**Instituto Superior de Engenharia**
Politécnico de Coimbra

# META 2
## SUPERVISED E UNSUPERVISED LEARNING

Departamento De Engenharia Informática E De Sistemas
Mestrado Em Engenharia Informática
UC: Machine Learning
2022/2023
André Proença - 2016018783
Isabel Catarina Castro – 2018013160

# Índice

# Introdução

Este relatório tem como objetivo descrever a abordagem sistemática aplicada aos modelos de utilizados na aplicação de algoritmos de aprendizagem supervisionada e não-supervisionada sobre o *dataset* escolhido para a primeira meta do trabalho prático. Foi criado para este fim, uma lista de todos os algoritmos aplicados no projeto, a explicação da sua utilização e quais os seus resultados. Por fim, apresentámos as conclusões para cada um dos algoritmos de aprendizagem.

# Supervised Learning

Visto que o dataset utilizado se trata de um problema de classificação binária, foi feita uma seleção dos algoritmos que, na nossa opinião, poderão ter melhores resultados em comparação com os demais.

Os algoritmos selecionados foram:

### Logistic Regression
Mede a relação entre a variável dependente/target no contexto do problema, e uma ou mais variáveis independentes, estimando as probabilidades através de uma função logística. Este método é frequentemente utilizado para problemas de classificação binária, utilizando conceitos de estatística e probabilidade.

### SVM
Algoritmo versátil e eficaz em problemas com grande dimensão

### KNN
Um dos algoritmos fundamentais em Machine Learning, sendo um dos mais simples e usado especialmente em problemas de classificação

### Naïve Bayes
Tem vantagens em relação a outros. Por exemplo, converge mais rapidamente que a regressão logística e em relação ao Random Forest, o tamanho do modelo é menor

### Random Forest
Adequado para datasets onde existe grande conjunto de dados, como é o caso

### Decision Trees
Algoritmo muito popular, utilizado como algoritmo de treino para Supervised Learning, nomeadamente em problemas de classificação binária

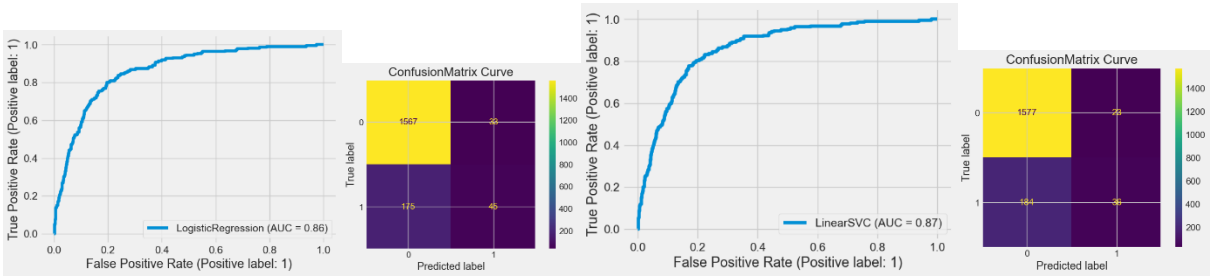
## Resultados dos modelos

De seguida são apresentados os resultados de cada classificador com os parâmetros *default*.

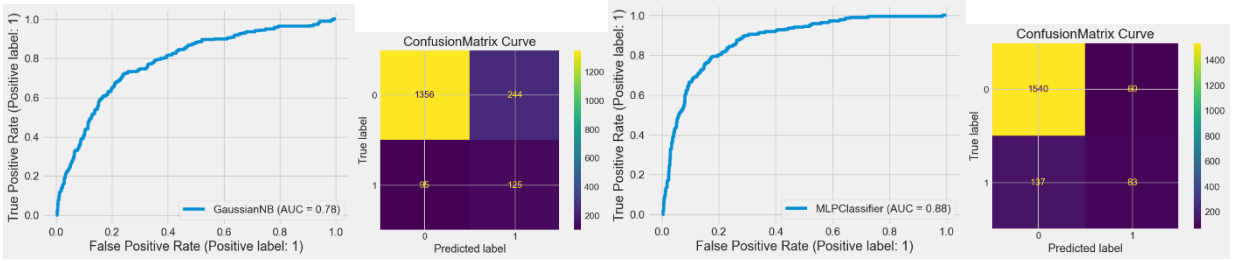
| | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Logistic Regression | 0.88 | 0.47 | 0.21 | 0.29 |
| Linear SVM | 0.88 | 0.46 | 0.15 | 0.22 |
| Gaussian Naive Bayes | 0.80 | 0.28 | 0.45 | 0.34 |
| MLP Classifier | 0.89 | 0.51 | 0.41 | 0.45 |
| Decision Tree Classifier | 0.89 | 0.56 | 0.34 | 0.43 |
| KNeighbors Classifier | 0.89 | 0.55 | 0.26 | 0.35 |
| Random Forest Classifier | 0.90 | 0.67 | 0.29 | 0.41 |

