Combinação de Modelos

Departamento de Engenharia Informática (DEI/ISEP)
Fátima Rodrigues
mfc@isep.ipp.pt

Combinação de Modelos

Ideia Base

- Constrói-se um conjunto de modelos a partir de dados de treino
- Fazem-se previsões combinando as previsões feitas pelos vários modelos

Razões para combinar modelos:

- Maior habilidade preditiva
- Maior robustez, previsões mais estáveis

Porquê Combinação de Modelos?

Resultados teóricos e experimentais comprovam que não existe nenhum algoritmo que apresente um melhor desempenho para todos os problemas

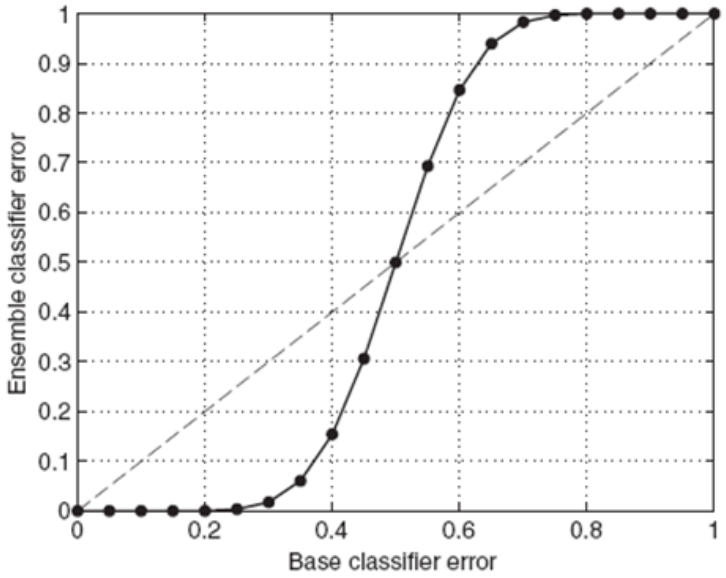
A ideia principal subjacente à combinação de modelos é baseada na observação de que diferentes algoritmos de aprendizagem exploram:

- Diferentes linguagens de representação
- Diferentes espaços de procura
- Diferentes funções de avaliação de hipóteses

A **Combinação de Modelos** visa explorar estas diferenças e desenvolver uma combinação de modelos que trabalhando em conjunto obtêm melhor desempenho do que cada um dos modelos individuais

Justificação para Combinação de Modelos

Duas condições são necessárias para que a combinação de modelos seja superior aos modelos individuais:



- Os modelos individuais devem ser **independentes** entre si, ou seja, cometem erros não correlacionados, ou negativamente correlacionados

↪Combinar modelos idênticos é inútil

- A previsão dos classificadores individuais deve ser **superior à previsão aleatória** (taxa de erro < 50%)

Justificação para Combinação de Modelos

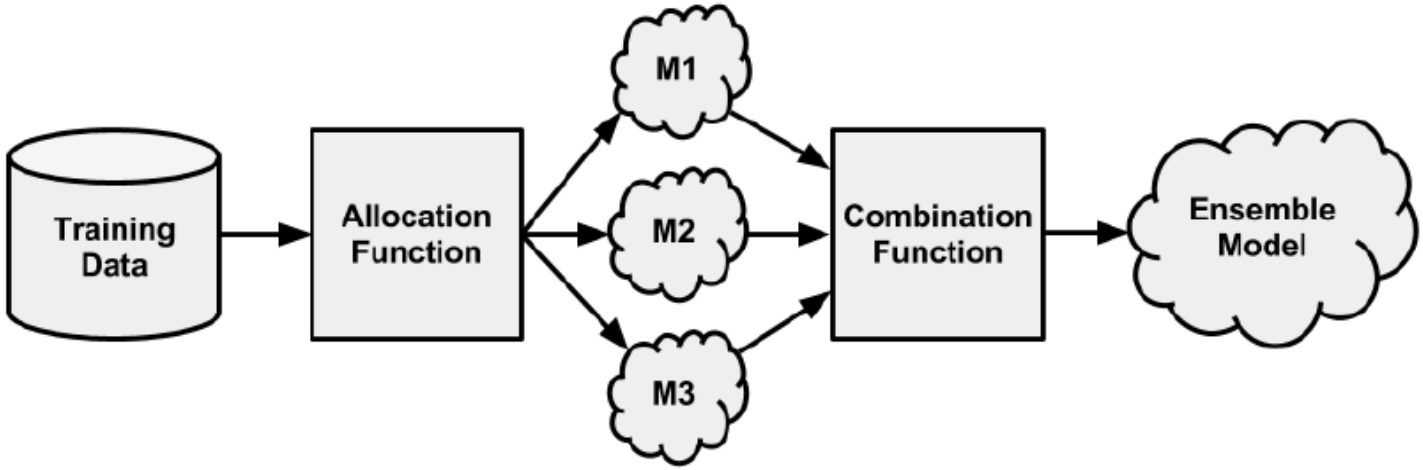
- Supondo que são construídos 25 classificadores base
- Cada classificador apresenta uma taxa de erro , **ε = 0.35**

