Facultad de Informática de
Barcelona

Detección de Púlsares

Aprendizaje Automático

Autores:
Ferran Velasco Olivera
Joaquín Gómez Sánchez

Enero 2020

Índice

1. Introducción

Los púlsares son estrellas de neutrones caracterizadas por emitir radiación de forma periódica de acuerdo a su periodo de rotación y la intensidad del campo magnético que provocan.

El interes por su estudio es debido a que la existencia de estas estrellas recae en la presión de degeneración cuántica. Es decir, no se convierten en un agujero negro gracias a que el principio de exclusión de Pauli, el cual dice que dos fermiones con los mismos números cuanticos no pueden existir en el mismo punto, evita que la estrella se siga comprimiendose hasta convertirse en un agujero negro, es decir, esta exclusion actua como un mecanismo de presión inversa.

1.1. Objetivo

El objetivo del presente trabajo es realizar la clasificación de estrellas en púlsares y no púlsares. Para ello se emplea el **Perfil de la estrella**, es decir, la densidad de flujo por cada fase; y la curva **Medida de Dispersion-Relación señal-ruido** (curva DM-SNR).

1.2. Conjunto de datos

Los datos utilizados [HTRU2], obtenidos durante el proyecto *High Time Resolution Universe Survey (South)*, presentan por cada fila una caracterización de la estrella, y si es un púlsar o no.

Dicha caracterización se realiza mediante las siguientes medidas del perfil integrado del púlsar y de la curva DM-SNR:

- Media del perfil integrado
- Desviación estandar del perfil integrado
- Kurtosis excesivo del perfil integrado
- Skewness del perfil integrado
- Media de la curva DM-SNR
- Desviación estandar de la curva DM-SNR
- Kurtosis excesivo de la curva DM-SNR
- Skewness de la curva DM-SNR

2. Trabajos relacionados

En la red se pueden encontrar distintos trabajos que emplean el mismo conjunto de datos para estudiar diversos métodos de aprendizaje automático. Los hay con finalidad puramente educativa y también científica.

El trabajo previo más relevante es el original [Lyon] sobre los datos que nos ocupan. En el se realiza una exhaustiva explicación sobre la obtención de estos, los diferentes *features* que lo componen, además de exponer ejemplos de utilización en técnicas de *Machine Learning*.

3. Exploración de los datos

En esta sección se analizan los datos para conocerlos mejor y enfocar de forma adecuada el diseño de los modelos de clasificación que se desarrollaran.

3.1. Preprocessing

Como se ha mencionado, el *data set* [HTRU2] empleado esta compuesto de 17.898 ejemplos, con un total de 1.639 púlsares. Como se puede observar, la proporcion de púlsares es del 9,16 %.

Todas las columnas del *data set* tienen todos los valores y ninguna presenta valores incorrectos o mal formados, por lo que no se ha requerido ningún tipo de manipulación para su corrección.

Como se puede observarvar en la Tabla 1 y la Tabla 2 los diferentes *features* del *data set* difieren mucho en cuanto a valores medios por cada clase, avarcando a su vez valores muy amplios como indican las desviaciones estdandar. Debido a esto, se ha decidido no aplicar ningún tratamiento de *outliers*, ya que podría afectar a la calidad de conjunto de los datos, dada la amplia dispersión que presentan, comparando una clase con la otra.

	Mean		**Stdev**		**Skewness**		**Kurtosis**	
	μ	σ	μ	σ	μ	σ	μ	σ
Púlsar	56,69	30,00	38,71	8,0	3,13	1,87	15,55	13,99
No púlsar	116,56	17,47	47,33	6,18	0,21	0,33	0,38	1,02

Tabla 1: Medias y desviaciones de la caracterización del perfil

	Mean		**Stdev**		**Skewness**		**Kurtosis**	
	μ	σ	μ	σ	μ	σ	μ	σ
Púlsar	49,82	45,28	56,46	19,73	2,75	3,10	17,93	50,89
No púlsar	8,86	24,41	23,28	16,65	8,86	4,23	113,62	106,72

Tabla 2: Medias y desviaciones de la caracterización de la curva DM-SNR

Como muchas de las técnicas de Machine Learning son sensibles a la escala de los datos [Grus], y vista la variabilidad de estos, se ha decidido estandarizarlos, con tal de reducir el efecto de la escala de cada *feature* en los modelos.

