Trabajo Fin de Máster

Aplicación de GNNs na clasificación de coníferas a partir de nubes de puntos LiDAR

Martín Fernández Pérez

Máster en Técnicas Estadísticas

Curso 2021-2022

* Propuesta de Trabajo Fin de Máster

Título en galego: Aplicación de GNNs na clasificación de coníferas a partir de nubes de puntos LiDAR
Título en español: Aplicación de GNNs en la clasificación de coníferas a partir de nubes de puntos LiDAR
English title: Aplication of GNNs in coniferous classification context using LiDAR data clouds
Modalidad: Modalidad A
Autor/a: Martín Fernández Pérez, Universidade de Santiago de Compostela
Tutor/a: Javier Roca Pardiñas, Universidade de Vigo; Julia Armesto Gonzalez, Universidade de Vigo
Breve resumen del trabajo: Neste traballo comparáronse dous métodos de clasificación para nubes de puntos LiDAR co obxectivo de identificar especies de coníferas presentes en Galicia. O primeiro método esta basado na extracción manual de features e emprega un XGBoost como clasificador mentras que o segundo esta basado en redes neuronais permite evitar a extracción de features.
Recomendaciones: Coñecementos básicos de aprendizaxe estatístico e de arquitectura de redes neuronais.
Otras observaciones:

*

Don/doña Javier Roca Pardiñas, Profesor titular Universidade de Vigo de Universidade de Vigo, y don/doña Julia Armesto Gonzalez, Profesora titular Universidade de Vigo de Universidade de Vigo, informan que el Trabajo Fin de Máster titulado

Aplicación de GNNs na clasificación de coníferas a partir de nubes de puntos LiDAR

fue realizado bajo su dirección por don/doña Martín Fernández Pérez para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En Santiago de Compostela, a 2 de Septiembre de 2022.

El/la tutor/a:

ROCA PARDIÑAS JAVIER - 77592181W

Firmado digitalmente por ROCA PARDIÑAS JAVIER - 77592181W
Fecha: 2022.08.31 14:40:16 +02'00'

Don/doña Javier Roca Pardiñas

El/la tutor/a:

ARMESTO GONZALEZ JULIA - 53114334C

Firmado digitalmente por ARMESTO GONZALEZ JULIA - 53114334C
Fecha: 2022.09.02 10:33:53 +02'00'

Don/doña Julia Armesto Gonzalez

El/la autor/a:

Firmado por FERNANDEZ PEREZ MARTIN - ***7627** el día 02/09/2022 con un certificado emitido por AC FNMT Usuarios

Don/doña Martín Fernández Pérez

Agradecementos

En primeiro lugar quero agradecerlles aos meus titores Javier e Julia o tempo que adicaron a axudarme na realización do traballo e pola libertade que me deron a hora de facer tantas probas como quixen. A Celestino por orientarme durante a escritura do traballo e ao meu amigo Pedro por introducirme as GNN's e todos os bos consellos sen os cales este traballo non seria posible. A miña aboa Rosario por regalarme este flamante ordenador co que aprendin todo o que sei de programación. A meus pais e amigos por todo o apoio e paciencia nos meses mais complicados do Máster, especialmente a miña moza Darlyn que ademais me recordou cantidade de datas importantes que a min se houberan esquecido e a Diego, Lidia e Breixo pola sua compañía tanto nas horas de estudo coma de celebración en Pexigo.

Índice xeral

Resumo

Resumo en galego

Neste traballo comparáronse dous métodos de clasificación para nubes de puntos LiDAR co obxectivo de identificar especies de coníferas presentes en Galicia *P. sylvestris*, *P. pinaster* e *P. radiata*. O primeiro método esta basado na extracción manual de variables descriptivas da distribución de alturas para posteriormente empregar un XGBoost como clasificador. O segundo método esta basado nas recentes arquitecturas de redes neuronais de grafos e permite traballar directamente coas coordenadas xyz. Ambos métodos foron comparados na clasificación punto a punto en segmentos obtendo mellores resultados co clasificador XGBoost.

English abstract

In this work two lidar point cloud classification methods were compared in order to identify three species of coniferous trees present in Galicia, P. sylvestris, *P. sylvestris*, *P. pinaster* e *P. radiata*.The first method is based on the manual extraction of descriptive variables of the height distribution to be used as input of a XGBoost classifier. The second method is based on recent graph neural network architectures and allows working directly with xyz coordinates. Both methods were compared in point-to-point and segment classification obtaining better results with the XGBoost classifier.

Capítulo 1

Introducción

Nas explotacions forestais é indispensable ter un coñecemento actualizado dos recursos presentes a través dos inventarios. A elaboración de estes inventarios é un proceso realizado de forma preferentemente manual, cos custos monetários e temporais que isto conleva [Graham, 2008]. Galicia é unha rexión na que as explotacións forestais teñen unha gran importáncia económica, pero presenta un particionamento elevado das parcelas forestais, conducindo a unha alta heteroxenidade que dificulta a creación manual de inventarios. Este e un problema que non só afecta a Galicia senón que é unha problemática comun a gran parte das explotacións forestais europeas [Alonso et al., 2020b].

Os altos custos da métodoloxia tradicional e os avances tecnolóxicos en sistemas de detección remota levan propiciando nas últimas duas décadas a busca de novos métodos que fagan uso de estas tecnoloxias para minimizar os custos e optimizar o proceso de creación de inventarios. Entre estas tecnoloxias de detección remota atopase o LiDAR. Este tipo de datos son recollidos por un sensor colocado na cara inferior de avións ou outro tipo de instrumento de voo de baixa altura. Mediante GPS e un medidor de inercia IMU é posible rexistrar a localización e altura simultaneamente. Dito sensor emite pulsos de luz periodicamente contra a superfície da terra e recibe esta sinal reflexada, coñecendo o tempo de retorno e a altura do avión e recuperamos a altura da superficie que reflexou o pulso permitindo obter unha representación da superfice sobrevoada en forma de nube de puntos. Os avances tecnolóxicos propiciaron o descenso nos custos de aplicación de este tipo de técnicas de detección remota, o cal permitiu que se levaran a cabo proxectos cartográficos como o LiDAR-PNOA.

O Instituto Cartográfico Español en conxunto co Plan Nacional de Ortofotografía Aérea levou a cabo entre os anos 2008 e 2014 unha serie de voos empregando tecnoloxia LiDAR. Esto permitiu capturar unha instantánea en forma de nube de puntos tridimensional de toda a superficie de España, incluindo a península, Ceuta, Melilla e as illas Canarias e Baleares. As nubes de puntos obtidas mediante esta tecnoloxía comprenden areas de 2×2 km e presentan unha densidade de 0.5 puntos/m^2. Foi empregado o ETRS89 como sistema xeodésico de referéncia na rexión peninsular e o sistema de co-

ordenadas UTM. Actualmente esta base de datos é gratuita e está accesible na paxina do Instituto Xeográfico Nacional.

Este tipo de nubes de puntos de baixa resolución non permiten detectar características presentes a baixa escala como a forma das follas, os patróns de ramificación ou a presenza de froitos nas árbores, pero sí permiten detectar outras características como poden ser a cobertura vexetal, a densidade da copa, a xeometria das copas ou a presenza de especies arbustivas ou ramas nas zonas cercanas ao chan.

En este proxecto propuxémonos atopar un método de clasificación capaz de identificar as tres especies de piñeiro mais representativas dos montes galegos, a través das características xerais a gran escala que permiten identificar aos montes de estas especies. *Pinus pinaster* ou piñeiro autóctono é unha arbore de mediano tamaño, entre 20 e 35 m, cuxo porte esta caracterizado pola autopoda dando lugar a un tronco descuberto de ramas dende o chan ata o inicio da copa na cima. Esta copa presenta un tamaño reducido en relación a toda a arbore. *Pinus radiata*, tamén unha arbore mediana pero capaz de alcanzar os 45 m, o seu porte xuvenil presenta forma piramidal mentres que na madurez tende a aplanarse, pero sempre mantendo as ramas ao longo do tronco. Por ultimo temos o *Pinus sylvestris* que presenta unhas características intermedias, a sua copa pode alcanzar os 30 metros, de novo presenta forma cónica mentras que na madurez tende a aplanarse e a perder as ramas das zonas baixas do tronco.

En estudos anteriores obtivéronse resultados prometedores na identificación de especies forestais a partir de datos LiDAR. Realizando unha pixelación do terreo e analizando estadísticos descriptivos da distribución de alturas empregando métodos de machine learning clásicos como random forest, gradient boosting ou support vector machines. No artigo [Alonso et al., 2020a] conseguiuse un accuracy do 76 % na identificación de castiñeiros mais foi necesaria información adicional, empregando tamén variables radiométricas tomadas en distintos momentos temporais, o cal aumenta os custos do proceso. Por outra parte o uso de redes neuronais convolucionais en este campo é un tópico comun na literatura cientifica. En [Mizoguchi et al., 2017] foi proposto con éxito un método de clasificación basado en CNNs mais para levar a cabo este método é necesario posuir datos de maior resolución xa que a clasificación se realiza sobre recortes contendo a cada arbore e as nubes de puntos de LiDAR-PNOA non alcanzan unha resolución suficiente como para identificar árbores individualmente.

Os modelos de clasificación basados en CNNs precisan de datos estructurados en unha grella. En este caso parte da información util para realizar a identificación das nosas especies reside na xeometria tridimensional polo que unha transformación a unha imaxe en 2d de resolución fixa implicaria unha perda de información que dada a baixa resolución do LiDAR-PNOA que é necesario evitar.

Ambas métodoloxias teñen inconvintes para traballar con este tipo de datos. Por unha parte a extracción de estadísticos da distribución de alturas non permite escalar adecuadamente se incrementamos o número de clases a identificar, ou se pretendemos identificar obxectos con formas diversas, non unicamente árbores. Asimismo a conversión da imaxe tridimensional a duas dimensions para usar técnicas de clasificación de imaxes basadas en filtros convolucionais, non permite reter características as descriptivas das especies debido a baixa resolución das nubes de puntos de esta base de datos, polo que tampouco e un método axeitado.

Coa finalidade de traballar con datos non tabulares, como ocorre no procesamento de textos, analise molecular, visión computerizada ou analise de redes sociais; xorden as graph neural networks (GNN). Esta nova arquitectura de rede e capaz de traballar sobre datos estructurados en grafos cunha estratexia basada na difusión de información. O grafo é procesado mediante unha serie de unidades, cada unidade de procesamento correspondese con un nodo. Estas unidades actualizan o seu estado intercambiando información coa rede ata atopar un equilibro. Se ben esta arquitectura foi deseñada para traballar con datos estructurados como grafos de fora natural podemos convertir as nubes de puntos en grafos de forma heurística empregando a distancia euclídea entre puntos como critério de unión mediante algoritmos como o *knn* ou o grafo de gabriel [Gabriel and Sokal, 1969]

Neste traballo propoñemos dous modelos de classificación para afrontar este problema. Un clasificador XGboost seguindo o método proposto en [Alonso et al., 2020a] basado na extracción manual de features cunha modificación que permite realizar a clasificación punto a punto e un clasificador basado nas recentes arquitecturas GNN, que permite evitar o paso previo de extracción de features, o cal podería permitir que o modelo escale mais facilmente a problemas mais complexos.

Capítulo 2

Graph Neural Networks

2.1. Antecedentes

Tras o éxito das redes neuronais convolucionais en problemas de clasificación de imaxes ou segmentación semántica de imaxes, onde a información se atopa representada en estructuras regulares tipo grella, houbo números intentos de trasladar estas técnicas a casos nos que a información non esta representada de esta forma, como e o caso dos grafos. Para chegar a orixe de este tipo de arquitectura foron necesarios certos avances en este tipo de técnicas.

Estas novas técnicas foron propostas inicialmente no campo da transducción de secuencias, problemas típicos de este campo seria o modelado de linguaxe o a traducción automática. A hora de traducir, por exemplo, partimos de unha secuencia de entrada en un idioma a cal transformamos en unha secuencia de saída en outro idioma. Este tipo de problemas son enfocados mediante unha arquitectura de codificación e decodificación, na cal a información contida nas palabras e traducida a vectores que son decodificados de novo en palabras. Unha primeira aproximación en esta dirección foron as redes neuronais recursivas (RNN)[Scarselli et al., 2009]. As RNN son capaces de traballar con estructuras xerárquicas como os grafos acíclicos, arbores ou cadeas lineais. Estas redes actuan combinando de forma xerárquica as representacións fillas coas representacións pais.

Por outra banda as cadeas de Markov son capaces de emular procesos nos cales as conexións causais entre elementos son representadas mediante grafos. En concreto, Os camiños aleatorios foron empregados con exito para na creación de rankings en resultados web segundo a sua importancia relativa frente a unha búsqueda.

As GNN propostas en [Scarselli et al., 2009] xorden como unha extensión de estas dúas métodoloxias, RNN e camiños aleatorios. Esta arquitectura e unha extensión das redes recursivas xa que e capaz de traballar sobre unha clase mais xeral de grafos incluindo grafos cíclicos, acíclicos, dirixidos en non dirixidos en tarefas como a predicción

a nivel nodo ou a nivel grafo. Por outra parte e unha ampliación dos camiños aleatorios xa que introduce o aprendizaxe automático e amplia a clase de procesos que e capaz de modelar.

2.2. Arquitectura base

O modelo base sobre o que están implementadas todas as variantes empregadas en este traballo emprega un método de difusión da información contida no grafo a través da sua conectividade. Esto posibilita que, a hora de facer predicciónss sobre un nodo, teñamos información sobre as suas veciñanzas ou se buscamos predicciónss a nivel grafo, empregemos toda a información propagada a través del.

