Optimización de traslados hacia clínicas de emergencia

Movilízate Salud - Hospitals Nearb-AI Me

Calderón Zetter Inés	ines@saturdays.ai
Farfan Perdomo Jorge	jorgefarfanperdomo@hotmail.com
Roman Chitala Luis Ernesto	Irchitala@gmail.com
Martínez Vargas Zaira Carolina	zaira.mar13@gmail.com
Alaniz Fabian Carlos Felix	carlos.alpha88@gmail.com
Angelina Alarcón	angelina@saturdays.ai

Resumen

Las emergencias son situaciones donde ocurren circunstancias que incluso llegan a poner en riesgo la vida. Aún así estas situaciones son muy comunes en la cotidianidad, registrándose en los primeros tres meses del año actual en México más de 500 mil llamadas médicas. Ante esto, se vuelve necesario encontrar métodos que nos ayuden a mitigar esta problemática. Por otro lado, los métodos de Machine Learning nos han mostrado soluciones óptimas en variedad de situaciones, por lo que en este trabajo se usan los métodos “Unsupervised Nearest Neighbors Clustering” y “Bayesian Ridge Regression” para encontrar los hospitales ideales para el usuario con base en la distancia y el tiempo de traslado, los consultorios y cantidad de médicos en estos. Además, se desarrolló una aplicación web con Django para la interfaz del usuario con el objetivo de que la aplicación fuera más personalizada y que pudiera ser consultada en un teléfono móvil.

Introducción

Uno de los aspectos más importantes en la vida es la salud, una buena salud es esencial para una contribución importante al progreso económico ya que personas sanas son más productivas y ahorran más (World Health Organization). No obstante, puede ocurrir la repentina aparición de una afección médica que hace que sea necesaria la atención médica inmediata, es decir, la asistencia de un especialista en salud.

Sin embargo, en el artículo de Oche MO[1] se ha mostrado uno de los problemas principales que afectan a este servicio: los tiempos de espera. Este es un importante indicador de la calidad de los servicios que brinda el hospital, ya que tener pacientes esperando más tiempo del necesario puede ser una causa de estrés para el personal de salud y pacientes. Por ello, muchos de estos estudios concluyen que para solucionarlo es necesario incrementar el número de trabajadores de la salud, pero ésta se vuelve una solución complicada de implementar.

Al tener presentarse una emergencia médica, el paciente o personas presentes deben de buscar a qué hospital dirigirse, y después, esperar que éste tenga disponibilidad para atenderlo lo más pronto posible o recurrir a una llamada de emergencia, aunque este servicio en el mes de abril en Jalisco recibió más de 18 mil llamadas de las cuales dos de cada diez fueron falsas[2]. Ésto tuvo como consecuencia que los pacientes recibieran una atención más tardada.

Ante una situación repentina que requiere atención médica de emergencia, no se cuenta con una herramienta más allá de las aplicaciones de mapas, que permita encontrar de manera ágil los servicios médicos de emergencia disponibles a nuestro alrededor así como tampoco una herramienta para saber los mejores hospitales para atenderse.

Por consiguiente, en este trabajo se utilizaron los métodos “Unsupervised Nearest Neighbors Clustering” y “Bayesian Ridge Regression” de Machine Learning para encontrar los hospitales de emergencia públicos (IMSS, SSA, ISSSTE) más óptimos para el paciente dentro de los siguientes municipios de la Zona Metropolitana de Guadalajara (ZMG): Tlaquepaque, Tonalá, Zapopan, El Salto, Tlajomulco y Guadalajara . De la misma manera, se desarrolló una aplicación web basada en Django que al ingresar la ubicación del usuario se generarán los hospitales más recomendables para el usuario con base en la distancia, tiempo de traslado, consultorios y médicos del hospital.

Marco Teórico

En la actualidad, ya existen aplicaciones similares en Play Store para encontrar el hospital más cercano y cuyas características son muy similares a las de este proyecto. Por ejemplo, por mencionar algunas tenemos: App CMDX[3] que muestra los hospitales más cercanos y el nivel de ocupación para pacientes con covid-19; Hospital Finder[4] de Arogya Online Health Care que localiza hospitales en un radio de 2, 4 y 8 kilómetros de distancia, proporciona información detallada de cada hospital con la ruta calculada como la más rápida así como el tiempo estimado de llegada; Nearby Near Me Hospital[5] por King Coder que da una lista de hospitales en un radio de diez kilómetros y Hospital Finder[6] por Dashan que proporciona información acerca de si los hospitales son públicos o privados, además de un sitio web y correo electrónico.

Sin embargo, estas aplicaciones tienen puntos en contra como la publicidad que aparece en cuanto se abre la aplicación además de mostrar los consultorios privados, nutriólogos y centros de salud no adecuados para emergencias.

