to define how much error is acceptable in the model. This parameter is called *epsilon* (ϵ). Therefore, the model creates a line that fits the data under the limits given to the error. In this case, the model creates a hyperplane to fit the data ([5] Sharp, 2020). This hyperplane can be chosen between different options to obtain the most suitable to fit the data. Another parameter to take into account is *C*, which controls the tolerance for points outside the interval given by ϵ. Therefore, this can help control or mitigate the presence of outliers (For more information, see *Appendix II*).

In addition, in the same way, as with the Neural Network model, it was beneficial to normalize the data before introducing it to the model, increasing the accuracy of the predictions. Therefore, all the input values were scaled using the maximum values of each variable as the scalar values.

The final model used was the following:

SVR(kernel =´poly´, epsilon=1e-5, C=1.0)

The final model chosen was a SVR that used a polynomial curve as the hyperplane or fitting curve. After some trial and error, this kernel was giving a better approximation to the actual data. Once again, the model configuration remains the same for both Translations and Rotations.

4.2.2.4 Gradient Boost Regression

This model uses boosting, which is a method for creating an ensemble. It starts by fitting an initial model and afterward, it builds a second model that focuses on accuracy where the previous model performed poorly. After developing different models, the outcome is expected to be better than any of these previous models alone ([6] Hoare, 2020). For this project, the individual models were set to be Decision Trees (For more information, see *Appendix II*), being the standard approach with this algorithm.

The final model used was the following:

```
GradientBoostingRegressor(n_estimators=100,
                          max_depth=3,
                          min_sample_split=5,
                          learning_rate=0.1, loss='ls')
```

Where *n_estimators* corresponds to the number of Decision Trees used in the model; *max_depth* limits the number of nodes in the tree; *min_sample_split* is the minimum number of samples required to split an internal node; the *learning_rate* shrinks the contribution of each tree; and the *loss* function to be optimized refers to least squares regression (For more information, see *Appendix II*).

4.2.3 MODIFICATIONS

Due to the results obtained (shown in the section of Results), two different approaches were done.

Initially, as explained before, the training data set was created from the data cloud corresponding to only lateral collisions. Therefore, the model was both trained and tested with lateral collisions.

Secondly, after analyzing the results, a second training set was created. This set was chosen from the original data cloud before differentiating between lateral and non-side collisions. This means that the second training data set had lateral and non-side collisions including

frontal, rear, and oblique collisions. Therefore, the model was trained with any type of collision but tested only with lateral collisions.

These two approaches aimed to verify that in the first approach, the models were not overfitting the data, as the amount of data introduced was limited, and maybe, biased. To have a better understanding and check how reliable these predictions were, the second approach was applied.

In the next figure, a representation of the new training vs the test data is shown. In blue circles is the gathered data, and in red, the test data:

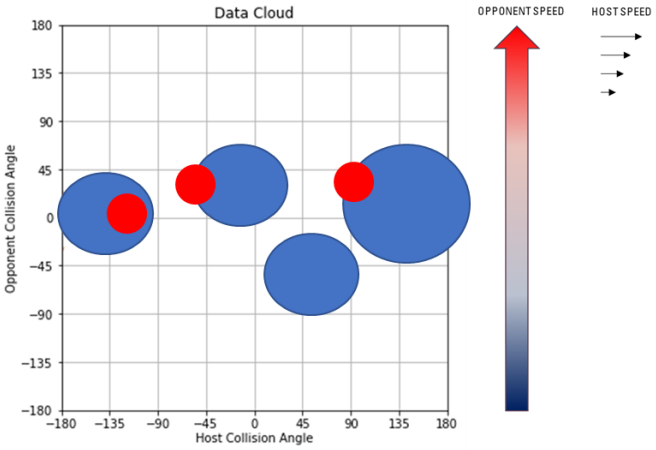

Figure 18: *Test data cloud in red, on top of the data cloud of the different Simulations (blue), gathered (HCPA vs OCPA with OYA, HS, and OS included).*

Chapter 5. RESULTS

As it has been mentioned before, the final metrics used to evaluate the predictions and the accuracy of the models were the RMSE (Root Mean Square Error) and the R^2. Before analyzing and comparing the results, a brief explanation of these two parameters will follow:

- RMSE (Root Mean Squared Error):

$$RMSE = \sqrt{\sum_{i=0}^{n-1} \frac{(y_i - \hat{y}_i)^2}{n}}$$

Where n represents the number of samples; $\hat{y}_i$ the predicted value for the i-th sample; and y_i the corresponding true value.

This metric provides the amount of error that the module obtains with the prediction, avoiding some biased results present when using other metrics. This is an absolute metric and can be directly compared with other models using the same dataset ([7] Scikit-Learn,).

- R^2 or Coefficient of Determination:

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \overline{y})^2}$$

Where SS_{RES} represents the Squared Sum of Residuals regarding the predictions; SS_{TOT} represents the Total Sum of Squared Errors; $\hat{y}_i$ the predicted value for the i-th sample; y_i the corresponding true value; and $\overline{y}$ the mean of the true values.

In other words, R^2 explains how well the model explains the variance of the dataset y. The SS_{TOT} represents the variance of the data and the SS_{RES} represents the variance of the predicted dataset. When computing this metric, it provides an indication of goodness of fit and therefore a measure of how well the validation or unseen samples are likely to be predicted by the model, throughout the proportion of explained variance. The maximum

value for this metric is 1.0, and the metric can be negative, as the model can be arbitrarily worse ([7] Scikit-Learn,).

5.1 *APPROACH WITH LATERAL SIMULATIONS AS THE TRAINING SET*

The first approach with the training set containing only lateral collisions.

These graphs represent the different values of R^2 and RMSE during the different intervals or points predicted for the 16 point-crash pulses. With these graphs, a better understanding and general overview of the results can be seen. For other graphs, see *Appendix I*.

5.1.1 TRANSLATIONS

5.1.1.1 Neural Networks with MLP

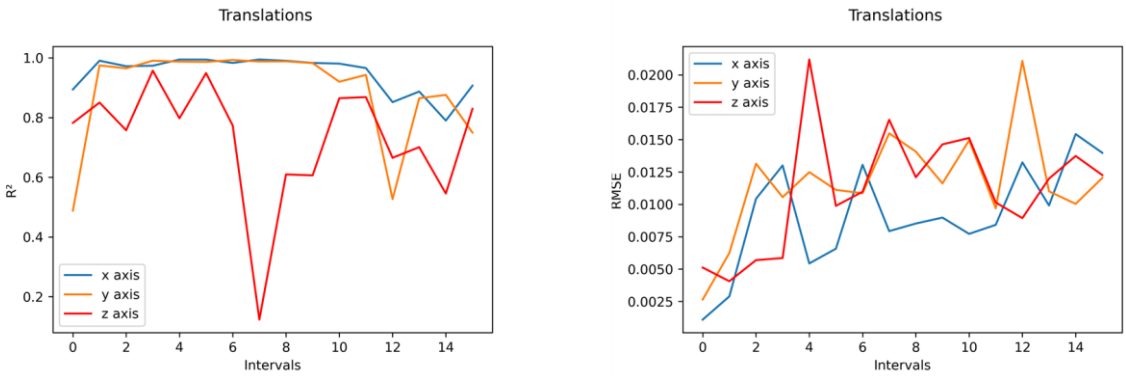

Figure 19: *R^2 and RMSE for Translations using NN.*

5.1.1.2 Random Forest Regression

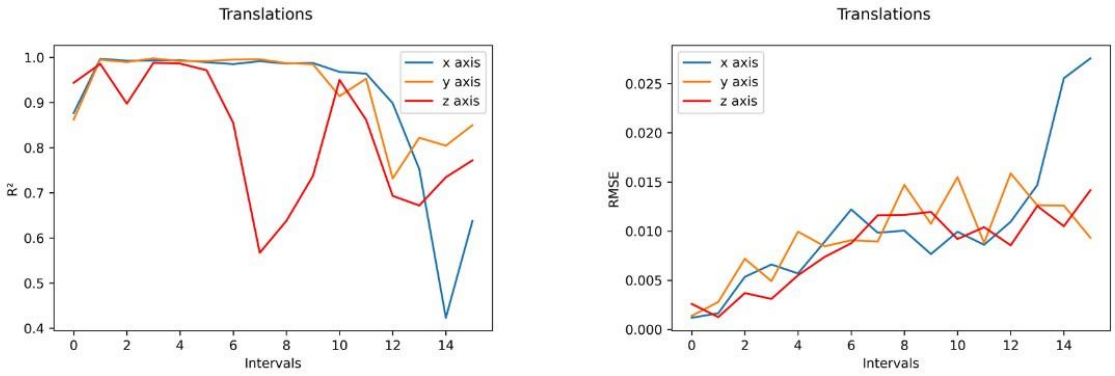

Figure 20: *R^2 and RMSE for Translations using Random Forest.*

5.1.1.3 Support Vector Machine Regression

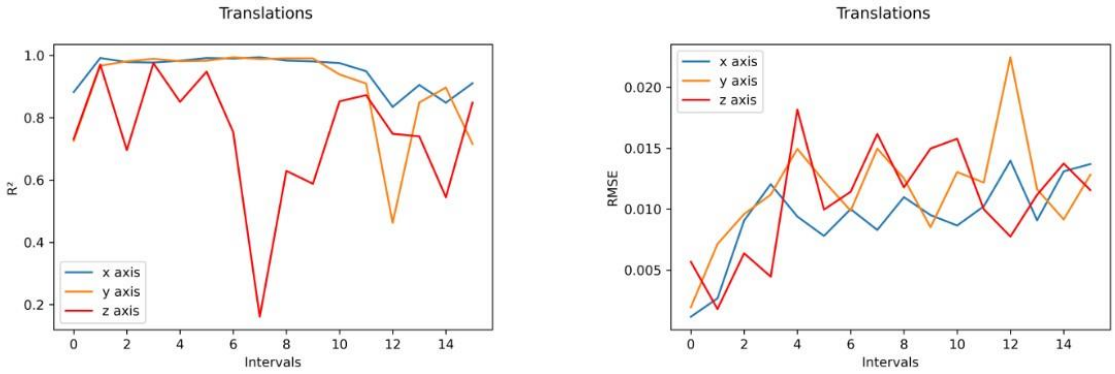

Figure 21: *R^2 and RMSE for Translations using SVR.*

5.1.1.4 Gradient Boost Regression

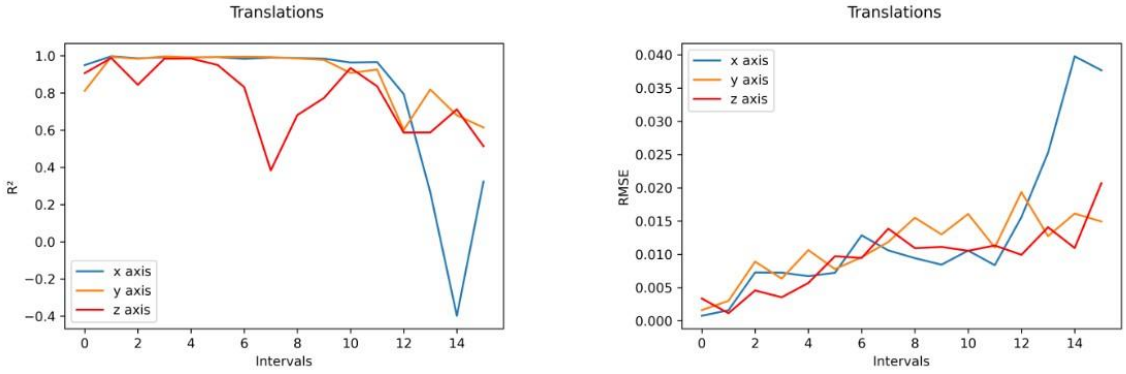

Figure 22: *R^2 and RMSE for Translations using Gradient Boost Regression.*

The first comparison was performed between the results of the models predicting the translational crash pulses. To choose what model suits best the data and makes the best prediction possible, different aspects were taken into account.

- Initially, the R^2 can give a first idea of how good the fit is. In general terms, it could be said that all of the models had a relatively good approximation of the true values. However, the Random Forest model was the one that had the highest values throughout all of the intervals in the different axes.
- Following the previous trend, it is the Random Forest model the one that has the lower magnitude for the RMSE.

- In general terms, it was seen that the z-axis translation was the regression whose R2 was the lowest. One possible reason for this to happen is the difference in magnitude in comparison to the x- and y-axis. As in translation, these two axes represent the majority of the motion, it is with the z-axis where these accelerations perceived have smaller magnitudes and with more noise.

