Universidad de Granada

E.T.S Ingeniería Informática y de Telecomunicación
Grado en Ingeniería Informática

Inteligencia de Negocio - IN
Grupo A1 (Miércoles 09:30 - 11:30)
Curso 2020/2021

Práctica 1:
Resolución de problemas de clasificación y análisis experimental

Javier Rodríguez Rodríguez
78306251Z
e.doblerodriguez@go.ugr.es

1. Introducción:

Dentro del campo de la salud, la precisión en los diagnósticos es fundamental para el correcto tratamiento de las personas. Esto es particularmente importante para enfermedades con altos índices de mortalidad, como lo es el cáncer. En particular, el cáncer de mama depende en su diagnóstico de la detección de tumores. Sin embargo, es común que la persona presente otro tipo de tumor mucho menos peligroso, denominados benignos, en contraposición de los cancerígenos tumores malignos.

Determinar de qué tipo es un tumor dado es sumamente importante para establecer el curso de acción adecuado en el tratamiento de la persona. Si bien es posible determinar esto mediante exámenes posteriores o evaluando cómo se desarrolla a lo largo del tiempo, lo ideal es poder determinarlo apenas se tiene conocimiento de la existencia del tumor. Por esta razón se persigue el desarrollo de mecanismos que determinen de qué tipo de tumor se trata a partir de ciertos patrones. Es aquí cuando el campo de la computación, y en particular el de la inteligencia artificial, aporta mecanismos de aprendizaje con los cuales desarrollar algoritmos que encuentren dichos patrones y realicen el diagnóstico de forma automatizada y precisa.

La presente práctica consiste en el desarrollo de tal algoritmo, utilizando en particular estrategias de aprendizaje supervisado sobre un conjunto de datos de mamografías proporcionado. Se contempla el uso de diversos algoritmos y técnicas de preprocesado para interpretar adecuadamente la información proporcionada y comparar con cuál se obtienen los mejores resultados, si estos son aceptables, y por qué.

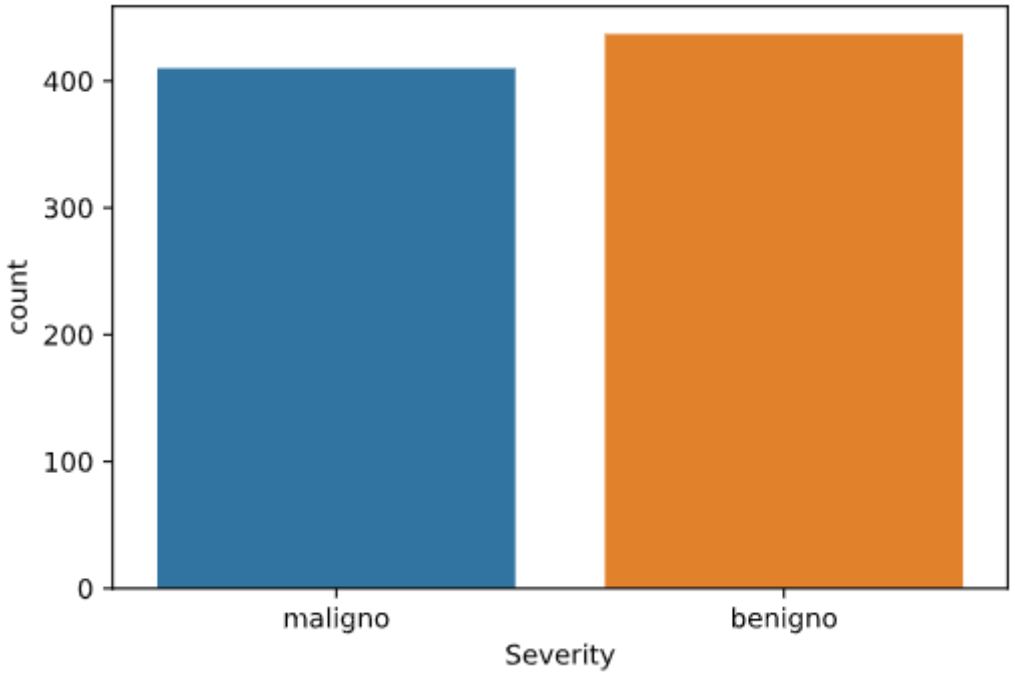
Para el desarrollo utilizamos el lenguaje Python, principalmente por su sencillez de uso y el acceso a la biblioteca Scikit-Learn, que implementa los algoritmos a utilizar.

