PONTIFICIA UNIVERSIDAD JAVERIANA

Ingeniería Electrónica

Inteligencia Artificial

Proyecto final Inteligencia Artificial

Presentador por:

Diego Alejandro López Rodríguez

diegoa.lopezr@javeriana.edu.co

Bogotá

2022

1. Introducción problema y con junto de datos:

Para el proyecto final se utilizó un ***Dataset*** relacionado con la predicción de Derrames cerebrales el cual colecciona entre sus columnas datos a considerar para poder asemejar posibles causas que puedan provocar este tipo de enfermedades en la vida de una persona, para este caso, se cuenta con 10 tipos de características en el dataset encontrado y dos tipos de etiqueta.

Entre sus características, como los datos que permitirán clasificar el riesgo o no de una persona que pueda sufrir un derrame cerebral, se encuentra:

- Género: Se encuentra dividido entre masculino y femenino
- Edad: Corresponde a la edad de cada uno de los sujetos a los cuales se les tomó esta variedad de datos.
- Hipertensión: esta característica se clasifica de manera binaria, ya que esta descrita como un "1" si la persona sufre de hipertensión, y "0" si la persona no sufre de ninguna manera de dicha afección.
- Enfermedad del corazón: al igual que la hipertensión, se referencia de la misma forma dando positivo a alguna enfermedad del corazón con "1" y de lo contrario "0".
- Alguna vez casado: No es más que un dato de tipo string que afirma con "Yes" si la persona ha estado casad@ o "No" de lo contrario.
- Tipo de trabajo: No podría faltar esta característica ya que se puede entender que según algún trabajo específico se pueden tener más posibilidades o menos de sufrir esta afección, ya que con trabajos más desgastantes y estresantes se puede sufrir de hipertensión y llevar directamente a una posición más cercana a clasificar en riesgo por un derrame cerebral; este se clasifica mediante 3 tipos los cuales son Privado, Gobierno o Independiente.
- Tipo de residencia entre Urbano y Rural
- Promedio del nivel de glucosa: En general: Menos de 100 mg/dL (5,6 mmol/L) se considera normal. Entre 100 y 125 mg/dL (5,6 a 6,9 mmol/L) se diagnostica como prediabetes.
- BMI: Esta medida indica que nos ayuda a saber si tenemos un peso correcto con respecto a nuestra estatura, por lo cual, este número se basa tanto en peso como en estatura, el cual para adultos mayores de 20 años se puede clasificar de la siguiente manera.

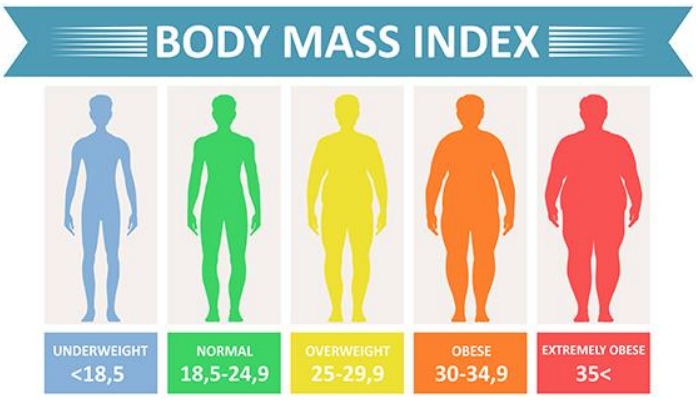

Fig 1. IBM mundial

- Estado de fumador: corresponde a un grupo de 4 posibilidades las cuales son Desconocido, Antiguo fumador, Fumado y No fumador.
- Stroke (Resultado): Indicia la posibilidad de sufrir un derrame cerebral mediante “1” como positivo, y “0” negativo no posible, el cual, será la columna tomada como etiquetas para clasificar cada una de las filas de datos en el riesgo descrito anteriormente.

El problema que se espera resolver es poder generar un modelo de aprendizaje supervisado de clasificación que permita predecir el riesgo que tiene una persona en su vida diaria de sufrir un derrame cerebral a partir del análisis de diferentes características personales y de salud de la misma como se pudo verificar anteriormente, y así poder tomar medidas para mejorar alguno de los parámetros descritos que lo catalogan entre ese grupo de persona con riesgo, algo que permita concientizar de igual manera a las personas sobre el peligro en el que se pueden encontrar debido a esto, en donde se le dé una prioridad a la salud de la persona con respecto a algunos cambios en su vida personal para no presentarse en el grupo de riesgo conocido como stroke (derrame cerebral).

2. Desarrollo del problema a partir de modelos:

Para el desarrollo de este problema a partir de los datos, se analizaron los modelos correspondientes que permitían actuar como clasificadores de aprendizaje supervisado, ya que se debía definir la etiqueta esencial de “Derrame” para un número determinado de características y se clasifica de forma binaria entre 1 y 0. Para ese problema se utilizaron 5 modelos los cuales corresponden a **Regresión Logística, Máquinas de soporte vectorial, Decision Tree, Random Forest y Knn** de la siguiente forma.

Antes de proceder a revisar cada modelo se realizó la lectura de del Dataset mostrado anteriormente en donde se puede ver la cantidad de datos correspondiente.

