Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

"Yo, como integrante de la comunidad estudiantil del Tecnológico de Monterrey, soy consciente de que la trampa y el engaño afectan mi dignidad como persona, mi aprendizaje y mi formación, por ello me comprometo a actuar honestamente, respetar y dar crédito al valor y esfuerzo con el que se elaboran las ideas propias, las de los compañeros y de los autores, así como asumir mi responsabilidad en la construcción de un ambiente de aprendizaje justo y confiable"

"Inteligencia artificial avanzada para la ciencia de datos I"

Evidencias del reto - Módulo 4

Interfaz

Equipo:

Frida Cano Falcón A01752953

Jorge Javier Sosa Briseño A01749489

Guillermo Romeo Cepeda Medina A01284015

Daniel Saldaña Rodríguez A00829752

Profesor:

Antonio Carlos Bento

Fecha de entrega: 13 de septiembre de 2023

Introducción:

El reto del curso consiste en la utilización del método CRISP-DM (Cross Industry Standard Process for Data Mining) para generar una solución que integre ciencia de datos e inteligencia artificial utilizando herramientas computacionales de vanguardia. Esto implica la obtención de una base de datos confiable que nos permita hacer un análisis de los mismos, con la finalidad de predecir una variable. En este caso, se encuentra con el reto de predecir si una persona que atendió al Titanic sobrevivió o no.

Lo primero que se realizó para empezar a solucionar el problema, fue identificar el problema, así como los requerimientos que tiene el reto y herramientas que utilizaremos para completar los requerimientos.

En primera instancia fue necesario descargar la base de datos de los pasajeros del Titanic con sus principales características, como lo es nombre, número de ticket, sexo, edad, etc.

Lo segundo fue utilizar una plataforma que permite utilizar herramientas de matemáticas, así como inteligencia artificial para darnos una mejor oportunidad de resolver el problema, así como poder trabajar en conjunto. Para esto se decidió utilizar el lenguaje de programación Python, el cuál desarrollamos en la plataforma de Google Colab, por medio de un tipo de archivo llamado Jupiter Notebook.

También se identificó que el primer paso para generar una predicción del deceso o sobrevivencia de los pasajeros del Titanic es la limpieza de la base de datos, esto debido a que se tienen variables que no aportan información útil acerca del evento que ocasionó la muerte de muchos (en específico quién entró a las barcas de emergencia y quién no).

Estructura del código.

- **Directorio**

En esta sección, se utiliza en un entorno de Google Colab, que es una plataforma de Google para ejecutar código en la nube utilizando Notebooks de Jupyter. Esta sección tiene el propósito de montar (conectar) la cuenta de Google Drive a Google Colab para poder acceder y trabajar con archivos almacenados en la unidad de Google Drive desde el entorno de Colab.

- **Limpieza de datos**
- **Extracción de datos**
- **Modelado**

En esta etapa, se presenta el proceso seguido en el proyecto, centrándose en la generación y comparación de modelos de aprendizaje máquina para abordar el reto planteado. En esta fase, se exploran diferentes tipos y configuraciones de modelos con el objetivo de seleccionar los

más efectivos, cuyas decisiones se detallarán en la documentación. Para esto se hizo una investigación acerca de los modelos de clasificación, se encontraron varios modelos que cumplen con las características de nuestros datos para predecir de manera binaria un resultado, en nuestro caso la predicción de la muerte de un individuo en el Titanic. Entre estos modelos, se seleccionaron: KNeighborsClassifier, Support Vector Classifier (SVC), LogisticRegression, DecisionTreeClassifier, GaussianNB, RandomForestClassifier, GradientBoostingClassifier, DecisionTreeClassifier, MLPClassifier.

Entre estos modelos, se tiene la intención de seleccionar sólo tres modelos, por lo que se realizó un proceso para seleccionar el modelo que será explicado en el documento, así mismo se documentó el proceso para la preparación de datos para cada modelo, así como la separación del dataset para entrenar y predecir. A continuación se describe el proceso de la selección e implementación de los modelos.

