Machine Learning.

Angel Chaico yo
Facultad de ciencias, UNI[1]

23 de julio de 2022

[1]yo

Índice general

Capítulo 1

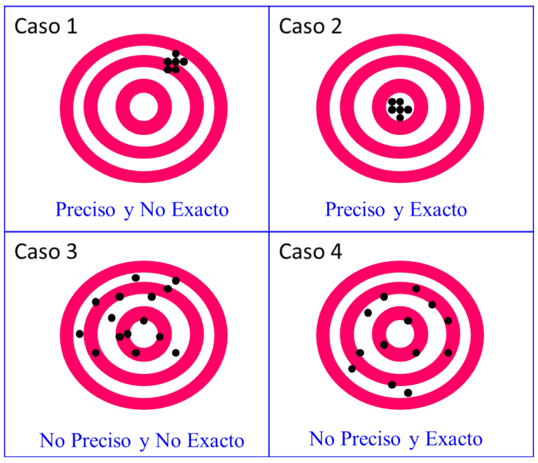
aora

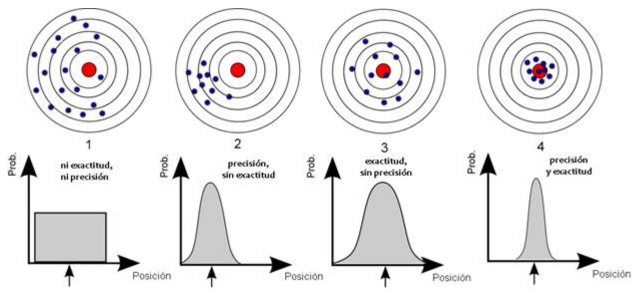
random forest, neural network, regresion logisticy hacer bibliografi por capitulo.

En loc codigos iran ejemplos graficos de cada modelo aplicado.

Parte I

Statistical and probabiliti (EDA).





1.1. Estimates and explorind data.

1.2. Distributions.

1.3. Statistical experiment.

Parte II

Resamplin(train and test set).

Capítulo 2

Sampli:stratied

Capítulo 3

Imbalance data.

3.1. Undersampling

3.2. Oversamplig

3.2.1. Bootstraping

3.2.2. data generation: SMOTE algorithm

3.3. Weighting.

3.4. Cost-based classification.

3.5. bibliografi

revisar
practical sttistical for data science, by peter bruce.

Parte III

Supervised (model)

Capítulo 4

Linear regression.

Capítulo 5

Logistic regression.

5.1. Regularization.

5.2. Código.

Capítulo 6

Desicion tree.

6.1. regression

:

- We divide the predictor space- that is, the set of posible values for $X_1, X_2, \cdots, X_p$- into J distinct and non-overlapping regions, $R_1, R_2, \cdots, R_J$.
- For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

The goal es to find $R_1, \cdots, R_J$ that minimize the RSS given by:

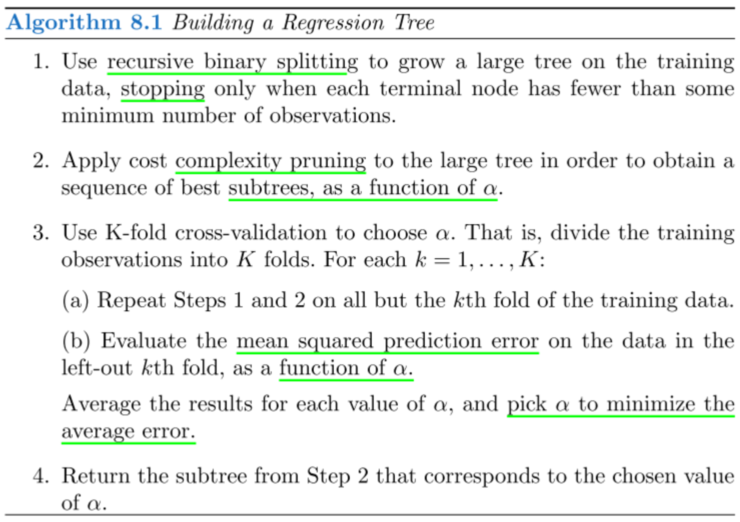
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{6.1}$$