Lectura de datos

```
[152] stroke_data = pd.read_csv('Data/full_data.csv')
      stroke_data.head(-1)
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4975	Male	31.0	0	0	No	Private	Urban	215.07	32.7	smokes	0
4976	Male	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	Male	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	Female	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	Male	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0

4980 rows × 11 columns

Fig 2. Lectura datos

Posterior a esto se procedió a cambiar la etiqueta de stroke a Resultado lo cual es más sencillo para entender si el riesgo es positivo o no de sufrir un derrame cerebral.

Cambiar nombre columna

```
stroke_data = stroke_data.rename(columns={'stroke' : 'Derrame'})
stroke_data.head(-1)
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	Derrame
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4975	Male	31.0	0	0	No	Private	Urban	215.07	32.7	smokes	0
4976	Male	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	Male	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	Female	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	Male	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0

4980 rows × 11 columns

Fig 3. Cambio de etiqueta

Posterior a esto, se realiza la asignación de características y etiquetas, para esto, se toma cualquiera cosa que no corresponda a Resultado como una característica, y en el caso contrario, una etiqueta. Además, los datos referenciados como textos se separan para crear más columnas que se clasifiquen por valores binarios, como ejemplo a esto, si se tiene un género que agrupa Masculino y Femenino, este se separa creando las columnas ***gender_Male*** y ***gender_Female*** respectivamente, y esto mismo procedimiento se repite en cada columna que presente este tipo de opción, generando solo opciones binarias para el análisis.

Asignación con junto etiquetas y características

Separación en valores binarios de características textuales

```
[154] target = 'Derrame'
      X = stroke_data.loc[:,stroke_data.columns!=target]
      y = stroke_data.loc[:,stroke_data.columns==target]
      columnas = ['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
             'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
             'smoking_status']
      X = pd.get_dummies(X[columnas])
```

Fig 4. Asignaciones características y etiquetas

Al hacer esto, se puede realizar la partición de datos entre conjunto de entrenamiento y conjunto de validación, para este caso se utiliza una semilla (***random_state = 0***) junto con la partición por defecto de 70% entrenamiento y 30% validación, además se asigna el conjunto de características ***X*** y la etiqueta ***y***. Para esto, se procede posteriormente a normalizar teniendo en cuenta ambos conjuntos a partir de los momentos iniciales para normalizar (***standarscaler***).

Particición por defecto conjunto de validación y entrenamiento

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

#partición del conjunto de muestras en validación y entrenamiento
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
print(X_train.shape)
print(X_test.shape)
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
(3735, 19)
(1246, 19)
```

Fig 5. Partición del conjunto de datos

Con esto ya se puede revisar cada uno de los modelos nombrados anteriormente, los cuales se permitirán entrenar el conjunto de entrenamiento referenciado y así poder realizar la respectiva predicción.

Regresión Logística:

La regresión logística es una técnica de análisis de datos que utiliza las matemáticas para encontrar las relaciones entre dos factores de datos. Luego, utiliza esta relación para predecir el valor de uno de esos factores basándose en el otro. Normalmente, la predicción tiene un número finito de resultados, como un sí o un no. La regresión logística es un modelo estadístico para estudiar las relaciones entre un conjunto de variables cualitativas Xi y una variable cualitativa Y. Se trata de un modelo lineal generalizado que utiliza una función logística como función de enlace.

Un modelo de regresión logística también permite predecir la probabilidad de que ocurra un evento (valor de 1) o no (valor de 0) a partir de la optimización de los coeficientes de regresión. Este resultado siempre varía entre 0 y 1. Cuando el valor predicho supera un umbral, es probable que ocurra el evento, mientras que cuando ese valor está por debajo del mismo umbral, no es así. [1]

SVM:

Es un algoritmo de aprendizaje automático supervisado que se utiliza tanto para la clasificación como para la regresión. Aunque decimos que los problemas de regresión también son los más adecuados para la clasificación. El objetivo del algoritmo SVM es encontrar un hiperplano en un espacio N-dimensional que clasifique claramente los puntos de datos. La dimensión del hiperplano depende del número de características. Si el número de entidades de entrada es dos, entonces el hiperplano es solo una línea. Si el número de entidades de entrada es tres, el hiperplano se convierte en un plano 2D. Se vuelve difícil de imaginar cuando el número de funciones supera las tres. [2]

Decision Tree:

El modelo Decision Tree es una técnica de aprendizaje supervisado que se puede utilizar tanto para problemas de clasificación como de regresión, pero se prefiere sobre todo para resolver problemas de clasificación. Es un clasificador con estructura de árbol, donde los nodos internos representan las

características de un conjunto de datos, las ramas representan las reglas de decisión y cada nodo hoja representa el resultado. En este modelo, hay dos nodos, que son el nodo de decisión y el nodo hoja. Los nodos de decisión se utilizan para tomar cualquier decisión y tienen múltiples ramas, mientras que los nodos de hoja son el resultado de esas decisiones y no contienen más ramas. [3]

Random Forest:

Los bosques aleatorios son un algoritmo de aprendizaje supervisado. Se puede utilizar tanto para clasificación como para regresión. También es el algoritmo más flexible y fácil de usar. Un bosque está compuesto de árboles. Se dice que cuantos más árboles tiene, más robusto es un bosque. Los bosques aleatorios crean árboles de decisión en muestras de datos seleccionadas al azar, obtienen predicciones de cada árbol y seleccionan la mejor solución mediante votación. También proporciona un indicador bastante bueno de la importancia de la función. Técnicamente es un método de conjunto (basado en el enfoque divide y vencerás) de árboles de decisión generados en un conjunto de datos dividido aleatoriamente. Esta colección de clasificadores de árboles de decisión también se conoce como el bosque. Los árboles de decisión individuales se generan utilizando un indicador de selección de atributos. [4]

Knn:

Es un método que simplemente busca en las observaciones más cercanas a la que se está tratando de predecir y clasifica el punto de interés basado en la mayoría de los datos que le rodean. En K-Nearest Neighbor la "K" significa la cantidad de "puntos vecinos" que tenemos en cuenta en las cercanías para clasificar los "n" grupos -que ya se conocen de antemano, pues es un algoritmo supervisado. Puede usarse para clasificar nuevas muestras (valores discretos) o para predecir (regresión, valores continuos). Al ser un método sencillo, es ideal para introducirse en el mundo del Aprendizaje Automático. Sirve esencialmente para clasificar valores buscando los puntos de datos "más similares" (por cercanía) aprendidos en la etapa de entrenamiento y haciendo conjeturas de nuevos puntos basado en esa clasificación. [5]

3. Resultados con respecto a la evaluación de cada modelo:

Para evaluar cada uno de los modelos se implementó una función que a partir del modelo, permitiera obtener métricas de evaluación como Accuracy, ROC AUC, MCC y f1 score, con el fin de poder dar a conocer el mejor modelo modelo a partir de los valores de estas métricas por medio de sus resultados y de manera gráfica.

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import f1_score

def run_model(model, X_train, y_train, X_test, y_test):
  #Entrenamiento del modelo especificado y predicción
  model.fit(X_train,y_train)
  y_pred = model.predict(X_test)
```

```
  #Accuracy metric
  accuracy = accuracy_score(y_test, y_pred)
  #ROC AUC metric
  roc_auc = roc_auc_score(y_test, y_pred)
  #MCC Metric
  MCC = matthews_corrcoef(y_test, y_pred)
  #F1 score metric
  F1 = f1_score(y_test, y_pred, average='micro')

  model_ev = pd.DataFrame({'Metric': ['ACC','ROC AUC','MCC','f1 score'
],
                           'Score': [accuracy,roc_auc,MCC,F1]})

  print("Accuracy = {}".format(accuracy))
  print("ROC Area under Curve = {}".format(roc_auc))
  print("MCC = {}".format(MCC))
  print("F1 SCORE = {}".format(F1))
  print("--------------------------------")
  print("\n")

  return model, accuracy, roc_auc, MCC, F1, model_ev
```