### Gráfico da performance de cada classificado

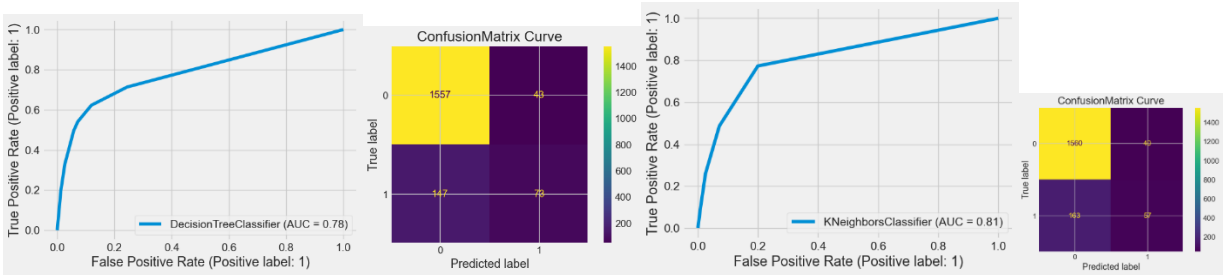
#### LogisticRegression e LinearSVC



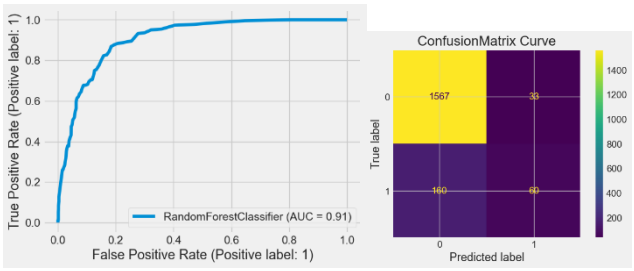

#### NB Gaussian e NN MLP




#### DecisionTree e Kneighbors



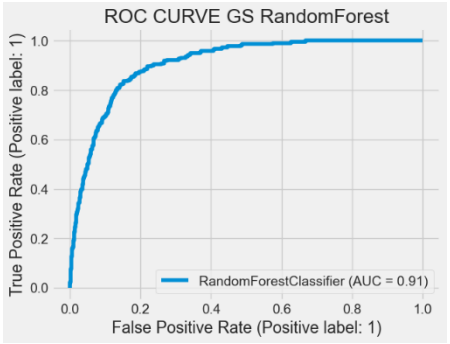

## RandomForest




## Grid Seach

Através do *gridsearch* fomos tentar obter os melhores valores para os *hiper parâmetros* de alguns dos algoritmos que tínhamos corrido previamente com os valores *default*. Os resultados estão descritos abaixo.

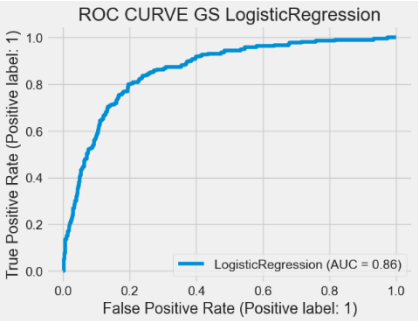
## Random Forest




```
Melhores Hyperparametros: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 200}
Accuracy: 0.8972527472527473
Precision: 0.6774193548387096
Recall: 0.2863636363636364
F1: 0.40255591054313095
```

```
param_grid = {
    'n_estimators' : [10, 50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

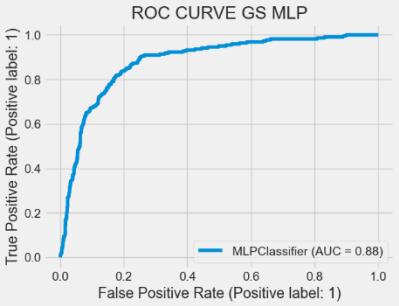
### Logistic Regression




```
Melhores Hyperparametros: {'C': 5}
Accuracy: 0.8846153846153846
Precision: 0.5625
Recall: 0.20454545454545456
F1: 0.3
```

```
param_grid = {
    'C': [0.1, 1, 5, 10, 20, 30, 50, 80, 100]
}
```

### NN MLP




```
Melhores Hyperparametros: {'alpha': 0.001, 'hidden_layer_sizes': (10,), 'learning_rate_init': 0.01}
Accuracy: 0.8868131868131868
Precision: 0.5406976744186046
Recall: 0.42272727272727273
F1: 0.4744897959183673
```

