UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

SELECCIÓ D'AUTOENCODERS AMB VALIDACIÓ CREUADA

GRAU EN ESTADÍSTICA

Autor: Carlos Segura Ramiro
Director: Ferran Reverter Comes
Departament: Genètica, Microbiologia i Estadística
Convocatòria: 2023-2024

RESUM

Aquest treball de fi de grau se centra en l'ús d'autoencoders per a la reducció de dimensions i l'anàlisi de la reconstrucció de les dades. Els autoencoders són una tècnica d'aprenentatge automàtic que permet codificar i descodificar dades, preservant la informació essencial mentre se'n redueix la dimensió. L'estudi investiga la determinació del nombre òptim de nodes a la capa oculta mitjançant la validació creuada i compara diferents mètriques com ara són l'AIC, l'R^2 i el SSE per a la selecció de models. Utilitzant diferents bases de dades, el treball avalua l'efectivitat dels autoencoders a la reconstrucció d'aquestes. S'implementa el model utilitzant el paquet Keras, destacant-ne la capacitat per crear xarxes neuronals avançades i entrenar-les eficientment. Els resultats mostren que és possible reduir significativament la dimensió de les dades sense una gran pèrdua d'informació, optimitzant el balanç entre variabilitat explicada i la suma dels errors quadràtics.

Paraules clau

Autoencoders, Reconstrucció, Machine learning, Capa oculta, Validació creuada, AIC, R^2, SSE, Xarxes neuronals avançades.

ABSTRACT

This final degree project focuses on the use of autoencoders for dimensionality reduction and data reconstruction analysis. Autoencoders are a machine learning technique that allows for the encoding and decoding of data, preserving essential information while reducing its dimensionality. The study investigates the optimal number of nodes in the hidden layer using cross-validation and compares different metrics such as AIC, R^2, and SSE for model selection. Using various databases, the project evaluates the effectiveness of autoencoders in data reconstruction. The model is implemented using the Keras package, highlighting its ability to create and efficiently train advanced neural networks. The results show that it is possible to significantly reduce the dimensionality of the data without a substantial loss of information, optimizing the balance between explained variance and the sum of squared errors.

Keywords

Autoencoders, Reconstruction, Machine learning, Hidden layer, Cross validation, AIC, R^2, SSE, Advanced neural networks.

ÍNDEX

Introducció i motivació del treball

Introducció

Avui dia, el *machine learning* o més concretament els *autoencoders*, són una eina molt utilitzada per professionals del camp de l'anàlisi de dades i tractament d'aquestes. Aquesta eina permet codificar i descodificar les dades per reduir la dimensió d'aquestes sense que es perdi informació i també ajuda en la detecció d'anomalies en les dades. Aquest algorisme ha estat utilitzat en camps diversos com podrien ser el biomèdic, la generació de dades artificials, la bioinformàtica, el reconeixement i generació de veus i la generació d'art, entre d'altres.

Motivació del treball

Aquest treball ve motivat pel fet que cada vegada s'utilitzen més algorismes d'aquest tipus per causes diverses com ara són: escollir clients per vendre un tipus de producte concret, veure si les dades biomèdiques tenen algun tipus de patró, etc. Per tant, aquests *autoencoders* semblen una bona solució per tots els problemes anteriors. Ens agradaria aprofundir principalment en la reducció de dimensió, on veurem el potencial dels *autoencoders* en la generació de les dades, després de la seva codificació en una dimensió menor que la d'entrada i comparant diferents tipus de bases de dades amb més variables i més individus.

En el moment de la realització d'aquest estudi, esperem aportar informació al camp del aprenentatge automàtic com una opció viable i molt robusta per reduir dimensions de diverses bases de dades i promoure aquests algorismes pel seu ús en diferents àmbits.

Objectius

Els objectius del treball són els següents, on cadascun d'ells tenen subobjectius que estudiarem a continuació.

- Investigar a partir de la cross-validació la determinació del nombre de nodes de la capa oculta d'un autoencoder.
    - Estudiar o implementar un autoencoder
    - Estudiar o implementar la metodologia de la cross-validació
- Comparar mètriques per la selecció de models basats en autoencoders
    - AIC
    - R2
    - SSE

El *Deep Learning*

Introducció al *Deep Learning*

El *deep learning*, o aprenentatge profund, ha esdevingut una de les àrees més revolucionàries dins de la intel·ligència artificial i l'aprenentatge automàtic. El desenvolupament ha estat possible gràcies a la combinació de grans volums de dades, avenços en el maquinari computacional i l'evolució d'algorismes d'aprenentatge. Inspirat per l'estructura i el funcionament del cervell humà, el deep learning utilitza xarxes neuronals artificials per processar i analitzar dades complexes, permetent a les màquines aprendre de manera autònoma a partir d'exemples.

A diferència dels mètodes tradicionals d'aprenentatge automàtic, que sovint requereixen la creació manual de característiques i regles per interpretar les dades, l'aprenentatge profund pot aprendre representacions jeràrquiques directament a partir de les dades brutes. Això es tradueix en una major precisió i eficiència en tasques com ara el reconeixement d'imatges, el processament del llenguatge natural i la predicció d'esdeveniments. Ha demostrat ser especialment efectiu en problemes on les característiques importants no són evidents immediatament, permetent descobrir patrons ocults a grans conjunts de dades.

El creixement del deep learning ha estat impulsat per la disponibilitat de grans conjunts de dades etiquetades i l'accés a recursos computacionals poderosos, com les unitats de processament gràfic (GPU). Les GPU permeten la paral·lelització d'operacions matemàtiques complexes necessàries per entrenar xarxes neuronals profundes, reduint significativament el temps d'entrenament. A més, la comunitat de recerca i desenvolupament ha contribuït a la creació de biblioteques i marcs de treball com TensorFlow i PyTorch, que simplifiquen la implementació del models.

L'impacte del deep learning es pot observar en una varietat de camps, des de la medicina fins a la indústria automotriu. Per exemple, a la medicina, les xarxes neuronals profundes s'utilitzen per analitzar imatges mèdiques i ajudar en el diagnòstic de malalties. A la indústria automotriu, és fonamental per al desenvolupament de vehicles autònoms, ja que permet als cotxes interpretar dades de sensors i prendre decisions en temps real. Aquestes aplicacions demostren el potencial transformador de l'aprenentatge profund a diverses indústries.

El deep learning és una tecnologia clau que ha canviat la manera com abordem problemes complexos de dades. La seva capacitat per aprendre automàticament representacions jeràrquiques a partir de dades brutes ha portat a avenços significatius en múltiples disciplines. A mesura que la tecnologia continua evolucionant, és probable que vegem un impacte encara més gran d'aquest a la nostra vida quotidiana i en el progrés científic i tecnològic.

Fonaments del Deep Learning

Els fonaments del deep learning es basen en l'ús de xarxes neuronals artificials inspirades en l'estructura del cervell humà. Una xarxa neuronal està formada per múltiples capes de neurones artificials, que són unitats de càlcul que processen la informació. Aquestes capes es divideixen en tres tipus principals: la capa d'entrada, les capes ocultes i la capa de sortida. La capa d'entrada rep les dades brutes, les capes ocultes processen i transformen aquestes dades, i la capa de sortida genera la resposta final del model.

El procés d'aprenentatge en una xarxa neuronal profunda implica ajustar els pesos de les connexions entre neurones per minimitzar un error o funció de pèrdua. Això s'aconsegueix mitjançant un algorisme anomenat retropropagació, que calcula el gradient de la funció de pèrdua respecte als pesos i actualitza els pesos a la direcció oposada al gradient per reduir l'error. Aquest procés iteratiu continua fins que el model arriba a una precisió satisfactòria en la tasca per a la qual ha estat entrenat.

Una característica clau del deep learning és la seva capacitat per fer aprenentatge no supervisat i supervisat. A l'aprenentatge supervisat, el model s'entrena amb un conjunt de dades etiquetades, on cada entrada està associada amb una sortida desitjada. En l'aprenentatge no supervisat, el model intenta identificar patrons i estructures a les dades sense la guia d'etiquetes explícites. També hi ha l'aprenentatge per reforç, on el model aprèn a prendre decisions seqüencials per maximitzar una recompensa acumulada al llarg del temps.

La profunditat d'una xarxa neuronal, és a dir, el nombre de capes ocultes, és el que diferencia l'aprenentatge profund d'altres mètodes d'aprenentatge automàtic. Les xarxes profundes poden aprendre representacions més complexes i abstractes de les dades, cosa que els permet dur a terme tasques que són difícils o impossibles per als models més superficials. Tot i això, entrenar xarxes profundes requereix una quantitat significativa de dades i poder computacional, així com tècniques avançades per evitar problemes com el sobreajustament i la desaparició del gradient.

Podríem dir que els fonaments del deep learning rauen en la seva estructura de xarxes neuronals profundes i la seva capacitat per aprendre de grans volums de dades a través de la retropropagació i l'ajustament iteratiu de pesos. Aquesta tecnologia permet la creació de models altament precisos i versàtils, capaços d'abordar una àmplia gamma de problemes complexos a diverses àrees, des de la visió per ordinador fins al processament del llenguatge natural.

Arquitectures i Models a Deep Learning

Les arquitectures i models de deep learning són variats i estan dissenyats per abordar diferents tipus de problemes i dades. Una de les arquitectures més conegudes és la xarxa neuronal convolucional (CNN), utilitzada principalment al camp de la visió per ordinador. Les CNN són especialment efectives per al reconeixement d'imatges i la detecció d'objectes perquè poden capturar característiques espacials mitjançant l'ús de filtres de convolució. Aquests filtres permeten a la xarxa detectar vores, textures i altres característiques importants a les imatges.

Una altra arquitectura important és la xarxa neuronal recurrent (RNN), que es fa servir per al processament de seqüències i sèries temporals. Les RNN tenen connexions cícliques que els permeten mantenir informació sobre estats anteriors, cosa que és crucial per a tasques com el reconeixement de veu i la traducció automàtica. No obstant això, les RNN tradicionals tenen limitacions, com el problema del gradient esvaint. Per mitigar aquests problemes, s'han desenvolupat variants com les LSTM (Long Short-Term Memory) i GRU (Gated Recurrent Unit), que milloren la capacitat de la xarxa per capturar dependències a llarg termini.

Les xarxes generatives adversàries (GAN) són una altra arquitectura notable al deep learning. Introduïdes per Ian Goodfellow el 2014, les GAN consisteixen en dues xarxes neuronals que competeixen entre si: una xarxa generadora i una xarxa discriminadora. La xarxa generadora crea dades falses que imiten les dades reals, mentre que la xarxa discriminadora intenta distingir entre dades reals i falses. Aquest enfocament ha demostrat ser extremadament efectiu per generar imatges, vídeos i altres tipus de dades sintètiques d'alta qualitat.

Les xarxes neuronals d'autoencoders també tenen un paper important en el deep learning, especialment en l'aprenentatge no supervisat. Un autoencoder és una xarxa neuronal que intenta aprendre una representació comprimida de les dades d’entrada i després reconstruir les dades originals a partir d’aquesta representació. Aquests es fan servir per a la reducció de dimensió, la detecció d'anomalies i la generació de dades. Algunes de les seves variants, com els autoencoders variacionals (VAE), introdueixen probabilitats en el procés de codificació, millorant la capacitat de la xarxa per generar dades noves i variades.

Aplicacions del Deep Learning

Les aplicacions del deep learning són moltes i abasten una àmplia gamma de sectors, transformant indústries senceres amb les seves capacitats avançades. Al camp de la visió per ordinador, el deep learning ha revolucionat tasques com el reconeixement d'imatges, la detecció d'objectes i la segmentació d'imatges. Models basats en CNN s'utilitzen per identificar i classificar objectes en imatges i vídeos amb una precisió que

supera amb escreix els mètodes tradicionals. Això ha permès avenços significatius en àrees com la vigilància, l'automoció i la realitat augmentada.

En el processament del llenguatge natural (NLP), l'aprenentatge profund ha millorat dràsticament la comprensió i la generació de llenguatge humà. Models com els transformadors, incloent-hi BERT i GPT, han demostrat una capacitat sense precedents per executar tasques de traducció automàtica, anàlisi de sentiments, resposta a preguntes i generació de text coherent. Aquestes tecnologies estan integrades en assistents virtuals, sistemes d'atenció al client i eines de redacció, facilitant interaccions més naturals i efectives entre humans i màquines.

El deep learning també està tenint un impacte significatiu a la medicina. Les xarxes neuronals s'utilitzen per analitzar imatges mèdiques, com ara radiografies i ressonàncies magnètiques, per detectar malalties i condicions amb una precisió comparable a la dels metges especialistes. A més, el deep learning ajuda al descobriment de fàrmacs en analitzar grans volums de dades biològiques i químiques per identificar possibles candidats a medicaments. Aquestes aplicacions estan accelerant el diagnòstic i el tractament de malalties, millorant els resultats per als pacients.

A la indústria automotriu, és un component crucial per al desenvolupament de vehicles autònoms. Les xarxes neuronals permeten als cotxes interpretar dades de sensors, com ara càmeres i lidar, per entendre el seu entorn i prendre decisions en temps real. Això inclou la detecció de vianants, la identificació de senyals de trànsit i la planificació de rutes. Les empreses de tecnologia i automòbils estan invertint fortament en deep learning per avançar en la conducció autònoma i millorar la seguretat viària.

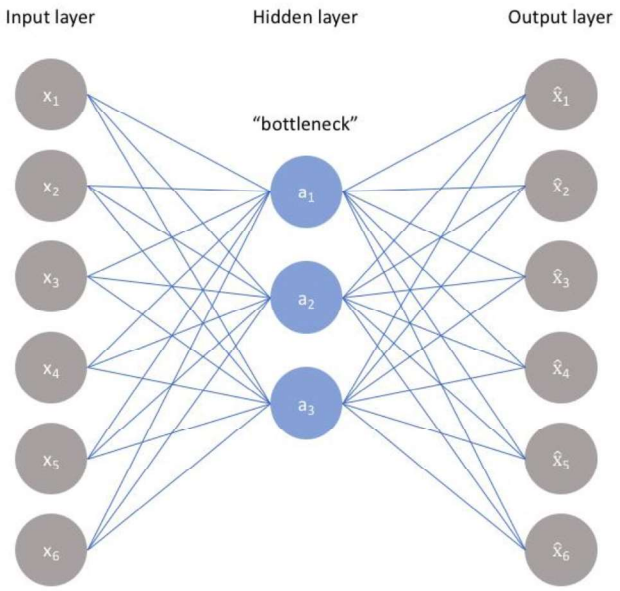
El deep learning també s'utilitza al sector financer per a tasques com la detecció de fraus, la gestió de riscos i la personalització de serveis financers. Els models poden analitzar patrons en grans conjunts de dades transaccionals per identificar comportaments anòmals i predir tendències del mercat. Això no només millora la seguretat i l'eficiència de les operacions financeres, sinó que també permet oferir productes i serveis més adaptats a les necessitats individuals dels clients.

Algunes de les aplicacions del deep learning estan transformant una àmplia varietat d'indústries, des de la visió per ordinador i el processament del llenguatge natural fins a la medicina, l'automoció i les finances. La capacitat del deep learning per aprendre de grans volums de dades i descobrir patrons complexos impulsa innovacions que milloren l'eficiència, la precisió i la capacitat de presa de decisions en múltiples dominis.

Mètodes

En aquest treball, com bé s'ha mencionat anteriorment, utilitzarem *autoencoders* per reduir la dimensió de les dades en diversos escenaris i veure com afecten a les reconstruccions la quantitat d'individus de la mostra o el nombre de variables.

Aquest procés de reducció de dimensions, implica la compressió de dades en una representació de menor dimensió sense perdre informació rellevant. Els *autoencoders* ho aconsegueixen aprenent a codificar les dades d'entrada en una representació compacta al "coll d'ampolla" de la xarxa.



Imatge 1: exemple de codificació de l'autoencoder

En aquest exemple s'observa com la dimensió de les dades d'entrada i de sortida són iguals, ja que volem fer una reconstrucció de les dades per veure que bona és aquesta transformació i en la part Inter mitja veiem la capa oculta o *hidden layer* on es redueix la dimensió d'entrada per ser descodificada un altre cop al final.

Un autoencoder pot tenir múltiples capes ocultes, però nosaltres treballarem en autoencoders d'una única capa oculta on canviarem el nombre de nodes d'aquesta i intentarem reduir el màxim possible la base de dades sense perdre molta informació.

Aquesta capacitat és especialment útil en tasques de preprocessament de dades, on la reducció de dimensió pot facilitar l'anàlisi i la visualització de dades complexes. Per exemple, a l'anàlisi d'imatges, els autoencoders poden reduir la quantitat de

característiques necessàries per representar una imatge, mantenint la major part de la informació visual rellevant.

Els autoencoders també són efectius en l'eliminació de soroll de les dades, una tècnica coneguda com a *denoising*. En aquesta aplicació, s'entrenen per reconstruir dades netes a partir de versions amb soroll. Durant el procés d'entrenament, el model aprèn a distingir entre el soroll i el senyal rellevant, i permet reconstruir una versió més clara i precisa de les dades originals. L'ús d'aquest, és particularment valuós en el processament d'imatges i senyals, on l'eliminació de soroll pot millorar la qualitat de les dades i facilitar tasques posteriors com l'anàlisi d'aquestes.

Intentarem reconstruir diferents bases de dades a partir d'un *autoencoder* de vàries capes internes per veure la capacitat de reconstrucció per diferents dimensions. Una vegada tornem a tenir la base de dades descompilades quina és la que ens aporta una millor reconstrucció amb una reducció de dimensions més gran fixant-nos sobretot en les correlacions i altres estadístics.

Pel que fa a les bases de dades utilitzades, totes són bases de dades sobre el Càncer de mama cadascuna amb diferents proporcions de variables per veure com afecta el fet d'augmentar el nombre de variables, al nombre de nodes òptim de la capa oculta del *autoencoder*.

Per tant, s'ajusten els paràmetres de l'*autoencoder* de tal manera que sigui capaç de construir el model per diferents reduccions de dimensió que utilitzarem entre el 33% i el 67% de les variables de la base de dades d'entrada. D'aquesta manera, intentarem que la diferència entre els valors d'entrada i la reconstrucció feta pel model sigui mínima.

S'han emprat també estadístics com per exemple el AIC per veure com és de bo el model i mètodes com la separació de les dades en dos conjunts (entrenament i test).

Implementació del autoencoder

Paquet Keras

El paquet utilitzat per tal de construir i analitzar els autoencoders és el paquet *keras*, paquet molt utilitzat per crear models per capes o *layers* i també per entrenar i avaluar prediccions dels autoencoders.

Un altre avantatge important del paquet, és la seva capacitat per aprofitar l'àmplia gamma de tècniques i models preentrenats disponibles a la comunitat d'aprenentatge profund. A través de keras, els usuaris poden accedir a models preentrenats per a tasques comunes com per exemple la classificació d'imatges, el processament del llenguatge natural o inclús el reconeixement de veu. Aquests models poden ser fàcilment personalitzats i ajustats per adaptar-se a les necessitats del projecte o treball, estalviant temps i recursos.

Aquest paquet principalment s'utilitza per crear xarxes neuronals avançades com per exemple: xarxes neuronals recurrents (RNN), xarxes neuronals convolucionals (CNN), xarxes generatives adversàries (GAN), entre altres arquitectures avançades.