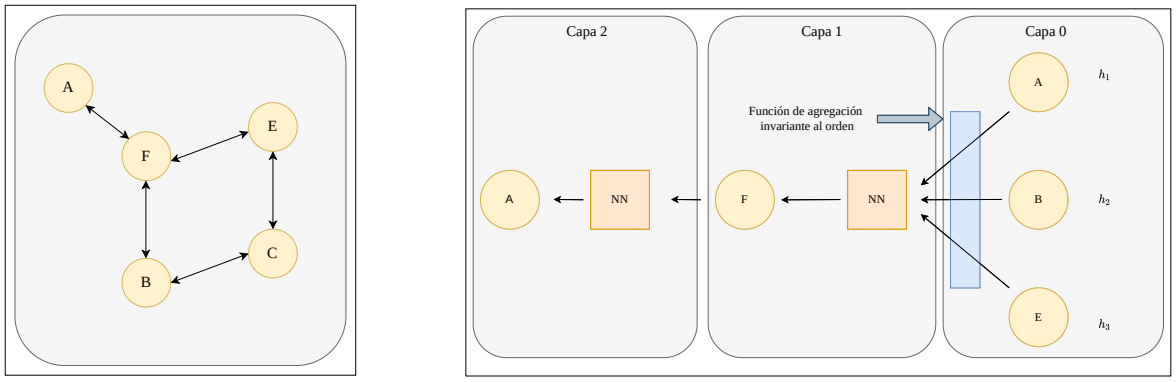

Figura 2.1: Propagación de mensaxes: (a) Grafo de exemplo; (b) Grafo computacional

Este mecanismo denominase paso de mensaxes. A continuación descríbese unha iteración de este modelo 2.1. Partimos de un grafo con N nodos e E arcos, cada nodo esta dotado de un vector de atributos $h = \{\overrightarrow{h}_1, \overrightarrow{h}_2, \dots, \overrightarrow{h}_N\}$ $h_i \in \mathbb{R}^F$ onde F é o tamaño de cada vector de atributos. Para facer predicciónss sobre un nodo i esta arquitectura toma os vectores $\overrightarrow{h}_n$ dos nodos veciños e agrégaos mediante unha función invariante a orde, aplica unha transformación lineal adestrable W e pasa a nova representación a través de unha función de activación σ 2.1. Apilando capas con esta arquitectura consíguese propagar información contida en nodos de veciñanzas lonxanas, asimismo, o tamaño da matriz de pesos W coa que se realiza a transformación lineal permite variar a dimensión de ditas features.

$$\overrightarrow{h}'_i = \sigma(W \times Aggr(\overrightarrow{h}_j, \dots, \overrightarrow{h}_n)) \tag{2.1}$$

Mediante esta arquitectura conseguimos empregar a conectividade do grafo para obter features de alto nivel que inclúen información de rexións distantes no grafo. A

continuación pasamos a describir as modificacións de esta arquitectura básica que deron lugar a arquitectura das capas empregada na rede neuronal deste traballo. Os dous tipos de arquitectura empregados son a GAT (Graph Attention Network) proposta en [Veličković et al., 2017], que emprega mecanismos de atención para incluir na propagación de mensaxes as importancias relativas dentro de cada veciñanza mediante o uso de pesos adestrables na fase de agregación de mensaxes. A outra arquitectura empregada, denominada GraphSAGE[Hamilton et al., 2017] (Sample Aggregation), foi deseñada para a xeración de embeddings de forma inductiva, é decir, representacions vectoriais do grafo, normalmente asociadas aos nodos, que permiten reter a topoloxía do grafo en unha dimensión potencialmente menor. O termo inductivo fai referencia a que unha vez adestrado o método de xeración de embeddings este pode actuar sobre novos nodos do grafo ou en grafos completamente novos

2.3. Graph Attention Network

2.3.1. Mecanismos de atención

Partindo das ideas de atención nas redes neuronais e das RNN surxe o transformador, este tipo de arquitectura supuxo un punto de inflexión no mundo do aprendizaxe profundo [Vaswani et al., 2017] xa que permite crear representacións das secuencias secuencias de entrada e saída sen empregar a recursividade ou convolucions. En unha rede recurrente sen mecanismos de atención cando buscamos facer predicción sobre un elemento a rede fai uso dos elementos contiguos a posición de interés dotándoos da mesma importancia. Ao empregar un mecanismo de atención a rede aprende tamén a analizar o resto de elementos que teñen algun tipo de relación co noso obxectivo e asignarlles a súa importancia relativa, no contexto dos grafos as relacións entre elementos, ou nodos, están definidas pola conectividade do grafo.

Unha forma intuitiva de describir estos mecanismos de atención atención seria unha capacidade de dar maior relevancia ou atender mais aos elementos mais imortantes a hora de facer inferencia sobre un determinado aspecto, sendo este mecanismo de atención unha función adestrable 2.2. No contexto das GNNs os mecanismos de atención aplicanse na fase de agregación das features na veciñanza, empregando a importancia relativa para obter coeficienetes ou pesos cos que se pondera dita agregación.

2.3.2. Modelo

A continuación descríbense as compoñentes e o funciónamento da capa atencional para grafos descrita en [Veličković et al., 2017] e a modificación proposta en [Brody et al., 2021] que a dota de atención dinámica. A entrada da nosa capa esta formada polo conxunto

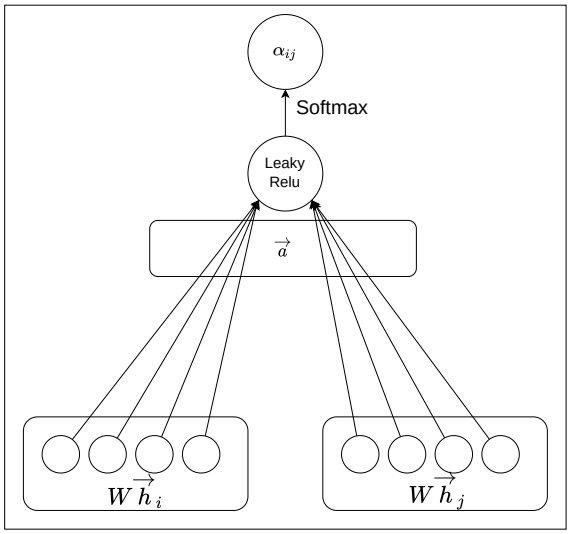

Figura 2.2: Mecanismo de atención

de características dos nodos $h = \{\overrightarrow{h}_1, \overrightarrow{h}_2, \ldots, \overrightarrow{h}_N\}$ $h_i \in \mathbb{R}^F$ onde N é o número de nodos e F o tamaño do vector de características dos nodos. Esta capa simple permite obter unha nova representación $h' = \{\overrightarrow{h'}_1, \overrightarrow{h'}_2, \ldots, \overrightarrow{h'}_N\}$ $h'_i \in \mathbb{R}^{F'}$ tamaño de F' potencialmente diferente a F.

Para poder acadar unha transformación das características de entrada nun novo conxunto de características con maior nivel de abstracción, é preciso realizar unha transformación lineal que poida ser adestrada. Con esta intención aplícase unha transformación lineal sobre as características do nodo e dos seus veciños de xeito compartido en todo o grafo. Esta transformación está parametrizada mediante unha matriz de pesos $W \in \mathbb{R}^{F'} \times \mathbb{R}^F$. Posteriormente aplícase un mecanismo de atención compartido $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \longrightarrow \mathbb{R}$ que se encarga de computar as importancias, ou coeficientes de atención na veciñanza. Este mecanismo de atención esta formado por un vector de pesos adestrable $\overrightarrow{a}$ con dimensión $R^{2F'}$, de forma que ao realizar o producto escalar entre este vector e as features concatenadas $[W\overrightarrow{h}_i || W\overrightarrow{h}_j]$ obtemos un escalar e_{ij}, que será o coeficiente de atención entre os nodos i e j

$$e_{ij} = a(W\overrightarrow{h}_i, W\overrightarrow{h}_j) \tag{2.2}$$

Estos coeficientes indican a importancia das características do nodo j sobre as características do nodo i. Si ben estos coeficientes poden ser obtidos para cada par de nodos no grafo, en esta implementación so se calcularon os coeficientes para o primeiro entorno de veciñanza. Para reescalar estos coeficientes entre os nodos veciños j aplicamos unha función de normalización softmax 2.3.

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in N_i} exp(e_{ik})} \tag{2.3}$$

Finalmente esta capa obtén unha media ponderada das características transformadas de cada nodo veciño seguida de función de activación non lineal σ a cal será unha nova representación de do nodo i tendo en conta as importancias aportadas polos coeficientes de atención normalizados 2.4.

$$\overrightarrow{h}'_i = \sigma(\sum_{j \in N_i} \alpha_{ij} \cdot W \overrightarrow{h}_j) \tag{2.4}$$

Inicialmente o mecanismo de atención proposto por [Veličković et al., 2017] carecia da capacidade de adaptarse a cada entorno, computando únicamente unha atención estática ao aplicar consecutivamente as transformacións lineais por W e a poden ser colapsadas en unha unico paso. No paper [Brody et al., 2021] propoñen unha modificación na orde de operacións que permite aumentar a capacidade de atención de esta rede. Esta modificación consiste en aplicar a transformación por W tras a concatenación dos vectores de features $\overrightarrow{h}_i$ e $\overrightarrow{h}_j$ e aplicar a transformación mediante $\overrightarrow{a}$ tras a función de activación LeakyReLU 3.15. A continuación podemos observar o mecanismo orixinal 2.6 e a modificación para realizar atención dinámica 2.6

$$e(\overrightarrow{h}_i, \overrightarrow{h_j}) = LeakyReLU(a^T \cdot [W \overrightarrow{h}_i || W \overrightarrow{h}_j]) \tag{2.5}$$

$$e(\overrightarrow{h}_i, \overrightarrow{h}_j) = a^T LeakyReLU(W \cdot i || \overrightarrow{h}_j]) \tag{2.6}$$

Mediante esta modificación a rede e capaz de axustarse a problemas de maior complexidade. Ademais, a atención dinámica dota a esta arquitectura de unha maior robustez ao ruido nos arcos, sendo capaz de discriminar arcos espúreos ou non relevantes. Esta modificación mellorou os resultados acadados pola versión inicial da GAT en todas as probas realizadas, e decir, é estrictaménte mellor, polo que esta será a versión empregada neste traballo.

2.4. Graph SAGE

A Graph SAGE, acrónimo de SAmple and aggreGatE, e unha arquitectura proposta en [Hamilton et al., 2017] e deseñada para obter embeddings de forma inductiva, e decir, poder adestrar transformacións de grafos capaces de traballar con nodos ou grafos completos novos. E unha arquitectura deseñada no campo dos sistemas de recomendación en redes sociais, onde as relacións están contidas en grafos de gran tamaño que cambian constantemente, polo cal, e un problema re adestrar o modelo cada vez que se

introduce un cambio.

A continuación móstrase o algoritmo de xeración de estos embeddings supoñendo que a rede foi previamente adestrada 1. De forma intuitiva para obter a representación da na capa i de esta rede toma cada nodo do grafo e samplea un número N de nodos veciños, agrega os seus vectores de features e, tras concatenar coa representación actual do nodo, aplica unha tansformación lineal mediante unha matriz de pesos W. Tras a transformación lineal aplícase unha función de activación non lineal. Finalmente esta representación divídese pola sua norma mapeando os embeddings a unha hiperesfera de radio 1 e mesma dimensión que os embeddings para normalizalos. Tras sucesivas capas a información propagase a través das veciñanzas do grafo.

Algorithm 1 Xeración de embeddings mediante GraphSAGE

Input: Grafo $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\overrightarrow{x}_v, \forall v \in \mathcal{V}\}$; numero de veciñanzas no que se expande a agregación K; matriz de pesos $\mathbf{W}_k, \forall k \in \{1, \dots, K\}$; función de activación σ; función de agregación diferenciable $AGGREGATE_k, \forall k \in \{1, \dots, K\}$; función de veciñanza $\mathcal{N} : v \longrightarrow 2^{\mathcal{V}}$
Output: Vector de representacións $\overrightarrow{z}_v \forall v \in \mathcal{V}$
$\overrightarrow{h}_v^0 \longleftarrow \overrightarrow{x}_v, \forall v \in \mathcal{V}$;
for $k = 1, \dots, K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\overrightarrow{h}_{\mathcal{N}(v)}^k \longleftarrow AGGREGATE_k(\{\overrightarrow{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\overrightarrow{h}_v^k \longleftarrow \sigma(\mathbf{W}_k \cdot CONCAT(\overrightarrow{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$
  end for
  $\overrightarrow{h}_v^k \longleftarrow \overrightarrow{h}_v^k / ||\overrightarrow{h}_v^k||_2, \forall v \in \mathcal{V}$
end for
$\overrightarrow{z}_v \longleftarrow \overrightarrow{h}_v^K, \forall v \in \mathcal{V}$

Este e o algoritmo descrito no artigo [Hamilton et al., 2017]. Na fase de agregación eles propoñen empregar unha función de mostraxe uniforme dentro de cada veciñanza para reducir o custo computacional, xa que en nodos con un alto grado de conectividade o grafo computacional pode ser moi grande. Como no noso problema traballamos con grafos creados de forma heurística, ben por **knn** ou mediante a triangulación de Delunay [Gabriel and Sokal, 1969], a conectividade nos nosos grafos e coñecida e relativamente uniforme polo que non é problemático realizar a agregación na veciñanza completa. Como operador de agregación empregamos o max pooling tras a transformación lineal da veciñanza, de este modo a agregación é adestrable de forma conxunta coa transformación lineal. No artigo [Hamilton et al., 2017] propoñen outros métodos de agregación como a media ou LSTM (Long-short-term-memory) pero o max pooling demostrou dar mellores resultados de forma computacionalemnte eficiente.

Capítulo 3

Métodos

3.1. Procesado dos datos

3.1.1. Datos de adestramento

Os datos que imos a empregar proveñen da base de datos LiDAR-PNOA. Orixinalmente os arquivos empregados son nubes de puntos tridimensionais cunha densidade de 0,5 puntos/m^2 e cunha extensión de 2×2 km. Con esta información construíronse tres conxuntos de datos contendo as nubes de puntos das parcelas identificadas como *P. sylvestris*, *P. pinaster* e *P. radiata* na zona de Lourizán (Pontevedra). Forman un total de 86 parcelas correspondendo 23 a *P. sylvestris*, 25 a *P. pinaster* e 38 a *P. radiata*. Encanto ao número de puntos, temos un total de 91171 pertencendo a cada especie 22653, 27363 e 41155 respectivamente.

Estas nubes de puntos estan formadas polas coordenadas xyz. As nubes de puntos foron previamente normalizadas, é decir, identificouse o chan e transformáronse as alturas de forma que o chan se situase a altura cero para todas as rexións, de esta forma elimínase o efecto da orografía sobre as alturas.

Para poder comparar de forma xusta as duas metodoloxías de clasificación que empregamos, realizamos unha etapa inicial do procesamento en comun que nos permite replicar as particións de adestramento e validación de forma idéntica en ambos casos. A diferencia radica nos pasos posteriores. No caso do XGboost realizamos unha extracción de features de forma manual en un entorno cilíndrico a cada punto e no caso das GNNS creamos unha estructura de grafo que conten a información do entorno a cada punto para que sexa a rede a que se encargue a rede da extracción de features.

A etapa inicial do procesamento e a seguinte:

- Filtrar copas: Eliminar todos os puntos cunha altura inferior a 0,25m.

