Basile Álvarez Andrés José

No. Cuenta: 316617187

Email: andresbasile123@gmail.com

Fecha: 10/12/21

Inteligencia Artificial

Grupo III

Semestre 2022-1

Reporte Práctica 16: Modelos de Clasificación

Objetivo: Clasificar si una persona tiene diabetes o no, en función de otros parámetros disponibles, como número de embarazos, glucosa, presión arterial, índice de masa corporal, los niveles de insulina, entre otros. Este es un problema de clasificación y se requiere obtener el mejor modelo de aprendizaje automático para predecir la diabetes.

Fuente de datos:

https://www.kaggle.com/saurabh00007/diabetescsv

Variables:

- Número de embarazos (Pregnancies): número de veces que ha estado embarazada la persona.
- Concentración de glucosa en plasma (Glucose): cantidad de glucosa en la sangre. Cuando una persona ha ingerido alimento los valores normales son menores a 140 mg/DL y cuando los resultados se dan entre 140 a 190 son indicativos de diabetes.
- Presión arterial diastólica (BloodPressure): es la cantidad de presión que hay en las arterias ente un latido y otro.
- Espesor del pliegue cutáneo (SkinThickness): es un procedimiento frecuentemente utilizado, en combinación con el índice de masa corporal (IMC), para estimar la grasa corporal. Medir los pliegues cutáneos permite valorar los depósitos de grasa del cuerpo humano. A modo de referencia, según la medicina el espesor normal: ♂ 12 mm; ♀ 23 mm.
- Insulina (Insulin): es una prueba de insulina que consiste analizar antes de administrar la glucosa y 2 horas después. La razón por la que se realizan estas pruebas es para ver la curva de respuesta a la glucosa.
- Índice de masa corporal (BMI): es utilizado para estimar la cantidad de grasa corporal, y determinar si el peso está dentro del rango normal, o por el contrario, se tiene sobrepeso o delgadez.
- Función pedigrí de la diabetes (DiabetesPedigreeFunction): es una función que califica la probabilidad de diabetes según los antecedentes familiares.
- Edad en años (Age).
- Resultado (Outcome): si es positivo o negativo al diagnóstico de diabetes.

Características Generales:

En esta práctica, utilizaremos distintos algoritmos de clasificación con inteligencia artificial dividir una población de personas entre los que tienen diabetes y los que no. La regresión logística, a través de una regresión lineal, permite transformar los datos para clasificarlos. Los árboles de decisión utilizan un modelo jerárquico donde los datos se van clasificando según decisiones realizadas en cada nodo y las máquinas de soporte vectorial permiten hacer una transformación de los datos para su clasificación.

Desarrollo

Primeramente, tenemos que definir aquellas bibliotecas de Python que nos serán útiles para importar, limpiar y analizar los datos contenidos en el archivo separado por comas *Diabetes.csv*. Estas serán: *pandas* (manipulación y análisis de datos), *matplotlib* (para la creación de gráficas y visualización de los datos), *numpy* para utilizar vectores y matrices de *n* dimensiones.

Más tarde, importamos el archivo *Diabetes.csv* y lo primero que hacemos es guardarlo en un DataFrame. La importación del archivo se realizó a partir del explorador de archivos que abrimos utilizando el comando *files.upload()*. Una vez importados los datos, mostramos el DataFrame que los contiene:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Figura 1: Data Frame con algunos de los datos de diabetes.

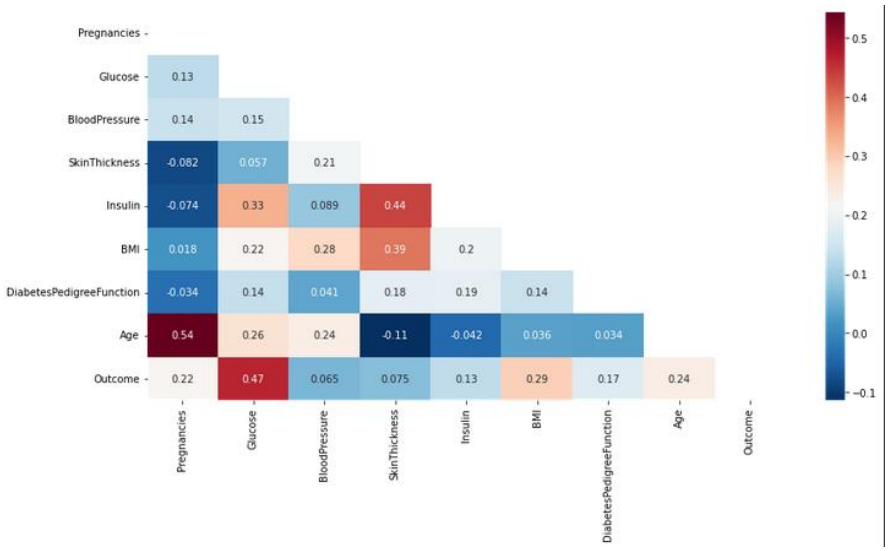
Una vez hecho lo anterior, se hace una agrupación de los datos:

```
Outcome
0    500
1    268
dtype: int64
```

