PORTADA

Nombre Alumno / DNI	**FRANCISCO JAVIER ROIG MINGUELL**
Título del Programa	**CIBERSECURITY AND HACKING**
Nº Unidad y Título	**3ºPD CYBER SECURITY / COMPUTER SCIENCE / DATA SCIENCE** **UNIT 25. APPLIED MACHINE LEARNING**
Año académico	**2025-2026**
Profesor de la unidad	**Rabindranath Andujar**
Título del Assignment	**AB ENTREGA FINAL**
Día de emisión	**25-09-2025**
Día de entrega	**20-01-2026**
Nombre IV y fecha	
Declaración del estudiante	**Certifico que la presentación del assignment es completamente mi propio trabajo y entiendo completamente las consecuencias del plagio. Entiendo que hacer una declaración falsa es una forma de mala práctica.** **Fecha: 20-01-2026** **Firma del alumno: FRANCISCO JAVIER ROIG MINGUELL** 

Plagio

El plagio es una forma particular de hacer trampa. El plagio debe evitarse a toda costa y los alumnos que infrinjan las reglas, aunque sea inocentemente, pueden ser sancionados. Es su responsabilidad asegurarse de comprender las prácticas de referencia correctas. Como alumno de nivel universitario, se espera que utilice las referencias adecuadas en todo momento y mantenga notas cuidadosamente detalladas de todas

sus fuentes de materiales para el material que ha utilizado en su trabajo, incluido cualquier material descargado de Internet. Consulte al profesor de la unidad correspondiente o al tutor del curso si necesita más consejos.

Tabla de contenido

Adjunto a continuación los entregables correspondientes al proyecto final y los ejercicios intermedios solicitados:

1. Archivos Adjuntos y Enlaces:

- **Red Neuronal en Excel:** Se adjunta el archivo .xlsx donde se simula el proceso de entrenamiento (forward propagation y ajuste de pesos).
- **Ejercicios de Kaggle:** Se adjuntan los notebooks completados del curso de *Intro to Machine Learning* (Decision Trees, Random Forests, Model Validation) basados en el dataset de predicción de precios de viviendas.
- **Enlace al Proyecto OpenCV (GitHub):** https://github.com/franciscojavierroigminguell/Crear-aplicaci-n-con-OpenCV
- **Elección de Dataset:** Dogs vs. Cats (Kaggle). https://www.kaggle.com/c/dogs-vs-cats/data?authuser=0

A. Definición del Problema (Business Understanding)

- **Contexto:** El proyecto aborda la clasificación automática de imágenes. En el volumen actual de datos digitales, diferenciar contenido visual automáticamente es vital para motores de búsqueda y organización de archivos.
- **Hipótesis:** Se parte de la hipótesis de que un modelo de Visión por Computador puede detectar patrones de texturas y formas (orejas, hocico) para diferenciar un perro de un gato con mayor precisión que un humano promedio en velocidad.
- **Valor:** El usuario final (plataformas de contenido o apps de veterinaria) obtiene una herramienta que automatiza el etiquetado, ahorrando horas de trabajo manual.

B. Fundamentación Teórica

Apoyándome en el ejercicio realizado en **Excel**, el modelo implementado funciona bajo la misma lógica matemática:

Justificación de Algoritmos: El Modelo de Red Neuronal

La elección de una **Red Neuronal Artificial** (o Perceptrón) se justifica por su capacidad iterativa para ajustar parámetros internos y minimizar el error de predicción. A diferencia de los sistemas basados en reglas fijas, este algoritmo "aprende" de los datos históricos.

1. Los Pesos (*Weights*): La Memoria del Modelo

Los pesos representan la importancia de cada variable de entrada (input 1, input 2, input 3) para determinar la salida. En el Excel, observamos que estos no son estáticos, sino dinámicos.

