**Trabajo Final de Master**

Edgar Andrés Marín Vega

Universidad Internacional de Valencia

Desarrollo de Aplicaciones y Servicios Web

MASW10

**Índice**

[1. Introducción 1](#_Toc111459274)

[2. Marco Teórico 1](#_Toc111459275)

[2.3 Stack MEAN 3](#_Toc111459276)

[2.3 Machine Learning 5](#_Toc111459277)

[2.3.1 Fuente de Datos 6](#_Toc111459278)

[3. Diseño de la Solución 7](#_Toc111459279)

[3.1 Persistencia 8](#_Toc111459280)

[3.2 Negocio 8](#_Toc111459281)

[3.3 Presentación 8](#_Toc111459282)

[4. Desarrollo de la Aplicación 9](#_Toc111459283)

[4.1 Desarrollo Interfaz de Usuario 9](#_Toc111459284)

[Luego se ejecuta el comando que permite crear el proyecto con sus dependencias: 9](#_Toc111459285)

[4.2 Desarrollo Aplicación lado Servidor 9](#_Toc111459286)

[4.3 Capa de Persistencia 9](#_Toc111459287)

[4.4 Modelo de Machine Learning 10](#_Toc111459288)

[4.4.1 Adicionando la variable de respuesta: 11](#_Toc111459289)

[4.4.2 Agrupación de entrenamiento y pruebas: 11](#_Toc111459290)

[4.4.3 Preprocesado de variables predictivas: 12](#_Toc111459291)

[6. Otros modelos de interés 15](#_Toc111459292)

[7. Biografía 16](#_Toc111459293)

# 1. Introducción

El propósito principal del proyecto aquí presentado es el de desarrollar una aplicación web utilizando el stack MEAN, en cumplimiento con la propuesta del trabajo final de máster presentada, siguiendo el procedimiento establecido para ello. Las tecnologías involucradas en dicho stack serán tratadas a profundidad en el presente informe.

Con relación al dominio de la aplicación, consiste en una herramienta de tipo triage que sirva de apoyo al personal médico, para determinar, de acuerdo con el estado de salud de un paciente, si este debe ser internado o no en un centro médico. Los datos se obtendrán de fuentes públicas las cuales cuenten con normas internacionales de tratamientos de datos con el fin de evitar infringir alguna normatividad relacionada. Se utilizarán algoritmos de Machine Learning con el propósito de realizar predicciones, de acuerdo con las condiciones médicas del paciente.

Objetivo general:

Diseñar e implementar una aplicación que permita mejorar el proceso de triage en un centro de emergencias, utilizando el stack de tecnologías MEAN.

Objetivos específicos:

* Desarrollar un modelo de Machine Learning que permita determinar de forma temprana si un paciente debe o no ser admitido en el área de emergencias
* Diseño de una interfaz gráfica que contenga un formulario que permita ingresar la información necesaria para evaluar el modelo de machine learning.

# 2. Marco Teórico

**2.1 Dominio de la aplicación**

El proyecto aquí presentado hace parte del dominio de la salud, en el cual la tecnología ha comenzado a hacer sus aportes desde hace aproximadamente 200 años, con el descubrimiento de los Rayos X (Contribuidores colegiomedicodemexico.org, 2019). En la época actual, la información cada vez juega un rol más importante en los avances tecnológicos y la salud no es la excepción. La internet y los avances en las tecnologías de almacenamiento de información han permitido que cualquier persona o entidad tenga la posibilidad de almacenar grandes volúmenes de datos sin la necesidad de invertir en grandes sumas de dinero. Esto ha llevado a que se desarrollen nuevas formas de IA (Inteligencia Artificial) a partir de dichos volúmenes de datos.

La IA se define como la ciencia de computación que desarrolla sistemas diseñados para realizar determinadas operaciones que se consideran propias de la inteligencia humana. Dentro de la IA existen diferentes campos, dentro de los cuales se encuentra el campo de Machine Learning, en el cual nos centraremos en adelante.

