Predicción de actividad inhibitoria de moléculas pequeñas sobre el receptor serotoninérgico 5-HT2$_A$ mediante modelos de machine learning.

Aramis Adriana Rojas Mena
arrojas@uoc.edu
Máster Universitario en Bioinformática y Bioestadística

Nombre Consultor/a**: Melchor Sánchez Martinez**

Barcelona, junio del 2020.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Predicción de actividad inhibitoria de moléculas pequeñas sobre el receptor serotoninérgico 5-HT2A mediante modelos de machine learning.*
Nombre del autor:	*Aramis Adriana Rojas Mena*
Nombre del consultor/a:	*Melchor Sánchez Martinez*
Nombre del PRA:	*Marc Maceira Duch* *Javier Luis Cánovas Izquierdo*
Fecha de entrega:	25/06/2020
Titulación:	*Máster Universitario en Bioinformática y Bioestadística.*
Área del Trabajo Final:	*Bioinformática y Bioestadística*
Idioma del trabajo:	*Español.*
Palabras clave / Keywords	*Predicción, Ki, receptores serotoninérgicos.* *Prediction, Ki, serotoninergic receptors.*

Resumen (Spanish):

El presente trabajo se engloba dentro del campo de las ciencias de la computación y su aplicación en las ciencias de la salud, específicamente en el desarrollo de modelos de aprendizaje automático (machine learning) que permitan predecir la actividad de moléculas pequeñas sobre el receptor serotoninérgico 5-HT2$_A$. Este receptor se asocia a patologías psiquiátricas ya conocidas, para las cuales existe tratamiento con moléculas pequeñas que lo inhiben, pero con frecuentes efectos adversos. Se prueban algoritmos tanto de clasificación como de regresión, para comprobar cuáles podrían tener una capacidad de predicción mejor sobre la constante de inhibición, Ki, y poder realizar un cribado computacional sobre grupos de moléculas pequeñas que pudieran suponer una alternativa terapéutica.

Los hallazgos sugieren que, a partir de un conjunto de datos cribado y balanceado, los algoritmos de clasificación tienen, en general, muy buena capacidad de predicción de la actividad (activa o inactiva) de las moléculas pequeñas sobre este receptor 5-HT2$_A$. El mejor algoritmo es el SVM, con una exactitud y precisión por encima del 93%. Los algoritmos de tipo regresión no resultan útiles para predecir la actividad. Para ambos casos, será necesario reproducir estudios similares sobre este receptor, con diferente procedencia de

los datos y otros algoritmos o diferente configuración de sus hiperparámetros, para poder inferir un conocimiento más robusto.

Abstract (English):

The present work is included within the field of computer science and its application in the health sciences, specifically the development of machine learning models that allow predicting the activity of small molecules on the serotonergic receptor 5- $HT2_A$. This receptor is associated with already known psychiatric pathologies, for which there is treatment with small molecules that inhibit it, but with frequent adverse effects. Both classification and regression algorithms are tested to see which ones could have a better predictive capacity on the inhibition constant, Ki, and be able to perform computational screening on groups of small molecules that could be a therapeutic alternative.

The findings suggest that, based on a balanced and screened data set, the classification algorithms generally have a very good ability to predict the activity (active or inactive) of small molecules on this 5-$HT2_A$ receptor. The best algorithm is SVM, with an accuracy and precision of over 93%. Regression algorithms are not helpful in predicting activity. For both cases, it will be necessary to reproduce similar studies on this receptor, with different sources of data and other algorithms or different configuration of its hyperparameters, in order to infer more robust knowledge.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE GRÁFICOS DE RESULTADOS

1. INTRODUCCIÓN

1.1 Contexto y justificación del trabajo

El proyecto se basa en el uso de los algoritmos de *machine learning* (aprendizaje automático) mediante técnicas de aprendizaje supervisado, tanto de clasificación como de regresión para la predicción de una respuesta, dentro del campo de las ciencias farmacéuticas y químicas, en tanto que dicha respuesta se trata de una constante de inhibición de una diana terapéutica de interés médico.

El campo de la quimioinformática se viene desarrollando desde los últimos 15-20 años, siendo los últimos en los que ha emergido de forma notoria, debido a las mejoras físicas de las máquinas computadoras para procesar grandes volúmenes de datos, y el perfeccionamiento de las técnicas estadísticas, con métodos de trabajo más precisos y robustos, que permiten dar una mayor fiabilidad respecto de los resultados obtenidos.

Dicha quimioinformática se centra especialmente en el campo del *machine learning*, como herramienta para acelerar los procesos de innovación y desarrollo de fármacos nuevos. Permite realizar un cribado de cuáles podrían ser potenciales moléculas activas de un conjunto amplio y diverso respecto de sus propiedades físicas y químicas. De forma adicional, este campo cada vez tiene un mayor reconocimiento por su potencial utilidad por parte de las agencias regulatorias del medicamento de cada región

A partir de las características fisicoquímicas de un conjunto grande de moléculas de síntesis química, que son tratadas como las variables de entrada de información en los modelos que se desarrollan y analizan el comportamiento de diversos algoritmos. La finalidad es estudiar cuáles son los que tienen una mejor capacidad predictiva de la variable respuesta para el conjunto de datos con el que se trabaja.

1.2 Objetivos del trabajo

El objetivo de este trabajo es saber si con los datos disponibles en una base de datos en abierto relativos a la estructura química de moléculas de síntesis con potencial interés biomédico se pueden desarrollar y mejorar modelos de *machine learning* para predecir la actividad inhibitoria de un determinado subconjunto de dichas moléculas, para una diana terapéutica específica.

Los objetivos de este trabajo son los siguientes:

- Manejar la base de datos ChEMBL en entorno UNIX/Linux de forma solvente para hacer el volcado, consultas en las tablas y exportación de los datos a otras plataformas o aplicaciones.
- Realizar operaciones de limpieza del conjunto de datos óptima para que estos pueden ser usados en los modelos predictivos respecto a una diana

terapéutica. Escoger la variable respuesta de interés, K_i (constante de inhibición) o IC_{50} (concentración inhibitoria media) bajo el criterio imparcial del mayor número de datos disponibles y para los que se ofrezca una mayor fiabilidad.

- Desarrollar y validar varios modelos predictivos de aprendizaje supervisado, a partir de los datos relativos a la estructura química de las moléculas, para una respuesta de naturaleza química y diana terapéutica, escogida de forma arbitraria.

1.3 Enfoque y método seguido

El enfoque se establece en dos bloques principales:

La primera, la interrogación de una base de datos relacional de libre acceso, aquí ChEMBL[1] (que contiene la información respecto de las moléculas de síntesis química. Se hace a través de la construcción de dicha BBDD en el gestor PostgreSQL en el servidor local, en plataforma UNIX/Linux, con el archivo contenedor de toda la información (formato .dmp), junto con una aplicación web cliente que se conecta al mismo servidor local (pgAdmin4), como una interfaz gráfica que facilita la ejecución de dichas preguntas en el lenguaje SQL (*Structured Query Language*) y la posterior exportación del subconjunto de datos de interés. Esto permite amoldar las preguntas acordes a la necesidad propia, pudiendo modificar el orden de presentación de las columnas en las tablas, tipo de registros, criterios de cerca, ...etc.

La segunda, una vez se tienen los datos que interesa, con respecto al volumen y calidad suficientes, para la diana terapéutica que se esté estudiando, y en un formato portable a plataforma Windows, se desarrollan los pertinentes modelos predictivos en lenguaje Python, en un entorno de desarrollo interactivo multiplataforma de código abierto, como lo es Spyder. La razón de elección de este lenguaje de programación se asienta en su uso creciente en el entorno tanto académico como profesional, lo que puede resultar muy interesante para su pronta aplicación por empresas del ámbito del desarrollo de fármacos.

1.4 Planificación del trabajo

Primer bloque:

- Instalar en la máquina física (Windows 10) la aplicación de virtualización de máquinas VMWare Workstation 16.
- Descargar la imagen ISO de instalación de la distribución de UNIX/Linux escogida (distribución Ubuntu Mate 18.04 LTS) e instalarla a través de VMWare.
- Actualizar la máquina virtual con las últimas versiones, así como instalar el gestor de bases de datos PostgreSQL. La instalación se realiza a través de la consola de comandos de Ubuntu Mate (shell) con permisos de

[1] https://www.ebi.ac.uk/chembl/

superusuario.

- Realizar el volcado de los datos comprimidos a una base de datos creada en dicha máquina virtual, a través del gestor de BBDD (PostgreSQL).
- Proceder al cuestionario de esta BBDD a través de pgAdmin4 para obtener la información necesaria acerca de las variables descriptoras y la variable respuesta escogida para los posteriores modelos predictivos. Imponer condiciones para eliminar determinados tipos de registros.
- Realizar la exportación de los resultados en un formato de tabular de texto plano, como ".csv" (*comma separated values*).

Segundo bloque:

- Instalación del entorno de desarrollo interactivo Spyder, óptimo para el lenguaje de programación Python. Instalación de librerías propias de *machine learning*, como RDKit, que permitirá transformar la información relativa a la estructura tridimensional de las moléculas (en formato SMILES) en los denominados *fingerprints* (cadenas de números, generalmente 0 y 1, que recogen dicha información tridimensional)
- Curado/filtrado de los datos de los resultados. Inclusión del término InChiKey, transformado con el software Open Babel GUI desde el término SMILES, para eliminar registros duplicados.
- Desarrollo de modelos predictivos variables predictoras – variable respuesta, en particular de aprendizaje supervisado, tanto de clasificación como de regresión.
- Elección de los mejores modelos predictivos, en cuanto a criterios técnicos/estadísticos Uso de métricas apropiadas para cada tipo de algoritmo. Aplicación de técnicas de validación de los modelos para garantizar su robustez.

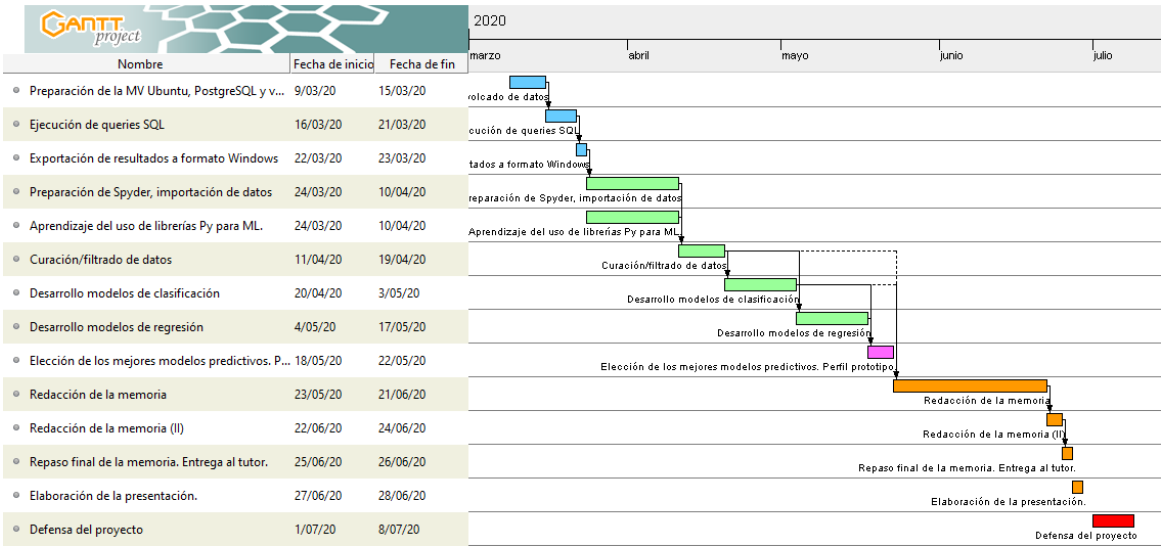

Ilustración 1. Diagrama de Gantt con los hitos del trabajo.

2. ESTADO DEL ARTE[2] [3] [4] [5]

En las últimas décadas, el descubrimiento de nuevos fármacos ha experimentado un cambio de paradigma respecto de cómo se había realizado desde su auge en el siglo XIX, mediante el método de ensayo y error, con un grupo reducido de moléculas, en un laboratorio de una forma totalmente empírica.

Esto se ha debido a la eclosión de una nueva disciplina de investigación y desarrollo de nuevos fármacos, la quimioinformática, que une dos campos de estudio, las ciencias de la computación y las ciencias farmacéuticas.

Una de sus aplicaciones más desarrolladas hasta el día de hoy es el denominado *screening* molecular, que mediante unos protocolos y procedimientos definidos, permite construir modelos predictivos para el descubrimiento de nuevas moléculas con potencial actividad farmacológica, con el máximo rigor estadístico.

Esto es debido al auge y perfeccionamiento de las bases de datos de índole química, que contienen información tanto de moléculas probadas experimentalmente, y de las que se conoce su actividad farmacológica o propiedades fisicoquímicas, como de otras moléculas de las cuáles solo se tiene catalogada su estructura.

Las relaciones cuantitativas estructura – actividad, o más bien conocidas como técnicas QSAR (por las siglas en inglés de *Quantitative Structure-Activity Relationships*) son las que engloban un conjunto de operaciones probadamente eficaz para explorar y explotar la relación entre la estructura química y su actividad farmacológica, apoyadas en las matemáticas y en la estadística.

El objetivo de cualquier modelado QSAR es el de establecer un algoritmo cualquiera que se dé en los valores de la/s variable/s que actúen como descriptoras en el modelo para otra variable respuesta (como clases o numérica). Para todos estos modelos, se sigue la suposición del principio de similaridad, por el cual, compuestos con estructuras similares han de tener actividades farmacológicas similares.

Los aspectos más críticos de este tipo de desarrollo computacional de fármacos son la selección y desarrollo del modelo matemático que enlace las variables descriptoras con la respuesta, así como las técnicas de validación de estos.

Es importante considerar que, aunque el esfuerzo se concentre en conseguir modelos con el máximo poder predictivo y rigor estadístico, todos los potenciales descubrimientos que se pudieran hallar con esta metodología habrán de ser contrastados mediante la práctica empírica en laboratorio.

---

[2] (Tropsha, 2010)
[3] (Eriksson, y otros, 2003)
[4] (Gortari, García-Jacas, Martinez-Mayorga, & Medina-Franco, 2017)
[5] (Administration, 2020)

2.1 Flujo de operaciones habitual previo al *screening* computacional en desarrollo de nuevos fármacos[6]

Los modelos predictivos con datos relativos a moléculas tienden a ser tan buenos como lo son los datos de partida con los que estos se construyen. Cualquier error en la transcripción de las estructuras de las moléculas puede suponer errores, a su vez, en las fracciones de estas que pueden servir como variables descriptoras de los modelos, los descriptores químicos, y, por tanto, incidir en la calidad de los modelos predictivos.

Estos errores, que pueden parecen pequeños en número o magnitud, suponen pérdidas de calidad en las predicciones de cualquier modelo QSAR. Es por eso por lo que resulta muy importante el proceso completo de preparación de los datos previo a la ejecución de las técnicas QSAR.

Hoy en día existen pocos procesos establecidos con claridad relativos a asegurar la calidad mínima para un set de datos que se use para esto, pero el mínimo si parece poderse dividir en varias etapas:

2.1.1 Curado de datos

Se han de eliminar aquella fracción de todo el conjunto de datos que no pueda ser procesada mediante técnicas quimioinformáticas. Aquí se incluyen compuestos químicos como los inorgánicos, compuestos organometálicos, sales y mezclas de sales, compuestos aromáticos, formas tautoméricas y eliminación de duplicados.

2.1.2 Tamaño muestral y balanceado

El tamaño y proporción óptimo de las diferentes clases que pueda haber en un set de datos dependen de varios factores. El límite superior suele estar determinado, por norma general, por la capacidad computacional del equipo con el que se realicen los cálculos. El límite inferior, por cambio, se rige considerando el número de particiones que se hagan del set de datos (en fracciones de entrenamiento, test y validación), considerando que un tamaño muy pequeño rinde problemas de sobreajuste del modelo a los datos de entrenamiento, así como correlación casual.

Estos son los mínimos que deben de cumplir los conjuntos de datos balanceados:

- Si se trata de modelos para predecir una variable numérica continua, la proporción (en tanto por ciento) debe ser 50:25:25, con un mínimo total a partir de 40 compuestos para todo el set de datos.

[6] (Tropsha, 2010)

- Si se trata de modelos para predecir una variable categórica, debe de haber 10 compuestos de cada clase en la partición de entrenamiento, 5 en la partición de test y otros 5 en la de validación. Para estos, en cualquier caso, lo mejor es que el número de compuestos de cada categoría sea aproximadamente igual en cada una de ellas.

Muchos de los sets de datos que se usan en el campo de la quimioinformática suelen estar desbalanceados (el tamaño de las diferentes clases existentes es diferente); de ahí que sea importante la necesidad de realizar los modelos predictivos tanto en el set de datos desbalanceado como en el balanceado.

Si no se hace, se estaría pasando por alto el hecho que, en un set desbalanceado, tanto el modelado como la validación estarán siempre sesgadas hacia una predicción correcta de la clase mayoritaria. Las técnicas de corrección del desbalanceo de datos suelen ser el submuestreo de la clase mayoritaria o el sobremuestreo de la clase minoritaria.

2.1.3 Eliminación de valores atípicos (*outliers*)[7]

También resulta importante de cara a mejorar la calidad de los modelos predictivos la eliminación de aquellas moléculas que, en el espacio descriptor, muestren unas propiedades diferentes a las del resto de grupo/s existente/s en el set de datos. Suelen tratarse de grupos de moléculas donde la suposición básica de las técnicas QSAR, por la que compuestos similares tienen actividades similares, no se sostiene.

Estos valores atípicos, relativos a la actividad biológica a predecir, se podrían estudiar y eliminar aplicando técnicas de clustering clásicas sobre un set de datos. El proceso seguido para este proyecto se especifica en *3.3 Obtención de los fingerprints y estructura de los datos para ser usados en los modelos.*

2.1.4 Validación del modelo

La capacidad de predicción de un modelo QSAR puede ser establecida por la validación del modelo con una fracción del set de datos que no hayan sido utilizados en la construcción de los diferentes modelos, generalmente.

Esto tiene su contrapartida, y es que no garantiza que los modelos seleccionados con las métricas de R^2 de estos vayan a realizar predicciones exactas para compuestos totalmente fuera del conjunto de datos usados. Además, diversos modelos de machine learning necesitan ajustar sus hiperparámetros. Si los valores de los hiperparámetros se ajustan demasiado a un conjunto de test, también es posible obtener resultados con diferentes métricas.

Por esta razón, es recomendable realizar un proceso de validación; en lugar de la frecuente división del conjunto de datos en entrenamiento y test, hacerlo:

- En 3 partes, como entrenamiento + test + validación externa.

[7] (Wallach & Heifets, 2018)

- O bien, en los métodos de división del set de datos en varios subconjuntos de entrenamiento y test.

Si se escoge el primer método, la fracción de validación externa ha de ser un subconjunto de datos que no hayan sido utilizados ni en el entrenamiento ni en el test. Puede ser seleccionado de forma aleatoria del conjunto inicial de datos, y un valor óptimo puede estar en torno al 15-20%.

Si se escoge el segundo método, los dos métodos más probados y de mayor capacidad predictiva han demostrado ser el *5-fold-cross-validation* y *el 10-fold-cross-validation*. Se explica con mayor detenimiento en *3.6 Aplicación de método de validación sobre los modelos.* cuáles son los más usados, las razones técnicas detrás de estos, y el aplicado en este trabajo.

2.2 Necesidad del uso de fingerprints[8] [9]

Cualquiera de las moléculas sintéticas existentes se trata de un objeto tridimensional, con abundantes detalles en cuanto a posesión de un tipo de átomos pesados u otros, que pueden ser especialmente relevantes en cuanto a determinar las propiedades físicas y químicas de la molécula sobre sus receptores o su actividad, así como existencia de diferentes tipos de enlaces que unen los diferentes átomos (sencillos, dobles, triples), formación de estructuras cerradas como anillos...etc.

Esto convierte a las moléculas en objetos físicos muy complejos, de los cuales es necesario recoger toda esta información en una estructura numérica codificada que permita realizar los modelados matemáticos que relacionen la estructura química con propiedades fisicoquímicas, actividad farmacológica u otras propiedades experimentales.

Se conocen como descriptores moleculares, y desde el inicio de su estudio y desarrollo, en la década del 2000, se han reportado cientos de tipos diferentes de ellos. Se pueden clasifican acorde a la dimensionalidad de la molécula que van a representar o por su naturaleza:

- **Dimensionalidad:** pueden ser descriptores 1D (una dimensión) si recogen características generales, como el peso molecular, la refractividad molar, el logP (logaritmo del coeficiente de partición octanol/agua)...etc.; también pueden ser descriptores 2D (dos dimensiones) si recogen información relativa a número de átomos, número de enlaces, conexiones entre átomos…etc..; o bien ser descriptores 3D, si la información recoge datos de la conformación de las moléculas, momentos de inercia, volumen de Van der Waals...etc.

- **Naturaleza:** pueden ser constitucionales, si recogen características generales de la molécula, o bien topológicos, cuando usan la teoría matemática de grafos aplicada al esquema de conexiones de átomos de la molécula. Las demás subcategorías existentes (geométricos,

[8] (Bajorath)
[9] (Fara & Oprea, s.f.)

electrónicos o cuánticos) son más laxas, en cuanto derivan de cálculos empíricos o de orbitales moleculares, y codifican la capacidad de la molécula de participar en interacciones polares o de enlaces de hidrógeno (como donadores o aceptores de estos).

En cualquier caso, estos descriptores pueden ser codificados desde un valor numérico, discreto o continuo, con un rango de interés predefinido; o bien, desde la presencia o ausencia de un elemento estructural (normalmente en codificación binaria de 0 y 1). El fundamento básico es que, la presencia o ausencia de los posibles valores de dicha variable se representen en una cadena de bits de longitud fija.

El factor más determinante para la construcción de estos descriptores moleculares (o fingerprint, como serán referidos más adelante), es que pueden ser asignados de 3 formas diferentes:

- **Directa:** se asignan los diferentes grupos o descriptores a áreas separadas de la cadena de bits (fingerprint).

- **Diccionario**: la asignación se hace mediante un diccionario que especifica la correspondencia entre una estructura y una posición determinadas en la cadena de bits.

- **Por partición:** una función de partición (*hash function*) de la cadena de bits crea fragmentos como para que quepan estos en la longitud predeterminada de dicha cadena. Cuantos más fragmentos genere la función de partición, mayor es la posibilidad de generar un patrón único en la cadena de bits.

De cualquiera de las formas que se obtengan estos descriptores moleculares, y la longitud de la cadena de caracteres que tuvieran, lo común a todos ellos será la división de los elementos de la cadena en cada uno de los bits que la conforman, de forma que puedan utilizarse como datos de entrada para la construcción de modelos de *machine learning* posteriores.

2.3 Elección de la diana terapéutica 5-HT2A y la constante de inhibición Ki[10] [11] [12] [13] [14]

La diana terapéutica 5-HT2$_A$ se trata de un receptor celular a nivel de membrana, que generalmente se localiza en tejidos neurológicos, tanto en el axón como las dendritas de las neuronas, así como vesículas citoplasmáticas y presinápticas.

[10] (Wacker D., 2017)
[11] (UniProtKB - P28223 - 5-hydroxytryptamine receptor 2A, s.f.)
[12] (5-hydroxytryptamine receptor 2A, s.f.)
[13] (Xu, y otros, 2018)
[14] (Enzyme inhibitors, s.f.)

Sus funciones son como receptor de varios fármacos, así como receptor endógeno del neurotransmisor serotonina, relacionado con el estado anímico y el comportamiento en los seres humanos y otras especies animales.

Otras funciones alternativas en las que está involucrado son: la detección del estímulo de la temperatura en la percepción sensorial del dolor, regulación positiva de la vasoconstricción, regulación negativa de la transmisión sináptica glutamatérgica o la constricción del músculo liso de la vejiga urinaria.

Dado su papel fundamental en el sistema nervioso, esto hace que este receptor esté presente en el desarrollo de las principales enfermedades mentales en la sociedad actual, tales como la depresión mayor, esquizofrenia, trastorno bipolar, trastorno obsesivo-compulsivo,etc.

Para este tipo de enfermedades se han venido utilizando diversos grupos de fármacos, que actúan tanto en el citado receptor, como en el D2 dopaminérgico. Los resultados de la práctica clínica, en cambio, han arrojado la existencia de efectos adversos en forma de síntomas extrapiramidales (parkinsonismo inducido por fármacos, discinesia tardía), que se han relacionado con dicho receptor D2 dopaminérgico.

También, otros fármacos como los antidepresivos tricíclicos (TCAs), inhibidores de la monoaminoxidasa (MAOIs), inhibidores de la recaptación de la serotonina (SSRIs) o los inhibidores de la recaptación de serotonina-norepinefrina (SNRIs), debido a sus elevadas afinidades a receptores muscarínicos, adrenérgicos e histaminérgicos se han relacionado con efectos adversos adicionales.

Es por esta razón que se sostiene la hipótesis por la cual, fármacos que solo actuasen en el receptor 5-HT2_A, evitarían en buena parte estos efectos adversos, dándose así un vacío terapéutico con respecto a aquellos pacientes que sufren con mayor intensidad y frecuencia los efectos adversos de los fármacos clásicos.[15] [16]

Para realizar dicha actividad de innovación y desarrollo, es aquí donde entran las técnicas de cribado computacional de moléculas, que permiten acelerar este proceso, con datos ya recogidos previamente en la práctica experimental relativos a la actividad biológica que desarrollan estos fármacos, para definir modelos de *machine learning*, de clasificación o regresión.

Estos podrán predecir si se producirá tal actividad biológica esperada (clasificación) o con qué magnitud lo hará (regresión) a partir de un set de moléculas nuevas, no usadas previamente para entrenar los modelos. Ej.: en este proyecto, la unión al receptor para desencadenar una actividad inhibitoria.

La variable de estudio de interés será el “pchembl value”, que se trata de una normalización logarítmica de la variable “Ki” (constante de inhibición), indicador de la potencia inhibitoria de una molécula sobre una diana terapéutica determinada. Como se verá en el apartado *3.1 Obtención de datos para el*

[15] (Mishra, 2019)
[16] (Xu, y otros, 2018)

trabajo. Planteamiento para su explotación, esta constante, para esta diana terapéutica, es de la que se extrae un mayor número de datos.

La normalización logarítmica permite hacer una interpretación más sencilla de dicha constante relativa a la concentración, pues esta abarca un amplio rango de valores, desde unidades hasta las centenas (en nm, nanomolar), así como fomentar una mayor precisión en los modelos de *machine learning* que se vayan a desarrollar.

3. DESARROLLO

3.1 Obtención de datos para el trabajo. Planteamiento para su explotación

Los datos que se usan en este trabajo proceden de la base de datos ChEMBL, mantenido por el Instituto Europeo de Bioinformática (EBI, por sus siglas en inglés), que aloja más de 1.9 millones de compuestos bioactivos, como pequeñas moléculas, con potencial acción farmacológica.

Existen dos vías esenciales de obtención de los datos: 1) con el formato preestablecido que se puede obtener a través del servidor web, o 2) mediante un volcado en un gestor de bases de datos relacionales. Por conveniencia se exponen ambos, aunque el ejecutado para este proyecto ha sido el segundo.

3.1.1 Datos a partir del servidor web de ChEMBL

Habría que introducir la proteína/diana terapéutica deseada en el buscador y seleccionar aquella que marca "*single protein*".

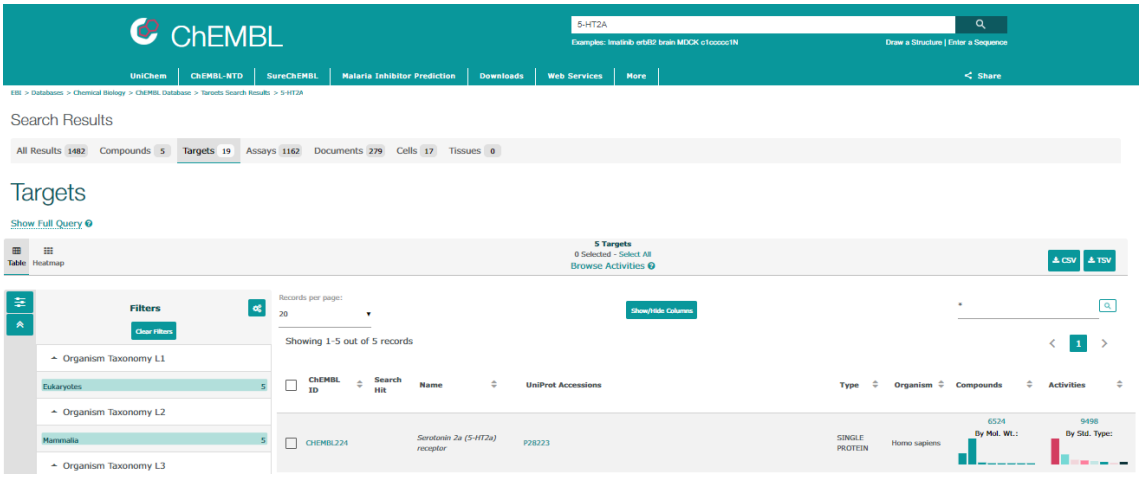

Ilustración 2. Página inicial de búsqueda en ChEMBL del receptor de interés.

En la siguiente página, bajar el explorador hasta “*associated bioactivities*”, y seleccionar aquellas respecto al Ki;

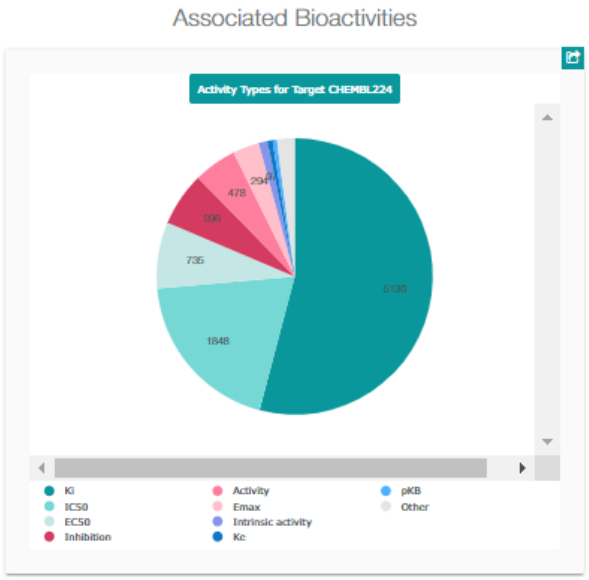

Ilustración 3. Propiedades del receptor escogido en ChEMBL para las cuales hay datos y su proporción gráfica.

