Лабораторна робота №6.
Ідентифікація користувача за допомогою логістичної регресії

In [1]:
```
import pickle
import numpy as np
import pandas as pd
from tqdm import tqdm_notebook
from scipy.sparse import csr_matrix, hstack
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score
from sklearn.linear_model import LogisticRegression
%matplotlib inline
from matplotlib import pyplot as plt
import seaborn as sns
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Спочатку налаштуємо доступ до даних на google drive (якщо ви відкриваєте блокнот в google colab, а не на PC) шляхом монтування google drive

In [2]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

Перевіримо шлях до папки з матеріалами лабораторної роботи на google drive. Якщо у вас шлях відрізняється то відредагуйте

In [3]:
```
!ls gdrive/'My Drive'/TEACHING/IntroDataScience/intro_to_data_science/Lab_5_6/da
```

```
adult.data.csv  adult_train.csv  telecom_churn.csv  train_sessions.csv
adult_test.csv  site_dic.pkl     test_sessions.csv
```

Перемістимо матеріали лабораторної роботи з google drive на віртуальну машину google colab

In [4]:
```
!cp -a gdrive/'My Drive'/TEACHING/IntroDataScience/intro_to_data_science/Lab_5_6/
!ls
```

```
data                            lab5_decision_trees.ipynb
gdrive                          lab5_decision_trees.pdf
lab0_intro_decision_tree.ipynb  lab6_intruder_detection.ipynb
lab0_intro_decision_tree.pdf    sample_data
```

1. Завантаження і перетворення даних

Дані можна самостійно завантажити за посиланням сторінка (https://inclass.kaggle.com/c/catch-me-if-you-can-intruder-detection-through-webpage-session-tracking2). Однак можна цього не робити, оскільки дані вже завантажені для проведення лабораторної роботи

In [5]:
```
# завантажимо навчальну і тестову вибірки
train_df = pd.read_csv('data/train_sessions.csv',
                       index_col='session_id')
test_df = pd.read_csv('data/test_sessions.csv',
                      index_col='session_id')

# приведемо колонку time1, ..., time10 до часового формату
times = ['time%s' % i for i in range(1, 11)]
train_df[times] = train_df[times].apply(pd.to_datetime)
test_df[times] = test_df[times].apply(pd.to_datetime)

# відсортуємо дані за часом
train_df = train_df.sort_values(by='time1')

# подивимося на заголовок навчальної вибірки
train_df.head()
```

Out[5]:

session_id	site1	time1	site2	time2	site3	time3	site4	time4	site5	time5	site6
21669	56	2013-01-12 08:05:57	55.0	2013-01-12 08:05:57	NaN	NaT	NaN	NaT	NaN	NaT	NaN
54843	56	2013-01-12 08:37:23	55.0	2013-01-12 08:37:23	56.0	2013-01-12 09:07:07	55.0	2013-01-12 09:07:09	NaN	NaT	NaN
77292	946	2013-01-12 08:50:13	946.0	2013-01-12 08:50:14	951.0	2013-01-12 08:50:15	946.0	2013-01-12 08:50:15	946.0	2013-01-12 08:50:16	945.0
114021	945	2013-01-12 08:50:17	948.0	2013-01-12 08:50:17	949.0	2013-01-12 08:50:18	948.0	2013-01-12 08:50:18	945.0	2013-01-12 08:50:18	946.0
146670	947	2013-01-12 08:50:20	950.0	2013-01-12 08:50:20	948.0	2013-01-12 08:50:20	947.0	2013-01-12 08:50:21	950.0	2013-01-12 08:50:21	952.0

В навчальній вибірці містяться наступні ознаки:

- site1 – індекс першого відвідування сайту в сесії
- time1 – час відвідування першого сайту в сесії
- ...
- site10 – індекс 10-го відвідування сайту в сесії
- time10 – час відвідування 10-го сайту в сесії
- target – цільова змінна, 1 для сесій Еліс, 0 для сесій інших користувачів

Сесії користувачів виділені таким чимном, щоб вони не можут бути довші півгодини чи 10 сайтів. Тобто сесія вважається закінченою або коли користувач відвідав 10 сайтів підряд або коли сесія зайняла за часом більше 30 хвилин.

В таблиці зустрічаються пропущені значення, це значить, що сесія містить менше, ніж 10 сайтів. Замінимо пропущені значення нулями і приведемо ознаки до цільового типу. Також заванатажимо словник сайтів і подивимося, як він виглядає:

In [6]:
```
# приведемо колонки site1, ..., site10 до цілочислового формату і замінимо пропу
sites = ['site%s' % i for i in range(1, 11)]
train_df[sites] = train_df[sites].fillna(0).astype('int')
test_df[sites] = test_df[sites].fillna(0).astype('int')