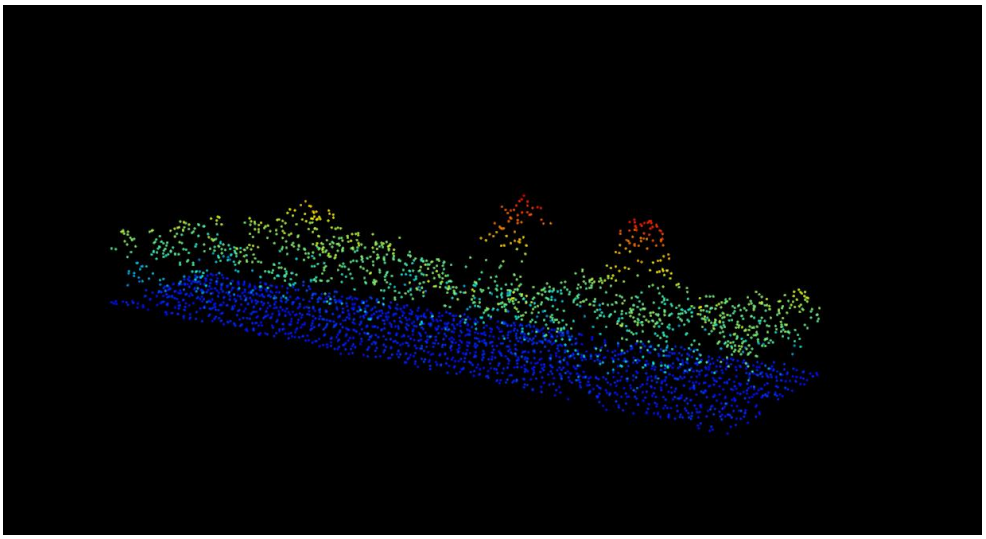

Figura 3.1: Parcela

- Segmentación: Para cada especie separar en distintos conxuntos de datos os puntos corespondentes a cada unha das parcelas. Segmentar cada parcela en fragmentos empregando unha grella de 8m de lado e eliminar os fragmentos con menos de 30 puntos.

- Matriz de distancias: Obtención da matriz de distancias de cada un de estos fragmentos.

O paso de segmentación e necesario por dous motivos. Por unha parte permite homoxenizar os datos de adestramento e validación facendo que todos os fragmentos teñan o mesmo tamaño xa que as parcelas orixinais son de tamaños moi diferentes. Por outra banda ao dispor de un conxunto de datos relativamente pequeno se creamos as particións de adestramento e validación empregando as parcelas orixinais e moi probable que características que estén presentes en unha ou poucas parcelas queden incluidas unicamente en adestramento ou validación dificultando a xeralización.

3.1.2. Extracción de features - XGBoost

A extracción de estadísticos descriptivos da distribución de alturas foi o método empregado en [Alonso et al., 2020b] para a identificación de plantacións de castiñeiros. Estos estadísticos permiten caracterizar as os montes compostos por cada unha das tres especies a partir da distribución de alturas observadas nas nubes de puntos. A distribucion de alturas de cada especie pode observarse no anexo 6.1. Se ben no artigo empregaron a maiores variables radiométricas obtidas mediante satélite en distintas estacións do ano, en este traballo imos empregar unicamente as variables relacionadas

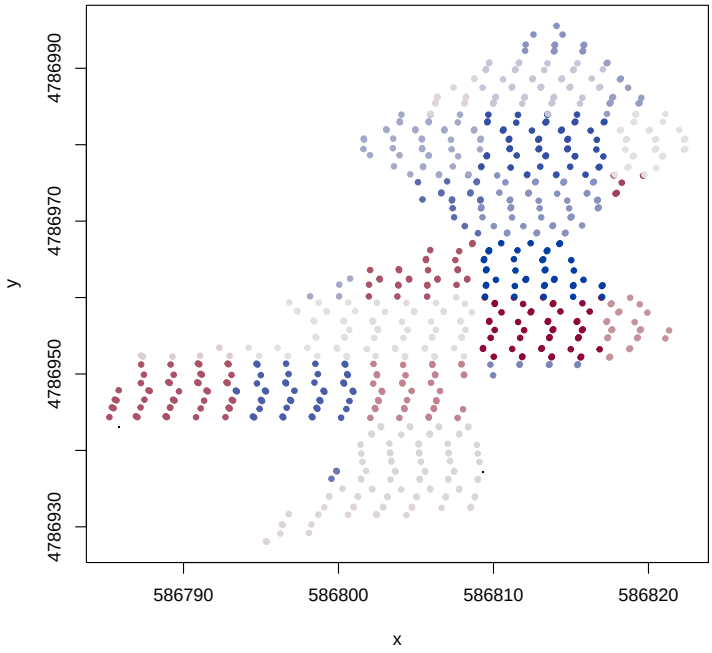

Figura 3.2: Segmentación de unha parcela en fragmentos de $8m^2$

coas coordenadas xyz da nube de puntos.

Para adaptar a extracción de features ao obxectivo de clasifcar os datos punto a punto definimos un entorno no que extraer ditas features a partir de cada punto. A estratexia empregada foi definir un entorno *knn* a cada punto no plano horizontal. Decidimos empregar un entorno *knn* en vez de un entorno cilíndrico xa que de esta forma o número de puntos que conforma o entorno e constante independentemente da posición do puno que estemos a analizar. De esta forma evitamos que se vexan afectadas as features dos bordes das parcelas polas zonas vacías. Na figura 3.3 podemos ver como se comportan un entorno *knn* e un entorno cilíndrico de tamaños equivalentes cando son empregados en rexións centrais e no borde da parcela.

Os estadísticos descriptivos que empregamos son os seguintes:

- Altura máxima: Altura máxima rexistrada no entorno de cada punto.
- Altura media: Media das alturas rexistradas no entorno de cada punto.
- Desviación estándar das alturas: Desviación estándar das alturas rexistradas no entorno de cada punto.

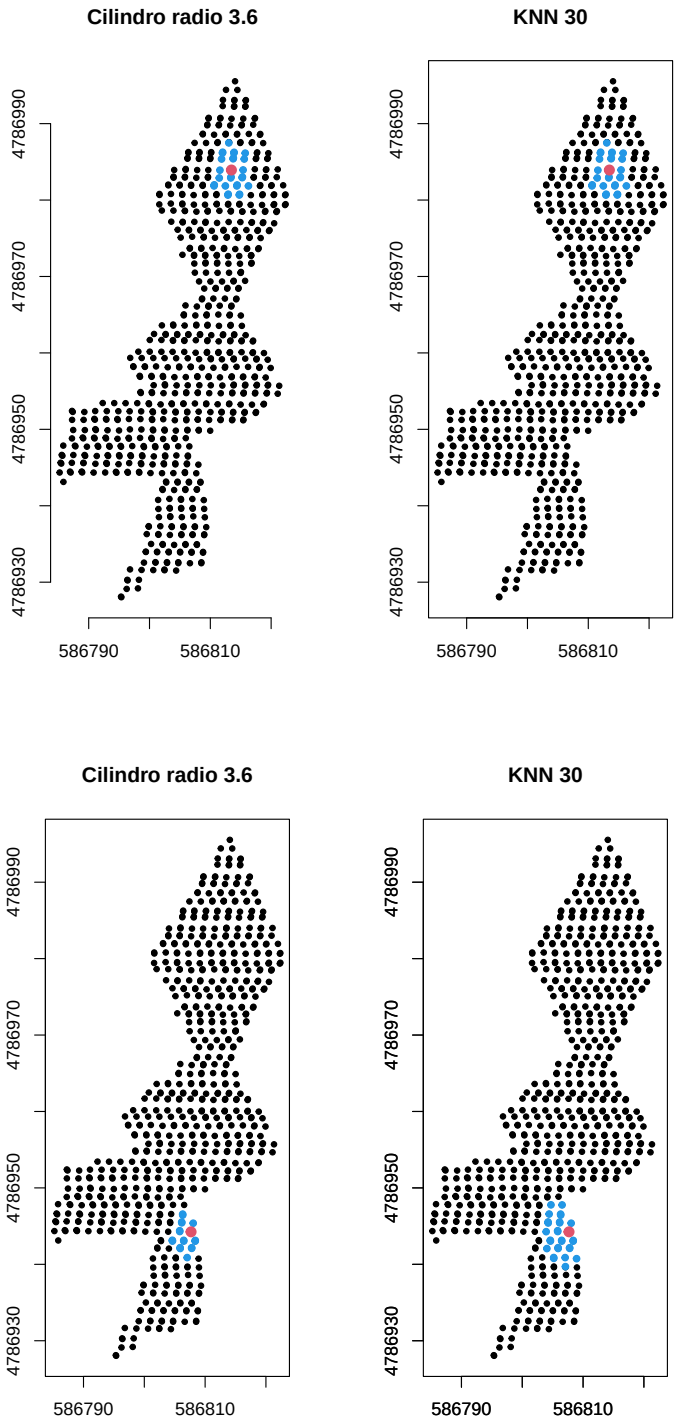

Figura 3.3: Comparativa entorno *knn* vs entorno cilíndrico

- Cuantís 1º, 2º, 5º e 9º: Alturas nas que se acumulan as probabilidades 0.1, 0.2, 0.5, 0.9 dentro do entorno de cada punto.
- Altura base da copa: Altura mínima rexistrada por enriba da altura do diámetro base (1.37m).
- Altura media da copa. Altura media dos puntos situados por encima de 1.37 m no entorno de todos os puntos.
- Desviación estándar das alturas da copa. Desviacion estándar computada sobre as alturas situadas por enriba de 1.37 m no entorno entornos de cada punto.
- Cobertura de copa: Ratio de puntos situados sobre 1.37 m frente a todos os punto rexistrados no entorno para cada punto.
- Kurtosis da copa: Kurtosis avaliada nos puntos situados por encima de 1.37 m no entorno de cada punto.
- Asimetria da copa: Asimetria avaliada nos puntos situados por encima de 1.37 m no entorno de cada punto.
- Densidade arbustiva: número de puntos situados entre 0.5 e 2 m frente a todos os puntos situados por debaixo de 2 m no entorno de cada punto.

Dado o gran volume de datos e en previsión a posibilidade de escalar este método de clasificación a parcelas de maior tamaño decidiuse implementar en c++ as funcións necesarias para a extracción de características a partir das alturas. Mediante o paquete Rcpp e posible facer interfaz dende R a este código en c++. De esta forma podemos ler os ficheiros .las dende R e realizar a extración de características de forma interactiva na propia sesión de R pero coa ventaxa computacional que nos aporta c++.

3.1.3. Construcción de grafos - GNN

Para poder adestrar unha rede con unha arquitectura GNN é preciso contar con un input en forma de grafo, polo cal necesitamos transformar a información contida nas nubes de puntos en grafos. Os grafos cos que imos a traballar poden ser definidos da seguinte forma:

Un grafo dirixido $\mathcal{G} = (\nu, \varepsilon)$ conten os vertices $\nu = \{1, \dots, n\}$ e aristas ou arcos $\varepsilon \subseteq \nu \times \nu$, onde $(i, j) \in \varepsilon$ denota un arco dende o nodo i ao nodo j. Asúmese que cada nodo $i \in \nu$ ten unha representación inicial $\overrightarrow{h}_i^{(0)} \in \mathbb{R}^{d_0}$. Como en este caso o grafo que define a nosa nube de puntos non e dirixido, para cada par de nodos unidos empregamos un arco bidirecional.

Para definir a conectividade do grafo empregamos dous métodos distintos, ambos basados na distancia euclídea entre puntos:

- k veciños mais próximos (knn). Cada punto unese de forma bidireccional aos seus k puntos mais proximos. Este método e dependente de k, polo cal probamos a xerar os grafos con $k = \{3, 6, 8\}$.

- Grafo de Gabriel. É un grafo non paramétrico plano. Para crear este grafo avaliase entre cada par de puntos se podemos crear un disco vacío entre eles (esfera ou hiperesfera en dimensions superiores) cuxo diámetro é a distancia que une a ditos puntos, e se é posible, trazase o arco.

Se ben en etapas iniciais de este traballo empregamos a distancia nas tres dimensións para a creación dos grafos, vimos que empregando unicamente a distancia no plano horizontal os resultados melloraron considerablemente polo que só se incluiron os resultados obtidos mediante este segundo método.

O nivel de conectividade pode ser un aspecto relevante para a eficacia da rede xa que o método de propagación de mensaxes depende directaménte da conectividade. Por outra parte o mecanismo de atención da rede demostrou no artigo [Brody et al., 2021] que e capaz de ignorar arcos ruidosos ou irelevantes.

Ainda que a ausencia de conectividade parece ser problemática, xa que limita o paso de mensaxes a través da rede, o exceso de conectividade poderia non ser un problema. Este tipo de arquitectura, grazas ao mecanismo de atención dinámica, permite discriminar segundo a importancia dos arcos polo que a presenza de arcos innecesarios non ten por que dificultar o proceso de clasificación.

Unha vez definida a conectividade do grafo e necesario dotar aos arcos e nodos de propiedades que os definan. Como cada nodo se corresponde con un punto da nube orixinal, a cada nodo asignaselle a altura do punto ao que corresponde. No caso dos arcos hai mais información dispoñible, a cada arco asignamoslle un vector con catro variables, a distancia euclídea en xyz, a distancia euclídea en xy. a distancia euclídea en z e o coseno do ángulo que forma o arco coa horizontal.

Chegado a este punto temos o conxunto de datos segmentado en fragmentos de tamaño homoxéneo e un grafo definido para cada un de estes fragmentos. De forma análoga a como fixemos no caso anterior, para realizar a clasificación punto a punto creamos un subgrafo a a partir de cada un dos nodos. Este subgrafo esta composto polo nodo central mais os puntos das suas V veciñanzas.

3.2. Modelos de clasificación

A continuación descríbense os dous métodos que empregamos para resolver este problema de clasificación. O modelo XGboost adestrado sobre a extracción manual de estadísticos descriptivos das alturas e a rede neuronal basada en grafos capaz de extraer features de forma automática reducindo a importancia da etapa de preprocesado de datos.

3.2.1. Clasificador XGBoost

O nome XGBoost fai referencia a Extreme gradient boosting. É unha variante mais eficiente dos modelos de gradient boosting proposta por [Chen and Guestrin, 2016]. A metodoloxía XGboost entra dentro dos modelos de ensemble onde se usan distintos predictores de forma aditiva. A diferencia dos modelos de bagging onde cada modelo e adestrado paralelamente e de forma independente aos demais, a metodoloxía de boosting trata de adestrar secuencialemnte ditos modelos, cada un condicionado polos errores de predicción do modelo anterior. Os predictores empregados en esta metodoloxía adoitan ser predictores débiles, é decir, predictores moi sinxelos con sesgo elevado xa que a capacidade predictiva surxe da suma de ditos modelos. O termo gradient fai referencia a que cada predictor debil é creado minimizando o valor de unha función de pérdida, e decir, seguindo o gradiente negativo da función de pérdida empregada.