Recursive binary splitting, we seek the value of j and s that minimize the equation

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{6.2}$$

Cost complexity pruning (weakest link pruning). For each value of α there corresponds a subtree $T \in T_0$, T_0 is when $\alpha = 0$, such that is as small as possible. The $|T|$ indicate the number of terminal nodes of the tree T.

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \tag{6.3}$$



Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Figura 6.1: Although CV error is computed as a function of α, it is convenient to display the result as a function of $|T|$.

6.2. Classification

Tree vs linear

Capítulo 7

Random Forest.

When building desicion trees, each time a split in a tree is considered, a random sample of m predictor is chosen as split candidates from the fulll set of p predictors (decorrelating trees).

7.1. Regularization.

- As with bagging, random forests will not overfit if we increase B trees, so in in practis we use a value of B sufficiently large for the error rate to have settled down.

Metod	default	observation
random sample of m predictors	m = $\sqrt{p}$	
bootstrapped (max_sample)		
Out-of-bag (OOB)		
criterion	gini	
max depth		

7.2. Código.

Importance attributes

feature_importances_	
n_outputs_	

Observactiona

- Para cada X_train diferente se obetiene features importances diferentes en el modelo resultalte, como determinar las features importances si el modelo cambia al cambiar los X_train?

Capítulo 8

Bagging

Capítulo 9

Boosting.

Capítulo 10

Bayesian additive regression trees.

Capítulo 11

Supert Vector Machine.

11.1. Regularization.

11.2. Código.

Parte IV

Unsupervised (model)

Capítulo 12

K-means

Capítulo 13

Hierarchical Clustering.

Parte V

Deep learning (model)

Capítulo 14

Neural Networks

14.1. Activation function.

Sigmoid

ReLU

ELU

$$ELU_\alpha = \begin{cases} \alpha(e^z - 1) & , z < 0 \\ z & , z \geq 0 \end{cases} \tag{14.1}$$

14.2. Convulation layer.

14.3. Poling layer.

Capítulo 15

Regularization.

Problems:

- Vanishing gradient or exploding gradients.
- Not have enought trainig data.
- Slow.
- Overfitting.

15.1. The Vanishing/Exploring gradients

Glorot and He initialization

Glorot and Yoshua show the with logistic sigmoid function activation and weight initialization technique, the variance of the output of each layer is much greater than the variance of its inputs.

Glorot show that the connection weights of each layer must be initialized randomly as[1]:

$$\text{Normal distribution}, \mu = 0, \sigma^2 = \frac{1}{fan_{avg}} \tag{15.1}$$

$$\text{Uniform distribution between} - r, r \text{ with } r = \sqrt{\frac{1}{fan_{avg}}} \tag{15.2}$$

$$fan_{avg} = \frac{fan_{int} + fan_{out}}{2} \tag{15.3}$$

[1]fan_in is the number of input and fan_out the number of neuron of the layer

initialization	activation function	$\sigma^2(Normal)$
Glorot	None, tanh, logistic, softmax	$\frac{1}{fan_{avg}}$
He	ReLU and variant	$\frac{2}{fan_{int}}$
Lecun	SELU	$\frac{1}{fan_{int}}$

Cuadro 15.1: Initialization parameters for each type of activation function; $r = \sqrt{3\sigma^2}$.

```
keras.layers.Dense(10,, activation="relu", kernel_initializer="he_normal")
```

```
% fan_in->fan_avg
he_avg_init=keras.initializers.VarianceScaling(scale=2,mode="fan_avg")
keras.layers.Dense(10, activation="sigmoid", kernel_initializer=he_avg_init)
```

Nonsaturing Activation Function

Glorot and Bengio show that the problems with unstable gradients were in part due to a poor choice of activation function.