En la última ventana, filtrar los resultados por las propiedades deseadas, y entonces, pinchar en el botón del lado superior derecho donde marca “CSV”, para descargar el archivo en formato “.csv”.

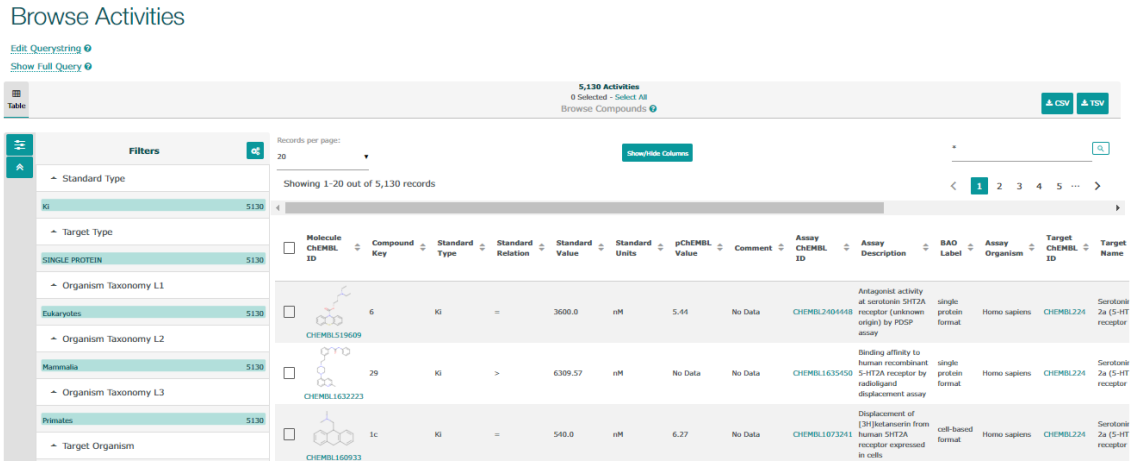

Ilustración 4. Moléculas disponibles y ligadas al receptor que se haya buscado.

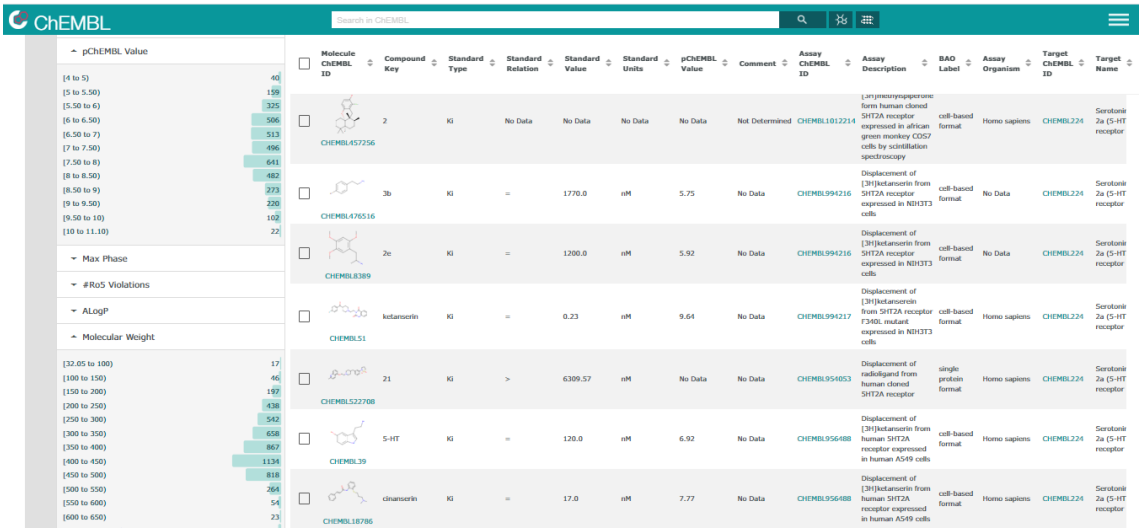

Ilustración 5. Página con las pestañas laterales abiertas para filtrar el listado de moléculas disponibles y ligadas al receptor buscado.

3.1.2 Datos a partir de un archivo formato ".dmp" y un gestor de bases de datos

Para obtener los datos de esta forma, es necesario obtener el correspondiente archivo ".dmp" de la página del EBI, instalar una máquina virtual con alguna de las distribuciones de Linux, junto con el gestor de bases de datos, y, si es posible, se recomienda configurar pgAdmin4 como interfaz gráfica para ejecutar las preguntas en formato SQL (lenguaje utilizado para el manejo de bases de datos relacionales). Aquí se ha escogido Postgres, debido a ser el que ofrecía un tamaño del archivo de volcado acorde con la capacidad existente en la memoria virtual que se ha dispuesto desde la máquina física (que corre bajo Windows 10)

Preparación de la MV de Ubuntu, PostgreSQL (pgAdmin4) y volcado de datos de ChEMBL.

Máquina virtual: se instala a través del software VMware Workstation 16, con la imagen ISO correspondiente al sistema operativo que queremos simular, correspondientemente descargado, de forma libre, de la página web de la fundación que lo publica y mantiene. Se establece la configuración del SO al gusto del usuario.

PostgreSQL y pgAdmin4: son los programas que nos permitirán crear la BBDD y realizar las preguntas que hemos considerado oportunas (*queries*, como se conocen en el ámbito SQL). Se realiza a través de la consola de comandos del SO Ubuntu (*Shell*) con permisos de superusuario.

```
sudo apt update
sudo apt install postgresql postgresql-contrib
sudo -u postgres psql
sudo -u postgres createdb chembl_26
postgres=# \q

sudo apt-get install pgadmin4
```

Ilustración 6. Comando en el Shell de Ubuntu para instalar Postgres y crear la base de datos para trabajar dentro de este.

Volcado de datos de ChEMBL para crear la BBDD que actúa como servidor en la máquina virtual

También se realiza a través del Shell de Ubuntu, llamando previamente al programa PostgreSQL, y con el archivo de extensión ".dmp" obtenido de la página del consorcio que mantiene esta BBDD, ejecutar un comando idéntico a este:

```
postgres=# \list
postgres=# \q
aramis18@ubuntu:~$ pg_restore --no-owner -h localhost -p 5432 -U postgres -d che
mbl_26 /home/aramis18/TFM/chembl_26/chembl_26_postgresql/chembl_26_postgresql.dm
p
Password:
aramis18@ubuntu:~$ sudo -u postgres psql
[sudo] password for aramis18:
psql (10.12 (Ubuntu 10.12-0ubuntu0.18.04.1))
Type "help" for help.

postgres=# \list
```

Ilustración 7. Comando prototipo en el Shell de Ubuntu para hacer el volcado del archivo dmp en Postgres.

Ejecución de preguntas SQL en pgAdmin4 como cliente en navegador web

Este programa funciona a través del navegador web que tengamos predeterminado (Firefox, por defecto). Hay que configurarlo la primera vez para que este cliente web se conecte al servidor propio del localhost (la propia máquina virtual). Nos permite hacer preguntas a la BBDD de una forma más personalizada, en comparación a las opciones que permite ChEMBL desde su web. Hay dos condiciones que son parte esencial del filtrado/curado de los datos.

Seleccionar los compuestos en base a un criterio de actividad/inactividad mínima.

Aquí, el valor pChembl, considerando mínimo un valor de 5. Esta ha sido la estructura y condiciones impuestas a la pregunta ejecutada aquí:

```sql
SELECT md.molregno, cms.canonical_smiles, cmp.hba, cmp.hbd, cmp.psa,
cmp.rtb, cmp.full_mwt, cmp.heavy_atoms, cmp.aromatic_rings,
td.pref_name, ass.confidence_score, act.bao_endpoint,
act.standard_value, act.standard_units, act.pchembl_value, act.assay_id
FROM activities act
JOIN assays ass ON ass.assay_id = act.assay_id
JOIN target_dictionary td ON td.tid = ass.tid
JOIN molecule_dictionary md ON md.molregno = act.molregno
JOIN compound_properties cmp ON cmp.molregno = md.molregno
JOIN compound_structures cms ON cms.molregno = md.molregno
LEFT JOIN bioassay_ontology bo ON bo.bao_id = act.bao_endpoint
LEFT JOIN assay_type at ON at.assay_type = ass.assay_type
WHERE act.standard_value IS NOT NULL
AND ass.confidence_score >= 9
AND td.pref_name LIKE 'Serotonin 2a%'
AND bo.label = 'Ki'
AND md.molecule_type = 'Small molecule'
AND td.organism = 'Homo sapiens'
AND cms.canonical_smiles NOT LIKE '%.%'
AND act.pchembl_value >=5
```

Ilustración 8. Pregunta en SQL para obtener el listado de moléculas ligadas al receptor 5-HT2A.

Exportación de resultados

Desde el programa web pgAdmin4, que hace que la máquina virtual sea servidor de la base de datos y cliente al mismo tiempo, dentro de la herramienta "*Query tool*", una vez ejecutada y obtenidos los resultados, existe un botón en la esquina superior derecha que permite descargar los resultados obtenidos en formato ".csv" (*comma separated values*), que es portable a entorno Windows.

Preparación de Python 3.7/Spyder, importación de datos.

La instalación del lenguaje de programación Python, como del IDE Spyder *(graphical user interface)* se hacen mediante descarga directa de las páginas[17] indicadas al pie, y ejecución de sus ayudantes de instalación. Spyder está contenido dentro del entorno Anaconda, que ofrece otros IDE útiles para otros usos (*data mining*, *data visualization*, creación de notebooks en explorador web…etc.)

La importación del archivo ".csv" con los datos necesarios para trabajar se puede ya realizar en el IDE Spyder, con un comando similar a este:

```python
import os
os.chdir(r"C:/Users/usuario/OneDrive/EstadisticaUOC/4-SEMESTRE/TFM/Datos_recuperados_ChEMBL")

import pandas as pd

df = pd.read_csv("5ht2a_definitive_nosalts.csv")
```

Ilustración 9. Importación del ".csv" definitivo al entorno de Python/Spyder

17 https://www.python.org/downloads/release/python-370/
https://www.anaconda.com/distribution/

3.2 Cribado de datos. Exploración descriptiva (clustering, desbalanceado)

3.2.1 Cribado de datos

Un paso fundamental antes de comenzar a trabajar con los datos es hacer un cribado de estos, pues pueden contener valores que añadan ruido de fondo, así como duplicados, que influirían en los resultados de los modelos.

Se han de obtener los InChIKey de cada molécula con OpenBabel IDE o mediante la librería de Python correspondiente, desde la notación SMILES que hemos extraido en el ".csv" mediante pgAdmin4.
Con estos, guardados en formato ".txt" (con un editor de texto), añadirlos al dataframe ya disponible de las moléculas (Python).

Los dos pasos fundamentales son realizar la eliminación de todos aquellos duplicados, para quedarse únicamente con uno de ellos, mediante la notación InChiKey, así como eliminar las sales del dataset, si no se hubiera incluido en la query de SQL pertinente, para obtener los datos para los modelos; de esta forma solo trabajaremos con moléculas neutras.

```
import numpy as np

df_2 = np.genfromtxt(fname="inchikeys.txt", dtype="str", skip_header=1)

df_2df = pd.DataFrame(data=df_2, columns=["InChIKey_notation"])

df_final = pd.concat([df, df_2df], axis=1)

df_final.drop_duplicates(subset="InChIKey_notation",
                         keep = 'first', inplace = True)

df_final.to_csv("results_unique_p5.csv", index=False)

df_final.head()
```

Ilustración 10. Cribado de datos en Spyder.

Finalmente, el set de datos tendrá las siguientes columnas: *molregno,* ***canonical_smiles****, hba, hbd, psa, rtb,* ***full_mwt****, heavy_atoms, aromatic_rings, pref_name,* ***confidence_score****, bao_endpoint, standard_value, standard_units,* ***pchembl_value****, assay_id,* ***InChIKey_notation****.*

3.2.2 Exploración descriptiva

De los pasos anteriormente ejecutados obtenemos un set de datos de 2628 observaciones, de las cuales vamos a hacer una exploración descriptiva, para entender mejor la estructura de los datos que se van a analizar. La variable por estudiar será el valor "*pchembl value*" que hemos introducido previamente en el apartado *2.3 Elección de la diana terapéutica 5-HT2A y la constante de inhibición Ki* Se utilizan las siguientes librerías de Python para ello*: Pandas, Yellowbrick, Seaborn, Matplotlib y Scikit-learn cluster* (DBSCAN).

La variable "pchembl value" (en adelante, pChembl) se va a estudiar tanto como una variable numérica continua como una variable categórica, donde aquellos

compuestos con un pChembl igual o mayor a 6, han sido calificados como activos, y aquellos con un valor inferior a 6, como inactivos.

- **pChembl ≥ 6: activos.**
- **pChembl < 6: inactivos.**

Estudio de outliers[18]

El estudio de los puntos atípicos (o *outliers*) también resulta importante, para saber si hay demasiados elementos en el set de datos que puedan distorsionar los modelos y sus predicciones. Se realiza mediante la librería Scikit-learn clustering – DBSCAN, que es un algoritmo de clustering, o aprendizaje no supervisado, cuya finalidad es remarcar aquellos valores atípicos tras generar clusters o grupos de puntos de alta densidad.

Este algoritmo considerará *outlier* todo punto alejado y en zonas poco densas, en base a unos parámetros que se le han de definir; son el parámetro épsilon (eps) y el número mínimo de muestras (min_samples).

A partir de un punto arbitrario, si hay una cantidad de puntos mayor o igual al número mínimo de muestras establecido a una distancia épsilon de dicho punto, todos esos puntos se considerarán parte de un cluster (grupo). Luego, a partir de cada uno de esos puntos, se repite el mismo proceso, e igual, si dicho punto subyacente tiene tantos o más puntos como se ha marcado en el mínimo de muestras a distancia épsilon, el cluster irá creciendo de forma recursiva.

Si en alguno de los puntos analizados, este tiene menos puntos del mínimo de muestras establecido a la distancia épsilon, y tampoco pertenece a ningún otro cluster, entonces se considerará un punto atípico.

Para ejecutar este algoritmo es necesario que utilizemos dos variables escalares, las cuales habrán de ser normalizadas para que sus valores estén entre el 0 y 1; una es la variable de interés previamente nombrada, y otra es el peso molecular, que esta disponible con la extracción de los datos.

Los primeros parámetros se pueden establecer sin hacer ninguna prueba adicional, sabiendo que el épsilon (*eps*) es conveniente partir de un valor no superior a 0,1. Con las diferentes combinaciones de *eps* y *min_samples* se conformarán diferentes cluster y sus posibles recurrentes, así como diferentes puntos atípicos. Esta primera representación es con *eps* = 0.08 y *min_samples* = 15.

[18] (Wallach & Heifets, 2018)

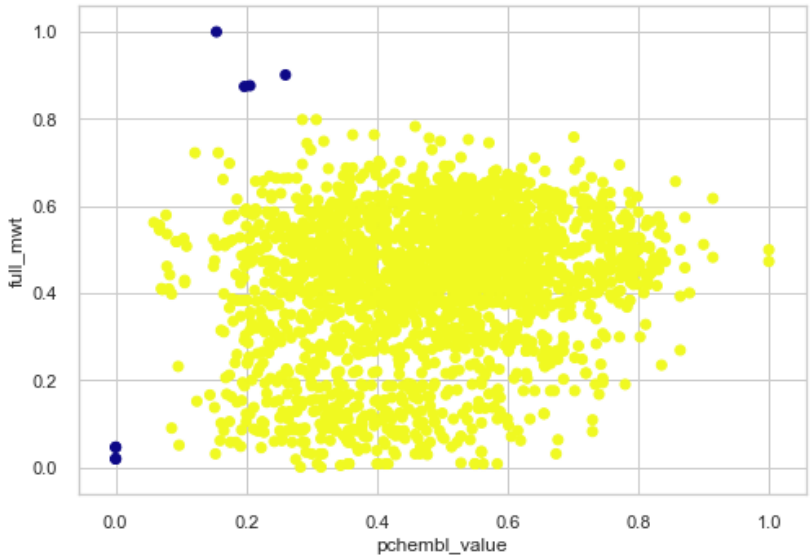

Gráfico 1. Scatterplot de todas las observaciones con unos pocos puntos atípicos.

De la representación anterior, se podría identificar y eliminar estos puntos atípicos, pero si se afinan un poco más los parámetros, se podrán recoger algunos puntos atípicos más, que pueden hacer más precisos los modelos.

Una técnica habitual es la de representar el gráfico del codo, fijando el parámetro min_samples y representando todos los radios épsilon de los puntos ordenados por distancia (eje horizontal). Cuando los radios comienzan a aumentar de forma exponencial y se dibuja el “codo”, significa que se está pasando de la zona de alta densidad de puntos a la zonas de baja densidad de puntos. Aquí, *eps* parece estar entre 0.04 y 0.05.

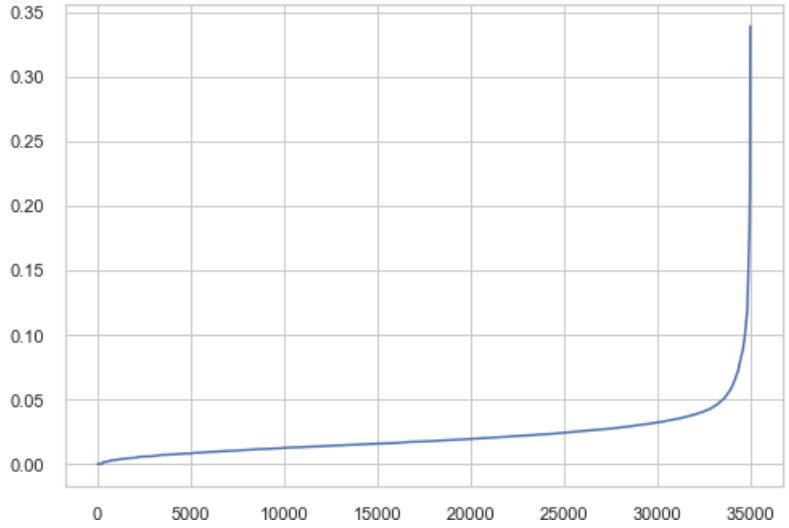

Gráfico 2. Representación en codo con los componentes principales, para conocer el valor de eps.

Con este paso, se redibuja el gráfico anterior, observando que recoge un mayor número de puntos atípicos. Del set de datos utilizado para estos análisis, que

contenía 2335 muestras, se identifica un total de 103 puntos atípicos (4.6% del total) que se podrán eliminar para mejorar los análisis a continuación; quedan en el set 2232 observaciones, de las cuáles, otras 359 se eliminan por tener algún valor NaN (not a number). El número de observaciones total es **1873**.

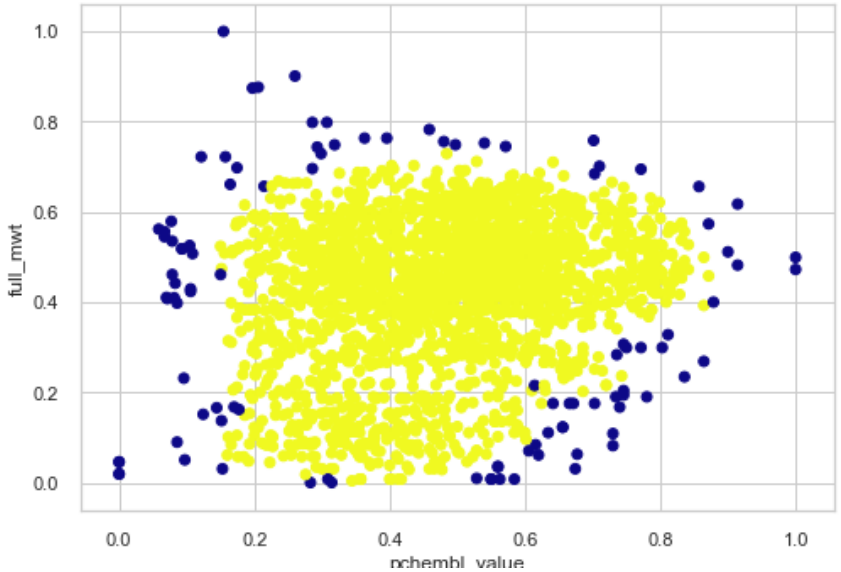

Gráfico 3. Scatterplot de todas las observaciones con más puntos atípicos detectados.

Las características principales del set de datos original son las siguientes:

- Valor mínimo de pChembl = 4.0
- Valor máximo de pChembl = 11.0
- Número de elementos por clasificación categórica: 198 inactivos, 1675 activos.
- Media aritmética de pChembl por categoría: 5.61 para los inactivos, 7.62 para los activos.
- Desviación típica de pChembl por categoría: 0.23 para los inactivos, 0.91 para los activos.
- Porcentaje de presencia de compuestos inactivos en el set de datos: 10.57%.
- Porcentaje de presencia de compuestos activos en el set de datos: 89.43%.

Como se puede apreciar, la distribución de las clases de compuestos (activo e inactivo) está muy desbalanceada; un desequilibrio como este, de tomarse los datos como tal, generaría modelos de *machine learning* que producirían predicciones claramente arbitrarías hacia la clase mayoritaria. Es por esto que, para realizar unos modelos lo más robustos posibles, se habrá de buscar un método para balancear ambas clases.

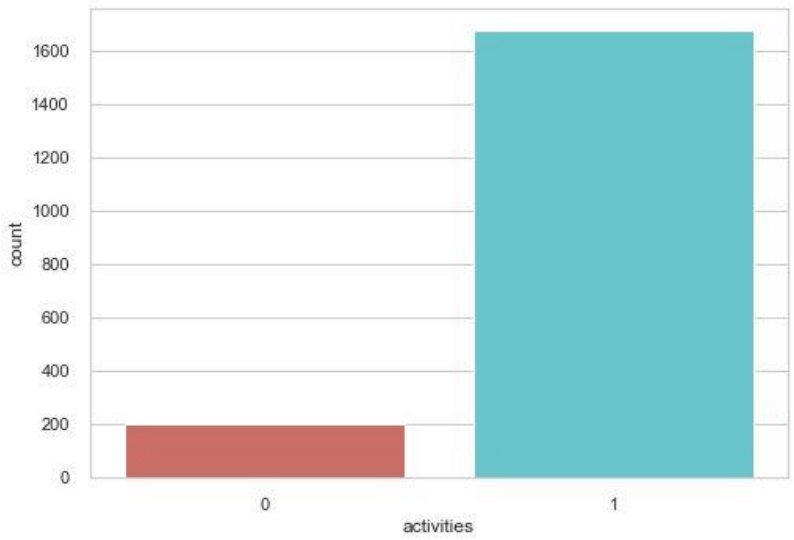

Gráfico 4. Distribución desbalanceada de las clases.

Del estudio de la variable pChembl como tipo numérica continua, podemos además saber que los valores de esta no siguen una distribución normal. Si se puede apreciar que una proporción importante de los registros tienen valores de pChembl entre 6 y 8, siendo estos compuestos categorizados como activos.
Para verlo de forma más evidente, se puede dibujar un gráfico de tipo violín, donde se aprecia mejor la distribución de pChembl por las diferentes clases.

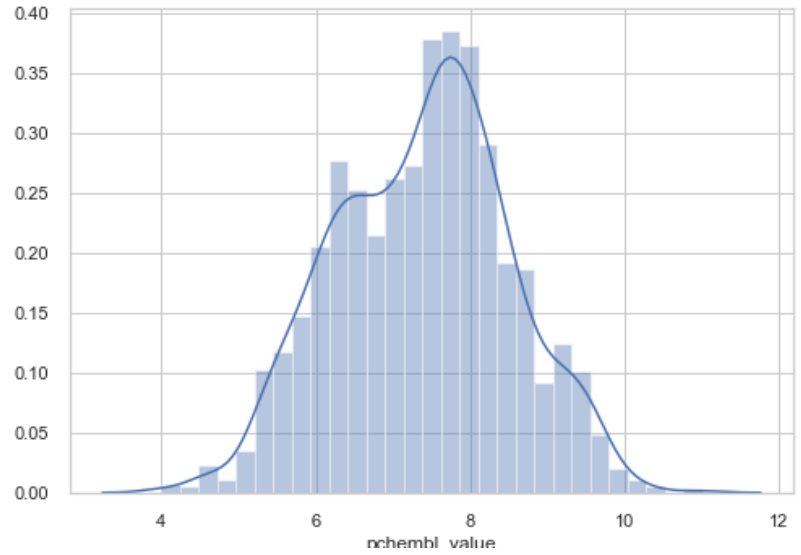

Gráfico 5. Distribución del pChembl para todo el conjunto.

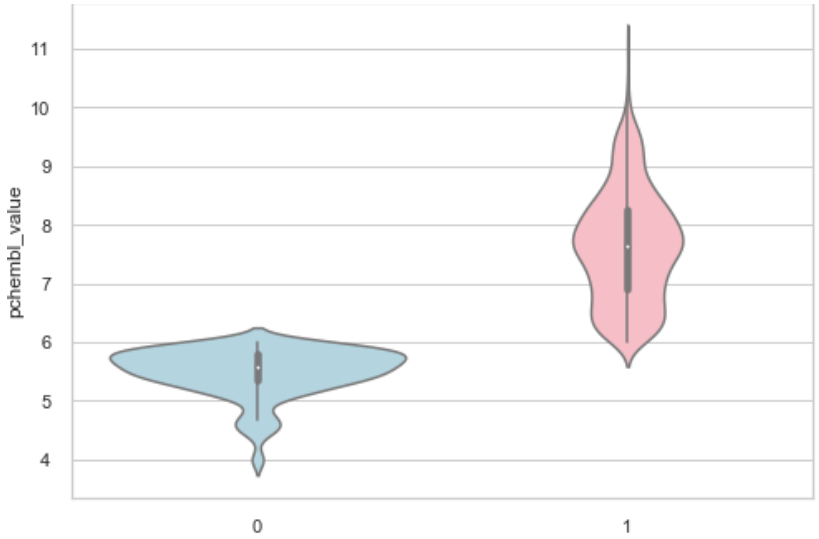

Gráfico 6. Distribución del pChembl por clases.

Balanceado

Para realizar el balanceo entre clases, se usa la librería *Imbalance-learn* de Python, que nos permitirá realizar, en este caso, acorde a la necesidad, un remuestreo aleatorio de los datos de la clase mayoritaria, en este caso, la activa. Se trata de reducir el número de componentes de los compuestos activos hasta que haya tantos como de los compuestos inactivos. Así, el set de datos balanceado, que será de vital importancia para los modelos de *machine learning* del tipo de clasificación, queda así:

- Número de elementos por clasificación categórica: 198 inactivos, 198 activos. Total observaciones: n = 396.
- Porcentaje de presencia de compuestos inactivos en el set de datos: 50%.
- Porcentaje de presencia de compuestos activos en el set de datos: 50%.

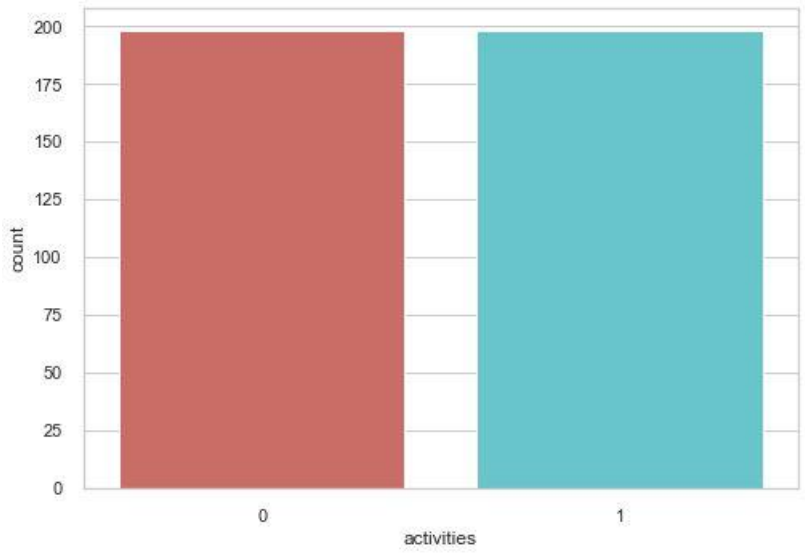

Gráfico 7. Distribución balanceada de las clases.

3.3 Obtención de los fingerprints y estructura de los datos para ser usados en los modelos.

El siguiente paso es obtener los fingerprints (traducción de la estructura 3D de la molécula a un único vector binario de 0 y 1) de todas las moléculas del set de datos, mediante la librería *rdkit*, de instalación directa en el entorno Anaconda, en la consola de comandos propia de este entorno. El comando utilizado ha sido este:

La columna de interés para obtener los fingerprints es la relativa a la notación de la molécula en formato SMILES (canonical_smiles), puesto que el primer paso es utilizar esta notación para añadir la representación de cada una de las moléculas, en una nueva columna, al set de datos (tipo de objeto *RD molecular object* o *ROMol*).

Cuando está preparada esta columna, el siguiente paso es obtener columnas, para todas las muestras, con diferentes fingerprints; es decir, diferentes modos de recoger la información más relevante de la molécula. Existen muchas, siendo incluso un área de la química computacional abierta a la mejora y constante innovación de nuevas formas de representación.

En cualquier caso, hay unos cuantos tipos de fingerprints que suelen ser los más utilizados, y, para los cuales, existe reportada abundante bibliografía de estudios de QSAR. Las más conocidas y trabajadas son: Morgan, MACCS y los fingerprints topológicos.

3.3.1 Fingerprints Morgan y MACCS

Morgan/ECFP [19] [20] [21]

Este tipo de fingerprint también se conoce como ECFP, por las siglas en inglés de *Extended-Connectivity Fingerprint*, que hace alusión a la forma de crear el vector binario para cada molécula.

El fingerprint sigue unos pasos para ser construido, que se resumen en estos puntos:

- Se asigna cada átomo de la molécula con un identificador, como un número entero, que recoge información de: nº de átomos vecinos que no son hidrógenos, número de enlaces unidos al átomo (sin contar los de hidrógeno), número atómico, masa atómica, número de hidrógenos conectados al átomo y si dicho átomo forma parte de un anillo o no. El proceso se repite para todos los átomos de la molécula.
- Realizar la actualización del valor numérico de cada átomo como iteraciones, de forma que, en cada vuelta, se alcance un átomo más de

[19] (Fara & Oprea, s.f.)
[20] (Laksh, 2019)
[21] (Bajorath)

distancia respecto al que se considera. De esta forma, se consigue saber, no solo la conectividad del átomo considerado inicialmente, sino como lo hacen aquellos a los que este se conecta. Esto permite definir con mayor exactitud las subestructuras químicas que se van conformando en la molécula.
    - El resultado es un array que reúne los identificadores del átomo considerado con los adyacentes. Este array se divide y recalcula de forma aritmética para dar lugar a un nuevo identificador para ese átomo.
    - El nuevo fingerprint es un array donde se han unido los identificadores iniciales de los átomos observados con aquellos identificadores "recalculados"
- En cada iteración o vuelta, se repite el paso anterior, y además se eliminan aquellas subestructuras duplicadas, que al ir considerando cada uno de los diferentes átomos, se han anotado 2 o más veces, pero hacen referencia a la misma, en las mismas posiciones. Las iteraciones marcan el "radio" que se asigna para calcular este fingerprint, que normalmente se fija entre 2 y 4.
- Con las iteraciones deseadas realizadas, el array resultante de todos los átomos y sus iteraciones se transforma, mediante un algoritmo más complejo, en un vector binario de 0 y 1 de 2048 bits de longitud.