Metodología

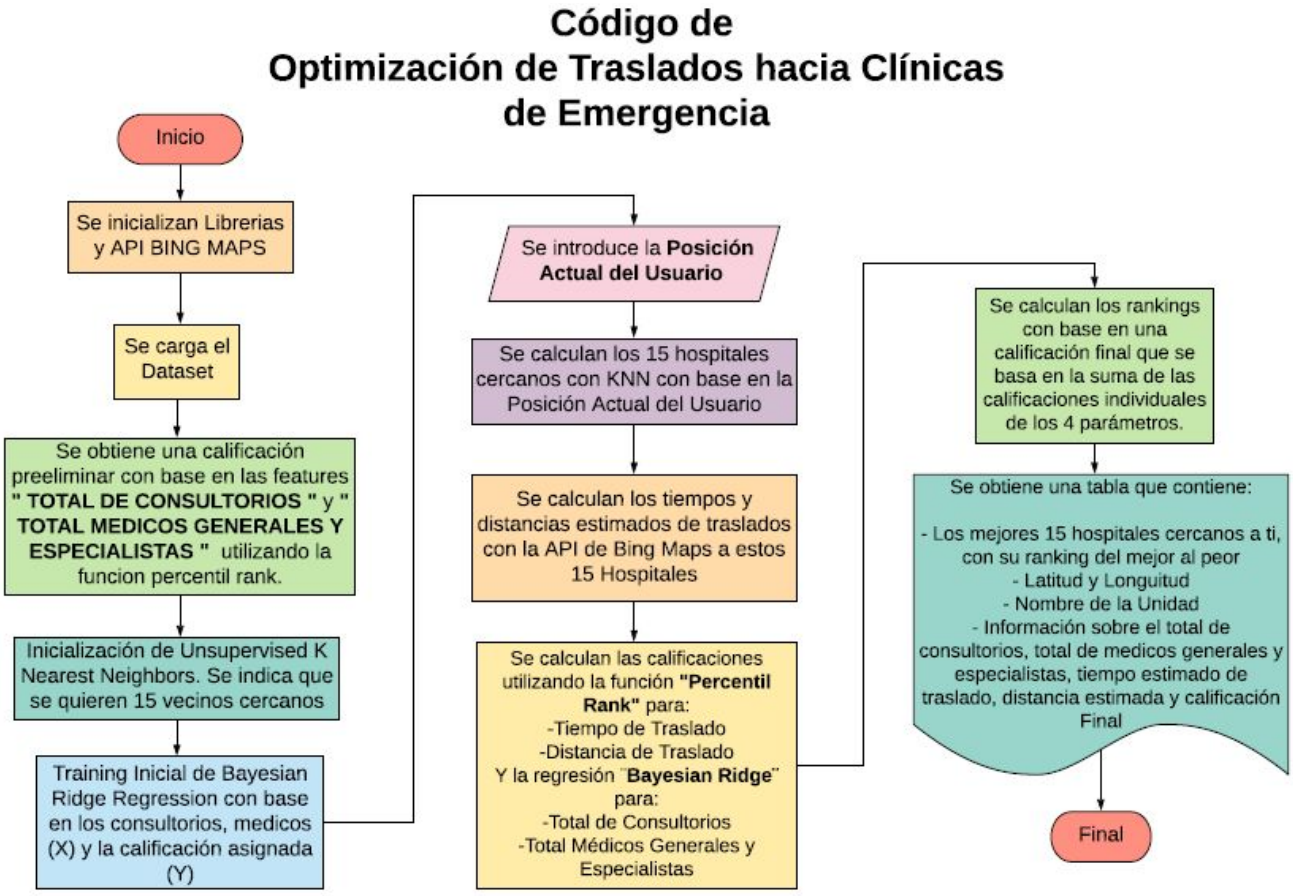

Fig. 1. Diagrama de flujo del programa donde se explican las distintas fases del procesamiento las cuales se presentarán en este trabajo.

1. **LIBRERÍAS**

Se utilizó el lenguaje de programación *Python* (Versión 3.7.4) y las librerías:

- **Pandas (Versión 0.25.1):** Provee herramientas para la manipulación de información, muy útil cuando se trabaja en ciencia de datos, ya que muchos algoritmos de ML, aceptan pandas como input.
- **Numpy (Versión 1.16.5):** Contiene herramientas que facilitan manipular información al igual que pandas, además de métodos matemáticos para poder hacer ciertas operaciones más fácilmente.
- **Matplotlib (Versión 3.2.1):** Contiene herramientas para realizar visualización de datos a través de gráficas, tablas, etc.

- **Scikit-Learn (Versión 0.21):** Contiene algoritmos ya desarrollados de machine learning, que facilitan mucho la ciencia de datos

2. Modelos de Machine Learning y API

El primer problema fue encontrar el hospital más cercano al usuario, por lo que se empleó una API y un algoritmo de clustering.

Una **API** (**A**pplication **P**rogram **I**nterface) es un conjunto de rutinas que permite el acceso a determinadas funciones de un software . Es decir, al utilizarla un programa hace un conjunto de servicios disponibles para usarlo en otras aplicaciones. En este caso, la API usada fue *Bing maps* que sirvió para encontrar la ruta y el tiempo de traslado.

3. DATA SET

Se construyó un dataset tomando información de bases de datos de páginas de la secretaria de salud[7] del gobierno de México del año 2018. Se redujo el problema al aplicarlo a seis municipios del estado de Jalisco: Tonalá, Tlaquepaque, Zapopan, Tlajomulco, El Salto y Guadalajara. Las cinco primeras filas se muestran en el siguiente cuadro:

	AÑO	CLUES	CLAVE INSTITUCION	CLAVE MUNICIPIO	LATITUD	LONGITUD	NOMBRE DE LA UNIDAD	TOTAL DE CONSULTORIOS	TOTAL MEDICOS GENERALES Y ESPECIALISTAS
0	2018	JCIMS000296	IMS	39	20.686922	-103.328075	BCO. SANGRE OBLATOS	0	4
1	2018	JCIMS001112	IMS	120	20.729193	-103.390566	C.C.S. MENTAL 1	15	9
2	2018	JCSSA013646	SSA	97	20.475000	-103.447000	CARAVANA DE LA SALUD TLAJOMULCO	0	2
3	2018	JCSSA009211	SSA	39	20.713143	-103.353109	CENTRO DE ATENCIÓN PRIMARIA EN ADICCIONES GUAD...	2	0
4	2018	JCSSA013465	SSA	97	20.518600	-103.376000	CENTRO DE ATENCIÓN PRIMARIA EN ADICCIONES NUEV...	2	0