**2.2 Machine Learning**

El ML o aprendizaje automático, es un subcampo de la IA que permite a un sistema aprender de los datos en lugar de aprender mediante programación explícita. Conforme el algoritmo de aprendizaje ingiere datos de entrenamiento, es posible producir modelos más precisos. Un modelo de ML es la salida de información que se genera cuando se entrena el algoritmo de ML con datos.

En resumen, los algoritmos de ML aprenden de forma automática a realizar unatarea o hacer predicciones a partir de datos y mejorar su rendimiento con el tiempo (datascientest contributors, 2021).

**Principales Algoritmos de ML**

Existe una amplia variedad de algoritmos de ML. Los algoritmos de regresión, lineal o logística, permiten comprender relaciones entre los datos. La regresión lineal se utiliza para predecir el valor de una variable dependiente en función de una variable independiente, por ejemplo, predecir el clima en función de las condiciones meteorológicas.

La regresión logística a su vez se utiliza cuando las variables dependientes son binarias, que es nuestro caso ya que deseamos predecir si el paciente debe ser o no admitido en sala de emergencias.

Otro algoritmo muy popular es el árbol de decisiones el cual permite establecer recomendaciones basadas en un conjunto de reglas de decisión. Por ejemplo, es posible recomendar un candidato en un proceso de selección de una compañía basándose en su experiencia.

Los algoritmos de regresión lineal o logística se encuentran dentro de una técnica de ML denominada **aprendizaje supervisado**, en la que los datos son etiquetados o clasificados antes del entrenamiento, para indicar al modelo que patrones debe buscar.

Cuando se tienen datos no etiquetados, se utilizan algoritmos de Clustering, los cuales permiten identificar grupos con características similares y etiquetar estos registros según el grupo al que pertenecen. Este tipo de algoritmos hacen parte de las técnicas de aprendizaje **no supervisado**, el los cuales se busca encontrar patrones y relaciones que no se han identificado previamente en los datos.

**Machine Learning (ML) en Medicina:**

En la medicina existen diferentes propósitos para los que un modelo de ML puede ser aplicado, entre estos propósitos se encuentran:

* Efectividad del diagnóstico
* Efectividad de un tratamiento
* Prevención de afectaciones a la salud
* Mejora de la calidad de vida
* Reducción de costos en los sistemas de salud
* Mejorar la calidad de los servicios de salud

De acuerdo con el objetivo principal planteado en nuestro proyecto, nuestro modelo de ML permitiría reducir el tiempo de triage en los centros de salud donde sea implementado. Esto permitiría reducir los costos de operación de dichos centros de salud y por ende mejorar la calidad del servicio.

# 2.3 Stack MEAN

Se utilizará el stack MEAN, tal como se definió en la propuesta del presente proyecto. Dicho stack se compone de las siguientes tecnologías:

* **M**ongoDB (Persistencia)
* **E**xpress (Servidor)
* **A**ngular (Presentación)
* **N**odeJS (Servidor)

**2.2.1 MongoDB**

MongoDB es una base de datos multiplataforma, orientada a documentos. Utiliza documentos en formato JSON y está clasificada como una base de datos NoSQL. (Wikipedia Contributors, 2022)

A comparación de las bases de datos tradicionales de tipo SQL, en las cuales los datos se almacenan de forma estructurada en registros y columnas con una estructura predefinida, las bases de datos documentales como MongoDB permiten almacenar datos de tipo semiestructurado en donde no es necesario definir la estructura de dichos documentos para su almacenamiento.

**2.2.2 NodeJS**

Es un entorno de ejecución de JavaScript orientado a eventos asíncronos el cual está diseñado para crear aplicaciones escalables y que operan en la red. Este es un sistema que contrasta con el modelo de concurrencia utilizado en otros lenguajes en los que se emplean hilos del sistema operativo. Este sistema ofrece algunas ventajas, como el hecho de que las operaciones no sean bloqueantes, lo cual contribuye a la escalabilidad de dichas aplicaciones y además, los desarrolladores están libres de preocuparse por el bloqueo de dichos procesos.