3.2. Feature extraction/selection

Como se puede observar en la Figura 1 los *features* $Profile\_skewness - Profile\_kurtosis$ y $DM\_skewness - DM\_kurtosis$ presentan una correlación superior a 0.9.

	Profile_mean	Profile_stdev	Profile_skewness	Profile_kurtosis	DM_mean	DM_stdev	DM_skewness	DM_kurtosis
Profile_mean	1	0.547137	-0.873898	-0.738775	-0.298841	-0.307016	0.234331	0.144033
Profile_stdev	0.547137	1	-0.521435	-0.539793	0.00686873	-0.0476316	0.0294294	0.0276915
Profile_skewness	-0.873898	-0.521435	1	0.945729	0.414368	0.43288	-0.341209	-0.214491
Profile_kurtosis	-0.738775	-0.539793	0.945729	1	0.412056	0.41514	-0.328843	-0.204782
DM_mean	-0.298841	0.00686873	0.414368	0.412056	1	0.796555	-0.615971	-0.354269
DM_stdev	-0.307016	-0.0476316	0.43288	0.41514	0.796555	1	-0.809786	-0.5758
DM_skewness	0.234331	0.0294294	-0.341209	-0.328843	-0.615971	-0.809786	1	0.923743
DM_kurtosis	0.144033	0.0276915	-0.214491	-0.204782	-0.354269	-0.5758	0.923743	1

Figura 1: Matriz de Correlaciones

Puesto que algunos métodos se ven afectado por los *features* irrelevantes, o por correlaciones elevadas entre ellos [Kotsiantis14], se ha decidido comparar la calidad de los modelos entrenados con el *data set* completo y con un subconjunto de este sin variables correlacionadas, pudiendo asi determinar el conjunto de *features* adecuados para cada modelo.

Para la selección de *features* no correlacionados, se ha procedido eliminando uno de dos correlacionados si y solo si estos presentan una correlación superior a 0.9.

3.3. Visualization

Como contamos con un dataset de 8 *features*, para poderlo visualizar realizamos *Principal Component Analysis*, proyectando los datos en un espacio de 2 dimensiones.

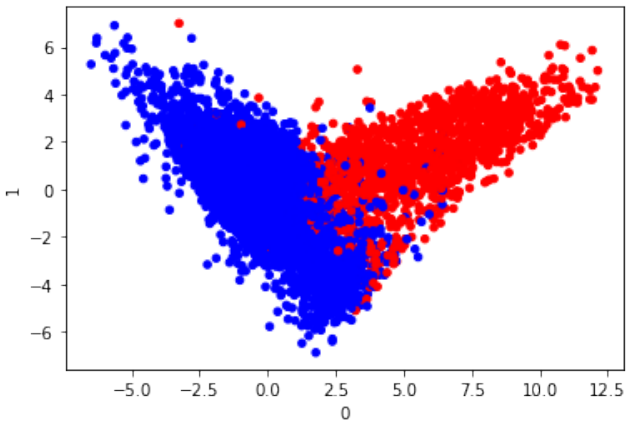

Figura 2: Scatter plot del PCA (2-dimensiones)

Como se puede observar en la Figura 2 existen a simple vista 2 grupos de datos (azul para los no púlsares y rojo para los púlsares). Se puede apreciar a demás que la franja entre ambos grupos parece ser clara, pudiendose dibujar una linea entre ambas clases, aunque existe una clara mezcla entorno a la linea de división. Se puede observar también como algunos puntos de ambas clases directamente escapan del conglomerado al que pertenecen.

