Лабораторная работа

Построение системы ИИ с помощью инструментов Scikit-Learn для решения задачи классификации

Выполнила: Короткова Инга Сергеевна

2020 год

Построение системы ИИ с помощью инструментов Scikit-Learn для решения задачи классификации

Цель работы:

научиться строить модели классификации.

Задачи:

- Научиться строить классификаторы
- Научиться оценивать его качество
- Изучить вклад используемых переменных в решения классификатора
- Научиться визуализировать классификатор (дерево решений)

Импортируем библиотеки

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import graphviz
# для изменения параметров визуализируемого дерева
import pydotplus
from sklearn.tree import export_graphviz
from IPython.display import SVG
```

```
# распакуем архив с данными
!unzip data.zip
```

```
Archive:  data.zip
  inflating: data.csv
```

Загрузим данные из файла csv

```
data = pd.read_csv('data.csv').drop(columns=['Unnamed: 0', 'id', 'key']).set_index(['artists', 'name'])
```

```
data.head(7)
```

artists	name	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	liveness	loudness	mode	popularity	rel
['Dennis Day']	Clancy Lowered the Boom	0.732	0.819	180533	0.341	0	0.000000	0.1600	-12.441	1	8	
['Sergei Rachmaninoff', 'James Levine', 'Berliner Philharmoniker']	Piano Concerto No. 3 in D Minor, Op. 30: III. Finale. Alla breve	0.982	0.279	831667	0.211	0	0.878000	0.6650	-20.096	1	5	
['John McCormack']	The Wearing of the Green	0.996	0.518	159507	0.203	0	0.000000	0.1150	-10.589	1	6	
['Sergei Rachmaninoff', 'James Levine', 'Berliner Philharmoniker']	Piano Concerto No. 3 in D Minor, Op. 30: III. Finale. Alla breve	0.982	0.279	831667	0.211	0	0.878000	0.6650	-20.096	1	4	
['Phil Regan']	When Irish Eyes Are Smiling	0.957	0.418	166693	0.193	0	0.000002	0.2290	-10.096	1	4	
	Come Back To Erin	0.957	0.259	186467	0.212	0	0.000222	0.2360	-13.300	1	2	
['Sergei Rachmaninoff', 'Zubin Mehta', 'Vladimir Feltsman', 'Israel Philharmonic Orchestra']	Rhapsody on a Theme of Paganini, Op. 43: Var. XVIII, Andante cantabile	0.992	0.241	167867	0.109	0	0.853000	0.0771	-19.286	1	3	

Изменим тип признака release_date на datetime и извлечем только год.

Это делается для того, чтобы привести признак к тому виду, чтобы решающее дерево могло использовать его как численный признак при построении дерева.

```
data.release_date = pd.to_datetime(data.release_date, yearfirst=True)
data.release_date = data.release_date.dt.year
```

Выберем целевой переменной - popularity, убрав ее из набора данных для обучения X.

```
X = data.drop(columns=['popularity'])
y = data.popularity
```

Создадим классификатор - решающее дерево, ограничим его глубиной 3 и минимальным количеством листьев - 5.

Также следует отметить, что используется критерий разбиения Gini impurity является мерой, как часто случайно выбранный элемент из набора неверно помечается, если он случайным образом помечается согласно распределению меток в подмножестве.

Решающее дерево это алгоритм машинного обучения (обучение с учителем).

В общем виде они воспроизводят иерархическую древовидную структуру, в узлах которой содержится какой-либо вопрос (да-нет, больше-меньше), причём вопрос задаваемый на следующем уровне зависит от ответа, полученного на предыдущем уровне.

```
clf = DecisionTreeClassifier(random_state=0, max_depth=3, min_samples_leaf=5)
```

Обучим классификатор

```
clf.fit(X,y)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=3, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=5, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

Посмотрим, какие переменные каким весом обладают при построении решающих правил:

```
clf.feature_importances_
```

```
array([0.01841238, 0.        , 0.00499254, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.00514254, 0.        , 0.        , 0.97145253])
```

Для более удобного воспринятия информации о важности переменных построим датафрейм с названиями переменных и их важностью.

```
pd.DataFrame(data = clf.feature_importances_, index=X.columns, columns=['feature_importances'])
```

