



Instituto Tecnológico y de Estudios Superiores de Monterrey

Escuela de Ingeniería y Ciencias

Inteligencia Artificial Avanzada para la Ciencia de Datos

Ingeniería en Ciencias de Datos y Matemáticas

Music Genre Classification

Francia García Romero

A01769680

Monterrey, Nuevo León. 10 de septiembre 2023

Introducción

La toma de decisiones basadas en datos se han convertido en elementos esenciales para organizaciones de todos los sectores. La capacidad de clasificar y predecir eventos futuros a partir de datos históricos es una habilidad valiosa que puede tener una amplia gama de aplicaciones.

La clasificación es una tarea fundamental en el aprendizaje automático que se utiliza para etiquetar o categorizar datos en clases o categorías discretas. Estos modelos son ampliamente aplicables en diversos campos, desde la detección de spam en el correo electrónico hasta el diagnóstico médico, la segmentación de clientes y la detección de fraudes financieros.

En este reporte se tiene como propósito analizar y comparar algunos algoritmos de clasificación, evaluar su rendimiento y comprender sus ventajas y limitaciones en un contexto de clasificación de canciones.

Los datos utilizados para este proyecto fueron obtenidos de [Music Genre Classification](#)). Este conjunto de datos, el cual consta de 17,996 registros y 17 columnas, está compuesto por variables numéricas continuas y categóricas. Cada registro corresponde a una canción que esta clasificada dentro de uno de 10 géneros musicales por la variable *Class*:

Etiqueta	Género
0	Acoustic/Folk
1	AltMusic
2	Blues
3	Bollywood
4	Country
5	HipHop
6	Indie Alt
7	Instrumental
8	Metal
9	Pop

Abordaremos cuestiones fundamentales, como la selección de características, la preparación de datos, el análisis de sesgo y varianza en los resultados, así como su relación con el manejo de *overfitting* y *underfitting* en el modelo.

Las métricas que se utilizarán para la evaluación de los modelos serán:

- **Accuracy:** Número de aciertos correctos.

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- **Precision:** Un mayor *precision* indica un bajo número de FP.

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** Un mayor *recall* indica un bajo número de FN.

$$recall = \frac{TP}{TP + FN}$$

- **Matriz de Confusión:** La diagonal de la matriz indica las clasificaciones de cada clase realizadas correctamente, y el resto de los valores corresponden a instancias de una clase clasificadas como otra. El valor en la posición (i, j) es la cantidad de miembros de la clase i clasificados como pertenecientes a la clase j.

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$$

En estas métricas, TP y TN hacen referencia a *true positives* y *true negatives* respectivamente, y son las clasificaciones correctas, mientras que FP y FN (*false positives* y *false negatives*) hacen referencia a los errores de cada tipo [2].

Preprocesamiento de los Datos

El conjunto de datos original contaba con 16 columnas (14 numéricas continuas y 2 categóricas nominales), la columna de *Class* y 13047 registros (ver figura 1). Se manejaron datos faltantes y atípicos para poder analizar las variables, y se realizó una estandarización de los valores. Para las columnas numéricas se realizó un análisis de correlación para descartar aquellas variables dependientes entre sí, asimismo se realizó un análisis de varianza para evaluar el efecto de la categoría del género en cada variable. Una vez realizado el análisis, se obtuvo que las variables que tenían una relación más significativa con el género de las canciones fueron: *danceability*, *loudness*, *speechiness*, *acousticness*, *valence* y *duration_in min/ms* (ver figura 2).

	Artist Name	Track Name	Popularity	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_in min/ms	time_signature	Class
0	Bruno Mars	That's What I Like (feat. Gucci Mane)	60.0	0.854	0.564	1.0	-4.964	1	0.0485	0.01710	NaN	0.0849	0.899	134.071	234596.0	4	5
2	The Raincoats	No Side to Fall In	35.0	0.434	0.614	6.0	-8.334	1	0.0525	0.48600	0.000196	0.3940	0.787	147.681	109667.0	4	6
3	Deno	Lingo (feat. J.I & Chunkz)	66.0	0.853	0.597	10.0	-6.528	0	0.0555	0.02120	NaN	0.1220	0.569	107.033	173968.0	4	5
5	The Stooges	Search and Destroy - Iggy Pop Mix	53.0	0.235	0.977	6.0	0.878	1	0.1070	0.00353	0.006040	0.1720	0.241	152.952	208133.0	4	6
6	Solomon Burke	None Of Us Are Free	48.0	0.674	0.658	5.0	-9.647	0	0.1040	0.40400	0.000001	0.0981	0.677	143.292	329387.0	4	2