5.1.2 ROTATIONS

5.1.2.1 Neural Networks with MLP

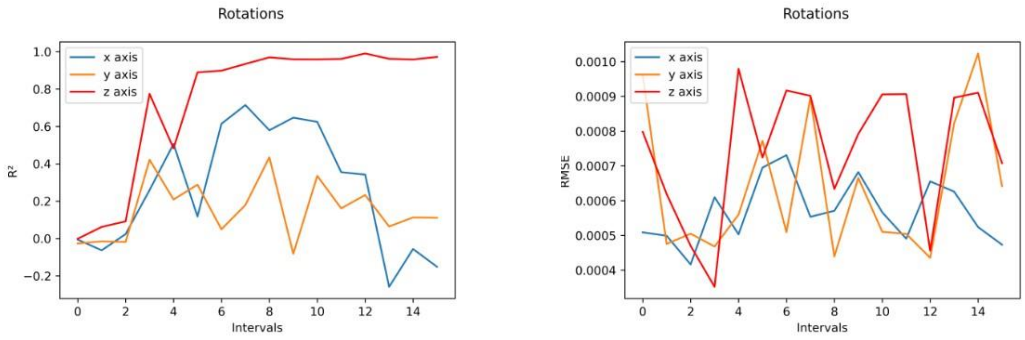

Figure 23: *R^2 and RMSE for Rotations using NN.*

5.1.2.2 Random Forest Regression

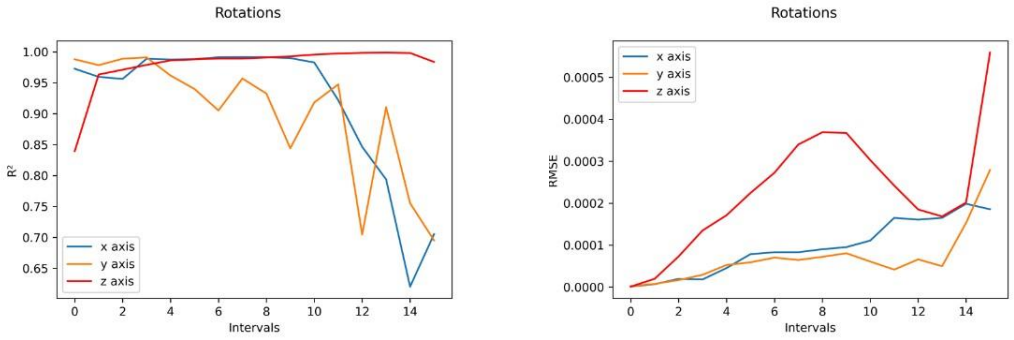

Figure 24: *R^2 and RMSE for Rotations using Random Forest Regression.*

5.1.2.3 Support Vector Machine Regression

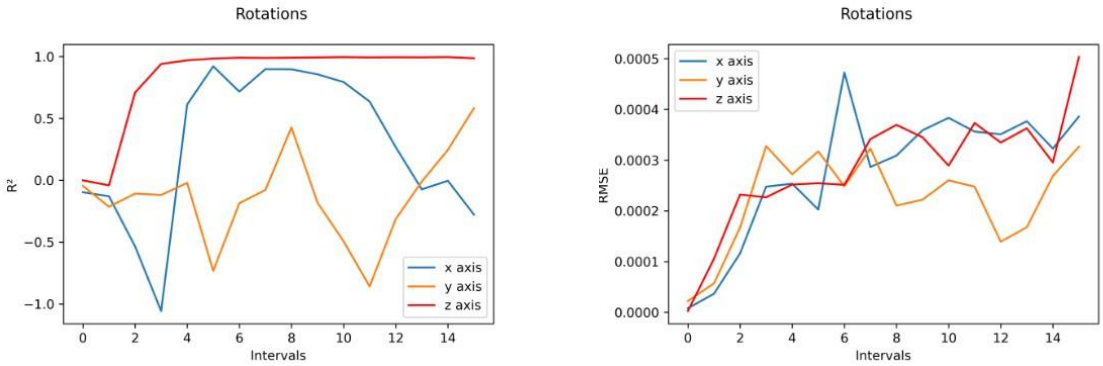

Figure 25: *R^2 and RMSE for Rotations using SVR.*

5.1.2.4 Gradient Boost Regression

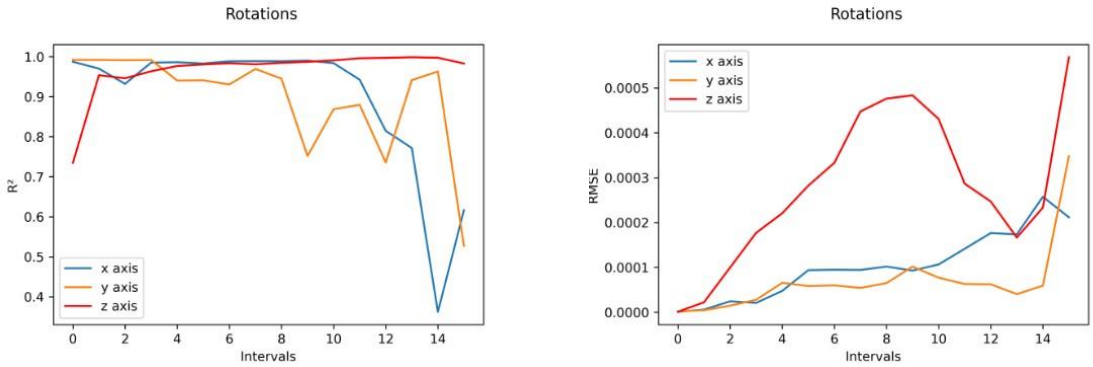

Figure 26: *R^2 and RMSE for Rotations using Gradient Boost Regression.*

In this case, the same procedure to evaluate the Rotational models was performed.

- In general terms, the Random Forest model was the one that performs best in terms of R^2 as the values were obtained in a narrower range throughout the intervals and axes.
- Following the previous trend, it is the Random Forest model the one that has the lower magnitude for the RMSE, very similar to the Gradient Boost. However, it was also a close result in this case the SVR model, whose RMSE had a more continuous trend and with a low magnitude as well.

- In general terms, it was seen that the z-axis translation was the regression whose R^2 was the largest. A possible reason was that due to the crash configuration (side collision). It might be expected to have larger rotations in the z-axis rather than in the other axes. Not only that but also the other two rotations of the x- and y-axis will have values closer to zero and more noise will be present. Therefore, the algorithm will be capable of making more accurate predictions for the z-axis.

5.2 *APPROACH WITH TRAINING SET FROM ALL SIMULATIONS*

The second approach with the training set containing any type of collisions. As mentioned before, this training set was chosen from the general data cloud containing all types of crash configurations.

5.2.1 TRANSLATIONS

5.2.1.1 Neural Networks with MLP

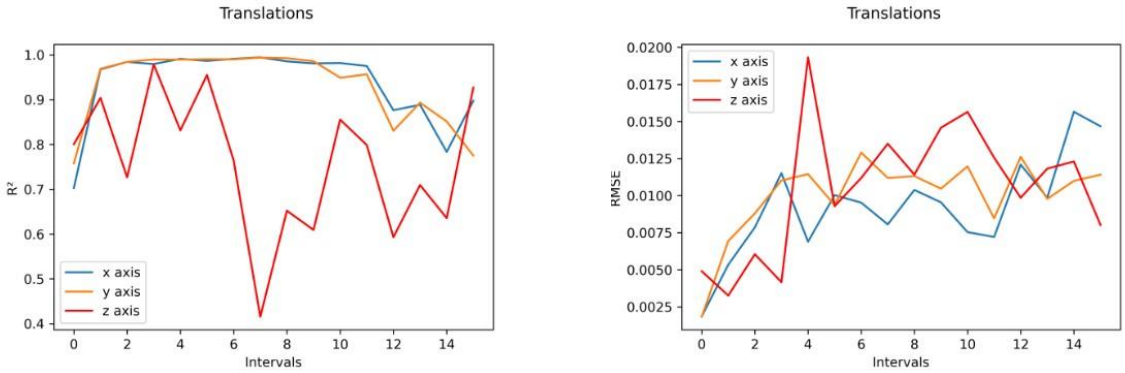

Figure 27: *R^2 and RMSE for Translations using NN.*

5.2.1.2 Random Forest Regression

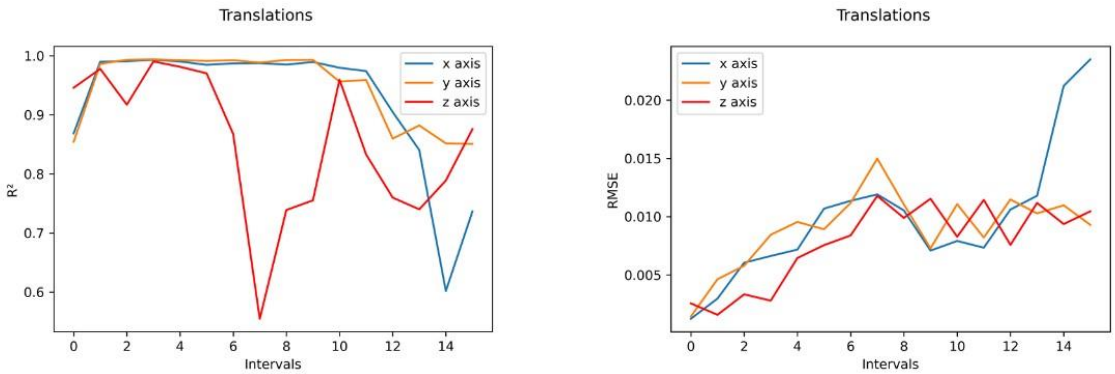

Figure 28: *R^2 and RMSE for Translations using Random Forest Regression.*

5.2.1.3 Support Vector Machine Regression

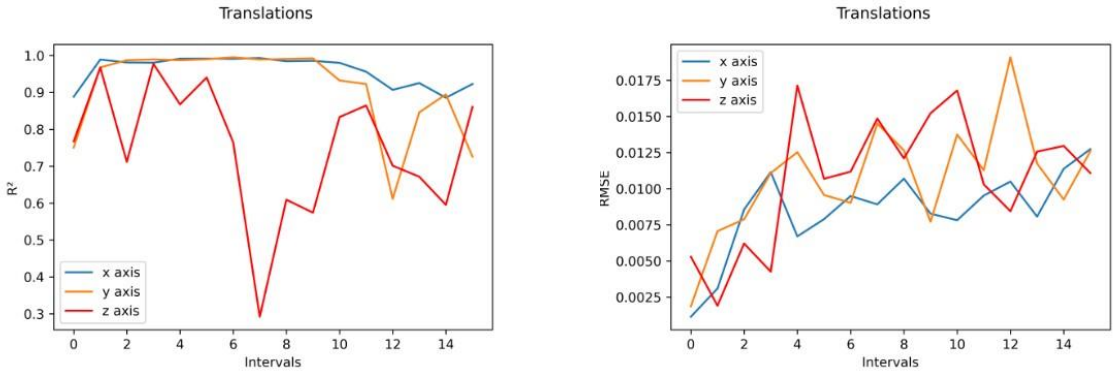

Figure 29: *R^2 and RMSE for Translations using SVR.*

5.2.1.4 Gradient Boost Regression

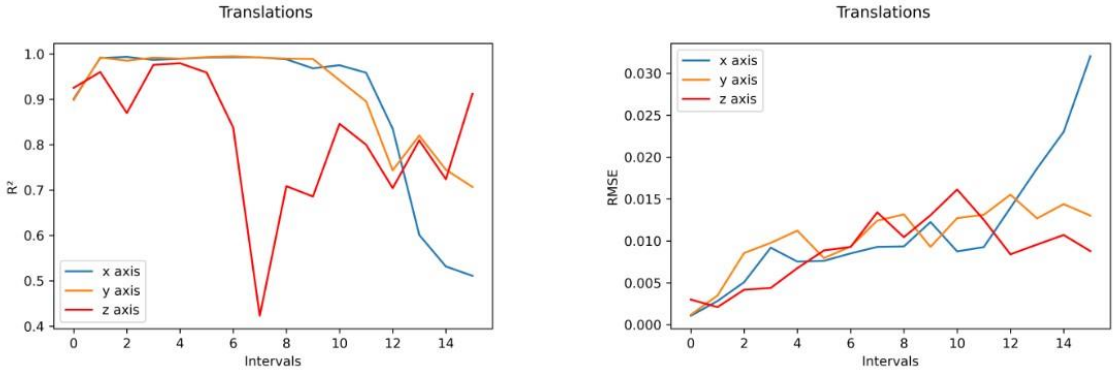

Figure 30: *R^2 and RMSE for Translations using Gradient Boost Regression.*

In these new models, the curves for the R^2 improved in terms of the range reached. It was seen that this metric was closer to 1 at the first intervals, and when incrementing, the accuracy of the coefficient was reduced. In terms of the RMSE, in all of the models, a reduction of the error can be seen.

5.2.2 ROTATIONS

5.2.2.1 Neural Networks with MLP

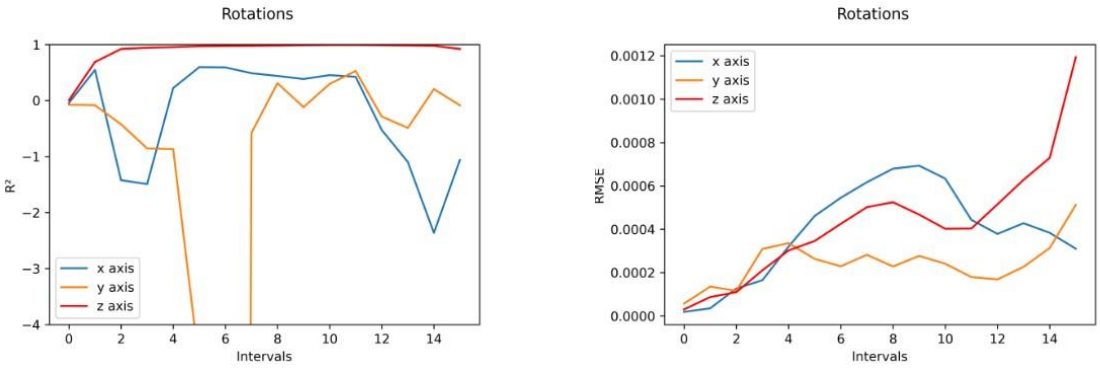

Figure 31: *R^2 and RMSE for Rotations using NN.*

5.2.2.2 Random Forest Regression

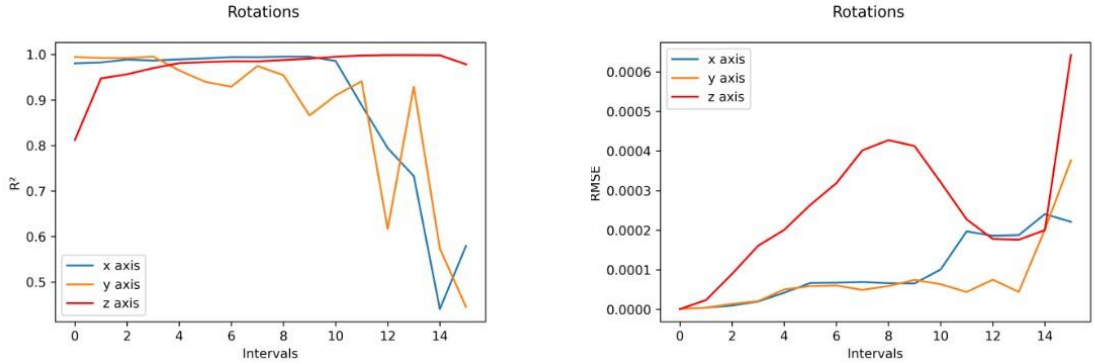

Figure 32: *R^2 and RMSE for Rotations using Random Forest Regression.*

5.2.2.3 Support Vector Machine Regression

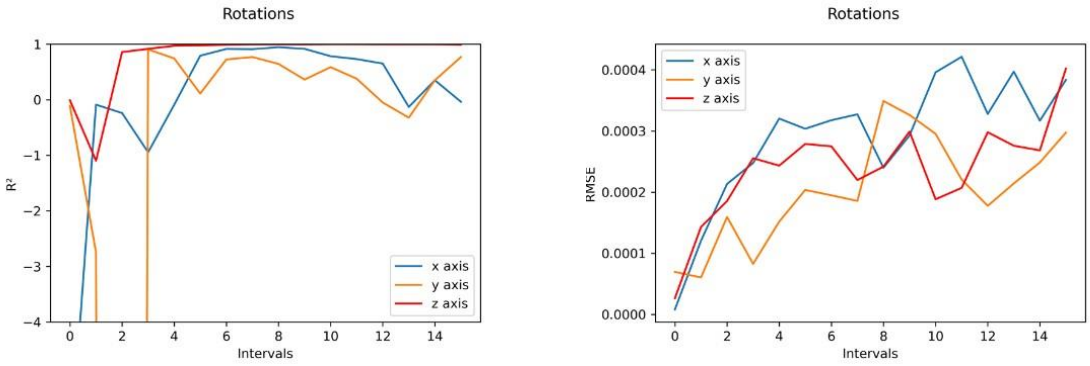

Figure 33: *R^2 and RMSE for Rotations using SVR.*

5.2.2.4 Gradient Boost Regression

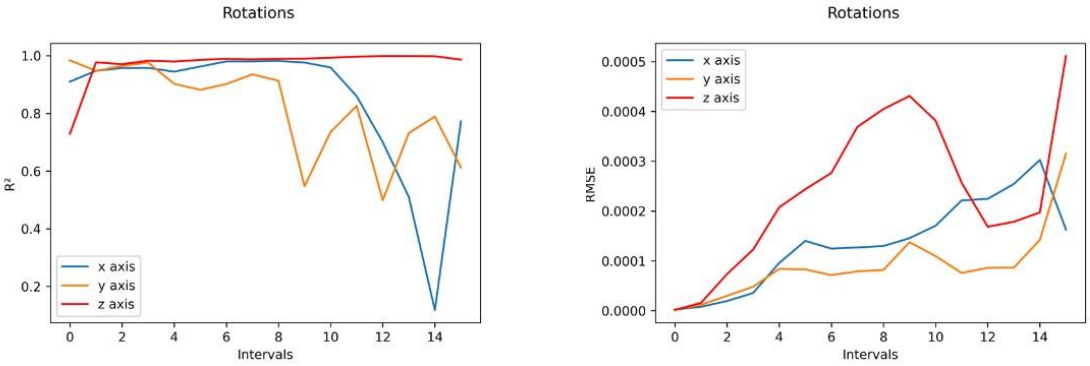

Figure 34: *R^2 and RMSE for Rotations using Gradient Boost Regression.*

In this section, some models had trouble when predicting the rotations, obtaining values for the R^2 coefficient that was below 0 at some intervals. As for the range where the RMSE was obtained, it had a similar range of values between the same models when changing the training dataset.