- **ReLU**, problem dying ReLU.
- **leaky ReLU**, $LeakyReLU_\alpha(z) = max(\alpha z, z)$, usual fixed $\alpha = 0{,}01$, from one paper, it conclusions was that the leaky variants always outperform the estric ReLU, $\alpha = 0{,}2$ seemed to resutl in better performance.
- **RReLU**,randomize leaky ReLU, α is random during the training and fixed to an average value during testing; seems to act regularizer reducing the overfitting of the training set.
- **PReLU**, parametric leaky ReLU, α to be learned during training; was reported to strongly outperform ReLU on large image datasets, but on smaller it runs the risk of overfitting the training set.
- **ELU** exponential linear unit, better that all varians ReLU. ELU network will be sslower than a ReLU network.
- **SELU**, Scaled ELU, in stack of dense layer result self-normalize: the output of each layer will tend to preserve a mean of 0 and standar desviation of 1 during trainig, which solve the grading problem. condiction:input staandarized 0,1; initialized with LeCu; architecture must be sequential,all layer must be dense

```
keras.layers.Dense(10, kernel_initializer="he_normal")
keras.layers.LeakyReLU(alpha=0.2)\PReLU()
%
keras.layers.Dense(10, activation="selu", kernel_initilizer="lecun_normal")
```

Bach Normalization

The before not guarantee that gradient come back during trainig. Sergey and Christian proposed BN, consist of adding an operation in the model just before of after the activation fuction of each hidden layer. The algorithm needs to estimate each input's mean and standar desviation (does over mini-bach)

$$\mu_B = \frac{1}{m_B}\sum_{i=1}^{m_B} x^{(i)} \tag{15.4}$$

$$\sigma_B^2 = \frac{1}{m_B}\sum_{i=1}^{m_B} \left(x^{(i)-\mu_B}\right)^2 \tag{15.5}$$

$$\hat{x}^{(i)} = \frac{x^(i) - \mu_B}{\sqrt{\sigma_B^2} + \epsilon} \tag{15.6}$$

$$z^{(i)} = \gamma \cdot \hat{x}^{(i)} + \beta \tag{15.7}$$

γ (outputscaler vector), β (output offset vector) are learned through regular backpropagation, and μ (the final input mean vector) and σ (the final input standar desviation vector) are stimated using an exponential moving average (estimed during training but used after).

The vanishing gradients problem was strongly reduced; BN acts like a regularizer; there is a runtime penalty (evoiding updating the previos layer's weight and biases).

The aoutor of the BN paper arqued in favor of adding the BN layers before the activation function, rather than after. There is some debate about this, as which is preferable seems to depend on the task. Since BN layer include one offset parameter per input, you can remove the bias term from the previos layer.

momentum this parameter is used by the BN layer when it updates the exponential moving averages.

BN has become one of the most-used layers in deep neural networks.[2]

```
model = keras.models.Sequatial([
```

[2]Now fixed-updated by Hogyi Zhang is view recently

```
keras.layers.Flatten(input_shape=[28,28]),
keras.layers.BatchNormalization(),
keras.layers.Dende(300, kernel_initializatizer="he_normal", use_bias=False),
keras.layers.BatchNormalization(),
keras.layers.Activation("elu")
keras.layers.Dende(100, kernel_initializatizer="he_normal", use_bias=False),
keras.layers.BatchNormalization(),
keras.layers.Activation("elu"),
keras.layers.Dense(10, activation="sofmax")
])
```

Gradient Clipping

```
optimizer= keras.optimizers.SGD(clipvalue=1.0)
model.compile(loss="mse", optimizer=optimizer)
```

15.2. Reusing Petrained Layer

15.3. Faster Optimizationers.

15.4. Overfitting.

Even though Batch Normalization was designed to solve the unstable gradients problems, it also acts like a pretty good regularizer.