- Se os classificadores base forem **idênticos**:
  - a combinação dos classificadores fará previsões idênticas às dos classificadores base, e apresenta idêntica taxa de erro (ε = 0.35)

- Se os classificadores base forem **independentes**
- A **taxa de erro** dos classificadores individuais for menor 50%
  - A combinação dos classificadores fará uma previsão errada se pelo menos metade dos classificadores base prever erradamente, e portanto apresentará uma taxa de erro:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Fases na Combinação de Modelos



1. **Diversificação:** escolhem-se diferentes modelos para cobrir diferentes regiões do espaço de procura

2. **Integração:** combinam-se os vários modelos por forma a maximizar o desempenho do modelo combinado

Técnicas de Diversificação

Amostragem

Varia os dados usados para treinar o algoritmo

Combinação homogénea

Um **único algoritmo** de aprendizagem é usado para desenvolver modelos sobre **diferentes sub-conjuntos de dados**

Diferentes algoritmos

Varia os algoritmos treinados sobre o conjunto de dados

Combinação heterogénea

Usa **diferentes algoritmos** de aprendizagem

Técnicas de Integração

- **Estática:** o procedimento de integração é fixo por exemplo:
  - votação (Max-Voting)
  - média/moda das previsões individuais
  - média pesada

- **Dinâmica:** as previsões base são combinadas através de um procedimento adaptativo - **meta-aprendizagem**

Combinação Heterogénea

Aprendizagem Multi-Estratégica

Diversificação

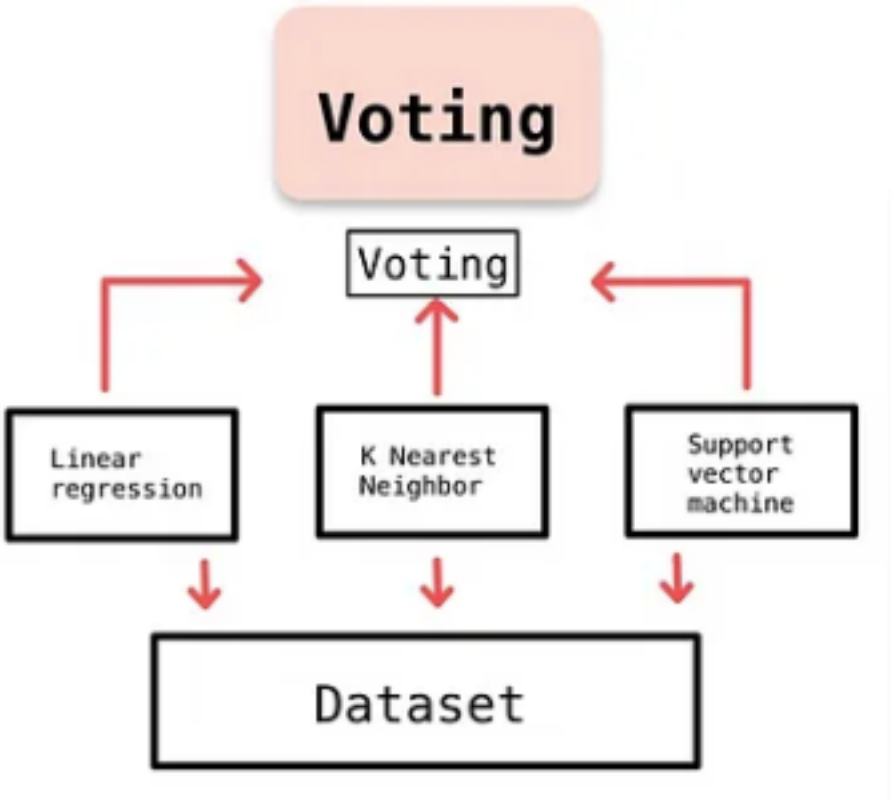
- Treinar M algoritmos de aprendizagem em subconjuntos de um conjunto de treino e medir o seu desempenho num conjunto de teste
- Selecionar J modelos com **erro de correlação mínimo**

