Estudio comparativo de modelos de machine learning para la detección de dianas microARN

Jordi Martínez Rodríguez
Máster universitario en Bioinformática y bioestadística UOC-UB
TFM-Estadística y Bioinformática 11

Nombre Consultor/a: Albert Pla Planas

Nombre Profesor/a responsable de la asignatura: Jose Antonio Morán Moreno / Carles Ventura Royo

Fecha Entrega: 02/01/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Estudio comparativo de modelos de machine learning para la detección de dianas microARN*
Nombre del autor:	*Jordi Martínez Rodríguez*
Nombre del consultor/a:	*Albert Pla Planas*
Nombre del PRA:	*Jose Antonio Morán Moreno / Carles Ventura Royo*
Fecha de entrega (mm/aaaa):	01/2018
Titulación::	Máster universitario en Bioinformática y bioestadística UOC-UB
Área del Trabajo Final:	*TFM-Estadística y Bioinformática 11*
Idioma del trabajo:	*Castellano*
Palabras clave	*Computer science, Machine learning, microRNAs,*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Los microARN son pequeñas moléculas de ARN de una longitud aproximada de 23 nucleótidos, de cadena sencilla y no codificantes que actúan en la regulación post-transcripcional inhibiendo la expresión génica uniéndose a sitios específicos de ARNm diana. En este documento se ha evaluado la viabilidad de distintos modelos basados en machine learning existentes para la predicción de las dianas de miARN. Se evaluó la viabilidad de seis modelos distintos: *k-nearest neighbors*, Naive Bayes, *artificial neural network*, *support vector machine*, *decision tree* y *random forest*. Para el entrenamiento de los modelos se construyó una base de datos con más de 14.000 interacciones de miARN-ARNm con 19 *features* de las mismas, de las cuales, se seleccionaron 3.173 interacciones reales y 3.173 interacciones no reales. El modelo *random forest* presento la mayor eficiencia en la predicción de interacciones, a la vez que fue uno de los modelos más sólidos y más sencillos de entrenar. Por las características de los modelos basados en *random forest* y debido a la utilización de un gran número de *features* para la predicción de dianas de miARN se recomienda el uso de estos modelos para futuros estudios de detección y predicción de dianas de miARNs.

Abstract (in English, 250 words or less):

MicroRNAs (miRNA) are small, non-coding RNA molecules of an approximate length of 23 nucleotides, which are involved in post-transcriptional regulation by inhibiting genetic expression by binding to specific target mRNA sites. The present document evaluates the viability of different existing machine learning models for the prediction of miRNA targets. The viability of six different models was examined: k-nearest neighbors, Naive Bayes, artificial neural network, support vector machine, decision tree and random forest. For model training, a database was built with more than 14000 miRNA-mRNA interactions with 19 features, from which 3173 real and 3173 unreal interactions were selected. The random forest model showed the highest efficiency in the prediction of interactions, and was as well one of the most solid and simple to train. Considering the characteristics of models based in random forest and due to their ability to use a great number of features for miRNA target predictions, these models are recommended for future studies of detection and prediction of miRNA targets.

Índice

1. Introducción 8
1.1. Contexto y Justificación del Trabajo 8
1.1.1. Descripción general 8
1.1.2. Justificación del TFM 9
1.2. Objetivos 9
1.2.1. Objetivos generales 9
1.2.2. Objetivos específicos 9
1.3. Enfoque y método a seguir 10
1.4. Planificación del trabajo 10
1.4.1 Tareas 10
1.4.2. Calendario 11
1.5 Breve sumario de productos obtenidos 12
1.6 Breve descripción de los otros capítulos de la memoria 12
2. Los microARNs 13
2.1. Descubrimiento de los microARNs 13
2.2. Biogénesis y procesamiento 14
2.3. Métodos de represión génica 15
2.4. Detección de dianas 16
3. Herramientas bioinformáticas de *machine learning* 18
4. Materiales y métodos 21
4.1. Base de datos 21
4.2. Selección de modelos 23
4.2.1. Naive Bayes 23
4.2.3. Support vector machine 27
4.2.4. Decision tree 29
4.2.5. Random Forest 30
4.2.6. k-Nearest Neighbors 31
4.3. Construcción y entrenamiento de los modelos 32
4.4. Métricas de evaluación de modelos 33
4.4.1 Variables de efectividad 33
4.4.2. Comparación de modelos 35
5. Resultados 35
5.1 Preparación de la base de datos 35
5.2. Elección de parámetros 36

5.3. Valoración de los modelos 38
5.4. Comparación de modelos 44
6. Discusión 50
7. Conclusión 51
8. Glosario 52
9. Referencias 53
10. Material suplementario 56

Lista de figuras

Figura 1: Tabla de la planificación del TFM 11
Figura 2: Diagrama de Grantt de la planificación del TFM 12
Figura 4: Biogénesis de los miRNA 14
Figura 3: Estructura en *hairpin stem-loop* de un pre-miARN 14
Figura 5: Interaccion miARN-ARNm diana 15
Figura 6: Esquema de la estructura secundaria de dúplex miARN-ARNm. 22
Figura 7: Diagrama de red de una ANN 24
Figura 8: Tipos de funciones de umbral 25
Figura 9: Diagrama de la estructura de una ANN 26
Figura 10: División del espacio por hiperplanos 27
Figura 11: Cambio de dimensiones mediante la utilización de un kernel gaussiano 28
Figura 12: Estructura jerárquica en forma de árbol de un decisión tree 29
Figura 13: Gráfico de valores de *accuracy* en función de *k* 36
Figura 14: Estructura de la *artificial neural network* construida. 36
Figura 15: Gráfico de valores de *accuracy* en función del número de *boosting* 37
Figura 16: Gráfico de importancia de las *features* 38
Figura 17: Distribución de las principales variables de eficiencia del modelo k-nn 39
Figura 18: Curva ROC del modelo k-nn 39
Figura 19: Distribución de las principales variables de eficiencia del modelo k-nn 40
Figura 20: Curva ROC del modelo Naive Bayes 40
Figura 21: Distribución de las principales variables de eficiencia del modelo ANN 41
Figura 22: Curva ROC del modelo ANN 41
Figura 23: Distribución de las principales variables de eficiencia del modelo SVM 42
Figura 24: Curva ROC del modelo SVM 42
Figura 25: Distribución de las principales variables de eficiencia del modelo *decision tree* 43
Figura 26: Curva ROC del modelo *decision tree* 43
Figura 27: Distribución de las principales variables de eficiencia del modelo *random forest* 44
Figura 28: Curva ROC del modelo *random forest* 44
Figura 29: Distribución de la variable *accuracy* en los modelos estudiados. 45
Figura 30: Distribución de la variable *sensitivity* en los modelos estudiados. 45
Figura 31: Distribución de la variable *specificity* en los modelos estudiados. 45
Figura 32: Distribución de la variable kappa en los modelos estudiados. 46
Figura 33: Distribución de la variable PPV en los modelos estudiados. 46
Figura 34: Distribución de la variable NPV en los modelos estudiados. 46
Figura 35: Curva ROC de todos los modelos 47

1. Introducción

1.1. Contexto y Justificación del Trabajo

1.1.1. Descripción general

Los microARN (miRNA) han sido catalogados como una de las moléculas más importantes que podemos encontrar relacionadas con regulación de la expresión génica. Estas moléculas son cadenas pequeñas de ARN no codificante, de 21-23 nucleótidos, que uniéndose a moléculas de ARN mensajero (mRNA), regulan el procesamiento post-transcripcional. Son capaces de silenciar por degradación o por represión la síntesis de proteínas.

Las moléculas de microARN se descubrieron en los años setenta por Lee, Feinbaum y Ambros [1]. Estos autores estaban interesados en un mutante de lombriz (*C. elegans*) lin-4. Gracias a la clonación de este mutante consiguieron secuenciar un fragmento de 700pb. A pesar de los experimentos realizados con este fragmento no fueron capaces de eliminar su función. Fue entonces cuando se dieron cuenta de que lin-4 no podía codificar para ninguna proteína. La clonación de ADN complementario (ADNc) y el posterior análisis de las secuencias han permitido la identificación de más de un ciento de miARNs en una gran diversidad de especies, desde nemátodos hasta mamíferos.

Estas moléculas participan en procesos que regulan el desarrollo embrionario y tisular (especialmente hematopoyesis y desarrollo neural), la apoptosis, proliferación y diferenciación celular. También están involucrados en la respuesta al estrés, la angiogénesis y ciertos trastornos cardiovasculares; además de que participan en la iniciación y progresión del cáncer y pueden funcionar como oncogenes o supresores tumorales.

Conocemos que más del 30% de todos los genes humanos están regulados por mecanismos dependientes de miARNs y que un solo miARN es capaz de regular diferentes transcriptos que pueden funcionar en diferentes vías celulares, así como que un mismo ARNm puede ser regulado por diversos miARNs.

Gracias a los avances en sistemas bioinformáticos hemos sido capaces superar las dificultades de identificación de genes y moléculas ARNm diana de los miARN y así predecir cada vez con más exactitud los sitios de unión de los miARN.

El principal obstáculo en la investigación de los miARN es detectar las secuencias específicas de los genes diana donde las interacciones miARN-ARNm son total o parcialmente complementarias y descubrir como los miARNs son capaces de reconocer esas secuencias. Además, la complementariedad parcial de las interacciones causa que el número de genes diana aumente para una única molécula de miARN. Para solucionar estos obstáculos utilizamos métodos bioinformáticos.

La predicción de dianas de miARN mediante modelos de *machine learning* es un proceso difícil dado el excesivo número de genes y la cantidad de información. Los diferentes algoritmos para estas predicciones generan resultados diversos creando pocos resultados comunes entre ellos. La sensibilidad y especificidad de los algoritmos actuales varían entre ellos además de que muy pocos algoritmos son capaces de integrar los datos resultantes de diferentes estudios de las funciones de los miARN.

En este trabajo se estudiarán los modelos bioinformáticos basados en *machine learning* que se utilizan en la comunidad científica a la hora de predecir las dianas de estos miARN. Se evaluará la capacidad de predecir de estos modelos y se intentará llegar a una comparación de los mismos para descubrir qué modelos predicen de forma más exacta y eficiente estas dianas. También se estudiará aplicar nuevos modelos de *machine learning*, todavía no utilizados por la comunidad, para intentar conseguir nuevos modelos que puedan aportar métodos de predicción más precisos.

1.1.2. Justificación del TFM

Durante la realización del máster en bioinformática y bioestadística hemos trabajado con algoritmos de clasificación de *machine learning* a la hora de realizar predicciones y de forma independiente también aprendimos a trabajar con secuencias de genes para conocer toda la información que podríamos necesitar para nuestros futuros estudios. La elección del tema presentado en este trabajo se caracteriza por mezclar dos frentes de estudio que normalmente se han presentado de forma separada durante el máster. Con el trabajo dedicado a los miARN, será posible aplicar los algoritmos de *machine learning* a la genética y así empezar a unir todos los conceptos aprendidos en un único trabajo.

Los métodos bioinformáticos actuales utilizan para la predicción de dianas, principalmente, la complementariedad de secuencias o cálculos de estabilidad termodinámica (energía libre) y la conservación evolutiva para determinar la probabilidad de la interacción miARN-ARNm. Con este trabajo se pretenderá también trabajar con toda la información disponible de las interacciones de forma conjunta para llegar a predecir las dianas.

1.2. Objetivos

1.2.1. Objetivos generales

- Evaluar la viabilidad de los distintos modelos basados en *machine learning* existentes para la predicción de las dianas de los miARN.

- Cuantificar y evaluar el funcionamiento de los métodos viables para la predicción de las dianas.

1.2.2. Objetivos específicos

- Obtener una base de datos que permita comparar distintos algoritmos de *machine learnig* para la predicción de dianas.

- Evaluar los modelos que se utilizan actualmente para la predicción de dianas.

- Conseguir nuevos modelos basados en *machine learning* para la predicción de dianas de miRNA.

- Definir un *benchmark* para la comparación de algoritmos de *machine learning.*

1.3. Enfoque y método a seguir

Para la realización de este trabajo es necesario un estudio exhaustivo de los modelos utilizados por la comunidad científica para la predicción de dianas de los miARN. Necesitaremos conocer aquellos métodos que utilicen *machine learning* para realizar la predicción y qué características y problemas contienen estos modelos.

Una vez tengamos los modelos localizados y clasificados será necesaria la creación de una base de datos. Las bases de datos existentes actualmente ya están preparadas para ser utilizadas en modelos concretos. Cada base de datos está preparada para ser utilizada en uno o varios modelos, por lo tanto, no existe una base de datos que pueda ser utilizada en todos los modelos. Una opción sería utilizar diferentes bases de datos para los diferentes modelos pero los resultados serían difícilmente comparables, por lo tanto en este trabajo se ha decidió crear una única base de datos para utilizarla con el máximo número de modelos posible. Esta base de datos contendrá las secuencias de los miARN y las secuencias dianas ARNm además de diferentes características (*features*) de la interacción miARN-ARNm. Una vez construida la prepararemos para entrenar los modelos.

Para entrenar los modelos utilizaremos el entorno y lenguaje de programación R. R se distribuye gratuitamente facilitando su uso. Principalmente elegiremos esta plataforma por esta característica en contraposición a Matlab u otras plataformas. Además, R posee muchas funciones para análisis estadístico y creación de gráficos y aún más importante, contiene muchas librerías creadas por la comunidad enfocadas para *machine learning*. Las librerías creadas por la comunidad nos facilitarán el trabajo a la hora de estudiar un gran número de modelos, ya que gracias a esto, podremos centrarnos en el análisis de los resultados, y en intentar mejorar los modelos si es posible, y no en la implementación de los mismos.

En el entrenamiento de los modelos dividiremos la base de datos en dos conjuntos de datos, uno para entrenar el modelo (*train*) y otro para evaluarlo (*test*). Para evitar que esta partición pueda influenciar en la evaluación final del modelo utilizaremos la validación cruzada (*cross-validation*) y así, aseguraremos que los resultados son independientes de la partición

1.4. Planificación del trabajo

1.4.1 Tareas

- **Revisión bibliográfica**
  - Revisión bibliográfica y estudio de las características de los modelos utilizados por la comunidad científica para la predicción de dianas de los miARN.
  - Revisión bibliográfica de técnicas de *machine learning* i métodos no utilizados actualmente para la predicción de miARNs.

- Identificar librerías de R necesarias para realizar la implementación y comparativa de los modelos

❖ **Base de datos**

- Buscar una base de datos con interacciones miARN-ARNm y sus *features*.
- Buscar bases de datos con secuencias de miARN y su secuencia diana ARNm.
- Construir una base de datos completa y funcional para ser utilizada para entrenar los modelos.

❖ **Entrenamiento de los modelos**

- Entrenar los modelos y analizar los resultados
- Desarrollo del código, "*benchmark"* necesario para comparar los resultados y para automatizar la comparación.

❖ **Interpretar y discutir los resultados**.

❖ **Redacción de la memoria**

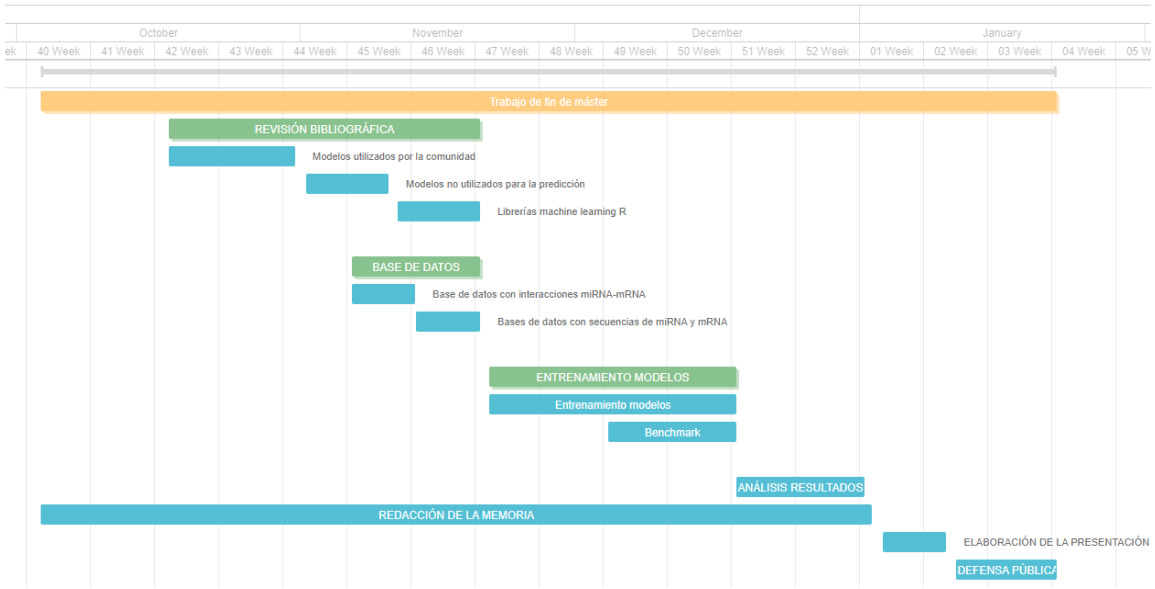
❖ **Elaboración de la presentación**

1.4.2. Calendario

Figura 1: Tabla de la planificación del TFM

TAREA	INICIO	FIN	DURACIÓN (dias)
REVISIÓN BIBLIOGRÁFICA	17/10/2017	20/11/2017	34
Modelos utilizados por la comunidad	17/10/2017	31/10/2017	14
Modelos no utilizados para la predicción	01/11/2017	10/11/2017	9
Librerías machine learning R	11/11/2017	20/11/2017	9
BASE DE DATOS	06/11/2017	20/11/2017	14
Base de datos con interacciones miRNA-mRNA	06/11/2017	13/11/2017	7
Bases de datos con secuencias de miRNA y mRNA	13/11/2017	20/11/2017	7
ENTRENAMIENTO MODELOS	21/11/2017	18/12/2017	27
Entrenamiento modelos	21/11/2017	18/12/2017	27
Benchmark	04/12/2017	18/12/2017	14
ANÁLISIS RESULTADOS	18/12/2017	01/01/2018	14
REDACCIÓN DE LA MEMORIA	03/10/2017	02/01/2018	91
ELABORACIÓN DE LA PRESENTACIÓN	03/01/2018	10/01/2018	7
DEFENSA PÚBLICA	11/01/2018	22/01/2018	11

Figura 2: Diagrama de Grantt de la planificación del TFM



1.5 Breve sumario de productos obtenidos

En los primeros pasos del comienzo del trabajo se consiguió y estudió una bibliografía amplia y variada sobre los miARN y la predicción de dianas de miARN. Se consiguió crear una base de datos funcional con más de 14.000 interacciones miARN-ARNm con 19 *features* de las mismas. Esta base de datos contenía casos donde las interacciones no existían (interacciones negativas) y casos donde las interacciones eran reales (interacciones positivas). A la conclusión de este trabajo también se obtuvo un código en R para la creación de esta base de datos, el cálculo de las *features* y el entrenamiento y evaluación de los modelos estudiados en este trabajo.

