Proyecto Final Inteligencia Artificial - Aplicación de clasificadores binarios para la predicción de ataques al corazón

Juan Sebastian Corredor Angarita, Juan Diego Llano Miraval
juan_corredor@javeriana.edu.co, llanojuan@javeriana.edu.co

Abstract—Este proyecto consiste en comparar 3 clasificadores binarios en diferentes configuraciones para determinar cual tiene un comportamiento más óptimo. Para realizar el entrenamiento de cada uno se parte de una base de datos extraída de Kaggle, la cual relaciona diferentes características del paciente como la edad, sexo, colesterol, azúcar en sangre, entre otras. La predicción final debe retornar un valor de 0 si el paciente no es propenso a sufrir un paro cardiaco, y 1 si el paciente es propenso a sufrirlo.

Index Terms—Classification algorithms, K Nearest Neighbor, Logistic Regression, Machine Learning, Neural Networks, Probability Computing.

1 Introducción

Este proyecto emplea tres tipos de clasificador: Regresión logística multivariada, K vecinos más cercanos y redes neuronales. Cada clasificador utilizará la misma base de datos extraída de Kaggle, un ejemplo de esta se presenta a continuación.

age	sex	cp	trestbps	chol	fbs	resteeg	thalach	exang	oldpeak	slope	ca	thal	target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
44	1	1	120	263	0	1	173	0	0	2	0	3	1
52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
54	1	0	140	239	0	1	160	0	1.2	2	0	2	1

Fig. 1: Datos existentes en la base extraída de Kaggle.

Los datos mostrados en la Figura 1 corresponden a:

1) edad
2) sexo
3) tipo de dolor de pecho (4 opciones)
4) presión arterial en reposo
5) colesterol sérico en mg / dl
6) azúcar en sangre en ayunas menor a 120 mg / dl
7) resultados electrocardiográficos en reposo (valores 0,1,2)
8) frecuencia cardíaca máxima alcanzada
9) angina inducida por ejercicio
10) oldpeak: depresión del ST inducida por el ejercicio en relación con el reposo
11) la pendiente del segmento ST de ejercicio pico
12) número de vasos principales (0-3) coloreados por la floración
13) thal: 0 = normal; 1 = defecto fijo; 2 = defecto reversible
14) objetivo: 0 = menos probabilidad de ataque cardíaco 1 = más probabilidad de ataque cardíaco

Las etiquetas de esta base de datos serán el numeral 14, correspondiente a 1 si el paciente es propenso a un paro cardiaco o 0 si no lo es. El objetivo es entrenar 3 algoritmos supervisados, observando las etiquetas a la salida mediante conjuntos de entrenamiento y validación. Los 3 algoritmos serán los siguientes:

1) Regresión lineal
2) K vecinos más cercanos
3) Perceptron multicapa

Para cada uno de los algoritmos se define la matriz preprocesada con la cual se entrenarán, la división entre entrenamiento y validación, la selección de hiperparámetros y cuales serán los resultados que calificarán el funcionamiento de cada modelo.

2 Desarrollo

Antes de entrenar cualquier algoritmo se prepara la matriz de entrada a cada uno. Se leen los datos del archivo CSV adquirido y se separa la salida Y de la matriz X, de esta manera se conocen las salidas para cada fila de datos que serán el target, y las entradas que será la matriz X, la cual es la misma mostrada en la Figura 1 sin la última columna.

