Treinar modelo de machine learning para classificação de espécies de Iris (flor)

JOHNSON TAVARES PINTO

MANAUS – AM

2021

Treinar modelo de machine learning para classificação de espécies de Iris (flor).

Para isso você deverá seguir as instruções abaixo:

1) Usar a linguagem python e suas bibliotecas, especialmente, a biblioteca scikit-learn de suporte a machine learning;

2) Ajustar o dataset (pré-processamento);

3) Treinar modelo baseado em Árvore de Decisão;

4) Executar as predições (classificação de espécies de Iris) usando o modelo treinado;

5) Avaliar a acurácia do modelo;

6) Apresentar por meio de um relatório os resultados obtidos;

7) Disponibilizar no github, o código-fonte e o relatório.

Obs.: você deverá utilizar o dataset abaixo:
http://archive.ics.uci.edu/ml/datasets/Iris

A seguir será apresentado as células do código de modelo de machine learning para classificação de espécies de Iris (flor).

```
import pandas as pd
from sklearn import tree
from sklearn.model_selection import train_test_split

iris = 'http://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data'
df = pd.read_csv(iris, sep=',')
attributes = ["sepal_length", "sepal_width", "petal_length", "petal_wid
th", "class"]
df.columns = attributes
n_dados = df_treino.count()
print(n_dados)
```

```
sepal_length    149
sepal_width     149
petal_length    149
petal_width     149
class           149
dtype: int64
```

Primeiramente é importado a biblioteca pandas, tree, e train_test_split. A variável iris recebe o endereço da base de dados, em seguida df recebe o database convertido para csv, utilizando a biblioteca pandas, é criada uma lista denominada attributes que recebe os nomes das 4 features da flor Iris
"sepal_length", "sepal_width", "petal_length", "petal_width", "class"
e a classificação no database.

Para um melhor entendimento da database é utilizado o método count(), este método retorna o número de amostras de cada feature e classe que são 149.

```
X_treino, X_teste, y_treino, y_teste = train_test_split(df.drop('class'
, axis=1), df['class'], test_size=0.3)
```

O método train_test_split faz uma partição randômica em dois subsets para treinar e testar os dados. Com esta função não é necessário dividir o dataset manualmente. Por padrão é possível também especificar o estado randômico para a operação, haverá uma amostra de 70% para treinar o modelo e 30% das amostras serão para teste.

```
model = tree.DecisionTreeClassifier()
    model.fit(X_treino, y_treino)
```

DecisionTreeClassifier é uma classe capaz de realizar a classificação multiclasse em um conjunto de dados, o DecisionTreeClassifier leva como entrada duas matrizes uma matriz

X contendo as amostras de treinamento e uma matriz Y contendo os rótulos de classe para as amostras de treinamento.

```
model.feature_importances_
```

A feature_importances_ tem a finalidade de estimar a importância de cada feature. Ele retorna um score para cada atributo, quanto maior o score, maior é a importância desse atributo.

```
display(X_teste)
```

O display(X_teste) tem a função de mostrar a matriz de teste formatada com suas features, vale notar que não há mais a coluna “class” pois esses dados serão usados para as predições(teste) , a figura a seguir mostra os dados para teste.

	sepal_length	sepal_width	petal_length	petal_width
118	6.0	2.2	5.0	1.5
41	4.4	3.2	1.3	0.2
54	5.7	2.8	4.5	1.3
122	6.3	2.7	4.9	1.8
64	6.7	3.1	4.4	1.4
128	7.2	3.0	5.8	1.6
31	5.2	4.1	1.5	0.1
43	5.1	3.8	1.9	0.4
142	6.8	3.2	5.9	2.3

```
y_pred = model.predict(X_teste)
y_pred
```

y_pred recebe as predições da base de X_teste, os valores dessas predições podem ser vistos nas figura a seguir.

```
array(['Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa'], dtype=object)
```

```
y_teste == y_pred
```

O comando acima compara os rótulos da base de teste original com os rótulos identificados nas predições.

102	True	118	False
58	True	41	True
106	True	54	True
40	True	122	True
73	True	64	True
89	True	128	True
92	True	31	True
18	True	43	True
12	True	142	True
23	True	29	True
34	True	126	True
16	True	77	True
0	True	104	True
52	True	113	True
75	True	139	True
13	True	37	True
32	True	141	True
71	True	136	True
83	True	112	True
35	True	85	True
127	True	81	True

Os que tiveram um valor acompanhado de “True” significa que a predição foi correta e “False” foi uma predição incorreta.

```
from sklearn import metrics
print(metrics.classification_report(y_teste, y_pred))
```

metrics.classification_report() retorna as métricas do modelo, a figura abaixo apresenta essas métricas

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	16
Iris-versicolor	0.93	1.00	0.97	14
Iris-virginica	1.00	0.93	0.97	15
accuracy			0.98	45
macro avg	0.98	0.98	0.98	45
weighted avg	0.98	0.98	0.98	45

A seguir é plotado a matrix de confusão nela é possível ver o algoritmo de classificação, nela é possível ver o número de falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos. Isso permite uma análise mais detalhada.

```
from sklearn.metrics import confusion_matrix
  print(confusion_matrix(y_pred,y_teste))
```

	Setosa predição	**Versicolar predição**	**Virginica predição**
Setosa real	18	0	0
Versicolar real	0	11	**1**
Virgínica real	0	**2**	13

Na tabela é possível ver as instâncias de testes, de acordo com a tabela é possível ver que:

- Para Iris Setora houve 100% de precisão nas predições.
- 2 plantas Vigínica foram classificadas como versicolar.
- 1 planta versicolar foi classificada como virginica.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(40,40))
tree.plot_tree(model, filled=True, fontsize=42)
plt.savefig("arvore_decisao1.png")
plt.close()
```

A célula acima com o auxílio da biblioteca matplot lib tem a finalidade de criar a árvore de decisão

Caso x[3] seja <= 0.75 a árvore irá pelo caminho da esquerda, caso x[3] seja o contrário a árvore irá pela direita.