Clasificación de especies de hojas mediante aprendizaje supervisado

Gonzalo Lope Carrasco

Diciembre 2023

Contents

1 Introducción **2**
- 1.1 Descripción del problema 2
- 1.2 Metodología 3

2 Preprocesado de los Datos **4**

3 Resultados **4**
- 3.1 Regresión Logística 4
- 3.2 Árbol De Decisión 5
- 3.3 K Nearest Neighbors 7
- 3.4 SVM 8

4 Discusión **9**

5 Conclusión **10**

1 Introducción

1.1 Descripción del problema

Los datos que he escogido para hacer la clasificación en esta práctica son datos sobre distintos tipos de hojas (Link). Estos datos están incluidos dentro de un archivo csv, que incluye 16 características extraídas de 340 imágenes de las hojas. A continuación, podemos ver cuales son los valores de las 5 primeras características de las 5 primeras hojas.

Class	Eccentricity	Aspect Ratio	Elongation	Solidity	Stochastic Convexity
1	0.72694	1.4742	0.32396	0.98535	1.00000
1	0.74173	1.5257	0.36116	0.98152	0.99825
1	0.76722	1.5725	0.38998	0.97755	1.00000
1	0.73797	1.4597	0.35376	0.97566	1.00000
1	0.82301	1.7707	0.44462	0.97698	1.00000

De cada fotografía, el creador de los datos obtuvo las siguientes 16 características.

1. Class (Species)
2. Specimen Number
3. Eccentricity
4. Aspect Ratio
5. Elongation
6. Solidity
7. Stochastic Convexity
8. Isoperimetric Factor
9. Maximal Indentation Depth
10. Lobedness
11. Average Intensity
12. Average Contrast
13. Smoothness
14. Third moment
15. Uniformity
16. Entropy

Cada hoja pertenece a una especie de planta, que será la variable a predecir a partir del resto de los datos. Existen 30 tipos de planta dentro de estos datos, a continuación podemos ver un ejemplo de hoja para cada uno de ellos (Existen fotos de 40 especies pero solo datos de 30).

1.2 Metodología

Para la predicción del tipo de planta a partir de los datos de cada hoja se preprocesarán los datos, se dividirán entre un 40 datos de test y el resto de entrenamiento. A continuación, se probarán distintos métodos de aprendizaje supervisado como SVM, regresión logística, Arboles de decisión o KNN. A su vez, cabe destacar que en todos los casos se usará la tasa de acierto como medida a optimizar en todos los modelos.

2 Preprocesado de los Datos

Escalo los datos usando el StandarScaler de scikitlearn

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df.drop('Class',axis=1))
scaled_features = scaler.transform(df.drop('Class',axis=1))
df_feat = pd.DataFrame(scaled_features,columns=df.columns[1:])
```

Finalmente, divido los datos en un grupo de test, de 40 muestras, y el resto lo uso para el entrenamiento de los modelos.

```
train_X, test_X, train_y, test_y = train_test_split(df_feat, df['Class'],
↪  test_size=40,random_state=12)
```

3 Resultados

3.1 Regresión Logística

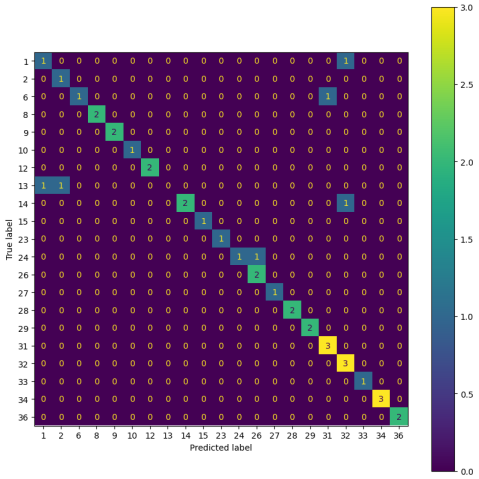
Entreno el modelo de regresión logística probando distintos valores de C