MACCS [22]

Este tipo de fingerprint se trata de un string (cadena de caracteres) de 166 bits, que se basa en la estructura 2D de la molécula en cuestión, y se basa en realizar una serie de preguntas relativas a la presencia o ausencia de subestructuras químicas determinadas.
Las preguntas abarcan la presencia de un determinado número de átomos de elementos clave (ej. O, F, Cl, Br, I), si existen enlaces entre átomos determinados, si hay alguna subestructura en forma de anillo, si existe algún grupo funcional químico determinado (ej.: alcoholes, cetonas, carboxilos, aminas, amidas...) ...etc.

3.3.2 Obtención del string binario y preparación para su uso con algoritmos.

Con la librería rdkit y sus funciones, obtendremos un vector de bits, únicamente como valores posibles 0 y 1 (objeto tipo *BitVect*) desde la columna que alberga la representación de la molécula (objeto *ROMol*).

Este tipo de vector binario, aunque pueda ser visualizado por pantalla o consola, necesita hacerle una modificación adicional, que es pasarlo del tipo de objeto BitVect a un string de texto plano; esto se consigue con la librería *rdkit.DataStructs* y la función *BitVectToText*.

Una vez ya esté el string de 0 y 1, el último paso es romper dicha cadena en todos los bits que la componen. Según el tipo de fingerprint escogido, el número de bits puede variar, desde 32 a 2048 bits; lo que es importante considerar es que, a mayor número de bits, la información que está recopilando de la estructura

[22] (Scientific, 2013) (Li, 2017)

de la molécula será más detallada y amplia, que cuando menor sea el número de estos bits.

3.3.3 Estructura de los datos para ser usados en los modelos

Con esta partición del string en tantas columnas como bits tenga este, tendremos las variables de entrada que necesitamos para probar sobre los diferentes algoritmos de clasificación y regresión.

Por último, es importante localizar de nuevo, y asignar a una variable dedicada dentro Python, en el set de datos la variable que se quiere predecir, en este caso, pChembl, como un valor numérico escalar para aquellos modelos de regresión, así como un valor categórico (activo:1; inactivo: 0) para trabajar con los modelos de clasificación.

3.4 Desarrollo de modelos de machine learning. Clasificación.

3.4.1 Logistic regression[23] [24]

Este algoritmo toma su nombre de la función matemática que se usa para llevar a cabo la clasificación, la función logística o sigmoide. Es una curva en forma de S que puede tomar cualquier valor numérico real y mapearlo a un valor entre 0 y 1. Se escribe así:

$$1/(1+e^{-valor})$$

Funciona de forma que los valores que se le aportan como las variables descriptoras (x) se hace una combinación lineal de ellas, con pesos o coeficientes para cada una de ellas (que se obtienen del subconjunto de datos del entrenamiento), para predecir la variable regresora (y), que solo podrá tener valores binarios, 0 o 1, en vez de un valor numérico.

$$y = e^{(b0+b1*x)}/\left(1+e^{(b0+b1*x)}\right)$$

Este algoritmo lo que hace es modelar la probabilidad de que ocurra una de las dos clases, dada una o más condiciones. Ej. Que se dé la clase por defecto, 1, dada una variable de entrada X.

$$P(X) = P(Y = 1|X)$$

Con la combinación lineal, queda así:

$$P(X) = e^{(b0+b1*x)}/\left(1+e^{(b0+b1*x)}\right)$$

[23] (Li, 2017)
[24] (Brownlee, Logistic Regression for Machine Learning, 2016)

A la que se le puede realizar una transformación con logaritmo neperiano para eliminar la e del término de la derecha, y simplificar la expresión:

$$ln\left(P(X) - \frac{1}{P(X)}\right) = b0 + b1 * x$$

A la derecha quedará el cálculo de la variable regresora (y), que vuelve a ser lineal de nuevo, y a la izquierda quedará el logaritmo neperiano de la probabilidad de la clase por defecto escogida. Esto no es más que la probabilidad de que se produzca tal evento (clase 1 dados una serie de valores de las x), dividido por la probabilidad de que no se produzca tal evento.

3.4.2 kNN[25]

Se trata de un algoritmo de clasificación que clasifica ejemplos sin etiquetar asignándoles la clase de ejemplos similares. Es muy sencillo, pero muy potente, y su buen poder predictivo ha sido ampliamente contrastado en investigación y aplicaciones prácticas. No necesita que el conjunto de datos siga ningún tipo de distribución en particular.

Usa la información de los vecinos adyacentes (ya etiquetados) al punto que se quiere clasificar (no etiquetado), considerando un número *k* variable de vecinos, siendo este el parámetro sensible de este algoritmo.

kNN identifica *k* casos en el subconjunto de datos destinados al entrenamiento que están cerca del punto que se quiere etiquetar; a este punto se le asigna la clase mayoritaria entre esos *k* vecinos cercanos.

kNN trata las variables descriptoras como características en un espacio multidimensional. Los *k* vecinos cercanos se consideran en dicho espacio como distancias euclídeas, para poder seleccionar aquellos *k* vecinos más cercanos con sus respectivas etiquetas.

Para evitar el sobreajuste o infraajuste del algoritmo relativo a la elección del parámetro *k,* y así evitar que la clase mayoritaria siempre prevalezca a la minoritaria o que los puntos atípicos puedan etiquetar incorrectamente un caso, se da en la práctica algunas praxis, como empezar usando la raíz cuadrada del número de casos que haya en el subconjunto de entrenamiento. Otra buena práctica es probar varios valores para la *k* en una variedad de subconjuntos de test diferentes.

3.4.3 Naive Bayes[26]

Se trata de un algoritmo que aplica el teorema de Bayes a problemas de clasificación; estudia la probabilidad de un evento condicionado por la existencia de otro, es decir, probabilidades condicionales.

[25] (Lantz, 2015)
[26] (Lantz, 2015)

El teorema dice que la estimación de la probabilidad de un evento A dado B, P(A|B), debe basarse en la probabilidad que A y B ocurran juntos (P(A ∩ B)) y la probabilidad de que B se observe en general (P(B)). Si B es poco frecuente, P(B) y (P(A ∩ B)) siempre serán pequeñas; si A y B casi siempre ocurren juntas, P(A | B) será alta sin importar la probabilidad de B.

Realiza la asunción por la que considera igual de importantes todas las variables descriptoras, así como que estas variables son independientes entre sí. Si bien no es habitual en aplicaciones del mundo real, puede ajustarse al conjunto de datos que se tienen, especialmente si hay muchas variables descriptoras que considerar.

Para aplicarlo al *machine learning*, lo que se hace es añadir variables descriptoras adicionales, de forma que los cálculos de las probabilidades condicionadas se van complicando sucesivamente, donde A sería la clase de la que queremos conocer su probabilidad condicionada por varios eventos B.

En estos cálculos, se hace uso de la suposición de la independencia entre eventos A, B y otros; lo que simplifica los cálculos, pudiendo expresar tal probabilidad condicional como el producto de cada uno de los eventos B, por separado, dado A.

3.4.4 Gradient Boosting[27] [28]

Se trata de una técnica, más que de un algoritmo en sí, tanto de clasificación como de regresión, que se fundamenta en la idea de utilizar numerosas veces un algoritmo de aprendizaje débil, o *aprendiz* (aquel cuyo desarrollo es ligeramente mejor que cualquier otro por azar)

La idea básica es la de filtrar las observaciones, dejando aquellas que el algoritmo de aprendizaje débil prediga sin esfuerzo aparte, para centrarse, en sucesivas vueltas, en predecir las observaciones más complejas.

Desde la posición más estadística, el objetivo es minimizar las pérdidas del modelo añadiendo aprendices débiles, siguiendo un procedimiento de descenso del gradiente. Por eso se conocen como “modelos aditivos por etapas”; un nuevo aprendiz débil se añade a cada etapa, mientras que el resto se mantienen inalterados.

Tiene, por lo general, 3 componentes básicos:

- La función de pérdida, que cuantifica el fallo en las predicciones, y para problemas de clasificación suele ser de tipo logarítmico;
- Un aprendiz débil, los árboles de decisión, a los cuales se les puede aplicar diferentes parámetros de restricción en su comportamiento
- Un modelo aditivo, por lo general, un procedimiento de descenso de gradiente, que busca minimizar la pérdida de capacidad predictiva conforme se añaden árboles de decisión. Se añade un árbol, se calcula la pérdida residual, y seguidamente se añade un nuevo árbol que reduzca

[27] (Brownlee, A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning, 2016)
[28] (Friedman, 2001)

la pérdida, lo cual se consigue modificando algunos parámetros de dicho árbol que consigan que la toma de decisiones se mueva en la dirección correcta, reduciendo la pérdida residual.

Se suele fijar un número máximo de árboles a añadir, o un criterio de parada de adición (ej. un nivel de pérdida residual aceptable) o se alcanza el tope de mejora del modelo sobre un conjunto de datos de validación.

3.4.5 SVM[29]

Se trata de un algoritmo tanto de clasificación como de regresión, cuyo objetivo es el de trazar una frontera imaginaria entre los puntos de datos existentes en un espacio multidimensional, técnicamente llamada hiperplano, que divide el espacio dibujado en particiones homogéneas a cada lado de este.

Para los supuestos de clasificación binaria, el objetivo del algoritmo es identificar la línea en el espacio multidimensional que separe las dos clases dejando la máxima separación posible entre ellas.

Esto garantiza que el futuro modelo generalizará mejor cuando se le pasen nuevos datos para clasificar, y si hay observaciones en el conjunto de entrenamiento que pudieran ser parte del ruido, los puntos a clasificar se coloquen en el lado correcto del hiperplano.

Para los casos en que el espacio multidimensional no se pueda separar de forma lineal, existe la posibilidad de introducir una variable de holgura, que dibuja un hiperplano que permite que algunos puntos caigan en el lado equivocado del plano, aplicándoles a estos una función de coste. Entonces, el algoritmo, en lugar de intentar buscar la máxima separación entre ellas, busca la forma de minimizar el coste total que se derive de los puntos “mal clasificados”.

Otra alternativa es cambiar el *kernel* (o núcleo) en el sentido de redibujar las variables o características en el espacio multidimensional, de forma que el espacio que inicialmente parece no poderse separar de forma lineal, si se realiza una transformación matemática, como combinación de dos variables, se obtiene una perspectiva diferente, desde la cual, si es viable la separación lineal de las observaciones en sus clases.

En el trabajo, se ha probado esta última técnica con la modificación del *kernel*, tanto con transformación polinómica, sigmoide como gaussiana.

3.4.6 Random Forest[30] [31]

Se trata de una combinación o “bosque” de predictores en forma de árbol donde cada uno de ellos tiene unos parámetros extraídos de muestras al azar, con la misma distribución en todos los árboles del bosque. La idea clave es añadir diversidad adicional a los árboles de decisión.

[29] (Lantz, 2015)
[30] (Breiman, Random Forests, 2001)
[31] (Breiman, Arcing The Edge, 1998)

Selecciona las variables descriptoras que considera más importantes, y por eso se puede usar en datos con numerosas variables descriptoras u observaciones.

Cuanto mayor es el número de árboles de decisión en un bosque o la fuerza predictiva de los árboles de forma individual, los errores en la generalización que puedan cometer los bosques tienden a un límite.

Estas características los hacen especialmente complicados de interpretar, y para conseguir los mejores resultados, puede ser necesario invertir bastante tiempo, por ensayo y error, con los diferentes parámetros del modelo. Como normal general, los procesos de aprendizaje más complejos y con conjuntos de datos más grandes (en variables u observaciones) funcionan mejor con un mayor número de árboles.

3.5 Desarrollo de modelos de machine learning. Regresión.

3.5.1 Regresión lineal. Regresión con penalizaciones (*ridge regression*) [32] [33]

Se trata de un algoritmo de regresión donde se pretende encontrar la relación lineal $y = X_i * \beta$, donde y es la variable regresora que se quiere predecir, y Xi las variables descriptoras, con su parámetro de regresión propio, β, para cada una de ellas.

$$y = Xi * \beta$$

La diferencia con cualquier otro tipo de análisis bioinformático es que, cuando se intentan analizar conjuntos de datos que presentan tantas variables descriptoras (X_i) como observaciones (n), $X_i \approx n$, o incluso más variables descriptoras que observaciones, $X_i > n$, es que la matriz de datos a usarse como entrada de datos para los análisis resulta impracticable en términos computacionales (matriz de elevada dimensionalidad), y resulta necesario examinar **modelos lineales alternativos** que puedan asimilar dicha elevada dimensionalidad de la matriz de datos.

A grandes rasgos matemáticos, cuando la matriz de diseño es de elevada dimensionalidad, las columnas se dice que sufren colinealidad, es decir, que 2 o más de dichas columnas están linealmente relacionadas, y no son independientes, lo que hace matemáticamente imposible separar la contribución de cada variable descriptora de forma individual. Este hecho se refleja en el ajuste de la regresión lineal clásica mostrado un error de gran tamaño, que corresponde a las variables descriptoras linealmente dependientes.

Una colinealidad de una matriz X de diseño *"n x p"* implica que el rango de la *"p x p"* matriz X^TX es más pequeño que p, y, en consecuencia, esta matriz es singular (su determinante es 0) y no tiene inversa. Entonces, cuando la matriz de diseño X es de elevada dimensionalidad, el parámetro de la regresión β no se puede estimar, y se hace oportuno buscar dicha alternativa.

[32] (Kim, 2019)
[33] (James, Witten, Hastie, & Tibshirani, 2017)

Una de ellas es la ***ridge regression*** (o de cresta), descrita por primera vez en 1970, que busca resolver el problema de la singularidad de tal matriz X^TX añadiendo un parámetro a esta, λ, tal que $X^TX + \lambda I$ (donde $\lambda \in [0,\infty)$). El parámetro lambda, λ, es un valor escalar que se conoce como el parámetro de penalización, y que permitirá poder hallar el parámetro de la regresión β. Cada valor de lambda, λ, dará lugar a unos parámetros de la regresión β's diferente.

$$\hat{\boldsymbol{\beta}}(\lambda) = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{pp})^{-1} \mathbf{X}^\top \mathbf{Y}$$

Además, el uso de la *ridge regression* se justifica por el hecho que, al cambiar el parámetro lambda, λ, y los coeficientes del modelo, β's, significa que se puede construir un modelo parcializado, donde las variables descriptoras del modelo puedan ser tratadas con diferentes pesos, según la importancia de cada una de ellas sobre la variable regresora y.

El parámetro lambda λ también se conoce como la penalización, porque lo que hace, si bien permite la solución matricial de la cuestión, es, a su vez, incrementar la suma de los residuos (del modelo) al cuadrado, medido con el error cuadrático medio. La lambda λ que minimice tal error deberá ser la seleccionada para el modelo final.

Si nos fijamos una última vez en la fórmula de arriba, se puede ver que, cuanto mayor sea lambda λ, los coeficientes del modelo β's deben disminuir, sin que ello suponga que ninguno de ellos sea 0. Este tipo de regresión da diferente importancia a los pesos de las variables descriptoras, pero no elimina ninguna de ellas.

3.5.2 Gradient Boosting

Sigue los mismos conceptos básicos que el algoritmo de clasificación, con la diferencia que la función de pérdida suele ser el error cuadrático, en lugar de cuantificar la pérdida de forma logarítmica.

3.5.3 SVM (regresión) y Random Forests (regresión)

Siguen los mismos conceptos básicos que el algoritmo de clasificación, pero siendo la variable regresora de tipo numérica continua.

3.6 Aplicación de método de validación sobre los modelos.[34] [35] [36]

Como se había comentado en el punto *2.1 Flujo de operaciones habitual previo al screening computacional en desarrollo de nuevos fármacos*, cuando se han comentado las operaciones habituales en el screening computacional, existen varias técnicas para realizar la validación de los modelos de *machine learning* desarrollados, pero no se había ahondado en su importancia y el porqué de su

[34] (Anguita, Ghelardoni, Ghio, Oneto, & Ridella, 2012)
[35] (Fushiki, 2011) (Khandelwal, 2018)
[36] (Krishni, 2018)

ejecución para poder hacer una lectura de los resultados que se obtengan con el suficiente rigor científico.

Una de las formas habituales de validación de los modelos ha sido la partición del conjunto de datos original en 2 subconjuntos de entrenamiento y test, pudiendo, en ocasiones, mejorarse a una partición de 3 subconjuntos incluso, con subconjunto de entrenamiento, test y otro de validación. El problema con este tipo de técnicas es que la exactitud que se obtiene no es verídica del comportamiento que tendrá el modelo clasificador o regresor sobre un conjunto de datos que se le pueda pasar y no haya visto antes.

Por esta razón, en la literatura científica y en el ámbito investigacional se opta prioritariamente por una técnica más exacta y precisa, denominada ***k-fold cross validation*** (validación cruzada con k subconjuntos), que es la escogida en este trabajo, con $k = 10$.

3.6.1 *K-fold cross validation*

Se trata de una técnica de validación basada en dividir el conjunto original de datos en *k* partes, donde cada una de las partes es usada una vez como un subconjunto de test en cada una de las *k* veces que se repite el procedimiento de testado del modelo en cuestión; mientras, las *k-1* partes restantes de la partición son las que se usan para entrenar el modelo de interés.

Esta técnica suele predominar frente a la validación *leave-one-out,* que se detalla más a continuación, porque no supone una sobrecarga computacional para ninguna máquina física, especialmente cuando se trabaja en el campo de la bioinformática o ciencia de datos, con numerosas variables predictoras u observaciones.

Funciona especialmente bien con valores para la $k \leq 5$, siendo este de los más utilizados, junto con ***k = 10***. La elección del parámetro *k* se ha hecho hasta ahora de una forma experimental, mediante ensayo y error, pero ya existen algunas técnicas para hallar el mejor valor posible de esta; por ejemplo, una es mediante la introducción del parámetro *k* como uno más de los hiperparámetros que se puedan probar del modelo que se quiera estudiar, durante la fase de entrenamiento y mejora del modelo.

Su contrapartida es el sesgo hacia arriba al que suele inducir, dando lugar a la posibilidad de hacer lecturas de las métricas de un modelo excesivamente halagüeñas. En dichos casos, y para mejorar su rendimiento, es oportuno mencionar los avances en investigación matemática que redefinen esta técnica con recursos para corregir tal sesgo.

3.6.2 *Leave-one-out validation*

Se trata de una técnica de validación total, donde el parámetro *k* es igual al número de observaciones (n) en el set de datos. Supone que k = n veces se va a entrenar el modelo que se esté estudiando, donde se tomarán n-1 observaciones para la fase de entrenamiento del modelo, y el punto eliminado

se utilizará para realizar una predicción, como si fuera un subconjunto de test de un solo punto.

La contrapartida de esta técnica está en que supone una gran carga computacional para la máquina física, sobre todo cuando se trabaja con sets de datos de miles de observaciones.

4. RESULTADOS

A continuación, se presentan los resultados de los modelos introducidos anteriormente de forma teórica, pero con los diferentes parámetros con los que se ha ido probando, de forma justificada en cada uno de los casos.

Para una comprensión mayor, se definen de forma sencilla las métricas que se exponen a continuación:

- **Sensibilidad:** capacidad de un modelo de evitar los falsos negativos; o lo que es lo mismo, de predecir correctamente todas las observaciones positivas en todas las observaciones existentes en dicha clase.
- **Especificidad:** capacidad de un modelo de evitar los falsos positivos
- **Exactitud:** predicciones correctamente realizadas del total de observaciones.
- **Precisión:** capacidad de un modelo de ofrecer resultados consistentes.
- **F1**: es la media ponderada de la precisión y la sensibilidad. Esta métrica considera tanto los falsos positivos como los falsos negativos.

En cuanto a la sensibilidad y especificidad de todos los modelos es importante considerar que es un resultado de una sola de las posibles particiones del conjunto de datos en subconjunto de entrenamiento y test (por la complejidad inherente al código), mientras que la exactitud y F1 son validación *10-fold-cross-validation* (CV), y para realizar los posteriores juicios de valor, se optará arbitrariamente por estos últimos.[37]

4.1 Evaluación de modelos de clasificación sobre el set de datos una vez balanceado.

A continuación, se exponen los resultados de los modelos de regresión explicados anteriormente en el punto *3.4 Desarrollo de modelos de machine learning. Clasificación.*; para todos ellos, el conjunto de datos para probar los diferentes modelos ha sido el expuesto en el punto *3.3 Obtención de los fingerprints y estructura de los datos para ser usados en los modelos*., donde = 1873, con los puntos atípicos ya eliminados, así como aquellas filas donde la variable pChembl presentaba un valor "*nan*" (*not a number*)

[37] (Shung, 2018)

4.1.1 Fingerprints tipo Morgan.

Logistic regression

Los hiperparámetros más importantes a la hora de configurar el modelo son:

- ***class_weight = None***: no añade ninguna corrección en los pesos de las diferentes clases.
- **random_state**: generador de números aleatorios, que decidirá como se divide el set de datos en subconjunto de entrenamiento y testado, si fuera necesario, para asegurar que se realiza de la misma forma tras varias ejecuciones de una misma parte de código. Necesario para la validación de cualquier modelo.

Con esto, se obtienen los siguientes resultados:

Sensibilidad: 0.5714

Especificidad: 0.4524

Exactitud (*accuracy*) con 10-fold-CV: 0.8686

F1 con 10-fold-CV: 0.927

De este modelo de clasificación, para este conjunto de datos, se puede decir que es un buen modelo, bastante exacto, con resultados consistentes, y una

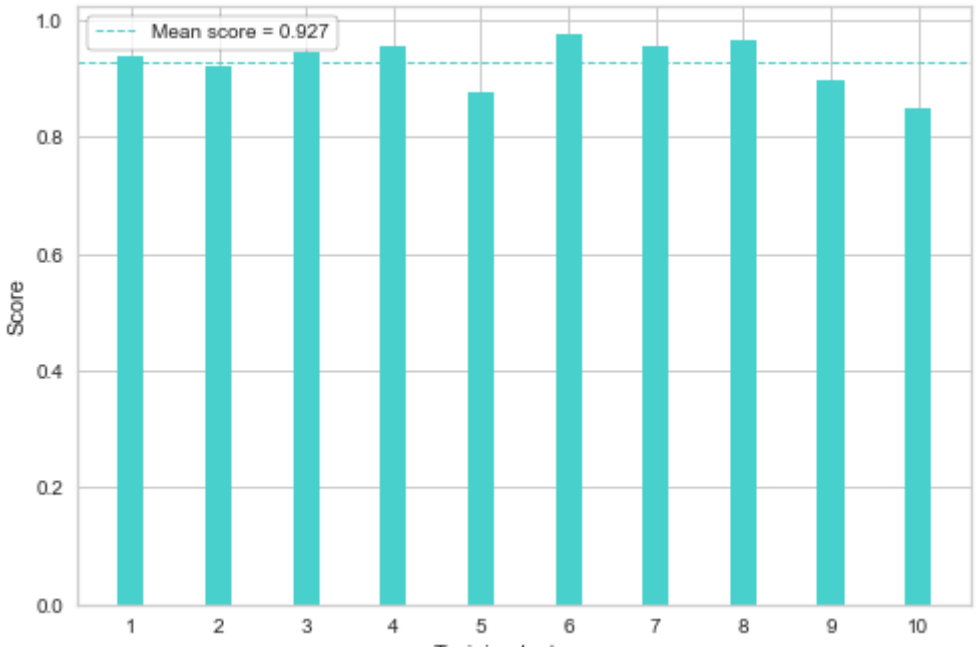

Gráfico de resultados 1. Métricas de exactitud para cada partición de la validación cruzada en Logistic Regression (Morgan). Resultado medio de la exactitud tras validación cruzada.

sensibilidad aceptable. La baja sensibilidad es fruto de una partición entrenamiento – test, y no global, de todas las posibles.

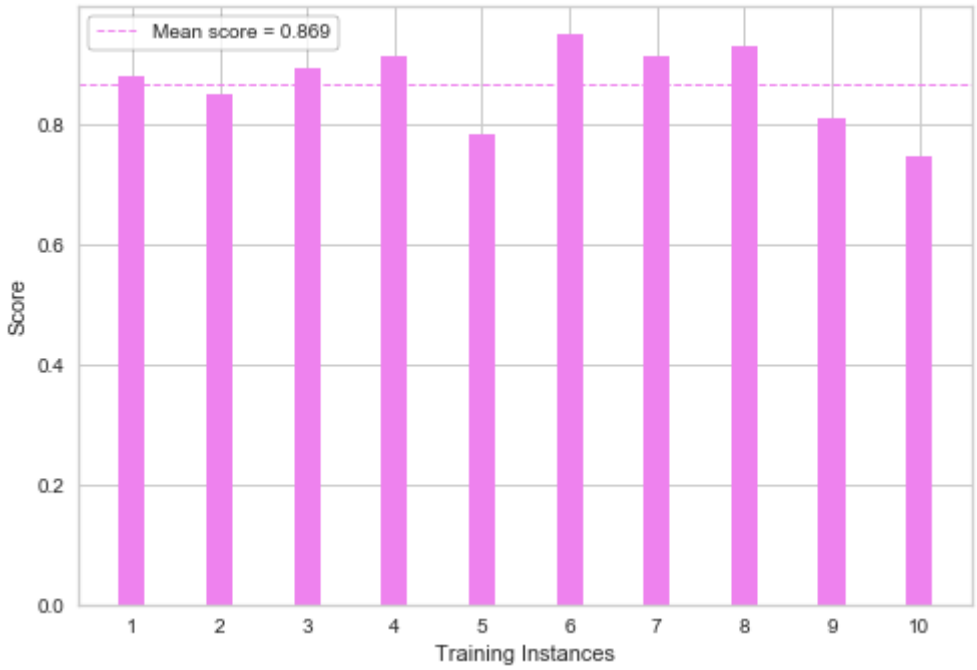

Gráfico de resultados 2. Métricas de F1 para cada partición de la validación cruzada en Logistic Regression (Morgan). Resultado medio de la F1 tras validación cruzada.

La curva ROC, así como la matriz de confusión, que da idea de lo mejor o peor clasificador que es un modelo, también es un resultado de una sola de las

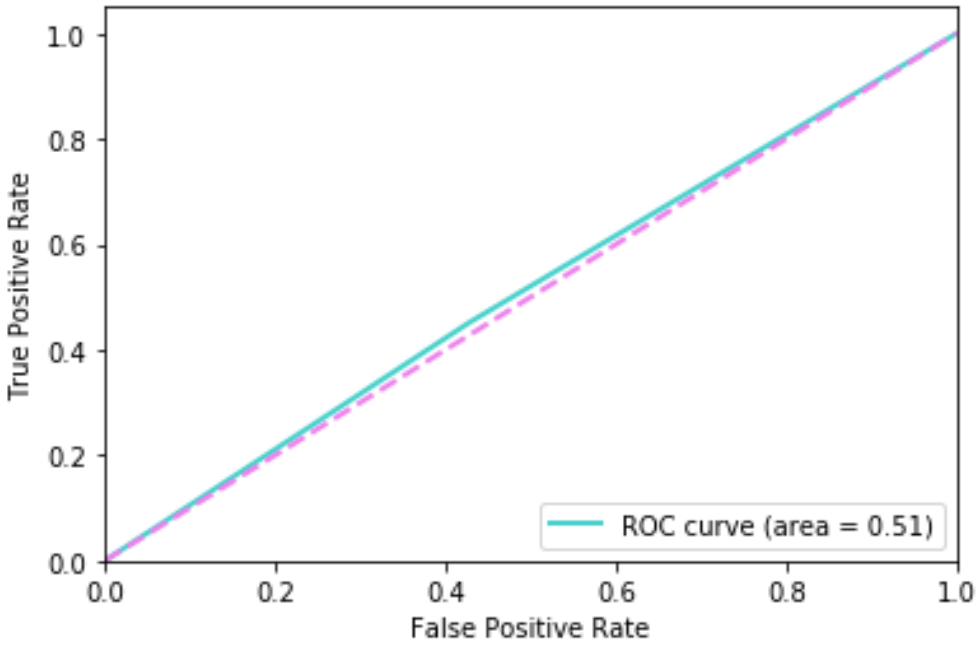

Gráfico de resultados 3. Curva ROC en Logistic Regression (Morgan) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.51

posibles particiones del conjunto de datos en subconjunto de entrenamiento y test. Ayuda para realizar un juicio sobre el modelo, pero no se debe considerar como una métrica determinante para ninguno de los modelos a continuación expuestos.

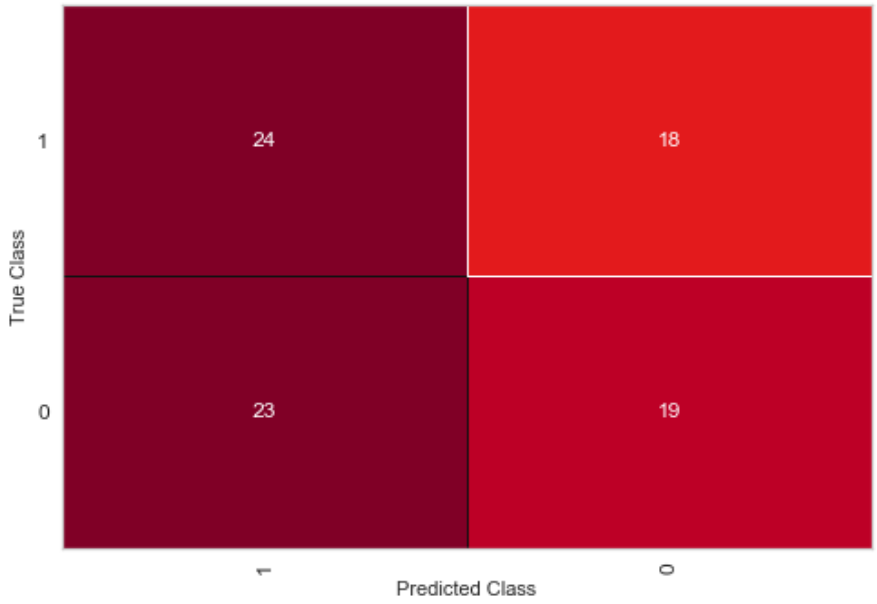

Gráfico de resultados 4. Matriz de confusión en Logistic Regression (Morgan) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

kNN

Los hiperparámetros más importantes a la hora de configurar el modelo son:

- ***n_neighbors = 5***: número de vecinos adyacentes de los que se consideran las etiquetas para clasificar la nueva observación.
- ***weights = 'uniform'***: no añade ninguna corrección en los pesos de las diferentes clases.