1.6 Breve descripción de los otros capítulos de la memoria

La memoria de este trabajo de fin de máster se divide en:

- Los microARN. Descripción detallada de la biología de los miARN y de las *features* que se utilizan habitualmente para la detección de dianas de miARN
- Herramientas bioinformáticas de *machine learning*. Pequeño resumen de diferentes herramientas creadas por la comunidad para la detección de dianas de miARN
- Materiales y Métodos. Descripción de los principales modelos elegidos para este estudio y pasos seguidos para la creación y comparación de los mismos
- Discusión. Comentarios de los resultados obtenidos y valoración de los modelos entrenados.
- Conclusiones. Valoración final de los objetivos planteados al inicio del trabajo.
- Referencias. Referencias y recursos bibliográficos utilizados.
- Anexo. Materiales adjuntos a la memoria.

2. Los microARNs

Los microARN son pequeñas moléculas de ARN endógeno (se sintetizan en la misma célula) de una longitud aproximada de 23 nucleótidos, de cadena sencilla y no codificantes (no tienen información para la codificación de proteínas) que actúan en la regulación post-transcripcional inhibiendo la expresión génica. Estas moléculas ejercen su función emparejándose con los ARNm de los genes que codifican proteínas para dirigir su represión. La represión de estos ARNm regulatorios conduce a una disminución de la eficacia de traducción y / o una disminución de los niveles de los mismos.

En la actualidad el estudio de los miARNs se ha convertido en uno de los focos principales de investigación en el campo de la biología. Aunque estas moléculas de ARN han sido descritas mayoritariamente en seres multicelulares tanto de origen vegetal como de origen animal y se pensó durante mucho tiempo que eran exclusivos de estos organismos recientemente se han descubierto en algas unicelulares [2] que podrían indicar que los miARN son probablemente evolutivamente más antiguos de lo que se imaginaba inicialmente

En el ser humano se han identificado más de 1500 moléculas de microARNs, que han sido descritas y almacenadas en la base de datos miRBase, cifra que a día de hoy continúa aumentando. Los microRNAs son codificados a partir de regiones intergénicas o bien a partir de intrones de genes que codifican para proteínas. Los miRNAs en forma de complejos con proteínas específicas dirigen las actividades efectoras de estos complejos denominados RISC, "*RNA-induced silencing complex*".

2.1. Descubrimiento de los microARNs

El descubrimiento de los miARNs surgió en los años setenta de la estrecha colaboración entre científicos, en concreto los tres científicos que descubrieron el primer miARN (lin-4) en *C. elegans* fueron Lee, Feinbaum y Ambros [1]. Estos autores comenzaron a investigar un mutante de lombriz lin-4. El grupo de Ambos consiguió clonar el mutante lin-4 en 1988 consiguiendo así un fragmento de 700 pares de bases (pb). Intentaron encontrar una pauta abierta de lectura en el fragmento pero no lo consiguieron y a pesar de la realización de numerosos experimentos no consiguieron tampoco eliminar la función de lin-4. Esto los llevó a considerar que lin-4 no podría codificar para ninguna proteína. A pesar de no conseguir eliminar su función, comprobaron que era esencial para la sincronización del desarrollo postembrionario de los gusanos. En aquel momento se conocía que la regulación génica era llevada a cabo por proteínas pero no por moléculas de ARN. En 1992 el grupo de Ambros confirmó que el producto génico de lin-4 era de una molécula de 22 pb.

Al mismo tiempo que el grupo de Ambos realizaba sus investigaciones con lin-4 el grupo de Ruvkun [3] mapeó la mutación lin-14. Los dos grupos se intercambiaron las secuencias de lin-4 y lin-14 y al mismo tiempo reconocieron la complementariedad antisentido de las dos secuencias en la región no traducida 3' de lin-14.

En el año 2000 fue descubierto el segundo miARN, let-7, otra molécula implicada también en el desarrollo de *C. elegans* [4]. En aquel momento se comenzó a descubrir las secuencias genómicas de *D. melanogaster* y de *H. sapiens* y esto llevó a buscar homologías de la secuencia de let-7 en el genoma de estas especies. Estos estudios revelaron una fuerte conservación filogenética de let-7 en estas dos especies y en

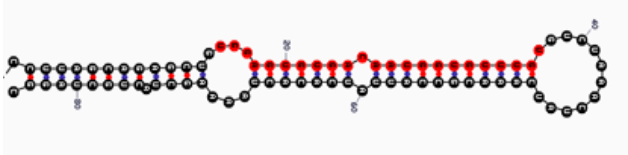
muchas otras. El descubrimiento de let-7 provocó una fuerte búsqueda e identificación de nuevos miARNs en otras especies.

2.2. Biogénesis y procesamiento

Los genes que codifican miARNs son mucho más largos que los miARNs procesados maduros. Algunos de estos genes se encuentran distribuidos a lo largo del genoma, pero un gran número se encuentra de forma adyacente formando grupos y se expresa de forma coordinada. La regulación y la transcripción de estos genes se realizan mediante la síntesis de un transcrito policistrónico que se fragmenta para generar múltiples miARN.

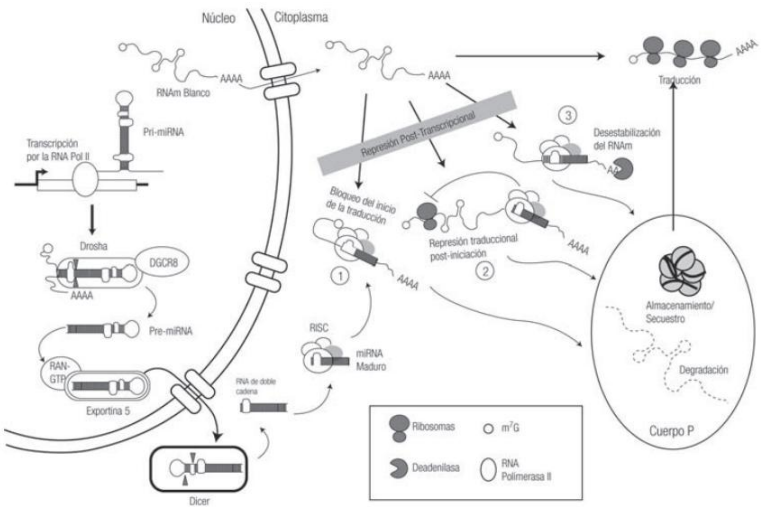
Los precursores de los miARN se encuentran localizados dentro de exones e intrones de ARN no codificante además de en intrones de ARN codificante. Históricamente, estas regiones se conocían como "ADN basura", ya que se desconocía su función. El descubrimiento de los miARNs implicó que el "ADN basura" no era tan inútil como se pensaba originalmente. Los precursores de miARNs se encuentran con menos frecuencia dentro de los exones de las transcripciones y en las transcripciones antisentido.

Figura 3: Estructura en *hairpin stem-loop* de un pre-miARN. En rojo secuencia del miARN maduro. Tomada de http://mirnamap.mbc.nctu.edu.tw/html/tutorial.html



La vía de biosíntesis de los miARN está formada por varias etapas (fig 4). En la primera fase, los miARNs son transcritos por la RNA polimerasa II para generar moléculas precursoras o pri-miARN con una caperuza en 5' y una cola de poli-adeninas (poly-A) en 3', y se procesan en estructuras cortas de 70 nucleótidos en forma de tallo-bucle (*stem-loop*) conocidas como pre-miARN (fig. 3) en el núcleo celular por el complejo proteico denominado “microprocesador” que consta de la nucleasa Drosha y la proteína de unión a ARN de doble hélice Pasha.

Figura 4: Biogénesis de los miRNA y mecanismos de regulación de expresión génica. Tomado de Lugo-Trampe et al. [5].



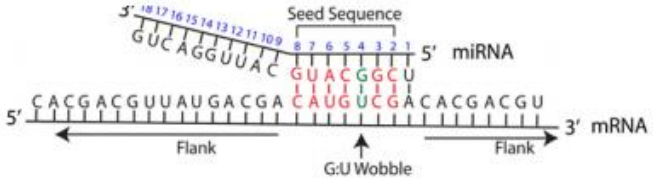
Después de la etapa de procesamiento en el núcleo, los pre-miARN son reconocidos por el factor nuclear de exportación Exportina-5 (Exp5), el que junto con la proteína de unión a GTP Ran forman un complejo de transporte nuclear que conduce a los pre-miARN a través de los poros del núcleo al citoplasma. Los pre-miARN son luego procesados a miARN maduros en el citoplasma mediante la interacción con la ribonucleasa Dicer. Esta enzima corta el tallo y bucle del pre-miARN para generar un ARN dúplex formado por una cadena de miARN maduro y una cadena de miARN complementaria de ~22 nucleótidos de longitud. En seguida, el miARN maduro es incorporado al "*RNA-induced silencing complex*", RISC [6].

Este complejo RISC está compuesto por varias proteínas, entre ellas Dicer, TRBP (*transactivation-response element RNA-binding protein*) y, principalmente, por las proteínas Argonauta. Las proteínas Argonauta se encuentran en regiones específicas del citoplasma denominadas *P-bodies*, que pueden ser definidos como sitios de almacenamiento de mARN. Este complejo RISC se ensambla con la proteína Argonauta, para formar el "*miRNA-induced silencing complex*" (miRISC), el cual selecciona la cadena madura o guía del ARNm.

2.3. Métodos de represión génica

La unión entre una secuencia de miARN y su diana es idéntica al emparejamiento entre moléculas de ADN y ARN. Este proceso puede o no ser canónico, por lo tanto pueden existir bases que queden desemparejadas.

Figura 5: Interaccion miARN-ARNm diana. En azul se muestra la posición del nucleótido en el miARN. La secuencia semilla del miARN se consideran los nucleótidos de la posición 2 a la 8. En rojo las uniones perfectas. En verde se muestran las uniones "*wobble*". Tomado de Peterson et al. [7].



Los miARNs se suelen unir a la región 3' UTR del tránscrito, donde inician la degradación del mARN o inhiben la traducción. El silenciamiento mediado por miARNs se lleva a cabo mediante inhibición de la traducción o por degradación dependiente del grado de complementariedad entre el miARN y la región 3'UTR del tránscrito. La utilización de un mecanismo u otro dependerá del ARNm diana.

Una vez que el miARN forme parte del complejo RISC, si la complementariedad de la interacción miARN-ARNm es elevada o total se producirá la degradación del ARNm [8], mientras que si la interacción no es total solo se producirá la interrupción e inhibición de la traducción del ARNm. Actualmente se considera que para un nivel de afinidad elevada son necesarios 11 pb en la interacción miARN-ARNm. Doench y Sharp [9] introdujeron que la habilidad de los miARNs para suprimir la traducción dependía enormemente de la energía libre de unión entre los primeros 8 nucleótidos del extremo 5' del miARN y el ARNm diana. Estudios con microarrays mostraron que los miARNs podían inhibir dianas donde sus secuencias fueran complementarias en las posiciones 2-7 del extremo 5' de un miARN. Esta región se conoce como *seed sequence* [10]. La *seed sequence* corresponde a las primeras 8 bases de la interacción y son especialmente importantes para reconocer la diana. Si la interacción

entre las dos moléculas se realiza en esta región se considera que la diana es canónica pero también existen regiones donde puede existir una interacción fuera de la *seed sequence* entre el miARN y el ARNm. Si se producen este tipo de interacciones se considerara que la diana no es canónica

Después de la degradación del ARNm, el miARN puede continuar generando interacciones adicionales y continuar destruyendo ARNm. La complementariedad se puede dar tanto en regiones codificantes como en regiones 3' UTR mientras que cuando actúa inhibiendo la traducción suele unirse a la región 3' UTR del ARNm.

El elemento principal del mecanismo de inhibición traduccional es RCK, uno de los componentes del complejo RISC. Este componente puede iniciar un evento de oligomerización que secuestra el ARNm y lo transporta a los puntos citoplasmáticos *P-bodies* que contienen complejos proteicos. Una vez allí el ARNm inhibido puede permanecer almacenado o podría ser degradado mediante proteínas *cap-binding* o por complejos con enzimas *decapping*.

2.4. Detección de dianas

En el estudio para la detección de dianas objetivo para los miARNs se descubrió que existe una gran cantidad de sitios de unión para un solo miARN. Además de este factor el proceso de validación de posibles dianas en un laboratorio conlleva mucho tiempo de investigación y tiene un coste muy elevado. Dados estos problemas se consideró un enfoque computacional para la investigación de dianas. El método computacional para la detección de dianas facilita el proceso y reduce el número de los posibles sitios de unión para la validación experimental.

Los métodos computaciones utilizan para la predicción y detección de posibles dianas la selección de características o *features* que aparecen en la formación de la interacción miARN-ARNm y que contienen un gran valor predictivo. En la actualidad existen cuatro grupos de *features* principales para la predicción de dianas: la coincidencia de las semillas, la conservación entre especies, la energía libre y la accesibilidad del sitio de unión.

Coincidencia de semillas

Las secuencias semilla o *seed sequence* se definen como los primeros nucleótidos (2-8) del extremo 5' de la secuencia del miARN. Una coincidencia de semillas se considera una coincidencia de Watson-Crick (WC) entre la secuencia de un miARN y la secuencia del ARNm diana en la *seed sequence*, por lo tanto, se considera coincidencia cuando una adenosina (A) se empareja con un uracilo (U) y una guanina (G) se empareja con una citosina (C). Un emparejamiento perfecto no tiene errores en la alineación de sus secuencias.

Existen diferentes tipos de coincidencias iniciales [11], [12]:

- 6-mer: Coincidencia de semillas perfecta de 6 pb entre el miARN y el ARNm en la *seed sequence*
- 7-mer-m8: Coincidencia de semillas perfecta de 2-8 pb entre el miARN y el ARNm en la *seed sequence*

- 7mer-A1: Coincidencia de semillas perfecta de 2-7 pb entre el miARN y el ARNm en la *seed sequence* además de una A en el primer nucleótido del miARN.
- 8-mer: Coincidencia de semillas perfecta de 2-8 pb entre el miARN y el ARNm en la *seed sequence*, además de una A en el primer nucleótido de miARN.

Los emparejamientos de bases anteriores pertenecen a dianas canónicas donde la interacción se produce en la *seed sequence,* pero también existen otro tipo de interacciones fuera de esa *seed sequence* que pertenecerían a dianas no canónicas (compensatorias, centradas, etc.).

Conservación

La conservación se refiere al mantenimiento de una secuencia entre especies desde el punto de vista evolutivo. La conservación es una característica que analiza las regiones de la secuencia del miARN y se compara con las mismas regiones en otras especies. Se ha encontrado que la *seed secuence* está más conservada que otras regiones. El análisis de la conservación de las regiones genómicas puede proporcionar la evidencia de que una diana de miARN predicha es funcional, ya que se está seleccionando y conservando y por lo tanto es altamente probable que tenga un rol en el desarrollo celular.

Energía libre

La energía libre es una medida de la estabilidad de un sistema biológico. En el estudio de las interacciones de miARN-ARNm, se utiliza la medida de la *minimum free energy* (o energía libre de Gibbs) que indica la estabilidad de unión de la interacción. Es más probable que se confirme que una diana es una diana verdadera de un miARN si se consigue predecir que la unión de un miARN a un mARN diana es un candidato estable. Existe una elevada dificultad para medir la energía libre directamente, así que se considera el cambio en la energía libre durante una reacción (ΔG). Los sistemas con mayor estabilidad son aquellos que presentan un ΔG negativo ya que tendrán menos energía disponible para volver a reaccionar en el futuro.

Accesibilidad del sitio

La accesibilidad del sitio de unión es una medida que indica la facilidad que presenta un miARN para localizar e hibridarse con una diana. El ARNm después de la transcripción se pliega sobre sí mismo formando una estructura secundaria. Una vez se forme esta estructura puede haber sitios de unión que estén más expuestos y sean más accesibles para un miARN o también puede que el ARNm no esté plegado sobre sí mismo con lo que sus bases serán accesibles para el miARN. Existe una tercera posibilidad en la cual el pliegue de la estructura secundaria no sea estable, y por lo tanto el miARN puede interferir y acceder a sitio de unión. Para poder evaluar la probabilidad de que un ARNm sea diana de un miARN se puede medir la cantidad de energía necesaria para hacer que el sitio de unión sea accesible a un miARN.

Existen otras muchas *features* que pueden ser utilizadas para la predicción de dianas:

-Abundancia de sitos de unión diana
-Contenido de AU en la *seed secuence*
-Contenido del emparejamiento GU en la *seed secuence*
-Emparejamiento 3' compensatorio
-Estabilidad de emparejamiento de semillas
-Posición de una secuencia diana dentro del mARN
-Etc.

En la actualidad aún no se conocen todos los elementos que podrían tener importancia en el desarrollo y funcionalidad de la interacción entre miARN y sus dianas, por lo que todavía hoy se continúan identificando e investigando nuevos descriptores.

3. Herramientas bioinformáticas de *machine learning*

En los primeros pasos, la identificación de dianas de miRNAs se realizaba a través del experimento seguido por la identificación de los sitios de unión potenciales de forma manual. Estos sitios de unión finalmente se confirmaban médiate mutagénesis dirigida al sitio de unión u otras técnicas. En la actualidad existe mucha información disponible sobre las propiedades de los miARNs como para poder reconocer un conjunto de características distintivas. Decenas de métodos computacionales y algoritmos se han desarrollado para la identificación de dianas de miRNA (tabla 1). Los programas basados en los patrones del emparejamiento de las bases son los más comunes, además de otras *features* comentadas anteriormente como la conservación evolutiva, la estructura secundaria del ARNm diana y la composición de nucleótidos de las secuencias. Estas *features* a menudo se agregan para aumentar la precisión de los métodos.

Tabla 1: Métodos computacionales par la predicción de dianas de microARN. Tomado de Min y Yoon [13].