En aquest treball principalment s'ha utilitzat aquest paquet en el context de la creació d'un model per capes, capaç de reduir la dimensió de la base de dades principal, i ser capaç de tornar a la mateixa dimensió sense perdre molta informació. Per fer això també s'ha fet ús de paquets com *ggplot* per l'anàlisi gràfic dels residus i de les correlacions i càlcul d'estadístics com bé podria ser el Criteri d'informació d'Akaike (AIC).

Entorn de treball

A causa de problemes amb la instal·lació del paquet Keras o més concretament del paquet TensorFlow que és un paquet intern de keras que permet crear entorns per fer els autoencoders, finalment s'ha fet aquest treball des del servidor UPC, connectant-nos amb un programa VPN anomenat FortiClient VPN.

Per culpa d'això en alguns moments de l'anàlisi s'ha prioritzat la rapidesa del model en entrenar-se que no pas sobrecarregar el RStudio o els autoencoders, ja que sinó el servidor no funcionava correctament.

Base de dades 1

A continuació veurem un petit anàlisi descriptiu de la primera base de dades utilitzada per mostrar el funcionament dels autoencoders.

És una base de dades sobre el càncer de mama, extreta de la pàgina web de Kaggle on podem trobar moltes bases de dades de diferents tipus. VID. Bibliografia [3]

Trobem que tenim 32 variables, les quals seran les dimensions que voldrem reduir. Per part dels individus o files en tenim 569, per tant, tenim una base de dades amb 18.208 entrades. Per l'anàlisi hem tret de la base de dades les dues primeres columnes, ja que la primera era una columna d'identificació de l'individu i la segona era el tipus de diagnosi. Aquestes dues variables no han participat en els autoencoders ni en els models.

Les variables són les següents:

ID: numero identificador del individu, de tipus numèric.

Diagnosis: Diagnosi donada al pacient, de tipus categòric amb "M" per diagnosi Maligne i "B" per diagnosi Benigne.

Radius_mean: mitjana del radi dels lòbuls, de tipus numèric.

Texture_mean: mitjana de la textura superficial, de tipus numèric.

Perimeter_mean: mitjana del perímetre exterior dels lòbuls, de tipus numèric.

Area_mean: mitjana del àrea dels lòbuls, de tipus numèric.

Smoothness_mean: mitjana dels nivells de suavitat, de tipus numèric.

Compactness_mean: mitjana de la compacitat, de tipus numèric.

Concavity_mean: mitjana de la concavitat, de tipus numèric.

Concave points_mean: mitjana dels punts còncaves, de tipus numèric.

Symmetry_mean: mitjana de la simetria, de tipus numèric.

Fractal_dimension_mean: mitjana de la dimensió fractal, de tipus numèric.

Radius_se: error estàndard del radi dels lòbuls, de tipus numèric.

Texture_se: error estàndard de la textura superficial, de tipus numèric.

Perimeter_se: error estàndard del perímetre exterior dels lòbuls, de tipus numèric.

Area_se: error estàndard del àrea dels lòbuls, de tipus numèric.

Smoothness_se: error estàndard dels nivells de suavitat, de tipus numèric.

Compactness_se: error estàndard de la compacitat, de tipus numèric.

Concavity_se: error estàndard de la concavitat, de tipus numèric.

Concave points_se: error estàndard dels punts còncaves, de tipus numèric.

Symmetry_se error estàndard de la simetria, de tipus numèric.

Fractal_dimension_se: error estàndard de la dimensió fractal, de tipus numèric.

Radius_worst: Pitjor radi dels lòbuls, de tipus numèric.

Texture_worst: Pitjor textura superficial, de tipus numèric.

Perimeter_worst: Pitjor perímetre exterior dels lòbuls, de tipus numèric.

Area_worst: Pitjor àrea dels lòbuls, de tipus numèric.

Smoothness_worst: Pitjor nivell de suavitat, de tipus numèric.

Compactness_worst: Pitjor compacitat, de tipus numèric.

Concavity_worst: Pitjor concavitat, de tipus numèric.

Concave points_worst: Pitjor punt còncau, de tipus numèric.

Symmetry_worst: Pitjor simetria, de tipus numèric.

Fractal_dimension_worst: Pitjor dimensió fractal, de tipus numèric.

Una vegada explicades les variables mirarem algunes estadístiques descriptives que ens donin informació de cadascuna d'aquestes variables. Per això utilitzarem la funció *summary* de R i veurem com són aquestes variables i comentarem els més destacats.

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
Min. : 8670	Length:569	Min. : 6.981	Min. : 9.71	Min. : 43.79	Min. : 143.5
1st Qu.: 869218	Class :character	1st Qu.:11.700	1st Qu.:16.17	1st Qu.: 75.17	1st Qu.: 420.3
Median : 906024	Mode :character	Median :13.370	Median :18.84	Median : 86.24	Median : 551.1
Mean : 30371831		Mean :14.127	Mean :19.29	Mean : 91.97	Mean : 654.9
3rd Qu.: 8813129		3rd Qu.:15.780	3rd Qu.:21.80	3rd Qu.:104.10	3rd Qu.: 782.7
Max. :911320502		Max. :28.110	Max. :39.28	Max. :188.50	Max. :2501.0

Imatge 2: Summary de les columnes 1-6

La variable id és numèrica, no són valors des d'1 fins al nombre de files perquè segurament és una mostra del total de pacients.

La variable diagnosi és categòrica i si fem un *table* observem que la distribució no està equilibrada, tenim 357 pacients amb diagnosi benigne i 212 amb diagnosi maligne.

Pel que fa a les altres variables veiem que els valors de *radius_mean* estan entre 6.981 i 28.110, la mitjana dels valors està en 14,127 i la mediana o quartil 50 està en 13.370.

En *area_mean* veiem que els valors es troben entre 143.5 i 2501.0 i la mitjana es troba en 654.9. La mediana es troba en canvi en 551.1. Molta més diferencia que les altres variables.

smoothness_mean	compactness_mean	concavity_mean	concave.points_mean	symmetry_mean	fractal_dimension_mean
Min. :0.05263	Min. :0.01938	Min. :0.00000	Min. :0.00000	Min. :0.1060	Min. :0.04996
1st Qu.:0.08637	1st Qu.:0.06492	1st Qu.:0.02956	1st Qu.:0.02031	1st Qu.:0.1619	1st Qu.:0.05770
Median :0.09587	Median :0.09263	Median :0.06154	Median :0.03350	Median :0.1792	Median :0.06154
Mean :0.09636	Mean :0.10434	Mean :0.08880	Mean :0.04892	Mean :0.1812	Mean :0.06280
3rd Qu.:0.10530	3rd Qu.:0.13040	3rd Qu.:0.13070	3rd Qu.:0.07400	3rd Qu.:0.1957	3rd Qu.:0.06612
Max. :0.16340	Max. :0.34540	Max. :0.42680	Max. :0.20120	Max. :0.3040	Max. :0.09744

Imatge 3: Summary de les columnes 7-12

Veiem que aquestes sis variables tenen valors molt petits, inclús *concave.points_mean* i *symmetry_mean* el seu valor mínim és 0 encara que no agafen valors negatius, si que tenen valors nuls.

radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se
Min. :0.1115	Min. :0.3602	Min. : 0.757	Min. : 6.802	Min. :0.001713	Min. :0.002252
1st Qu.:0.2324	1st Qu.:0.8339	1st Qu.: 1.606	1st Qu.: 17.850	1st Qu.:0.005169	1st Qu.:0.013080
Median :0.3242	Median :1.1080	Median : 2.287	Median : 24.530	Median :0.006380	Median :0.020450
Mean :0.4052	Mean :1.2169	Mean : 2.866	Mean : 40.337	Mean :0.007041	Mean :0.025478
3rd Qu.:0.4789	3rd Qu.:1.4740	3rd Qu.: 3.357	3rd Qu.: 45.190	3rd Qu.:0.008146	3rd Qu.:0.032450
Max. :2.8730	Max. :4.8850	Max. :21.980	Max. :542.200	Max. :0.031130	Max. :0.135400

Imatge 4: Summary de les columnes 13-18

concavity_se	concave.points_se	symmetry_se	fractal_dimension_se	radius_worst	texture_worst
Min. :0.00000	Min. :0.000000	Min. :0.007882	Min. :0.0008948	Min. : 7.93	Min. :12.02
1st Qu.:0.01509	1st Qu.:0.007638	1st Qu.:0.015160	1st Qu.:0.0022480	1st Qu.:13.01	1st Qu.:21.08
Median :0.02589	Median :0.010930	Median :0.018730	Median :0.0031870	Median :14.97	Median :25.41
Mean :0.03189	Mean :0.011796	Mean :0.020542	Mean :0.0037949	Mean :16.27	Mean :25.68
3rd Qu.:0.04205	3rd Qu.:0.014710	3rd Qu.:0.023480	3rd Qu.:0.0045580	3rd Qu.:18.79	3rd Qu.:29.72
Max. :0.39600	Max. :0.052790	Max. :0.078950	Max. :0.0298400	Max. :36.04	Max. :49.54

Imatge 5: Summary de les columnes 19-24

En les imatges 4 i 5 trobem totes les variables “se” on veiem que tenen valors petits, a excepció de la variable *area_se* que és més gran les altres tenen valors més propers al 0.

perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave.points_worst
Min. : 50.41	Min. : 185.2	Min. :0.07117	Min. :0.02729	Min. :0.0000	Min. :0.00000
1st Qu.: 84.11	1st Qu.: 515.3	1st Qu.:0.11660	1st Qu.:0.14720	1st Qu.:0.1145	1st Qu.:0.06493
Median : 97.66	Median : 686.5	Median :0.13130	Median :0.21190	Median :0.2267	Median :0.09993
Mean :107.26	Mean : 880.6	Mean :0.13237	Mean :0.25427	Mean :0.2722	Mean :0.11461
3rd Qu.:125.40	3rd Qu.:1084.0	3rd Qu.:0.14600	3rd Qu.:0.33910	3rd Qu.:0.3829	3rd Qu.:0.16140
Max. :251.20	Max. :4254.0	Max. :0.22260	Max. :1.05800	Max. :1.2520	Max. :0.29100

Imatge 6: Summary de les columnes 25-30

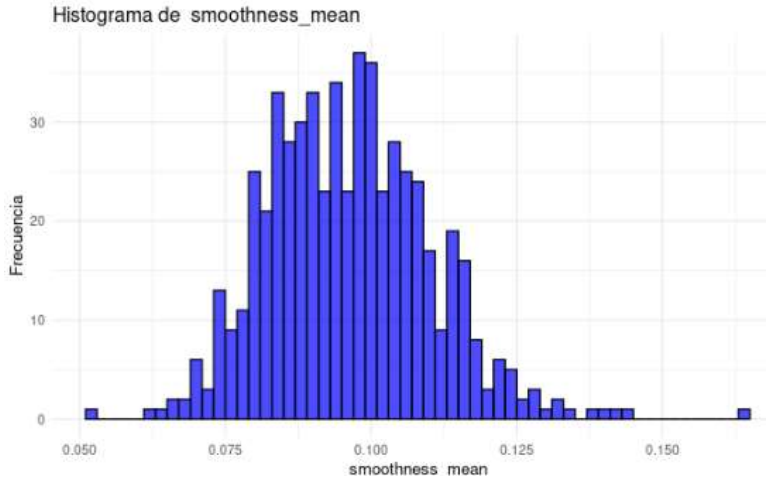
```
symmetry_worst   fractal_dimension_worst
Min.   :0.1565   Min.   :0.05504
1st Qu.:0.2504   1st Qu.:0.07146
Median :0.2822   Median :0.08004
Mean   :0.2901   Mean   :0.08395
3rd Qu.:0.3179   3rd Qu.:0.09208
Max.   :0.6638   Max.   :0.20750
```

Imatge 7: Summary de les columnes 31 i 32

Si observem les variables de tipus “worst” veiem que els valors tendeixen a ser més grans que els anteriors, encara que hi ha variables on encara tenim valors petits, però la majoria ja en tenen algun més gran.

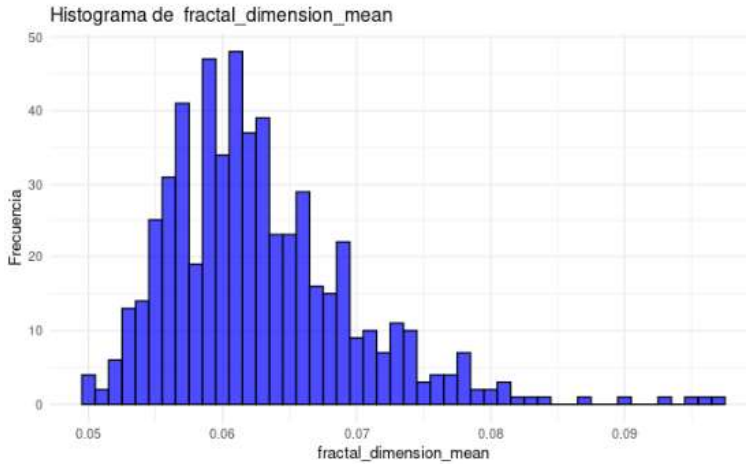
Veurem ara també els histogrames de les variables per veure com estan distribuïdes, però ja no tindrem en compte les dues primeres, ni la variable id, ni la variable diagnosi. Utilitzarem el paquet de R *ggplot* que ens permet fer aquests histogrames per a cada variable.

Comentarem els histogrames de les 5 variables més destacables, les altres 25 es deixaran en annexos VID. Imatge 1.A-3.A



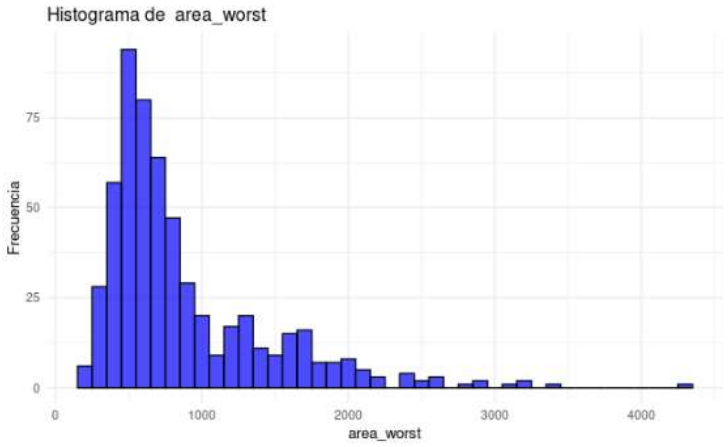
Gràfic 1: Histograma de la variable smoothness_mean

Veiem que la variable smothness_mean té una distribució bastant semblant a la normal i veiem que els seus valors són bastant petits en general.



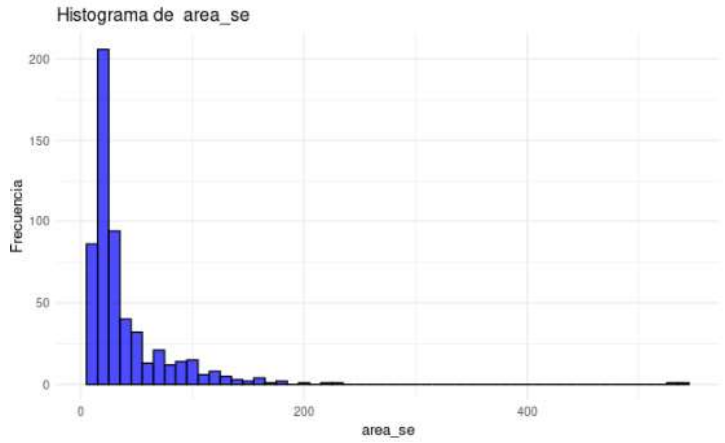
Gràfic 2: Histograma de la variable fractal_dimension_mean

Pel que fa a la variable fractal_dimension_mean veiem que els valors són encara més petits i la distribució seria semblant a la gamma



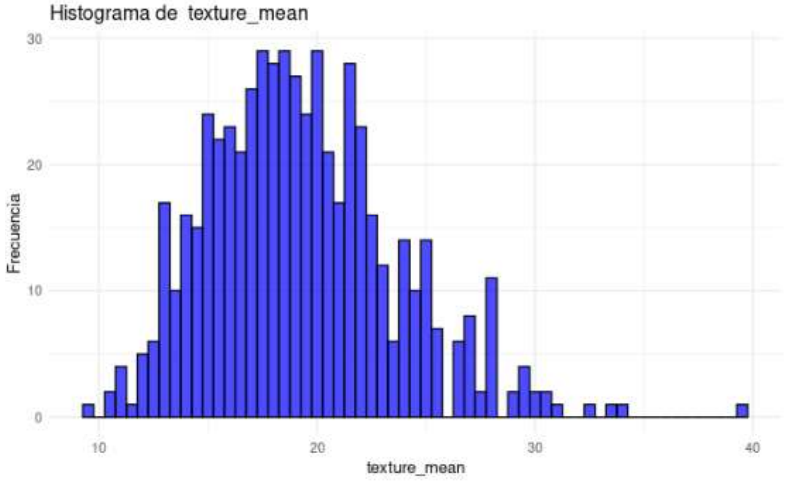
Gràfic 3: Histograma de la variable area_worst

En la variable area_worst trobem que els valors que agafa la variable són molt grans, arribant alguns fins a 4300 aproximadament. La distribució seria semblant a la gamma un altre cop.



Gràfic 4: Histograma de la variable area_se

Veiem ara en la variable area_se que aquesta sembla una distribució exponencial però amb algun valor *outlier*. Els valors de la variable són més grans que els anteriors però el valor extrem sembla que els fa més grans.



Gràfic 5: Histograma de la variable texture_mean

Per últim trobem una altre distribució normal encara que desplaçada a la dreta. Veiem que aquesta variable té valors més grans que smothness_mean, però una distribució semblant.

Base de dades 2

Pel que fa a la segona base de dades, on provarem també la reconstrucció de l'autoencoder després de reduir la dimensió d'entrada, tenim un total de 143 variables amb 410 individus de les quals utilitzarem 142, ja que la primera columna és una variable identificadora de cada individu.

En la base de dades utilitzada, anomenada *protein_abundance.csv* les 142 variables restants, utilitzades per l'autoencoder, són numèriques. Aquesta base de dades està extreta de l'enllaç de la Bibliografia [4].

Base de dades 3

Aquesta última base de dades és la base amb més variables de les tres que tenim, el fitxer es diu *gene_expression.csv*. Aquesta base de dades també està extreta de la pàgina de la Bibliografia [4].

Té un nombre de files de 526 i un total de 17815 variables. Aquest nombre és massa gran per a calcular amb un autoencoder quina seria la dimensió òptima, ja que per fer validació creuada trigaria molt a executar les diferents dimensions de la capa oculta, per tant, el que farem és quedar-nos només amb una part d'aquesta base de dades reduint-la a 3466 variables, un 20% de la base de dades completa. Comprovarem també quina és la millor dimensió pel mètode del colze i com són algunes de les reconstruccions, ja que tenim moltes variables.

Per reduir aquesta base de dades el que hem fet, veient que tenia valors faltants, ha sigut primer treure aquelles columnes que tinguessin un *NA* per no tindre problemes amb l'autoencoder i les reconstruccions.