Se partimos de un modelo cunha arquitectura fixa o método de optimización por descenso de gradiente busca os parámetros P de dito modelo que minimizan a función de pérdida que estemos a empregar. Se introducimos na búsqueda o propio modelo F a búsqueda pode expresarse así:

$$F(x|P) = min_{F,P} Loss(y, F(x|P)) \tag{3.1}$$

Gradiet boosting busca o modelo óptimo cos parámetros óptimos en esta función de pérdida. A búsqueda por forza bruta en este espazo e inpracticable na maioria dos casos. O algoritmo de gradient boosting actua de forma aditiva axustando modelos simples de forma secuencial. En unha etapa inicial axustariamos un modelo h_0 a un conxunto (x, y).

$$F(x|P) = min_{P} Loss(y, F(x|P)) \tag{3.2}$$

O axuste dos seguintes modelos en vez de realizarse de forma paralela a este, axustaranse sobre o gradiente do erro respecto as predicciónss $\partial Loss/\partial \hat{y}$. Neste caso, empregando a pérdida de entropía cruzada 3.3 con $i, ..., k \in K$ clases (Cross Entropy Loss) e normalizando os pesos $\hat{y}_i$ mediante unha función $softmax$ 3.4 para convertilos en probabilidades como se mostra a continuación:

$$L_{CE} = \sum_{i=1}^{n} y_i log(\hat{y}_i) \tag{3.3}$$

$$p_i = \frac{e^{\hat{y}_i}}{\sum_{j=1}^{K} e^{\hat{y}_j}} \tag{3.4}$$

Unha vez definida a función de pérdida e o seu gradiente, para cada modelo axustado secuencialemente obteremos o gradiente da función de pérdida en cada dato para no seguinte modelo facer un axuste a estes mesmos gradientes empleando as mesmas features. O modelo h1 axustase minimizando a derivada negativa da sua pérdida frente a derivada do modelo anterior 3.5.

$$F_i(x) = -\frac{\partial L_{CE}}{\partial F_{i-1}} \tag{3.5}$$

A predicción de este modelo na etapa i viria dada por 3.6. Sendo η o hiperparámetro encargado de controlar a importancia de cada etapa na predicción final.

$$\hat{y}_i = F_{i-1} + \eta \cdot F_i \tag{3.6}$$

Segundo [Chen and Guestrin, 2016] definindo o modelo de ensemble como 3.7, para un conxunto de datos con n individuos e m variables $\mathcal{D} = \{(x_i, y_i)\}$ $(|\mathcal{D}| = n, x_i \in \mathbb{R}^m, y_i \in \mathbb{R}$ un modelo ensemble usará K funcións de forma aditiva para realizar predicions.

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} F_k(x_i), F_k \in \mathcal{F} \tag{3.7}$$

Onde $\mathcal{F} = \{F(x) = w_{q(x)}\}(q : \mathbb{R}^m \longrightarrow T, w \in \mathbb{R}^T)$ e o espacio das arbores de regresión. q Representa a estructura de cada arbore, mapeando un exemplo ao índice de unha folla. T é o número de follas da arbore. Cada F_k corresponde a unha arbore q con pesos w nas follas diferente. A diferencia das arbores de decisión, para obter a clasificación a partir das arbores CART empregamos a suma dos pesos obtidos nos nodos terminais para cada dato. Como estes pesos w toman valores continuos podemos aplicar regularización a hora de optimizar ditos modelos. A función obxectivo a optimizar incluido o termo de regularización toma a forma 3.8. Onde o segundo termo Ω penaliza a complexidade do modelo. De forma intuitiva o obxectivo regularizado tendera a seleccionar modelos mais simples.

No noso caso os predictores debiles que axustamos en cada iteración son arbores de regresión CART, en vez de determinar unha clase en cada folla temos un peso w_k

continuo asígnado a cada clase de forma que a clasificación final se realiza mediante a suma de ditos pesos en todos os modelos. Empregar pesos nos nodos terminais permite incluir regularización na función obxectivo final 3.8. O segundo termo Ω e o termo de regularización, penaliza a complexidade do mediante os hiperparámetros γ e λ que actuan sobre a complexidade do modelo e a magnitude dos pesos respectivamente.

$$\mathcal{L}(\phi) = \sum_i L_{CE}(\hat{y}_i, y_i) + \sum_k \Omega(F_k) \tag{3.8}$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

A función obxectivo 3.8 inclúe funcións como parámetros polo que non e posible optimizala mediante métodos do espacio euclídeo. No seu lugar e preciso adestrar o modelo de forma aditiva. sendo f_t o modelo na etapa t e $\hat{y}_i^t$ a predicción da instancia i en na etapa t podemos incluir o modelo f_t na seguinte función obxectivo a minimizar.

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} L_{CE}(y_i, \hat{y}_i^{(t-i)} + F_t(x_i)) + \Omega(F_t) \tag{3.9}$$

De esta forma seleccionaremos o modelo F_t que minimiza a pérdida de forma voraz. Para optimizar dita función de forma xeral empregaremos as derivadas de primeiro e segundo orde da función de pérdida $g_i = \partial_{\hat{y}^{(t-1)}} L_{CE}(y_i, \hat{y}^{(t-1)})$ e $h_i = \partial^2_{\hat{y}^{(t-1)}} L_{CE}(y_i, \hat{y}^{(t-1)})$ obtendo:

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [L_{CE}(y_i, \hat{y}^{(t-1)}) + g_i F_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{3.10}$$

Podemos eliminar as constantes para obter a seguinte versión simplificada da función obxectivo na etapa t.

$$\bar{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{3.11}$$

Para seleccionar os modelos candidatos en cada etapa definimos $I_j = \{i | q(x_i) = j\}$ como o conxunto das instancias do nodo j. Para unha estructura de arbore fixada $q(x_i)$ podemos obter o peso óptimo w_i^* na folla j como:

$$w_i^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{3.12}$$

e obter o seu valor óptimo minimizando:

$$\bar{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma\lambda \tag{3.13}$$

Esta última expresión pode ser empregada para avaliar a calidade de cada arbore q. Como enumerar cada arbore posible e avaliala mediante esta función e unha labor imposible na maioria dos casos, empregase un algoritmo voraz que fai crecer a arbore división a división seleccionando en cada etapa a división que minimiza a pérdida. Se $I_L e I_R$ son os conxuntos de nodos ubicados nas ramas esquerda e dereita tras unha división, consideramos $I = I_L \cup I_R$ dando lugar a unha reducción da pérdida tras a división igual a:

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \tag{3.14}$$

En vez de atacar a este problema de forma voraz avaliando todos os posibles puntos de corte para todas as variables a implementación de XGboost emprega os percentis das distribucións de cada variable como candidatos para os puntos de corte e almacena esta información entre iteracións axilizando a otimización.

Outro método implementado en esta libraría para reducir o sobreaxuste e o submostraxe de variables. De forma similar a o metido de submostraxe empregado nos random forest, no cal se mostrean aleatoriamente grupos de individuos para axustar cada arbore, en este caso mostreanse variables e so se propoñen ramificacións a partir de ditas variables. Este método ademais de impedir o sobreaxuste reduce o tempo computacional ao eliminar aleatoriamente moitos modelos candidatos en cada iteración.

3.2.2. GNN

A rede neuronal deseñada para este traballo está representada na figura 3.4 e consta dos seguintes compoñentes: Unha primeira sección basada en GNNs destinada a obter representacións de alto nivel para os nodos dos grafos de entrada. Esta sección consta de unha primeira capa atencíonal con 16 cabezas, cada cabeza obtén 8 features a partir das alturas, estas features son concatenadas a saída da capa obtendo unha representación de 128 elementos por nodo, seguida de unha capa SAGE que transforma ditas representación mantendo as dimensións. A cabeza esta seguida de duas capas de pooling paralelas. estas capas están destinadas a colapsar as features contidas en cada nodo do grafo en un único vector de features para o grafo. A agregación nunha das capas realizase mediante a media e en outra mediante o máximo. Ambas agregacións son

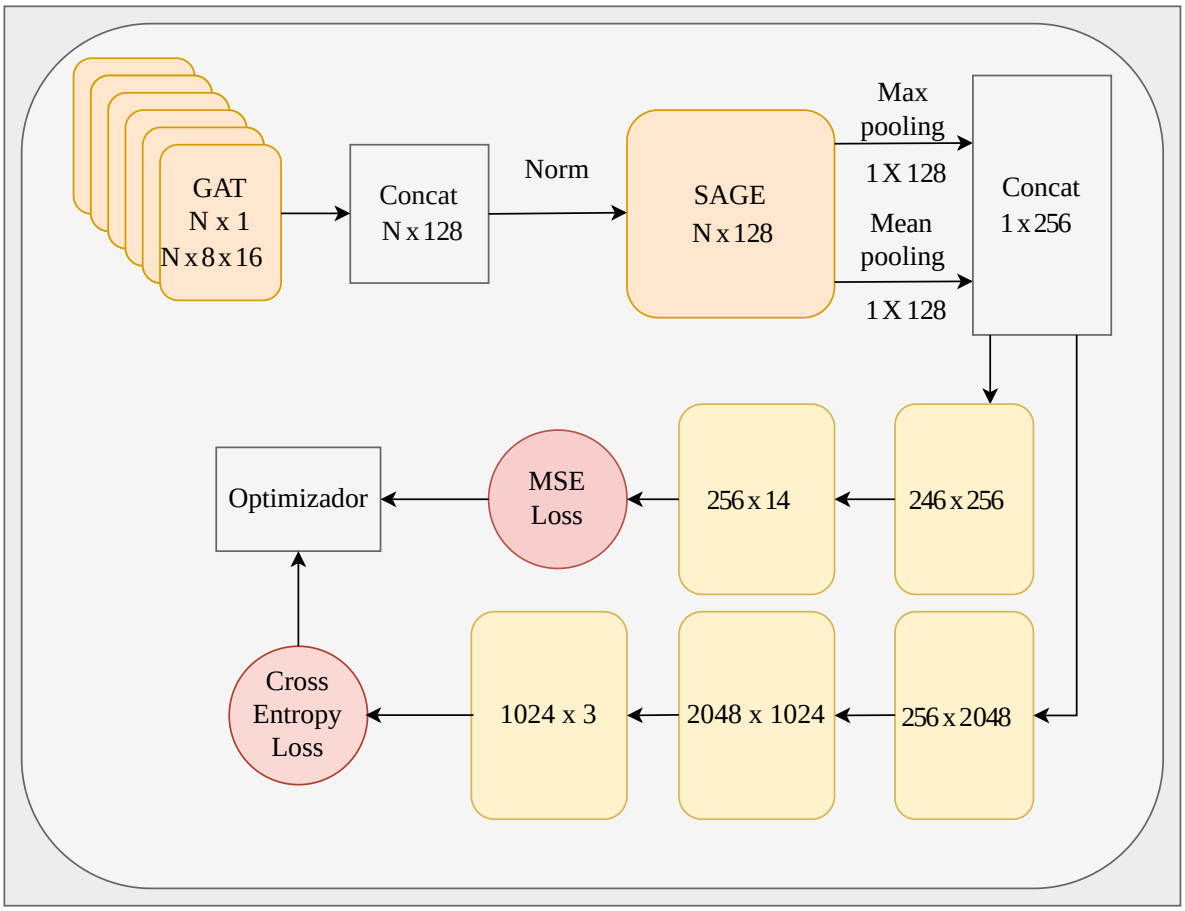

Figura 3.4: Deseño da rede

concatenadas obtendo unha representación de 256 elementos por cada grafo de entrada.

Unha vez obtida a representación final para ao grafo, esta representación pasa paralelamente a duas seccions diferentes da rede. Por unha parte temos un MLP destinado a clasificación que está composto de tres capas 256×2048, 2048×1024 e 1024×3. Paralelamente a hay outro MLP de regresión que so sera empregado na etapa de preadestramento para inicializar os pesos da cabeza da rede. Este MLP tamén consta de duas capas cuxas dimensións son 256×256 e 256×14. Esta capa esta deseñada para facer regresión sobre os estadísticos da distribución de alturas obtidos sobre todo o grafo, de esta forma podemos preadestrar o a rede para que extraia features relacionadas con estes estadísticos facilitando o aprendizaxe da rede. Unha vez adestrada a rede a capa de regresión xa non e necesaria, tampouco e necesario obter ditos estadísticos sobre os novos datos.

A función de activación empregada entre todas as capas da rede foi a Leaky ReLu

3.15. Esta función de activación e unha modificacion da ReLu (Rectified Linear Unit) para evitar a sua morte durante a optimización. A morte dunha ReLu ocorre cando o seu valor de saída se torna cero sexa cal sexa a entrada. Para evitar este problema a leaky ReLu engade unha lixeira pendente $\alpha = 0{,}001$ na rexión negativa para que o gradiente non sexa cero cando traballa con imputs negativos. De esta forma a derivada para valores positivos e 1 e para valores negativos α polo que a obtención dos gradientes e computacionalmente moi sinxela. Ao ter un comportamento lineal para valores positivos facilita tamén a aprendizaxe e optimización, ademáis posibilita facer regresión sobre variables positivas xa que os seus valores de saída non están restrinxidos por enriba de 0.

$$f(x) = \begin{cases} \text{máx}(0, x) & x > 0 \\ \alpha x & x \leq 0 \end{cases} \tag{3.15}$$

Entre todas as capas da rede aplicouse un mecanismo de dropout como método de regularización [Srivastava et al., 2014]. Este mecanismo torna a 0 o valor de saída das neuronas con unha probabilidade p definida e multiplica o valor de saída das neuronas restántes por $\frac{1}{1-p}$ para manter constante a suma dos valores de saída da capa. Este mecanismo esta deseñado para evitar o problema da coadaptación, e decir, evita que distintas neuronas nunha mesma capa tomen funcións similares devido a sua conectividade. De esta forma evitase a aparición de features nas representacións internas moi similares. A problemática da coadaptación radica en que repetición de unha mesma feature no modelo aumenta o seu peso nas predicción conducindo ao modelo ao sobreaxuste. A diferencia de outros métodos de regularización, o dropout impide que a rede sobreaxuste ao ser adestrada nun número elevado de épocas. Esto permite aforrar a búsqueda do número de épocas óptimo, deixando que a rede adestre ata que os erros en adestramento e validación converxan.

As funcións de pérdida empregadas para optimizar a rede foron duas. En primeiro lugar na etapa de preadestramento buscamos predecir as features obtidas en cada grafo polo que empregamos o MSE como función de pérdida 3.16.

$$L_{MSE} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2, i = 1, \ldots, m \in Mvariables \tag{3.16}$$

Para adestrar o modelo no problema de clasificación empregamos a mesma función de pérdida que no caso anterior co XGboost, a pérdida de entropía cruzada 3.3.

O optimizador empregado nas etapas de preadestramento e de adestramento foi ADAM (Adaptative moment estimation). Este optimizador combina duas modificacions do método de descenso de gradiente estocástico, permitindo converxer no óptimo en menos iteracións. De forma intuitiva, cando adestramos en lotes mediante descenso de gradiente estocastico, a actualización dos pesos ocorre de forma errática, baixando en

zigzag ata o óptimo. O optimizador adam combina as técnicas RMSProp (Root Mean Square Propagation) e Momentum para minimizar as variacións dos pesos que menos acercan a pérdida ao óptimo permitindo que o descenso de gradiente sexa máis rápido.

O algoritmo momentum toma a media movil exponencialmente ponderada (EWMA, Exponentially weighted moving average) dos gradientes 3.17. Onde m_t coresponde ao valor dos gradientes agregados no instante t, w_t son os pesos no instante t, α_t e o learning rate no instante t e β o hiperparámetro para a media movil, que controla a influencia da media no instante anterior para o instante t.

$$w_{t+1} = w_t + \alpha m_t \tag{3.17}$$

$$m_t = \beta m_{t-1} + (1-\beta)\frac{\partial L}{\partial w_t}$$

Obtendo un promedio ponderado dos gradientes en cada instante suaviza os cambios nos pesos do modelo de tal forma que se manten a dirección de cambio cara o mínimo da función de coste minimizando o efecto zig-zag causado polo adestramento en lotes.

