



universidad
de león



Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Memoria del Proyecto RateMyCar Sistemas de Información de Gestión y Business Intelligence

Autor: Bechari Ayoub 1/2/2025

Índice

1. Introducción	3
2. Contexto, Problema y Solución	4
3. Herramientas y metodologías empleadas	5
3.1. Scikit-learn (sklearn)	5
3.2. Streamlit	5
3.3. Groq (para Llama 3)	5
3.4. Llama 3	6
3.5. API de Google (para imágenes de los modelos de coches)	6
3.6. Entornos de desarrollo	6

3.7. Herramientas estudiadas pero no utilizadas:	6
4. Flujo de la aplicación desarrollada	7
5. Modelo de Machine Learning (Random Forest Regression)	11
5.1. Preprocesamiento de Datos	11
5.1.1. Carga y exploración del conjunto de datos	11
5.1.2. Limpieza de Datos	13
5.1.3. División de Datos para Entrenamiento y Prueba	14
5.1.4. ¿Por qué usar Scikit-Learn?	14
5.2. Selección del Modelo	14
5.2.1. ¿Por qué usar Random Forest Regression?	14
5.2.2. Entrenamiento con Random Forest	14
5.2.3. Optimización de Hiperparámetros con RandomizedSearchCV	15
5.3. Evaluación del Modelo	15
5.3.1. Distribución de errores	15
5.3.2. Comparación de valores reales vs predichos	15

5.3.3. Guardado del Modelo.....	15
5.4. Resumen del Proceso en un Diagrama.....	16
5.5. Conclusión	16
6. Análisis DAFO	17
6.1. Debilidades.....	17
6.2. Amenazas	17
6.3. Fortalezas	17
6.4. Oportunidades	18
7. Lecciones aprendidas	20
8. Bibliografía, referencias y recursos	21

1. Introducción

En la presente memoria se detalla el desarrollo de la aplicación web RateMyCar, diseñada para predecir el valor de automóviles usados utilizando modelos de Machine Learning. Este proyecto ha sido desarrollado como parte del estudio de técnicas avanzadas de análisis de datos y sistemas de inteligencia artificial. A lo largo de este documento, se analizará el problema que la aplicación busca resolver, las tecnologías utilizadas, la arquitectura del sistema y los algoritmos empleados. Además, se describirá el funcionamiento general de la aplicación mediante ejemplos prácticos. Se llevará a cabo un análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) para evaluar los puntos clave del sistema, así como posibles mejoras futuras. Finalmente, se presentarán conclusiones basadas en la experiencia obtenida durante el desarrollo del proyecto.


2. Contexto, Problema y Solución


Cuando se habla de valoración de autos usados en línea, a menudo se piensa en las herramientas clásicas de estimación que proporcionan un valor indicativo basado en parámetros genéricos. Sin embargo, el proceso de venta y compra de un automóvil es mucho más complejo y no se limita solo a la valoración económica. El verdadero problema no es solo saber cuánto vale un automóvil, sino comprender **qué alternativas podrían ser mejores** con el mismo presupuesto.

En los últimos años, el mercado de autos usados ha crecido rápidamente, lo que ha llevado al surgimiento de numerosas plataformas de compraventa. Sin embargo, estas solo ofrecen filtros y búsquedas por características técnicas, dejando al usuario solo en el proceso de toma de decisiones. Incluso con herramientas avanzadas de búsqueda, la cantidad de opciones puede resultar **abrumadora**, lo que dificulta encontrar el automóvil más adecuado para sus necesidades.

Por esta razón, el proyecto propone un sistema que **no solo estima el valor del automóvil del usuario**, sino que también genera de manera inteligente una página personalizada con información útil. Gracias a un modelo **Random Forest Regressor**, se calcula el valor estimado del vehículo, mientras que un agente basado en **Llama 3** analiza el modelo del automóvil y sugiere dos alternativas mejores al mismo precio, destacando también posibles problemas comunes del vehículo actual.

El objetivo no es competir con las plataformas tradicionales de compraventa, sino ofrecer una **herramienta complementaria** que pueda integrarse en estos sistemas, mejorando la experiencia del usuario e incentivando la compraventa tanto para los clientes como para los vendedores.