Name	URL	Reference
DIANA-microT	http://diana.pcbi.upenn.edu/cgi-bin/micro_t.cgi	Kiriakidou *et al*., 2004
EIMMo	http://www.mirz.unibas.ch/EIMMo	Gaidatzis *et al*., 2007
miRanda	http://www.microrna.org	Enright *et al*., 2003
MirTarget2	http://mirdb.org	Wang and El Naqa, 2008
miTarget	http://cbit.snu.ac.kr/~miTarget	Kim *et al*., 2006
PicTar	http://pictar.mdc-berlin.de	Grün *et al*., 2005
rna22	http://cbcsrv.watson.ibm.com/rna22.html	Miranda *et al*., 2006
RNAhybrid	http://bibiserv.techfak.uni-bielefeld.de/rnahybrid	Rehmsmeier *et al*., 2004
TargetScan	http://genes.mit.edu/targetscan	Lewis *et al*., 2003
TargetScanS	http://genes.mit.edu/targetscan	Lewis *et al*., 2005

Cada uno de estos métodos construye una serie de clasificadores que pueden medir cómo un candidato de miARN es similar a los miARNs conocidos en base a varias características. TargetScan y TargetScanS, por ejemplo, son algoritmos desarrollados por Lewis et al. [11] que integran el modelado basado en la termodinámica de las interacciones miRNA-mRNA y el análisis de la secuencia comparativa para predecir objetivos de miRNA conservados en genomas de múltiples especies como humanos, ratones, ratas y pez globo. Otro ejemplo es PicTar [14], que comprueba las alineaciones 3' UTR para aquellas dianas que muestran coincidencia inicial con miARNs, filtra las alineaciones retenidas en función de su estabilidad termodinámica y calcula un puntaje de máxima verosimilitud de modelo de Markov para cada objetivo previsto. Como último ejemplo nombraremos miRanda [15]. Este método fue

desarrollado originalmente para predecir los genes diana de genes miARN en *D. melanogaster,* pero también se usó para predecir dianas de miARN humanos. Para cada miARN, miRanda seleccionó genes diana sobre tres propiedades: complementariedad de secuencia usando un algoritmo de alineación local, energías libres del dúplex ARN-ARN y conservación de sitios de unión diana en genomas relacionados.

Una forma alternativa de predecir los objetivos de miARN consiste en reunir todas las interacciones miARN-ARNm y calcular un cierto número de características o *features* que describan estas asociaciones para construir un modelo estadístico que se ajustaría a las mismas. En este trabajo nos centraremos en aquellos métodos creados por la comunidad científica que se basan es esta idea de método de predicción de dianas. Estos métodos de predicción se basan en modelos de *machine learning*.

El objetivo del *machine learning* (aprendizaje automático) es desarrollar algoritmos para los ordenadores de forma que estos sean capaces de generalizar comportamientos y conocer patrones comunes a partir de la información suministrada previamente. Llamamos aprendizaje supervisado a aquel en el cual los elementos con los que se intenta enseñar ya están asociados a un valor conocido. Por otro lado cuando no conocemos estos valores estamos ablando de aprendizaje no supervisado. En la estadística podemos ver el aprendizaje automático como un problema general de inferencia estadística: se parte de observaciones particulares y se llega a descripciones generales mediante la construcción de un modelo.

En la actualidad existen una gran numero de herramientas informáticas creadas por la comunidad que implementan modelos de *machine learning* para predecir dianas. A continuación se presentan algunas de estas herramientas y sus características

TargetSpy

TargetSpy fue desarrollado por Sturm et al. [16]. Esta herramienta informática utiliza un enfoque computacional para predecir sitios de unión en ARNm independientemente de la presencia de coincidencia de semillas. Esta herramienta utiliza un amplio espectro de características de la composición y estructura de las secuencias y de emparejamiento de las bases. A diferencia de los métodos de predicción no basados en *machine learning* no utiliza la conservación evolutiva para predecir, lo cual le permite detectar interacciones específicas de especies y hace que sea adecuado para analizar secuencias genómicas no conservadas.

Esta herramienta fue entrenada con diferentes modelos (SVM, Naive Bayes, C4.5, AdaBoost con C4.5 y MultiBoost con C4.5). Finalmente seleccionaron el modelo de *decision tree* C4.5 con MultiBoost ya que presentaba mejores resultados que el resto.

miTarget

miTarget fue desarrollado Kim et al. [17]. miTarget es un clasificador basado en modelos SVM para la predicción de genes diana. Utiliza un *kernel radial basis function* (RBF) como una medida de similitud para las características de SVM, clasificadas por características estructurales, termodinámicas y de posición de nucleótidos. Esta herramienta predijo funciones significativas de miARN en humanos.

NBmiRTar

NBmiRTar fue desarrollado por Yousef et al. [18]. Esta herramienta proporciona un método de predicción de dianas que no utiliza la conservación de la secuencia,

NBmiRtar utiliza un modelo de *machine learning* basado en un clasificador de Naive Bayes. NbmiRTar genera un modelo de predicción de dianas objetivo a partir de dianas validadas y ejemplos negativos generados artificialmente. Para la identificación de dianas se utilizaron las secuencias de los segmentos de "*seed*" como "*out-seed*" del dúplex miARN-ARNm.

SVMicrO

Liu et al. [19] buscaban construir un conjunto de datos de entrenamiento más rico y confiable y desarrollar un algoritmo que explotase adecuadamente este conjunto de datos para mejorar los algoritmos de predicción de dianas del momento. Para ello desarrollaron un nuevo algoritmo de predicción de dianas de miARN basado en SVM de 2 etapas llamado SVMicrO. El modelo está formado por tres pasos. En un primer paso, aplican un filtro para encontrar posibles lugares de unión del miARN. En un segundo paso Site-SVM extrae *features* de cada lugar de unión potencial y asigna puntos que indican la confianza de la predicción del sitio de unión con un sitio real. En el último paso la UTR-SVM considera los puntos del lugar de unión junto con otras *features* UTR para producir la predicción final del UTR como diana. Gracias a este sistema el modelo puede reducir el número de falsos positivos identificados por las reglas de coincidencia de semillas.

Este algoritmo hace la predicción basada en 21 *features* del sitio de unión y 18 *features* UTR, seleccionadas mediante el entrenamiento de una colección completa de 113 *features* del sitio y 30 UTR.

MiRTDL

MiTRDL fue desarrollado por Cheng et al. [20]. MiRTDL es un clasificador basado en "*the constraint relaxation method"* y "*the Convolutional Neural Network"* (CNN). El grupo logró una *specificity* de 96.44%, *sensitivity* de 88.43% y una *accuracy* de 89.98%. Para entrenar el modelo seleccionaron 20 *features* involucradas en la coincidencia de semilla, en la conservación y en las características de accesibilidad del sitio de unión objetivo del miARN.

MBSTAR

Bandyopadhyay et al. [21], utilizando un clasificador basado en un modelo de *random forest,* desarrollaron MBSTAR. MBSTAR presentaba un modelo de *Multiple-instance learning random forest* (MIL-RF) con 50 árboles en el bosque. En *machine learning*, el *Multiple-instance learning* (MIL) es una variación de aprendizaje supervisado. En lugar de recibir un conjunto de casos que están etiquetados individualmente, el modelo recibe un conjunto de paquetes etiquetados, cada uno, con muchos casos.

Este clasificador fue capaz de predecir los sitios de unión diana en un ARNm de los miRNA. Para el entrenamiento de MBSTAR seleccionaron las 40 características más relevantes de un conjunto con más de 350 características estructurales y de secuencias. Para este modelo se consideró seis técnicas diferentes de *Multiple instance learning* y se aplicó un *5-fold cross-validation*. Con este modelo Bandyopadhyay consiguió una *acurracy* del 78.24%.

deepTarget

deepTarget fue desarrollado por Lee et al. [22]. En su artículo propone un modelo de predicción basado en auto-codificación profunda basada en redes neuronales

recurrentes (RNN) para la predicción de dianas de miARN. Para el entrenamiento de su modelo utilizaron 4.735 interacciones positivas y 1.225 interacciones negativas.

Gracias a esta combinación de enfoques de aprendizaje supervisados y no supervisados, deepTarget elimina la necesidad de extracción manual de características.

TarPmiR

Ding et al. [23] aplicaron cuatro enfoques de *machine learning* diferentes a sus datos. Consiguieron identificar siete nuevas *features* de sitios dianas de miARN, que al combinarlas con las *features* comúnmente utilizadas por los algoritmos de predicción de objetivos de miARN existentes, desarrollaron un enfoque llamado TarPmiR para la predicción del sitio de dianas de miARN. Al probar en dos conjuntos de datos humanos y de un ratón, demostraron que TarPmiR podía predecir más del 74.2% de los sitios diana de miARN en cada conjunto de datos.

4. Materiales y métodos

4.1. Base de datos

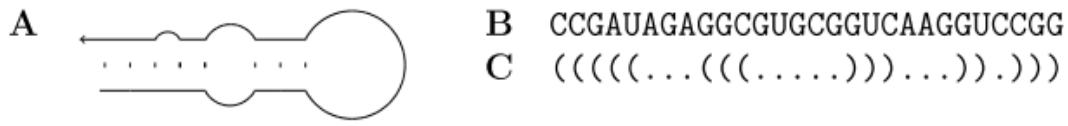
Nuestro objetivo es la comparación de modelos basados en *machine learning* para la predicción de dianas de miARNs. Para poder realizar una comparación efectiva de los modelos es necesaria la construcción de una base de datos común para todos los modelos y entrenarlos con ella. En este trabajo se construyó una base de datos con interacciones miARNs-ARNm positivas y negativas.

La interacciones miARN-ARNm y las secuencias de los mismos fueron obtenidas del artículo de Lu y Leslie [24]. En este artículo se presenta un modelo para predecir las interacciones miARN-ARNm basándose únicamente en las secuencias, mapeando las interacciones de las proteínas Argonauta con los miARN con técnicas de *machine learning*, en concreto un modelo basado en SVM. Su base de datos contenía una lista de miARN y sus genes dianas con sus secuencias. Además esta base de datos contenía dos tipos de interacciones miARN-ARNm, unas donde esta interacción no existía y otras donde la existencia de esta interacción había sido comprobada previamente por el grupo de Lu y Leslie [24]. Elegimos esta base de datos debido a que ya nos proporcionaba las transcripciones del sitio de unión del ARNm y un número significativo de interacciones positivas, pero aún más importante, interacciones negativas, ya que existe una dificultad añadida a la hora de buscar una muestra de interacciones negativas.

Una vez se filtró la base de datos de Lu y Leslie, eliminado los miARN de ratón y considerando solo un miARN por gen, se obtuvo la estructura secundaria del dúplex miARN-ARNm para el posterior cálculo de las *features* necesarias para entrenar el modelo. Para conseguir la estructura dúplex se utilizó el paquete ViennaRNA [25], [26], más concretamente el programa ViennaRNA Cofold. El paquete ViennaRNA ha sido muy utilizado por toda la comunidad científica. Este paquete consiste en una biblioteca de códigos en C y varios programas independientes para la predicción y comparación de estructuras secundarias de ARN.

ViennaRNA Cofold nos proporciona la estructura secundaria con *bracket notation* (fig. 6). La estructura secundaria con *bracket notation* nos muestra una secuencia de paréntesis y puntos con la cual podemos saber qué nucleótidos están unidos formando pares de bases en la estructura en forma de dúplex de la interacción. Los paréntesis abiertos hacia la derecha nos muestran qué nucleótidos, en sentido ascendente de la secuencia, se unirán con los nucleótidos marcados con un paréntesis abierto hacia la izquierda en un sentido descendiente del dúplex. Los puntos nos indican qué nucleótidos no han sido complementados en el dúplex y que acabaran formado *loops* o bultos en la estructura secundaria.

Figura 6: Esquema de la estructura secundaria de dúplex miARN-ARNm (A) y braket notation (B y C).



Gracias a esta anotación podemos conocer y realizar el cálculo de las *features* de coincidencia de semillas propias de cada interacción. En este trabajo se calcularon 19 *features* con las cuales se han entrenado cada uno de los modelos.

- **Coincidencia de semilla y estructura de la secuencia**
  - Pares de bases encajadas en los primeros 8 nucleótidos de la semilla
  - Pares de bases encajadas en toda la secuencia del miARN
  - Pares de bases no encajadas en toda la secuencia del miARN
  - Pares de bases antes de un codón de parada
  - Porcentajes de A,U,G y C en la secuencia del miARN
  - Porcentajes de pares de bases AU, CG y GU
  - Ratio de pares de bases GC
  - Pares de bases seguidas en la toda la secuencia del miARN
  - Pares de bases seguidas en la semilla del miARN
  - Pares de bases seguidas fuera de la semilla del miARN
  - Posición del primer par de bases encajadas en el miARN
  - Posición del último par de bases encajadas en el miARN

- **Energía libre**

- **Energía de accesibilidad del sitio**

La *feature* de la energía libre de Gibbs también la obtenemos gracias al programa ViennaRNA Cofold que, por cada una de las estructuras secundarias que proporciona, también adjunta la información de la energía libre de Gibbs que presenta y nos sirve como indicador de la fortaleza del vínculo de la interacción miARN-ARNm. La energía

de accesibilidad del sitio de unión, en cambio, la obtenemos de otro programa que contiene el paquete ViennaRNA, ViennaRNA fold.

4.2. Selección de modelos

Para la realización de este trabajo se seleccionaron 5 modelos que aparecen de forma regular en herramientas bioinformáticas creadas por la comunidad científica para la predicción de dianas de miRNA, además de ser utilizados también de forma habitual para el estudio de la predicción de nuevos precursores de miRNA a partir de miRNA ya conocidos [27], [28]. Además se seleccionó un modelo del cual no se encontró ninguna referencia o herramienta que lo incorporara para predecir dianas de miRNA (*k-nearest neighbor*s). Todos estos modelos utilizados están considerados dentro del campo de la predicción por *machine learning*.

4.2.1. Naive Bayes

Los clasificadores basados en métodos Bayesianos (uso del Teorema de Bayes) utilizan datos de entrenamiento para calcular la probabilidad de cada resultado en base a la evidencia provista por los valores de las características. Cuando el clasificador se aplica luego a datos no etiquetados, usa las probabilidades observadas para predecir la clase más probable para las nuevas características.

El principal objetivo consiste en determinar a qué clase, dentro del conjunto C = (c1, c2, …, c*m*), pertenece un elemento descrito por un vector de características x = (x1, …, x*n*)

Por lo general, los clasificadores bayesianos se aplican mejor a problemas en los que los casos a clasificar presentan un gran número de atributos que deben considerarse simultáneamente para estimar la probabilidad general de un resultado. Si bien muchos algoritmos de aprendizaje automático ignoran las características que tienen efectos débiles, los métodos bayesianos utilizan todos los recursos disponibles para cambiar sutilmente las predicciones. Pero en contraposición, si una gran cantidad de características tienen efectos secundarios, en conjunto, su impacto combinado podría ser bastante grande y producir predicciones erróneas.

En la tabla 2 se muestran las fortalezas y debilidades del modelo. La implementación de Naive Bayes se realizó en R mediante el paquete “e1071”. Esta librería fue desarrollada Meyer et al. [29].

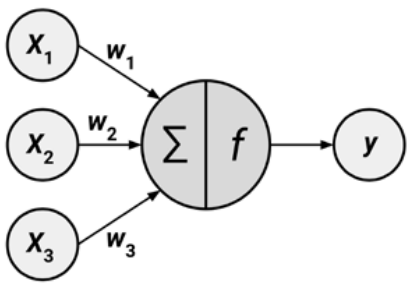
Tabla 2: Tabla de fortalezas y debilidades de los modelos de Naive Bayes. Tomado de Lantz [30].

Fortalezas	Debilidades
• Simple, rápido y muy efectivo • Es tolerante al ruido y a la ausencia de datos • Requiere pocos ejemplos para entrenamiento, pero también funciona bien con un gran número de ejemplos	• Se basa en la suposición de que los atributos son independientes y que tienen una importancia similar, la cual no siempre se cumple de forma estricta • No es ideal para conjuntos de datos con muchas características numéricas. • Las probabilidades estimadas son menos confiables que las clases predichas

4.2.2. Artificial neural network

Las redes de neuronas artificiales o *artificial neural networks* (ANN) constituyen una técnica de aprendizaje automático inspirada en el comportamiento del cerebro humano. Se crea una red con diferentes capas interconectadas para procesar la información. Cada capa está formada por un grupo de nodos que transmite la información a los otros nodos de las capas siguientes. El modelo de una sola neurona artificial se puede entender en términos muy similares al modelo biológico. Como se muestra en la figura 7, un diagrama de red dirigido define una relación entre las señales de entrada recibidas por las dendritas (variables *x*) y la señal de salida (variable *y*). Al igual que con la neurona biológica, la señal de cada dendrita se pondera (valores *w*) de acuerdo con su importancia. Las señales de entrada son sumadas por el cuerpo de la célula y la señal se transmite de acuerdo con una función de activación indicada por *f*.

Figura 7: Diagrama de red de una ANN. Tomado de Lantz [30].



Una red neuronal artificial se caracteriza por:

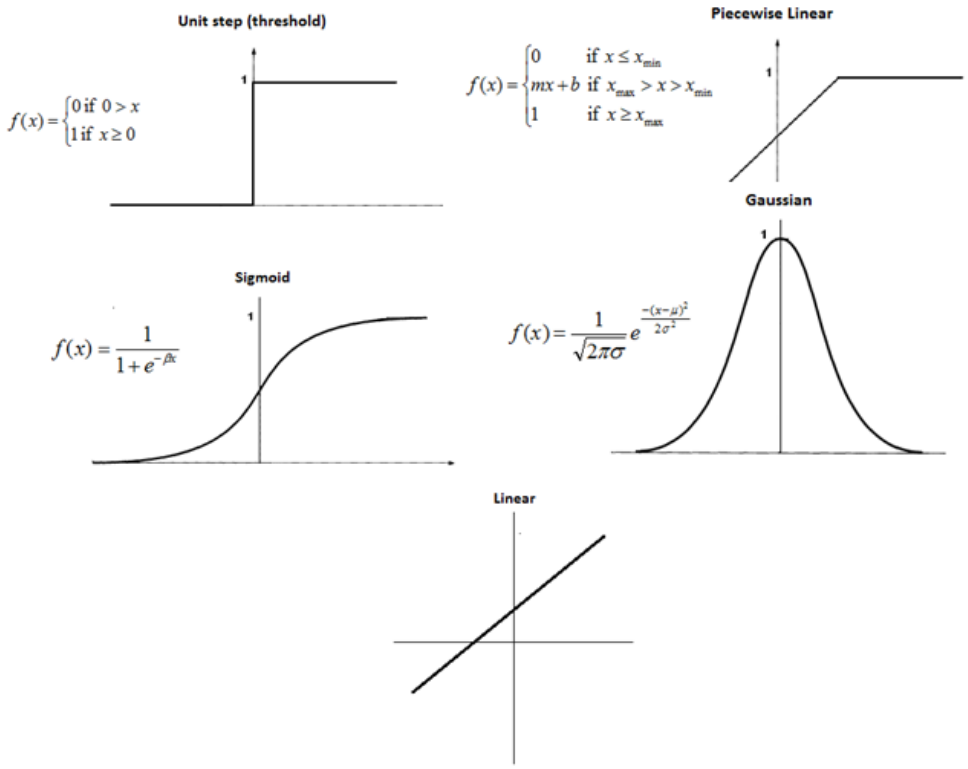
- La topología: Esto corresponde al número de capas y de nodos, además de la dirección en que la información pasa de un nodo al siguiente, dentro de capas o entre capas. Generalmente, las redes más grandes y más complejas son capaces de identificar patrones más sutiles y límites de decisión complejos.