L1,L2 regularization

```
layer = keras.layers.Dense(100, activation="elu", kernel_initializer="he_normal",
kernel_regularizer=keras.regularizers.l2(0.01))
```

```
from functools import partial
RegularizedDense = partial(
keras.layers.Dense,
activation="elu",
kernel_initializar="he_normal",
kernel_regularizer=keras.regularizers.l2(0.01)
)
model = keras.models.Sequential([
keras.layers.Flatten(input_shape=[28,28]),
RegularizedDense(300),
RegularizedDense(100),
RegularizedDense(10,activation="softmax",
kernel_initializer="glorot_uniform")
])
```

Dropout

At every training step, every neuron(including the input neurons, but always excluding the output neurons) has a probability p of being temporarily "dropped out", meaning it will be entirely ignored during this training step, but it may be active during the next step. The hyperparameter p is called the dropout rate, and it is typically set between $\%10 - \%50$. The state-of-the-art neural networks get $1 - 2\,\%$ accuracy boost simply by adding dropout. Also. we need to multiply each input connection weight by the keep probability (1-p) after training. Alternatively, we can divide each neuron's output by the keep probability during training(don't equvalent but they work equally well).

```
model = keras.model.Sequential([
keras.layers.Flatten(input_shape=[28,28]),
keras.layers.Dropout(rate=0.2),
keras.layers.Dense(300, activation="elu",kernel_initializer="he_normal"),
keras.layers.Dropout(rate=0.2),
keras.layers.Dense(100, activation="elu", kernel_initializer="he_normal"),
keras.layers.Dropout(rate=0.2),
keras.layers.Dense(10,activation="softmax")

])
```

If you observed that the model is overfitting, you can increase the dropout rate.Conversly, you should try decreasing the dropout rate if the model underfits the training set. It can also help to increase the dropout rate for large layers, and reduce it for small ones. Moreover, many state-of-the-art architectures only use dropout after the last hidden layer, so you may want to try this if full dropout is too strong. Dropout does tend to significanty slow down convergence, but it usually result in a much better model when tuned properly. So, it is in generally well worth the extra time and effort.

observation Dropout is only active during training, made sure to evaluate the training loss without the dropout (after training).

MC Dropout

Max-Norm Regularization.

15.5. Summary

Lasso, Ridge, slow learning λ	tunning parameter
dropout learning	remove fraction of unit per layer and scaler weight
early stopping, epoch, batch size	details of stochastic gradient descent
number of layer, and unit fro layer	

15.6. bibliografy

2014 paper Nitish Srivastava (dropout)

Parte VI

Optimization (opt model).

Capítulo 16

First order methods.

16.1. Adagrad.

16.2. Adam.

Capítulo 17

Second order methods.

17.1. Newton's method.

17.2. Secant method.

Capítulo 18

Direct Methods.

18.1. Nelder-mead simplex method.

Capítulo 19

Stochastic methods.

19.1. Noise descent.

19.2. Cross-Entropy method.

Parte VII

Regularization (overfitting).

Parte VIII

Metrics (evaluation)

Capítulo 20

Regression

Capítulo 21

Classification

- Confusion matrix
- ROC.
- PR
- Gain and lift chart.

