PROYECTO DE BIOINFORMÁTICA – DESARROLLO COMPUTACIONAL DE FÁRMACOS

Para el desarrollo de este proyecto me baso en la **relación cuantitativa estructura-actividad** (*Quantitative structure-activity relationship*, **QSAR**). Esto quiere decir que la estructura química de los compuestos se correlaciona cuantitativamente con la actividad biológica. La actividad biológica es define como la unión de un fármaco con un receptor (proteína, organismo …)

Permite entender el origen biológico de la actividad e interpretar los modelos QSAR tiene un gran valor e importancia en el desarrollo y diseño de nuevos fármacos.

1.SELECCIONAR LOS DATOS Y PRE-PROCESADO

- **Base de datos ChEMBL**

En este caso uso la base de datos ChEMBL de la qu descargamos los datos de bioactividad de los compuestos una vez seleccionada la proteína objetivo ***Human Acetylcholinesterase.***

De la base de datos obtengo los compuestos (fármacos) que tienen un cierto tipo de efecto en la proteína/organismo objetivo. Este efecto puede ser de inhibición o de aceleración de la actividad de la proteína en este caso ***Human Acetylcholinesterase.***

Estos son los pasos que he seguido:

1. Instalar el cliente de CheMBL
2. Importar las librerías
3. Buscar los resultados disponibles para la proteína objetivo (Filtro: single protein)
4. Buscar los datos de bioactividad para dicha proteína (*standard_type = "IC50"*)
   Es importante mencionar que selecciono el tipo stanard IC5 para que mi dataset tenga unidades uniformes de bioactividad (nM)
5. He salvado los datos en un fichero *bioactivity_data.csv*

- **Pre-procesado de los datos de bioactividad**

El valor de la bioactividad lo encuentro en la columna *standard_value.* Este valor indica la potencia del fármaco. Indica la concentración de fármaco necesaria para la inhibición del 50% de la proteína. A menor valor, menor cantidad de fármaco necesitamos para crear el mismo efecto por ello, mayor potencia. Idealmente deseamos encontrar el menor valor posible.

Estos son los pasos que he seguido:

1. Si alguno de los compuestos tiene valores nulos en la columna standard_value, lo elimino
2. Etiqueto los compuestos como activo, intermedio e inativo.

Los datos de bioactividad son valores IC50 (Concentración para la inhibición de 50%) . Los compuestos con valores inferiores a 1000 nM se consideran ACTIVOS. Los que tengan valores superiores a 10000 nM se consideran INACTIVOS. Los compuestos entre ambos valores (mayores que 1000 e inferiores a 10000) son INTERMEDIOS.

3. Combino las 3 columnas *molecue_chembl_id, canonical_smiles y standard_value* después de haber eliminado las filas duplicadas en un DataFrame y salvo esa información en bioactivity_preprocessed_data.csv

2. ANÁLISIS EXPLORATORIO DE LOS DATOS

En esta sección calculo los descriptores de Lipinski y hago un análisis exploratorio de los datos.

- **Descriptores de Lipinski**

Estos descriptores dan una visión global de las propiedades de las moléculas para saber si pueden ser una buen fármaco para via oral. Christopher Lipinski, es un científico en Pfizer, que estableció una serie de reglas para evaluar la efectividad de un fármaco. Esta efectividad sebasa en el perfil farmacokinetico también conocido como ADME por sus siglas en inglés (Absorption, Distribution, Metabolism, & Extraction). Este científico analizó todos los fármacos activos de via oral aprovado por la FDA americana y estabeció lo que se conoce como the Rule-of-Five of Lipinski's Rule.

The Lipinski's Rule dice lo siguiente:

- El peso molecular (MW) <500 Dalton
- EL coefficiente de particion Octanol-agua (LogP)<5
- Enlaces de hidrogeno donantes <5
- Enlaces de hidrógeno aceptadores <10

Estos son los pasos que he seguido:

1. Instalo conda en Google Colab e importo la librería rdkit, librería de software libre para química-informática.
2. Calculo los descriptores de Lipinski usando una función modificada de Codeocean

   Los descriptores de Lipinski son el peso molecular (Molecular Weight, MW), la solubilidad (LogP) el número de enlaces de hidrogeno cedidos (NumHDonors) y el número de enlaces de hidrógeno acceptados (NumHAcceptors).