Chapter 6. CONCLUSIONS

After the analysis of results and comparison between the different models of the first approach, some conclusions were made. First of all, the models had different results and each of these was better at some parts than others. However, it was the Random Forest Regression model that seemed to be predicting closer values to the true values throughout all the intervals. One possible reason for these apparent good results could be that the data set used for the training was very similar to the one used in testing. Because of this, as explained before, a second approach was performed, using the general training dataset gathered (including all types of simulations) to determine if the results could be trusted or if not, what kind of results could be expected under unbiased conditions.

Some new insights appeared after this new approach was performed:

- In terms of the translations, it was seen that the RMSE values were reduced slightly. This fact was surprising as, after introducing new and more varied sets of data for the training process, a larger RMSE could be expected. However, this meant that all the models could adapt to this new training set, despite having more variations to learn from. As for the R^2, in general terms, the values were slightly larger, meaning an improvement from the first approach. However, the general behavior remained very similar, having values close to 1 for the x and y axes and seeing larger variations for the z-axis.
- In terms of the rotations, the results were not as promising as in the translations. In this case, the RMSE increased slightly, despite seeing that the general trend of these curves was not as noisy (noisy understood as curves with a lot of peaks values and not clear trend appreciated). As for the R^2, the models obtained drastically different results between them. While the Random Forest Regression and the Gradient Boost Regression remained to obtain good results, the Neural Network and SVR saw their performance affected. These two last models seemed to have problems obtaining good R^2 values at some intervals despite keeping relatively good results at some

others. This new behavior could be a result of the introduction of these new datasets for the training process which, in rotations, resulted in worse performance in some models than in the first approach. These R^2 values were on some occasions lower than 0, which means that the prediction model is not capable of making better predictions than establishing the predicted value as the mean of the target values.

After all these approaches, the final decision was to choose the Random Forest as the final algorithm to be used, as this model performed good results in both approaches. One of the main concerns of this model is that it easily overfits its predictions, making it complex to train. However, after the second approach, it was seen that despite introducing a different variety of simulations, the model was still capable of making good predictions for these specific lateral crash configurations. Furthermore, it was seen that this model was also capable of having the same performance values when introducing the input data without a previous phase of normalization, which can be very helpful when using this model afterward.

Another decision to make was what approach was the most suitable. The final decision was to keep the second approach, as it helped the model learn from different crash configurations, which could be helpful for future developments. Not only that but also it was chosen because the first approach had some limitations, which will be explained in the next section.

Therefore, the chosen model was capable of making good predictions, having relatively good R^2 and RMSE scores. To have a better feeling of how the model was performing, the following figures of three different simulations were developed:

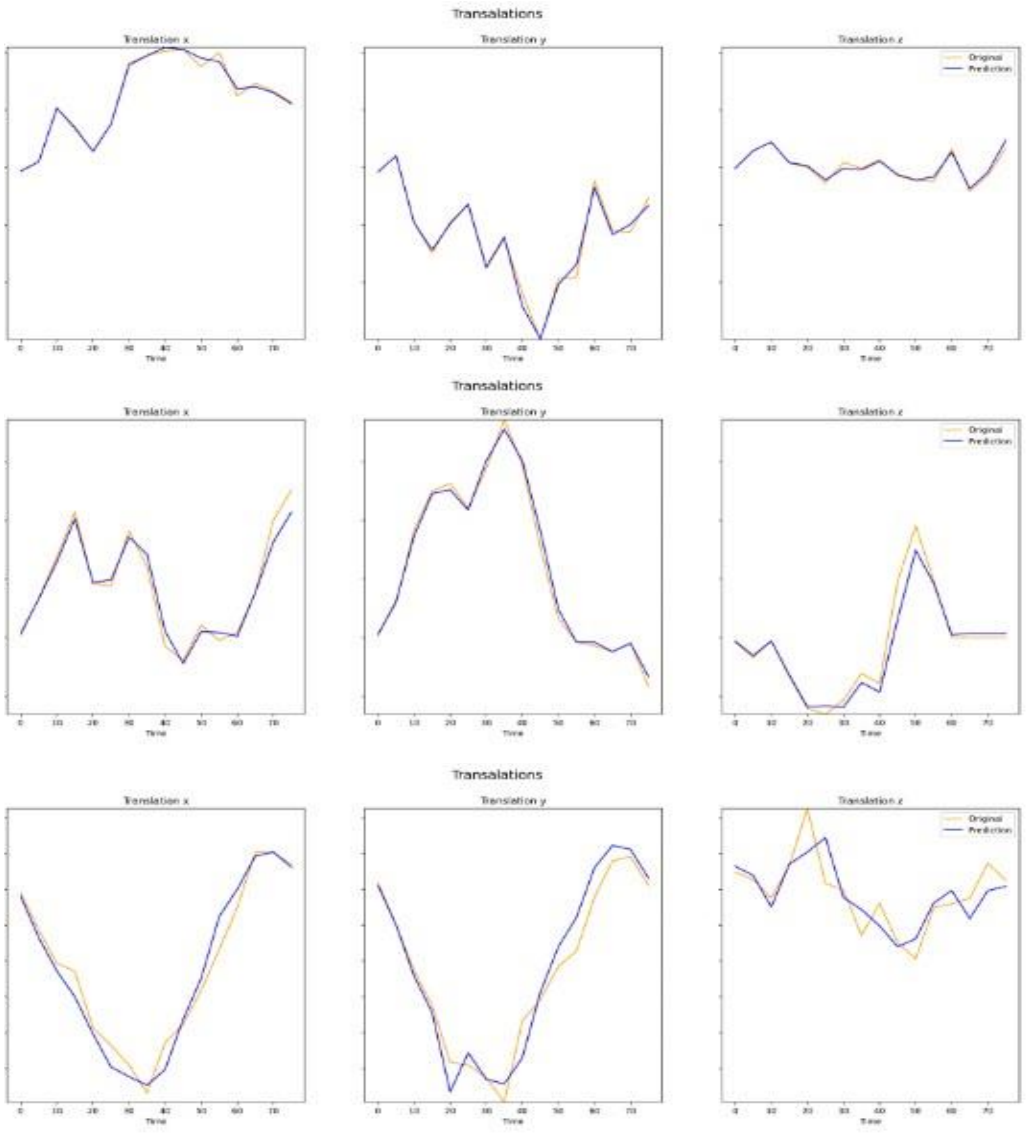

Figure 35: *Three test samples. Predictions vs True Values for Translations using Random Forest Regression.*

As it can be seen in this figure, it was for the translations in the x and y axes where the similarity between the curves was better. However, when predicting the z-axis, a larger difference between the curves can be appreciated. Nevertheless, despite not being 100% perfect predictions, the general trend that the curves described was acceptable.

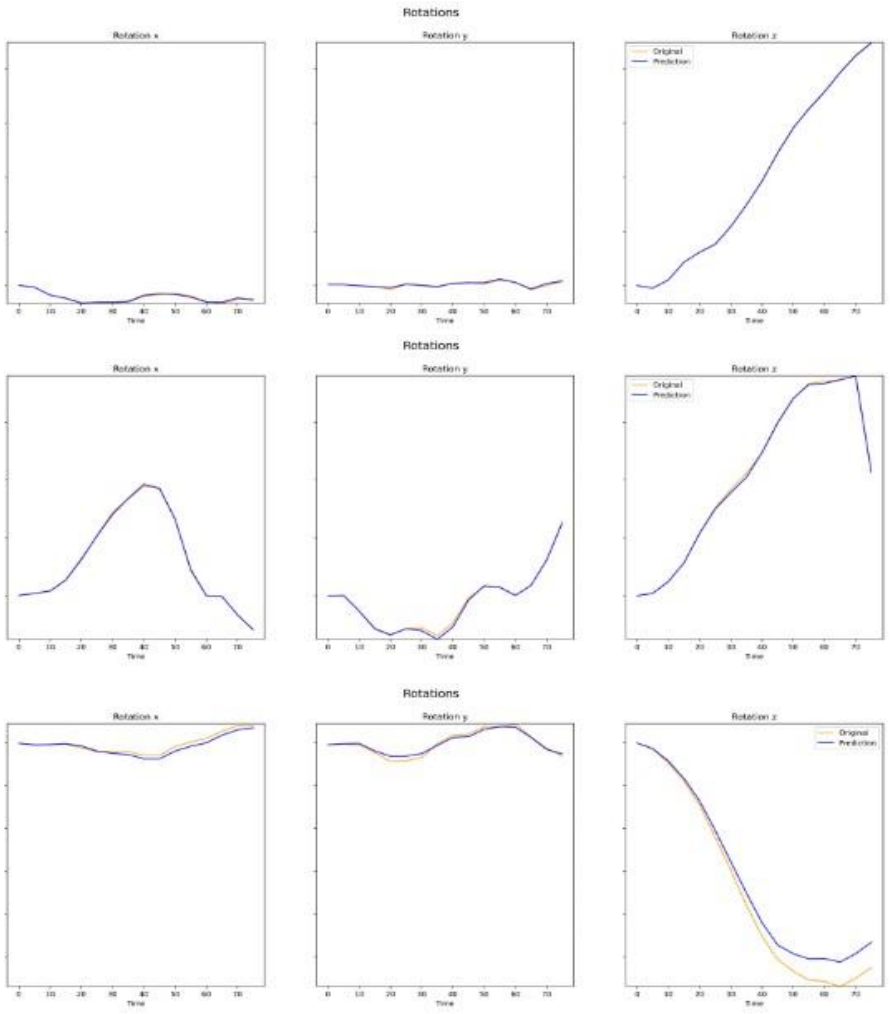

Figure 36: *Three test simples. Predictions vs True Values for Rotations using Random Forest Regression*

In this case, no obvious differences between the predictions of the three axes were observed. The z-axis for the third prediction was perhaps the one that could be having some trouble predicting the last intervals, but it was considered as an acceptable difference.

6.1 *LIMITATIONS*

In terms of the limitations found throughout the project, the main ones will be listed below:

- The amount of data gathered, as seen in some figures, were not diverse enough, making it difficult to train and test the model with certainty that this model will not be overfitting the data. Therefore, for the next developments, not only more data but more varied data would be needed to be gathered.
- The model approach was based on supervised learning. However, nowadays exist several approaches with unsupervised learning that could be suitable for this project. With so, the crash pulses might not need to be cropped and discretize in different intervals, increasing the direct use of this model to be used afterward.
- The input data would need to be more varied when introducing more car models. The model is working with 7 parameters but if this model were to be used with different car brands, car types, mechanical properties such as stiffness, or other parameters, it could result in a more viable and robust algorithm.

Chapter 7. REFERENCES

[1] Wikipedia, S. C., 2009. [En línea] Available at: https://en.wikipedia.org/wiki/Side_collision

[2] Leledakis, A., 2020. “A method for predicting crash configurations using counterfactual simulations and real-world data".

[3] Wågström, L., 2019. “Integrated Safety: Establishing Links for a Comprehensive Virtual Tool Chain”.

[4] Bakshi, C., 2019. *GitConnencted, “Random Forest Regression".* [En línea] Available at: https://levelup.gitconnected.com/random-forest- regression-209c0f354c84

[5] Sharp, T., 2020. *Toward Data Science, "An Introduction to Support Vector Regression (SVR)".* [En línea] Available at: https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2

[6] Hoare, j., 2020. *DisplayR, “Gradient Boosting Explained – The Coolest Kid on The Machine Learning Block".* [En línea] Available at: https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block/

[7] Scikit-Learn, U. G., . *"Metrics and scoring: quantifying the quality of predictions".* [En línea]
Available at: https://scikit-learn.org/stable/modules/model_evaluation.html#

APPENDIX I. EXTRACTED DATA FROM MODELS

I. APPROACH WITH LATERAL SIMULATIONS AS THE TRAINING SET

The first approach with the training set containing only lateral collisions.

a. TRANSLATIONS

i. Neural Networks with MLP

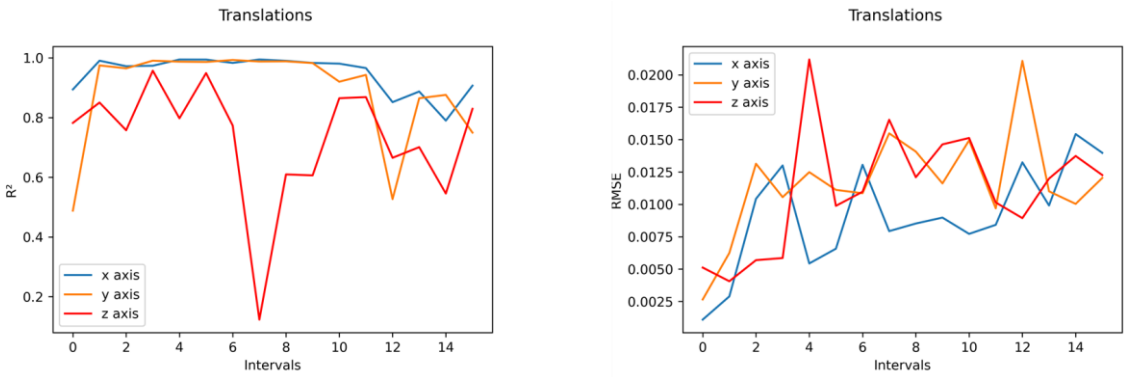

Figure 1: *R^2 and RMSE for Translations using NN.*

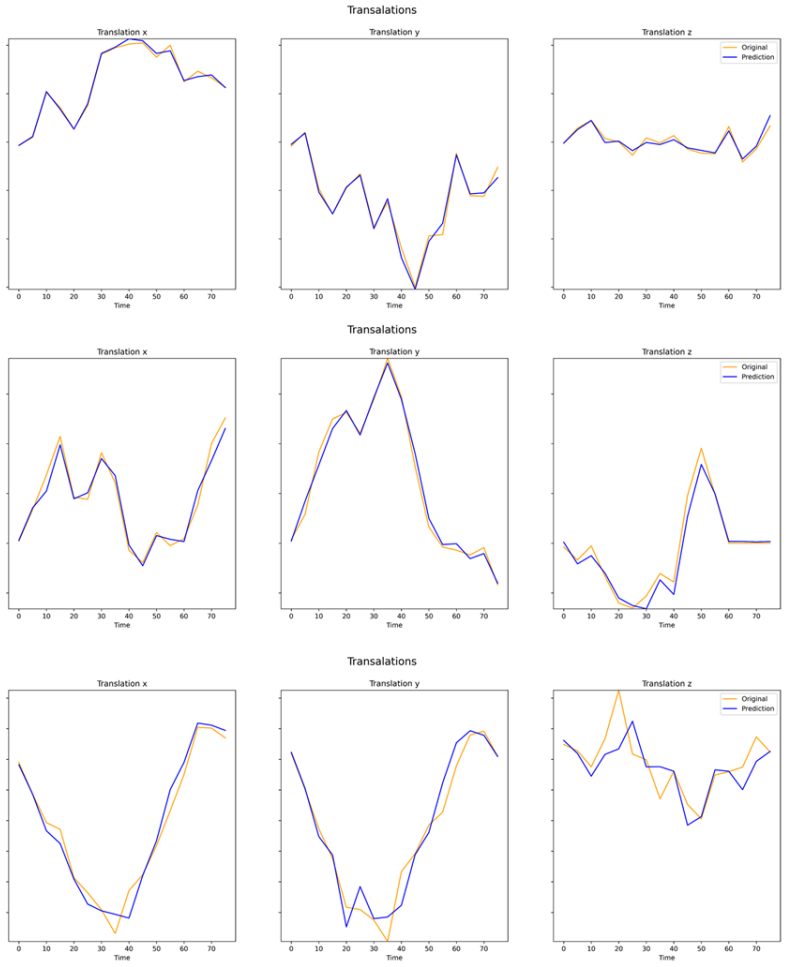

Figure 2: *Three test simples Predictions vs True Values for Translations using NN.*

