PROYECTO FINAL INTELIGENCIA ARTIFICIAL

AUTORES:
Federico Sierra Gaitán

PRESENTADO A:
Francisco Carlos Calderón Bocanegra, Ing, PhD

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRÓNICA
BOGOTÁ D.C.
26/11/2021

Algoritmo de Machine Learning para la predicción de muerte por ataque al corazón

En el presente documento se mostrará el tratamiento de un dataset formado por un total de 11 características o datos tomados en 294 pacientes etiquetados con lo que supone su estado actual, vivo o muerto. Con ello se busca realizar un algoritmo de machine learning que sea capaz de clasificar y predecir la supervivencia de una persona dados sus datos tales como edad, sexo, presión en la sangre, plaquetas, etc.

Así pues, se propone una solución basada en algoritmos de regresión y clasificación supervisado con codificación en el lenguaje de programación Python, siguiente todos los lineamientos y requisitos dados por el profesor de la asignatura.

```
[227] #Visualizacion de los datos del dataset
      data = pd.read_csv(r"heart.csv")
      data.head(-5)
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	1
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	1
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	1
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	1
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
289	90.0	1	337	0	38	0	390000.00	0.9	144	0	0	256	0
290	45.0	0	615	1	55	0	222000.00	0.8	141	0	0	257	0
291	60.0	0	320	0	35	0	133000.00	1.4	139	1	0	258	0
292	52.0	0	190	1	38	0	382000.00	1.0	140	1	1	258	0
293	63.0	1	103	1	35	0	179000.00	0.9	136	1	1	270	0

294 rows × 13 columns

Si fuese necesario en este paso transformaríamos características de un valor categórico a numérico, pero en este caso el data set ya cuenta con esto, donde:

- **Anaemia:** 0 significa que la persona no tiene anemia, 1 si tiene.
- **Diabetes:** 0 significa que la persona no tiene diabetes, 1 si tiene.
- **High_blood_pressure:** 0 significa que la persona no tiene alta presión sanguínea, 1 si tiene.
- **Smoking:** 0 significa que la persona no fuma, 1 si fuma.
- **Sex:** 0 mujer, 1 hombre.
- **DEATH_EVENT :** 0 vivo, 1 muerto.

Primero realizamos la visualización y limpieza de nuestro data set, donde eliminamos valores nulos o vacíos de las características, o se rellenan con el promedio de los valores existentes para esa característica.

```
#Revisar si existen datos nulos o espacios de columnas sin rellenar
print("Valores nulos en train: \n", data.isna().sum(),"\n")
```

```
Valores nulos en train:
 age                         0
anaemia                      0
creatinine_phosphokinase     0
diabetes                     0
ejection_fraction            0
high_blood_pressure          0
platelets                    0
serum_creatinine             0
serum_sodium                 0
sex                          0
smoking                      0
time                         0
DEATH_EVENT                  0
dtype: int64
```

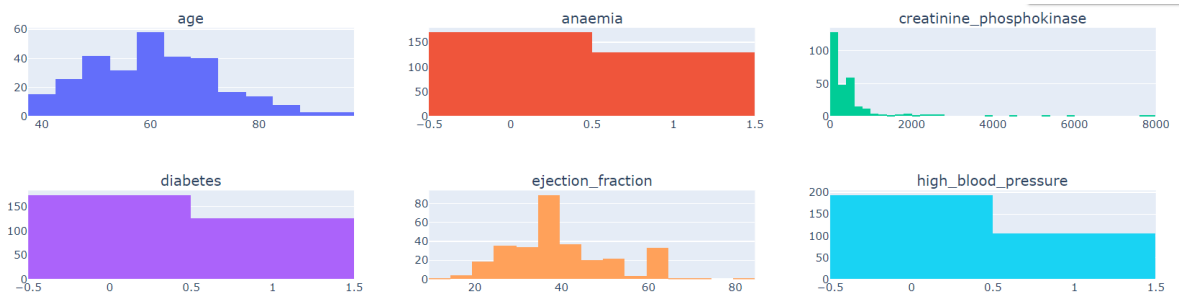
Como se puede ver no encontramos ningún valor nulo necesario de eliminar o vacío para rellenar, sin embargo, si es necesario eliminar una columna, la de time, ya que esta no representa ninguna información relevante y además sería un dato imposible de conseguir si se quisiese ampliar el data set.

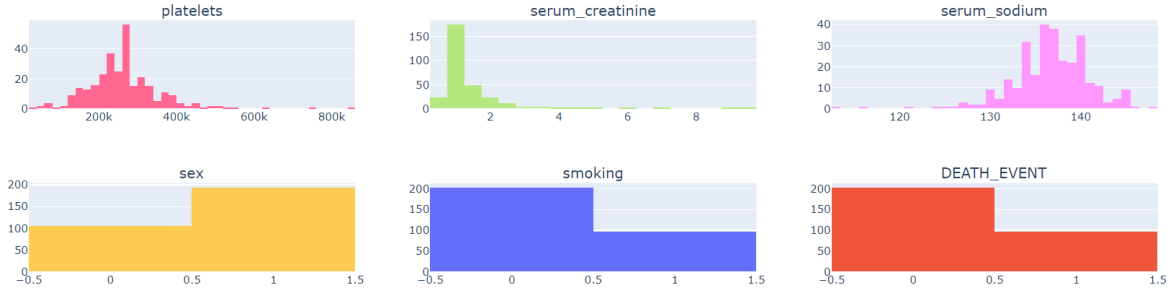
```
[229] #Se eliminar de las columnas de caracteristicas aquellas que no son útiles para nuestro proyecto
      data.drop(['time'], axis = 1, inplace = True)
      data.head(-5)
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	DEATH_EVENT
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	1
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	1
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	1
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	1
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
289	90.0	1	337	0	38	0	390000.00	0.9	144	0	0	0
290	45.0	0	615	1	55	0	222000.00	0.8	141	0	0	0
291	60.0	0	320	0	35	0	133000.00	1.4	139	1	0	0
292	52.0	0	190	1	38	0	382000.00	1.0	140	1	1	0
293	63.0	1	103	1	35	0	179000.00	0.9	136	1	1	0

