**DZ\_ Анализ данных\_Классификация с использованием Python. Деревья решений.**

**Part 1: Анализ и обучение.**

Библиотека scikit-learn предоставляет реализацию целого ряда алгоритмов для обучения с учителем (Supervised Learning) и обучения без учителя (Unsupervised Learning) через интерфейс для языка программирования Python. Данная библиотека распространяется под лицензией "Simplified BSD License" и имеет дистрибутивы для множества различных версий Linux, поощрая тем самым академическое и коммерческое использование scikit-learn. Scikit-learn построена поверх [SciPy](http://www.scipy.org/) (Scientific Python), который должен быть установлен перед использованием scikit-learn. Данный стек включает в себя:

* **NumPy**: расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых математических функций для операций с этими массивами
* **SciPy**: открытая библиотека высококачественных научных инструментов для языка программирования Python
* **Matplotlib**: библиотека на языке программирования Python для визуализации данных двумерной (2D) графикой (3D графика также поддерживается)
* **IPython**: интерактивная оболочка для языка программирования Python, которая предоставляет расширенную интроспекцию, дополнительный командный синтаксис, подсветку кода и автоматическое дополнение
* **Sympy**: библиотека для символьных вычислений
* **Pandas**: различные структуры данных и анализ

Библиотека scikit-learn ориентирована в первую очередь на моделирование данных, а не на загрузку, манипуляцию и обобщение данных. Для таких целей, как раз-таки, лучше использовать NumPy и Pandas.

Вот несколько популярных функциональных областей, в которых scikit-learn помогает решать поставленные задачи:

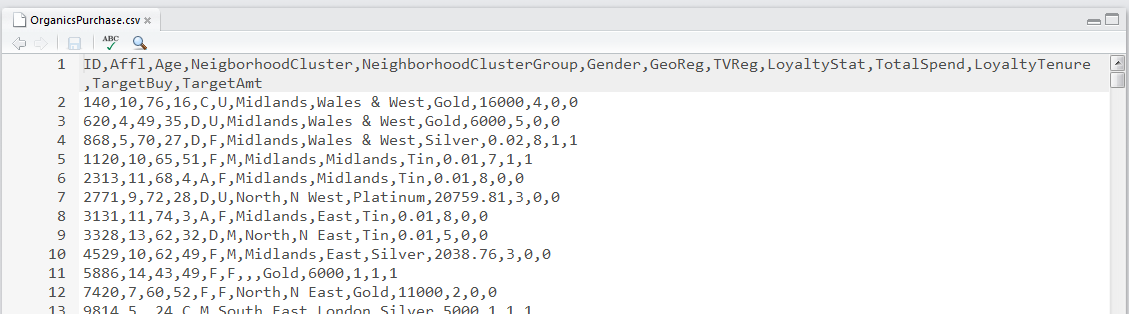
* **Кластеризация (Clustering):** для группировки неразмеченных данных, например, метод k-средних (k-means)
* **Перекрестная проверка (Cross Validation):** для оценки эффективности работы модели на независимых данных
* **Наборы данных (Datasets):** для тестовых наборов данных и для генерации наборов данных с определенными свойствами для исследования поведенческих свойств модели
* **Сокращение размерности (Dimensionality Reduction):** для уменьшения количества атрибутов для визуализации и отбора признаков (Feature Selection), например, метод главных компонент (Principal Component Analysis)
* **Алгоритмические композиции (Ensemble Methods):** для комбинирования предсказаний нескольких моделей
* **Извлечение признаков (Feature Extraction):** определение атрибутов в изображениях и текстовых данных
* **Отбор признаков (Feature Selection):** для выявления значимых атрибутов на основе которых будет построена модель
* **Оптимизация параметров алгоритма (Parameter Tuning):** для получения максимально эффективной отдачи от модели
* **Множественное обучение (Manifold Learning):** для нелинейного сокращения размерности данных
* **Алгоритмы обучения с учителем (Supervised Models):** огромный набор методов не ограничивается обобщенными линейными моделями (Generalized Linear Models), дискриминантным анализом (Discriminate Analysis), наивным байесовским классификатором (Naive Bayes), нейронными сетями (Neural Networks), методом опорных векторов (Support Vector Machines) и деревьями принятия решений (Decision Trees).

1. **Прочитайте данные для анализа в python с помощью пакета pandas**

Для этого упражнения вам понадобится файл: Organics.csv (файл данных).