2.1 Preprocesamiento

Una vez leído el archivo CSV, siguen unos pasos generales que se aplicarán a la matriz de entrada para obtener los datos que serán entregados a cada algoritmo. Al observar nuevamente la matriz generada se evidencia que esta no requiere un preprocesamiento complejo, ya que este dataset contiene absolutamente todos los datos necesarios en cada fila y columna, además de esto, todos los datos son numéricos, lo que facilita el no tener que reemplazar letras o palabras por números. Se define entonces la escalización para este dataset, se elige escalizar por mínimo y máximo de cada columna entre los rangos de -1 a 1.

age	sex	cp	trestbps	chol	fbs	estec	thalach	exang	oldpeak	slop	ca	thal
0.41666667	1	1	-0.03773585	-0.51141553	1	-1	0.20610687	-1	-0.25806452	-1	-1	-0.33333333
-0.66666667	1	0.33333333	-0.32075472	-0.43378995	-1	0	0.77099237	-1	0.12903226	-1	-1	0.33333333
-0.5	-1	-0.33333333	-0.32075472	-0.64383562	-1	-1	0.54198473	-1	-0.5483871	1	-1	0.33333333
0.125	1	-0.33333333	-0.50943396	-0.49771689	-1	0	0.63358779	-1	-0.74193548	1	-1	0.33333333
0.16666667	-1	-1	-0.50943396	0.04109589	-1	0	0.40458015	1	-0.80645161	1	-1	0.33333333
0.16666667	1	-1	-0.13207547	-0.69863014	-1	0	0.17557252	-1	-0.87096774	0	-1	-0.33333333
0.125	-1	-0.33333333	-0.13207547	-0.23287671	-1	-1	0.2519084	-1	-0.58064516	0	-1	0.33333333
-0.375	1	-0.33333333	-0.50943396	-0.37442922	-1	0	0.55725191	-1	-1	1	-1	1
-0.04166667	1	0.33333333	0.47169811	-0.66666667	1	0	0.38931298	-1	-0.83870968	1	-1	1
0.16666667	1	0.33333333	0.05660377	-0.80821918	-1	0	0.57251908	-1	-0.48387097	1	-1	0.33333333
0.04166667	1	-1	-0.13207547	-0.48401826	-1	0	0.35877863	-1	-0.61290323	1	-1	0.33333333
-0.20833333	-1	0.33333333	-0.32075472	-0.3196347	-1	0	0.03816794	-1	-0.93548387	1	-1	0.33333333
-0.16666667	1	-0.33333333	-0.32075472	-0.36073059	-1	0	0.52671756	-1	-0.80645161	1	-1	0.33333333
0.45833333	1	1	-0.69811321	-0.61187215	-1	-1	0.11450382	1	-0.41935484	0	-1	0.33333333
0.20833333	-1	1	0.05660377	-0.28310502	1	-1	0.38931298	-1	-0.67741935	1	-1	0.33333333
-0.125	-1	0.33333333	-0.50943396	-0.57534247	-1	0	0.32824427	-1	-0.48387097	0	-1	0.33333333
0.20833333	-1	0.33333333	-0.50943396	-0.02283105	-1	0	0.54198473	-1	-1	1	-1	0.33333333
0.54166667	-1	1	0.05660377	-0.543379	-1	0	-0.34351145	-1	-0.16129032	-1	-1	0.33333333
-0.41666667	1	-1	0.05660377	-0.44748858	-1	0	0.52671756	-1	-0.51612903	1	-1	0.33333333
0.66666667	-1	1	-0.13207547	-0.48401826	-1	0	0.22137405	-1	-0.41935484	1	0	0.33333333
0.25	1	-1	-0.22641509	-0.50684932	-1	0	0.3740458	-1	-0.83870968	0	-1	1
-0.375	1	0.33333333	-0.32075472	-0.51141553	-1	0	0.64885496	1	-0.87096774	1	-1	0.33333333
-0.45833333	1	-1	-0.13207547	-0.543379	-1	0	0.63358779	-1	-1	1	-1	0.33333333
0.33333333	1	0.33333333	0.05660377	-0.46575342	1	0	0.00763359	1	-0.67741935	0	-1	0.33333333
-0.54166667	1	1	-0.13207547	-0.66666667	-1	0	0.63358779	1	-0.5483871	1	-1	1
0.75	-1	-0.33333333	0.24528302	-0.19634703	-1	0	0.38931298	-1	-0.87096774	1	0	0.33333333

Fig. 2: Datos existentes en la base extraída de Kaggle normalizados.