ii. Random Forest Regression

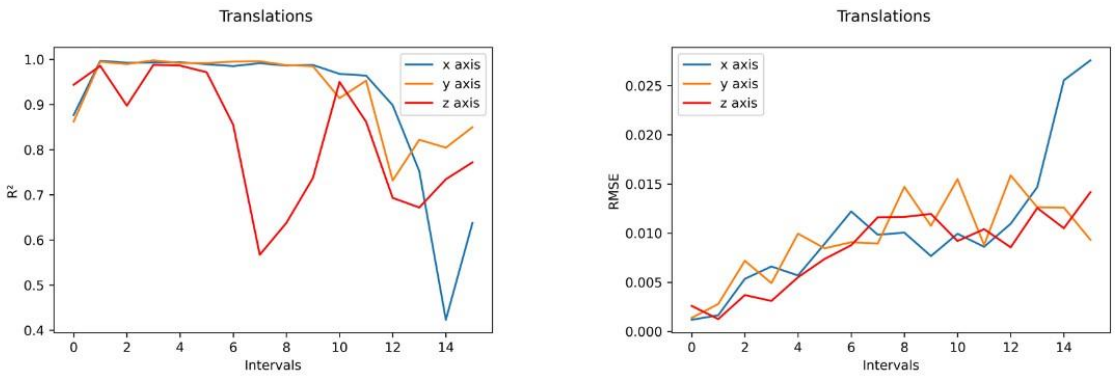

Figure 3: *R^2 and RMSE for Translations using Random Forest.*

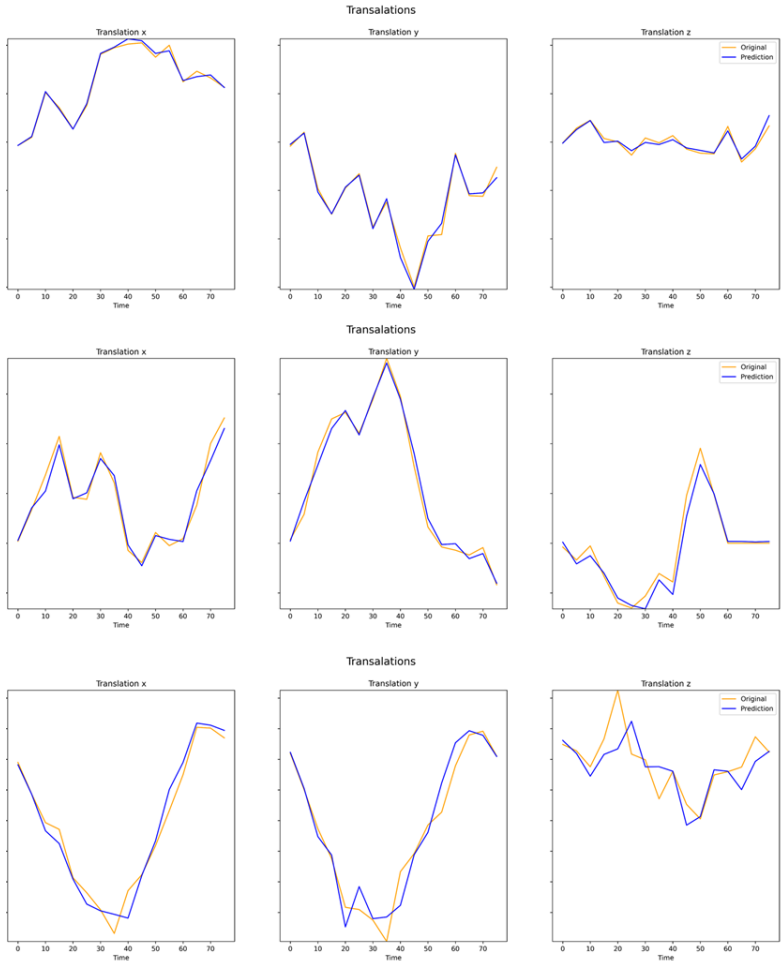

Figure 4: *Three test simples Predictions vs True Values for Translations using NN.*

iii. Support Vector Machine Regression

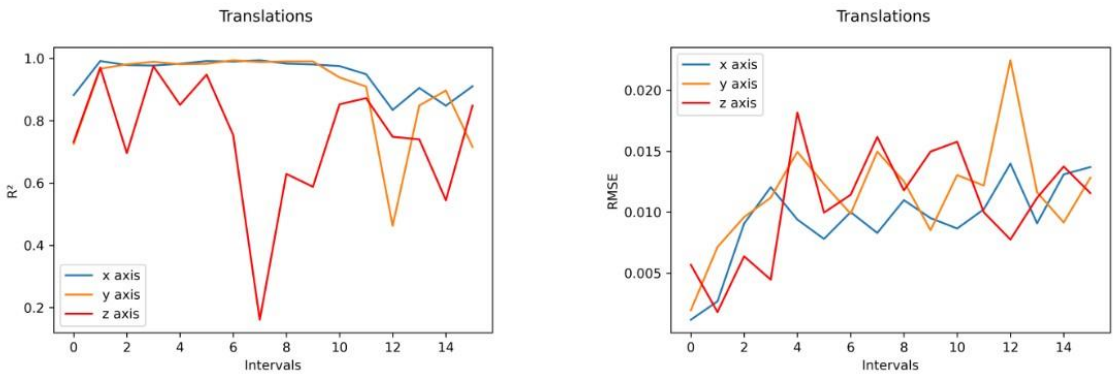

Figure 5: *R^2 and RMSE for Translations using SVR.*

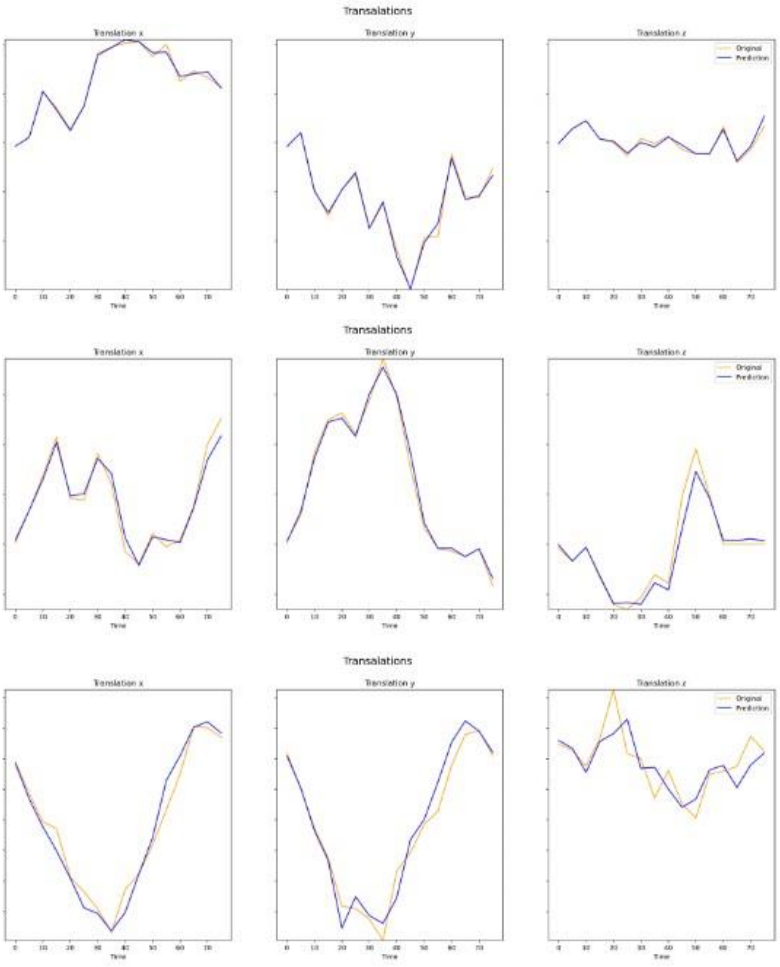

Figure 6: *Three test simples Predictions vs True Values for Translations using NN.*

iv. Gradient Boost Regression

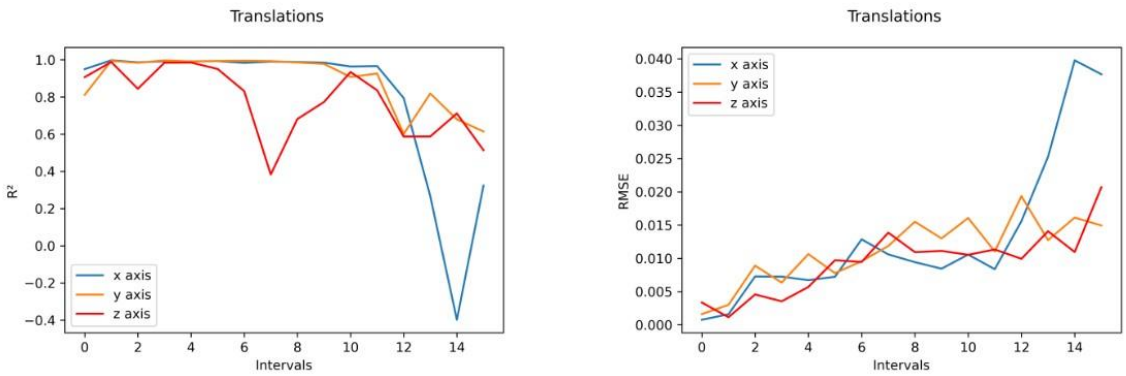

Figure 7: *R^2 and RMSE for Translations using Gradient Boost Regression.*

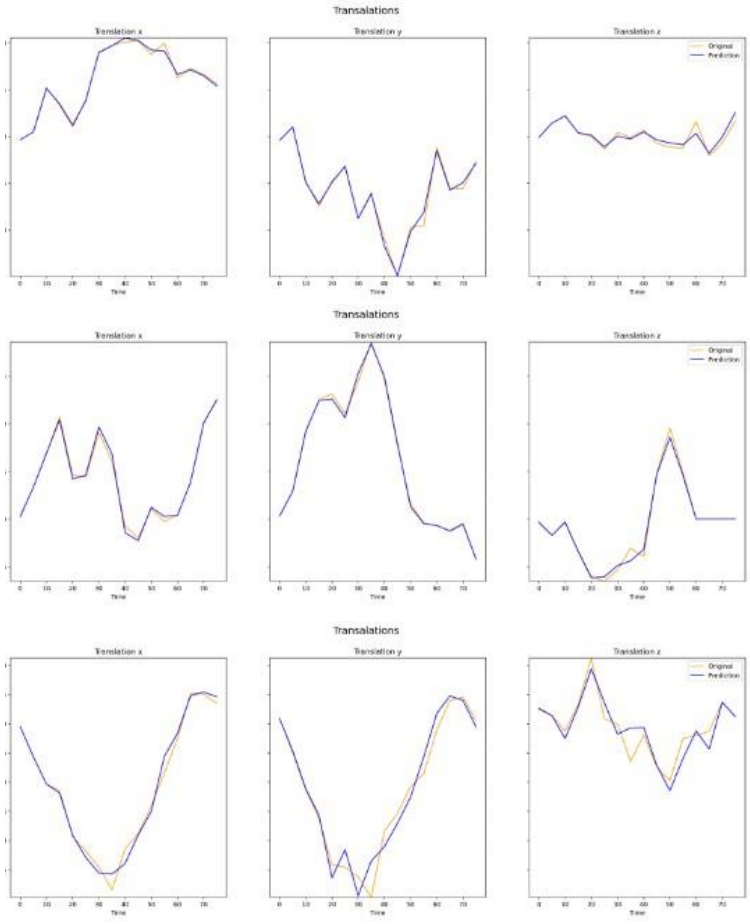

Figure 8: *Three test simples Predictions vs True Values for Translations using NN.*