3. Combino los dataframes de la bioactividad con el resultado de los descriptores de Lipinski.
4. Fijo los valores de IC50 a un máximo de 100000000 para la posterior transformación usando la función norm_value

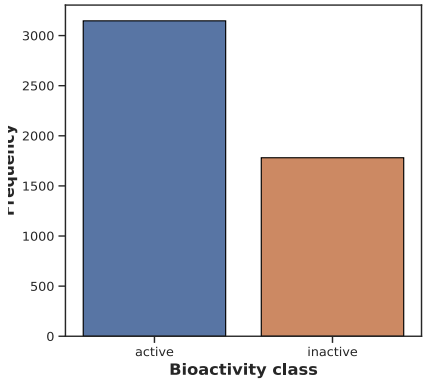
5. Convierto el valor IC50 en pIC50 aplicando una transformación logarítmica negativa para lograr una distribución de los datos más uniforme. Uso la función pIC50

6. Elimino la clase “intermediate” y me quedo sólo con las clases de compuestos activos e inactivo que había definido previamente.
7. Guardo los datos en *final_df_ipc50.csv*

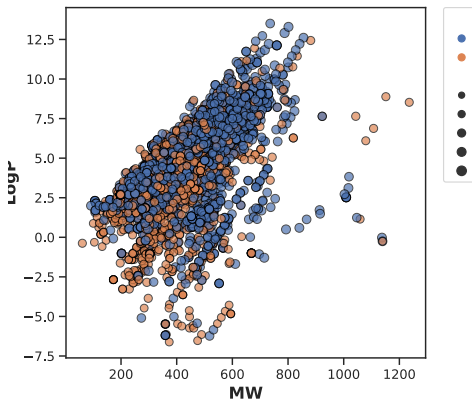
- **Análisis exploratorio del espacio químico basándome en los descriptores de Lipinski calculados.**

  Uso la función estadística Mann-Whitney U Test función modificada de [Machine Learning Mastery]

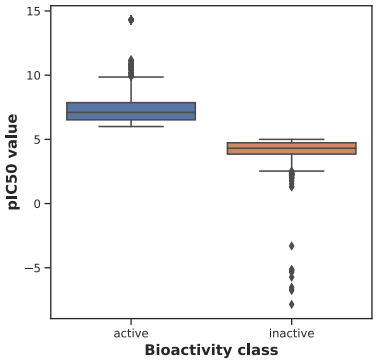
  - Gráfico de frecuencia de las 2 clases de bioactividad



  - Gráfico de dispersión peso molecular (MW - moecular weight) VS LogP (Solubilidad)

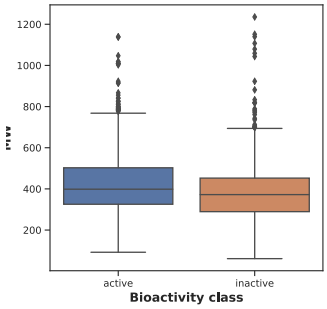


  - Gráfico de caja para pIC50 y análisis estadístico Mann-Whitney U Test



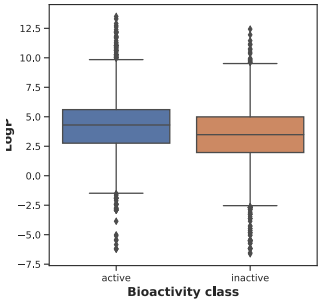
	Descriptor	Statistics	p	alpha	Interpretation
0	pIC50	0.0	0.0	0.05	Different distribution (reject H0)

- Gráfico de caja para peso molecular (Molecular weight, MW) y análisis estadístico Mann-Whitney U Test



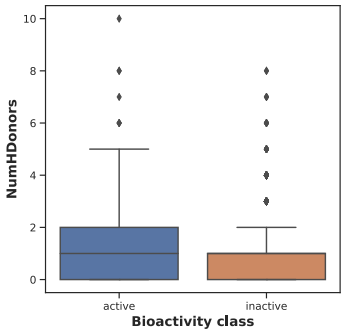
	Descriptor	Statistics	p	alpha	Interpretation
0	MW	2368074.5	7.005948e-20	0.05	Different distribution (reject H0)

- Gráfico de caja para solubilidad (LogP) y análisis estadístico Mann-Whitney U Test