Quan ja hem tret les columnes on hi hagués un NA, la base de dades es queda amb 17328 variables. Per reduir la base de dades a 3466 variables, el que hem fet és agafar aquelles que tenien una variància més alta, per tal d'analitzar aquelles amb més variància. Una vegada calculada la variància de cada columna, les hem ordenat i ens hem quedat amb el 20% amb aquelles més altes.

Una vegada fet això, ja tenim una base de dades de 3466 variables amb 526 individus que utilitzarem per comprovar quina és la millor dimensió de la capa oculta i com surten les reconstruccions amb aquesta dimensió.

Normalització

Abans de començar amb l'anàlisi de la reconstrucció dels autoencoders, parlarem del mètode utilitzar per normalitzar les dades anomenat normalització min-max.

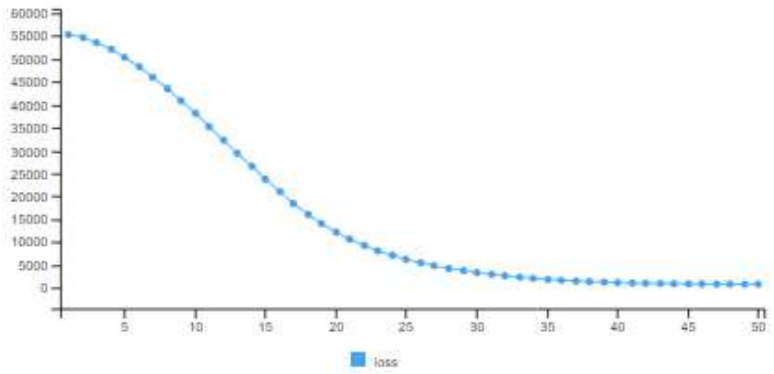
Aquest mètode s'utilitza molt en el context de l'aprenentatge d'autoencoders i consisteix a restar a cadascun dels valors d'una columna (variable), el valor mínim, i dividir-ho entre la diferència del valor màxim i el valor mínim de la variable (rang).

$$N_j = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

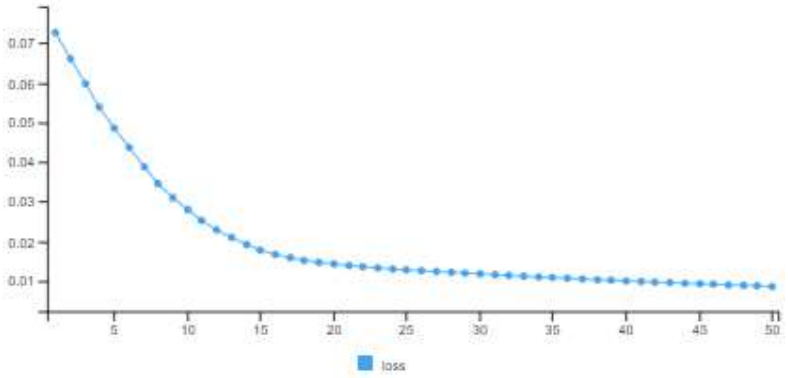
D'aquesta forma normalitzem les dades perquè l'aprenentatge de l'autoencoder sigui millor i minimitzar els errors, ja que així el gradient es propaga millor.

En els gràfics 6 i 7, podem observar el fet de la normalització de les mateixes dades en un autoencoder com fa disminuir l'error. En el primer gràfic on les dades no estan normalitzades veiem que a la primera iteració l'error és de l'ordre de 55000 i baixa fins a arribar a 800 aproximadament, mentre que els errors del segon gràfic amb les mateixes dades, però ara normalitzades comença per sobre de 0.07 i acaba a menor de 0.01.

No només aquest fet sinó que veiem com la corba d'error baixa més ràpidament amb dades normalitzades que amb dades sense normalitzar, veiem que l'error a les dades sense normalitzar sobre les èpoques número 25-30 s'estabilitza l'error, mentre que a les dades normalitzades això passa sobre l'època 15, fent que el model no necessiti tantes repeticions per arribar a minimitzar l'error. Això és gràcies a la millor eficiència del gradient, ja que tots ells són més uniformes i milloren l'eficiència de l'entrenament.



Gràfic 6: Pèrdua del model sense normalitzar



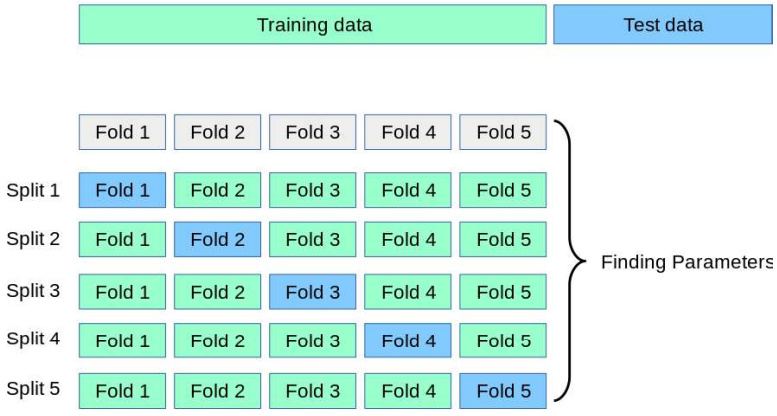
Gràfic 7: Pèrdua del model normalitzat

Validació creuada

Per part de la validació dels models que crearem amb diferents dimensions a la capa oculta, farem *cross-validation* o validació creuada.

La validació creuada consisteix a fer un número “k” de particions de les files de la base de dades, com si fos la separació entrenament i validació però amb més particions. El nombre de particions depèn de la grandària de la base de dades, però el més important és que les particions estiguin equilibrades i no siguin molt petites, sinó pot ser que alguna de les validacions tingui *overfitting* o sobre ajust en aquella partició.

En les nostres dades aquest nombre de particions ha estat 5 per a totes les bases de dades, ja que totes elles tenien un nombre semblant de files.



Imatge 8: Esquema d’exemple de validació creuada

La imatge 8 seria l’exemple de les nostres dades, primerament fem 5 particions equilibrades de les dades de forma aleatòria. Seguidament, la primera partició la reservem per la validació i les altres 4 particions les ajuntem per entrenar el model.

Una vegada realitzat veiem que tan bona ha estat la reconstrucció validant amb la primera partició. Ara tornem a fer el mateix, però la segona partició serà la que reservem per fer la validació i les altres 4 per entrenar el model, així fins a acabar amb totes.

Una vegada realitzades totes les validacions fem la mitja d'aquesta per assegurar-nos que no hi ha alguna part de les dades que té algun valor extrem ni tenim sobre ajust.

Determinació de la dimensió de la capa oculta

Els estadístics utilitzats per decidir quin és el millor model seran l'R^2, la suma dels errors quadràtics (SSE) i l'AIC. L'R^2 és un estadístic utilitzat en models per veure quin és el percentatge de la variabilitat de les dades originals que és explicada pel model. Aquest pren valors entre 0 i 1 sent 1 un model que explica tota la variabilitat de les dades i 0 un model que no explica res la variabilitat de les dades.

L'Akaike Information Criterion o AIC, és una mesura utilitzada en estadística per trobar el model que millor explica les dades, equiparant la complexitat del model amb el nombre de paràmetres i la capacitat d'ajustar-se a les dades. La seva fórmula és:

$$AIC = -2\log(L) + 2k$$

On *L* és la funció de versemblança del model i *k* és el nombre de paràmetres del model. S'ha utilitzat com a funció de versemblança el mean squared error o MSE.

En el nostre cas, l'R^2 més gran ha estat aquell que ha tingut una dimensió de la capa oculta més gran, per tant, utilitzarem el mètode del colze per decidir quina serà la millor dimensió de la capa oculta a partir dels R^2. Aquest mètode s'utilitza molt en el context de l'agrupació de dades, o més conegut com *clustering,* que es basa en calcular la suma dels errors quadràtics per cadascuna de les dimensions utilitzades en la capa oculta i quedar-se amb la que ha fet disminuir l'error més ràpidament al principi, i que després ja no baixa amb el mateix pendent, aquest punt s'anomena "colze".

Farem 5 particions de la base de dades per fer la validació creuada, i calcularem per a diferents nombres de nodes de la capa oculta l'R^2, el SSE i l'AIC que seran la mitjana de les 5 validacions fetes en cada part de la validació creuada d'un mateix nombre de nodes de la capa oculta.

Autoencoder utilitzat

Per arribar al nostre objectiu de comprovar la reconstrucció d'un autoencoder primer hem de crear-lo. El autoencoder utilitzat està format per dues parts, que després s'ajunten per crear l'autoencoder, la part codificadora, i la part descodificadora.

Hem utilitzat un autoencoder de només una capa oculta que serà la dimensió a la qual reduirem la base de dades per posteriorment veure com ha estat la reconstrucció.

En la part codificadora fixem la dimensió d'entrada de la base de dades en un valor p, equivalent al nombre de variables (en la primera la dimensió és 30, en el cas de la segona 142 i en la tercera 3466). La dimensió de la capa oculta, que anirà canviant per comprovar quina serà la millor reducció de dimensió i una activació on hem utilitzat l'activació "relu" o ReLU que significa *Rectified Linear Unit*. Segons tots els mètodes disponibles d'activació hem considerat que la millor opció seria aquesta, ja que les seves principals funcions són la reducció del temps de computació i la mitigació de l'esvaïment del gradient per tal de millorar els entrenaments, que siguin més estables i més ràpids de reproduir.

En la part descodificadora invertim les dimensions de forma que la dimensió d'entrada serà la dimensió a la qual hem reduït les dades en la part codificadora i la dimensió de sortida serà la dimensió inicial de la base de dades. També utilitzem una activació en aquesta part, però en aquest cas hem utilitzat l'activació "linear", ja que aquesta activació no restringeix les dades codificades al ser descodificades com altres activacions, de manera que la reconstrucció pot abastar tots els valors possibles de les dades originals.

Finalment, ajuntem aquestes dues parts en una sola en forma d'*autoencoder* per tal que sigui més fàcil veure com surten les prediccions, ja que només hem de cridar un únic model que ajunta les dues parts, i a més podem entrenar les dues parts a la vegada i no per separat.

Una vegada aquest autoencoder està creat, el compilem i l'entrenem amb les nostres dades. La part de la compilació també està formada per dues parts importants, el mètode de la pèrdua del model o el que volem reduir per considerar que un model està entrenant correctament, i l'optimitzador utilitzat.

En el nostre cas per tots els *autoencoders* hem utilitzat la pèrdua del *mean squared error* (mse) o error quadràtic mig per les seves propietats a l'hora d'optimitzar el model i la facilitat de la interpretació d'aquest.

Per altra part, hem utilitzat l'optimitzador *Adaptive Moment Estimation* o "Adam", ja que és una combinació dels dos optimitzadors més utilitzats per autoencoders, com

són el *RMSProp* i el *Momentum*. Les seves millors característiques serien la seva adaptació a la taxa d'aprenentatge, pel fet que utilitza una part dels gradients utilitzats en l'optimització anterior per millorar la següent i l'eficiència computacional. Per l'última base de dades amb un nombre molt gran de variables, aquest optimitzador ens ajuda a minimitzar els temps d'entrenament.

Resultats

Elecció dimensió base de dades 1

Una vegada fet l'anàlisi descriptiu de les dades, conformat l'autoencoder i normalitzat aquestes, hem fet ús de la validació creuada mencionada anteriorment per decidir quina serà la dimensió a la qual reduirem les dades.

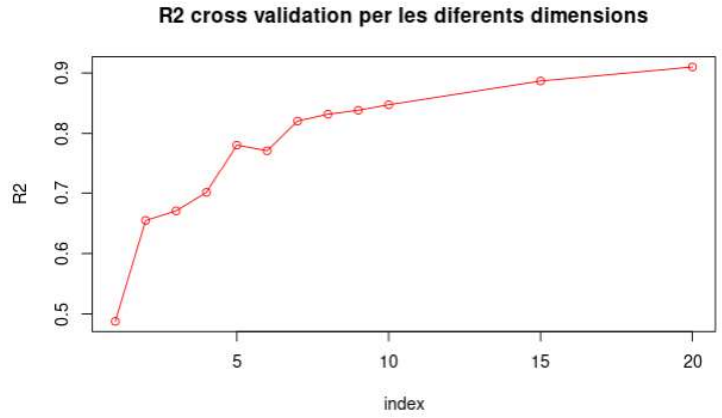
Seguidament, fem la validació creuada i veiem a la taula 1 tres columnes. *Index*, que representa el nombre de nodes de la capa oculta pels quals hem realitzat validació creuada en cadascun d'ells. *R2* que seria l'R^2 mitjà de les validacions fetes en aquell índex en les validacions creuades i finalment la variable *e* que seria la mitjana dels SSE de cada validació per cada nombre de nodes de la capa oculta.

index	R2	e
1	0.4877471	51.95003
2	0.6547544	35.40039
3	0.6705658	33.89038
4	0.7011114	30.97677
5	0.7806243	22.45950
6	0.7713666	23.65932
7	0.8205887	18.50761
8	0.8318439	17.17423
9	0.8384191	16.61104
10	0.8474600	15.77509
15	0.8867876	11.72842
20	0.9100079	9.28939

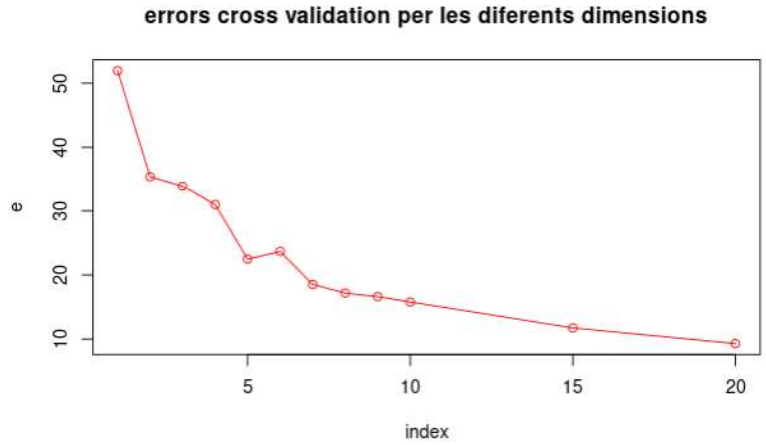
Taula 1: informació de la validació creuada de la base de dades 1

Observem, per tant, que el R^2 més gran és on el nombre de nodes de la capa oculta és més gran. Passa el mateix si mirem on està la suma dels errors quadràtics, ja que aquesta suma és mínima al nombre més gran de nodes. Com volem quedar-nos amb el node que augmenti la variabilitat explicada i minimitzi el SSE, però també volem

disminuir la dimensió original de les dades, ens quedarem amb el punt mencionat anteriorment que fa de colze, com observem als gràfics 8 i 9.



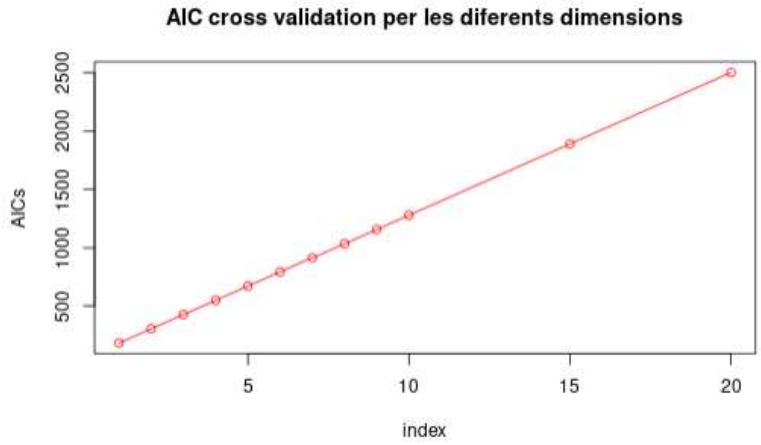
Gràfic 8: R^2 de les diferents dimensions de la base 1



Gràfic 9: SSE de les diferents dimensions de la base 1

En aquests dos gràfics observem que el punt del colze que disminueix el SSE i que augmenta l'R^2 seria amb un nombre de nodes a la capa oculta igual a 5. Aquest seria el punt "colze" que ens permet disminuir la dimensió original de les dades sense tindre una pèrdua molt gran de les dades originals, veient que explica gairebé el 80% de la variabilitat de les dades originals.

Com el nostre objectiu és reduir la dimensió de les dades originals i no arribar a fer una reconstrucció perfecta de les dades, aquest valor de l'R^2 ja seria prou bo per a quedar-nos amb ell, ja que una variabilitat explicada del 80% ens indica que s'ha aconseguit preservar la major part de l'estructura i les característiques de les dades originals de manera raonable.



Gràfic 10: AIC de les diferents dimensions de la base 1

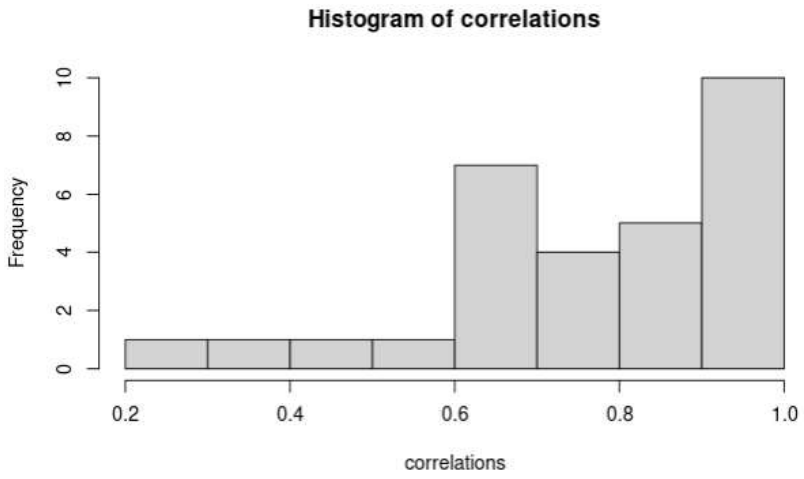
Es pot observar que l'AIC no ens ajuda per trobar quina és la millor dimensió, ja que ens diu que el millor seria reduir a 1 dimensió, el fet d'augmentar el nombre de nodes de la capa oculta no fa reduir suficient els MSE.

Anàlisi reconstrucció base de dades 1

Com ja hem vist quin serà el nostre nombre de nodes a la capa oculta per aquesta base de dades, ara veurem amb aquest nombre de nodes que bona és la reconstrucció amb diferents eines gràfiques i estadístiques.

