Pràctica 2

Aprenentatge Computacional

Enginyeria Informàtica 2021/2022 - UAB

Martí Caixal Joaniquet - 1563587

Ricard López Olivares - 1571136

Jairo Villarroel Rodriguez - 1571069

Introducció **3**

Apartat B **4**
Comparant models 4
Precision-Recall 5
ROC Curve 6
Comparant Kernels 7
Comparant Gamma i C al kernel RBF 9
Gamma 9
C 10
Gamma i C conjuntats 11
Comparant graus al kernel polinòmic 12
Resum sobre l'ús de paràmetres 12

Apartat A **13**
1. EDA (exploratory data analysis) 13
2. Preprocessing (normalitzation, outlier removal, feature selection..) 16
Normalització 16
Valors nuls 19
Codificació de dades categòriques 20
First Component Analysis 22
Anomalies 24
Classes balancejades 26
Polynomial Features 27
3. Model Selection 28
Regresor logístic 28
SVM 29
Random Forest 30
XGBoost 31
SVM lineal 32
knn 33
Ensambles 34
4. Cross Validation 35
Importància de cross validar 35
Proporció Train-Test 35
K-Fold Cross Validation 37
Leave One Out 38
5. Metric Analysis 39
Mètriques disponibles 39
Corbes PR i ROC 40
Classification Report 41
6. Hyperparameter Search 42

Introducció

Aquest informe està dividit en 2 apartats. Al primer es compara diferents models de Machine Learning i com, a través dels seus paràmetres, modifiquen el seu comportament. Al segon apartat es dona més importància a analitzar les dades amb les que es treballa i modificar-les. També es mostren eines i mètodes per fer front a problemes vists al primer apartat.

A continuació es llisten els objectius de la pràctica:

- Aplicar models de classificació, ficant l'èmfasi en:
    - Aplicar diferents classificadors (regressors logístics, SVM, Random Forest, etc …) i entendre les millores d'aplicar kernels.
    - Avaluar correctament l'error del model.
    - Visualitzar les dades i el model resultant.
- Ser capaç d'aplicar tècniques de classificació en casos reals.
- Validar els resultats en dades reals.

Per ambdós apartats es treballa sobre la mateixa base de dades. Es tracta d'un seguit d'observacions meteorològiques diàries a Australia.

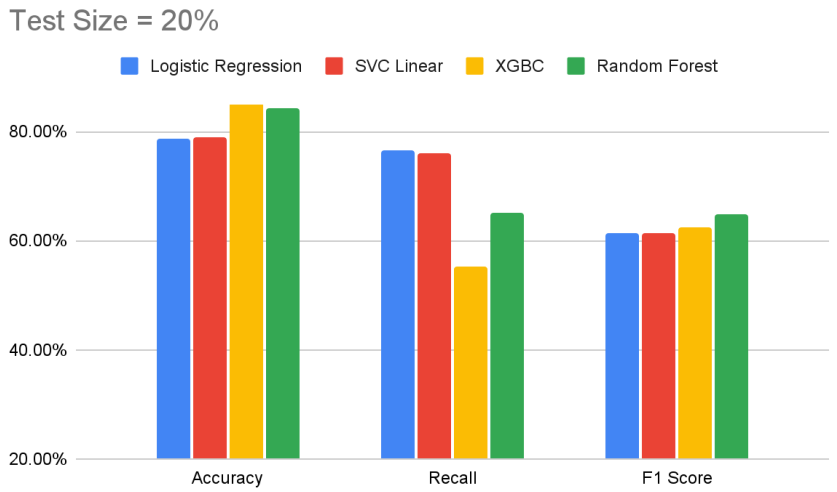
Apartat B

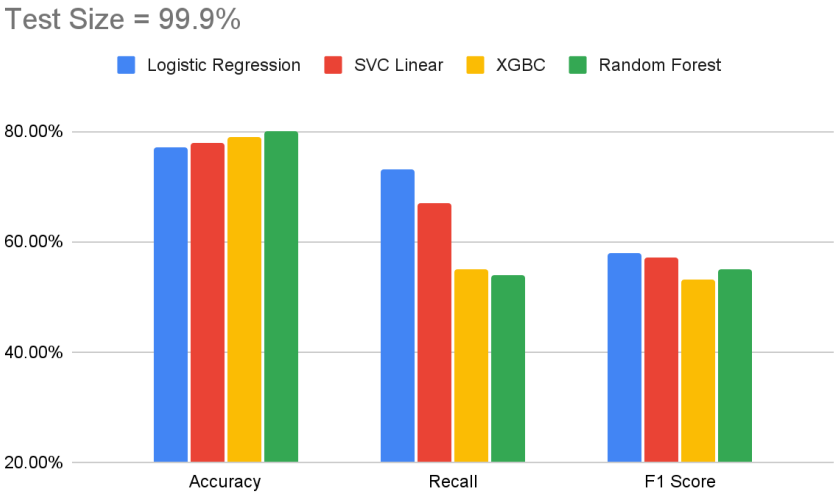
Comparant models

Es comença fent una comparació inicial de diferents models. Cal anotar que per ara no es toquen els varis paràmetres per modificar el comportament de cada un.

La mètrica important no és l'Accuracy. Les dades amb les que tractem tenen un 80% de dies solejats. Si un model fes una predicció on sempre és un dia solejat, l'Accuracy seguiria sent alt. És important, doncs, tenir també una bona predicció per quan ha de ploure. Amb la mètrica F1 Score podem veure de forma més clara si realment està fent bé les prediccions.

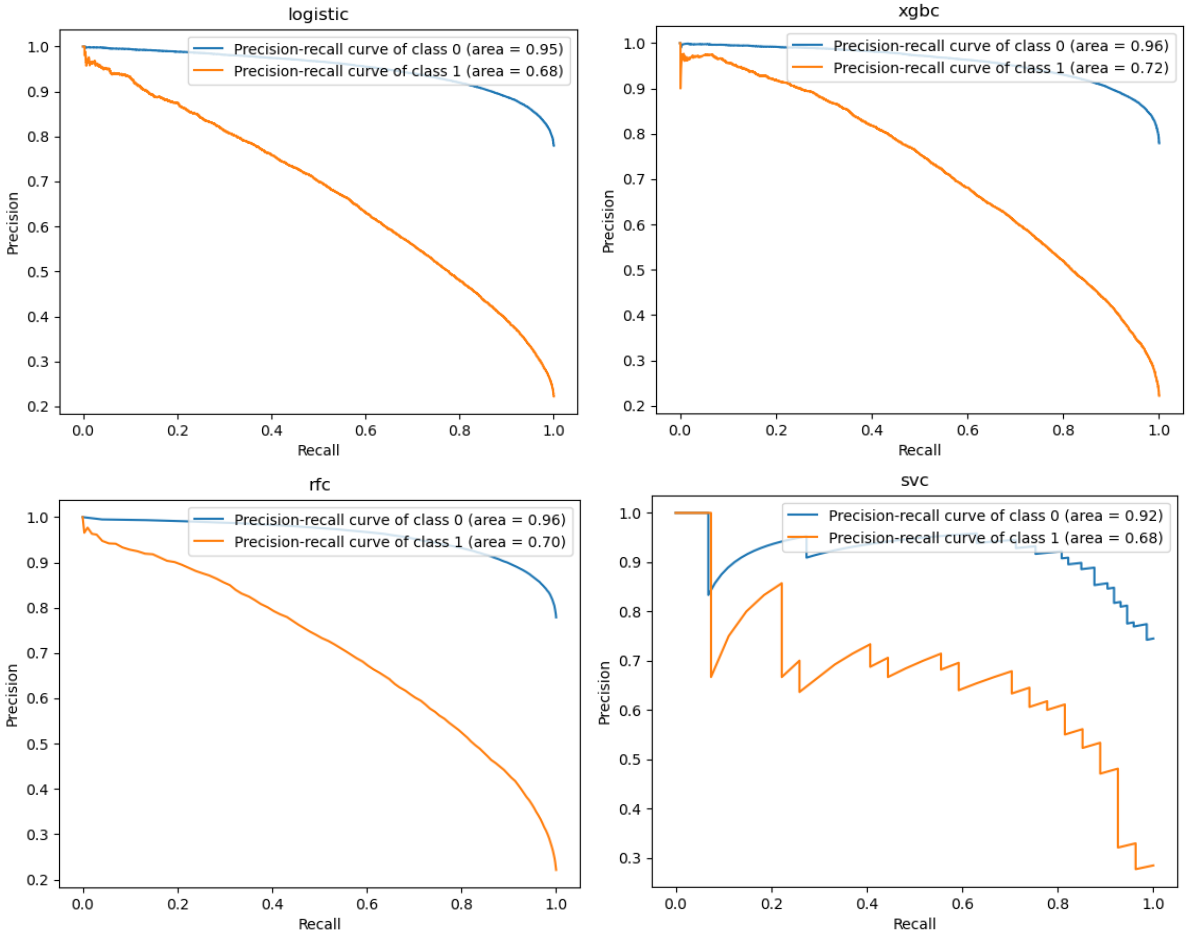
El principal punt a destacar és la poca variació de resultats tot i fer servir moltes dades pel test o no. (20% vs 99%). Les mètriques han sigut idèntiques fins que la mida del conjunt Test no ha sigut superior al 95%. La raó és que el dataset està compost per 150000 registres, sent la gran majoria pràcticament idèntics (mateix dia, diferent any).





Precision-Recall

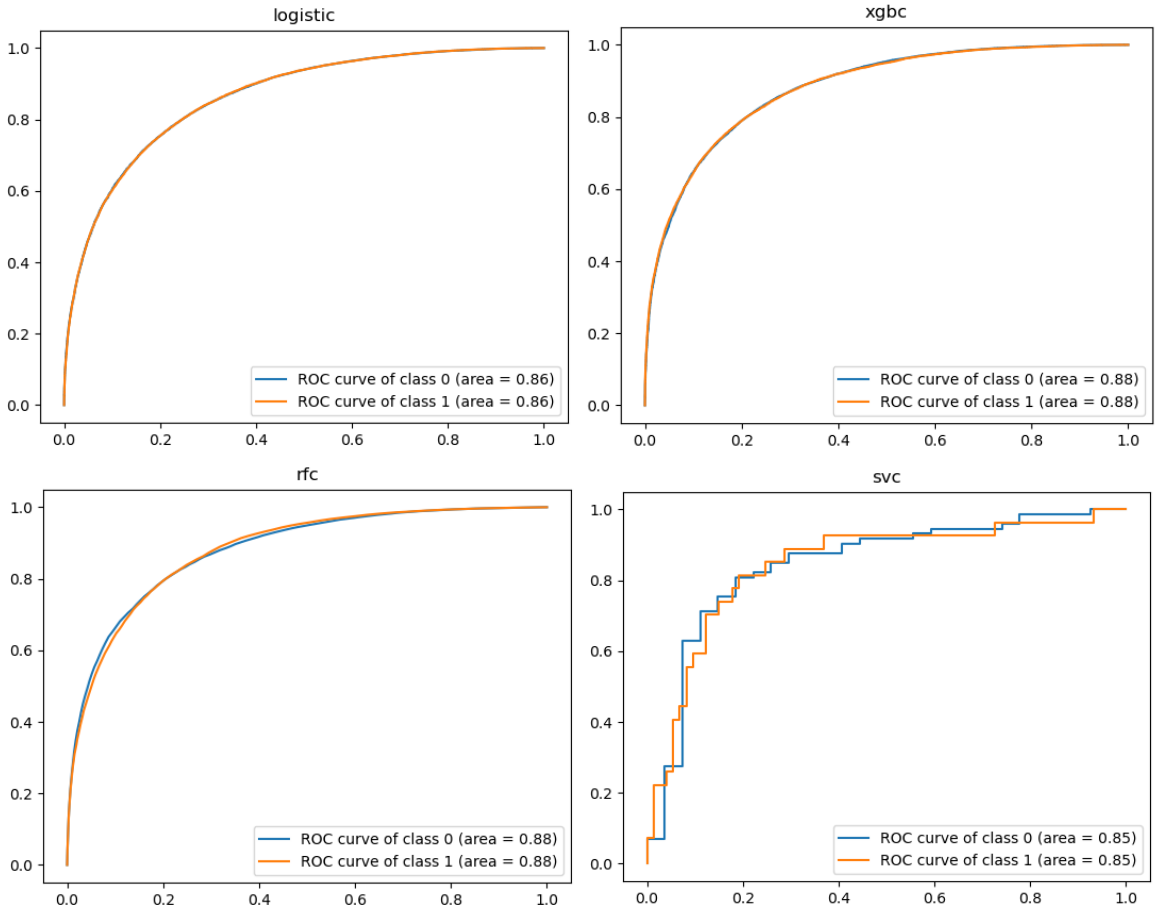
A continuació es mostra la curva Precision-Recall per cadascun dels models. Es pot veure més fàcilment el problema esmentat anteriorment sobre dades no balancejades.