- **Librerías de modelos.**

La continuación del proyecto viene después de la limpieza de datos que se realizó en el entregable pasado, el siguiente paso fue investigar los modelos de predicción que nos interesan.

K-Neighbors Classifier: Utiliza el método de vecinos más cercanos para la clasificación. Las predicciones se basan en las clases de los ejemplos cercanos en el espacio de características. Requiere ajustar el número de vecinos y posiblemente otras métricas de distancia.

SVC (Support Vector Classifier): Un clasificador que encuentra un hiperplano de separación óptimo entre clases. Puede usar diferentes funciones de kernel para transformar los datos en un espacio de mayor dimensión. Es útil para problemas de clasificación lineal y no lineal.

Logistic Regression: Realiza la clasificación utilizando la función logística para modelar la probabilidad de pertenencia a una clase. Es efectivo para problemas de clasificación binaria y multiclase cuando las clases son linealmente separables.

Decision Tree Classifier: Construye un árbol de decisión que divide recursivamente el espacio de características en regiones más puras. Es fácilmente interpretable, pero puede ser propenso al sobreajuste si no se controla adecuadamente.

Gaussian NB: Implementa el clasificador Bayesiano Ingenuo Gaussiano, asumiendo que las características son independientes y distribuidas normalmente. A pesar de su simplicidad, puede funcionar sorprendentemente bien en muchos casos.

Random Forest Classifier: Crea múltiples árboles de decisión y combina sus predicciones para mejorar la robustez y precisión del modelo. Puede manejar características categóricas y numéricas, y es menos propenso al sobreajuste que un solo árbol.

Gradient Boosting Classifier: Construye una secuencia de modelos que corrigen los errores del modelo anterior. Es útil para problemas de clasificación y regresión, y tiende a producir modelos de alta calidad.

Decision Tree Classifier: Construye un árbol de decisión que divide recursivamente el espacio de características en regiones más puras. Es fácilmente interpretable, pero puede ser propenso al sobreajuste si no se controla adecuadamente.

Multi-layer Perceptron Classifier (MLP): Este clasificador implementa el algoritmo de aprendizaje profundo supervisado MLP que se entrena mediante el método de retropropagación (backpropagation) está compuesto por múltiples capas de neuronas: capas de entrada, capas ocultas y capas de salida, cada una conectada a través de conexiones ponderadas (Neural network models (supervised), s. f.). Se escogió este modelo para realizar la predicción de los decesos en el titanic debido a su popularidad actual en la sociedad, puesto que este modelo es la base de las nuevas tecnologías que estamos usando como la nueva

La librería cross_val_score se utiliza para generalizar los análisis con gráficas y predicciones, así como se importa accuracy_score y metrics para obtener información de nuestros modelos y predicciones

Después de seleccionar lo mejores modelos para cada modelo se realizaron diferentes pruebas utilizando diferentes hiperparámetros y utilizando dos bases de datos diferentes: una en donde eliminamos los pasajeros que contienen datos nulos y otra en donde a través del algoritmo de K-vecinos predecimos los valores nulos para no eliminar dichos datos. Para cada modelo se hicieron tres pruebas con diferentes híper parámetros para encontrar el mejor rendimiento.

Se determinaron pues las variables dependientes e independientes de la base de datos así como los datos de entrenamiento y los de testeo. Después se crea un array de modelos el cuál tiene la función de almacenar los modelos para utilizar un ciclo para preparar, entrenar y predecir cada uno de nuestros modelos que importamos, además se hace un análisis de precisión con la librería accuracy score. A continuación presentamos los scores de los modelos:

Accuracy de los modelos.

K-Neighbors Classifier	*0.643357*
Support Vector Classifier	*0.671329*
Logistic Regression - LR	*0.755245*
Decision Tree Classifier	*0.713287*
Gaussian NB-GNB	*0.755245*
Random Forest Classifie	*0.769231*
Gradient Boosting Classifier - GB	*0.783217*
Multi-Layer Perceptron Classifier - MLP	*0.762238*

Tabla X. Accuracy de los modelos.