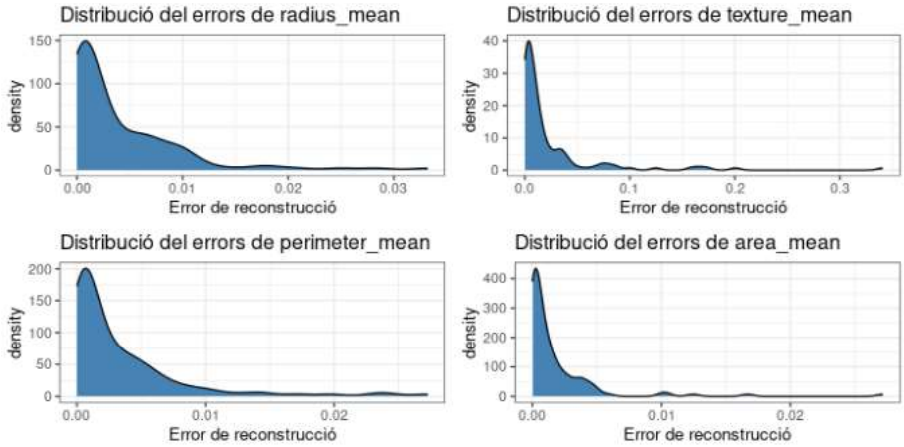
Tornarem a fer ús del autoencoder per ara no farem validació creuada, sinó que agafarem la base de dades i la separarem en entrenament i validació directament, un 80% de la base de dades anirà a entrenament i un 20% a validació. Així és com avaluarem la reconstrucció de les diferents variables per veure si ha estat prou bé.

Una vegada realitzat, veurem com a estat a reconstrucció general de la base de dades, i algunes de les variables més concretament, comentant alguns dels resultats més destacables. Al gràfic 11 veiem la correlació entre variables de la base de dades original i de la reconstrucció, veiem que algunes de les variables tenen una correlació bastant baixa, però la majoria tenen una correlació per sobre del 0.6, només 4 estan per sota d'aquest valor. També veiem que un terç de les variables de la base de dades tenen una correlació entre 0.9 i 1.



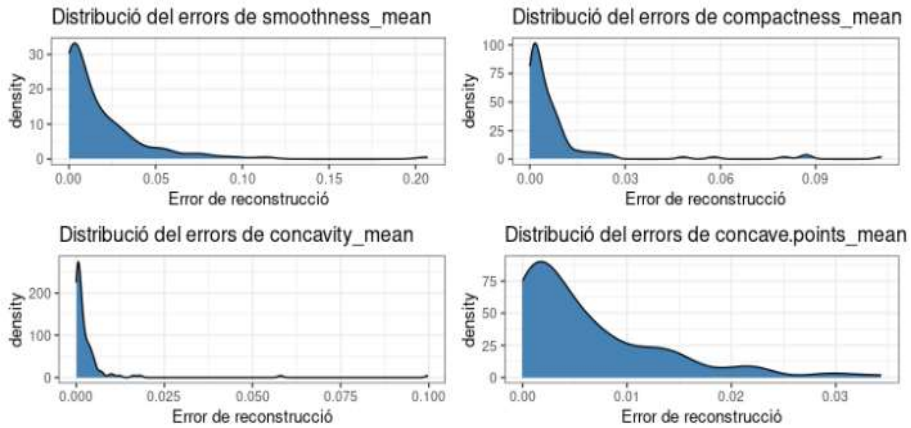
Gràfic 11: Histograma de les correlacions de la base 1

Comentarem ara la distribució dels errors per a cadascuna de les variables en grups de 4 variables i comentarem els gràfics més destacables.



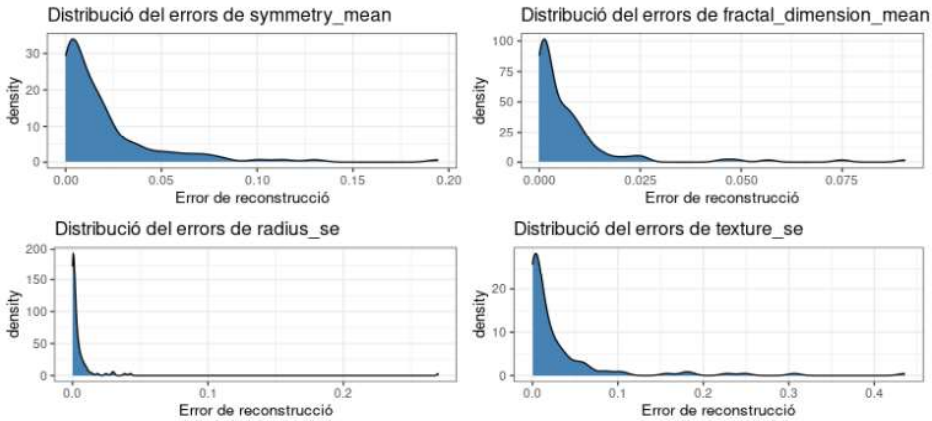
Imatge 9: Gràfics de densitat dels errors de les variables 1-4 de la base 1

En la imatge 9 veiem que les quatre variables tenen una distribució bastant similar, però la variables *texture_mean* sembla tenir algun error més gran que les altres tres.



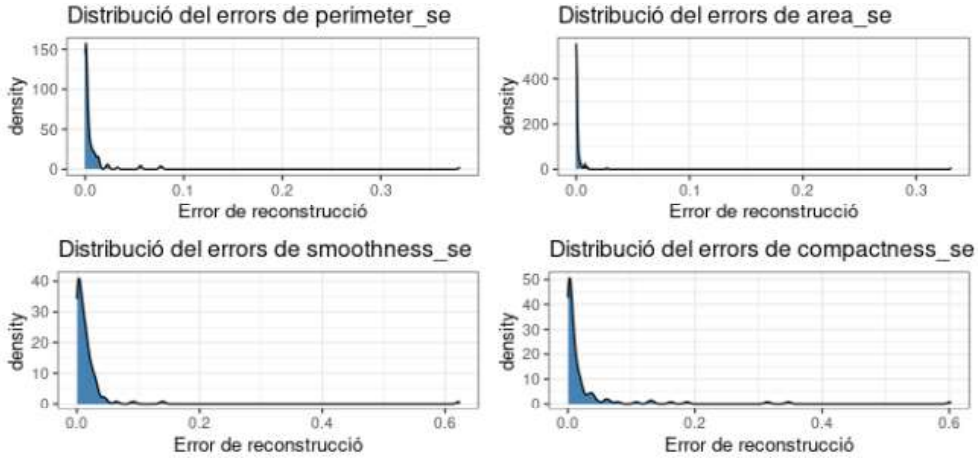
Imatge 10: Gràfics de densitat dels errors de les variables 5-8 de la base 1

En la imatge 10 veiem que la variable *concavity_mean* té uns errors molt petits, en canvi la variable *concave.points_mean* té una variància més alta que les altres. També té errors petits en comparació a la variable d'abans *texture_mean*.



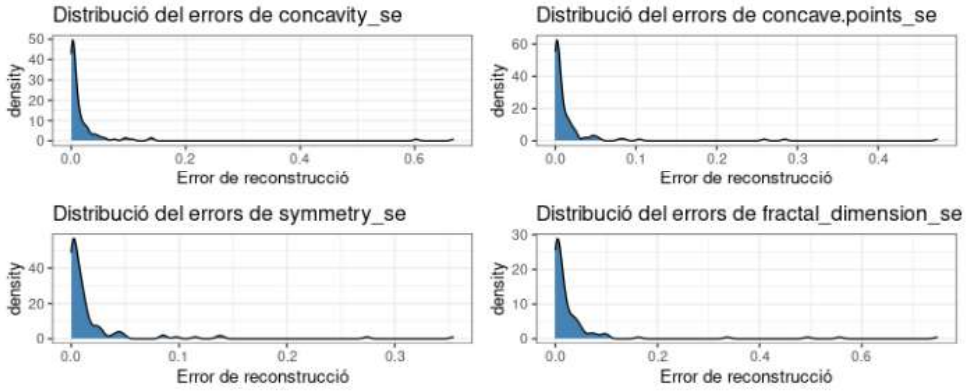
Imatge 11: Gràfics de densitat dels errors de les variables 9-12 de la base 1

En aquestes distribucions s'observa com els errors de la variable *radius_se* són molt petits, gairebé 0, encara que hi ha algun valor extrem. De les altres variables destacar també la poca variabilitat dels errors de la variable *fractal_dimension_mean*.



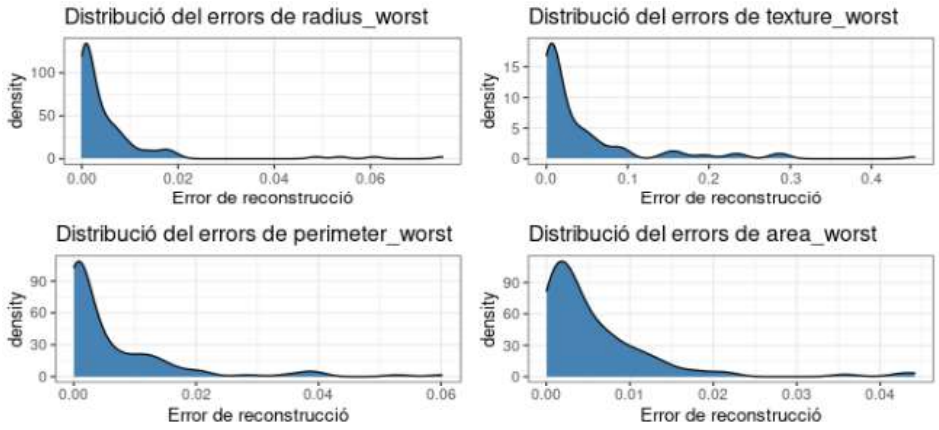
Imatge 12: Gràfics de densitat dels errors de les variables 13-16 de la base 1

En aquestes quatre variables de la imatge 12 veiem que els errors són molt petits, amb algun *outlier*, però en general petits tant els errors com la variància dels mateixos.



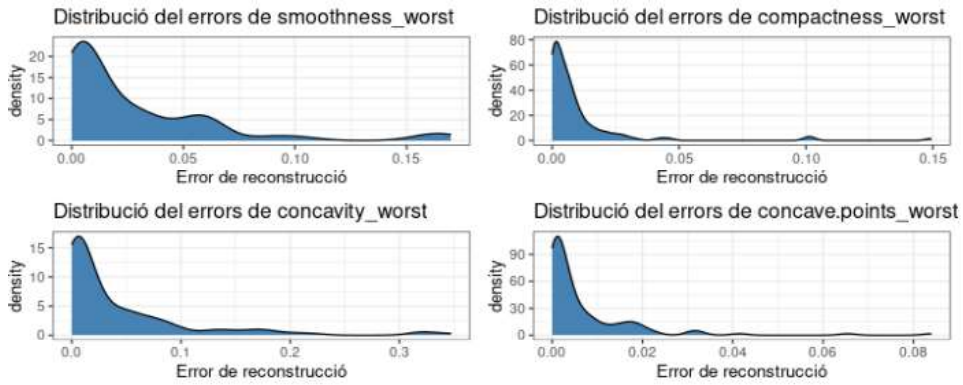
Imatge 13: Gràfics de densitat dels errors de les variables 17-20 de la base 1

Veiem ara en la imatge 13 que les quatre variables són bastant similar a les anteriors, errors petits amb algun valor extrem.



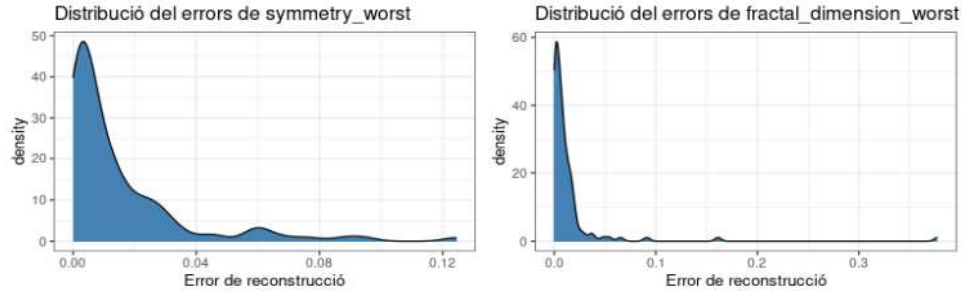
Imatge 14: Gràfics de densitat dels errors de les variables 21-24 de la base 1

En aquestes variables veiem que ara augmenta més la variabilitat que en les anteriors, encara que els errors son petits, la variable *texture_worst* sembla tenir una mica més d'errors que les altres tres.



Imatge 15: Gràfics de densitat dels errors de les variables 25-28 de la base 1

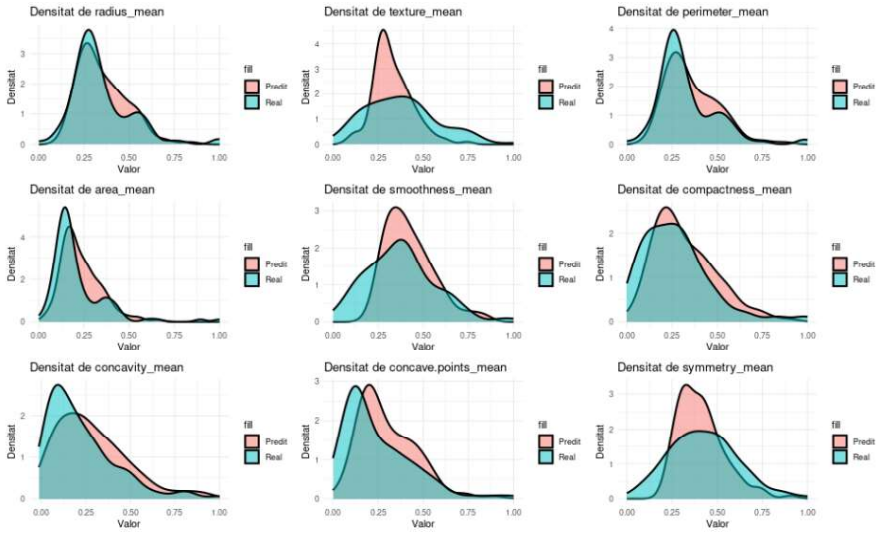
Observem que tenim també alguna variable amb més errors que abans amb les variables “se”, encara que no tenen tants ouliers.



Imatge 16: Gràfics de densitat dels errors de les variables 29 i 30 de la base 1

En les últimes dues variables trobem que els errors tenen algun valor extrem però no molt gran i la variabilitat augmenta una mica més.

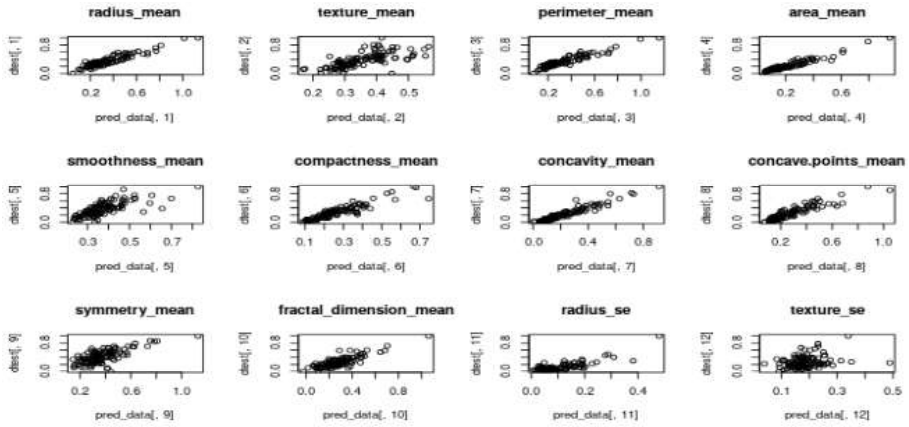
En conclusió, podríem dir que hi ha una diferenciació clara entre les variables del tipus “mean”, “se” i “worst” pel que fa als errors, sembla que les variables “se” estan més a prop del 0, però tenen més valors extrems, mentre que les variables “worst” tenen més errors però no tants *outliers.* Pel que fa a les variables tipus “mean” sembla que són un punt intermedi de les altres dos, tenen una variància més alta que les variables “se” però tenen més outliers que les variables “worst”. Encara que en el global sembla que tots els errors estan molt a prop del 0 en general.



Imatge 17: Gràfics de densitat reals vs predits de les variables 1-9 de la base 1

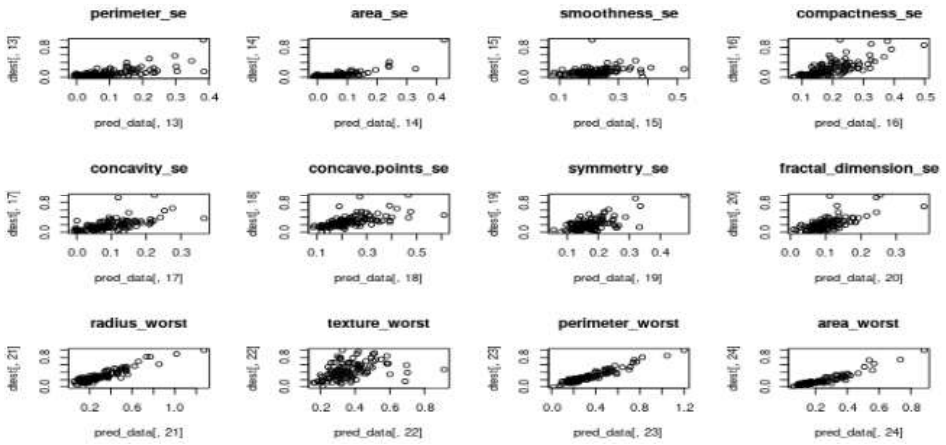
Veiem que a excepció d'alguna variable, la majoria de variables tenen una distribució semblant a la de les dades originals, els altres gràfics es deixaran a Annexos. VID. Imatges 4.A-6.A

Per últim, veurem els gràfics de dispersió entre la base de dades real i les dades predites per veure com han estat les prediccions.



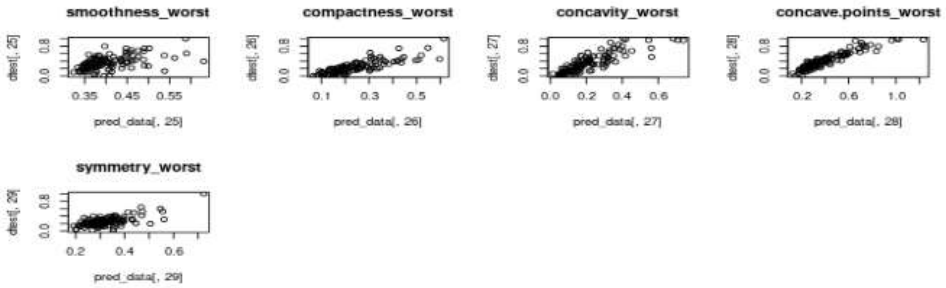
Imatge 18: Gràfics de dispersió reals vs predits de les variables 1-12 de la base 1

Observem que en general les prediccions són bastant bones excepte en la variable *texture_se* que sembla bastant aleatòria la assignació i la variable *texture_mean* que per valors elevats augmenta la variància.



Imatge 19: Gràfics de dispersió reals vs predits de les variables 13-24 de la base 1

En la imatge 18 veiem que les pitjors reconstruccions són les variables *texture_worst*, *compactness_se* i *concave.points_se*, encara que aquesta no tant malament com les altres dos.



Imatge 19: Gràfics de dispersió reals vs predits de les variables 25-30 de la base 1

En aquestes ultimes variables la pitjor seria *smoothness_worst* i *concavity_worst*, les altres semblen bastant millor representades.