Con esto, se evaluó el modelo de **Regresión Logística:**

```
from sklearn.linear_model import LogisticRegression

model_LR = LogisticRegression(penalty='l2',max_iter=1000, C=1000,rando
m_state=0)
print('Logistic Regression Metrics:')
model_LR, accuracy_LR, roc_auc_LR, MCC_LR, F1_LR, model_ev_LR = run_mo
del(model_LR, X_train, y_train, X_test, y_test)
#Visualize
plt.figure(figsize=(6,6))
plt.title("Represent Metrics of Logistic Regression model", fontweight
 = 'bold', fontsize = 20)
plt.ylabel("Score %", fontweight = 'bold', fontsize = 15)
plt.xlabel("Metrics", fontweight = 'bold', fontsize = 15)
plt.xticks(rotation=45)
plt.bar(model_ev_LR['Metric'],model_ev_LR['Score'], color ='teal', wid
th = 0.5)
plt.show()
```

Obteniendo los siguientes resultados:

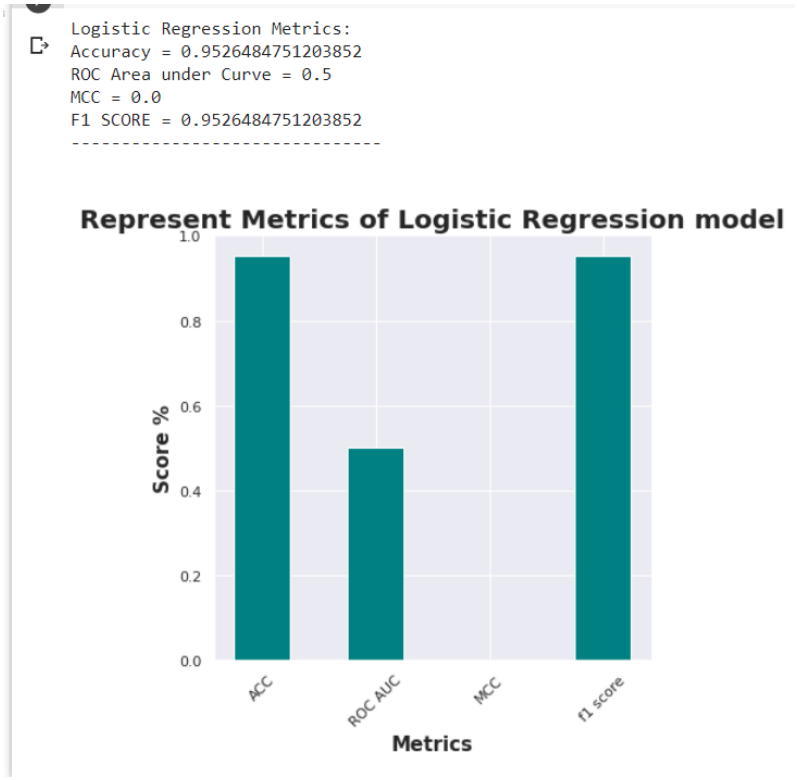

Fig 6. Resultados evaluación modelo Regresión Logística

Donde el valor más alto corresponde al Accuracy el cual comparte resultados con el f1 score e indicia una buena medida.