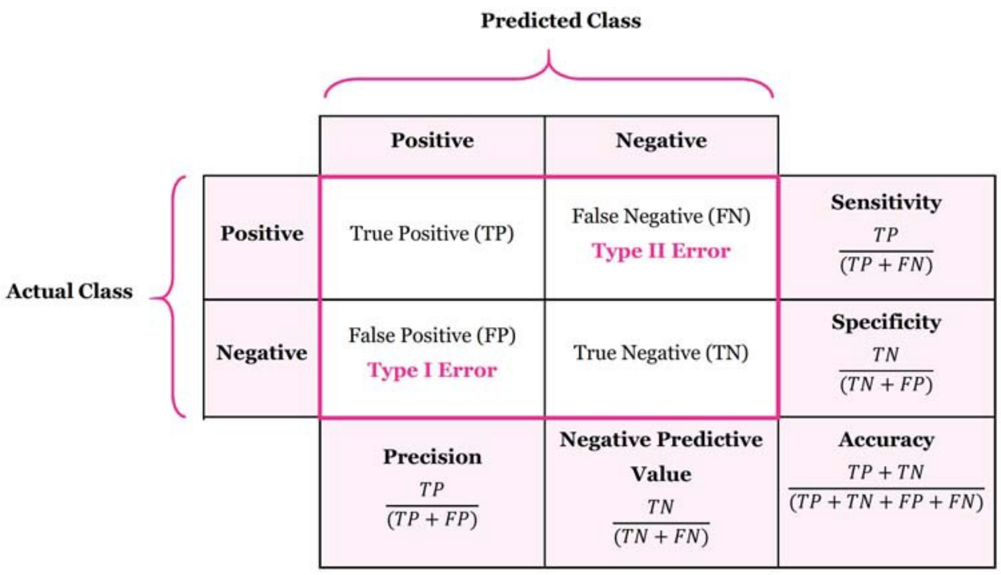
21.1. Matrix confusion

Fundamentally, the assessment process atempts to learn which model produces the most accurate and useful predictions.

When 1s are rare, the ratio of false positives to all predicted positives can be high, leading to the unintuitive situation in which a predicted 1 is most likely a 0.[1]

In inbalance classes the rare class is usually the class of more interestet and is typically designed 1, in contrast to more prevalent 0s. The 1s are the more important case, in the sense that misclassifying them as 0s is costlier than misclassifying 0s as 1s (exp legitimate insurance claims 0, fraudulent 1). The most accurate classification model may be one that simply classifies everything as a 0.

[1]Default desicion point or cutoff is 0.5



		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Figura 21.1: Matrix cunfusion, sensitivity is recall.

21.2. ROC curve.

The Geometric Mean or G-Mean is a metric for imbalanced classification that, if optimized, will seek a balance between the sensitivity and the specificity.

$$G - mean = \sqrt{Sensitivity \cdot Specificity} \tag{21.1}$$

Given that we have already calculated the Sensitivity (TPR) and the complement to the Specificity when we calculated the ROC Curve, we can calculate the G-Mean for each threshold directly.

It turns out there is a much faster way to get the same result, called the Youden's J statistic.