Se observa la escalización realizada en la matriz de entrada, esto nos permitirá mejorar los datos y facilitará el entrenamiento de los modelos y su optimización ya que acotamos los datos entre el pico más alto y el más bajo. Posteriormente, se aplica PCA. Para evaluar la reducción dimensional primero se descompone en 13 componentes (debido a las 13 características de entrada) y se observa la varianza explicada por característica:

varianza explicada por cada característica después de PCA: [0.27038127 0.18743662 0.1252547 0.08969606 0.08284437 0.06152327 0.05556614 0.03414654 0.03067008 0.02228809 0.01599219 0.01391614 0.01028452]

Al observar la varianza explicada se decide no realizar reducción dimensional, la sumatoria de las varianzas de los 13 componentes tendrá un valor de exactamente 1. Se aplica entonces PCA sin reducción dimensional a la matriz de entrada.

age	sex	cp	trestbps	chol	fbs	resteog	thalach	exang	oldpeak	slope	ca	thal
-0.35668595	-0.91665005	2.27444362	1.08559778	0.80777925	-0.09472806	-0.7060286	0.66664075	-0.26875588	0.12951321	0.00064331	0.22499041	-0.03789612
-0.26950604	-0.77038305	-0.00959798	1.46227082	0.79144235	0.72473618	-0.56116475	0.07692548	0.70138242	0.55367871	0.18841063	0.32327531	0.11475588
-1.32010981	0.86009118	-0.33314059	-0.28894905	-0.09325815	-0.56066508	-0.39053335	-0.08124056	0.42338034	0.2323664	0.46588376	-0.18533269	-0.1831273
-0.56092124	-0.87791116	-0.59100159	-0.32388672	-0.06627907	0.33953122	-0.11064514	0.07462701	-0.24958332	0.09704664	0.12412772	0.38317681	-0.12156304
0.36873364	1.77115943	-0.60888702	-1.10897498	0.24708855	0.21275283	0.00014708	0.02887558	-0.13366515	0.03775061	0.04684379	0.49882278	0.23850635
-0.12845911	-0.53472653	-0.46307404	0.38538093	-0.4442997	0.60266553	-0.67414005	0.79335874	-0.49614246	0.3267473	-0.28226267	-0.00491647	-0.11625156
-1.02136091	1.10409398	0.01285936	0.61053929	-0.08484334	-0.47772497	-0.42771742	-0.04958442	-0.04818649	0.11759809	-0.0072973	0.04538775	0.10094932
-0.52832468	-0.94548103	-0.69382372	-0.42660118	-0.05820869	0.37569514	-0.14501912	-0.5183697	0.15096856	-0.06556224	-0.07074382	0.05015946	0.13409164
-0.59949735	-1.16877178	1.44494231	-0.80759851	0.16069553	0.55974427	-0.08946737	-0.74036151	-0.35954118	0.41525394	-0.00490388	-0.40161297	-0.08725867
-0.73200599	-0.99808328	-0.32576862	-0.11116097	0.45110769	0.24141946	0.21637625	0.00330388	-0.43166966	0.49573121	0.26095782	0.03366646	-0.3488861
-0.26604245	-0.69679453	-0.69779061	-0.29989397	-0.58154681	0.43082574	-0.31006759	0.03064668	-0.44186152	0.38086577	0.21793123	0.06847255	-0.00533958
-1.58205813	0.79212393	-0.26894656	-0.33141512	0.44633694	0.21608781	0.20082665	0.00134909	-0.07575209	-0.13023135	0.14334986	-0.29343068	0.27706338
-0.56406387	-0.88097278	-0.60418858	-0.35684253	-0.0428538	0.33965127	-0.16248464	0.04879818	-0.15299032	0.23849993	0.08540072	0.15473218	0.1369127
