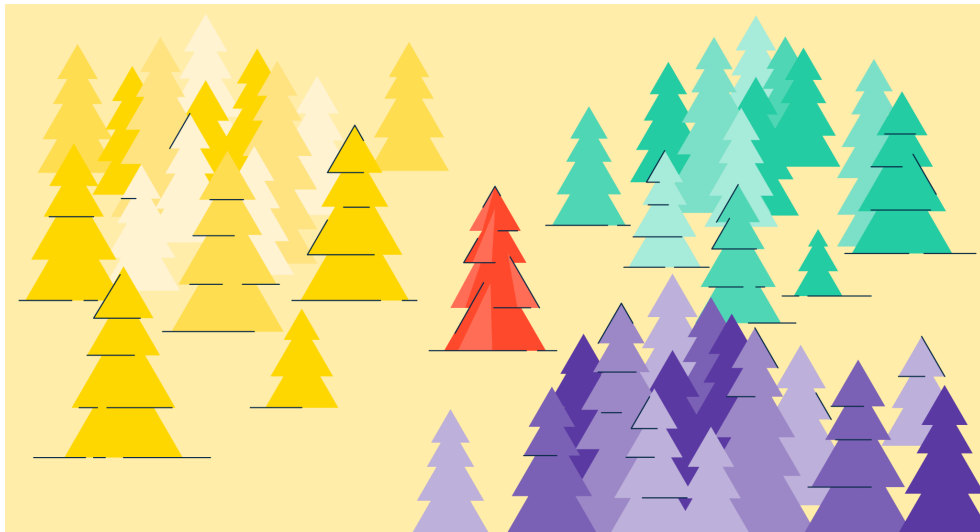


# Aprenentatge Computacional

## Practica 2: Classificació



Grup GPA603-1530

Mario González	1566235
Ferran Bernabé	1564845
Daniel Gutiérrez	1563389

## Apartat A: Comparativa de Models

En aquest apartat farem un primer anàlisi de la nostra base de dades, amb l'objectiu de mesurar els resultats d'aplicar els diferents models, mètriques i veure quin d'ells dona millor o pitjor rendiment, veurem mètriques com l'accuracy, i altres maneres de mesurar el rendiment del model com les corbes ROC i PR.

Finalment mostrarem una classificació del nostre model de forma gràfica.

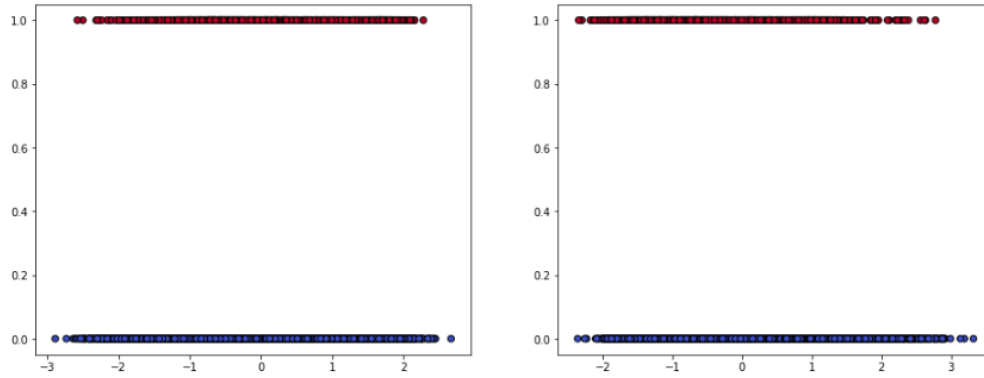
### Model selection

En quant a la selecció dels models a aplicar, hem utilitzat **Logistic Regression, SVM, KNN i Perceptron**. Hem entrenat aquests diferents models amb el conjunt de proporcions de test - validació proporcionats, amb els paràmetres per defecte, obtenint els següents resultats:

	<b>Logistic Regression</b>	<b>SVM</b>	<b>KNN</b>	<b>Perceptron</b>
<b>50% test 50% validation</b>	0.7892	0.792	0.7644	0.7636
<b>80% test 20% validation</b>	0.794	0.7895	0.772	0.7885
<b>70% test 30% validation</b>	0.7873	0.791	0.7616	0.6683

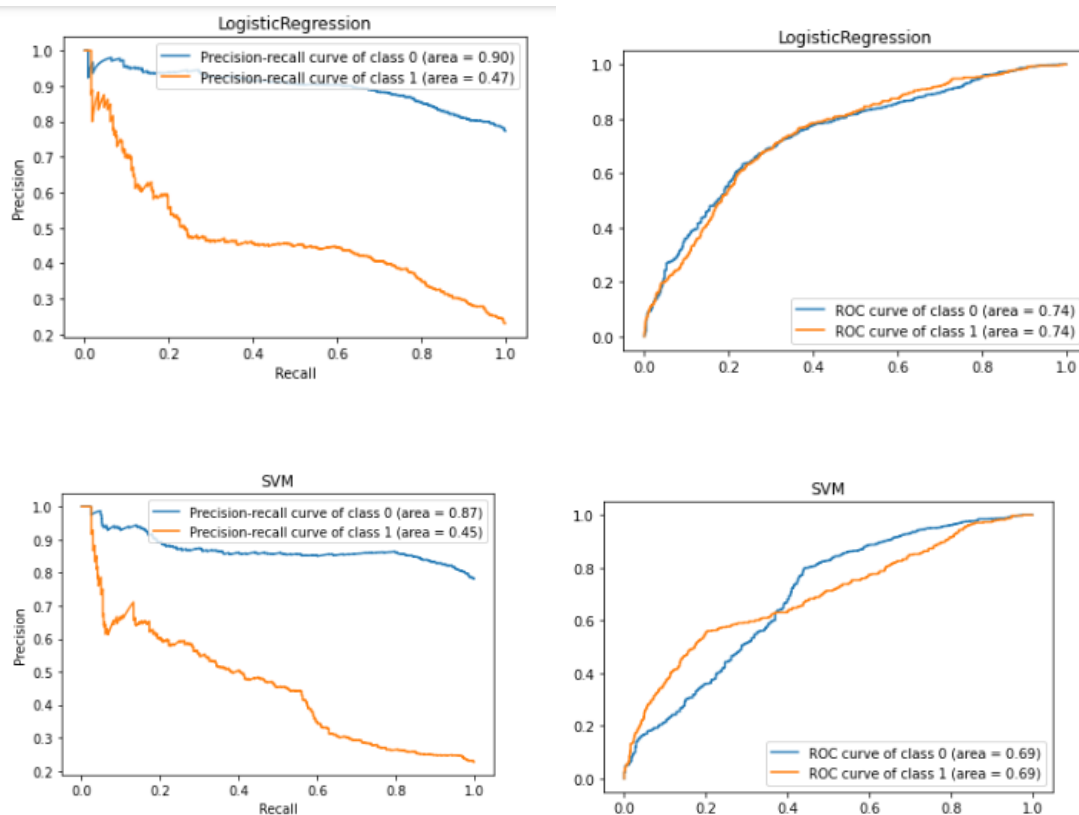
Podem observar que els resultats amb les tres distribucions diferents de conjunt de test i predicció obtenen resultats similars per a cadascun dels models, exceptuant el cas del Perceptró, on el conjunt amb 70% de test dona un resultat considerablement inferior. En quant als models, podem observar que tenen resultats similars, sent SVM el que dona un resultat més gran de mitjana, i el Perceptró el més petit en quant a la mitja de les tres distribucions.

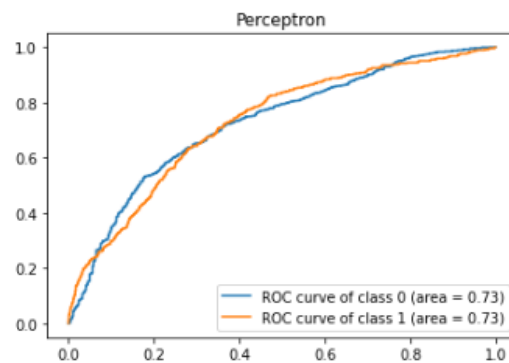
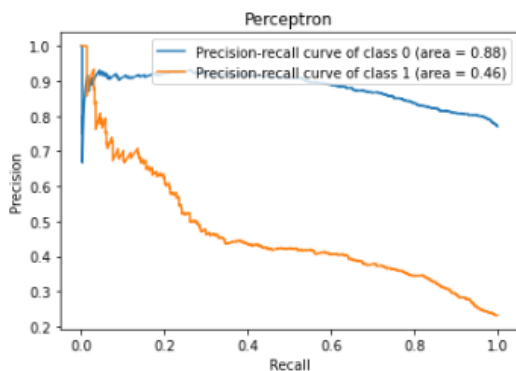
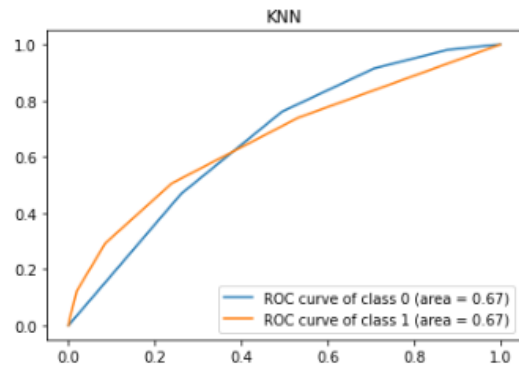
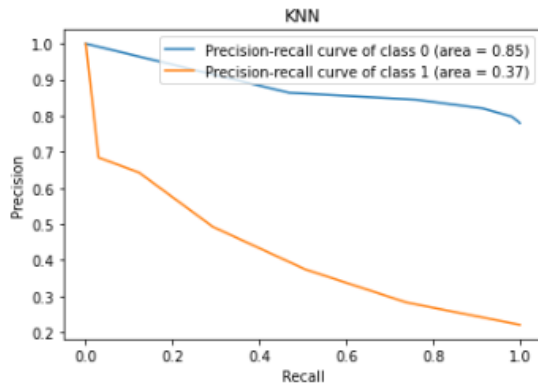
Les gràfiques executades en el primer apartat són les següents:



En aquest diagrama de punts podem observar la diferència entre les nostres dues classes. En el nostre cas, la nostra variable a classificar (y) era una **variable categòrica binària (Yes / No)**, la qual hem transformat a **1 i 0** respectivament per poder fer aquesta representació gràfica.

