



Bootcamp: Analista de Machine Learning

Enunciado da Atividade Modular

Módulo 2: Técnicas Avançadas para Machine Learning

Descrição do Módulo

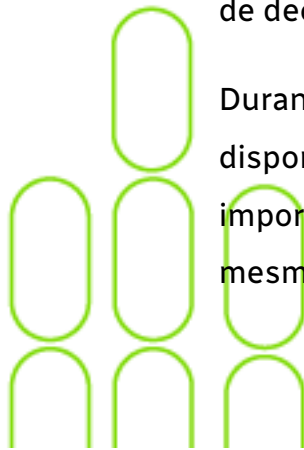
Para a nossa atividade modular, vamos utilizar um dataset público, de casos de câncer de mama, da universidade de Wisconsin, que pode ser encontrado na plataforma Kaggle no seguinte link:


<https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data>

O objetivo dessa atividade será construir um modelo de classificação, para detecção da doença. Para isso, iremos passar por diversas etapas, na seguinte ordem:

1. Separação do dataset em partições de treino e teste, com 70% dos dados no treino e 30% dos dados no teste;
2. Redução de dimensionalidade aplicando PCA e uso de componentes que representem pelo menos 70% da variância dos dados;
3. Teste de hiperparâmetros em um classificador de árvore de decisão, utilizando validação cruzada;
4. Construção de um método ensemble Bagging aplicando a melhor árvore de decisão da etapa anterior.

Durante todo o processo, parâmetros importantes aos códigos serão disponibilizados, que devem ser usados nas funções. Isso é muito importante para garantir a reprodutibilidade dos resultados, usando os mesmos seeds random. Atente-se aos comandos pois nem tudo estará





preenchido, o que aparecer com reticências deverá ser preenchido corretamente de acordo com o enunciado.

- Instalação das bibliotecas:

```
pip install pandas sklearn
```

- Bibliotecas a importar:

```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import accuracy_score
```

- Preparação do dataset:


Considerando que o dataset seja lido e armazenado em um dataframe Pandas df, devemos separá-lo em X e y:

- Para os features X, vamos remover as 2 primeiras colunas e a última, pois essas não são valores de interesse;
- Para os valores de referência y, usamos a coluna de índice 1 do dataset original. Além disso aplicamos um dicionário para mapear 'B' de Benigno para 0 e 'M' de Maligno para 1, pois o modelo precisa de variáveis numéricas ao invés de texto. Ignorar os avisos warning do sistema.

```
X = df.iloc[:, 2:-1]
y = df.iloc[:, 1]
y = y.replace( {'B': 0, 'M': 1} )
```

- Divisão treino-teste:





Para a divisão em treino e teste, vamos separar 70% para treino e 30% para teste, declarando o seed random. Use os seguintes parâmetros, completando corretamente:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=...,
random state=10)
```

- Redução Dimensionalidade com PCA:

Para a aplicação do PCA é importante lembrar que primeiro devemos aplicar o Standard Scaler, usando fit_transform na base de treino, mas apenas o transform na base de teste:

```
scaler = StandardScaler()
X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train),
columns=X_train.columns)
X_test_scaled = pd.DataFrame(scaler.transform(X_test),
columns=X_test.columns)
```


A seguir é possível aplicar a transformação com PCA nos dados já escalonados:

```
pca = PCA(n_components=10, random_state=10)
pca.fit(X_train_scaled)
```

Após isso, será possível analisar a explicabilidade das variâncias nos dados, a partir dos componentes:

```
pca.explained_variance_ratio_
```

Em nosso problema, queremos criar os modelos de classificação utilizando apenas os primeiros componentes que representem pelo menos 70% da variância dos dados. Para isso você deve analisar as variâncias das componentes. Feita a análise, criamos então os datasets X_train_pca e



X_test_pca, transformados pelo PCA e utilizando o número de componente identificado por você, em n_components.

```
pca = PCA(n_components=..., random_state=10)
pca.fit(X_train_scaled)
X_train_pca = pca.transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
```

- Busca de Hiperparâmetros com Validação Cruzada

Iremos utilizar um DecisionTreeClassifier como modelo base e os hiperparâmetros que queremos testar serão:


Hiperparâmetros	Valores
max_depth	3, None
min_samples_split	2, 10

Você deve definir qual o tipo de Busca irá aplicar: GridSearch ou RandomSearch. A busca deve ser realizada utilizando-se validação cruzada com 5 partições e a acurácia ('accuracy') como métrica. O seed random do modelo deve ser declarado:

```
estimator = DecisionTreeClassifier(random_state=10)
grid = ...Search(estimator = estimator,
                 param_grid/param_distributions = ...,
```

```
scoring = ...,
cv = ...)
```

Lembre-se que após a busca pelos hiperparâmetros, é possível analisar os resultados visualizando as saídas:



```
grid.cv_results_  
grid.best_index_  
grid.best_params_  
grid.best_score_  
grid.best_estimator_
```

- Método Ensemble

Vamos aplicar o método Bagging, utilizando como modelo base nossa árvore de decisão com os melhores hiperparâmetros encontrados na etapa anterior. Portanto será necessário criar um modelo base, com os melhores hiperparâmetros:

```
estimator = DecisionTreeClassifier(random_state = 10,  
                                   max_depth = ...,  
                                   min_samples_split = ...)
```

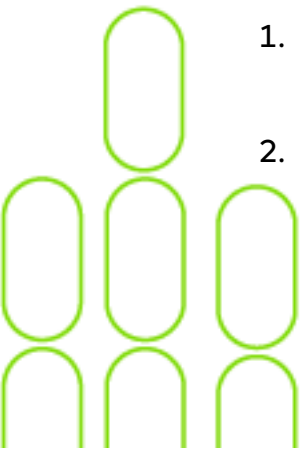
Construir o modelo ensemble com 100 estimadores:

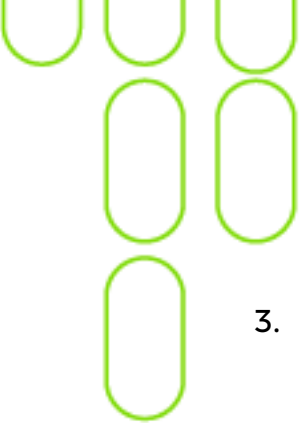

```
ensemble = BaggingClassifier(estimator = estimator,  
                             n_estimator = ...,  
                             random_state = 10)
```

E por fim deve-se treiná-lo com `X_train_pca` e `y_train`, usá-lo para prever com `X_test_pca` e comparar as previsões `y_pred` com o `y_test` utilizando a métrica acurácia, chamando a função `accuracy_score`.

Objetivo de Ensino

Espera-se que o aluno consiga, ao final deste Módulo:

- 
1. Preparar o dataset e separar corretamente as partições de treino e teste
 2. Aplicar redução de dimensionalidade com PCA e identificar os componentes que representem 70% da variância dos dados

- 
- 
3. Buscar pelos melhores hiperparâmetros de um modelo de árvore de decisão
 4. Criar um modelo ensemble Bagging com o melhor modelo de árvore de decisão encontrado na etapa anterior