```
param_grid = {
    'hidden_layer_sizes': [(10,), (50,), (100,)],
    'alpha': [0.0001, 0.001, 0.01],
    'learning_rate_init': [0.001, 0.01, 0.1]
}
```

## Emsemble techniques

Nesta parte vamos combinar os nossos classificadores com técnicas de *Emsemble*, neste caso foram apenas utilizadas duas técnicas o *voting* que assim como o nome indica é uma votação de cada algoritmo e o *Adaboost* que a cada iteração vai mudando os pesos de cada elemento dos nossos dados. Tornando todas essas iterações numa única final onde cada iteração vota o peso de cada elemento do nosso conjunto de dados.

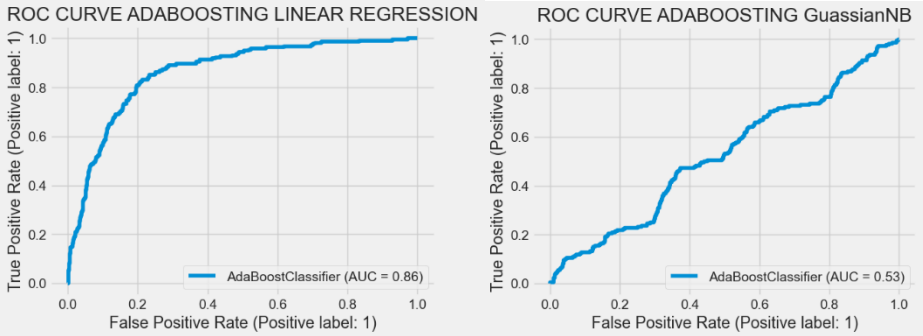
### Voting

Através dos melhores valores encontrados no Gridseach para os classificadores (mlp,logistic regression, random forest)

```
Accuracy: 0.8906593406593407
Precision: 0.5981308411214953
Recall: 0.2909090909090909
F1: 0.39143730886850153
```

AdaBoost – Boosting

Logistic regression e NaiveBayes Guassian




## Undersampling

```
------------------------RANDOM FOREST----------------------
[UNDERSAMPLED_DATASET]Accuracy: 0.8204545454545454 [NORMAL_DATASET]Accuracy: 0.8972527472527473
[UNDERSAMPLED_DATASET]Precision: 0.7974683544303798 [NORMAL_DATASET]Precision: 0.6774193548387096
[UNDERSAMPLED_DATASET]Recall: 0.8590909090909091 [NORMAL_DATASET]Recall: 0.2863636363636364
[UNDERSAMPLED_DATASET]F1: 0.8271334792122538 [NORMAL_DATASET]F1: 0.40255591054313095


------------------------LINEAR REGRESSION----------------------
[UNDERSAMPLED_DATASET]Accuracy: 0.5363636363636364 [NORMAL_DATASET]Accuracy: 0.8846153846153846
[UNDERSAMPLED_DATASET]Precision: 1.0 [NORMAL_DATASET]Precision: 0.5625
[UNDERSAMPLED_DATASET]Recall: 0.07272727272727272 [NORMAL_DATASET]Recall: 0.20454545454545456
[UNDERSAMPLED_DATASET]F1: 0.13559322033898305 [NORMAL_DATASET]F1: 0.3


------------------------MLP REGRESSION----------------------
[UNDERSAMPLED_DATASET]Accuracy: 0.8159090909090909 [NORMAL_DATASET]Accuracy: 0.8868131868131868
[UNDERSAMPLED_DATASET]Precision: 0.8008658008658008 [NORMAL_DATASET]Precision: 0.5406976744186046
[UNDERSAMPLED_DATASET]Recall: 0.8409090909090909 [NORMAL_DATASET]Recall: 0.42272727272727273
[UNDERSAMPLED_DATASET]F1: 0.8203991130820399 [NORMAL_DATASET]F1: 0.4744897959183673
```

Neste fase tentamos comparar os melhores valores encontrados através do *gridsearch* para os classificadores (*mlp,linear regression e random forest*) e com os melhores valores encontrados também com o *gridsearch* mas neste caso fazendo *undersampling* ao *target*, pelo facto de ser bastante desbalanceado.

## Unsupervised Learning Clustering

Todo o tratamento dos dados para este tipo de algoritmos foi o mesmo que foi aplicado ao algoritmo anterior, sendo que esta decisão é apoiada pela documentação referente aos algoritmos de *clustering*.

Estes algoritmos (Birch,KMeans) foram aplicados de duas formas, isoladamente ou como pré-processamento para tentar melhorar os resultados dos classificadores anteriores. Os restantes apenas foram utilizados de forma isolada.

Para tal, tivemos em conta um dos melhores classificador dos algoritmos Supervisionados, o KNN.

Para cada um dos algoritmos foi calculado as seguintes métricas: *'ARI', 'AMI', 'Homogenity', 'Completeness', 'V-measure', 'Silhouette'*

### KMeans

Em relação a este algoritmo, foi calculado o silhoute score de forma isolada assim como quando aplicado como pré processamento num pipeline. Os resultados foram os seguintes:

```
Average silhouette score:  0.20907216152580915
Accuracy Pipeline: 0.8708791208791209
Pipeline silhouette score:  0.10048579113813007
```

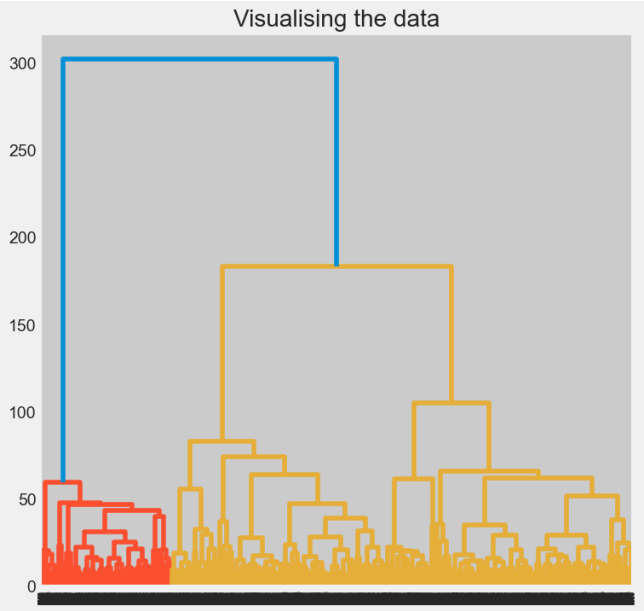
Como podemos comprovar pelos valores encontrados, apesar da accuracy no pipeline implementado ser razoável, podemos verificar que o valor da Silhouette Score no pipeline baixou em relação à implementação isolada

### BIRCH

Em relação a este algoritmo, foi calculado o silhoute score de forma isolada assim como quando aplicado como pré processamento num pipeline. Os resultados foram os seguintes:

```
Birch
-------------
Accuracy Pipeline: 0.8895604395604395
Pipeline silhouette score:  0.09893757452870304
```

## Agglomerative




### Métricas calculadas de forma isolada

Devido a dificuldades na implementação, estes valores mostram que não foi possível associar corretamente as *labels* do dataset.

| | ARI | AMI | Homogenity | Completeness | V-Measure | Silhouette |
|---|---|---|---|---|---|---|
| KMeans | 0.0 | 0.0 | 0.01 | 0.00 | 0.00 | 0.21 |
| BIRCH | 0.0 | 0.0 | 0.00 | 0.00 | 0.00 | 0.20 |
| Agglomerative | 0.0 | 0.0 | 0.00 | 0.02 | 0.00 | 0.29 |
| Spectral | 0.02 | 0.0 | 0.01 | 0.00 | 0.00 | 0.19 |

Segundo o Silhouette Score, o algoritmo com melhores resultados a nível individual foi o Agglomerative Clustering.

## Conclusão

Em relação aos algoritmos de Supervised Learning, apesar da nossa accuracy ter baixado, em geral os valores dos *scores* de *undersampling* foram melhores que os do *dataset* inicial(desbalanceado), isto pelo facto de a accuracy apenas medir a taxa de acerto do algoritmo, um modelo poucos valores da classe positiva vai ter sempre a accuracy alta, mas isto porque o algoritmo apenas prevê a classe negativa, e neste caso o modelo torna-se pouco útil pois não consegue identificar os exemplos positivos é por isso que é preciso olhar para uma variedade de métricas de avaliação e não só para a taxa de acerto dos algoritmos.

Para conseguir melhorar a previsão destes algoritmos para o nosso problema, teríamos de ter mais dados com elementos da classe positiva (clientes que iriam subscrever um deposito bancário).

Uma métrica bastante melhor que a accuracy para avaliar os nossos algoritmos é o *AUC* (área abaixo da curva), mede a capacidade de um classificador distinguir entre duas classes, e é calculado através do *true positive ratio* e do *false positive ratio*, quando os valores estão próximos de 1 indicam um bom modelo, quando estão próximos de 0.5 indicam que o modelo é aleatório.

Em relação, ao modelo de Unsupervised Learning devido a problemas no cálculo da *accuracy* não foi possível confirmar qual o algoritmo com melhor resultado. Assim sendo, apenas a métrica da *Silhoute Score* nos deu uma ligeira ideia dos valores que poderiam ter sido encontrados.