- El algoritmo de entrenamiento: Establece la importancia de cada conexión para transmitir o no la señal a los nodos correspondientes. El más usado es el algoritmo "*backpropagation*". El nombre indica que para corregir los errores de predicción va hacia atrás de la red corrigiendo los pesos de los nodos.

- La función de activación: Función que recibe un conjunto de entradas e integra la señales para transmitir la información a otro nodo/capa.

En las neuronas biológicas la función de activación equivaldría a un proceso que implica sumar las señales de entrada y determina si sobrepasa el umbral de transmisión de la señal. Si es así, la neurona pasa la señal a otra neurona; de lo contrario, no hace nada. En las ANN la señal de salida solo se da una vez que se ha alcanzado un umbral de entrada especificado. Existen diferentes funciones de umbral para las ANN (fig. 8).

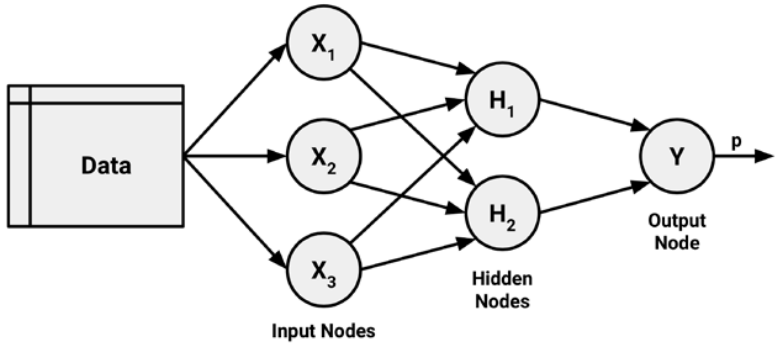
Figura 8: Tipos de funciones de umbral. Tomado de : http://chem eng.utoronto.ca/~datamining/dmc/artificial_neural_network.htm



La capacidad de una red neuronal para aprender está enraizada en su topología (fig. 9), o en los patrones y estructuras de las neuronas interconectadas. Aunque existen innumerables formas de arquitectura de red, se pueden diferenciar por tres características clave:

• El número de capas
• Si la información en la red puede viajar hacia atrás
• La cantidad de nodos dentro de cada capa de la red

Figura 9: Diagrama de la estructura de una ANN y los flujos de información entre los nodos. Tomado de Lantz [30].



Actualmente, no existe una regla fiable para determinar el número de neuronas en las capa ocultas. El número apropiado depende de la cantidad de nodos de entrada, la cantidad de datos de entrenamiento, la cantidad de datos ruidosos y la complejidad de la tarea de aprendizaje, entre muchos otros factores.

En la tabla 3 se muestran las fortalezas y debilidades del modelo. La implementación del modelo basado en ANN se realizó en R mediante el paquete “neuralnet”. Esta librería fue desarrollada por Stefan Fritsch y Frauke Guenther [31].

Tabla 3: Tabla de fortalezas y debilidades de los modelos basados en ANN. Tomada de Lantz [30].

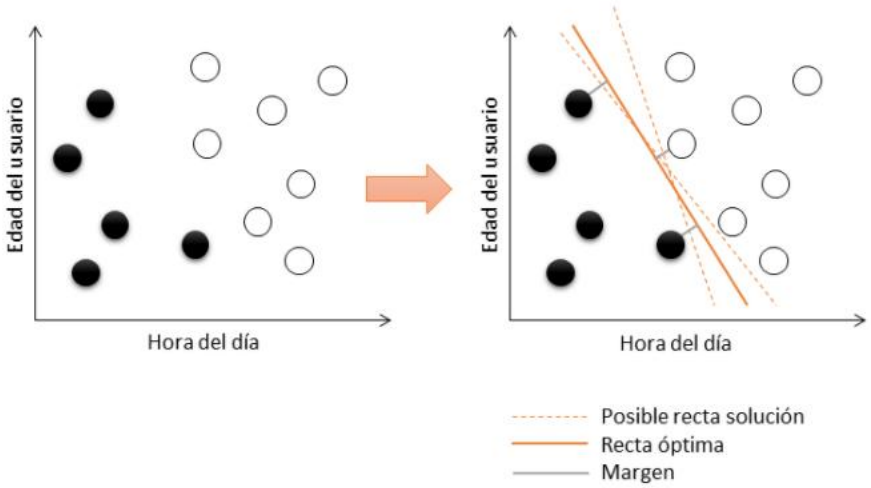
Fortalezas	Debilidades
• Adaptable a clasificaciones o problemas de predicción numérica • Capaz de modelar patrones más complejos que casi cualquier otro algoritmo • No necesita muchas restricciones acerca de las relaciones subyacentes de los datos	• Requiere de gran potencia computacional y en general es de aprendizaje lento, particularmente si la topología es compleja. • Propenso a sobreajustar los datos de entrenamiento. • Es un modelo de caja negra complejo, que es difícil, si no imposible, de interpretar.

4.2.3. Support vector machine

Las máquinas de vectores de soporte o *support vector machines* (SVM) son un conjunto de algoritmos de aprendizaje supervisado, dirigido tanto a la resolución de problemas de clasificación como de regresión, indistintamente.

Los algoritmos de SVM se pueden entender como una superficie en la cual se crea un límite e inducen separadores lineales o hiperplanos entre los puntos de datos que representan ejemplos y sus características. Estos hiperplanos dividen el espacio para crear particiones homogéneas.

Figura 10: Representación bidimensional de división del espacio por hiperplanos**.** Tomada de http://www.analiticaweb.es/machine-learning-y-support-vector-machines-porque-el-tiempo-es-dinero-2/



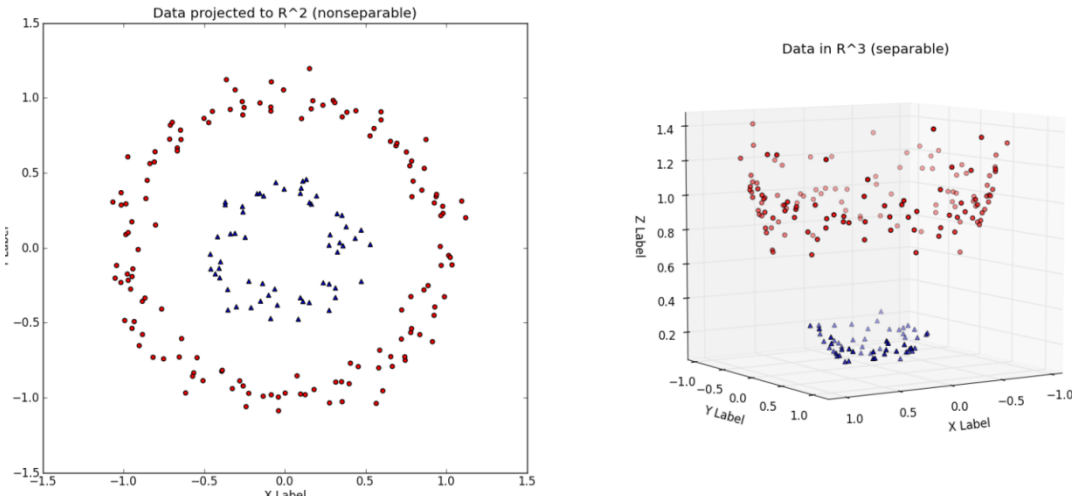
Inicialmente estos algoritmos estaban pensados y preparados para clasificaciones binarias pero esto no se suele presentar de forma idílica, por lo tanto deben tratar con:

- Más de dos variables predictoras
- Clasificaciones en más de dos categorías
- Conjuntos de datos no separables
- Curvas no lineales de separación

Cuando los datos no son separables de forma lineal o presentan alguna de las características anteriores, es necesario el uso de *kernels* o funciones de similitud, y especificar un parámetro *C* para minimizar la función de coste. La elección de este parámetro es a base de ensayo/error, pero se buscan valores que no sean extremos en la búsqueda del equilibrio sesgo/varianza. La representación por medio de funciones *kernel* ofrece una solución a este problema. La utilización de un *kernel* permite transformar los datos a un espacio con más dimensiones. Esto hace que un conjunto de datos que inicialmente no era linealmente separable, tras ser transformado con un *kernel*, sea linealmente separable. Los *kernels* más populares son el lineal y el gaussiano, aunque hay otros como el polinomial, *kernel string*, *kernel chi-square*, etc.

La siguiente imagen (fig. 11) muestra como un conjunto de datos 2D linealmente no separable, tres ser transformado mediante un *kernel* gaussiano a un espacio 3D, puede ser linealmente separable.

Figura 11: Representación gráfica de cambio de dimensiones mediante la utilización de un kernel gaussiano. Tomada de http://www.eric-kim.net/eric-kim-net/posts/1/imgs/data_2d_to_3d.png



La tabla 4 muestra las fortalezas y debilidades del modelo. La implementación del modelo basado en SVM se realizó en R mediante el paquete “kernlab”. Esta librería fue desarrollada por Karatzoglou et al. [32].

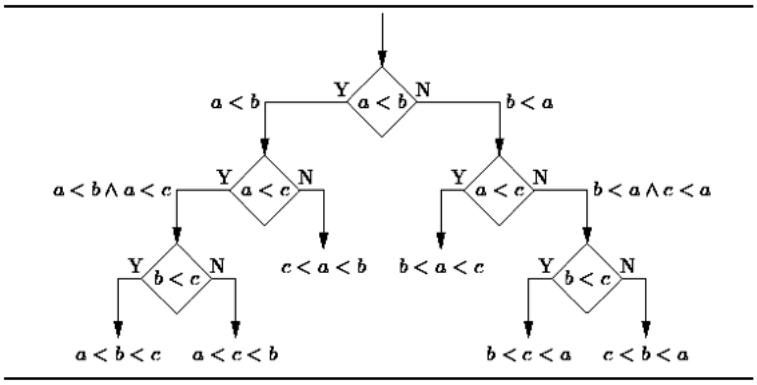
Tabla 4: Tabla de fortalezas y debilidades de los modelos basados en SVM. Tomada de Lantz [30].

Fortalezas	Debilidades
• Se puede usar para problemas de clasificación o predicción numérica • Funciona bastante con datos ruidosos y no es muy propenso al sobreajuste • Puede ser más fácil de usar que las redes neuronales, en particular debido a la existencia de varios algoritmos SVM bien soportados.	• Encontrar el mejor modelo requiere probar diferentes kernels y parámetros del modelo prueba (prueba y error) • Lento de entrenar, sobre todo a medida que aumenta el número de características • Los resultados del modelo son difícil, si no imposible, de interpretar (caja negra)

4.2.4. Decision tree

Los modelos que emplean arboles de decisión (*decision tree*) son modelos clasificadores potentes, que utilizan una estructura jerárquica para modelar las relaciones entre las características y los posibles resultados. Estos modelos construyen una estructura que se asemeja a la forma de un árbol (fig. 12) donde un caso comienza en el nodo raíz desde el que posteriormente irá recorriendo diferentes nodos de decisión donde se requiere que se realicen elecciones basadas en las características del caso. Estas elecciones dividen los datos en ramas que indican posibles resultados de las decisiones. En caso de existir una decisión final el árbol es terminado en nodos terminales que denotan la clase a la que pertenece el caso resultado de las serie de decisiones.

Figura 12: Esquema de la estructura jerárquica en forma de árbol de un decisión tree. Tomada de http://blog.raona.com/los-10-algoritmos-esenciales-machine-learning/



Los arboles de decisión se construyen realizando particiones recursivas. Los datos se dividen en un primer subconjunto, y a continuación se continúan dividiendo repetidamente cada vez en conjuntos más pequeños hasta que el algoritmo determina que los datos colocados dentro de los subconjuntos son homogéneos.

Existen varias medidas que se pueden usar para identificar la mejor característica para dividir el árbol de decisión. El algoritmo de árbol de decisión C5.0 usa entropía, un concepto tomado de la teoría de la información que cuantifica la aleatoriedad o desorden dentro de un conjunto de valores. Los conjuntos con alta entropía son muy diversos y brindan poca información sobre otros elementos que también pueden pertenecer al conjunto, ya que no existe una similitud aparente.

El árbol de decisiones espera encontrar divisiones que reduzcan la entropía y, en última instancia, aumenten la homogeneidad dentro de los grupos. Típicamente, la entropía se mide en bits. Si solo hay dos clases posibles, los valores de entropía pueden oscilar entre 0 y 1. Para n clases, la entropía varía de 0 a log2 (n). En cada caso, el valor mínimo indica que la muestra es completamente homogénea, mientras que el valor máximo indica que los datos son tan diversos como sea posible.

La tabla 5 muestra las fortalezas y debilidades de este modelo. La implementación del modelo basado en *decision tree* se realizó en R mediante el paquete “C50”. Esta librería fue desarrollada por Kuhn y Quinlan [33].

Tabla 5: Tabla de fortalezas y debilidades de los modelos basados en decision tree. Tomada de Lantz [30].

Fortalezas	Debilidades
• Funcionan bien en la mayoría de problemas de clasificación • Puede utilizar características numéricas o categóricas, y datos faltantes • Excluye características con baja o sin importancia • Se pueden utilizar con conjuntos de datos pequeños y grandes • Los resultados pueden interpretarse de forma sencilla sin conocimientos matemáticos	• A menudo están sesgados hacia características que tienen una gran cantidad de niveles • Es fácil sobreajustar o subajustar el modelo • Pequeños cambios en los datos de entrenamiento pueden generar grandes cambios en la lógica de decisión • Los árboles grandes pueden ser difíciles de interpretar y las decisiones que toman pueden parecer contradictorias • No funciona bien en problemas complejos

4.2.5. Random Forest

Los modelos basados en *random forest* se centran en conjuntos de *decisión trees*. *Random forest* es una técnica que mejora la precisión en la clasificación mediante la incorporación de aleatoriedad en la construcción de cada clasificador individual. La aleatorización se introduce en la construcción de la partición del espacio (durante la construcción del árbol de decisión) así como en la muestra de entrenamiento. El algoritmo de *random forest* introduce de forma aleatoria en cada nodo un conjunto de variables de todas las originales y de estas selecciona la mejor para realizar la partición. El proceso del algoritmo es el siguiente:

-Selecciona individuos al azar.
-Crea arboles de decisión con un número de variables al azar para cada nodo del árbol.
-Crea un árbol de decisión con cada set de datos, generando así, diferentes árboles con diferentes datos y variables.
-Predice los nuevos datos usando un voto donde clasificará como positivo si la mayoría de los árboles predicen la observación como positiva.

La tabla 6 muestra las fortalezas y debilidades del modelo. La implementación del modelo basado en *random forest* se realizó en R mediante el paquete “randomForest”. Esta librería fue desarrollada por Liaw y Wiener [34].

Tabla 6: Tabla de fortalezas y debilidades de los modelos basados en *random forest*. Tomada de Lantz [30].

Fortalezas	Debilidades
• Funcionan bien en la mayoría de problemas de clasificación • Puede utilizar características numéricas o categóricas, y datos faltantes • Excluye características con baja o sin importancia • Se pueden utilizar para una cantidad extremadamente grande de características	• No son tan fáciles de interpretar a diferencia de los modelos basados en arboles de decisión. • Puede requerir un trabajo adicional para ajustar el modelo a los datos.

4.2.6. k-Nearest Neighbors

Este es quizás uno de los algoritmos de aprendizaje automático más simples y que, aun en la actualidad, se sigue utilizando ampliamente. El modelo *k-nearest neighbors* (k-nn) es un tipo de *lazy learning*. En los modelos *lazy learning* no se entrena un modelo, retrasa el procesamiento de la base de datos de entrenamiento hasta que se recibe una solicitud explícita de información. El algoritmo de clasificación realiza la clasificación de un nuevo ejemplo basándose en los ejemplos de la muestra de entrenamiento más cercanos al que se pretende clasificar; de forma más concreta, se asigna al nuevo caso sin etiquetar a la clase más frecuente entre los k vecinos más próximos. Este modelo es la base sobre la que se implementan los sistemas de clasificación de razonamiento basado en casos (*case-based reasoning*) [35].

La letra k es un término variable que implica el número de casos vecinos que se elegirán para realizar la clasificación y asignación del grupo. Después de elegir el valor de k, el algoritmo requiere un conjunto de datos de entrenamiento compuesto por ejemplos que han sido clasificados en varias categorías. Luego, para cada registro de prueba o test, k-nn identifica k registros en los datos de entrenamiento que son los "más cercanos" en similitud. A la instancia de prueba sin etiqueta se le asigna la clase de la mayoría de los k vecinos más cercanos. La elección de un valor óptimo de k se puede realizar mediante prueba y error.

La tabla 7 muestra las fortalezas y debilidades del modelo. La implementación del modelo basado en *k-nearest neighbors* se realizó en R mediante el paquete “class”. Esta librería fue desarrollada por Venables y Ripley [36].

Tabla 7: Tabla de fortalezas y debilidades de los modelos basados en k-nn. Tomado de Lantz [30].

Fortalezas	Debilidades
• Simple y efectivo • No hace suposiciones sobre la distribución de datos subyacente • Fase de entrenamiento rápida	• No produce un modelo, lo que limita la capacidad de encontrar nuevos conocimientos en las relaciones entre las características • Fase de clasificación lenta • Requiere una gran cantidad de memoria • Variables nominales y los datos faltantes requieren un procesamiento adicional

4.3. Construcción y entrenamiento de los modelos

Para la creación de los modelos de clasificación comentados en puntos anteriores se siguió una metodología conocida como Método H (*hold out*) o *Training-test*. Esta metodología consiste en dividir de manera aleatoria la base de datos de estudio en dos bases de datos separadas. Una de ellas estará formada por aquellos casos que se utilizarán para entrenar los modelos y los casos restantes formarán otra base de datos que se utilizará como casos de prueba para conocer la precisión de los modelos.