Se puede apreciar que los modelos con mejor precisión son: Gradient Boosting, Multi-Layer Perceptron y Decision Tree Classifier, por lo que serán uno de los modelos que se utilizaran para hacer el análisis y predicción. Sin embargo, se harán algunos cambios en las parámetros para mejorar los resultados de la predicción. Una vez determinados los mejores modelos, se realizaron tres pruebas por modelo de aprendizaje para cada base de datos. A continuación se presentan los mejores resultados de precisión obtenidos en cada modelo:

	Eliminando los valores nulos	Preservando los valores nulos.
Gradient Boosting	0.82	0.78
Xtreme Gradient Boosting	0.84	0.80
Multi-Layer Percepton	0.76	0.79
Decision Tree	0.82	0.84

Tabla X. Comparativa de modelos con y sin valores nulos.

En específico, en Gradient Boosting tomamos los mejores hiperparámetros e hicimos un análisis de significancia de las variables como input, y nos entregó la siguiente gráfica, este nos dá a entender que no necesitamos la variable de embarking, ya que no parece tener relación con el resultado de decesos.

Vemos que los resultados son diferentes de acuerdo con la base de datos utilizada, sin embargo no se ve un patrón preciso que determine que una base es mejor que otra, solamente que en la mayoría de los modelos la base de datos en la que se preservan los datos con valores nulos predichos la precisión de los modelos es mayor. Así mismo se puede notar que el clasificador de árbol de decisión es el que tiene mayor precisión.

A lo largo de esta entrega, se ha llevado a cabo un proceso exhaustivo de selección, implementación y comparación de modelos de aprendizaje automático con el fin de lograr el objetivo de predecir la supervivencia de los pasajeros en el Titanic. En el transcurso de esta fase, se estudiaron una variedad de algoritmos de clasificación y se evaluaron en diferentes configuraciones para determinar cuales ofrecen el mejor rendimiento en materia de las métricas de evaluación.

Entre el conjunto de los modelos analizados, destacan dos enfoques particulares: el algoritmo Xtreme Gradient Boosting (XGBoost) y el uso de redes neuronales a través del Multi-Layer Perceptron Classifier (MLP). Ambos métodos demostraron resultados prometedores en términos de precisión, recall y F1-Score. El algoritmo XGBoost, que implementa un sistema de decisiones basado en árboles, se resaltó por su capacidad para manejar conjuntos de datos complejos y generar predicciones precisas. En contraste, la red neuronal MLP mostró una capacidad para aprender patrones en los datos y adaptarse a relaciones no lineales, lo que contribuyó a su alto rendimiento.

Es importante mencionar que, los mejores rendimientos encontrados en los modelos fueron extremadamente mejores cuando se trabajó con los datos que no contenían valores nulos. La eliminación de registros con datos faltantes permitió a los modelos centrarse en relaciones significativas en los datos disponibles y generar predicciones más precisas. Así mismo, se experimentó con diferentes valores de hiper-parámetros y técnicas de regularización. Las

iteraciones constantes permitieron identificar combinaciones óptimas que maximizaron la precisión y la generalización de los modelos.

- **Cambio de base de datos**
- **Firebase**

Se implementó la plataforma de Base de datos FireBase para utilizar una interfaz de celular que tenga una interacción con nuestro modelo de prediccion del reto

A continuación una explicación detallada del codigo para la implementación de la plataforma Fire-base

Este código está diseñado para interactuar con la plataforma Firebase y realizar predicciones sobre la supervivencia de un pasajero en el Titanic utilizando un modelo de árbol de decisiones.

