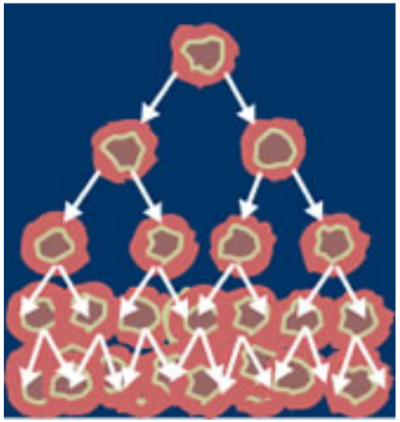
Pipeline de aprendizagem

Breast Cancer Wisconsin (Diagnostic) Data Set

Download: Data Folder, Data Set Description

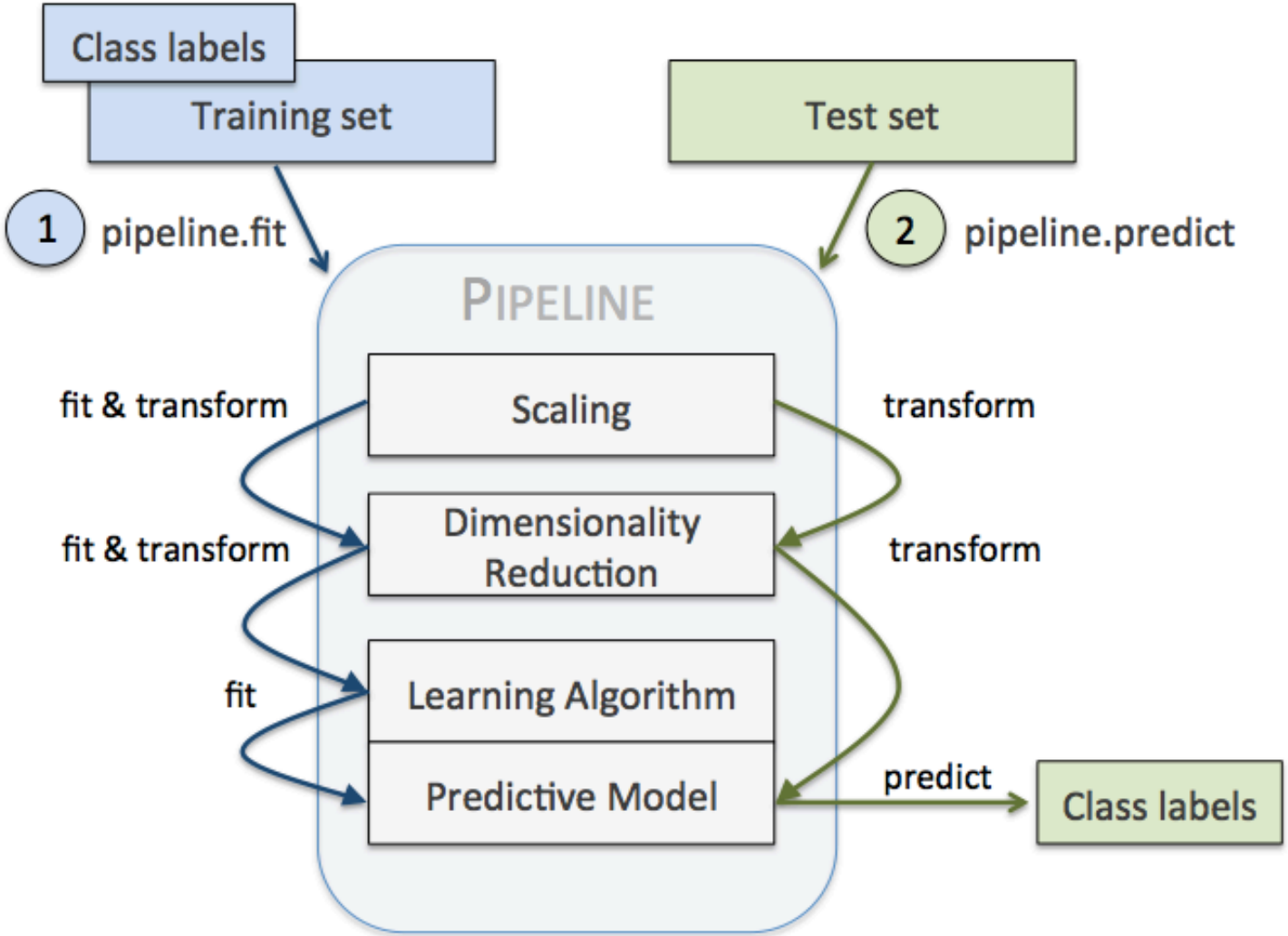


Abstract: Diagnostic Wisconsin Breast Cancer Database

Data Set Characteristics:	Multivariate	**Number of Instances:**	569	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	32	**Date Donated**	1995-11-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	546676

842302,**M**,17.99,10.38,122.8,1001,0.1184,0.2776,0.3001,0.1471,0.2419, ...
89842517,**M**,20.57,17.77,132.9,1326,0.08474,0.07864,0.0869,0.07017,0.1812, ...
84300903,**M**,19.69,21.25,130,1203,0.1096,0.1599,0.1974, ...
...
8510426,**B**,13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,0.1885,...
8510653,**B**,13.08,15.71,85.63,520,0.1075,0.127,...
...

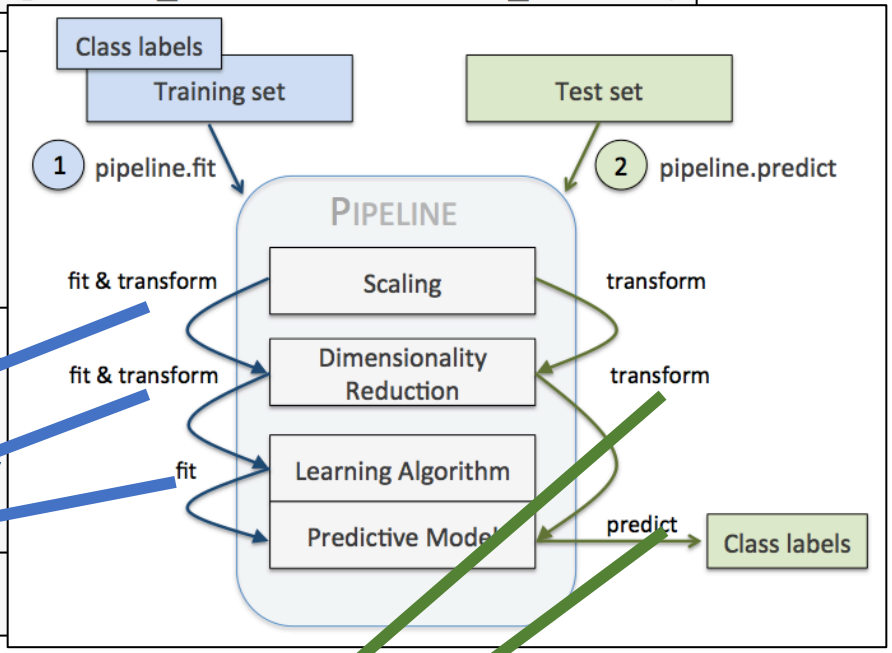
a) Radius b) texture c) perimeter d) area e) smoothness f) compactness g) concavity h) concave points i) symmetry j) fractal dim