294 rows × 12 columns

Luego realizamos un poco de análisis del dataset graficando un histograma por cada característica en el





Como tenemos definido nuestro método como uno supervisado, vamos a usar un 75% para entrenar y un 25% para validar, seleccionamos estos a partir de la cantidad de datos, de tal manera que el conjunto de validación que suele ser el más pequeño de los dos tenga por lo menos una cantidad significativa de representantes de todas las clases.

```
X = data[['age','anaemia','creatinine_phosphokinase','diabetes','ejection_fraction',
          'high_blood_pressure','platelets','serum_creatinine','serum_sodium','sex','smoking']].values
y = data['DEATH_EVENT'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0,test_size=0.25)
```

Inmediatamente después pasamos a normalizar nuestros datos debido a que en si es una buena práctica y además observamos que valores como plaquetas se encuentran muy desproporcionados en relación con los valores de características dados por 0 o 1.

```
[290] scaler = StandardScaler() #Se escalizan los datos
      scaler.fit(X_train) #El fit de los datos se hace con el conjunto de entrenamiento!
      X_train_scaled = scaler.transform(X_train)
      X_test_scaled = scaler.transform(X_test)
```

Con los datos ya normalizamos se procede a realizar un análisis por componentes principales (PCA), para evaluar si este se viera beneficiado por reducir su dimensionalidad a partir de la varianza explicada:

```
#PCA solamente al conjunto de entrenamiento
n = 11
pca = decomposition.PCA(n_components=n,whiten=True,svd_solver='auto')
pca.fit(X_train_scaled)
X_train_PCA = pca.transform(X_train_scaled)
X_test_PCA = pca.transform(X_test_scaled)
print("Pesos de PCA:",pca.explained_variance_ratio_)
print("Componentes:",pca.components_)
```

```
Pesos de PCA: [0.15224169 0.13211175 0.12555635 0.09786508 0.0881707  0.08532696
 0.07642814 0.07058003 0.06662873 0.05694353 0.04814706]
```

```
Componentes: [[ 0.24590246 -0.04349115 -0.06868628 -0.30679713 -0.20115128 -0.14236438
  -0.25484092  0.33391293 -0.18222272  0.57048892  0.49291409]
 [ 0.20790155  0.32094545 -0.24189507  0.28306447 -0.26436156  0.03645558
  -0.11197987  0.47499899 -0.49530005 -0.22769328 -0.32952435]
 [-0.43498787 -0.38032561  0.3975485   0.35414347 -0.44176688 -0.28701955
   0.02948507 -0.03997317 -0.3081458   0.08318589  0.02822104]
 [ 0.14378335 -0.39083431  0.11956605 -0.09088571  0.03594753  0.53594295
   0.60918469  0.28992677 -0.20332471 -0.05244523  0.12742294]
 [ 0.51117627 -0.18080103  0.66311491 -0.08120642  0.11749034 -0.05643785
  -0.27750804  0.10006075  0.07429883  0.00537263 -0.3874696 ]
 [ 0.11709208  0.19944777  0.09012105  0.12393439  0.32433884 -0.6656435
   0.54979694  0.25451525  0.0395652   0.04348366  0.06512002]
 [-0.18458907  0.38147901  0.27432245 -0.53681764 -0.53360563 -0.01697714
   0.20648132  0.15219404  0.2469546  -0.20299967 -0.06291627]
 [ 0.07716224  0.56606618  0.37396039  0.45210493 -0.03589405  0.36361799
   0.09337146 -0.15793406  0.0465638   0.33082956  0.22491247]
 [-0.44355031  0.22994427  0.2594467  -0.34130113  0.5050935   0.0765874
  -0.14355746  0.05143978 -0.5309168   0.00751726 -0.02051055]
 [-0.40773866 -0.08001479  0.01622601  0.21171977  0.16880104  0.15046455
  -0.19304729  0.66034069  0.48150505  0.1369425  -0.08489676]
 [ 0.09162022 -0.00109079  0.18414636  0.12741362  0.06756237 -0.03886296
  -0.2490484   0.12502196  0.03489604 -0.66456095  0.64410468]]
```