O algoritmo RMSProp segue unha estratexia similar. En vez de tomar o promedio ponderado toma o promedio ponderado do cuadrado dos gradientes. Gracias ao termo $\frac{\partial L}{\partial w_t}]^2$ de v_t os parámetros con un gradiente elevado ven reducidas as suas variacións mentras que nos parámetros que presentan un menor gradiente permítese un avance maior. En 3.18 vemos como se formula RMSProp sendo ϵ unha constante moi baixa para evitar dividir por cero e v_t a suma acumulada do cuadrado dos gradientes ponderados.

$$w_{t-1} = w_t - \frac{\alpha_t}{(v_t+\epsilon)^{\frac{1}{2}}}\frac{\partial L}{\partial w_t} \tag{3.18}$$

$$v_t = \beta v_{t-1} + (1-\beta)\frac{\partial L}{\partial w_t}^2$$

O optimizador ADAM emprega estas duas metodoloxías para adaptar o tamaño do salto en cada iteración permitindo evitar óptimos locais e reducindo o tempo en converxer no óptimo global.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial L}{\partial w_t}, v_t = \beta_2 v_{t-1} + (1-\beta_2)\frac{\partial L}{\partial w_t}^2 \tag{3.19}$$

Nesta expresión β_1 e β_2 controlan a influencia dos gradientes de etapas anteriores en ambos métdos. Como m_t e v_t se inicializan a cero ambos parámetro tenden a estar sesgados cara este valor. Este optimizador corrixe este problema mediante as expresións 3.20. Intuitivamente, adaptamos o descenso de gradiente tras cada iteración controlando o seu sesgo.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{3.20}$$

ADAM emprega estas correccións $\hat{w}_t$ e $\hat{v}_t$ na ecuación 3.19 dando lugar aos pesos correxidos:

$$w_{t+1} = w_t - \hat{m}_t(\frac{\alpha}{\sqrt{\hat{v}_t}+\epsilon}) \tag{3.21}$$

3.2.3. Preparación das particións de adestramento e validación

Para poder realizar optimización de hiperparámetros con medidas honestas de erro realizamos un particionamento do conxunto de datos en 5 kfolds empregando un 90 % dos datos para adestramento e o 10 % restante para validación.

Previa a optimización de hiperparámetros de ambos modelos fixemos unha búsqueda inicial para o parámetro k que define os entornos de extracción de features no caso do XGBoost e a conectividade do grafo *knn* para a GNN.

Para avaliar o efecto de este hiperpárametro nos dous casos empregouse un modelo sen optimizar e avaliouse o erro na partición de validación variando o valor de k. Para o conxunto de datos da GNN tamén se comparou do mesmo xeito o uso do grafo de Gabriel xunto aos Grafos *knn*. Esta búsqueda realizouse empregando a mesma partición para evitar o elevado tempo computacional da validación cruzada, xa que non pretendemos obter unha medida de erro xeral, senón o valor de dito parámetro que mellor resultados aporta.

3.3. Adestramento e búsqueda de hiperparámetros

A continuación describirase o procedemento seguido para adestrar os dous modelos de clasificación ew o procedemento de optimización sobre os seus hipérparámetros. Dentro da optimización de hiperparámetros diferenciamos dous grupos: Por unha parte os parámetros empregados durante o procesado dos datos; Tamaño do entrno *knn* empregado na extracción de features para o XGboost e método da creación dos grafos para a GNN, incluindo o grafo de gabriel e distintos niveis de conectividade para o grafo *knn* e por outra parte temos os propios hiperparámetros de cada un dos modelos.

Como optimizar todo o conxunto de hiperparámetos conxuntaménte requeriría de un esforzo computacional enorme ao ter que preprocesar os datos multitude de veces decidiuse facer unha primeira búsqueda sobre os parámetros de preprocesado avaliando a clasificación con modelos sen optimizar e, posteriormente, optimizar cada modelo

sobre os parámetros de preprocesado que mellor resultados den en esta primeira etapa.

A optimización de hiperparámetros nos dous modelos realizouse en validación cruzada mediante un procedemento de búsqueda bayesiana. De forma intuitiva este método emprega a información obtida nos modelos axustados en cada iteración para crear un modelo probabilístico que mapea os hiperparámetros aos valores da función obxectivo $P(L_{CE}|hiperparametros)$. Este modelo probabilistico denominase surrogado. O modelo surrogado crease a partir de un número de iteracións iniciais nas que se explora o espacio de hiperparámetros. Unha vez creado o surrogado determinase un novo candidato a avaliar e unha vez avaliado actualizase o surrogado. Continuase iterando ata que a función coste non mellora ou non hai variabilidade na función coste como para determinar novos candidatos. A continuación describirase o método empregado para crear os surrogados e a función de adquisición empregada para determinar os novos candidatos.

Asumese que a nosa función obxectivo e un proceso aleatorio estacionario que sigue unha distribución normal. A matriz de covarianzas do proceso gausiano obtense mediante o kernel RBF (Radial Basis Function):

$$K(x_i, x_j) = exp(-\frac{||x_i - x_j||^2}{2\sigma^2}) \tag{3.22}$$

Un proceso gausiano é un un un proceso estocastico, é decir, un conxunto de variables aleatorias, no cal calquera subconxunto de dittas variables posue unha distribucion normal multivariant e polo tanto, calquera conbinacion normal de elas ten distribución normal. En este caso empregamos procesos gausianos para modelar a nosa funcion de coste frente aos hiperparametros $f(x)$, tal que calquera conxunto de inputs $x_1, ... x_n$ ten unha distribución conxunta gausiana. O conxunto de variables aleatorias $f(x)$ queda definido por unha función de media $m(x)$ e unha función de covariazas que nos proporciona a similaridade entre cada par de puntos $k(x, x')$.

Dita funcion queda modelada mediante un proceso gausiano $f(x) = \mathcal{N}(m(x), k(x, x')$ onde $E(f(x)) = m(x)$ e $E(f(x) - m(x))(f(x') - m(x') = k(x, x')$ sendo a función de covarianzas a función kernel definida en 3.22.

Sendo $\overrightarrow{\mathbf{f}} = (f(x_1), ..., f(x_n))$ o vector dos valores da función coste avaliados sobre os puntos x buscamos predecir o valor da función coste sobre un novo punto x'. Este novo valor $' = f(x')$ esa normalmente distribuido conxuntamente coas observacions contidas en $\overrightarrow{\mathbf{f}}$, polo tanto:

$$P\left(\begin{bmatrix} \overrightarrow{\mathbf{f}} \\ f' \end{bmatrix}\right) = \mathcal{N}\left[0, \begin{bmatrix} \mathbf{K}[\mathbf{X},\mathbf{X}] & \mathbf{K}[\mathbf{X},\mathbf{x}'] \\ \mathbf{K}[\mathbf{x}',\mathbf{X}] & \mathbf{K}[\mathbf{x}',\mathbf{x}'] \end{bmatrix}\right]$$

Onde $\mathbf{K}[\mathbf{X},\mathbf{X}]$ é unha matriz $n \times n$ onde cada elemento (i,j) ven dado por $k(x_i,x_j)$, $\mathbf{K}[\mathbf{X},\mathbf{x}']$ e un vector de dimension $n \times 1$ onde cada elemento i ven dado por $k(x_i,x')$ e analogamente con $\mathbf{K}[\mathbf{x}',\mathbf{X}]$ e $\mathbf{K}[\mathbf{x}',\mathbf{x}']$.

Como os valores da expresión anterior son normais, a distribución condicional $P(f'|\overrightarrow{\mathbf{f}})$ tamén é normal polo que podemos empregar as formulas para a media e varianza da distribución condicionada.

$$P(f'|\overrightarrow{\mathbf{f}}) = \mathcal{N}(\mu(x'),\sigma^2(x'))$$

$$\mu(x') = \mathbf{K}[\mathbf{x}',\mathbf{X}]\mathbf{K}[\mathbf{X},\mathbf{X}]^{-1}\overrightarrow{\mathbf{f}}$$

$$\sigma^2(x') = \mathbf{K}[\mathbf{x}',\mathbf{x}'] - \mathbf{K}[\mathbf{x}',\mathbf{X}]^{-1}\mathbf{K}[\mathbf{X},\mathbf{x}']$$

Mediante esta expresion podemos estimar a distribución da función coste en calquera punto x'. A estimación viria dada polo valor medio $\mu(x')$ cunha incerteza $\sigma^2(x')$

Unha vez que temos modelada a función coste e a sua incerteza e necesario definir unha estratexia para determinar que novos valores avaliar. Un método moi empleado para seleccionar novos candidatos e o UCB ou Upper Confidence Bound **??**. Este método selecciona como candidatos os puntos que presentan un valor medio alto ou unha gran incerteza favorecendo a exploración de rexións de interese, o parámetro β controla o balance entre os dous termos da expresión.

$$UCB(x') = \mu x' + \beta^{\frac{1}{2}}\sigma(x')$$

Este método de optimización e relativamente costoso computacionalmente xa que en cada iteración e necesario simular un número elevado de procesos gausianos compatibles co noso modelo surrogado para determinar o novo candidato a avaliar. Esto fai que este método sexa útil unicamente cando a función que pretendemos optimizar e mais costosa de avaliar que as propias iteracións no método, ademais, o número de parámetros que podemos optimizar mediante este método e limitado. Non se recomenda optimizar funcións con mais de 20 parámetros xa que con un número elevado de parámetros caemos no desastre da dimensión e o número de iteracións necesarios para poder atopar unha solución boa crece moi rapidamente. Pese a estas limitacions, a optimización de hiperparámetros en modelos de aprendizaxe estadístico é un problema adecuado para resolver mediante este método, xa que é moito mais eficiente que a búsqueda en grella

ou a búsqueda aleatorizada.

A hora de realizar a búsqueda de hiperparámetros empregáronse duas implementacións diferentes. Para o XGboost, empregamos a implementación presente no paquete tidyverse de R [W, 2021]. A rede neuronal foi implementada en python coa librería pytorch geometric [Fey and Lenssen, 2019] e a librería de aprendizaxe distribuido e optimización de hiperpárametros Ray [team, 2019]. A configuración inicial da rede foi deseñada de forma manual e unha vez definida unha estructura base composta de (2-3 capas na cabeza GNN $\longrightarrow$ pooling $\longrightarrow$ 2-3 capas MLP), o modelo foi enviado ao CESGA para buscar os axustes de configuración óptimos no número e tamaño das capas, dropout e parametros do oprimizador mediante a librería Ray en validación cruzada.

Coa rede neuronal probamos duas formas de adestramento: Por unha parte adestramos directamente a rede na tarefa de clasificación, inicializando os pesos da rede de forma aleatoria e por outra preadestramos previamente a cabeza da rede facendo regresión sobre os estadísticos descriptivos da distribución de alturas para inicializar os pesos antes de pasar a tarefa de clasificación.

Capítulo 4

Resultados

A continuación móstranse os resultados obtidos na búsqueda dos hiperparámetros de procesado dos datos e os resultados de clasificación obtidos finalizada a etapa de optimización. As métricas que empregamos en esta etapa móstranse a continuación.

- Accuracy = $\frac{TP_A+TP_B+TP_C}{Total}$
- Micro recall = $\frac{TP_A+TP_B+TP_C}{TP_A+TP_B+TP_C+FN_A+FN_B+FN_C}$
- Macro recall = $\frac{Recall_A+Recall_B+Recall_C}{3}$
- Micro F1 score $\frac{1}{2}(\frac{TP_A+TP_B+TP_C}{P_A+TP_B+TP_C+FP_A+FP_B+FP_C} + \frac{TP_A+TP_B+TP_C}{TP_A+TP_B+TP_C+FN_A+FN_B+FN_C})$
- Macro F1 = $\frac{F1_A+F1_B+F1_C}{3}$

4.1. Optimización do preprocesado

Para determinar o tamaño óptimo do entorno *knn* no que realizar a extracción de features, avaliamos a clasificación empregando un modelo XGboost cos parámetros por defecto sobre os datos procesados con $k = (10, 20, 30, 40, 50)$. Os resultados de esta búsqueda poden verse na táboa 4.1.

O valor do parámetro k que mellores resultados da en todas as métricas da clasificación e 40. Observase un incremento en todas as métricas da clasificación a medida que aumentamos o tamaño do entorno *knn* ata superar knn = 40, onde diminue bruscamente. Se ben os resultados de clasificaión son prometedores empregar un entorno tan amplio pode ser contraproducente se tratasemos de clasificar un monte mixto.

knn	Accuracy	Micro recall	Macro recall	Micro F1 score	Macro F1 score
10	0.596	0.596	0.601	0.596	0.577
20	0.623	0.623	0.629	0.624	0.607
30	0.640	0.640	0.651	0.640	0.626
40	0.761	0.761	0.774	0.761	0.760
50	0.609	0.609	0.633	0.608	0.592

Cadro 4.1: Efecto do tamaño do entorno **knn** sobre os resulados de clasificación do modelo XGboost

No caso da rede neuronal comparamos distintos tamaños dos subgrafos empregados na clasificación. Unha vez creado o grafo *knn* a partir da nube de puntos con $k = 4$ creamos os subgrafos que inclúen 2, 4, 6 e 8 a partir de cada punto. Ademais comparamos este método de xeración de grafos co grafo de gabriel empregando o número de veciñanzas que mellor funcionou de entre estos catro valores.

Conectividade	Tamaño medio	Accuracy	Micro recall	macro recall	Micro F1	Macro F1
2	8.95	0.679	0.690	0.706	0.690	0.695
4	17.90	0.714	0.714	0.723	0.714	0.718
6	27.29	0.728	0.728	0.713	0.728	0.714
8	33.73	0.737	0.738	0.732	0.736	0.739
GG	40.28	0.719	0.706	0.719	0.718	0.719

Cadro 4.2: Efecto do tamaño do método de formación de subgrafos sobre os resultados de clasificación na GNN

Os resultados na búsqueda do método óptimo para a creción de grafos mostranse na táboa 4.2. O método que mellores resultados aporta e expandir en 8 veciñanzas dende cada nodo para crear o seu subgrafo correspondente. Ainda que este foi o tamaño óptimo, o tamaño en memoria do conxunto de datos procesado de esta forma foi demasiado grande para poder realizar a optimización do modelo nun espazo de hiperparámetros adecuado, xa que para realizar esta proba inicial empregamos un modelo pequeno sen

optimizar. Por este motivo decidiuse empregar o entorno inmediatamente inferior, xa que a diferencia observada nos resultados non foi moi diferente (1 % aprox), pero con grafos de este tamaño sí e posible obter modelos de maior complexidade cos recursos dispoñibles (4Gb de memoria grafíca).

4.2. Optimización de hiperparámetros

A búsqueda de hiperparámetros en ambos modelos foi realizada mediante optimización bayesiana. No caso do XGboost xeramos 35 iteracións iniciais e buscamos durante 25 iteracións adicionais. En cada iteración foron xerados 5000 procesos gausianos para determinar o seguinte candidato. O espazo de hiperparámetros onde se realizou a búsqueda e o valor óptimo atopado para cada un deles móstrase na táboa 4.3.