En este trabajo se seleccionaron 2/3 de los datos para formar la base de datos de entrenamiento (*train*) y 1/3 de los datos como muestras de prueba (test). Adicionalmente se generaron diez particiones aleatorias diferentes, lo cual produjo diez bases de datos de entrenamiento y diez bases de datos de test a partir de la base de datos original. Este método se conoce como validación cruzada o *cross-validation*.

El método de validación cruzada o *cross-validation* es una técnica utilizada de forma habitual para evaluar que los resultados de un análisis de predicción son independientes de la partición aleatoria de los datos de entrenamiento y prueba. Una vez realizadas las medidas de evaluación con cada una de las particiones se considera la media aritmética de las mismas.

Siempre que se realizó la selección al azar de casos, se mantuvieron las frecuencias de interacciones miARN-ARNm positivas y negativas en cada una de las particiones.

Adicionalmente se realizó la técnica de *boostrap* para el modelo *decision tree*. El *boostrap* se basa en la noción de que al combinar un número de modelos con bajo rendimiento, puede crear un equipo mucho más potente que los modelos por sí solos. Cada uno de los modelos tiene un conjunto de fortalezas y debilidades que pueden, al combinarse varios modelos, mejorar enormemente la precisión de la clasificación.

4.4. Métricas de evaluación de modelos

4.4.1 Variables de efectividad

Una vez construido y entrenado el modelo elegido, se debe comprobar la fiabilidad y la precisión del modelo. Para ello se utilizan los datos de prueba que habíamos dividido anteriormente. Con esta selección de casos de prueba se comprueba si el modelo es capaz de predecir correctamente la clase a la que pertenece cada uno de los casos.

Para comprobar la validez de las predicciones de un modelo se emplea la matriz de confusión (tabla 8). La matriz de confusión genera una matriz que permite la visualización del funcionamiento del clasificador, mostrando de forma resumida la efectividad y la precisión del modelo. Las columnas de la matriz representan el número de elementos reales por clase y cada fila el número de elementos predichos.

Tabla 8: Tabla representativa de una matriz de confusión

	Reference	
Prediction	Interacción	No interacción
Interacción	817 (TP)	387 (FP)
No interacción	237 (FN)	675 (TN)

A partir de las matrices de confusión se pueden extraer varias variables de efectividad con las que podremos comparar la precisión y viabilidad de los modelos utilizando las siguientes métricas:

Numero de verdaderos positivos (TP) y verdaderos negativos (TN):

Los elementos de la primera casilla (1,1) y la última (2,2) corresponden al número de elementos clasificados de forma correcta. El modelo ha predicho correctamente el número de elementos positivos y el número de negativos que realmente eran positivos y negativos. Los elementos clasificados en la clase positiva (Interacción) y que realmente eran de la clase positiva serían considerados verdaderos positivos y aquellos elementos clasificados como clase negativa (No interacción) y que pertenecen a la clase negativa en la realidad serian catalogados como verdaderos negativos.

Numero de falsos positivos (FP) y falsos negativos (FN):

Los elementos de la segunda (2,1) y tercera (1,2) casilla corresponden al número de elementos clasificados de forma incorrecta. El modelo ha predicho erróneamente el número de elementos positivos y los ha clasificado como negativos (Elementos con interacción como no interacción) y el número de elementos negativos los ha clasificado como positivos (elementos sin interacción como elementos con interacción)

Sensibilidad y especificidad:

La sensibilidad (*sensitivity*) es la probabilidad de clasificar un elemento como la clase positiva y que realmente pertenezca a la clase positiva. La sensibilidad se calcula como la proporción de casos positivos que obtuvieron un resultado positivo en la predicción.

Sensitivity = TP/(TP +FN)

La especificidad (*specificity*) es la probabilidad de clasificar un individuo como la clase negativa y que realmente pertenezca a la clase negativa. La especificidad se calcula como la proporción de casos negativos que obtuvieron un resultado negativo en la predicción.

Specificity = TN/(TN+FP)

Accuracy:

La variable *accuracy* es la tasa total de aciertos con las predicciones, es decir, número de elementos clasificados correctamente.

Accuracy = TP+TN / (TP+TN+FN+FP)

Precision:

La variable *precision* mide la tasa de predicciones verdaderas. Se divide en dos variables, *Positive Predictive Value* (PPV) y *Negative Predictive Value* (NPV).

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

Kappa:

El coeficiente Kappa oscila desde 0 hasta un máximo de 1, lo que indica un concordancia perfecta entre las predicciones del modelo y los valores verdaderos.

- Concordancia pobre = < 0.20
- Concordancia justa = 0.20 a 0.40
- Concordancia moderada = 0.40 a 0.60
- Buena concordancia = 0.60 a 0.80
- Muy Buena concordancia = 0.80 a 1.00

Curva ROC:

El *Receiver Operating Characteristic* (ROC) se usa comúnmente para examinar la compensación entre la detección de verdaderos positivos, evitando al mismo tiempo los falsos positivos.

Las curvas se definen en un gráfico con la proporción de verdaderos positivos en el eje vertical y la proporción de falsos positivos en el eje horizontal. El modelo clasificador perfecto tendría una curva que pasa por el punto donde la tasa de verdaderos positivos es del 100% y la tasa de falsos positivos es de 0%. El modelo sería capaz

identificar todos los casos positivos antes de clasificar incorrectamente cualquier negativo.

Cuanto más cerca esté la curva de un modelo clasificador perfecto, mejor identificará y clasificará el modelo. Esto se puede medir calculando el área debajo de la curva ROC (*area under the curve*, AUC). Actualmente se considera que si el valor AUC se encuentra entre 0.9 y 1 el modelo es excepcional, si se encuentra entre 0.8 y 0.9 es excelente o bueno, entre 0.7 y 0.8 es aceptable, entre 0.6 y 0.7 un modelo pobre y un modelo sin discriminación si AUC es menor que 0.6.

4.4.2. Comparación de modelos

Como último paso para el estudio de los modelos construidos se realizó la prueba de los rangos con signo de Wilcoxon (Wilcoxon Signed-Rank Test). Esta prueba es una prueba no paramétrica que se utiliza para comprobar si hay diferencias entre las distribuciones de dos poblaciones a partir de dos muestras dependientes o relacionadas. La hipótesis nula de esta prueba postula que las muestras proceden de poblaciones con la misma distribución de probabilidad; la hipótesis alternativa establece que hay diferencias respecto a la tendencia central de las poblaciones y puede ser direccional o no.

La prueba fue aplicada para comparaciones dos a dos de los modelos con cada una de las variables de efectividad anteriormente comentadas para los 10 entrenamientos de un mismo modelo.

5. Resultados

5.1 Preparación de la base de datos

La base de datos utilizada para la creación de los modelos contenía más de 14.000 interacciones miARN-ARNm positivas y negativas. Exactamente la base de datos contenía 14.186 interacciones, de las cuales en 11.013 no existía una interacción real y en 3.173 la interacción había sido comprobada previamente por Lu y Leslie [24]. El número de interacciones negativas era tan elevado que los resultados estaban sesgados y los modelos tendían a clasificar los casos como la clase más numerosa, es decir que la variable *specificity* (probabilidad de clasificar un individuo como la clase negativa y que realmente pertenezca a la clase negativa) era enormemente superior que la variable *sensitivity* (probabilidad de clasificar un elemento como la clase positiva y que realmente pertenezca a la clase positiva).

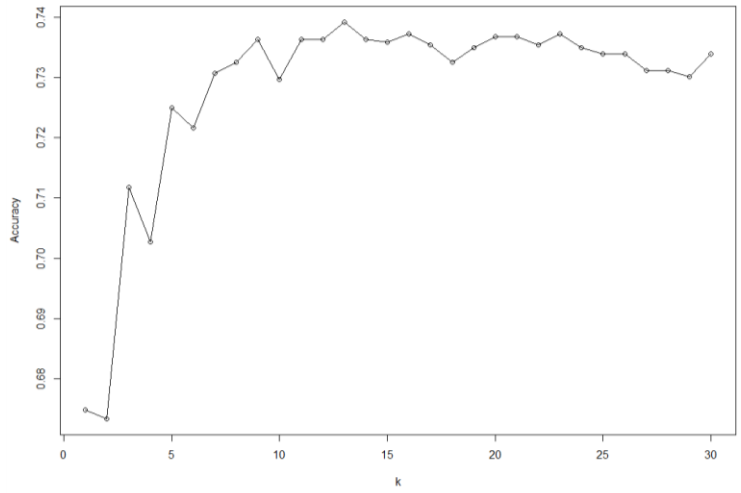
Para solucionar este problema se decidió utilizar el mismo número de interacciones donde sí que existía interacción y de las que no existía interacción. Finalmente la base de datos se formó con 6.347 interacciones de las cuales 3.173 eran interacciones reales y 3.173 interacciones no reales.

Las *features* de las interacciones se encontraban en diferentes escalas. Por ejemplo la energía libre se encuentra en una escala negativa que se situaba entre -5 y -44.1 mientras que las *features* estructurales se encontraban en una escala positiva entre 0 y 24. También encontramos *features* que eran tasas (A%, AU%, etc.) y su escala se encontraba entre 0 y 1. Esta diferencia en escalas produce problemas en nuestros clasificadores por ello las *features* fueron normalizadas. Posteriormente todas las *features* presentaban un rango entre 0 y 1.

5.2. Elección de parámetros

Para el modelo de *k-nearest neighbors* se crearon diferentes modelos otorgando un valor de *k* de 1 a 30 (fig. 13). A partir de una *k* superior a 9 el modelo ya no presentaba mejoras en las variables de eficiencia.

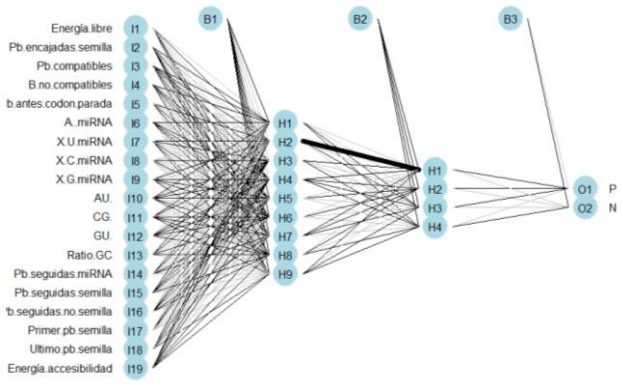
Figura 13: Grafico de valores de *accuracy* en función de *k*



El modelo de Naive Bayes se entrenó sin el estimador de Laplace, ya que los modelos entrenados con y sin estimador de Laplace presentaban los mismos resultados. El estimador de Laplace básicamente agrega una unidad a cada uno de los recuentos en la tabla de frecuencias, lo que garantiza que cada característica tenga una probabilidad distinta de cero de ocurrir en cada clase. Normalmente, el estimador de Laplace se establece en 1, lo que garantiza que cada combinación de características de clase se encuentre en los datos al menos una vez.

El modelo ANN fue entrenado con dos capas ocultas, la primera con 9 neuronas y la segunda con 4 (fig. 14). Esta elección fue debida a la regla de la pirámide geométrica. Para la creación de redes neuronales artificiales no existe un número correcto de capas ocultas ni de número de neuronas, por lo tanto optamos por esta regla. La regla de la pirámide geométrica sigue la ecuación *in - in/2 - in/4 - out* donde *in* es el número de *features* y neuronas de la primera capa y *out* el número de clases y neuronas de salida.

Figura 14: Estructura de la *artificial neural network* construida.

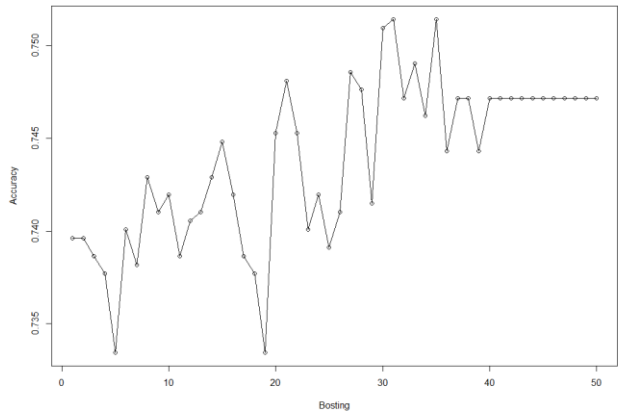


La red neuronal creada con estos parámetros requería mucha potencia de procesamiento y el hardware disponible para ello no podía proporcionárselo. Se intentó con hardware más potente pero tampoco se logró crear el modelo. Finalmente se decidió reducir el número de muestras para entrenar el modelo. Se consiguió crear la red neuronal con un 10% de los datos totales.

El modelo SVM fue entrenado con diferentes *kernels* para comprobar qué *kernel* se ajustaba mejor a los datos y presentaba una eficiencia del modelo más elevada. Fue entrenado con el *Radial Basis kernel "Gaussian"*; *kernel* polinomial de grado 1, 2 y 3; *kernel* lineal; *kernel* de tangente hiperbólica; *kernel* Laplaciano; *kernel* ANOVA RBF; *kernel* Spline y *kernel* String. El modelo que presentó mejores resultados fue el modelo entrenado con el *kernel* polinomial de grado 2.

En el modelo de *decisión tree* se aplicó un número diferente de *boosting* para decidir el mejor valor de *boosting* para el modelo. Se aplicaron desde 1 hasta 50 (fig. 15). El modelo que presentó una mayor eficiencia de predicción de los datos test fue el modelo con un *boosting* de 35. A partir de 35 no se observaron mejoras en la eficiencia del modelo.

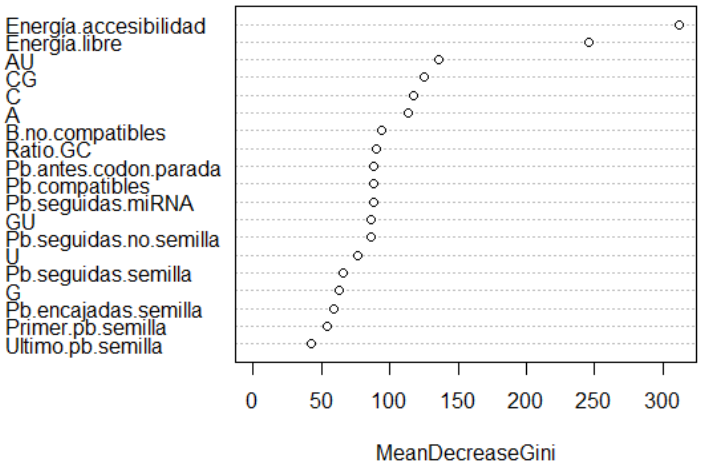
Figura 15: Gráfico de valores de *accuracy* en función del número de *boosting*



El modelo de *random forest* fue entrenado con diversos números de árboles para saber qué número de árboles era el más adecuado para obtener una mayor eficiencia del modelo. Se entrenó el modelo con 250 árboles, 500, 750, 1000, 1250, 1500, 1750 y 2000. El modelo que presentó mejores resultados fue el modelo entrenado con 1500 árboles.

El modelo de *random forest* ofrece una función para conocer qué *features* de los miARN calculados tienen mayor importancia a la hora de predecir si una interacción de la base de datos es una interacción verdadera o si no existe interacción entre el miARN y el ARNm. La figura 16 muestra cada *feature* en el eje *y*, y su importancia en el eje *x*. Se ordenan de arriba a abajo de mayor a menor importancia. Por lo tanto, las *features* más importantes están en la parte superior y una estimación de su importancia en el eje *x*.

Figura 16: Gráfico de importancia de las *features*



5.3. Valoración de los modelos

k-nearest neighbors

El modelo de k-nn, con un *cross-validation* de 10, presentó una *accuracy* media de 0.73 (tabla 9, fig. 17), por lo tanto encontramos un modelo capaz de predecir correctamente si existe interacción o no en el 73% de los casos de los datos de prueba. Las variables de efectividad, *sensitivity* y *specificity* presentaron valores prácticamente iguales, alrededor de 0.73. El valor de kappa fue de 0.46, y por lo tanto el modelo presentó una buena concordancia entre los valores predichos y los valores reales. Las desviaciones típicas de la media fueron muy bajas, lo que indica que la selección aleatoria de los datos de prueba y test no afectó al entrenamiento del modelo de forma sustancial. Por otro lado la curva ROC (fig. 18) del modelo se situó cerca de la diagonal y el valor AUC del modelo fue de 0.57. Esto muestra que el modelo era un modelo con poca capacidad de clasificación y por lo tanto un modelo con poca validez.

Figura 17: Distribución de las principales variables de eficiencia del modelo k-nn

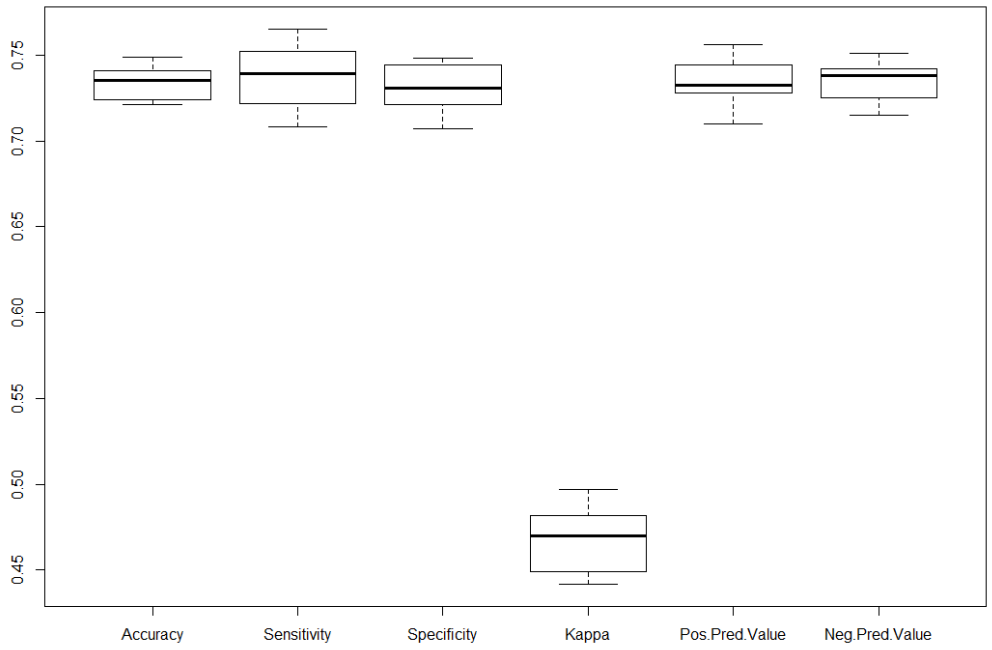
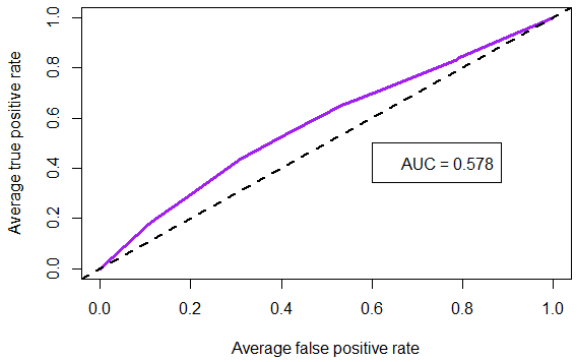


Tabla 9: Variables de eficiencia del modelo k-nn

k-NN	Media	Desviación típica
Accuracy	0.7341	0.0092
Sensitivity	0.7372	0.0183
Specificity	0.7310	0.0135
Kappa	0.4682	0.0181
Pos Pred Value	0.7339	0.0138
Neg Pred Value	0.7344	0.0120

Figura 18: Curva ROC del modelo k-nn



Naive Bayes

El modelo de Naive Bayes, con un *cross-validation* de 10, presentó una *accuracy* media de 0.65 (tabla 10, fig. 19). El modelo solo fue capaz de predecir si existe interacción en el 65% de los casos de los datos de prueba. La *sensitivity* también presentó unos valores bajos, con una media de 0.51. En cambio, el valor medio de *specificity* fue de 0.79. El modelo clasificó mejor los casos donde no existe interacción miARN-ARNm. El valor de kappa también fue bajo, 0.30. El modelo presentó una concordancia baja o justa entre los valores predichos y los valores reales. Las desviaciones típicas de las variables de efectividad fueron muy bajas. La curva ROC (fig. 20) y el valor AUC, 0.70, nos indicaron que el modelo presentaba una capacidad de clasificación pobre.