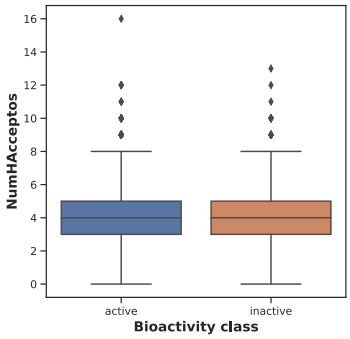
	Descriptor	Statistics	p	alpha	Interpretation
0	LogP	2206860.5	1.122125e-35	0.05	Different distribution (reject H0)

- Gráfico de caja para NumHDonors y análisis estadístico Mann-Whitney U Test



	Descriptor	Statistics	p	alpha	Interpretation
0	NumHDonors	2532960.0	1.511152e-09	0.05	Different distribution (reject H0)

- Gráfico de caja para NumHAcceptors y análisis estadístico Mann-Whitney U Test



	Descriptor	Statistics	p	alpha	Interpretation
0	NumHAcceptors	2758827.0	0.177832	0.05	Same distribution (fail to reject H0)

Interpretación de los resultados estadísticos

- pIC50 values

Mirando los valores de pIC50, las clases active y inactive muestran diferencia estadística significativa, lo cual era de esperar ya que habíamos fijado los valores de umbral previamente para crear las clases. (IC50 < 1000 nM = Active, IC50 > 10000 nM = Inactive, corresponde a pIC50 > 6 Actives , and pIC50< 5 = Inactives)

- Descriptores Lipinski

De los 4 descriptores (Lipinski's descriptors,MW, LogP, NumHDonors, NumHAcceptors) sólo LogP no muestra diferencia estadística significativa entre clases. Los otros 3 descriptores todos mostraron diferencia estadística significativa entre las clases de bioactividad "active" y "inative"

3. CÁLCULO DE LOS DESCRIPTORES PUBCHEM FINGERPRINTY PREPARACIÓN DE LOS DATOS PARA ML

En esta parte he calculado unos descriptores que describen las características locales de las moléculas que usaré para construir los modelos de ML. Estos descriptores se llaman descriptores PubChem Fingerprint que describen cuantitativamente los compuestos del dataset.

Estos son los pasos que he seguido:

1. Descargar el software PaDEL y aplicarlo a los datos. Genera un resultado *descriptors_output.csv.*

   El software PaDEL usa java y un fichero PaDEL_descriptor.jar que ejecuto. Se encarga de limpiar los valores de la variable canonical_smiles que contiene información de la estructura de los compuestos. Elimina las sales Cloro y Socio (Cl, Na) y elimina también los ácidos orgánicos para que no haya impurezas.
2. Separo los datos en matriz X e Y (target). Elimino el nombre de los compuesto (Name) de la matriz X . Esta matrix contiene los descriptores PubChem Fingerprint. La Y contiene el valor pIC50 de bioactividad.

3. CONSTRUIR MODELO SIMPLE DE REGRESIÓN RANDOM FOREST

Con la X e Y creadas anteriormente pruebo a construir un modelo de regresión de Random Forest. Estos son los pasos que he seguido:

1. Elimino las variables con varianza baja.
2. Divido los datos en train (80%) y test (20%)
3. Creo un modelo con el siguiente código y obtengo un valor de 0.47 para R^2.