2. Procesado de datos:

El procesado del fichero .csv es bastante sencillo al utilizar la biblioteca pandas, que implementa una función de lectura y almacena los datos en una estructura de datos propia conocida como DataFrame.

Primero que nada, se contempla qué hacer con los datos perdidos. Dado que tan solo 114 instancias tienen datos perdidos, se opta por eliminarlas. Luego, hay que transformar los parámetros representados por cadenas de caracteres, pues los algoritmos de aprendizaje utilizan exclusivamente valores numéricos. En particular, se transforman las variables 'Shape' y 'Severity' a través del objeto LabelEncoder().

Una vez leídos los datos lo primero es comprobar qué tan balanceadas están las clases a predecir, pues de ello dependerá la mejor métrica de evaluación.



Como vemos, la clase objetivo si bien no está perfectamente balanceada, sí lo está lo suficiente como para que no sea mayor problema a la hora de interpretar los datos.

Inicialmente, estas son todas las transformaciones realizadas, pues son las mínimas necesarias para realizar pruebas. Sin embargo, también se considera la estandarización, normalización y la reducción de dimensionalidad, en búsqueda de eliminar información innecesaria y mejorar el proceso de aprendizaje.

En cuanto a los resultados, estos son presentados en una tabla donde se comparan 5 métricas para cada algoritmo: accuracy (tasa de aciertos), precision (tasa de veracidad de los positivos), recall (tasa de veracidad de los negativos) , f1 score (media armónica de precision y recall) y roc auc (área

bajo la curva roc). A su vez, para cada algoritmo este valor es calculado como el promedio del obtenido para cada una de las 5 iteraciones realizadas, dado que el proceso de aprendizaje es llevado a cabo utilizando validación cruzada de 5 particiones (5-fold cross validation). Además, cada tabla identificará al final qué algoritmo funciona mejor en cada experimento para cada métrica.

3. Configuración de algoritmos:

Los algoritmos seleccionados a evaluar fueron:

3.1. Clasificador aleatorio o "Dummy": Este clasificador simplemente asigna aleatoriamente variables en función de la proporción de clases presente en los datos de entrenamiento, de forma que si en los datos de entrenamiento la proporción de etiquetas es 60/40, asignará aleatoriamente con 60% de probabilidad de escoger la primera etiqueta y 40% de escoger la segunda. Se prefiere al ser el más "inteligente" de los clasificadores aleatorios, en especial considerando que ya se concluyó que las clases están suficientemente balanceadas. En cualquier caso, su uso es simplemente como medida referencial de que el resto de algoritmos han de proporcionar mejores resultados. No tiene ningún parámetro de interés para variar.

3.2. K-Nearest Neighbors: El primero de los clasificadores de interés real, comúnmente abreviado kNN, tiene dos parámetros de particular interés. El primero es simplemente el mejor k, la cantidad de vecinos contemplados para la asignación de un valor. Por defecto se contemplan 5, pero se considerará 1, 2, 3, 5 y 10. El segundo parámetro es la consideración de pesos para darle mayor o menor valoración a ciertos datos respecto a otros. Para ello se modifica weights, que por defecto viene dado por 'uniform', o peso uniforme, a 'distance', o pesos según la distancia.

Los mejores parámetros son k=10 y peso uniforme para datos no preprocesados. Una vez estandarizados, es mejor k=5

3.3. Regresión Logística: Si bien el nombre regresión genera confusión inicialmente, este modelo de clasificación es uno de los más sencillos dentro del campo de aprendizaje supervisado, pero en caso de que los datos fuesen linealmente separables nos daría excelentes resultados. El parámetro a considerar aquí es el valor de C, la intensidad de la regularización que hace el algoritmo sobre los datos. Se considerarán 0.01, 0.1, 1, 10, 100.

El mejor parámetro es C = 0.1 para datos no preprocesados, mientras que tras la estandarización es mejor C=1

3.4. Red Neuronal: También conocido como perceptrón multicapa, la red neuronal utilizada es la dada por defecto por Scikit-Learn, pero se utilizará como función de activación la función logística, además de la función unitaria lineal rectificada, que es la dada por defecto.

El mejor parámetro es activación con función logística para datos no

preprocesados, mientras que tras la estandarización es mejor la función unitaria lineal rectificada.

