**INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA**

**TECNICATURA EN CIENCIA DE DATOS E INTELIGENCIA ARTIFICIAL**

**MÓDULO: CIENTÍFICO DE DATOS 2022**

**PRÁCTICA PROFESIONALIZANTE**

**PROFESOR: FACUNDO OLIVA CÚNEO**

**INFORME CRISP-DM**

**Proyecto App Información al Turista**

**‘Muéstrame una Imagen y te diré dónde ir’**

**GRUPO 18**

**Mariano Ledezma**

**Natalia Lamia**

**Viviana Farabollini**

CRISP-DM (Cross-Industry Standard Process for Data Mining) es una metodología ampliamente utilizada en proyectos de minería y análisis de datos. Su objetivo es proporcionar un enfoque estructurado y sistemático para gestionar estos proyectos, asegurando que se lleven a cabo de manera eficiente y efectiva. El presente informe se estructura siguiendo sus seis fases: Entendimiento del Negocio, Entendimiento de los Datos, Preparación de los Datos, Modelado, Evaluación y Despliegue.

1. **Entendimiento del Negocio**

**Objetivo del proyecto**

Se nos solicita desde una Agencia de Turismo el desarrollo de una aplicación para que, a partir de una imagen proporcionada por el usuario-turista, pueda sugerir un destino turístico en alguna de las regiones de Argentina. Las regiones consideradas son: Patagonia, Noroeste, Noreste, Cuyo, Buenos Aires y Córdoba.

**Requisitos del Negocio**

* Proporcionar sugerencias precisas y confiables basadas en la imagen cargada por el usuario.
* Aumentar el interés y la interacción de los usuarios con la aplicación y los destinos turísticos ofrecidos.
* Mejorar la experiencia del cliente y facilitar la planificación de viajes.

**Plan del Proyecto**

* Recolección y etiquetado de datos mediante scraping.
* Preparación de los datos para entrenamiento, validación y prueba.
* Construcción y entrenamiento de un modelo de redes neuronales convolucionales (CNN).
* Evaluación del modelo y optimización.
* Implementación del modelo en una aplicación móvil.
* Despliegue y mantenimiento del sistema.

1. **Entendimiento de los Datos**

**Recolección de datos**

* Se utilizó scraping a <https://www.shutterstock.com/> para recopilar imágenes representativas de las seis regiones turísticas de Argentina: Patagonia, Noroeste, Noreste, Cuyo, Buenos Aires y Córdoba
* Las imágenes fueron almacenadas en una estructura de carpetas bajo el nombre "***Turismo\_Argentina***", con subcarpetas para cada región.

**Análisis Exploratorio**

* Se descargaron y etiquetaron imágenes de sitios web utilizando BeautifulSoup y Requests.
* Se verificó la calidad de las imágenes y su relevancia para cada región.

3. **Preparación de los Datos**

**Selección de Datos**

* Las imágenes fueron organizadas en subcarpetas específicas para cada región.
* Se implementó un script para dividir las imágenes en conjuntos de entrenamiento, validación y prueba.

**Limpieza y Transformación de Datos**

* Las imágenes se normalizaron y redimensionaron a un tamaño de 150x150 píxeles.
* Se crearon generadores de datos utilizando *ImageData Generator* de *Keras* para realizar aumentación y escalado de las imágenes.

**4. Modelado**

**Selección de Técnicas de Modelado**

* Se optó por una red neuronal convolucional (CNN) debido a su efectividad en la clasificación de imágenes.

**Construcción del Modelo**

Se construye una Red Neuronal Convolucional (CNN) con el objetivo de clasificar imágenes en diferentes categorías, en este caso para identificar regiones turísticas de Argentina, optimizando su rendimiento mediante varias capas convolucionales, de pooling, densas y dropout, y utilizando el optimizador Adam para ajustar los pesos durante el entrenamiento.

**Detalles del Modelo**

1. **Definición del Modelo**

* *model = Sequential():* Crea un modelo secuencial, que es una pila lineal de capas.