# завантажимо словник сайтів
with open(r"data/site_dic.pkl", "rb") as input_file:
    site_dict = pickle.load(input_file)

# датафрейм словника сайтів
sites_dict_df = pd.DataFrame(list(site_dict.keys()),
                             index=list(site_dict.values()),
                             columns=['site'])
print(u'всього сайтів:', sites_dict_df.shape[0])
sites_dict_df.head()
```

```
всього сайтів: 48371
```

Out[6]:

	site
25075	www.abmecatronique.com
13997	groups.live.com
42436	majeureliguefootball.wordpress.com
30911	cdt46.media.tourinsoft.eu
8104	www.hdwallpapers.eu

Виділимо цільову змінну і об'єднаємо вибірки, щоб разом привести їх до розрідженого формату.

In [7]:
```
# наша цільова змінна
y_train = train_df['target']

# об'єднана таблиця вхідних даних
full_df = pd.concat([train_df.drop('target', axis=1), test_df])

# індекс, за яким будемо відокремлювати навчальну вибірку від тестової
idx_split = train_df.shape[0]
```

Для самої першої моделі ми використовуємо лише відвідувані сайти в сесіях (але не будемо звертати увагу на часові ознаки). В основі такого вибору даних для моделей лежить така ідея: * у Еліс є свої улюблені сайти, і якщо ви ще побачите ці сайти в сесіях, тим вище ймовірність, що це сесія Еліс і навпаки. *

Підготуємо дані, з усієї таблиці виберемо лише ознаки `site1, site2, ..., site10` . Нагадуємо, що пропущені значення замінені нулем. Ось як виглядатимуть перші рядки таблиць:

In [8]:
```
# таблиця з індексами відвіданих сайтів в сесії
full_sites = full_df[sites]
full_sites.head()
```

Out[8]:

session_id	site1	site2	site3	site4	site5	site6	site7	site8	site9	site10
21669	56	55	0	0	0	0	0	0	0	0
54843	56	55	56	55	0	0	0	0	0	0
77292	946	946	951	946	946	945	948	784	949	946
114021	945	948	949	948	945	946	947	945	946	946
146670	947	950	948	947	950	952	946	951	946	947

Сессии представляют собой последовательность индексов сайтов и данные в таком виде неудобны для линейных методов. В соответствии с нашей гипотезой (у Элис есть излюбленные сайты) надо преобразовать эту таблицу таким образом, чтобы каждому возможному сайту соответствовал свой отдельный признак (колонка), а его значение равнялось бы количеству посещений этого сайта в сессии. Это делается в две строчки:

Сесії представляють собою послідовність індексів сайтів і дані в такому вигляді невдалі для лінійних методів. Відповідно до нашої гіпотези (у Еліс є улюблені сайти) необхідно перетворити цю таблицю таким чином, щоб кожен можливий веб-сайт відповідав своєму окремому призначенню (колонка), а його значення зростало за кількістю відвідувачів цього веб-сайту в сесіях. Це робиться в два рядки:

In [9]:
```
from scipy.sparse import csr_matrix
```

In [10]:
```
csr_matrix?
```

In [11]:
```
# послідовність з індексами
sites_flatten = full_sites.values.flatten()