Class labels
Training set
Test set
1
pipeline.fit
2
pipeline.predict
PIPELINE
fit & transform
Scaling
transform
fit & transform
Dimensionality Reduction
transform
fit
Learning Algorithm
Predictive Model
predict
Class labels

```
3 X_train, X_test, y_train, y_test = \
4         train_test_split(X, y, test_size=0.20, random_state=1)
```

```
1 from sklearn.preprocessing
2 X = df.loc[:, 2:].values
3 y = df.loc[:, 1].values
4 le = LabelEncoder()
5 y = le.fit_transform(y)
6 le.transform(['M', 'B'])
```

```
4 from sklearn.pipeline import Pipeline
5
6 pipe_lr = Pipeline([('scl', StandardScaler()),
7             ('pca', PCA(n_components=2)),
8             ('clf', LogisticRegression(
9
10 pipe_lr.fit(X_train, y_train)
```




```
11 print('Test Accuracy: %.3f' % pipe_lr.score(X_test, y_test))
12 y_pred = pipe_lr.predict(X_test)
```

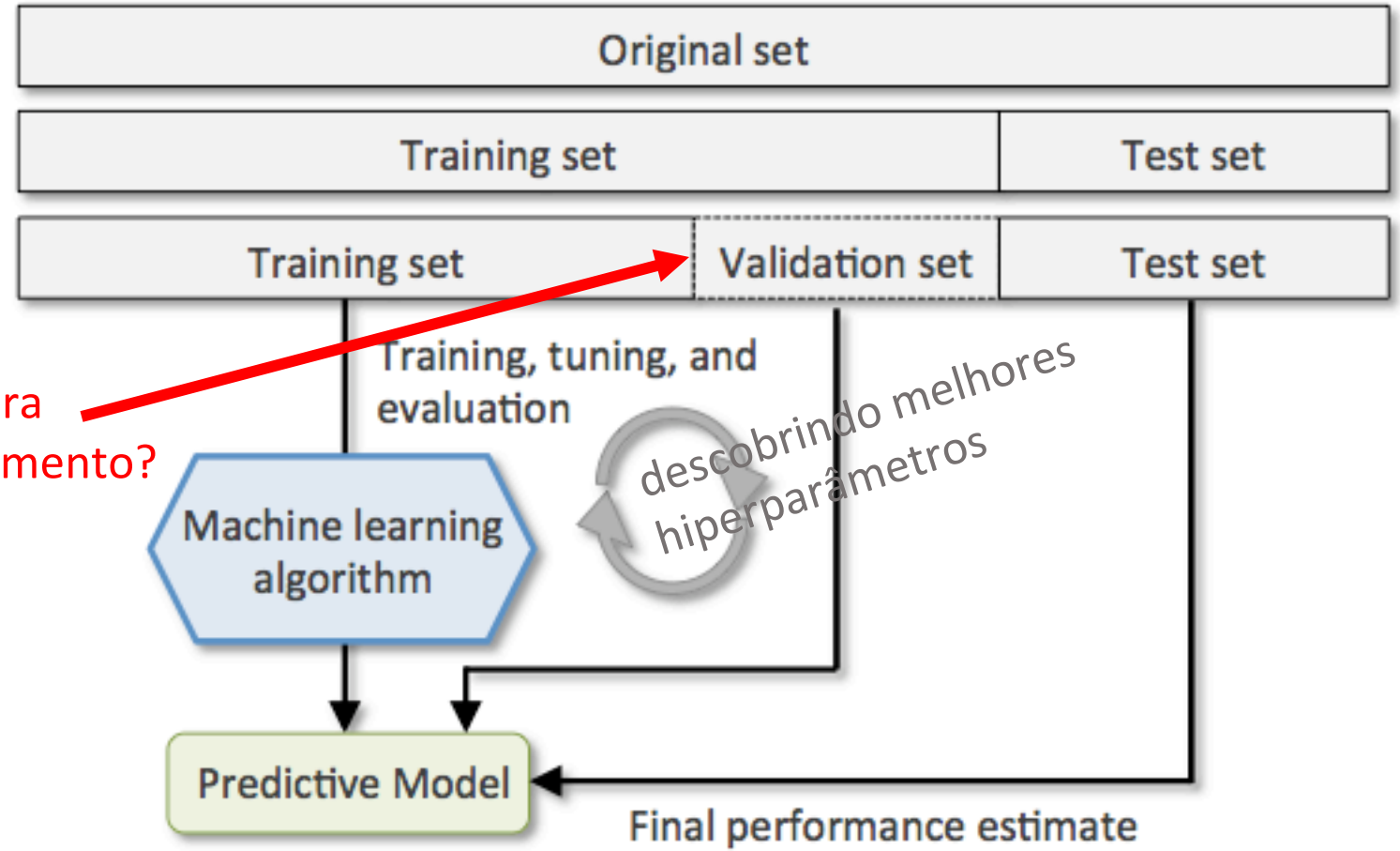
Estimativa de performance do modelo

Objetivo: permitir variação de parâmetros de treinamento para otimizar modelo (descobrir melhores hiperparâmetros)