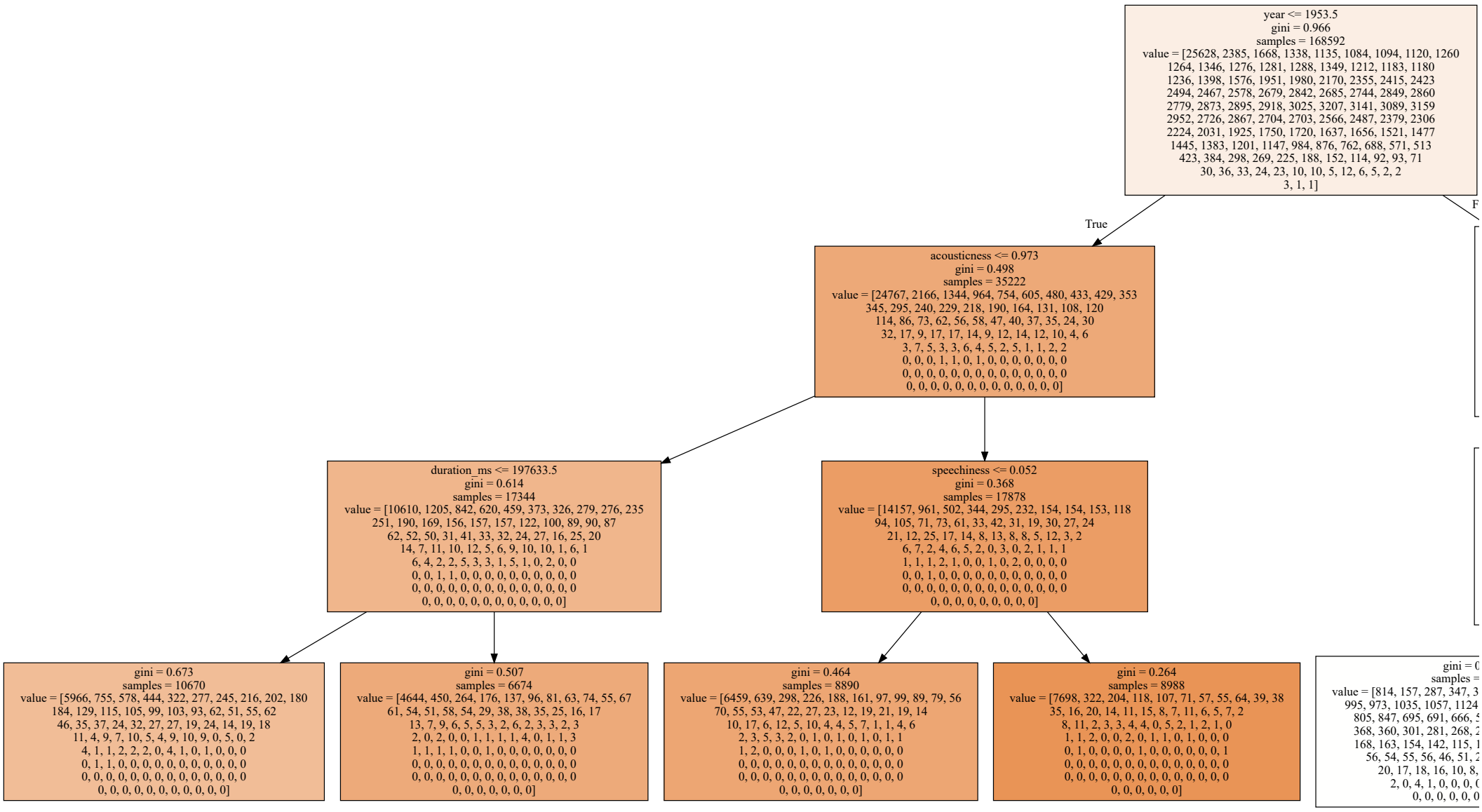
	feature_importances
acousticness	0.018412
danceability	0.000000
duration_ms	0.004993
energy	0.000000
explicit	0.000000
instrumentalness	0.000000
liveness	0.000000
loudness	0.000000
mode	0.000000
release_date	0.000000
speechiness	0.005143
tempo	0.000000
valence	0.000000
year	0.971453

Визуализируем получившееся дерево

```
# выделим названия колонок для их отображения
col_names = X.columns
dot_data = export_graphviz(clf, out_file=None, filled=True, feature_names=col_names)
```

```
# изменим размер, для более удобного отображения
pydot_graph = pydotplus.graph_from_dot_data(dot_data)
pydot_graph.set_size('"20,25!"')


# graph = graphviz.Source(dot_data)
# display(SVG(graph.pipe(format='svg')))
# визуализируем дерево.
gvz_graph = graphviz.Source(pydot_graph.to_string())
gvz_graph
```



Как видно из дерева, наиболее важным признаком оказался год, а затем acousticness.

Заключение.

В этой работе мы познакомились с таким алгоритмом машинного обучения как решающие деревья.

Решающее дерево использовалось для построения и отображения набора признаков, которые наиболее эффективно делят набор данных на части.

Для этого, мы:

- разделили набор данных на входные данные модели и целевую переменную
- построили классификатор
- оценили вклад переменных
- визуализировали дерево решений

К достоинствам деревьев решений можно отнести:

- легко интерпритировать результаты.
- быстрый процесс обучения.
- классификатор хорошо интепритирует признаки, которые достаточно сложно эксперту (человеку) быстро формализовать.

```
```