Figura 19: Distribución de las principales variables de eficiencia del modelo k-nn

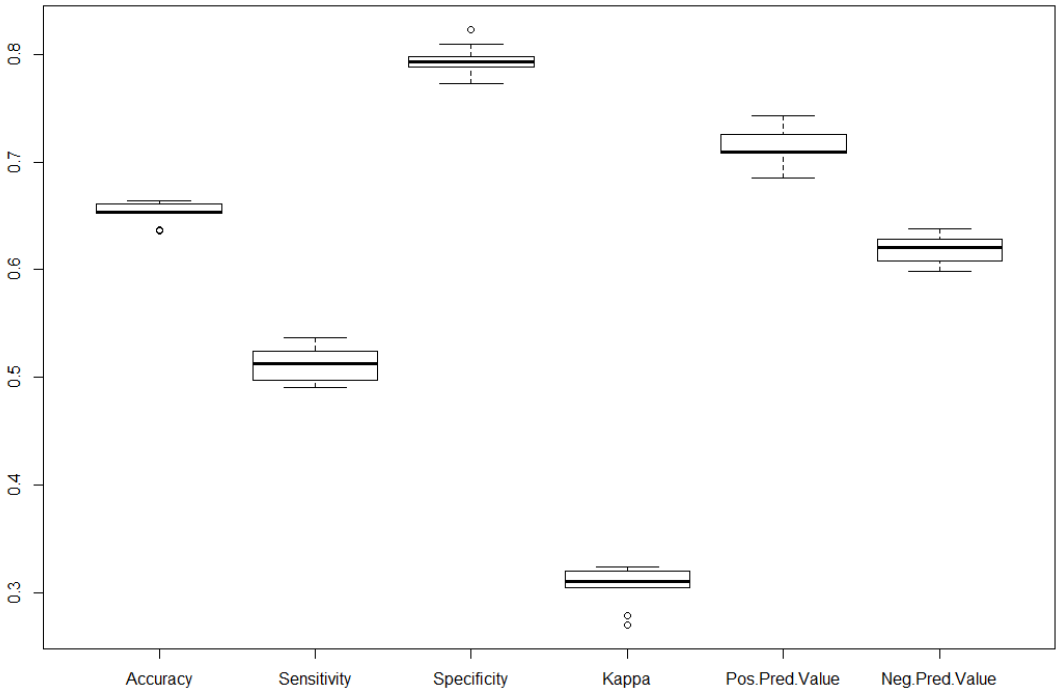
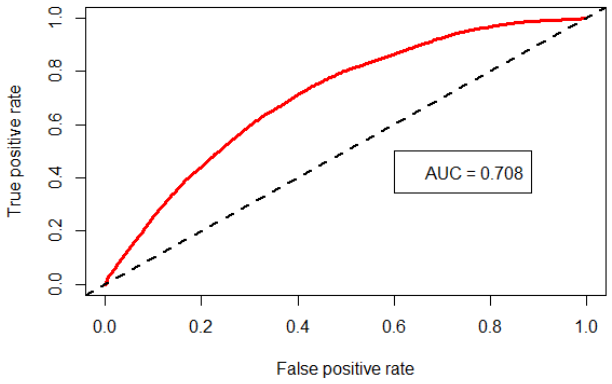


Tabla 10: Variables de eficiencia del modelo NB

Naive Bayes	Media	Desviación típica
Accuracy	0.6526	0.0094
Sensitivity	0.5116	0.0157
Specificity	0.7948	0.0139
Kappa	0.3059	0.0182
Pos Pred Value	0.7150	0.0175
Neg Pred Value	0.6177	0.0134

Figura 20: Curva ROC del modelo Naive Bayes



Artificial neural network

El modelo de ANN presentó una *accuracy* media de 0.61 con unos valores de *sensitivity* y *specificity* prácticamente iguales, 0.612 y 0.623 (tabla 11, fig. 21). Estos valores nos reflejan que el modelo fue capaz de predecir si existe interacción o no en un 61% de los casos y que no tuvo preferencia por clasificar los casos como positivos o negativos. El valor de kappa fue el más bajo de entre todos los modelos, 0.23, por lo tanto este modelo no presentó concordancia entre los valores predichos y los reales. Las desviaciones típicas de las variables de efectividad del modelo fueron más elevadas que en el resto de modelos, lo que nos puede indicar que en este caso la selección aleatoria de los casos que forman los datos de entrenamiento y los datos de prueba sí que pudieron afectar a la efectividad del modelo. La curva ROC (fig. 22) cercana a la diagonal y el valor AUC, 0.62, nos indicó que el modelo de ANN fue un modelo pobre.

Figura 21: Distribución de las principales variables de eficiencia del modelo ANN

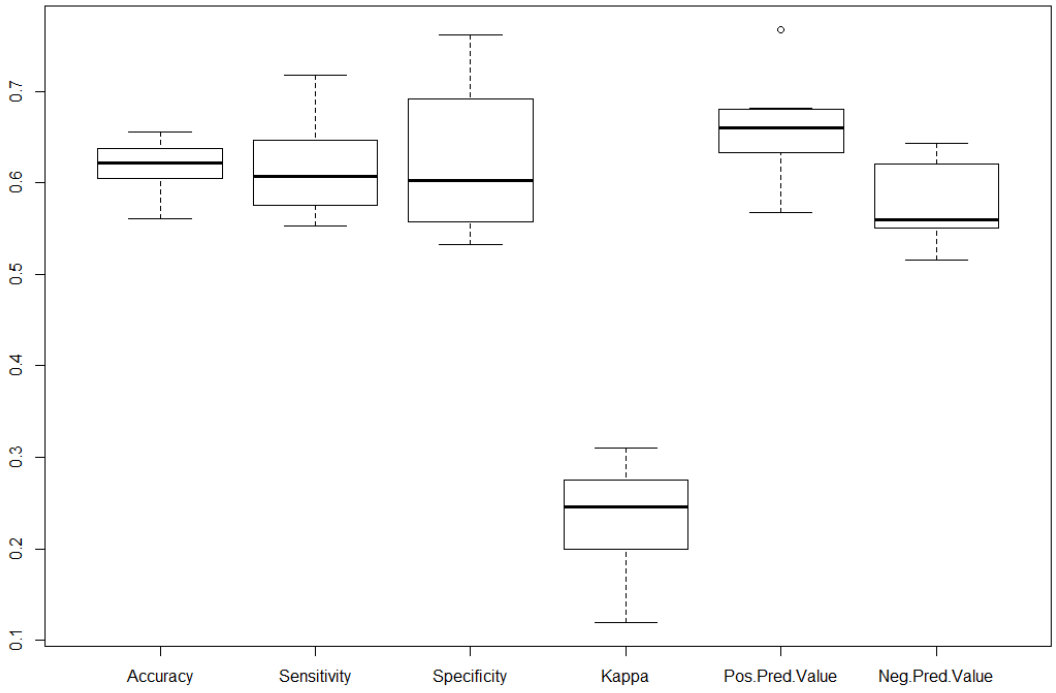
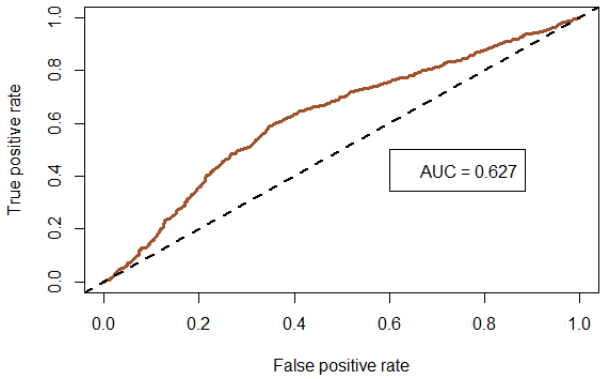


Tabla 11: Variables de eficiencia del modelo ANN

ANN	Media	Desviación típica
Accuracy	0.6175	0.0293
Sensitivity	0.6128	0.0499
Specificity	0.6231	0.0819
Kappa	0.2336	0.0619
Pos Pred Value	0.6578	0.0524
Neg Pred Value	0.5790	0.0454

Figura 22: Curva ROC del modelo ANN



Support Vector Machine

El modelo SVM presentó una *accuracy* de 0.74 y unos valores de *sensitivity* y *specificity* de 0.75 y 0.73 (tabla 12, fig. 23). Este modelo fue capaz de predecir correctamente el 74% de los casos de prueba de la base de datos y predijo con mayor eficiencia, aunque levemente, los casos donde existe interacción. El valor de kappa fue de 0.48, por lo tanto la concordancia entre los valores predichos y los valores reales fue buena. Las bajas desviaciones típicas de las variables de efectividad reflejan que las predicciones realizadas fueron independientes de la partición aleatoria de los datos de entrenamiento del modelo. La curva ROC (fig. 24) y el valor AUC, 0.80, nos afirmaron que el modelo de predicción SVM fue un buen clasificador.

Figura 23: Distribución de las principales variables de eficiencia del modelo SVM

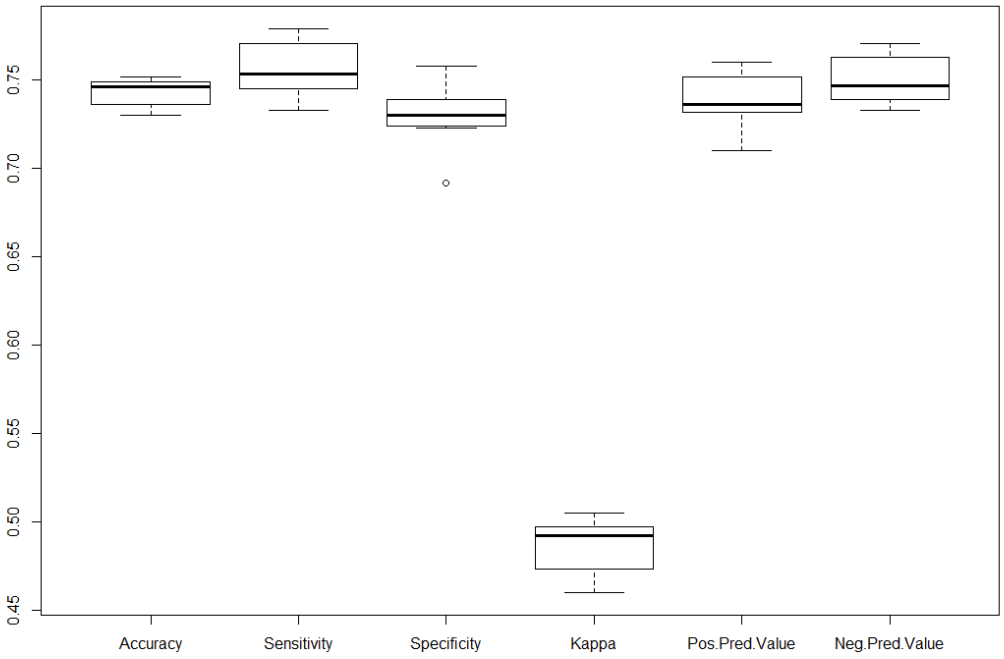
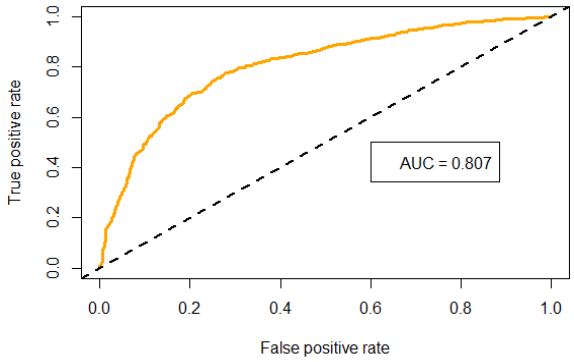


Tabla 12: Variables de eficiencia del modelo SVM

SVM	Media	Desviación típica
Accuracy	0.7437	0.0075
Sensitivity	0.7564	0.0163
Specificity	0.7312	0.0183
Kappa	0.4875	0.0149
Pos Pred Value	0.7392	0.0153
Neg Pred Value	0.7490	0.013

Figura 24: Curva ROC del modelo SVM



Decisión Tree

El modelo de *decision tree* presentó una *accuracy* media de 0.74 y una *specificity* más elevada que la *sensitivity*, 0.72 y 0.76 respectivamente (tabla 13, fig. 25). Este modelo fue capaz de predecir correctamente el 74% de los casos de prueba y podríamos afirmar que este modelo fue capaz de predecir mejor los casos donde no existe interacción miARN-ARNm. El valor de kappa fue similar al valor del modelo SVM, 0.48, indicando que la concordancia entre los valores predichos y los reales era buena. Las desviaciones típicas fueron un poco elevadas en comparación con el resto de modelos, lo que podría reflejar que para este modelo la partición aleatoria para los datos de entrenamiento y los datos de prueba sí que pudo afectar a la efectividad del modelo. La curva ROC (fig. 26) y el valor de AUC, 0.81, nos indicó que el modelo predictivo basado en *decision tree* construido para nuestro trabajo fue un buen clasificador.

Figura 25: Distribución de las principales variables de eficiencia del modelo *decision tree*

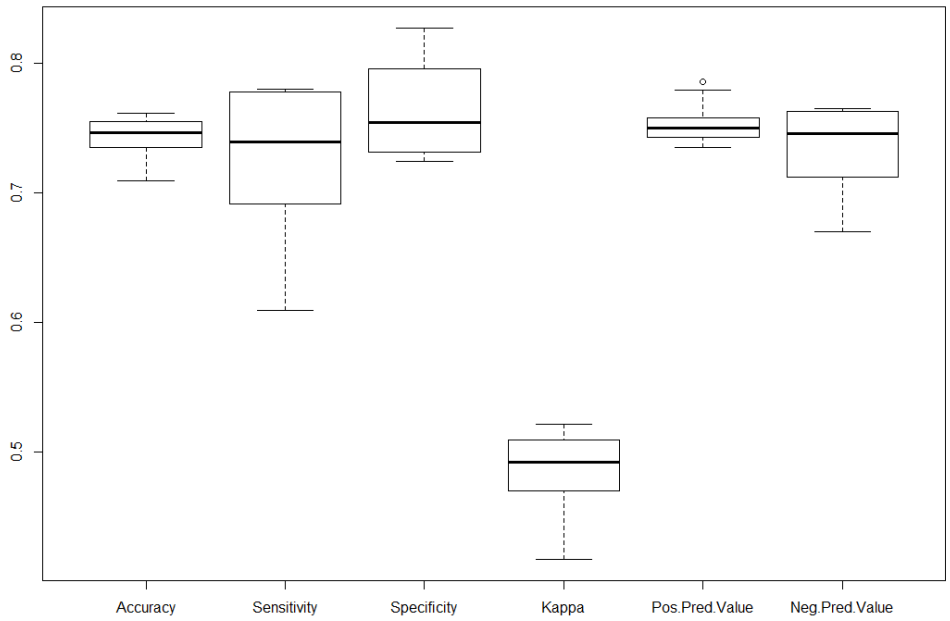
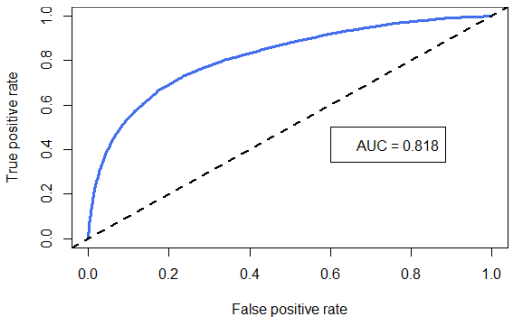


Tabla 13: Variables de eficiencia del modelo *decision tree*

Decision tree	Media	Desviación típica
Accuracy	0.7417	0.0172
Sensitivity	0.7209	0.0630
Specificity	0.7628	0.0345
Kappa	0.4835	0.0340
Pos Pred Value	0.7544	0.0164
Neg Pred Value	0.7336	0.0349

Figura 26: Curva ROC del modelo *decision tree*



Random forest

El modelo de *random forest* presentó un valor de *accurracy* medio de 0.74 y un valor de *sensitivity* y *specificity* de 0.77 y 0.72 respectivamente (tabla 14, fig. 27). Observamos que el modelo predijo correctamente el 74% de los casos de la base de datos de prueba y que el modelo predijo mejor aquellos casos que presentan interacción con respecto a los casos que no presentan interacción. Presentó un valor de kappa próximo a 0.5 lo que nos dice que la concordancia entre los valores reales y los valores predichos fue buena. La desviación típica de las variables de efectividad fue baja, por lo tanto podríamos concluir que la partición aleatoria de los casos para los datos de entrenamiento del modelo y los datos de prueba no interfirió en la efectividad del modelo. La curva ROC (fig. 28) y el valor AUC, 0.81, reflejaron un modelo de clasificación muy bueno o incluso excelente.