Figura 1: Conjunto de datos inicial.

	danceability	loudness	speechiness	acousticness	valence	duration_in min/ms	Class
8676	0.407002	0.594976	0.024848	0.684549	0.642969	0.494506	9
11957	0.404814	0.710302	0.259259	0.002242	0.018736	0.503115	8
1401	0.507659	0.430539	0.535865	0.185622	0.494119	0.450064	5
4951	0.493435	0.423370	0.085091	0.004174	0.033413	0.815886	6
3124	0.464989	0.316624	0.039850	0.944206	0.067555	0.398829	9
...
13116	0.176149	0.558058	0.148148	0.000051	0.090247	0.733767	8
5770	0.226477	0.698956	0.587436	0.007167	0.317165	0.229141	8
13199	0.335886	0.762856	0.088842	0.000078	0.709587	0.375854	8
15057	0.575492	0.595371	0.041491	0.273605	0.867805	0.388293	6
10425	0.647702	0.318431	0.055321	0.714592	0.323410	0.482999	5

Figura 2: Conjunto de datos listo para entrenamiento del modelo.

Regresión Logística

En la primera implementación de modelos de predicción, se considerará el problema como uno de **clasificación binaria**, en donde el propósito será clasificar las canciones entre aquellas que pertenezcan a un género (9: Pop) y las que no. Para esto, se reestablecerán las etiquetas de la variable *Class* con 0's y 1's: 1 si el valor es igual a 9, 0 para cualquier otro valor. Una vez realizado esto, se dividirá el dataset completo en set de entrenamiento y de pruebas (70% para entrenamiento y 30% para pruebas). Se establece una tasa de aprendizaje de 0.0001 y 2000 *epochs* para el entrenamiento. Una vez realizado el ajuste del modelo a los datos, se procede a la evaluación del modelo. Se realizan predicciones para los datos de entrenamiento, con lo cual se obtienen las siguientes métricas:

Accuracy: 0.52
Precision: 0.50
Recall: 0.002

Para las predicciones realizadas con los datos de pruebas, se obtiene lo siguiente:

Accuracy: 0.47
Precision: 0.51
Recall: 0.005

Como se puede observar, en general no hay un buen desempeño al implementar el modelo para este conjunto de datos en específico. Podemos observar que hubo un *performance* deficiente tanto para los datos de entrenamiento, como para los datos de pruebas, por lo que el **sesgo** o *bias* del modelo es muy alto. Por otro lado, la diferencia entre el error de las predicciones con los datos de entrenamiento y el error de las predicciones con los datos de prueba es bajo, es decir, en ambos casos se tuvieron métricas con valores parecidos, por lo que el modelo tiene una baja **varianza**. Esto nos lleva a la conclusión de que tenemos un caso de *underfitting* en el modelo.

Lo anterior se debe principalmente a que, al ser un dataset con 10 categorías originalmente, hay un gran desbalance en la cantidad de canciones que si pertenecen al genero *Pop*, y a las que no. Es por esto que aunque se tiene un *accuracy* del 50%, el *recall* es del 0.2%: al haber demasiados casos en donde la canción no pertenece a la clase, el modelo aprendió que clasificando la mayoría como que no pertenece, va a acertar con más predicciones.

Para abordar el problema de *underfitting* existente en el modelo, se utilizarán 2 técnicas: balanceo de clases y aumento de complejidad del modelo.

Over/Under Sampling

Para el balanceo de clases se eliminan 6 de las 10 clases en el dataset, ya que, en conjunto, solamente comprenden el 20% de los datos.