Con esto, se obtienen los siguientes resultados:

Sensibilidad: 0.5714

Especificidad: 0.4524

Exactitud (*accuracy*) con 10-fold-CV: 0.8777

F1 con 10-fold-CV: 0.933

De este modelo de clasificación, para este conjunto de datos, se puede decir, también, que es un buen modelo, bastante exacto, con resultados consistentes, y una sensibilidad aceptable. La baja sensibilidad es fruto de una partición entrenamiento – test, y no global, de todas las posibles.

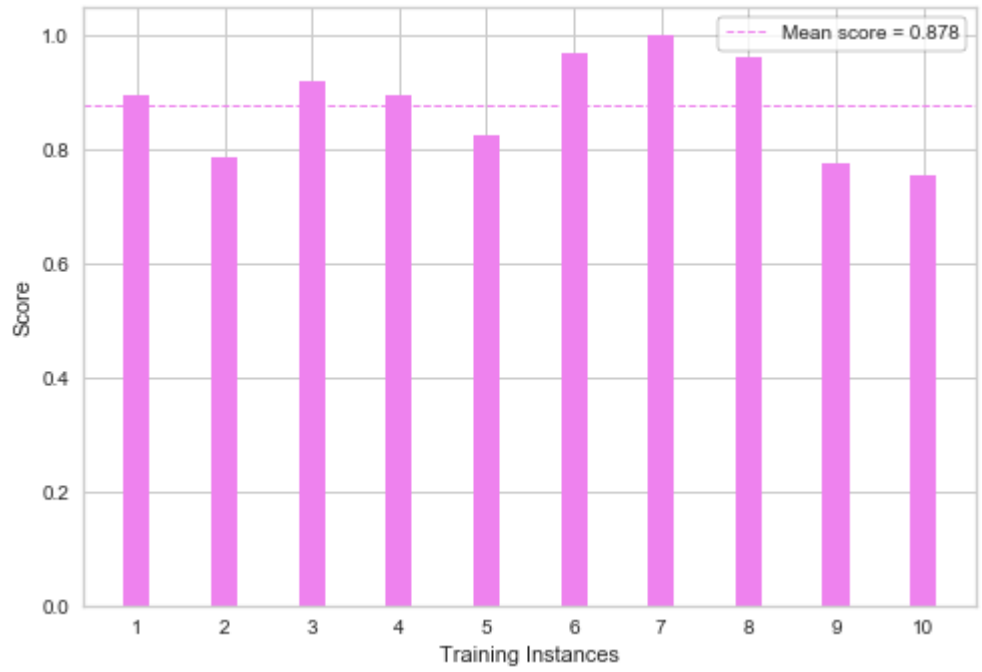

Gráfico de resultados 5. Métricas de exactitud para cada partición de la validación cruzada en kNN (Morgan). Resultado medio de la exactitud tras validación cruzada.

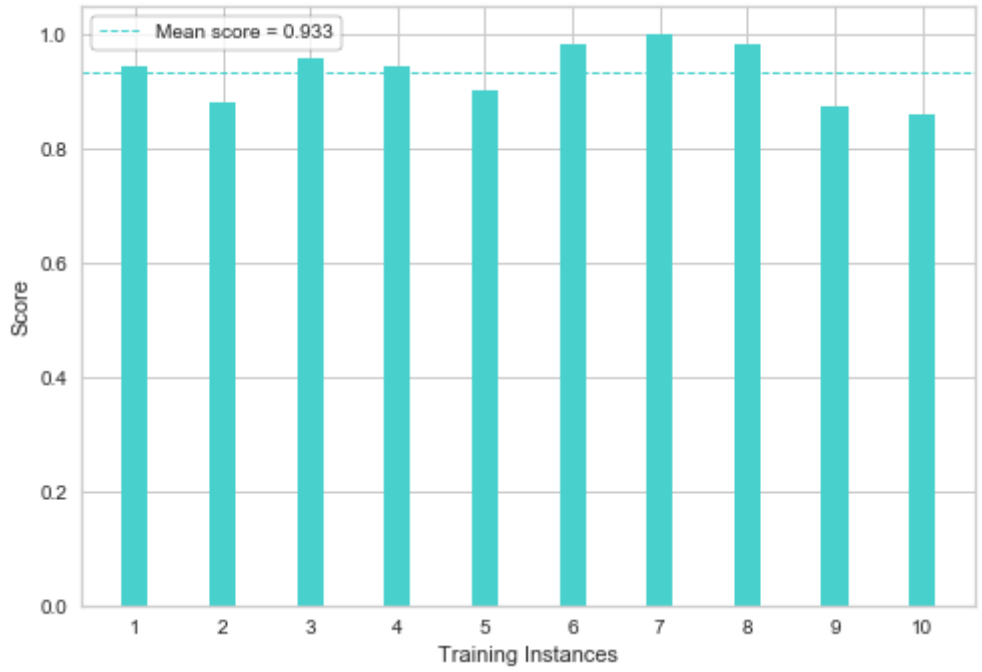

Gráfico de resultados 6. Métricas de F1 para cada partición de la validación cruzada en kNN (Morgan). Resultado medio de la F1 tras validación cruzada.

Aquí, la curva ROC y el área bajo esta, arroja un equilibrio mejor entre los verdaderos positivos y los falsos positivos, pero igualmente, sigue siendo un resultado apenas aceptable.

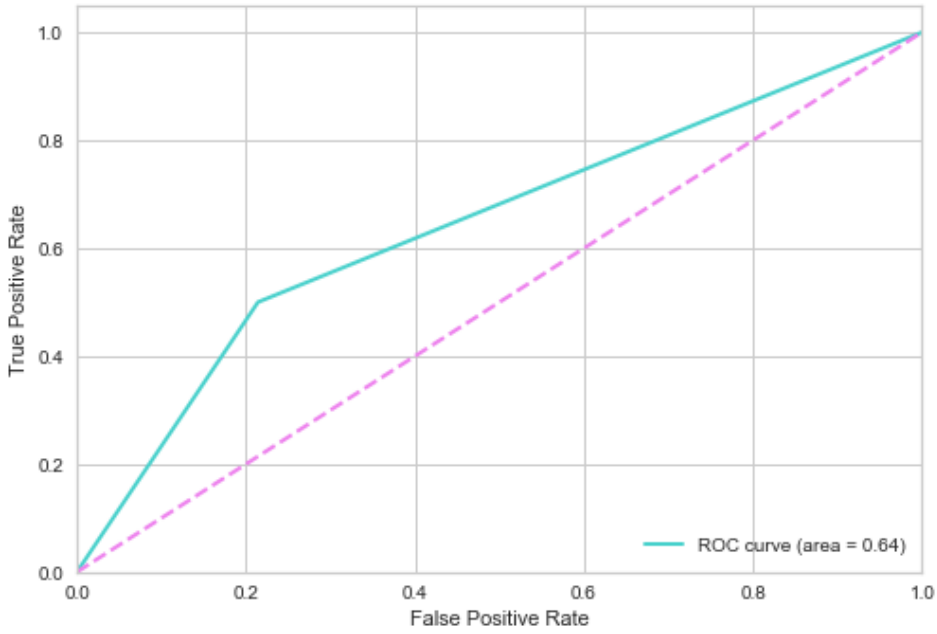

Gráfico de resultados 7. Curva ROC en kNN (Morgan) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.64

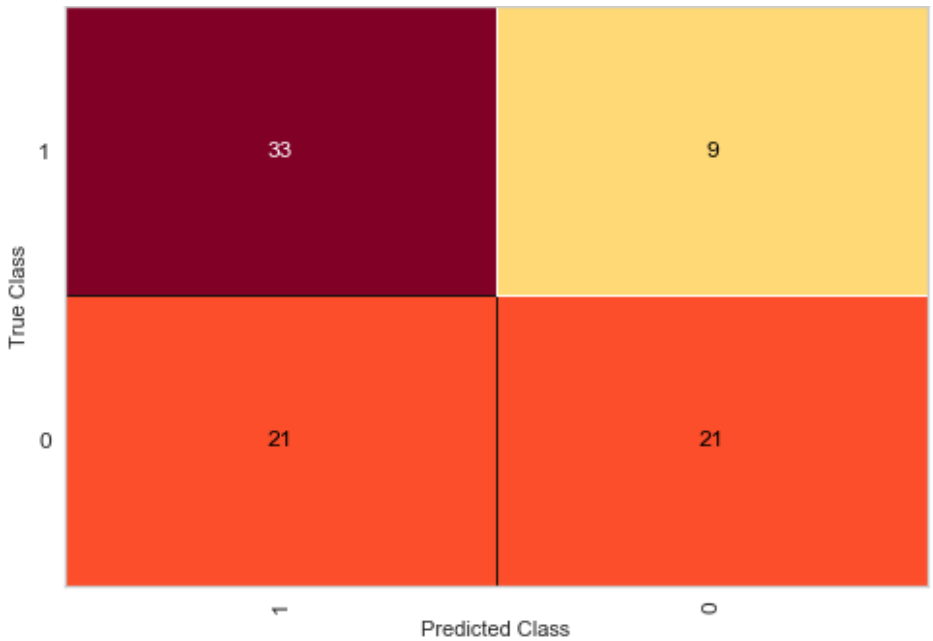

Gráfico de resultados 8. Matriz de confusión en kNN (Morgan) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

Naive Bayes

No se modifican los hiperparámetros por defecto que ofrece la librería *scikit-learn* por defecto a la hora de configurar el modelo. El tipo de Naive Bayes utilizado es el multinomial.

Con esto, se obtienen los siguientes resultados:

Sensibilidad: 0.5714

Especificidad: 0.6190

Exactitud (*accuracy*) con 10-fold-CV: 0.7233

F1 con 10-fold-CV: 0.809

De este modelo de clasificación, para este conjunto de datos, se puede decir, que es un modelo aceptable, pero con una exactitud y precisión notablemente menor que los dos modelos presentados anteriormente, de forma global. La métrica F1 es buena, aunque cabe destacar que, de todos los modelos, es el que menos sensible y preciso es, comparándolo incluso con los modelos cuyos fingerprints son de tipo MACCS, como se verá a partir de *4.1.2 Fingerprints tipo MACCS.*

Aquí, la curva ROC y el área bajo esta, arroja un equilibrio pobre entre los verdaderos positivos y los falsos positivos y es un resultado apenas aceptable.

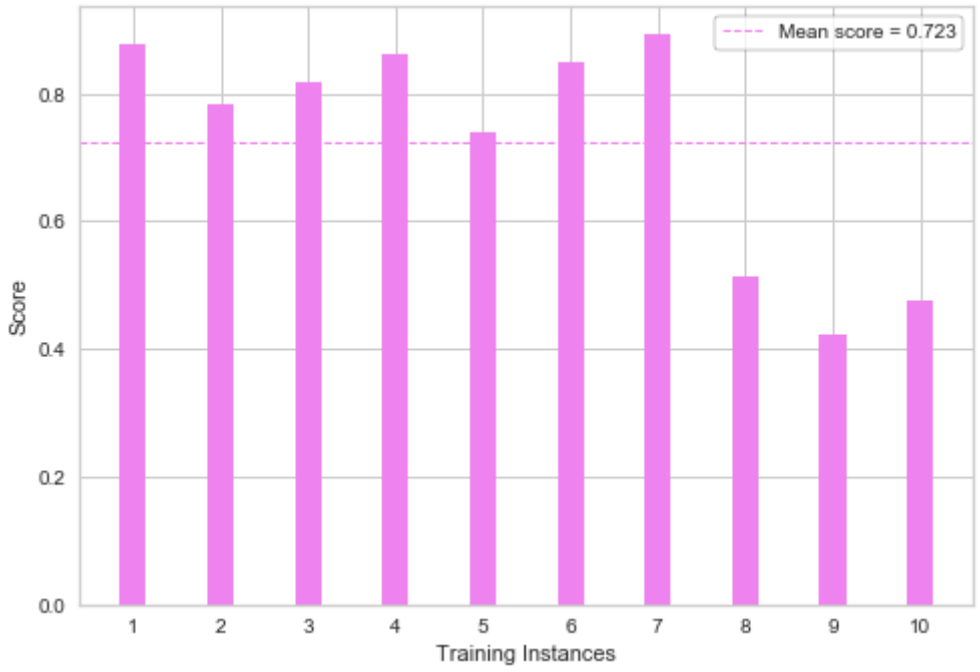

Gráfico de resultados 9. Métricas de exactitud para cada partición de la validación cruzada en Naive Bayes (Morgan). Resultado medio de la exactitud tras validación cruzada.

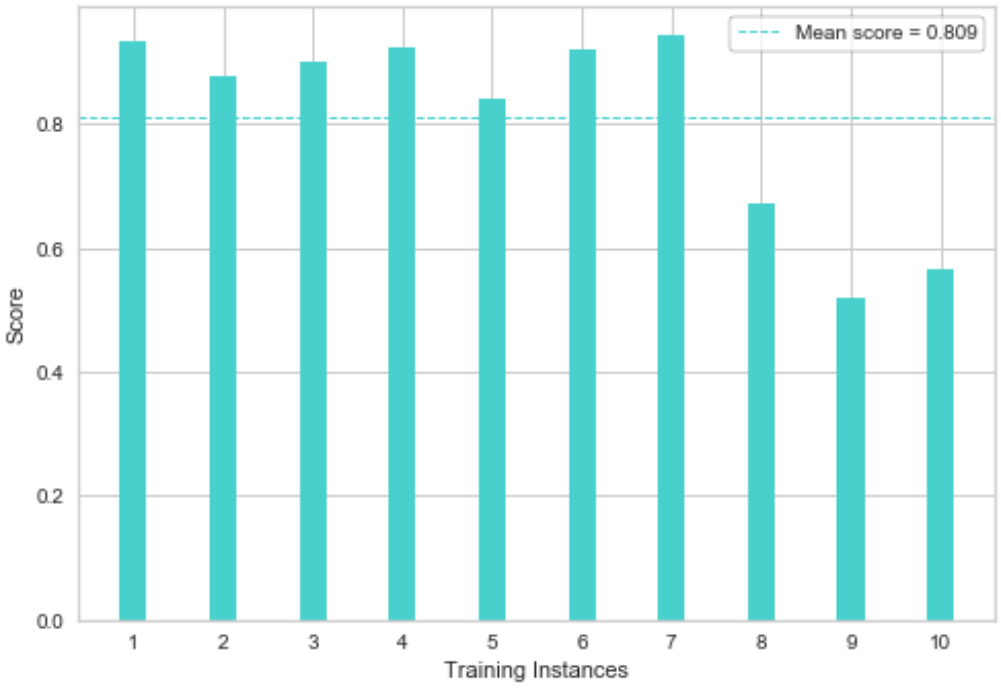

Gráfico de resultados 11. Métricas de F1 para cada partición de la validación cruzada en Naive Bayes (Morgan). Resultado medio de la F1 tras validación cruzada.

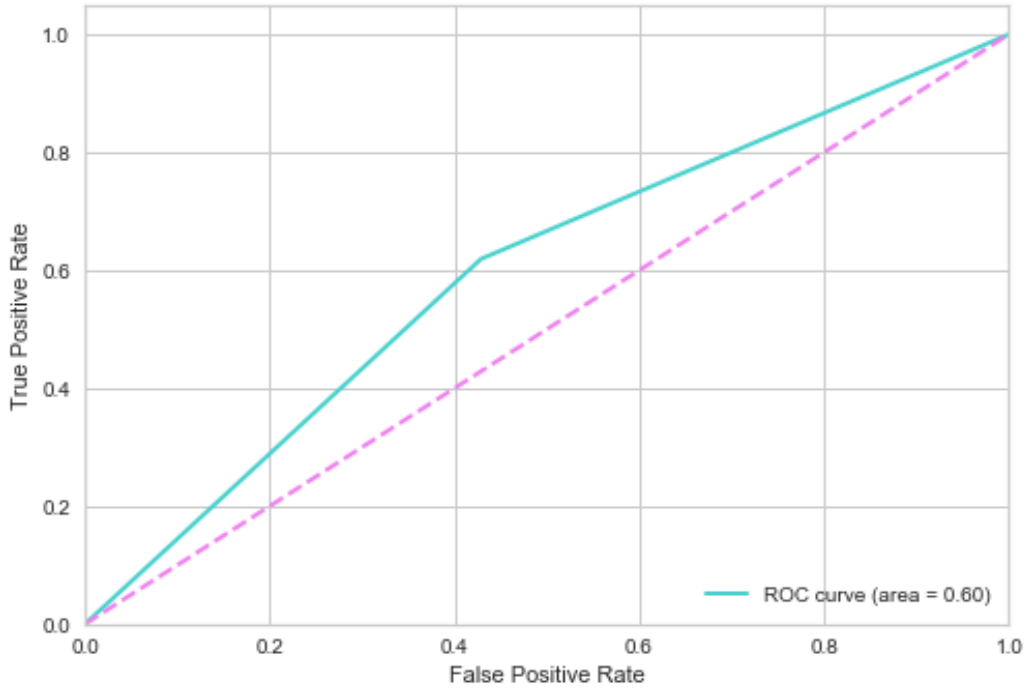

Gráfico de resultados 10. Curva ROC en Naive Bayes (Morgan) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.60

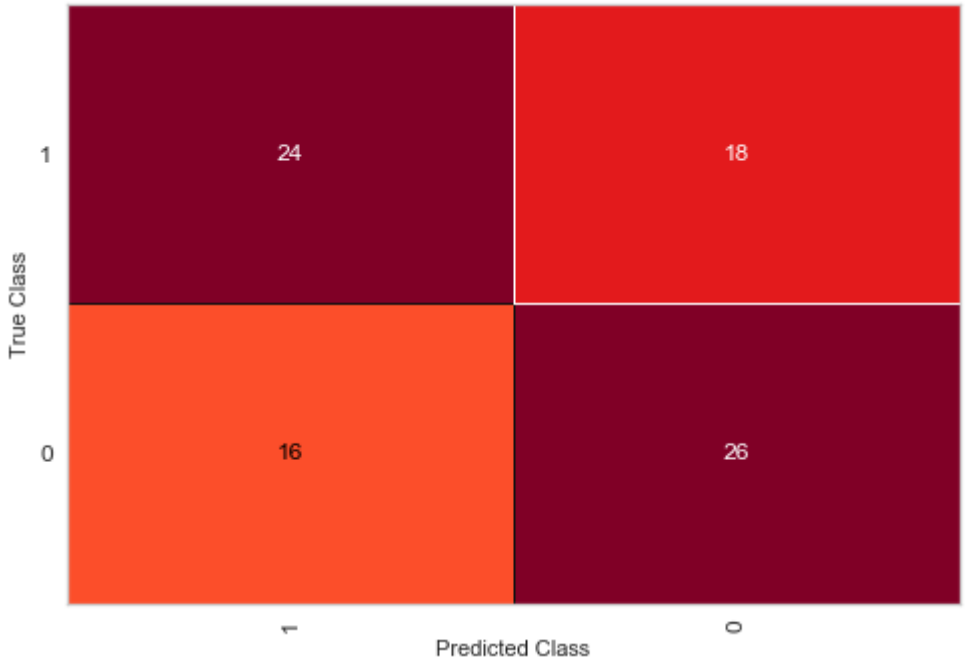

Gráfico de resultados 12. Matriz de confusión en Naive Bayes (Morgan) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

Gradient Boosting

Los hiperparámetros más importantes a la hora de configurar el modelo son:

- ***loss = exponential***: configura el modelo para que opere como se explica en el punto *3.4 Desarrollo de modelos de machine learning. Clasificación.*; de otra forma, por defecto, aplicaría una *Logistic regression.*
- ***learning_rate:*** medida de la aportación de cada árbol al modelo final; está ligado a n_estimators. Cuanto mayor learning_rate, menor la contribución de cada árbol al ajuste del modelo final.
- ***n_estimators = 100***: número de árboles con los que conformar el modelo.
- ***random_state***: generador de números aleatorios, que decidirá como se divide el set de datos en subconjunto de entrenamiento y testado, si fuera necesario, para asegurar que se realiza de la misma forma tras varias ejecuciones de una misma parte de código. Necesario para la validación de cualquier modelo.

Con esto, se obtienen los siguientes resultados:

Ajuste con diferentes valores para el hiperparámetro learning_rate.

***learning_rate* =0.1**
Exactitud (entrenamiento): 0.926
Exactitud (test): 0.548

***learning_rate* =0.25**
Exactitud (entrenamiento): 0.958
Exactitud (test): **0.571**

***learning_rate* =0.5**
Exactitud (entrenamiento): 0.978
Exactitud (test): 0.512

***learning_rate* =0.75**
Exactitud (entrenamiento): 0.981
Exactitud (test): 0.548

Solo para el valor *learning_rate* = 0.25:
Sensibilidad: 0.6190

Especificidad: 0.5238

Exactitud (accuracy) con 10-fold-CV: 0.875

F1 con 10-fold-CV: 0.931

De este modelo de clasificación, para este conjunto de datos, se puede decir, también, que es un buen modelo, bastante exacto, con resultados consistentes, y una sensibilidad aceptable. Se obtienen resultados muy similares como con el tipo Logistic o kNN, pero como se ha explicado anteriormente en el punto *3.4 Desarrollo de modelos de machine learning. Clasificación.*, los fundamentos teóricos sobre los que se asienta son notablemente diferentes.

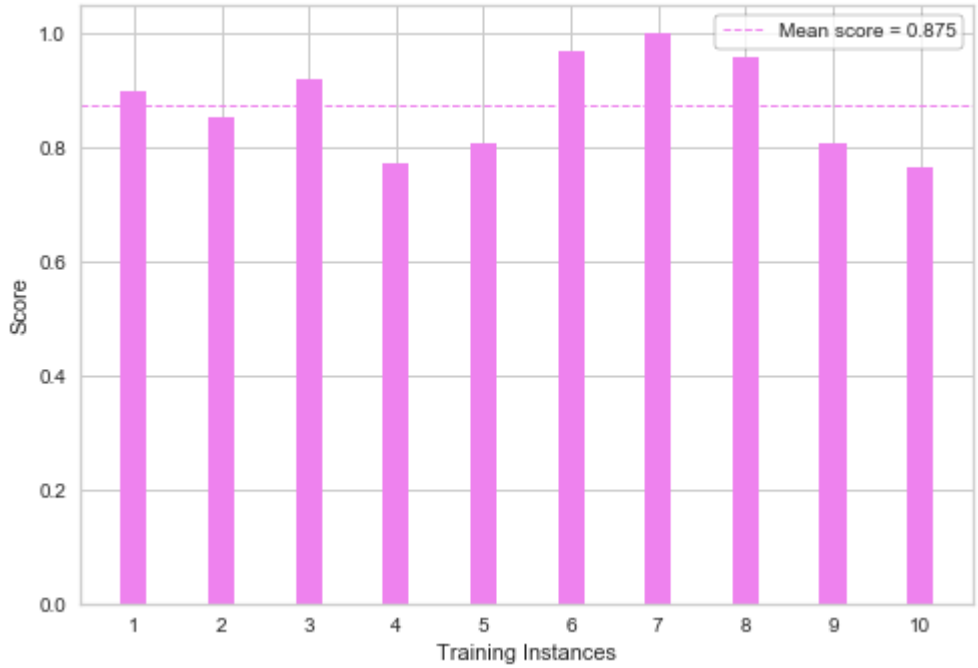

Gráfico de resultados 13. Métricas de exactitud para cada partición de la validación cruzada en Gradient Boosting (Morgan). Resultado medio de la exactitud tras validación cruzada.

Aquí, la curva ROC y el área bajo esta, arroja un equilibrio pobre entre los verdaderos positivos y los falsos positivos y es un resultado apenas aceptable.

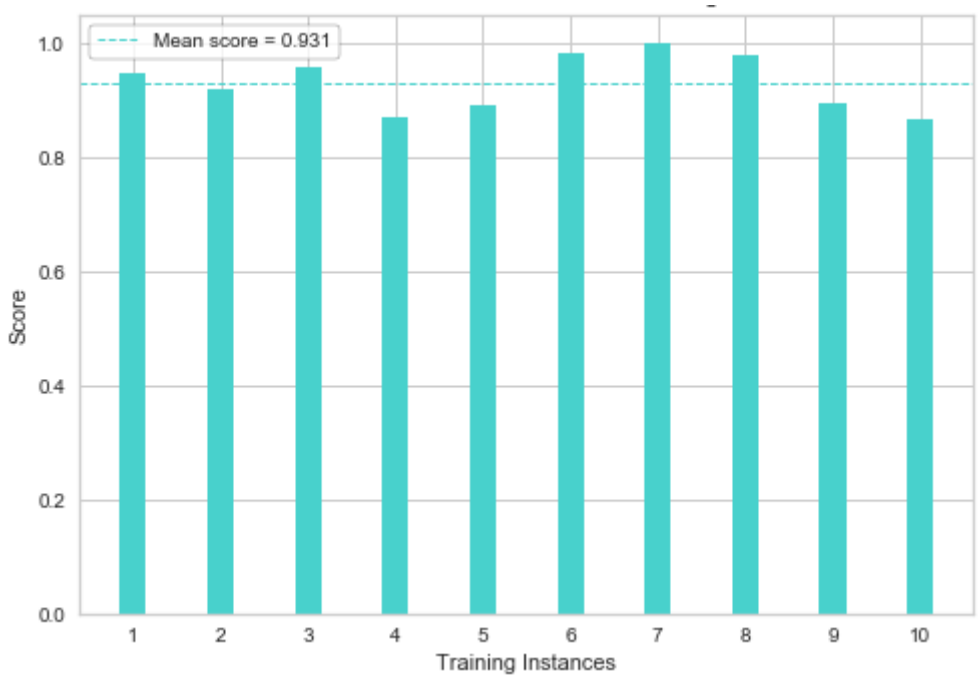

Gráfico de resultados 14. Métricas de F1 para cada partición de la validación cruzada en Gradient Boosting (Morgan). Resultado medio de la F1 tras validación cruzada.

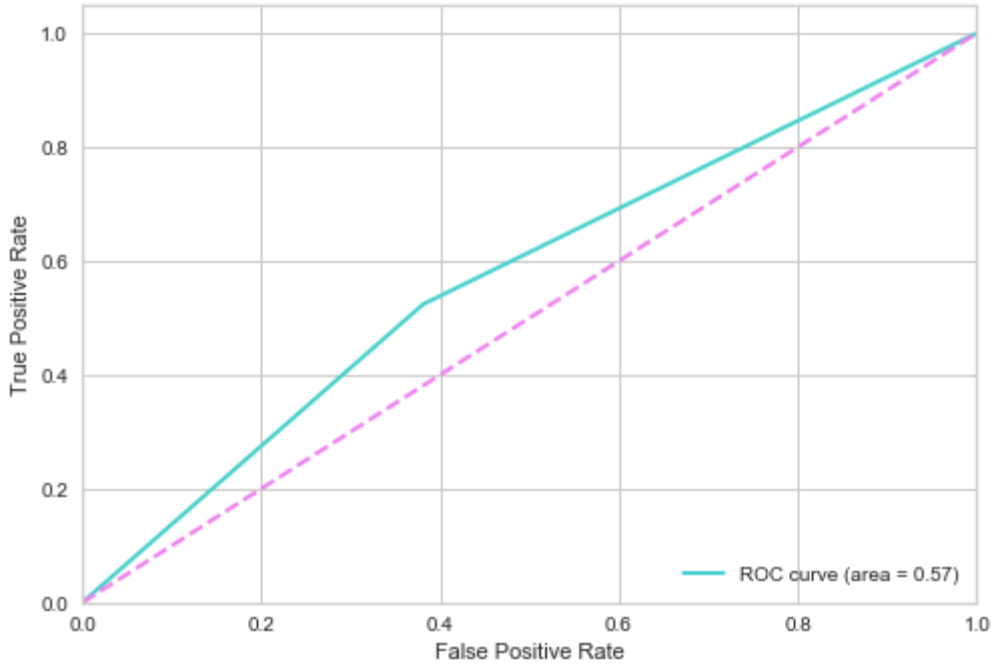

Gráfico de resultados 15. Curva ROC en Gradient Boosting (Morgan) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.57

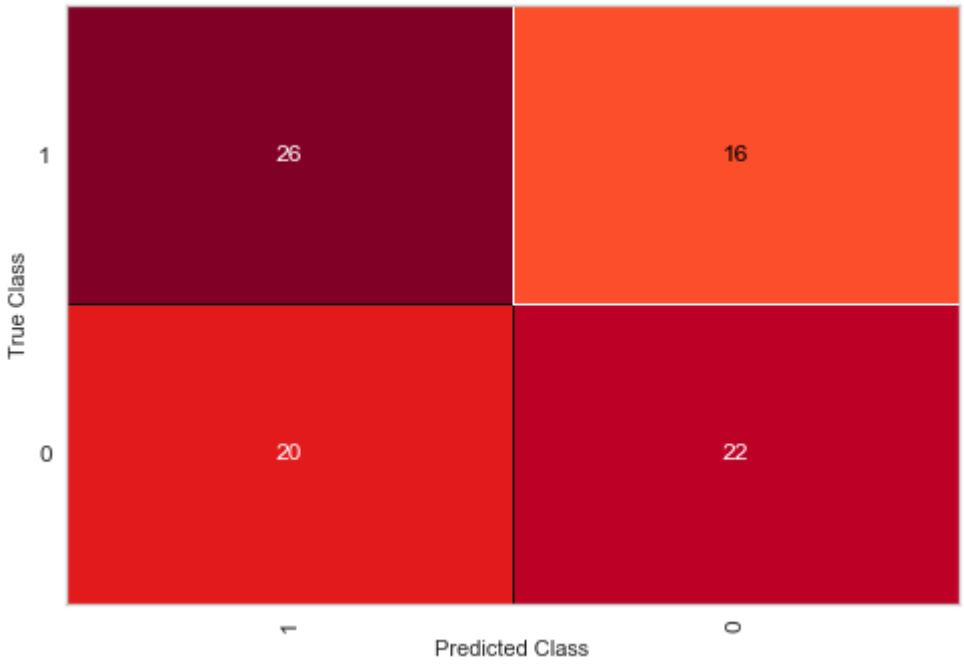

Gráfico de resultados 16. Matriz de confusión en Gradient Boosting (Morgan) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

SVM

Los hiperparámetros más importantes a la hora de configurar el modelo son:

- ***C***: parámetro de penalización para aquellas observaciones mal clasificadas. Por defecto C = 1.0
- ***kernel***: función matemática que se usa para dibujar el hiperplano de separación entre las diferentes clases. Aquí, es 'rbf' (función radial).
- ***random_state***: generador de números aleatorios, que decidirá como se divide el set de datos en subconjunto de entrenamiento y testado, si fuera necesario, para asegurar que se realiza de la misma forma tras varias ejecuciones de una misma parte de código. Necesario para la validación de cualquier modelo.