Figura 27: Distribución de las principales variables de eficiencia del modelo *random forest*

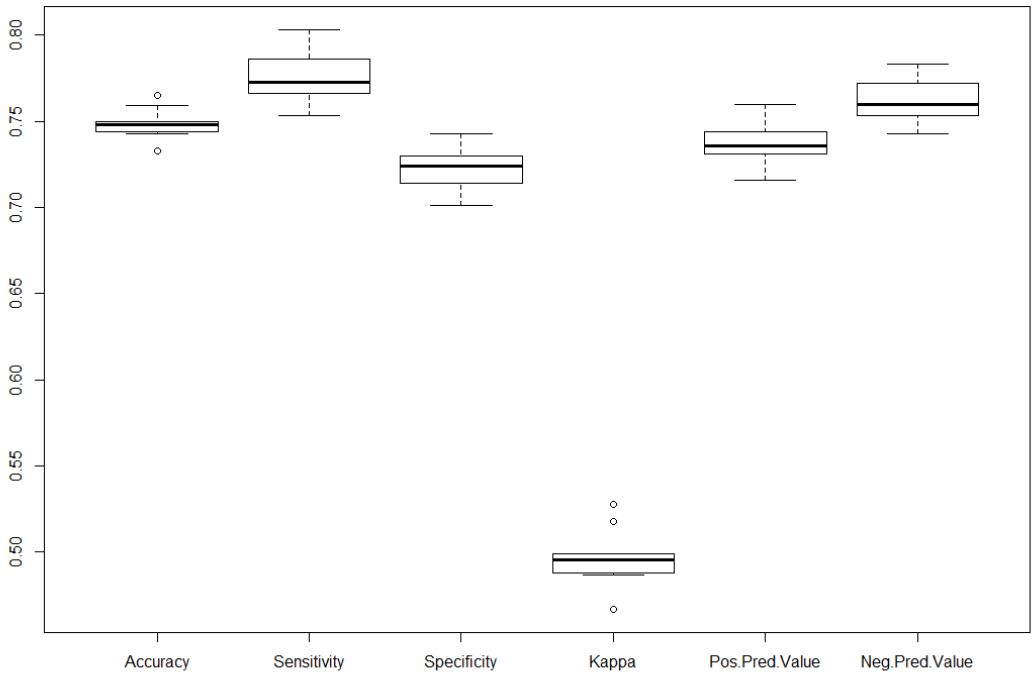
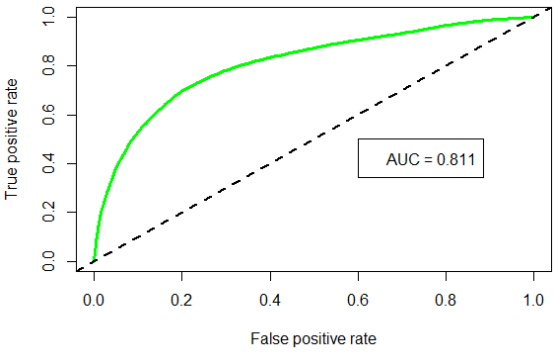


Tabla 14: Variables de eficiencia del modelo *random forest*

Random forest	Media	Desviación típica
Accuracy	0.7485	0.0087
Sensitivity	0.7753	0.0143
Specificity	0.7217	0.0132
Kappa	0.4969	0.0167
Pos Pred Value	0.7371	0.0138
Neg Pred Value	0.7615	0.0119

Figura 28: Curva ROC del modelo *random forest*



5.4. Comparación de modelos

Los modelos presentaron una *accuracy* entre 0.6 y 0.75 (tabla 15, fig. 29) donde el valor de *accuracy* más elevado lo obtuvo el modelo de *random forest,* seguido por el modelo de *decisión tree* y el modelo de SVM. El modelo que presentó peores resultados en *accuracy* fue el modelo basado en ANN, el cual también obtuvo la máxima desviación típica en los 10 entrenamientos.

La variable de efectividad *sensitivity* presentó valores más diversos que el resto de variables de efectividad (tabla 15, fig. 30). El modelo que presentó un valor más elevado de esta variable fue el modelo de *random forest*, seguido por el modelo SVM y el modelo k-nn. El modelo que presentó los valores más bajos fue el modelo de Naive Bayes seguido por el modelo ANN. Las desviaciones típicas en esta variable fueron elevadas indicando que la *sensitivity* puede estar afectada por la partición aleatoria de los datos para el entrenamiento de los modelos.

Con respecto a la variable de efectividad *specificity* presentó valores similares entre todos los modelos excepto el modelo ANN que presentó los valores más bajos (tabla 15, fig. 31). El modelo de Naive Bayes observamos que obtuvo el valor más elevado de todos los modelos. Las desviaciones típicas en esta variable también fueron elevadas.

Figura 29: Distribución de la variable *accuracy* en los modelos estudiados.

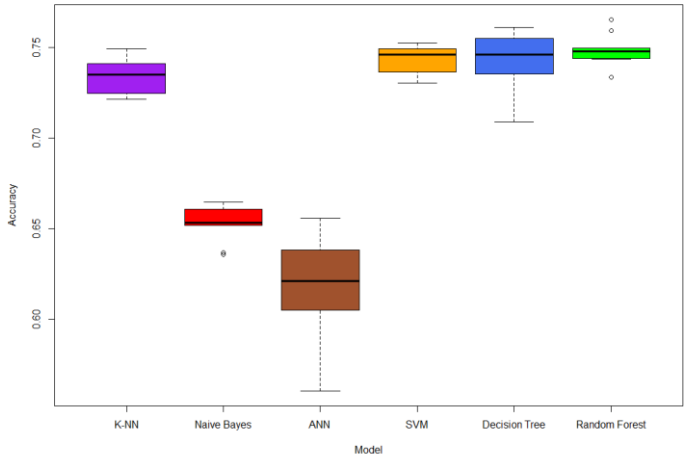


Figura 30: Distribución de la variable *sensitivity* en los modelos estudiados.

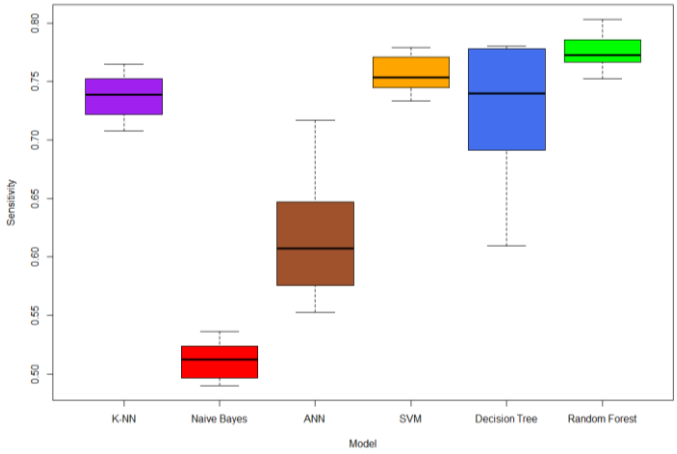
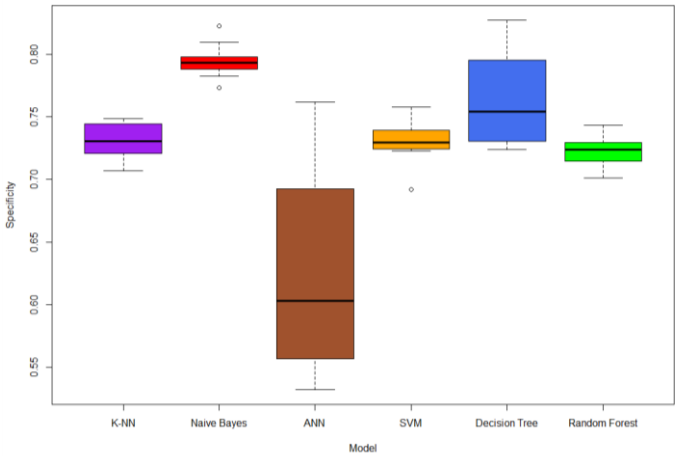


Figura 31: Distribución de la variable *specificity* en los modelos estudiados.



El valor kappa presentó valores muy similares en los modelos de k-nn, SVM, *decision tree* y *random forest*, siendo este el modelo con el valor kappa más elevado (tabla 15, fig. 32). El modelo ANN obtuvo el valor kappa más bajo de todos los modelos seguido del modelo de Naive Bayes.

Los valores de PPV y NPV nos indican la probabilidad de que una predicción positiva sea realmente positiva, y que una predicción negativa realmente sea negativa. Entre todos los modelos, el que obtuvo un valor de PPV más elevado fue el modelo de *decisión tree* (tabla 15, fig. 33), pero en cambio el modelo con un valor de NPV más elevado fue el modelo de *random forest*. El modelo de ANN obtuvo los valores más bajos de las dos variables de eficiencia (fig. 34).

Figura 32: Distribución de la variable kappa en los modelos estudiados.

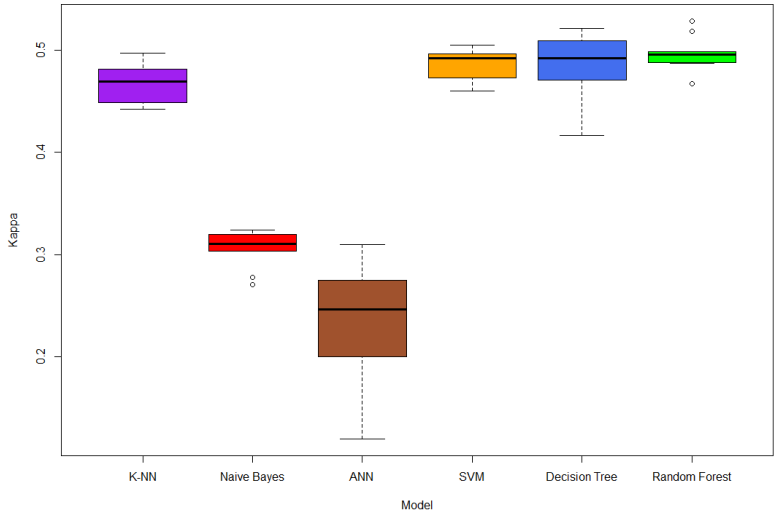


Figura 33: Distribución de la variable PPV en los modelos estudiados.

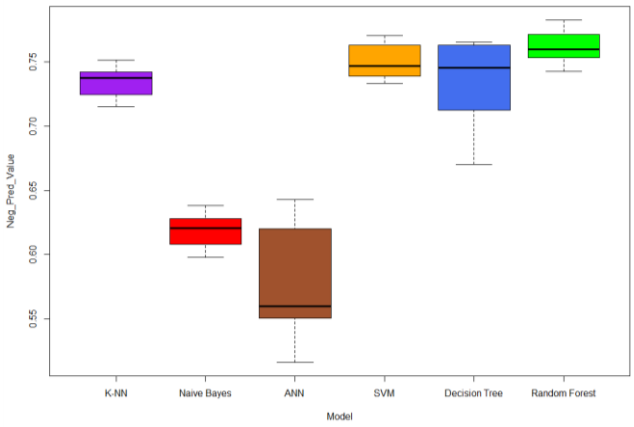


Figura 34: Distribución de la variable NPV en los modelos estudiados.

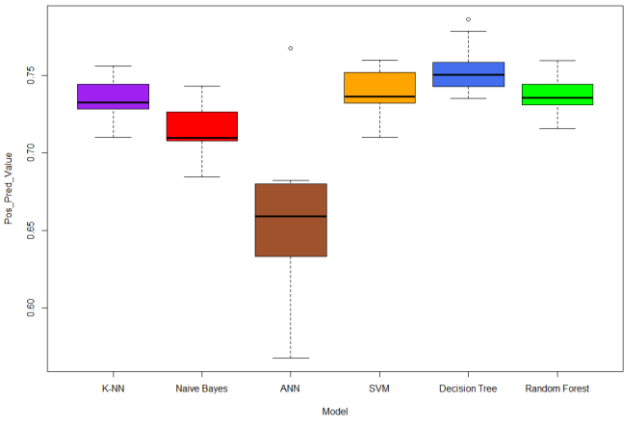


Tabla 15: Tabla de variables de efectividad de todos los modelos. En color rojo se indican los valores más bajos por variable de efectividad y en verde se indican los valores más elevados.

	Accuracy	Sensitivity	Specificity	Kappa	Pos Pred Value	Neg Pred Value
K-NN	0.734	0.737	0.731	0.468	0.734	0.734
Naive Bayes	0.653	0.511	0.795	0.306	0.715	0.618
ANN	0.617	0.613	0.623	0.233	0.658	0.579
SVM	0.744	0.756	0.731	0.487	0.739	0.749
Decision Tree	0.742	0.721	0.763	0.484	0.754	0.734
Random Forest	0.749	0.775	0.722	0.497	0.737	0.761

Al observar las curvas ROC (fig. 35) y los valores AUC (tabla 16) pudimos ver que los modelos que presentaron los valores AUC más bajos fueron los modelos de k-nn y Naive Bayes, cuyas curvas ROC se aproximaban a la diagonal. Los modelos de SVM, *random forest* y *decision tree* presentaron valores de AUC por encima de 0.8, siendo el modelo de *decisión tree* el que obtuvo el valor más elevado. Estos tres modelos podrían considerarse excelentes modelos.

Figura 35: Curva ROC de todos los modelos

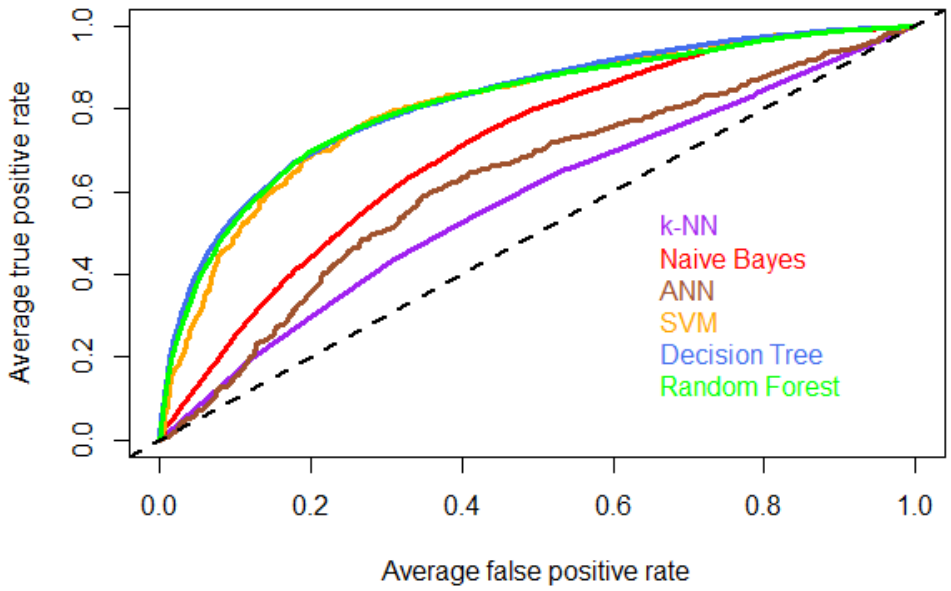


Tabla 26: Valores AUC de todos los modelos

	k-NN	Naive Bayes	ANN	SVM	Decision Tree	Random Forest
AUC	0.578	0.708	0.627	0.807	0.818	0.811

Para comprobar que existen diferencias significativas entre los modelos, y así poder discernir qué modelo es mejor para la predicción de dianas de miARN, se realizaron pruebas de suma de rangos de Wilcoxon. En las siguientes tablas se presentan los p-valores de las pruebas realizadas dos a dos para cada una de las variables de eficiencia.

Se encontraron diferencias significativas con respecto a la variable de efectividad *accuracy* entre todos los modelos excepto en cuatro casos (tabla 17). El modelo k-nn y el modelo *decision tree* no presentaron diferencias significativas. Tampoco se encontraron diferencias significativas entre los modelos de *decision tree*, SVM y *random forest.*

Tabla 17. Resultados de la suma de rangos de Wilcoxon para la variable *accuracy*. Se muestra en rojo aquellos casos donde el p-value de la prueba es inferior a 0.05.

Accuracy	k-NN	Naive Bayes	ANN	SVM	Decision Tree	Random Forest
k-NN	-	0.00589	0.00195	0.00915	0.1389	0.00586
Naive Bayes		-	0.00195	0.00589	0.00195	0.00589
ANN			-	0.00195	0.00195	0.00195
SVM				-	0.83836	0.13086
Decision Tree					-	0.19336
Random Forest						-

Se encontraron diferencias significativas en la variable *sensitivity* entre todas las parejas de modelos excepto en dos casos (tabla 18). Los modelos *decision tree* y k-nn no presentaron diferencias significativas entre ellos, al igual que *decisión tree* y el modelo SVM.

Tabla 18: Resultados de la suma de rangos de Wilcoxon para la variable *sensitivity*. Se muestra en rojo aquellos casos donde el p-value de la prueba es inferior a 0.05.

Sensitivity	k-NN	Naive Bayes	ANN	SVM	Decision Tree	Random Forest
k-NN	-	0.00195	0.00195	0.00391	0.76953	0.00195
Naive Bayes		-	0.00195	0.00195	0.00195	0.00195
ANN			-	0.00195	0.00391	0.00195
SVM				-	0.13086	0.00195
Decision Tree					-	0.00586
Random Forest						-

La variable *specificity* presentó diferencias significativas en todos los modelos excepto en un caso (tabla 19). Los modelos k-nn y SVM no presentaron diferencias significativas.

Tabla 19: Resultados de la suma de rangos de Wilcoxon para la variable *specificity*. Se muestra en rojo aquellos casos donde el p-value de la prueba es inferior a 0.05.

Specificity	k-NN	Naive Bayes	ANN	SVM	Decision Tree	Random Forest
k-NN	-	0.00195	0.01367	0.92188	0.01953	0.00195
Naive Bayes		-	0.00195	0.00195	0.03711	0.00195
ANN			-	0.02734	0.00586	0.01953
SVM				-	0.01953	0.04883
Decision Tree					-	0.00195
Random Forest						-

Se encontraron diferencias significativas entre los modelos con respecto al valor de eficiencia kappa (tabla 20). Al igual que con la variable *accuracy* no se encontraron diferencias significativas entre el modelo k-nn y el modelo *decision tree* y entre los modelos de *decision tree*, SVM y *random forest*.

Tabla 20: Resultados de la suma de rangos de Wilcoxon para la variable *kappa*. Se muestra en rojo aquellos casos donde el p-value de la prueba es inferior a 0.05.