Validação cruzada

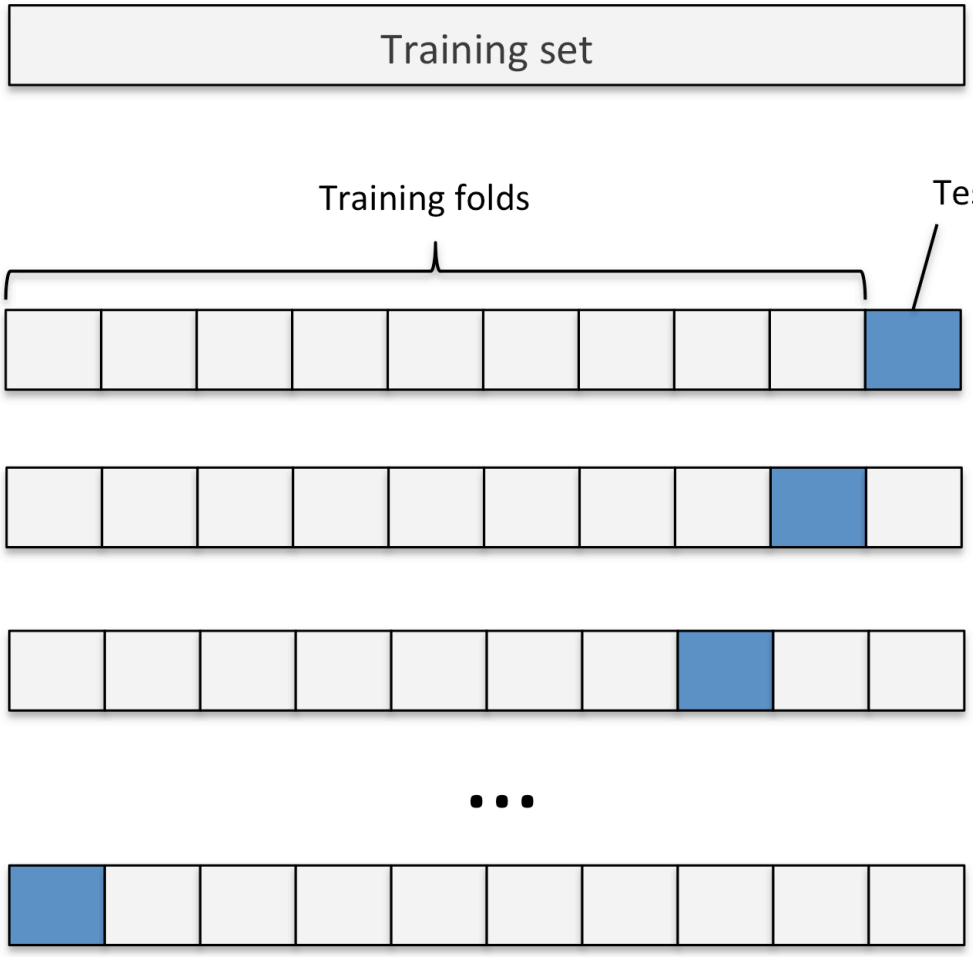
- Método *holdout*
- Método *k-fold*

Holdout



Que critério para este particionamento?