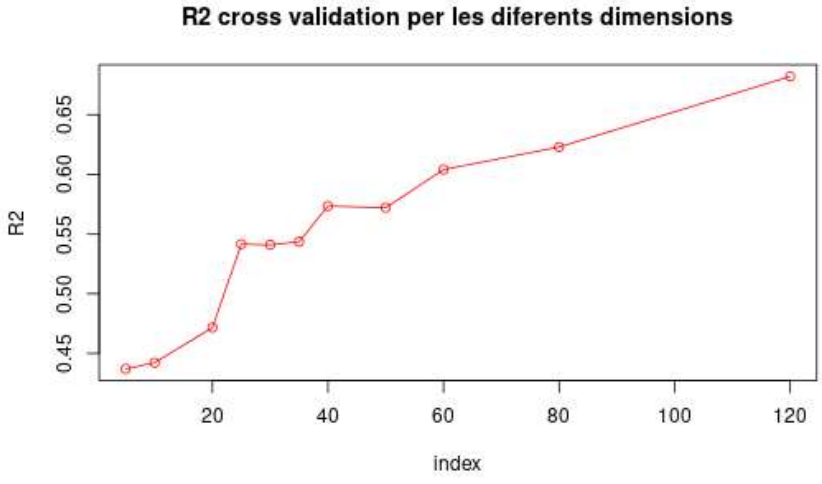
Elecció dimensió base de dades 2

Utilitzarem el mateix mètode de selecció de la millor dimensió que en la primera base de dades. Primerament, normalitzarem les variables amb el mètode min-max i farem 5 *folds* de les files per fer la validació creuada.

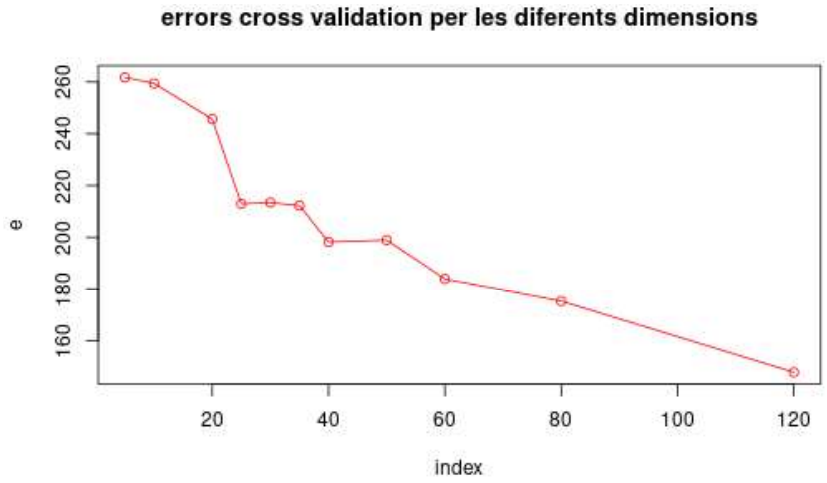
index	R2	e
5	0.4369688	261.7363
10	0.4421584	259.4498
20	0.4715755	245.6490
25	0.5416915	212.9111
30	0.5410369	213.4086
35	0.5437148	212.2351
40	0.5735233	198.1810
50	0.5722554	198.9164
60	0.6041297	183.8406
80	0.6230357	175.2712
120	0.6820407	147.7276

Taula 2: informació de la validació creuada de la base de dades 2

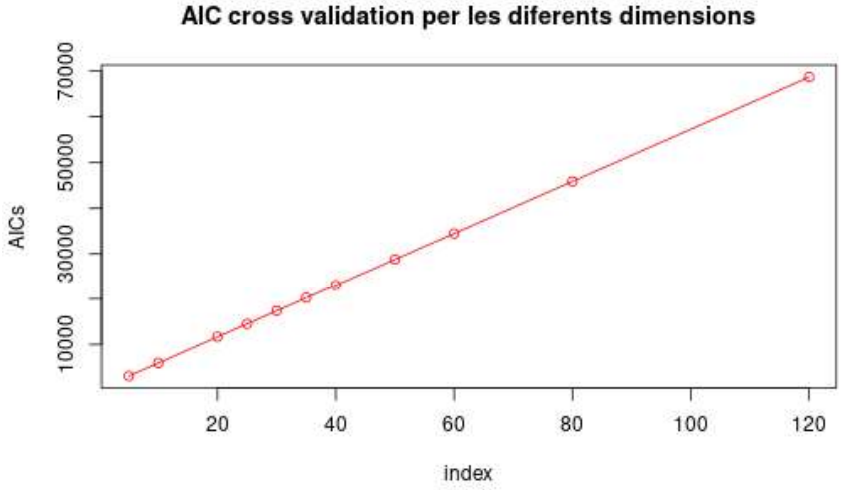
Si observem els gràfics 12 i 13 veiem que sembla que hi ha dos punts que podrien ser vàlids com colzes, però ens quedarem amb el segon punt, on el nombre de nodes a la capa oculta és 40 ja que considerem que el fet d'augmentar la dimensió fins aquest punt augmenta la variabilitat explicada i redueix els SSE suficient per considerar-lo millor punt "colze"



Gràfic 12: R^2 de les diferents dimensions de la base 2



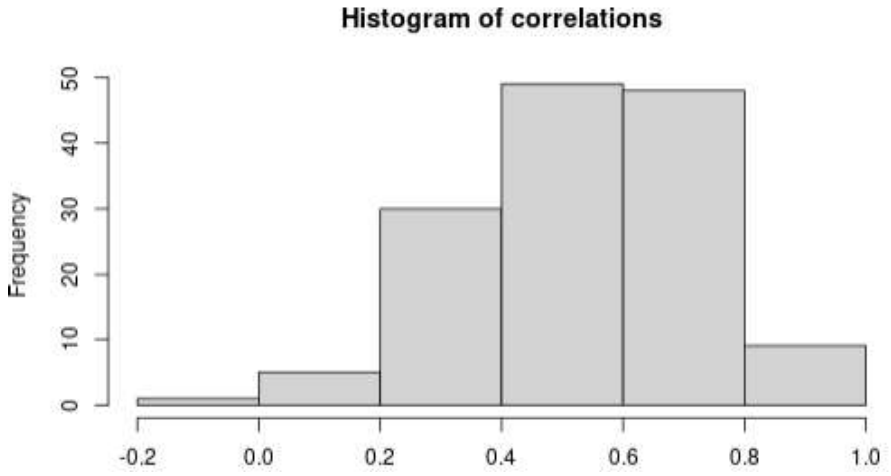
Gràfic 13: SSE de les diferents dimensions de la base 2



Gràfic 14: AIC de les diferents dimensions de la base 2

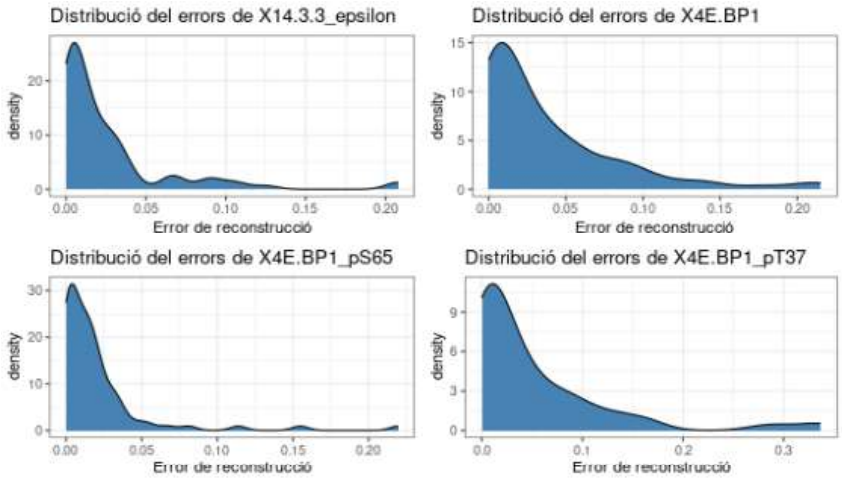
Veiem que pels AIC tenim el mateix problema que amb la base de dades 1, veiem que a mesura que augmentem la dimensió de la capa oculta fa que l'AIC augmenti i per tant ens indicaria que el fet d'agregar més dimensions no disminueix el MSE prou com perquè redueixi l'AIC.

Anàlisi reconstrucció base de dades 2



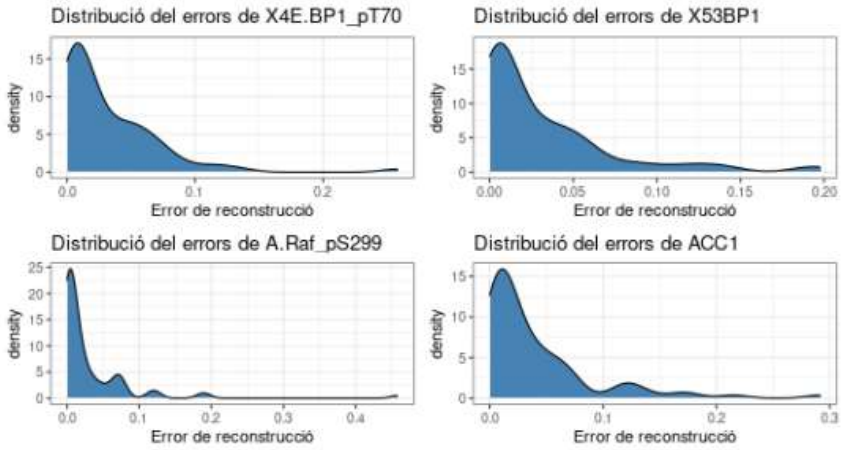
Gràfic 15: Histograma de les correlacions de la base 2

Observant el gràfic 15 veiem que han disminuït el nombre de variables amb correlació més gran de 0.8 i han augmentat les que tenen entre 0.2 i 0.4 i les que tenen entre 0.4 i 0.6 respecte a la primer base de dades.



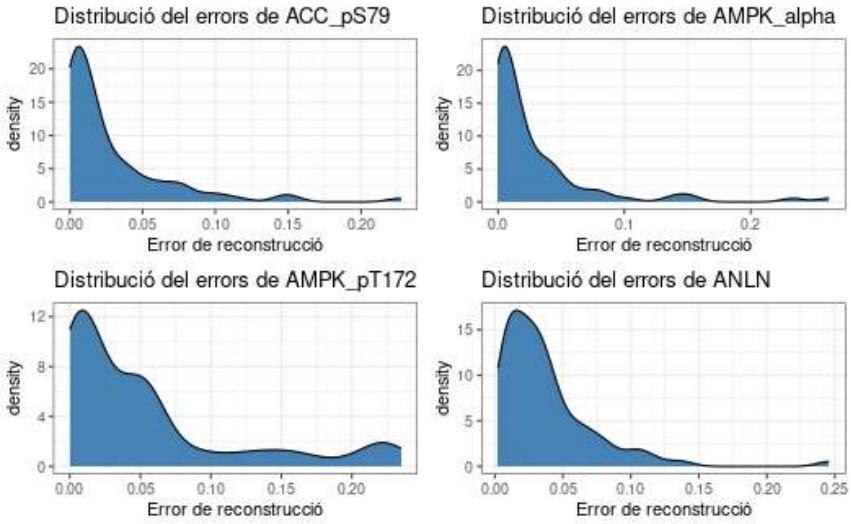
Imatge 20: Gràfics de densitat dels errors de les variables 1-4 de la base 2

Veiem que els errors en aquestes quatre variables és més gran que els que teníem a la base de dades anterior en general, i sembla que tenen més valors extrems.



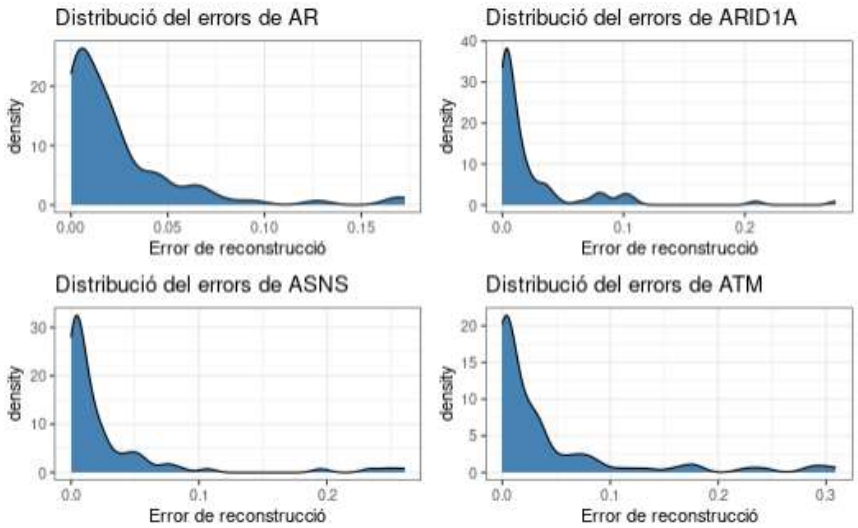
Imatge 21: Gràfics de densitat dels errors de les variables 5-8 de la base 2

En la imatge 21 veiem que la variable A.Raf_pS299 te uns errors més petits que les altres, i en general veiem que els errors han pujat.



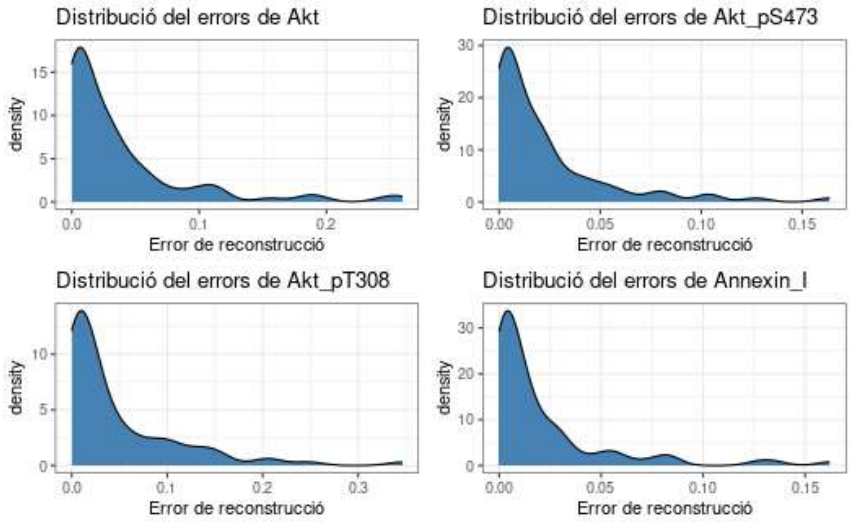
Imatge 22: Gràfics de densitat dels errors de les variables 9-12 de la base 2

Observem que les variables tenen errors bastants semblants, però veiem que la variable AMPK_pT172 té una cua bastant gran i bastants valors extrems.

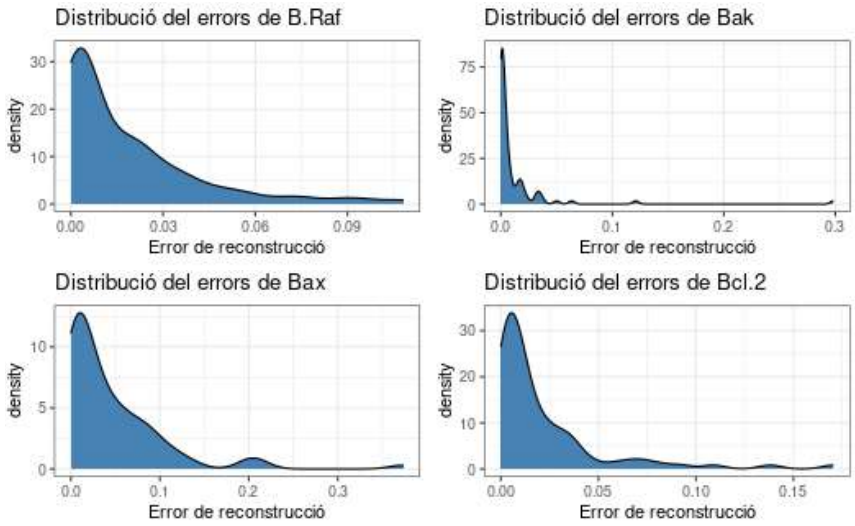


Imatge 23: Gràfics de densitat dels errors de les variables 13-16 de la base 2

En la imatge 23 els errors no són tan grans com en les variables anteriors però sembla que hi ha encara algun valor extrem.

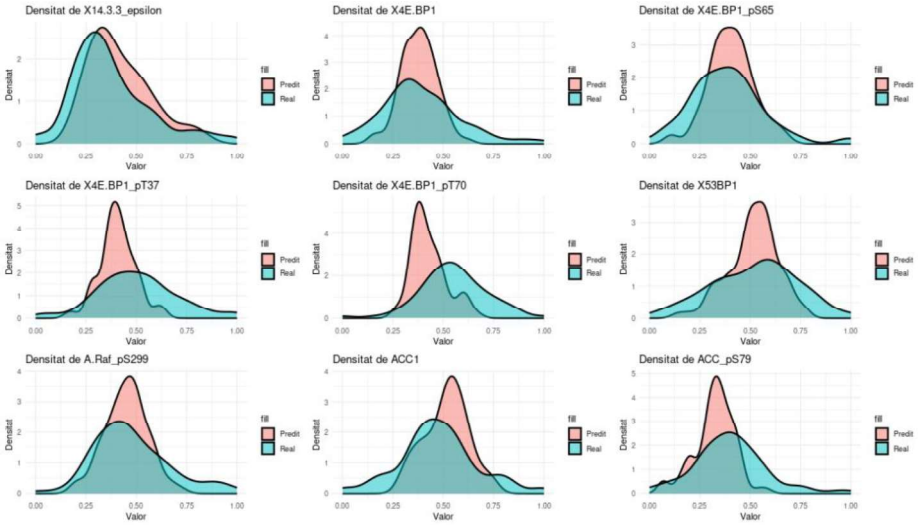


Imatge 24: Gràfics de densitat dels errors de les variables 17-20 de la base 2



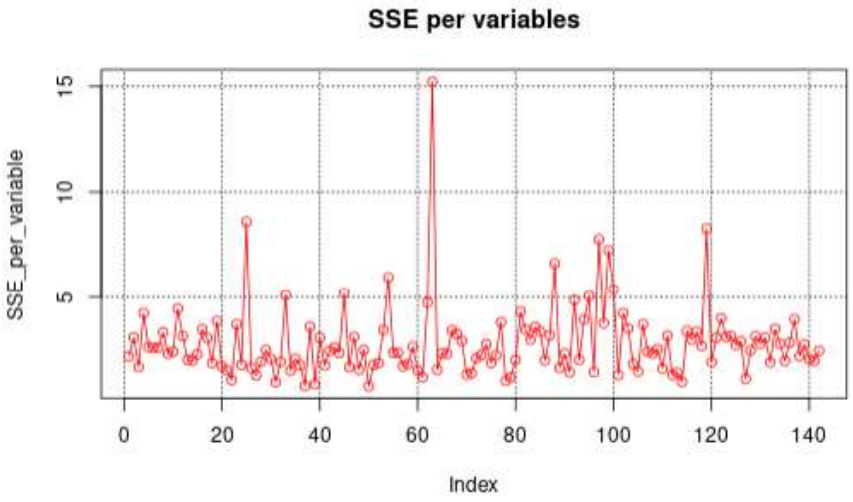
Imatge 25: Gràfics de densitat dels errors de les variables 21-24 de la base 2

En les imatges 24 i 25 veiem que els errors són bastants similars excepte la variable *Bak* que sembla tenir pocs errors però amb algun *outlier.*