Si embargo, a pesar de que NodeJS está diseñado para trabajar sin hilos, es posible generar subprocesos utilizando la API child\_process.fork() y de esta forma aprovechar múltiples nucleaos en su entorno (OpenJS Foundation and NodeJS contributors, 2022).

**2.2.3 Express**

Express es un servidor web desarrollado para Node.js. Es parte de la infraestructura requerida para la ejecución de aplicaciones web Node.JS y proporciona un conjunto de características para las aplicaciones web y móviles (expressjs.com contributors, s.f.).

Sus principales características se encuentran:

* Una capa de enrutamiento que opera sobre la capa de HTTP de NodeJS
* Enrutamiento declarativo
* Componente Middleware

**2.2.4 Angular**

Es un framework de diseño de aplicaciones y plataforma de desarrollo para crear aplicaciones móviles y aplicaciones web de tipo SPQ (Single Page Application). Está desarrollado en TypeScript, es de código abierto y actualmente es mantenido por Google (Colaboradores de Wikipedia, 2022).

Angular hace uso del matrón MVC (Modelo-Vista-Controlador) el cual separa la capa de datos y la lógica de la aplicación de la interfaz de usuario.

* Modelo: Representación de la información y gestión del acceso a dicha información
* Vista: Interfaz de usuario
* Controlador: Gestiona el flujo de información entre el modelo y la vista y gestiona los eventos.

# 2.3 Machine Learning

Machine Learning es una rama de la IA (Inteligencia Artificial), cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender.

El propósito de nuestro modelo es el de ofrecer una herramienta que permita mejorar el proceso de triage en el área de urgencias de un hospital, permitiendo predecir de forma temprana si el paciente debe ser o no internado.

En el dominio médico existen diferentes áreas en las cuales se pueden aplicar algoritmos de machine learning que son:

**Screening:** Se define como la identificación de una enfermedad de forma previa a la aparición de alguna señal o síntoma. Un ejemplo en esta área son los algoritmos de machine learning utilizados para el análisis de imágenes de rayos X.

**Diagnóstico:** Este tipo de analíticas están enfocadas principalmente al apoyo en el tratamiento de un síntoma o enfermedad preexistente.

**Pronóstico:** Este tipo de resultado está orientado a determinar la probabilidad de sobrevivencia a cierto padecimiento en términos de un periodo de tiempo, dadas las condiciones de un paciente.

**Respuesta al Tratamiento:** Este tipo de analíticas son también conocidas como medicina personalizada, dado que tienen en cuenta factores particulares de cada paciente. Este tipo de analíticas son recientes y actualmente están ganando popularidad.

**Desenlace:** Este tipo de analítica pretende determinar la probabilidad de un desenlace dado Ej. Ingreso por urgencias o Muerte. Este tipo de analíticas en ocasiones no tienen cuenta la causa, por lo que en estos casos no se enfoca en los síntomas específicos o signos vitales de un paciente. El algoritmo de Machine Learning desarrollado en este proyecto se encuentra dentro de esta categoría.

# 2.3.1 Fuente de Datos

Se utilizarán datos públicos obtenidos del National Center for Health Statistics desde el sitio <https://www.cdc.gov/nchs/ahcd/about_ahcd.htm>, de los cuales se utilizará el dataset NHAMCS (National Hospital Ambulatory Medical Care Survey) dado que es uno de los más completos en cuanto a documentación y volumen de datos que se encuentran en internet.

Los dataset seleccionados están disponibles a partir del año 2013 (ED2013.ZIP) hasta el año 2019 (ED2019.ZIP) y se encuentran en el siguiente directorio FTP.

<https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Datasets/NHAMCS/>

La documentación de estos archivos se podrá encontrar en el siguiente enlace:

<ftp://ftp.cdc.gov/pub/Health_Statistics/NCHS/Dataset_Documentation/NHAMCS>

Se muestra a continuación el resumen de nuestro dataset, el cual se encuentra en la documentación de el mismo, en el archivo doc13\_ed.pdf.