Parámetro	Rango	Óptimo
Número de variables por árbore	(1-14)	5
Tamaño mínimo dos nodos	(10-300)	151
Número de arbores	(10-200)	103
Profundidade máxima das árbores	(1-10)	6
η	(0.0001-1)	0.2468601
γ	(0.001-2)	1.85

Cadro 4.3: Espazo de hiperparámetros e configuración óptima

Analogamente levamos a cabo a optimización de hiperparámetros na rede. Grazas ao CESGA e ao software de aprendizaxe distribuido Ray [team, 2019], foi posible realizar a búsqueda de hiperparámetros de este modelo en validación cruzada xa que os custos computacionais en tempo e memoria que conleva optimizar un modelo de este tipo fan imposible o uso de un equipo convencional. Adestrouse un modelo por partición de validación cruzada de forma simultánea empregando a pérdida promedio de ditas particións. Esto permite empregar un criterio de parada de forma simultánea en todas as particións e aforrar épocas de adestamento. O número de épocas de adestramento fixouse a 50 xa que previamente se comprobou que o modelo tende a converxer nun número de épocas menor (35-45) sen sobreaxustar e optimizar este parámetro conxuntamente aos do modelo pode dificultar o proceso.

Sección	Parámetro	Rango	Óptimo
GNN	GAT Features	(2-64)	6
	SAGE1 features	(32-256)	160
	SAGE2 features	(32-256)	-
	Heads GAT	(1-32)	24
	FC1 features	(128-4096)	2100
	FC2 features	(128-4096)	1506
	Additional SAGE	(0-1)	0
	Additional FC	(0-1)	0
	GAT dropout	(0.05-0.4)	0.21
	SAGE dropout	(0.05-0.4)	0.18
	FC1 dropout	(0.05-0.4)	0.095
	FC2 dropout	(0.05-0.4)	0.351
	Batch size	(10-1000)	595
Adam	β_1	(0.5-0.999)	0.982
	β_2	(0.5-0.999)	0.973
	Learning rate	log(1e-7-1e-1)	0.012
	Weight decay	log(1e-9,5e-3)	0.004

Cadro 4.4: Búsqeda de hiperparámetros para a GNN

Empregaronse 256 iteracións no proceso de optimización e os resultados da búsqueda poden verse na táboa 4.4. De forma similar ao que ocurriu no paso anterior os valores óptimos do modelo non apenas melloran os resultados da rede axustada de forma manual (Acuraccy = 0.754 no modelo óptimo frente 0.735 no modelo anterior), pero supoñen un incremento notable no número de parámetros do modelo, e por tanto, nos recursos necesarios para realizar prediccióǹs con el. Por este motivo decidiuse empregar o modelo

previamente axustado que pode verse na figura 3.4, pero mantendo os parámetros do optimizador e de regularización obtidos no óptimo.

4.2.1. Clasificación XGBoost

A continuación móstranse as métricas de avaliación da clasificación en validación do modelo optimizado. Esta clasificación foi avaliada de duas formas: Clasificando a mostra de validación punto a punto 4.5 e sumando as predicciónss en cada segmento de $8m^2$ para clasificar a clase maioritaria 4.6. As matrices de confusion obtidas en cada unha das partións de validación cruzada na clasificación punto a punto e na clasificación a nivel segmento poden verse en anexos nas figuras 6.4 e 6.5

Métrica	Media	Desviación estandar
Accuracy	0.818	0.036
Macro Recall	0.817	0.039
Micro Recall	0.818	0.035
Macro F1 score	0.819	0.038
Micro F1 score	0.818	0.035

Cadro 4.5: Resultados da clasificación punto a punto do modelo XGboost en validación cruzada

Métrica	Media	Desviación estandar
Accuracy	0.859	0.046
Macro Recall	0.875	0.054
Micro Recall	0.859	0.045
Macro F1 score	0.856	0.050
Micro F1 score	0.859	0.046

Cadro 4.6: Resultados da clasificación a nivel segmento do modelo XGboost en validación cruzada

Vemos que os resultados da clasificación melloran cando facemos a agregación das prediccións a nivel segmento, incrementando todas as métricas avaliadas en un 4 % aproximadamente. Esto pode ser interesante a hora de clasificar plantacións onde sabemos que non hai especies mixtas pero non podemos extrapolar estas métricas a montes que non cumplan estas características.

Na figura 4.1 podemos ver a importancia de cada variable empregada no XGBoost. Esta importancia obtense como a ganacia en precisión, é decir, canto se incrementa a precisión do modelo cando a variable é empregada nas divisións. As variables de maior importancia son a altura máxima e a a altura no percentil 90. Que estas variables sexan as mais importantes para a clasificación pode ser problemático, xa que se ven afectadas moi facilmente pola presenza de outras especies mais altas ou por outros obxectos que que podan alterar estas variables como postes ou tendido eléctrico.

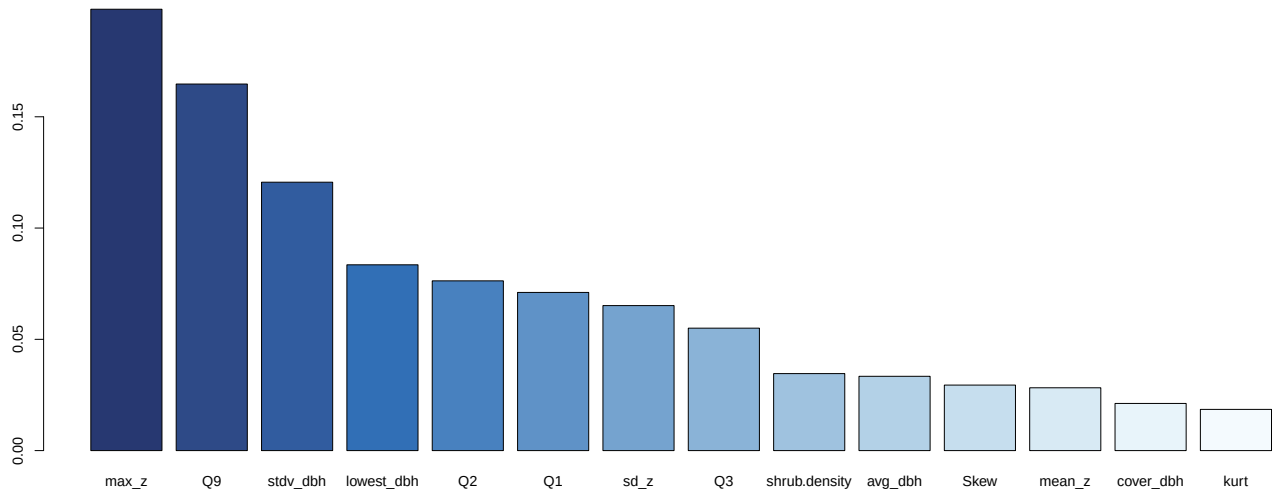

Figura 4.1: Importancia das variables en XGboost

4.2.2. Clasificación GNN

Como comentamos antes, a configuración da rede que imos empregar para a clasificación non foi a óptima, senón unha versión coa mesma estructura pero de menor tamaño, para poder ser empregada fora do CESGA.

Sobre o modelo 3.4 probamos duas estratexias de adestramento diferentes. En primeiro lugar adestramos durante 50 epocas a rede, omitindo a seccion de regresión, cos

parámetros determinados na etapa anterior e avaliamos o erro 4.7. De forma analoga ao modelo xgboost avaliamos os resultados de clasificación sumando as predicciónsen cada segmento 4.8.

Métrica	Media	Desviación estandar
Accuracy	0.735	0.030
Macro Recall	0.741	0.016
Micro Recall	0.735	0.030
Macro F1 score	0.730	0.025
Micro F1 score	0.735	0.030

Cadro 4.7: Resultados da clasificación punto a punto en validación cruzada da GNN sen preadestrar

Métrica	Media	Desviación estandar
Accuracy	0.775	0.045
Macro Recall	0.764	0.037
Micro Recall	0.763	0.045
Macro F1 score	0.753	0.047
Micro F1 score	0.767	0.049

Cadro 4.8: Resultados da clasificación a nivel segmento en validación cruzada da GNN sen preadestrar.

As métricas da classificación de este modelo sitúanse un 8 % por debaixo do modelo XGboost. O incremento nas métricas observado ao sumar as predicciónsen cada segmento ten unha magnitude similar ao modelo anterior. As matrices de confusión obtidas sobre cada partición de validación cruzada para os dous métodos atópanse en 6.6 e 6.7.

Sobre este mesmo modelo probamos realizar unha etapa previa de adestramento, o preadestramento consiste en adestrar o modelo en unha tarefa mais sinxela relaciona-

da coa finalidade real, de esta forma os pesos da rede xa están adaptados para obter features útiles dos datos facilitando o adestramento no propósito inicial [Li et al., 2019].

Para preadestrar o modelo decidimos os estadisticos descriptivos da distribucion de alturas que empregamos no modelo XGBoost extraidas sobre os puntos de cada subgrafo e adestrar a rede para predecir ditas variables. A rede 3.4 incluía unha capa de regresión con este propósito que conectaba coa saída da capa de pooling, de forma que ao adestrar a rede nesta tarefa estaremos so actuando sobre os pesos da cabeza da rede, deixando o adestramento do MLP de clasificación para unha segunda etapa. Para a etapa de preadestramento empregaronse 70 épocas en cada partición e como función de pérdida empregouse o MSE. Os parámetros empregados no optimizador foron os mesmos en nas duas etapas, pero reinicializando o optimizador entre etapas.

A os resultados da clasificación punto a punto e a nivel segmento co modelo preadestrado pode observarse nas táboas 4.9 e 4.10 respectivamente. Se ben a converxencia do modelo foi mais rápida tras esta etapa de preadestramento, os resultados de clasificación non se incrementaron de forma notable. Apenas un 1 % na clasificación punto a punto e sin diferencias apreciables na clasificación a nivel segmento. As matrices de confusión de cada partición obtidas mediante este modelo atopanse nos anexos 6.6 e 6.2.

Métrica	Media	Desviación estandar
Accuracy	0.746	0.041
Macro Recall	0.755	0.027
Micro Recall	0.745	0.041
Macro F1 score	0.741	0.036
Micro F1 score	0.745	0.042

Cadro 4.9: Resultados da clasificación punto a punto en validación cruzada da GNN con preadestramento

Un posible motivo polo cal a rede poderia estar funcionando peor que o XGboost e que ao estar traballando grafos xerados a partir de segmentos troceados de forma artificial, os subgrafos xerados a partir dos nodos próximos aos bordes dos segmentos poderíanse ver afectados por este corte brusco, afectando tamén a clasificación.

Para comprobar se existe este efecto sobre a clasificación obtivemos a excentricidade de cada un dos puntos do segmento no grafo que forman. Esta excentricidade obtívose

Métrica	Media	Desviación estandar
Accuracy	0.775	0.055
Macro Recall	0.782	0.038
Micro recall	0.764	0.048
Macro F1 score	0.769	0.056
Micro F1 score	0.775	0.055

Cadro 4.10: Resultados da clasificación a nivel segmento en validación cruzada da GNN con preadestramento

como o máximo dos camiños mais curtos entre cada nodo e os demais fixando a distancia dos arcos en 1.

Na figura 4.2 pode observarse un histograma bivariante da pérdida de clasificación nos datos de validación frente a excentricidade. Sobre este histograma representouse a densidade kernel nestas variables. O gráfico non parece reflexar esta hipótese, mais ben o contrario, nas zonas con alta excentricidade a pérdida non alcanza valores tan elevados. O coeficiente de correlación de pearson entre estas duas variables e -0.163 con un pvalor cercano $2{,}49 \times 10^{-25}$ confirmando unha relación contraria ao que esperábamos.

O gráfico mostra que a rexión de maior densidade situase en valores da pérdida cercanos a 0 e valores da excentricidade situados entorno a 7. Os puntos con alta excentricidade non alcanzan pérdidas tan elevadas como os de menor excentricidade, unha posible explicación para esto e que se teñen unha alta excentricidade pertencen a grafos de maior tamaño, permitindo a rede extraer información de rexións mais amplias e minimizando o erro.

Por último para tratar de avaliar a mellora potencial de ambos modelos se dispuxéramos de mais datos de adestramento, realizamos un estudo da eficacia de ambos modelos con distintos tamaños para o conxunto de adestramento.

Para realizar este estudo tomamos unha división de adestamento e validación e aplicamos submostraxe sobre a partición de adestramento. Adestramos progresivamente con mostras do 5,10,20,40,60,80 e 100 % dos segmentos contidos en dita partición. Os segmentos empregados para adestrar en cada etapa foron seleccionados de forma aleatoria fixando a mesma semente de aleatoriedade para os dous modelos para que as submostras sexan idénticas. Na figura 4.3 poden verse os resultados de este estudo.

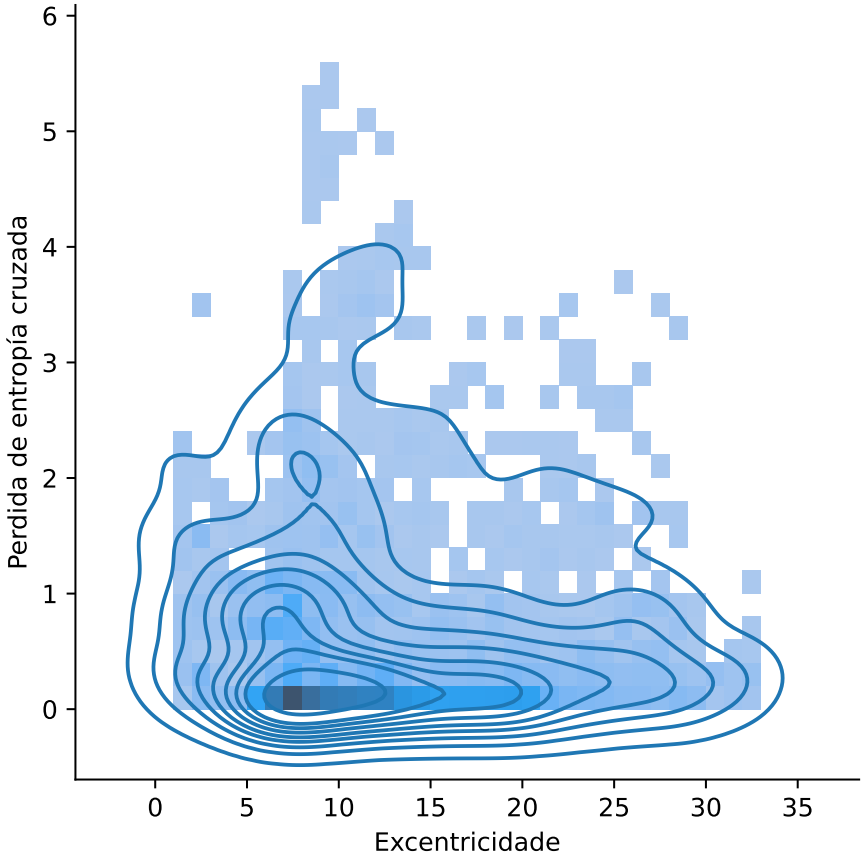

Figura 4.2: Relación entre excentricidade e pérdida na clasificación.