Imatge 26: Gràfics de densitat reals vs predits de les variables 1-9 de la base 2

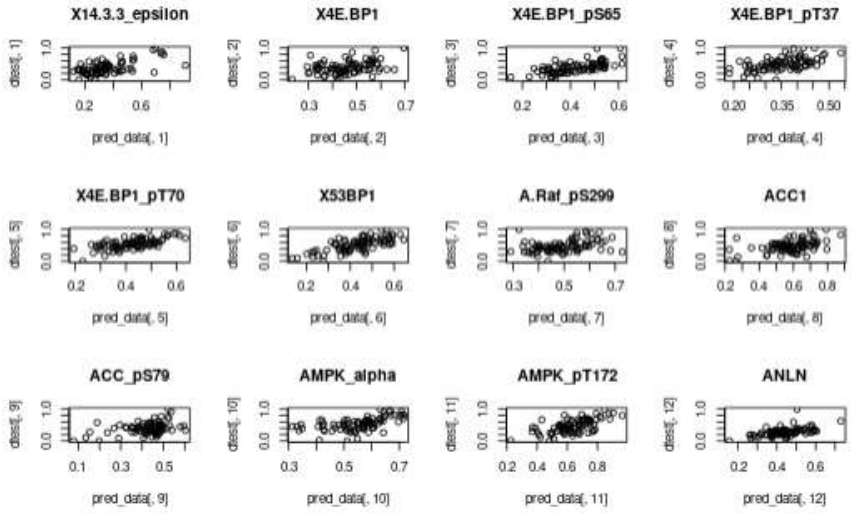
En la imatge 26, veiem que ha excepció de la primera variable, les altres han perdut bastant la forma de la seva distribució es pot veure com aquesta segona base de dades no ha aconseguit gaire bé mantenir les distribucions de les dades originals. Els altres gràfics es poden veure a l'annex. VID. Imatges 7.A-9.A



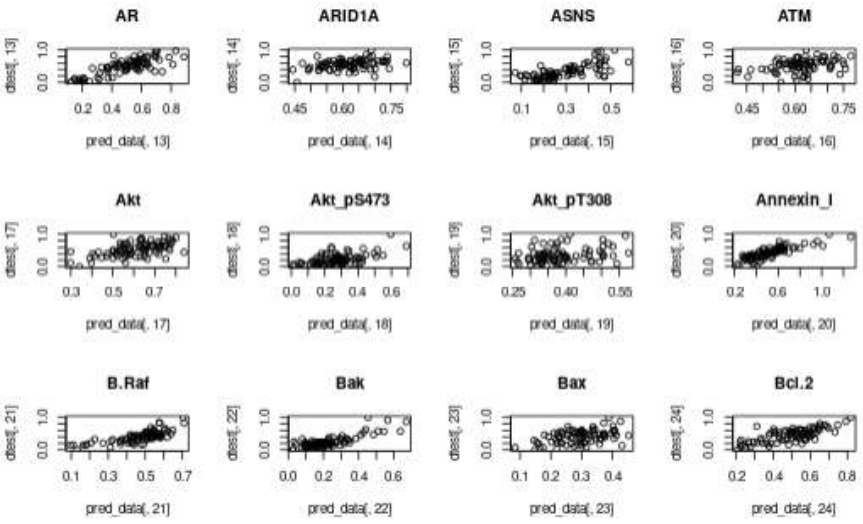
Gràfic 16: SSE per variable de la base 2

Pel que fa als SSE, veiem que no són massa elevats, a excepció d'alguns que estan una mica per sobre, inclús una variable que té un SSE superior a 15, però en general estan entre 0 i 5.

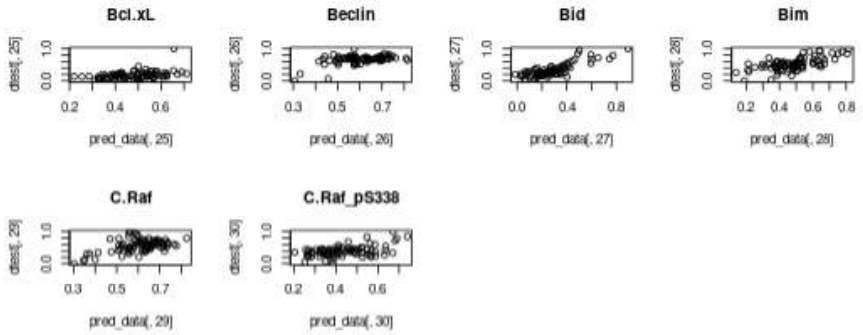
En les imatges 27, 28 i 29 següents veiem que els gràfics de dispersió ja no surten tan bé com a la primera base de dades, però bastants mantenen la forma de la recta encara que amb una variància més elevada.



Imatge 27: Gràfics de dispersió de les variables 1-12 de la base 2



Imatge 28: Gràfics de dispersió de les variables 13-24 de la base 2



Imatge 29: Gràfics de dispersió de les variables 25-30 de la base 2

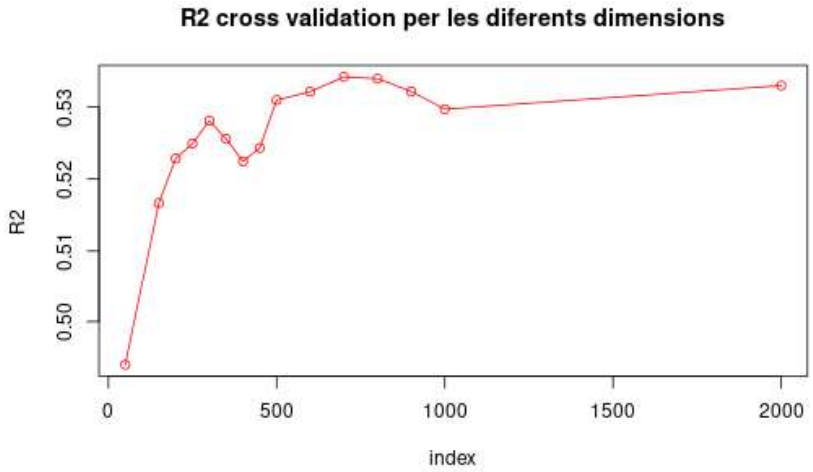
Elecció dimensió base de dades 3

En aquesta base de dades també utilitzarem el mateix mètode de les bases de dades anteriors, veurem com surten els R^2 i els SSE de les diferents validacions en la validació creuada.

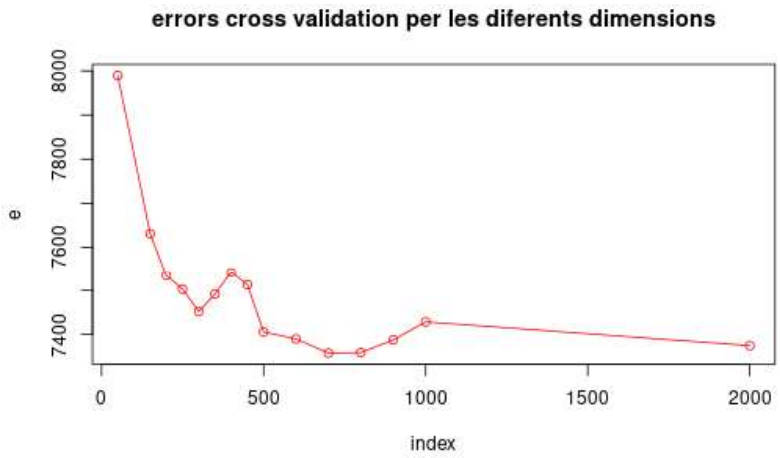
index	R2	e
50	0.4226316	5763.755
150	0.4663284	5328.716
200	0.4736442	5254.577
250	0.4804629	5186.269
300	0.4810876	5179.501
350	0.4806336	5184.252
400	0.4889478	5101.261
450	0.4838629	5153.412
500	0.4814229	5177.033
600	0.4868437	5125.236
700	0.4981258	5010.485
800	0.4969484	5022.031
900	0.4990148	5001.140
1000	0.5006328	4984.740
2000	0.4970955	5020.853

Taula 3: informació de la validació creuada de la base de dades 3

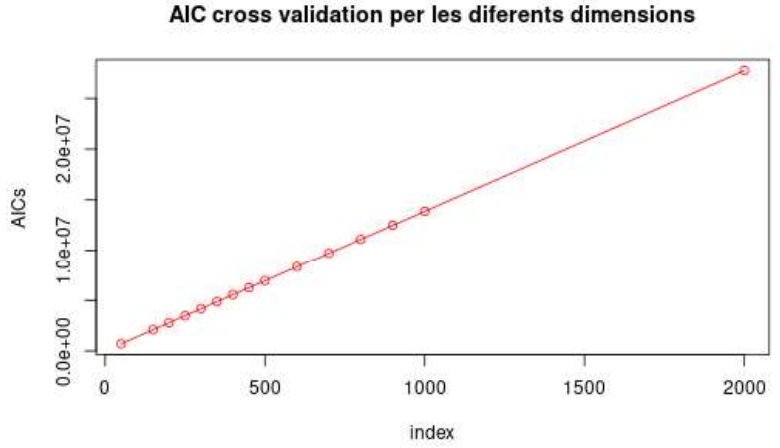
Veiem que tenint un nombre tan gran de variables, fa que l'R^2 i SSE no augmentin ni disminueixin tant a l'augmentar el nombre de nodes de la capa oculta, respectivament.



Gràfic 17: R^2 de les diferents dimensions de la base 3



Gràfic 18: SSE de les diferents dimensions de la base 3

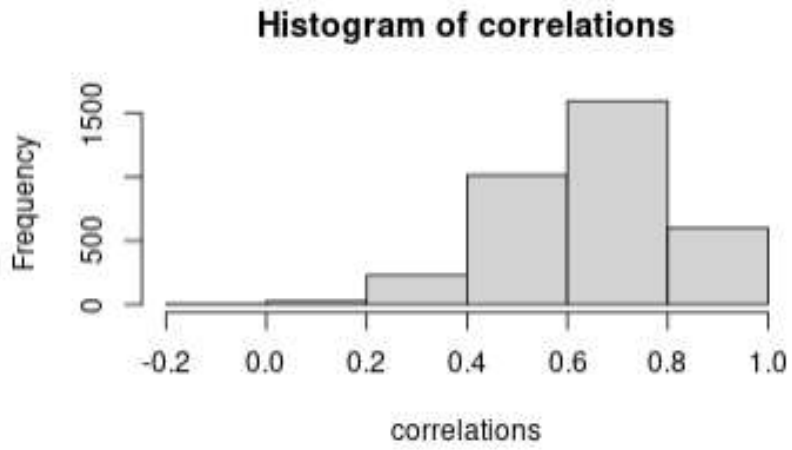


Gràfic 19: AIC de les diferents dimensions de la base 3

Veient els gràfics 17 i 18 podem dir que el nombre de nodes de la *hidden layer* que fa de colze en aquests gràfics seria 400 nodes. Veiem que el fet d'augmentar 300 el nombre de nodes de la capa oculta, només fa augmentar la variabilitat explicada en un 1%, per tant, ens quedarem amb 400 variables per la capa oculta com a millor nombre de nodes.

Anàlisi reconstrucció base de dades 3

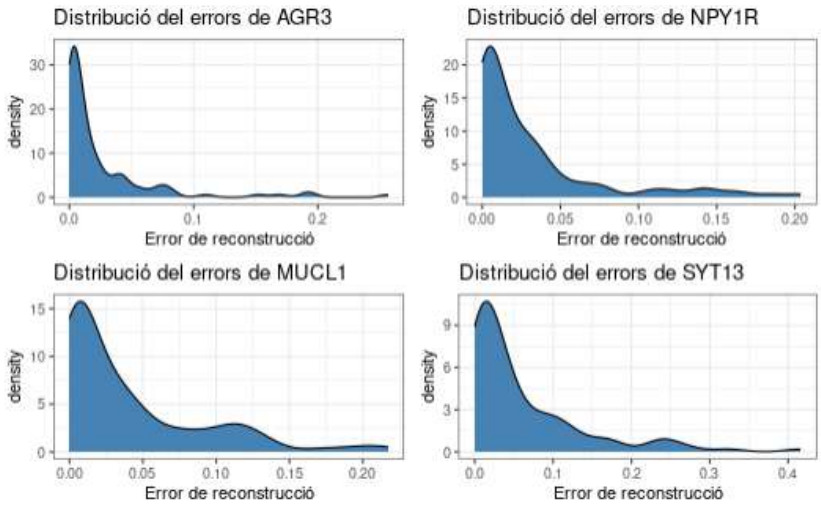
Analitzarem ara la reconstrucció de la base de dades reduïda anterior, amb un nombre de nodes a la capa oculta de 400 calculat anteriorment.



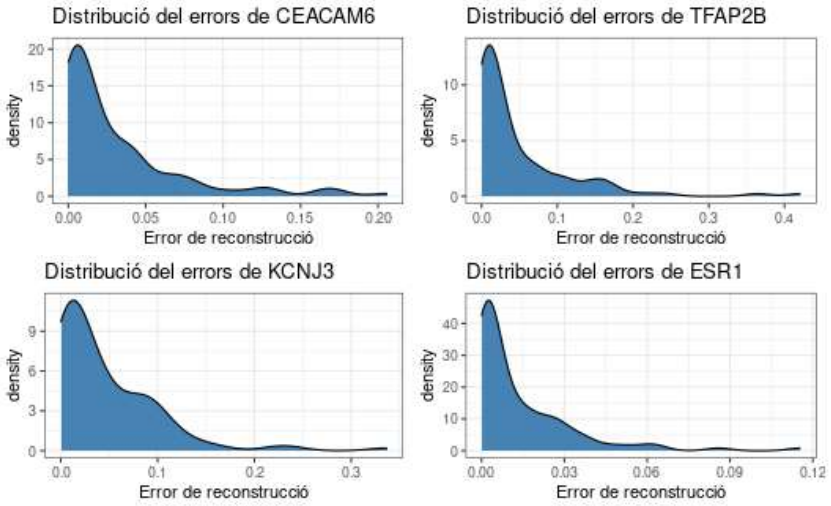
Gràfic 20: Histograma de les correlacions de la base 3

Observem que tenim la gran majoria de variables entre 0.4 i 0.8 de correlació, i sembla que la distribució de les correlacions és més o menys la mateixa que a la base de dades 2.

Veiem ara que les distribucions dels errors de reconstrucció d'algunes de les variables i comentarem els més destacables.

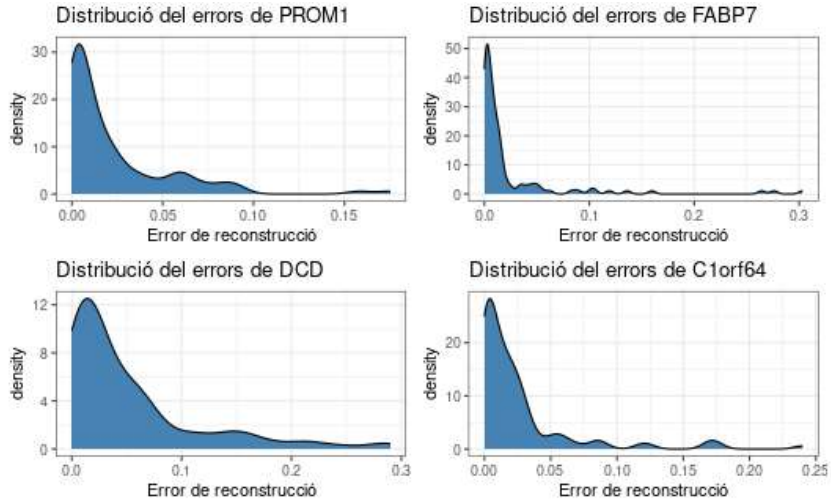


Imatge 30: Gràfics de densitat dels errors de les variables 1-4 de la base 3

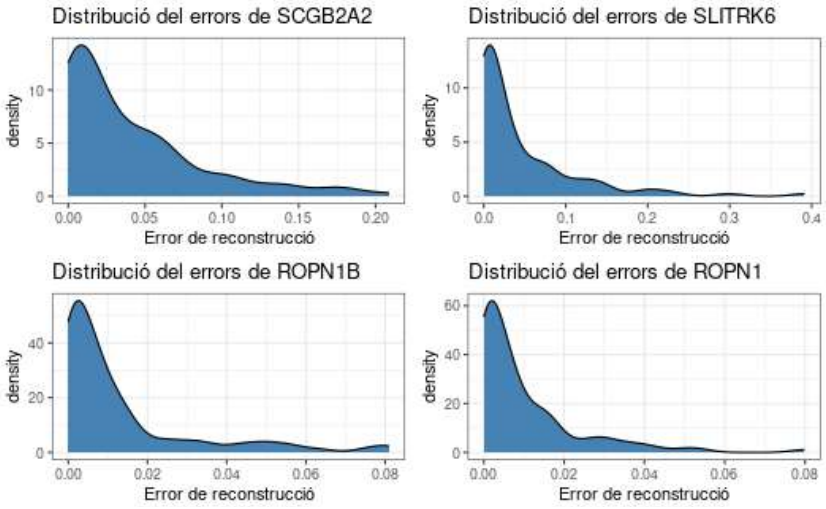


Imatge 31: Gràfics de densitat dels errors de les variables 5-8 de la base 3

En les imatges 30 i 31 veiem que les variables SYT13 i TFAP2B, tenen valors extrems molt grans, arribant a 0,4 d'error en alguns casos.

