Ensembles

Descrição: São *meta-algoritmos* que combinam várias técnicas de aprendizado de máquina em um único modelo preditivo (Wisdom of the crowd).

Objetivo: Prover estabilidade. Diminuir a variância (bagging), viés (boosting) ou melhorar as predições (stacking).

Estratégia: Combinar as predições de modelos especializados. Um grupo de modelos *fracos* juntos formam um modelo *forte* (United we stand).

Problemas: Podem ser aplicados tanto em problemas de classificação quanto em regressão.

Estruturas:

- Paralelo: Os modelos (heterogêneos ou homogêneos) são treinados paralelamente de forma indepentende e então combinados para gerar a previsão.
- Sequencial: Os modelos são treinados sequencialmente de modo que o erro seja minimizado ao longo da sequência.

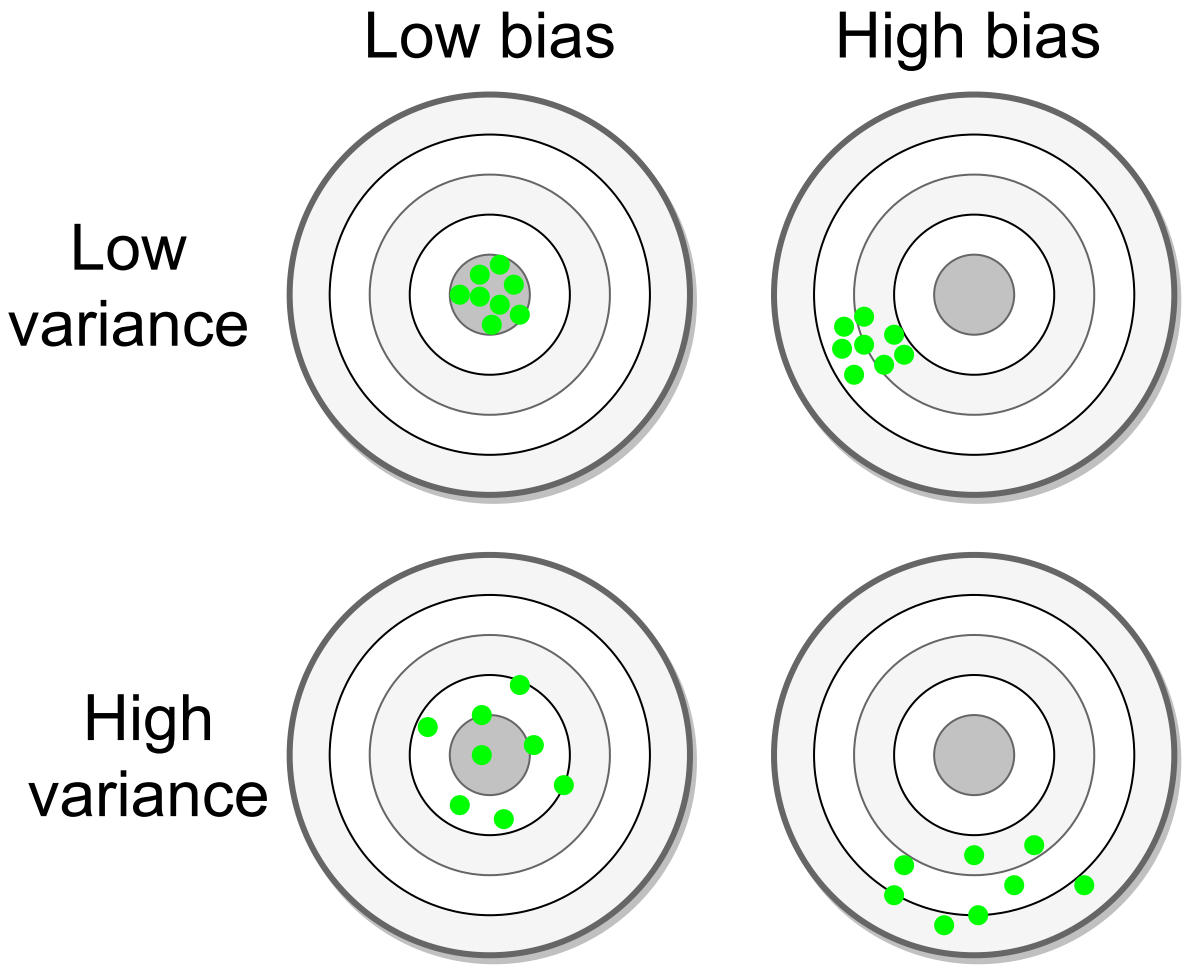
Premissas por *James Michael Surowiecki sobre Wisdom of the crowd*:

- Independência: Capacidade de formar uma opinião independente da opinião alheia.
- Decentralização: Capacidade de especializar e tirar conclusões baseado em informação local.
- Diversidade: Capacidade de manter informação privada mesmo que seja uma opinião enviesada.
- Agregação: Algum mecanismo capaz de tornar julgamentos privados em uma decisão coletiva.