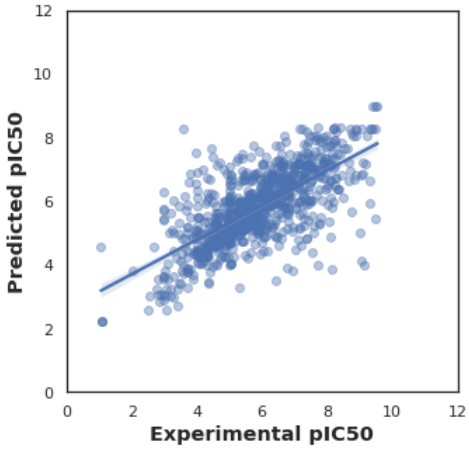
```
model = RandomForestRegressor(n_estimators=100)
model.fit(X_train, Y_train)
r2 = model.score(X_test, Y_test)
r2
```

```
0.47790169996302834
```

```
Y_pred = model.predict(X_test)
Y_pred
```

```
array([7.16037167, 4.2203357 , 5.95399928, 6.05265718, 4.36899365,
       5.62061571, 6.91673308, 6.05848668, 6.03501957, 7.38607628,
       5.14093255, 5.902377  , 6.74353551, 4.56137882, 5.22722398,
       5.58079553, 5.89873062, 4.58952284, 6.779797  , 5.35993709,
       5.40687747, 6.83380314, 6.75900428, 5.71213682, 6.11559223,
       6.71775439, 5.04374842, 5.74226371, 5.14103778, 6.258384  ,
       5.23779936, 5.27968879, 4.09975188, 7.95312671, 6.98783157,
       6.71491805, 5.79703944, 5.38189259, 5.14615861, 4.81294073,
       5.14543544, 5.68354758, 5.84827515, 6.99903652, 5.23367232,
       5.70220509, 4.74403937, 7.13587812, 5.1094284 , 6.42292015,
       7.59546272, 5.42281093, 6.62937751, 6.15571642, 7.68240808,
       8.31542212, 6.73715792, 7.89519161, 8.06607686, 1.06939512,
       5.18549413, 5.24148605, 4.71000891, 4.56939017, 5.46662778,
       3.78109562, 5.82145576, 4.98845423, 6.00110104, 8.03410307,
       6.41494684, 4.98333372, 4.09951225, 4.5459131 , 7.66722   ,
       5.95540225, 5.4813585 , 5.6006411 , 6.71775439, 6.40944372,
       4.76726264, 5.60762815, 5.46474816, 7.64040489, 4.42433982,
       4.02496464, 5.8052954 , 5.46017554, 5.81489633, 5.34815503,
       6.34065847, 4.88566011, 6.42514598, 6.75101848, 5.22016036,
       5.64159055, 5.24468252, 8.29762306, 6.91111279, 5.23179325,
       7.25392078, 6.78388652, 4.74635757, 4.80194732, 8.28445034,
       6.32269667, 5.44002986, 6.40014202, 6.01019399, 5.71707536,
       6.08332937, 5.60972609, 7.05035882, 8.29762306, 8.10044703,
       4.75361649, 6.53619333, 6.04265911, 5.83057821, 4.80642792,
```

4. Gráfico de dispersión de los valores experimentales Vs los valores predichos.
   Se puede observar que el modelo no predice con demasiada exactitud los valores de pIC50.



5.MODELOS DE REGRESIÓN MACHINE LEARNING

En esta sección voy a construir modelos de ML para los datos de bioactividad para la proteína Human Acetylcholinesterase basándome en los predictores PubChem Fingerprint. Estos predictores son huellas únicas para cada compuesto que los identifican y les confieren propiedades únicas. Esto es esencial para el desarrollo de fármacos y su diseño.

La conectividad entre las moléculas da lugar a estructuras únicas del compuesto y también propiedades. Necesitamos encontrar una forma de reajustar dichas moléculas de tal manera que el compuesto final, el fármaco sea lo más potente posible a la vez que lo menos tóxico y más seguro para el consumo.

El algoritmo de ML aprende de los descriptores PubChem Fingerprint, de las propiedades moleculares únicas de cada compuesto. El objetivo es crear un modelo que sepa distinguir correctamente entre compuesto ACTIVO/INACTIVOS. Así podremos ver que grupos funcionales son esenciales para diseñar un fármaco potente y seguro. Es por eso que la variable dependiente, Y (target) será el valor pIC50 con el que hemos estado trabajando. Este es el valor de la concentración de compuesto necesario para la inhibición del 50% de la proteína.

Para este propósito he usado la librería Lazypredict que me permite construir 39 algoritmos básicos presentes en scikit-learn simulatenamente. Es importante decir que son entrenado con los parámetro por defecto y que se podría mejorar su rendimiento aplicando optimización de hyperparámetros con GridSearch por ejemplo.

```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
import lazypredict
from lazypredict.Supervised import LazyRegressor
```

```
clf = LazyRegressor(verbose=0,ignore_warnings=True,custom_metric=None)
train,test = clf.fit(X_train, X_test, Y_train, Y_test)
```

```
100%|██████████| 42/42 [00:54<00:00,  1.29s/it]
```