Class	Distribución (%)
6	19.828313
9	19.345443
8	14.210163
5	11.090672
1	10.523492
2	9.749368
0	4.790373
7	4.414808
3	3.081168
4	2.966199

Con las clases restantes, la distribución queda de la siguiente manera:

Class	Distribución (%)
6	30.753685
9	30.004755
8	22.039943
5	17.201617

Debido a que ya no hay clases que puedan considerarse insignificantes, utilizando *oversampling* para las clases 8 y 5, y *undersampling* para las clases 6 y 9, es posible modificar el *dataframe* de tal manera que la distribución final sea del 25% para cada clase. Esto se realiza tomando una fracción aleatoria de registros de las clases predominantes, y duplicando registros de manera aleatoria para las otras clases. Habiendo realizado esto, puede implementarse una **regresión logística multiclase**, que sería un modelo lineal de mayor complejidad que el anterior.

Regresión Logística Multiclase

A diferencia del modelo anterior, en este caso se tienen 4 clases con igual distribución para clasificación. Utilizando la librería *sklearn* en Python, se realiza una implementación del modelo utilizando como hiper-parámetros principales una regularización L2 y un máximo de 50 iteraciones. La división del dataset se establece nuevamente como 30% para pruebas y 70% para entrenamiento. Utilizando la configuración mencionada, se obtienen los siguientes resultados para las predicciones del set de entrenamiento:

Accuracy: 0.6546
Precision: 0.6464
Recall: 0.6539
Matriz de Confusión:

$$\begin{bmatrix} 898 & 72 & 30 & 175 \\ 85 & 448 & 278 & 285 \\ 4 & 160 & 891 & 23 \\ 141 & 256 & 50 & 717 \end{bmatrix}$$

Los resultados para las predicciones del set de pruebas son:

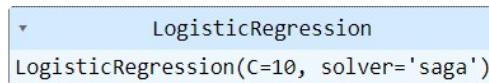
Accuracy: 0.6615
Precision: 0.6519
Recall: 0.6602
Matriz de Confusión:

$$\begin{bmatrix} 384 & 35 & 13 & 62 \\ 45 & 200 & 110 & 136 \\ 0 & 59 & 416 & 19 \\ 54 & 94 & 28 & 280 \end{bmatrix}$$

Nuevamente podemos observar un gran sesgo (*bias*) y baja varianza en los resultados, lo que significa que el modelo no está propiamente ajustado a los datos (*underfitting*). Para poder evaluar si el modelo puede ser mejorado, o si se necesita de un modelo más complejo, se utilizará **Grid Search** para ajustar los hiper-parámetros del modelo de acuerdo a los datos,

Grid Search

Se hará uso de la función *GridSearchCV* de *sklearn* [4][5], la cual internamente utiliza *cross-validation* para ajustar los hiper-parámetros del modelo. Según las métricas promedio obtenidas haciendo la rotación del set de validación en cada repetición del entrenamiento para distintos valores de parámetros, la función evalúa cuáles son los mejores valores para configurar el modelo. Al implementar esta búsqueda con validación cruzada de 5 subconjuntos para regresión logística se obtiene que la configuración óptima de parámetros es:



```
LogisticRegression(C=10, solver='saga')
```

Figura 3: Grid Search para Regresión Logística Multiclase

Utilizando la configuración propuesta, obtenemos la siguiente evaluación para las predicciones con el set de entrenamiento y de pruebas, respectivamente:

Accuracy: 0.6554
Precision: 0.6473
Recall: 0.6549
Matriz de Confusión:

$$\begin{bmatrix} 900 & 79 & 24 & 172 \\ 85 & 449 & 276 & 286 \\ 3 & 155 & 897 & 23 \\ 142 & 266 & 44 & 712 \end{bmatrix}$$

Accuracy: 0.6594
Precision: 0.6491
Recall: 0.6580
Matriz de Confusión:

$$\begin{bmatrix} 383 & 36 & 13 & 62 \\ 46 & 200 & 113 & 132 \\ 0 & 59 & 417 & 18 \\ 58 & 96 & 26 & 276 \end{bmatrix}$$

Nuevamente hay poca varianza y alto sesgo, por que el *underfitting* identificado probablemente se deba a la baja complejidad intrínseca al modelo. Por lo mismo, el siguiente paso es probar un modelo que no sea lineal.