Файл данных содержит 22 223 записей о клиентах с демографической информацией и информацией о том, покупал ли клиент органические продукты. Содержание проблемы: супермаркет начинает предлагать новую линию органических продуктов. Руководство супермаркета хотело бы определить, какие покупатели могут приобрести эти продукты.

В супермаркете действует программа лояльности клиентов. В качестве первоначального плана стимулирования покупателей супермаркет предоставлял купоны на свои органические продукты всем участникам программы лояльности и теперь собирает данные, которые включают информацию о том, приобрели ли эти клиенты какие-либо из органических продуктов или нет.



.

**Необходимые библиотеки**

*import numpy as np*

*import pandas as pd*

*import pylab as pl*

*import matplotlib.pyplot as plt*

*plt.style.use('ggplot')*

*%matplotlib inline*

*import seaborn as sns*

Полный список переменных модели:

|  |  |
| --- | --- |
| **Variable Name** | **Variable Description** |
| **ID** | Customer loyalty identification number |
| **Affl** | Affluence grade on a scale from 1 to 30 (1 = least affluent, 30 = most affluent) |
| **Age** | Age, in years |
| **Cluster** | Identifier for residential neighborhood |
| **ClusterGroup** | A set of similar neighborhoods  (A-F = neighborhood type, U=unknown) |
| **Gender** | M = male, F = female, U = unknown |
| **GeoReg** | Geographic region |
| **TVReg** | Television region |
| **Promclass** | Loyalty status: Tin, Silver, Gold, or Platinum |
| **Spend** | Total amount spent (in British pounds) |
| **TargetBuy** | Organics purchased? 1 = Yes, 0 = No |
| **TargetAmt** | Number of organic products purchased |

TargetBuy - это целевая переменная результата, потому что она описывает дискретное событие (1 = сделал покупку, 0 = не купил).

Некоторые переменные, такие как ID, не имеют отношения к анализу. Другие переменные, такие как TargetAmt, также бесполезны, поскольку они не дают дополнительной информации о результате (очевидно, если TargetAmt больше 0, тогда TargetBuy будет 1).

**Вывести данные, описательную статистику , проверить корреляции между признаками.**

*df=pd.read\_csv("D:\ORGANICS.csv",encoding="cp1251")*

*print(df.shape)*

*print(df.info())*

**В некоторых строках отсутствуют данные. Провести дальнейший анализ по трем ситуациям:**

1. Удаление Nan (*dropna*())
2. Замена Nan усредненными значениями (*fillna*())
3. Замена Nan наиболее часто встречаемыми значениями

**Для анализа в этой части используем пока только количественные признаки (Affl , Age, Cluster, Promspend,PromTime). Для этого выбрать только нужные столбцы Dataframe – x . Выбрать целевую - y.**