$$J = Sensitivity + Specificity\breve{}1 \tag{21.2}$$

We can then choose the threshold with the largest J statistic value

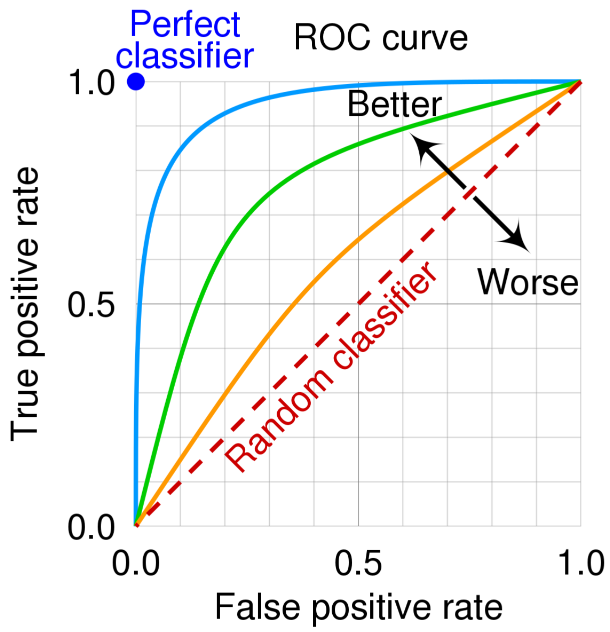
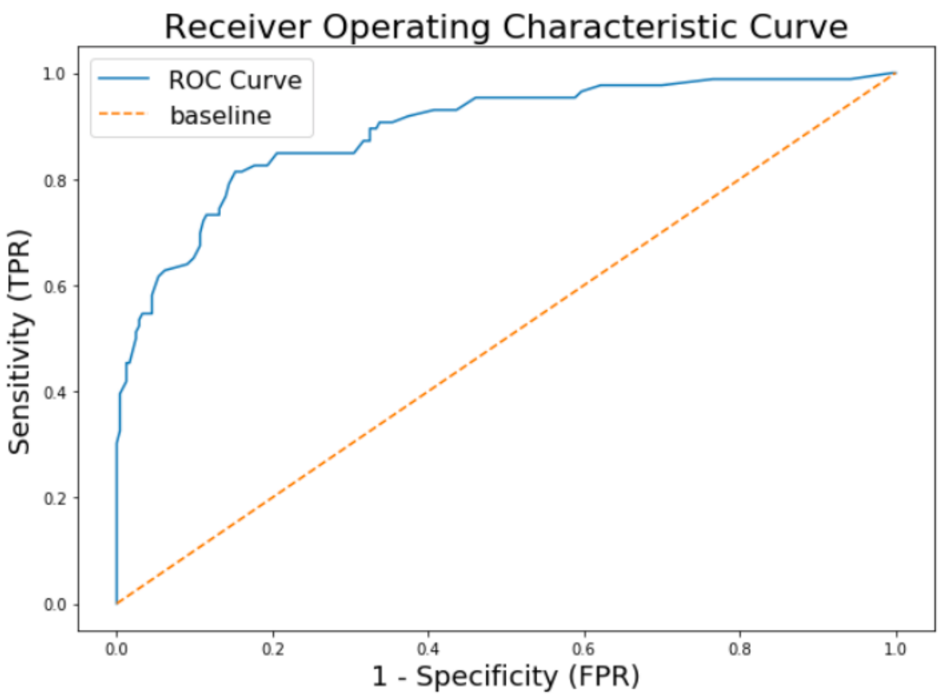


Figura 21.2: If ROC (sensity-recall) hugs the upperleft corner it will identify lots of 1s without misclassifying lots of 0s as 1s. Prevalence is y=1 / total. false positive rate is 1-specifity.



21.3. PR curve

PR curves are specially useful in evaluating data with highly unbalanced out-comes.

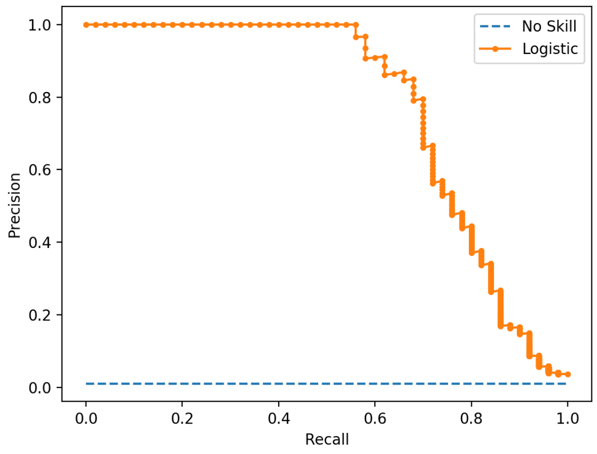


Figura 21.3:

Fuentes A Gentle Introduction to Threshold-Moving for Imbalanced Classification

If we are interested in a threshold that results in the best balance of precision and recall, then this is the same as optimizing the F-measure that summarizes the harmonic mean of both measures.