k-fold

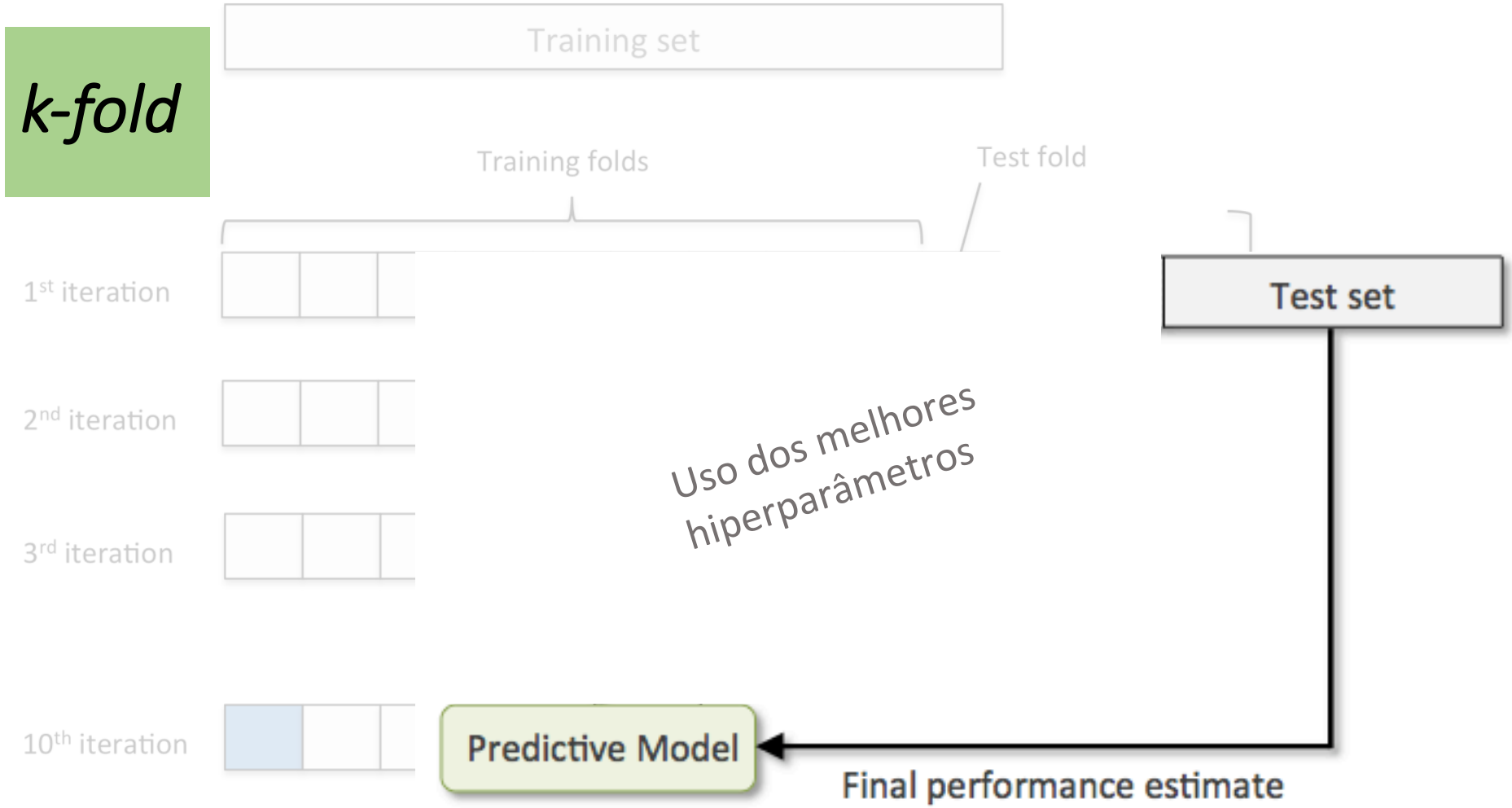


particionamento aleatório sem reposição

$$E = \frac{1}{10}\sum_{i=1}^{10} E_i$$

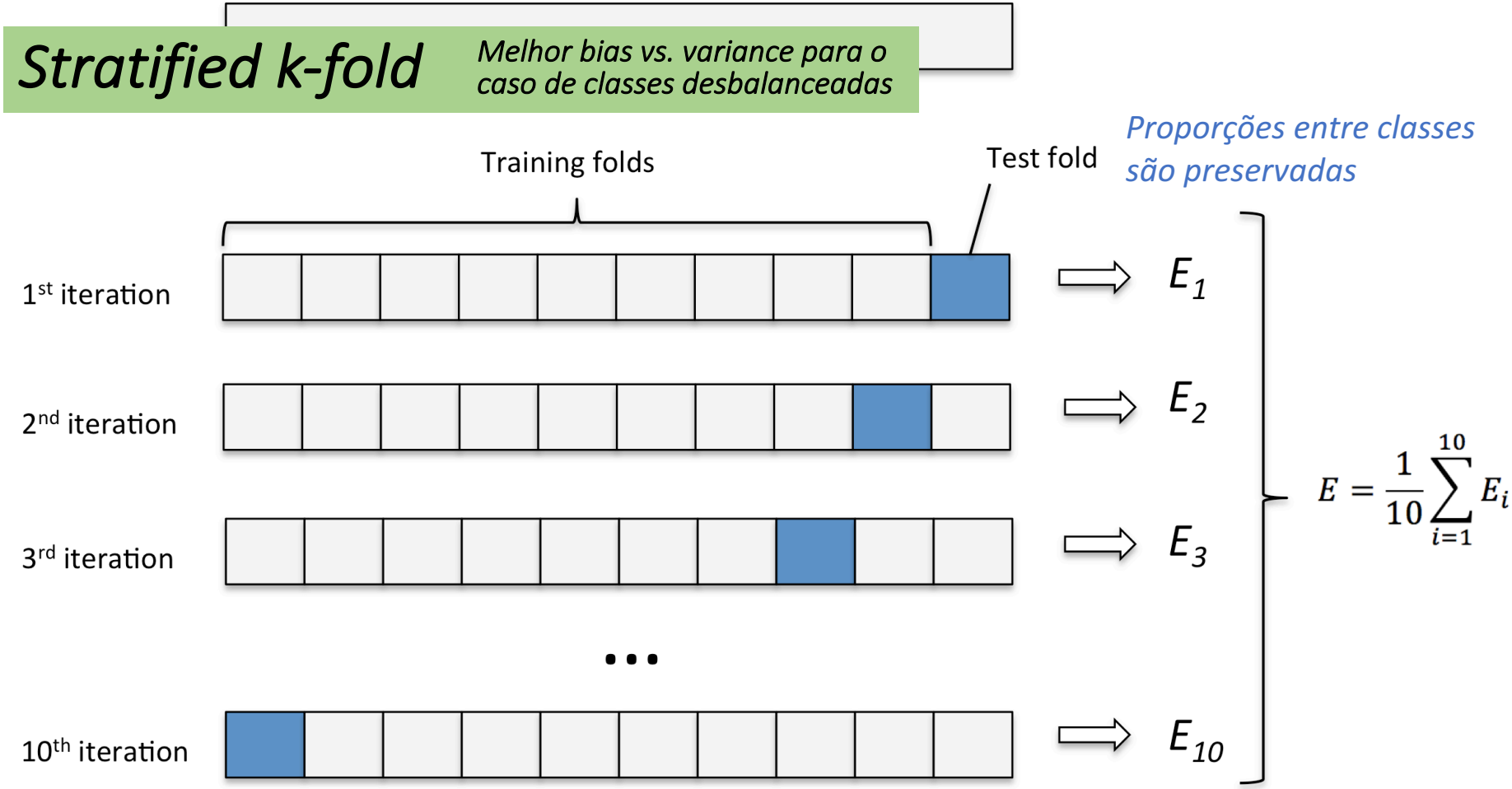
menor sensibilidade ao particionamento

k-fold


Training set
Training folds
Test fold
1st iteration
2nd iteration
3rd iteration
10th iteration
Test set
Uso dos melhores hiperparâmetros
Predictive Model
Final performance estimate

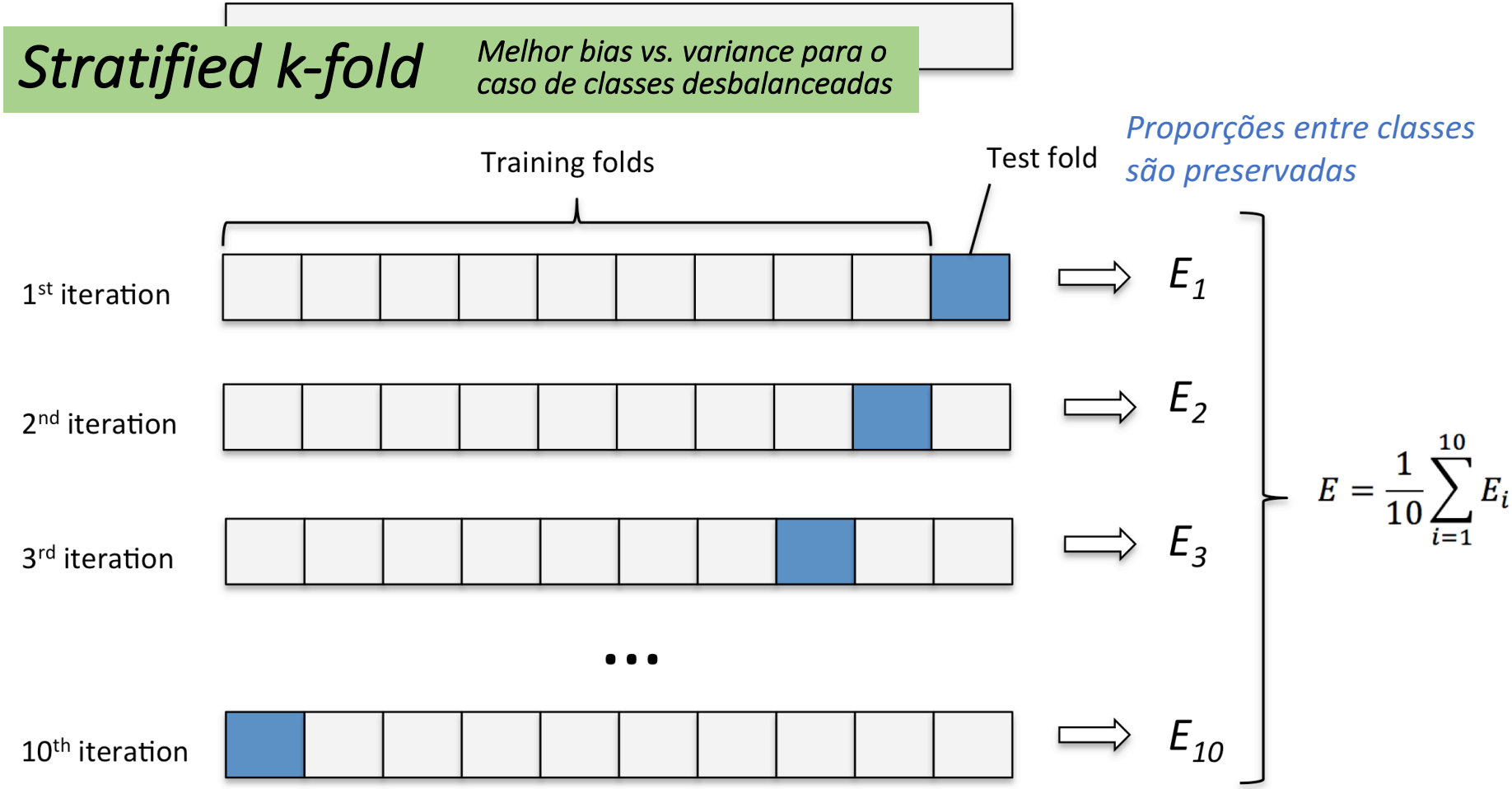
Stratified k-fold

Melhor bias vs. variance para o caso de classes desbalanceadas



Stratified k-fold

Melhor bias vs. variance para o caso de classes desbalanceadas



k-fold


```
1 from sklearn.cross_validation import cross_val_score
2
3 scores = cross_val_score(estimator=pipe_lr,
4                          X=X_train,
5                          y=y_train,
6                          cv=10,
7                          n_jobs=1)
8 print('CV accuracy scores: %s' % scores)
9 print('CV accuracy: %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
```