Kappa	k-NN	Naive Bayes	ANN	SVM	Decision Tree	Random Forest
k-NN	-	0.00195	0.00195	0.00391	0.16016	0.00195
Naive Bayes		-	0.00195	0.00195	0.00195	0.00195
ANN			-	0.00195	0.00195	0.00195
SVM				-	0.92188	0.13086
Decision Tree					-	0.23242
Random Forest						-

La variable PPV no mostró diferencias significativas entre los modelos de k-NN y SVN y entre los modelos *random forest* y SVM (tabla 21).

Tabla 21: Resultados de la suma de rangos de Wilcoxon para la variable *PPV*. Se muestra en rojo aquellos casos donde el p-value de la prueba es inferior a 0.05.

PPV	k-NN	Naive Bayes	ANN	SVM	Decision Tree	Random Forest
k-NN	-	0.00195	0.00391	0.08398	0.00391	0.03711
Naive Bayes		-	0.00586	0.00195	0.00195	0.00195
ANN			-	0.00391	0.00391	0.00391
SVM				-	0.00977	0.8457
Decision Tree					-	0.00391
Random Forest						-

En la variable de efectividad NPV no se encontraron diferencias significativas entre los modelos de *decision tree* y k-nn, ANN y Naive Bayes y entre SVM y *decision tree* (tabla 22).

Tabla 3: Resultados de la suma de rangos de Wilcoxon para la variable *NPV*. Se muestra en rojo aquellos casos donde el p-value de la prueba es inferior a 0.05.

NPV	k-NN	Naive Bayes	ANN	SVM	Decision Tree	Random Forest
k-NN	-	0.00195	0.00195	0.00391	0.8457	0.00195
Naive Bayes		-	0.06445	0.00195	0.00195	0.00195
ANN			-	0.00195	0.00195	0.00195
SVM				-	0.27539	0.00195
Decision Tree					-	0.01367
Random Forest						-

6. Discusión

El objetivo de este estudio era realizar una evaluación de la viabilidad de distintos modelos basados en técnicas de *machine learning* que pudieran utilizarse para la predicción de dianas de miARN y a su vez cuantificar y comparar los modelos construidos para detectar qué modelos tienen mayor precisión a la hora de identificar interacciones miARN-ARNm.

Para este estudio se consiguió crear seis modelos con metodologías de *machine learning* distintas para la predicción de dianas. Cinco de estos modelos ya habían sido utilizados de forma regular en la predicción de dianas de miARN y también para la detección de pre-miARN a partir de miARNs maduros. Adicionalmente se eligió un sexto modelo (k-nn) que no aparece habitualmente en este tipo de estudios. Para todos los modelos se utilizó la misma base de datos que contenía una lista de interacciones miARN-ARNm y el cálculo de sus *features* y fueron divididos 10 veces de forma aleatoria, siguiendo la técnica de *cross-validation*, en una base de datos para entrenar los modelos y en una base de datos para comprobar su precisión.

Todos los modelos presentaron valores de efectividad significativos para predecir dianas de miARN con la base de datos construida para ello. El modelo que obtuvo peores resultados fue el modelo clasificador ANN que en 5 de las 6 variables de efectividad estudiadas mostró los niveles más bajos comparado con el resto de modelos. Por el contrario el modelo basado en *random forest* presentó los mejores resultados en 4 de 6 de las variables de efectividad, con la *accuracy* más elevada de todos los modelos. Las curvas ROC indicaron que los modelos más sólidos fueron SVM, *decision tree* y *random forest*.

Con el modelo ANN se encontraron dificultades desde el comienzo del estudio. Los modelos basados en redes neuronales requieren mucha capacidad de procesamiento. El tamaño de la base de datos era muy grande, a su vez que el número de *features* de estudio también era un poco elevado. Estas condiciones llevaron a que no se pudiera entrenar el modelo con toda la muestra y se tuviera que reducir hasta que el hardware fuera capaz de procesar y crear el modelo. Finalmente se logró entrenar el modelo con solo el 10% de la muestra y por lo tanto se perdieron el 90% de los datos para el entrenamiento. Este problema pudo provocar que el modelo presentara una efectividad muy reducida, dándonos como resultado el peor modelo de entre todos los entrenados. Este resultado discrepa con la bibliografía estudiada. Probablemente con más potencia de procesamiento y más pruebas, considerando diferentes parámetros para el modelo, se podría lograr una mejora sustancial en las predicciones del mismo y podríamos obtener, probablemente, unos valores de efectividad muy parecidos a SVM y *decisión tree*.

El modelo propuesto de *k-nearest neighbors,* aun mostrando unos valores de eficiencia elevados, no consiguieron ser equiparables a los valores de eficiencia que consiguieron los modelos de SVM, *decision tree* y *random forest*. Uno de los principales problemas de los modelos clasificadores basados en k-nn es que al aplicarse con bases de datos con muchas variables pierde efectividad y capacidad de predicción. Actualmente en los estudios de predicción de dianas de miARN se utilizan un gran número de *features* para ello, debido a que todavía no se conocen aquellas que mejor determinan una interacción. En el futuro, si se llegan a conocer estas *features* podría volver a considerarse, y probar, este tipo de modelos para la predicción de dianas miARN.

Las pruebas realizadas de suma de rangos de Wilcoxon indicaron que no existían diferencias significativas entre los modelos SVM, *decision tree* y *random forest* en la variable de efectividad *accuracy* pero sí en las variables *sensitivity* y *specificity*, con la *sensitivity* más elevada para el modelo de *random forest*.

Podríamos afirmar que el modelo que obtuvo una mejor predicción de dianas de miARN fue el modelo de *random forest* (75%). Dados los buenos resultados del modelo de *random forest* podríamos considerar este modelo como el más adecuado para el estudio de predicción de dianas de miARN. En contraposición con el resto de modelos, el *random forest* presenta un buen funcionamiento en la mayoría de problemas de clasificación que involucran grandes bases de datos, y en la actualidad, ya comentado anteriormente, todavía nos encontramos en una situación de desconocimiento de qué *features* son las determinantes para lograr predicciones precisas, por lo que se hace necesario el uso de grandes cantidades de información.

Las *features* que presentaron una mayor importancia a la hora de la identificación y predicción de dianas de miARN, en el modelo de *random forest*, fueron las *features* de energía libre de Gibbs y la energía de accesibilidad al sitio de unión, seguido de aquellas relacionadas con la estructura de la semilla y la secuencia del miARN, porcentajes de pares de bases AU y CG y porcentajes en la secuencia de nucleótidos C y A. Aunque en la bibliografía siempre se otorga mucha importancia al número de pares de bases encajadas en la secuencia de la semilla, en nuestro estudio fueron las *features* con menos importancia para predecir si existe o no interacción del miARN con el ARNm.

El número de *features* utilizadas para este estudio es reducido en comparación con el que suelen utilizar los modelos construidos por la comunidad. El número de *features* en nuestro estudio estuvo limitado por el tiempo de realización del trabajo y la enorme dificultad de cálculo de las mismas. Algunas de las variables que no se han tenido en cuenta podrían ser importantes, como por ejemplo el número de *loops* o bultos en el dúplex miARN-ARNm o la conservación evolutiva. Probablemente si se aumentara el número de *features* se conseguiría aumentar la capacidad de predicción de los modelos, pero estudios anteriores reflejaban que a partir de un número de *features* no se mostraba un aumento significativo de la precisión. En estudios futuros sería más relevante centrar la atención en estudiar qué *features* realmente afectan de forma significativa a la predicción de dianas en vez de generar cientos de *features* e introducirlas todas en la creación de una herramienta de predicción.

7. Conclusión

Se ha demostrado que los miARNs están involucrados en un amplio espectro de procesos biológicos. A pesar de que miles de microRNA se identifican experimentalmente de forma continua, para muchos de ellos las dianas objetivo todavía no se han identificado. Es importante desarrollar algoritmos eficientes para la predicción de dianas de miARN, y así ampliar nuestra comprensión sobre la regulación génica post-transcripcional. Con el estudio realizado se ha confirmado que los modelos basados en técnicas de *machine learining* son viables para el estudio de predicción de dianas de miARN y pueden aportar una vía de investigación muy potente e importante. Estos métodos facilitarán la identificación de nuevas dianas objetivo y vías post-trascripcionales de miARNs sin los enormes costes que conlleva la identificación de forma experimental.

8. Glosario

Machine learning: Disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente.

Artificial Neural Network (ANN): Modelos de aprendizaje supervisado inspirados en las redes neuronales biológicas.

Support Vector machine (SVM): Modelos de aprendizaje supervisados que analizan los datos utilizados para la clasificación, regresión e identificación de valores atípicos de los mismos.

k-nearest neighbors (k-nn): Modelos de clasificación no paramétricos, que estiman el valor de la función de densidad de probabilidad o directamente la probabilidad a posteriori de que un elemento *x* pertenezca a la clase *c* a partir de la información de otros elementos previos.

Decision tree: Modelos de aprendizaje supervisados donde los datos se dividen continuamente según un parámetro determinado. El árbol puede ser explicado por dos entidades, a saber, nodos de decisión y hojas. Las hojas son las decisiones o los resultados finales y los nodos de decisión, donde se dividen los datos.

Random forest: Modelos de clasificación que combinan árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos.

miARN: ARN monocatenario, de una longitud de entre 21 y 25 nucleótidos, y que tiene la capacidad de regular la expresión de otros genes mediante diversos procesos.

Cross-validation: Método para estimar la precisión (o error) de un modelo al dividir los datos en *k* subconjuntos mutuamente excluyentes El modelo se entrena y se prueba *k* veces. La estimación de precisión es la precisión promedio para los *k* modelos.

Feature: Característica, atributo.

Modelo: Estructura e interpretación correspondiente que resume o resume parcialmente un conjunto de datos.

Supervised learning: Técnicas utilizadas para aprender la relación entre atributos independientes y un atributo dependiente designado.

Confusion matrix: Una matriz que muestra las clasificaciones predichas y reales.

ARN mensajero: Ácido ribonucleico que transfiere el código genético procedente del ADN del núcleo celular a un ribosoma en el citoplasma.

Dúplex miARN-ARNm: estructura secundaria formada por la interacción del microARN con su ARNm diana.

9. Referencias

[1] R. C. Lee, R. L. Feinbaum, and V. Ambros, “The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14,” *Cell*, vol. 75, no. 5, pp. 843–854, 1993.

[2] T. Zhao, G. Li, S. Mi, S. Li, G. J. Hannon, X. J. Wang, and Y. Qi, “A complex system of small RNAs in the unicellular green alga Chlamydomonas reinhardtii,” *Genes Dev.*, vol. 21, no. 10, pp. 1191–1203, 2007.

[3] B. Wightman, I. Ha, and G. Ruvkun, “Posttranscriptional regulation of the heterochronic gene lin-14 by lin-4 mediates temporal pattern formation in C. elegans,” *Cell*, vol. 75, no. 5, pp. 855–862, 1993.

[4] B. J. Reinhart, F. J. Slack, M. Basson, A. E. Pasquinelll, J. C. Bettlnger, A. E. Rougvle, H. R. Horvitz, and G. Ruvkun, “The 21-nucleotide let-7 RNA regulates developmental timing in Caenorhabditis elegans,” *Nature*, vol. 403, no. 6772, pp. 901–906, 2000.

[5] A. Lugo-Trampe and K. Trujillo-Murillo, “MicroRNAs: reguladores clave de la expresión génica,” *Med. Univ.*, vol. 11, no. 44, pp. 187–192, 2009.

[6] S. Gu and M. A. Kay, “How do miRNAs mediate translational repression?,” *Silence*, vol. 1, no. 1, 2010.

[7] S. M. Peterson, J. A. Thompson, M. L. Ufkin, P. Sathyanarayana, L. Liaw, and C. B. Congdon, “Common features of microRNA target prediction tools,” *Frontiers in Genetics*, vol. 5, no. FEB. 2014.

[8] S. Bagga, J. Bracht, S. Hunter, K. Massirer, J. Holtz, R. Eachus, and A. E. Pasquinelli, “Regulation by let-7 and lin-4 miRNAs results in target mRNA degradation,” *Cell*, vol. 122, no. 4, pp. 553–563, 2005.

[9] J. G. Doench and P. a Sharp, “Specificity of microRNA target selection in translational repression,” *Genes Dev.*, vol. 18, no. 5, pp. 504–511, 2004.

[10] L. P. Lim, N. C. Lau, P. Garrett-Engele, A. Grimson, J. M. Schelter, J. Castle, D. P. Bartel, P. S. Linsley, and J. M. Johnson, “Microarray analysis shows that some microRNAs downregulate large numbers of-target mRNAs,” *Nature*, vol. 433, no. 7027, pp. 769–773, 2005.

[11] B. P. Lewis, I. H. Shih, M. W. Jones-Rhoades, D. P. Bartel, and C. B. Burge, “Prediction of Mammalian MicroRNA Targets,” *Cell*, vol. 115, no. 7, pp. 787–798, 2003.

[12] B. P. Lewis, C. B. Burge, and D. P. Bartel, “Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are microRNA targets,” *Cell*, vol. 120, no. 1, pp. 15–20, 2005.

[13] H. Min and S. Yoon, “Got target?: computational methods for microRNA target prediction and their extension,” *Exp. Mol. Med.*, vol. 42, p. 233, Feb. 2010.

[14] D. Grün, Y.-L. Wang, D. Langenberger, K. C. Gunsalus, and N. Rajewsky, “microRNA Target Predictions across Seven Drosophila Species and Comparison to Mammalian Targets,” *PLoS Comput. Biol.*, vol. 1, no. 1, p. e13, 2005.

[15] A. J. Enright, B. John, U. Gaul, T. Tuschl, C. Sander, and D. S. Marks, “MicroRNA targets in Drosophila.,” *Genome Biol.*, vol. 5, no. 1, p. R1, 2003.

[16] M. Sturm, M. Hackenberg, D. Langenberger, and D. Frishman, “TargetSpy: A supervised machine learning approach for microRNA target prediction,” *BMC Bioinformatics*, 2010.

[17] S.-K. Kim, J.-W. Nam, J.-K. Rhee, W.-J. Lee, and B.-T. Zhang, “miTarget: microRNA target gene prediction using a support vector machine.,” *BMC Bioinformatics*, vol. 7, p. 411, 2006.

[18] M. Yousef, S. Jung, A. V. Kossenkov, L. C. Showe, and M. K. Showe, “Naïve Bayes for microRNA target predictions - Machine learning for microRNA targets,” *Bioinformatics*, vol. 23, no. 22, pp. 2987–2992, 2007.

[19] H. Liu, D. Yue, Y. Chen, S. J. Gao, and Y. Huang, “Improving performance of mammalian microRNA target prediction,” *BMC Bioinformatics*, vol. 11, 2010.

[20] S. Cheng, M. Guo, C. Wang, X. Liu, Y. Liu, and X. Wu, “MiRTDL: A Deep Learning Approach for miRNA Target Prediction,” *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 13, no. 6, pp. 1161–1169, 2016.

[21] S. Bandyopadhyay, D. Ghosh, R. Mitra, and Z. Zhao, “MBSTAR: multiple instance learning for predicting specific functional binding sites in microRNA targets,” *Sci. Rep.*, vol. 5, p. 8004, Jan. 2015.

[22] B. Lee, J. Baek, S. Park, and S. Yoon, “deepTarget: End-to-end Learning Framework for microRNA Target Prediction using Deep Recurrent Neural Networks,” *Bioinformatics*, vol. 10, 2016.

[23] J. Ding, X. Li, and H. Hu, “TarPmiR: a new approach for microRNA target site prediction,” *Bioinformatics*, vol. 32, no. 18, pp. 2768–2775, Sep. 2016.

[24] Y. Lu and C. S. Leslie, “Learning to Predict miRNA-mRNA Interactions from AGO CLIP Sequencing and CLASH Data,” *PLoS Comput. Biol.*, vol. 12, no. 7, 2016.

[25] D. H. Mathews, M. D. Disney, J. L. Childs, S. J. Schroeder, M. Zuker, and D. H. Turner, “Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure,” *Proc. Natl. Acad. Sci.*, vol. 101, no. 19, pp. 7287–7292, 2004.

[26] D. H. Turner and D. H. Mathews, “NNDB: The nearest neighbor parameter database for predicting stability of nucleic acid secondary structure,” *Nucleic Acids Res.*, vol. 38, no. SUPPL.1, 2009.

[27] J. Ding, S. Zhou, and J. Guan, “MiRenSVM: Towards better prediction of microRNA precursors using an ensemble SVM classifier with multi-loop features,” *BMC Bioinformatics*, 2010.

[28] P. Jiang, H. Wu, W. Wang, W. Ma, X. Sun, and Z. Lu, “MiPred: Classification of real and pseudo microRNA precursors using random forest prediction model with combined features,” *Nucleic Acids Res.*, vol. 35, no. SUPPL.2, 2007.

[29] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, “e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien.” 2017.

[30] B. Lantz, *Machine learning with R*, Second edi. Birmingham - Mumbai: Packt Publishing, 2015.

[31] S. Fritsch and F. Guenther, “neuralnet: Training of Neural Networks.” 2016.

[32] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis, “kernlab – An S4 Package for Kernel Methods in R,” *J. Stat. Softw.*, 2004.

[33] M. Kuhn and R. Quinlan, “C50: C5.0 Decision Trees and Rule-Based Models.” 2017.

[34] A. Liaw and M. Wiener, “Classification and Regression by randomForest,” *R news*, 2002.

[35] J. Kolodner, *Case-Based Reasoning*. 1993.

[36] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S Fourth edition*. Springer-Verlag New York, 2002.

10. Material suplementario

- **Bdatos.csv**: Base de datos con todas las interacciones y sus *features* utilizada para entrenar los modelos.

- **DatosSeC.Rmd**: Script de R para el procesamiento de las bases de datos de Lu y Leslie.

- **Features.R**: Script de R para el cálculo de las *features* de las interacciones.

- **hek293_clip_neg_new.66nt.fa**: Base de datos de Lu y Leslie con interacciones negativas

- **hek293_clip_pos_new.66nt.fa**: Base de datos de Lu y Leslie con interacciones positivas

- **miRNA.fa**: Base de datos con las secuencias de las interacciones miARN-ARNm para el programa ViennaCofold

- **Mlearning**:Script de R para la creación, entrenamiento y comparación de los modelos

- **resultsVC.txt**: Resultados del programa ViennaRNA Cofold con las energías libres de las interacciones y *bracket notation*.

- **RNA.fa**: Base de datos con las secuencias de las interacciones miARN-ARNm para el programa ViennaRNA fold

- **RNAconf.txt**: Resultados del programa ViennaRNA fold para el cálculo de energía de accesibilidad

- **tabfinal.txt:** Base de datos solo con los datos descriptivos de las interacciones y sus secuencias