 Ingresa los datos de tu coche

Marca del coche

wagon

Año de fabricación

2013

2003 2017

Kilómetros recorridos

159165

500 500000


Tipo de combustible

Petrol


Tipo de transmisión

Manual

Precio estimado: €11331



wagon

 Nuestros consejos

Basado en la información proporcionada, yo como experto asesor de automóviles, te proporciono la siguiente respuesta:

Problemas comunes: Con 159,165 km recorridos, se pueden presentar problemas comunes en el wagon, como:

3. Herramientas y metodologías empleadas

El desarrollo de este proyecto ha requerido el uso de diversas herramientas y metodologías para la recopilación, procesamiento y análisis de datos, así como para la implementación del modelo de predicción y la generación de recomendaciones. A continuación, se detallan los principales recursos utilizados.

3.1. Scikit-learn (sklearn)

Scikit-learn es una de las bibliotecas más populares de Python para el machine learning. En este proyecto, se utilizó el modelo *Random Forest Regressor* de Scikit-learn para predecir el valor de los automóviles en base a un conjunto de datos proporcionado. El modelo *Random Forest* es especialmente adecuado para tratar con conjuntos de datos complejos y no lineales, como es el caso del valor de los vehículos, ya que maneja bien tanto variables numéricas como categóricas.

Nota: Scikit-learn es ampliamente utilizado por su simplicidad, versatilidad y su excelente documentación, lo que facilita la implementación y evaluación de modelos de machine learning.

3.2. Streamlit

Streamlit es una herramienta *open-source* que permite crear aplicaciones web interactivas directamente con Python. En este proyecto, se utilizó para desarrollar una interfaz de usuario intuitiva que permite al usuario interactuar con el modelo de predicción de valor de automóviles y obtener recomendaciones de vehículos alternativos. Streamlit se destacó por su simplicidad, permitiendo crear la interfaz de usuario sin la necesidad de escribir código HTML, CSS o JavaScript.

Nota: Streamlit es ideal para desarrolladores de machine learning que desean crear rápidamente aplicaciones interactivas con pocos esfuerzos adicionales en el front-end.

3.3. Groq (para Llama 3)

Groq es una plataforma de cómputo especializada en la aceleración de modelos de inteligencia artificial. Para el componente de procesamiento del lenguaje natural del proyecto, esto permitió un procesamiento más eficiente.

3.4. Llama 3

Es un modelo de procesamiento de lenguaje natural altamente avanzado, utilizado para generar respuestas y recomendaciones basadas en texto.

3.5. API de Google (para imágenes de los modelos de coches)

3.6. Entornos de desarrollo

El desarrollo del proyecto se llevó a cabo utilizando dos entornos principales de desarrollo:

- **Google Colab:** Una plataforma que permite escribir y ejecutar código Python en la nube, aprovechando recursos como GPUs y TPUs para entrenar modelos de machine learning de manera eficiente.
- **Visual Studio Code:** Un editor de código fuente altamente versátil.

3.7. Herramientas estudiadas pero no utilizadas:

- TensorFlow y PyTorch (se consideraron pero Scikit-learn fue suficiente para este caso)
- Flask
- Selenium para scraping de datos (descartado por restricciones de acceso a fuentes de datos confiables)

4. Flujo de la aplicación desarrollada

La aplicación sigue un flujo de trabajo estructurado en tres etapas principales:

Entradas:

- Datos ingresados por el usuario: modelo del vehículo, kilometraje, año de fabricación.

Procesamiento (analizado en el próximo capítulo):

1. Se ingresa la información en el modelo de regresión *Random Forest* para estimar el precio de venta del automóvil.
2. Se envía la información a un agente con Llama 3 alojado en Groq, que proporciona:
 - Posibles problemas del vehículo.
 - Dos alternativas de compra con mejor relación calidad-precio.
3. Se consulta la API de Google Custom Search para obtener imágenes del vehículo ingresado por el usuario.

Salidas:

- Precio estimado de venta del automóvil.

 Precio estimado: €8194

- Imagen del vehículo proporcionada por la API de Google.



- Evaluación de posibles problemas del vehículo.

