UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

TEMA:

Evaluacion: Segundo Parcial

PRESENTADO POR:

Samonte Mercado Jeremy

GRUPO:

9A

MATERIA:

Extracción y conocimiento de bases de datos

PROFESOR:

Florencio López Cruz

Tijuana, Baja California, 12 de noviembre del 2025

I. Resumen del proyecto

Objetivo: Desarrollar un modelo de Machine Learning capaz de predecir si un paciente presenta alto o bajo riesgo de cáncer de pulmón, a partir de síntomas y hábitos registrados en el dataset survey_lung_cancer.csv.

Tecnologías: Python, Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn, Flask, y Joblib.

Resultados esperados:

- Pipeline guardado (best_pipeline.joblib y lung_cancer_model.pkl) con preprocesador + modelo.
- Aplicación web Flask que recibe inputs del usuario y retorna predicción + probabilidad.
- Carpeta reports/ con gráficos EDA y métricas.

II. Dataset y observaciones iniciales

Archivo: survey_lung_cancer.csv.

Observación clave: Contiene columnas con información sobre edad, género, síntomas (tos, dolor en el pecho, dificultad al respirar, etc.) y hábitos (fumar, consumo de alcohol, etc.), junto con la variable objetivo LUNG_CANCER.

Observación importante: El dataset utiliza valores 1 y 2 para indicar respuestas binarias. En este caso:

- 1 = No
- 2 = Sí

Por ello, en los scripts se realizó una conversión de valores y codificación para normalizar los datos antes de entrenar el modelo.

III. Ciclo de Machine Learning

3.1 Carga y EDA (Exploratory Data Analysis)

Script: generate_reports.py

Este script realiza un análisis exploratorio automatizado y genera gráficos dentro de la carpeta reports/.

Acciones que se realizaron:

1. Carga del dataset con pd.read_csv().
2. Generación de estadísticas descriptivas (describe() → dataset_description.csv).
3. Creación de gráficas con Matplotlib y Seaborn:
    - Distribución de edades (age_distribution.png)
    - Distribución por género (gender_distribution.png)
    - Conteo de fumadores (smoking_status.png)
    - Mapa de correlaciones (correlation_matrix.png)
    - Distribución de casos positivos/negativos (lung_cancer_distribution.png)
4. Exportación automática de todas las imágenes a reports/.

```
PS C:\Users\Jeres\repositorio\BD_Jeremy_9A\lung_cancer_project> python generate_reports.py
Cargando dataset...
Dataset cargado correctamente con 309 filas y 16 columnas.
Archivo 'dataset_description.csv' generado.
Gráficas generadas correctamente en la carpeta 'reports/'.
PS C:\Users\Jeres\repositorio\BD_Jeremy_9A\lung_cancer_project>
```

Justificación: El análisis exploratorio permitió comprender mejor las relaciones entre variables y detectar correlaciones relevantes entre factores de riesgo y la aparición del cáncer de pulmón.

3.2 Preprocesamiento y transformación de datos

Script principal: train.py

Pasos realizados:

1. Limpieza de nombres de columnas (eliminación de espacios).
2. Conversión del objetivo LUNG_CANCER de valores “YES/NO” a 1/0.
3. Detección de variables numéricas y categóricas.
4. Imputación de valores faltantes:
    - Numéricos -> mediana (SimpleImputer(strategy='median'))
    - Categóricos -> valor más frecuente (SimpleImputer(strategy='most_frequent'))
5. Codificación categórica mediante OneHotEncoder.
6. Escalado de variables numéricas con StandardScaler.
7. Integración de todo en un ColumnTransformer y Pipeline.