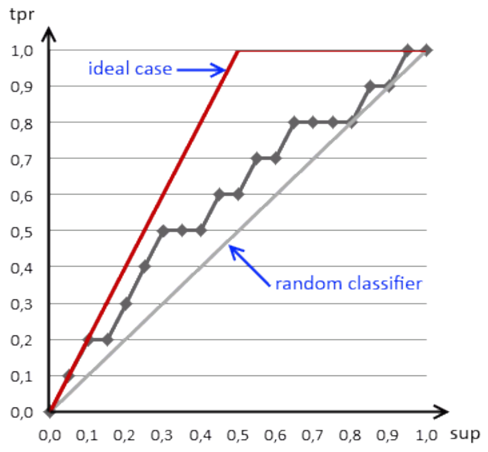
$$F - Measure = \frac{(2 \cdot Precision \cdot Recall)}{(Precision + Recall)} \tag{21.3}$$

first calculates the F-measure for each threshold, then locates the score and threshold with the largest value.

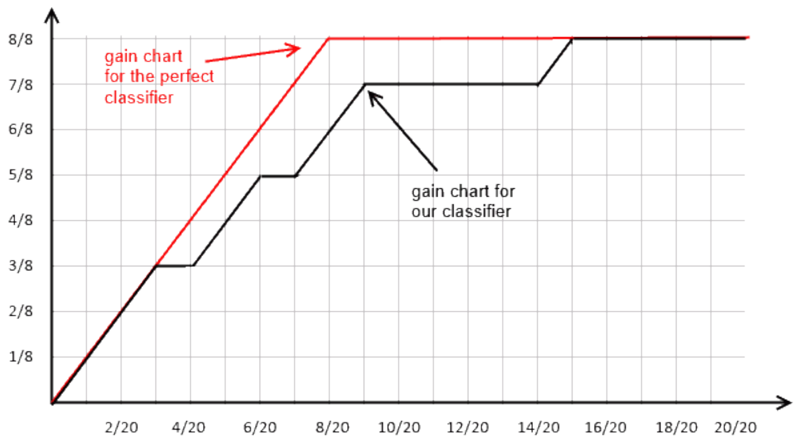
21.4. Gain chart and lift chart.

Gain and Lift charts are two approaches used while solving classifications problems with imbalanced data sets. The gain chart and lift chart is the measures in logistic regression that will help organizations to understand the benefits of using that model.Cumulative Gain Chart

$$Gain = \frac{CumulativeNumberOfPositiveObservationsUptodecibels}{TotalNumberOfPositiveObservationInData} \tag{21.4}$$



it says how much population we should sample to get the desired sensitivity of our classifier i.e. if we want to direct 40 % of potential repliers to our targeting campaign, we should select 20



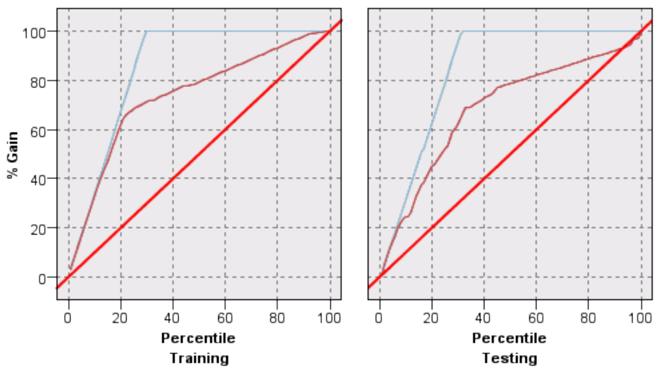


Figura 21.4: we can easily see if a classifier overfits on the test set, but underperforms on the testing

$$lift = \frac{Cumulative number of positive observations up to decile i using ML model}{Cumulative Number of Positive Observations up to Decile i using Random Model.} \tag{21.5}$$

falta analisar el lift, el libro donde encontre es el data mining for eider franck, revisar luego, usare lo mencionado en ROC, PR y el tratamiendo de datos no balanceados por mietras, el lift es para saber los costos, vere mas adelante.