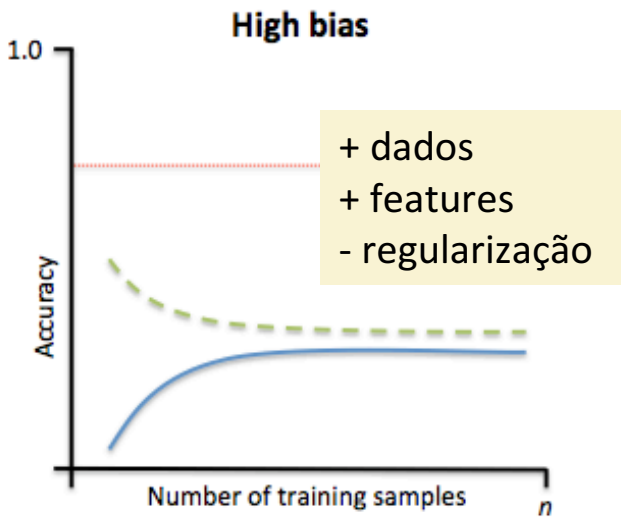
1 CPU
2 CPUs, ...
-1 → todas CPUs

```
CV accuracy scores: [ 0.89130435  0.97826087  0.97826087  0.91304348  0.93478261  0.97777778
  0.93333333  0.95555556  0.97777778  0.95555556]
CV accuracy: 0.950 +/- 0.029
```

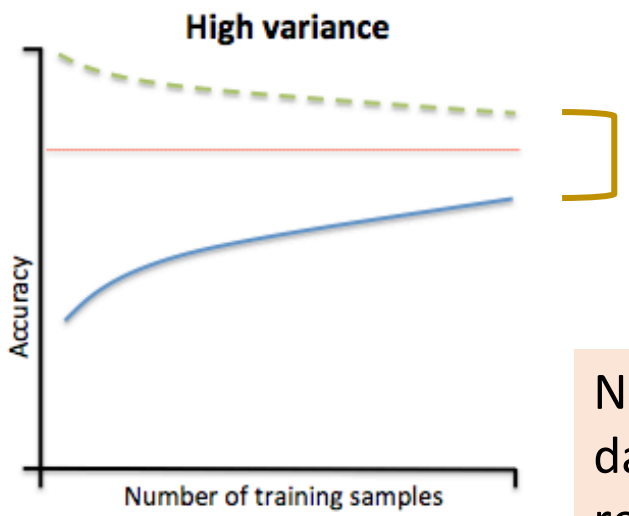
Ferramentas de diagnóstico

Objetivo: observar problemas com *underfitting* (alto bias) ou *overfitting* (alta variância)

Curvas de aprendizado
Curvas de validação
Grid search

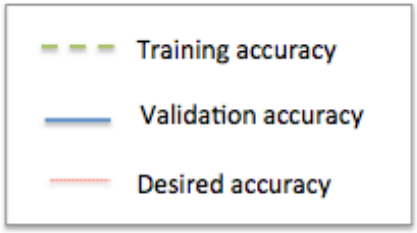


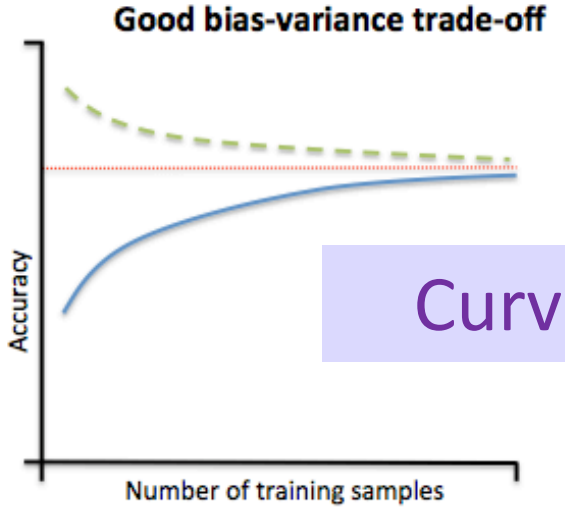
+ dados
+ features
- regularização



+ dados
+ regularização
- features (via seleção)
- feaures (via extração)

Nem sempre coletar mais dados para treinamento resolve o problema

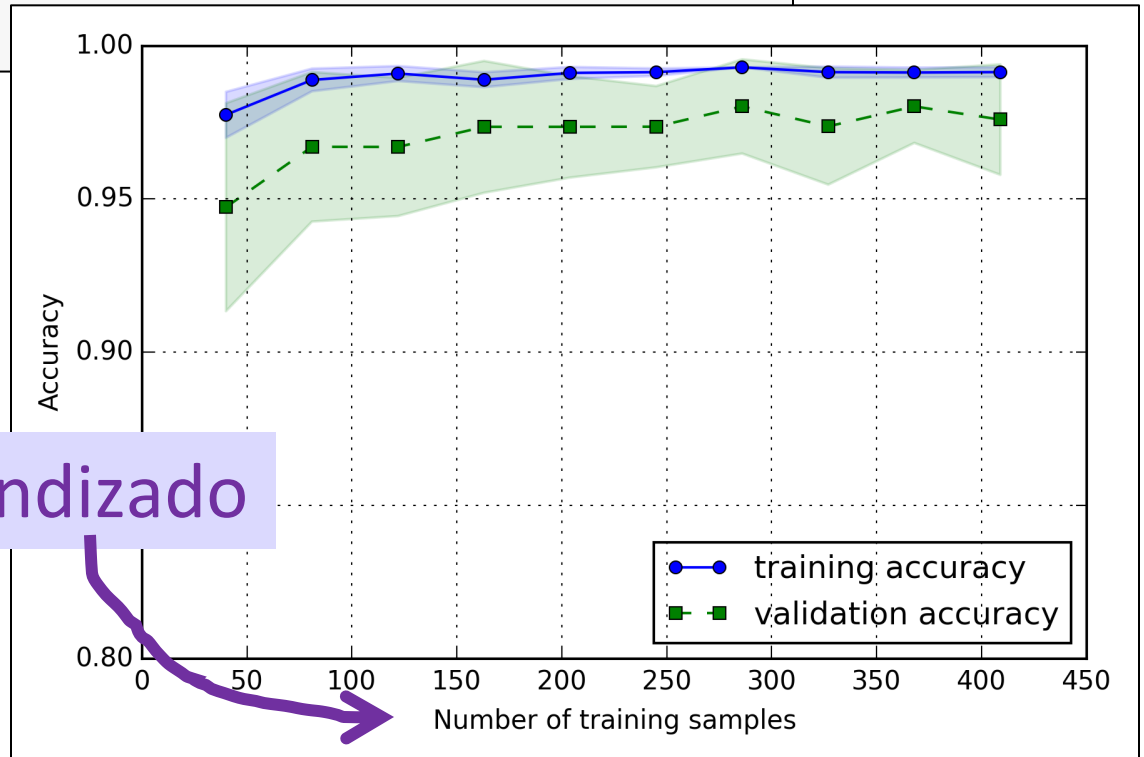




Curvas de aprendizado

```
pipe_lr = Pipeline([('scl', StandardScaler()),
            ('clf', LogisticRegression(penalty='l2', random_state=0))])

train_sizes, train_scores, test_scores =\
                learning_curve(estimator=pipe_lr,
                X=X_train,
                y=y_train,
                train_sizes=np.linspace(0.1, 1.0, 10),
                cv=10,
                n_jobs=1)
```