0.91396598	-0.28755871	0.12940691	0.2762703	1.75249541	-0.69835866	0.2286452	0.35043236	-0.16445648	-0.45166837	0.13164054	0.33410896	-0.26613152
-1.65973831	0.52799292	1.93402585	-0.68076867	0.56196272	-0.53892411	-0.03546086	-0.24755331	-0.17602313	0.06943446	0.32786322	-0.02987806	0.06227886
-1.33532335	0.91071652	-0.0469354	0.54192401	0.58582989	0.4520805	-0.07347547	0.04744985	0.23685411	-0.08640469	0.14586557	0.04085262	-0.10394054
-1.65020447	0.79127312	-0.25555463	-0.38858925	0.4140579	0.13907222	0.26565706	-0.06909659	-0.15255486	-0.12748714	-0.05150392	0.404042	0.25017567
-1.14951281	1.05674969	0.55821168	1.72678654	1.09527516	0.47832025	0.22736085	0.08469735	-0.46363151	-0.07472726	0.002453	-0.22356745	-0.02484972
-0.29514642	-0.73555433	-0.73428266	-0.34500742	-0.51550027	0.45570558	-0.42517826	-0.03802681	-0.11487872	0.71398711	0.28673981	-0.0661258	0.11782537
-1.59478252	0.71442922	0.23226061	0.11887315	0.46907406	0.00742521	1.46764906	0.15267071	-0.24234107	0.0412795	0.29624908	0.10063543	-0.17554015
0.01414542	-0.58834879	-0.50268746	0.45146864	-0.53057816	0.60513071	-0.44081147	-0.47774116	-0.29795826	0.03141921	-0.40093133	0.14803086	-0.12756776
0.67690815	-0.39386331	-0.51429576	-1.00282378	1.36262032	0.13424836	0.21850209	0.17599625	0.17719701	0.28535405	-0.03373226	0.14440778	-0.01334378
-0.38802225	-0.78654621	-0.81020202	-0.54331943	-0.50589908	0.44764588	-0.5040233	0.03775014	0.01284177	0.49326906	-0.1014154	-0.06517001	0.02863115
1.14362199	-0.26397267	1.67363432	-0.59096116	1.10054148	0.60485164	-0.14685134	0.14529211	-0.45175492	0.06675919	-0.15585389	0.01162418	0.04139296
0.57346627	-0.56231902	-0.34037132	-0.78629515	1.86151997	0.0886072	0.56825516	-0.47783299	0.36034067	0.31233968	0.19483754	-0.10991686	-0.12492347
-1.21150191	1.00029751	-0.0770741	-0.17743449	-0.56641831	0.09646316	1.0019972	0.06385098	-0.45529645	0.42312974	-0.24611701	0.18007562	0.00721402
-0.61324074	-1.10323267	1.48110699	-0.70127085	0.16178684	0.57369755	-0.12860756	-0.08507683	-0.53979497	0.25241924	0.40902187	-0.00372045	-0.10123898

Fig. 3: Matriz de entrada después de realizar PCA.

Finalmente esta será la matriz que se utiliza para entrenar los modelos, después de realizar PCA se divide entre entrenamiento y validación, a partir de estos conjuntos se empezará el entrenamiento.

2.2 Métricas utilizadas

Para evaluar todos los modelos en este proyecto se emplean las siguientes métricas:

- F1 score
- ROC curve
- Matthews correlation coefficient

Mediante estas tres métricas se determinará qué tan bueno es cada clasificador, y se realizará una comparación entre todos para hallar el mejor. Se explicará el significado de cada una en la siguiente sección, y se realizará el análisis de cada una partiendo de los resultados dados por los clasificadores, se explicará cuales son valores buenos y cuales pueden mejorarse dado el caso.