*This material provides documentation for users of the Emergency Department public use micro-data file of the 2013 National Hospital Ambulatory Medical Care Survey (NHAMCS). NHAMCS is a national probability sample survey of visits to hospital outpatient and emergency departments, and hospital-based ambulatory surgery centers, conducted by the National Center for Health Statistics, Centers for Disease Control and Prevention. The survey is a component of the National Health Care Surveys, which measure health care utilization across a variety of health care providers.*

Este dataset contiene datos de tipo estructurado, los cuales se tabulan durante el triage de un paciente a su llegada a los centros de salud. Cada registro se compone de una serie de preguntas que se realizan a todos los pacientes al llegar a sala de emergencias para lo cual cuentan con unas encuestas cuyo formato ha sido preestablecido. Esta serie de preguntas y sus correspondientes respuestas se pueden caracterizar en dos grandes grupos que son:

* Datos Subjetivos: Información que puede estar sujeta al criterio del personal médico
* Datos Objetivos: Corresponde a resultados de muestras de laboratorio y pruebas diagnósticas.

# 3. Diseño de la Solución

La aplicación se dividirá en 3 capas que son, capa de presentación, servicios y persistencia.

Mas adelante se dará una descripción en detalle de cada uno de estos componentes, los cuales se incluyen en el siguiente diagrama el cual identifica dichos componentes dentro de nuestra aplicación.

Diagram

Description automatically generated

**3.1 Interfaz de usuario**

Para el diseño de la interfaz gráfica se ha considerado crear un diseño simplista, propio de un mínimo producto viable, teniendo en cuenta que un diseño más elaborado requiere de conceptos mucho más profundos, entre los cuales se contempla el diseño de una marca y el diseño de la experiencia de usuario, los cuales, se encuentran fuera del alcance de este proyecto.

**Creación del Logo**

Se ha escogido el nombre TriageIT como nombre de la página que permitirá acceder a la aplicación de triage. Una vez definido el nombre procedemos a la creación del logo para lo cual se ha utilizado la herramienta Dafont (https://www.dafont.com/) la cual permite comparar diferentes estilos de fuentes que podrán utilizarse para este propósito a partir del nombre de nuestra marca.

**Logo, company name

Description automatically generated**

# 3.1 Persistencia

Esta capa se es la encargada de almacenar los datos de nuestra aplicación. Estos datos provienen de un proceso ETL que se encargará de extraer los datos desde fuentes de datos abiertos, los cuales podrán ser archivos CSV o servicios REST.

**ETL**

Para el componente ETL se desarrollará una aplicación en Java utilizando las librerías de SpringBoot, la cual permite crear una aplicación escalable de forma rápida y es en estas tecnologías con las cuales tengo mayor experiencia.

# 3.2 Negocio

En esta capa se crearán todos los servicios que proveerán los datos a la capa de presentación. Estos servicios contendrán la lógica de negocio de nuestra aplicación, cargar los datos desde la capa de persistencia y todas las validaciones que sean requeridas.

# 3.3 Presentación

La capa de presentación contiene las interfaces de usuario con las que el usuario final va a interactuar y que le permitirán visualizar la información deseada. Toda la lógica relacionada con la interfaz gráfica estará definida en esta capa.

# 3. Desarrollo de la Aplicación

**3.1 Interfaz de Usuario**

**3.2 Aplicación lado Servidor**

En el lado del servidor serán implementados los servicios de nuestra aplicación, los cuales contendrán la lógica necesaria que permitirá procesar las peticiones recibidas desde la interfaz de usuario y entregar la información requerida por el usuario.

Los servicios implementados para el formulario de triage son los siguientes:

survey\_sex

survey\_country

…

**3.3 Capa de Persistencia**

# 4. Instalación de la Aplicación

# 4.1 Interfaz de Usuario

Para el desarrollo de la interfaz de usuario se utilizó la versión de Angular 4. Para ello se ha utilizado la versión de Node.JS 16.16

**4.1.1 Creación de proyecto Angular**

Para crear aplicaciones con Angular, es necesario contar con un administrador de dependencias para lo cual se ha utilizado npm versión 8.17.0. Para realizar la instalación de la interfaz de comandos de Angular se utiliza el siguiente comando:

npm install -g @angular/cli

# Luego se ejecuta el comando que permite crear el proyecto con sus dependencias:

ng new triageit --routing --style=scss

Como entorno de desarrollo se utilizó Visual Studio Code, versión 1.64. Una vez cargado el proyecto en el IDE.