```
logmodel = LogisticRegression(solver='lbfgs', max_iter=10000)
param_grid = {'C': [0.1,1, 10, 100, 1000]}
grid = GridSearchCV(logmodel,param_grid,refit=True,cv=5)
grid.fit(train_X,train_y)
```

Los resultados de clasificación con la partición de test creada son los siguientes:

- Accuracy: 85.00%
- Recall: 85.75%
- Precision: 86.51%

Este modelo usa el mejor valor de C posible que es 100. Podemos ver mejor los aciertos y los fallos en las predicciones en la matriz de confusión:



Se puede observar que, una de las clases con más fallos es la trece, que el modelo confunde con la 1 y la 2, esto cuadra si observamos la Fig.1.1 en la que la hoja 13 se parece a estas otras y es posible que en la imágenes de test incluso más.

3.2 Árbol De Decisión

A continuación, entreno el árbol de decisión usando otra vez GridSearchCV para probar diferentes parámetros como el numero mínimo de instancias por nodo, la profundidad máxima del árbol o el numero mínimo de instancias por partición.

```
from sklearn.tree import DecisionTreeClassifier
param_grid = {'max_depth': np.arange(3, 20), 'min_samples_leaf': [5,20,
↪  40, 30], 'min_samples_split': [5,20, 30, 40]}
grid = GridSearchCV(DecisionTreeClassifier(),param_grid,refit=True,cv=5)
grid.fit(train_X,train_y)
```

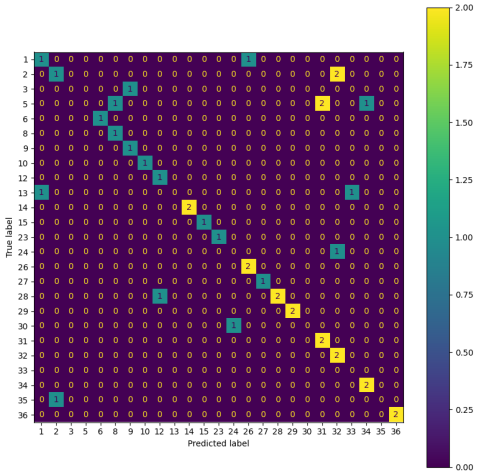
Tras entrenar modelos con todas las posibles combinaciones de parámetros, obtenemos la mejor combinación de estos:

{'max_depth': 15, 'min_samples_leaf': 5, 'min_samples_split': 5}

Cabe resaltar que a pesar de que el valor de la máxima profundidad es de 15, la profundidad real del mejor árbol es de 9. Observamos las métricas sobre las predicciones de los datos del test.

- Accuracy: 65.00%
- Recall: 54.93%
- Precision: 66.00%

Y la matriz de confusión:



Más tarde, compararemos los resultados de este modelo con todos los demás. A pesar de esto todavía podemos interpretar las decisiones para las clases que mejor precisión tienen como por ejemplo la clase 29:

1. Solidity $\leq$ -2.50: False
2. Eccentricity $\leq$ 0.49: False
3. Stochastic Convexity $\leq$ -0.15: False
4. Aspect Ratio $\leq$ 0.92: True
5. Entropy $\leq$ 0.12: True
6. Solidity $\leq$ 0.61: True
7. Uniformity $\leq$ -0.80: True

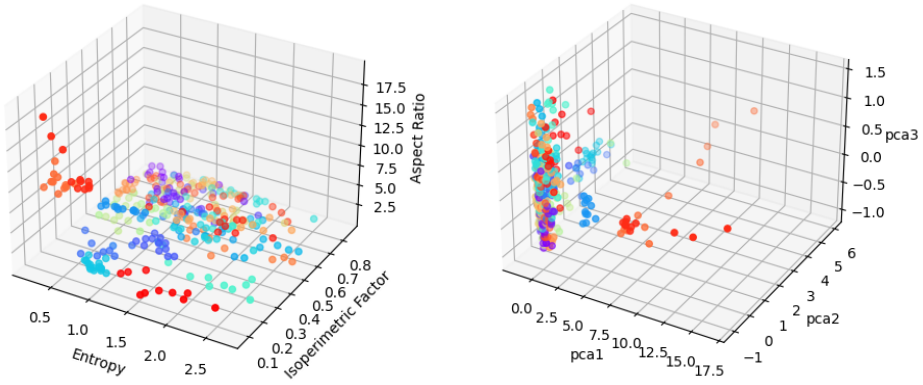
Como podemos observar, el camino que sigue es: ¿Solidity muy baja?: No, ¿es una hoja poco convexa, con Eccentricity (redondeo) bajo?: No, ¿Aspect Ratio bajo?: Si, ¿Entropía baja?: Si, ¿Solidity no muy alta?: Si, ¿Uniformity muy baja?: Si. Esto cuadra con la hoja 29, una hoja bastante equilibrada, redonda y con forma convexa.

3.3 K Nearest Neighbors

Volvemos ha entrenar múltiples modelos usando GridSearchCV, esta vez para obtener que número de vecinos tenemos que tener en cuenta a la hora de predecir la clase de hoja de una instancia dada.