2.3 Regresión logística

La regresión logística es un método de clasificación cuyos hiperparámetros se configuran probando diferentes combinaciones y observando la documentación, se elige como solver 'liblinear' debido a que este funciona bastante bien para datasets pequeños como el nuestro. Esta configuración permite utilizar pesos de clase balanceados y penalty 'L2'. Posteriormente mediante ciclos se determina cual debe ser el factor de regularización C.

```
corcoef = []
regularization = []
iter = []
for C in np.linspace(0.1,10,100):
  clf = LogisticRegression(penalty='l2',max_iter=10000,random_state=0, C = C, class_weight = 'balanced', solver = 'liblinear').fit(X_train, np.ravel(y_train))
  y_test_predicted = clf.predict(X_test)
  y_test_scores = clf.predict_proba(X_test)
  corcoef.append(matthews_corrcoef(y_test, y_test_predicted))
  regularization.append(C)
max_score = max(corcoef)
best_C = corcoef.index(max_score)
print(max_score)
print(regularization[best_C])
```

```
0.6517059369140351
3.4000000000000004
```

Fig. 4: Configuración de los hiperparámetros - Regresión logística.

Se observa que este método retorna cual es la mejor correlación encontrada, y cual es el factor de regularización que optimiza este parámetro. Se utiliza esta configuración y se entrena el modelo de regresión logística. Posteriormente, se hallan los valores de las métricas establecidas.

```
Accuracy of LR classifier on training set: 0.85
Accuracy of LR classifier on test set: 0.83
coeficiente de matthews: 0.6517059369140351
F1 Score:  0.8289473684210527
```

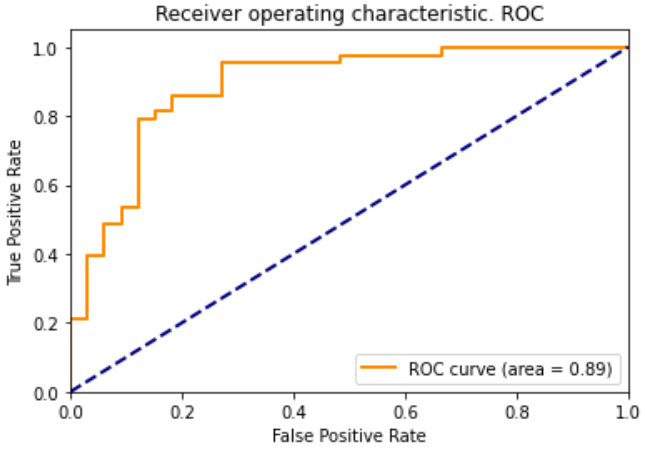

Fig. 5: Valores de las métricas establecidas - Regresión logística.

Se observan las tres métricas importantes del modelo: el coeficiente de Matthews, el valor de F1 y la curva ROC. Comenzando por el coeficiente de Matthews, este valor es

mayor a 0.6, es decir, se tiene correlación intensa entre las variables, por ende este resultado es bastante bueno, ya que esta métrica se calcula utilizando las salidas predichas por el modelo, y las salidas ya establecidas anteriormente, al tener esta correlación intensa significa que ambas salidas son muy similares entre sí, por ende, se busca tener la mayor correlación posible. Posteriormente se analiza el valor F1, siendo este 0.83, este valor agrupa la precisión y la exhaustividad, es decir, retorna un valor que indica la calidad general del modelo. Entonces utilizando el valor F1 se puede comparar más fácilmente el rendimiento entre varios modelos, buscando siempre tener el mayor puntaje, esta métrica se utilizará más adelante para hallar el mejor clasificador. Finalmente evaluamos la curva ROC, con un area de 0.89, esta curva permite evaluar el rendimiento del clasificador en todos los umbrales de clasificación, y se representa con dos parámetros: TPR (true positive rate) y FPR (false positive rate). Se busca entonces tener la mayor cantidad de TPR frente a un FPR menor, ya que el TPR son las predicciones correctas, por ende el área bajo la curva debe ser lo más cercano a 1 que se pueda, determinando que se tiene un muy buen TPR. Como el area bajo la curva ROC en este modelo es 0.89, se puede determinar que el modelo cuenta con una muy buen TPR frente al FPR, de hecho es posible identificar dos saltos significativos que suceden cerca a 0.2, cambiando el TPR de 0.55 a 0.8, y en 0.3, cambiando el TPR de 0.85 a 0.95, para posteriormente acercarse más a 1. Una vez explicados estas métricas y analizados los valores retornados por el modelo de regresión logística, se procede a evaluar y comparar los demás modelos.