3.4. Clustering

Por lo mencionado sobre en la Figura 2 del apartado anterior, cabría esperar la existencia de 2 clusters.

Scikit-learn proporciona diferentes métodos de clustering. Se ha decidido emplear k-means ya que trabaja correctamente con un número elevado de muestras y con un número medio de clusters [Scikit:Clustering], además de ser un método de propósito general. Para medir la calidad del clustering se ha empleado el índice de Calinski-Harabasz (CH).

Ejecutando 50 veces k-means para $k \in [2.,12]$, realizando la media de las ejecuciones, se ha obtenido que la mejor k par el conjunto de datos estandarizados es $k = 4$.

Como k-means trabaja con la distancia euclidiana, y puesto que esta tiende a ïnflarseçon dimensiones elevadas [Scikit:Clustering], realizamos el clustering sobre el conjunto de datos tras realizar PCA.

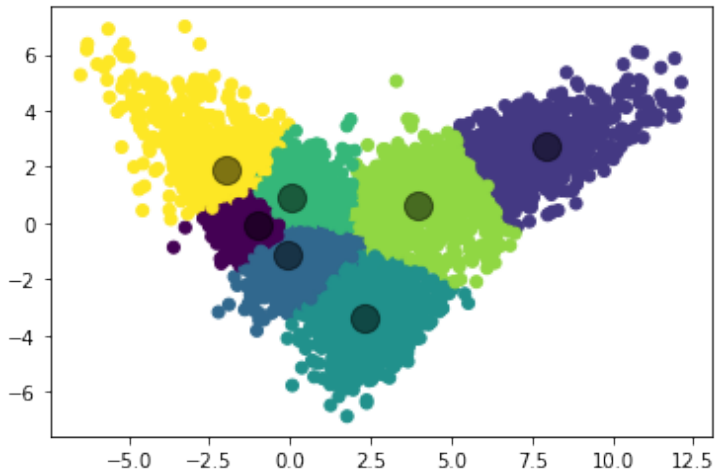

Figura 3: Scatter plot del k-means de PCA

El valor óptimo obtenido es $k = 7$ (Figura 3), con un sub-óbtimo destacado para $k = 4$. Esto nos permite darnos cuenta de la complejidad del problema, donde la variabilidd de los datos no púlsares es muy amplia, y aunque diferenciada de

los púlsares, presenta subconjuntos. El conjunto de datos de púlsares parece ser más homojeneo, con dos subconjuntos en él.

4. Resampling Protocol

El protocolo de remuestreo empleado en todo momento en el trabajo ha sido *10-Fold Cross Validation*. Este se ha aplicado en el entrenamiento de los diferentes métodos sobre un subconjunto de los datos, el *training set*, del 80 % de los ejemplos, reservado el 20 % restantes para el *test set*.

Se ha decidido emplear $k = 10$ ya que al tener 17.898 ejemplos, aunque se haya empleado CV con el 80 % de los datos, queda jsutificado que son suficientes como para emplear dicha k.

Excepcionalmente, no se ha utilizado este método para el modelo de Random Forests, ya que para optimizar los parámetros de este, se ha optado por utilizar como guía la medida *Out of Bag* (OOB), ya que supone un coste computacional menor y la fiabilidad del resultado es prácticamente la misma.

5. Modelos de Clasificación

Los modelos de clasificación escogidos para el presente trabajo son: Nearest-neighbor, Logistic Regression, Lineas Support Vector Machines; en cuanto a modelos lineales, y Redes MLP y Random Forest, en cuanto a modelos no lineales.

Se decidido estudiar la adecuación de estos modelos para el problema presentado de forma gradual, es decir, empleando desde modelo menos complejos a modelos más complejos, con tal de poder comparar tanto la adecuación de estos al problema, como la necesidad de emplear modelos más o menos complejos.