Al entrenar el modelo obtenemos lo siguiente:

- Descripción estadística de las características

```
--- Descripción estadística ---
                       count unique  top freq       mean       std   min   25%   50%   75%   max
GENDER                   309      2    M  162        NaN       NaN   NaN   NaN   NaN   NaN   NaN
AGE                    309.0    NaN  NaN  NaN  62.673139  8.210301  21.0  57.0  62.0  69.0  87.0
SMOKING                309.0    NaN  NaN  NaN   1.563107  0.496806   1.0   1.0   2.0   2.0   2.0
YELLOW_FINGERS         309.0    NaN  NaN  NaN   1.569579  0.495938   1.0   1.0   2.0   2.0   2.0
ANXIETY                309.0    NaN  NaN  NaN   1.498382  0.500808   1.0   1.0   1.0   2.0   2.0
PEER_PRESSURE          309.0    NaN  NaN  NaN   1.501618  0.500808   1.0   1.0   2.0   2.0   2.0
CHRONIC_DISEASE        309.0    NaN  NaN  NaN   1.504854  0.500787   1.0   1.0   2.0   2.0   2.0
FATIGUE                309.0    NaN  NaN  NaN   1.673139  0.469827   1.0   1.0   2.0   2.0   2.0
ALLERGY                309.0    NaN  NaN  NaN   1.556634  0.497588   1.0   1.0   2.0   2.0   2.0
WHEEZING               309.0    NaN  NaN  NaN   1.556634  0.497588   1.0   1.0   2.0   2.0   2.0
ALCOHOL_CONSUMING      309.0    NaN  NaN  NaN   1.556634  0.497588   1.0   1.0   2.0   2.0   2.0
COUGHING               309.0    NaN  NaN  NaN   1.579288  0.494474   1.0   1.0   2.0   2.0   2.0
SHORTNESS_OF_BREATH    309.0    NaN  NaN  NaN   1.640777  0.480551   1.0   1.0   2.0   2.0   2.0
SWALLOWING_DIFFICULTY  309.0    NaN  NaN  NaN   1.469256  0.499863   1.0   1.0   1.0   2.0   2.0
CHEST_PAIN             309.0    NaN  NaN  NaN   1.556634  0.497588   1.0   1.0   2.0   2.0   2.0
LUNG_CANCER            309.0    NaN  NaN  NaN   0.873786  0.332629   0.0   1.0   1.0   1.0   1.0
```

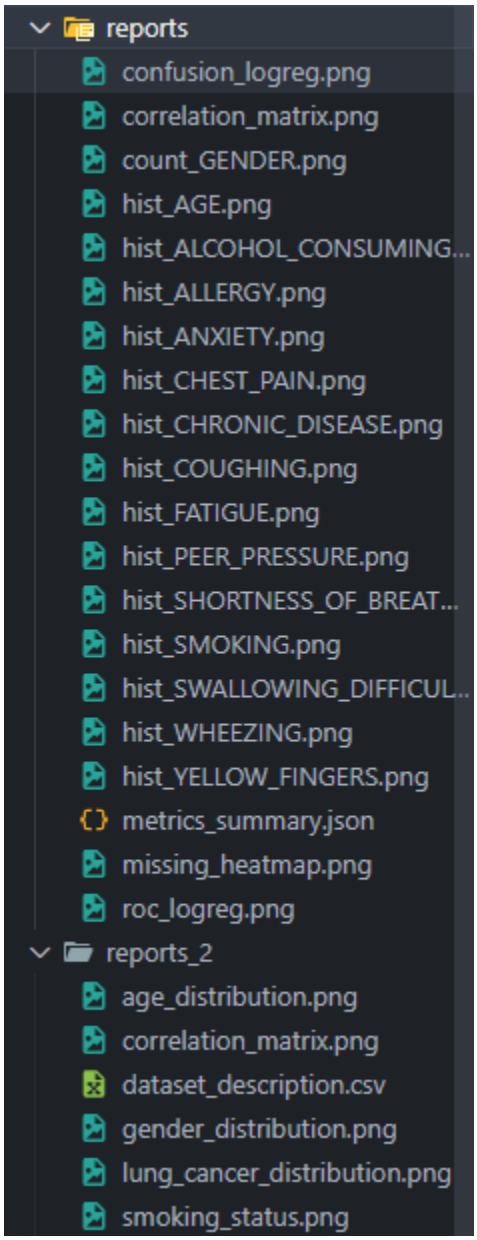
- Comentarios sobre el entrenamiento de modelos, su accuracity, recall, f1 y roc

```
Entrenando logreg...
C:\Users\Jeres\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\sklearn\preprocessing\_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
logreg -> acc: 0.9032, recall: 0.9444, f1: 0.9444, roc_auc: 0.9467592592592593

Entrenando knn...
C:\Users\Jeres\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\sklearn\preprocessing\_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
knn -> acc: 0.8871, recall: 0.9444, f1: 0.9358, roc_auc: 0.9039351851851851

Entrenando rf...
C:\Users\Jeres\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\sklearn\preprocessing\_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
rf -> acc: 0.8871, recall: 0.9259, f1: 0.9346, roc_auc: 0.9444444444444444
```