2.4 K vecinos más cercanos

El método de clasificación K vecinos mas cercanos es un modelo con una cantidad de hiperparámetros considerables que requieren de un algoritmo para la obtención de la mejor combinación de estos. Para esto se realiza un bucle para evaluar el modelo con 50 diferentes k, de 1 a 50. Ademas de esto se va variando la métrica y la distancia, así obteniendo los mejores K para cada combinación.

```
k_range = range(1, 50)
scores = []
distance=['minkowski','euclidean','manhattan','chebyshev','seuclidean','mahalanobis']
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k,weights='distance',metric=distance[1], metric_params=None,algorithm='auto')
    knn.fit(X_train, np.ravel(y_train))
    scores.append(knn.score(X_test, y_test))
max_score = max(scores)
best_k = scores.index(max_score)+1
print("max_score",max_score)
print("best_k",best_k)
plt.figure()
plt.xlabel('k')
plt.scatter(k_range, scores)
plt.xticks([0,5,10,15,20])

print('minkowski-distance - K=4 SCORE=0.868421')
print('minkowski-uniform - K=4 SCORE=0.881578')

print('euclidean-distance - K=4 SCORE=0.868421')
print('euclidean-uniform - K=4 SCORE=0.881578')

print('manhattan-distance - K=6 SCORE=0.855263')
print('manhattan-uniform - K=4 SCORE=0.855263')

print('chebyshev-distance - K=3 SCORE=0.842105')
print('chebyshev-uniform - K=3 SCORE=0.842105')
```

```
max_score 0.8684210526315791
best_k 4
```

Fig. 6: Configuración de los hiperparámetros - KNN.

Los hiperparámetros que mas destacaron en el puntaje fueron un K=4, una métrica minkowski y un peso por distancia. Se utilizo esta configuración para el entrenamiento del modelo de K vecinos mas cercanos y posteriormente se encontraron los valores de las métricas de evaluación.

```
Accuracy of K-NN classifier on training set: 0.85
Accuracy of K-NN classifier on test set: 0.88
coeficiente de matthews: 0.7601021471234662
F1 Score:  0.8815789473684211
```

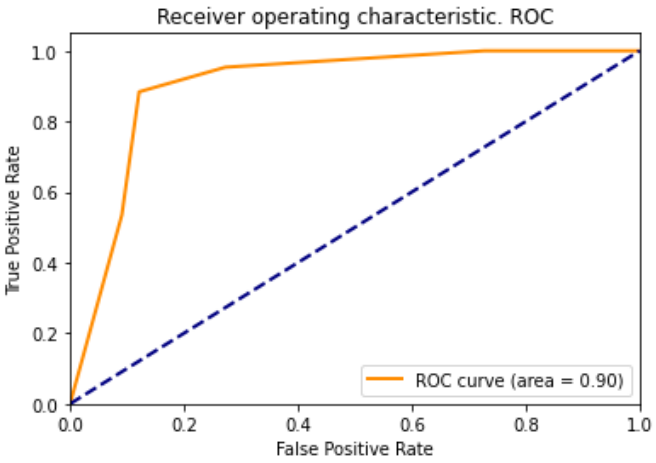

Fig. 7: Valores de las métricas establecidas - KNN.