Integração

- **Dinâmica:** através de **meta-aprendizagem**

Combinação por votação

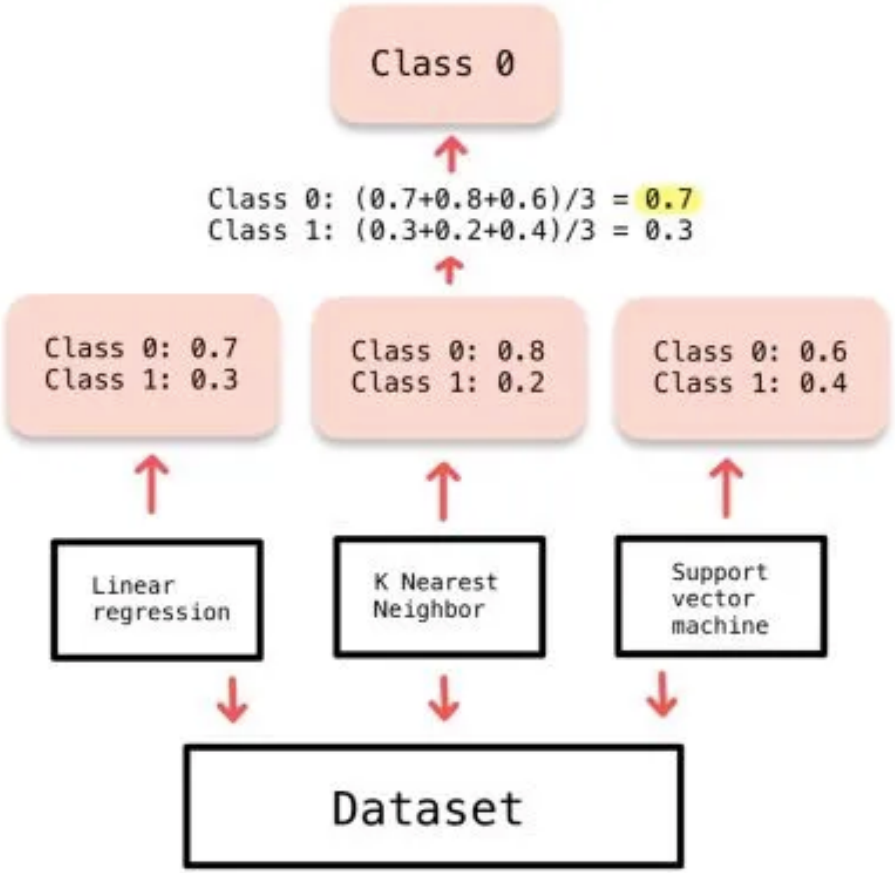
Combinação de modelos por votação



Combina previsões de diferentes modelos, independentes

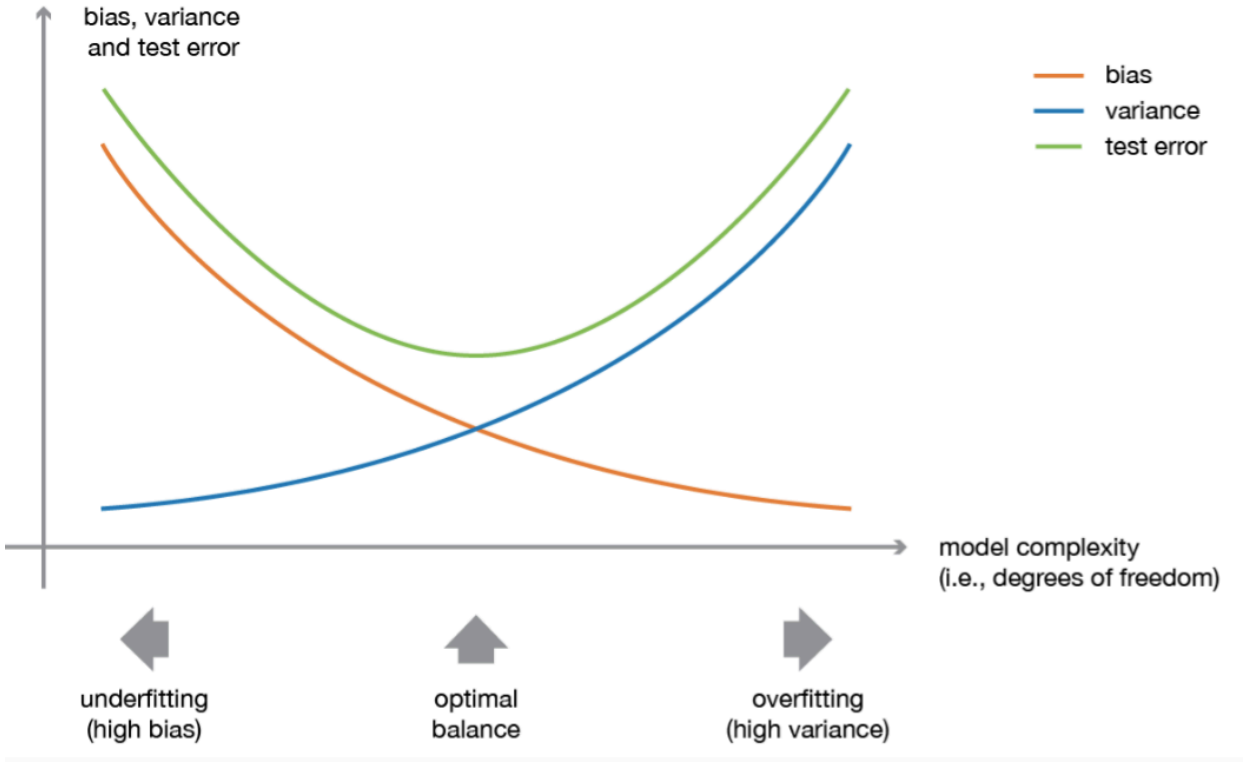
Combinação: hard/soft voting

- **Hard Voting** (votação forte): é equivalente à votação mioritária
- **Soft Voting** (votação suave): é a média da saída dos vários modelos



Combinação por stacking

bias-variance tradeoff



- Modelos base (high variance)
- Modelos de base (High bias)