Cuadro. 1. Primeros cinco datos del dataset que incluyen las columnas más importantes

4. **Unsupervised Nearest-Neighbor**

Mediante un algoritmo de clustering, podemos hacer subgrupos tales que las observaciones entre cada grupo son similares, y en este caso, esta similitud se basó en su ubicación. Se utilizó el método **Unsupervised Nearest-Neighbor Clustering**[8] que se usa para la agrupación de nuevas muestras, delimitar el espacio de búsqueda de nuestra aplicación, optimizar el consumo de datos de la API y la eficiencia del mismo. Un algoritmo Nearest-Neighbor puede ser supervisado, no supervisado, de Clasificación o Regresión, hay de diferentes tipos. Unsupervised NN es un algoritmo que busca las observaciones más cercanas a partir del punto de interés basado en la mayoría de datos que le rodea. El número de muestras puede ser mediante una constante o basado en la densidad local de puntos. La distancia puede ser cualquier métrica de medida, la distancia Euclidiana es la distancia más común . En este caso, al utilizar el método no supervisado, el algoritmo busca aquellos puntos que se encuentran a la distancia más cercana hasta obtener los 15 datos más cercanos. Al ser no supervisado, no se requiere una columna de predicción Y.

En este caso, el dataset contaba con 172 hospitales y se escogió que el número de nearest neighbors fuera igual a 15. El punto central de estos 15 puntos sería la posición actual del usuario. En K-means se podíamos especificar el número de clusters, pero no el número de elementos que iba a tener cada cluster. Además, desde el punto de vista computacional, NN es 35 veces más rápido que K-means.

Tiempo Nearest Neighbors: 0.005604582000160008 [s]
Tiempo Kmeans:: 0.1997328359993844 [s]

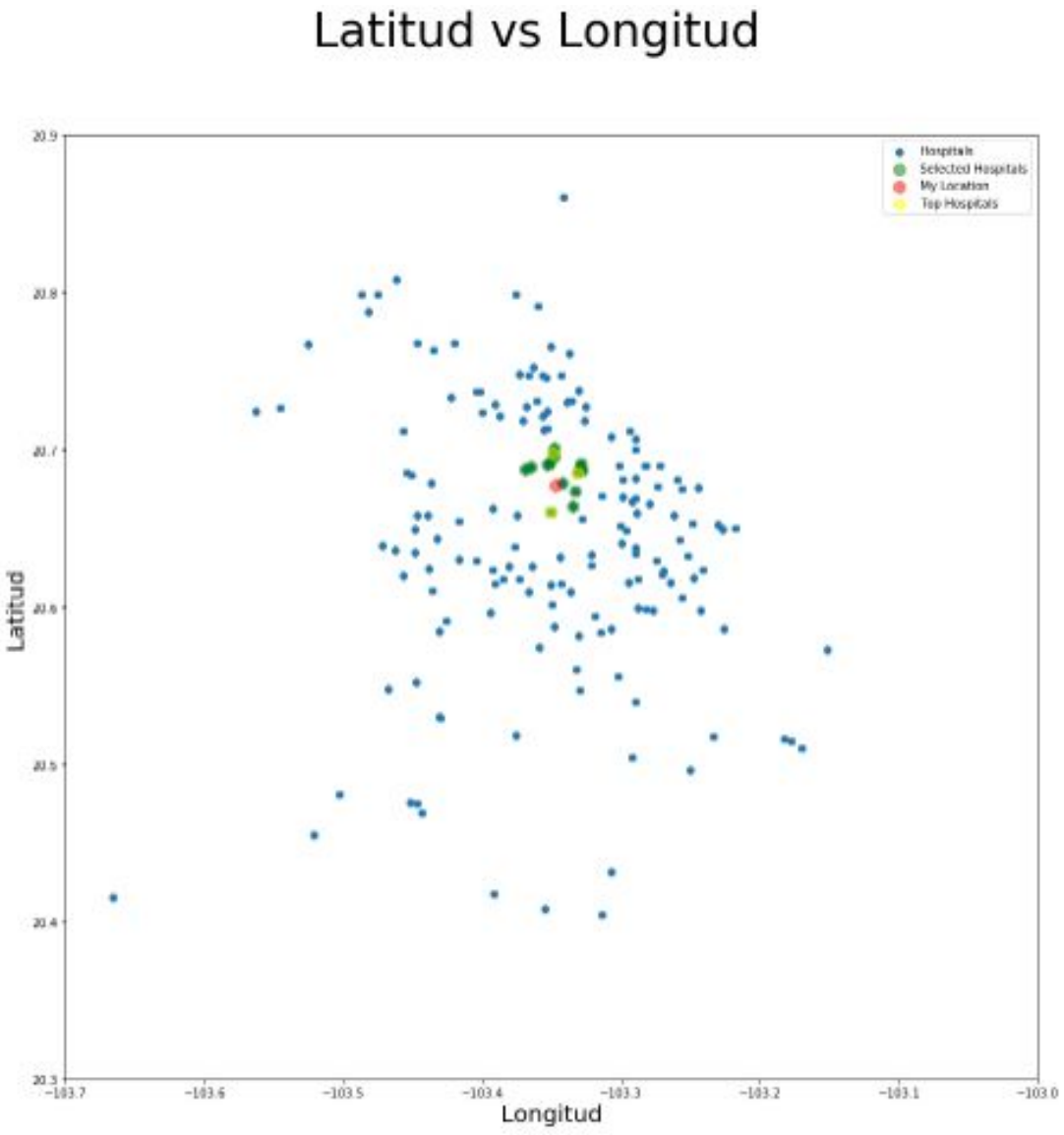

Fig. 2. Resultados después de haber aplicado el método de Unsupervised Nearest Neighbors. El punto rojo, es en donde se encuentra el usuario actualmente. En azul, los hospitales. En verde, los 15 hospitales más cercanos. Y en amarillo, los mejores 3 hospitales.