Para la modelización se ha empleado únicamente la librería scikit-learn de Python. Como librerías de apoyo se han empleado Pandas, Numpy, Matplotlib, Seaborn, Pickle, StatsModels y Graphviz.

5.1. Métodos Lineales

5.1.1. Nearest-neighbors

Scikit-learn ofrece dos tipos de métodos nearest-neighbor supervisado:

- **KNeighborsClassifier**: Algoritmo basado en el número k de vecinos.
- **RadiusNeighborsClassifier**: Algoritmo basado en el número de vecinos dentro de un radio fijo de cada punto de entrenamiento.

Se ha decidido emplear el primero ya que es el algoritmo básico estandard de propósito general.

Este método permite ajustar diferentes parámetros del modelo: el **algoritmo para el computo de los vecinos más cercanos** (BallTree, KDTree, Brute-force Search), el **tipo de métrica a emplear en las distancias** (Manhattan o Euclidea), y la **forma asignar pesos** (uniforme o inverso de la distancia).

Empleando *GridSearchCV*, como se ha descrito en la Sección 4, se ha buscado la mejor parametrización del modelo, empleando como posibles valores de vecinos el rango $[2, 50]$.

Entrenando con los parámetos descritos, y los datos estandarizados, se ha encontrado que el mejor modelo, con un error de training medio de 2.11 %, es el que cuenta la parametrización de la tabla 3.

k	**Weights**	**Algoritmo**	**Métrica**
18	distance	Ball Tree	Euclidea

Tabla 3: Mejores parámetros obtenidos de KNeighborsClassifier

En test el error del mejor modelo encontrado tiene un ligero aumento hasta 2.21 %, lo cual se puede considerar muy aceptable dada la complejidad del problema y la sencillez de este primer modelo.

Cabe destacar, como se podría esperar y se puede observar en la Tabla 4, la clase Púlsar es la más perjudicada en predicciones, ya que es la que cuenta con menos ejemplos.

Predicción Real	**No púlsar**	**Pulsar**
No púlsar	3230	19
Pulsar	60	271

Tabla 4: Matriz de confusión del mejor modelo KNN en test

Buscando el mejor modelo, pero con los *features* seleccionados, es decir, sin los correlacionados, se ha obtenido el mismo hypertuning óptimo, pero con $k = 17$.

Entrenando con los dato sin correlación se ha obtenido el mejor modelo con un error de training medio igual que con los datos correlacionados, pero un error de test muy ligeramente superior, 2.23 %.

Ni el error de test en ambos, ni el tiempo de entrenamiento medio (0.092s con *features* correlacionados y 0.081s sin correlacionados), son muy determinantes para decantarse por un conjunto de *features* u otro.

5.1.2. Logistic Regression

Se ha decidido emplear regresión logística debido a que la salida que queremos predecir es binaria y resulta intuitivo utilizar esta técnica para dicho tipo de predicciones.

La librería scikit-learn ofrece un método de regresión logísitca llamado LogisticRegression. Este permite introducir una serie de parámetros que pueden ser ajustados. Para el presente trabajo, los parámetros que se han considerado más relevantes para estudiar son: **la norma utilizada para la penalización (L1 o L2)**, *penalty*; **el parámetro de regularización inversa**, C; y **el algoritmo a utilizar en la optimización del modelo**, (*liblinear* o *saga*).

Se ha empleado *GridSearchCV* con valores posibles de C entre 1×10^{-4} y 1×10^{4} generados en una escala logaritmica.

Entrenando el modelo con los parámetros indicados y utilizando los datos estandarizados, el mejor modelo que se ha encontrado con CV tiene como parámetros: penalty L1, el algoritmo *liblinear* y el valor de $C = 4{,}28$. Curiosamente, para los datos sin *features* correlacionados, los parámetros obtenidos resultan ser los mismos, lo que nos indica que la extracción de los *features* es correcta ya que claramente no influyen en el modelo.