3.5. Árbol de decisión: Consideraremos como parámetros a variar el criterio de selección de parámetro, en el que usaremos gini y entropía, y la máxima cantidad de características a considerar, que serán todas o la raíz cuadrada de la cantidad. El resto de parámetros serán los dados por la implementación por defecto.

Los mejores parámetros son selección usando entropía como criterio y máxima cantidad de características todas para ambos grupos de datos.

3.6. Support Vector Machine: Muy similar a Regresión Logística en cuanto a naturaleza, y por ende, en cuanto a parámetros considerados. De nuevo, se variará la intensidad de la regularización, C, con valores 0.01, 0.1, 1, 10 y 100. Además, consideraremos dos funciones de pérdida distintas, la llamada hinge loss simple y hinge loss cuadrática.

Los mejores parámetros son C = 1 y pérdida hinge loss cuadrática para datos no preprocesados, mientras que una vez estandarizados es mejor C=0.1 y pérdida hinge loss.

La determinación de los mejores atributos para cada algoritmo se hace automáticamente a partir del método GridSearchCV() de Scikit-Learn, que ejecuta un proceso de 5-fold Cross Validation previo probando todas las posibles combinaciones de atributos y seleccionando aquella que proporcione mayor accuracy. La razón por la que accuracy fue la métrica deseada fue porque la clase está balanceada, como se notó previamente.

4. Resultados obtenidos:

4.1: Clasificador aleatorio:

Se crea a través de `DummyClassifier()` y dada su sencillez y uso estrictamente referencial no se toca ninguno de sus parámetros.

Los resultados son:

	Accuracy	Precision	Recall	F1-Score	ROC AUC
Datos sin preprocesar	0.515907	0.500000	0.560976	0.528736	0.517296
Datos normalizados y estandarizados	0.515907	0.500000	0.560976	0.528736	0.517296

4.2. K-Nearest Neighbors:

Se crea a través de

```
GridSearchCV(KNeighborsClassifier(),
```

```
param_grid={'n_neighbors': [1, 2, 3, 5, 10],
'weights':['uniform', 'distance']}, n_jobs=-1)
```

de forma tal que se obtengan los parámetros que maximicen el accuracy del modelo.

Los resultados son:

	Accuracy	Precision	Recall	F1-Score	ROC AUC
Datos sin preprocesar	0.800508	0.791995	0.800000	0.795283	0.800562
Datos normalizados y estandarizados	0.814626	0.816633	0.804878	0.806177	0.814443

4.3. Regresión Logística:

Se crea a través de

```
GridSearchCV(LogisticRegression(random_state=rng,
max_iter=10000), param_grid={'C': [0.01, 0.1, 1, 10, 100]},
n_jobs=-1)
```

de forma tal que se obtengan los parámetros que maximicen el accuracy del modelo.

Los resultados son:

	Accuracy	Precision	Recall	F1-Score	ROC AUC
Datos sin preprocesar	0.824072	0.811700	0.834146	0.820472	0.824505
Datos normalizados y estandarizados	0.828771	0.834739	0.814634	0.818867	0.828477

4.4. Red Neuronal:

Se crea a través de

```
GridSearchCV(MLPClassifier(random_state=rng, max_iter=10000),
param_grid={'activation': ['logistic', 'relu']}, n_jobs=-1)
```

de forma tal que se obtengan los parámetros que maximicen el accuracy del modelo.

Los resultados son:

	Accuracy	Precision	Recall	F1-Score	ROC AUC
Datos sin preprocesar	0.818176	0.786710	0.858537	0.820219	0.819511
Datos	0.829969	0.841364	0.809756	0.818646	0.829473

normalizados y estandarizados					

4.5. Árbol de decisión:

Se crea a través de

```
GridSearchCV(DecisionTreeClassifier(random_state=rng),
param_grid={'criterion': ['gini', 'entropy'], 'max_features':
['auto', None]}, n_jobs=-1)
```

de forma tal que se obtengan los parámetros que maximicen el accuracy del modelo.

Los resultados son:

	Accuracy	Precision	Recall	F1-Score	ROC AUC
Datos sin preprocesar	0.760278	0.771310	0.721951	0.741888	0.759265
Datos normalizados y estandarizados	0.766196	0.779665	0.726829	0.749542	0.765100

4.6. Support Vector Machine:

Se crea a través de

```
GridSearchCV(LinearSVC(random_state=rng, max_iter=10000),
param_grid={'C': [0.01, 0.1, 1, 10, 100], 'loss': ['hinge',
'squared_hinge']}, n_jobs=-1)
```

de forma tal que se obtengan los parámetros que maximicen el accuracy del modelo.