*x=df[['DemAffl', 'DemAge', 'DemCluster', 'PromSpend', 'PromTime']]*

*y=df.values[::,11]*

1. **Обучение модели**

**Функция *DecisionTreeClassifier* Описание:**

|  |  |
| --- | --- |
| **Параметры :** | **criterion** : string, optional (default = "gini")  Функция для измерения качества раскола. Поддерживаемые критерии - «gini» для примеси Gini и «энтропии» для получения информации.  **splitter** : string, optional (default = "best")  Стратегия, используемая для выбора разделения на каждом узле. Поддерживаемые стратегии «лучше всего», чтобы выбрать лучший раскол и «случайный», чтобы выбрать лучший случайный сплит.  **max\_depth** : int или None, необязательно (по умолчанию = Нет)  Максимальная глубина дерева. Если None, то узлы расширяются до тех пор, пока все листья не станут чистыми или пока все листья не будут содержать меньше, чем min\_samples\_split.  **min\_samples\_split** : int, float, optional (по умолчанию = 2)  Минимальное количество выборок, необходимых для разделения внутреннего узла:   * Если int, рассмотрите min\_samples\_split как минимальное число. * Если float, то min\_samples\_split - это процент, а ceil(min\_samples\_split \* n\_samples) - это минимальное количество выборок для каждого разделения.   Изменено в версии 0.18: Добавлены значения float для процентов.  **min\_samples\_leaf** : int, float, optional (default = 1)  Минимальное количество образцов, которые должны быть у листового узла:   * Если int, тогда считайте min\_samples\_leaf как минимальное число. * Если float, то min\_samples\_leaf - это процент, а ceil(min\_samples\_leaf \* n\_samples) - это минимальное количество выборок для каждого узла.   Изменено в версии 0.18: Добавлены значения float для процентов.  **min\_weight\_fraction\_leaf** : float, optional (по умолчанию = 0.)  Минимальная взвешенная доля суммарного веса (всех входных выборок) должна быть у листового узла. Образцы имеют одинаковый вес, когда sample\_weight не предоставляется.  **max\_features** : int, float, string или None, optional (default = None)  Количество функций, которые нужно учитывать при поиске лучшего раскола:   * Если int, то рассмотрите функции max\_features в каждом раздвоении. * Если float, то max\_features - это процент, а int(max\_features \* n\_features) функции учитываются при каждом расколе. * Если «auto», то max\_features=sqrt(n\_features) . * Если «sqrt», то max\_features=sqrt(n\_features) . * Если «log2», то max\_features=log2(n\_features) . * Если None, то max\_features=n\_features .   Примечание: поиск разделения не прекращается до тех пор, пока не будет найден хотя бы один действительный раздел выборок узлов, даже если он требует эффективного max\_features функциями max\_features .  **random\_state** : int, случай RandomState или None, необязательный (по умолчанию = нет)  Если int, random\_state - это семя, используемое генератором случайных чисел; Если экземпляр RandomState, random\_state - генератор случайных чисел; Если None, генератор случайных чисел является экземпляром RandomState, используемым np.random .  **max\_leaf\_nodes** : int или None, необязательный (по умолчанию = нет)  Вырастите дерево с max\_leaf\_nodes наилучшим образом. Наилучшие узлы определяются как относительное уменьшение примеси. Если None, то неограниченное количество листовых узлов.  **min\_impurity\_decrease** : float, optional (default = 0.)  Узел будет разделен, если этот раскол индуцирует уменьшение примеси, большее или равное этому значению.  Уравновешенное уравнение уменьшения примеси следующее:  N\_t / N \* (impurity - N\_t\_R / N\_t \* right\_impurity  - N\_t\_L / N\_t \* left\_impurity)  где N - общее количество выборок, N\_t - количество выборок в текущем узле, N\_t\_L - количество выборок в левом дочернем N\_t\_R , а N\_t\_R - количество выборок в правом дочернем N\_t\_R .  N , N\_t , N\_t\_R и N\_t\_L все относятся к взвешенной сумме, если sample\_weight передан.  Новое в версии 0.19.  **min\_impurity\_split** : float,  Порог для ранней остановки роста дерева. Узел будет разделяться, если его примесь выше порога, иначе это лист.  Устаревший с версии 0.19: min\_impurity\_split устарел в пользу min\_impurity\_decrease в 0.19 и будет удален в 0.21. min\_impurity\_decrease этого используйте min\_impurity\_decrease .  **class\_weight** : dict, список dicts, "сбалансиров |

**Разбиение на обучение и контроль:**

Чтобы собрать тестовую и обучающую выборку из исходных данных, воспользуемся удобной функцией кросс-валидации train\_test\_split, реализованной в scikit-learn. С готовыми выборками работаем дальше — импортируем RandomForestClassifier из ensemble в sklearn. Этот класс содержит в себе все необходимые для обучения и тестирования машины методы и функции. Вызовом функции fit() обучаем машину из класса clf, где X\_train — признаки категорий y\_train.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3)

**Обучение модели:**

*from sklearn.tree import DecisionTreeClassifier*

*tree=DecisionTreeClassifier(max\_depth=4, random\_state=0)*

*tree.fit(X\_train, y\_train)*

1. **Визуализация модели**

После обучения мы можем экспортировать дерево в формате [Graphviz](https://www.graphviz.org/) , используя экспортер [export\_graphviz](https://code-examples.net/ru/docs/scikit_learn/modules/generated/sklearn.tree.export_graphviz#sklearn.tree.export_graphviz) . Ниже приведен пример exportviz экспорта вышеупомянутого дерева, обученного по всему набору данных; результаты сохраняются в выходном файле \*.png :

from sklearn.tree import export\_graphviz

import subprocess

export\_graphviz(tree, out\_file='E:/Organics\_tree.dot',

rounded = True, proportion = False,

precision = 2, filled = True)

import pydot

(graph, ) = pydot.graph\_from\_dot\_file('E:/Organics\_tree.dot')

graph.write\_png('E:/Organics\_tree.png')

