{desafío}
latam_

Mecanismos de Votación _

Itinerario

Activación de conceptos

Desarrollo Desafío

Panel de discusión

{desafío}
latam_

Activación de conceptos

¿Cuál de las siguientes frases es correcta?

- Un ensamble secuencial entre múltiples modelos y posteriormente promedia
- Un ensamble paralelo refuerza el error de clasificación de los modelos
- Un ensamble secuencial pondera en función a la pérdida y ajuste la exactitud

¿Qué algoritmos busca actualizar el modelo en base a las predicciones erróneas?

- Adaptative Boosting
- Gradient Boosting
- Multivariate Adaptative Regression Splines

Dentro de `sklearn.ensemble.GradientBoostingClassifier`, ¿Cómo se define la pérdida?

- Se debe declarar explícitamente por el usuario
- No se declara
- Se define en función al vector objetivo y la norma de pérdida definida por el usuario

Tipos de Ensambles

- **Modelos de instancia única**: Se entrena un modelo.

- **Modelos de ensambles paralelos**: Se entrena un modelo múltiples veces y se agregan resultados.

- **Modelos de ensambles secuenciales**: Se entrena un modelo múltiples veces y se corrige la tasa de error/gradiente

Novedad de los mecanismos de votación

Permiten agregar distintos tipos de modelos

¿Cuáles son las razones para implementar mecanismos de votación?

1. Poder representacional
2. Poder estadístico
3. Poder computacional

Comités

Rudimentos

1. Naturaleza del vector objetivo
2. Si es discreto, ver el tipo de votación

Setup

- Tenemos un modelo t $\in \mathcal{T}$.
- Cada t entrega una función candidata $h_t(\mathbf{x})$.
- Buscamos resolver **H(x)**

Vectores objetivos continuos

La función candidata entrega un resultado en $h_t(\mathbf{x}) \in \mathbb{R}$

Promedio simple

$$\mathcal{H}(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} h_t(\mathbf{x})$$

Promedio ponderado

$$\mathcal{H}(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} w_t h_t(\mathbf{x})$$

Vectores objetivos discretos

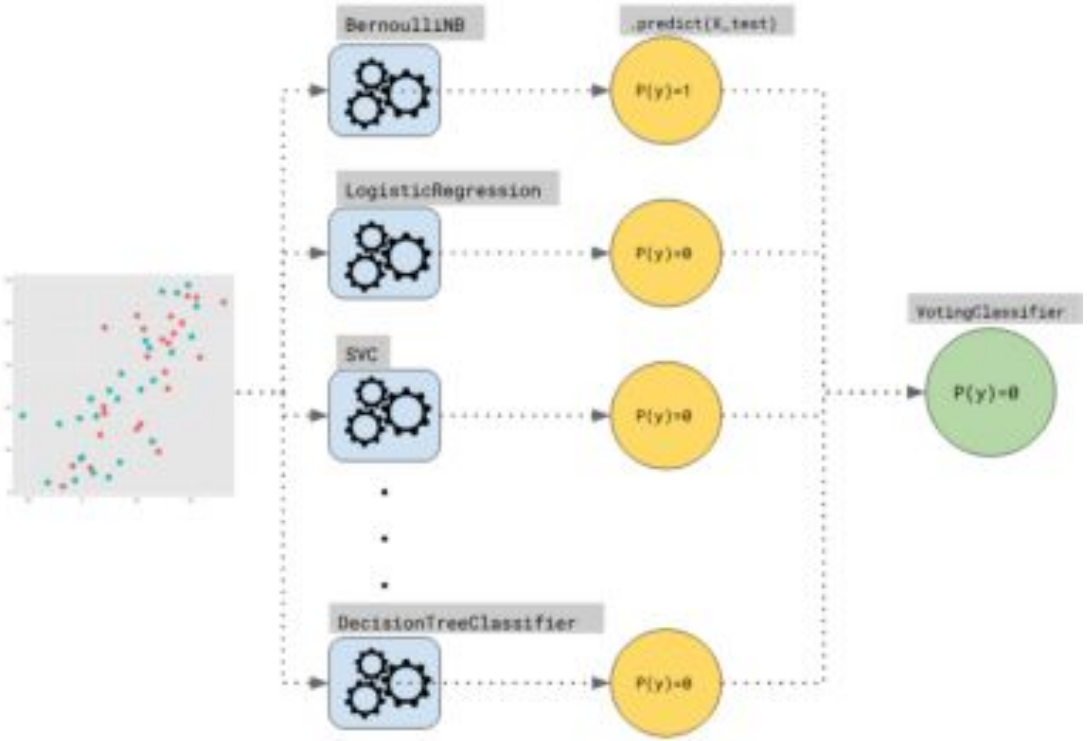
- La función candidata puede entregar dos resultados:
  - Clase: $h_i^c(\mathbf{x}) \in \{0, 1\}$
  - Etiqueta: $h_i^c(\mathbf{x}) \in [0, 1] \rightarrow \Pr(c|\mathbf{x})$
- Dependiendo de lo que se entregue en la función, podemos implementar estrategias de voto duro o blando.