Se utilizó este algoritmo Unsupervised Nearest Neighbors (NN), debido a que NN da un mejor control a la hora de seleccionar el número de hospitales cercanos.

5. Bayesian Ridge Regression

Se utilizó el método de regresión **Bayesian Ridge Regression** que estima un modelo probabilístico del problema de regresión a través de la estimación de los parámetros ω α λ. Los parámetros de regularización α λ son estimados al maximizar lo que se llama *log marginal*

likelihood. Asimismo, **Ridge Regression** es una técnica para analizar datos de **regresión múltiple**, ya que cuando hay multicolinealidad, las estimaciones de mínimos cuadrados son insesgadas, por lo que su variación es grande y puede estar lejos del valor real[5].

En el entrenamiento inicial, se utilizaron la totalidad de los datos. Las variables que se ocuparon fueron “TOTAL DE CONSULTORIOS” y “TOTAL MÉDICOS GENERALES Y ESPECIALISTAS”, ésto conformaría el bloque de features X. El bloque Y, que es la columna de predicción, sería la CALIFICACIÓN que fue obtenida mediante la función **Percentil Rank** . Las métricas obtenidas de este entrenamiento fueron las siguientes:

Variance Score: 0.9319793338810748

Mean Absolute Error: 0.013237854269503531

Root Mean Squared Error: 0.018178524831451007

R2: 0.9311509376469557

A los 15 hospitales que se seleccionaron con NN se les asignó una calificación a los hospitales con base en las columnas “TOTAL DE CONSULTORIOS” y “TOTAL MÉDICOS GENERALES Y ESPECIALISTAS” mediante el método de regresión.

En este paso, se pudo haber obtenido este score utilizando la función **Percentil Ranking** y tomando todos los datos, pero esto computacionalmente era más costoso al utilizar todos los datos en lugar de obtener la información a partir del modelo de regresión. Mediante el método de regresión, la información se puede obtener 5 veces más rápido.

Tiempo Evaluación mediante Percentile Rank y todos los datos: 0.04411361400000047 [s]
Time Bayesian Ridge Regression con los 15 hospitales: 0.008890186999906291 [s]

Finalmente, a esta calificación se le sumaron las calificaciones individuales que se obtuvieron con la función percentile ranking utilizando la columna de ”Tiempo_Estimado(seg)” y “Distancia_Estimada(km)” que se calculó mediante la API BING MAPS. Este procedimiento se hace hasta este punto, debido a que el Tiempo estimado y la Distancia Estimada son variables dinámicas y no se pueden utilizar dentro del entrenamiento, debido a que si el usuario se mueve, el entrenamiento ya no sería válido.

Resultados

Al ingresar el usuario su longitud y su latitud, se obtiene el Cuadro 2. En este se muestran los resultados obtenidos después de aplicar ambos métodos, la última columna muestra el Ranking que se basa en la calificación final de los cuatro parámetros utilizados: Tiempo_estimado(seg), Distancia_Estimada(km), TOTAL DE CONSULTORIOS y TOTAL MEDICOS GENERALES Y ESPECIALISTAS. El mejor hospital será aquel que tenga un mejor ranking, en otras palabras, el que tenga un número más bajo siendo 1 el mejor resultado posible.

Indice	Latitud	Longuitud	NOMBRE DE LA UNIDAD	TOTAL DE CONSULTORIOS	MEDICOS GENERALES Y ESPECIALISTAS	Tiempo_Estimado(seg)	Distancia_Estimada(km)	Calificacion_Final	Ranking
170	20.679007	-103.342022	UNIDAD MÓVIL DE COLPOSCOPIA	1.0	1.0	242.0	0.783	0.578557	6.0
37	20.690800	-103.353000	CENTRO DE SALUD GUADALAJARA 3	13.0	26.0	474.0	2.067	0.609958	4.0
69	20.673800	-103.333000	CENTRO DE SALUD N°1	9.0	15.0	541.0	2.286	0.465255	9.0
157	20.691573	-103.352040	UMF 79 GUADALAJARA	10.0	15.0	571.0	2.765	0.408871	10.0
139	20.660603	-103.350850	UMF 1 GUADALAJARA	26.0	51.0	360.0	1.920	0.820690	1.0

Cuadro. 2. Resultados que tienen los quince hospitales con mejor ranking para la ubicación del usuario.

Funcionamiento de la Aplicación Web

La aplicación web recibe las coordenadas de tu móvil a través del navegador. Al presionar el botón de **“Buscar Hospitales!”** ejecuta el código antes mencionado y en una pantalla posterior te muestra en el mapa los 3 mejores hospitales, la información de estos y el enrutamiento hacia el mejor hospital.

