Random Forest - Prática

Ciclo	Ciclo 06: Algoritmos baseado em árvores
# Aula	48
Created	@March 9, 2023 10:11 AM
Done	☐
Ready	☑

Objetivo da Aula:

- ☐ Random Forest Classifier
- ☐ Resumo
- ☐ Próxima aula

Conteúdo:

▼ 1. Random Forest Classifier

▼ 1.1 Training RF Classifier

```
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import roc_curve, roc_auc_score
from matplotlib import pyplot as plt
```

```
# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, random_state=1)

# split into train/test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_st
ate=2)

# fit a model
model = RandomForestClassifier(n_estimators=100, max_depth=2, random_state=0)
model.fit(X_train, y_train)
```

▼ 1.2 ROC curve from RF Classifier

```
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import roc_curve, roc_auc_score
from matplotlib import pyplot as plt

# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, random_state=1)

# split into train/test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_st
ate=2)

# fit a model
model = RandomForestClassifier(n_estimators=100, max_depth=2, random_state=0)
model.fit(X_train, y_train)
y_scores = model.predict_proba(X_test)[:, 1]

# Calculate ROC curve values
fpr, tpr, thresholds = roc_curve(y_test, y_scores)

# Calculate AUC score
auc_score = roc_auc_score(y_test, y_scores)

# Plot ROC curve
plt.plot( fpr, tpr, color='b', label=f'ROC curve (AUC = {auc_score:.2f})')
plt.plot([0, 1], [0, 1], color='r', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```

▼ 1.3 Choose best threshold from ROC curve

```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import roc_curve, roc_auc_score
from matplotlib import pyplot as plt
from sklearn.metrics import precision_recall_curve, f1_score, auc, roc_curve, roc_
auc_score, accuracy_score


# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, random_state=1)

# split into train/test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_st
ate=2)

# fit a model
model = RandomForestClassifier(n_estimators=100, max_depth=2, random_state=0)
model.fit(X_train, y_train)
y_scores = model.predict_proba(X_test)[:, 1]

# Calculate ROC curve values
fpr, tpr, thresholds = roc_curve(y_test, y_scores)

# Calculate AUC score
auc_score = roc_auc_score(y_test, y_scores)

# Plot ROC curve
plt.plot( fpr, tpr, color='b', label=f'ROC curve (AUC = {auc_score:.2f})')
plt.plot([0, 1], [0, 1], color='r', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()

# Find the best threshold based on ROC curve
distances = np.sqrt((1 - tpr)**2 + fpr**2)
best_threshold = thresholds[np.argmin(distances)]
print(f"Best Threshold: {best_threshold}")

# Calculate accuracy using the best threshold
y_pred = (y_scores >= best_threshold).astype(int)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

2. Hiperparametros de controle

Os parâmetros que regulam a Decision Tree são:

- *max_depth:* controle o tamanho máximo das quebras, ou seja, o tamanho máximo do crescimento da árvore ou ainda o número de recortes espaciais.
- *min_samples_leaf:* O número mínimo de amostras do nó deve ter, antes de fazer uma nova separação e gerar nós filhos.
- *min_weight_fraction_leaf:* A mesma definição do parâmetro min_samples_leaf, mas definida como a fração do número total dos pesos das evidências.
- *max_features:* O número máximo de atributos que são avaliados para a divisão de cada nó.
- *class_weight:* é usado para lidar com classes desbalanceadas em um problema de classificação.

3. Resumo

1. O aprendizado não-supervisionado tem o objetivo de agrupar indivíduos com características ou comportamentos semelhantes, para encontrar padrões.

4. Próxima aula

Métricas de avaliação I: Curva ROC