Les gràfiques resultants d'executar el segon apartat són les següents:



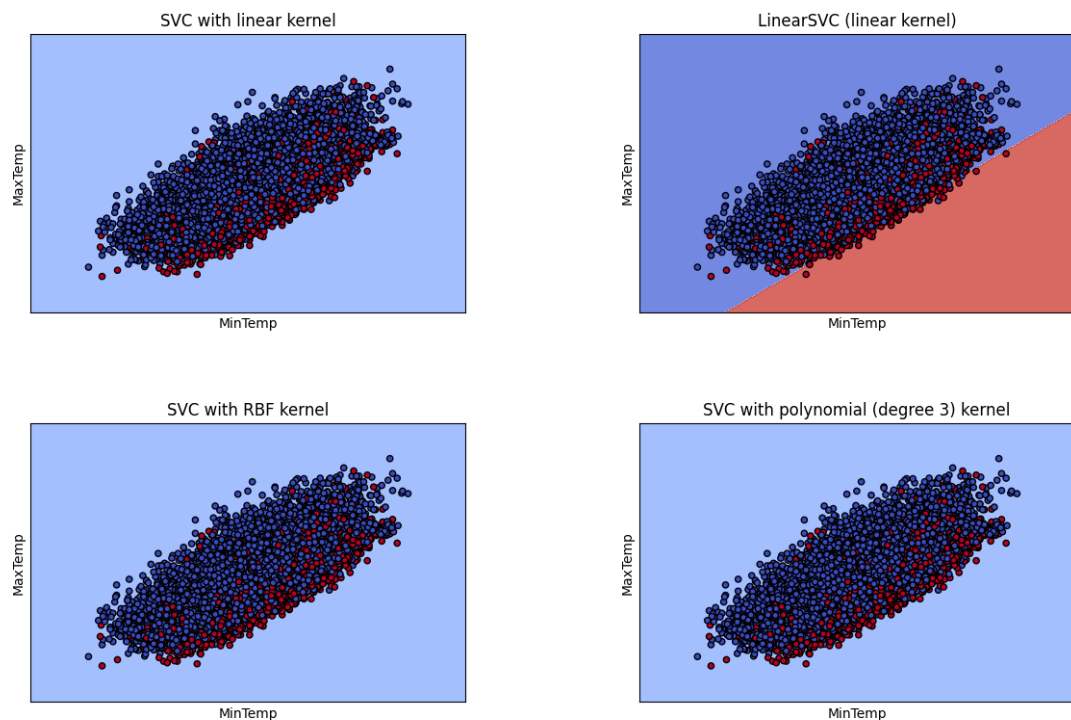


Aquestes corbes PR i ROC han estat generades amb el percentatge de training - validation que ens donava el millor resultat en el apartat anterior (**80% training, 20% validation**).

Com podem veure a les gràfiques PR i ROC dels quatre models analitzats, els millors en totes dues són Logístic Regression i Perceptron, obtenint una ROC Curve del 0.73 i 0.74, i una Precision-Recall Curve màxima de 0.88 i 0.90 per la classe que millor score ha obtingut.

Els altres models (KNN i SVM) també han obtingut valors de ROC i PR força alts, pero han estat una mica inferiors a aquests models.

El resultat d'executar l'apartat 3 és el següent:



Com podem veure cap d'aquests models classifica molt bé les nostres dades, pensem que es degut a que cap dels atributs escollits presenten cap diferència notoria i com podem veure a apartats següents, no hi ha cap atribut amb alta correlació, això significa que podem trobar qualsevol o qüasi qualsevol valor que es trobi tant en la classe **No** com a la classe **Yes**, i això fa una mica difícil aquesta classificació.

Tot i així, podem veure una tendència de que els valors **No** (blaus) ocupen la part superior esquerra del cluster de punts, mentre que els valors **Yes** (vermells), es troben a la part inferior dreta, per la qual cosa sí que hi ha una petita divisió entre aquestes classes.

## Apartat B: Classificació Numèrica

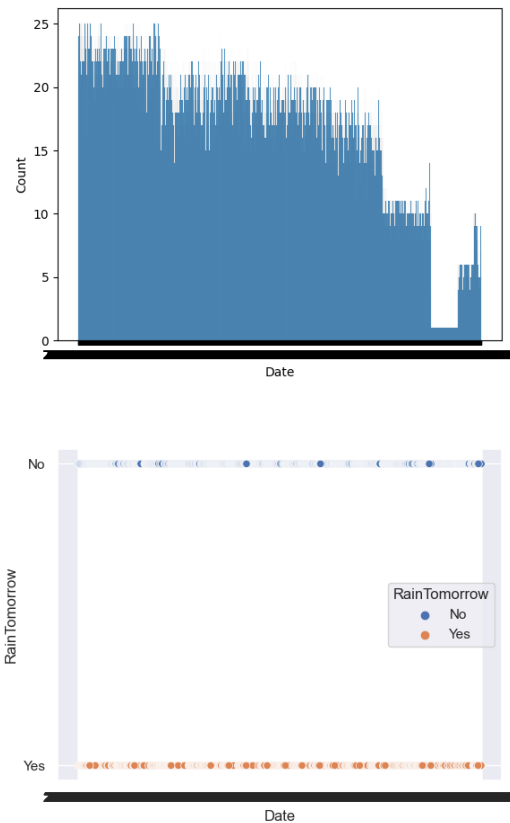
En aquest apartat de la pràctica analitzarem una base de dades amb l'objectiu d'aplicar diferents models de classificació i veure quin és més adient per classificar l'atribut objectiu en el nostre cas.

### 1. EDA (Exploratory Data Analysis)

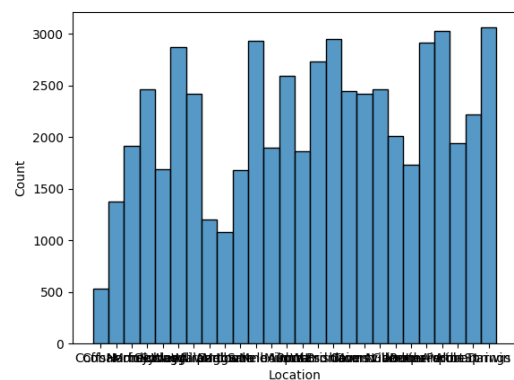
- **Quants atributs té la vostra base de dades? Quin tipus d'atributs tens? (Númeric, temporals, categòrics, binaris...)**

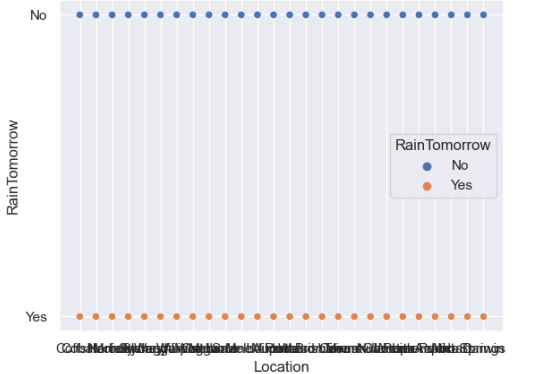
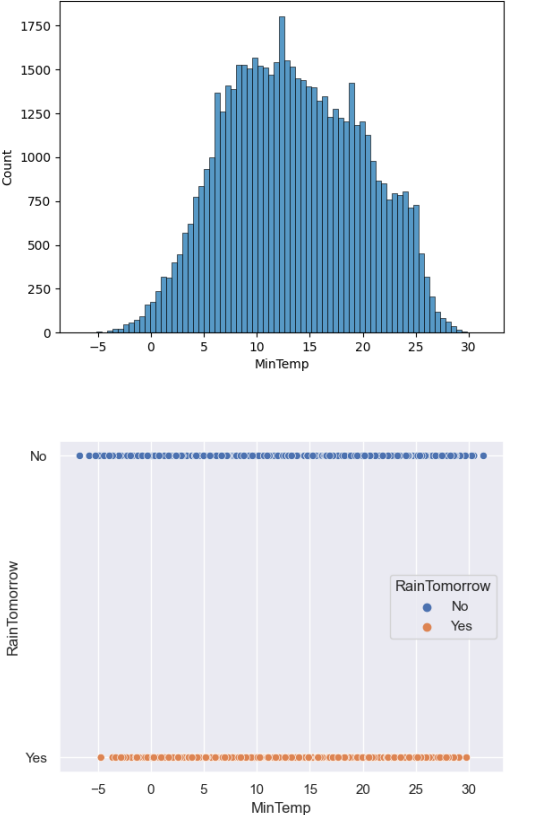
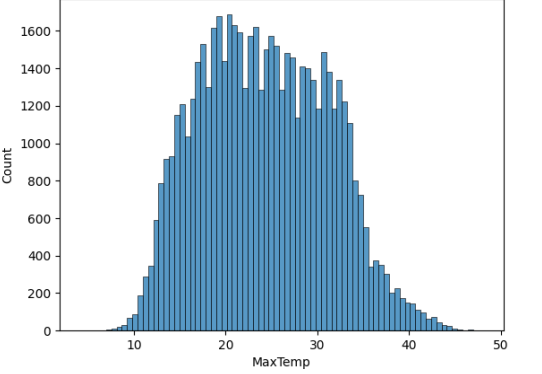
La nostra base de dades es una base de dades que disposa de dades de la climatologia a Australia. En el nostre cas, disposem de **23 atributs**:

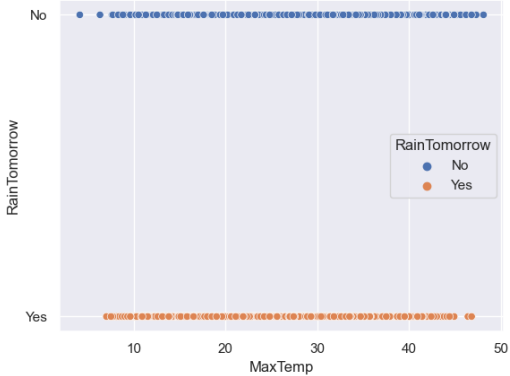
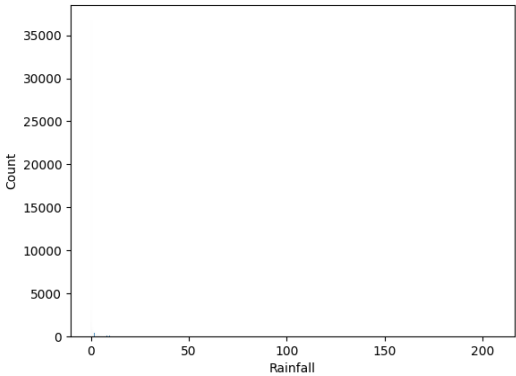
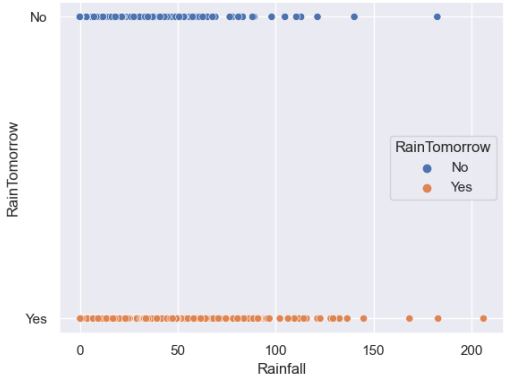
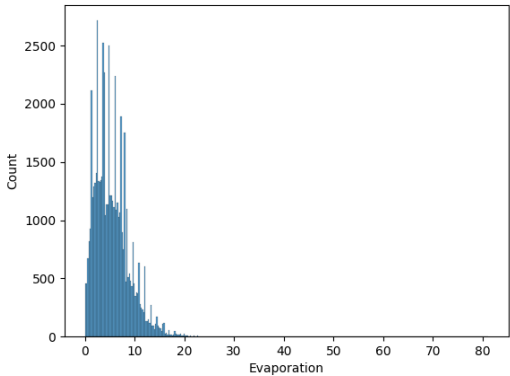
**Date:** atribut temporal, es la data a la que es va mesurar aquest conjunt de dades climatologiques (**date**)