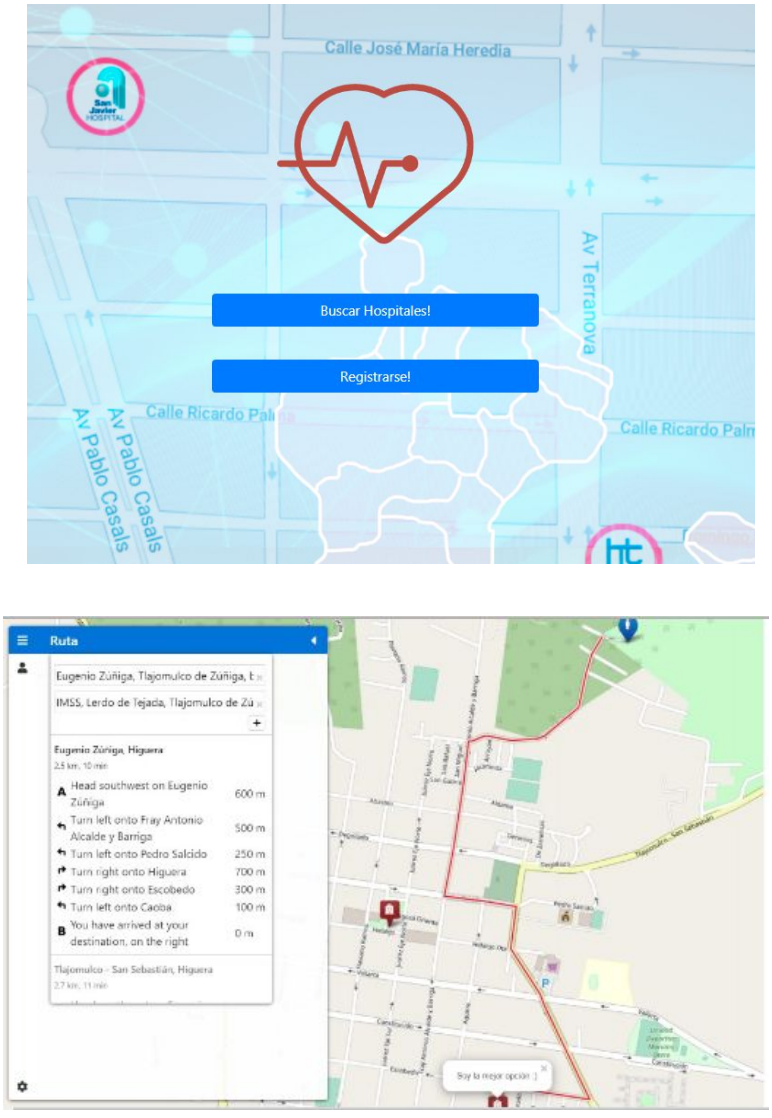

Figura 2. Prototipo de la aplicación

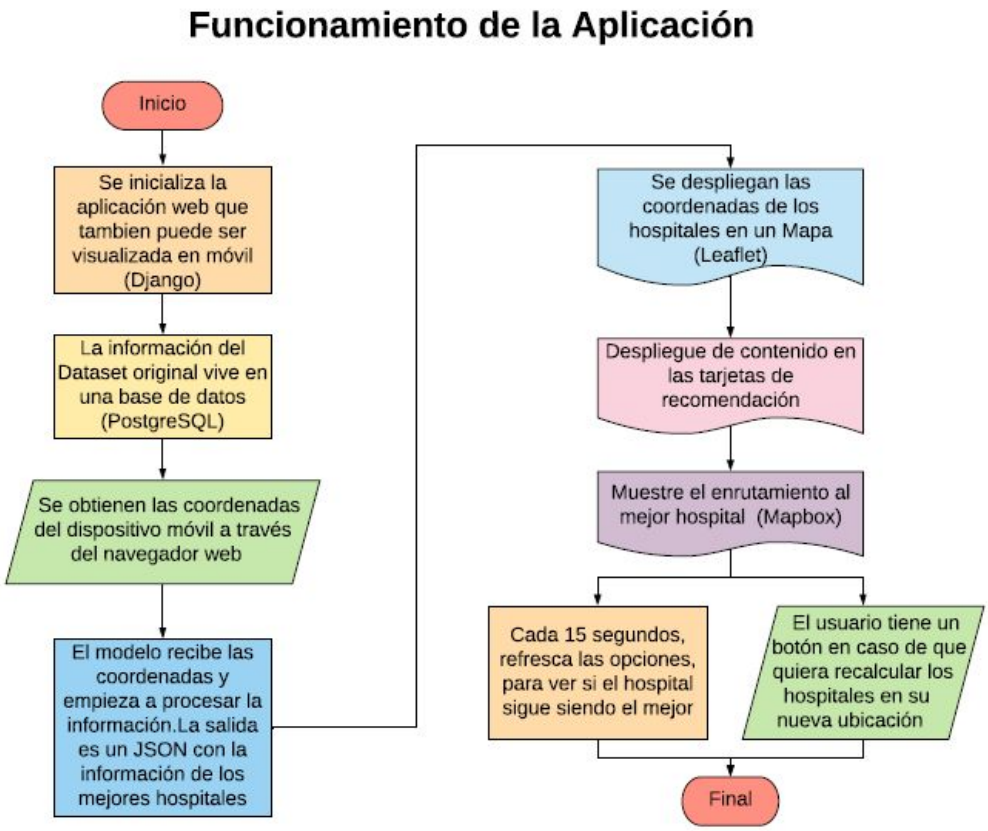

Figura 3. Funcionamiento de la Aplicación

Discusión y conclusión

Una importante comparación es que durante el mes de abril, el gobierno de la ciudad de México implementó una sección de la aplicación “App cmdx” que busca los hospitales más cercanos a ti y que puedan atenderte en el caso de tener síntomas del virus COVID-19, sin embargo, esta sólo funcionará para la pandemia ya que se eliminará esta opción en cuanto cese la declaratoria de emergencia en el país. Por ello, al implementar la aplicación de este trabajo, se espera que su funcionamiento sea a largo plazo y que después los datos recaudados sobre la disponibilidad de los hospitales y la calidad sean de los usuarios, teniendo así un mejor sistema de predicción. También, si se compara con el uso de google maps, en éste encontramos la ruta y el hospital más cercano, pero no sabemos cual tiene mejor probabilidad de ser mejor y en *Movilízate Salud* se obtiene un ranking basado en las características de los hospitales.

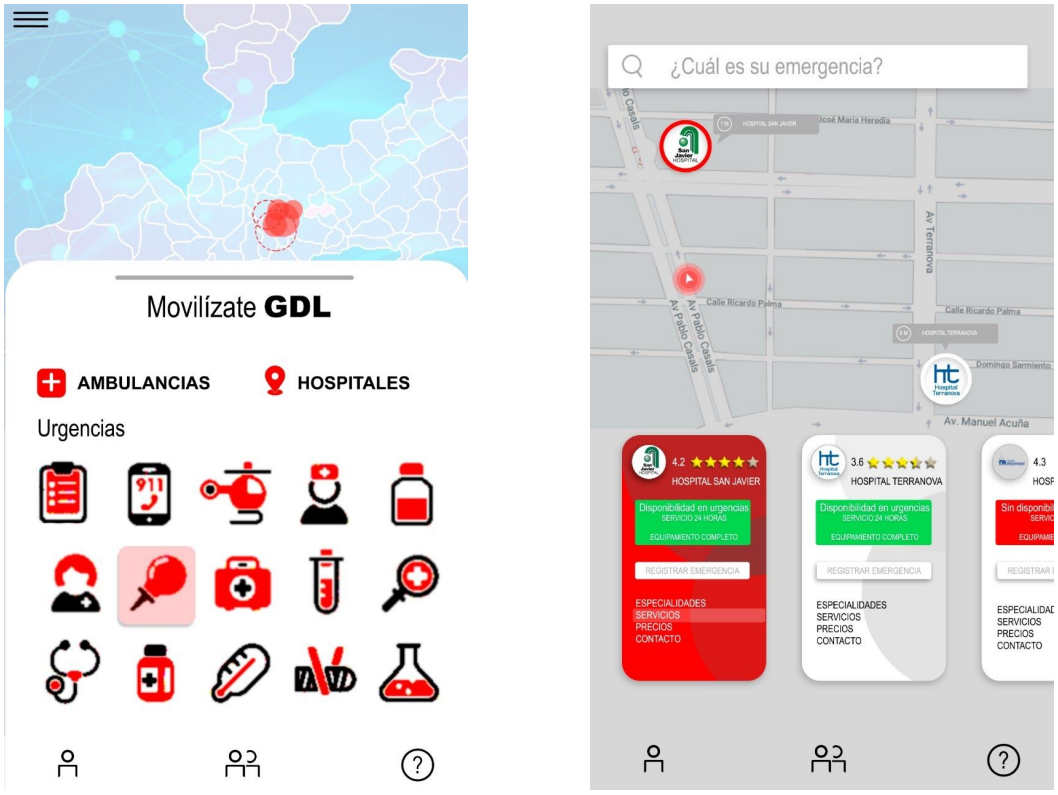

Fig. 4. Prototipo de aplicación a futuro

Se espera que a futuro la aplicación tome más puntos en cuenta y se vea como las imágenes de la Figura 4. Los símbolos de la primera imagen muestran el tipo de emergencia que el usuario pueda tener y que al seleccionar esta opción se filtre a un hospital que pueda atender esa emergencia. También, añadir un indicador de los precios de los hospitales y que los usuarios califiquen los servicios recibidos sería algo que robustecería la aplicación.

Código

- **Inicialización de variables, librerías y dataset**

Esta inicialización sólo se realiza una vez al ejecutar la aplicación

```
#Importando librerias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors
import pybingmaps
from sklearn.metrics import explained_variance_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import timeit
from sklearn.model_selection import train_test_split
from sklearn import linear_model

#Key API Maps
bing = pybingmaps.Bing('Key API Bing')

#Porcentajes de Preferencia, dados por el usuario. En esta etapa fueron
definidos por el equipo
pref_Tiempo = 0.4
pref_Distancia = 0.15
pref_Consultorio = 0.15
pref_Medicos = 0.3
#Importando los datos y convirtiendolo en un dataframe
df_Hospitals = pd.read_excel("./Master_Dataset_AI_Saturday.xlsx")
# Juntando las coordenadas de los hospitales en un solo dataframe que es
utilizado por el algoritmo de Nearest Neighbors
df_LatitudHospitals = pd.DataFrame(df_Hospitals['LATITUD'])
df_LongitudHospitals = pd.DataFrame(df_Hospitals['LONGITUD'])
df_coordenadasHospitals = df_LatitudHospitals.join(df_LongitudHospitals)

Ev_hospitales = df_Hospitals

# Percentrank(Evaluacion) de los parametros Total de Consultorios y Medicos
Generales (Variables Estaticas)
```

```
Ev_hospitales['Ev_Consultorios'] = Ev_hospitales.reset_index() \
                                [['TOTAL DE CONSULTORIOS']] \
                                .apply(lambda x: (x.rank(method='dense') -
1) / (x.nunique() - 1) ) \
                                .values

Ev_hospitales['Ev_Medicos'] = Ev_hospitales.reset_index() \
                            [['TOTAL MEDICOS GENERALES Y ESPECIALISTAS']] \
                            .apply(lambda x: (x.rank(method='dense') - 1) /
(x.nunique() - 1) ) \
                            .values

# Se añade Calificacion al dataframe de evaluación la cual suma de estos
dos parametros multiplicado por
#su factor de preferencia
Ev_hospitales["Calificacion"] =( (Ev_hospitales['Ev_Consultorios']*
pref_Consultorio)

+(Ev_hospitales['Ev_Medicos']* pref_Medicos))
```