- **Evidencia en el Excel:**
  - En la **Fila 1**, el weight 1 comienza con un valor de **0.71**.
  - Tras varias iteraciones de aprendizaje, hacia la **Fila 11**, el weight 1 ha subido a **0.918**.
- **Interpretación:** El algoritmo ha "aprendido" que el input 1 es fuertemente predictivo para el resultado positivo (1), por lo que le asigna progresivamente más "peso" o importancia.

2. La Activación y el Sesgo (*Bias/Weighted Sum*)

En tu hoja de cálculo, la columna etiquetada como BIAS (que matemáticamente en tu hoja actúa como la **Suma Ponderada** o z) representa la señal total que recibe la neurona antes de tomar una decisión.

- **Evidencia en el Excel:**
  - Observamos la columna OUTPUT (la predicción). En la primera fila, con un valor de suma ponderada (BIAS en la hoja) de **0.71**, la red predice un **0.67** (67% de probabilidad).
  - Esto demuestra el uso de una **Función de Activación** (probablemente Sigmoide), que transforma números brutos en probabilidades manejables entre 0 y 1.

3. El Aprendizaje mediante el Error (*Backpropagation*)

La parte más crítica del algoritmo es cómo corrige sus errores. Las columnas W1 CHANGE, W2 CHANGE, etc., representan el **Gradiente Descendente**. El modelo calcula la diferencia entre lo que predijo y la realidad, y ajusta los pesos en consecuencia.

- **Evidencia en el Excel:**
  - **Caso de Error:** En la **Fila 2**, la red predice aprox. **0.83** pero el objetivo real es **1**. Como hay un error ("falta" llegar a 1), el sistema genera un ajuste positivo:
    - W1 CHANGE: **0.023**
    - W3 CHANGE: **0.017**

    - **Caso de Acierto:** En la **Fila 3** (entrada 0,1,1 -> salida 0), si la red predice correctamente o la entrada es 0, el ajuste es nulo. Vemos que W1 CHANGE es **0**.
- **Justificación:** Esto demuestra que el algoritmo es **eficiente**: solo gasta recursos computacionales ajustando sus parámetros cuando comete un error, refinando su precisión con cada ejemplo procesado.
- "Nos hemos decantado por este algoritmo de la Red Neuronal ya que, tal y como queda reflejado en la simulación con Excel, el modelo presenta capacidad plástica de ir modificando sus pesos (w) conforme al error que se observa. Esto permite que el modelo evolucione desde una configuración ya inicial aleatoria (pesos de 0.71) hacia una configuración optimizada (pesos de 0.94) que garantiza una mejor generalización frente a nuevos datos."

C. Ingeniería y Análisis de Datos

Aplicando lo aprendido en los ejercicios de **Kaggle** sobre limpieza y validación:

En esta sección detallamos el ciclo de vida del dato, desde su exploración inicial hasta la evaluación crítica de los modelos predictivos generados (Decision Trees y Random Forests).

1. EDA: Exploración Visual de los Datos

Para comprender la naturaleza del problema, analizamos el dataset "Home Data for ML".

- Descripción Estadística: Utilizamos describe() para identificar la distribución de las variables numéricas. Esto nos permitió detectar escalas dispares (ej. LotArea vs YearBuilt) y calcular métricas base, como el tamaño promedio del lote.
- Análisis del Target (SalePrice):
    - *(Recomendación: Aquí deberías incluir un histograma de la variable objetivo 'SalePrice'. Si la distribución tiene una "cola larga" hacia la derecha, menciónalo).*
- Relación Variables-Target:
    - *(Recomendación: Incluye un Scatter Plot de 'LotArea' vs 'SalePrice' o un Mapa de Calor de correlaciones. Esto justifica por qué elegiste las features que elegiste).*

2. Preprocesamiento y Selección de Features

Basándonos en el EDA, realizamos una selección manual de características predictivas.