Estrategias de voto duro

Votación por mayoría simple

Criterio:

Dado en ensamble **T** de clasificadores binarios, éste será correcto si por lo menos **T/2 + 1** eligen la misma clase.



Votación por mayoría relativa

Criterio:

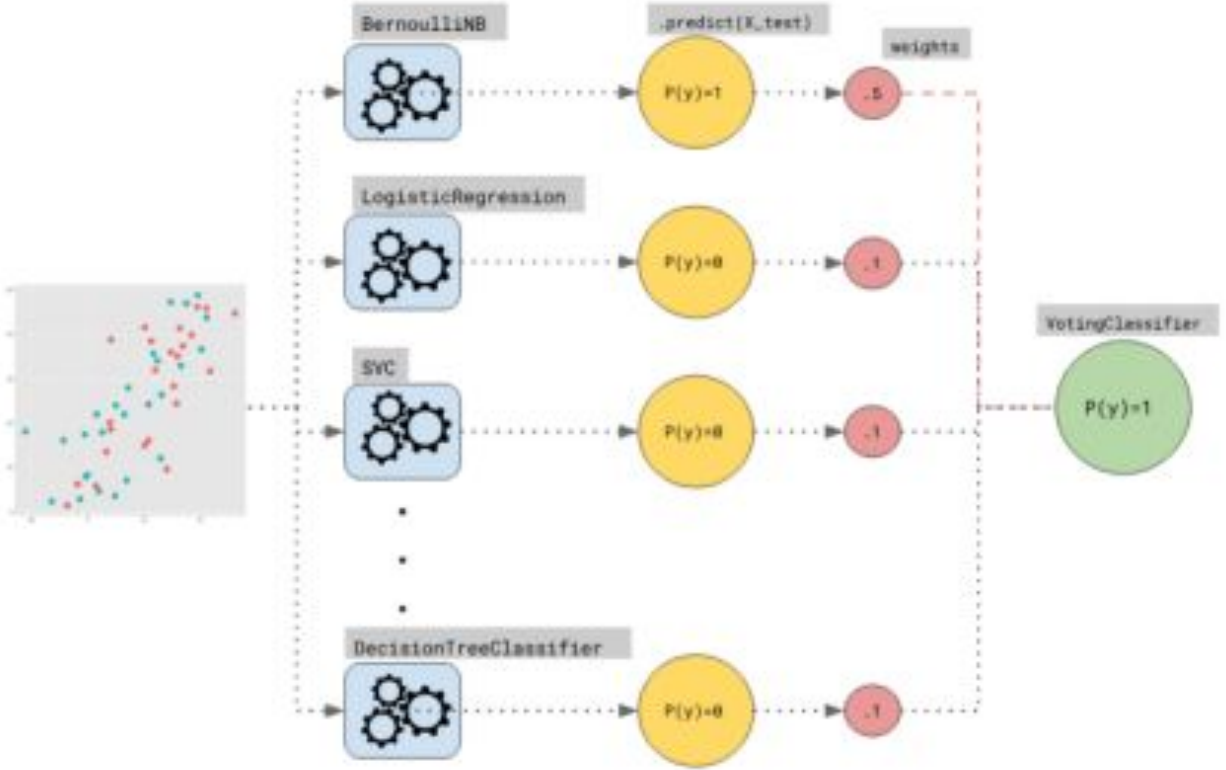
$$H(\mathbf{x}) = \underset{c}{\text{argmax}} \sum_{t=1}^{T} h_t^c(\mathbf{x})$$

Ponderación

- Por defecto asumimos equiprobabilidad en cada función candidata.
- Podemos afectar la relevancia de cada función candidata en el ensamble con los ponderadores.