# 4.2 Configuración del Servidor

Para la instalación del servidor Express se utiliza el siguiente comando:

npm install --save express body-parser

// Crea el directorio dist y compila el proyecto

ng build --base-href triageit

Se crea el archivo server,js en el cual se definen las configuraciones de enrutamiento del servidor:

const express = require('express');

const bodyParser = require('body-parser');

const path = require('path');

const http = require('http');

const app = express();

// Parsers

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: false}));

// Archivo de estadísticas del servidor

app.use(express.static(path.join(\_\_dirname, 'dist')));

// Establece las rutas de api

//app.use('/api', api);

// Retornar otras rutas index.thml

app.get('\*', (req, res) => {

res.sendFile(path.join(\_\_dirname, 'dist/triageit/index.html'));

});

// Set port

const port = process.env.PORT || '3000';

app.set('port', port);

// Create the HTTP Server

const server = http.createServer(app);

server.listen(port, () => console.log('Running on localhost'));

Ejecución del servidor:

// Ejecuta el servidor

node server.js

**4.2.1 API Rest**

La API Rest permite la ejecución de comandos desde Angular con el fin de interactuar con MongoDB. Para configurar el Api Rest se crea el archivo api.js al interior del directorio “server/routes” del proyecto Angular, el cual hace referencia a la librería mongoose. Esta librería permite la conexión entre el servidor node.js y la instancia de base de datos.

// Instalación de la librería mongoose

npm install mongoose --save

# 4.3 Capa de Persistencia

Se utilizará la librería spring-boot-starter-data-mongodb, la cual contiene los drivers necesarios para establecer la conexión con la base de datos MongoDB y permite abstraer las operaciones de tipo CRUD que permitirán insertar los datos en la base de datos. <https://www.baeldung.com/spring-data-mongodb-reactive>.

En primera instancia se implementará el flujo de la ETL, tomando los datos desde archivos de tipo CSV los cuales se cargarán a un Bucket de S3. Para acceder a los objetos almacenados en el bucket se utilizará el SDK de Java disponible en las herramientas para desarrolladores de AWS.

A picture containing diagram

Description automatically generated

**MongoDB**

Para nuestra instancia de MongoDB se utilizará el servicio en la nube MongoDB cloud, en modalidad Serverless, el cual es gratuito siempre y cuando no se exceda la cuota establecida. <https://www.mongodb.com/docs/atlas/billing/serverless-instance-costs/>

Se crea un usuario específico para la aplicación, con el fin de no utilizar el usuario por defecto ‘admin’ lo cual representaría una brecha de seguridad

A picture containing application

Description automatically generated

La cadena de conexión a la instancia es la siguiente:

mongodb+srv://usuario\_aplicacion\_0001:<password>@triageit.khwbq.mongodb.net/triageit

# 4.4 Modelo de Machine Learning

Los datos serán importados y procesados con Python, utilizando una sesión de Jupyter. Para la importación del archivo, fue necesario un archivo de ayuda (ED\_metadata.csv) el cual contiene el ancho, nombre y tipo de variable de cada columna del datasource y fue tomado de (Kumar, 2018).

Importando la Metadata:

import pandas as pd

pd.set\_option('mode.chained\_assignment',None)

HOME\_PATH = 'D:\\maestria\\MASW10 TFM\\Datos\\Jupyter\\'

df\_helper=pd.read\_csv(HOME\_PATH + 'ED\_metadata.csv', header=0,dtype={'width': int, 'column\_name': str, 'variable\_type': str})

Importando el Dataset:

width=df\_helper['width'].tolist()

col\_names=df\_helper['column\_name'].tolist()

var\_types=df\_helper['variable\_type'].tolist()

df\_ed=pd.read\_fwf(HOME\_PATH + 'ED2013', widths = width, header=None, dtype='str')

df\_ed.columns=col\_names

# 4.4.1 Adicionando la variable de respuesta:

Para nuestro caso, se desea predecir cuales son los pacientes que serán hospitalizados tras presentarse a la sala de emergencias, lo cual incluye:

* Pacientes remitidos a sala de hospitalización
* Pacientes remitidos a otro hospital
* Pacientes que son dejados en observación

Teniendo en cuenta estos casos, se crea una columna adicional llamada ADMITFINAL, cuyo valor será 1, en aquellos registros que se cumpla una de las anteriores condiciones.