Con esto, se obtienen los siguientes resultados:

Para C = 1.0

Sensibilidad: 0.3333

Especificidad: 0.7143

Exactitud (accuracy) con 10-fold-CV: 0.8943

F1 con 10-fold-CV: 0.943

Para C = 10.0

Exactitud (accuracy) con 10-fold-CV: 0.8943

F1 con 10-fold-CV: 0.943

Para C = 1.0, kernel = 'poly', degree = 3:

Exactitud (accuracy) con 10-fold-CV: 0.8943

F1 con 10-fold-CV: 0.943

Para C = 1.0, kernel = 'poly', degree = 4:

Exactitud (accuracy) con 10-fold-CV: 0.8943

F1 con 10-fold-CV: 0.943

De este modelo de clasificación, para este conjunto de datos, se puede decir, también, que es un buen modelo, bastante exacto, con resultados consistentes, y una sensibilidad aceptable. Si bien parece que su sensibilidad es pobre, su especificidad es mejor (evitando falsos positivos). En cualquier caso, lo mejor y más destacable vuelven a ser los resultados de exactitud y F1 tras la validación cruzada, que dan idea del buen modelo que es, lo que hace pensar que esa baja sensibilidad sea fruto de una partición entrenamiento – test, y no global, de todas las posibles.

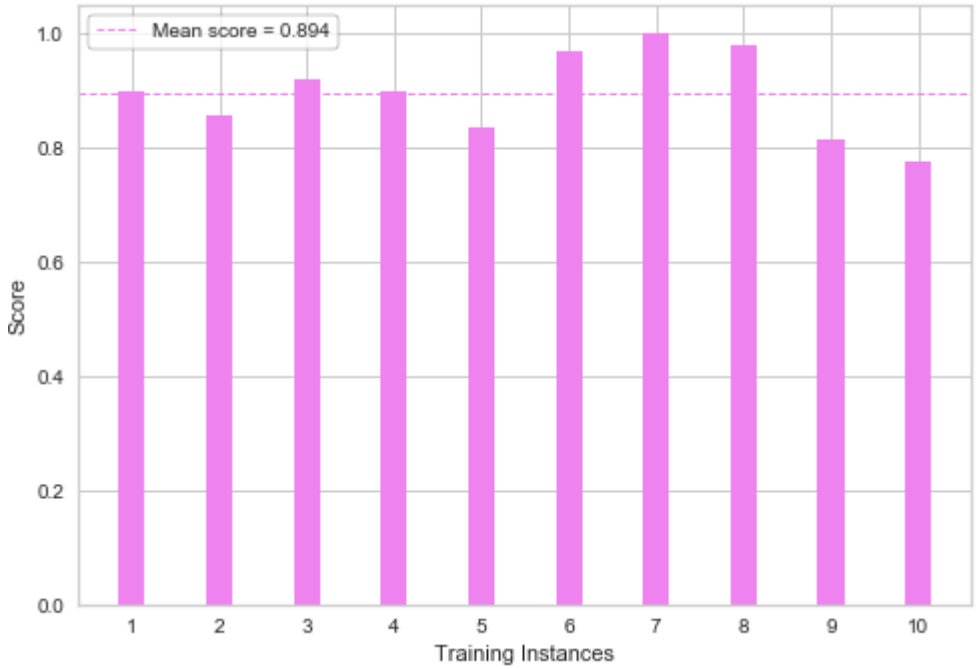

Gráfico de resultados 17. Métricas de exactitud para cada partición de la validación cruzada en SVC (Morgan). Resultado medio de la exactitud tras validación cruzada.

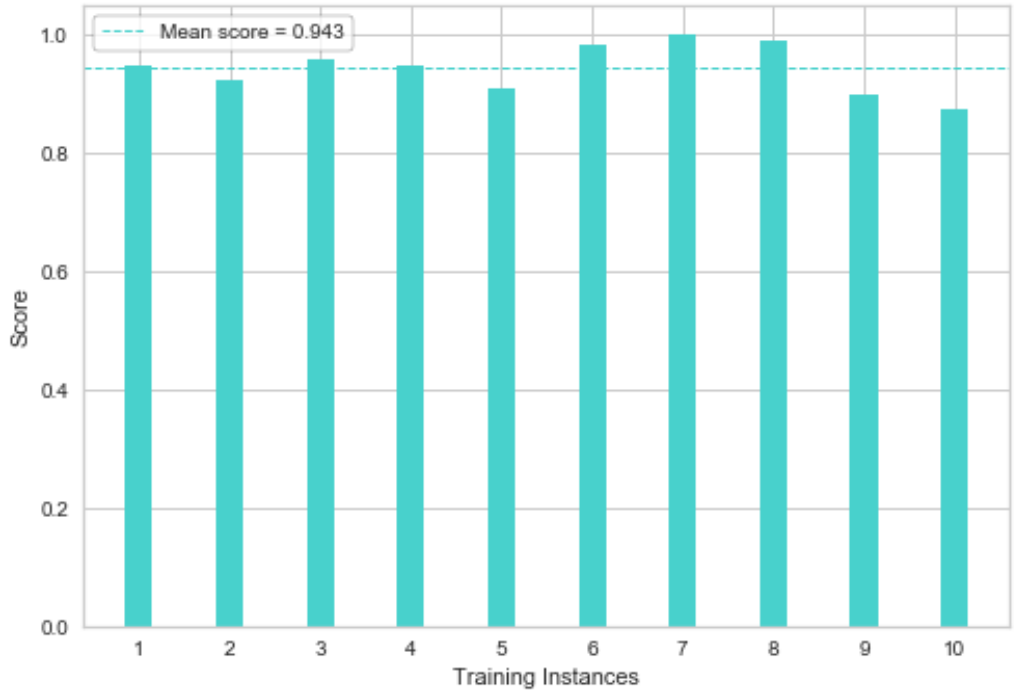

Gráfico de resultados 20. Métricas de F1 para cada partición de la validación cruzada en SVC (Morgan). Resultado medio de la F1 tras validación cruzada.

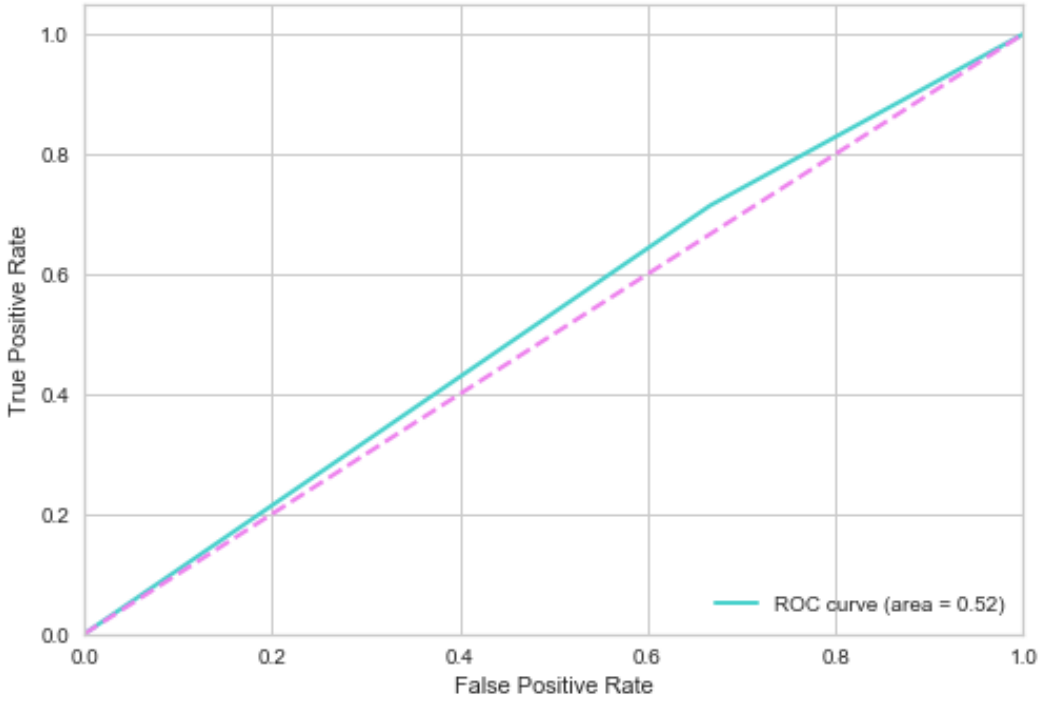

Gráfico de resultados 18. Gráfico de resultados 19. Curva ROC en SVC (Morgan) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.52

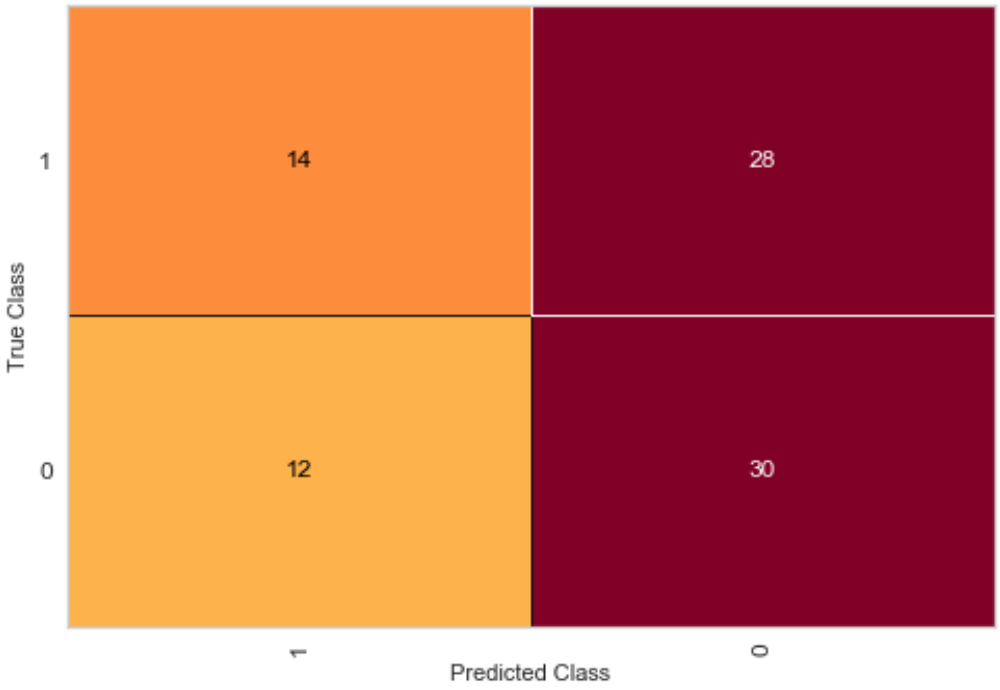

Gráfico de resultados 21. Matriz de confusión en SVC (Morgan) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

Random Forest

Los hiperparámetros más importantes a la hora de configurar el modelo son:

- ***n_estimators = 50***: número de árboles con los que conformar el modelo, para cada uno de los bosques que se conformen.
- ***random_state***: generador de números aleatorios, que decidirá como se divide el set de datos en subconjunto de entrenamiento y testado, si fuera necesario, para asegurar que se realiza de la misma forma tras varias ejecuciones de una misma parte de código. Necesario para la validación de cualquier modelo.

Con esto, se obtienen los siguientes resultados:

Para n = 50

Sensibilidad: 0.6429

Especificidad: 0.5238

Exactitud (accuracy) con 10-fold-CV: 0.889

F1 con 10-fold-CV: 0.940

Para n = 100

Sensibilidad: 0.6429

Especificidad: 0.5476

Exactitud (accuracy) con 10-fold-CV: 0.886

F1 con 10-fold-CV: 0.938

De este modelo de clasificación, para este conjunto de datos, se puede decir, también, que es un buen modelo, bastante exacto, con resultados consistentes, y una sensibilidad aceptable. Su fundamento teórico tiene cosas en común con el modelo de Gradient Boosting, y vemos que, de hecho, rinde una exactitud y F1 bastante similares tras la validación cruzada.

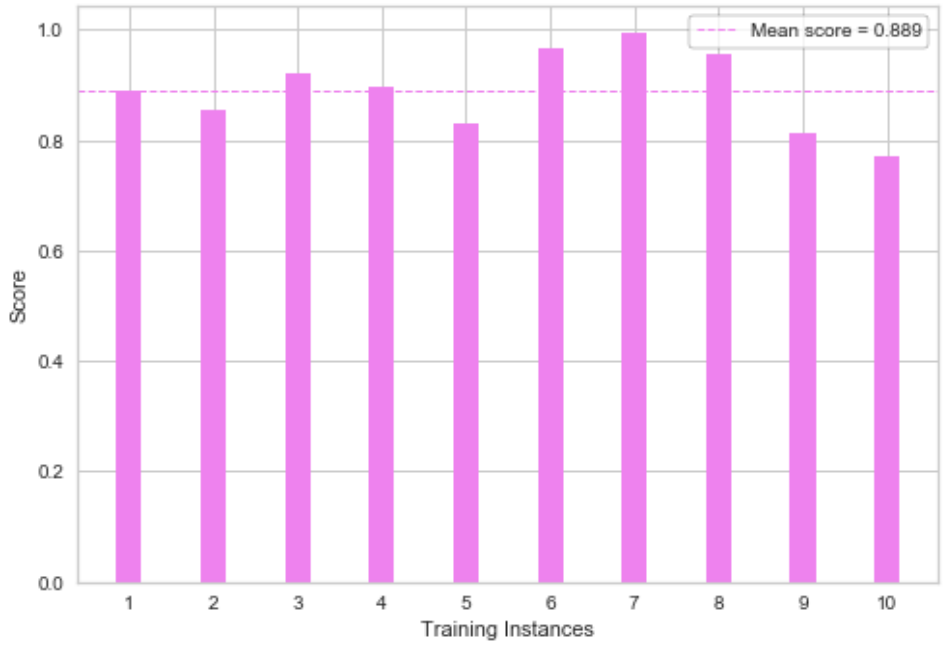

Gráfico de resultados 22. Métricas de exactitud para cada partición de la validación cruzada en Random Forest (Morgan). Resultado medio de la exactitud tras validación cruzada.

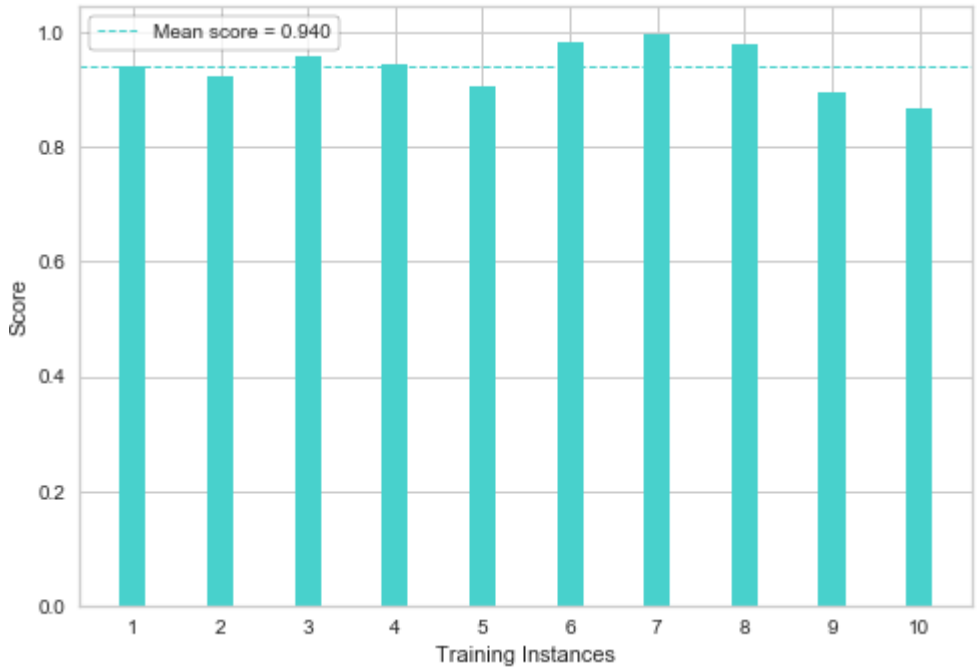

Gráfico de resultados 23. Métricas de F1 para cada partición de la validación cruzada en Random Forest (Morgan). Resultado medio de la F1 tras validación cruzada.

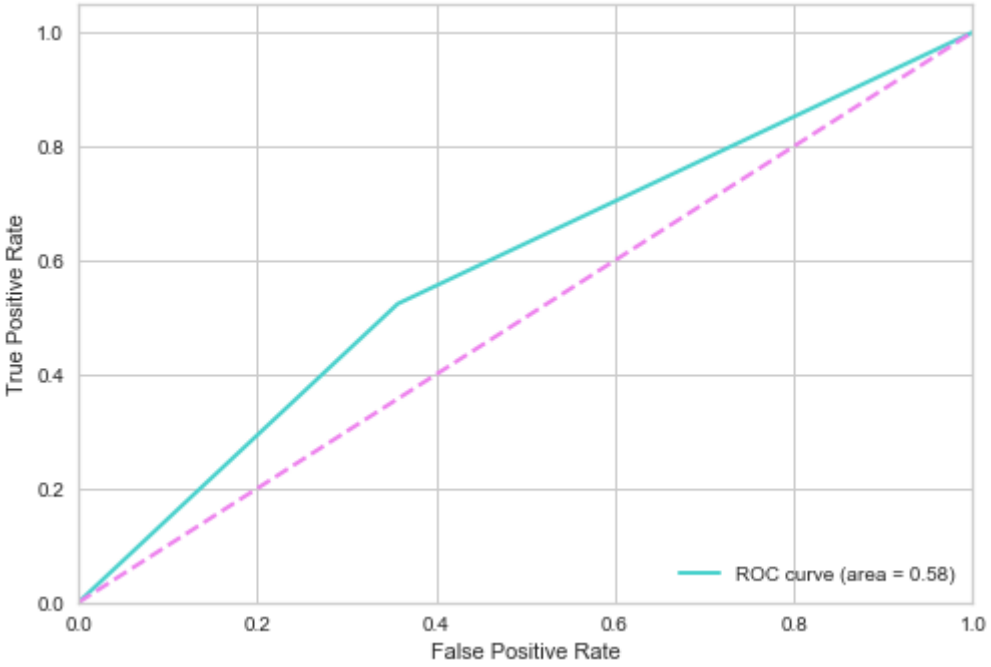

Gráfico de resultados 24. Curva ROC en Random Forest (Morgan) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.58

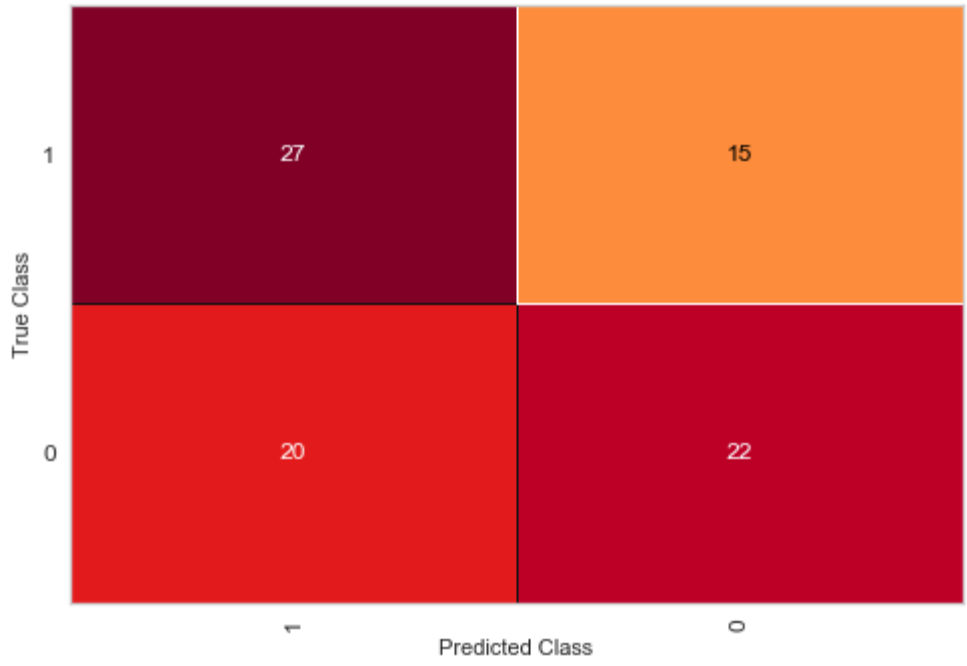

Gráfico de resultados 25. Matriz de confusión en Random Forest (Morgan) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

	Exactitud	F1 (precisión y sensibilidad)
Logistic regression	0.8686	0.927
kNN	0.8777	0.933
Naive Bayes	0.7233	0.809
SVM	**0.8943**	**0.943**
Gradient Boosting (100 árboles en total)	0.875	0.931
Random Forests (100 árboles por bosque)	0.889	0.940

Tabla 1. Métricas de exactitud y F1 para fingerprints tipo Morgan.

4.1.2 Fingerprints tipo MACCS.

Logistic regression

Con los hiperparámetros configurados igual que para los fingerprints de tipo Morgan, se obtienen los siguientes resultados:

Sensibilidad: 0.6190

Especificidad: 0.5238

Exactitud (*accuracy*) con 10-fold-CV: 0.8900

F1 con 10-fold-CV: 0.940

De este modelo de clasificación, para este conjunto de datos, se puede decir que es un buen modelo, bastante exacto, con resultados consistentes, y una sensibilidad aceptable. Fijando la atención en el F1, con validación cruzada, resulta tan buen modelo como su homólogo con fingerprints Morgan, siendo incluso unas décimas mejor.

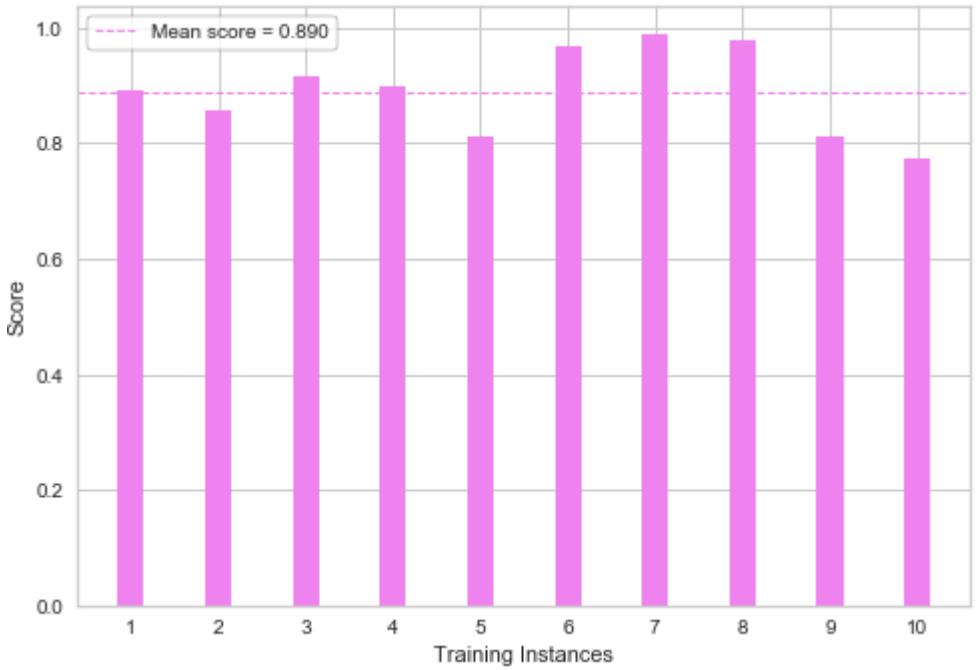

Gráfico de resultados 26. Métricas de exactitud para cada partición de la validación cruzada en Logistic Regression (MACCS). Resultado medio de la exactitud tras validación cruzada.

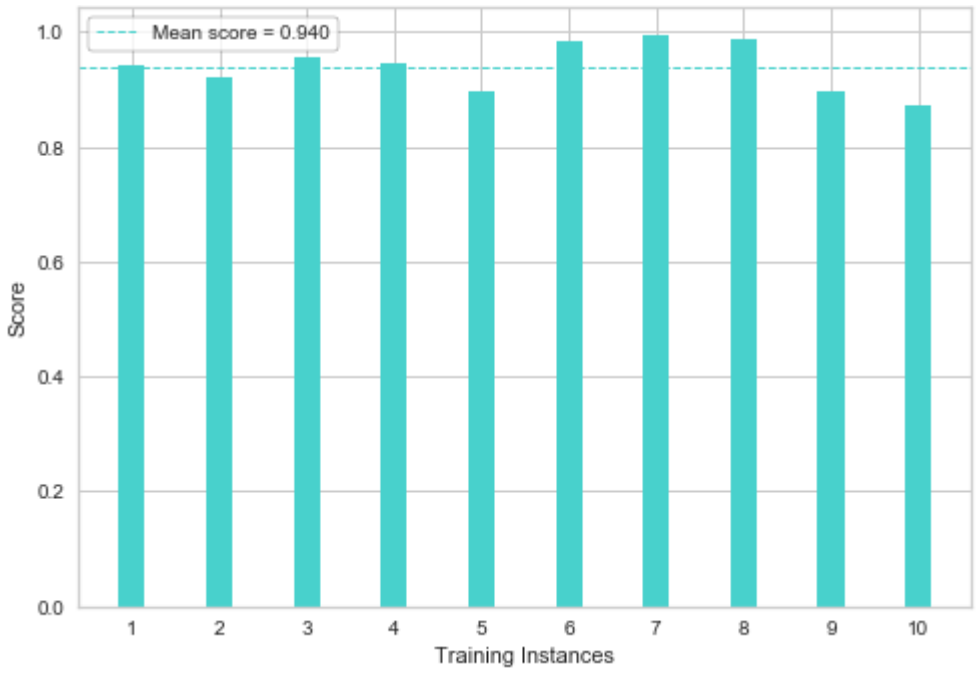

Gráfico de resultados 27. Métricas de F1 para cada partición de la validación cruzada en Logistic Regression (MACCS). Resultado medio de la F1 tras validación cruzada.

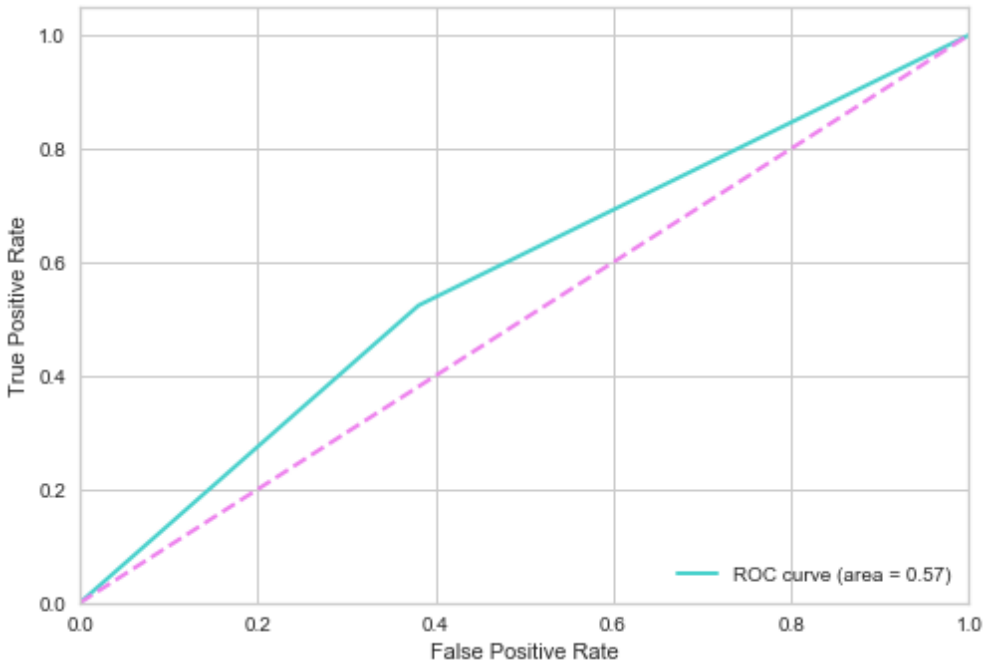

Gráfico de resultados 28. Curva ROC en Logistic Regression (MACCS) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.57

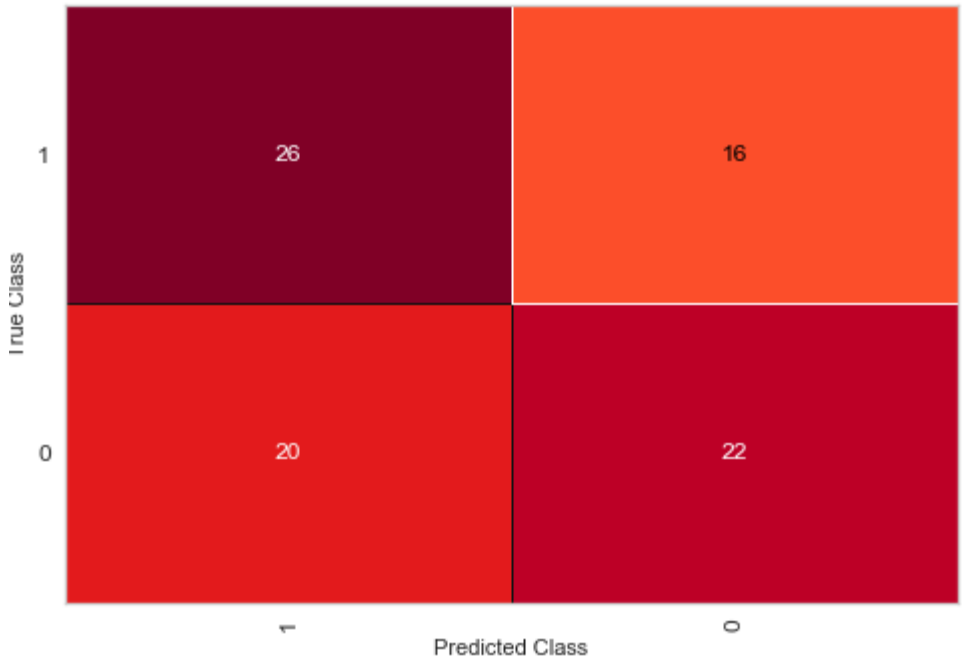

Gráfico de resultados 29. Matriz de confusión en Logistic Regression (MACCS) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

kNN

Con los hiperparámetros configurados igual que para los fingerprints de tipo Morgan, se obtienen los siguientes resultados:

Sensibilidad: 0.7619

Especificidad: 0.5476

Exactitud (*accuracy*) con 10-fold-CV: 0.8767

F1 con 10-fold-CV: 0.877

De este modelo de clasificación, para este conjunto de datos, se puede decir, que es un buen modelo, bastante exacto, con resultados consistentes, y una sensibilidad aceptable. A pesar de esto, de todos los modelos de clasificación probados con los fingerprints MACCS, es el que arroja una métrica F1 más baja de todos ellos.