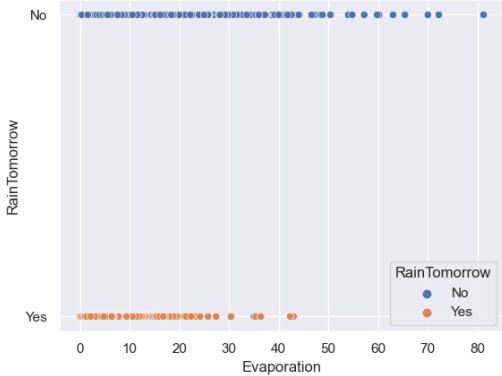
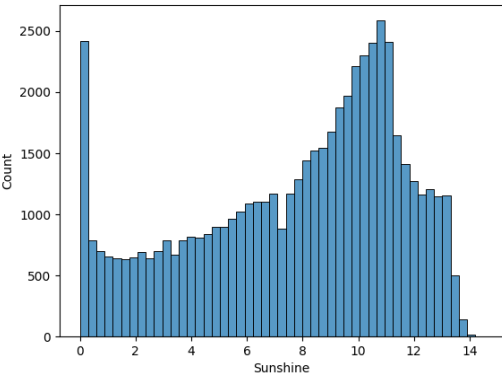
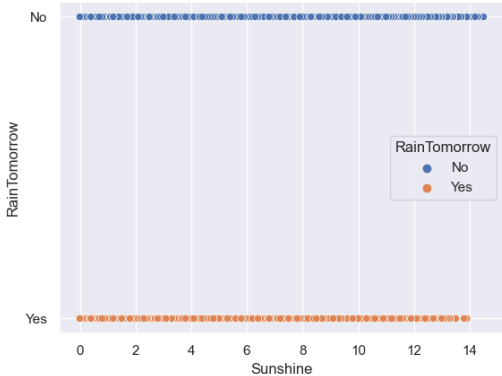
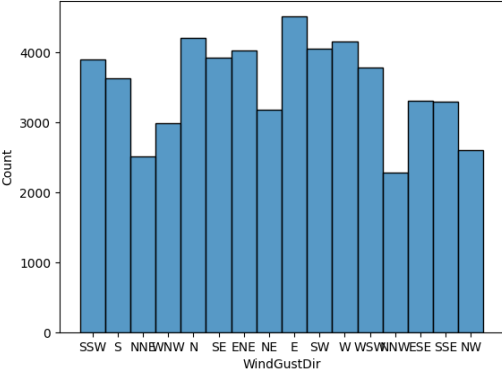
**Location:** atribut referent a la localització de les mesures climatologiques (**string**)

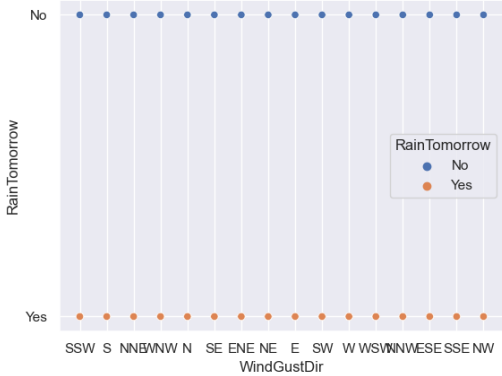
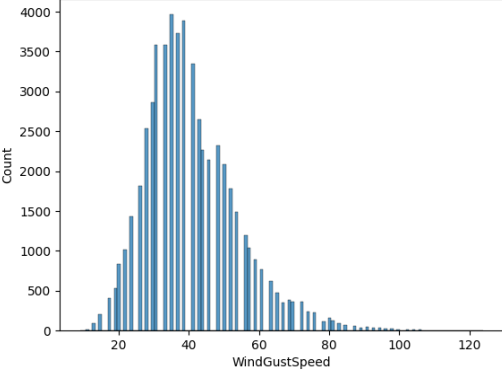
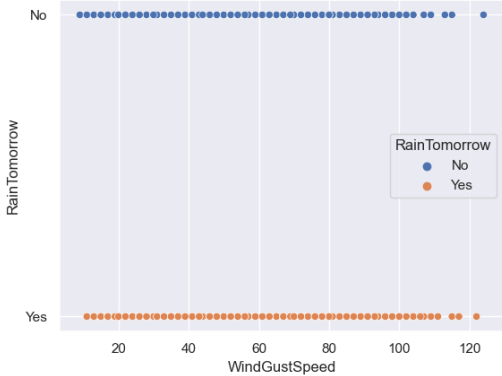
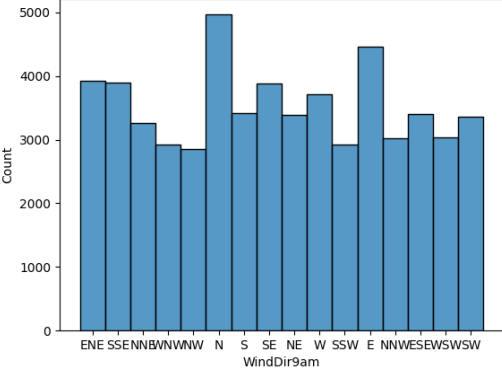


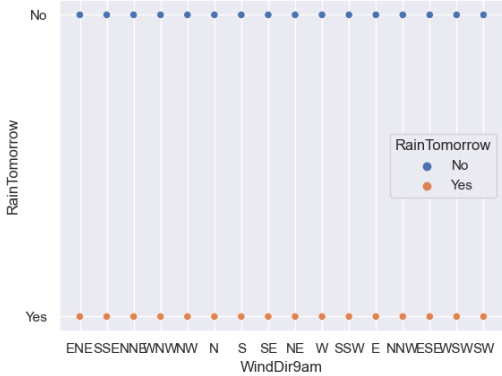
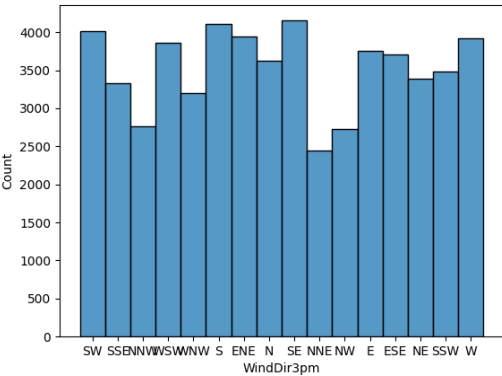
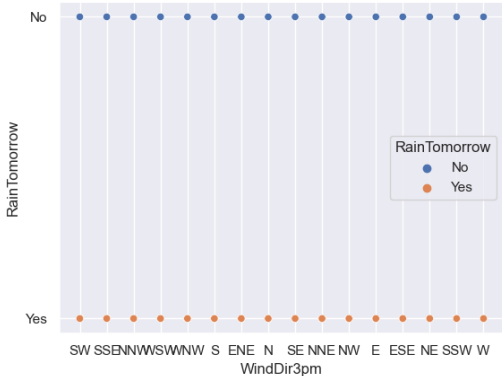
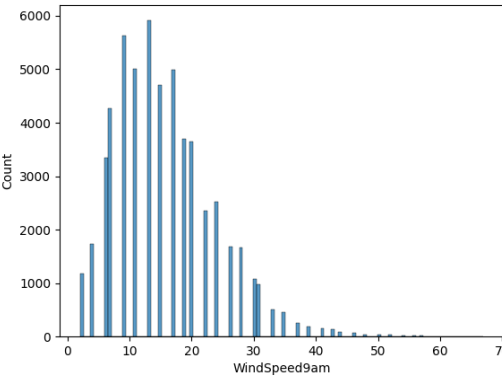
	 <p>A scatter plot showing the relationship between 'RainTomorrow' (Y-axis, with 'No' at the top and 'Yes' at the bottom) and 'Location' (X-axis). The plot displays a horizontal line of blue dots at the 'No' level and a horizontal line of orange dots at the 'Yes' level. The legend indicates that blue dots represent 'No' and orange dots represent 'Yes'.</p>
<p><b>MinTemp:</b> atribut que indica la temperatura minima del dia (<b>float</b>)</p>	 <p>Two plots are shown. The top plot is a histogram of 'MinTemp' (X-axis, ranging from -5 to 30) with 'Count' (Y-axis, ranging from 0 to 1750). The distribution is roughly bell-shaped, peaking around 10-12. The bottom plot is a scatter plot of 'RainTomorrow' (Y-axis, with 'No' at the top and 'Yes' at the bottom) against 'MinTemp' (X-axis, ranging from -5 to 30). The plot shows a horizontal line of blue dots at the 'No' level and a horizontal line of orange dots at the 'Yes' level. The legend indicates that blue dots represent 'No' and orange dots represent 'Yes'.</p>
<p><b>MaxTemp:</b> atribut que indica la temperatura màxima del dia (<b>float</b>)</p>	 <p>A histogram of 'MaxTemp' (X-axis, ranging from 10 to 50) with 'Count' (Y-axis, ranging from 0 to 1600). The distribution is roughly bell-shaped, peaking around 20-22.</p>