Aquí hay una descripción detallada del código:

Importación de Librerías:

- **firebase_admin y credentials: Estas son librerías de Firebase que se utilizan para autenticar la aplicación en la plataforma.**
- **db: Esta es la biblioteca de Firebase que permite interactuar con la base de datos en tiempo real.**
- **requests: Se utiliza para hacer solicitudes HTTP a las bases de datos Firebase y obtener datos.**
- **pandas y train_test_split de sklearn.model_selection: Se utilizan para manipular y dividir los datos.**
- **DecisionTreeClassifier de sklearn.tree: Se utiliza para construir un modelo de árbol de decisiones.**

Credenciales y Configuración de Firebase:

- **cred: Contiene las credenciales necesarias para autenticar la aplicación en Firebase. Estas credenciales se obtienen del archivo JSON proporcionado.**
- **app: Inicializa la aplicación de Firebase con las credenciales.**

Obtención de Referencia a la Base de Datos:

- **ref: Obtiene una referencia a la base de datos en tiempo real de Firebase.**

Lectura de Datos:

- **data: Lee los datos de la base de datos.**
- **URLs de los Datos:**
- **Define URLs para obtener datos específicos de la base de datos.**

Preparación de Datos de Entrenamiento:

- **Lee un archivo CSV que contiene datos del Titanic y realiza varias operaciones de preprocesamiento, incluyendo el manejo de variables categóricas y la selección de características relevantes.**

División de Datos de Entrenamiento y Prueba:

- **Divide los datos en un conjunto de entrenamiento y un conjunto de prueba para evaluar el rendimiento del modelo.**

Construcción y Entrenamiento del Modelo de Árbol de Decisiones:

- **Crea y entrena un modelo de árbol de decisiones utilizando las características seleccionadas.**

Loop de Predicción:

- **Inicia un bucle infinito para hacer predicciones repetidamente.**

Obtención de Datos del Usuario:

- **Obtiene datos de usuario sobre la clase, edad, embarque y sexo del pasajero.**

Procesamiento de Datos del Usuario:

- **Convierte los datos a formatos adecuados y asigna un valor de cuota (Fare) en función de la clase del pasajero.**

Predicción y Publicación del Resultado:

- **Utiliza el modelo de árbol de decisiones entrenado para predecir si el pasajero sobrevivirá o no. Luego, publica el resultado en la base de datos.**

Manejo de Errores:

- **Captura y maneja cualquier error que pueda ocurrir durante el proceso de obtención y procesamiento de datos.**

A continuación se aprecia la implementación de la interfaz del titanic:

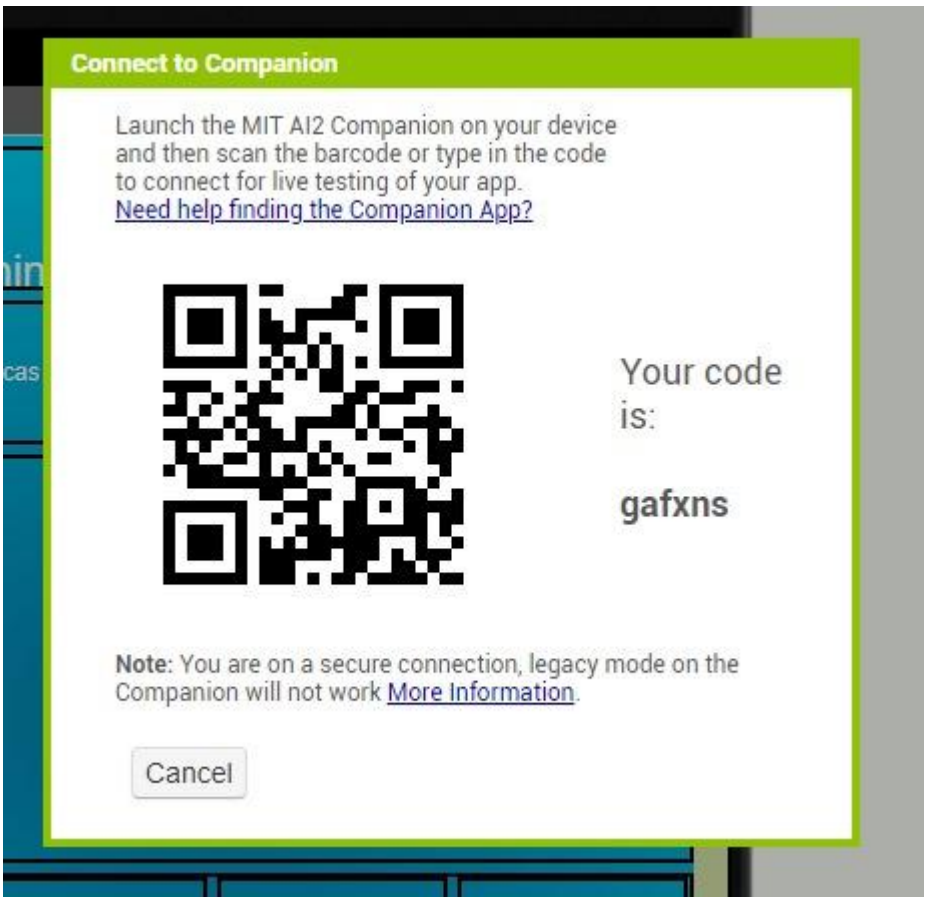

Imagen 1.1 Codigo QR para la aplicación

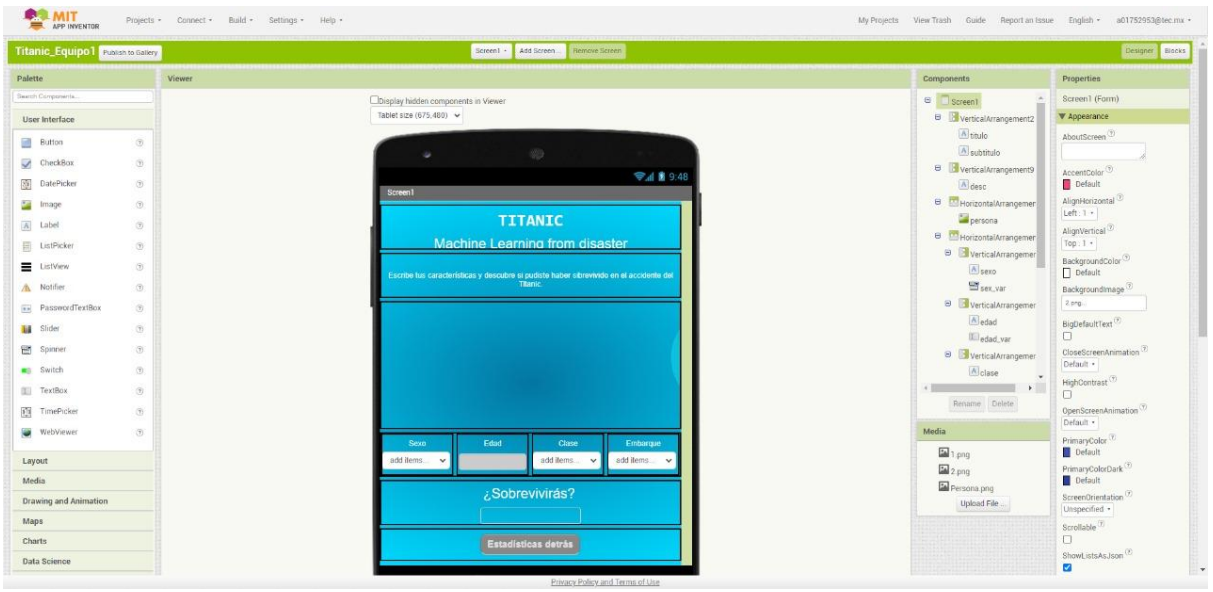

Imagen 1.2 Diseño para la aplicación en MIT

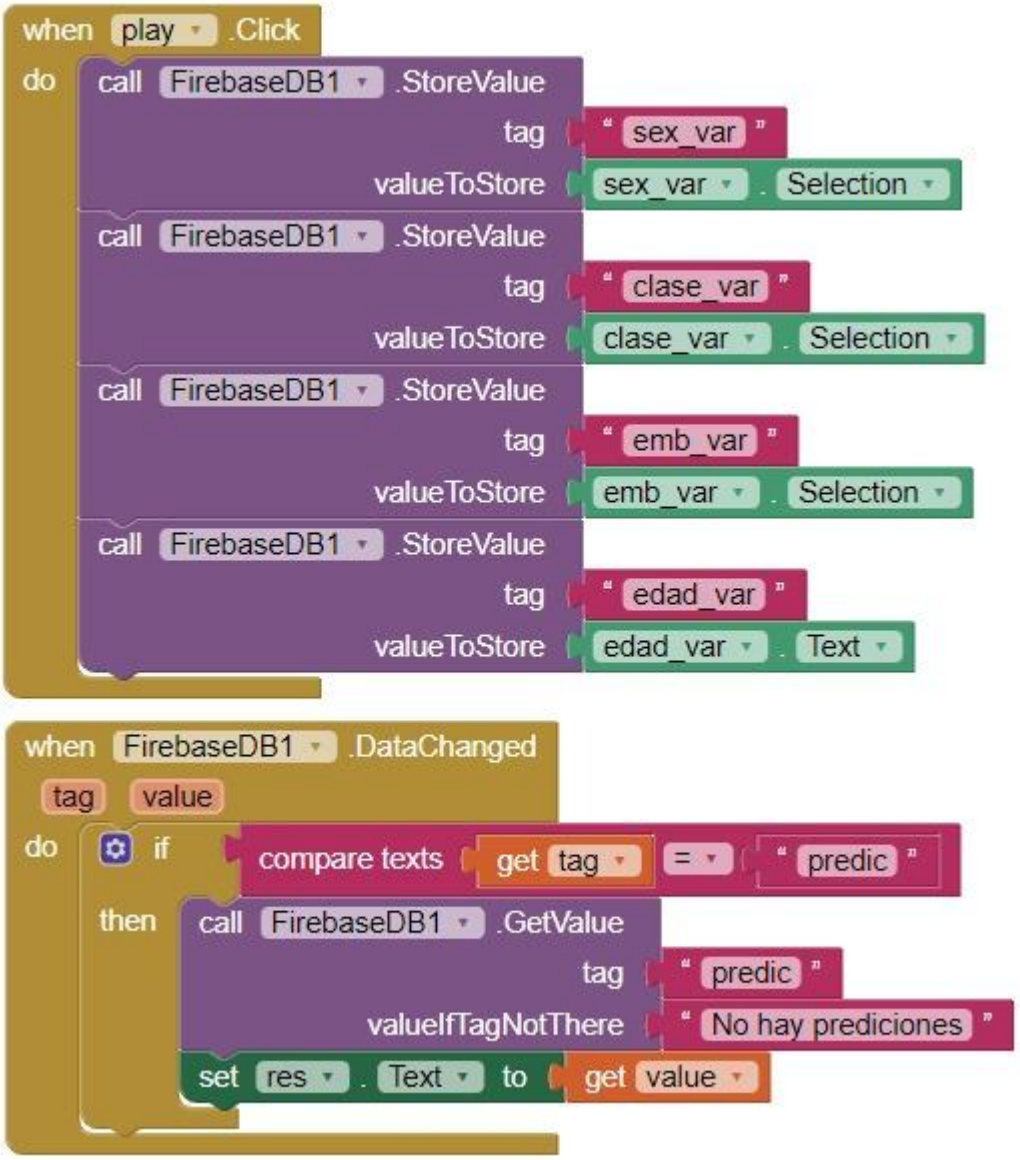

Imagen 1.4 Esquema de Aplicación móvil (Codigo en bloques)

Imagen 1.4 Aplicación móvil

Conclusiones :

Tras haber implementado la aplicación de MIT y la plataforma de FireBase, en el reto como el microcontrolador, el implementarlo para nuestro modelo de Machine Learning como solución del reto del Titanic fue bastante sencillo en ejecución, nos gustaría hacer algunas

modificaciones para hacer un poco más completa la experiencia y robusto el esquema en el que trabaja para que el usuario cuente con un mejor uso en la aplicación.