21.5. K-S Chart

21.6. bibliografia.

A Gentle Introduction to Threshold-Moving for Imbalanced Classification
Cumulative Gain Chart
Practical statistical for data sicience, peter bruce and andrew bruce

Parte IX

Python.

Capítulo 22

sklearn.preprocessing

Preprocessing and Normalization
The sklearn.preprocessing module includes scaling, centering, normalization, binarization methods.

- sklearn.preprocessing.StandardScaler().

$$\frac{x_i - mean(x)}{sd(x)} \tag{22.1}$$

- sklearn.preprocessing.MinMaxScaler().

$$\frac{x_i - min(x)}{max(x) - min(x)} \tag{22.2}$$

- sklearn.preprocessing.RobustScaler().

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)} \tag{22.3}$$

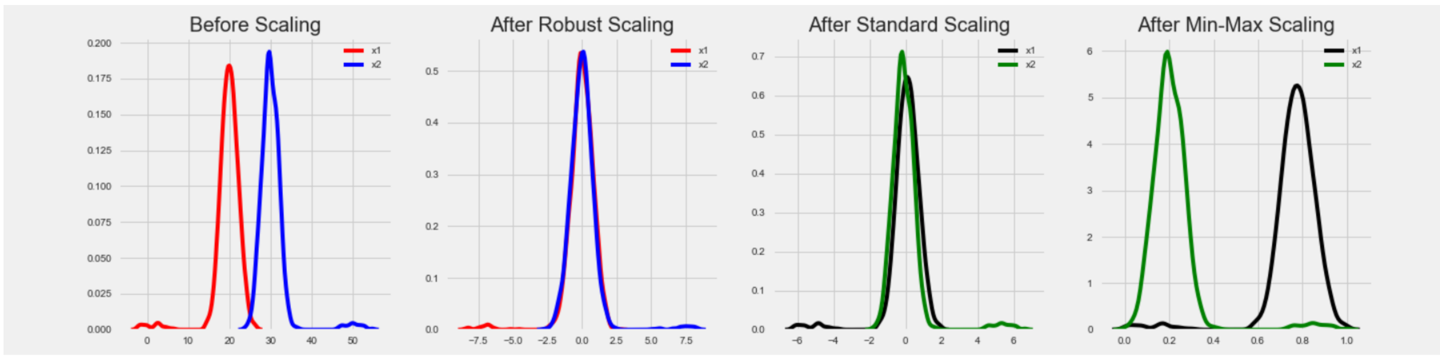


Figura 22.1: from Geeksfor geeks.

Capítulo 23

sklearn.pipeline

The sklearn.pipeline module implements utilities to build a composite estimator, as a chain of transforms and estimators.

Capítulo 24

sklearn.ensemble

The sklearn.ensemble module includes ecnsemble-based methods for classification, regression and anomaly detection.

24.1. sklearn.emsemble.RandomForestClassifier

Capítulo 25

sklearn.model_selection

See the Cross-validation: evaluating estimator performance, Tuning the hyper-parameters of an estimator and Learning curve sections for further details.

- sklearn.model_selection.train_test_split

Capítulo 26

sklearn.metrics

The sklearn.metrics module includes score functions, performance metrics and pairwise metrics and distance computations.

- metrics.classification_report
- metrics.confusion_matrix
- metrics.roc_curve
- metrics.roc_auc_score

Capítulo 27

tf.keras.model

Model groups layers into an object with training and inference features.

Capítulo 28

ft.keras.layer

Parte X

R.

Capítulo 29

ggplot

Capítulo 30

dplyr

Capítulo 31

tidyr

Parte XI

SQL.

Parte XII

Julia.

Parte XIII

C++.

Bibliografía

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer series in statistics, Springer, 2009.

[2] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.