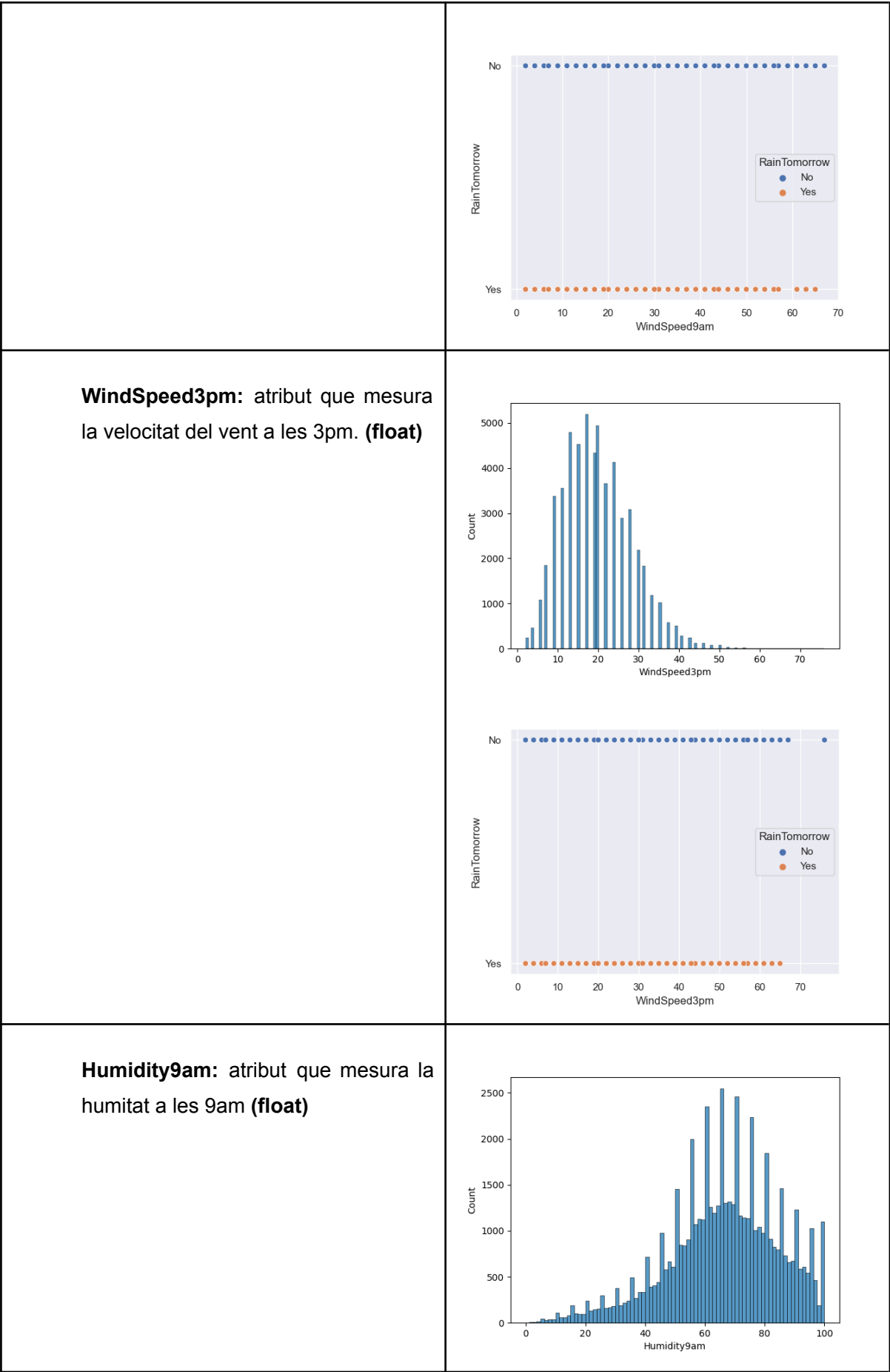
	 <p>A scatter plot showing the relationship between 'MaxTemp' (x-axis, 0 to 50) and 'RainTomorrow' (y-axis, 'No' and 'Yes'). The 'No' series (blue dots) is concentrated at the top, while the 'Yes' series (orange dots) is concentrated at the bottom. A legend in the top right corner identifies the series.</p>
<p><b>Rainfall:</b> atribut que indica la pluja mesurada en aquell dia (<b>float</b>)</p>	 <p>A histogram showing the distribution of 'Rainfall' (x-axis, 0 to 200) with 'Count' on the y-axis (0 to 35000). The distribution is highly right-skewed, with a peak count of approximately 35,000 at low rainfall values.</p>  <p>A scatter plot showing the relationship between 'Rainfall' (x-axis, 0 to 200) and 'RainTomorrow' (y-axis, 'No' and 'Yes'). The 'No' series (blue dots) is concentrated at the top, while the 'Yes' series (orange dots) is concentrated at the bottom. A legend in the top right corner identifies the series.</p>
<p><b>Evaporation:</b> atribut que indica la evaporació en mm d'aquell dia (<b>float</b>)</p>	 <p>A histogram showing the distribution of 'Evaporation' (x-axis, 0 to 80) with 'Count' on the y-axis (0 to 2500). The distribution is highly right-skewed, with a peak count of approximately 2,500 at low evaporation values.</p>

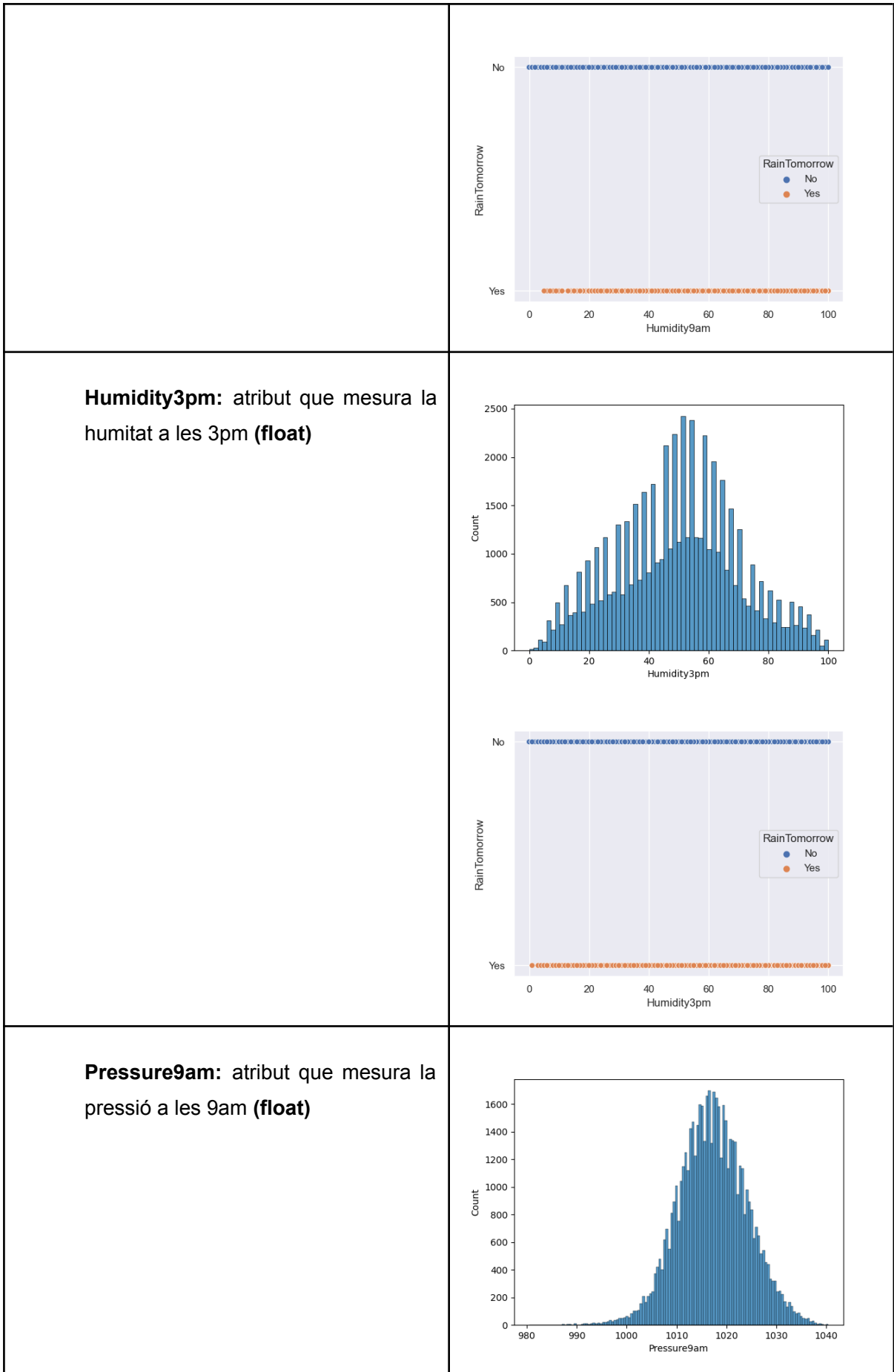


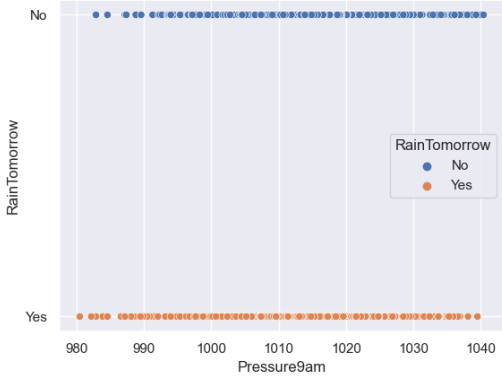
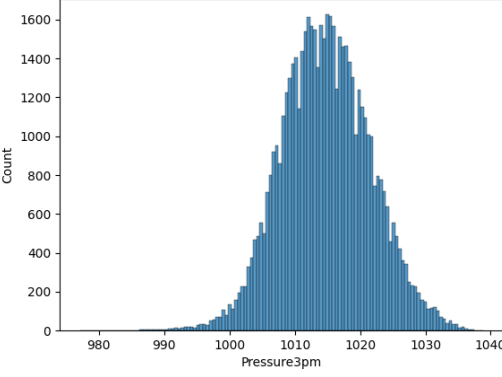
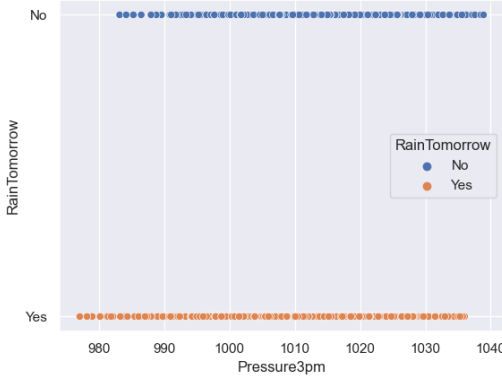
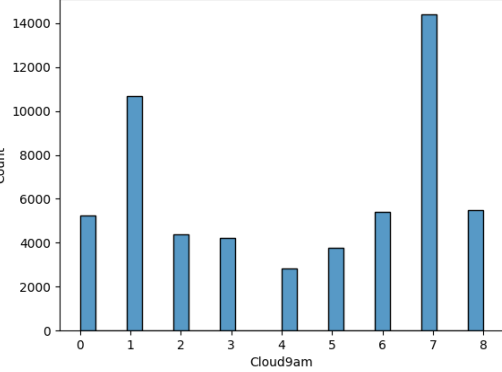
	
<p><b>Sunshine:</b> atribut que indica el nº d'hores de sol en aquell dia (<b>float</b>)</p>	 
<p><b>WindGustDir:</b> atribut que indica la direcció de la ratxa mes forta de vent mesurada aquell dia (<b>string</b>)</p>	