Criterio: $\sum_{i=1}^{N} w_i = 1$

La función de agregación se actualiza a: $H(\mathbf{x}) = \underset{c}{\text{argmax}} \sum_{i=1}^{T} w_i h_i^c(\mathbf{x})$


BernoulliNB
.predict(X_test)
weights
P(y)=1
.5
LogisticRegression
P(y)=0
.1
VotingClassifier
SVC
P(y)=0
.1
P(y)=1
DecisionTreeClassifier
P(y)=0
.1

Estrategias de voto suave

Si el clasificador produce una probabilidad x donde $h_i^c(\mathbf{x}) \in [0, 1]$ se puede entender como $\Pr(c|\mathbf{x})$

- Ponderador específico a nivel de clasificador $$\mathcal{H}^c(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i^c(\boldsymbol{x})$$
- Ponderador específico a nivel de clasificador y clase $$\mathcal{H}^c(\boldsymbol{x}) = \sum_{i=1}^{T} w_i^c h_i^c(\boldsymbol{x})$$
- Ponderador específico a nivel de observación, clasificador y clase $$\mathcal{H}^c(\boldsymbol{x}) = \sum_{i=1}^{T} \sum_{j=1}^{m} w_{ij}^c h_i^c(\boldsymbol{x})$$

Implementación

Elementos necesarios

- Preprocesadores clásicos (train_test_split)
- Alguna métrica (f1, precision, etc...)
- `sklearn, emsemble, VotingClassifier`
- Una secuencia de modelos a componer el comité

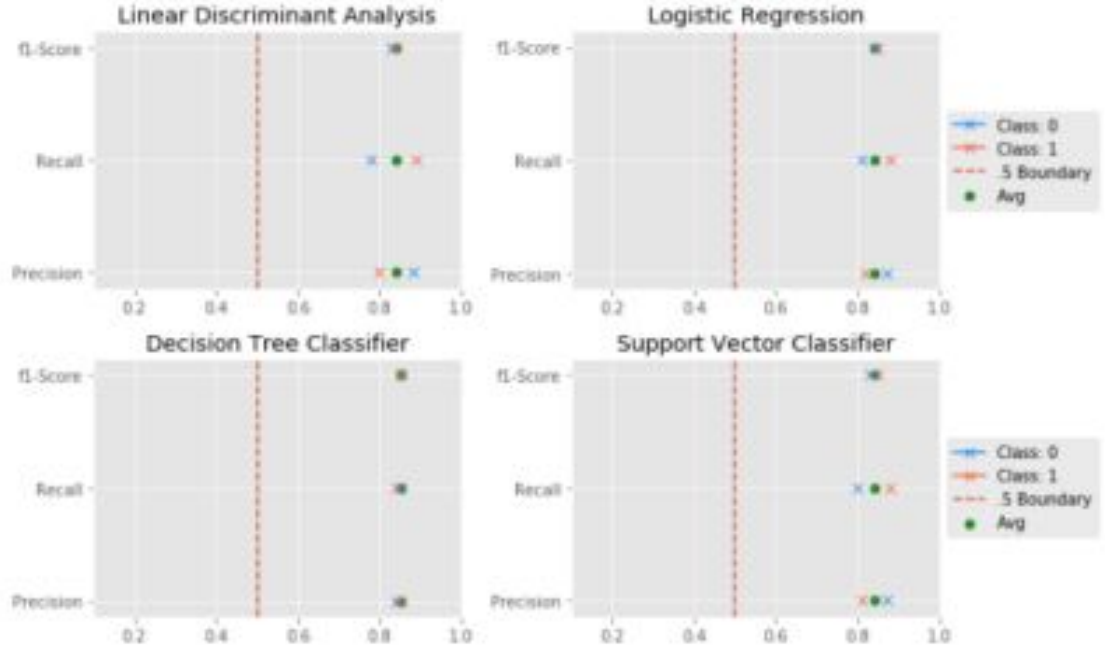
Definición del comité

In [2]:

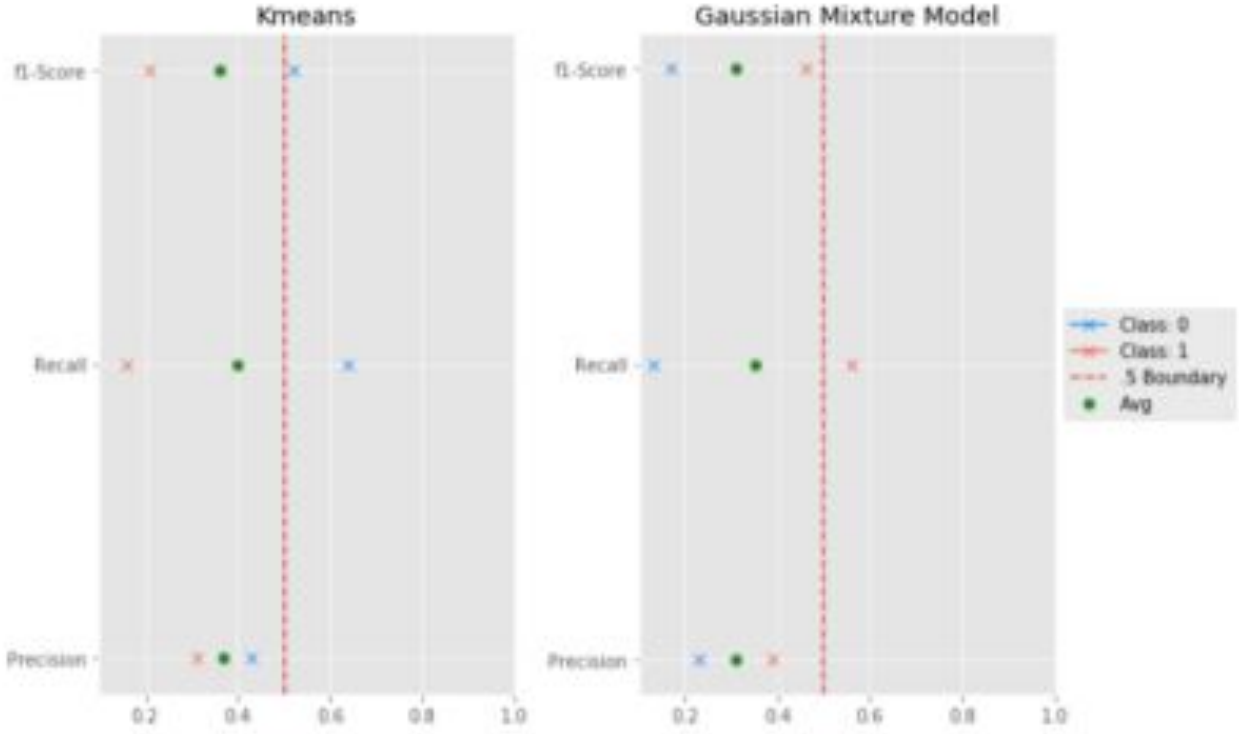
```
# definimos el comité de clasificadores en una lista de tuplas
estimators = [('Linear Discriminant Analysis', LinearDiscriminantAnalysis()),
('Logistic Regression', LogisticRegression(random_state=rep_seed)),
('Decision Tree Classifier', DecisionTreeClassifier(random_state=rep_seed)),
('Support Vector Classifier', SVC(kernel='linear', random_state=rep_seed)),
# para el caso de kmeans y GMM es necesario definir la cantidad de clusters a inferir
('Kmeans', KMeans(n_clusters=2, random_state=rep_seed)),
('Gaussian Mixture Model', GaussianMixture(n_components=2,random_state= rep_seed))]
```

Evaluación individual del comité

```
In [11]: fair_performance_clfs(estimators)
```