Posterior a esto, se evaluó el modelo de **Máquinas de soporte vectorial** tomando en cuenta un kernel que definía si era de tipo lineal, cuadrático y RBF, como se puede verificar a continuación.

```
from sklearn import svm
kernels=['linear', 'poly', 'rbf', 'sigmoid']
#SVM LINEAL
Kernel=0
model_svm_lineal = svm.SVC(kernel=kernels[Kernel],gamma=0.01)
print('SMV LINEAL METRICS')
model_svm_lineal, accuracy_svm_lineal, roc_auc_svm_lineal, MCC_svm_lin
eal, F1_svm_lineal, model_ev_svm_lineal = run_model(model_svm_lineal,
X_train, y_train, X_test, y_test)

#SVM POLINOMIAL CUADRATICO
Kernel=1
model_svm_C = svm.SVC(kernel=kernels[Kernel],degree=4,coef0=1)
print('SMV POLINOMIAL METRICS')
```

```
model_svm_C, accuracy_svm_C, roc_auc_svm_C, MCC_svm_C, F1_svm_C, model
_ev_svm_C = run_model(model_svm_C, X_train, y_train, X_test, y_test)

#SVM RBF
Kernel=2
model_svm_RBF = svm.SVC(kernel=kernels[Kernel],gamma=(1/30)*100)
print('SMV RBF METRICS')
model_svm_RBF, accuracy_svm_RBF, roc_auc_svm_RBF, MCC_svm_RBF, F1_svm_
RBF, model_ev_svm_RBF = run_model(model_svm_RBF, X_train, y_train, X_t
est, y_test)

#Visualize
# set width of bar
barWidth = 0.25
plt.figure(figsize=(10,10))
plt.title("Represent Metrics of SVM model", fontweight = 'bold', fonts
ize = 20)
plt.xticks(rotation=45)

# Set position of bar on X axis
br1 = np.arange(len(model_ev_svm_lineal['Score']))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]

# Make the plot
plt.bar(br1, model_ev_svm_lineal['Score'], color ='r', width = barWidt
h,
    edgecolor ='grey', label ='SVM Lineal')
plt.bar(br2, model_ev_svm_C['Score'], color ='g', width = barWidth,
    edgecolor ='grey', label ='SVM Polinomial')
plt.bar(br3, model_ev_svm_RBF['Score'], color ='b', width = barWidth,
    edgecolor ='grey', label ='SVM RBF')

# Adding Xticks
plt.xlabel('Metrics', fontweight ='bold', fontsize = 15)
plt.ylabel('Score %', fontweight ='bold', fontsize = 15)
plt.xticks([r + barWidth for r in range(len(model_ev_svm_lineal['Score
'']))],
    ['ACC','ROC AUC','MCC','f1 score'])
plt.legend()
plt.show()
```

Esto permite evaluar cada uno de sus tipos referenciados y así, poder comparar el mejor de manera gráfica y visual.

```
Accuracy = 0.9526484751203852
ROC Area under Curve = 0.5
MCC = 0.0
F1 SCORE = 0.9526484751203852
--------------------------------


SMV POLINOMIAL METRICS
Accuracy = 0.9454253611556982
ROC Area under Curve = 0.5123156226350435
MCC = 0.0514854507927018
F1 SCORE = 0.9454253611556982
--------------------------------


SMV RBF METRICS
Accuracy = 0.9510433386837881
ROC Area under Curve = 0.4991575400168492
MCC = -0.008939339966957898
F1 SCORE = 0.9510433386837881
--------------------------------
```

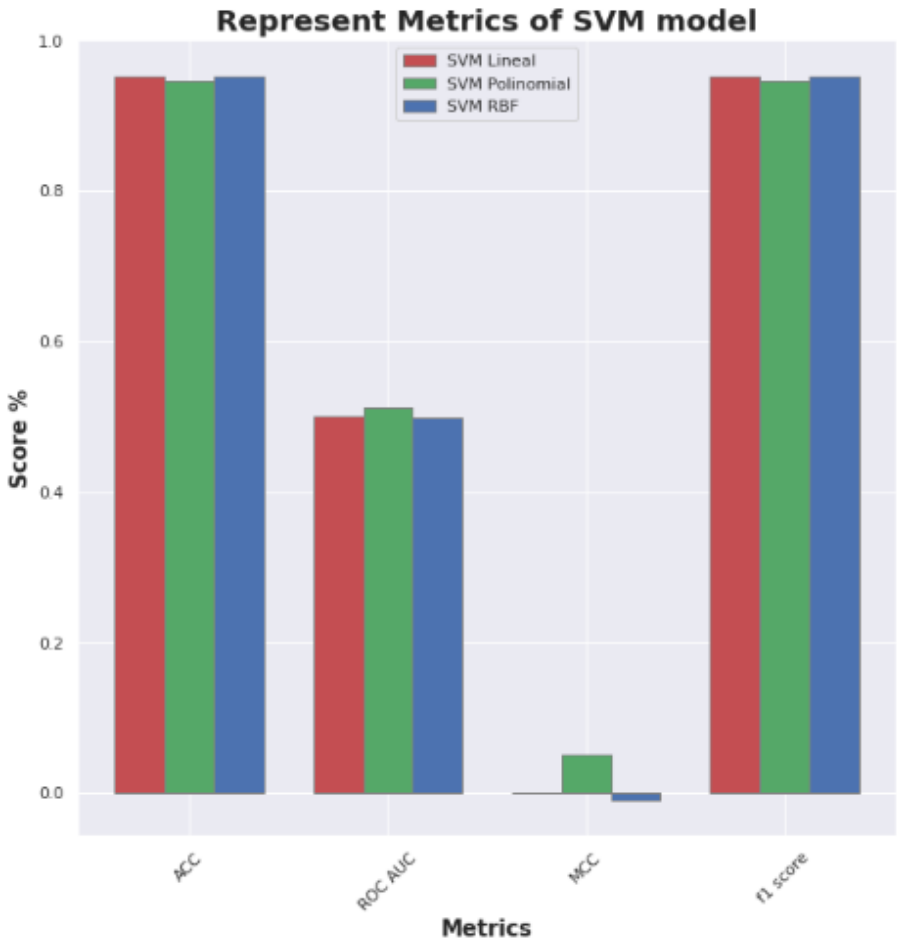

Fig 7. Resultados evaluación modelo SVM

Como se puede observar tanto su Accuracy como f1 score presentan valores parecidos para cada modelo específico, pero solo se logran diferenciar en la métrica del coeficiente de correlación de Matthews, el cual se utiliza en el aprendizaje automático como medida de la calidad de las clasificaciones binarias y multiclase, donde el MCC es en esencia un valor de coeficiente de correlación entre -1 y +1. Un coeficiente de +1 representa una predicción perfecta, 0 una predicción aleatoria promedio y -1 una predicción inversa. Por tal motivo, el mejor modelo para las máquinas de soporte vectorial corresponde a SVM polinomial cuadrático, ya que su valor de coeficiente es el más cercano a tener una predicción perfecta.