- Se realiza comparación de modelos, y se selecciona el mejor de ellos. Al seleccionarlos se guarda el pipeline en un archivo.joblib y de igual manera se crean graficas en el folder de /reports

```
Comparación de modelos:
logreg -> accuracy: 0.9032 f1: 0.9444 recall: 0.9444 roc_auc: 0.9467592592592593
knn -> accuracy: 0.8871 f1: 0.9358 recall: 0.9444 roc_auc: 0.9039351851851851
rf -> accuracy: 0.8871 f1: 0.9346 recall: 0.9259 roc_auc: 0.9444444444444444
rf_grid -> accuracy: 0.8548 f1: 0.9189 recall: 0.9444 roc_auc: 0.9490740740740741

Mejor modelo seleccionado: logreg
Pipeline guardado en: models/best_pipeline.joblib
Reportes y gráficas guardadas en reports
```

- Reportes en png donde se muestra la distribución de características numéricas, categóricas y correlaciones entre características



3.3 Selección y entrenamiento de modelos

Modelos entrenados:

- Regresión Logística
- K-Nearest Neighbors (KNN)
- Random Forest

Procedimiento:

- División de datos en entrenamiento y prueba (train_test_split, 80/20).
- Entrenamiento de cada modelo en un Pipeline completo.
- Evaluación por métricas: Accuracy, Recall, F1, y ROC-AUC.
- Búsqueda de hiperparámetros con GridSearchCV para optimizar el Random Forest.

Criterio de selección:

El mejor modelo se seleccionó por Regresion Logistica/F1, al priorizar el balance entre precisión y sensibilidad.

Salida generada:

- models/best_pipeline.joblib
- Métricas resumidas en reports/metrics_summary.json
- Matriz de confusión
- Curva ROC
- Histogramas

3.4 Evaluacion de modelos

Script adicional: train_save_model.py

Se entrenó un modelo base con Regresión Logística balanceada (class_weight='balanced') para comparación.

Resultados:

- Reporte de clasificación impreso en consola.
- Archivo del modelo exportado como lung_cancer_model.pkl (en flask se utiliza el .joblib)

```
PS C:\Users\Jeres\repositorio\BD_Jeremy_9A\lung_cancer_project> python train_save_model.py
Matriz de confusión:
 [[ 1  1]
 [ 6 54]]

Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.14      0.50      0.22         2
           1       0.98      0.90      0.94        60

    accuracy                           0.89        62
   macro avg       0.56      0.70      0.58        62
weighted avg       0.95      0.89      0.92        62

Modelo guardado correctamente como lung_cancer_model.pkl
```

IV. Despliegue web (Flask)

Archivo: app.py

Flujo de funcionamiento:

1. Carga el modelo models/best_pipeline.joblib al iniciar.
2. Muestra un formulario HTML (templates/index.html) donde el usuario ingresa:
    - Género, edad, y síntomas (tos, dolor de pecho, etc.)
3. Envía los datos a la ruta /predict.
4. El servidor procesa la información, genera la predicción y muestra el resultado en result.html.