Aparentemente os dous modelos teñen un comportamento diferente ao reducir o tamaño da mostra de adestramento. O modelo XGboost presenta un accuracy estable entorno ao 80 % sempre que o tamaño da mostra supere o 10 % dos datos de adestramento, reducíndose só cando empregamos o 5 % dos datos. Esto pode deberse a unha alta correlación entre os datos, e decir, os estadísticos extraidos de distintas rexións son moi semellantes sendo redundantes entre eles.

Pola contra, ainda que a GNN non alcanza un accuracy tan elevado na clasificación, presenta unha tendencia positiva a medida que aumentamos o tamaño da mostra de adestramento e esta tendencia mantense ata chegar ao 100 % dos datos. Esto podería indicar que de posuir mais datos de adestramento a rede podería beneficiarse de eles e mellorar a clasificación.

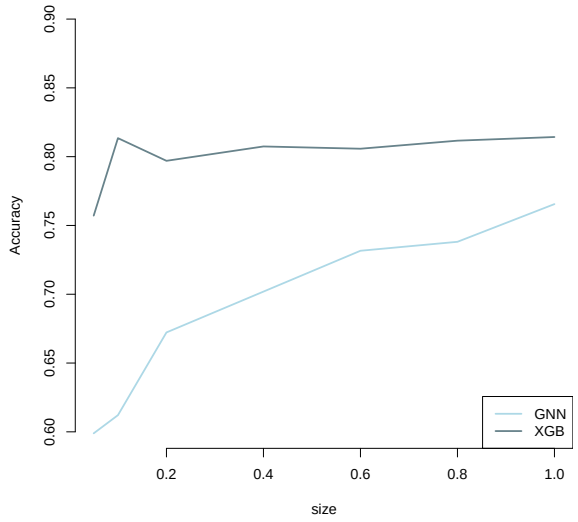

Figura 4.3: Efecto do tamaño da mostra de adestramento sobre a clasificación

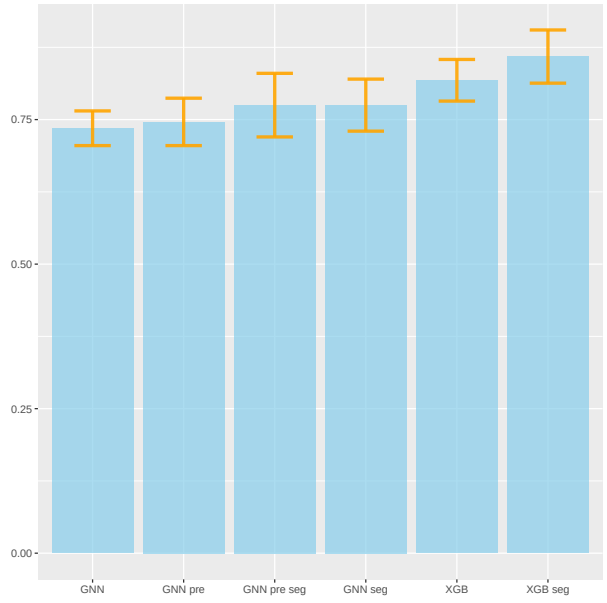

Figura 4.4: Resumen dos modelos de clasificación

Capítulo 5

Discusión

Neste traballo conseguimos adestrar dous clasificadores con capacidade de identificar as tres especies de piñeiro *Pinus pinaster*, *Pinus radiata* e *Pinus sylvestris* con unha eficacia aceptable. En primeiro lugar, empregando o método de extracción de features manual, de forma analoga ao método descrito en [Alonso et al., 2020b], acadamos un accuracy en validación cruzada do 81 % na clasificación punto a punto e 85 % sobre os segmentos de $8m^2$ frente ao 81 % acadado na identificación de castiñeiros en dito artigo, no cal empregan segmentos de $20m^2$ e variables radiometricas adicionais en distintos momentos temporais. Compre ter en conta que a clasificación punto a punto que levamos a cabo con este método emprega como entorno de extraccion de features os 40 veciños mais proximos polo que a identificacion de especies en montes mixtos pode dar peores resultados, de todas formas aporta maior flexibilidade que clasificar a nube de puntos segmentada direcamente. ademais e posible variar o tamaño dependendo da variablilidade presente nas parcelas que vaiamos a clasificar.

En [Li et al., 2010] tamen aplicaron un método de extracción de features a partir da distribucion de alturas. O seu conxunto de datos estaba composto de nubes de puntos cunha densidade de 40 puntos/m^2 con unha arbore por instancia e 5 especies diferentes. Como clasificador usaron unha arbore de decisión. O seu método acadou unha accuracy do 84.4 %, o cal se atopa proximo ao obtido co noso modelo a pesar da maior calidade dos datos, suxerindo que o clasificador XGboost e mais adecuado para esta metodoloxia.

Por outra parte o modelo de deep learning que adestramos ainda que non alcanza resultados tan altos na clasificación, quedando aproximadamente 8 puntos por debaixo en todas as métricas, é capaz de identificar as tres espécies sen a necesidade de extraer manualmente features. Ao non depender da extracción de features manual poderia ser adestrado no recoñecemento de outras estructuras ou especies arbustivas que non sexan identificables mediante as features plantexadas neste problema. Por outra parte vimos que este modelo se podería mellorar todavía os seus resultados de aumentar o tamaño do conxunto de datos de adestramento, a diferencia do modelo basado en features que non se via afectado pola redución dos datos de adestramento 4.3.

En [Liu et al., 2021] trataron de resolver un problema de clasificación similar, empregando un modelo de deep learning que permite evitar a extracción manual de features e que traballa directamente coa nube de puntos desordenada, sen colapsar os datos en imaxes 2D. Para adestrar o seu modelo empregaron nubes de puntos de alta densidade, cada arbore do seu conxunto de datos esta composto aproximadamente por 2000 puntos, o cal permite identificar características morfolóxicas da especie a simple vista. No seu artigo acadaron un Accuracy do 88 % na clasificación arbore a arbore, resultados superiores aos da nosa rede ao modelo XGboost, pero compre ter en conta a diferencia de densidade entre o conxunto de datos que empregaron eles e o noso.

En relación a outros intentos de abordar este problema parece que o método proposto en este traballo pode dar bos resultados pero e necesario mellorar varios aspectos do adestramento.

O volume de datos de adestramento é insuficiente, a nube de puntos empregada neste traballo consta de 90.000 puntos mentres que o habitual na literatura en este tipo de problemas e traballar con millóns de puntos. Duas formas de incrementar o tamaño de adestramento serian aumentar a densidade das nubes de puntos ou a extensión das parcelas empregadas. Como o interés de este traballo radica en poder empregar unha base de datos gratuita e se demostrou que un aumento do tamaño resultaria beneficioso para o modelo a estratexia a seguir seria incluir mais parcelas de esta mesma base de datos.

No conxunto de datos empregado para realizar este traballo as tres especies estaban perfectamente separadas. O verdadeiro interese de aplicación de estes modelos estaria en detectar en que rexions se atopa cada especie e poder segregar adecuadamente cando fagan fronteira entre especies. Para poder acadar un modelo capaz de separar adecuadamente as especies cando sexan conlindantes é necesario adestrar o modelo en un conxunto de datos con parcelas mixtas. As metricas de clasificación que obtivemos ao tratar con parcelas sen mistura de especies poden cambiar moito se tratamos de clasificar mediante este modelo parcelas con arbores de varias especies, e posiblemente o tamaño dos entornos óptimo para realizar a clasificación en esa situación sexan diferentes.

De cumplírse estas duas situacións o modelo poderia modificarse lixeiramente para realizar a clasificación punto a punto sen considerar entornos de agregación artificiais como fixemos en este trabllo xa que non seria necesario aumentar os datos incluindo grafos para cada punto e ao ter especie mixtas o interes radicaria en segregar semanticamente ditos grafos. A modificación proposta en ese caso sería traballar co grafo completo, transformalo a traves da cabeza da gnn e aplicar o MLP de clasificación sobre as features xeradas sobre cada un dos nodos do grafo en vez de aplicalo sobre a agregación por pooling das features de cada subgrafo.

Outra estratexia que pode ser interesante probar sería aplicar outro método de preadestramento. No traballo vimos que preadestrar o modelo na prediccion das features derivadas das alturas supón un incremento lixeiro na accuracy, entorno a un 1 %. Esta estratexía pode non ser a optima xa que formzamos ao modelo a aprender unha feartures que poden non ser as mellores para o problema de clasificación e implican a extraccion previa de features, pero hai outras formas de abordar o preadestramento. Seria interesante probar a preadestrar o modelo identificando parcelas forestais de outras zonas sen arbores, de este modo o modelo estaria preadestrado en recoñecer os datos que posteriormente clasificará.

Capítulo 6

Anexos

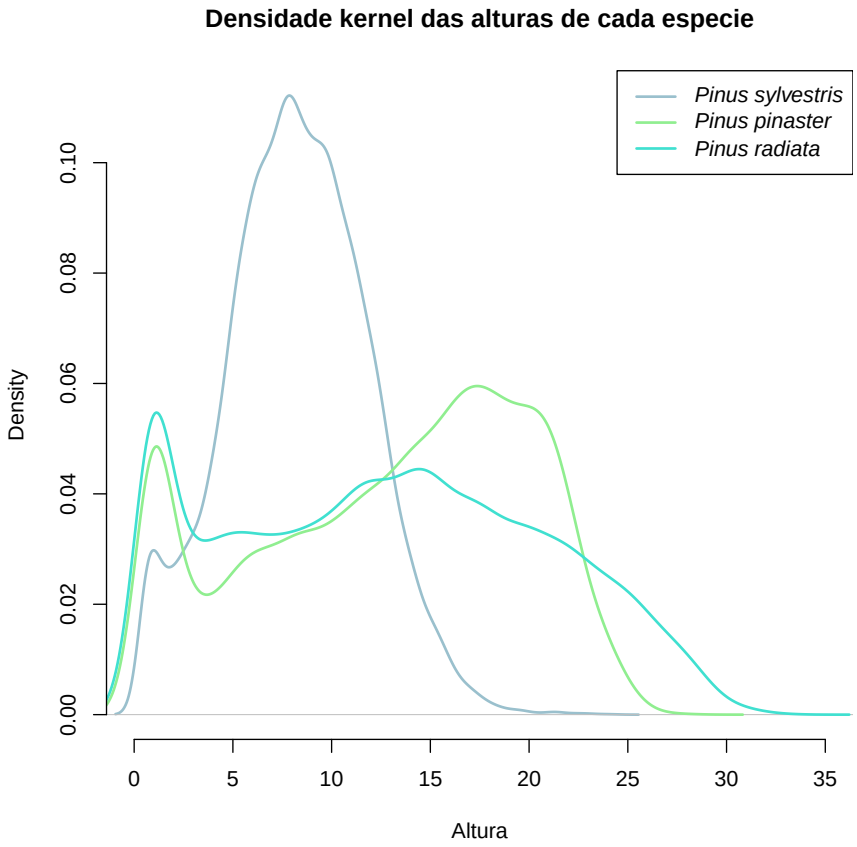

Figura 6.1: Distribución de alturas das tres especies eliminando os retornos con altura inferior a $25cm$

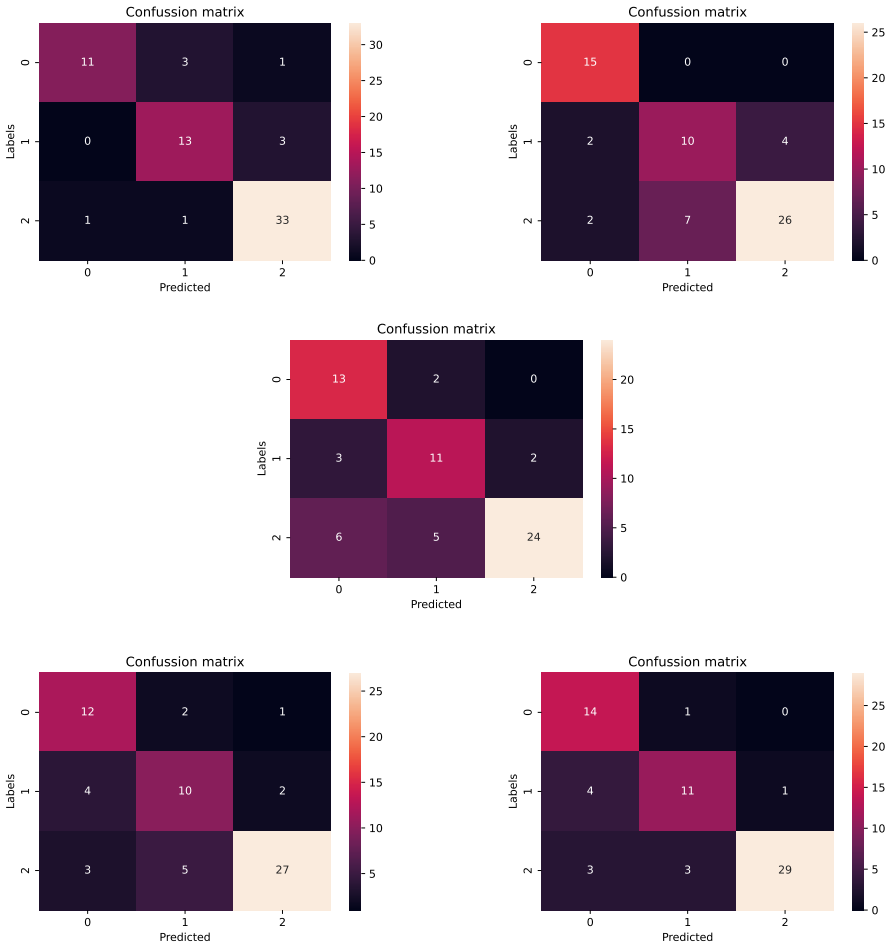

Figura 6.2: Matrices de confusión a nivel segmento da GNN con preadestramento

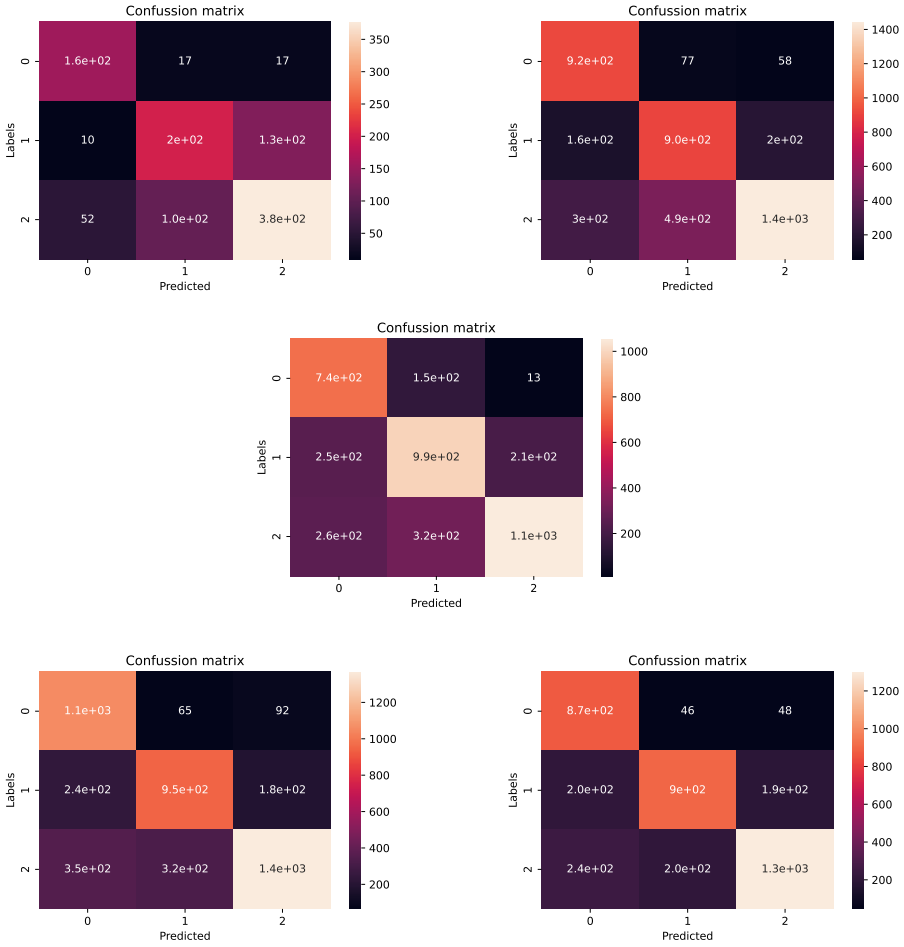

Figura 6.3: Matrices de confusión da clasificación punto a punto da GNN con preadestramento