O método de combinação deve contrariar a tendência dos modelos base

Stacking

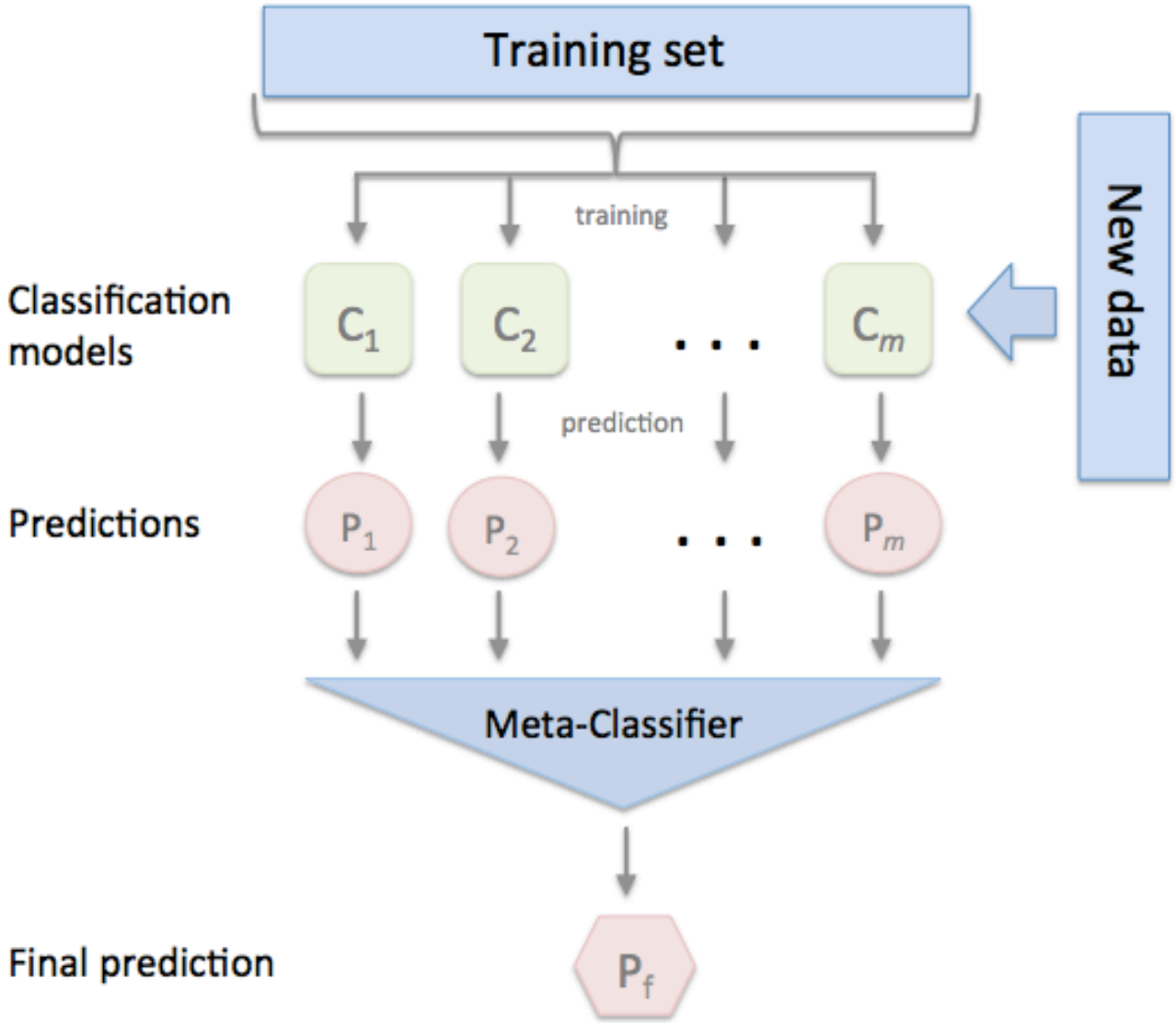
Stacking envolve:

Modelos de nível 0 (modelos base): modelos desenvolvidos com os dados de treino e cujas previsões são compiladas

Modelo de Nível 1 (meta-modelo): modelo que aprende como combinar melhor as previsões dos modelos básicos.

O meta-modelo é treinado com as previsões feitas pelos modelos base com os dados de teste

Stacking – representação conceptual



Combinação de Modelos por Amostragem

Combinação de Modelos por Amostragem

Esta técnica funciona bem com **algoritmos de aprendizagem instáveis**, ou seja, algoritmos cuja saída sofre grandes alterações em resposta a pequenas mudanças nos dados de treino: árvores de decisão, regressão, redes neuronais

Métodos mais usuais baseados em amostragem dos exemplos treino:

- **Bagging**: "bootstrap agregação" (Breiman 1996)
- **Boosting**: reamostragem através de ponderação adaptativa

Homogeneous Parallel Ensembles

Bagging

Bagging = Bootstrap AGGregatING

O Bagging consiste em duas etapas:

1. durante o treino:

   amostragem bootstrap ou amostragem com substituição é usada para gerar diferentes subconjuntos de treino, extraídos do conjunto de dados original