Figura 2: Agrupación de los datos según el diagnóstico.

Una vez hecho lo anterior, utilizamos una matriz de correlaciones con el propósito de seleccionar variables significativas en el conjunto de datos y reducir la dimensionalidad de este. En la figura 3 se muestra la matriz de correlación de Pearson, utilizando colores para distinguir las variables que tienen una mayor dependencia y así poder realizar la selección de variables.

Figura 3: Matriz con las correlaciones entre variables.



Ante la no presencia de correlaciones altas, se consideran todas las variables para la construcción de los modelos. Definimos entonces un *array* con las variables predictoras que nos permitirán pronosticar el valor del diagnóstico en la clasificación, como se muestra en la figura 4.

Figura 4: Parte de la matriz con las variables predictoras.

```
#Variables predictoras
X = np.array(Diabetes[['Pregnancies',
                       'Glucose',
                       'BloodPressure',
                       'SkinThickness',
                       'Insulin',
                       'BMI',
                       'DiabetesPedigreeFunction',
                       'Age']])
pd.DataFrame(X)
```

	0	1	2	3	4	5	6	7
0	6.0	148.0	72.0	35.0	0.0	33.6	0.627	50.0
1	1.0	85.0	66.0	29.0	0.0	26.6	0.351	31.0
2	8.0	183.0	64.0	0.0	0.0	23.3	0.672	32.0
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0
4	0.0	137.0	40.0	35.0	168.0	43.1	2.288	33.0
...	...	...	...	...	...	...	...	...
763	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63.0
764	2.0	122.0	70.0	27.0	0.0	36.8	0.340	27.0
765	5.0	121.0	72.0	23.0	112.0	26.2	0.245	30.0
766	1.0	126.0	60.0	0.0	0.0	30.1	0.349	47.0
767	1.0	93.0	70.0	31.0	0.0	30.4	0.315	23.0

768 rows × 8 columns

Para poder aplicar los algoritmos, importamos lo siguiente:

```
from sklearn.tree import DecisionTreeClassifier

#from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC                              #Support vector classifier

from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Figura 5: Importaciones para los algoritmos.

```
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y,
                                                                                test_size = 0.2,
                                                                                random_state = 0,
                                                                                shuffle = True)
```

```
pd.DataFrame(X_train)
```

	0	1	2	3	4	5	6	7
0	7.0	150.0	78.0	29.0	126.0	35.2	0.692	54.0
1	4.0	97.0	60.0	23.0	0.0	28.2	0.443	22.0
2	0.0	165.0	90.0	33.0	680.0	52.3	0.427	23.0
3	1.0	109.0	56.0	21.0	135.0	25.2	0.833	23.0
4	8.0	120.0	0.0	0.0	0.0	30.0	0.183	38.0
...	...	...	...	...	...	...	...	...
609	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63.0
610	7.0	159.0	66.0	0.0	0.0	30.4	0.383	36.0
611	4.0	94.0	65.0	22.0	0.0	24.7	0.148	21.0
612	11.0	85.0	74.0	0.0	0.0	30.1	0.300	35.0
613	5.0	136.0	82.0	0.0	0.0	0.0	0.640	69.0

614 rows × 8 columns

Figura 6: División de los datos para entrenamiento y prueba.

Para el primero de los modelos, utilizamos la regresión logística, en donde, a partir de la función *logisticRegression()* se entrena al modelo con los datos de entrenamiento previamente definidos. Hecho lo anterior, obtuvimos una exactitud promedio del 82.46%

```
#A manera de referencia se calcula la exactitud promedio
Clasificacion.score(X_validation, Y_validation)
```