Anotació: Pel SVC s'ha hagut de reduir el nombre de dades per qüestions de rendiment, d'aquí la diferència respecte les altres gràfiques

La curva per la classe 0 (no pluja) és molt bona per tots els models. En canvi, la curva per la classe 1 (pluja) és pitjor. Veiem ara de forma més gràfica que el model no té problemes per predir quan no plourà, però té bastantes dificultats per poder predir amb certesa quan ha de ploure.

ROC Curve



Anotació: Pel SVC s'ha hagut de reduir el nombre de dades per qüestions de rendiment, d'aquí la diferència respecte les altres gràfiques

Les ROC Curves són molt dolentes quan es té un dataset no balancejat com és el nostre cas. Es veu com per ambdues classes les prediccions semblen ser bones, però anteriorment ja s'ha vist que no és així.

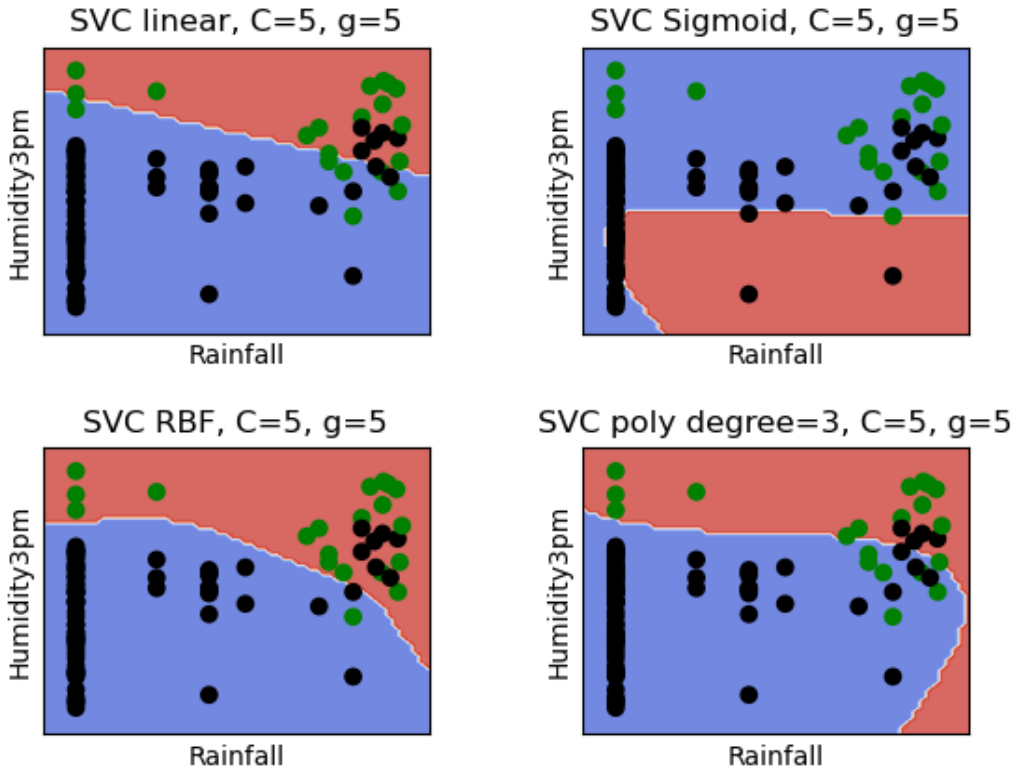
Comparant Kernels

Les SVM, quan es troben en situacions que no poden separar les classes linealment, han de recorrer als Kernels. Aquests aumenten la dimensionalitat de les dades per poder trobar un hiperplà que sí permeti separar les classes.

Per facilitar el mostreig dels resultats, l'entrenament dels models i les prediccions es fan basant-se en dos característiques i així es poden mostrar en 2 dimensions sense problema.

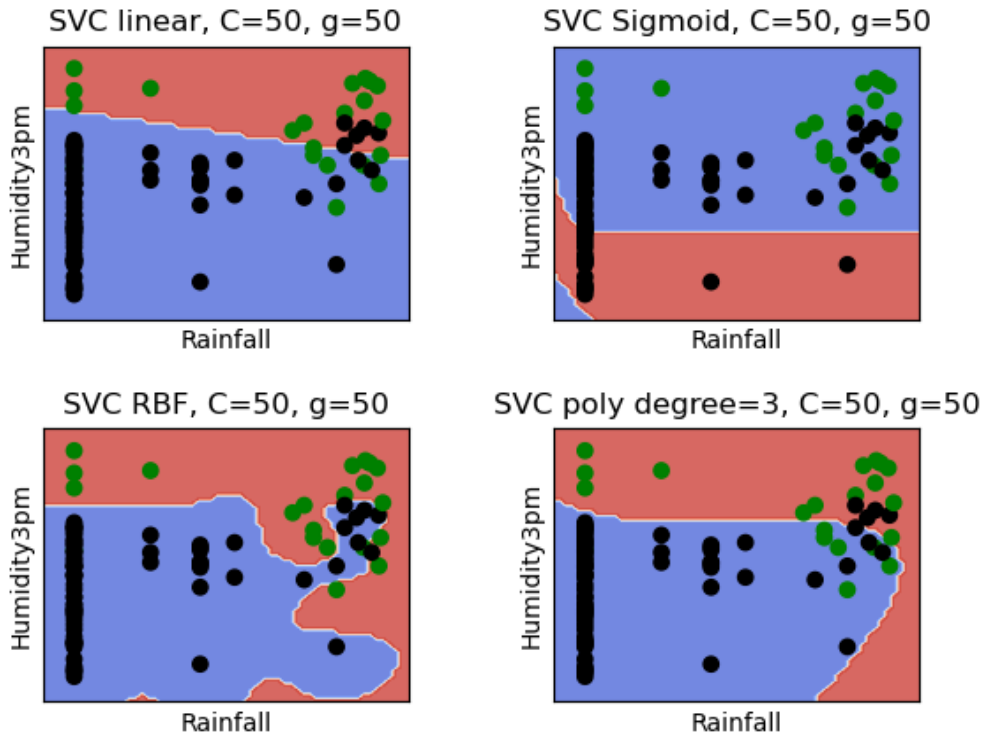
Aquestes característiques no s'han agafat de forma aleatòria. S'utilitza la funció "SelectKBest" de SkLearn i s'aconsegueix les dos característiques més importants. Una altra opció hauria sigut utilitzar Principal Component Analysis, però llavors el gràfic no mostraria dades d'atributs reals, sinó combinacions on no es podria veure sentit a simple vista.

A continuació es veu el comportament de 4 kernels diferents, cap d'ells té un bon resultat.



Anotació: les proves amb SVM s'han fet agafant 100 mostres aleatòries. Si bé no genera les millors prediccions, ajuda a veure i explicar el classificador.

Tal i com es veu a la següent imatge, els kernels tenen paràmetres per modificar el seu comportament.



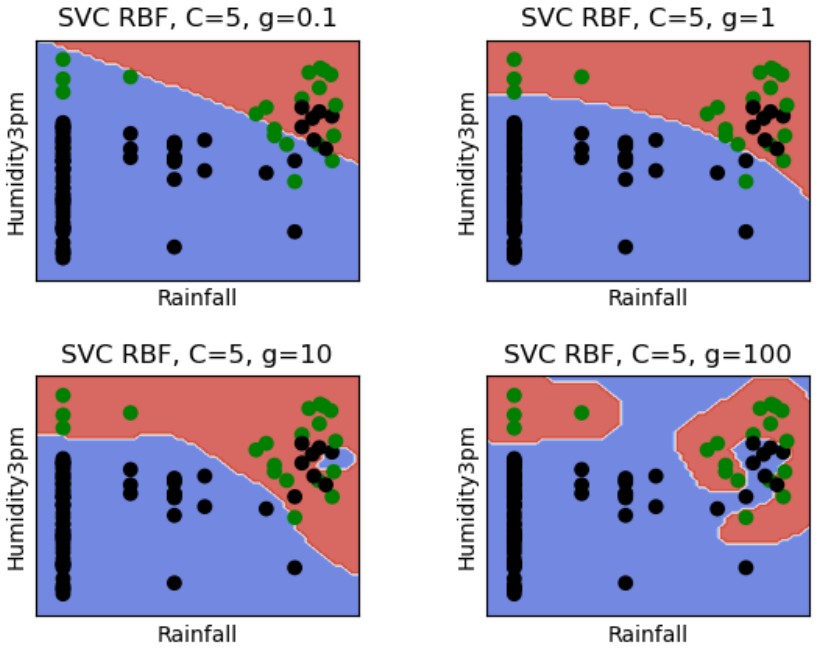
Tant el Linear com el Polinomial no canvien gaire degut a la seva pròpia naturalesa. El primer busca una funció lineal per separar les classes, així que no té marge de millora. El segon, en canvi, és polinomial. Si bé el nom indica que busca una funció polinomial, i així és, en cap moment s'ha indicat canviar el grau de la funció en qüestió. Així doncs, al model li resulta impossible adaptar-se millor a les dades sense poder aumentar el grau de la funció.

Els altres dos kernels sí que canvien respecte els paràmetres C i G (gamma). Als següents subapartats es fa èmfasi a cada un d'ells i es mostren més examples.

Comparant Gamma i C al kernel RBF

Gamma

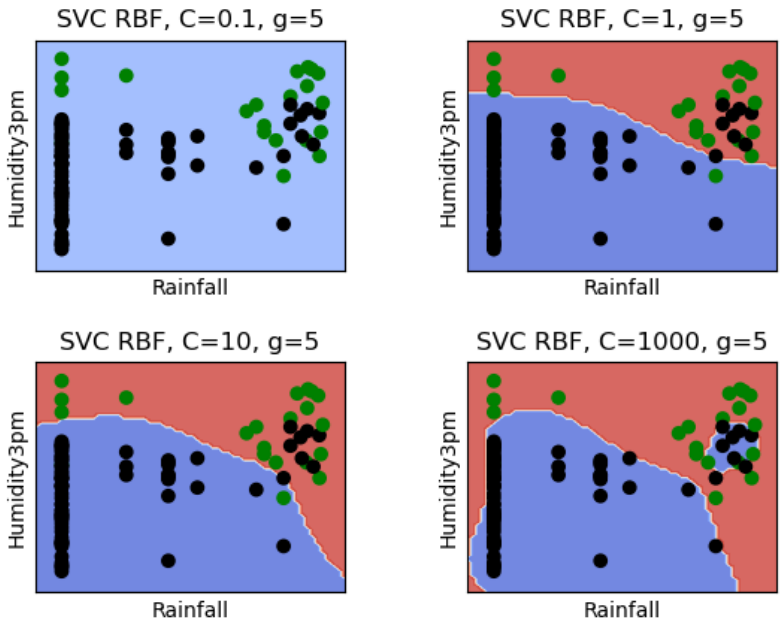
El paràmetre gamma s'utilitza per hiperplans no lineals. Com més gran és el valor, més importància dona en fer "fit" a les dades de training.



Es veu clarament que a mesura que el valor aumenta, el clasificador busca conseguir un "fit" de les dades training de manera més restrictiva i exacte. L'última imatge mostra un clar exemple d'overfitting.

C

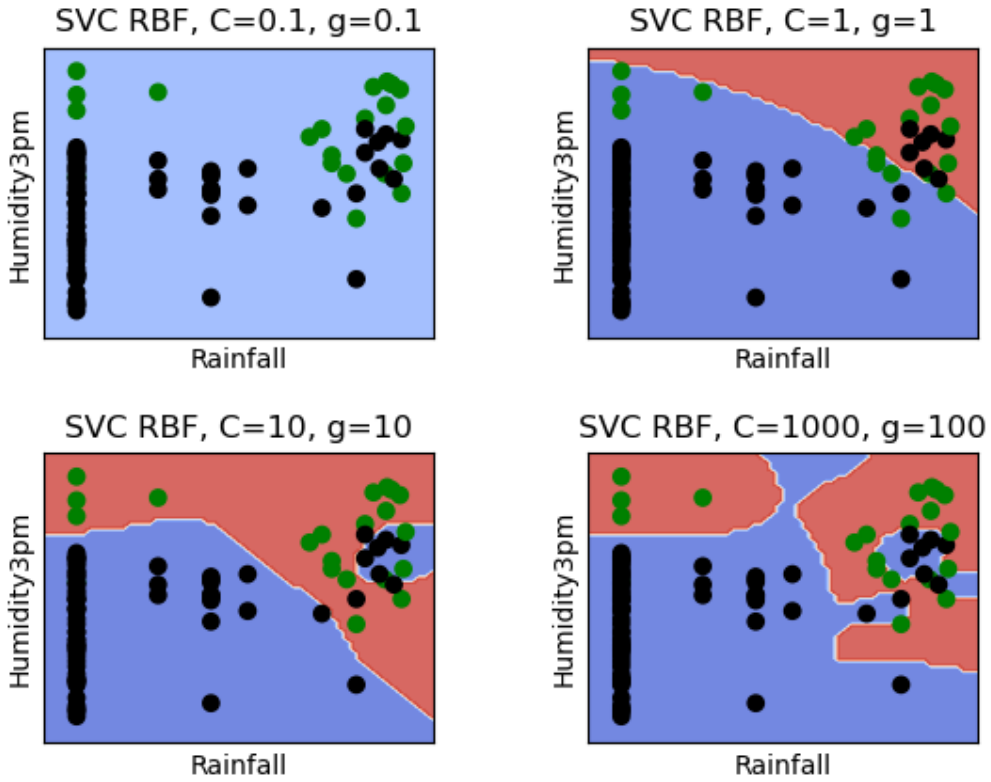
El paràmetre C indica el marge d'error permés. Controla l'equilibri entre una frontera entre classes suau i classificar les mostres correctament. Quan es tenen outliers, aquest paràmetre és molt útil, doncs permet ignorar alguns outliers que d'altre manera el SVM intentaria posar dins de la frontera de la seva classe.