```
from sklearn.neighbors import KNeighborsClassifier
param_grid = {'n_neighbors': np.arange(1, 20)}
grid = GridSearchCV(KNeighborsClassifier(),param_grid,refit=True,cv=5)
grid.fit(train_X,train_y)
```

En este caso, obtenemos que el mejor valor posible para el número de vecinos es 1, lo que significa que por cada instancia, se predice que su clase es la de la instancia de entrenamiento mas cercana. Si pensamos en la naturaleza de los datos, este parámetro tiene sentido ya que existe muy poca variabilidad entre las hojas de un mismo tipo de árbol en la mayoría de los casos y a su vez, existen muchas especies de hojas muy parecidas entre ellas. Todas estas conclusiones se ven reforzadas si vemos los siguientes graficos 3d.

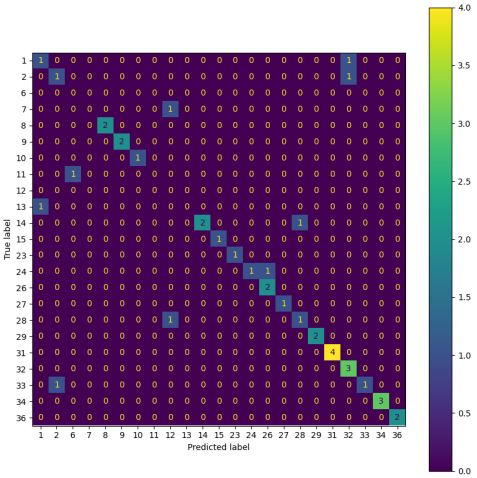


En estos gráficos podemos ver la relación entre las variables 'Entropy','Isoperimetric Factor' y 'Aspect Ratio' y por otro lado, todas las variables reducidas a tres dimensiones mediante PCA. En ambos casos, podemos observar que excepto algunos grupos muy apartados, la mayoría de los puntos están muy juntos, lo que podría justificar el uso de una sola instancia de entrenamiento como discriminante.

Podemos observar las métricas de la clasificación.

- Accuracy: 77.50%
- Recall: 68.55%
- Precision: 65.94%

Y la matriz de confusión:



De esta gráfica cabe destacar sobretodo la buena precisión en los grupos finales de clasificación, en especial el 31, que como observamos en Fig.1.1 es una hoja muy diferente al resto que puede ser uno de estos grupos separados mencionados anteriormente.

3.4 SVM

Finalmente, estudiaremos el uso de SVM o en este caso la variante para clasificación SVC.

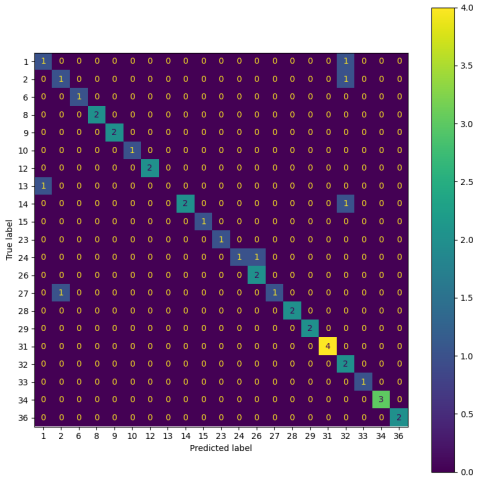
```
from sklearn.svm import SVC
param_grid = {'C': [1000,1500,2000,2500,3000], 'gamma':
↪   [0.01,0.008,0.005,0.002]}
grid = GridSearchCV(SVC(),param_grid,refit=True,cv=5)
grid.fit(train_X,train_y)
```

Tras este entrenamiento, obtenemos que los mejores parametros posibles son:

$$\gamma = 0.008 \; C = 1000$$

Y con estos parámetros obtenemos las siguiente métricas de clasificación y matriz de confusión.