Nuestros consejos

Problemas comunes

Con un ritz con 128134 km recorridos, pueden ocurrir problemas comunes como:

- Fallos en el sistema de aire acondicionado y escapes del motor
- Problemas de transmisión y cambio de marchas
- Desgaste prematuro de los neumáticos y llantas
- Fallas en la iluminación y sistema de luces
- Posibles problemas en el sistema de frenos

- **Consumo.**

- Consumo típico**

- El consumo típico (consumo medio de combustible) de un ritz es de aproximadamente 5-6 litros por 100 kilómetros en ciudad y 3-4 litros por 100 kilómetros en carretera.

- **Dos alternativas de compra con mejor relación calidad-precio.**

- Alternativas similares**

- Dos alternativas de autos similares seguras y económicamente viables según el valor de mercado en euro estimado para este vehículo (8194) son:

- Toyota Etios (8000-8500 €)
 - Ford Fiesta (7800-8200 €)

5. Estructura del Código

La estructura del proyecto está organizada de la siguiente manera:

/RateMyCare	
— /model	# Contiene el modelo de regresión
— /data	# Contiene i dati del dataset delle auto
— app.py	# Il file principale di Streamlit per la web app
— agent.py	# Contiene l'implementazione dell'agente LLaMA 3
— ricerca.py	# Implementazione della ricerca e delle alternative per l'auto

5.1 Vamos a analizar qué sucede en el código

1. app.py - Gestión de la interfaz de usuario (UI)

- La app está construida sobre Streamlit, que ofrece una interfaz de usuario interactiva para el usuario.
- Los datos del auto (marca, año, kilometraje, tipo de combustible, etc.) son ingresados por el usuario mediante controles como slider y select-box.

```
# Diseño con dos columnas (30% parámetros - 70% resultados)
col_parametros, col_resultados = st.columns([1, 2])

with col_parametros:
    st.markdown("<h3>🚗 Ingresa los datos de tu coche</h3>", unsafe_allow_html=True)
    car_name = st.selectbox("Marca del coche", car_data["Car_Name"].unique())
    year = st.slider("Año de fabricación", 2003, 2025)
    kms_driven = st.slider("Kilómetros recorridos", 500, 500000)
    fuel_type = st.selectbox("Tipo de combustible", car_data["Fuel_Type"].unique())
    transmission = st.selectbox("Tipo de transmisión", car_data["Transmission"].unique())
    owner = st.selectbox("Número de propietarios anteriores", car_data["Owner"].unique())
    present_price_euro = st.slider("Precio del coche nuevo (€)", 3000, 100000)
```

- Una vez que el usuario presiona el botón "Predecir precio", la app carga el modelo de regresión desde un archivo pickle y predice el valor de venta del auto.

```
# Cargar el modelo entrenado
model = pickle.load(open("models/random_forest_regression_model (1).pkl", "rb"))
```

```
# Realizar la predicción
selling_price_predicted = model.predict(x_test)
predicted_price_inr = round(selling_price_predicted[0] * (10 ** 5), 2)
st.session_state.predicted_price_eur = predicted_price_inr / 90
```

- En esta sección, el valor de venta se convierte a euros y la app devuelve la imagen del auto (busqueda.py), la predicción del precio y los consejos

generados por el agente LLaMA 3 a través del módulo agent.py.

```
with col_resultados:
    if st.session_state.predicted_price_eur is not None:
        st.success(f"💰 Precio estimado: €{round(st.session_state.predicted_price_eur)}")

    if st.session_state.car_image_url:
        st.image(st.session_state.car_image_url, caption=f"{car_name}", use_container_width=True)
    else:
        st.warning("❌ ¡No se encontró ninguna imagen!")

    if st.session_state.ai_tips:
        st.subheader("📝 Nuestros consejos")
        st.write(st.session_state.ai_tips)
```

2. agent.py - Agente LLaMA 3

- El agente se encarga de generar respuestas sobre los problemas comunes que podrían ocurrir con un auto específico y sugerir alternativas similares.
- El módulo utiliza Groq para interactuar con la API LLaMA 3, un modelo de lenguaje avanzado que proporciona respuestas basadas en un prompt. Por ejemplo, si el usuario ingresa datos como la marca, el kilometraje y el año del auto, el agente devuelve:
 - Los problemas comunes que podrían ocurrir con ese coche específico.
 - El consumo medio de combustible.
 - Dos alternativas de autos similares basadas en el valor de mercado estimado.
- Las respuestas se devuelven en un formato conciso y estructurado, con un máximo de 150 palabras.