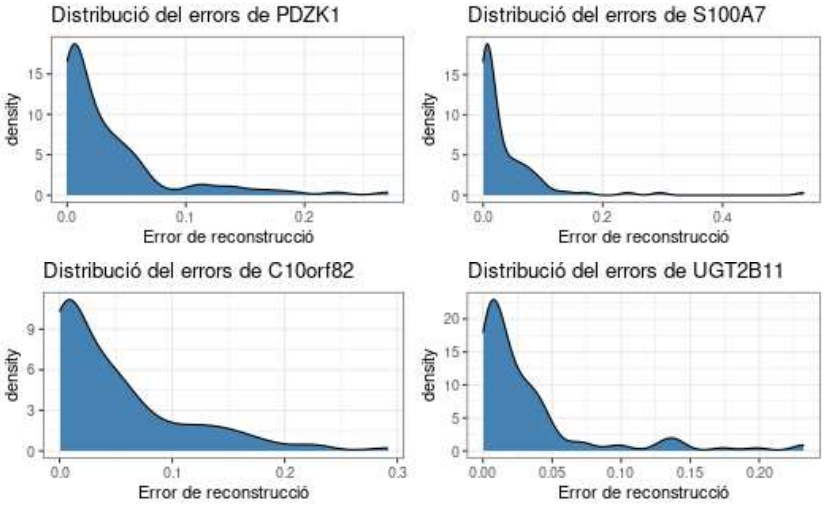


Imatge 32: Gràfics de densitat dels errors de les variables 9-12 de la base 3

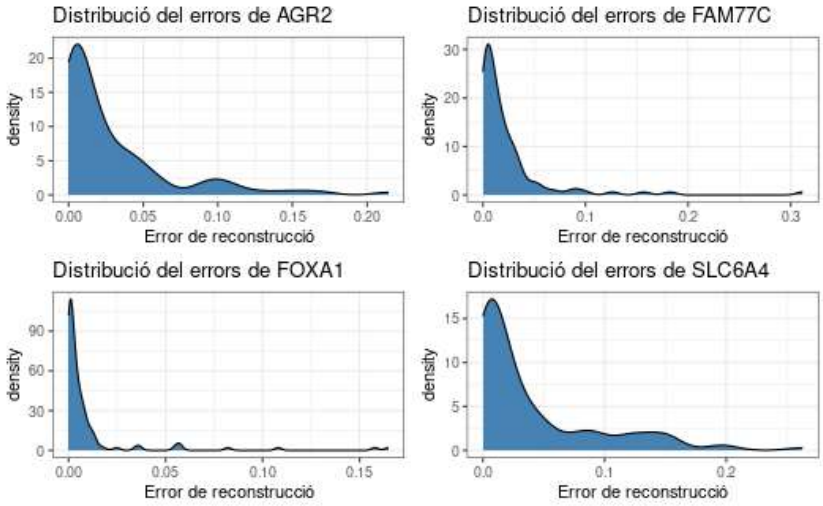


Imatge 33: Gràfics de densitat dels errors de les variables 13-16 de la base 3

En els gràfics 32 i 33 veiem que la variable SLITRK6 també té errors grans fins 0,4 i en canvi les variables ROPN1B i ROPN1 tenen errors molt petits, com a molt de 0,08

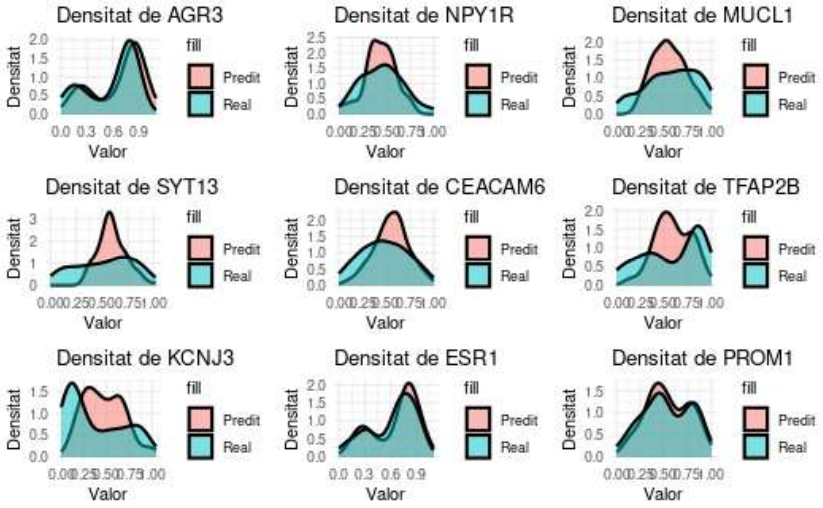


Imatge 34: Gràfics de densitat dels errors de les variables 17-20 de la base 3



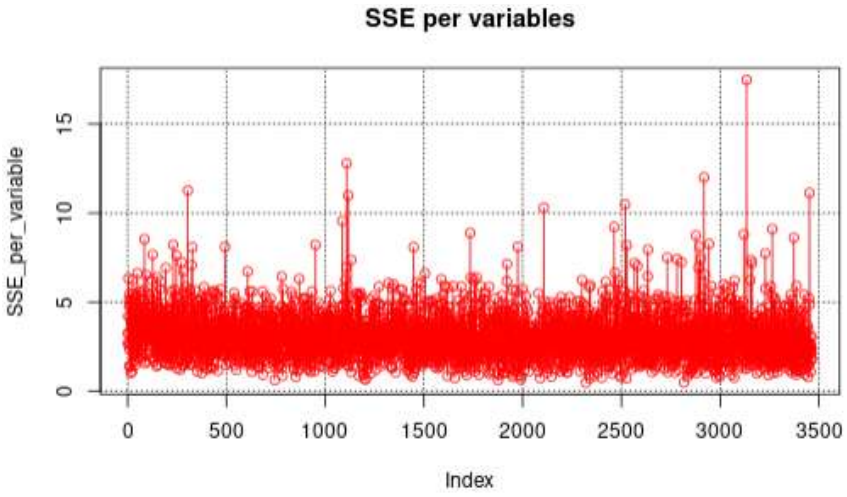
Imatge 35: Gràfics de densitat dels errors de les variables 21-24 de la base 3

En les imatges 34 i 35 veiem que la variable S100A7 té valors extrems encara més grans sobrepassant el 0,5. Les altres distribucions són semblants entre elles.



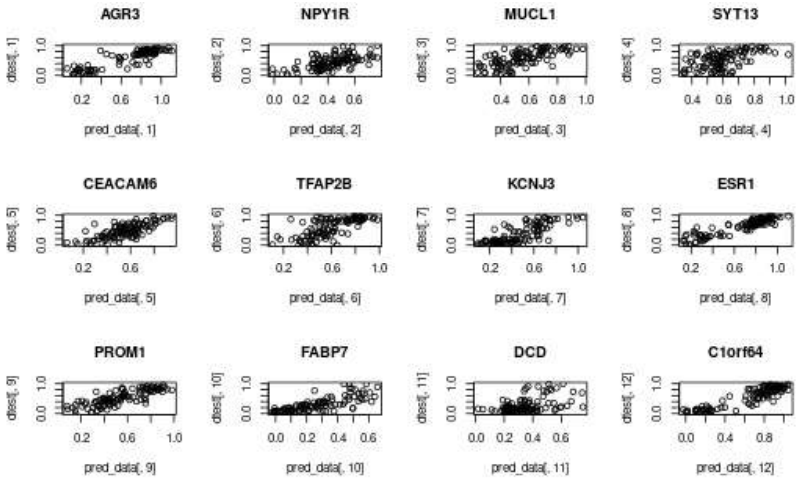
Imatge 36: Gràfics de densitat real vs predit de la base 3

En aquest cas, sembla que hi ha més variables ara que en la base de dades 2 que han mantingut la distribució original, encara que semblen una mica més canviades, però hi ha més variables que abans que ho mantenen. Les altres imatges amb les densitats queden a l'annex. VID Imatges 10.A-12.A

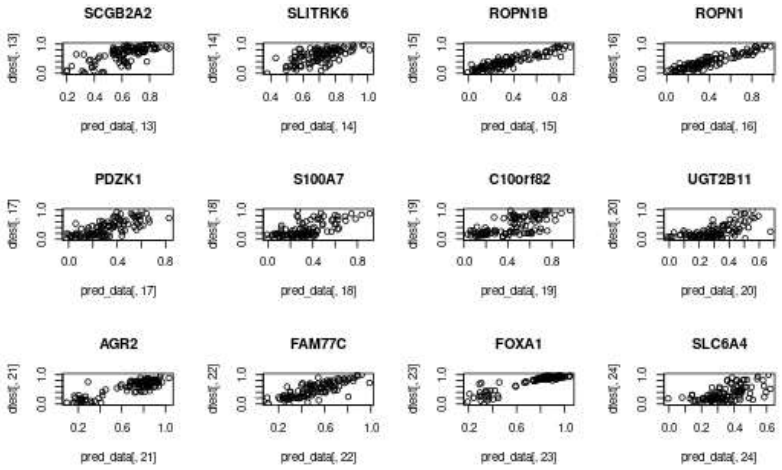


Gràfic 20: SSE per variable de la base 3

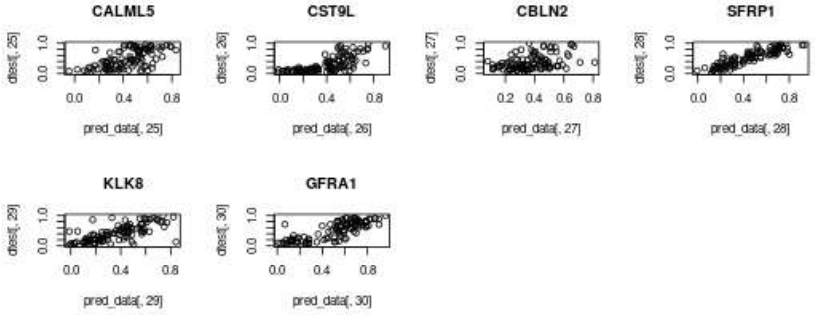
En el gràfic 20 veiem que la majoria de les variables tenen una suma d'errors quadràtics petita, però també ha augmentat el nombre de variables per sobre de 5 i fins i tenim una variable que supera els 15 de SSE. Sembla que ha augmentat el nombre de variables amb SSE més elevats o més lluny del 0 que amb les altres dos bases de dades.



Imatge 37: Gràfics de dispersió de les variables 1-12 de la base 3



Imatge 38: Gràfics de dispersió de les variables 13-24 de la base 3



Imatge 39: Gràfics de dispersió de les variables 25-30 de la base 3

En els gràfics de dispersió trobem que a excepció d'algunes variables malament relacionades com NUP62CL o C10orf93, les altres segueixen la tendència de les dades originals, encara que es pot observar com ha augmentat la variabilitat de les variables ja que les rectes són més amples.

Conclusions

Com a conclusions veiem que a les tres bases de dades hem reduït bastant les dimensions, a la primera base de dades ens hem quedat amb un 16,67% de les columnes de la base de dades completa, reduint de 30 a 5 variables, a la segona base de dades, on hem reduït menys la dimensió, veiem que ens hem quedat amb un 28,17% de les columnes originals i per últim a la tercera base de dades, on hem reduït més la dimensió, ens hem quedat amb un 11,54% de les variables de la base de dades real.

Podem dir que a la primera base de dades, l'autoencoder ha funcionat perfectament reduint les dades i mantenint gairebé un 0.8 de variabilitat explicada, mentre que per les altres aquesta variabilitat ha baixat considerablement.

Observem també que encara que a la tercera base de dades els SSE han estat per sobre que la resta, s'han mantingut les distribucions millor que en el cas de la segona base de dades, on les distribucions no s'han mantingut gaire encara que els errors eren en general més petits.

Podem observar que de les tres mesures utilitzades per decidir el millor nombre de nodes per fer la reconstrucció, l'AIC no ens ha ajudat, ja que en tots els casos ens sortia una recta que l'única informació que ens donava és que el millor node era el més petit possible, pel fet que augmentar el nombre de nodes no feia augmentar la versemblança, en el nostre cas reduir el MSE, prou com perquè el model sigui millor. Per tant, s'ha optat per utilitzar les altres dues mesures per decidir el millor nombre de nodes de la *hidden layer,* l'R^2 i el SSE.

Amb més temps es podrien haver provat autoencoders apilats, amb múltiples capes en el coll d'ampolla i fer reduccions progressives en cada part, ja que sobretot a la tercera base de dades la reducció de dimensió ha estat de 3466 variables a 400. El fet d'utilitzar múltiples capes podria millorar raonablement la reconstrucció de la última base de dades.

Bibliografia

[1]. Chollet F., Kalinowski T., & Allaire J. J. (2022). *Deep learning with R (Second Edition)*. https://www.simonandschuster.com/books/Deep-Learning-with-R-Second-Edition/Francois-Chollet/9781633439849

[2]. Anders U., & Korn O. (1996). *Model selection in neural networks*. Zentrum für Europäische Wirtschaftsforschung (ZEW), Mannheim. https://www.econstor.eu/handle/10419/29449

[3]. William H., W. Nick Street, & Olvi L. Mangasarian. (2023). Breast cancer. Kaggle. https://www.kaggle.com/datasets/imtkaggleteam/breast-cancer

[4]. Marco C., Nicole B., Alessia M., Giuseppe J., Alessandro Z. & Cesare F. (2020). Integrative Network Fusion: a multi-omics approach in molecular profiling. https://figshare.com/articles/dataset/Integrative_Network_Fusion_a_multi-omics_approach_in_molecular_profiling/12052995/1

[5]. Jordi C., Toni L. & Anna B. (2020). Deep Learning principios y fundamentos. https://www.editorialuoc.com/deep-learning

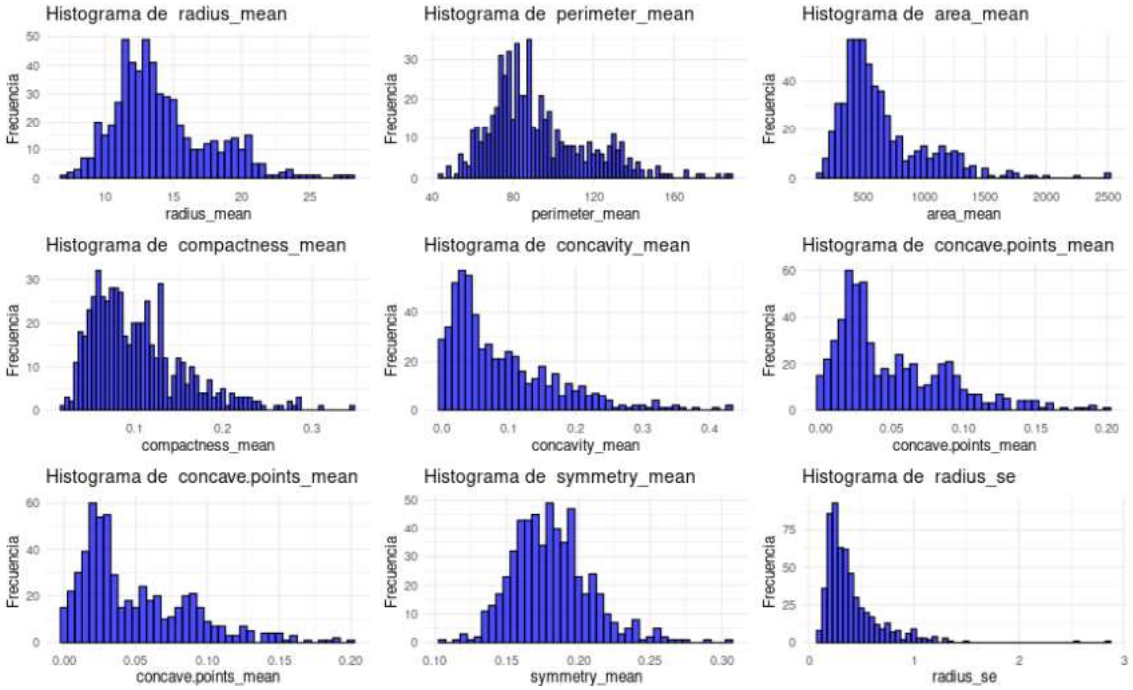
[6]. Fars S. & Thomas S. (2021). On estimating the optimal autoencoder model for denoising ECG using Akaike Information Criterion. https://www.researchgate.net/publication/353105368_On_estimating_the_optimal_autoencoder_model_for_denoising_ECG_using_Akaike_Information_Criterion

[7]. Jeremy J. (2018) Introduction to autoencoders. https://www.jeremyjordan.me/autoencoders/

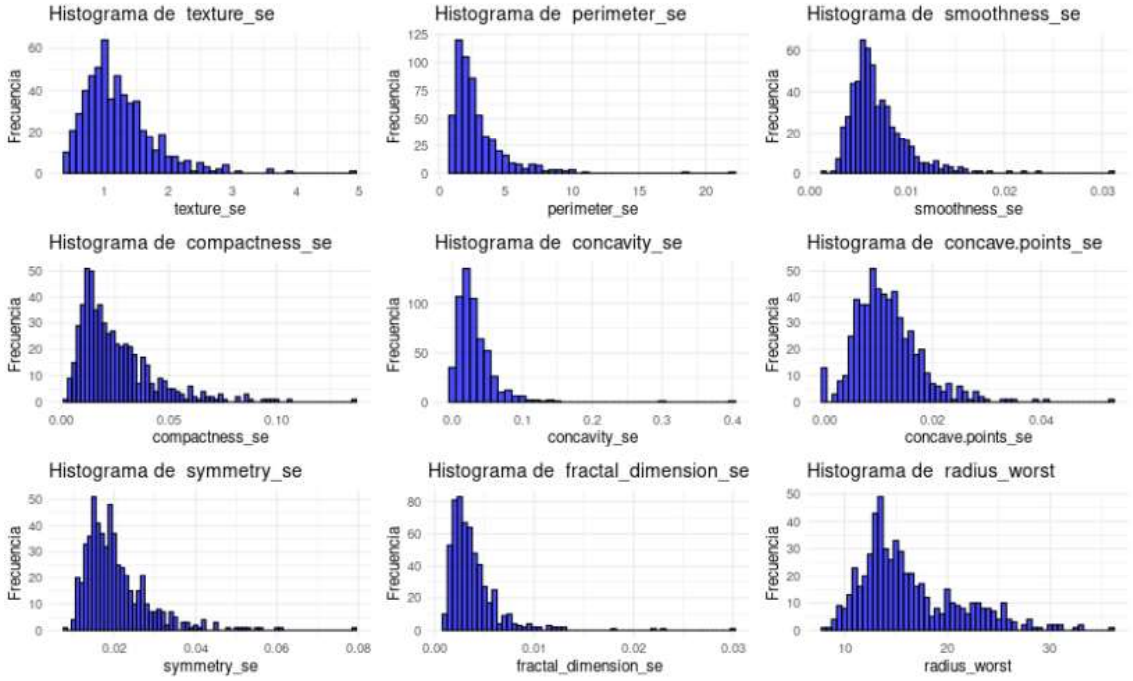
[8]. Guillaume A. & Yoshua B. (2014) What Regularized Auto-Encoders Learn from the Data Generating Distribution. https://arxiv.org/pdf/1211.4246

[9]. TensorFlow. (s. f.) TensorFlow for R – Basic Regression. https://tensorflow.rstudio.com/tutorials/keras/regression