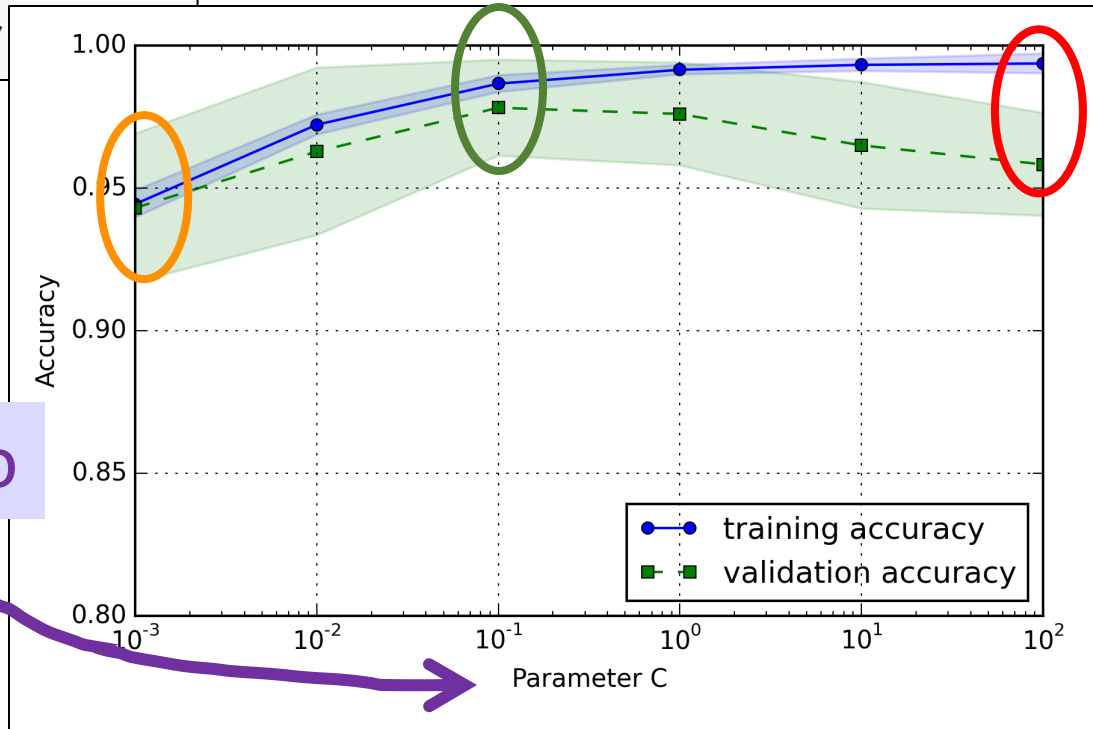


Curvas de aprendizado

```
5  pipe_lr = Pipeline([('scl', StandardScaler()),
6              ('clf', LogisticRegression(penalty='l2', random_state=0))])
```

```
3  param_range = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]
4  train_scores, test_scores = validation_curve(
5                  estimator=pipe_lr,
6                  X=X_train,
7                  y=y_train,
8                  param_name='clf__C',
9                  param_range=param_range,
10                 cv=10)
```

Curvas de validação



Grid search

Combinação exaustiva sobre lista de valores de hiperparâmetros

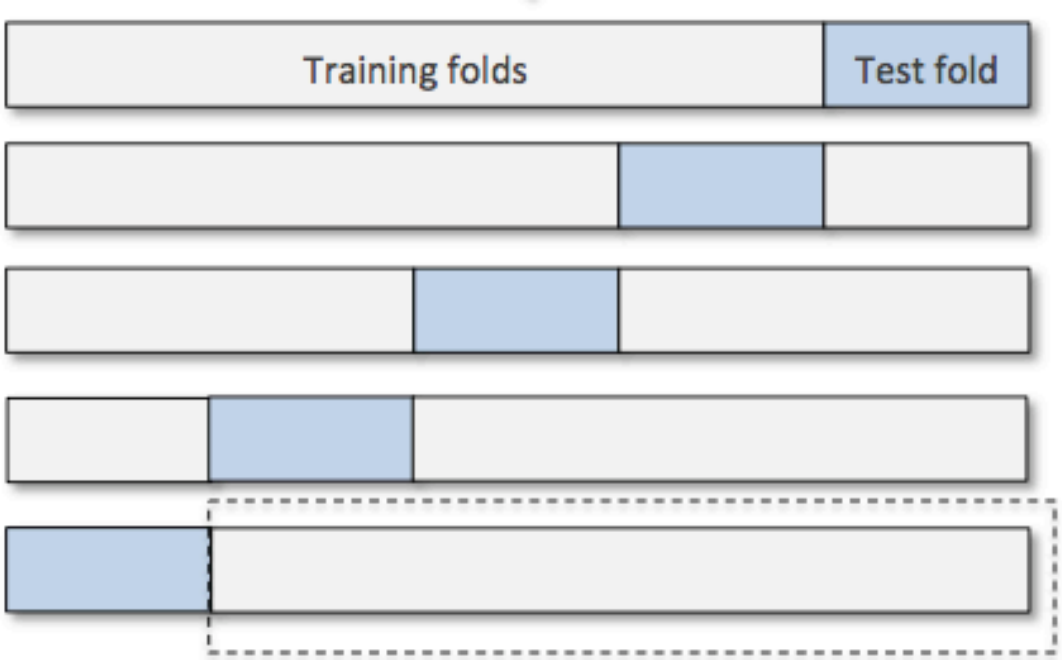
```
param_range = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]

param_grid = [{'clf__C': param_range,
               'clf__kernel': ['linear']},
              {'clf__C': param_range,
               'clf__gamma': param_range,
               'clf__kernel': ['rbf']}]

gs = GridSearchCV(estimator=pipe_svc,
                  param_grid=param_grid,
                  scoring='accuracy',
                  cv=10,
                  n_jobs=-1)
```

Comparar parâmetros

```
0.978021978022
{'clf__kernel': 'linear', 'clf__C': 0.1}
```

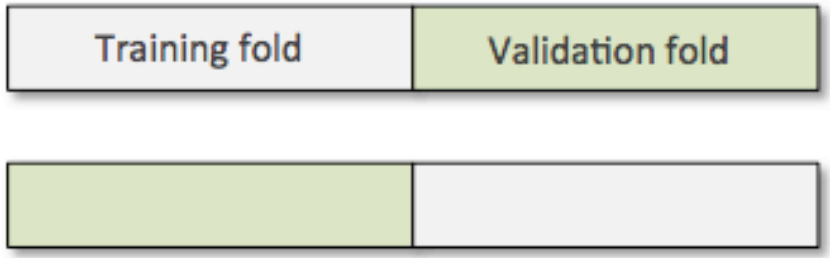


Grid search + Nested k-fold

Outer loop

Train with optimal parameters

Ex: 5x2 cross validation



Inner loop

Tune parameters

```
param_range = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]

param_grid = [{'clf__C': param_range,
               'clf__kernel': ['linear']},
              {'clf__C': param_range,
               'clf__gamma': param_range,
               'clf__kernel': ['rbf']}]

gs = GridSearchCV(estimator=pipe_svc,
                  param_grid=param_grid,
                  scoring='accuracy',
                  cv=10,
                  n_jobs=-1)
```