In [12]:
```
awful_performance_clfs(estimators)
```

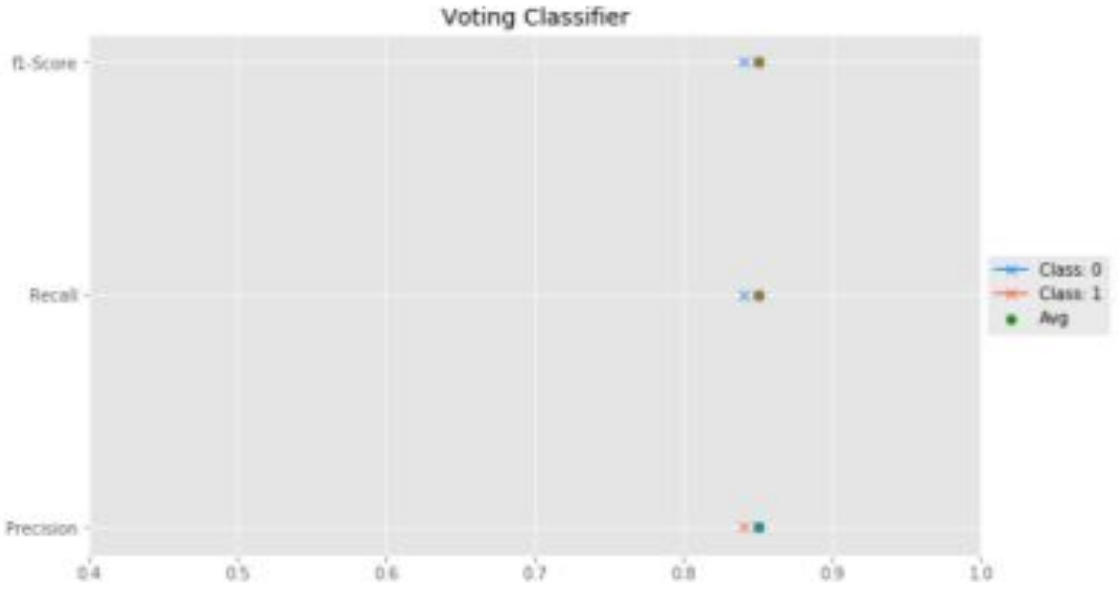


Implementación

```
In [3]: voting_classifier = VotingClassifier(estimators).fit(X_train, y_train)

In [4]: print([i[0] for i in voting_classifier.estimators])

['Linear Discriminant Analysis', 'Logistic Regression', 'Decision Tree
Classifier', 'Support Vector Classifier', 'Kmeans', 'Gaussian Mixture
Model']

In [5]: print(voting_classifier.voting)

hard

In [6]: print(voting_classifier.weights)

None
```

Desempeño del ensamble

```
In [13]: plot_voting_clf()

Out[13]: <matplotlib.legend.Legend at 0x1a26dda828>
```