Igual que amb el paràmetre anterior, incrementar el valor porta a overfitting. La gran diferència està en una frontera més suau i sense tantes corbes.

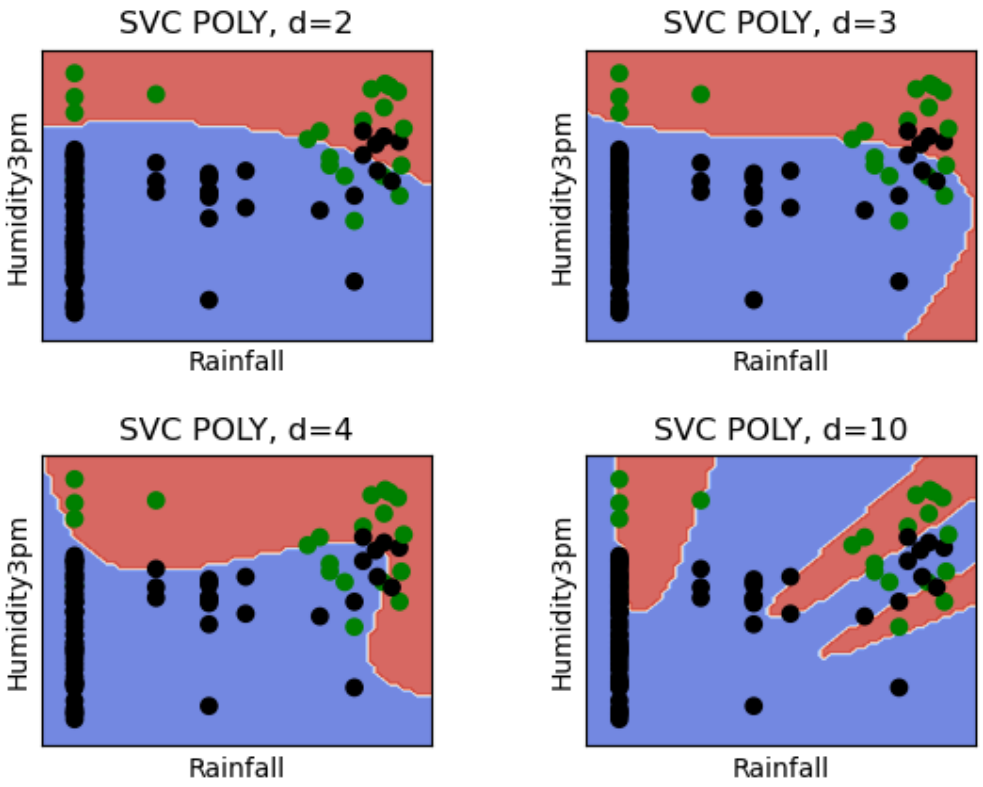
Gamma i C conjuntats

Com és d'esperar, incrementant el valor d'ambdós paràmetres s'aconsegueix tenir unes fronteres que semblen molt bones, però realment és overfitting.



Comparant graus al kernel polinòmic

El degree (d) és un paràmetre utilitzat al kernel polinòmic. Bàsicament és el grau de la funció polinòmica utilitzada per trobar un hiperplà que separa les classes.



Clarament és veu que aumentar el valor de degree fa que aumenti el nombre de corbes que pot agafar la funció. Això comporta donar-li més flexibilitat fins al punt de crear un overfitting molt sever.

Resum sobre l'ús de paràmetres

És fàcil veure com saber fer-ne un bon ús és molt important. Cal sempre buscar un equilibri per tal de no trobar-se amb un overfitting.

Apartat A

1. EDA (exploratory data analysis)

La base de dades utilitzada en aquest apartat és la mateixa amb les que a l'apartat anterior s'ha fet una comparació entre models. És anomenada weatherAUS i conté 142000 dades diàries d'observacions meteorològiques arreu d'Australia. L'objectiu principal que es planteja és podeu preveure si l'endemà serà un dia plujós o no.

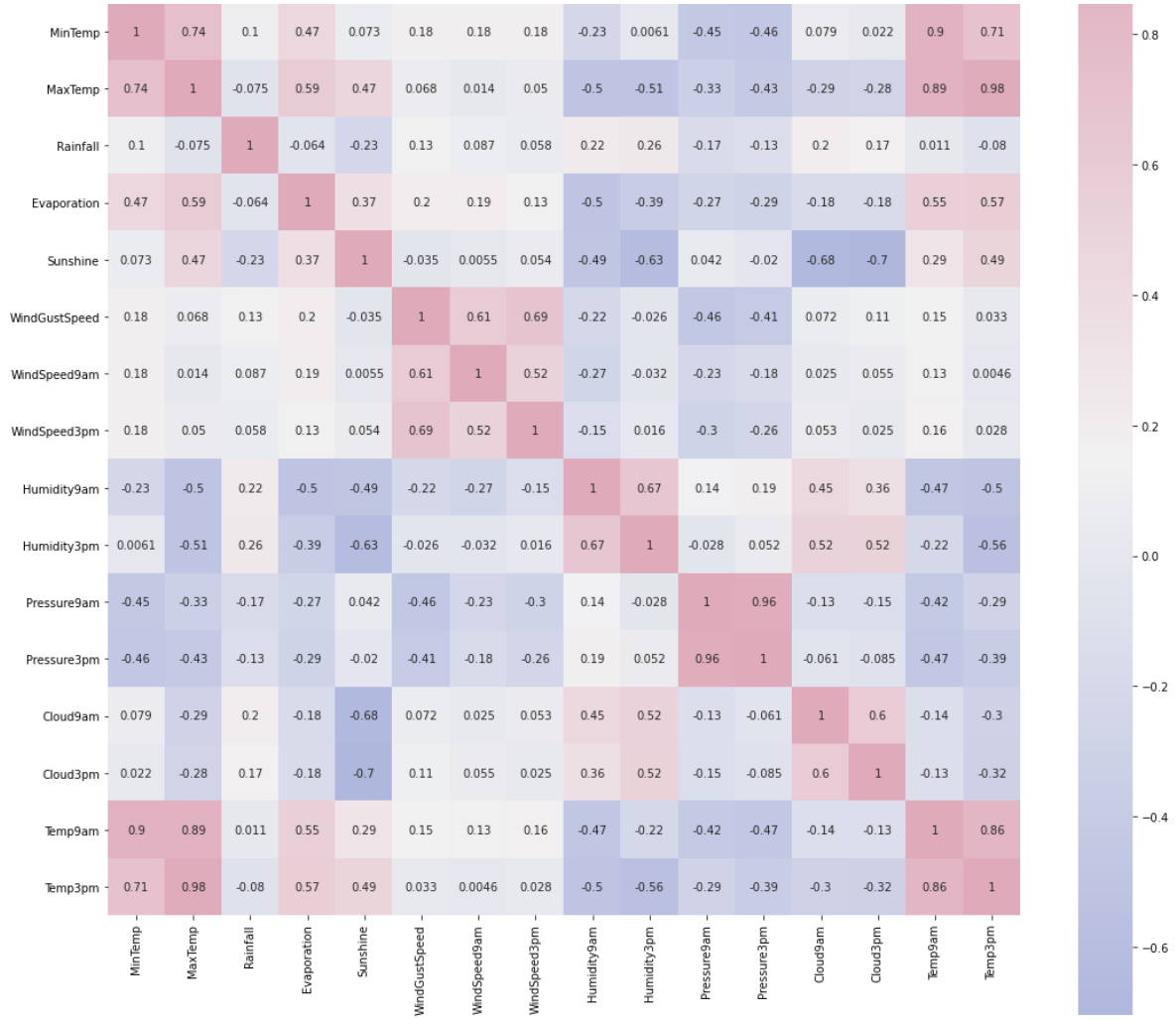
La base de dades està formada per 23 atributs. D'aquests, 16 són de tipus decimal continus i descriuen dades relacionades amb els núvols, temperatures, evaporació, etc. Els 7 atributs restant son categòrics i aporten dades com la direcció del vent (Nord, Sud-Est, …) o la ciutat (Sydney, Canberra, Adelaide, …).

#	Column	Non-Null Count	Dtype
0	Date	145460 non-null	object
1	Location	145460 non-null	object
2	MinTemp	143975 non-null	float64
3	MaxTemp	144199 non-null	float64
4	Rainfall	142199 non-null	float64
5	Evaporation	82670 non-null	float64
6	Sunshine	75625 non-null	float64
7	WindGustDir	135134 non-null	object
8	WindGustSpeed	135197 non-null	float64
9	WindDir9am	134894 non-null	object
10	WindDir3pm	141232 non-null	object
11	WindSpeed9am	143693 non-null	float64

#	Column	Non-Null Count	Dtype
12	WindSpeed3pm	142398 non-null	float64
13	Humidity9am	142806 non-null	float64
14	Humidity3pm	140953 non-null	float64
15	Pressure9am	130395 non-null	float64
16	Pressure3pm	130432 non-null	float64
17	Cloud9am	89572 non-null	float64
18	Cloud3pm	86102 non-null	float64
19	Temp9am	143693 non-null	float64
20	Temp3pm	141851 non-null	float64
21	RainToday	142199 non-null	object
22	RainTomorrow	142193 non-null	object

Entre aquests atributs se'n troba l'objectiu, "RainTomorrow", aportant un valor binari si l'endemà és plujós o no.

Fent una taula de correlació, a primera vista sembla que s'està davant d'unes dades amb atributs suficientment correlacionats entre ells.

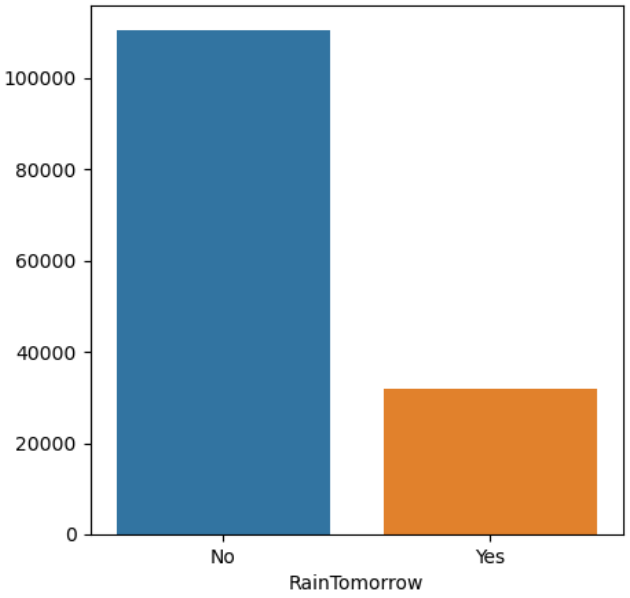


Aquest fet pot portar a pensar que els models a generar tindran unes bones mètriques i serà una tasca relativament simple.

Un aspecte a tenir en compte és el nombre de valors nuls per cada atribut. Com es veu a la imatge inferior, hi ha bastants valors nuls. Alguns atributs, com per exemple Evaporation i Sunshine, tenen pràcticament ⅓ dels valors nuls. Els atributs efectats s'hauran de tractar d'una manera o altre, però mai podran ser tant útils a les prediccions com si es tingués el valor real.