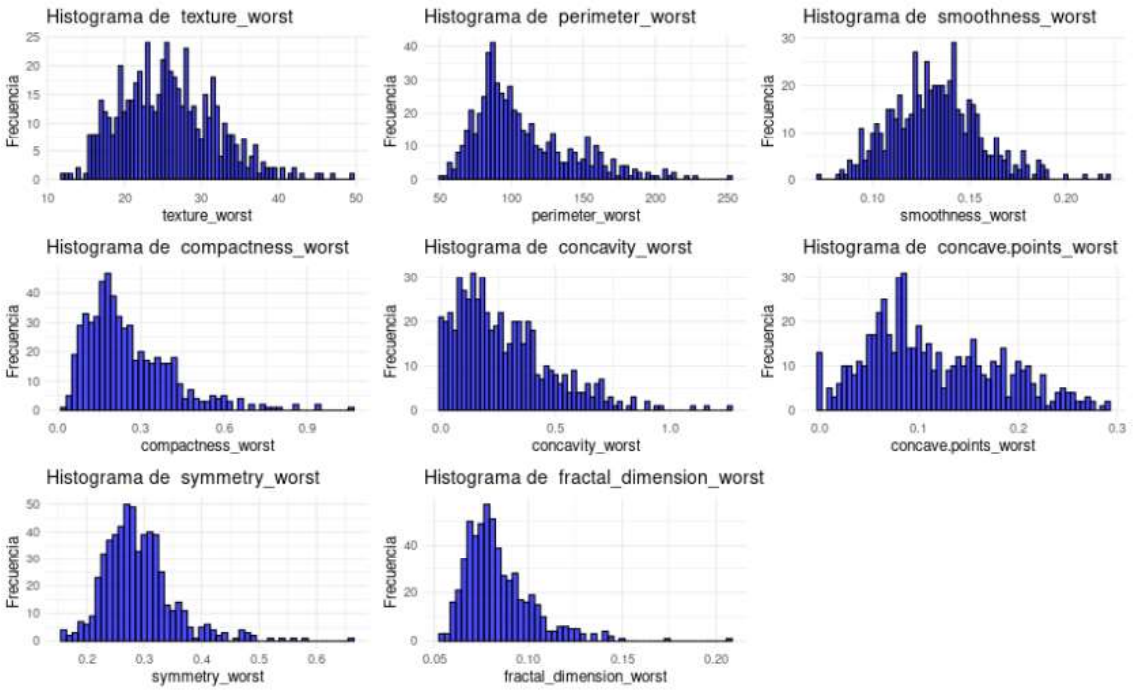
Annex 1: Gràfics addicionals



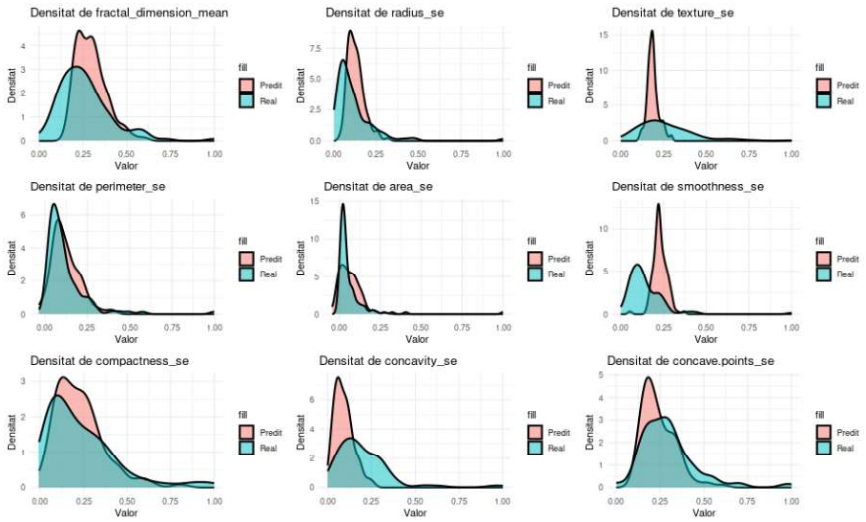
Imatge 1.A: Histogrames 1 de la base 1



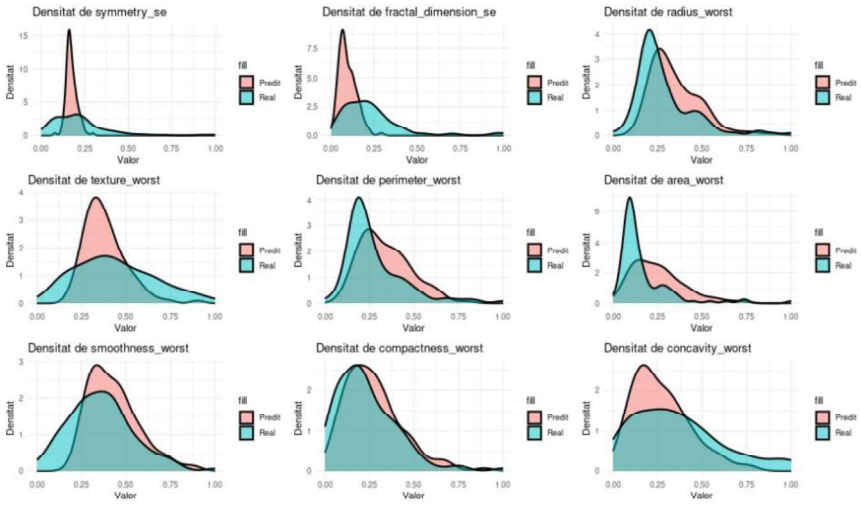
Imatge 2.A: Histogrames 2 de la base 1



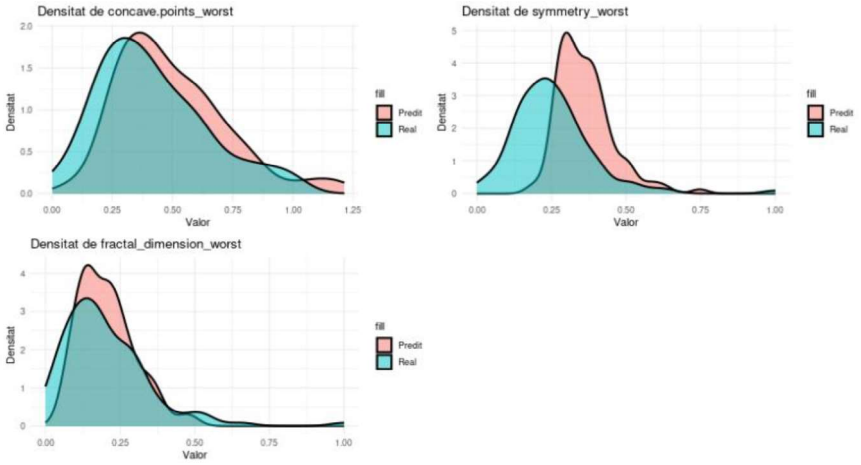
Imatge 3.A: Histogrames 3 de la base 1



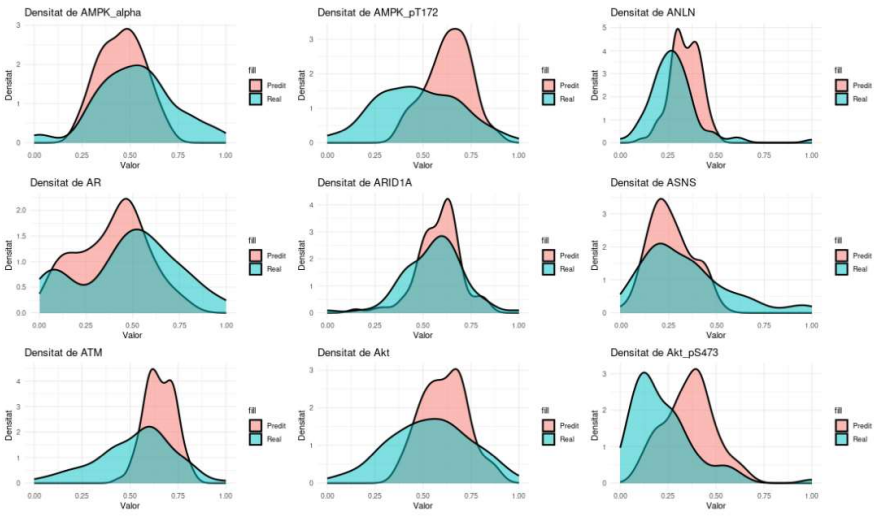
Imatge 4.A: Gràfics de densitat real vs predit variables 10-18 de la base 1



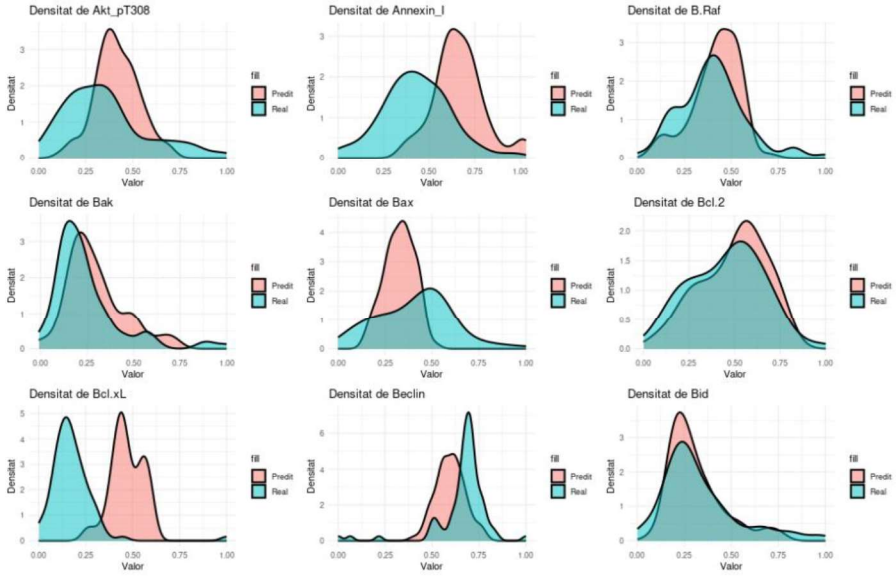
Imatge 5.A: Gràfics de densitat real vs predit variables 19-27 de la base 1



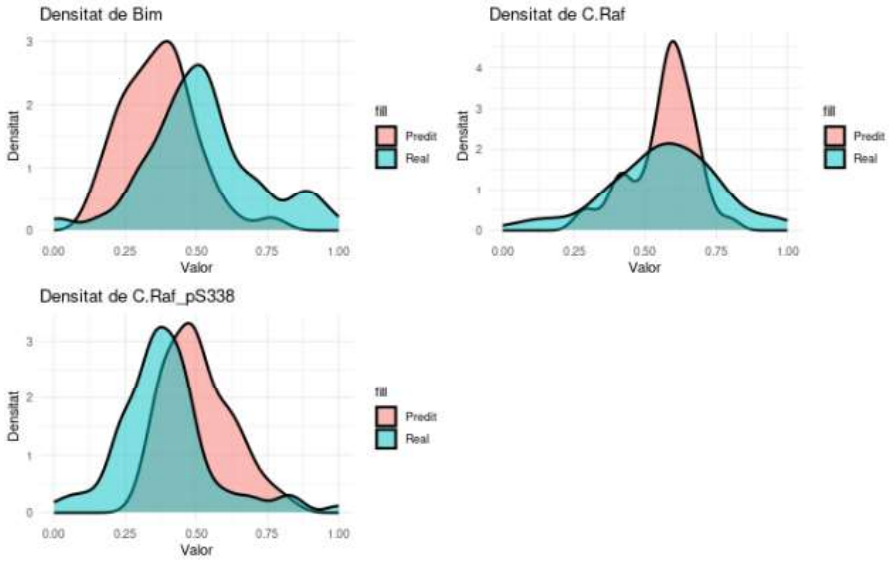
Imatge 6.A: Gràfics de densitat real vs predit variables 28-30 de la base 1



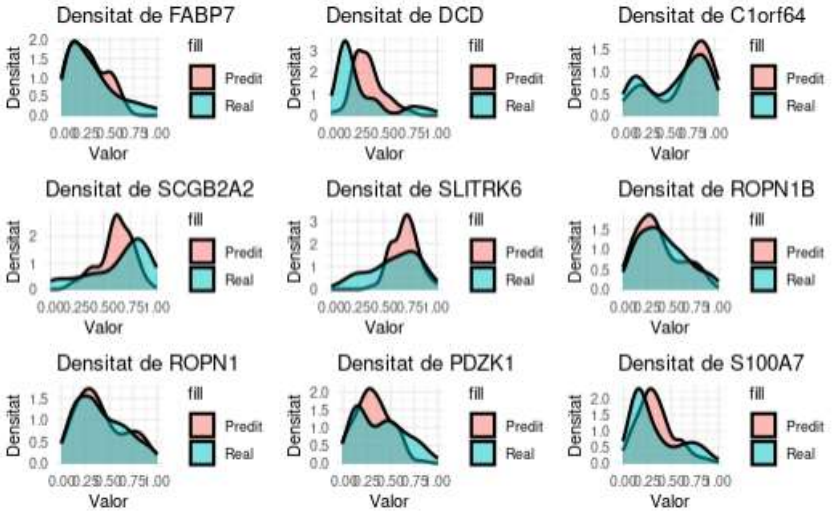
Imatge 7.A: Gràfics de densitat real vs predit variables 10-18 de la base 2



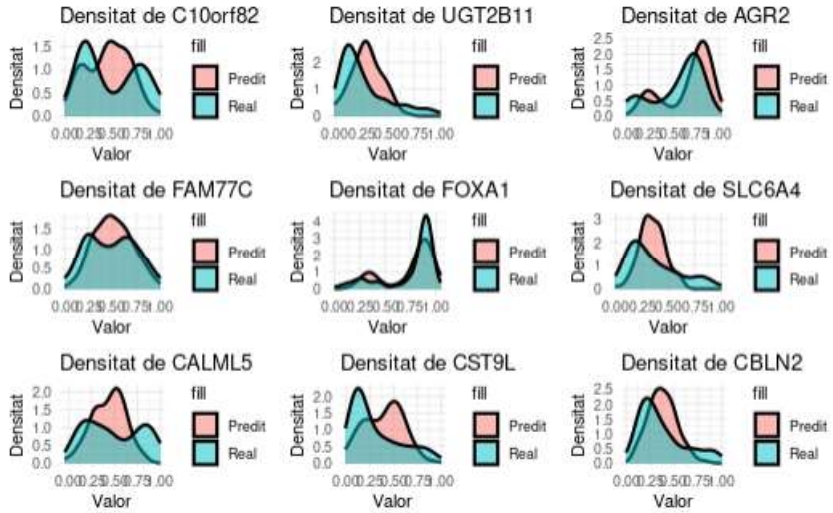
Imatge 8.A: Gràfics de densitat real vs predit variables 19-27 de la base 2



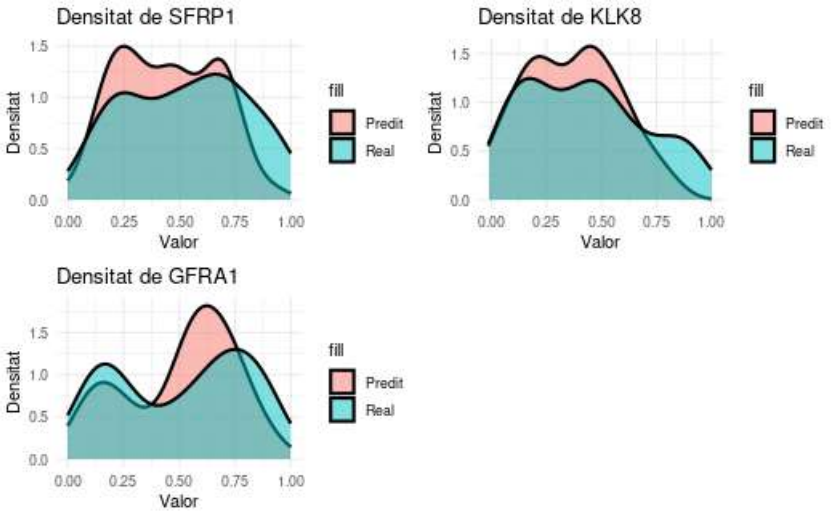
Imatge 9.A: Gràfics de densitat real vs predit variables 28-30 de la base 2



Imatge 10.A: Gràfics de densitat real vs predit variables 10-18 de la base 3



Imatge 11.A: Gràfics de densitat real vs predit variables 19-27 de la base 3



Imatge 12.A: Gràfics de densitat real vs predit variables 28-30 de la base 3

Annex 2: Codi

En aquest annex s'afegirà únicament el codi utilitzat per fer la validació creuada i els autoencoders de la primera base de dades, que són iguals per a les altres dues.

```{r, echo=FALSE, warning=FALSE}

set.seed(20326180)

ind = sample(1:nrow(data), size = round(0.8*nrow(data)))

dtrain = data[ind,]
dtrain = dtrain[,-(1:2)]
dtest = data[-ind,]
dtest<-dtest[,-(1:2)]
dtrain<- as.matrix(dtrain)
dtest<- as.matrix(dtest)
data2<-data[,-(1:2)]

```

```{r}
# function
normalize <- function(x, na.rm = TRUE) {
  return((x- min(x)) /(max(x)-min(x)))
}

dtrain<-apply(dtrain,2,normalize)
dtest<-apply(dtest,2,normalize)

```

```{r}
### K-fold Cros validation


K <- 5
index<-c(1,2,3,4,5,6,7,8,9,10,15,20)
input_size<- 30
n <- nrow(data2)
p <- ncol(data2)
r2<-matrix(ncol=length(index),nrow=5)
r2_ajust<-matrix(ncol=length(index),nrow=5)
errors<-matrix(ncol=length(index),nrow=5)
AIC<-matrix(ncol=length(index),nrow=5)

folds <- sample(rep(1:K, length.out=nrow(data2)))
table(folds)


data2<-apply(data2,2,normalize)
```

```{r,warning=FALSE}

for(l in 1:length(index)){

inx<-index[l]

for(i in 1:K){

 val <- data2[folds == i, ]
  tr <- data2[folds != i, ]
```

```
  tr<- as.matrix(tr)
val<- as.matrix(val)

  enc_input=layer_input(shape= input_size)
enc_output= enc_input %>%
  layer_dense(units=inx,activation="relu") %>%
  layer_activation_leaky_relu()

encoder=keras_model(enc_input,enc_output)

dec_input= layer_input(shape= inx)
dec_output= dec_input %>%
  layer_dense(units=input_size,activation ="linear") %>%
  layer_activation_leaky_relu()

decoder= keras_model(dec_input,dec_output)

aen_input = layer_input(shape=input_size)
aen_output= aen_input %>%
  encoder() %>%
  decoder()

aen= keras_model(aen_input, aen_output)

  aen %>% compile(optimizer="adam", loss="mse")
  aen %>% fit(tr,tr, epochs=35, batch_size=40)

  pred_data= aen %>% predict(val)

loss<-(val-pred_data)^2
n_val <- nrow(val)
```

```
errors[i,l]<-sum(loss)
r2[i,l] <- 1 - (sum((val - pred_data)^2) / sum((val - mean(val))^2))

mse<-mean(loss)
k<-count_params(aen)

AIC[i,l]<-2*k - 2*mse
}

}
```

```{r}
R2<-colMeans(r2)
e<-colMeans(errors)
AICs<-colMeans(AIC)
tab01<-cbind(index,R2,e,AICs)

kbl(tab01) %>%
   kable_styling()
```

```{r}
plot(x=index,y=R2,type="o",col="red",main="R2 cross validation per les diferents dimensions")

plot(x=index,y=e,type="o",col="red",main="errors cross validation per les diferents dimensions")

plot(x=index,y=AICs,type="o",col="red",main="AIC cross validation per les diferents dimensions")
```

```

```{r}
input_size<- 30


latent_size<-5
enc_input=layer_input(shape= input_size)
enc_output= enc_input %>%
  layer_dense(units=latent_size,activation="relu") %>%
  layer_activation_leaky_relu()

encoder=keras_model(enc_input,enc_output)

dec_input= layer_input(shape= latent_size)
dec_output= dec_input %>%
  layer_dense(units=input_size,activation ="linear") %>%
  layer_activation_leaky_relu()

decoder= keras_model(dec_input,dec_output)

aen_input = layer_input(shape=input_size)
aen_output= aen_input %>%
  encoder() %>%
  decoder()

aen= keras_model(aen_input, aen_output)


aen %>% compile(optimizer="adam", loss="mse")

aen %>% fit(dtrain,dtrain, epochs=50, batch_size=40)
```
```

```
decoded_data= aen %>% predict(dtest)

pred_data <- array_reshape(decoded_data, dim(dtest))

errors<-(dtest-pred_data)^2

SSE<-sum(errors)

r2 <- 1 - (sum((dtest - pred_data)^2) / sum((dtest - mean(dtest))^2))
```