Observações:

- Não são recomendados quando é importante ter interpretabilidade e explicabilidade
- Tempo de computação e desenvolvimento é alto!
- Selecionar modelos para criar ensembles não é trivial
- Nem sempre irá produzir resultados bons

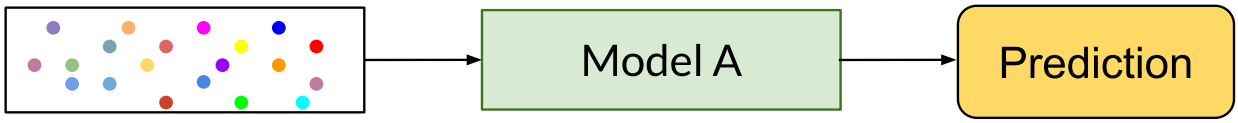
Bias-Variance Tradeoff



$$\text{Erro} = \text{Vies}(\hat{\theta}, \theta) + \text{Variancia}(\hat{\theta}, \theta) + \text{Ruido}$$

Modelo Baseline

```
from mlxtend.evaluate import bias_variance_decomp
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier,
StackingClassifier
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.datasets import make_classification
```



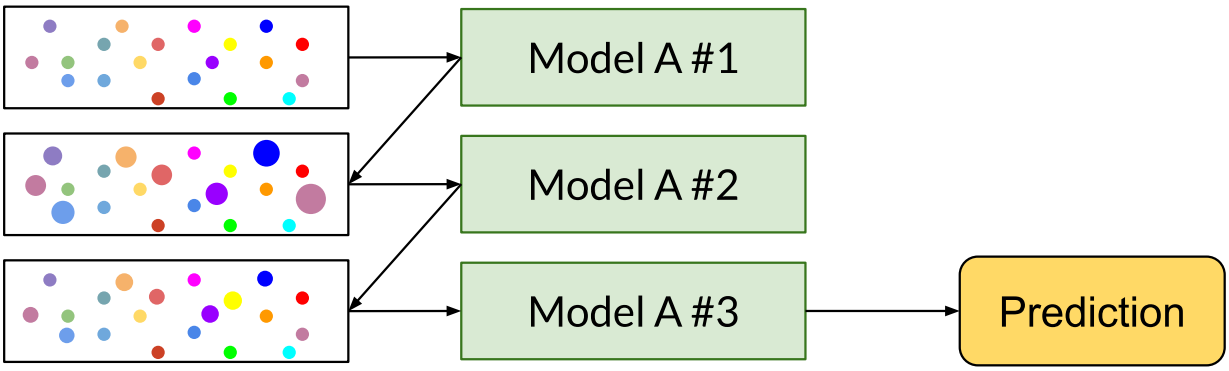

```
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15,
n_redundant=5, random_state=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=1, stratify=y)
```

```
tree = DecisionTreeClassifier(max_depth=3, random_state=1)

# A métrica 0-1_loss é calculada da seguinte forma:
# se y != ŷ então 1 senão 0
# O ruido é ignorado por uma questão de simplificação
loss, bias, var = bias_variance_decomp(tree, X_train, y_train, X_test, y_test,
loss='0-1_loss', random_seed=0)

print('%.3f Average expected loss' % loss)
print('%.3f Average bias' % bias)
print('%.3f Average variance' % var)
```

```
0.223 Average expected loss
0.160 Average bias
0.157 Average variance
```

Boosting

- Estrutura sequencial
- Objetivo é reduzir o viés (bias) do modelo (não a variância)
- Adequado para modelos com baixa variância e alto viés (Baseados em Árvores)
- Baseado na Hypothesis Boosting a qual assume que um conjunto de modelos fracos são capazes de produzir modelos fortes.
- Exemplos comuns: ExtraTrees, GradientBoosting, HistGradientBoosting