El error de training medio obtenido en el CV utilizando los datos estandarizados con todos los *features* es de 2.06 % y de 2.12 % para los datos sin los *features* correlacionados.

En el caso del error de testing, con el modelo obtenido indicando los parámetros optimizados, es de 2.23 % para los datos estandarizados. Para los datos sin *features* correlacionados, en cambio, el error de testing sufre un ligero aumento hasta 2.56 %.

Dada la reducción de complejidad del problema que supone la eliminación de las *features*, el incremento del error en el segundo caso no supone un problema considerable. No obstante, debido a que el error es menor en el primer modelo y que el tiempo de ejecución de ambos resulta muy parecido, está justificada la decisión de quedarse con este antes que con el segundo.

5.1.3. Linear Support Vector Machine

Se ha decidido emplear el kernel lineal en SVM ya que, dadas las conclusiones extraidas del análisis de los datos, estos parecen estar suficiente separados y tener una distribución adecuada para el kernel lineal, por la visualización en dos dimensiones.

Scikit-learn, mediante el método LinearSVM permite hypertunear dos parámetros de especial relevancia: el **parámetro de regularización** C y el **peso de las clases** (cada clase el mismo peso o balanceado).

Empleando GridSearchCV, con posibles valores de $C \in [0{,}001, 0{,}01, 0{,}1, 1, 10, 100, 1000]$, y acotando el número de iteraciones posibles de entrenamiento a 5.000; se ha encontrado que el mejor modelo es el que contempla $C = 1$ y el mismo peso en ambas clases.

El error de training medio con dichos parámetros es 2.15 %, y el de test 2.37 %. Igual que con los modelos anteriores la precisión del modelo para no púlsar es mayor, 98 %, que la de púlsar, 93 %.

Entrenando con los datos sin *features* correlacionados se obtiene que el mejor modelo es con $C = 100$ y con el mismo peso en ambas clases. En este caso se presenta un mejor error en test, 2.29 %, consiguiendo un aumento en el *recall* de la clase púlsar. De 80 % a 82 %, manteniendo el *recall* de la clase no púlsar en 99 %.

5.2. Métodos no lineales

5.2.1. Redes MLP

Para el desarrollo de redes MLP, scikit-learn ofrece suficiente potencia como para no ser necesario recurrir a otras librerías, por lo que emplearemos el método MLPClassigier de scikit.

Este método ofrece varios parámetros a ajustar, de los cuales se trabajará únicamente con: **neuronas de las capas interiores**, **función de activación** y α.

Otro parámetros interesantes podrían ser el *solver*. Se empleará el disponible por defecto, *adam*, ya que trabaja correctamente con *datasets* relativamente grandes en cuanto a *samples* como HTRU2 [Scikit:MLPClassifier].

El resto de parámetros, como el learning rate o el momentum, o no estan disponibles para *adam* o se han considerado no relaventes para el problema, puesto que este no presenta una complejidad suficientemente elevada como para necesitar, por ejemplo, el momentum para la aceleración de la optimización.

Para lo parámetros si consideraros, se ha creado un grid con los siguientes elementos:

- $hidden\_layer\_sizes \in [(20,), (40,), (50,), (70,), (100,), (20, 20, 20), (50, 50, 50), (20, 50, 200), (50, 100, 50)]$
- $activation \in [logistic, tanh, relu]$
- $\alpha \in [1.e-01, 1.e-02, 1.e-03, 1.e-04, 1.e-05, 1.e-06]$

Realizando esta seleccion de parámetros se ha intentado cubrir una amplia gama de posibilidades, especialmente con $hidden\_layer\_sizes$, de acuerdo a la complejidad del problema (8 *features*).