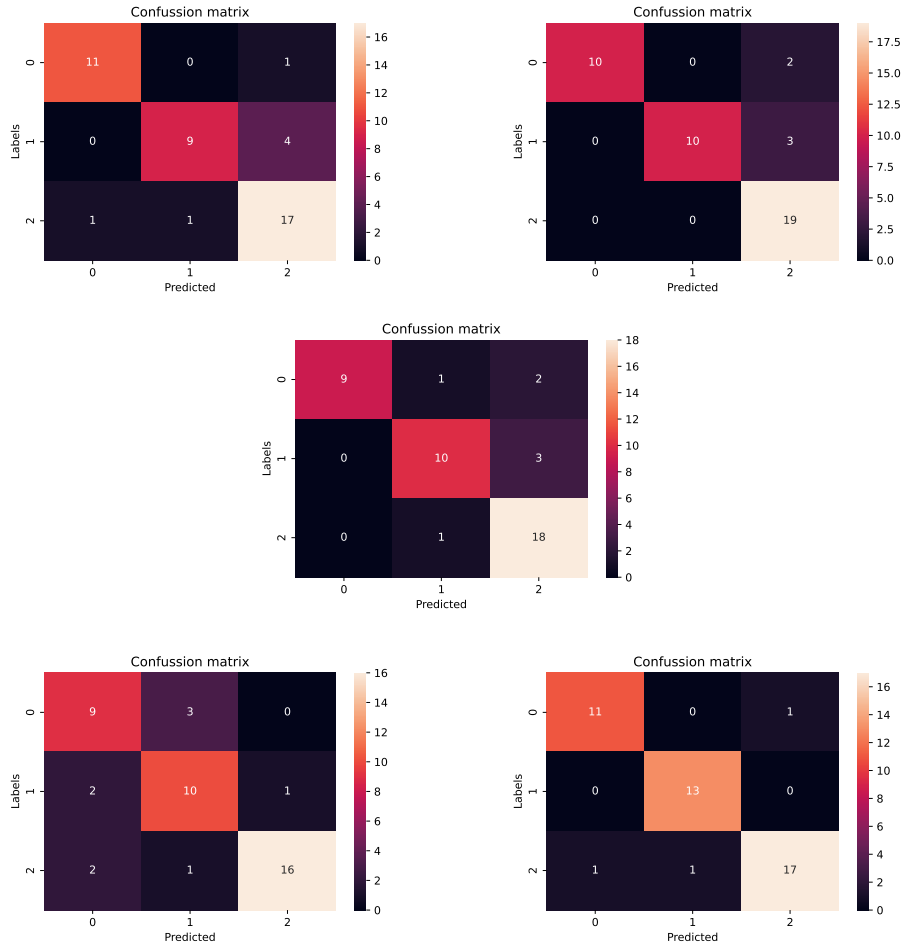

Figura 6.4: Matrices de confusión a nivel segmento do modelo XGboost

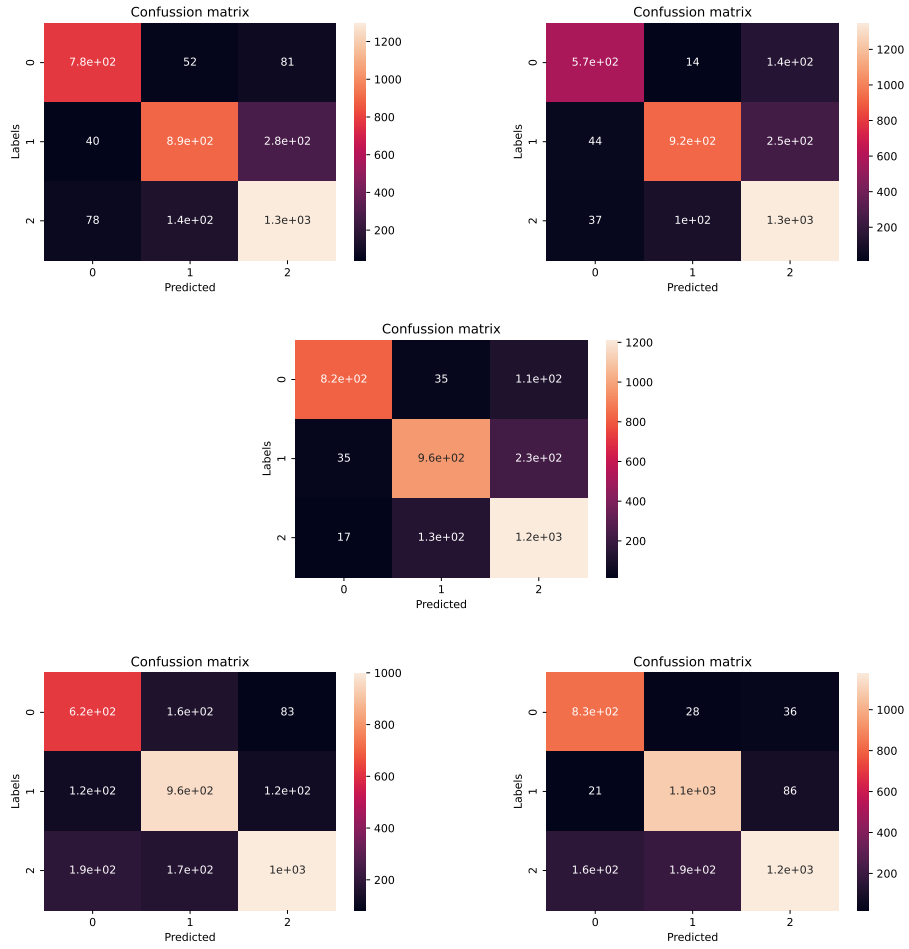

Figura 6.5: Matrices de confusión da clasificación punto a punto do modelo XGboost

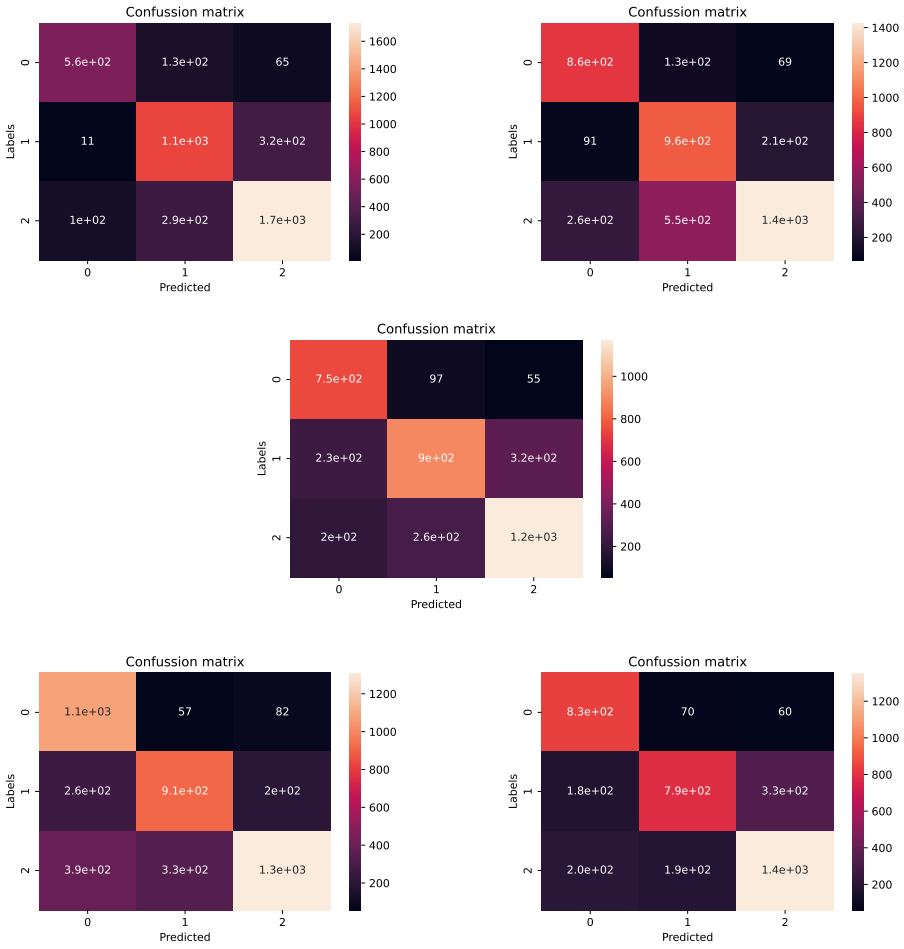

Figura 6.6: Matrices de confusión da clasificación punto a punto coa GNN

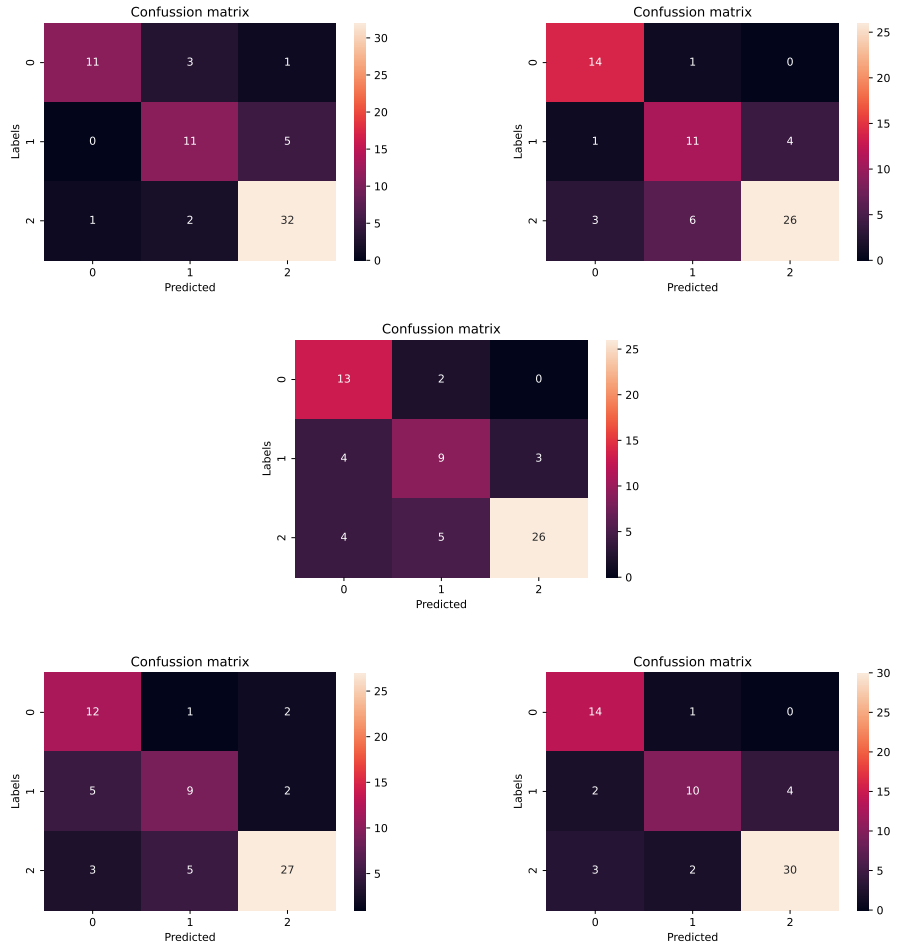

Figura 6.7: Matrices de confusión da clasificación a nivel segmento da GNN

Bibliografía

[Alonso et al., 2020a] Alonso, L., Armesto, J., and Picos, J. (2020a). Chestnut cover automatic classification through lidar and sentinel-2 multi-temporal data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-3-2020:425–430.

[Alonso et al., 2020b] Alonso, L., Picos, J., Bastos, G., and Armesto, J. (2020b). Detection of very small tree plantations and tree-level characterization using open-access remote-sensing databases. *Remote Sensing*, 12(14).

[Brody et al., 2021] Brody, S., Alon, U., and Yahav, E. (2021). How attentive are graph attention networks?

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. pages 785–794.

[Fey and Lenssen, 2019] Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric.

[Gabriel and Sokal, 1969] Gabriel, K. R. and Sokal, R. R. (1969). A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259–278.

[Graham, 2008] Graham, B. (2008). Using low density, small footprint lidar in forest inventory applications in the southeastern u.s. *All Theses*.

[Hamilton et al., 2017] Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive representation learning on large graphs.

[Li et al., 2019] Li, H., Singh, B., Najibi, M., Wu, Z., and Davis, L. S. (2019). An analysis of pre-training on object detection.

[Li et al., 2010] Li, J., Hu, B., Sohn, G., and Jing, L. (2010). Individual tree species classification using structure features from high density airborne lidar data. In *2010 IEEE International Geoscience and Remote Sensing Symposium*, pages 2099–2102.

[Liu et al., 2021] Liu, M., Han, Z., Chen, Y., Liu, Z., and Han, Y. (2021). Tree species classification of lidar data based on 3d deep learning. *Measurement*, 177:109301.

[Mizoguchi et al., 2017] Mizoguchi, T., Ishii, A., Nakamura, H., Inoue, T., and Takamatsu, H. (2017). Lidar-based individual tree species classification using convolutional neural network.

[Scarselli et al., 2009] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

[team, 2019] team, T. R. (2019). Ray, universal api for building distributed applications.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.Ñ., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

[Veličković et al., 2017] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2017). Graph attention networks.

[W, 2021] W, H. (2021). Tidyverse.