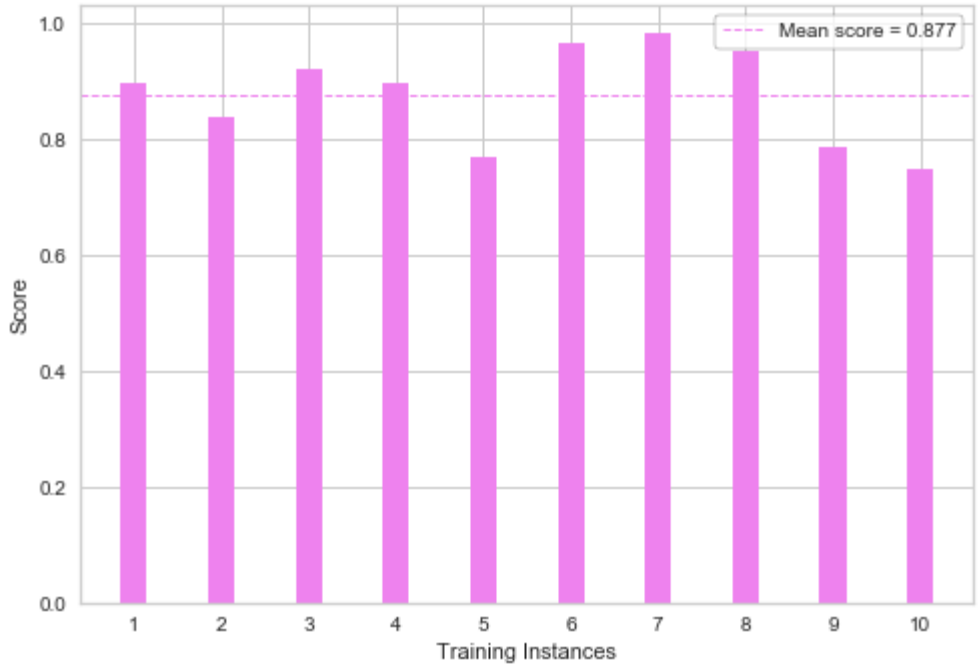

Gráfico de resultados 30. Métricas de exactitud para cada partición de la validación cruzada en kNN (MACCS). Resultado medio de la exactitud tras validación cruzada.

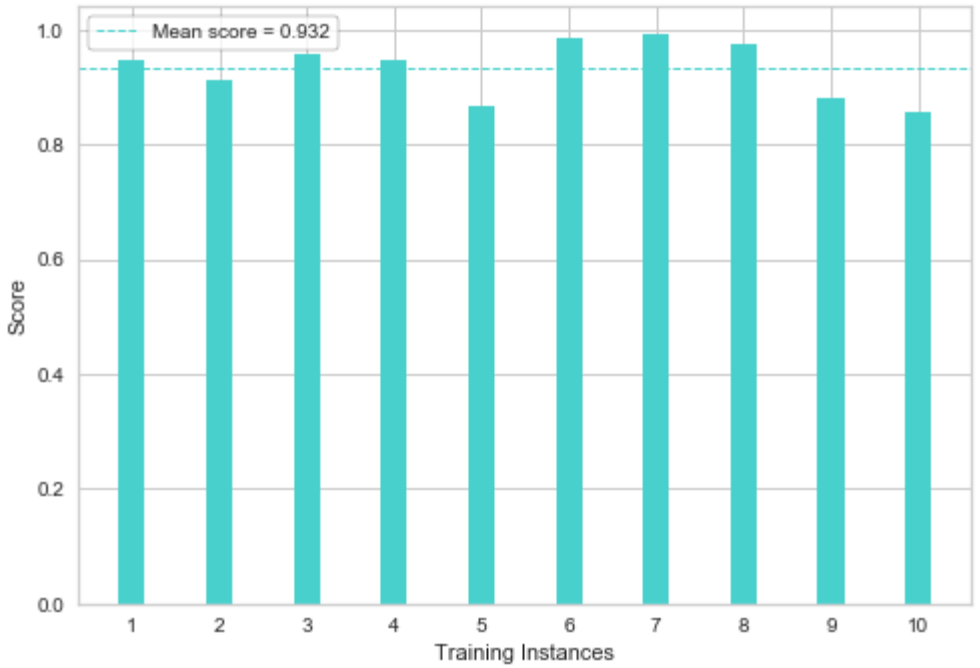

Gráfico de resultados 31. Métricas de F1 para cada partición de la validación cruzada en kNN (MACCS). Resultado medio de la F1 tras validación cruzada.

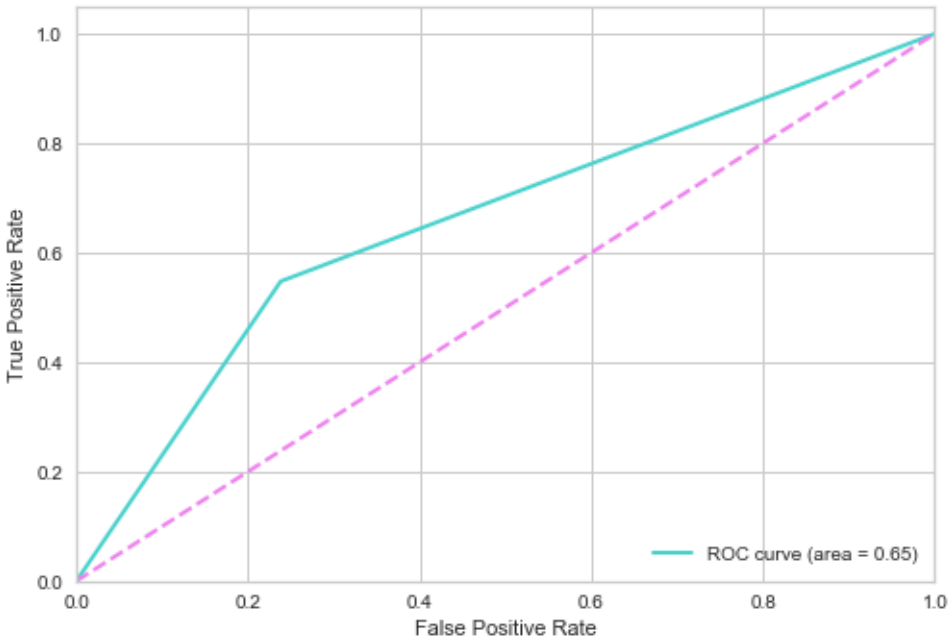

Gráfico de resultados 33. Curva ROC en kNN (MACCS) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.65

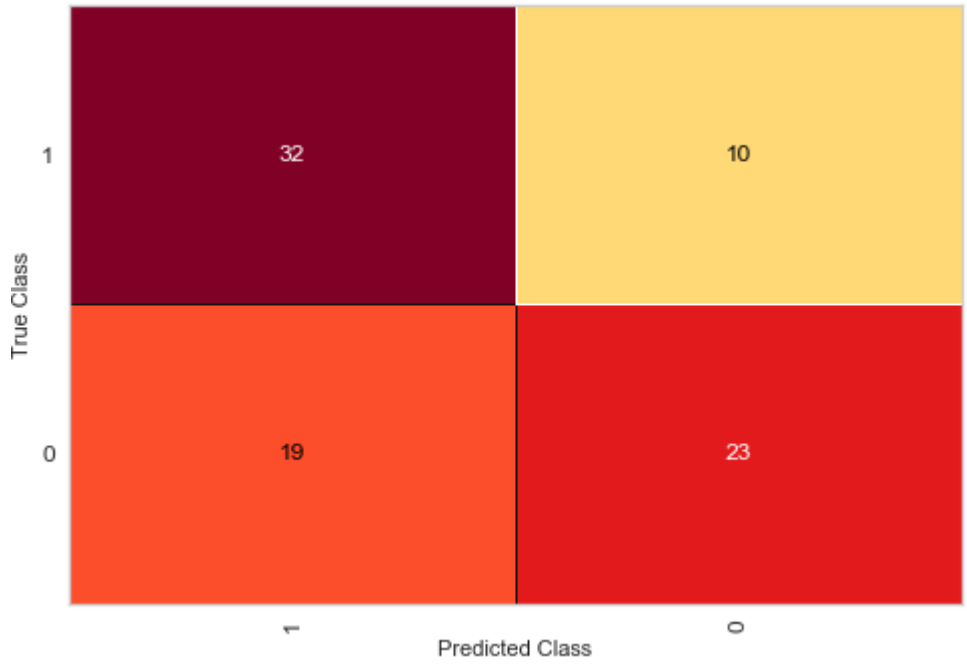

Gráfico de resultados 32. Matriz de confusión en kNN (MACCS) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

Naive Bayes

Con los hiperparámetros configurados igual que para los fingerprint de tipo Morgan, se obtienen los siguientes resultados:

Sensibilidad: 0.5714

Especificidad: 0.5

Exactitud (*accuracy*) con 10-fold-CV: 0.8349

F1 con 10-fold-CV: 0.904

De este modelo de clasificación, para este conjunto de datos, se puede decir, que es un modelo aceptable, con una exactitud y precisión similar al modelo Logistic, y mejores métricas que el modelo kNN.

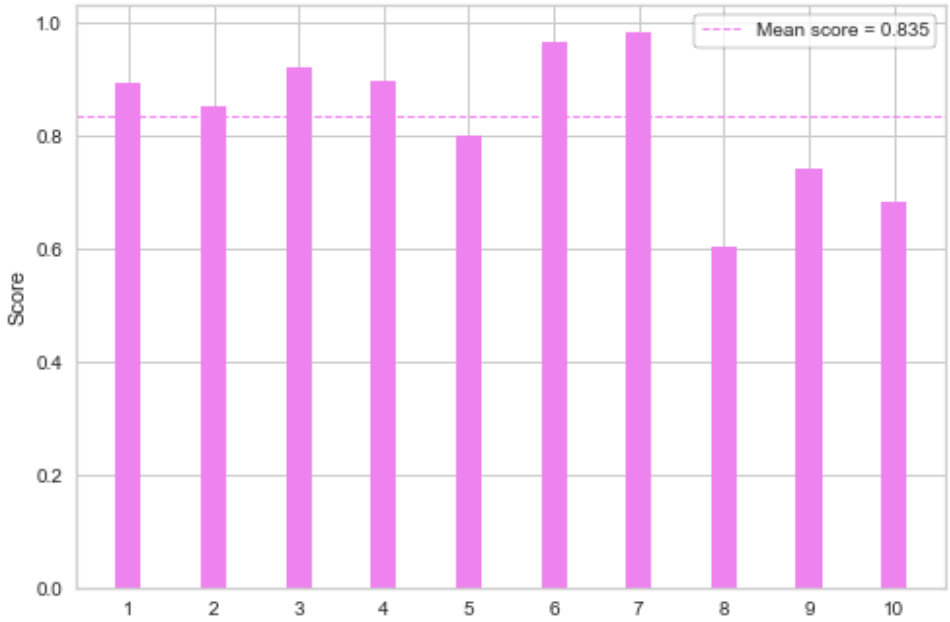

Gráfico de resultados 34. Métricas de exactitud para cada partición de la validación cruzada en Naive Bayes (MACCS). Resultado medio de la exactitud tras validación cruzada.

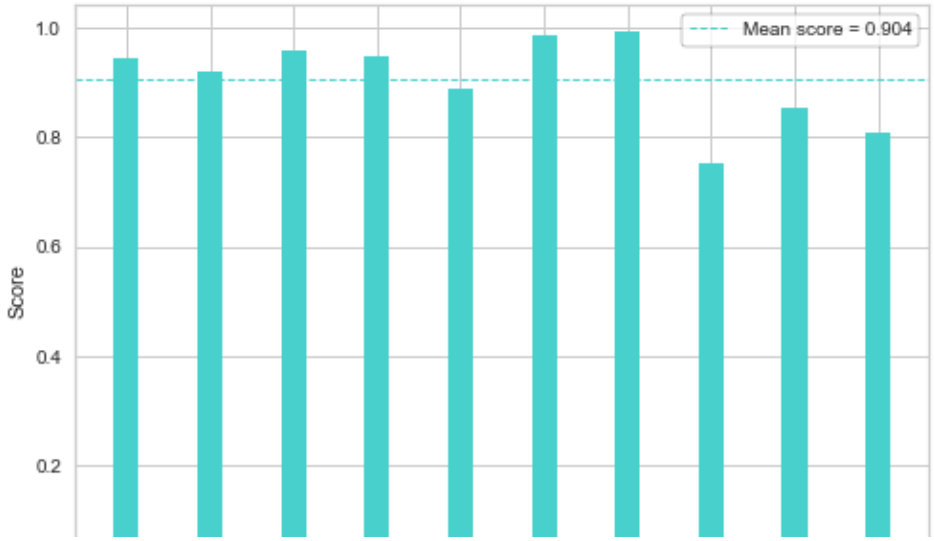

Gráfico de resultados 35. Métricas de F1 para cada partición de la validación cruzada en Naive Bayes (MACCS). Resultado medio de la F1 tras validación cruzada.

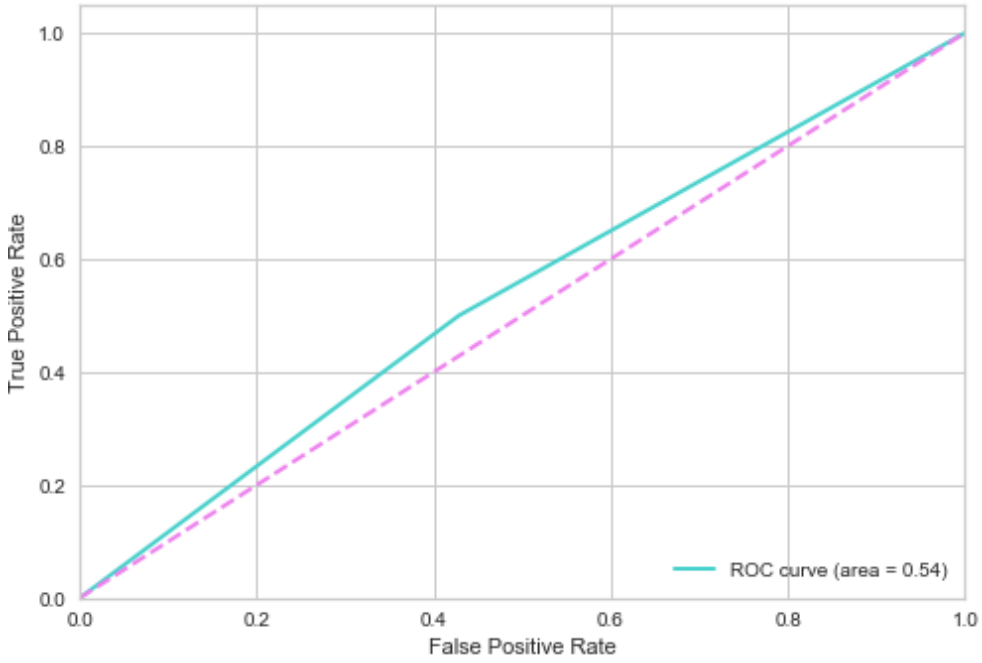

Gráfico de resultados 36. Curva ROC en kNN (MACCS) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.54

Gradient Boosting

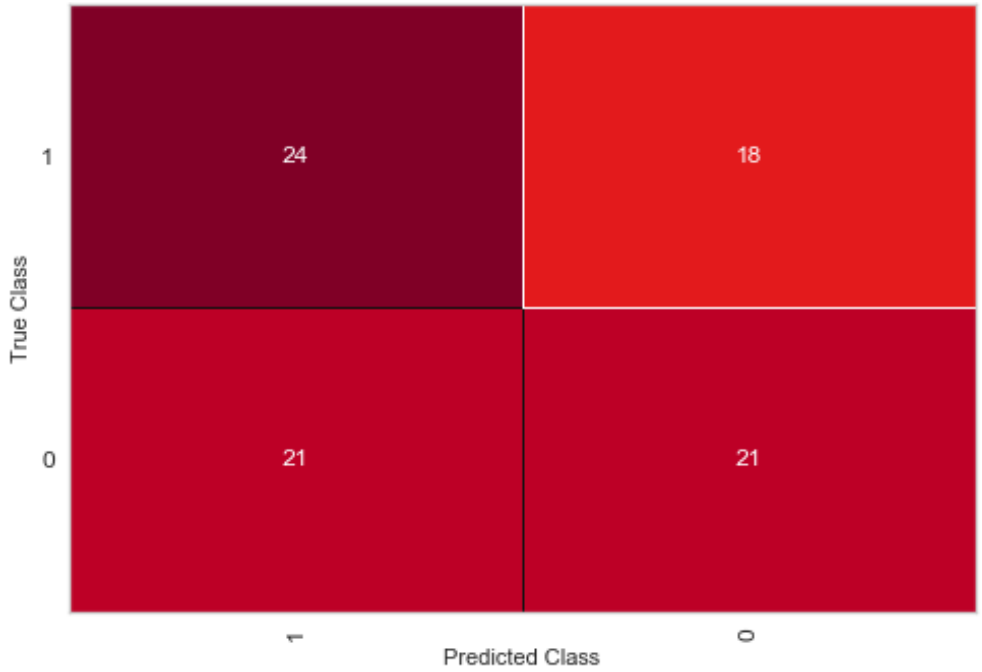

Gráfico de resultados 37. Matriz de confusión en Naive Bayes (MACCS) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

Con los hiperparámetros configurados igual que para los fingerprint de tipo Morgan, se obtienen los siguientes resultados:

Sensibilidad: 0.5952

Especificidad: 0.5476

Exactitud (*accuracy*) con 10-fold-CV: 0.8788

F1 con 10-fold-CV: 0.934

De este modelo de clasificación, para este conjunto de datos, se puede decir, también, que es un buen modelo, bastante exacto, con resultados consistentes, y una sensibilidad aceptable. Se obtienen resultados muy similares como con el tipo Logistic o Naive Bayes, pero como se ha explicado anteriormente en el punto *3.4 Desarrollo de modelos de machine learning. Clasificación*., los fundamentos teóricos sobre los que se asienta son notablemente diferentes.

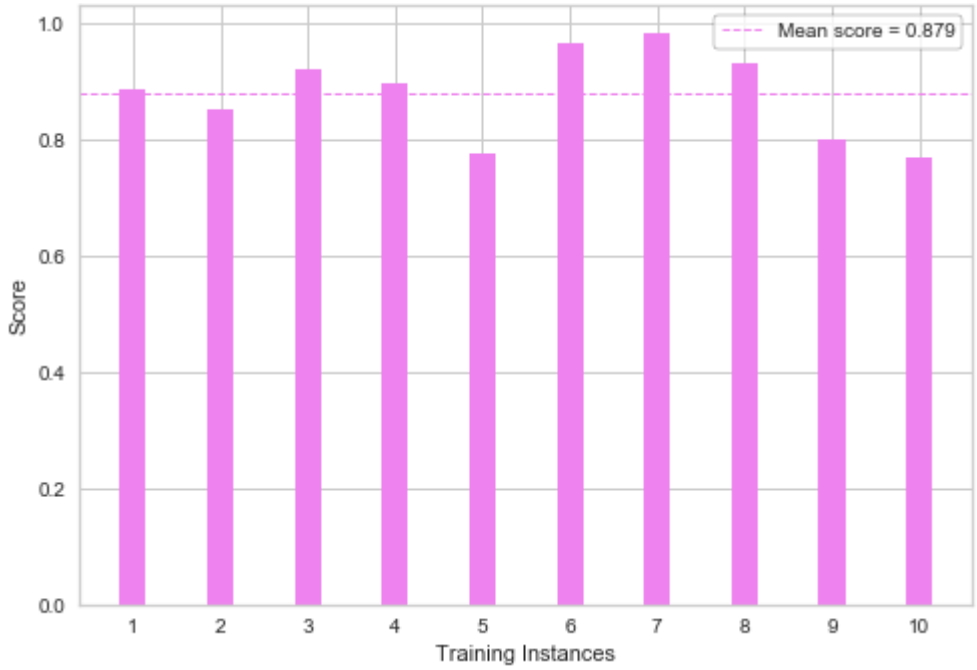

Gráfico de resultados 38. Métricas de exactitud para cada partición de la validación cruzada en Gradient Boosting (MACCS). Resultado medio de la exactitud tras validación cruzada.

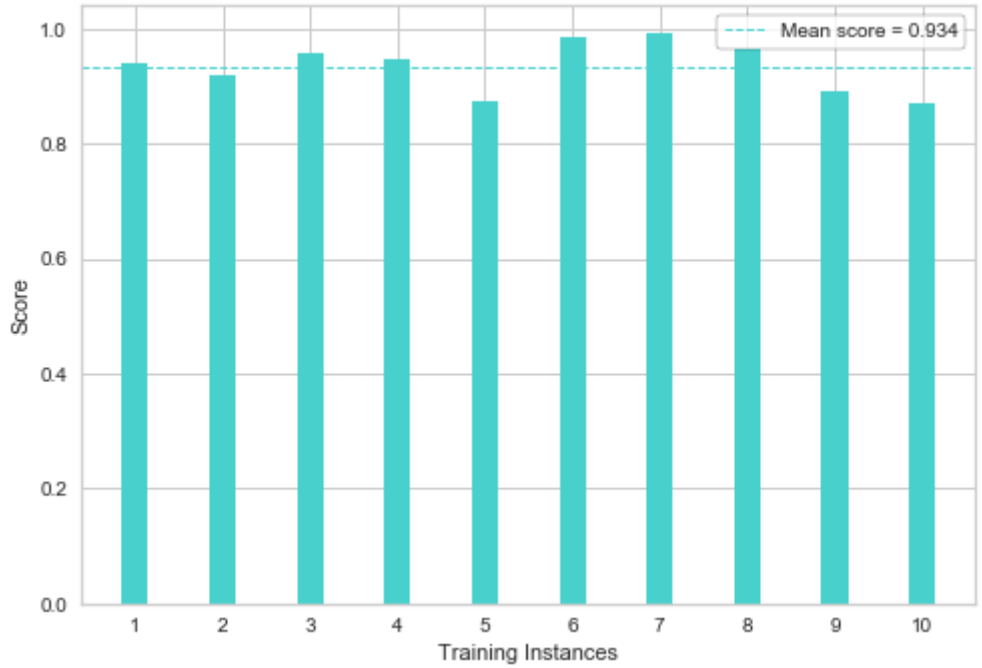

Gráfico de resultados 39. Métricas de F1 para cada partición de la validación cruzada en Gradient Boosting (MACCS). Resultado medio de la F1 tras validación cruzada.

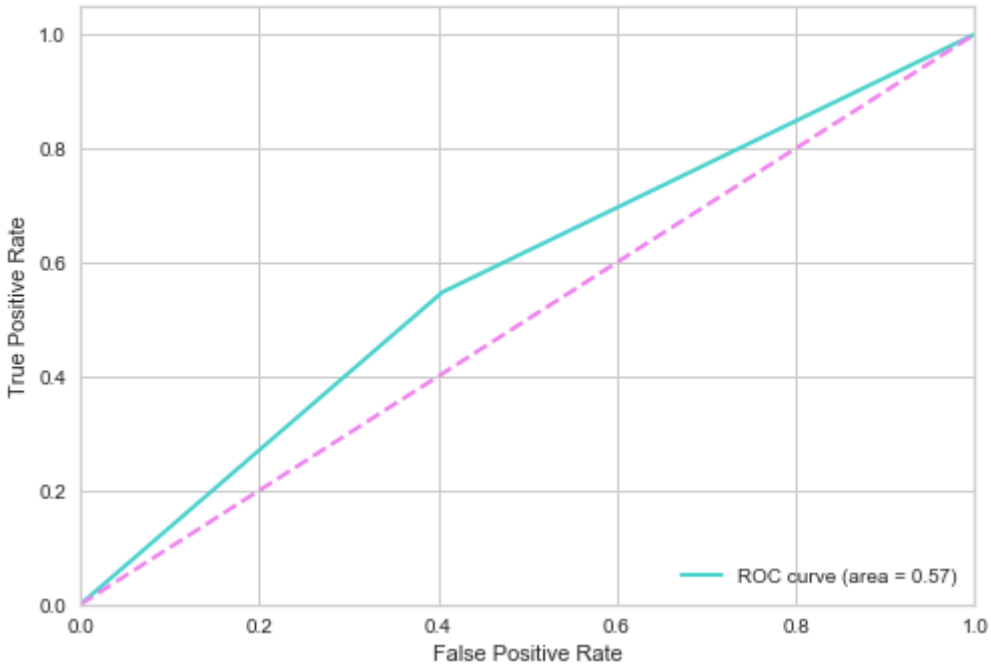

Gráfico de resultados 40. Curva ROC en Gradient Boosting (MACCS) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.57

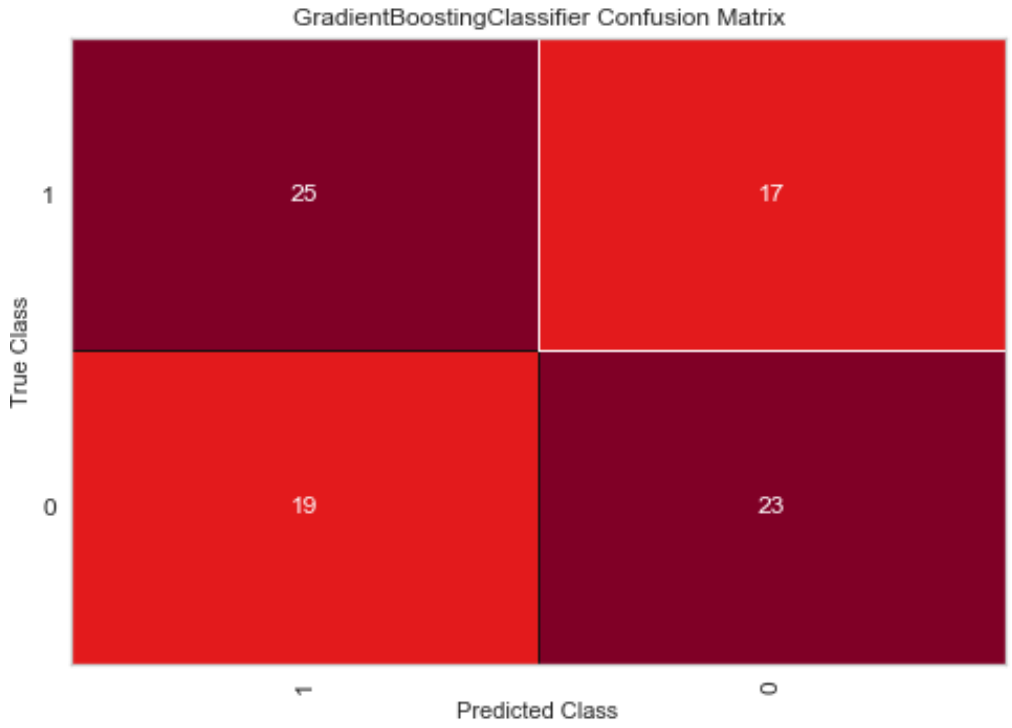

Gráfico de resultados 41. Matriz de confusión en Gradient Boosting (MACCS) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

SVM

Con los hiperparámetros configurados igual que para los fingerprint de tipo Morgan, se obtienen los siguientes resultados:

Para C = 1.0

Sensibilidad: 0.5714

Especificidad: 0.5476

Exactitud (accuracy) con 10-fold-CV: 0.8943

F1 con 10-fold-CV: 0.943

De este modelo de clasificación, para este conjunto de datos, se puede decir, también, que es un buen modelo, bastante exacto, con resultados consistentes, y una sensibilidad aceptable. Lo mejor y más destacable vuelven a ser los resultados de exactitud y F1 tras la validación cruzada, que dan idea del buen modelo que es.

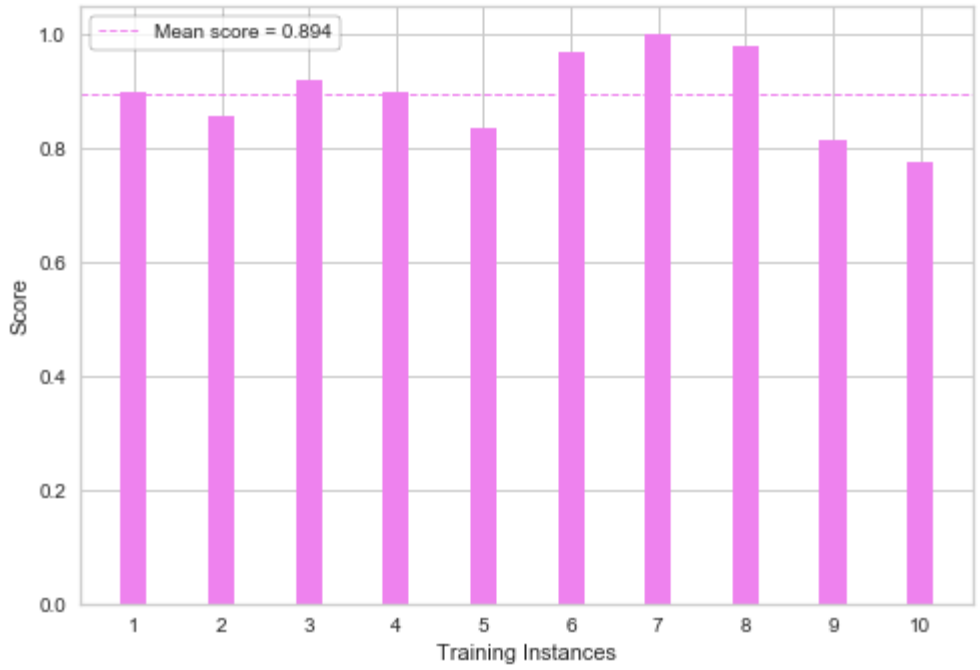

Gráfico de resultados 43. Métricas de exactitud para cada partición de la validación cruzada en SVC (MACCS). Resultado medio de la exactitud tras validación cruzada.