Se puede apreciar un coeficiente de Matthews de 0.76, que como mencionado anteriormente esta por encima de 0.6 teniendo un resultado bastante bueno debido a la correlación entre las variables, y un puntaje F1 de 0.88 que califica la calidad general, siendo este tambien un buen resultado. Otro valor importante tomado en cuenta es el area bajo la curva ROC que es de 0.9, se puede apreciar una linealidad hasta mas de 0.1, donde con un valor de 0.9 tiende a convertirse en constante.

2.5 Redes neuronales

Para redes neuronales se eligió el clasificador de redes neuronales perceptrón multicapa. Este nos da una variedad de parámetros los cuales ajustar a la solución. Los primeros parámetros ha encontrar fueron la activación y el solucionador. Esto se realizo un bucle que cambia la activación y a mano se cambiaba el solucionador para tener los mejores resultados de cada solucionador.

```
solvers = ["lbfgs", "sgd","adam"]
activations =['identity', 'logistic', 'tanh', 'relu']
scores=[]


for z in activations:
  clf = MLPClassifier(random_state=1, max_iter=10000, solver=solvers[0], activation=z).fit(X_train, np.ravel(y_train))
  scores.append(clf.score(X_test, y_test))
#clf.score(X_test, y_test)
max_score = max(scores)
best_act = scores.index(max_score)
print("max_score",max_score)
print("best_Activation",best_act)
```

```
max_score 0.8552631578947368
best_Activation 1
```

Fig. 8: Configuración de los primeros hiperparámetros - ANN.

La mejor configuración obtuvo un puntaje de 0.85, esta fue la pareja de solucionador lbfgs y activación logística. Con estos parámetros en mano se procede a encontrar la mejor cantidad de capas y unidades de las capas para el sistema. Al igual que en el caso anterior, se cambiaban las unidades a mano, mientras que las capas cambiaban de 1 a 20 buscando el mejor puntaje entre las combinaciones.

```
solvers = ["lbfgs", "sgd","adam"]
activations =['identity', 'logistic', 'tanh', 'relu']
scores=[]

for z in range(1,20):
  clf = MLPClassifier(random_state=1, max_iter=10000, solver=solvers[0], activation=activations[1], hidden_layer_sizes=(5, z)).fit(X_train, np.ravel(y_train))
  scores.append(clf.score(X_test, y_test))
#clf.score(X_test, y_test)
max_score = max(scores)
best_lay = scores.index(max_score)+1
print("max_score",max_score)
print("best_Layer",best_lay)
```

```
max_score 0.868421052631579
best_Layer 6
```

Fig. 9: Configuración de los segundos hiperparámetros - ANN.

En los resultados se encontró que 6 capas (sin incluir la entrada y salida) y 5 unidades en las capas eran la pareja que daba el mejor resultado con un puntaje de .86. Como últimos parámetros, se buscó el mejor ”learning rate” con un bucle que lo cambia de 0.001 a 0.1.

```
solvers = ["lbfgs", "sgd","adam"]
activations =['identity', 'logistic', 'tanh', 'relu']
learnrn=["constant","invscaling","adaptive"]
vec=[]
scores=[]
#lay =

for z in np.linspace(0.001,0.1,100):
  clf = MLPClassifier(random_state=1, max_iter=10000, solver=solvers[0], activation=activations[1], hidden_layer_sizes=(5, 6), learning_rate_init=z,learning_rate=learnrn[2]).fit(X_train, np.ravel(y_train))
  vec.append(z)
  scores.append(clf.score(X_test, y_test))
#clf.score(X_test, y_test)
max_score = max(scores)
best_learn = scores.index(max_score)
print("max_score",max_score)
print("best_initiation",best_learn)
print(vec[best_Act])
```

```
max_score 0.8684210526315789
best_Activation 0
0.001
```

Fig. 10: Configuración de los terceros hiperparámetros - ANN.