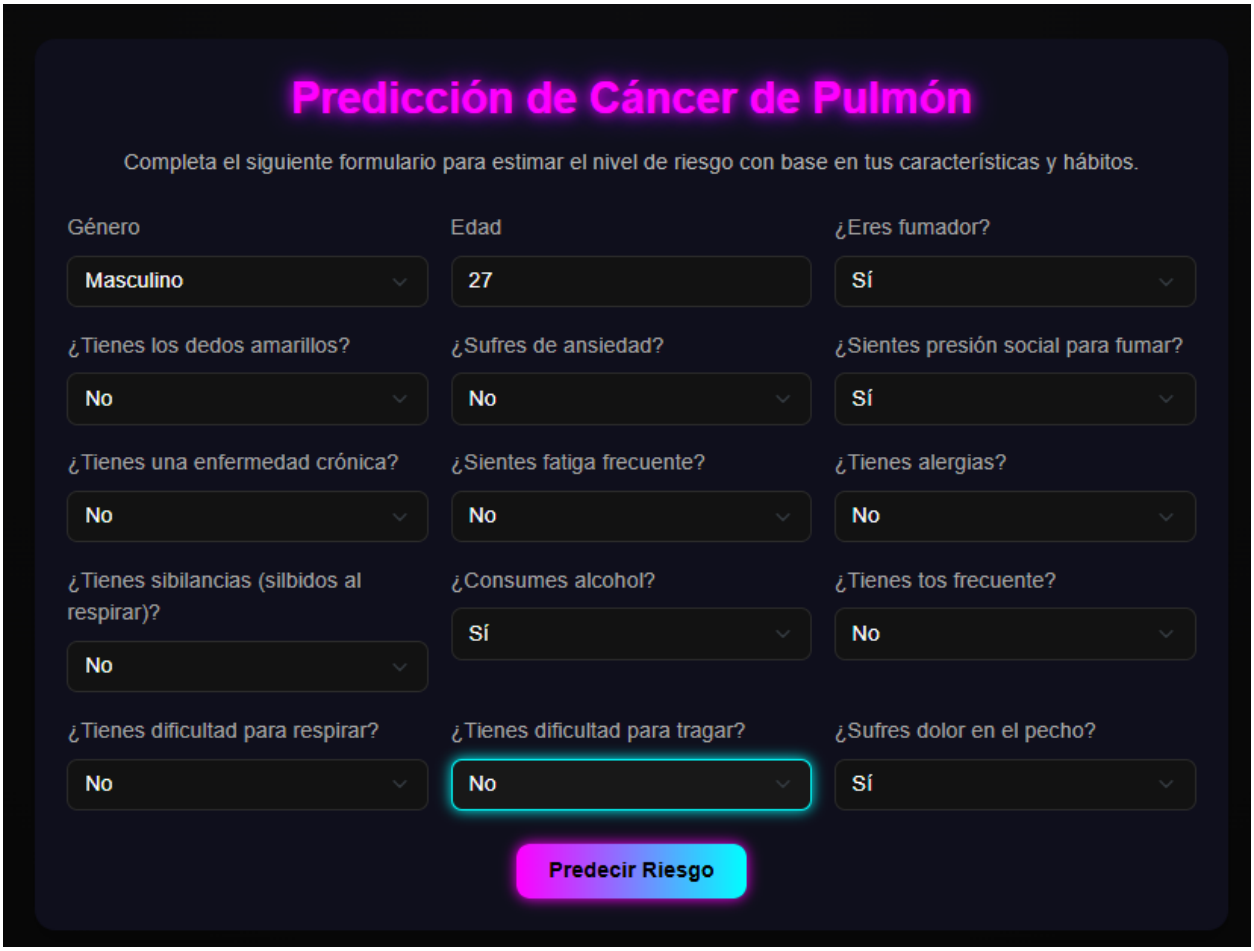
Salida de la aplicación:

- Predicción del modelo (“Alto riesgo” o “Bajo riesgo”)
- Porcentaje de probabilidad estimado.