Date	0	WindDir9am	10566	Pressure3pm	15028
Location	0	WindDir3pm	4228	Cloud9am	55888
MinTemp	1485	WindSpeed9am	1767	Cloud3pm	59358
MaxTemp	1261	WindSpeed3pm	3062	Temp9am	1767
Rainfall	3261	Humidity9am	2654	Temp3pm	3609
Evaporation	62790	Humidity3pm	4507	RainToday	3261
Sunshine	69835	Pressure9am	15065	RainTomorrow	3267
WindGustDir	10326	WindGustSpeed	10263		

Un altra aspecte que encara influeix més a l'hora d'avaluar un model predictor és tenir unes dades balancejades. És important tenir dades on totes les classes estiguin representades de manera uniforme entre totes les mostres. El dataset amb el que es treballa té ¼ de les mostres dient que l'endemà plourà i ¾ dient que no plourà.



Aquesta situació és anomenada "Accuracy Paradox". Per portar-ho a extrems, suposem el següent conjunt de dades:

Predicted class / Actual class	Terrorist	Not terrorist	Sum
Terrorist	**10**	0	10
Not terrorist	**990**	999000	999990
Sum	1000	999000	1000000

L'accuracy resultant és 99.9%, a priori un bon resultat, però fixant-nos millor, les prediccions estan lluny de ser òptimes. Dels 10 terroristes (la classe més important a predir), se'n ha trobat 1000 quan realment hi ha 10

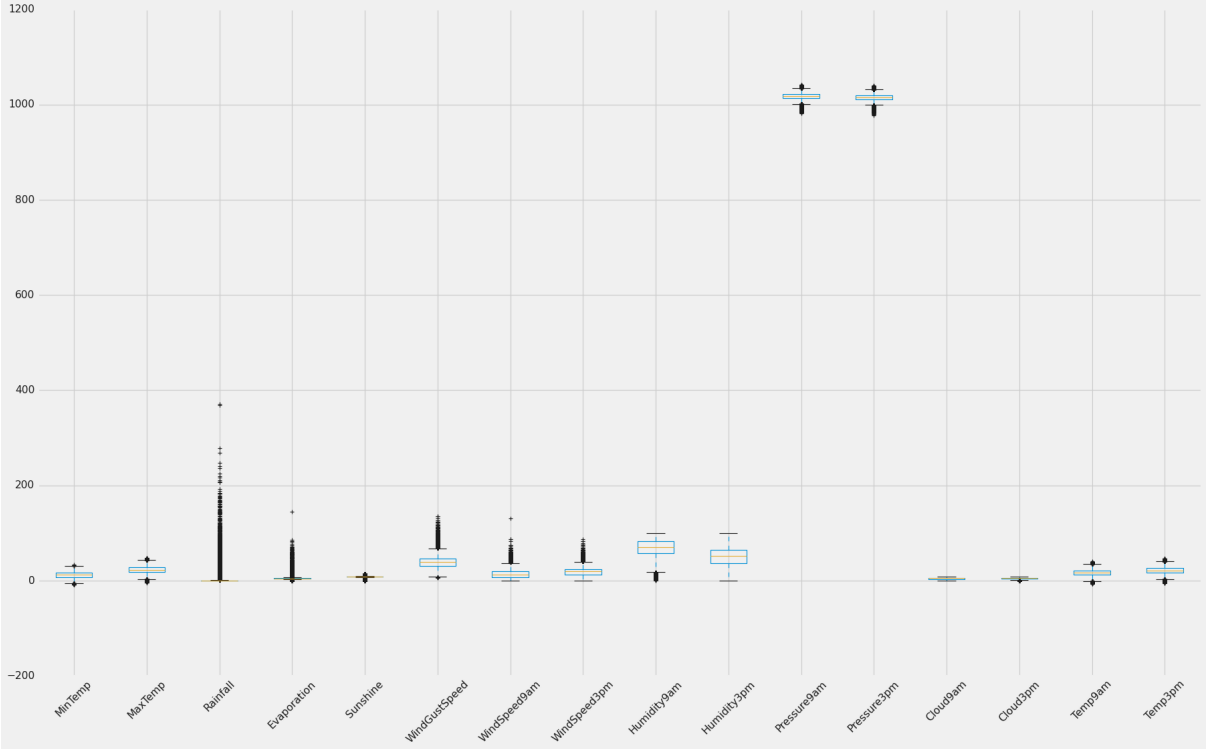
Com hem vist, tot i tenir dades correlacionades hi ha altres aspectes que dificulten la feina.

2. Preprocessing (normalitzation, outlier removal, feature selection..)

Normalització

Un dels primers passos abans de començar a entrenar models és entendre amb quines dades tractem.

Per assegurar-nos que tots els atributs comencen amb un mateix pes durant l'entrenament, cal tenir unes dades escalades. Utilitzant un boxplot es pot veure fàcilment la distribució de les dades, els seus valors i identificar els atributs amb outliers



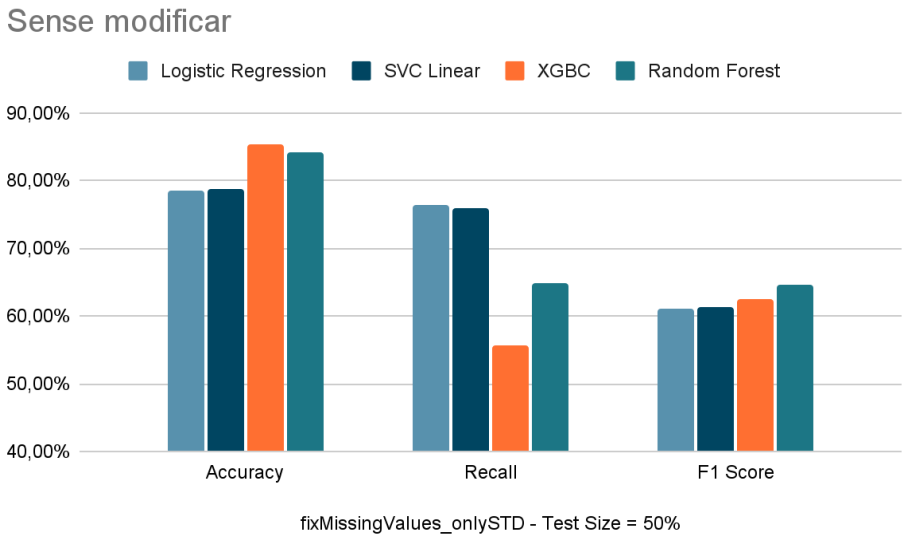
Clarament les no estan escalades. Per fer-ho hi ha dos opcions:

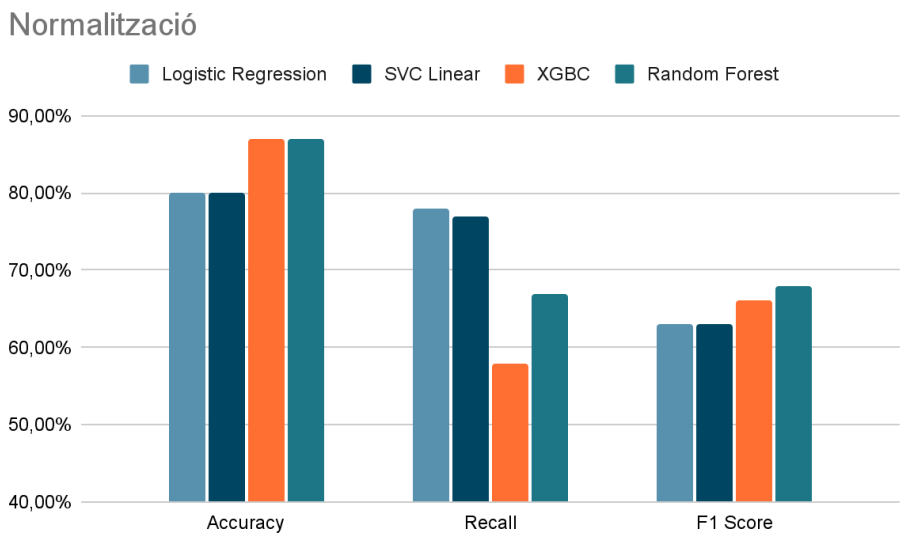
- Standarization: escalar les dades amb l'objectiu de tenir una mitjana de 0 i una desviació estàndard d' 1. Útil per la gran majoria de casos. És més fàcil seguir identificant els outliers.

- Normalization: escalar les dades amb l'objectiu de tenir-les totes entre l'interval [0,1]. Útil quan necessitem que els paràmetres tinguin valors positius.

Decidim fer-ho amb estandarditzación per mantenir valors positius. Són útils per aplicar, més endavant, un Chi-Squared Test.

Les següents imatges mostren resultats fent diferents reescalats a les dades.

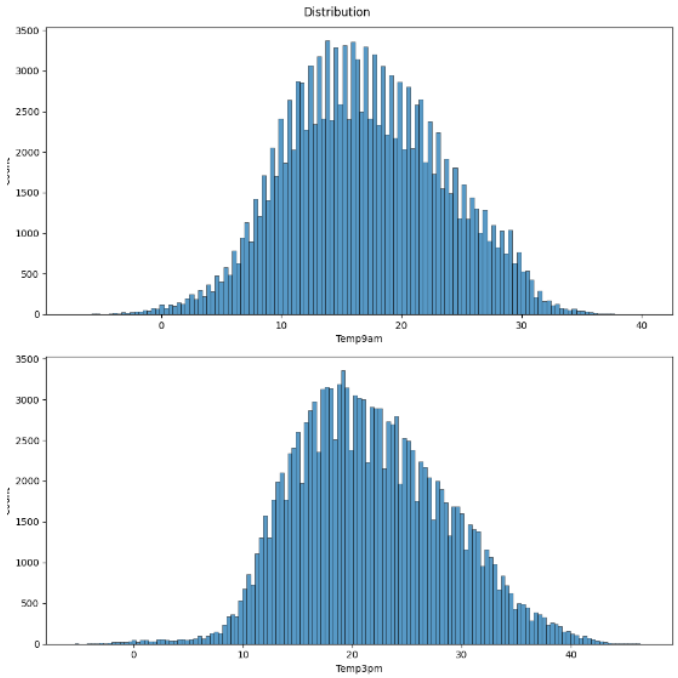




Aplicant normalització a les dades s'aconsegueix millorar les mètriques aproximadament entre un 1% i 4%. La raó que passi això és que tots els atributs parteixen d'un mateix valor pel pes. Veient el Boxplot anterior, molts atributs ja estaven dins d'uns rangs similars, només alguns quedaven lluny d'aquests valors. Segurament, si es tingués atributs més diversos, la millora seria encara més bona.

També es pot observar que s'aconsegueix una millora en els temps d'execució. Llavors una normalització, no només millora les mètriques, sinó que també ajuda a reduir el temps de còmput dels models. La raó que fa que sigui més ràpid és que es redueixen les probabilitats que els models es quedin estancats en un òptim local.

A l'hora de normalitzar, hem provat d'aplicar només aquelles dades que siguin asimètriques, és a dir, que no tinguin una distribució gaussiana.



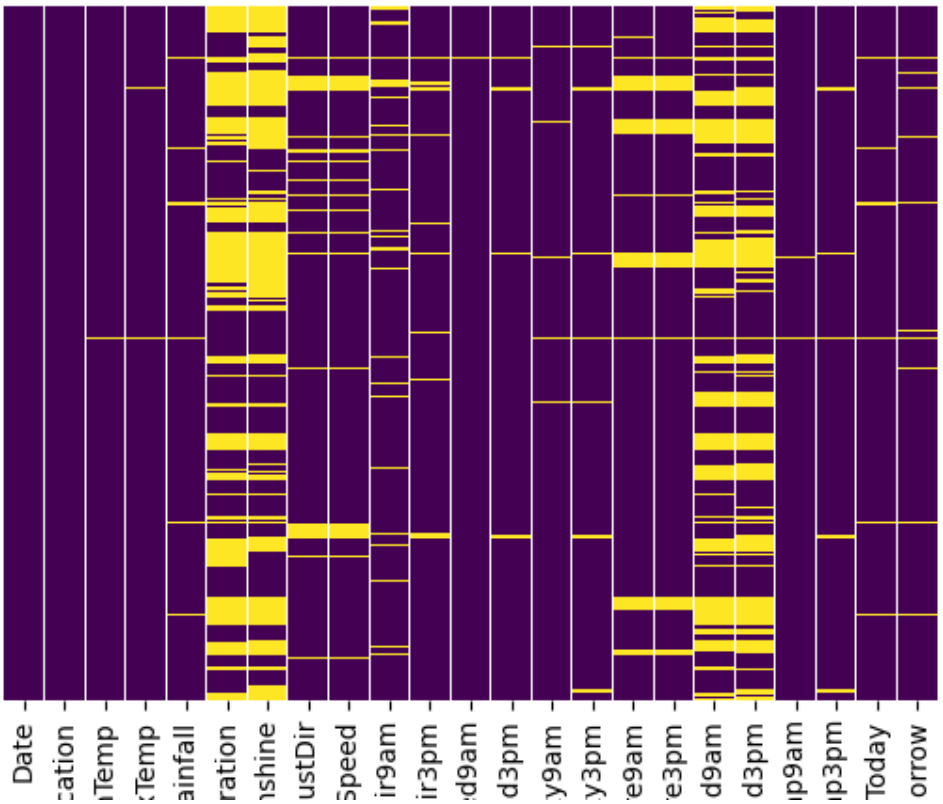
Per veure si les dades són asimètriques, podem utilitzar Skew. És una fórmula que diu quant d'asimètric és un conjunt de dades. Si és un nombre negatiu, vol dir que els valors per sota de la mitjana estan més dispersos. Si és positiu, llavors els valors que estiguin per sobre de la mitjana seran més dispersos. Un cop sabem els atributs no gaussians, podem fer la transformació a aquelles variables per normalitzar-les amb la llibreria PowerTransformer de sklearn.