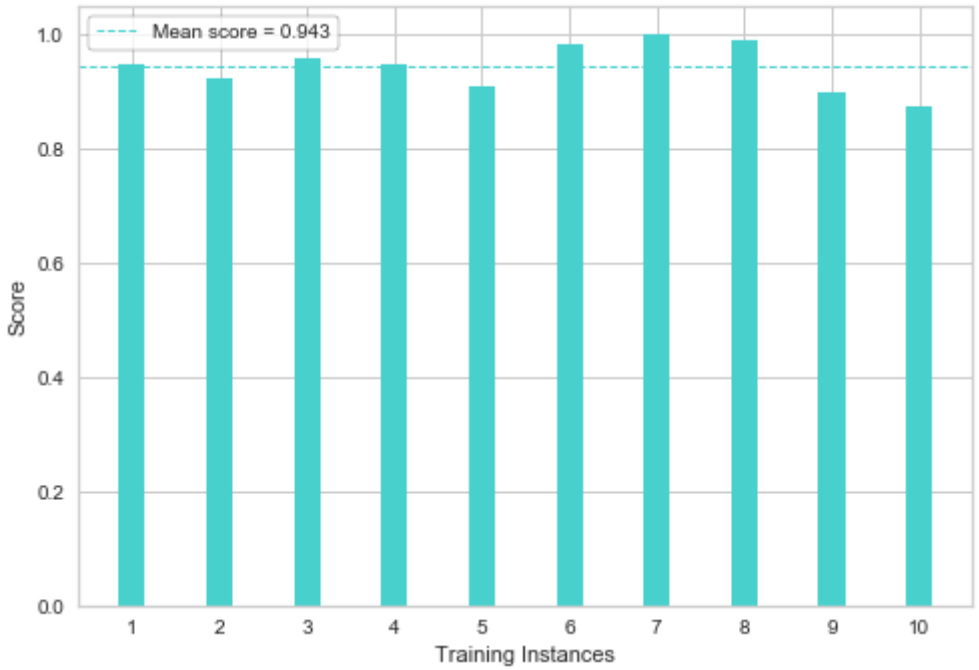

Gráfico de resultados 42. Métricas de F1 para cada partición de la validación cruzada en SVC (MACCS). Resultado medio de la F1 tras validación cruzada.

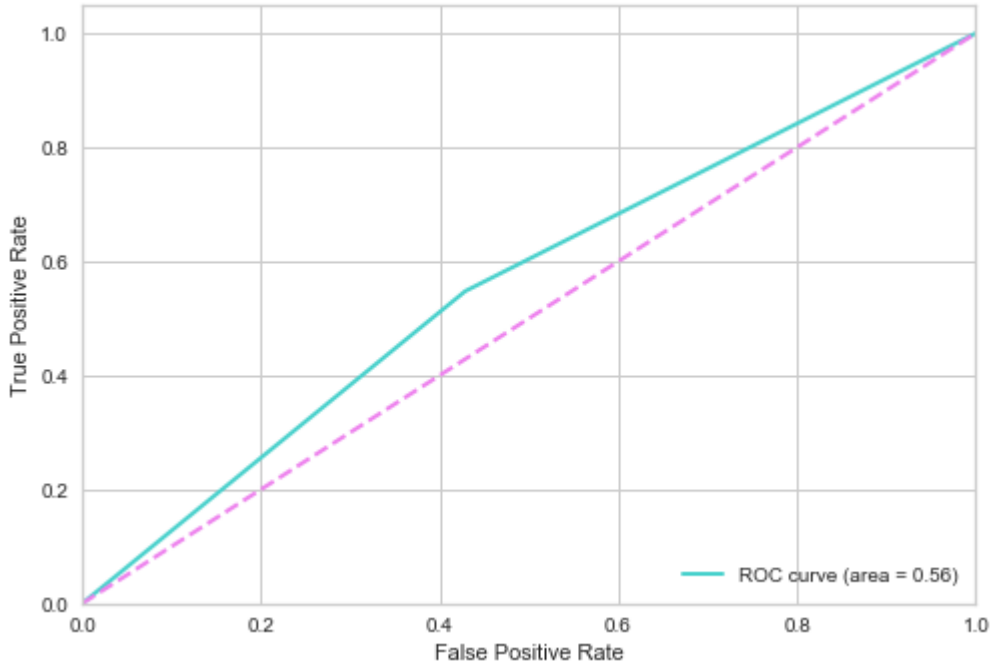

Gráfico de resultados 45. Curva ROC en SVC (MACCS) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.56

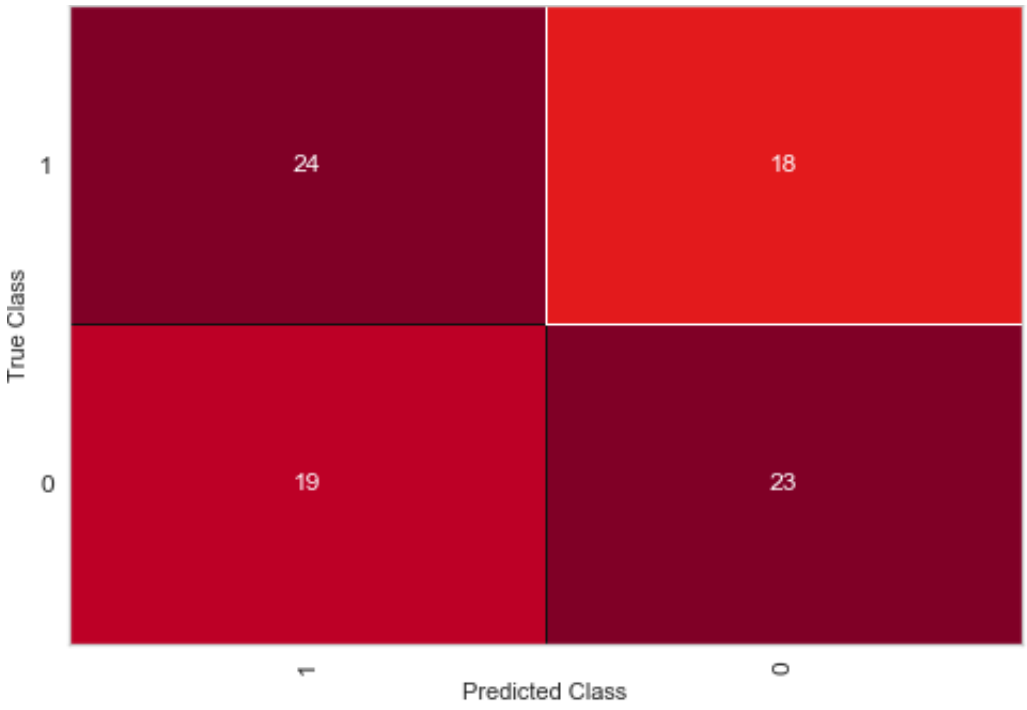

Gráfico de resultados 44. Matriz de confusión en SVC (MACCS) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

Random Forests

Con los hiperparámetros configurados igual que para los fingerprint de tipo Morgan, se obtienen los siguientes resultados:

Para n = 50

Sensibilidad: 0.6666

Especificidad: 0.5952

Exactitud (accuracy) con 10-fold-CV: 0.8799

F1 con 10-fold-CV: 0.934

Para n = 100

Exactitud (accuracy) con 10-fold-CV: 0.8799

F1 con 10-fold-CV: 0.934

De este modelo de clasificación, para este conjunto de datos, se puede decir, también, que es un buen modelo, bastante exacto, con resultados consistentes, y una sensibilidad aceptable. Su fundamento teórico tiene cosas en común con el modelo de Gradient Boosting, y vemos que, de hecho, rinde una exactitud y F1 bastante similares tras la validación cruzada. Existe un empate casi total entre el modelo Logistic, Naive Bayes, SVM, Gradient Boosting y Random Forests.

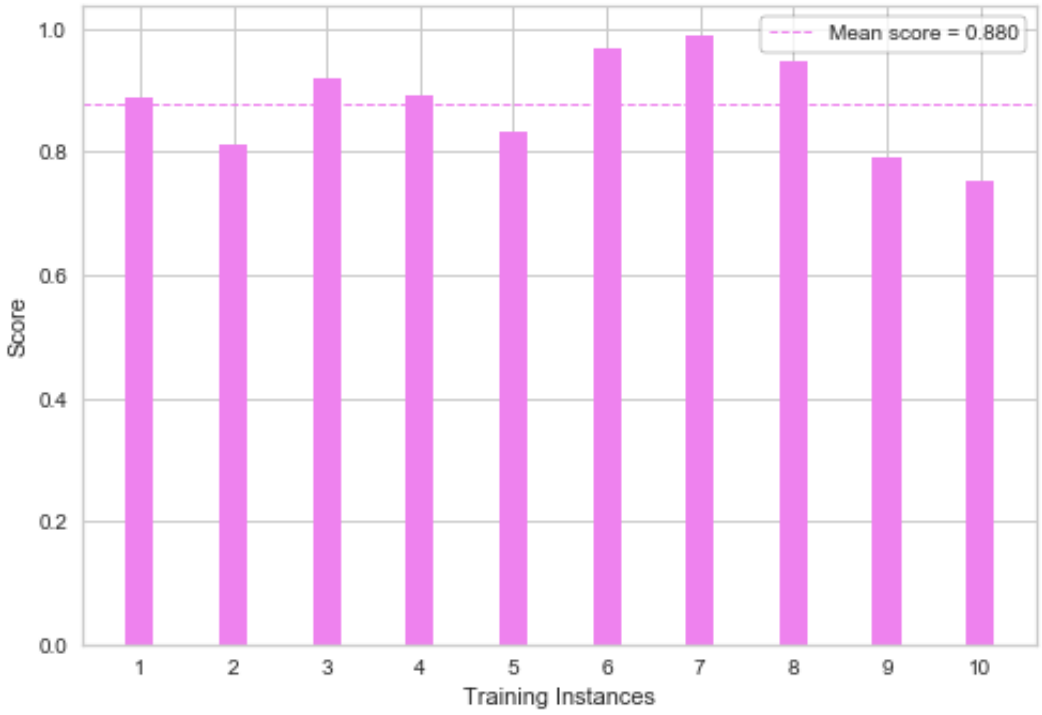

Gráfico de resultados 46. Métricas de exactitud para cada partición de la validación cruzada en Random Forest (MACCS). Resultado medio de la exactitud tras validación cruzada.

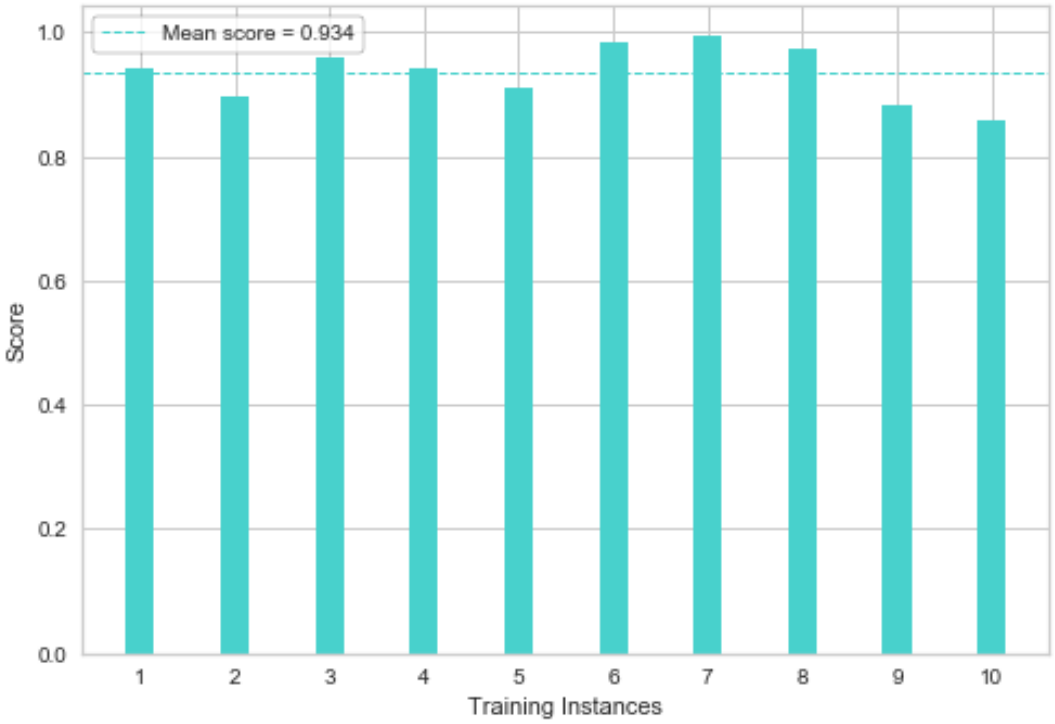

Gráfico de resultados 48. Métricas de F1 para cada partición de la validación cruzada en Random Forest (MACCS). Resultado medio de la F1 tras validación cruzada.

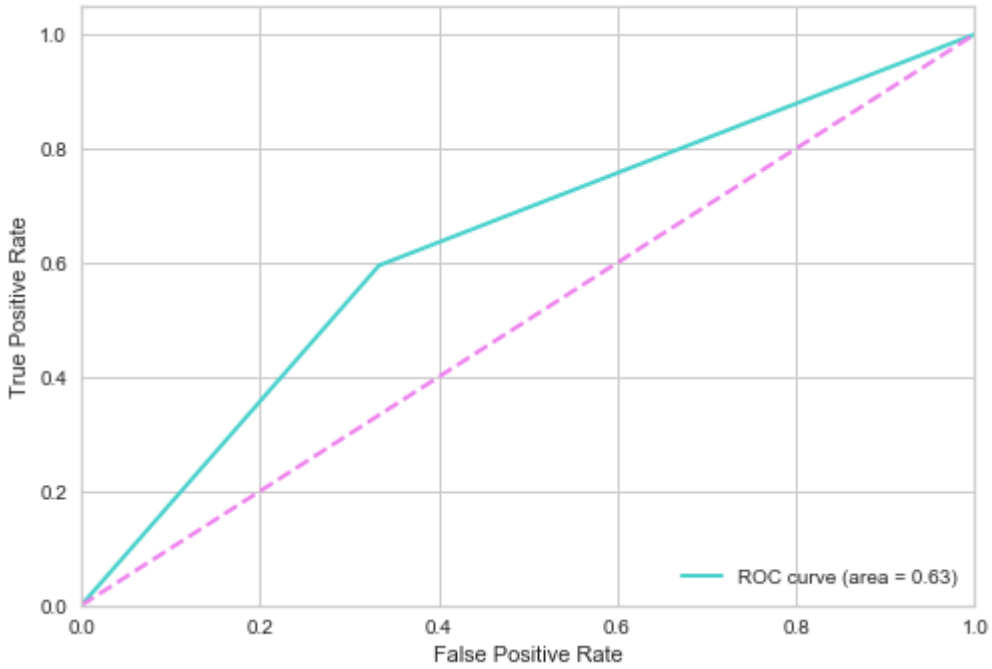

Gráfico de resultados 47. Curva ROC en Random Forest (MACCS) para una de las posibles particiones del conjunto de datos. Área bajo la curva, AUC = 0.63

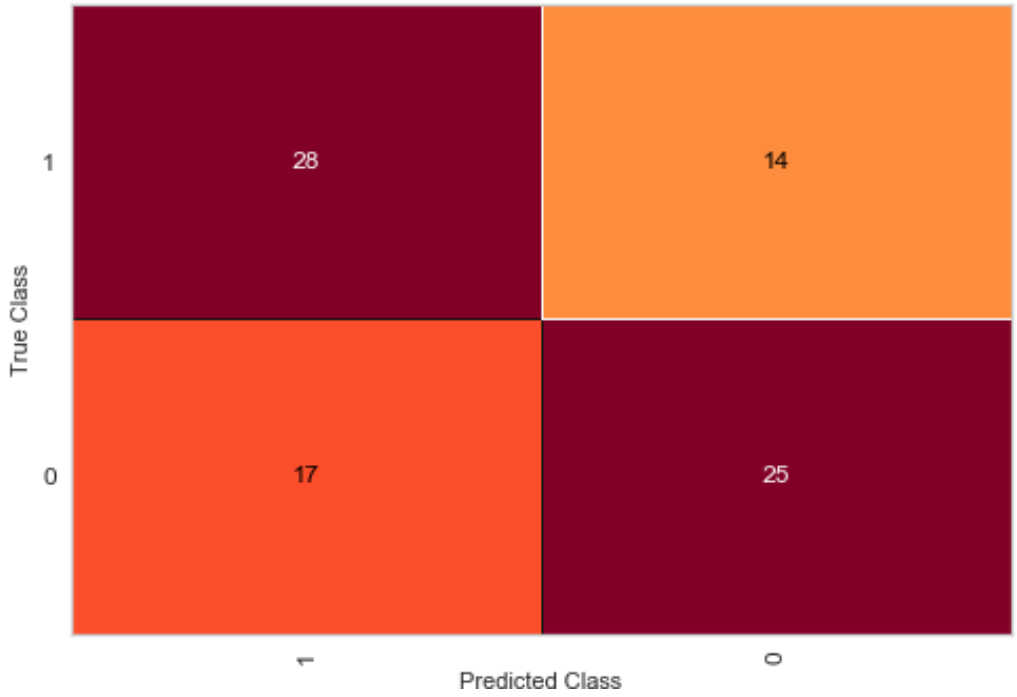

Gráfico de resultados 49. Matriz de confusión en Random Forest (MACCS) para una de las posibles particiones del conjunto de datos. En diagonal, de derecha a izquierda, se leen los falsos negativos y debajo, los falsos positivos.

Métricas de exactitud y F1 para fingerprints tipo MACCS

	Exactitud	F1 (precisión y sensibilidad)
Logistic regression	0.8900	0.940
kNN	0.8767	0.877
Naive Bayes	0.8349	0.904
SVM	**0.8943**	**0.943**
Gradient Boosting (100 árboles en total)	0.8788	0.934
Random Forests (100 árboles por bosque)	0.8799	0.934

Tabla 2. Métricas de exactitud y F1 para fingerprints tipo MACCS.

4.1.3 Evaluación global de los modelos de clasificación.

F1	Morgan	MACCS
Logistic regression	0.927	0.940
kNN	0.933	0.877
Naive Bayes	0.809	0.904
SVM	**0.943**	**0.943**
Gradient Boosting	0.931	0.934

Random Forests	0.940	0.934

Tabla 3. Métrica de F1 (precisión + sensibilidad) comparando fingerprints Morgan y MACCS con 10-fold-cross-validation[38]

Como se puede apreciar, a excepción del modelo Naive Bayes, que ofrece una métrica F1 más baja de todos los modelos estudiados, todos los demás están en el intervalo de 0.90 – 0.94, siendo valores muy buenos. Se puede decir, además, que no existen diferencias notables, quitando la excepción ya remarcada, entre utilizar un determinado tipo de fingerprint u otro.

4.2 Evaluación de modelos de regresión

A continuación, se exponen los resultados de los modelos de regresión explicados anteriormente en el punto *3.5 Desarrollo de modelos de machine learning. Regresión.*; para todos ellos, el conjunto de datos para probar los diferentes modelos ha sido el expuesto en el punto *3.3 Obtención de los fingerprints y estructura de los datos para ser usados en los modelos.*, donde = 1873, con los puntos atípicos ya eliminados, así como aquellas filas donde la variable pChembl presentaba un valor "*nan*" (*not a number*)

4.2.1 Fingerprints tipo Morgan

Regresión lineal. Regresión con penalizaciones (*ridge regression*)

Para este tipo de regresión, el hiperparámetro a configurar es el lambda, λ, como ya se ha explicado en *3.5.1 Regresión lineal. Regresión con penalizaciones (ridge regression)* Se prueba, para una división del conjunto de datos final en entrenamiento y test, los valores de R^2 para cada valor de lambda λ dado, para elegir uno de ellos con el que hacer la validación cruzada. Se aprecia que el mejor valor es λ = 2.

0.25	0.074
0.5	0.134
0.75	0.158
1	0.17
1.25	0.177
1.5	0.18
2	0.182
3	0.178
4	0.171
5	0.163

Tabla 4. Métrica R^2 para cada valor dado al parámetro lambda λ.

[38] (Shung, 2018)

Métricas para el modelo con lambda λ = 2.

R^2 : 0.1819

Ajustado R^2 : 0.2227

Error medio cuadrático: 0.9676

R^2 con 10-fold-CV: -0.470

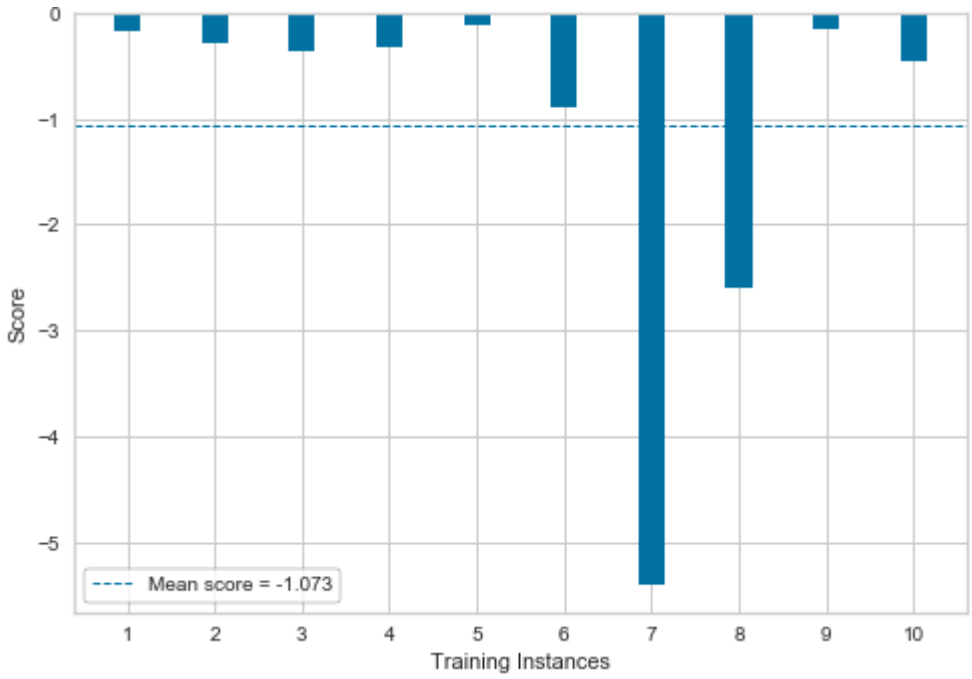

Gráfico de resultados 50. Métricas de R^2 para cada partición de la validación cruzada en ridge regression. Resultado medio de estas R^2 tras validación cruzada.

SVM (regresión, Morgan)

Hiperparámetros: RBF & C=1.0

R^2 : 0.0592

Ajustado R^2 : 0.0725

Error medio cuadrático: 1.1127

R^2 con 10-fold-cross-validation: -0.2729

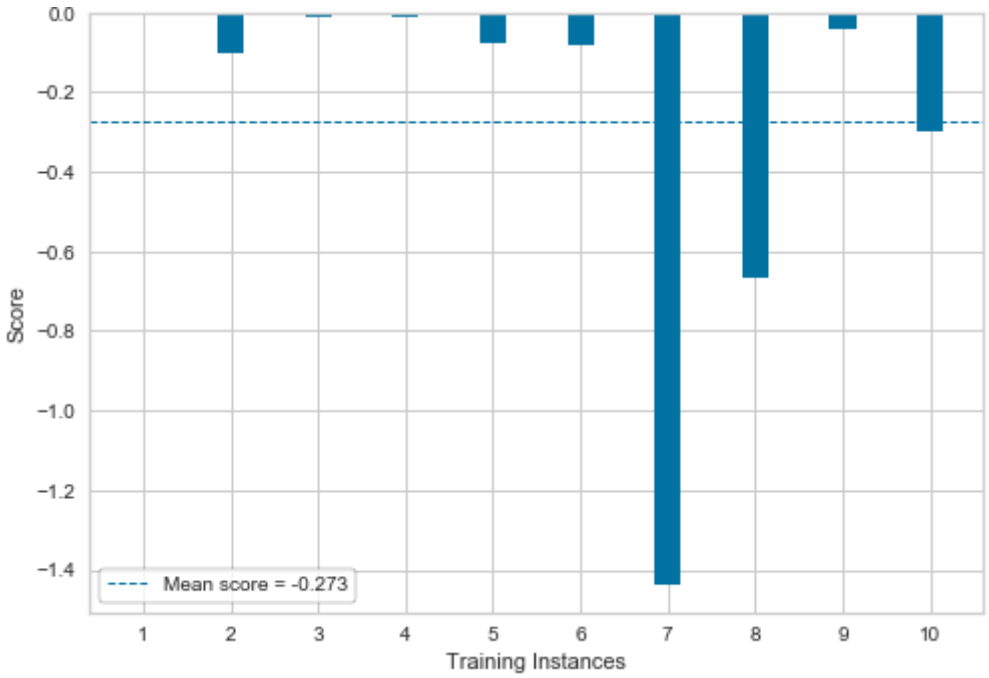

Gráfico de resultados 51. Métricas de R^2 para cada partición de la validación cruzada en SVM. Resultado medio de estas R^2 tras validación cruzada.

Hiperparámetros: RBF & C=10.0

R^2 : 0.0967

Ajustado R^2 : 0.1183

Error medio cuadrático: 1.0685

R^2 con 10-fold-cross-validation: -0.4962

Hiperparámetros: Poly, degree=3, C=1.0

R^2 : -0.0266

Ajustado R^2 : -0.0326

Error medio cuadrático: 1.2143

Hiperparámetros: Poly, degree=3, C=10.0

R^2 : -0.0266

Ajustado R^2 : -0.0326

Error medio cuadrático: 1.2143

Gradient Boosting (regresión, Morgan)

Learning rate = 0.1

R^2 : 0.1448

Ajustado R^2 : 0.1773

Error medio cuadrático: 1.0115

R^2 con 10-fold-CV: -0.5181

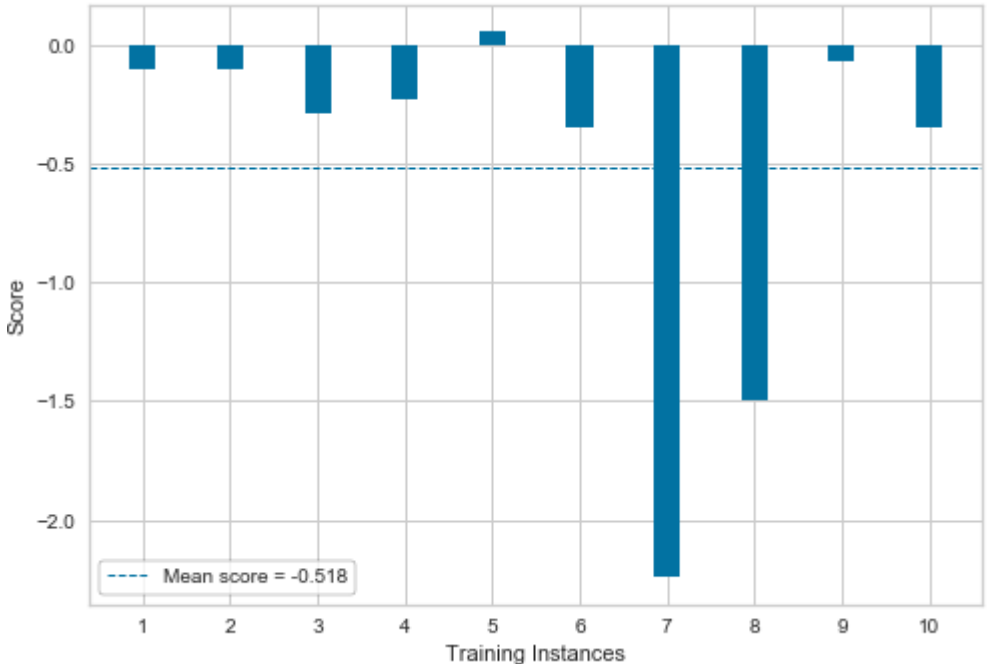

Gráfico de resultados 52. Métricas de R^2 para cada partición de la validación cruzada en Gradient Boosting. Resultado medio de estas R^2 tras validación cruzada.

Learning rate = 0.25

R^2 : 0.0938

Ajustado R^2 : 0.1148

Error medio cuadrático: 1.0719

R^2 con 10-fold-CV: -0.7823

Learning rate = 0.5

R^2 : 0.0346

Ajustado R^2 : 0.0424

Error medio cuadrático: 1.1419

R^2 con 10-fold-CV: -1.1360

Learning rate = 0.75

R^2 : 0.0276

Ajustado R^2 : 0.0337

Error medio cuadrático: 1.1502

R^2 con 10-fold-CV: -1.9181

Random Forests (regresión, Morgan)

Con 50 árboles

R^2 : 0.1287

Ajustado R^2 : 0.1575

Error medio cuadrático: 1.0306

R^2 con 10-fold-cross-validation: -0.6514

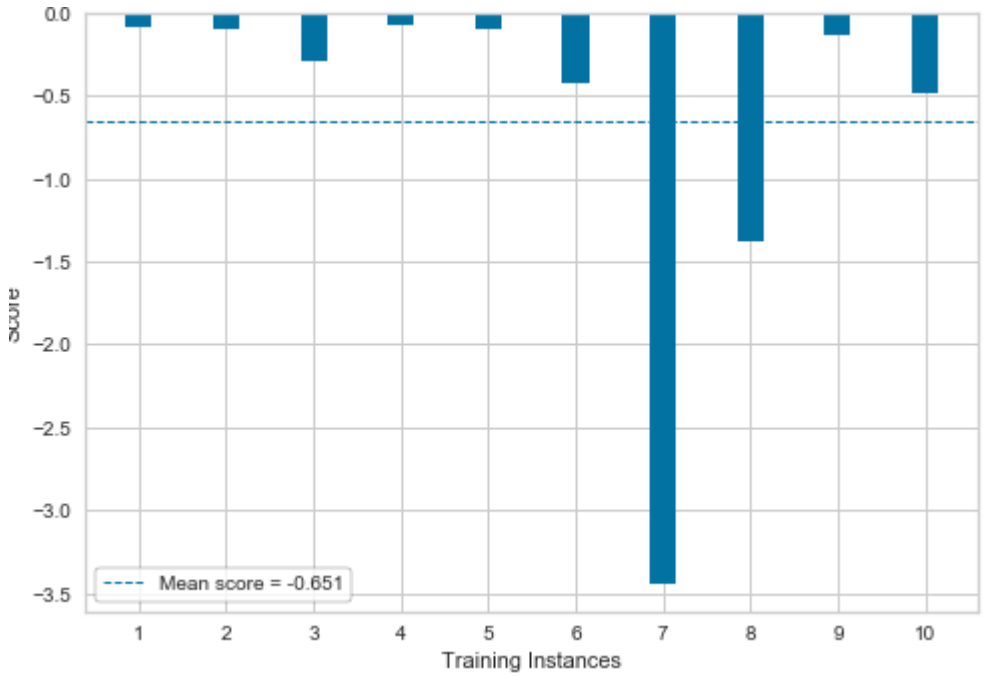

Gráfico de resultados 53. Métricas de R^2 para cada partición de la validación cruzada en Random Forest con 50 árboles. Resultado medio de estas R^2 tras validación cruzada.