b. ROTATIONS

i. Neural Networks with MLP

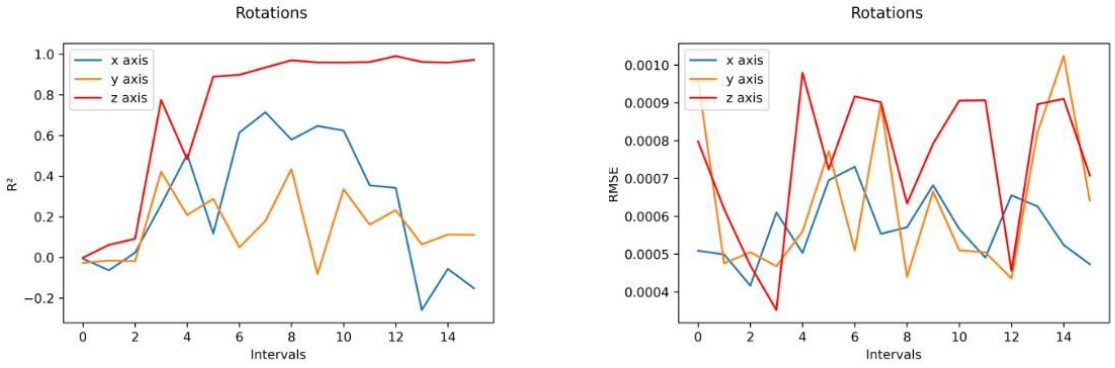

Figure 9: *R^2 and RMSE for Rotations using NN.*

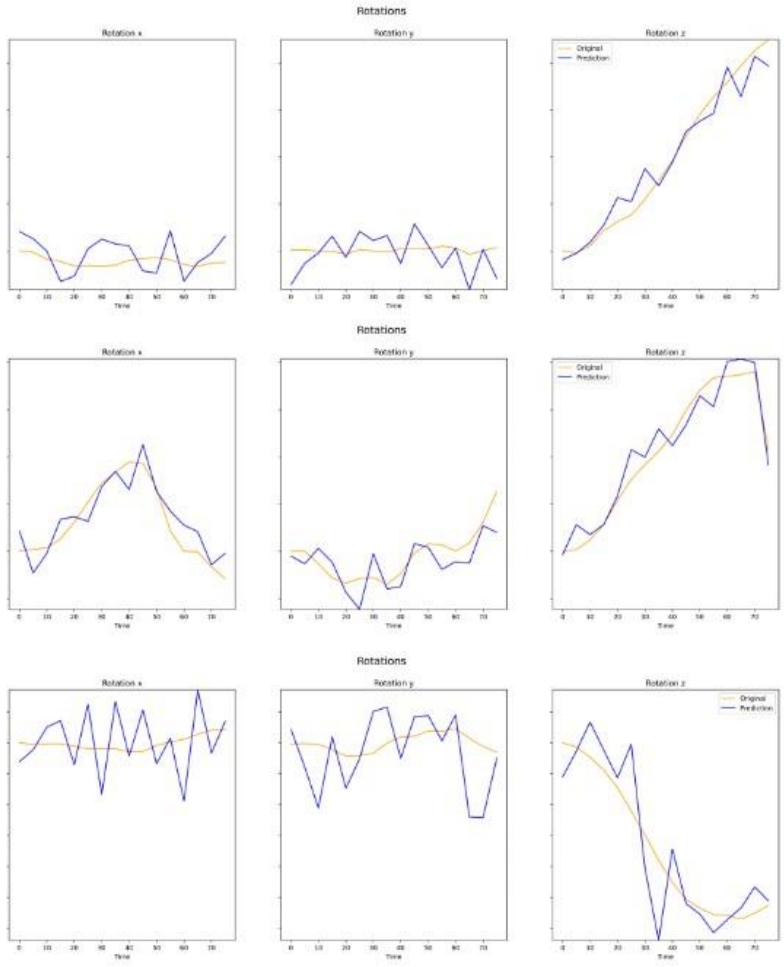

Figure 10: *Three test simples Predictions vs True Values for Rotations using NN.*

ii. Random Forest Regression

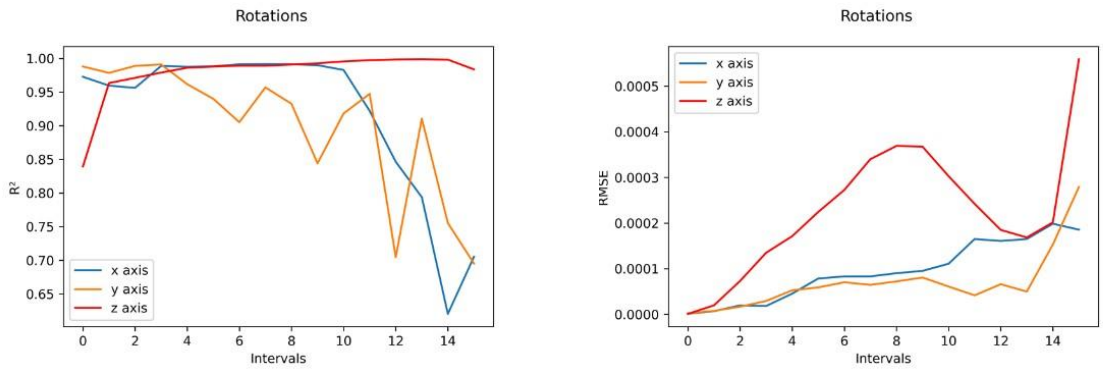

Figure 11: *R^2 and RMSE for Rotations using Random Forest Regression.*

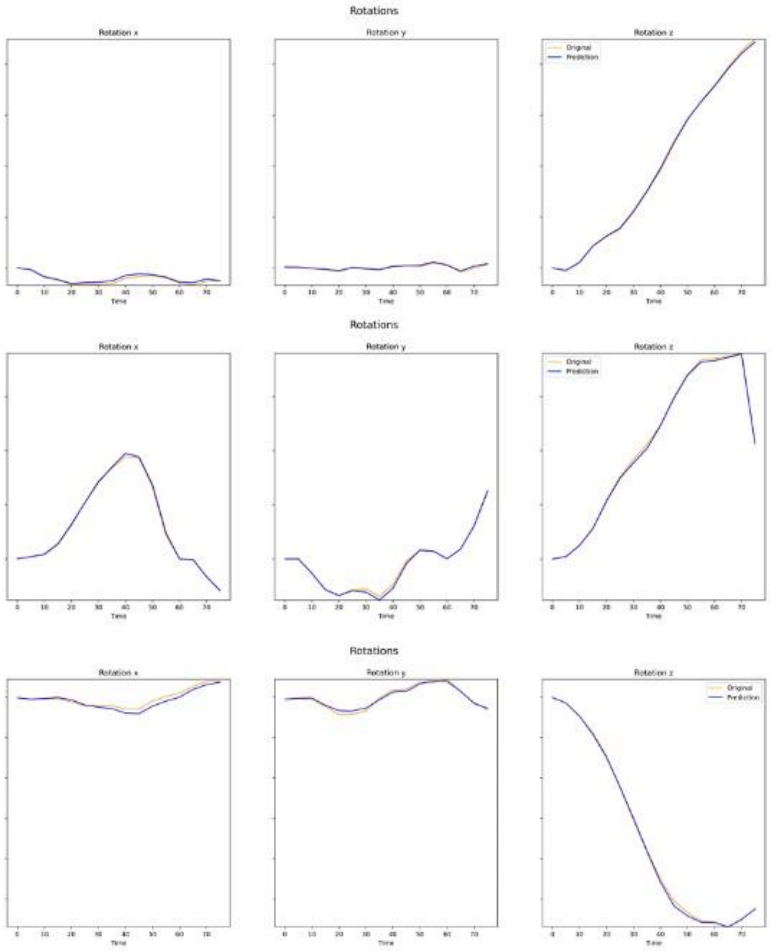

Figure 12: *Three test simples Predictions vs True Values for Rotations using NN.*

iii. Support Vector Machine Regression

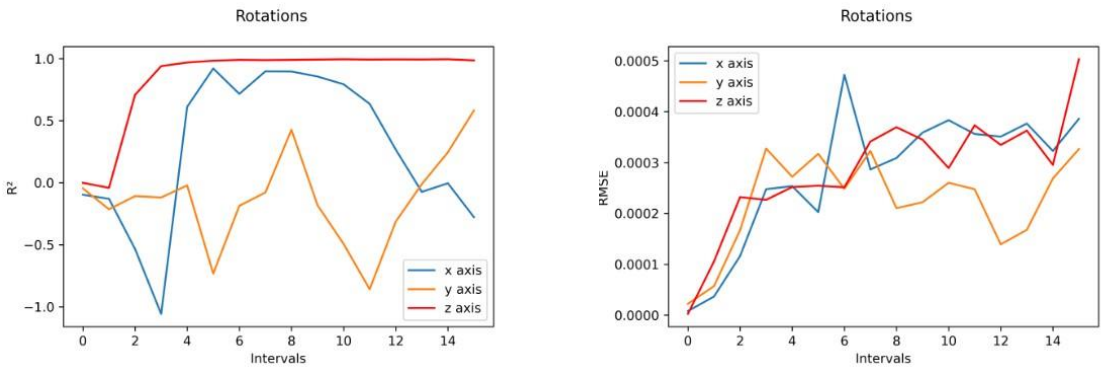

Figure 13: *R^2 and RMSE for Rotations using SVR.*

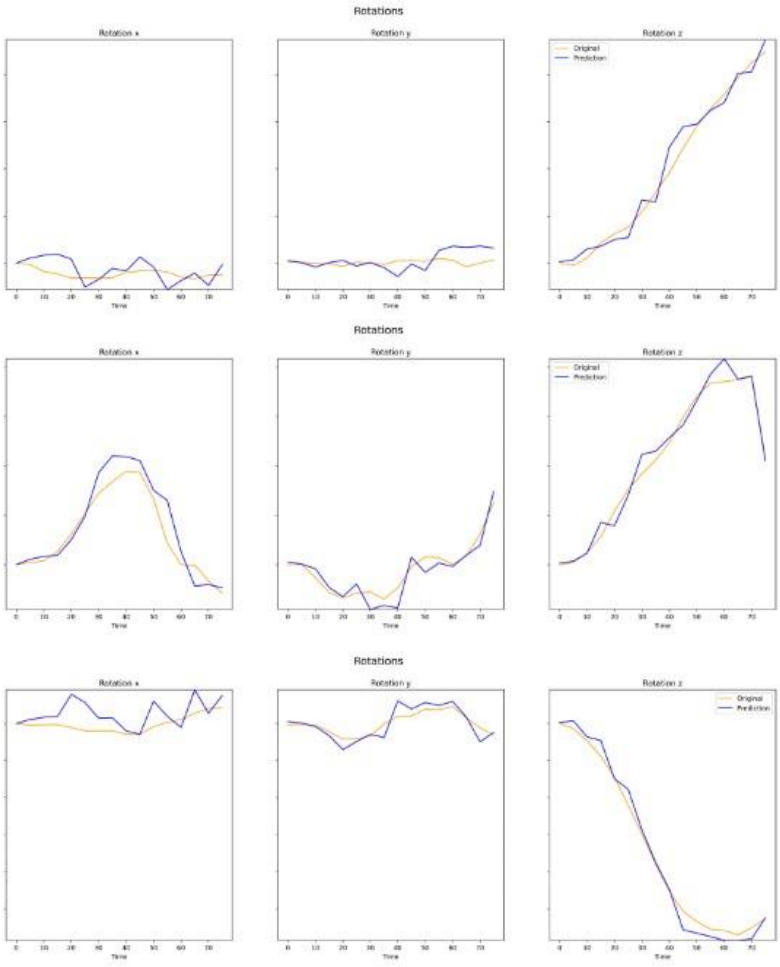

Figure 14: *Three test simples Predictions vs True Values for Rotations using NN.*

iv. Gradient Boost Regression

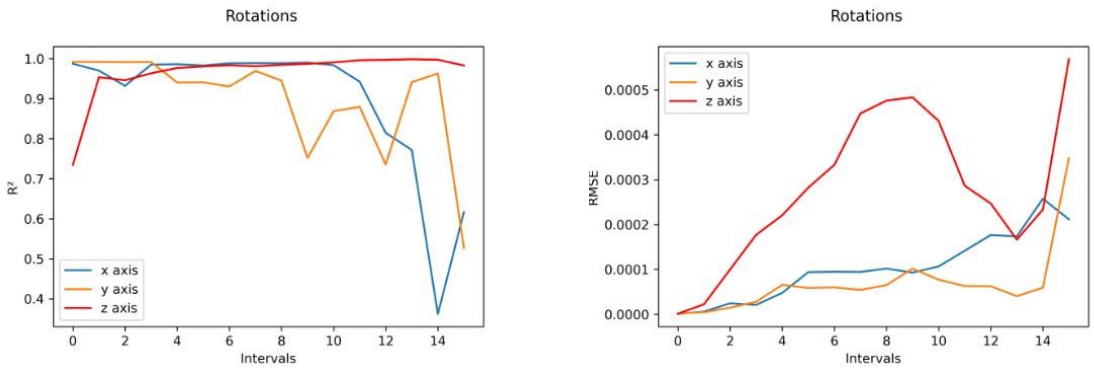

Figure 15: *R^2 and RMSE for Rotations using Gradient Boost Regression.*

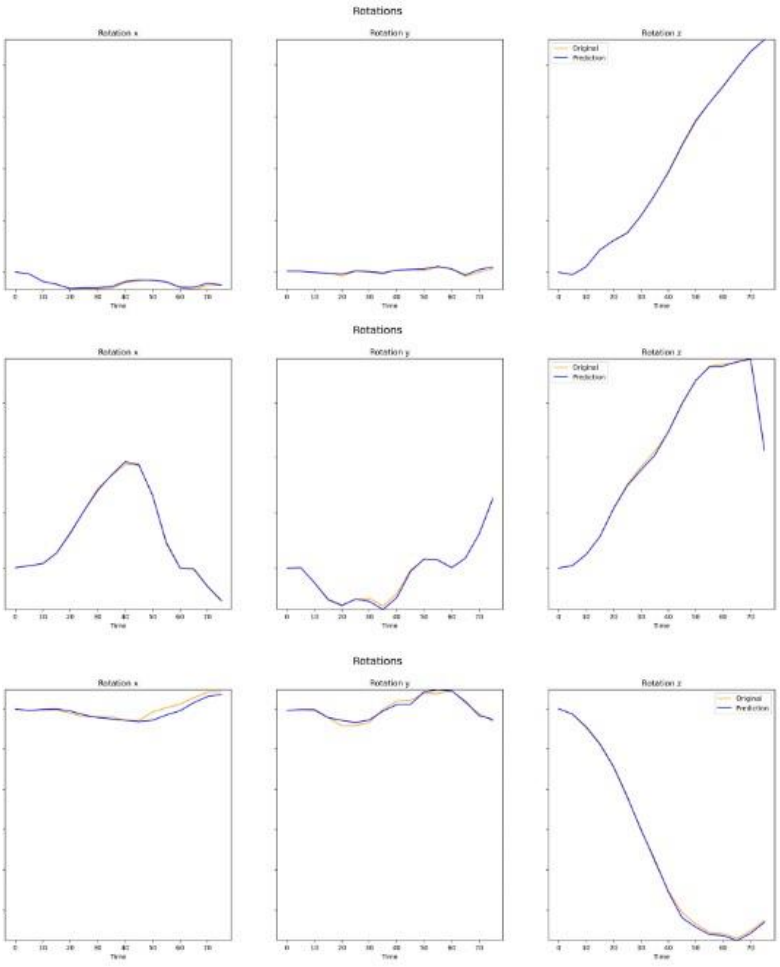

Figure 16: *Three test simples Predictions vs True Values for Rotations using NN.*

II. APPROACH WITH TRAINING SET FROM ALL SIMULATIONS

The second approach with the training set containing any type of collisions. As mentioned before, this training set was chosen from the general data cloud containing all types of crash configurations.