   ↪ garante que os modelos base treinados são diferentes

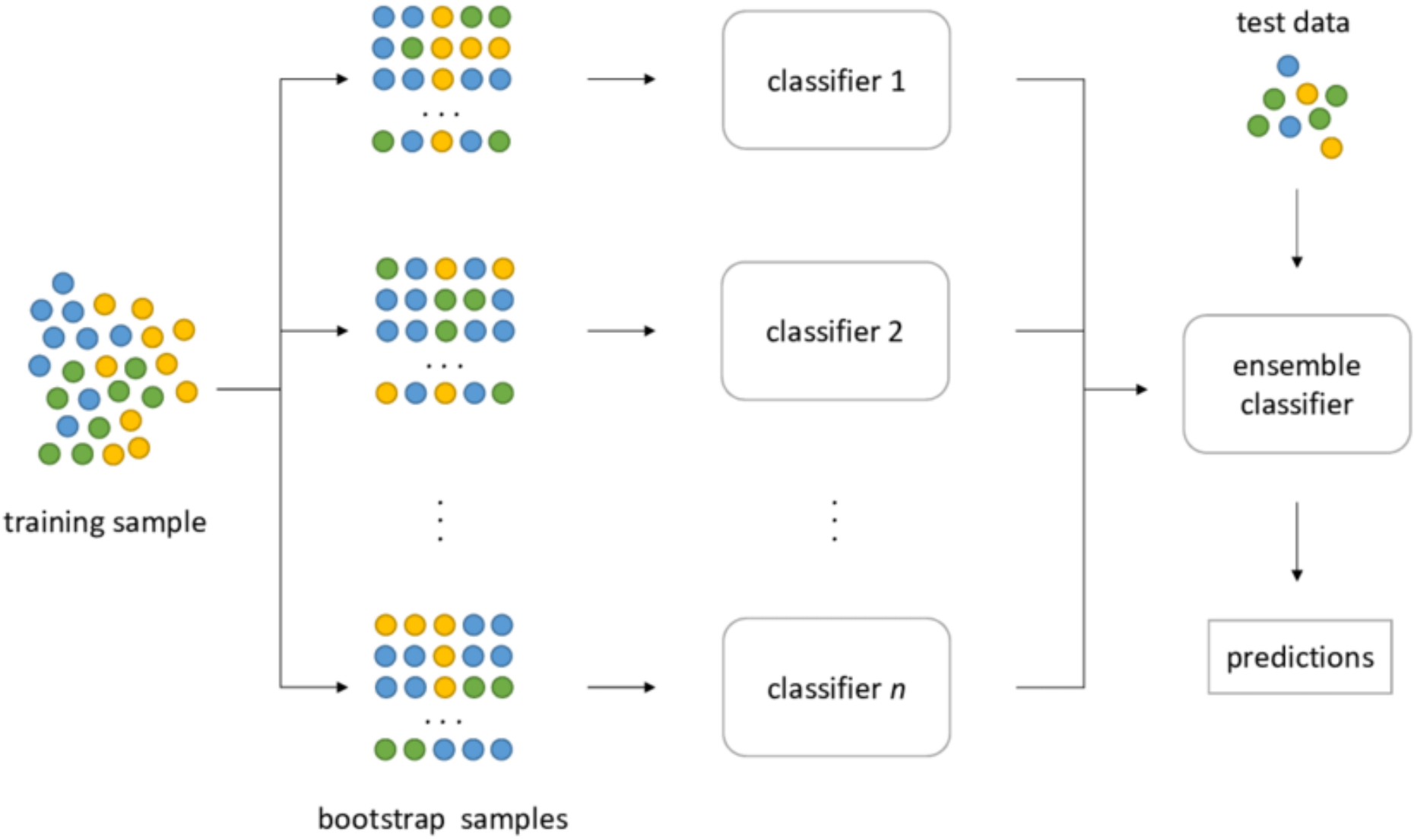
2. durante a previsão:

   são combinadas as previsões dos modelos base individuais

   Classificação: a previsão final é obtida pelo **voto maioritário** dos modelos base individuais

   Regressão: a previsão final é obtida pela **média** das previsões dos modelos base individuais

Bagging


test data
classifier 1
classifier 2
classifier n
ensemble classifier
predictions
training sample
bootstrap samples

Exemplificação Bagging

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 0.35$ $x \leq 0.75$

Sem *bagging* a melhor divisão que se pode conseguir deste conj^{to} de dados é $x \leq 0.35$ ou $x \leq 0.75$

que equivale a uma *accuracy* de **70%**

Aplicando o método *bagging* a este conjunto de dados, usando 10 amostras *bootstrap*