Los resultados son:

	Accuracy	Precision	Recall	F1-Score	ROC AUC
Datos sin preprocesar	0.821733	0.821903	0.824390	0.815797	0.821991
Datos normalizados y estandarizados	0.832315	0.866960	0.785366	0.815737	0.830980

4.7. Naïve Bayes:

Se crea a través de `GaussianNB()` y dada su sencillez no se toca ninguno de sus parámetros.

Los resultados son:

	Accuracy	Precision	Recall	F1-Score	ROC AUC

Datos sin preprocesar	0.809906	0.779954	0.848780	0.811959	0.811185
Datos normalizados y estandarizados	0.815774	0.799001	0.826829	0.811348	0.816184

5. Análisis de resultados:

Para los datos sin preprocesar:

	Accuracy	Precision	Recall	F1-Score	ROC AUC
Clasificador aleatorio	0.515907	0.500000	0.560976	0.528736	0.517296
K-Nearest Neighbors	0.800508	0.791995	0.800000	0.795283	0.800562
Regresión Logística	0.824072	0.811700	0.834146	0.820472	0.824505
Red Neuronal	0.818176	0.786710	0.858537	0.820219	0.819511
Árbol de Decisión	0.760278	0.771310	0.721951	0.741888	0.759265
Support Vector Machine	0.821733	0.821903	0.824390	0.815797	0.821991
Naïve Bayes	0.809906	0.779954	0.848780	0.811959	0.811185
Mejor clasificador	Regresión Logística	Support Vector Machine	Red Neuronal	Regresión Logística	Regresión Logística

Mientras que para los datos preprocesados

	Accuracy	Precision	Recall	F1-Score	ROC AUC
Clasificador aleatorio	0.515907	0.500000	0.560976	0.528736	0.517296
K-Nearest Neighbors	0.814626	0.816633	0.804878	0.806177	0.814443
Regresión Logística	0.828771	0.834739	0.814634	0.818867	0.828477
Red Neuronal	0.829969	0.841364	0.809756	0.818646	0.829473
Árbol de Decisión	0.766196	0.779665	0.726829	0.749542	0.765100
Support Vector	0.832315	0.866960	0.785366	0.815737	0.830980

Machine					
Naïve Bayes	0.815774	0.799001	0.826829	0.811348	0.816184
Mejor clasificador	Support Vector Machine	Support Vector Machine	Naïve Bayes	Regresión Logística	Support Vector Machine

Rápidamente los datos nos indican que los algoritmos que buscan separar linealmente el espacio, como lo son Regresión Logística y Support Vector Machine, son los que funcionan mejor. La normalización y estandarización de los datos mejora los resultados en todos los algoritmos excepto en el Dummy, que hace que permanezcan igual, y en la Regresión Logística. Posiblemente esto se deba a que los parámetros con mayor magnitud le permitan aprender más.

6. Interpretación de resultados:

La conclusión inmediata es suponer que preprocesar los datos y utilizar SVM es la mejor herramienta, ya que en 3 de 5 métricas da los mejores resultados de todo el estudio. Como las clases están bien balanceadas, accuracy es una métrica fiable, y por tanto es la conclusión inmediata. Sin embargo, dada la naturaleza médica del estudio es una línea de razonamiento adecuada priorizar el valor de recall, ya que los falsos negativos son lo más peligroso de cara a la salud de las personas. En ese aspecto, es interesante considerar refinar más el Naïve Bayes.

De hecho, si al usar GridSearchCV se adjunta scoring=make_scorer(recall_score), para que los hiperparámetros maximicen esta métrica, es nuevamente SVM el que triunfa en ella, pero en el resto de las métricas es la Red Neural la que da los mejores resultados. Ante esto, decimos que existen 3 modelos de interés, siendo el más relevante SVM.

Decision Tree funciona consistentemente como el peor de todos los que no son aleatorios, lo más probable es debido a que el dataset no contiene muchas características con las cuales diferenciar adecuadamente las instancias, aprendiendo menos que el resto.

7. Bibliografía

https://scikit-learn.org/stable/user_guide.html
https://towardsdatascience.com/machine-learning-classifiers-comparison-with-python-33149aecdbca