a. TRANSLATIONS

i. Neural Networks with MLP

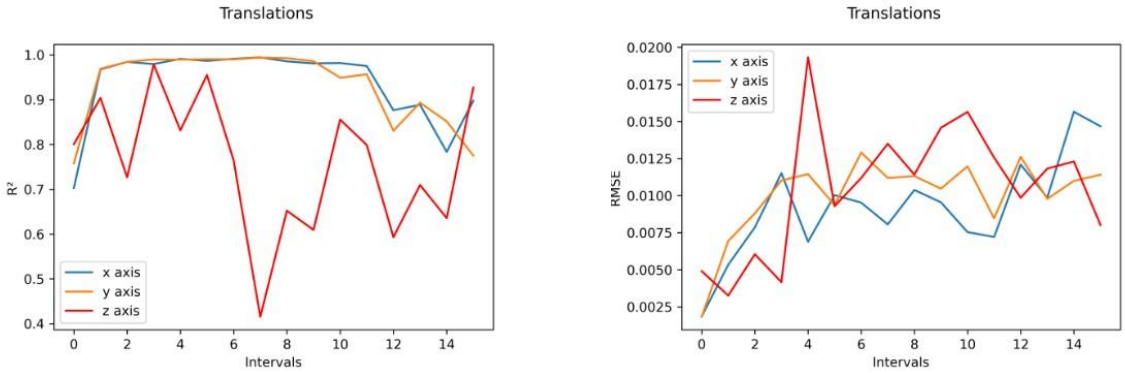

Figure 17: *R^2 and RMSE for Translations using NN.*

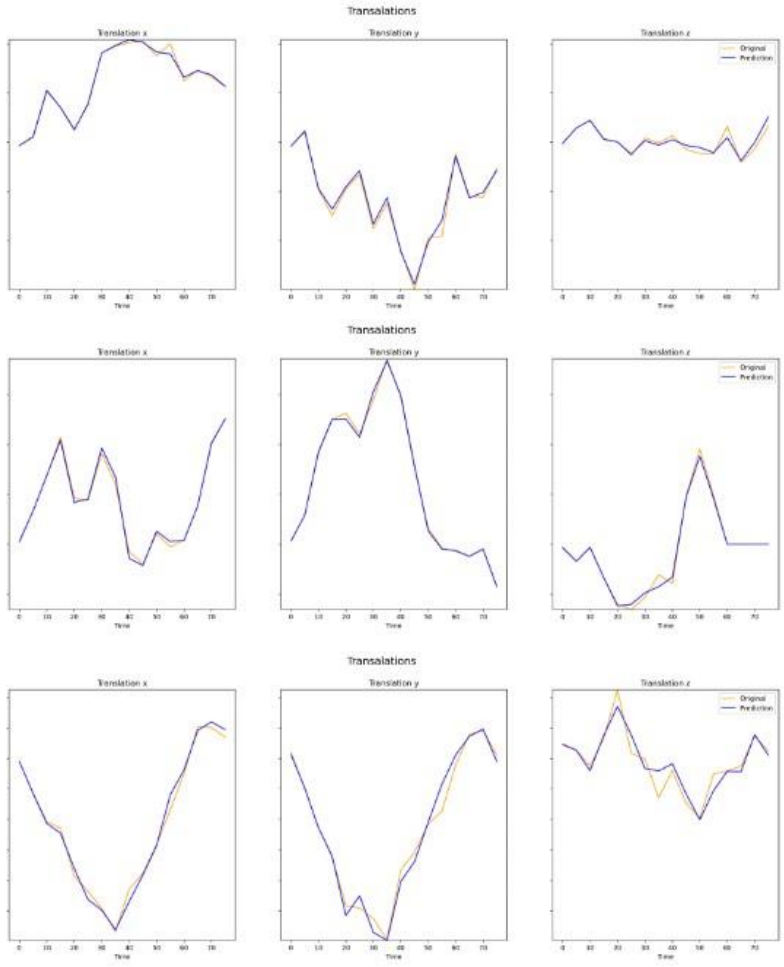

Figure 18: *Three test simples. Predictions vs True Values for Translational using NN.*

ii. Random Forest Regression

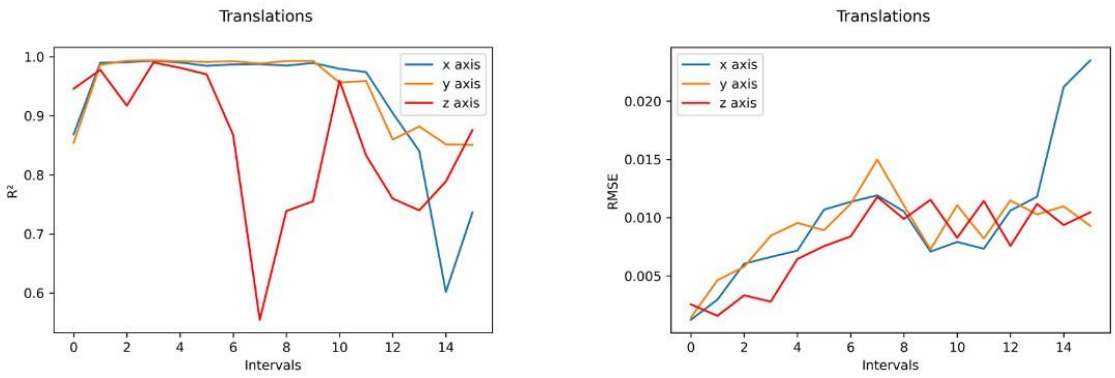

Figure 19: *R^2 and RMSE for Translations using Random Forest Regression.*

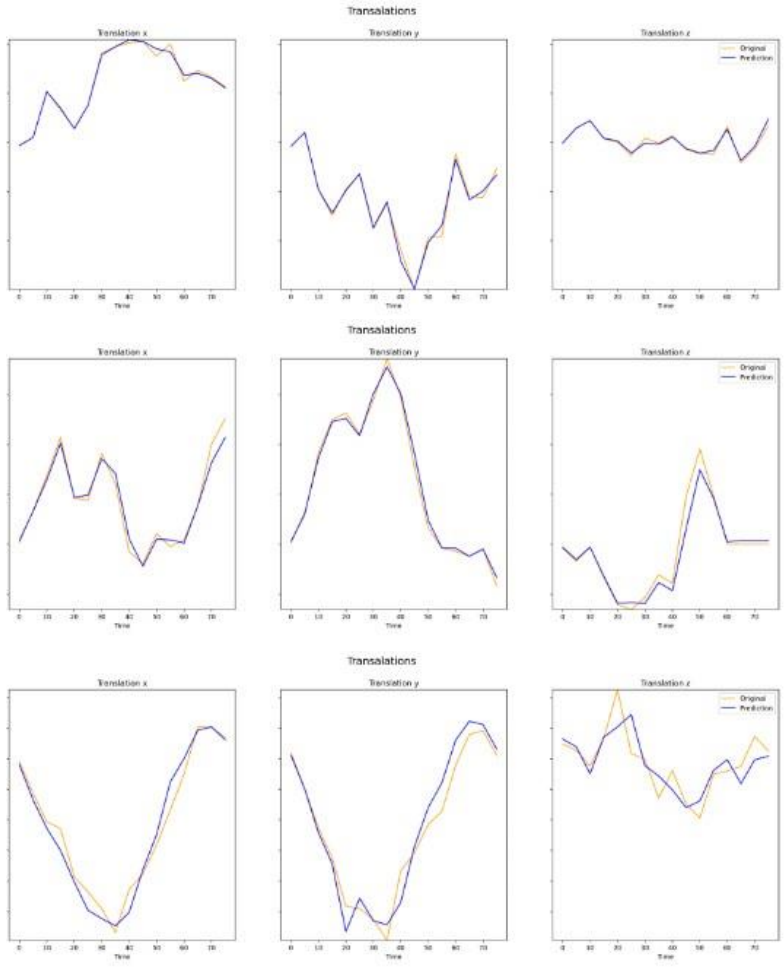

Figure 20: *Three test simples. Predictions vs True Values for Translations using Random Forest Regression.*

iii. Support Vector Machine Regression

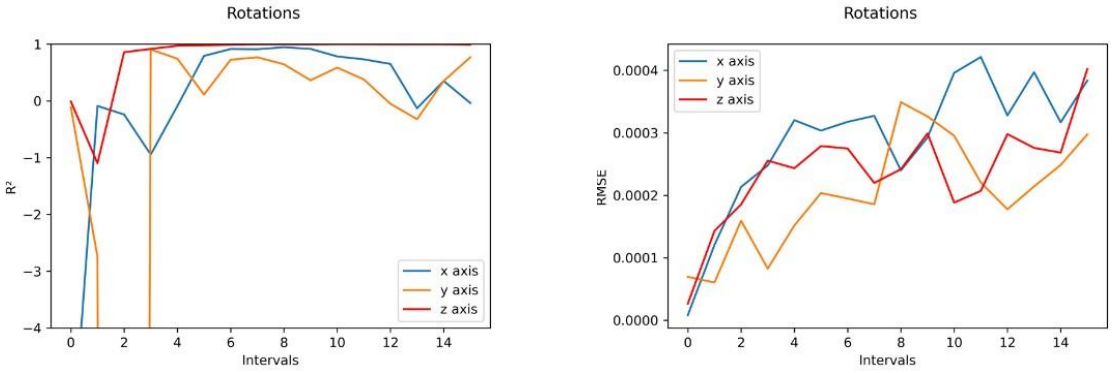

Figure 21: *R^2 and RMSE for Translations using SVR.*

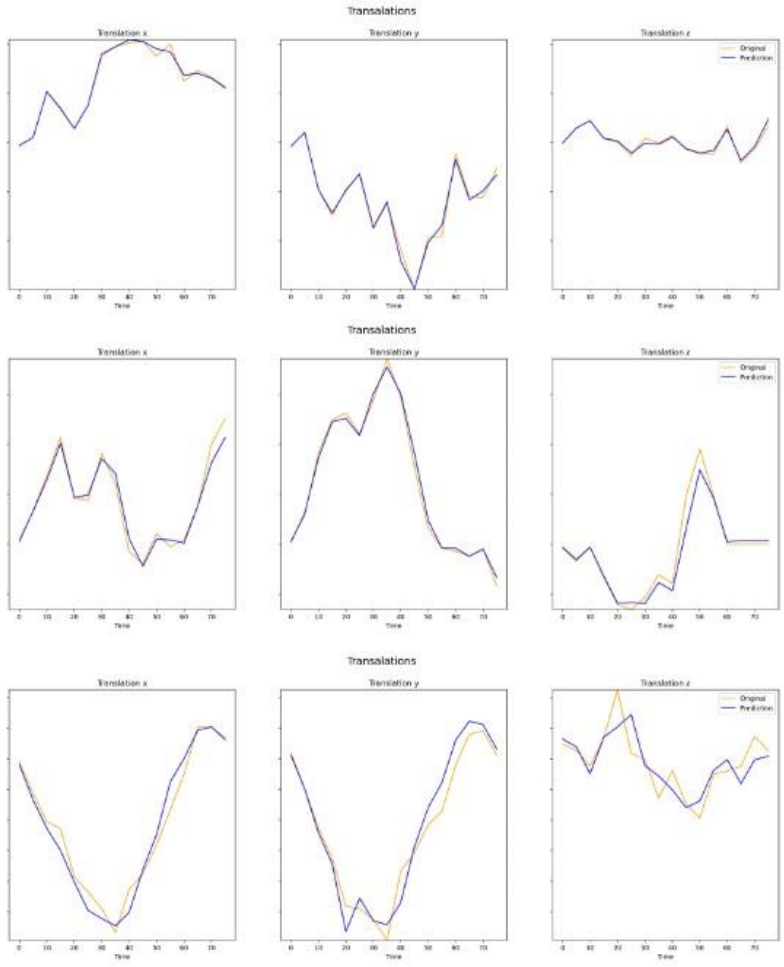

Figure 22: *Three test simples. Predictions vs True Values for Translations using SVR.*

iv. Gradient Boost Regression

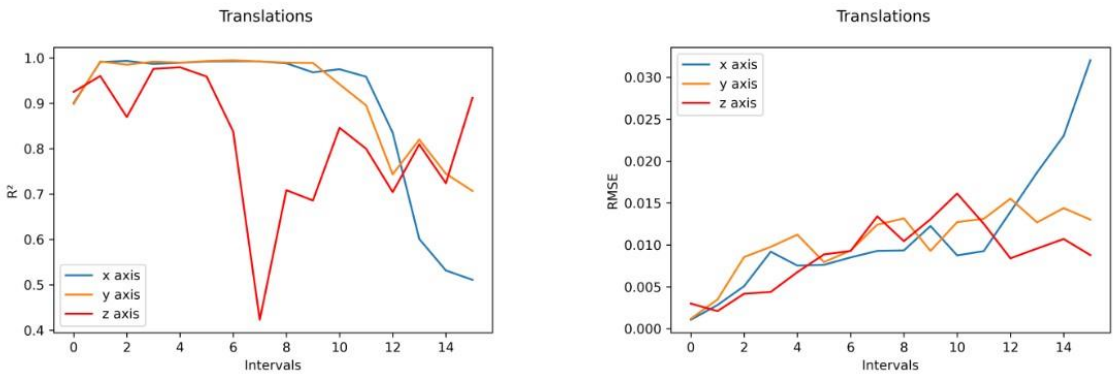

Figure 23: *R^2 and RMSE for Translations using Gradient Boost Regression.*

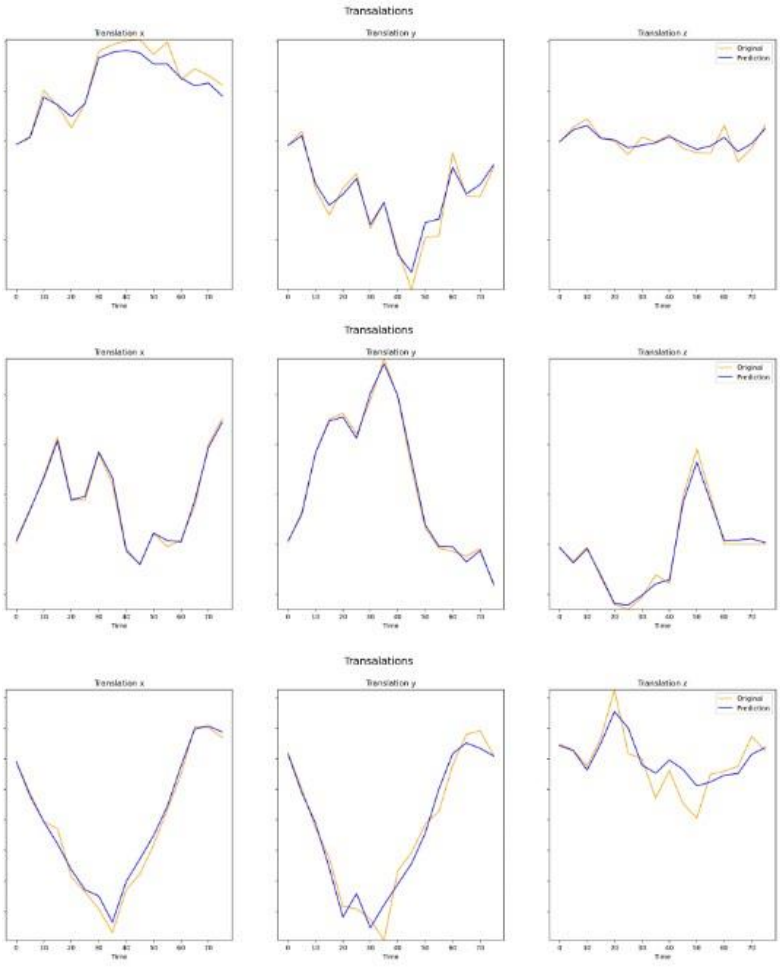

Figure 24: *Three test simples. Predictions vs True Values for Translations using Gradient Boost Regression.*