3. busqueda.py - Búsqueda de Imágenes a través de Google Custom Search

- Este módulo contiene una clase encargada de buscar imágenes de autos en Google, utilizando la API de Google Custom Search.
- Utiliza la API Key y el ID del motor de búsqueda para encontrar imágenes relevantes relacionadas con el modelo de auto seleccionado por el usuario.

```
def get_car_image_google(self, car_name):
    """Funzione per ottenere l'immagine di un'auto usando l'API di Google Custom Search"""
    url = f"https://www.googleapis.com/customsearch/v1?q={car_name}+car&cx={self.search_engine_id}&searchType=image&key={self.api_key}"

    response = requests.get(url).json()

    if "items" in response:
        return response["items"][0]["link"]
    else:
        return None # Nessuna immagine trovata
```

6. Arquitectura General

La arquitectura del proyecto está compuesta por tres componentes principales:

- **Interfaz de Usuario (UI):** Desarrollada con Streamlit.
- **Modelo de Machine Learning:** Utiliza un modelo de regresión Random Forest para estimar el precio de venta del vehículo.
- **Servicios Externos:** Interactúa con Llama 3 alojado en Groq para evaluar posibles problemas y con la API de Google Custom Search para obtener imágenes del vehículo.

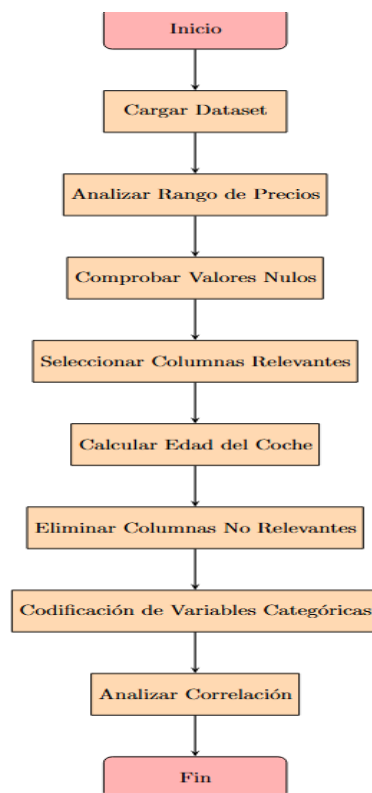
5. Modelo de Machine Learning (Random Forest Regression)

Para entrenar el modelo de predicción, se ha utilizado un dataset `car data.csv` de vehículos obtenidos de Kaggle. Este dataset contiene información detallada sobre automóviles, incluyendo características técnicas como:

- Año de fabricación
- Precio de venta
- Precio original
- Kilometraje recorrido
- Tipo de combustible
- Tipo de vendedor
- Transmisión
- Número de propietarios anteriores

La extracción y limpieza de los datos se realizó en Google Colab utilizando `pandas` para la manipulación de datos y `sklearn` para el preprocesamiento de variables. Se aplicaron técnicas de tratamiento de valores nulos y normalización para garantizar la calidad de los datos antes del entrenamiento del modelo.

El proceso de limpieza de los datos se lleva a cabo mediante los siguientes pasos representados en el diagrama de flujo:



5.1. Preprocesamiento de Datos

5.1.1. Carga y exploración del conjunto de datos

```
import pandas as pd
df = pd.read_csv('car data.csv') print(df.shape) #
Dimensiones del dataset print(df.head()) # Primeras filas
```

5.1.2. Limpieza de Datos

■ Verificación de valores nulos:

```
print(df.isnull().sum()) # Comprobamos valores faltantes
```

El conjunto de datos no tiene valores nulos, por lo que no es necesario manejarlos.

■ Transformaciones en los datos:

- Se añade una nueva columna para calcular la antigüedad del vehículo en años.
- Se eliminan columnas irrelevantes para mejorar la eficiencia del modelo.

```
import datetime
```

```
df['Current Year'] = datetime.datetime.now().year
df['Vehicle_Age'] = df['Current Year'] - df['Year']
df.drop(['Year', 'Current Year', 'Seller_Type'], axis=1, inplace=True)
```

■ Selección de Características:

- No todas las columnas del dataset original son relevantes para la predicción del precio de venta.
- Se eliminó Seller Type ya que no influye directamente en el valor de un automóvil.
- Car Name no se usa, ya que el nombre del automóvil es una variable categórica con muchas categorías, lo que podría agregar demasiada complejidad sin aportar mucho valor.
- Owner se conserva porque la cantidad de propietarios anteriores puede afectar el precio de venta.