Grid search + Nested k-fold

```
CV accuracy: 0.965 +/- 0.025
```

vs.

```
gs = GridSearchCV(estimator=DecisionTreeClassifier(random_state=0),
                  param_grid=[{'max_depth': [1, 2, 3, 4, 5, 6, 7, None]}],
                  scoring='accuracy',
                  cv=2)
```

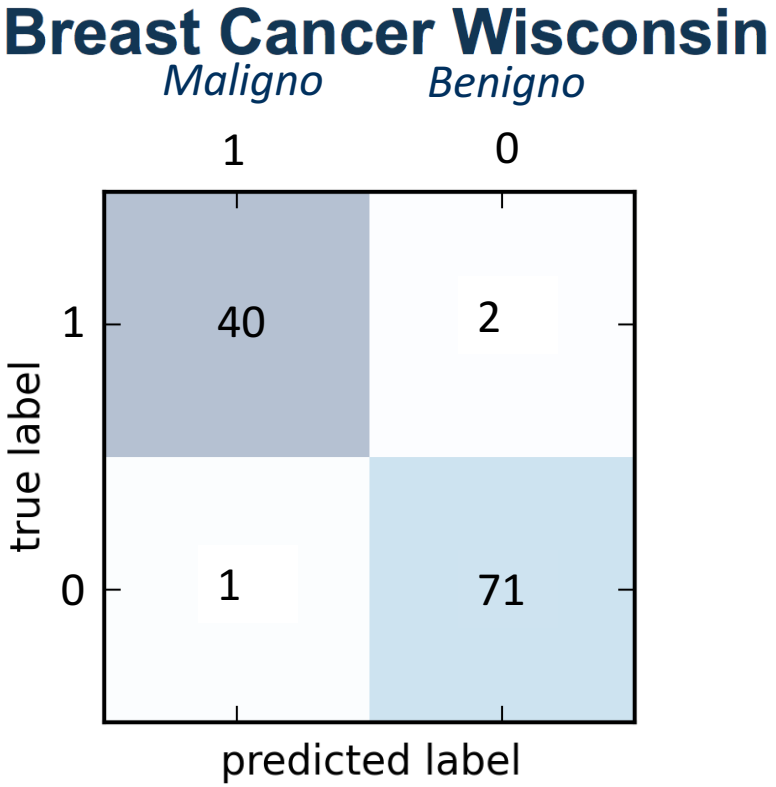
```
CV accuracy: 0.921 +/- 0.029
```

Tune parameters

Métricas de performance

Matriz de confusão

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)



Acurácia (*Accuracy*)

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR$$

Sensibilidade e Especificidade

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Breast Cancer Wisconsin

importante não preocupar o paciente

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

importante detectar para tratar

Precisão *(Precision)* e Cobertura *(Recall)*

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Breast Cancer Wisconsin

Dos detectados como malignos, quantos realmente o são?

$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

De todos os tumores malignos existentes, quantos foram de fato detectados?

Medida F *(F-Measure)*

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$\mathrm{F1} = 2 \times \frac{PRE \times REC}{PRE + REC}$$

$$\mathrm{F1} = \frac{(w+1) \times PRE \times REC}{PRE + w \times REC}$$

Exemplo

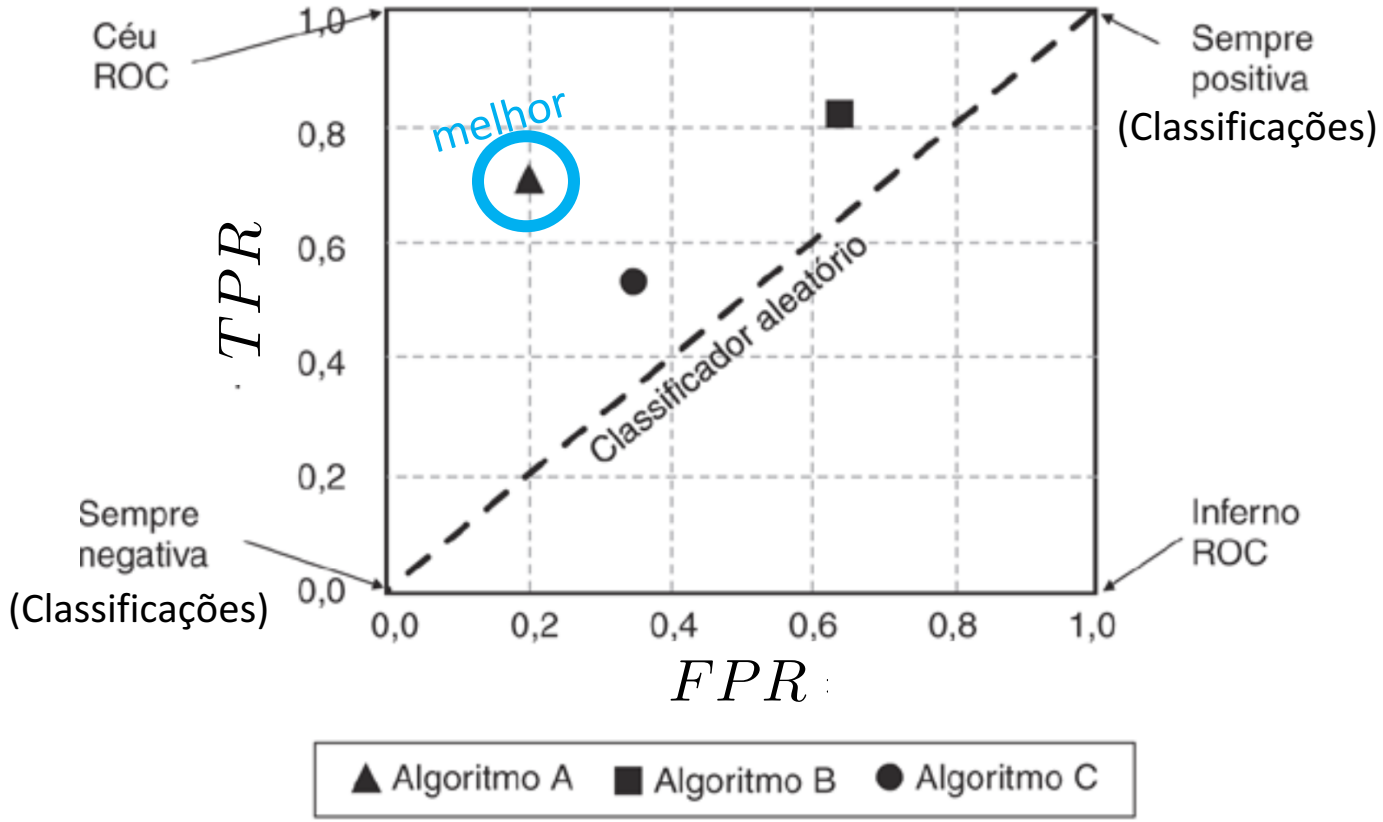
		Previsto como SPAM	
		+	-
Mensagem de SPAM	+	8	2
	-	16	974