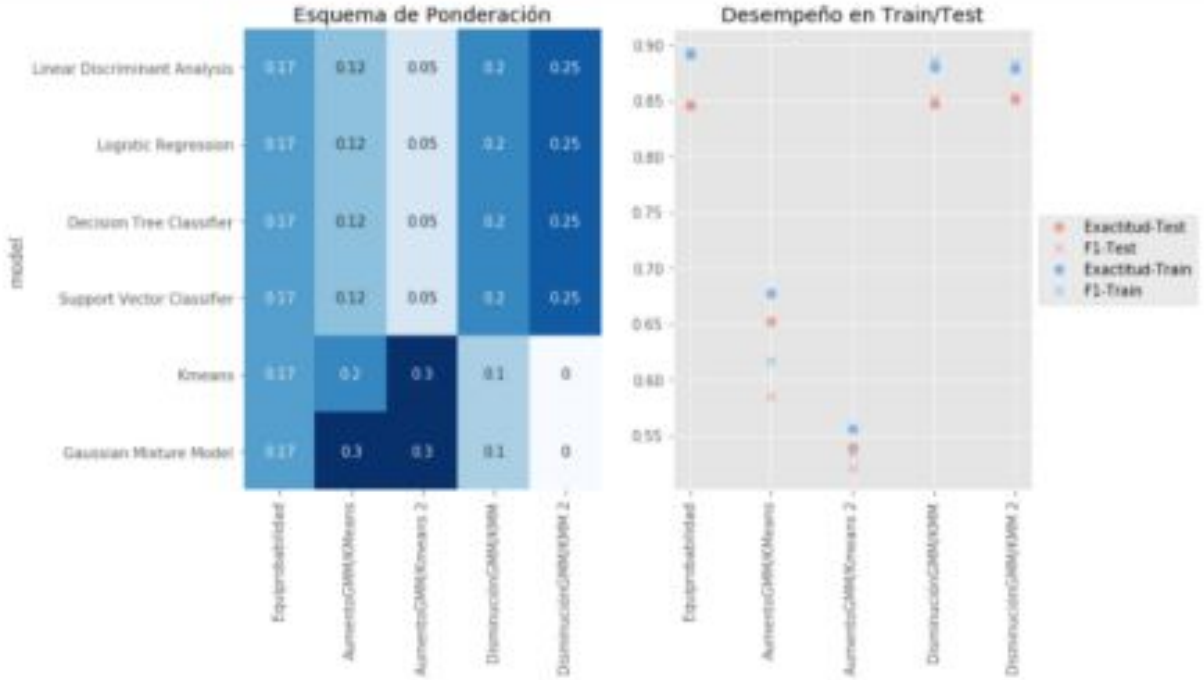
Estrategias de Ponderación

Setup del experimento

```
In [8]: weights_hyperparams = {'Equiprobabilidad': [.17, .17, .17, .17, .17, .17],
                               'AumentoGMM/KMeans': [.125, .125, .125, .125, .20, .30],
                               'AumentoGMM/Kmeans 2': [.05, .05, .05, .05, .30, .30],
                               'DisminuciónGMM/KMM': [.20, .20, .20, .20, .10, .10],
                               'DisminuciónGMM/KMM 2': [.25, .25, .25, .25, .0, .0]}
```

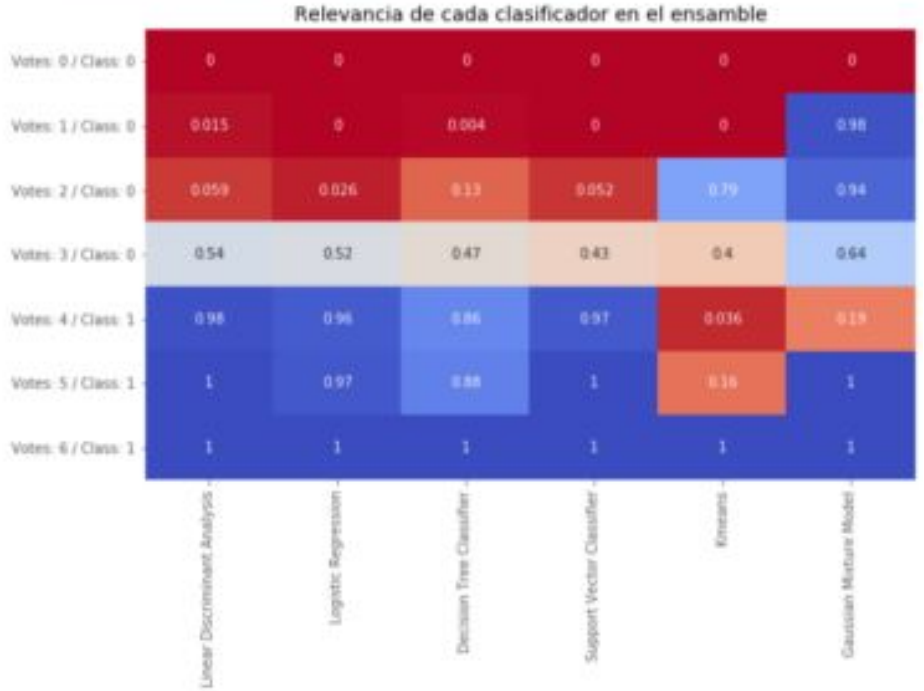
Evaluación

```
In [11]: afx.weighting_schedule(voting_classifier, X_train, X_test, y_train, y_test, weights_hyper
         params)
```



Relevancia del clasificador

```
In [12]: afx.committee_voting(voting_classifier, X_train, X_train, y_train, y_test)
         plt.title('Relevancia de cada clasificador en el ensamble');
```



/* Desafío */

Panel de discusión

{desafío} latam_

Academia de talentos digitales

www.desafiolatam.com