b. ROTATIONS

i. Neural Networks with MLP

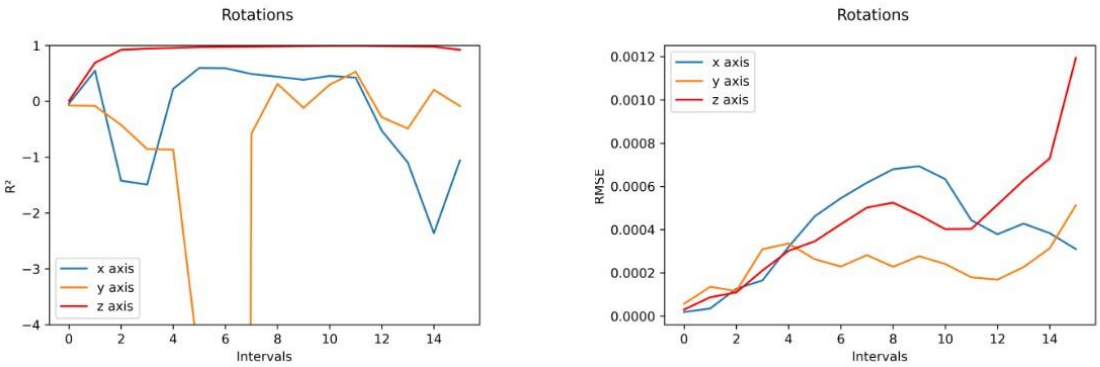

Figure 25: *R^2 and RMSE for Rotations using NN.*

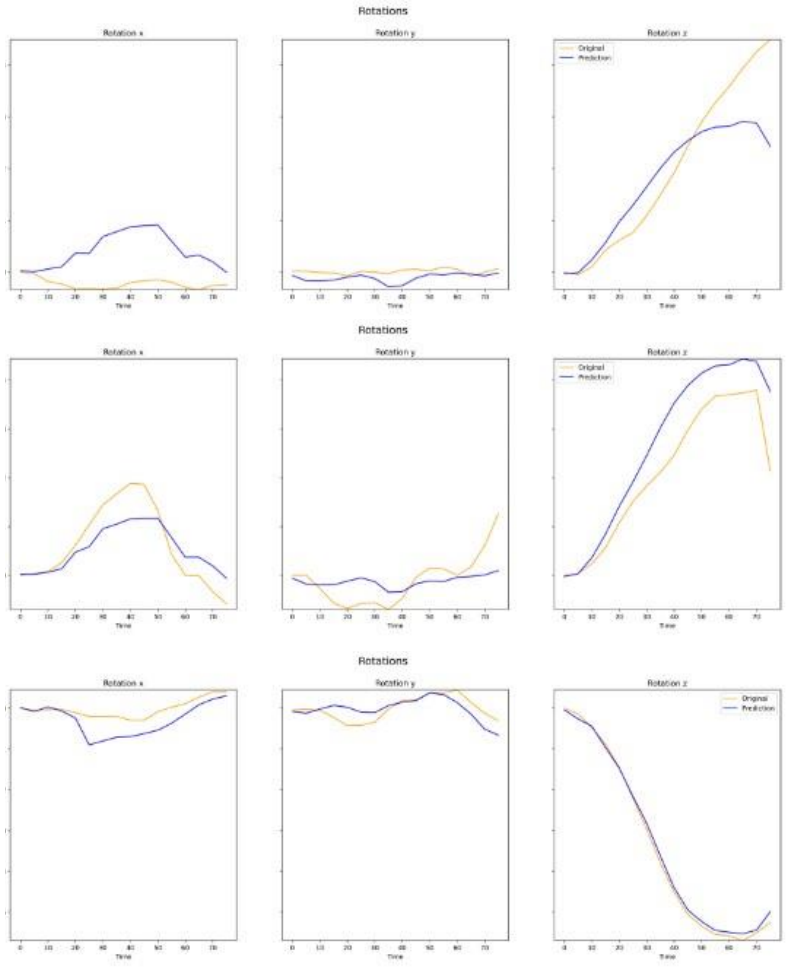

Figure 26: *Three test simples. Predictions vs True Values for Rotations using NN.*

ii. Random Forest Regression

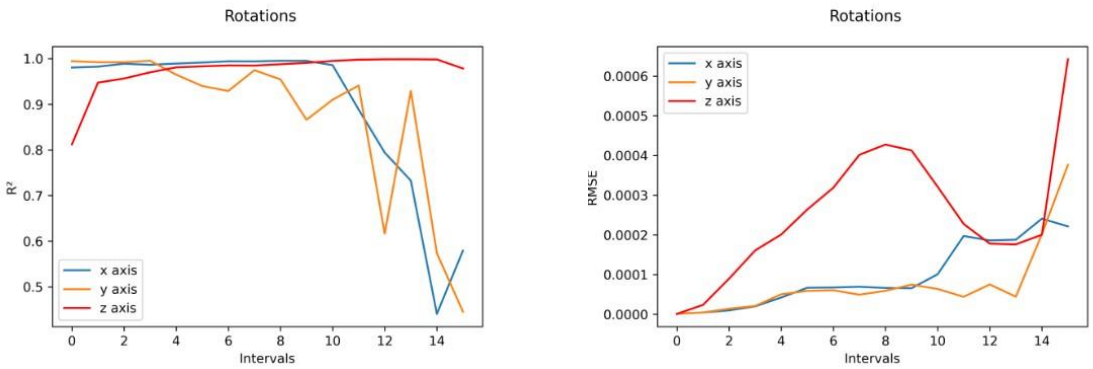

Figure 27: *R^2 and RMSE for Rotations using Random Forest Regression.*

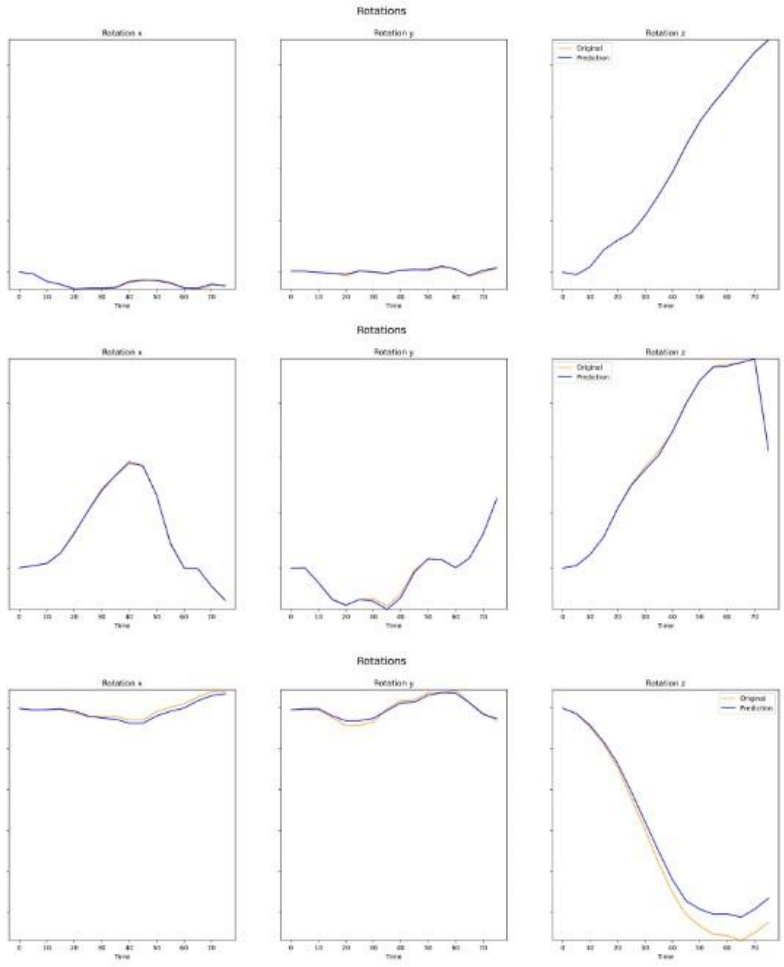

Figure 28: *Three test simples. Predictions vs True Values for Rotations using Random Forest Regression.*

iii. Support Vector Machine Regression

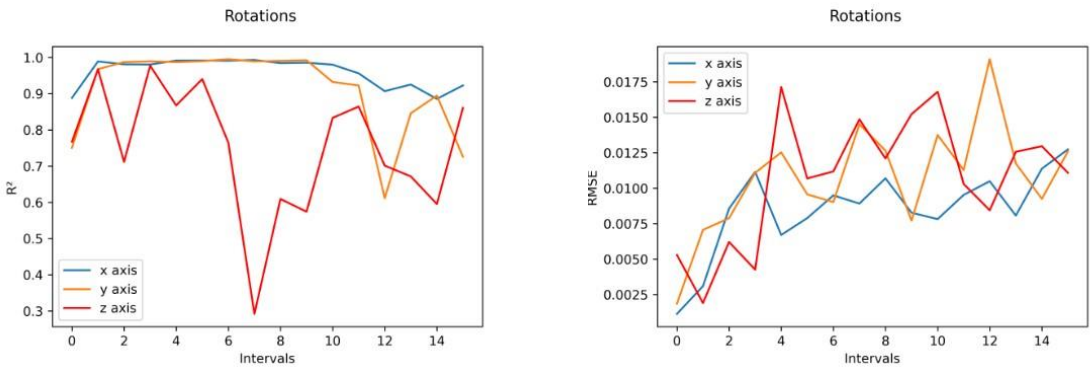

Figure 29: *R^2 and RMSE for Rotations using SVR.*

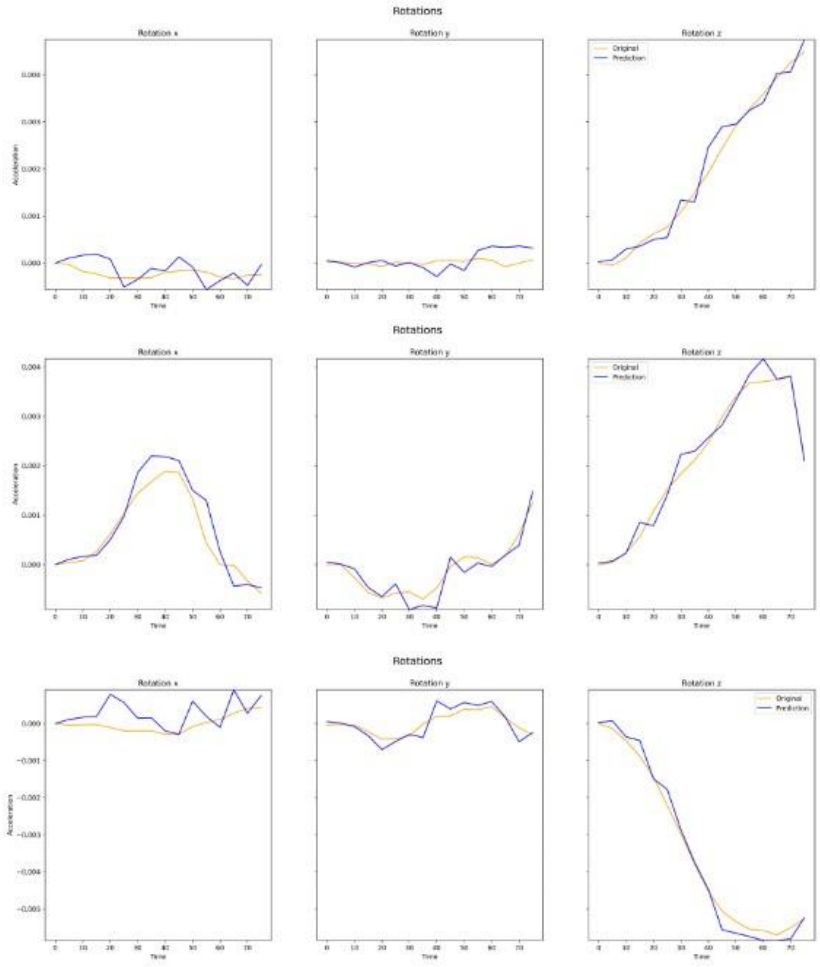

Figure 30: *Three test simples. Predictions vs True Values for Rotations using SVR.*

iv. Gradient Boost Regression

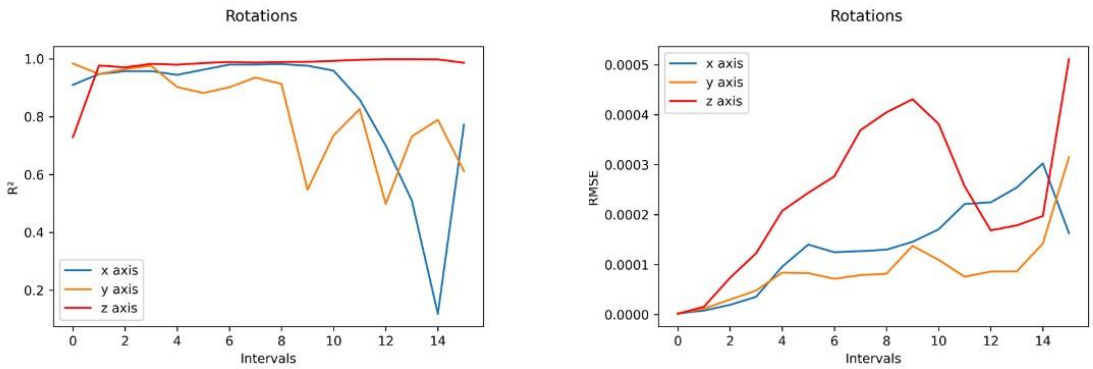

Figure 31: *R^2 and RMSE for Rotations using Gradient Boost Regression.*

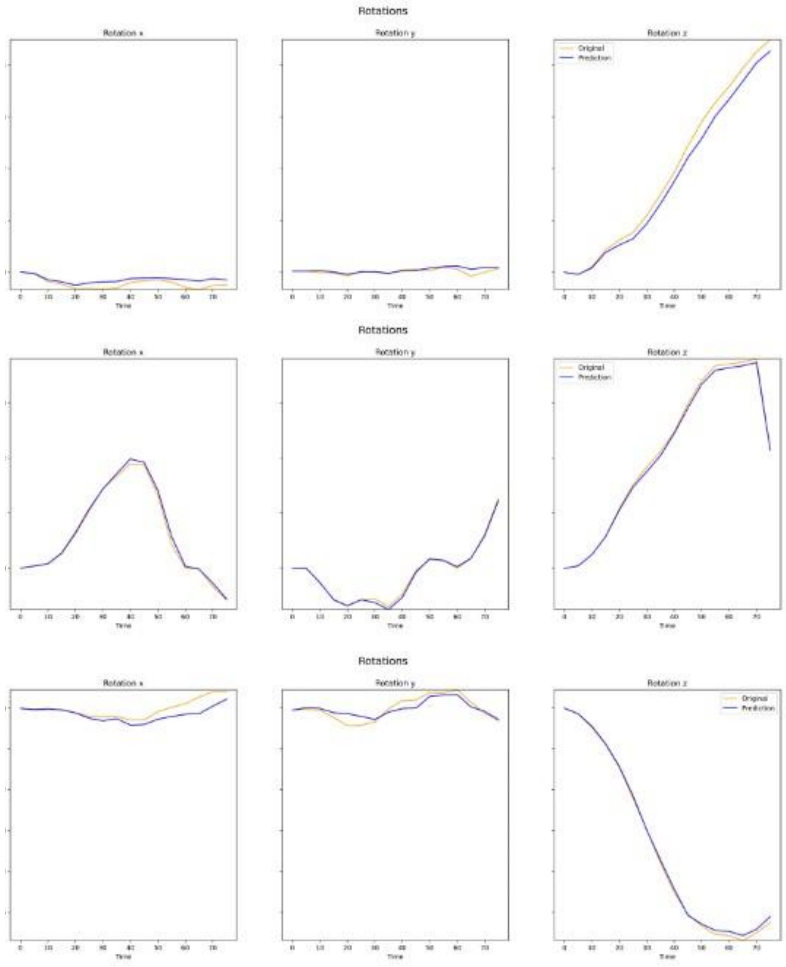

Figure 32: *Three test simples. Predictions vs True Values for Rotations using Gradient Boost Regression.*

APPENDIX II. MACHINE LEARNING TOOLS APPLIED

This appendix will explain different concepts used throughout the whole report: The main aim is to give a complete understanding of what the different methods used were, and how these were applied.