Realizando CV sobre el set con *featrues* correlacionados y sin, se ha encontrado que las mejores redes posibles son las siguientes:

	Capas Ocultas	Función de Activación	α
Corr	(50,100,50)	tanh	0.1
No corr	(20,20,20)	logistic	0.01

Tabla 5: Mejores parametros para MLPClasssifier (encontrados con CV)

Se puede apreciar en la Tabla 5 como la eliminación de los *features* correlacionados lleva a la obtención de un modelo más simplificado, reduciendose el número de neuronas por capa. En cuanto a la función de activación es remarcable como para ambos conjuntos de datos funciona mejor una función de tipo curva, que no una función de tipo recta como ReLU.

El modelo entrenado con el set de datos normal estandarizado ha presentado un error de entrenamiento medio de 1.89 % y un error de test superior, de 2.32 %. Mientras que para el modelo entrenado con datos sin *features* correlacionados el error medio de entrenamiento ha sido de 1.89 % y el de test de 2.23 %.

Aunque el error de test en ambos modelos se encuentra en la linea de los modelos lineales, MLP permite conseguir una mayor precisión, y se adapta mejor a la falta de ejemplos de púlsares como se puede observar en la Tabla 6, aunque empeora en la detección de no púlsares.

Predicción Real	**No púlsar**	**Púlsar**
No púlsar	3218	31
Púlsar	52	279

Tabla 6: Matriz de confusión del mejor modelo MLP (Corr) en test

5.2.2. Random Forest

Se ha decidido emplear este modelo ya que en Random Forests no es necesario estandarizar los datos, y además funciona bien en problemas de clasificación cuyos datos de diferentes clases estan desequilibrados, que es lo que ocurre en el problema que nos ocupa.

Los métodos de scikit-learn utilizados son dos. El método para el propio modelo, *RandomForestClassifier* y *compute_class_weight*, que permite calcular el mejor peso para las diferentes clases del problema. Estos pesos sirven para compensar el desequilibrio de datos que pueda haber entre diferentes clases.

El principal parámetro que admite *RandomForestClassifier* es *n_estimators*, que se corresponde con el número de *decision trees* que debe generar el Random Forest. Para este parámetro se ha decidido probado valores que van entre 10 y 1000, un rango considerablemente amplio. Posteriormente se ha probado entre 100 y 250, al observar primeramente que es entre esos valores donde la variabilidad del resultado cobraba importancia.

Como ha sido explicado en la Sección 4, para este modelo, utilizaremos la pun-

tuación del propio modelo, conocida como OOB. Ésta permite hypertunear el parámetro de número de árboles y así saber cuál es el valor de este parámetro que da un mejor resultado.

Para el caso de los datos normales (sin estandarizar y con todos los *features*), el número de árboles que mejor puntuación ha obtenido es 120, con una puntuación OOB de 0.019 y un error de training casi ínfimo, de 0.007. Al predecir los correspondientes datos de test con este modelo, el error de test resulta ser 2.37 %.

Predicción Real	**No púlsar**	**Pulsar**
No púlsar	3210	29
Pulsar	60	271

Tabla 7: Matriz de confusión del mejor modelo de RF en test

En cambio, para el caso de los datos sin *features* correlacionados, el número óptimo de arboles de decisión ha sido 100. La puntuación OOB en este caso ha sido 0.019 y el error de training ha sido 0.007, otra vez. En este caso, no obstante, el error de test ha sido de 2.56 %, un poco peor que el resultado anterior.

De la misma forma que ocurría con el modelo MLP, Random Forest se mantiene en la línea de los modelos lineales en cuanto a los errores de test, no obstante también empeora en la identificación de pulsares.