Como resultado se obtuvo que cualquier valor por debajo de 0.01 era mas que suficiente para darnos un puntaje de 0.86. Ya con todos los hiperparámetros calculados se entrena el modelo para obtener los valores de rendimiento.

```
Accuracy of MLPC classifier on training set: 0.96
Accuracy of MLPC classifier on test set: 0.87
coeficiente de matthews: 0.7400679502291563
F1 Score:  0.868421052631579
```

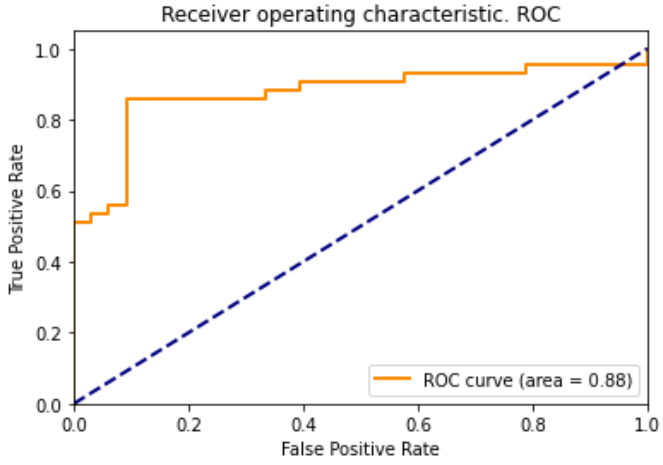

Fig. 11: Configuración de los terceros hiperparámetros - ANN.

Se puede apreciar un coeficiente de Matthews de 0.74, que como mencionado anteriormente esta por encima de 0.6 teniendo un resultado bastante bueno debido a la correlación entre las variables, y un puntaje F1 de 0.86. Para este caso el área bajo la curva ROC es de 0.88, en la cual se aprecia el pico importante en aproximadamente 0.1, donde pasa de 0.55 a 0.87 aproximadamente.

3 Conclusiones

Para concluir es importante mencionar que el modelo que sobresalio por encima de los demas fue KNN, obteniendo mejores puntajes en todo. Este fue el modelo que mostró mejores resultados teniendo en cuenta que todos fueron analizados y probados para obtener los hiperparámetros con los puntajes mas altos.

La tarea de realizar un modelo de aprendizaje de maquina va tambien desde entender como funciona cada modelo y cual es el que mejor se adapta para la solución. Esto es importante para buscar los hiperparámetros que mejor se comporten, y saber como buscarlos.

Los hiperparámetros son los mas relevantes a la hora de crear un modelo. Siendo estos capaces de cambiar los puntajes fácilmente y alterar la calidad del modelo para el dataset. Encontrar los mejores valores para los hiperparámetros es una tarea de bastante importancia y los primero que se debe pensar a la hora de implementar un modelo.

4 Bibliografía

1 ”sklearn.linear_model.LogisticRegression-scikit-learn 0.24.2 documentation”, Scikit-learn.org, 2021. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed: 02- Jun- 2021]

2 ”Clasificación: ROC y AUC — Curso intensivo de aprendizaje automático”, Google Developers, 2021. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419#: :text=Una%20curva%20ROC%20(curva%20de,Tasa%20de%20verdaderos%20positivos. [Accessed: 02- Jun- 2021]

3 ”Coeficientes de Correlación”, Es.slideshare.net, 2021. [Online]. Available: https://es.slideshare.net/KarlaGuzmn21/coeficientes-de-correlacin-201988310#: :text=COEFICIENTE%20DE%20CORRELACI%C3%93N%20PHI%20Tambi%C3%A9n,Pearson%20debido%20a%20su%20interpretaci%C3%B3n. [Accessed: 02- Jun- 2021]

4 ”K Neighbors Classifier documentation”, Scikit-learn.org, 2021. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. [Accessed: 02- Jun- 2021].

5 ”sklearn MLPClassifier documentation”, Scikit-learn.org, 2021. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier. [Accessed: 02- Jun- 2021].