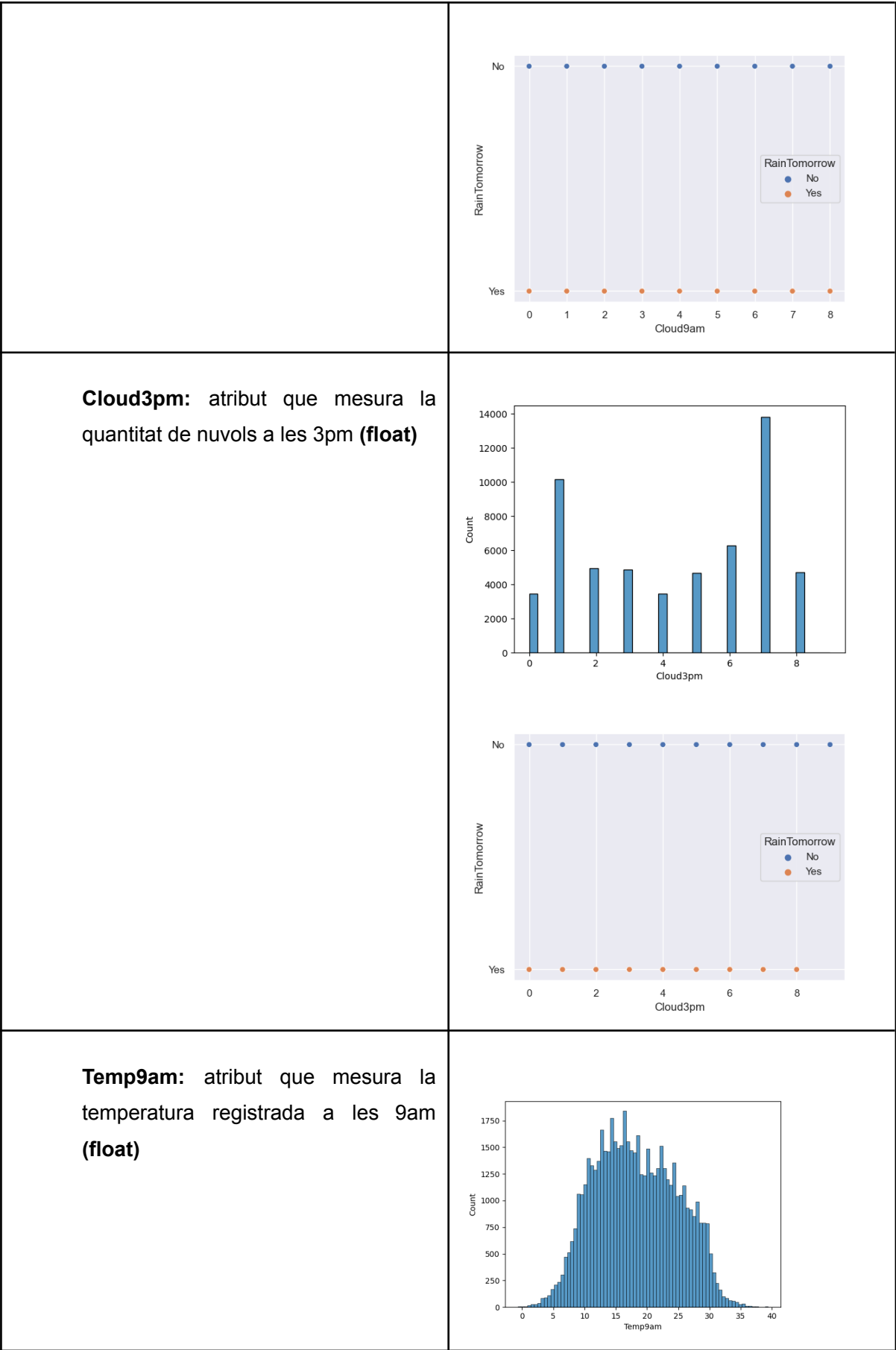
	
<p><b>WindGustSpeed:</b> atribut que mesura la velocitat de la ratxa de vent mes forta en km/h (<b>float</b>)</p>	 
<p><b>WindDir9am:</b> atribut que mesura la direcció del vent a les 9am. (<b>string</b>)</p>	

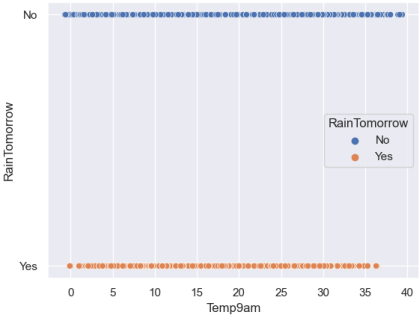
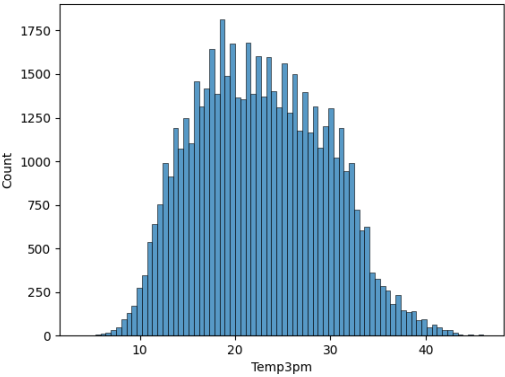
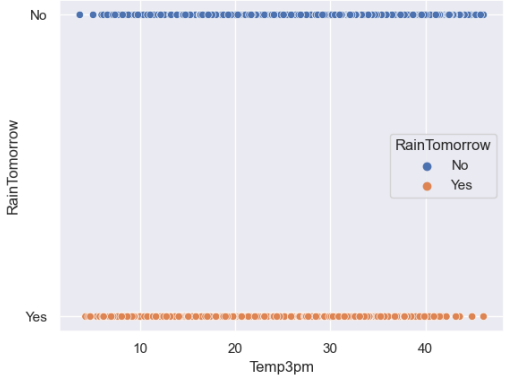
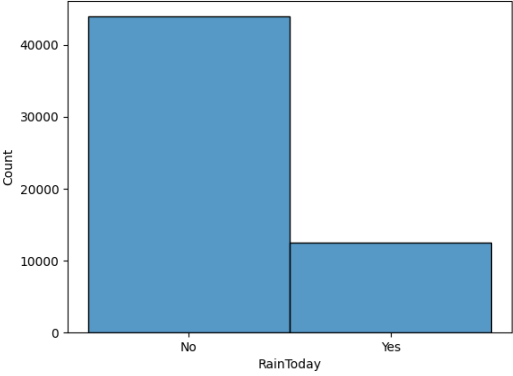
	
<p><b>WindDir3pm:</b> atribut que mesura la direcció del vent a les 3pm. <b>(string)</b></p>	 
<p><b>WindSpeed9am:</b> atribut que mesura la velocitat del vent a les 9am. <b>(float)</b></p>	



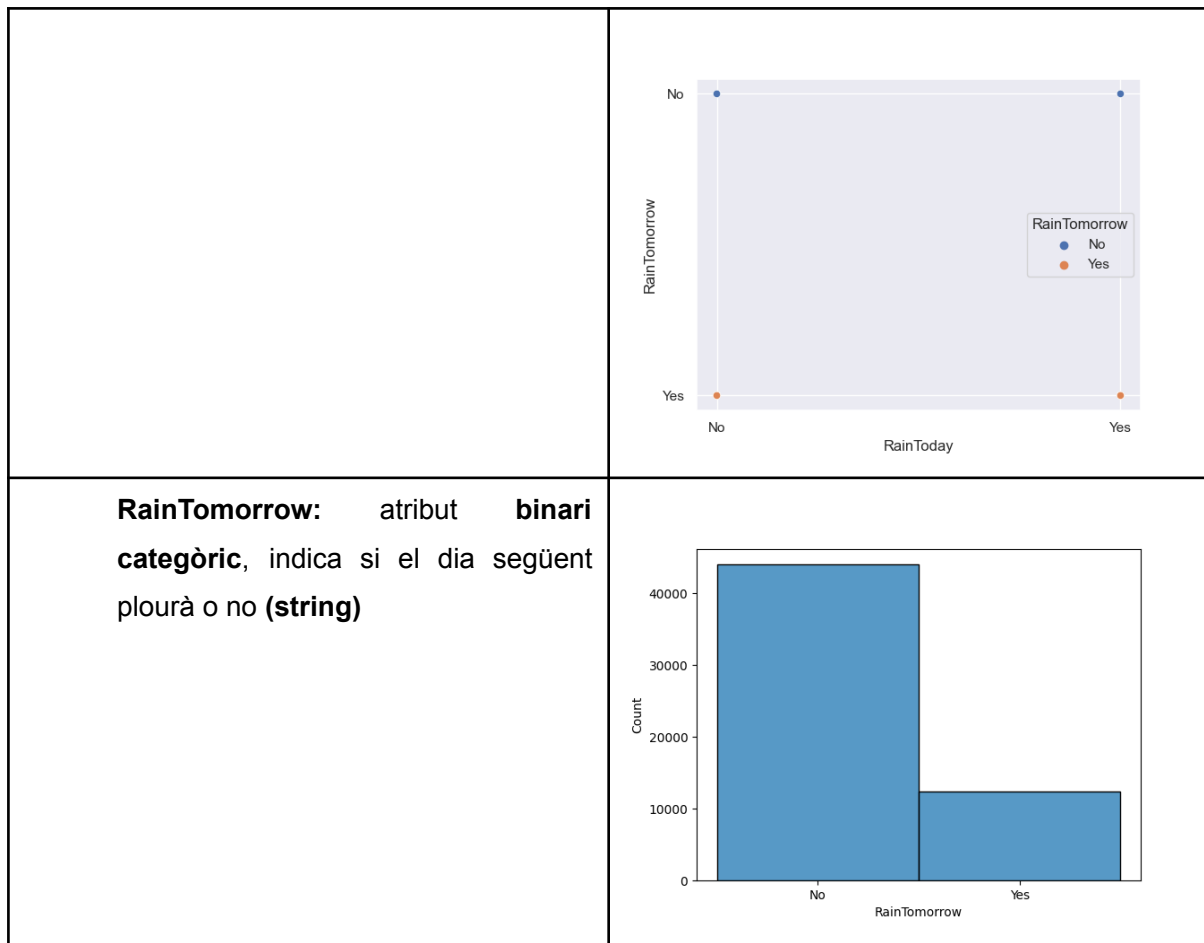


	 <p>A scatter plot showing the relationship between 'Pressure9am' (x-axis, ranging from 980 to 1040) and 'RainTomorrow' (y-axis, with categories 'No' and 'Yes'). The plot shows a dense horizontal line of blue dots at the 'No' level and a sparse line of orange dots at the 'Yes' level, indicating that rain tomorrow is generally independent of the pressure at 9am.</p>
<p><b>Pressure3pm:</b> atribut que mesura la pressió a les 3pm (<b>float</b>)</p>	 <p>A histogram showing the distribution of 'Pressure3pm'. The x-axis ranges from 980 to 1040, and the y-axis (Count) ranges from 0 to 1600. The distribution is unimodal and roughly bell-shaped, peaking around 1015.</p>  <p>A scatter plot showing the relationship between 'Pressure3pm' (x-axis, ranging from 980 to 1040) and 'RainTomorrow' (y-axis, with categories 'No' and 'Yes'). Similar to the first plot, it shows a dense horizontal line of blue dots at the 'No' level and a sparse line of orange dots at the 'Yes' level, suggesting no strong correlation.</p>
<p><b>Cloud9am:</b> atribut que mesura la quantitat de núvols a les 9am (<b>float</b>)</p>	 <p>A histogram showing the distribution of 'Cloud9am'. The x-axis ranges from 0 to 8, and the y-axis (Count) ranges from 0 to 14000. The distribution is multimodal, with prominent peaks at 1 (count ~10500) and 7 (count ~14200).</p>