- **Entrenamiento Unsupervised Nearest Neighbors**

El entrenamiento sólo se realiza una vez al ejecutar la aplicación

```
#DOCUMENTACION K NEAREST NEIGHBORS
#https://scikit-learn.org/stable/modules/neighbors.html
#https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.Neares
tNeighbors.html#sklearn.neighbors.NearestNeighbors

#Solamente se seleccionaran los 10 neighbors mas cercanos
nbrs = NearestNeighbors(n_neighbors=15,
algorithm='auto').fit(df_coordenadasHospitals)
```

- **Entrenamiento Bayesian Ridge Regression**

El entrenamiento sólo se realiza una vez al ejecutar la aplicación

```
#DOCUMENTACION BAYESIAN RIDGE REGRESSION
#https://scikit-learn.org/stable/modules/linear_model.html#bayesian-regress
ion
X = Ev_hospitales[['TOTAL DE CONSULTORIOS','TOTAL MEDICOS GENERALES Y
ESPECIALISTAS']]
y = Ev_hospitales['Calificacion']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=417)
regr = linear_model.BayesianRidge()
reg = regr.fit(X_train, y_train)
y_pred = reg.predict(X_test)
```

- **Método de Evaluación**

Este es el método principal que recibe la posición y la salida es la lista de los mejores hospitales con su ranking

```
def funcionEvaluacion(df_my_location):
    #Entrenamiento no supervisado del algoritmo K Nearest Neighbor
    tic=timeit.default_timer()
    distances, indices = nbrs.kneighbors(df_my_location)
    df_Out = pd.DataFrame(indices)
    df_Out = df_Out.T
    df_Out = df_Out.rename({0: "Indice"}, axis='columns')
    toc=timeit.default_timer()
    print("Time Bayesian Ridge Regression",toc-tic)

    # Se empieza crear el dataframe final el cual va a contener toda la
informacion de salida
    # La API añade los tiempos de traslado y distancia al dataframe

    for i in df_Out.index:
        indice = df_Out.iat[i,0]
        df_Out.at[i, 'Latitud'] = df_Hospitals.iat[indice,4]
        df_Out.at[i, 'Longitud'] = df_Hospitals.iat[indice,5]
        df_Out.at[i, 'NOMBRE DE LA UNIDAD'] = df_Hospitals.iat[indice,8]
        df_Out.at[i, 'TOTAL DE CONSULTORIOS'] = df_Hospitals.iat[indice,9]
```

```
        df_Out.at[i, 'TOTAL MEDICOS GENERALES Y ESPECIALISTAS'] =
df_Hospitals.iat[indice,12]
        hospital_location = (df_Hospitals.iat[indice,4],
df_Hospitals.iat[indice,5])
        df_Out.at[i, 'Tiempo_Estimado(seg)'] = bing.travelTime(my_location,
hospital_location )
        df_Out.at[i, 'Distancia_Estimada(km)'] =
bing.travelDistance(my_location, hospital_location)

    #Se hace la prediccion de las calificaciones mediante el algoritmo de
Bayesian Ridge Regressor
    pred = df_Out[['TOTAL DE CONSULTORIOS','TOTAL MEDICOS GENERALES Y
ESPECIALISTAS']]
    df_Out["Calificacion"] = reg.predict(pred)

    # Haciendo el PercentRank de las variables que no son estaticas sobre
el data frame ya filtrado de los nearest neighbors
    df_Out['Ev_Tiempo'] = df_Out.reset_index() \
                        [['Tiempo_Estimado(seg)']] \
                        .apply(lambda x: (x.rank(method='dense') - 1) /
(x.nunique() - 1) ) \
                        .values

    df_Out['Ev_Distancia'] = df_Out.reset_index() \
                            [['Distancia_Estimada(km)']] \
                            .apply(lambda x: (x.rank(method='dense') - 1) /
(x.nunique() - 1) ) \
                            .values

    # Aquellos parametros que sean los menores seran los que tengan la
maxima calificacion
    df_Out["Ev_Tiempo"] = (-1 * df_Out["Ev_Tiempo"])+1
    df_Out["Ev_Distancia"] = (-1 * df_Out["Ev_Distancia"])+1

    # Sumando todas las calificaciones y multiplicandolas por su factor de
preferencia para obtener la calificacion final
    for i in df_Out.index:
        df_Out["Calificacion_Final"] =( df_Out['Ev_Tiempo'] * pref_Tiempo
                                      +df_Out['Ev_Distancia']*
pref_Distancia
                                      +df_Out['Calificacion']+0.3)
```

```
    df_Out = df_Out.drop(['Calificacion'], axis=1)
    df_Out = df_Out.drop(['Ev_Tiempo'], axis=1)
    df_Out = df_Out.drop(['Ev_Distancia'], axis=1)


    #Haciendo el rankeo final con las calificaciones finales obtenidas
    df_Out["Ranking"] = df_Out['Calificacion_Final'].rank(method='dense',
ascending = False)

    result = df_Out
    return result
```