6. Modelo final

Método	Accuracy	Recall	Precision	F-Score
k-NN (Corr)	**0.98**	0.91	**0.96**	**0.93**
k-NN (No corr)	**0.98**	0.90	**0.96**	**0.93**
Logistic Regression (Corr)	**0.98**	0.91	**0.96**	**0.93**
Logistic Regression (No corr)	0.97	0.88	**0.96**	0.92
Linear SVM (Corr)	**0.98**	0.90	**0.96**	0.92
Linear SVM (No corr)	**0.98**	0.90	0.95	**0.93**
MLP (Corr)	**0.98**	**0.92**	0.94	**0.93**
MLP (No corr)	**0.98**	0.91	0.95	**0.93**
Random Forest (Corr)	**0.98**	0.91	0.95	**0.93**
Random Forest (No corr)	**0.98**	0.91	0.94	**0.93**

Tabla 8: Accuracy, Recall, Precision y F-Score obtenidos con cada conjunto de datos y método

Con tal de escoger un método final, se ha confeccionado la Tabla 8 para así poder comparar los diferentes resultados de Accuracy, Recall, Precision y F-Score para los distintos métodos y conjuntos de datos utilizados.

Al comparar estos resultados, se observa que todos los métodos obtienen resultados muy parecidos, lo que dificulta la elección de un método en concreto. Es por ello que también es necesario estudiar en profundidad los resultados para ambas clases de forma más concreta. Con este motivo se ha generado la Tabla 9.

Como se ha comentado en secciones anteriores, se observa una clara diferencia entre los modelos lineales y no lineales en cuanto al resultado que los segundos dan sobre la predicción de púlsares, la cual es claramente peor en términos de precisión. Sin embargo, para la predicción de no púlsares, los resultados son prácticamente iguales en todos los modelos.

También se observa que, en los casos en los que se utiliza el conjunto de datos sin *features* correlacionados, para la detección de púlsares, la precisión sube a costa del recall. Es por esto que es preciso rechazar los métodos en los que se utilizan datos sin *features* correlacionados. Siendo de especial interes también la reducción en cuanto a complejidad que aportan.

Finalmente, es intuible la selección de un método lineal. Por una parte, como se ha comentado, estos dan un mejor resultado para la predicción de púlsares y por otra, claramente su sencilleza ante los métodos no lineales hace que estos tengan un coste computacional mucho menor, interesante para aplicaciones en tiempo real, por ejemplo. Entre estos, se observa que k-NN y Logistic Regression tienen un comportamiento casi idéntico. Debido a la simplicidad algorítmica del primero y a la facilidad para entender lo que realmente está ocurriendo al ejecutarse, hemos decidido quedarnos con k-NN como mejor modelo, aunque hacemos incisión en el hecho de que Logistic Regression también es un método a tener en cuenta.

Método	Recall		Precision	
	No púlsar	Púlsar	No púlsar	Púlsar
k-NN (Corr)	0.99	0.82	0.98	0.93
k-NN (No corr)	0.99	0.81	0.98	0.94
Logistic Regression (Corr)	0.99	0.82	0.98	0.93
Logistic Regression (No corr)	**1.00**	0.77	0.98	**0.95**
Linear SVM (Corr)	0.99	0.80	0.98	0.93
Linear SVM (No corr)	0.99	0.82	0.98	0.93
MLP (Corr)	0.99	**0.84**	0.98	0.90
MLP (No corr)	0.99	**0.84**	0.98	0.91
Random Forest (Corr)	0.99	**0.84**	0.98	0.91
Random Forest (No corr)	0.99	**0.84**	0.98	0.90

Tabla 9: Recall y Precision para cada clase obtenidos con cada conjunto de datos y método

7. Conclusiones

7.1. Autoevaluación

El desarrollo de todo el conjunto del trabajo se ha realizado de forma pautada y premeditada, hecho que ha permitido un correcto desarrollo de este, pudiendose cumplir todos los objetivos prefijados al inicio.

Hemos podido analizar de forma adecuada los datos, pudiendo llegar a asegurar que existe una clara relación entre las variables de entrada dadas por el problema y la clasificación de estos datos en Púlsares o no Púlsares. Esta afirmación está justificada debido a que en todos los métodos de aprendizaje automatico que se han utilizado la predicción que se ha dado de los datos ha sido suficientemente fiable.