Como vemos en los pesos de PCA no es posible eliminar ninguna componente ya que no estaríamos perdiendo de alrededor de un 4%-5% de la varianza explicada, dejándonos así solo un 95% para analizar que no es lo deseado.

Finalmente aplicamos nuestros algoritmos escogidos para esta aplicación, Regresión Logística y SVM, iterando sus hiperparámetros para conseguir la mejor calificación en los métodos de evaluación de f1 score y MCC:

```
#Empieza el entrenanmiento

#Regresión logística
log = LogisticRegression(C=0.2, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=20, solver='liblinear', tol=0.1, verbose=0,
                   warm_start=False)

log.fit(X_train_PCA, y_train)
pred = log.predict(X_test_PCA)
```

```
LogisticRegression(C=0.2, max_iter=200, random_state=20, solver='liblinear',
                   tol=0.1)
```

Para la regresión lineal los hiperparámetros a iterar fueron la regularización y el solver, los mejores son ellos que se muestran en la imagen y con los que se obtuvieron los siguientes resultados:

```
print("F1 Score:", metrics.f1_score(y_test, pred,average='macro'))
```

```
F1 Score: 0.8783876500857634
```

```
[343] print("MCC:", metrics.matthews_corrcoef(y_test, pred))

      MCC: 0.790119475120320 25
```

Ahora con las máquinas de soporte vectorial:

```
#SVM
kernels=['linear', 'poly', 'rbf', 'sigmoid']
#lineal
#Kernel=0
#msv = svm.SVC(kernel=kernels[Kernel])

#polinomial cuadrático
#Kernel=1
#msv = svm.SVC(kernel=kernels[Kernel],degree=2)

#polinomial cúbico
#Kernel=1
#msv = svm.SVC(kernel=kernels[Kernel],degree=3)
#rbf
Kernel=2
msv = svm.SVC(kernel=kernels[Kernel],C=1)
#https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

msv.fit(X_train_PCA, y_train)

SVC(C=1)

[333] y_test_predicted = msv.predict(X_test_PCA)
```

El kernel elegido por brindar los mejores resultados fue el rbf, que combinado con la regularización obtuvimos los siguientes resultados:

```
[345] print("F1 Score:", metrics.f1_score(y_test, y_test_predicted,average='macro'))

      F1 Score: 0.8423529411764706
```

```
print("MCC:", metrics.matthews_corrcoef(y_test, y_test_predicted))

MCC: 0.77468516529888065
```

Conclusiones

- A pesar de la existencia de grandes librerías como scikit learn que, que ofrecen una gran facilidad a la hora de aplicar diferentes algoritmos de machine learning, el

conocimiento profundo de cada método, para que sirve, para que no, en que momentos es recomendable usar uno u otro

- Se demuestra que la iteración de parámetros es una parte fundamental a la hora de conseguir buenos resultados en la consecución de buenos resultados

- Para problemas de clasificación binaria tal como es el caso de este proyecto, y con un numero no tan bajo o alto de características SVM y regresión lineal son muy buenos métodos de clasificación