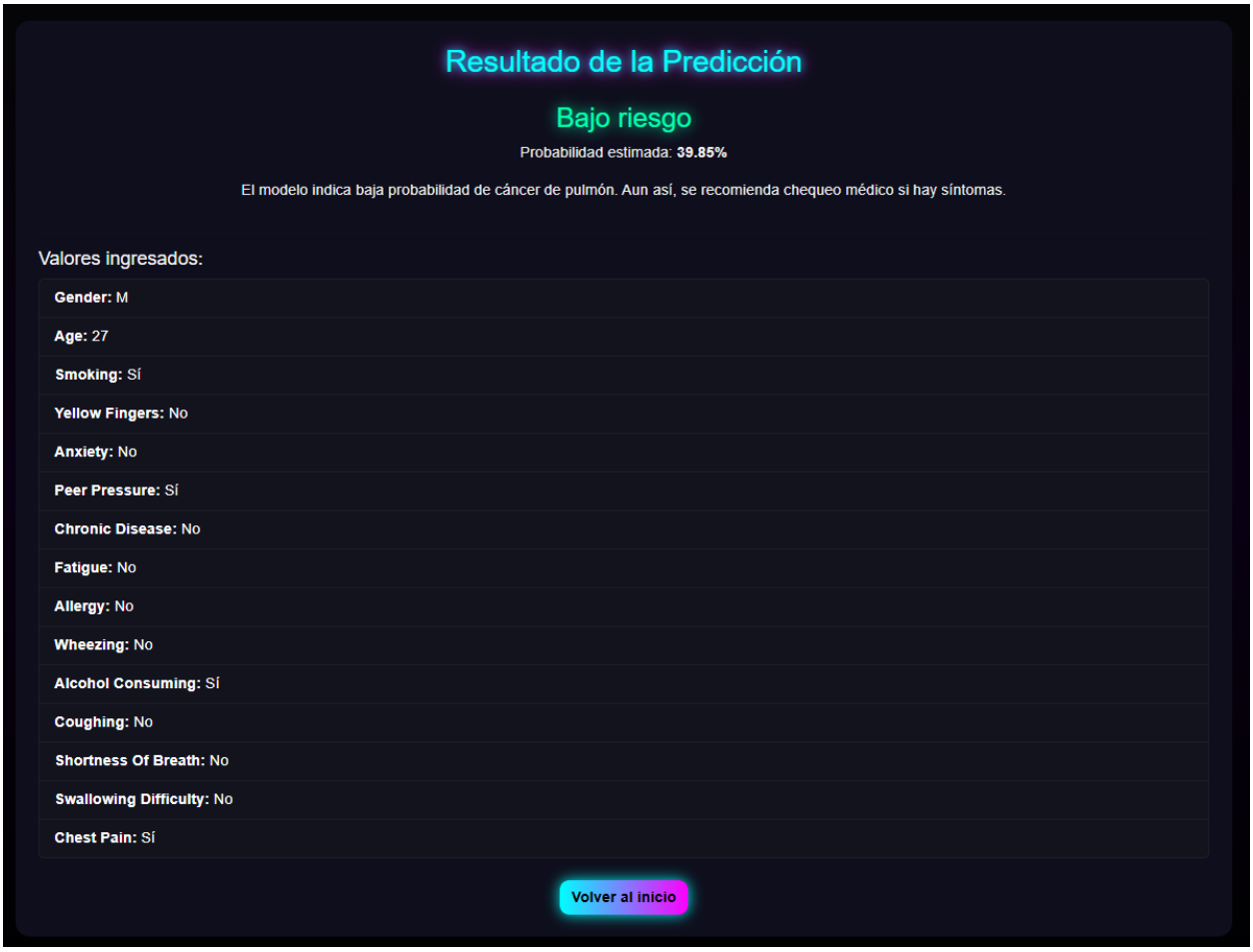
Ejemplo de uso:python app.py

Abrir en el navegador: http://127.0.0.1:5000