Estos son los resultados de entrenamiento

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
LGBMRegressor	0,335710182	0,432733322	1,144772973	0,411873102
HistGradientBoostingRegressor	0,335710181	0,43273332	1,144772975	9,669913769
RandomForestRegressor	0,333462308	0,430813762	1,146708222	5,475815296
BaggingRegressor	0,30732689	0,408495564	1,168973683	1,016167164
XGBRegressor	0,304099331	0,405739407	1,171693976	2,047224045
NuSVR	0,270576345	0,377112636	1,199583522	4,491662025
SVR	0,269864335	0,376504619	1,200168852	4,28390193
MLPRegressor	0,256561831	0,365145018	1,211052574	11,50372481
KNeighborsRegressor	0,233872906	0,345769933	1,229393693	1,616433144
GradientBoostingRegressor	0,231254787	0,343534205	1,231492531	2,113032818
LassoCV	0,118006123	0,246826124	1,319086741	3,845662594
RidgeCV	0,11722216	0,246156663	1,319672848	0,14512229
ElasticNetCV	0,117135746	0,246082871	1,319737437	5,277480841
TransformedTargetRegressor	0,116817374	0,245810998	1,319975373	0,0825212
LinearRegression	0,116817374	0,245810998	1,319975373	0,119082212
Ridge	0,116787029	0,245785085	1,319998049	0,053330898
BayesianRidge	0,115355429	0,244562578	1,321067408	0,297316313
LassoLarsCV	0,114274201	0,24363927	1,321874477	0,533579588
LassoLarsIC	0,113886042	0,243307803	1,322164094	0,128583431
PoissonRegressor	0,095602864	0,227694983	1,335734549	0,168862343
DecisionTreeRegressor	0,086931555	0,220290166	1,342122747	0,183002949
SGDRegressor	0,085859974	0,219375095	1,342910077	0,13952589
ExtraTreesRegressor	0,080786504	0,215042633	1,346631491	10,16059279
HuberRegressor	0,080052541	0,214415869	1,347169004	1,21950531
LarsCV	0,076530865	0,211408553	1,349745109	0,507240772
AdaBoostRegressor	0,056759617	0,194525003	1,364117445	0,673128605
OrthogonalMatchingPursuit	0,056741168	0,194509249	1,364130786	0,071005583
LinearSVR	0,053590454	0,191818714	1,36640715	0,931522369
OrthogonalMatchingPursuitCV	0,04962502	0,188432453	1,369266765	0,228033066
GeneralizedLinearRegressor	0,042428617	0,182287124	1,37444115	0,141920567
TweedieRegressor	0,042428617	0,182287124	1,37444115	0,079991817
GammaRegressor	0,040062416	0,180266519	1,376138255	0,06520009
ExtraTreeRegressor	0,025932185	0,168200086	1,386229596	0,183104038
Lars	0,020909365	0,163910875	1,389799075	0,119549274
Lasso	-0,171248227	-0,000181055	1,520075094	0,058181524
ElasticNet	-0,171248227	-0,000181055	1,520075094	0,070233822
DummyRegressor	-0,171248227	-0,000181055	1,520075094	0,072525024
LassoLars	-0,171248227	-0,000181055	1,520075094	0,060159445
PassiveAggressiveRegressor	-0,859843168	-0,588202961	1,915484896	0,15114212
GaussianProcessRegressor	-6,796742262	-5,657985663	3,921905807	6,941093445
KernelRidge	-16,57304305	-14,00640457	5,887949228	1,485174179
RANSACRegressor	-7,22847E+22	-6,17271E+22	3,77628E+11	1,650134802

Estos son los resultados de test:

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
LGBMRegressor	0,335710182	0,432733322	1,144772973	0,411873102
HistGradientBoostingRegressor	0,335710181	0,43273332	1,144772975	9,669913769
RandomForestRegressor	0,333462308	0,430813762	1,146708222	5,475815296
BaggingRegressor	0,30732689	0,408495564	1,168973683	1,016167164
XGBRegressor	0,304099331	0,405739407	1,171693976	2,047224045
NuSVR	0,270576345	0,377112636	1,199583522	4,491662025
SVR	0,269864335	0,376504619	1,200168852	4,28390193
MLPRegressor	0,256561831	0,365145018	1,211052574	11,50372481
KNeighborsRegressor	0,233872906	0,345769933	1,229393693	1,616433144
GradientBoostingRegressor	0,231254787	0,343534205	1,231492531	2,113032818
LassoCV	0,118006123	0,246826124	1,319086741	3,845662594
RidgeCV	0,11722216	0,246156663	1,319672848	0,14512229
ElasticNetCV	0,117135746	0,246082871	1,319737437	5,277480841
TransformedTargetRegressor	0,116817374	0,245810998	1,319975373	0,0825212
LinearRegression	0,116817374	0,245810998	1,319975373	0,119082212
Ridge	0,116787029	0,245785085	1,319998049	0,053330898
BayesianRidge	0,115355429	0,244562578	1,321067408	0,297316313
LassoLarsCV	0,114274201	0,24363927	1,321874477	0,533579588
LassoLarsIC	0,113886042	0,243307803	1,322164094	0,128583431
PoissonRegressor	0,095602864	0,227694983	1,335734549	0,168862343
DecisionTreeRegressor	0,086931555	0,220290166	1,342122747	0,183002949
SGDRegressor	0,085859974	0,219375095	1,342910077	0,13952589
ExtraTreesRegressor	0,080786504	0,215042633	1,346631491	10,16059279
HuberRegressor	0,080052541	0,214415869	1,347169004	1,21950531
LarsCV	0,076530865	0,211408553	1,349745109	0,507240772
AdaBoostRegressor	0,056759617	0,194525003	1,364117445	0,673128605
OrthogonalMatchingPursuit	0,056741168	0,194509249	1,364130786	0,071005583
LinearSVR	0,053590454	0,191818714	1,36640715	0,931522369
OrthogonalMatchingPursuitCV	0,04962502	0,188432453	1,369266765	0,228033066
GeneralizedLinearRegressor	0,042428617	0,182287124	1,37444115	0,141920567
TweedieRegressor	0,042428617	0,182287124	1,37444115	0,079991817
GammaRegressor	0,040062416	0,180266519	1,376138255	0,06520009
ExtraTreeRegressor	0,025932185	0,168200086	1,386229596	0,183104038
Lars	0,020909365	0,163910875	1,389799075	0,119549274
Lasso	-0,171248227	-0,000181055	1,520075094	0,058181524
ElasticNet	-0,171248227	-0,000181055	1,520075094	0,070233822
DummyRegressor	-0,171248227	-0,000181055	1,520075094	0,072525024
LassoLars	-0,171248227	-0,000181055	1,520075094	0,060159445
PassiveAggressiveRegressor	-0,859843168	-0,588202961	1,915484896	0,15114212
GaussianProcessRegressor	-6,796742262	-5,657985663	3,921905807	6,941093445
KernelRidge	-16,57304305	-14,00640457	5,887949228	1,485174179
RANSACRegressor	-7,22847E+22	-6,17271E+22	3,77628E+11	1,650134802

Gráfico de barras para los modelos valores VS R^2.