Random Forest

Utilizando la misma proporción para el set de entrenamiento y set de pruebas, se entrena el modelo utilizando 200 estimadores y sin límite de profundidad. Para el set de entrenamiento se obtienen los siguientes resultados:

Accuracy: 0.9918
Precision: 0.9920
Recall: 0.9919
Matriz de Confusión:

$$\begin{bmatrix} 1167 & 0 & 0 & 8 \\ 0 & 1085 & 0 & 11 \\ 0 & 0 & 1077 & 1 \\ 9 & 7 & 1 & 1147 \end{bmatrix}$$

Para el set de pruebas:

Accuracy: 0.8222
Precision: 0.8181
Recall: 0.8212
Matriz de Confusión:

$$\begin{bmatrix} 447 & 17 & 5 & 25 \\ 32 & 325 & 56 & 78 \\ 0 & 21 & 463 & 10 \\ 31 & 60 & 9 & 356 \end{bmatrix}$$

En este caso, tenemos que las predicciones con el set de entrenamiento tienen un 99% de accuracy. Ya que el bias es muy bajo para el set de entrenamiento, y relativamente bajo para el set de pruebas, pero la varianza entre ambos resultados es de casi 20%, es posible que, aunque los resultados sean buenos, el modelo tenga un cierto grado de *overfitting*. Si este es el caso, quiere decir que el modelo no está generalizando apropiadamente y únicamente se tenga un set de pruebas "afortunado". Para poder manejar esto se utilizarán 2 técnicas: selección de variables y disminución de complejidad del modelo utilizando cross-validation.

Selección de Features

Para poder realizar esta selección deben considerarse todas las combinaciones de las variables con las que contamos. Con cada una de estas combinaciones se entrena el modelo y se evalúa con qué combinación se tuvieron mejores resultados. En este caso, utilizando *accuracy* como indicador, se obtuvo que la mejor combinación es el set completo de variables que ya se tienen.

Cross-Validation

Para este caso, se establece un rango de valores para los parámetros de profundidad máxima y cantidad de estimadores. Posteriormente, se entrenan modelos con cada posible combinación, y se evalúan con validación cruzada utilizando la función *KFolds* de *sklearn* [1][3]. Primero, se obtiene el error del modelo manteniendo el número de estimadores como 100 y variando el límite de profundidad (ver figura 4).

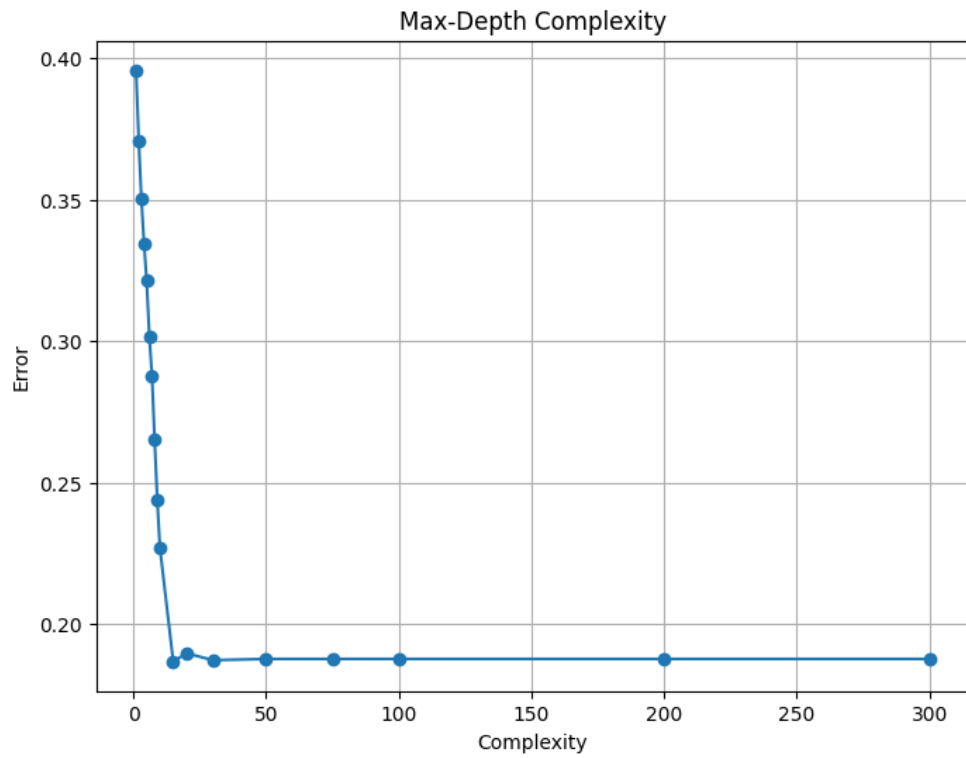


Figura 4: Error en función del límite de profundidad.