Después se procedió a evaluar el modelo **Decision Tree** de la siguiente forma.

```
from sklearn.tree import DecisionTreeClassifier

model_DT = DecisionTreeClassifier(random_state=0)
print('Decision Tree Metrics:')
model_DT, accuracy_DT, roc_auc_DT, MCC_DT, F1_DT, model_ev_DT = run_mo
del(model_DT, X_train, y_train, X_test, y_test)
#Visualize
plt.figure(figsize=(6,6))
plt.title("Represent Metrics of Decision Tree model", fontweight = 'bo
ld', fontsize = 20)
plt.ylabel("Score %", fontweight = 'bold', fontsize = 15)
plt.xlabel("Metrics", fontweight = 'bold', fontsize = 15)
plt.xticks(rotation=45)
plt.bar(model_ev_DT['Metric'],model_ev_DT['Score'], color ='y', width
= 0.5)
plt.show()
```

Dando como resultados la siguiente comparación de métricas:

```
Decision Tree Metrics:
Accuracy = 0.9109149277688604
ROC Area under Curve = 0.5264161181157454
MCC = 0.050833689692238455
F1 SCORE = 0.9109149277688604
--------------------------------
```

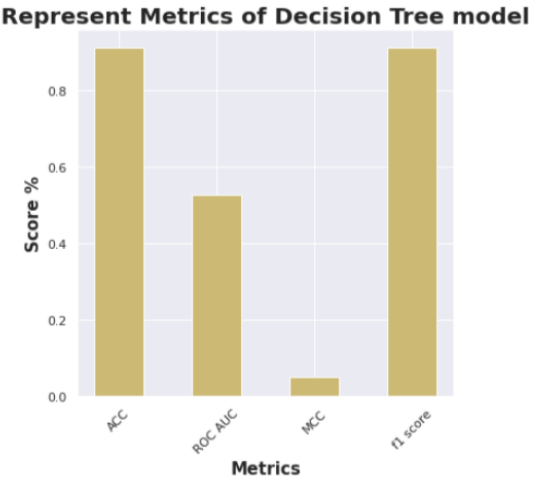

Fig 8. Resultados evaluación modelo Decision Tree

Con valores de Accuracy y f1 score más bajos que los anteriores modelos analizados.

Seguido a este, se analizó el modelo **Random Forest** de la siguiente manera.

```
from sklearn.ensemble import RandomForestClassifier

model_RF = RandomForestClassifier(max_depth=16, min_samples_leaf=1, mi
n_samples_split=2, n_estimators=100, random_state=0)
print('Random Forest Metrics:')
model_RF, accuracy_RF, roc_auc_RF, MCC_RF, F1_RF, model_ev_RF = run_mo
del(model_RF, X_train, y_train, X_test, y_test)
#Visualize
plt.figure(figsize=(6,6))
plt.title("Represent Metrics of Randon Forest model", fontweight = 'bo
ld', fontsize = 20)
plt.ylabel("Score %", fontweight = 'bold', fontsize = 15)
plt.xlabel("Metrics", fontweight = 'bold', fontsize = 15)
plt.xticks(rotation=45)
plt.bar(model_ev_RF['Metric'],model_ev_RF['Score'], color ='lime', wid
th = 0.5)
plt.show()
```

Obteniendo los siguientes resultados.

```
Random Forest Metrics:
Accuracy = 0.9518459069020867
ROC Area under Curve = 0.5076321162880356
MCC = 0.0661500456500721
F1 SCORE = 0.9518459069020866
--------------------------------
```

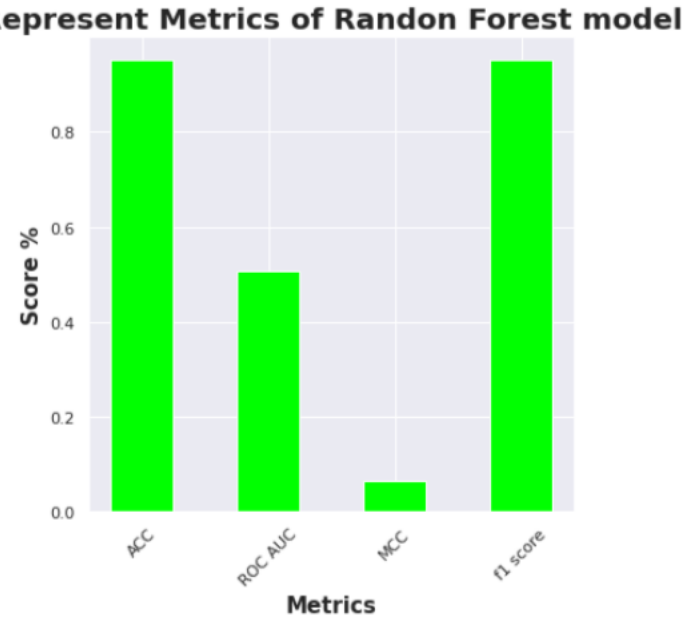

Fig 9. Random Forest

Por último, se evaluó el modelo de **KNN** primero evaluando el mejor K a partir de las métricas utilizadas anteriormente.

```
from sklearn.neighbors import KNeighborsClassifier

k_range = range(1, int(np.sqrt(len(y_train))))
print(k_range)
distance='minkowski'#podemos hacer un for que recorra las distancias q
ue queremos probar en un enfoque grid-search.

accuracy_knn=[]
roc_auc_knn=[]
MCC_knn=[]
F1_knn=[]

for k in k_range:#por ahora variemos K,
    model_knn = KNeighborsClassifier(n_neighbors = k,weights='distance
',metric=distance, metric_params=None,algorithm='brute')
    model_knn.fit(X_train, y_train)
```

```
    y_pred=model_knn.predict(X_test)

    #Accuracy metric
    accuracy_knn.append(accuracy_score(y_test, y_pred))
    #ROC AUC metric
    roc_auc_knn.append(roc_auc_score(y_test, y_pred))
    #MCC Metric
    MCC_knn.append(matthews_corrcoef(y_test, y_pred))
    #F1 score metric
    F1_knn.append(f1_score(y_test, y_pred, average='micro'))
```