**Самостоятельно:** найти другие варианты визуализации деревьев и функции для отображения разделяющей поверхности

1. **Оценка точности модели**

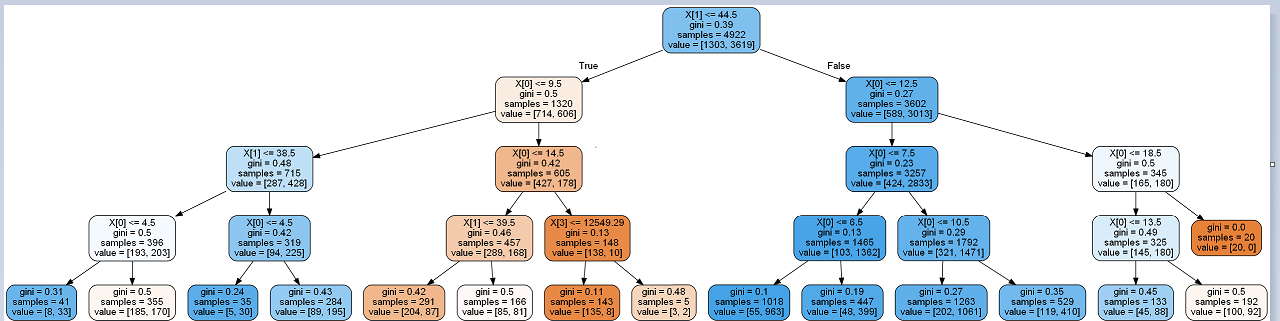
Используем встроенную в класс метрику score, чтобы определить точность предсказанных для X\_test категорий по истинным значениям этих категорий y\_test. При использовании данной метрики выводится значение точности от 0 до 1, где 1 <=> 100%

tree.score(X\_train, y\_train) # результат на обучающей выборке

0.9995936611133686

tree.score(X\_test,y\_test) # результат на тестовой выборке- число правильных результатов

0.706251088281386



1. Дерево правильно предскажет, купит ли кто-нибудь органику в 76,15% случаев. У дерева есть листовые узлы (узлы, под которыми ничего нет). Каждый из этих конечных узлов представляет прогноз, основанный на комбинации переменных предиктора.

**Задание:**

**DZ:**

**1) Поиграть с другими параметрами модели датасета Organics, сделать выводы**

1. Обучите решающее дерево с параметрами **criterion** = entropy и **max\_depth** =5. Какую точность дает данное дерево на контроле? Визуализируйте разделяющую поверхность.

*Полезные функции: sklearn.metrics.accuracy\_score*

1. Выполните предыдущий пункт для разных значений глубины дерева и минимального числа объектов в листе (параметры max\_depth и min\_samples\_leaf). Результаты запишите в виде таблицы. Как эти параметры влияют на вид разделяющей поверхности и точность модели?
2. Реализуйте бинарное кодирование категориальных признаков. Если у признака N возможных значений, то ему в соответствие ставится N бинарных признаков, каждый из которых соответствует одному значению исходного признака. На каждом объекте единице должен равняться строго один из них. Примените такое кодирование к столбцам c категориальными данными, добавьте результат к признакам. Удается ли с помощью новых признаков повысить качество классификации? (<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html> )
3. Построить дерево решений для датасетов по вариантам, оформить блокноты с описанием задачи, признаков:

а) titanic.csv (Описание https://www.kaggle.com/c/titanic/data)

b) telecom\_churn.csv(https://www.kaggle.com/spscientist/telecom-data)

c) diabetes.csv (https://www.kaggle.com/uciml/pima-indians-diabetes-database#diabetes.csv)

d) HR\_comma\_sep.csv (https: //www.kaggle.com/ludobenistant/hr-analytics)

e) Cars.csv (https://archive.ics.uci.edu/ml/datasets/Car+Evaluation)

f) bank.csv (<https://archive.ics.uci.edu/ml/datasets/Bank+Marketing>)

3) Ответить на вопросы:

* 1. Что такое решающее дерево? Как по построенному дереву найти прогноз для

объекта?

* 1. Зачем в вершинах нужны предикаты? Какие типы предикатов вы знаете? При-

ведите примеры.

* 1. Почему для любой выборки можно построить решающее дерево, имеющее ну-

левую ошибку на ней?

* 1. Почему не рекомендуется строить небинарные деревья (т.е. имеющие больше

двух потомков у каждой вершины)?

* 1. Как устроен жадный алгоритм построения дерева? Какие у него параметры?
  2. Зачем нужны критерии информативности?
  3. Как задается критерий ошибки классификации? Критерий Джини? Энтропий-

ный критерий? Какой у них смысл?

**Полезное:**

Работа с данными Datasets: <https://mlbootcamp.ru/article/tutorial/>

Категориальные переменные: <https://ru.coursera.org/lecture/supervised-learning/rabota-s-katieghorial-nymi-priznakami-pqU2j>

Обработка пропущенных данных: <https://scikit-learn.org/stable/modules/impute.html#impute>