- **Entrada de Parámetros**

Aquí el método principal recibe los argumentos de la posición y regresa la lista de los mejores hospitales

```
#Inicializacion de variables de tu posicion
my_latitude =20.677041
my_longitud =-103.347745
my_location = (my_latitude, my_longitud)
df_my_location = pd.DataFrame({'Latitud': [my_latitude],'Longitud':
[my_longitud]})
tic=timeit.default_timer()

df_Output = funcionEvaluacion(df_my_location)

toc=timeit.default_timer()

print("Tiempo de Busqueda del Mejor Hospital:",toc-tic)
```

- **Visualización**

Este código sirve para mostrar gráficamente la lista de los mejores hospitales

```
vissual = df_Output

df_LatitudHospitals = pd.DataFrame(vissual['Latitud'])
df_LongitudHospitals = pd.DataFrame(vissual['Longitud'])
```

```
vissual =df_LatitudHospitals .join(df_LongitudHospitals)
array_indices = vissual.to_numpy()

Sort = df_Output.sort_values(by='Calificacion_Final', ascending=False)

Selected = {'Latitud': [0.0,0.0,0.0], 'Longitud': [0.0,0.0,0.0]}
Selected = pd.DataFrame(Selected, columns = ['Latitud', 'Longitud'])
for i in range(3):
    Selected.at[i, 'Latitud'] = Sort.iat[i,1]
    Selected.at[i, 'Longitud'] = Sort.iat[i,2]
Top_Hospitals = Selected.to_numpy()

plt.subplots(figsize=(15, 15))
plt.scatter(df_Hospitals['LONGITUD'], df_Hospitals['LATITUD'],
cmap='viridis',label='Hospitals')
plt.scatter(array_indices[:, 1], array_indices[:, 0], c='green', s=100,
alpha=0.5,label='Selected Hospitals');
plt.scatter(my_longitud, my_latitude, c='red', s=100, alpha=0.5, label='My
Location');
plt.scatter(Top_Hospitals[:, 1], Top_Hospitals[:, 0], c='yellow', s=100,
alpha=0.5, label='Top Hospitals');

plt.suptitle('Latitud vs Longitud',fontsize=40)
plt.xlabel('Longitud', fontsize=20)
plt.ylabel('Latitud', fontsize=20)
plt.legend(loc='best')

plt.ylim(20.3, 20.9)
plt.xlim(-103.7, -103)
```

Referencias

1. Oche, M., & Adamu, H. (2013). Determinants of patient waiting time in the general outpatient department of a tertiary health institution in north Western Nigeria. *Annals of medical and health sciences research*, *3*(4), 588–592. https://doi.org/10.4103/2141-9248.122123
2. López Villalobos, I. (8 de abril de 2020). Call center de COVID-19 Jalisco ha recibido mil 394 llamadas. Noticias Udgtv. http://udgtv.com/noticias/call-center-covid-19-jalisco-ha-recibido-18-mil-394-llamadas/#

3. Gobierno de la Ciudad de México. (2020) App CMDX (Versión 1.35) [Aplicación Móvil] Descargado de: https://play.google.com/store/apps/details?id=mx.gob.cdmx.adip.apps&hl=es_MX
4. ArogyaOnline HealthCare. (2017). Hospital Finder (Versión 1.0.0) [Aplicación Móvil] Descargada de: https://play.google.com/store/apps/details?id=com.arogyaonline.myhospital&hl=es_MX
5. King Coder. (2018). Nearby Near Me Hospital (Versión 1.0.3) [Aplicación Móvil] Descargada de: https://play.google.com/store/apps/details?id=com.hospitalfinder.application&hl=es_MX
6. Darshan Institute of Engineering & Technology. (2017). Hospital Finder (Versión 1.0) [Aplicación Móvil] Descargada de: https://play.google.com/store/apps/details?id=com.aswdc_hospitalfinder&hl=es_MX
7. Datos abiertos gobierno Mexicano (2017). [Conjunto de datos]. Recuperado de https://datos.gob.mx/busca/dataset/recursos-en-salud-nivel-central
    a. Secretaria de Salud (2018). Recursos Salud Nivel central. 2020, de Secretaria de Salud Sitio web: https://datos.gob.mx/busca/dataset/recursos-en-salud-nivel-central
    b. Secretaria de Salud (2015). Tasa de Médicos por mil habitantes . 2020, de Secretaria de Salud Sitio web: https://datos.gob.mx/busca/dataset/indicadores-del-protocolo-de-san-salvador/resource/f2f69407-f5e2-4d9d-a3ab-fb13623d77e1
    c. Secretaria de Salud (2018). Egresos Hospitalarios. 2020, de Secretaria de Salud Sitio web: https://datos.gob.mx/busca/dataset/egresos-hospitalarios-de-la-secretaria-de-salud
    d. Secretaria de Salud (2018). Urgencias Hospitalarias. 2020, de Secretaria de Salud Sitio web: https://datos.gob.mx/busca/dataset/urgencias
8. Scikit-learn Documentation (2019). 1.6 Nearest Neighbors (2020) Sitio web: https://scikit-learn.org/stable/modules/neighbors.html