■ Codificación de variables categóricas:

- Se convierten variables como Fuel Type y Transmission en variables numéricas usando pd.get_dummies() para que puedan ser utilizadas por el modelo de machine learning.

```
df = pd.get_dummies(df, drop_first=True)
```

5.1.3. División de Datos para Entrenamiento y Prueba

```
from sklearn.model_selection import train_test_split
X=df.drop(['Selling_Price'],axis=1) #Características y =
df['Selling_Price'] # Variable objetivo
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.3, random
```

5.1.4. ¿Por qué usar Scikit-Learn?

- **Facilidad de uso:** Proporciona funciones preconstruidas para procesamiento de datos, entrenamiento de modelos y evaluación.
- **Optimización:** Incluye herramientas avanzadas como RandomizedSearchCV para encontrar hiperparámetros óptimos.
- **Escalabilidad:** Puede manejar grandes volúmenes de datos y múltiples algoritmos de aprendizaje automático.

5.2. Selección del Modelo

5.2.1. ¿Por qué usar Random Forest Regression?

El *Random Forest Regressor* es un algoritmo basado en múltiples árboles de decisión. Es preferido porque:

- Reduce el sobreajuste (overfitting) en comparación con un solo árbol de decisión.
- Maneja datos no lineales mejor que métodos como la regresión lineal.
- Es robusto a valores atípicos y puede manejar grandes volúmenes de datos.
- Ofrece interpretabilidad mediante la importancia de las características.

5.2.2. Entrenamiento con Random Forest

```
from sklearn.ensemble import RandomForestRegressor
rf= RandomForestRegressor(n_estimators=100, random_state=42) rf.fit(X_train,
y_train)
```

5.2.3. Optimización de Hiperparámetros con RandomizedSearchCV

```
from sklearn.model_selection import RandomizedSearchCV import  
numpy as np
```

```
param_grid = {  
    'n_estimators': np.arange(100, 1200, 100),  
    'max_depth': np.arange(5, 30, 5),  
    'min_samples_split': [2, 5, 10],  
    'min_samples_leaf': [1, 2, 5], 'max_features': ['sqrt',  
    'log2']  
}
```

```
rf_random = RandomizedSearchCV(estimator=rf, param_distributions=param_grid,  
                               n_iter=10, cv=5, scoring='neg_mean_squared_error',  
                               random_state=42, verbose=2)
```

```
rf_random.fit(X_train, y_train)  
print(rf_random.best_params_)
```

5.3. Evaluación del Modelo

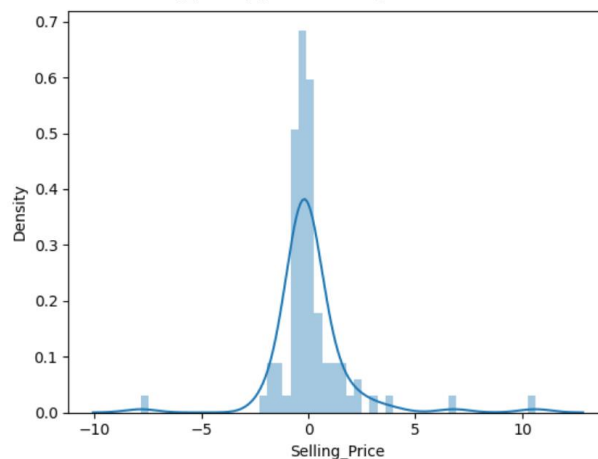
5.3.1. Distribución de errores

```
import seaborn as sns  
import matplotlib.pyplot as plt  
sns.histplot(y_test - rf_random.predict(X_test), kde=True) plt.show()
```

5.3.2. Comparación de valores reales vs predichos

```
plt.scatter(y_test, rf_random.predict(X_test))  
plt.xlabel("Valor Real") plt.ylabel("Predicción")  
plt.show()
```

Come se puede notar el model es bastante bueno



5.4. Conclusión

Este proyecto ha demostrado la eficacia del *Random Forest Regressor* para predecir precios de automóviles usados. Utilizando Scikit-Learn, logramos:

- Preprocesar eficientemente los datos.
- Seleccionar características relevantes.
- Optimizar el modelo mediante RandomizedSearchCV.
- Evaluar el rendimiento y guardar el modelo para futuras predicciones.