The first phase of the project consisted of gathering the data. During this phase, different tools or methods were used.

a. *K-Means Clustering*

During the project, it has been mentioned several times that the output files of the different simulations were used to extract information about the nodes. With this information, a clustering process was applied using the Python library of scikit-learn [1].

Clustering is referred to the process of grouping different objects or data into groups that share similar information. To do so, the objects have certain parameters that will help create the differentiation process to choose the clusters. This ML tool is an unsupervised learning algorithm, which means that the data is unlabeled, and the algorithm works by finding patterns in the introduced data.

For this project, the clustering process used was K-Means Clustering. This type of algorithm clusters the input data by separating the samples into groups with similar variance and minimizing the inertia between the points. This algorithm requires to introduce previously the number of clusters to be determined, which in this case, was set to be 2, as there were two vehicles.

The k-means algorithm chooses the centroid of the clusters to reduce the inertia or within-cluster sum of squares criterion [1]. The inertia concept is referred to as a measure of how internally coherent the clusters are. This is decided by the unlabeled information introduced to the algorithm. In this project, several dimensions of information were given. The input data were the nodes with the x-velocities, y-velocities, and the ids of the nodes. By applying this last dimension, the algorithm did not take only into account the velocities of the nodes corresponding to both vehicles, but also the numbered nodes. This last dimension was critical to cluster both vehicles accordingly, as the offset in the number of the nodes was of crucial importance to differentiate between the vehicles.

b. *Data split*

During the project, it has been mentioned several times that the data was split into training and test data sets. This process was used to differentiate the data from which the Ml algorithm would learn and the data from which the ML algorithm would be validated and tested.

For this purpose, the scikit-learn Python library was used once again [2]. This algorithm allows the user to set different parameters to divide the data into train and test data.

sklearn.model_selection.train_test_split*(*arrays,test_size,train_size)*

The algorithm needs to be introduced the two required arrays of data. These are the input array with the variables of the crash configuration; and the output arrays with the data of the crash pulses. By doing so, the algorithm allows the user to introduce the size or proportion of the corresponding training and test datasets. These were set to be 80% of the total data to be the training set and 20% of the data to be the test set.

c. *ML algorithms*

A brief introduction to the different algorithms used has been given. However, in this appendix, a more detailed explanation will be given.

Machine Learning is the science of getting computers to act, learn and develop certain processes without being specifically programmed to do so [3]. Multiples algorithms, applications, and tools have been developed and perfectioned during the last decade, becoming very useful and popular in different fields.

These algorithms have been used in different applications to make predictions, find patterns and give insights. For this reason, the introduction of these algorithms have been very useful for the development of this project, where four main tools or methods have been applied:

i. Regressions

Regression is a statistical method used in many fields that attempts to determine the connection between one variable and a series of other variables. This method takes a group of random variables to predict the output variable noted as Y, and it tries to find a mathematical relationship between them. The most common approach to relate these variables is a straight line An example can be seen below:

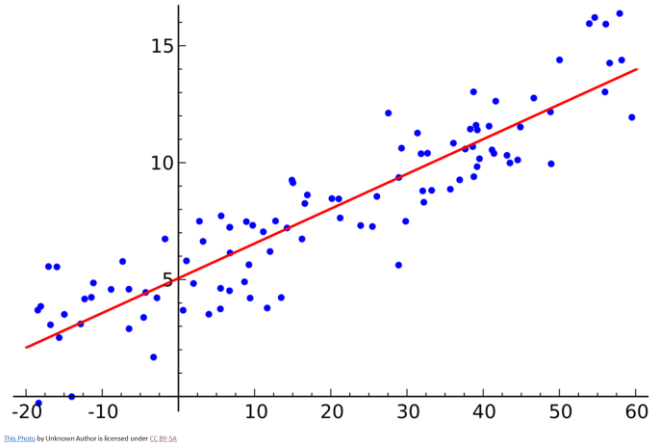

Figure 1: Linear Regression example [5].

Therefore, the ML algorithm tries to fit, in this case, a line that will describe the data in the most accurate way possible, trying to reduce the error between the predictions and the true values.

ii. Neural Networks.

Neural Networks correspond to a more complex algorithm. These are particularly called Deep Learning algorithms. A neural network is a series of algorithms that looks to recognize relationships in a set of data through a process that emulates the way the human brain operates. In this sense, neural networks refer to systems of neurons. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems [6].

Some of the key concepts to understand neural networks are the following:

- Layer: each of the sections of a Neural Network.
- Input layer: initial data for the neural network. The size of this layer must match the size of the input data.
- Hidden layer: intermediate layer or layers that are between the input layer and output layer. This is the main part of the computation is done.
- Output Layer: the layer that returns the results of the algorithm and must match the size of the desired output.

An example of a neural network can be seen below:

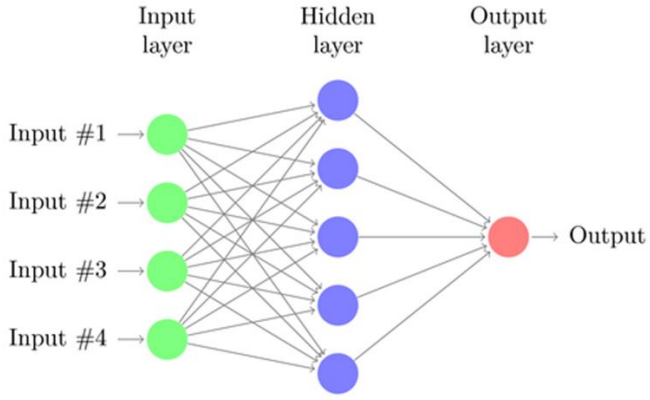

Figure 2: Neural Network example [5].

During the different iterations between the hidden layer and the input layer, the parameters are known as "weights" and "bias" are estimated and computed. The initial values of these parameters start as random values and then the algorithm makes corrections and changes to reduce the error function specified. This error function has been already explained in the report.

iii. Decision Trees.

One concept that has been mentioned throughout the report is the term of Decision Trees. This corresponds to the basics used in both, Random Forest Regression and Gradient Boost Regression.

The Decision Trees (DTs) are a non-parametric supervised learning method usually used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules from the input data [7].

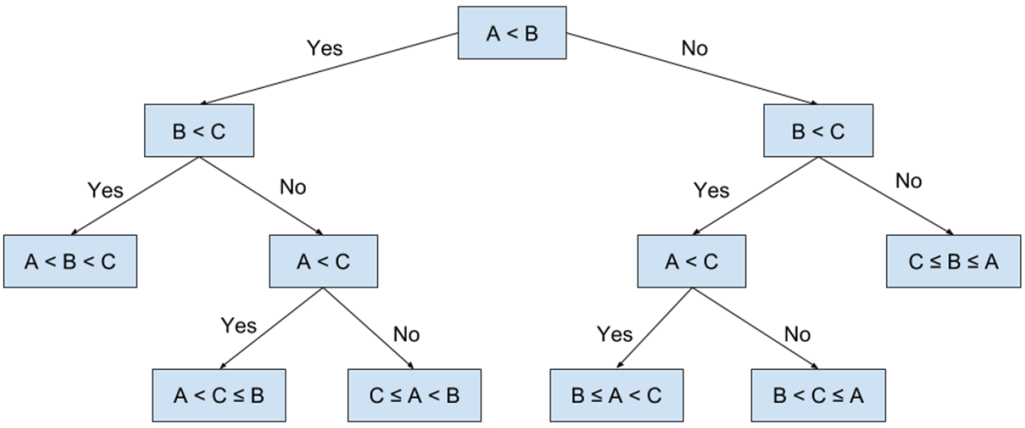

Figure 3: Decision Tree example [5].

A decision tree is drawn upside down with its root at the top. In the figure, the ramifications represent a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the A, B, or C satisfy the different conditions.

The main aim is, therefore, to reach a state where the data satisfies the conditions given. In terms of a Decision Tree used for Regression, the conditions applied are the error functions that will be computed at every branch. By doing so, the Decision Tree will be obtaining or predicting the output parameters.

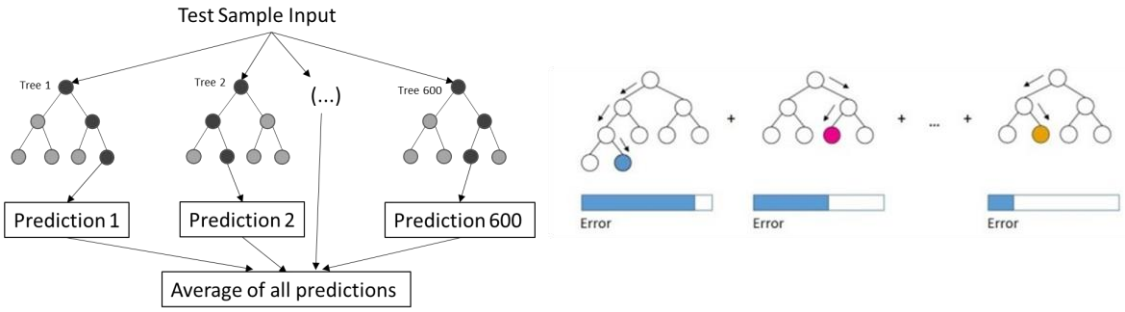

Figure 3: Random Forest example (left) and Gradient Boost example (right) [5].

These DTs are the core of the Random Forest Regression and the Gradient Boost Regression, wherein both cases, are used to make the desired predictions. As explained in the report, the Random Forest will be applying the specified number of DTs and computing the average of the predictions. And the Gradient Boost Regression will be overlapping DTs and focusing on those branches where the error function returned larger values.

iv. Support Vector Machine Regression

Commonly known as Support Vector Regression (SVR), it is a method that allows having a different approach than most linear regression algorithms. In these models, the main objective is to reduce the error understood as the sum of squared errors. However, the SVR gives the flexibility to define how much error is acceptable in the model. This parameter is called epsilon (ϵ). Therefore, the model creates a line that fits the data under the limits given to the error. In this case, the model creates a hyperplane to fit the data [8]. This hyperplane can be chosen between different options to obtain the most suitable to fit the data. Another parameter to take into account is C, which controls the tolerance for points outside the interval given by ϵ. Therefore, this can help control or mitigate the presence of outliers.

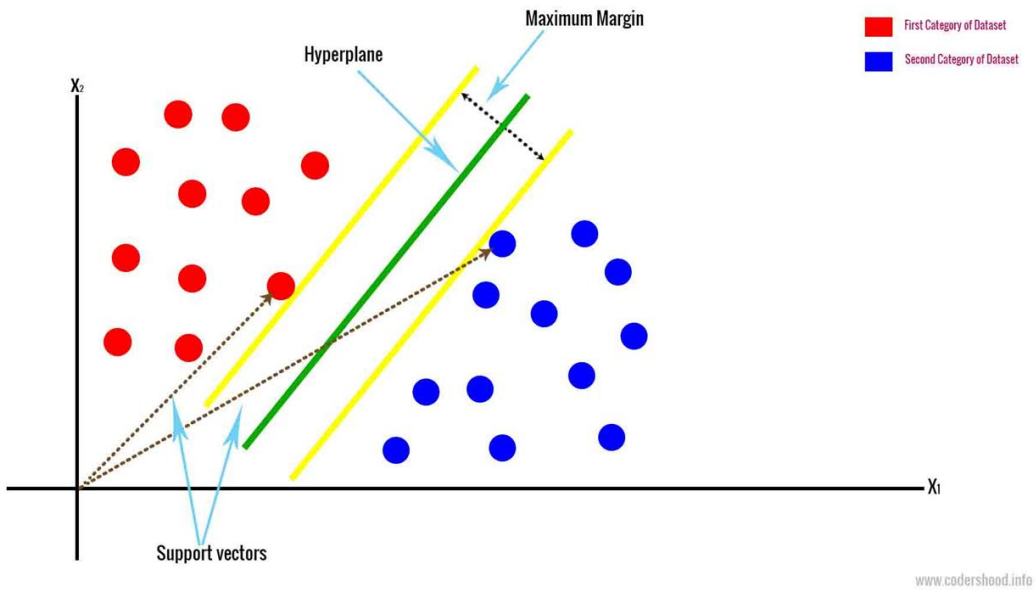

Figure 4: Support Vector Machine Regression [5].

In this example, the SVM creates a differentiation technique that will produce two different groups. This is performed with the mentioned hyperplane, a function that will be introduced to split the data with a specified maximum margin, making sure that these conditions are satisfied.

References

[1] Scikit learns, "2.3 Clustering". URL: https://scikit-learn.org/stable/modules/clustering.html.

[2] Scikit learn, "sklearn.model_selection.train_test_split". url: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[3] Coursera, "Machine Learning". URL: https://coursera.org/learn/machine-learning

[4] Investopedia, "Regression Definition", url: https://www.investopedia.com/terms/r/regression.asp#:~:text=Regression%20is%20a%20statistical%20method,(known%20as%20independent%20variables).

[5] Creative Commons. URL: https://creativecommons.org/

[6] Investopedia, "Neural Network", URL: https://www.investopedia.com/terms/n/neuralnetwork.asp#:~:text=Neural%20networks%20are%20a%20series,fraud%20detection%20and%20risk%20assessment.

[7] Scikit learn, "1.10 Decision Trees". URL: https://scikit-learn.org/stable/modules/tree.html

[8] Scikit learn, "sklearn.svm.SVR". URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

APPENDIX III. ALIGNMENT WITH THE SDGS

"The Sustainable Development Goals are the blueprint to achieve a better and more sustainable future for all. They address the global challenges we face, including poverty, inequality, climate change, environmental degradation, peace, and justice."[1]

As stated, these goals have a wide range of scope. For this purpose, the next Sustainable Development Goals can be related to the completion of this project:

- The SDG number 3 corresponds to Good Health and Well Being. In general terms, the main purpose of this project is to develop new technologies capable of assessing and predicting possible injuries or consequences of traffic collisions. Therefore, this project can be directly related to this SDG as it moves towards the health of the people in terms of the occupants of the vehicles. Specifically, the target that relates the most with the aim of this project is Target 3.6:" By 2020, halve the number of global deaths and injuries from road traffic accidents" [1].
- The SDG number 9 corresponds to Industry, Innovation, and Infrastructure. This goal is referred to build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation. These are linked to this project in terms of innovative solutions that aim to establish new ways or infrastructures to assess differently the traffic collisions and create a safer and friendlier environment.
- The SDG number 10 corresponds to Reduced Inequalities, which is highly related to the previous one, as one of the aims of this project is to make more accessible the research of the safety of current vehicles and trying to reach every country.

[1] United Nations. Sustainable Development Goals. 2018. url: https://www.un.org/sustainabledevelopment/.