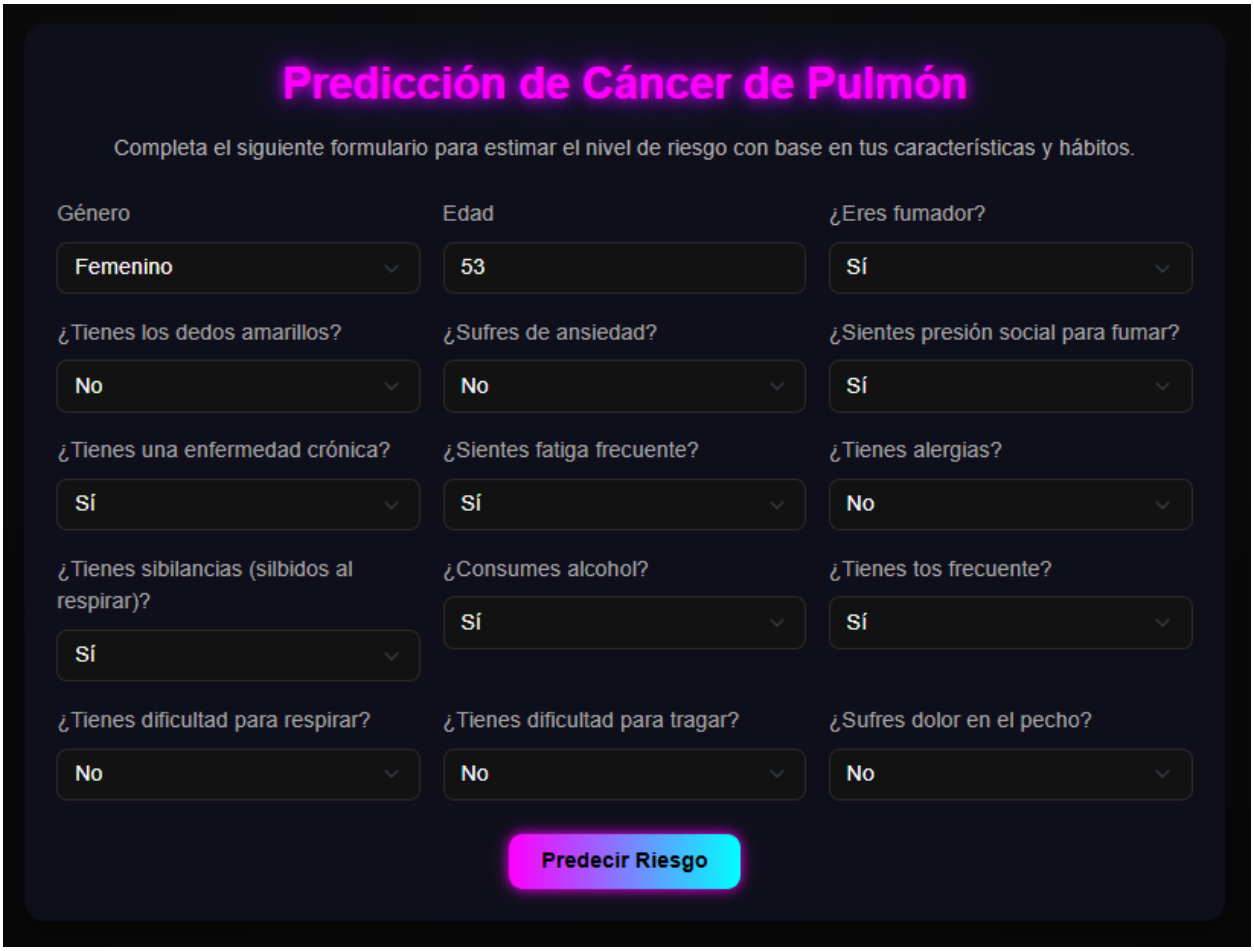
Ingresamos datos del formulario y damos predecir riesgo (bajo riesgo):