Este modelo puede ser mejorado con técnicas avanzadas como *Feature Engineering* y *Stacking Models* en futuros desarrollos.

6. Análisis DAFO

6.1. Debilidades

- Dependencia de datos históricos que pueden no reflejar cambios recientes en el mercado.
- Limitaciones en la precisión del modelo debido a la falta de datos específicos de algunos modelos de autos.
- Uso de APIs externas (Google Search y Groq) que pueden generar costos adicionales o limitaciones de uso.
- Dependencia de una conexión a Internet estable para obtener resultados en tiempo real.

6.2. Amenazas

- Cambios en las políticas de acceso de las APIs externas pueden afectar el funcionamiento de la aplicación.
- Aparición de competidores con modelos más avanzados o con acceso a bases de datos más completas.
- Posibles errores en la predicción del precio debido a fluctuaciones económicas o cambios en el mercado de autos usados.

6.3. Fortalezas

- Uso de un modelo de Machine Learning robusto (Random Forest Regression) con alta precisión.
- Implementación en Streamlit que permite una interfaz interactiva y fácil de usar sin necesidad de configuraciones complejas.
- Integración con Groq para aprovechar el poder de LLaMA 3 sin cargar el procesamiento en la máquina local.
- Generación automática de imágenes y consejos de IA, ofreciendo una experiencia enriquecida para el usuario.

6.4. Oportunidades

- Expansión del sistema incluyendo más datos y modelos de predicción avanzados.
- Posible monetización a través de recomendaciones personalizadas y servicios premium.
- Integración con concesionarios o plataformas de venta de autos para validar precios y mejorar la precisión del modelo.
- Mejora continua del modelo mediante técnicas avanzadas de aprendizaje automático.

Mejoras futuras

- **Mejorar el dataset:** Ampliar y enriquecer el conjunto de datos actual con más registros y variables relevantes, como la condición del vehículo, el historial de mantenimiento y datos de plataformas adicionales de compra-venta. Esto permitirá obtener predicciones más precisas y recomendaciones más personalizadas para los usuarios.
- **Crear un chat con una serie de agentes LLM:** Desarrollar un sistema de chat interactivo con varios agentes basados en Modelos de Lenguaje Grande (LLM), que ofrezcan respuestas personalizadas en tiempo real. Estos agentes podrán asesorar a los usuarios sobre el valor de los vehículos, posibles problemas, alternativas de compra y otros aspectos relacionados con el proceso de compra de automóviles.
- **Crear una base de datos gestionada por un agente de IA para las recomendaciones:** Implementar una base de datos robusta que contenga información detallada sobre los vehículos y las preferencias de los usuarios, gestionada por un agente de IA. Este agente será capaz de hacer recomendaciones personalizadas, actualizar la base de datos con nuevas entradas y optimizar las sugerencias en función de las tendencias del mercado y las necesidades individuales de cada usuario.

7. Lecciones aprendidas

Durante este curso, he adquirido una sólida comprensión de tecnologías avanzadas en el campo del machine learning, profundizando en la creación y aplicación de diversos modelos predictivos. He aprendido lo crucial que es disponer de un buen conjunto de datos, ya que este influye directamente en el rendimiento de los modelos. Además, he tenido la oportunidad de explorar y utilizar herramientas innovadoras que, antes de este curso, no habría tenido la oportunidad de descubrir, enriqueciendo así notablemente mis competencias tanto prácticas como teóricas.]

8. Bibliografía, referencias y recursos

Referencias

- [1] Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. <https://scikit-learn.org/>.
- [2] Streamlit Team (2021). Streamlit: The fastest way to build and share data apps. <https://streamlit.io/>.
- [3] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://link.springer.com/article/10.1023/A:1010933404324>.
- [4] Groq Inc. (2020). Groq: Accelerating AI workloads. <https://www.groq.com/>.
- [5] Kaggle Inc. (2021). Kaggle Datasets. <https://www.kaggle.com/datasets>.
- [6] Python Programmer (2022). Introduction to Machine Learning with Python. *YouTube*. <https://www.youtube.com/watch?v=7eh4d6sabA0>.