- Acurácia = (8+974)/1000 = 98,2%
- Sensibilidade = 8/(8+2) = 80,0%
- Precisão = 8/(8+16) = 33,3%
- Medida-F ($w=1$) = 2*0,80*0,33/(0,80+0,33) = 0,467

Análise ROC (*Receiver Operator Characteristic*)

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$



Curvas ROC (*Receiver Operator Characteristic*)

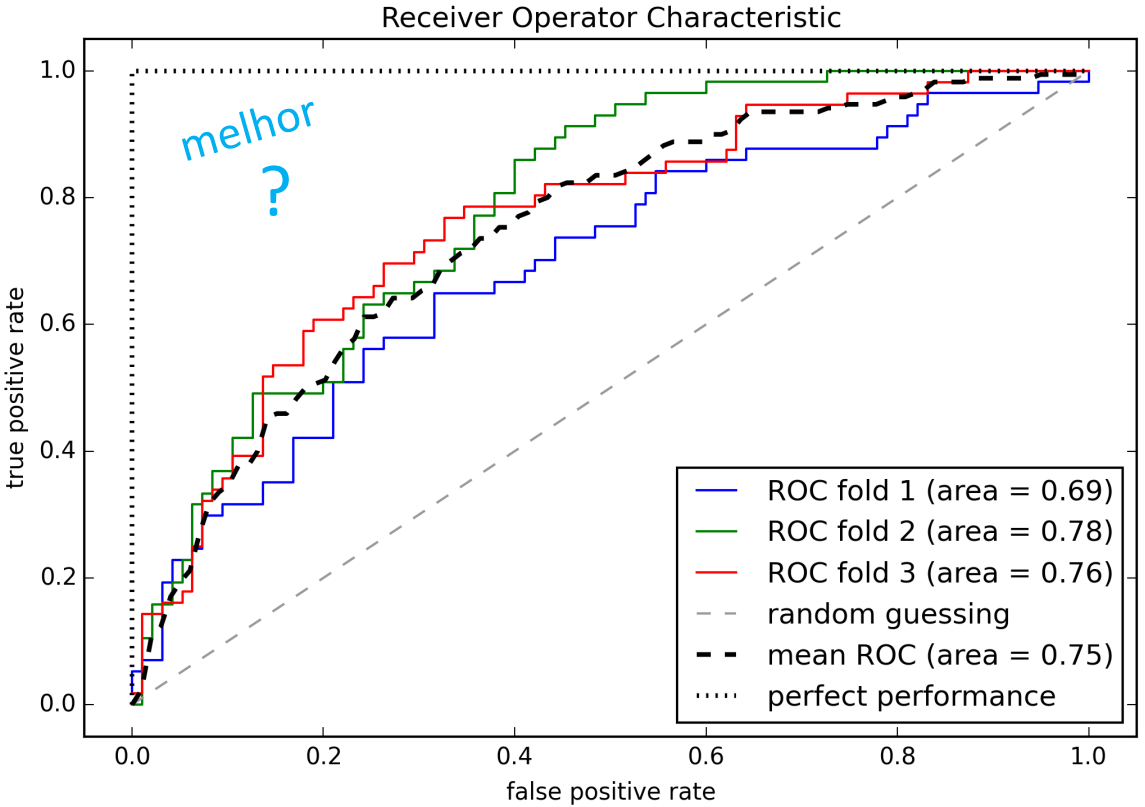
Varia-se o limiar de decisão do classificador

Naive bayes:
- limiar típico = 0.5
- {0.1, 0.2, ..., 0.9}

SVM, Regressão Logística:
- Função sinal

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$



Curvas ROC (*Receiver Operator Characteristic*)

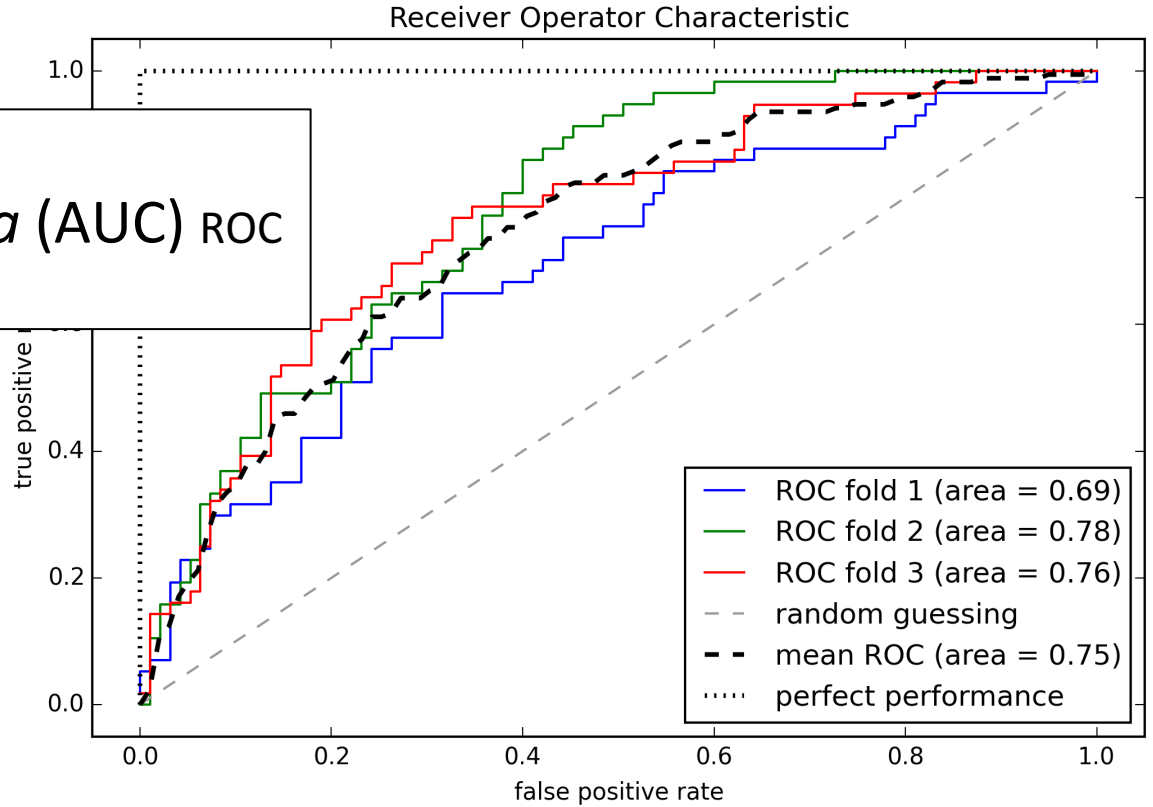
Melhor →

Maior *Área sob a Curva* (AUC) ROC

: produz valores em [0, 1]

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$



Divulgação científica

saense

http://www.saense.com.br
https://www.facebook.com/saense/

Hendrik Macedo
Escreve sobre Inteligência Artificial no Saense.
http://www.saense.com.br/autores/artigos-publicados-por-hendrik-macedo/