Aquesta prova no ha donat tant bon resultat, ni en velocitat ni en mètriques. Això passa pel fet que no totes les dades es transformen. Com que tenen una distribució gaussiana, al fer diferents transformacions, les dades no queden ben compensades entre elles..

Valors nuls

Un punt igual d'important és tenir les dades completes, cal assegurar-se que no hi hagi valors nuls, i si hi ha, posar-hi remei.

La següent imatge mostra en groc, per cada atribut, els valors que són nuls. Una primera solució és simplement ignorar els atributs amb gran quantitat de valors nuls, però ràpidament ens hem adonat que teniem una gran pèrdua d'informació.



Així doncs, per fer front al problema hem aplicat dos tècniques depenent del tipus d'atribut:

- Valors numèrics continus: assignar la mitjana de l'atribut
- Valors categòrics: assignar la moda.

Codificació de dades categòriques

Altres que dades que també hem de modificar són les dades categòriques, com per exemple la localització o les direccions del vent. La gran majoria d'algorisme de Machine Learning funcionen millor quan les dades són numèriques. Per "discretitzar-les" podem utilitzar diferents Encoders, com per exemple l'OrdinalEncoder o el OneHotEncoder. Però nosaltres fem servir el LabelEncoder.
L'OrdinalEncoder i el LabelEncoder fan pràcticament el mateix, que és assignar un valor numèric a cada un dels diferents valors categòrics que podia tenir l'atribut. Però el OneHotEncoder el que fa és crear noves taules, una per cada valor possible de l'atribut, i els emplena amb 0 i 1 depenen de si pertany o no a una categoria.

Quan les dades categòriques tenen una relació entre si, com per exemple els nivells de dificultat d'un joc, ens interessa fer ús del LabelEncoder per fer que els nivells de dificultat passin de [fàcil, intermedi, difícil] a [0, 1, 2]. Però quan volem discretitzar dades que no tenen relació entre elles com els noms d'uns països o ciutats, el millor és fer servir el OneHotEncoder, que converteix la taula [França, Itàlia, Alemanya] en 3 taules [1,0,0] [0,1,0] [0,0,1].

LabelEncoder

Country	Age	Salary	Purchased
France	44	72000	0
Spain	27	48000	1
Germany	30	54000	0
Spain	38	61000	0
Germany	40	nan	1
France	35	58000	1
Spain	nan	52000	0
France	48	79000	1
Germany	50	83000	0
France	37	67000	1

```
array([[0, 44, 72000, 0],
       [2, 27, 48000, 1],
       [1, 30, 54000, 0],
       [2, 38, 61000, 0],
       [1, 40, 61000, 1],
       [0, 35, 58000, 1],
       [2, 35, 52000, 0],
       [0, 48, 79000, 1],
       [1, 50, 83000, 0],
       [0, 37, 67000, 1]]
```

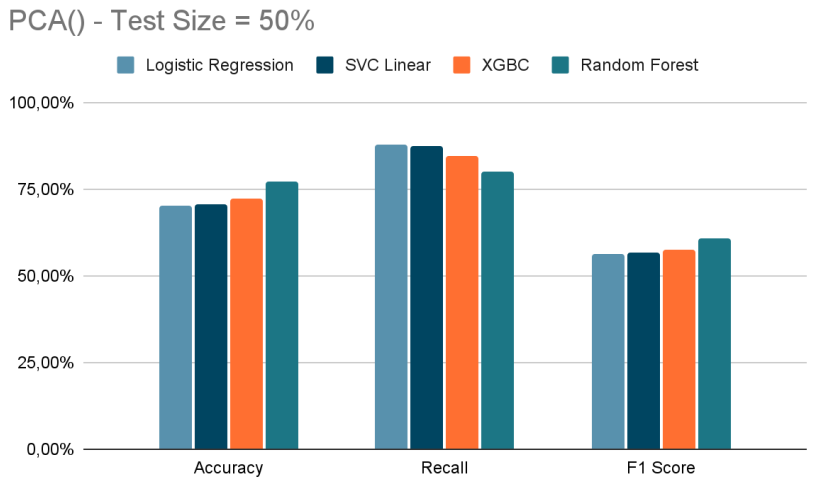
OneHotEncoder

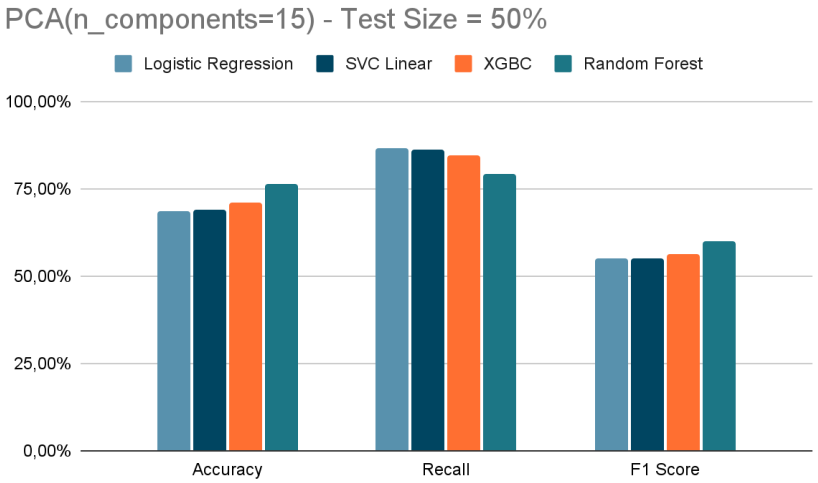
Country
France
Spain
Germany
Spain
Germany
France
Spain
France
Germany
France

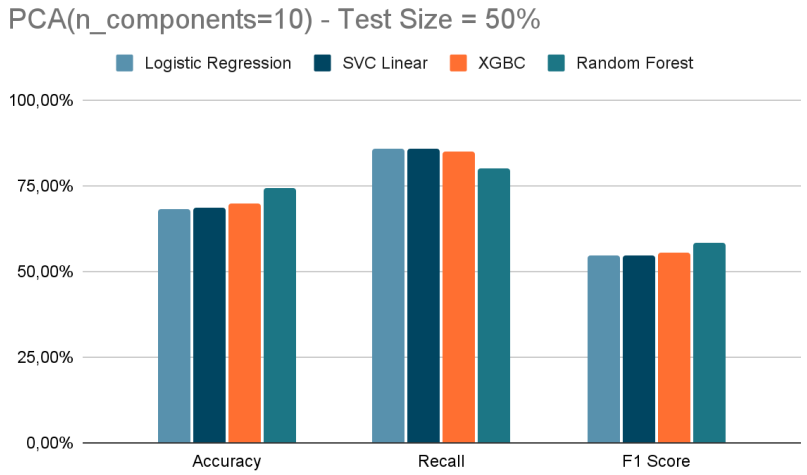
France	Germany	Spain
1	0	0
0	0	1
0	1	0
0	0	1
0	1	0
1	0	0
0	0	1
1	0	0
0	1	0
1	0	0

First Component Analysis

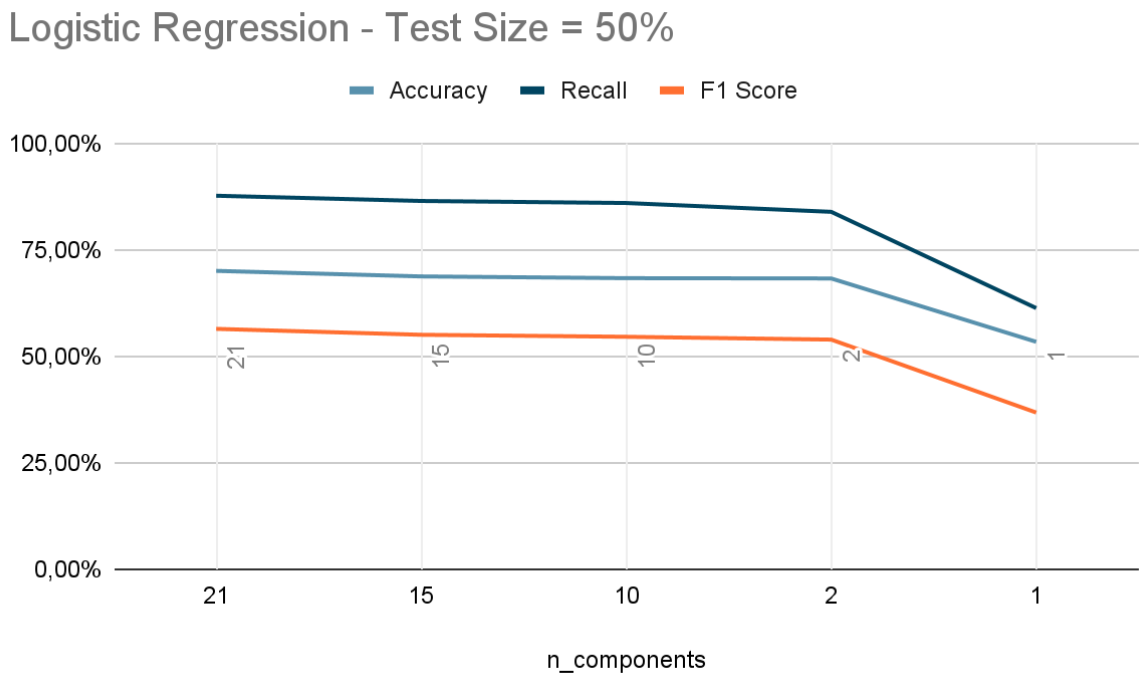
Fent un Principal Component Analysis (PCA) podem veure que no treu cap atribut, per tant, ens està dient que són necessaris tots els atributs per aconseguir una predicció correcta. Per comprovar que això és cert farem diferents proves amb valors diferents de n_components.





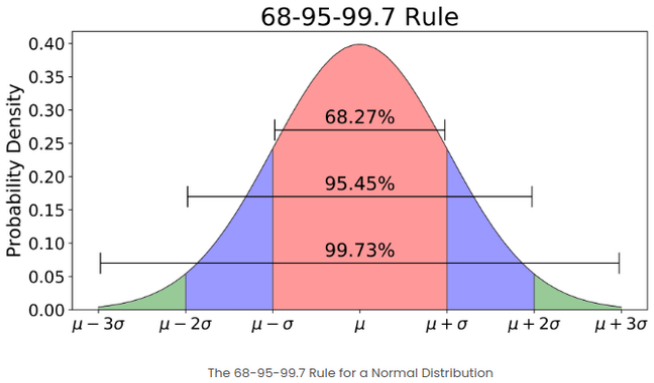


Podem observar que l'Accuracy i l'f1 score es veuen reduïdes, però aumentem el recall. Com més reduïm n_components, veiem que empitjores els valors d'Accuracy, f1 score i recall, però no gaire. Al següent gràfic podem veure com els valors es mantenen bastant semblants fins que arribem a n_components=1, on veiem que els valors es veuen reduïts.