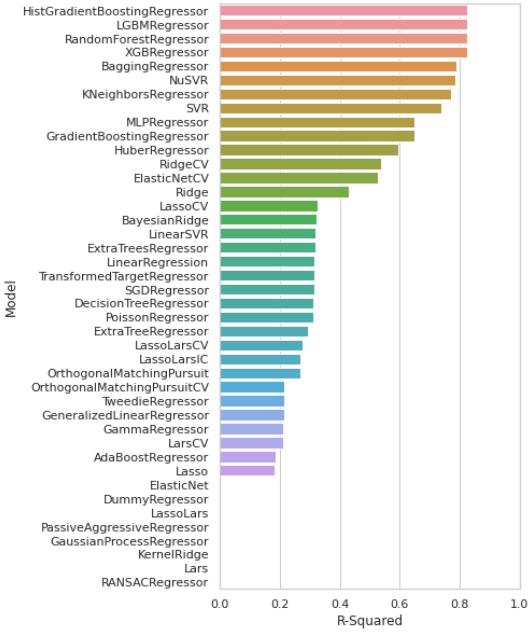


Gráfico de barras para los modelos valores VS RMSE

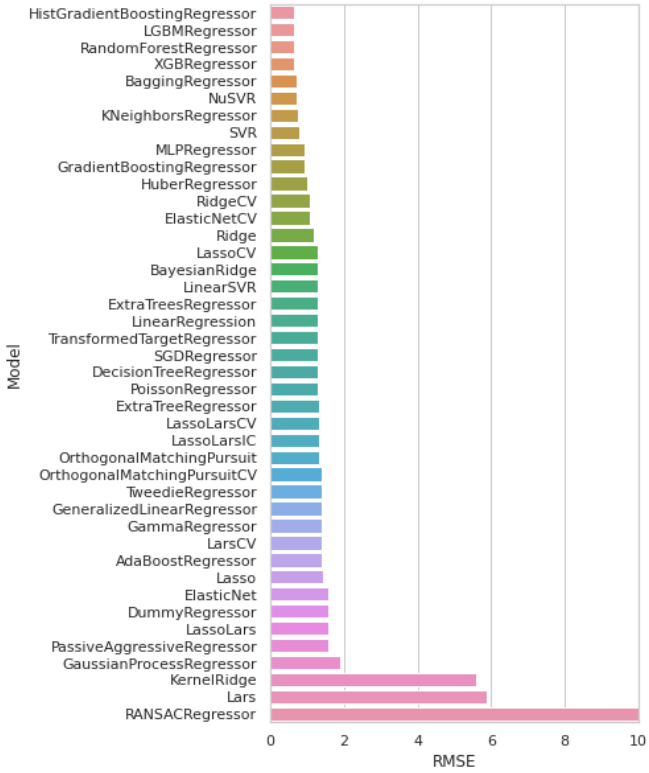
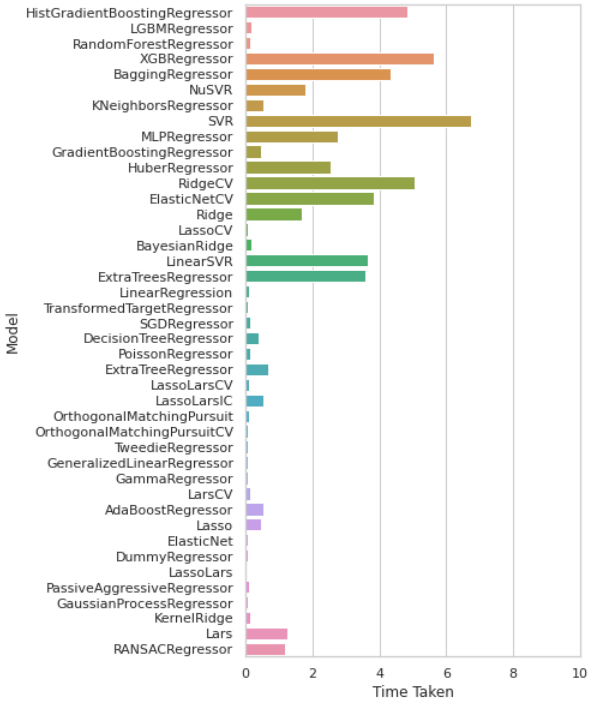


Gráfico de barras para los modelos VS tiempo



Conclusiones

El rendimiento de los 39 modelos no es para nada bueno. El mejor predictor es el modelo LGBMRegressor el cual alcanza un valor de R2 de 0.4. Esto puede deberse a que al aplicar Lazypredict estamos usando los parámetros que están establecidos por defecto para los algoritmos.

Mi estrategia sería aplicar un GridSearch individualmente a los 5 primeros modelos para ver si la predicción mejora. En segundo lugar intentaría encontrar una proteína con mayor cantidad de compuestos (+ 5000 registros) y haría un análisis más exhaustivo de las variables y su importancia.

Este ha sido un proyecto interesante del que he aprendido mucho aunque creo que la parte biológica y del EDA me ha ocupado 80% del tiempo del trabajo dejando 20% de dedicación a la parte de ML la cual podría ser mejorada en muchos aspectos.

Aprendizajes:

- Trabajar en Google Colab
- Base de datos ChEMBL
- Instalación de rdkit
- Descriptores Lipinski
- IC50/pIC50
- Man Whitney test estadístico
- PubChem fingerprint descriptores
- Software PaDEL
- LazyPredict librería

Problemas que he tenido que resolver:

- Incompatibilidad de librerías (ej: scikit-learn y LazyPredict)
- Transformación de object a string y a float
- Funciones que he usado de internet como Lipinski he tenido que modificarla para que funcoine con mi código