- Selección de Features: Se seleccionaron 7 variables clave que intuitivamente impactan el precio de la vivienda: LotArea, YearBuilt, 1stFlrSF, 2ndFlrSF, FullBath, BedroomAbvGr, TotRmsAbvGrd.
- Partición de Datos: Para evitar sesgos y medir la generalización, dividimos el dataset usando train_test_split con un random_state=1 para garantizar reproducibilidad.
    - Conjunto de Entrenamiento: Utilizado para ajustar los patrones del árbol.
    - Conjunto de Validación: Utilizado estrictamente para calcular el Error Absoluto Medio (MAE).

3. Resultados y Discusión

Evaluamos el rendimiento utilizando la métrica MAE (Mean Absolute Error), que ofrece una interpretación directa en dólares del error promedio por casa.

Iteración 1: Árbol de Decisión Base

- Entrenamos un DecisionTreeRegressor sin limitar la profundidad.
- Resultado: El modelo obtuvo un MAE de validación de aproximadamente $29,653.
- Observación: Al revisar las predicciones "in-sample" (con los datos de entrenamiento), el error era sospechosamente bajo, indicando un posible sobreajuste inicial.

Iteración 2: Ajuste de Hiperparámetros (Tuning)

- Para combatir el *overfitting*, experimentamos con el hiperparámetro max_leaf_nodes. Probamos valores en el rango: $[5, 25, 50, 100, 250, 500]$.
- Hallazgo: Identificamos que el punto óptimo se encuentra alrededor de 100 nodos.
    - *Pocos nodos (5)* causaban Underfitting (el modelo era demasiado simple para capturar patrones).
    - *Muchos nodos (500)* causaban Overfitting (el modelo memorizaba ruido de los datos de entrenamiento).

Iteración 3: Random Forest (Modelo Final)

- Evolucionamos hacia un modelo de ensamble RandomForestRegressor para mejorar la robustez.
- Mejora: El Random Forest redujo significativamente el error en comparación con el mejor Árbol de Decisión simple.
- Métrica Final: *(Inserta aquí tu MAE final del Random Forest, usualmente ronda los $22k-$24k).*

4. Análisis de Errores y Conclusiones Críticas

A pesar de la mejora con Random Forest, el modelo mantiene un margen de error.

- Análisis del Error: El error promedio de ~$20,000 - $30,000 sigue siendo significativo para una transacción inmobiliaria. Esto sugiere que las 7 variables seleccionadas no contienen toda la información necesaria (falta, por ejemplo, la ubicación o condición del vecindario).
- Fenómeno de Overfitting/Underfitting: Confirmamos la hipótesis de la curva en forma de "U" del error. Al aumentar la complejidad del modelo (max_leaf_nodes), el error de validación baja hasta cierto punto y luego vuelve a subir, demostrando que un modelo más complejo no siempre es mejor.
    - Limitación: El modelo actual asume que los datos históricos de Iowa (algunos muy antiguos) siguen siendo representativos para precios actuales, lo cual es una limitación temporal importante.

D. Arquitectura y Despliegue

Esta sección detalla cómo nuestra aplicación de visión artificial, construida con OpenCV y alojada en Vercel, interactúa con el usuario y gestiona el flujo de información.

1. Diagrama del Flujo de Datos

Como no puedo dibujar directamente en tu documento, aquí tienes una descripción textual estructurada que puedes convertir en un diagrama de bloques (usando herramientas como Draw.io, Canva o PowerPoint):
Paso 1: Interfaz de Usuario (Frontend)
Actor: Usuario.
Acción: Carga una imagen, activa la webcam o interactúa con un widget.
Tecnología: Navegador Web (HTML/JS o Streamlit UI).
Paso 2: Transmisión (Petición HTTP)
Acción: Los datos (imagen en base64 o archivo binario) viajan al servidor.
Protocolo: POST Request (/api/detectar).
Paso 3: Procesamiento (Backend en Vercel)
Motor: Python Serverless Function (Flask/FastAPI).
Lógica: OpenCV recibe la imagen → Preprocesamiento (Grayscale) → Inferencia del Modelo (Haar Cascade/DNN) → Extracción de resultado (ej. "3 caras").
Paso 4: Respuesta y Almacenamiento
Respuesta: JSON al Frontend ({ "caras": 3, "status": "ok" }).
Persistencia (Feedback): Envío asíncrono del metadato a Base de Datos Externa (Supabase/Firebase/Google Sheets).