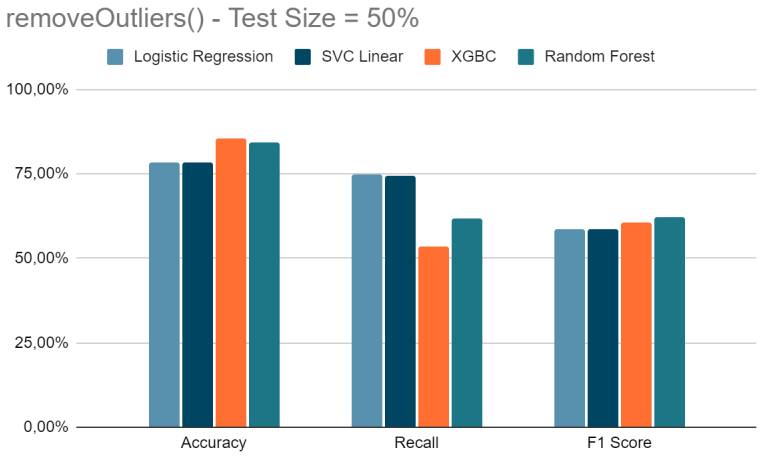


Anomalies

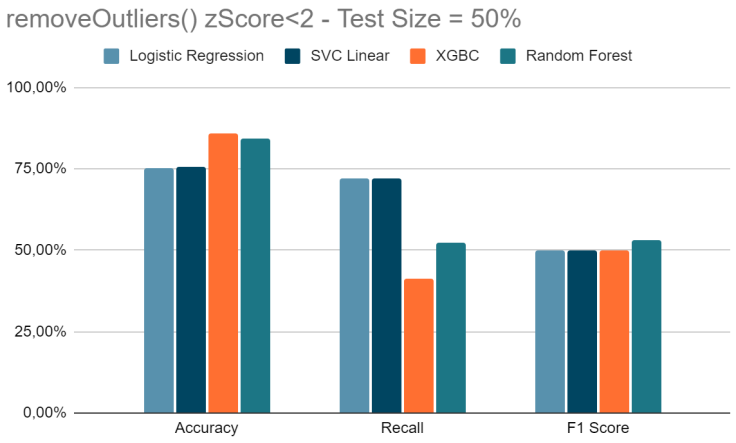
Una altra modificació a les dades que podem fer és l'OutlierRemoval, que consisteix a treure els Outliers, que són les dades que coneixem com anomalies, ja que se situen a la part més llunyana d'una distribució Gaussiana.



The 68-95-99.7 Rule for a Normal Distribution



En aquest gràfic podem observar que treient els valors amb una zScore superior a 3 realment no canvia els resultats. Ara comprovarem si reduint el llindar a 2 s'aconsegueix una millora.



En ningun dels dos casos s'ha aconseguit millora, realment els resultats són pitjors.

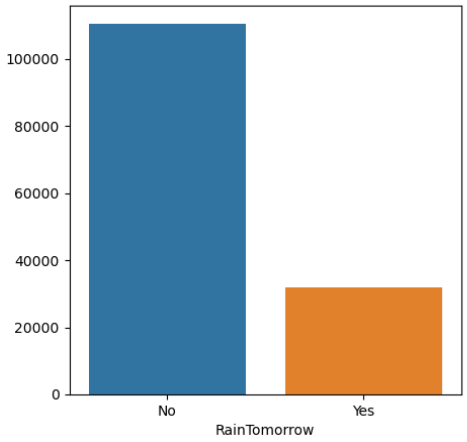
Finalment també s'ha decidit fer proves amb una SVM polinomial ja que té més flexibilitat a l'hora de definir fronteres. Els Outliers són especialment problemàtics a les màquines de vectors de suport pel fet que una sola mostra pot fer canviar totalment la funció generada.

A continuació es mostren 3 bancs de prova diferents per una SVM polinomial:

S'aprecia que el resultat millora, tant quan no tenim outliers, com quan es redueix el valor de la C.
Com ja s'ha comentat prèviament, el valor de la C permet més errors a les classificacions i, per tant, permet crear fronteres amb algunes mostres anòmales mal classificades.

Classes balancejades

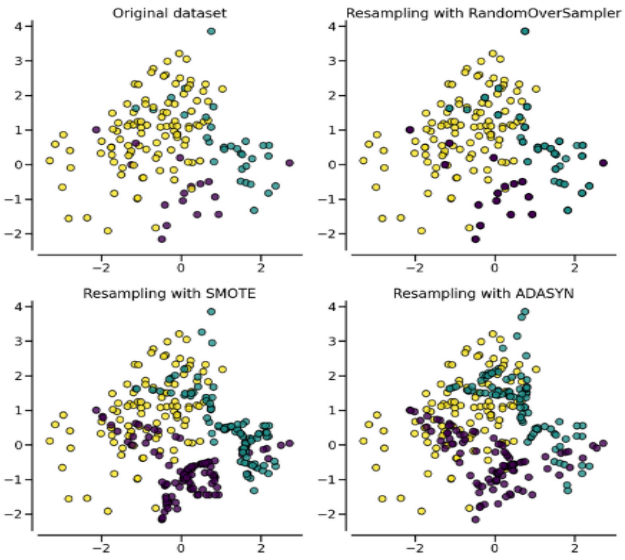
Ja prèviament comentat, quan volem predir si un event forma part d'una classe o d'una altre, hem de mirar que estiguin aquestes compensades. En el nostre cas, no ho estan gens.



Per fer front a aquest problema s'acostuma a generar noves dades basant-se en les que ja existeixen, amb la finalitat de balancejar ambdues classes.
Com podem veure en la gràfica estan molt descompensades la classe, per resoldre-ho hem utilitzat la llibreria SMOTE de Sklearn.

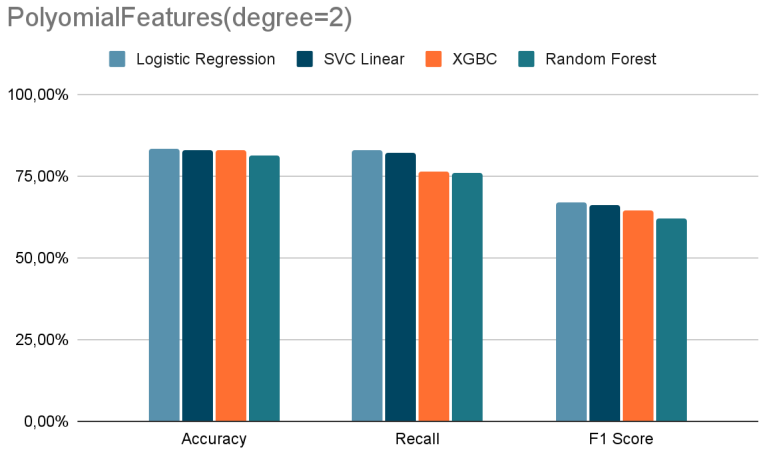
Cal deixar clar que només es pot aplicar a les dades Training, però mai a les dades Testing. Al cap i a la fi, quan sigui el moment de la veritat i s'hagi de fer prediccions reals, la distribució de les classes és la que és i no podem esperar que ens vingui balancejat.

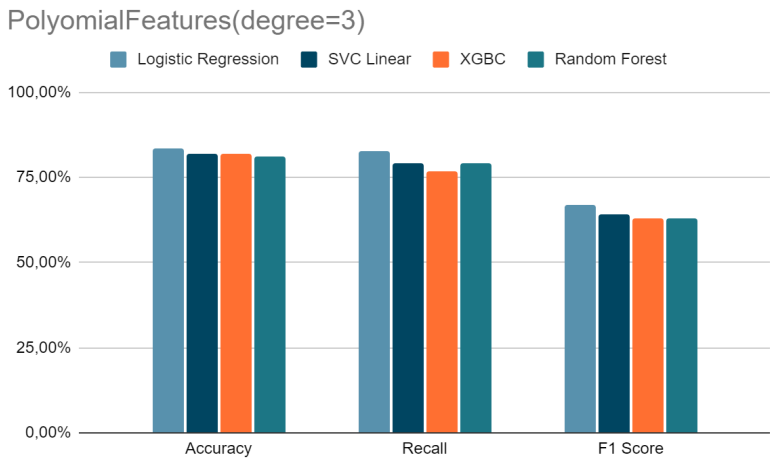
El fet de balancejar les dades no ha portat cap millora. Creies que, tot i no tenir la mateixa proporció de classes, tenim suficients dades per cada tipus. Ja hem vist prèviament que moltes es repeteixen. Així que crear noves dades que seran iguals a les que ja tenim no ajuda al resultat final.

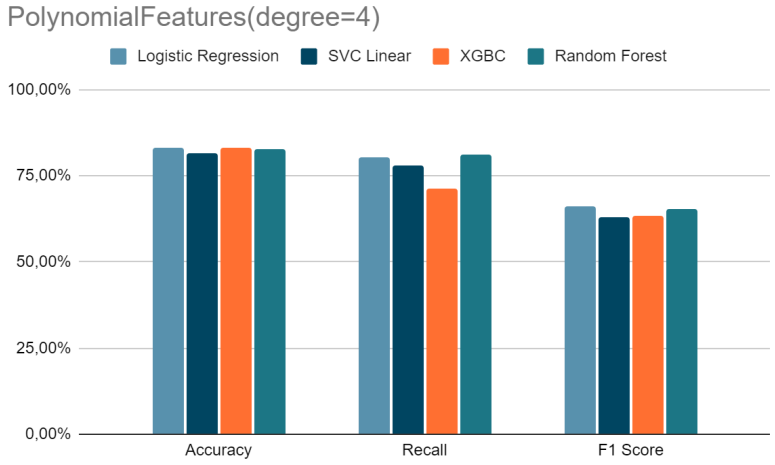


Polynomial Features

Com a última modificació provarem el Polynomial Features, que consisteix a crear noves columnes a partir de *features* que ja tenim elevant-los a un exponent.







Podem observar que respecte a l'execució normal hi ha una millora a l'f1 score del regresor logístic. Ja que afegir dades polinòmiques en un model lineal, li permet identificar patrons no lineals. Així i tot, en alguns models comporta un augment en el temps d'execució.

3. Model Selection

El nostre dataset té dues classes (plourà, no plourà), així que hem escollit models relacionats amb la predicció de categories, com:

- Regresor logístic
- SVM
- Random forest
- XGBoost
- SVM Lineal
- KNN

Regresor logístic

Metode de selecció	**Precicio(%) f1_score**	**C**	**fit_intercept**	**penalty**	**tol**
Básica					
80%(T) - 20%(V)	58.6	2	true	l2	0.001
70%(T) - 30%(V)	59.60	2	true	l2	0.001
60%(T) - 40%(V)	59.1	2	true	l2	0.001
50%(T) - 50%(V)	59.4	2	true	l2	0.001
K-Fold					
K=2	60.6	2	true	l2	0.001
K=3	60.8	2	true	l2	0.001
K=4	60.9	2	true	l2	0.001
K=5	61.2	2	true	l2	0.001
K=6	61.9	2	true	l2	0.001
CV	72.3	2	true	l2	0.001

SVM

Metode de selecció	Precicio(%) f1_score	C	kernel	max_iter	tol	random_state
Básica						
80%(T) - 20%(V)	38.8	2	rbf	500	0.0001	0
80%(T) - 20%(V)	46.5	2	poly	500	0.0001	0
80%(T) - 20%(V)	36.1	2	rbf	1000	0.0001	0
80%(T) - 20%(V)	45.1	2	poly	1000	0.0001	0
80%(T) - 20%(V)	44.4	0.2	rbf	500	0.0001	0
80%(T) - 20%(V)	42.7	0.2	poly	500	0.0001	0
70%(T) - 30%(V)	36.6	2	rbf	500	0.0001	0
70%(T) - 30%(V)	45.5	2	poly	500	0.0001	0
60%(T) - 40%(V)	43.2	2	rbf	500	0.0001	0
60%(T) - 40%(V)	45	2	poly	500	0.0001	0
50%(T) - 50%(V)	44.6	2	rbf	500	0.0001	0
50%(T) - 50%(V)	53.3	2	poly	500	0.0001	0
K-Fold						
K=2	48.3	2	poly	500	0.0001	0
K=3	49.7	2	poly	500	0.0001	0
K=4	51.3	2	poly	500	0.0001	0
K=5	53.2	2	poly	500	0.0001	0

K=6	51.7	2	poly	500	0.0001	0
CV	60.3	2	true	500	0.0001	0

A part d'aquest kernels hem utilitzats d'altre en l'apartat anterior

Random Forest

Metode de selecció	Precicio(%) f1_score	max_leaf_node	n_estimators	ccp_alpha	bootstrap	random_state
Básica						
80%(T) - 20%(V)	59	15	100	0.0	True	0
80%(T) - 20%(V)	36.1	15	100	0.7	True	0
70%(T) - 30%(V)	59.9	15	100	0.0	True	0
60%(T) - 40%(V)	59.6	15	100	0.0	True	0
50%(T) - 50%(V)	59.6	15	100	0.0	True	0
K-Fold						
K=2	60.2	15	100	0.0	True	0
K=3	59.9	15	100	0.0	True	0
K=4	61.3	15	100	0.0	True	0
K=5	61.8	15	100	0.0	True	0
K=6	62	15	100	0.0	True	0
CV	64.2	15	100	0.0	True	0

Observació: si no es fa la poda, cpp_alpha=0, triga bastant més temps. Ens sembla curiós perquè hem de tenir en compte que ha de mirar la probabilitat que podrà ser i decidir si posar-ho.