Exemplificação Bagging

Bagging 1	x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9	$x \leq 0.35 \Rightarrow y = 1$
	y	1	1	1	1	-1	-1	-1	-1	1	1	$x > 0.35 \Rightarrow y = -1$
Bagging 2	x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	0.2	0.3	0.8	$x \leq 0.65 \Rightarrow y = 1$
	y	1	1	1	-1	-1	1	1	1	1	1	$x > 0.65 \Rightarrow y = 1$
Bagging 3	x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9	$x \leq 0.35 \Rightarrow y = 1$
	y	1	1	1	-1	-1	-1	-1	-1	1	1	$x > 0.35 \Rightarrow y = -1$
Bagging 4	x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9	$x \leq 0.3 \Rightarrow y = 1$
	y	1	1	1	-1	-1	-1	-1	-1	1	1	$x > 0.3 \Rightarrow y = -1$
Bagging 5	x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	0.8	0.9	0.2	$x \leq 0.35 \Rightarrow y = 1$
	y	1	1	1	-1	-1	-1	-1	1	1	1	$x > 0.35 \Rightarrow y = -1$
Bagging 6	x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	0.1	$x \leq 0.75 \Rightarrow y = -1$
	y	1	-1	-1	-1	-1	-1	-1	1	1	1	$x > 0.75 \Rightarrow y = 1$
Bagging 7	x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	0.8	$x \leq 0.75 \Rightarrow y = -1$
	y	1	-1	-1	-1	-1	1	1	1	1	1	$x > 0.75 \Rightarrow y = 1$
Bagging 8	x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	0.2	$x \leq 0.75 \Rightarrow y = -1$
	y	1	1	-1	-1	-1	-1	-1	1	1	1	$x > 0.75 \Rightarrow y = 1$
Bagging 9	x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	0.3	0.9	$x \leq 0.75 \Rightarrow y = -1$
	y	1	1	-1	-1	-1	-1	-1	1	1	1	$x > 0.75 \Rightarrow y = 1$
Bagging 10	x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9	$x \leq 0.05 \Rightarrow y = -1$
	y	1	1	1	1	1	1	1	1	1	1	$x > 0.05 \Rightarrow y = 1$

Exemplificação Bagging

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	**1**	**1**	**1**	**-1**	**-1**	**-1**	**-1**	**1**	**1**	**1**
Bagging 1	1	1	1	-1	-1	-1	-1	-1	-1	-1
Bagging 2	1	1	1	1	1	1	1	1	1	1
Bagging 3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Bagging 4	1	1	1	-1	-1	-1	-1	-1	-1	-1
Bagging 5	1	1	1	-1	-1	-1	-1	-1	-1	-1
Bagging 6	-1	-1	-1	-1	-1	-1	-1	1	1	1
Bagging 7	-1	-1	-1	-1	-1	-1	-1	1	1	1
Bagging 8	-1	-1	-1	-1	-1	-1	-1	1	1	1
Bagging 9	-1	-1	-1	-1	-1	-1	-1	1	1	1
Bagging 10	1	1	1	1	1	1	1	1	1	1
Soma	2	2	2	-6	-6	-6	-6	2	2	2
Previsão	**1**	**1**	**1**	**-1**	**-1**	**-1**	**-1**	**1**	**1**	**1**

Accuracy: **100%**

Bagging:

- reduz a variância dos classificadores de base
- é menos susceptível a sobre-ajustamento dos modelos

Bagging

- Amostragem Bootstrap produz conjuntos de treino altamente sobrepostos

- Reduz o erro devido à **diminuição da variância**: melhora consideravelmente algoritmos altamente instáveis ou algoritmos com alta variância, isto é, onde pequenas variações de dados conduzem a modelos muito diferentes (por exemplo árvores de decisão, redes neuronais)

Algoritmos Bagging:

- Bagging
- Random Forest

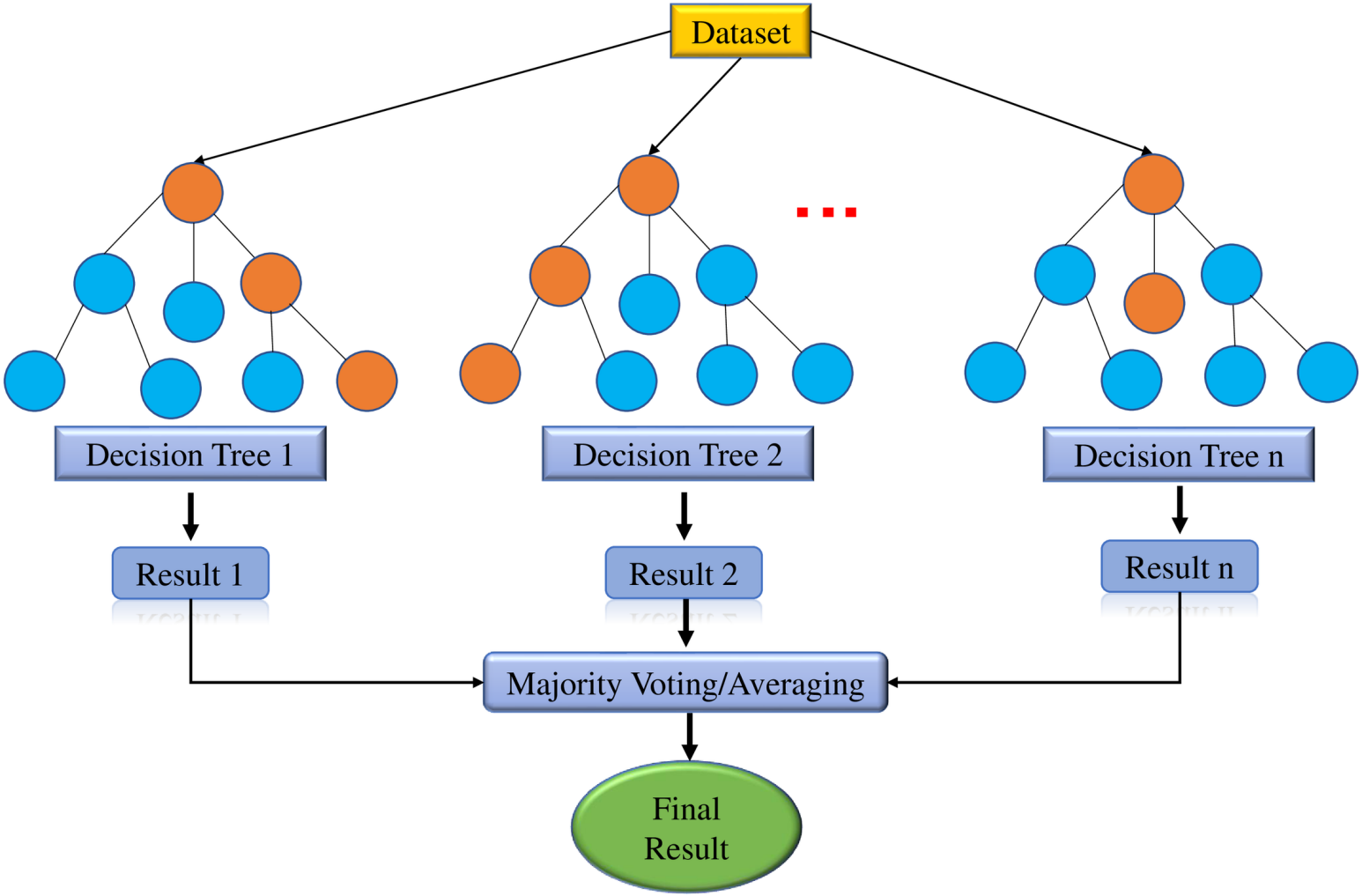
Random Forest – (Floresta aleatória)