	
<p><b>Temp3pm:</b> atribut que mesura la temperatura registrada a les 3pm (float)</p>	 
<p><b>RainToday:</b> atribut binari categòric, indica si aquell dia ha plogut o no (string)</p>	

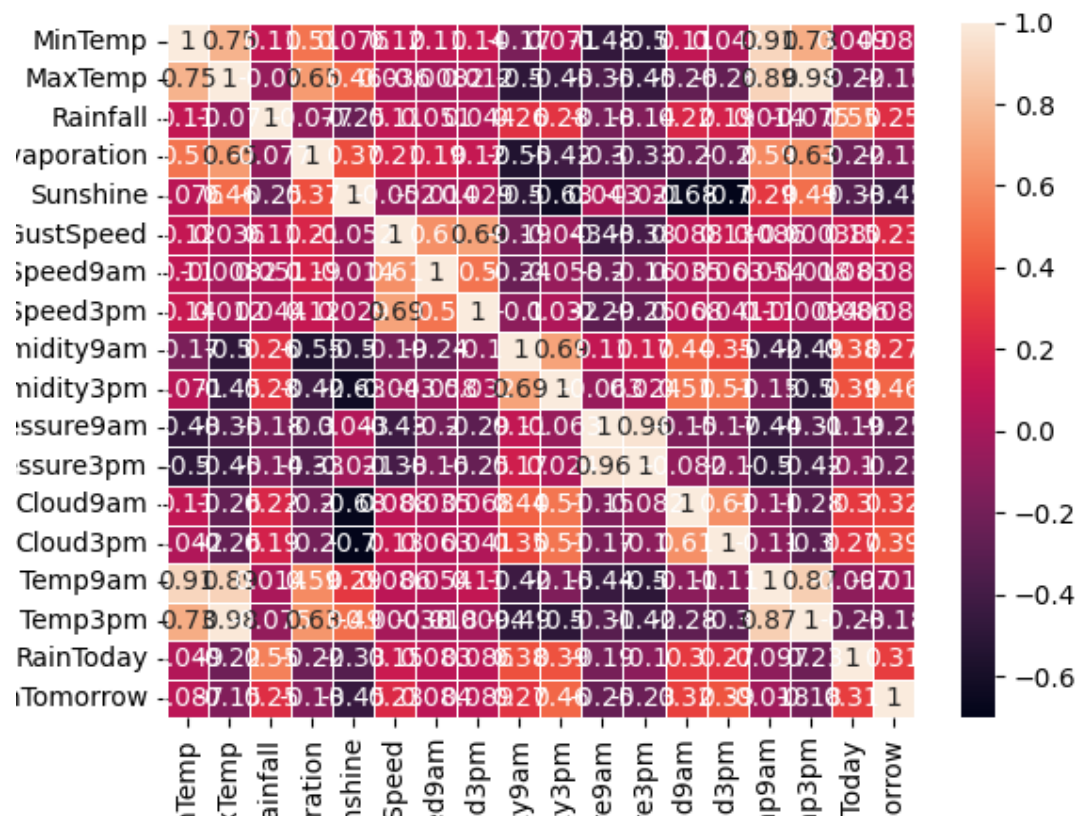




- Com es el target, quantes categories diferents existeixen?

L'atribut a predir serà **RainTomorrow**, al ser un atribut binari (Si, No), tindrem 2 **classes** a predir.

- Podeu veure alguna correlació entre X i y?



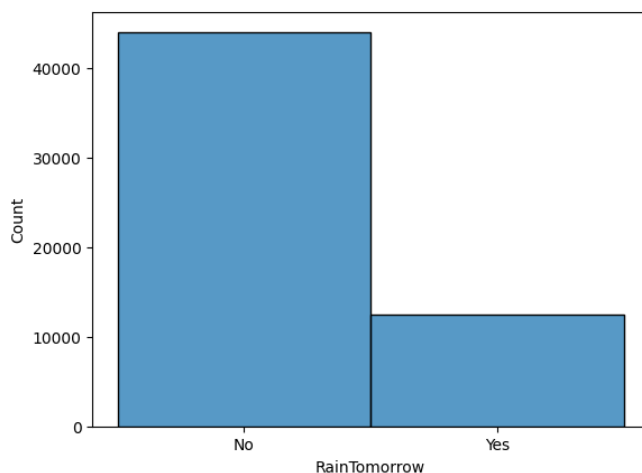
No tenim cap atribut que tingui molta correlació amb l'atribut a predir **RainTomorrow**. Els atributs que més correlació tenen son **Humidity3pm**, **Cloud9am**, **Cloud3pm**, **RainToday**, encara que tots ells tenen una correlació baixa ( $< 0.5$ ).

És a dir, els atributs que més relació tenen amb la predicció de precipitació son aquells que tenen més relació amb aquesta pluja (humitat, núvols, ...)

- Estan balancejades les etiquetes (distribució similar entre categories)? Creus que pot afectar a la classificació la seva distribució?

Les etiquetes en l'atribut a predir **RainTomorrow** estan força desbalancejades, en aquest cas l'atribut indica si el dia següent a les mesures climatològiques plourà o no (No / Yes), hem mesurat quina és la proporció de **No** i quina és la proporció de **Yes**, obtenint:

- **No:** 0.7797
- **Yes:** 0.2202



Podria afectar a la classificació, ja que disposem de moltes més mostres que al dia següent no plou, que mostres que plougui el dia següent.

## 2. Preprocessing (normalization, outlier removal, feature selection)

- Estan les dades normalitzades? Caldria fer-ho?

En el nostre dataset les dades no estan normalitzades, però al observar quins valors prenen la majoria d'atributs:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	Win
count	56420.000000	56420.000000	56420.000000	56420.000000	56420.000000	56420.000000	56420.000000	5
mean	13.464770	24.219206	2.130397	5.503135	7.735626	40.877366	15.667228	
std	6.416689	6.970676	7.014822	3.696282	3.758153	13.335232	8.317005	
min	-6.700000	4.100000	0.000000	0.000000	0.000000	9.000000	2.000000	
25%	8.600000	18.700000	0.000000	2.800000	5.000000	31.000000	9.000000	

Creiem que no cal fer una normalització, si ens fixem en la mitjana de valors que prenen els atributs, podem veure que la majoria d'ells estan en el mateix ordre (0-100) per la qual cosa normalitzar aquests valors no canviarà gaire el resultat.

El que hem fet ha estat un escalat de les dades, per donar la mateixa importància a tots els atributs a l'hora d'aplicar els diferents models (SVM, KNN, Regression, Perceptron). És important ja que situa els valors en un rang entre 0-1 de manera que un canvi d'una unitat en un dels atributs tindrà la mateixa importància que en un altre. D'aquesta manera, es normalitza utilitzant la fórmula:

$$X_{new} = \frac{X_{old}}{X_{max}}$$

- **Teniu gaires dades sense informació? Els NaNs a pandas? Tingueu en compte que hi ha mètodes que no els toleren durant el aprenentatge. Com afecta a la classificació si les filtrem? I si les reompliu? Com ho farieu? [Pista](#)**

Sí, hem observat que el dataset per defecte, disposa de massa atributs amb valor NaN:

Degut a això, hem decidit eliminar totes les files on algun dels atributs tingui un valor NaN / null, ja que no hem pogut omplir aquests valors amb 0, o la mitjana degut a que alguns valors no son de tipus **int** / **float** sinò que son dates, o direccions del vent.

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

- **Teniu dades categoriques? Quina seria la codificació amb més sentit? (OrdinalEncoder, OneHotEncoder, d'altres?)**

Si, tant **RainToday** com **RainTomorrow** (y) son variables categòriques, ja que només prenen els valors (No / Yes).

En el nostre cas com les variables categòriques només disposen de dos valors (**binaries**) la codificació **OrdinalEncoder** serà la més adient, simplement transformarem el **No** en un **0** i el **Yes** en un **1**.

- **Caldria aplicar `sklearn.decomposition.PCA`? Quins beneficis o inconvenients trobarieu?**

El PCA es una tècnica que ens permet reduir un espai de n atributs a un mes petit, com per exemple 2 o 3.

En el nostre cas no hem pogut aplicar-lo ja que hi ha bastantes variables les quals no estan suportades per l'algorisme **`sklearn.decomposition.PCA`**

- **Es poden aplicar `PolynomialFeatures` per millorar la classificació? En quins casos té sentit fer-ho?**

`PolynomialFeatures` genera un nou atribut que consisteix en la combinació lineal de diferents atributs.

En el nostre cas no hem aplicat `PolynomialFeatures` ja que com ens passava a PCA, al disposar de bastantes variables de tipus string (data, localització, direcció del vent) les quals no podem convertir a un valor float, no ens permet aplicar aquest tipus de preprocessaments.

### 3. Model Selection

En aquesta secció considerarem la implementació dels diferents models de regressió que hem après a classe, el perceptró, el KNN, SVM i regressió logística. Per cada model cal analitzar els resultats obtinguts i comparar els resultats.

Després d'executar els diferents models amb diferents percentatges de dades de test i dades de validació, tot i observar que en el cas de la nostre BD els resultats són bastant parells, el valor més alt obtingut és un 85,59% de precisió. Obtingut amb el model SVM utilitzat un 70% de test i 30% de validació.

Resultats obtinguts:

Test / Validació (%)	Regressió logística	KNN	SVM	Perceptró
50 / 50	85,38	83,91	85,35	80,50
80 / 20	85,31	84,45	85,32	78,33
70 / 30	85,31	84,45	85,59	80,74

- **Quins models heu considerat?**

Hem considerat provar cadascun dels models: **perceptró, KNN, SVM i logistic regression.**

- **Considereu les SVM amb els diferents kernels implementats.**

Un kernel (K) es una funció que torna un resultat del producte escalar entre dos vectors realitzat en un nou espai dimensional diferent al espai original en el que es troben els vectors. Existeixen diferents tipus de kernels, i entre ells hi ha els més coneguts com poden ser el kernel lineal, polinòmic o gaussià.

Si utilitzem un Kernel lineal, el classificador SVM obtingut és equivalent al Support Vector Classifier, SVC.

$$K(x, x') = x \cdot x'$$

En el cas del kernel polinòmic, si s'utilitza una  $d = 1$  i  $c = 0$  el resultat és igual que el lineal, però si  $d > 1$ , es generen límits de decisió no lineals, augmentant la no linealitat a mesura que augmenta la  $d$ .

$$K(x, x') = (x \cdot x' + c)^d$$

Finalment en el cas del kernel gaussià el valor  $\gamma$  controla el comportament del kernel, quan es molt petit, el model final equival a l'obtingut amb un kernel lineal, a mesura que va augmentant el seu valor, també ho fa la flexibilitat del model.

$$K(x, x') = \exp(-\gamma \|x - x'\|^2)$$

- **Quin creieu que serà el més precís?**

El tipus de kernel que retorna un valor més precís és el kernel gaussià, en canvi el kernel més ràpid és el lineal.