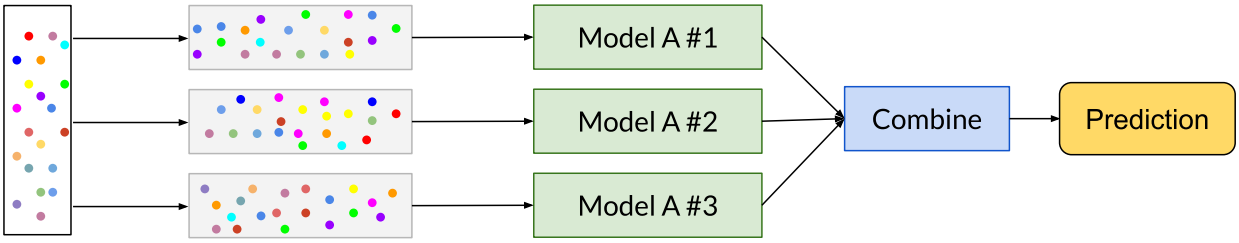



```
tree = DecisionTreeClassifier(max_depth=3, random_state=1)
boosting = AdaBoostClassifier(base_estimator=tree, n_estimators=15,
random_state=1)

loss, bias, var = bias_variance_decomp(boosting, X_train, y_train, X_test,
y_test, loss='0-1_loss', random_seed=1)

print('%.3f Average expected loss' % loss)
print('%.3f Average bias' % bias)
print('%.3f Average variance' % var)
```

```
0.150 Average expected loss
0.064 Average bias
0.131 Average variance
```

Bagging (aka Bootstrap AGGregatING)

- Estrutura paralela e faz a combinação dos modelos usando votação (classificação) ou média (regressão)
- Objetivo é reduzir a variância do modelo (não o viés)
- Adequado para modelos com alta variância e baixo viés (modelos complexos) e conjuntos de dados pequenos/moderados em número de instâncias.
- Baseado na estratégia de Bootstrapping a qual faz uma *amostragem com reposição* de mesma cardinalidade do conjunto de dados original permitindo assim reduzir a variância dos dados.
- Exemplo comum: RandomForest

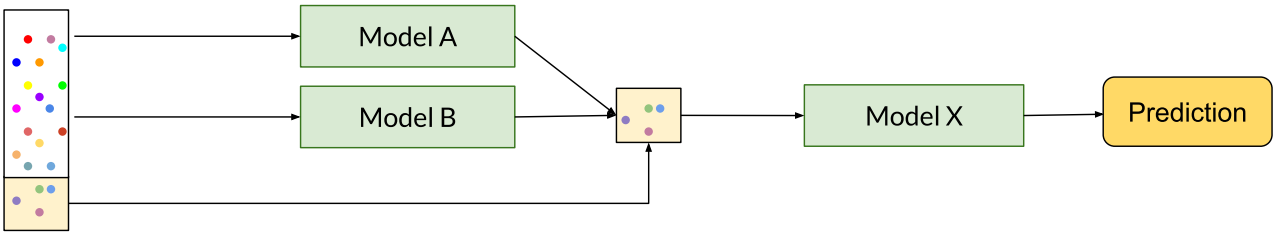



```
tree = DecisionTreeClassifier(max_depth=3, random_state=1)
bagging = BaggingClassifier(base_estimator=tree, n_estimators=15,
random_state=1)

loss, bias, var = bias_variance_decomp(bagging, X_train, y_train, X_test,
y_test, loss='0-1_loss', random_seed=1)

print('%.3f Average expected loss' % loss)
print('%.3f Average bias' % bias)
print('%.3f Average variance' % var)
```

```
0.168 Average expected loss
0.140 Average bias
0.096 Average variance
```

Stacking (aka Stacked Generalization)

- Estrutura paralela e faz combinação dos modelos usando um modelo (*e.g.* LogisticRegression)
- Objetivo é aumentar o poder preditivo (diminuir o erro) do modelo
- Baseado na ideia de que diferentes modelos podem aprender diferentes padrões nos dados mas não o suficiente para generalizar a maioria dos padrões. Dessa forma, a estratégia é usar um modelo que generalize os padrões encontrados por diferentes modelos.




```
estimators = [
    ('rdg', RidgeClassifier(normalize=True)),
    ('gnb', GaussianNB()),
    ('knn', KNeighborsClassifier(n_neighbors=3)),
    ('dt' , DecisionTreeClassifier(max_depth=10, random_state=1)),
    ('svm', make_pipeline(StandardScaler(), SVC(random_state=42)))
]

stacking = StackingClassifier(
    estimators=estimators,
    final_estimator=LogisticRegression()
)

loss, bias, var = bias_variance_decomp(stacking, X_train, y_train, X_test,
y_test, loss='0-1_loss', random_seed=1)

print('%.3f Average expected loss' % loss)
print('%.3f Average bias' % bias)
print('%.3f Average variance' % var)
```

```
0.048 Average expected loss
0.040 Average bias
0.019 Average variance
```

Referências

1. Bagging Ensemble with Python
2. Stacking Ensemble with Python
3. Ensemble Learning
4. Main Approaches for Ensemble Learning
5. Simple Guide for Ensemble Leraning Methods
6. Bias-Variance Decomposition
7. MIT - Boosting
8. Stacking and Blending
9. Imagem: Bias-Variance Tradeoff

Bibliotecas

1. vecstack
2. mlens