- Accuracy: 85.00%
- Recall: 86.03%
- Precision: 84.13%



Debido al alto número de dimensiones de este modelo, no se pueden interpretar las decisiones que toma el modelo a la hora de clasificar una instancia.

4 Discusión

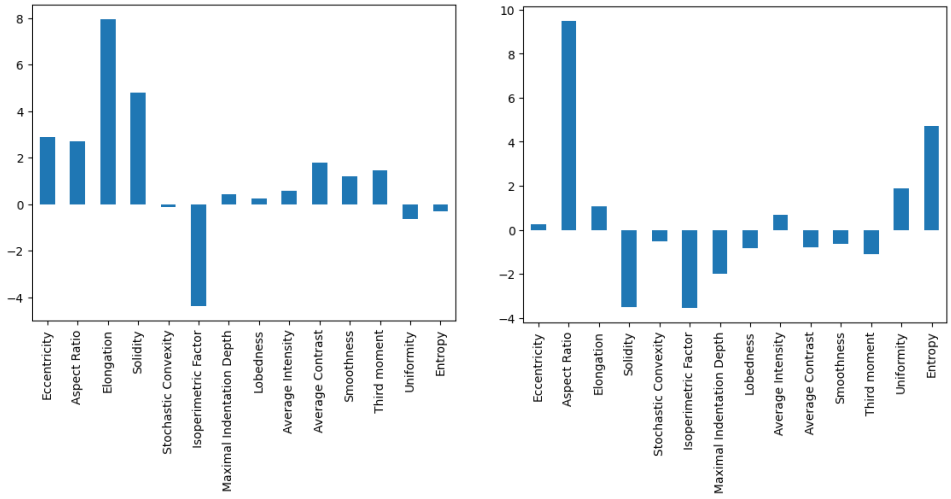
Observemos la siguiente tabla para saber cual es el mejor modelo de predicción.

	Accuracy	Recall	Precision
Regresion logisitca	85%	85.75%	86.51%
Arbol de decision	65%	54.93%	66%
KNN	77.5%	68.55%	65.94%
SVC	85%	86%	84%

Podemos observar que en este caso, los mejores modelos son el de regresión logística y el SVC, los cuales son prácticamente idénticos en cuanto a las métricas, excepto una pequeña diferencia con la precisión y el 'Recall'. Entonces, vamos a escoger el modelo de regresión logística debido al

criterio de la navaja de oca, es decir, debido a que es un modelo mas sencillo e interpretable que el SVM. Ahora pasemos a interpretar alguna de las clasificaciones del modelo de RL.

Se pueden interpretar las decisiones del modelo viendo los valores de los parámetros para cada tipo de hoja. Por ejemplo, para las hojas de la clase 8 y 34 el modelo escoge los siguientes parámetros respectivamente.



Como podemos observar para las hojas de la clase 8 se tiene muy en cuenta los valores de elongation y solidity, mientras que en las de la clase 34 se tiene mucho mas en cuenta el 'Aspect Ratio', esto es razonable si nos fijamos en la Fig.1.1, pues la clase 8 es un tipo de hojas muy alargado y solido que el tipo de hojas 34 que a pesar de ser también alargado es mucho mas remarcable su gran ratio entre su anchura y su altura, es decir, es muy fina.

5 Conclusión

En conclusión, se puede construir un buen modelo de predicción de la especie de una hoja a partir de datos empíricos obtenidos de una imagen de esta.Además, al usar un modelo de regresión logística, podemos interpretar los pesos que da el modelo a cada una de las variables a la hora de tomar una decisión, aumentando así la confianza que puede tener cualquier usuario del modelo.