## 4. Crossvalidation

- **Per què és important cross-validar els resultats?**

El crossvalidation és una tècnica que serveix per analitzar si l'entrenament que hem fet al nostre model ha estat efectiu.

S'utilitza un conjunt de validació, el qual anirà canviant de proporció en les diferents iteracions de entrenament que fem.

- **Separa la base de dades en el conjunt de train-test. Com de fiables serán els resultats obtinguts? En quins casos serà més fiable, si tenim moltes dades d'entrenament o poques?**

<b>Test / Validació (%)</b>	<b>Regressió logística</b>	<b>KNN</b>	<b>SVM</b>	<b>Perceptró</b>
<b>50 / 50</b>	85,38	83,91	85,35	80,50
<b>80 / 20</b>	85,31	84,45	85,32	78,33
<b>70 / 30</b>	85,31	84,45	<b>85,59</b>	80,74

En el nostre cas, com podem observar la fiabilitat no varia molt amb les diferents proporcions de **Test / Validació**, tot i així el valor que més fiabilitat ens ha donat ha estat **70% test i 30% validació** amb el model **Support Vector Machine**.

- Quin tipus de K-fold heu escollit? Quants conjunts heu seleccionat (quina k)? Com afecta els diferents valors de k?

Hem escollit el rang 2 → 6 de K, obtenint els següents resultats:

### Regressió logística

Regressió Logística	Precisió	C	fit_intercept	penalty	tolerance
<b>Bàsica</b>					
50% test 50%val	85,38	2.0	True	None	0.001
80% test 20% val	85,31	2.0	True	None	0.001
70% test 30% val	85,31	2.0	True	None	0.001
<b>K-Fold</b>					
K = 2	0,8507	2.0	True	None	0.001
K = 3	0,8517	2.0	True	None	0.001
K = 4	0,8530	2.0	True	None	0.001
K = 5	0,8532	2.0	True	None	0.001
K = 6	0,8518	2.0	True	None	0.001
LOOVC	0,8532	2.0	True	None	0.001

### KNN

KNN	Precisió	leaf_size	n_neighbors	metric	p
<b>Bàsica</b>					
50% test 50%val	83,91	30	5	minkowski	2



80% test 20% val	84,45	30	5	minkowski	2
70% test 30% val	84,45	30	5	minkowski	2
<b>K-Fold</b>					
K = 2	0.8270	30	5	minkowski	2
K = 3	0.8258	30	5	minkowski	2
K = 4	0.8325	30	5	minkowski	2
K = 5	0.8305	30	5	minkowski	2
K = 6	0.8315	30	5	minkowski	2
LOOVC	0,8445	2.0	True	None	0.001

## SVM

<b>SVM</b>	Precisió	C	fit_intercept	penalty	tolerance
<b>Bàsica</b>					
50% test 50%val	85,35	2.0	True	None	1e-5
80% test 20% val	85,32	2.0	True	None	1e-5
70% test 30% val	85,59	2.0	True	None	1e-5
<b>K-Fold</b>					
K = 2	0,8509	1.0	True	None	1e-5
K = 3	0,8520	1.0	True	None	1e-5
K = 4	0,8530	1.0	True	None	1e-5
K = 5	0,8529	1.0	True	None	1e-5
K = 6	0,8522	1.0	True	None	1e-5
LOOVC	0,8530	1.0	True	None	1e-5

## Perceptró

Perceptró	Precisió	$\alpha$	fit_intercept	penalty	tolerance
<b>Bàsica</b>					
50% test 50%val	80,50	0.0001	True	None	0.001
80% test 20% val	78,33	0.0001	True	None	0.001
70% test 30% val	80,74	0.0001	True	None	0.001
<b>K-Fold</b>					
K = 2	0,8522	0.0001	True	None	0.001
K = 3	0,7886	0.0001	True	None	0.001
K = 4	0,7789	0.0001	True	None	0.001
K = 5	0,7981	0.0001	True	None	0.001
K = 6	0,8049	0.0001	True	None	0.001
LOOVC	0,8522	0.0001	True	None	0.001

- Es viable o convenient aplicar **LeaveOneOut**?

El mètode de cross-validation **LeaveOneOut** és el més òptim / exacte, però com a desavantatge, el còmput necessari és molt alt.

## 5. Metric Analysis

- A teoria, hem vist el resultat d'aplicar el **accuracy\_score** sobre dades no balancejades. Podrieu explicar i justificar quina de les següents mètriques serà la més adient pel vostre problema? **accuracy\_score**, **f1\_score** o **average\_precision\_score**.

L'**accuracy\_score** és la mètrica que quantifica com de bé estan classificades tant les observacions de les classes **positives** o **negatives**. És recomanable no usar aquesta mètrica d'**accuracy\_score** en dades no balancejades, ja que es molt fàcil que ens doni un accuracy alt. En el nostre cas tenim dades **no balancejades**, ja que tenim molts més files de valors amb la variable target en **No**, que en valor **Yes**.

En quant al **f1\_score**, és una mètrica que combina tant **precisió** com **recall** en una sola mètrica. És una bona elecció si no tenim clar que prioritzar (o bé precisió o bé recall), ja que amb aquesta mètrica tindriem en compte les dues.

Finalment, el **average\_precision\_score** com el seu nom indica, ens donarà la mitjana de les precisions amb diferents thresholds. És una bona mètrica per comparar com de bé els models estan ordenant aquestes prediccions a l'hora de classificar.

Els resultats d'executar les mètriques d'anàlisis en la nostra base de dades han estat:

Metric Analysis	Accuracy score	F1 score	Average Precision Score
SVM	0,8336	0,7931	0,3715
Perceptró	0,7132	0,7373	0,4069
KNN	0,8423	0,8347	0,4673
Regressió Logística	0,8521	0,8436	0,4891

- **Mostreu la Precisió-Recall Curve i la ROC Curve. Quina és més rellevant pel vostre dataset? Expliqueu amb les vostres paraules, la diferencia entre una i altre [Pista](#)**

La **Precision-Recall Curve** es la corba que mostra l'equilibri entre els termes **precisió** i **recall**.

En aquest cas, **la precisió** es el ratio de classificacions correctes del nostre model.

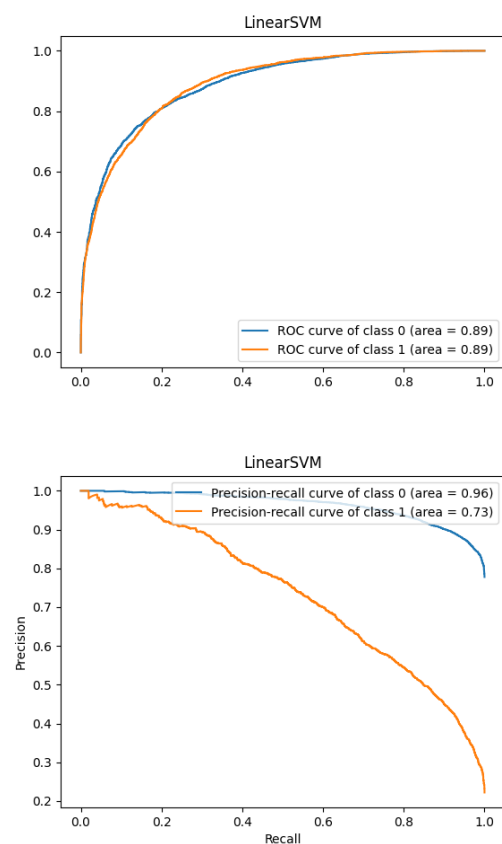
En canvi, **el recall**, que el podriem traduir com "sensibilitat" del model a l'hora de detectar **la classe positiva**.

Aquestes dues mètriques estan relacionades entre si, i quan una puja l'altre baixa.

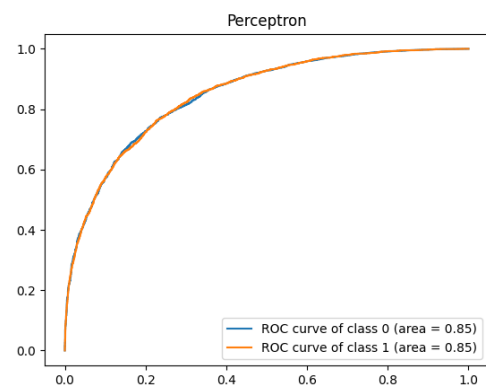
La **ROC Curve** , es semblant a la **Precision-Recall Curve**, en aquest cas relaciona **recall** (**sensibilitat**) amb el ratio de **falsos positius**. Es una mètrica que ens ajuda a veure, en quant anem augmentant la sensibilitat del model, com augmenta el ratio de falsos positius.

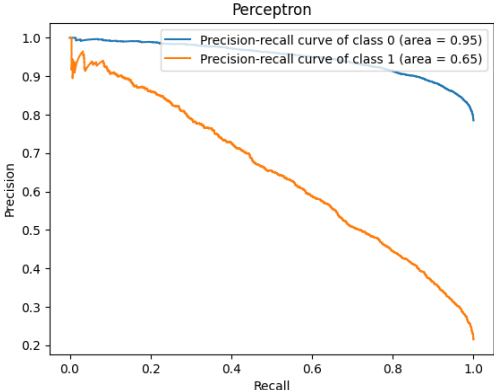
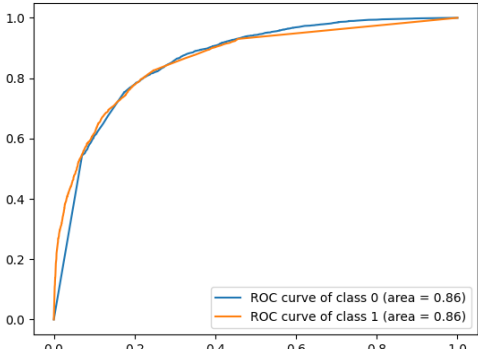
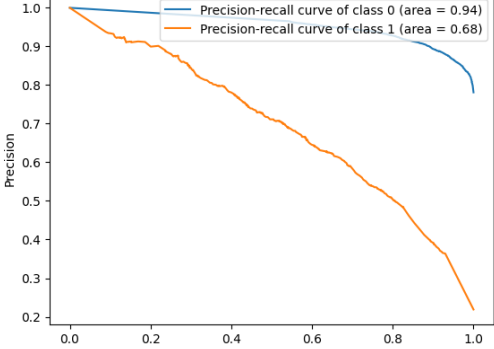
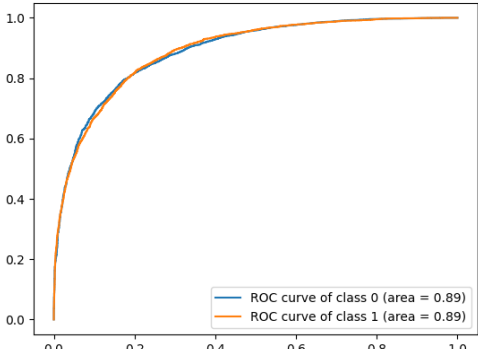
El resultat de les corbes **ROC** i **Precision-Recall** a les nostres dades han estat les següents:

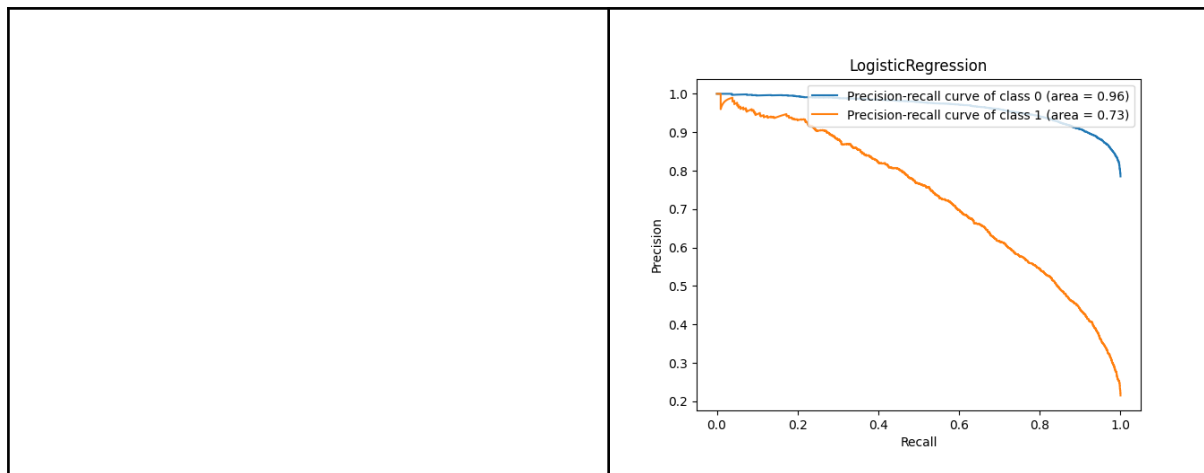
### Support Vector Machines



### Perceptron



	<p>Perceptron</p>  <p>Precision-Recall curve for class 0 (area = 0.95) Precision-Recall curve of class 1 (area = 0.65)</p> <p>This plot shows the precision-recall curves for the Perceptron model. The x-axis is Recall (0.0 to 1.0) and the y-axis is Precision (0.2 to 1.0). The blue curve for class 0 starts at (0, 1.0) and remains high until recall is approximately 0.8, then drops to about 0.8 at recall 1.0. The orange curve for class 1 starts at (0, 1.0), drops sharply to a precision of about 0.9 at recall 0.05, and then gradually declines to a precision of about 0.25 at recall 1.0.</p>
<b>KNN</b>	<p>KNN</p>  <p>ROC curve of class 0 (area = 0.86) ROC curve of class 1 (area = 0.86)</p> <p>KNN</p>  <p>Precision-Recall curve of class 0 (area = 0.94) Precision-Recall curve of class 1 (area = 0.68)</p> <p>This section contains two plots for the KNN model. The top plot is an ROC curve with both class 0 (blue) and class 1 (orange) curves showing an area of 0.86. The curves are very similar, starting at (0,0) and ending at (1,1), with a steep initial rise. The bottom plot is a precision-recall curve. The blue curve for class 0 starts at (0, 1.0) and remains high until recall is approximately 0.8, then drops to about 0.8 at recall 1.0. The orange curve for class 1 starts at (0, 1.0), drops sharply to a precision of about 0.9 at recall 0.05, and then gradually declines to a precision of about 0.25 at recall 1.0.</p>
<b>Logistic Regression</b>	<p>LogisticRegression</p>  <p>ROC curve of class 0 (area = 0.89) ROC curve of class 1 (area = 0.89)</p> <p>This plot shows the ROC curve for the Logistic Regression model. The x-axis is Recall (0.0 to 1.0) and the y-axis is Precision (0.0 to 1.0). Both the blue curve for class 0 and the orange curve for class 1 show an area of 0.89. The curves are very similar, starting at (0,0) and ending at (1,1), with a steep initial rise.</p>



En el cas de la **corba Precision-recall**, ens interessa que sigui el més propera a [1, 1] possible en la seva gràfica, això significarà que el nostre model està classificant correctament (precisió), i la seva sensibilitat (recall) per detectar la mostra positiva també es bona.

En el cas de la **corba ROC**, ens interessa que sigui el més propera possible a [0, 1], ja que això significarà que el ratio de falsos positius es petit, mentre que el ratio de true positives es alt, i això es el que volem que faci el nostre model a l'hora de classificar.

En el nostre cas, podem observar que tant el model de **Regressió Logística** com el **SVM**, son els dos models que ens donen una millor **corba ROC**, ahora que també tenen una bona **corba PR**. Això vol dir que aquests dos models, son els que millor estan classificant les mostres, amb la millor precisió i sensibilitat, i també que tenen el menor ratio de falsos positius.

- Què mostra [classification\\_report](#)? Quina mètrica us fixareu per tal d'optimitzar-ne la classificació pel vostre cas?

És una eina de sklearn que retorna un report de les mètriques d'anàlisis principals. D'aquesta manera se li passa com a paràmetres l'array de targets correctes i les prediccions fetes pels classificadors, retornant un text amb les mètriques precision, recall, f1\_score i support.

El resultat d'executar el classification\_report en la nostra base de dades ha estat el següent:

#### 1.SVM:

	precision	recall	f1-score	support
Class 0	0.81	0.99	0.89	8996
Class 1	0.90	0.21	0.34	2622
accuracy			0.82	11618
macro avg	0.86	0.60	0.62	11618
weighted avg	0.83	0.82	0.77	11618

#### 2.Perceptró:

	precision	recall	f1-score	support
Class 0	0.90	0.90	0.90	9075
Class 1	0.64	0.63	0.63	2543
accuracy			0.84	11618
macro avg	0.77	0.76	0.77	11618
weighted avg	0.84	0.84	0.84	11618

#### 3.KNN:

	precision	recall	f1-score	support
Class 0	0.87	0.93	0.90	9063
Class 1	0.67	0.52	0.58	2555
accuracy			0.84	11618
macro avg	0.77	0.72	0.74	11618
weighted avg	0.83	0.84	0.83	11618

#### 4.Logistic Regression:

	precision	recall	f1-score	support
Class 0	0.87	0.95	0.91	9058
Class 1	0.74	0.51	0.60	2560
accuracy			0.85	11618
macro avg	0.81	0.73	0.76	11618
weighted avg	0.84	0.85	0.84	11618

## 6. Hyperparameter Search

- Quines formes de buscar el millor parametre heu trobat? Són costoses computacionalment parlant? [documentació](#)

Les formes de buscar el millor paràmetre que hem trobat són el **GridSearchCV** i el **RandomizedSearchCV**. En el primer cas, troba els millors paràmetres combinant tots els valors possibles de la llista passada per l'usuari. En el segon en canvi, afegeix un factor "random" per a no haver de provar totes les combinacions possibles (computacionalment costós). Per contra, aquest últim model podria no trobar la combinació òptima de paràmetres.

- Si disposem de recursos limitats (per exemple, un PC durant 1 hora) quin dels dos mètodes creieu que obtindrà millor resultat final?

Entre els mètodes **GridSearchCV** i **RandomizedSearchCV** el que aconseguirà millors resultats quan es disposa de recursos limitats serà el **RandomizedSearchCV**. Aquest mètode està dissenyat per aconseguir un resultat bastant aproximat, però evita combinacions d'hiperparàmetres que troba "innecessaries" i genera hiperparàmetres a l'atzar.

- Existeixen altres mètodes de búsqueda més eficients ([scikit-optimize](#))?

Sí, Scikit-optimize proveeix un reemplaçament per a **GridSearchCV** on utilitza el mètode de "Bayesian Optimization" el qual permet modelar l'espai de cerca i s'utilitza per arribar a combinacions bones de paràmetres el més aviat possible.

- Cerca d'hiperparàmetres amb el nostre dataset

En aquest apartat farem una cerca d'hiperparàmetres amb un dels mètodes anteriorment explicats.

En el nostre cas utilitzarem el **Exhaustive Grid Search (GridSearchCV)**, ja que encara que és una mica més costós en quant a còmput respecte al **Randomized Parameter Optimization**,



ens permetrà provar amb totes les combinacions d'hiperparàmetres, per poder obtenir el resultat òptim entre totes aquestes combinacions.

Com als apartats anteriors, hem utilitzat els models **Perceptron**, **KNN**, **Logistic Regression** i **SVM** amb les següents combinacions de paràmetres:

<b>Perceptron</b>	{ 'penalty': ['l2','l1'], 'alpha': [0.0001, 0.001, 0.01, 0.1, 1], 'fit_intercept': [True, False], 'shuffle': [True, False] }
<b>KNN</b>	{ 'n_neighbors' : [5,7,9,11,13], 'weights' : ['uniform','distance'], 'metric' ['minkowski','euclidean','manhattan'] }
<b>Logistic Regression</b>	{'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000] }
<b>SVM</b>	{'C': [0.1,1, 10, 100, 1000]}

Per cadascun d'aquests hiperparàmetres, l'algorisme **GridSearchCV** ha escollit els següents com a millors hiperparàmetres:

<b>Perceptron</b>	{ 'penalty': 'l1', 'alpha': <b>0.0001</b> , 'fit_intercept': <b>True</b> , 'shuffle': <b>True</b> }
<b>KNN</b>	{ 'n_neighbors' : <b>13</b> , 'weights' : ' <b>uniform</b> ', 'metric': ' <b>manhattan</b> ' }
<b>Logistic Regression</b>	{'C': <b>1000</b> }
<b>SVM</b>	{'C': <b>10</b> }

En quant als resultats (accuracy) obtinguts amb aquests valors d'hiperparàmetres, obtenim les següents dades:

Model	Accuracy amb paràmetres per defecte	Accuracy amb paràmetres obtinguts amb GridSearchCV	Millora
Perceptron	0.7132	0.8224	15.31%
KNN	0.8423	0.8413	-0.11%
Logistic Regression	0.8521	0.8506	-0.17%
SVM	0.8336	0.8333	-0.03%

Com podem observar per aquests resultats, l'únic model que millora els seus resultats en quant a accuracy quan busquem els millors hiperparàmetres és el **Perceptron**, amb una molt bona millora de 15.31% d'accuracy, situant-lo més o menys amb el mateix accuracy que la resta de models.

Com podem veure a la resta de models, cap d'ells millora, sinó que empitjoren els seus resultats quan provem diferents combinacions d'hiperparàmetres. Això suposem que és degut a que els millors hiperparàmetres per aquests tres models (KNN, Logistic Regression i SVM) són els que utilitzen per defecte.

Aplicant aquesta cerca d'hiperparàmetres podem concluir que independentment del model escollit, amb les nostres dades la precisió de tots ells es troba entorn al **82 - 85%**.