El código encargado de validar estas 3 condiciones utilizando la variable response\_cols y crear la columna ADMITFINAL se muestra a continuación:

response\_cols = ['ADMITHOS','TRANOTH','TRANPSYC','OBSHOS','OBSDIS']

df\_ed.loc[:, response\_cols] = df\_ed.loc[:, response\_cols].apply(pd.to\_numeric)

df\_ed['ADMITTEMP'] = df\_ed[response\_cols].sum(axis=1)

df\_ed['ADMITFINAL'] = 0

df\_ed.loc[df\_ed['ADMITTEMP'] >= 1, 'ADMITFINAL'] = 1

df\_ed.drop(response\_cols, axis=1, inplace=True)

df\_ed.drop('ADMITTEMP', axis=1, inplace=True)

# 4.4.2 Agrupación de entrenamiento y pruebas:

Los datos históricos deben ser segmentados con el fin de que una fracción de estos datos sirvan para probar si nuestro modelo ha sido entrenado correctamente. Usualmente se toma entre un 20-30% de los datos para probar los modelos.

El siguiente código permite separar la variable de respuesta ‘ADMITFINAL’ de nuestro dataset, lo cual es necesario para entrenar el modelo

def split\_target(data, target\_name):

target = data[[target\_name]]

data.drop(target\_name, axis=1, inplace=True)

return (data, target)

X,y = split\_target(df\_ed, 'ADMITFINAL')

El proceso de seleccionar los datos de entrenamiento y prueba es realizado de forma automática por la librería scikit-learn. El entrenamiento de nuestro modelo se realiza con el siguiente código.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=1234)

Como resultado de la segmentación de los datos, se observa que hay proporcionalidad de la variable de respuesta entre los datos de entrenamiento y los de prueba. Es decir, la proporción que el algoritmo ha seleccionado entre los dos posibles resultados para los pacientes admitidos es muy similar entre los datos de entrenamiento 16,6% y los datos de validación 15,5%.

print(y\_train.groupby('ADMITFINAL').size())

ADMITFINAL

0 15996

1 2586

dtype: int64

print(y\_test.groupby('ADMITFINAL').size())

ADMITFINAL

0 5362

1 833

dtype: int64

# 4.4.3 Preprocesado de variables predictivas:

Algunas variables pueden ser preprocesadas con el fin de enriquecer el entrenamiento de nuestro modelo con información que no está explícita en nuestro dataset. Por ejemplo, la columna que contiene la hora de llegada del paciente por sí misma puede no es muy útil, dado que puede ser un entero entre 0 y 2359.

**Variable HOUR\_RANGE:**

Se crea una nueva variable que contenga los siguientes rangos de hora: 21:01-3:00, 3:01-6:00, 6:01-11:00, 11:01-16:00 y 16:01-21:00, a partir de la columna ARRTIME. Posteriormente se elimina esta última columna.

def hour\_range(arrtime):

arrtime\_int = int(arrtime)

if ((arrtime\_int >= 2100)):

return 1

elif ((arrtime\_int >= 0) & (arrtime\_int < 300)):

return 1

elif ((arrtime\_int >= 300) & (arrtime\_int < 600)):

return 2

elif ((arrtime\_int >= 600) & (arrtime\_int < 1100)):

return 3

elif ((arrtime\_int >= 1100) & (arrtime\_int < 1600)):

return 4

elif ((arrtime\_int >= 1600) & (arrtime\_int < 2100)):

return 5

else:

return 0

X\_train.loc[:,'HOUR\_RANGE] = df\_ed.loc[:,'ARRTIME'].apply(hour\_range)