2. Comunicación Cliente-Servidor

En nuestra arquitectura desplegada en Vercel, la comunicación se realiza mediante una arquitectura RESTful.
Captura de Datos: El frontend (la página web que ve el usuario) utiliza JavaScript (o componentes de Streamlit) para capturar el *input* visual. Si es una imagen estática, se convierte a formato binario o una cadena Base64.
El "Handshake" (Petición): Cuando el usuario pulsa "Procesar", el cliente emite una solicitud POST al endpoint del servidor (por ejemplo, misitio.vercel.app/api/procesar). Este método es necesario porque estamos enviando un cuerpo de datos (la imagen) que es demasiado grande para una petición GET.
Recepción en Vercel: Las *Serverless Functions* de Vercel despiertan una instancia de Python, leen el cuerpo de la solicitud, decodifican la imagen para que OpenCV la entienda (transformándola en un array de NumPy), ejecutan el algoritmo y devuelven la respuesta en formato JSON ligero, minimizando la latencia.

3. Ciclo de Feedback y Estrategia de Re-entrenamiento

Para que el modelo no sea estático y pueda mejorar, hemos diseñado un bucle de retroalimentación (Data Flywheel):

¿Cómo se guardan los nuevos datos?

Una vez que el modelo realiza una predicción (ej. cuenta 3 caras), la aplicación no descarta esa información. Se realiza una llamada secundaria a una base de datos en la nube (como Supabase o MongoDB Atlas).

Qué guardamos: Guardamos el resultado de la inferencia (número de detecciones), la fecha/hora, y opcionalmente (si la privacidad lo permite) la imagen original o los *embeddings* (características numéricas) de la imagen.
Validación del Usuario: Idealmente, la interfaz permite al usuario confirmar si el resultado fue correcto (ej. un botón "¿Es correcto? Sí/No"). Este "voto" se guarda junto con el registro.

¿Cómo se usarían para re-entrenar el modelo?
Esta base de datos acumulativa se convierte en nuestro nuevo *dataset*.

Auditoría: Periódicamente, revisamos las imágenes donde el usuario marcó "No es correcto" (Falsos Positivos/Negativos).
Etiquetado: Estas imágenes difíciles se etiquetan manualmente con la respuesta correcta.
Fine-Tuning: Si usamos un modelo de Deep Learning, usamos estas nuevas imágenes para re-entrenar las últimas capas del modelo, haciéndolo más robusto ante casos que fallaron previamente en el entorno real (iluminación real, ángulos extraños, etc.).

Bibliografia:

Los puntos son parte de los ejercicios de clase hecho por el profesor
https://github.com/franciscojavierroigminguell/ABFINAL-UNIT-25-APLIED-MACHINE-.git?authuser=0

https://medium.com/@epiclabs.io/machine-learning-for-video-transcoding-verification-in-livepeers-ecosystem-i-e020ed11d6b4
https://classroom.google.com/c/ODAwNjM1MTIwMzU3/m/ODE3OTM5MjE0ODQ1/details
https://classroom.google.com/c/ODAwNjM1MTIwMzU3/m/ODE4MTYxMjAwNjM3/details
https://www.kaggle.com/learn/intro-to-game-ai-and-reinforcement-learning
https://classroom.google.com/c/ODAwNjM1MTIwMzU3/m/ODE3MjYyMjUwMDAw/details
https://learnopencv.com/image-quality-assessment-brisque/
https://docs.opencv.org/3.4/d5/d10/tutorial_js_root.html
https://poloclub.github.io/transformer-explainer/
https://m.youtube.com/watch?v=QK4AHZTVf68