XGBoost

Metode de selecció	Precicio(%) f1_score	Objective	use_label_ encoder	n_estiators	gama	random_state
Básica						
80%(T) - 20%(V)	58.7	binary_ logistic	False	8	0.5	0
80%(T) - 20%(V)	59.3	binary_ logistic	False	5	0.5	0
80%(T) - 20%(V)	58.9	binary_ logistic	False	5	0.7	0
70%(T) - 30%(V)	59.1	binary_ logistic	False	5	0.5	0
60%(T) - 40%(V)	59.1	binary_ logistic	False	5	0.5	0
50%(T) - 50%(V)	58.7	binary_ logistic	False	5	0.5	0
K-Fold						
K=2	57	binary_ logistic	False	5	0.5	0
K=3	58.9	binary_ logistic	False	5	0.5	0
K=4	61.6	binary_ logistic	False	5	0.5	0
K=5	59.6	binary_ logistic	False	5	0.5	0
K=6	63.3	binary_ logistic	False	5	0.5	0
CV	72.4	binary_ logistic	False	5	0.5	0

Aquest és el més ràpid de tot amb diferència

SVM lineal

Metode de selecció	Precicio(%) f1_score	C	max_iter	penalty	random_state	tol
Básica						
80%(T) - 20%(V)	41.4	2	500	l2	0	0.0001
80%(T) - 20%(V)	12.3	2	1000	l2	0	0.0001
70%(T) - 30%(V)	1.1	2	500	l2	0	0.0001
60%(T) - 40%(V)	36.3	2	500	l2	0	0.0001
50%(T) - 50%(V)	44.6	2	500	l2	0	0.0001
K-Fold						
K=2	49	2	500	l2	0	0.0001
K=3	55.4	2	500	l2	0	0.0001
K=4	60.4	2	500	l2	0	0.0001
K=5	61.5	2	500	l2	0	0.0001
K=6	61.9	2	500	l2	0	0.0001
CV	61.6	2	500	l2	0	0.0001

knn

Metode de selecció	Precicio(%)	weights	n_neighbors	p
Básica				
80%(T) - 20%(V)	45.3	uniform	2	2
70%(T) - 30%(V)	46.2	uniform	2	2
60%(T) - 40%(V)	46.1	uniform	2	2
50%(T) - 50%(V)	43.2	uniform	2	2
K-Fold				
K=2	48	uniform	2	2
K=3	47.8	uniform	2	2
K=4	50.3	uniform	2	2
K=5	55	uniform	2	2
K=6	56.5	uniform	2	2
CV	61.7	uniform	2	2

Com podem veure en les taules tots els models milloren amb un cross-validation. Mirant els resultats, podem arribar a pensar que els millors models són la regressió logística i el boost classification. Entre aquest dos, si s'ha d'escollir un, agafaríem el boost classification pel fet que els temps d'execució són 3 vegades més ràpids.
Si fem la prova i unim aquests dos models, obtenim un model bastant ràpid en execució, però si fem la comprovació de si ha millorat, veurem que obtenim una mètrica pitjor, un 59.3 en F1_score.
Podem concloure, en aquest cas, que és millor utilitzar el XGBoost, que és un model ràpid i amb un f1-score lleugerament superior als altres.

Ensambles

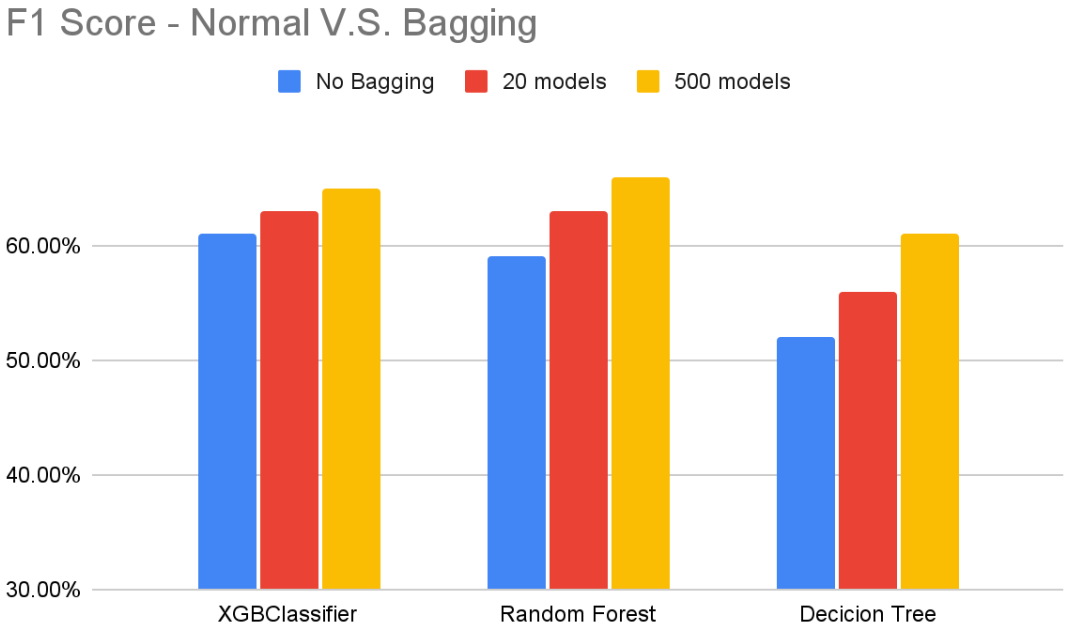
Una tècnica ampliament utilitzada és la de crear Ensambles, o dit d'altra manera, un conjunt de models de Machine learning. Cada model té una predicció diferent, però entre ells es combinen per obtenir una única predicció. L'avantatge de combinar models diferents és que, al funcionar diferents, els seus errors es compensen entre ells.

Pel nostre dataset es fa ús del Bagging. Es basa en combinar varis models i, per aconseguir que es corregeixin els errors els uns als altres, s'entrenen amb subconjunts diferents del conjunt Training. Els resultats es fan per votació, triant com a resultat final el que s'ha repetit més vegades. Al entrenar els models paral.lelment, aquests són independets entre ells i la combinació d'ells dona un resultat més robuts.

Cal deixar clar que per tal de veure el comportament i si hi ha millora, les comparacions es faran amb models bàsics, sense paràmetres òptims que sabem que milloren les prediccions. Tampoc s'ha fet un previ EDA per eliminar atributs no útils, eliminar Outliers, etc…

Per fer les proves s'ha fet dos baggins diferents per cada model. El primer tipus és amb 20 models, mentre que al segon tipus s'entrenen 500 models. Si bé en ambdós casos millora el resultat, no són dos execucions que es comportin igual.

Amb el baggins que s'entrenen 20 models, es veu una millora bastant considerable. Tot i no millorar molt en termes de mètriques, si es posa en consideració que el temps d'execució només incrementa uns pocs segons, s'aconsegueix millorar el resultat final sense ningun tipus de cost.
El segon bagging que s'ha provat entrena 500 models. Tot fa pensar que, basant-nos en el resultat anterior, amb aquest nombre tant elevat de models el F1 Score millorarà fins a percentatges a prop del 100%. No és així, com més models crea, el percentatge millora però de forma logarítmica. Malgrat tenir millores mètriques, no considerem que sigui un resultat final més bo pel que el temps d'execució és superior als 10 minuts.



4. Cross Validation

Importància de cross validar

Quan s'entrena un model sense tenir en compte la variació de les dades, com estan ordenades, si estan balancejades o no, és molt possible que els resultats obtinguts no representin realment al comportament que tindrà el predictor a l'hora de la veritat.

Hi ha, doncs, una tècnica anomenada Cross Validation que permet avaluar de forma més òptima i fidel a la realitat un conjunt de dades, evitant així generalització quan s'entrena sobre un sol conjunt.
Es basa en entrenar diversos models en subconjunts de les dades disponibles i avalua cada un d'ells en el subconjunt complementari restant. Mitjançant aquesta tècnica, hi ha moltes probabilitats que puguem detectar overfitting amb facilitat.

Proporció Train-Test

Un altre aspecte molt important a tenir en compte és la mida que es dona al conjunt Test i al Train. Normalment s'utilitza un 80% de les mostres per Training i un 20% per Testing.

Hi ha situacións, però, que poden fer canviar aquestes proporcions:

- Cost computacional durant l'entrenament del model
- Cost computacional durant l'avaluació del model
- Representativitat del conjunt Trainig
- Representativitat del conjunt Test

Com a norma general, com més mostres tingui un model per entrenar, millor podrà fer les prediccions. Tot i això, cal deixar una mostra considerable al Testing per assegurar-nos que es posarana prova la gran majoria de casos possibles que podria trobar-se un cop estigui implementat i en funcionament. D'altre manera, si no posem a prova suficients mostres, pot semblar que fa bones prediccions però realment no s'està entrant com a input tots els cassos reals.
Per altra banda, un conjunt de Training molt petit no tindrà suficients mostres per poder entendre i aprendre de totes les possibles entrades.

El nostre dataset conté 150000 registres aproximadament, una suma bastant considerable tenint en compte del que tracta. Aquest fet porta a una situació on hi ha molts registres similars. Com ja s'ha vist a l'apartat A, no hem obtingut diferència quan la mida del Traing era 80% o 10%. No ha sigut fins que hem posat una mida de Training de 95% que s'ha vist un empitjorament a les prediccions.

K-Fold Cross Validation

Una manera d'aplicar el Cross Validation és la coneguda per K-Fold Cross Validation. És la més popular d'entre elles i acostuma a donar un resultat bastant acceptable per la gran majoria de casos.

Abans de començar a explicar la tècnica, cal tenir clar una regla molt important. Mai s'ha de mesclar les dades Training i Test. Per tant, el pas previ és isolar les dades de Test i només utilitzar-les per l'avaluació final. Les dades que resten a Training seran les usades per aplicar el F-Fold Cross Validation.

Es comença partint les dades de forma aleatoria en K subconjunts iguals. El primer subconjunt és l'utilitzat per Testing, i els K-1 restants són utilitzats per entrenar el model (Training). Finalment el model és posat a prova amb les dades Testing.
Aquest procés es repeteix K vegades, cada cop emprant un subconjunt diferent per Testing. D'aquesta manera cada subconjunt té la mateixa oportunitat de ser inclòs al subconjunt d'entrenament.

El valor de K acostuma a ser entre 3 i 5 per la majoria de casos. Si més no, pot ser estès fins a valor com 10 0 15, tenint com a punt negatiu que el cost computacional incrementa considerablement.

Per la mateixa raó esmentada anteriorment (multitud de mostres molt semblants), aplicar un K-Fold no fa que canviï significativament el resultat dels models.

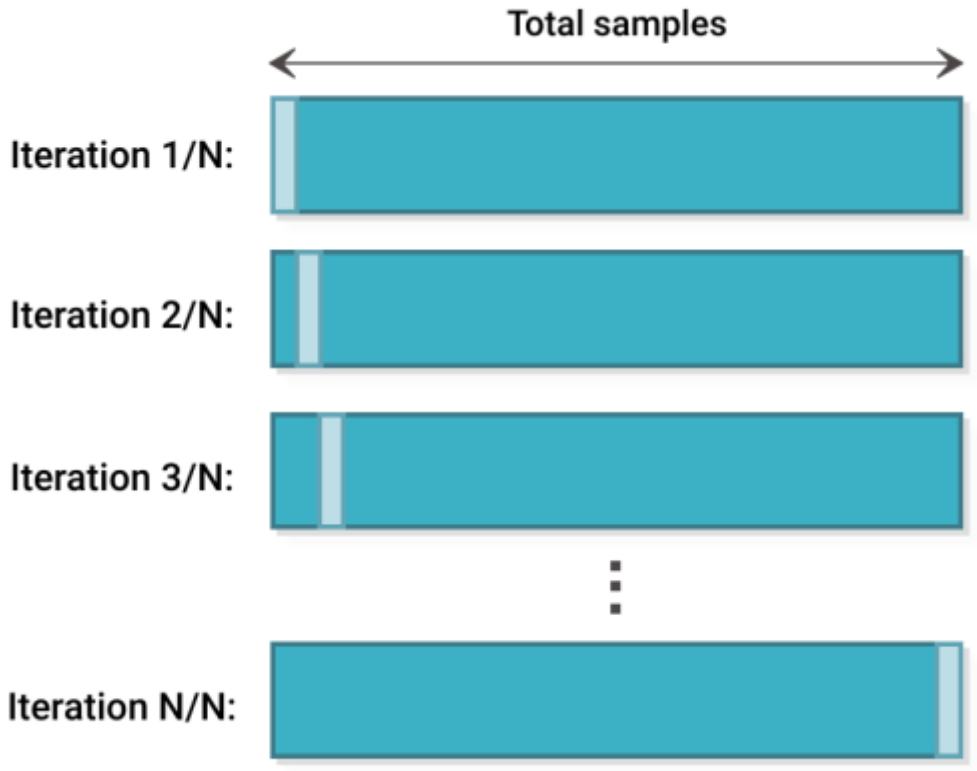
Leave One Out

Una altra manera d'aplicar el Cross Validation és utilitzar el mètode Leave One Out (LVOCV). És un cas extrem del K-Fold on K és igual a N, sent N la mida del conjunt de dades.