De igual manera, se obtiene el error manteniendo constante el limite de profundidad como 200, y variando el número de estimadores permitido (ver figura 5).

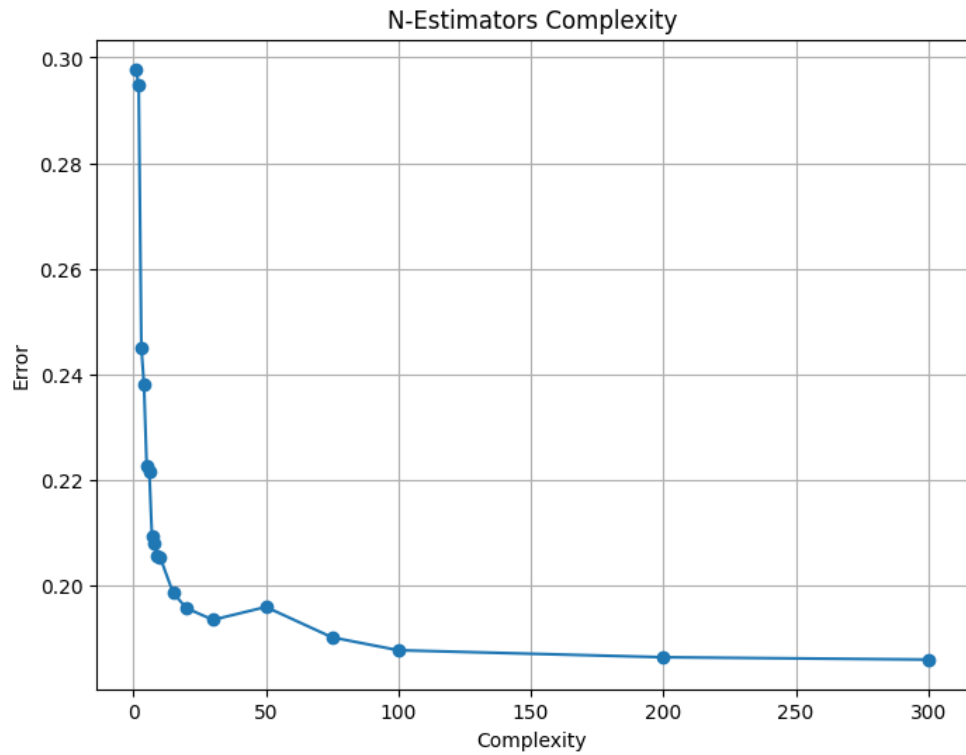


Figura 5: Error en función del número de estimadores.

Calculando el error según todas las combinaciones posibles de los valores contemplados para la cantidad de estimadores y la profundidad máxima (ver figura 6), se puede obtener que la mejor combinación de parámetros es una profundidad máxima de 15 con 20 estimadores. Esto coincide con el comportamiento visto en las figuras 4 y 5.