1. **Capas Convolucionales y de Pooling**

* *model.add(Conv2D(64, kernel\_size=(3, 3), strides=(2, 2), padding='same', activation='relu', input\_shape=input\_shape)):* Añade una capa convolucional con 64 filtros, un tamaño de *kernel* de 3x3, un paso (*stride*) de 2x2, relleno '*same*' (para mantener las dimensiones de la salida), función de activación *ReLU* y la forma de entrada especificada (150, 150, 3) (imágenes de 150x150 píxeles con 3 canales de color).
* *model.add(Conv2D(128, kernel\_size=(3, 3), strides=(2, 2), padding='same', activation='relu')):* Añade una segunda capa convolucional con 128 filtros, y características similares a la primera capa.
* *model.add(MaxPole 2D()):* Añade una capa de max pooling para reducir la dimensionalidad espacial (dimensiones de las imágenes) tomando el máximo valor en una ventana de 2x2.
* *model.add(Conv2D(256, kernel\_size=(3, 3), strides=(2, 2), padding='same', activation='relu')):* Añade una tercera capa convolucional con 256 filtros.
* *model.add(Conv2D(512, kernel\_size=(3, 3), strides=(2, 2), padding='same', activation='relu')):* Añade una cuarta capa convolucional con 512 filtros.
* *model.add(MaxPole 2D()):* Añade otra capa de *max pooling*.

1. **Capa de Aplanamiento (Flattening)**

* *model.add(Flatten()):* Aplana las salidas de la última capa de pooling en un vector de una dimensión, preparándolas para las capas densas (fully connected).

1. **Capas Densas (Fully Connected)**

* *model.add(Dropout(0.2)):* Añade una capa de dropout con una tasa del 20% para prevenir el sobreajuste, apagando aleatoriamente el 20% de las neuronas en cada actualización durante el entrenamiento.
* *model.add(Dense(1048, activation='relu')):* Añade una capa densa con 1048 neuronas y activación ReLU.
* *model.add(Dropout(0.2)):* Añade otra capa de dropout con la misma tasa.
* *model.add(Dense(len(class\_names), activation='softmax')):* Añade la capa de salida con un número de neuronas igual al número de clases (categorías de imágenes) y activación softmax para obtener probabilidades de clasificación.

**5. Compilación del Modelo**

* *model.compile(optimizer=Adam(), loss='categorical\_crossentropy', metrics=['accuracy']):* Compila el modelo usando el optimizador Adam, la función de pérdida categorical\_crossentropy (adecuada para clasificación multiclase) y la métrica de precisión (accuracy).

1. **Resumen del Modelo**

* *model.summary():* Imprime un resumen del modelo, mostrando las capas, sus formas de salida, y el número de parámetros entrenables.

**Entrenamiento del Modelo**

* El modelo se entrenó durante 30 épocas con datos de entrenamiento y validación.
* Se guardaron los pesos del modelo para su uso futuro.

**5. Evaluación**

**Evaluación del Modelo**

* Se utilizó una matriz de confusión para evaluar el rendimiento del modelo en el conjunto de validación.
* Los resultados mostraron una precisión alta en la clasificación de las imágenes por región con la utilización de un modelo proporcionado por Kaggle <https://www.kaggle.com/datasets/puneet6060/intel-image-classification>. , no así con el DB *Turismo\_Argentina* construido por el grupo utilizando scraping. Por lo que se decide modificar el proyecto en relación a las imágenes utilizadas para la clasificación y predicción.

**Optimización**

* Se ajustaron los hiperparámetros y se aplicaron técnicas de aumentación de datos para mejorar la precisión del modelo.

**6. Despliegue**

**Implementación**

* Se proyecta integrar el modelo en una aplicación móvil que permita a los usuarios turistas cargar imágenes y obtener una sugerencia de destino turístico por voz.
* Se desarrolló una función para convertir los resultados de la predicción en audio, proporcionando una experiencia interactiva y accesible para los usuarios.

**Mantenimiento**

* Se estableció la necesidad de un proceso continuo de monitoreo y actualización del modelo para mantener su precisión y relevancia a lo largo del tiempo por parte del equipo de técnicos en ciencia de datos e inteligencia artificial.

**OBSERVACIÓN**

Se adjunta el código utilizado en los archivos: ‘ScrapingTurismo’, ‘Turismo\_Argentina’, ‘III-ClasificaciónFotosTurismo\_ConRtaVoz’ y ‘ModeloAppTurismo’ (Este último en desarrollo).