Com a punt a favor d'aquest mètode cal dir que no malbarata dades. Només s'utilitza una mostra del dataset com a Testing, i la resta pel Training.
Té un gran desavantatge respecte al K-Fold, el cost computacional. Mentre que al K-Fold genera K models, el LOOCV en genera N.

Aquest mètode, per tant, només s'ha de fer servir quan la mida de les mostres és petita o quan es doni molta més importància a unes bones prediccions que al temps d'execució.

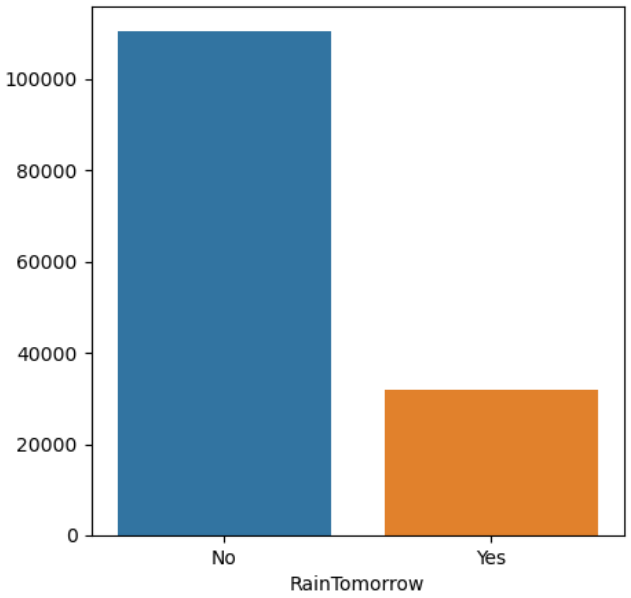
Al nostre dataset aquest tipus de Cross Validation no és, útil ja que a moltes iteracions s'estaria agafant pel Testing dades similars i, més important, el temps d'execució és massa elevat.



5. Metric Analysis

Mètriques disponibles

El dataset amb el que s'està treballant està considerablement desbalancejat. De les 2 classes que hi ha, aproximadament un 75% dels registres depèn a la primera i el 25% restant a la segona.



Utilitzar la mètrica accuracy no és recomanable per aquestes situacions, ja que pot portar a males interpretacions del resultat. Si un model prediu que l'endemà mai serà plujós, estaríem davant d'una accuracy bastant elevada, doncs encertaria pel 75% dels casos. Tanmateix, el model seria incapaç de preveure quan ha de ploure l'endemà.
No és només el fet que no pugui fer amb certesa prediccions per les dues classes, sinó que la més important (dia plujós) és la que fa malament.

Una altra mètrica que, tot i ser més adequada quan tenim dades no balancejades, en aquest cas no seria de gaire ajuda és Precision. Aquesta dada diu de totes les prediccions marcades com a "Yes", quantes s'han fet correctament. La raó per la qual no ens és d'ajuda és que, segons el nostre criteri, és més valuós estar segur que si diem no plourà, realment no plourà. Així doncs, no ens interessa tant si predim que un dia plourà, però realment fa sol. Per exemple, si una persona vols sortir a caminar i veu que el predictor prediu sol, sortirà a caminar. El problema se'l trobarà quan a mig passeig comenci a ploure.

La tercera mètrica a parlar és el Recall. Aquesta ens permet saber de tots els dies que plourà, quants s'han predit correctament. Creiem que és la més important ja que si una persona fa plans per l'endemà, és molt més important saber amb certesa si plourà. Seguint l'exemple anterior, si una persona mira el predictor i veu que plourà, ajornarà el passeig per un altre dia. Si al final resulta que no plou, tampoc té res a perdre.

Per últim, la mètrica F1 score és la que per la majoria de casos ens serà més útil. Aquesta mètrica es basa en la mitjana harmònica del Precision i del Recall. Així aconseguim que només sigui alta si les altres 2 mètriques són altes. Per exemple, si Precision és 1 i Recall és 0.01, el F1 score és 0.02. En dades no balancejades, aquesta mètrica és molt important i sempre substituirà al accuracy. Si donguèssim la mateixa importància a ambdues classes, llavors utilitzaríem F1 score.

Corbes PR i ROC

Un altre tipus de mètrica que es pot trobar són les corbes Precission-Recall (PR) i ROC.

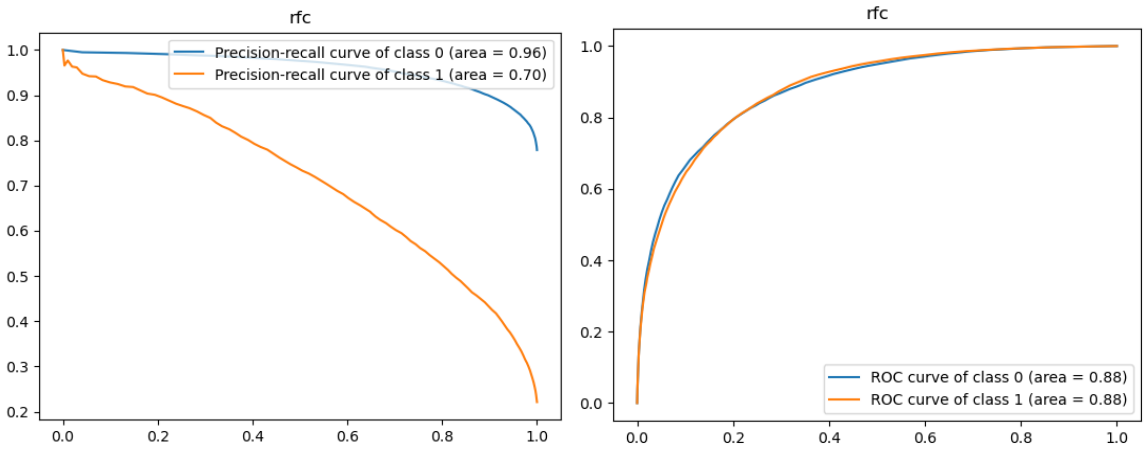
La PR permet veure la relació entre la precisió (positius ben predits sobre el total de positius predits) i el recall (positius ben predits sobre positius reals).
La ROC permet veure la relació entre recall (positius ben predits sobre positius reals) i fallout (falsos positius sobre negatius reals).

Ambdues corbes són aptes per la majoria de casos. Hi ha, si més no, situacions com la nostra on les dades no estan balancejades.
Fixant-nos més detalladament en el càlcul de la PR, tenim les dos fórmules següents:

- Precission = TP/(TP+FP)
- Recall = TP/(TP/FN)

Es pot apreciar com els True Negatives no apareixen per enlloc. Així doncs, la PR ens és molt més útil quan les dades estan balancejades tenint els casos positius com a minoria. De la mateixa manera, si els casos negatius són majoria, la PR no serviria.

Als dos gràfics inferiors es veu més visualment el resultat d'ambdues corbes. Mentre que la ROC sembla donar un bon resultat per les dos classes, la PR mostra la realitat. La classe 0 (més comuna) té bones prediccions i la classe 1 (minoritària) té pitjors prediccions.



Classification Report

SKlearn té implementada una eina anomenada “classification_report” que genera un informe amb les mètriques de les prediccions d’un model. La raó a ser utilitzada és que l’informe genera les mètriques per cada classe del conjunt dades.

Tal i com s’ha comentat anteriorment, el nostre model és capaç de preveure bé els dies que no plourà, però té problemes amb els que sí plourà. A la imatge inferior es veu la diferència a les mètriques entre les dos classes.

```
XGBC
Accuracy:  0.8103602364911315
Precision:  0.5566428302596914
Recall:  0.6911214953271028
f1 score:  0.616635397123202

Classification report:
              precision    recall  f1-score   support

           0       0.91      0.84      0.87     22672
           1       0.56      0.69      0.62      6420

    accuracy                           0.81     29092
   macro avg       0.73      0.77      0.75     29092
weighted avg       0.83      0.81      0.82     29092
```

Com a punts a destacar, trobem dos tipus de mitjanes:

- Weighted: la mitjana està calculada de forma proporcional a les classes que tenim:
  mitjanaClasse1 * P(classe1) + mitjanaClasse2 * P(classe2)

- Macro: la mitjana no està calculada de forma proporcional a les classes, es fa una simple mitjana aritmètica:
  mitjanaClasse1 * 0.5 + mitjanaClasse2 * 0.5

6. Hyperparameter Search

Com ja s'ha vist durant tota la pràctica, el tipus de model no és el que decidirà la qualitat de les prediccions, sinó els paràmetres que s'utilitzen a cada model. Són els encarregats de modificar el comportament (velocitat, estructura, rendiment, prediccions) dels models.
Un bon exemple és el learning rate. Quan és molt gran, l'aprenentatge no és suficientment sensitiu. Quan és molt petit, l'aprenentatge és tant sensitiu que el temps d'execució pot resultar sent infinit.

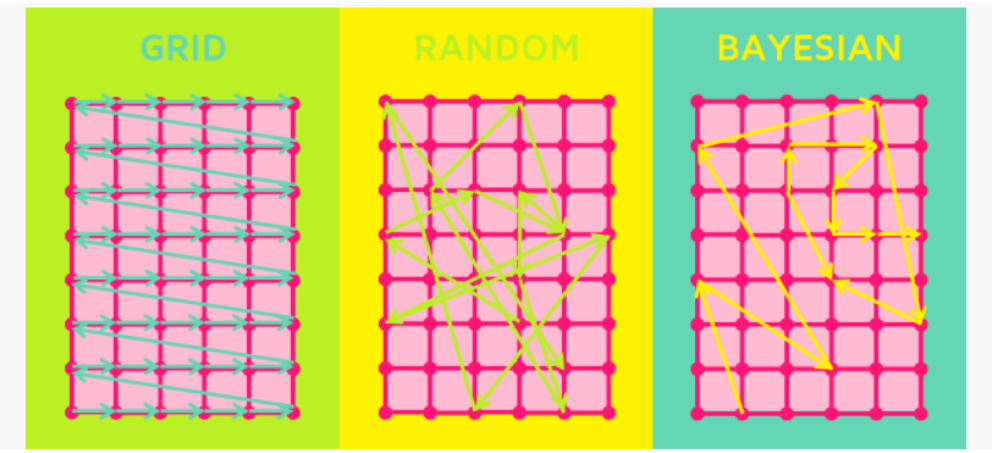
La tasca de trobar els paràmetres correctes és molt feixuga i comporta molt temps. Si bé, tenint un coneixement previ dels models, pot ser útil per aproximar el valor d'un paràmetre, la gran majoria de vegades es necessita recórrer a mètodes que provin un seguit de paràmetres, extreguin les mètriques de rendiment i es quedin amb el millor model.

Per aconseguir-ho, doncs, es pot aplicar les següents tècniques:

- Grid Search: És la manera més bàsica de fer "hyperparameter tuning", basant-se en fer totes les combinacions possibles d'un conjunt de paràmetres especificats prèviament.

- Random Search: En comptes d'especificar els valors que podrà agafar cada paràmetre, s'especifica un interval. Per cada iteració s'agafa un valor de l'interval de cada paràmetre i es fa la prova.

- Bayesian Search: Aquest mètodes es basa en un principi diferent. Cada iteració optimitza els valors dels paràmetres basant-se en amb les mètriques que l'anterior iteració. És útil quan es té un una gran quantitat de dades, l'aprenentatge és lent i es necessita minimitzar el "tuning time"



Quan es tenen recursos limitats, per exemple un ordinador personal, és recomanable provar el “Bayesian”. Al seleccionar els valors de forma informada prioritzant els que semblen més prometedors, aquest mètode pot trobar els millors paràmetres en poc temps. Els altres sí que poden arribar a un resultat més bo, però triguen molt més temps.

Hi ha, també, llibreries que permeten facilitar encara més trobar els paràmetres adequats. Una molt utilitzada és l’anomenada “scikit-optimize” i es basa en la “Bayesian Optimization”

Pel nostre cas s’ha optat per buscar els paràmetres òptims pel model que fins ara ens havia donat millors resultats, el Random Forest Classifier.

Fent proves amb el Random Search i el Bayesian Optimization s’ha arribat al mateix resultat. La diferència, però, està en que el Random ha trigat tota una tarda a executar i el Bayesian poc més d’una hora.

Els paràmetres que permeten tenir els següent resultats són:

- n_estimators = 100
- min_samples_split = 6
- min_samples_leaf = 3
- max_depth = 4