Con esto se graficaron los resultados de cada métrica como se puede ver a continuación.

```
#Comaparación mejor k accuracy
#print(accuracy_knn)
print("Max Accuracy:", max(accuracy_knn))
plt.figure()
plt.plot(k_range, accuracy_knn, color='aqua',lw=3)
plt.xlabel('K', fontweight = 'bold', fontsize = 15)
plt.ylabel('Accuracy %', fontweight = 'bold', fontsize = 15)
plt.title('ACC metric frome KNN model', fontweight = 'bold', fontsize
= 20)
plt.show()

#Comaparación mejor k ROC AUC
#print(roc_auc_knn)
print("Max AUC ROC:", max(roc_auc_knn))
plt.figure()
plt.plot(k_range, roc_auc_knn, color='green',lw=3)
plt.xlabel('K', fontweight = 'bold', fontsize = 15)
plt.ylabel('ROC AUC %', fontweight = 'bold', fontsize = 15)
plt.title('ROC AUC metric frome KNN model', fontweight = 'bold', fonts
ize = 20)
plt.show()

#Comaparación mejor k MCC
#print(MCC_knn)
print("Max MCC:", max(MCC_knn))
plt.figure()
plt.plot(k_range, MCC_knn, color='deeppink',lw=3)
plt.xlabel('K', fontweight = 'bold', fontsize = 15)
plt.ylabel('MCC %', fontweight = 'bold', fontsize = 15)
```

```
plt.title('MCC metric frome KNN model', fontweight = 'bold', fontsize
= 20)
plt.show()

#Comaparación mejor k MCC
#print(F1_knn)
print("Max f1 score:", max(F1_knn))
plt.figure()
plt.plot(k_range, F1_knn, color='k',lw=3)
plt.xlabel('K', fontweight = 'bold', fontsize = 15)
plt.ylabel('f1 score %', fontweight = 'bold', fontsize = 15)
plt.title('f1 score metric frome KNN model', fontweight = 'bold', font
size = 20)
plt.show()
```

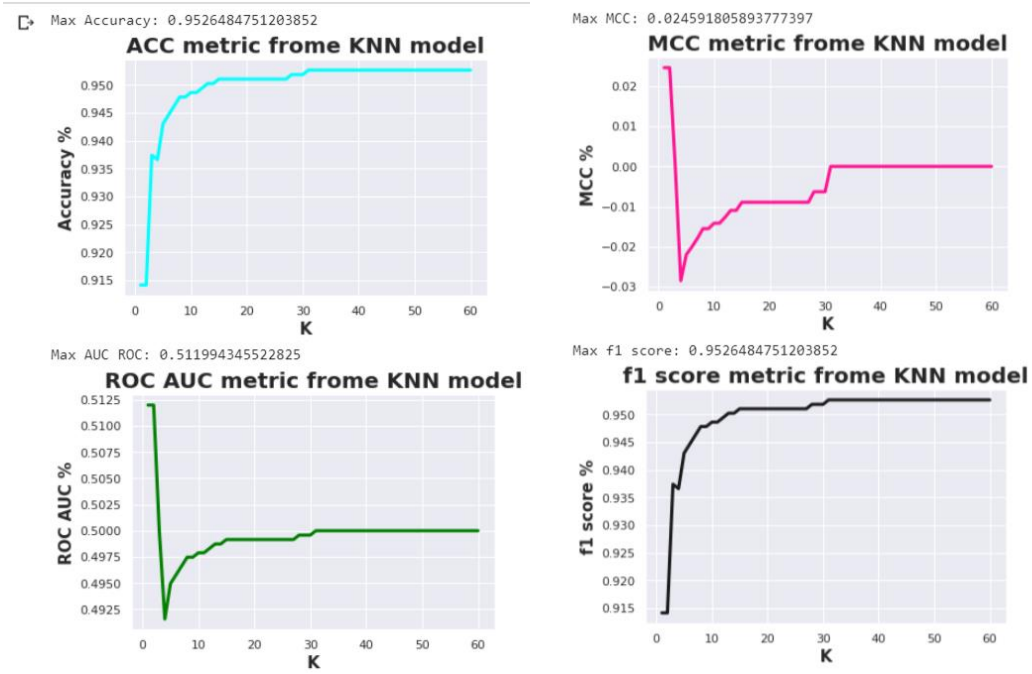

Fig 10. Análisis k ideal Knn

Como se puede verificar, lo ideal sería tomar un K por arriba de 30 ya que permite las mejores mediciones de Accuracy y f1 score, aunque no dan los mejores valores para la métrica de curva característica operativa del receptor (ROC AUC) junto con el MCC, generan valores que no se encuentran en una escala depreciable, por tal motivo, se tomará un K de 32 según los anteriores resultados. Con esto, se obtiene nuevamente las métricas para ese K en particular.

```
from sklearn.neighbors import KNeighborsClassifier
distance='minkowski'#podemos hacer un for que recorra las distancias q
ue queremos probar en un enfoque grid-search.
```

```
model_knn = KNeighborsClassifier(n_neighbors = 32,weights='distance',m
etric=distance, metric_params=None,algorithm='brute')
print('KNN Metrics:')
model_knn, accuracy_knn, roc_auc_knn, MCC_knn, F1_knn, model_ev_knn =
run_model(model_knn, X_train, y_train, X_test, y_test)

#Visualize
plt.figure(figsize=(6,6))
plt.title("Represent Metrics of KNN model", fontweight = 'bold', fonts
ize = 20)
plt.ylabel("Score %", fontweight = 'bold', fontsize = 15)
plt.xlabel("Metrics", fontweight = 'bold', fontsize = 15)
plt.xticks(rotation=45)
plt.bar(model_ev_RF['Metric'],model_ev_RF['Score'], color ='orange', w
idth = 0.5)
plt.show()
```