Random Forest é uma extensão especial de *bagging* que introduz ainda mais aleatoriedade para promover maior diversidade no conjunto de modelos base (árvores de decisão) criados

Random Forest

1. Começa por fazer **amostragem bootstrap** para gerar subconjuntos de treino diferentes (tal como bagging)
2. Com cada subconjunto de treino é gerada uma **árvore de decisão** como estimadores de base
3. As previsões das árvores são combinadas por voto

Random Forest



Random Forest

- Random Forest combina:
  - seleção aleatória de instâncias (bagging)
  - seleção aleatória de atributos

- A diversidade é garantida selecionando aleatoriamente em cada divisão, um subconjunto dos atributos originais durante o processo de geração de árvores

- Algoritmo base de aprendizagem: Alg. CART (árvores de decisão)

- Floresta = um conjunto de árvores T

[Breiman, 2001]

Homogeneous Sequential Ensembles

Boosting

Boosting: Ideia Base

Diversificação

Reamostragem sequencial adaptativa por ponderação das instâncias

- Inicialmente: todas as instâncias têm pesos iguais: 1/|TRN|
- Em cada uma das iterações:
  - aplicar o algoritmo e estimar o **erro**
  - Aumentar/diminuir o peso dos casos mal/corretamente classificados

Foco na aprendizagem dos casos difíceis

Boosting


Original Data
Weighted Data
Weighted Data
Classifer
Classifer
Classifer
Ensemble Classifer

Boosting

- Procedimento iterativo: os novos modelos são influenciados pelo desempenho dos modelos construídos anteriormente
  - O novo modelo é uma **especialização em instâncias classificadas incorretamente** pelos modelos anteriores
  - Justificação intuitiva: os novos modelos devem complementar os modelos anteriores

Algoritmos de Boosting

Existem diversos algoritmos de boosting essencialmente diferem na forma:

- Como os pesos são atribuídos às instâncias mal classificadas no final de cada iteração de boosting
- Como as previsões de cada classificador são combinadas

Algoritmos Boosting:

- AdaBoost (Adaptative Boosting)
- Gradient Boosting
- **XGBoost**
- **LightBoost**

Boosting: Sumário

- O poder do Boosting deve-se à **reamostragem adaptativa**
- Tal como o Bagging, o Boosting reduz a variância
- Também reduz o *bias*, obrigando os modelos a concentrarem-se nos casos mais difíceis
  ->Hipótese combinada mais flexível
- Rápida Convergência

Bagging vs. Boosting

	Bagging	Boosting
Similaridades	Classificação: por esquema de voto Combinam modelos do mesmo tipo	
Diferenças	Modelos base desenvolvidos separadamente	Cada novo modelo é influenciado pelo desempenho dos modelos previamente desenvolvidos
	igual peso atribuído a todos os modelos	O peso do modelo é definido pelo seu desempenho

Estratégias para Combinação de Modelos

Sumário

1. Manipulação do conjunto de treino, através de diferentes formas de amostragem

2. Manipulação dos atributos da amostra inicial

3. Usando um conjunto diversificado de algoritmos, tais como, rede neural, árvore de decisão, …

4. Através de diferentes funções de combinação, que determinam como divergências entre as previsões são reconciliadas: por votação, por ponderação, …