También hemos podido probar una gama amplia de posibilidades por cada uno de los modelos estudiados gracias a la librería scikit-learn, pudiendo así tratar de conseguir mejores modelos.

7.2. Conclusiones personales y científicas

Desde un punto de vista personal, consideramos que tener la posibilidad de estudiar un conjunto de datos como el estudiado nos ha permitido entender de una forma más profunda como funcionan los diferentes modelos de aprendizaje supervisado, como tratar con grandes cantidades de datos y como interpretar los diferentes resultados. Además, aplicando las tecnicas empleadas a un conjunto de datos real, se ha dado un sentido y aplicación real a dichos métodos, pudiendose profundizar más en ellos.

En el plano cientifico, queremos destacar la viabilidad de emplear los modelos desarrollados para este problema. Dada la precisión que estos presentan, no sería descabellado emplearlos en nuevos datos obtenidos sobre otras estrellas diferentes. Así, aquellos físicos que esten estudiando y recopilando datos sobre estrellas puedan utilizar estos modelos de predicción para enfocar y realizar sus estudios de forma más eficiente.

7.3. Posibles extensiones y limitaciones

La principal limitación del problema ha sido el desequilibrio que existe entre las clases dadas, ya que el número de ejemplares no-púlsares es mucho mayor que el de púlsares. Esto ha hecho que en los métodos utilizados la precisión de predicción de púlsares fuera mucho más limitada que la de no-púlsares. Aun asi se ha decidido no emplear tecnicas para mejorar el desequilibrio, puesto que los resultados obtenidos son de especial relevancia, aunque sería interesante contemplar la posibilidad de intentar mejorar el balanceo de clases para tratar de encontrar una mejora en la predicción.

Otra posible extensión de este problema sería probar otros modelos de aprendizaje supervisado sobre los datos y compararlos con los ya estudiados para observar posibles diferencias y/o mejoras. Algunos de estos modelos podría ser

versiones mejoradas o más rapidas de los aqui estudiados, para poderlos emplear en experimentos en tiempo real.

Referencias

[Scikit:Clustering] Varios Autores. *Clustering*. Accedido 28/12/2019. URL: https://scikit-learn.org/stable/modules/clustering.html.

[Scikit:LogisticRegression] Varios Autores. *LogisticRegression*. Accedido 5/01/2020. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[Scikit:MLPClassifier] Varios Autores. *MLPClassifier*. Accedido 11/01/2020. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[Scikit:NearestNeighbors] Varios Autores. *Nearest Neighbors*. Accedido 28/12/2019. URL: https://scikit-learn.org/stable/modules/neighbors.html.

[Scikit:RandomForestClassifier] Varios Autores. *RandomForestClassifier*. Accedido 11/01/2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[Scikit:SupportVectorMachines] Varios Autores. *Support Vector Machines*. Accedido 12/01/2020. URL: https://scikit-learn.org/stable/modules/svm.html.

[Grus] J. Grus. *Data Science from Scratch*. 1.ª ed. O'Reilly Media, 2015, págs. 132-133. ISBN: 978-1491901427.

[Kotsiantis14] S. B. Kotsiantis. "Feature selection for machine learning classification problems: a recent overview". En: *Artificial Intelligence Review* 42.1 (jun. de 2014), págs. 157-157. DOI: 10.1007/s10462-011-9230-1.

[HTRU2] R. J. Lyon. *HTRU2*. DOI: 10.6084/m9.figshare.3080389.v1.

[Lyon] R. J. Lyon y col. "Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach". En: *Monthly Notices of the Royal Astronomical Society* 459.1 (abr. de 2016), págs. 1104-1123. ISSN: 0035-8711. DOI: 10.1093/mnras/stw656.