Con 100 árboles

R^2 : 0.1048

Ajustado R^2 : 0.1283

Error medio cuadrático: 1.0589

R^2 con 10-fold-cross-validation: -0.6331

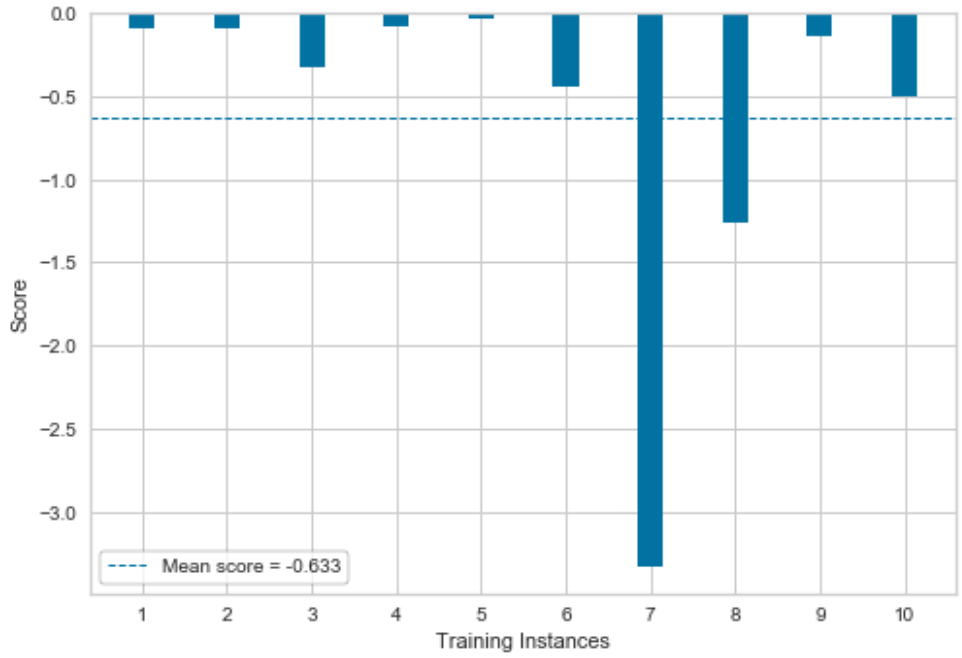

Gráfico de resultados 54. Métricas de R^2 para cada partición de la validación cruzada en Random Forest con 100 árboles. Resultado medio de estas R^2 tras validación cruzada.

4.2.2 Fingerprints tipo MACCS.

Regresión lineal. Regresión con penalizaciones (*ridge regression*). MACCS.

Para este tipo de regresión, el hiperparámetro a configurar es el lambda, λ, como ya se ha explicado en *3.5.1 Regresión lineal. Regresión con penalizaciones (ridge regression)* Se prueba, para una división del conjunto de datos final en entrenamiento y test, los valores de R^2 para cada valor de lambda λ dado, para elegir uno de ellos con el que hacer la validación cruzada. Se aprecia que el mejor valor es λ = 0.25.

0.25	0.088
0.5	0.087
0.75	0.084
1	0.08
1.25	0.076
1.5	0.072
2	0.065
3	0.055
4	0.048
5	0.042

Tabla 5 Métrica R^2 para cada valor dado al parámetro lambda λ.

Métricas para el modelo con lambda λ = 0.25.

R^2 : 0.0884

Ajustado R^2 : 0.1082

Error medio cuadrático: 1.0782

R^2 con 10-fold-CV: -0.4212

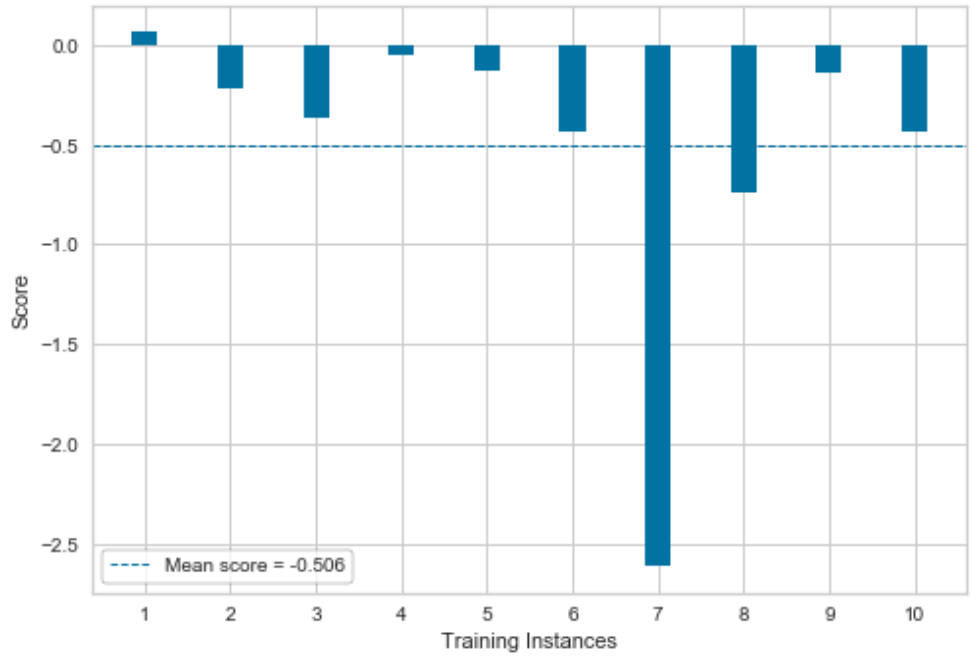

Gráfico de resultados 55. Métricas de R^2 para cada partición de la validación cruzada en ridge regression. Resultado medio de estas R^2 tras validación cruzada.

SVM (regresión, MACCS)

Hiperparámetros: RBF & C=1.0

R^2 : 0.0867

Ajustado R^2 : 0.1061

Error medio cuadrático: 1.0802

R^2 con 10-fold-cross-validation: -0.3935

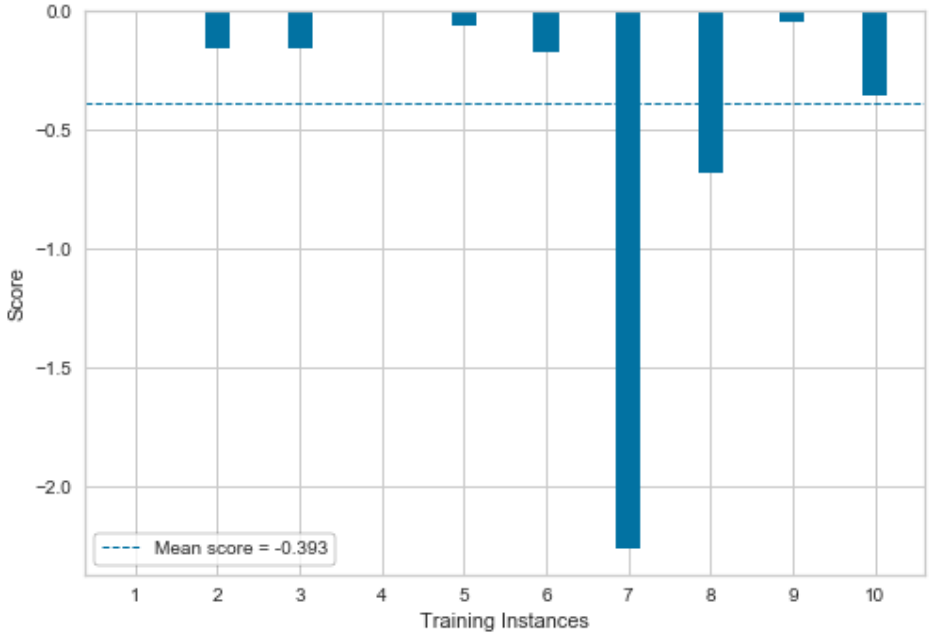

Gráfico de resultados 56. Métricas de R^2 para cada partición de la validación cruzada en SVM. Resultado medio de estas R^2 tras validación cruzada.

Hiperparámetros: RBF & C=10.0

R^2 : 0.0834

Ajustado R^2 : 0.1021

Error medio cuadrático: 1.0802

R^2 con 10-fold-cross-validation: -0.5468

Gradient Boosting (regresión, MACCS)

Learning rate = 0.1

R^2 : 0.1120

Ajustado R^2 : 0.1371

Error medio cuadrático: 1.0504

R^2 con 10-fold-CV: -0.6209

Learning rate = 0.25

R^2 : 0.0913

Ajustado R^2 : 0.1118

Error medio cuadrático: 1.0747

R^2 con 10-fold-CV: -0.9421

Learning rate = 0.5

R^2 : 0.0232

Ajustado R^2 : 0.0284

Error medio cuadrático: 1.1553

R^2 con 10-fold-CV: -1.1377

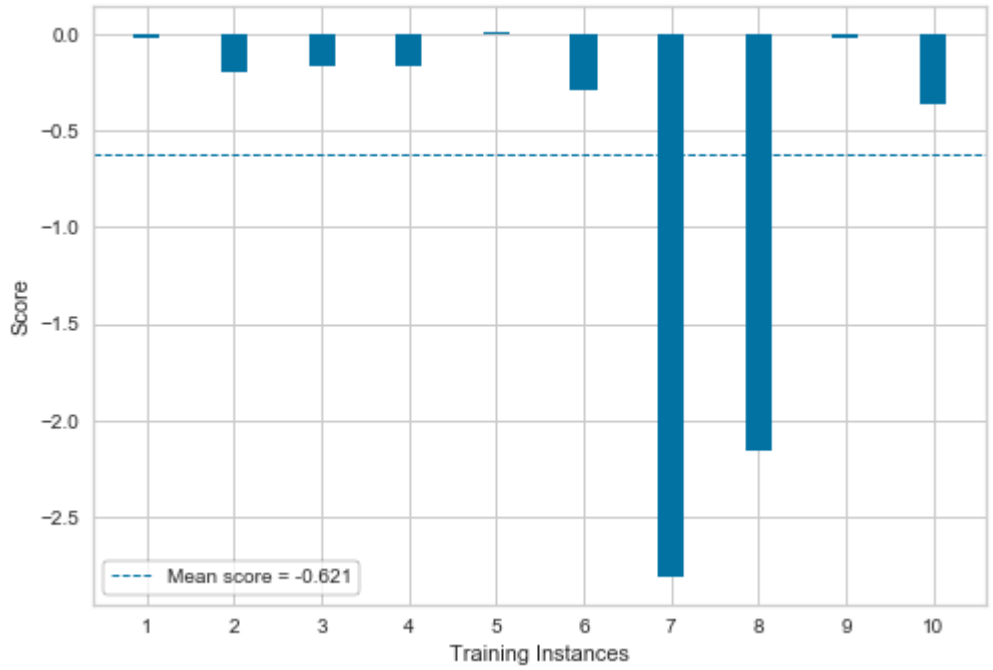

Gráfico de resultados 57. Métricas de R^2 para cada partición de la validación cruzada en Gradient Boosting. Resultado medio de estas R^2 tras validación cruzada.

Random Forests (regresión, MACCS)

Con 50 árboles

R^2 : 0.0369

Ajustado R^2 : 0.0453

Error medio cuadrático: 1.1391

R^2 con 10-fold-cross-validation: -0.6173

Con 100 árboles

R^2 : 0.0369

Ajustado R^2 : 0.0453

Error medio cuadrático: 1.1391

R^2 con 10-fold-cross-validation: -0.606

Métricas de R^2 de los modelos de regresión con 10-fold-cross-validation[39]

R^2	MORGAN	MACCS
Ridge	-0.470	-0.4212
SVM	-0.2728	-0.3935
Gradient Boosting	-0.5181	-0.6209
Random Forest	-0.6331	-0.606

Tabla 6. Métricas de R^2 de los modelos de regresión con 10-fold-cross-validation.

Como se puede apreciar, los modelos de regresión no resultan, en ningún caso, buenos predictores de la actividad de moléculas pequeñas sobre el receptor estudiado. Todos presentan valores negativos para R^2, incluso con los ajustes de los hiperparámetros aplicados sobre los modelos, con el objetivo de reducir el error medio cuadrático o los residuos cuadráticos (en el caso de la regresión lineal *ridge regression*)

No obstante, es necesario realizar más pruebas sobre este tipo de algoritmos y sus diferentes configuraciones de los hiperparámetros, aunque lo hallado anteriormente sugiere que no se darían mejoras notables.

[39] (Mishra, 2019)

5. DISCUSIÓN

De los resultados presentados, podemos hacer una interpretación tanto global, como por los dos tipos de modelos de *machine learning* probados, siempre considerando el hecho fundamental que estas lecturas han de suscribirse al conjunto de datos utilizado, para el receptor 5-HT2$_A$.

Una lectura en líneas generales nos permite ver que los modelos de clasificación son mejores que los modelos de regresión.
No se trata de ninguna diferencia menor, si no que los modelos de clasificación son especialmente buenos, con muy buenas métricas de exactitud [0.83-0.89] y de F1 (precisión + sensibilidad) [0.90-0.94], con un conjunto de datos balanceado y con una detección y eliminación de los puntos atípicos realizada. Aunque los resultados de sensibilidad y especificidad sean tan solo aceptables, su lectura debe ser limitada, pues son los resultados de una única de las posibles particiones del conjunto de datos en subconjuntos de entrenamiento y test, de las múltiples combinaciones posibles, tal y como sugieren los buenos valores para la métrica F1.

Por otro lado, los modelos de regresión no resultan útiles en ningún caso para predecir la actividad de las moléculas pequeñas sobre el receptor estudiado. Aunque existe una menor variedad de métricas para evaluar este tipo de modelos, la abundante bibliografía existente siempre apunta al error medio cuadrático, la R^2 (que considera el propio error cuadrático medio), R^2 ajustada a variables y observaciones…etc. como las métricas a observar para evaluar un modelo de regresión; con lo cual, con los resultados ofrecidos, que atienden al mismo conjunto de datos balanceado y sin valores atípicos que el usado en los modelos de clasificación, se puede hacer esta afirmación.
Los modelos de regresión se ajustan muy mal, tanto que hacen predicciones incluso peores que el modelo más básico con los valores medios de los puntos, pudiéndose explicar los valores negativos de la R^2 tras la validación cruzada.

De una forma más específica, dentro de los modelos de clasificación, para los datos utilizados, se puede ver que aquel que mejores métricas arroja es el **SVM** (de *support vector machine*, por sus siglas en inglés), que basa la clasificación en el trazado de un plano que pueda separar las dos clases existentes, con el menor número de observaciones mal clasificadas posibles. En particular, se trata de la configuración que usa la función radial para dibujar tal plano en el espacio multidimensional, con el hiperparámetro C = 1.0 para aplicar como penalización a aquellos puntos que se dispongan incorrectamente al lado del plano que no les corresponda.

Una de las características más notorias es que en este trabajo, se ha puesto en práctica, para un conjunto de datos determinado del receptor 5-HT2$_A$, diversos modelos de *machine learning* supervisado, tanto de clasificación, como de regresión, permitiendo la comparación entre estos.
De forma adicional, y a pesar de estar bastante difundida la gran utilidad de los fingerprints como una técnica novedosa de representar la composición de las moléculas en un plano bidimensional y tridimensional, la cantidad de estudios

con ellos, de forma específica para cualquier tipo de diana terapéutica o grupo farmacológico que se quisiera estudiar es aún minoritaria.

La mayoría de los estudios aún optan por intentar explicar comportamientos relativos a la unión a proteínas o dianas terapéuticas con variables descriptoras que recogen valores relativos al comportamiento cinético de los compuestos en el interior del organismo[40] [41], un hecho que probablemente, aunque pueda rendir modelos de *machine learning* aceptables, estará dejando de lado abundante información relativa a la molécula que explica su mayor o menor grado de unión a dicha proteína o diana terapéutica.[42] [43] La mayoría de los avances se da en cuanto a los fundamentos matemáticos sobre los que se asientan los modelos, pero no tanto en cuanto a su aplicación sobre otras ciencias empíricas, especialmente en el campo de las ciencias de la salud.[44]

Por estas razones, son necesarios más estudios como el realizado aquí, donde, para diversas dianas terapéuticas, con mayores o menores recursos farmacológicos (bien para buscar mejorar el arsenal terapéutico ya existente o descubrir nuevas moléculas activas sobre estos receptores) se realicen modelados basados en los fingerprints moleculares.

De esta forma, además, se podrá ahondar en cuestiones como si el uso de fingerprints moleculares rinde mejores resultados con un tipo de modelos u otros (clasificación o regresión). En este trabajo, como se ha comentado anteriormente, queda expuesto el mal rendimiento de los modelos de regresión, pero es necesario realizar más estudios, de forma que se pueda comprobar si existe algún tipo de relación entre el uso de los fingerprints y los modelos de regresión, o se debe a que es necesario realizar un ajuste más preciso de los hiperparámetros de los modelos o experimentar con otros modelos diferentes a los expuestos aquí.

Así mismo, es óptimo remarcar, que los resultados aquí obtenidos se ciñen a un solo conjunto de datos obtenido de un único repositorio, y si se quisiera inferir un conocimiento más robusto acerca de cualquier receptor farmacológico, sería oportuno realizar estudios paralelos con datos de diferente procedencia, pero con una estructura interna similar.

6. CONCLUSIONES

Al inicio de este trabajo, se había propuesto trabajar con ambos tipos de modelos de machine learning, tanto de clasificación como de regresión, con el objetivo de conocer qué tipo de estos dos era el que rendía mejores resultados con el conjunto de datos utilizado.

[40] (Doniger, Hofmann, & Yeh, 2002)
[41] (Periwal, Rajappan, Jaleel, & Scaria, 2011)
[42] (Maciukiewicza, y otros, 2018)
[43] (Ekins, y otros, 2017)
[44] (Lo, Rensi, Torng, & Altman, 2018)

A lo largo del trabajo, se ha podido comprobar que, para aquellos datos donde las variables descriptoras utilizadas son los fingerprints de cada una de las moléculas, los modelos de machine learning que mejor se comportan son los de tipo clasificador (activo/inactivo), con una exactitud y precisión/sensibilidad elevadas (0.90-0-94) incluso tras un proceso de validación cruzada, que permite hacer el cálculo de dichas métricas en diferentes particiones aleatorias del conjunto de datos en subconjuntos de entrenamiento y test diferentes.

Los modelos de machine learning de tipo regresor no se comportan bien, y de este trabajo, se puede deducir que su uso no está recomendado, pues los resultados de la métrica de R^2, tras validación cruzada, rinden resultados negativos, lo cual, si bien entra dentro de lo posible a nivel matemático, ha de ser interpretado como que tales modelos no se adaptan a la distribución de los datos, ni son capaces de establecer relaciones evidentes entre las variables descriptoras y la variable regresora de interés.

Los resultados para ambos tipos de modelos, tanto con los fingerprints de tipo Morgan como MACCS, a pesar de ser bastante diferentes en sus fundamentos teóricos para construirse a partir de las estructuras bi/tridimensionales de las moléculas, se puede apreciar que son muy parecidos, dándose una diferencia de centésimas, cuando no dan unos resultados relativos a la exactitud o precisión las predicciones exactamente igual. La única excepción se puede apreciar con el algoritmo de clasificación de Naive Bayes, que para el fingerprint de tipo MACCS rinde un resultado una décima mejor que el mismo con fingerprint de tipo Morgan; esto se puede traducir como que hace las predicciones un 10% mejor, hecho por el cual, si se optase usar este algoritmo en trabajos futuros, deberá ser un factor para tener en cuenta.

Las limitaciones más notorias relativas a este estudio son, por tanto, relativas a tres bloques: el tipo de fingerprints utilizado, los modelos utilizados y el establecimiento de los hiperparámetros dentro de cada uno de estos y los datos utilizados.

En cuanto a los fingerprints, es conveniente hacer más pruebas comparando el rendimiento de los diversos modelos expuestos aquí, u otros, para ver su comportamiento con un tipo de fingerprints u otros (ej. Con Morgan radio 3, otros tipos de fingerprints topológicos, nuevos fingerprints en desarrollo[45]…etc.) y comprobar si el hecho de obtener resultados similares entre los fingerprints de Morgan y MACCS, incluida la excepción del algoritmo de Naive Bayes, se repiten con otros conjuntos de datos, para el mismo receptor estudiado aquí o cualquier otro.
Relativo a los modelos probados, tanto para clasificación como para regresión, es conveniente hacer estudios más enfocados a ver el comportamiento de un tipo de algoritmos en particular, e incluso, de uno o varios de forma concreta, estableciendo diferentes configuraciones de hiperparámetros, para ver si existiese alguna configuración que diera mejores resultados de exactitud y F1 que los obtenidos aquí. (ej. Tomar el algoritmo de regresión Gradient Boosting y ajustar los hiperparámetros *loss* [tipo de función de perdida] o *learning_rate*

[45] (Gortari, García-Jacas, Martinez-Mayorga, & Medina-Franco, 2017)

[ponderación de cada árbol al aprendizaje en total] para ver si mejora o empeora el modelo con respecto al presentado aquí)

La tercera y última es la anteriormente destacada al uso de una única procedencia de los datos, y una recomendación para futuros estudios sería obtener los datos relativos a las moléculas y sus respectivos valores de Ki de repositorios de datos diferentes al utilizado aquí. También se aconseja reproducir estudios como este, o con las sugerencias hechas anteriormente, para cualquier otro tipo de receptor en el que se desee profundizar el conocimiento de sus mejores y peores ligandos.

7. BIBLIOGRAFÍA

5-hydroxytryptamine receptor 2A. (s.f.). Obtenido de DrugBank Canada: https://www.drugbank.ca/polypeptides/P28223

Academy, E. (2019). *Discussion Vs. Conclusion: Know the Difference Before Drafting Manuscripts*. Obtenido de Enago Academy: https://www.enago.com/academy/discussion-conclusion-know-difference-drafting-manuscript/

Administration, U. F. (2020). *Impact Story: CDER Assessment of Drug Impurity Mutagenicity by Quantitative Structure-Activity Relationship Modeling*. Obtenido de Regulatory Science In Action: https://www.fda.gov/drugs/regulatory-science-action/impact-story-cder-assessment-drug-impurity-mutagenicity-quantitative-structure-activity-relationship

Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., & Ridella, S. (2012). The 'K' in K-fold Cross Validation. *European Symposium on Artificial Neural Networks, Computational Intelligence.* Bruges, Belgium. Obtenido de https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2012-62.pdf

Bajorath, J. (s.f.). *Fingerprint Design And Molecular Complexity Effects.* Obtenido de Universität Bonn. LIMES Program Unit Chemical Biology: http://infochim.u-strasbg.fr/CS3/program/material/Bajorath.pdf

Breiman, L. (1998). Arcing The Edge. *Annals of Probability, 26*, 1683-1702.

Breiman, L. (2001). Random Forests. *Machine Learning*, 5-32.

Brownlee, J. (2016). *A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning*. Obtenido de Machine Learning Mastery: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

Brownlee, J. (2016). *Logistic Regression for Machine Learning*. Obtenido de Machine Learning Mastery: https://machinelearningmastery.com/logistic-regression-for-machine-learning/

Brownlee, J. (2020). *Tour of Evaluation Metrics for Imbalanced Classification*. Obtenido de Machine Learning Mastery: https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

Dodge, Y. (2008). Least Absolute Deviation Regression. En *The Concise Encyclopedia of Statistics.*

Doniger, S., Hofmann, T., & Yeh, J. (2002). Predicting CNS Permeability of Drug Molecules: Comparison of Neural Network and Support Vector Machine Algorithms. *Journal Of Computational Biology*, 849-864.

Ekins, S., Freundlich, J. S., Clark, A. M., Anantpadma, M., Davey, R. A., & Madrid, P. (2017). Machine learning models identify molecules active against the Ebola virus in vitro. *F1000Research, 4*(1091).

Enzyme inhibitors. (s.f.). Obtenido de University College London: https://www.ucl.ac.uk/~ucbcdab/enzass/inhibition.htm

Eriksson, L., Jaworska, J., Worth, A. P., Cronin, M. T., McDowell, R. M., & Gramatica, P. (2003). Methods for Reliability and Uncertainty Assessment and for Applicability Evaluations of Classification- and Regression-Based QSARs. *Environmental Health Perspectives, 111*(10), 1361-1375.

Fara, D. C., & Oprea, T. I. (s.f.). *Cheminformatics - Basics: Molecular Descriptors and Fingerprints*. Obtenido de University of New Mexico, Health Sciences Center. Division of Biocomputing: http://datascience.unm.edu/biomed505/Course/Cheminformatics/basic/descs_fingers/molec_descs_fingerprints.htm

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning. *Journal Of Computer And System Sciences*, 119-139. Obtenido de face-rec.org/algorithms/Boosting-Ensemble/decision-theoretic_generalization.pdf

Friedman, J. H. (2001). Greedy Function Approximation. A Gradient Boosting Machine. *Annals of Statistics, 29*(5), 1189-1232. Obtenido de https://statweb.stanford.edu/~jhf/ftp/trebst.pdf

Fushiki, T. (2011). Estimation of prediction error by using K-fold cross-validation. *Statistics And Computing*, 137-146. Obtenido de https://link.springer.com/content/pdf/10.1007/s11222-009-9153-8.pdf

Gortari, E. F., García-Jacas, C. R., Martinez-Mayorga, K., & Medina-Franco, J. L. (2017). Database fingerprint (DFP): an approach to represent molecular databases. *Journal Of Cheminformatics*. doi:10.1186/s13321-017-0195-1

Horvath Dragos, M. G. (2009). Predicting The Predictability: A Unified Approach To The Applicability Domain Problem Of QSAR Models. *Journal Of Chemical Information And Modeling*, 1762–1776.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An Introduction To Statistical Learning With Applications In R.* Springer.

Khandelwal, R. (2018). *L1 And L2 Regularization*. Obtenido de Medium: https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2

Kim, K. (2019). *Ridge Regression for Better Usage*. Obtenido de Towards Data Science: https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db

Knauer, C. S., Campbell, J. E., Chio, C. L., & Fitzgerald, L. W. (May de 2009). Pharmacological Characterization of Mitogen-Activated Protein Kinase Activation by Recombinant Human 5-HT2C, 5-HT2A, and 5-HT2B Receptors. *Naunyn Schmiedebergs Arch Pharmacol., 379*, 461-471. Obtenido de https://link.springer.com/content/pdf/10.1007/s00210-008-0378-4.pdf

Krishni. (2018). *K-Fold Cross Validation*. Obtenido de Medium: https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833

Laksh. (2019). *A Practical Introduction to the Use of Molecular Fingerprints in Drug Discovery*. Obtenido de Towards Data Science: https://towardsdatascience.com/a-practical-introduction-to-the-use-of-molecular-fingerprints-in-drug-discovery-7f15021be2b1

Lantz, B. (2015). *Machine Learning With R.* Packt Publishing.

Li, S. (2017). *Building A Logistic Regression in Python, Step By Step*. Obtenido de Towards Data Science: https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

Lo, Y.-C., Rensi, S. E., Torng, W., & Altman, R. B. (2018). Machine Learning In Cheminformatics And Drug Discovery. *Drug Discovery Today, 23*(8).

Maciukiewicza, M., S.Marsheab, V., Anne-ChristinHauschild, A.Foster, J., SusanRotzinger, L.Kennedy, J., . . . JosephGeraci. (2018). GWAS-based

machine learning approach to predict duloxetine response in major depressive disorder. *Journal of Psychiatric Research, 99*, 62-68.

Misdrahi, D., Tessier, A., Daubigney, A., Meissner, W., Schürhoff, F., & al.., e. (2019). Extrapyramidal side effects of antipsychotics: prevalence and risk factors. Results from the national FACE-SZ cohort. *The Journal Of Clinical Psychiatry, 80*. Obtenido de https://hal-amu.archives-ouvertes.fr/hal-02473295/document

Mishra, D. (2019). *Regression: An Explanation of Regression Metrics And What Can Go Wrong*. Obtenido de Towards Data Science: https://towardsdatascience.com/regression-an-explanation-of-regression-metrics-and-what-can-go-wrong-a39a9793d914

Mitra, P., Rastogi, A., Rajpoot, M., Kumar, A., & Srivastava, V. (2017). A QSAR model of Olanzapine derivatives As Potential Inhibitors For 5-HT2A Receptor. *Biomedical Informatics, 13*(10), 339-342.

Periwal, V., Rajappan, J. K., Jaleel, A. U., & Scaria, V. (2011). Predictive models for anti-tubercular molecules using machine learning on high-throughput biological screening datasets. *BMC Research Notes*.

Rees, E. V. (2017). *Pandas and NumPy arrays explained*. Obtenido de Medium: https://medium.com/@ericvanrees/pandas-series-objects-and-numpy-arrays-15dfe05919d7

Schneider, J. (1997). *Cross Validation*. Obtenido de Carnegie Mellon University - School of Computer Science: https://www.cs.cmu.edu/~schneide/tut5/node42.html

Scientific, D. (2013). *Fingerprints*. Obtenido de Dalke Scientific: http://www.dalkescientific.com/writings/NBN/fingerprints.html

Shung, K. P. (2018). *Accuracy, Precision, Recall or F1?* Obtenido de Towards Data Science.

Soni, D. (s.f.). Obtenido de Towards Data Science.

Tropsha, A. (2010). Best Practices for QSAR Model Development, Validation And Exploitation. *Molecular Informatics, 29*, 476-488.

UniProtKB - P28223 - 5-hydroxytryptamine receptor 2A. (s.f.). Obtenido de UniProt: https://www.uniprot.org/uniprot/P28223

Wacker D., W. S. (2017). Crystal structure of an LSD-bound human serotonin receptor. *Cell*, 377-389. Obtenido de https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5289311/pdf/nihms839215.pdf

Wallach, I., & Heifets, A. (2018). Most Ligand-Based Classification Benchmarks Reward Memorization Rather Than Generalization. *Journal Of Chemical Information And Modeling*, 916-932.

Xu, X., Wei, Y., Guo, Q., Zhao, S., Liu, Z., Xiao, T., . . . Wang, K. (2018). Pharmacological Characterization of H05, A Novel Serotonin And Noradrenaline Reuptake Inhibitor With Moderate 5-HT2A Antagonist Activity For The Treatment Of Depression. *Journal Of Pharmacology And Experimental Therapeutics*, 624-635.