Parametrização de Algoritmos

Otimização dos híper-parâmetros

O ajuste dos híper-parâmetros dos algoritmos de ML é importante para se conseguir obter melhor desempenho dos modelos

A otimização dos parâmetros envolve:

- Conhecer os parâmetros dos algoritmos
- De entre os parâmetros selecionar os mais importantes
- Testar as combinações possíveis de valores dos parâmetros

RandomForest parâmetros

```
clf = RandomForestClassifier()
clf.get_params().keys()

['bootstrap', 'ccp_alpha', 'class_weight', 'criterion', 'max_depth',
'max_features','max_leaf_nodes', 'max_samples','min_impurity_decrease',
'min_impurity_split','min_samples_leaf','min_samples_split',
'min_weight_fraction_leaf','n_estimators','n_jobs','oob_score',
'random_state','verbose','warm_start'])
```

- n_estimators = number of trees in the foreset
- max_depth = maximum depth of each tree in the forest
- max_features = max number of features considered for splitting a node
- min_samples_split = minimum number of samples required to split an internal node
- min_samples_leaf = min number of data points allowed in a leaf node

Estratégias para Otimização dos parâmetros

- Pesquisa manual
- Grid Search CV
- Random Search CV
- Bayesian Optimization

Grid Search CV

São executadas todas as combinações exaustivas de valores de parâmetros fornecidos e escolhida a melhor combinação

Exemplo, algoritmo randomForest principais parâmetros:

- n_estimators, max_depth, max_features

```
# Parameters to try
param_grid = { 'n_estimators': [100, 200, 300, 400, 500],
               'max_depth': [None, 10, 20, 30],
               'max_features': ['auto', 'log'] }
```

- Todas as combinações de valores dos híper parâmetros resulta em 5X4X2 = 60 x cv=5 **300** modelos diferentes
- Adicionar mais um híper parâmetro aumentará exponencialmente o número de combinações, aumentando drasticamente o tempo de execução
- Devem ser escolhidos apenas os parâmetros mais importantes a ajustar

Random Forest GridSearch optimization

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

rfm = RandomForestClassifier()

param_rfm = { 'n_estimators': [100, 200, 300, 400, 500],
              'max_depth': [None, 10, 20, 30],
              'max_features': ['auto', 'log'] }

rfm_gs = GridSearchCV(rfm, params_rfm, cv=5)

rfm_gs_fit = rfm_gs.fit(X_train, y_train)
rfm_best = rfm_gs.best_estimator_

rfm_preds = cross_val_score(rfm_best, X_test, y_test, cv=5)
```

Random Search CV

Escolhe aleatoriamente apenas algumas combinações de todas as combinações possíveis

```
# Parameters to try
param_grid = { 'n_estimators': [100, 200, 300, 400, 500],
               'max_depth': [None, 10, 20, 30],
               'max_features': ['auto', 'log'] }
```

- Das 60 combinações possíveis RandomSearchCV através do parâmetro ***n_iter*** permite especificar quantas combinações experimentar
- Se n_iter=20, significa que quaisquer 20 combinações aleatórias serão testadas

Random Forest RandomSearch optimization

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV

rfm = RandomForestClassifier()

param_rfm = { 'n_estimators': [100, 200, 300, 400, 500],
              'max_depth': [None, 10, 20, 30],
              'max_features': ['auto', 'log'] }

rfm_gs = GridSearchCV(rfm, params_rfm, cv=5)

# Create a RandomizedSearchCV object
rfm_rs=RandomizedSearchCV(rfm,
                          params_rfm,
                          n_iter=100, cv=5, n_jobs=-1)

rfm_rs_fit=rfm_rs.fit(X_train, y_train)
rfm_best = rfm_rs.best_estimator_

rfm_preds = cross_val_score(rfm_best, X_test, y_test, cv=5)
```

Bayesian Optimization

Faz uma escolha inteligente sobre a próxima combinação a ser tentada, por análise dos resultados das combinações anteriores

```
# Parameters to try
param_grid = { 'n_estimators': [100, 200, 300, 400, 500],
               'max_depth': [None, 10, 20, 30],
               'max_features': ['auto', 'log'] }
```

- Supondo que o teste relativo ao parâmetro n_estimators = 100, 200 e 300 produziu o melhor resultado com n_estimators=200
- O próximo conjunto de valores a ser testado será em torno de 200
- O mesmo com os outros parâmetros
- Alguns dos valores de hiper-parâmetros fornecidos podem não ser testados

Random Forest Bayesian optimization

```
from sklearn.ensemble import RandomForestClassifier
from skopt import BayesSearchCV

rfm = RandomForestClassifier()

param_rfm = { 'n_estimators': [100, 200, 300, 400, 500],
              'max_depth': [None, 10, 20, 30],
              'max_features': ['auto', 'log'] }

# Create a BayesSearchCV object
rfm_bs = BayesSearchCV (rfm, params_rfm,
                        n_iter=100,
                        cv=5,
                        n_jobs=-1)

rfm_bs_fit=rfm_bs.fit(X_train, y_train)
rfm_best = rfm_bs.best_estimator_

rfm_preds = cross_val_score(rfm_best, X_test, y_test, cv=5)
```