Los resultados de análisis son los siguientes:

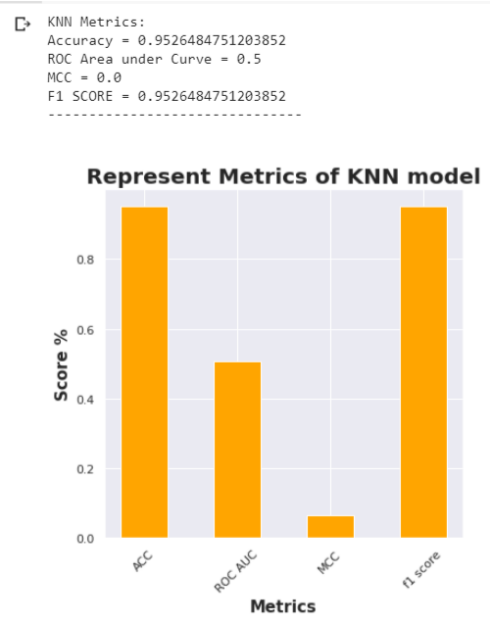

Fig 11. Resultados evaluación modelo KNN

Por último, se procede a realizar el análisis gráfico de cada una de las métricas con respecto al modelo referenciado de los cuales se han visto anteriormente.

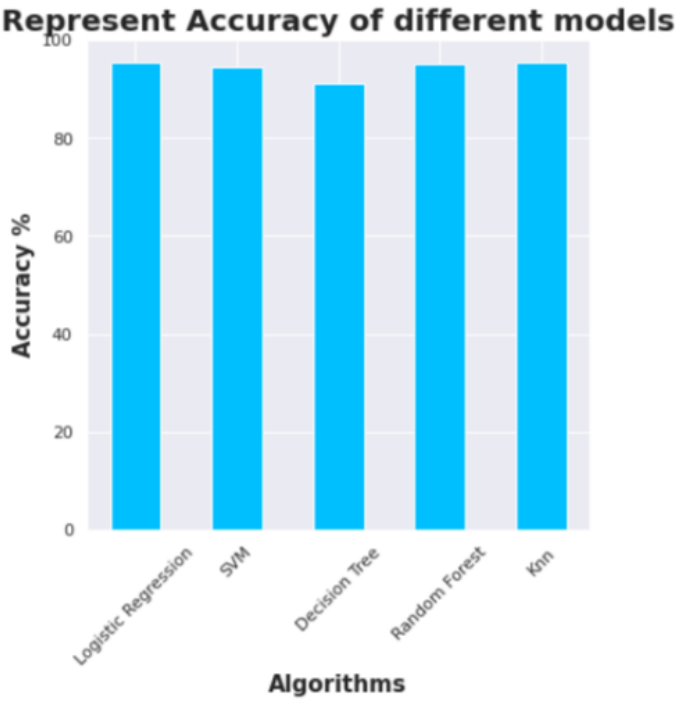

Fig 12. Representación Accuracy para todos los modelos

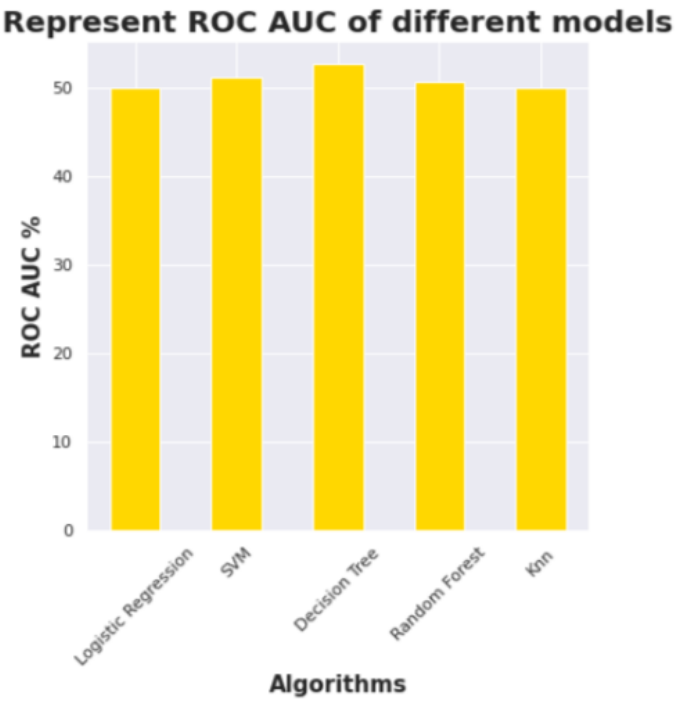

Fig 13. Representación ROC AUC para todos los modelos

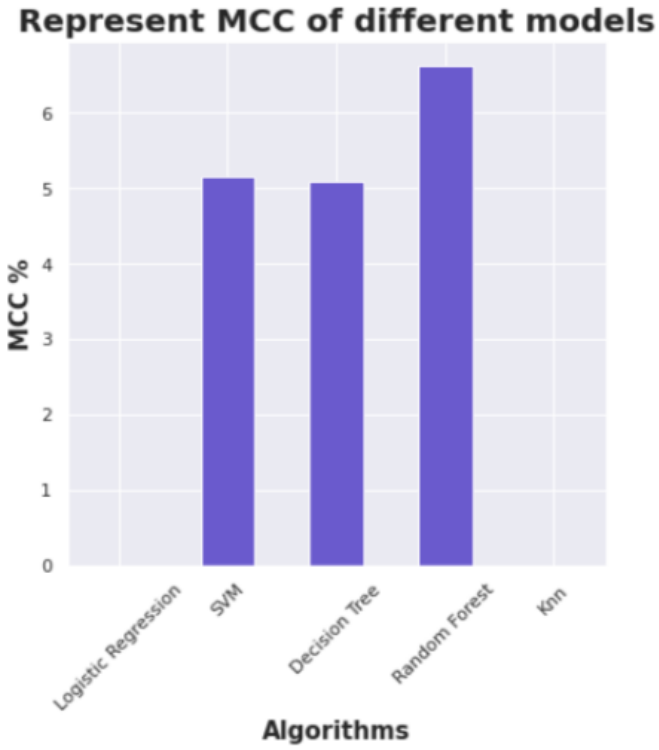

Fig 14. Representacion MCC para todos los modelos

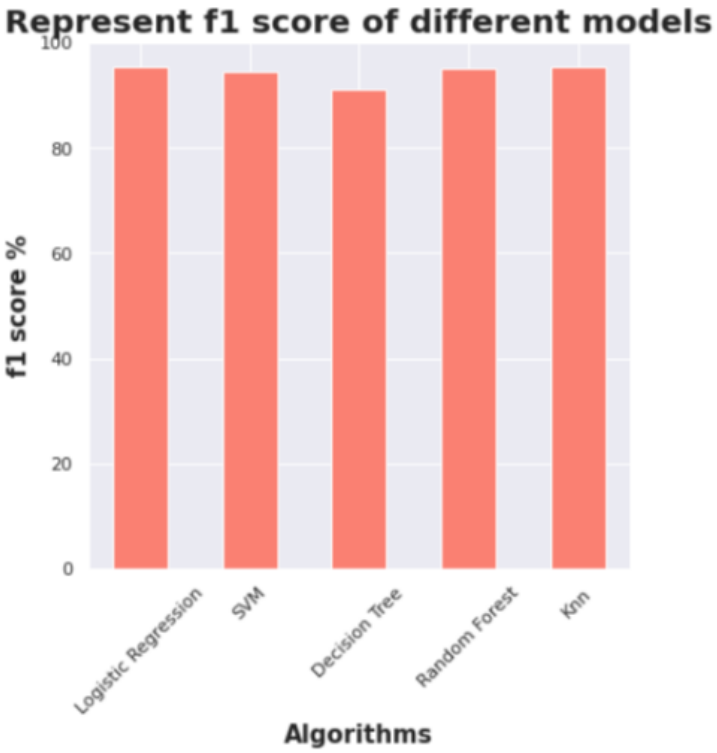

Fig 15. Representación f1 score para todos los modelos

4. Conclusiones:

Según todos los resultados presentados anteriormente, se pueden dar a conocer varias cosas, una de ellas corresponde al Accuracy de cada uno de los modelos presentados, siendo el de la Regresión Logística de los más altos con aproximadamente un valor del 95% lo que puede llegar a indicar que presenta 95 predicciones correctas de un total de 100 ejemplos). Eso significa que nuestro clasificador de Derrames está haciendo un buen trabajo en la identificación de personas con o sin riesgo, pero esto se puede cumplir para cada uno de los clasificadores ya que todos se encuentran por arriba del 90% de Accuracy. De igual manera se pueden comparar los modelos con respecto a la métrica ROC AUC que permite verificar el rendimiento de cualquier modelo de clasificación, lo que quiere decir que, permite medir el acierto en la predicción de eventos binarios, es decir, eventos que bien ocurren o no ocurren, lo que se puede observar que todos no suben a más del 60% por lo cual no corresponde en gran acierto esta métrica para cada uno de ellos, pero una de las mejores es de Decision Tree con un valor de aproximadamente 53%.