X\_test.loc[:,' HOUR\_RANGE'] = df\_ed.loc[:,'ARRTIME'].apply(hour\_range)

X\_train.drop('ARRTIME', axis=1, inplace=True)

X\_test.drop('ARRTIME', axis=1, inplace=True)

**WAITTIME:**

Debido a que esta variable requiere preprocesado puesto que contiene valores nulos y negativos que deben ser imputados, al igual que las variables ‘TEMPERATURE’, ‘PULSE’ y ‘RESPIRATORY\_RATE’.

X\_train.loc[:,'WAITTIME'] = X\_train.loc[:,'WAITTIME'].apply(pd.to\_numeric)

X\_test.loc[:,'WAITTIME'] = X\_test.loc[:,'WAITTIME'].apply(pd.to\_numeric)

def mean\_impute\_values(data,col):

temp\_mean = data.loc[(data[col] != -7) & (data[col] != -9), col].mean()

data.loc[(data[col] == -7) | (data[col] == -9), col] = temp\_mean

return data

X\_train = mean\_impute\_values(X\_train,'WAITTIME')

X\_test = mean\_impute\_values(X\_test,'WAITTIME')

**Variables Excluidas:**

Se excluirán algunas variables de nuestro entrenamiento, sea porque éstas han sido tomadas después de el triage del paciente y no nos sirven para predecir un evento previo, o porque estas variables son exclusivas de la población de EEUU, como el lugar de residencia y tipo de seguridad social, dado que el propósito es que este modelo sirva a poblaciones que se encuentren fuera de EEUU.

* Financial Variables
* Injury Codes
* Diagnostic Codes
* Tests
* Procedures
* Medication Codes
* Provider Information
* Disposition Information
* Electronic Medical Record Status

**One Hot Encoding:**

Ciertos clasificadores de la librería scikit-learn requieren que las variables de tipo categoría sean convertidas a columnas con valores binarios por lo cual, se utilizará la función de pandas get\_dummies() para realizar esta conversión.

categ\_cols = df\_helper.loc[

df\_helper['variable\_type'] == 'CATEGORICAL', 'column\_name'

]

one\_hot\_cols = list(set(categ\_cols) & set(X\_train.columns))

X\_train = pd.get\_dummies(X\_train, columns=one\_hot\_cols)

X\_test = pd.get\_dummies(X\_test, columns=one\_hot\_cols)

# 6. Otros modelos de interés

La propuesta consiste en la de presentar una herramienta que a partir de datos reales, se pueda obtener información valiosa y subjetiva, la cual pueda servir para tener un diagnóstico más personalizado y preciso, teniendo en cuenta una mayor cantidad de criterios, en comparación con los criterios generales aplicados por la medicina general.

La herramienta que se presenta tiene 2 componentes principales, destinados a mitigar esta situación que son, un componente de evaluación de riesgo de muerte por una causa específica y un componente que permita mostrar gráficas creadas a partir de datos reales y que se obtienen de fuentes de datos públicas.

Se consideran en principio las 3 principales causas de muerte, donde cada una de ellas cuenta con diferentes métodos de evaluación de riesgo, para lo cual se creó la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| Causa de Muerte | Método | Descripción |
| **Riesgo cardiovascular** | Framingham risk score | Se obtiene a partir de un análisis de unos factores de riesgo prestablecidos |
| **Insuficiencia cardiaca** | Ejection fraction | Requiere análisis por medio de electrocardiograma |
| **Cancer** | Combinación de factores de riesgo e imágenes | Se requiere análisis invasivo para determinar con precisión. |

# 7. Biografía

contributors, W. (2022, 08 12). *MongoDB*. Retrieved from Wikipedia: https://en.wikipedia.org/w/index.php?title=MongoDB&oldid=1104100098

Kumar, V. (. (2018). *Healthcare Analytics Made Simple: Techniques in healthcare computing using machine learning and Python.* Packt Publishing. Kindle Edition.