```
0.8246753246753247
```

Figura 6: Exactitud promedio regresión logística.

Para los árboles de decisión, tomando los mismos datos de entrenamiento, obtuvimos lo siguiente como exactitud promedio:

```
#Se calcula la exactitud promedio de la validación
ClasificacionAD.score(X_validation, Y_validation)
```

```
0.8051948051948052
```

Figura 7: Exactitud promedio árboles de decisión.

Para los bosques aleatorios, en donde esperábamos un resultado mejor que el árbol de decisión, obtuvimos la siguiente exactitud promedio:

```
ClasificacionBA.score(X_validation, Y_validation)
```

```
0.8311688311688312
```

Figura 8: Exactitud promedio bosque aleatorio.

Para las máquinas de soporte vectorial, obtuvimos, en el modelo lineal:

```
#Se calcula la exactitud promedio de la validación
ModeloSVM_1.score(X_validation, Y_validation)
```

```
0.8181818181818182
```

Figura 9: Exactitud promedio SVM lineal.

```
#Se calcula la exactitud promedio de la validación
ModeloSVM_2.score(X_validation, Y_validation)
```

```
0.7922077922077922
```

Figura 10 : Exactitud promedio SVM polinomial.

```
#Se calcula la exactitud promedio de la validación
ModeloSVM_3.score(X_validation, Y_validation)
```

```
0.7922077922077922
```

Figura 11 : Exactitud promedio SVM RBF.

```
#Se calcula la exactitud promedio de la validación
ModeloSVM_4.score(X_validation, Y_validation)
```

```
0.512987012987013
```

Figura 12 : Exactitud promedio SVM sigmoide.

No describiremos mucho más a detalle los procedimientos para obtener estos modelos y su exactitud promedio debido a que hemos hablado de todos ellos en prácticas anteriores.

Ahora, realizaremos la validación para el modelo de mayor exactitud que obtuvimos, en este caso, el bosque aleatorio de clasificación (83.11% exactitud promedio). Para la validación de este modelo, ocupamos la matriz de confusión siguiente:

Clasificación	0	1
Real		
0	96	11
1	15	32

Figura 13 : Matriz de confusión validación bosques aleatorios.

Para esta validación, vemos que obtuvimos una precisión del 86% para los valores de “0” (no diabetes) y del 74% para los valores de “1” (diabetes).

Además, podemos utilizar el modelo para generar nuevas clasificaciones:

```
7) Nuevas clasificaciones

#Paciente P-842302 (1) -Tumor Maligno-
PacienteID1 = pd.DataFrame({'Pregnancies':[6],
                            'Glucose': [147],
                            'BloodPressure': [72],
                            'SkinThickness':[35],
                            'Insulin': [0],
                            'BMI':[33.6],
                            'DiabetesPedigreeFunction':[0.627],
                            'Age':[50],
                             })
ClasificacionBA.predict(PacienteID1)

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:439: UserWarning: X ha
  f"X has feature names, but {self.__class__.__name__} was fitted without"
array([1])
```

Figura 14 : Nuevas clasificaciones.

Conclusiones

A lo largo de esta práctica, tuvimos como principal objetivo el obtener clasificaciones del diagnóstico de tumores utilizando bosques aleatorios. Al obtener los parámetros del modelo creado con bosques aleatorios, notamos que hay un mejor desempeño en la clasificación en comparación con la clasificación que puede realizar otro tipo de algoritmos analizados.

Vimos entonces que, de todos los algoritmos de clasificación estudiados a lo largo del curso, el bosque aleatorio fue el más exacto para este conjunto de datos en particular. Además, notamos que casi todos los conjuntos mostraron valores de exactitud relativamente bajos, lo cual creemos tiene que ver con que en el conjunto de datos encontramos una cantidad mucho mayor de casos negativos registrados con respecto de los positivos. Quizás podríamos trabajar con una muestra diferente para buscar hacer un balance entre los diagnósticos negativos y los diagnósticos positivos. La varianza de los datos también puede ser una de las causas por la que los resultados no fueron lo suficientemente alto. También podríamos hacer un filtrado de los datos, dividiendo por edades, por ejemplo.

En general, creo que se cumplieron los objetivos de la práctica y que ésta fue útil para comprender de mejor manera un ejemplo de aprendizaje supervisado sencillo y las posibles aplicaciones de este tipo de algoritmos.