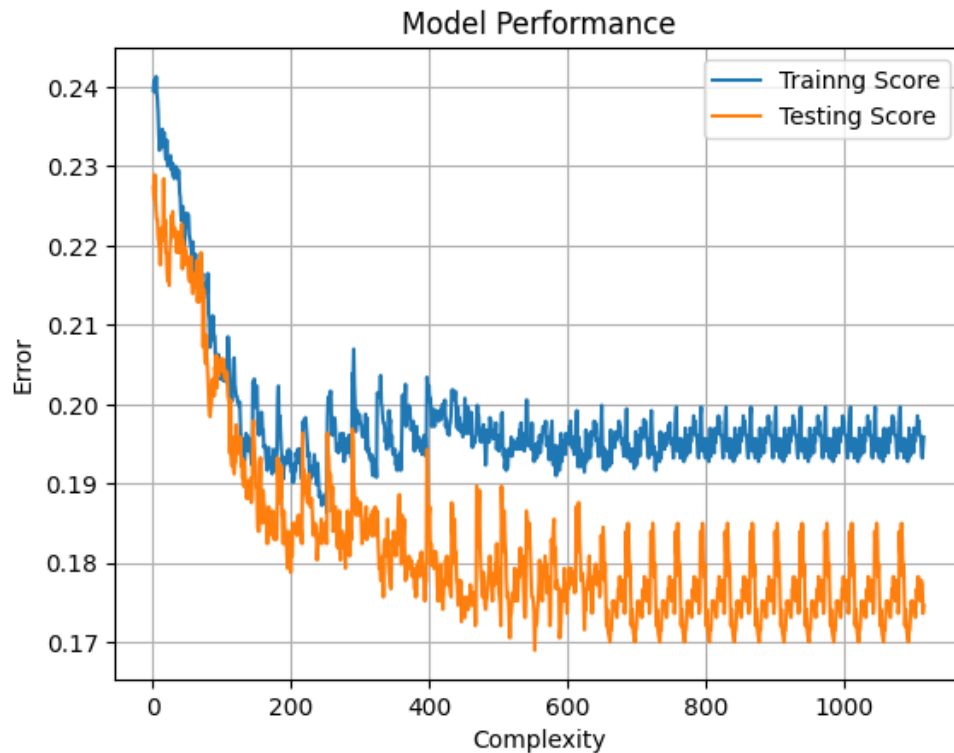


Figura 6: Error en función de la complejidad del modelo.

El criterio para poder llegar a la conclusión de que esta es la mejor combinación es el siguiente: se necesita de un par de valores en donde el error sea lo menor posible, sin que la diferencia entre los resultados del conjunto de entrenamiento y el de pruebas comiencen a divergir significativamente.

Implementando el modelo con los hiper-parámetros encontrados, se obtienen los siguientes resultados para el set de entrenamiento:

Accuracy: 0.8078
Precision: 0.8042
Recall: 0.8068

Para el set de pruebas se obtiene:

Accuracy: 0.8057
Precision: 0.8035
Recall: 0.8054

En esta implementación del modelo, ya no existe un sesgo significativo ni varianza alta entre los resultados, lo que quiere decir que el *performance* del modelo es eficiente, y que el modelo está generalizando en vez de aprender el ruido del conjunto de datos. En otras palabras, se ha logrado contrarrestar el *underfitting* y *overfitting* encontrado en las implementaciones previas para llegar al punto de equilibrio deseado.

Conclusión

Después de implementar regresión logística binaria y multiclase para el problema de clasificación de canciones por género musical, se observó que el problema requería de un modelo más robusto, por lo cual terminó por implementarse un clasificador del tipo Random Forest. En cada uno de los modelos iniciales, sin embargo, hubo situaciones de under y overfitting correspondientemente. Para poder abordar estas situaciones en cada caso y poder llegar a un modelo con un performance aceptable, se realizó un análisis del sesgo y la varianza en los resultados de los modelos con el conjunto de datos de entrenamiento y de pruebas. Esto fue una pieza clave en la detección de overfitting y underfitting, y permitió tomar las medidas necesarias para ajustar el modelo de una mejor manera haciendo uso de técnicas como selección de features, disminución de complejidad y cross-validation para overfitting, y aumento de la complejidad para underfitting.

Referencias

- [1] Cross-validation: evaluating estimator performance. (n.d.). Scikit-Learn. Retrieved September 10, 2023, from https://scikit-learn.org/stable/modules/cross_validation.html
- [2] Heras, J. M. (2019, November 17). Precision, Recall, F1, Accuracy en clasificación. Iartificial.net. <https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/>
- [3] Parra, F. (n.d.). 6 Métodos de clasificación. Bookdown.org. Retrieved September 10, 2023, from <https://bookdown.org/content/2274/metodos-de-clasificacion.html>
- [4] Sklearn.Model_selection.GridSearchCV. (n.d.). Scikit-Learn. Retrieved September 10, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- [5] Supervised learning. (n.d.). Scikit-Learn. Retrieved September 10, 2023, from https://scikit-learn.org/stable/supervised_learning.html