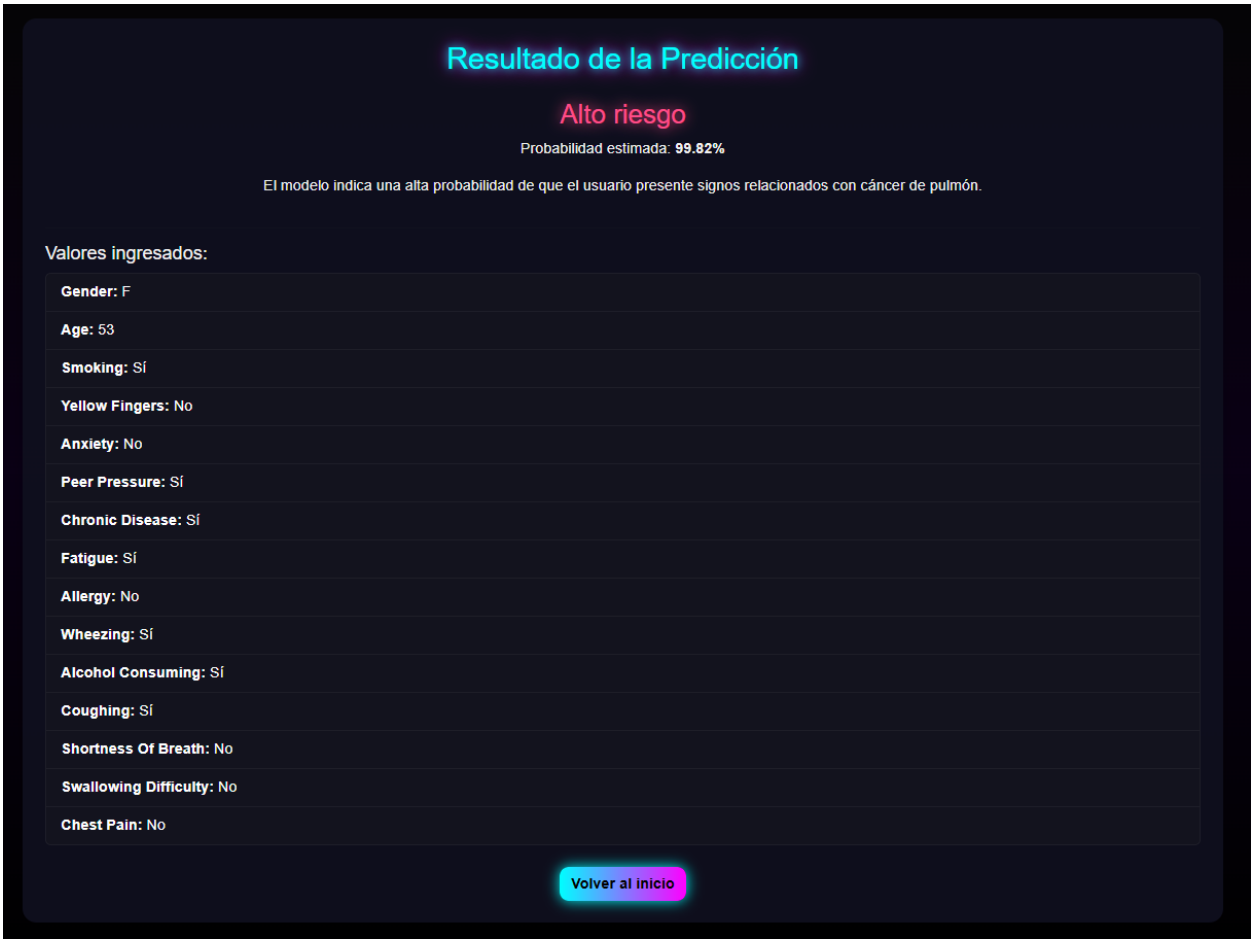
Obtenemos resultados de la predicción de bajo riesgo



Ingresamos datos del formulario y damos predecir riesgo (alto riesgo):



Obtenemos resultados de la predicción de alto riesgo



V. Conclusión

El proyecto implementa un flujo completo de Machine Learning aplicado a salud, desde la exploración y limpieza de datos hasta el despliegue de un modelo predictivo en una interfaz web. El modelo final logra un buen equilibrio entre recall y precisión, priorizando la detección de posibles casos positivos. Además, el EDA y los reportes automáticos permiten analizar el comportamiento del dataset y validar el desempeño del modelo de manera visual e interpretativa.

Archivos Finales