En adición a esto, como ya se dijo antes, para la métrica MCC entre más cercano se encuentre al valor de 1 es mejor ya que representa una predicción perfecta, lo cual no es muy acertado ya que todos los clasificadores cuentan con valores que tienden evidentemente a 0 indicando una predicción aleatoria promedio, de los cuales, el que mejor responde es el modelo Random Forest según los resultados mostrados anteriormente. Por último, el igual que el Accuracy obtenido, el f1 score presenta resultados muy parecidos, ya que se puede interpretar como una media armónica de precisión y recuperación, donde una puntuación F1 alcanza su mejor valor en 1 y su peor puntuación en 0, de las cuales todas pueden cumplir con este requisito para clasificar en buena medida.

A partir de los resultados obtenidos, se pudo evidenciar como este problema se puede resolver a partir de un método de clasificación supervisado ya que permite hacer un buen análisis con respecto a las posibilidades binarias que se presentan en cada una de las etiquetas, lo que facilita la medición de las métricas específicas y así, se pueden obtener de manera más factible los resultados con respecto a la etiquetas referenciadas en este caso la posibilidad de sufrir o no un Derrame, lo cual se dejó en evidencia a lo largo del presente informe dando la posibilidad abierta de utilizar más de un modelo como el óptimo para realizar la predicción y verificar sus resultados.

Referencias

[1] «DataScientist,» [En línea]. Available: https://datascientest.com/es/que-es-la-regresion-logistica. [Último acceso: 26 Noviembre 2022].

[2] «Geeksforgeeks,» 24 Noviembre 2022. [En línea]. Available: https://www.geeksforgeeks.org/support-vector-machine-algorithm/. [Último acceso: 26 Noviembre 2022].

[3] [En línea]. Available: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm. [Último acceso: 26 Noviembre 2022].

[4] «Datacamp,» [En línea]. Available: https://www.datacamp.com/tutorial/random-forests-classifier-python. [Último acceso: 26 Noviembre 2022].

[5] «Aprendemachinelearning,» 10 Julio 2018. [En línea]. Available: https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/. [Último acceso: 26 Noviembre 2022].