# шкана матриця
full_sites_sparse = csr_matrix(([1] * sites_flatten.shape[0],
                                sites_flatten,
                                range(0, sites_flatten.shape[0] + 10, 10)))[:, 1
```

Ще один плюс використання розріджених матриць у тому, що для них є спеціальні реалізації як матричних операцій, так і алгоритми машинного навчання, що дозволяє істотно прискорити операції за рахунок особливостей структур даних. Це стосується і логістичної регресії. Ось тепер у нас все готове для побудови наших перших моделей.

2. Побудова першої моделі

2. Побудова першої моделі

Отже, у нас є алгоритм та дані для нього, побудуйте нашу першу модель, використовуючи реалізацію [логістичної регресії] (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)) з пакета `sklearn` з параметрами за замовчуванням. Перші 90% даних будемо використовувати для навчання (навчальна вибірка, відсортована за часом), а також 10% для перевірки якості (validation).

** Напишіть просту функцію, яка поверне якість моделей на вкладеній вибірці, і навчіть наш перший класифікатор **.

In [ ]:
```
def get_auc_lr_valid(X, y, C=1.0, ratio = 0.9, seed=17):
    '''
    X, y – вибірка
    ratio – у якому співвідношенні поділити вибірку
    C, seed – коефіцієт регуляризації і random_state
              логістичної регресії
    '''

    # Ваш код тут
```

Подивіться, який отримано ROC AUC на відкладеній вибірці.

In [ ]:
```
# Ваш код тут
```

Будем вважати цю модель нашою першою відправною точкою (базовий рівень). Для побудови моделей для прогнозування на тестовій вибірці ** необхідно навчити модель заново вже на всій навчальній вибірці ** (покищо наша модель навчилася лише на частині даних), що підвищує її узагальнюючу здатність:

In [ ]:
```
# функція для запису прогнозів в файлі
def write_to_submission_file(predicted_labels, out_file,
                             target='target', index_label="session_id"):
    predicted_df = pd.DataFrame(predicted_labels,
                                index = np.arange(1, predicted_labels.shape[0] +
                                columns=[target])
    predicted_df.to_csv(out_file, index_label=index_label)
```

Навчіть модель на всій вибірці, зробіть прогноз для тестової вибірки і покажіть результат.

In [ ]:
```
# Ваш код тут
```

3. Покращення моделі, побудова нових ознак

Створіть таку ознаку, яка буде представлят собою число формату ГГГГММ від тої дати, коли відбувалась сесія, наприклад 201407 -- 2014 рік і 7 месяц. Таким чином, ми будемо

враховувати помісячний линейный тренд (http://people.duke.edu/~rnau/411trend.htm) за весь період наданих даних.

In [ ]:
```
# Ваш код тут
```

Додайте нову ознаку, попередньо нормалізуючи її за допомогою `StandardScaler`, і знову підрахуйте ROC AUC на відкладеній вибірці.

In [ ]:
```
# Ваш код тут
```

** Додайте дві нові ознаки: start_hour і morning. **

Ознака `start_hour` - це час у якому почалася сесія (від 0 до 23), а бінарна оознака `morning` рівна 1, якщо сесія почалася вранці і 0, якщо сесія почалася пізніше (будемо вважати, що це ранок, якщо `start_hour` рівний 11 або менше).

** Підрахуйте RUC AUC на відкладеній вибірці для вибірки з: **

- сайтами, `start_month` і `start_hour`
- сайтами, `start_month` і `morning`
- сайтами, `start_month`, `start_hour` і `morning`

In [ ]:
```
# Ваш код тут
```

4. Підбір коефіцієнта регуляризації

Отже, ми ввели ознаки, які покращують якість нашої моделі у порівнянні з першим бейслайном. Чи можемо ми домогтися більшого значення метрики? Після того, як ми сформували навчальну та тестову вибірки, майже завжди має сенс підібрати оптимальні гіперпараметри - характеристики моделі, які не змінюються під час навчання. Наприклад, ви вивчали вирішальні дерева, глибина дерева це гіперпараметр, а ознака, за якому відбувається розгалуження і її значення - ні. У використовуваної нами логістичної регресії ваги кожної ознаки змінюються і під час навчання знаходяться їх оптимальні значення, а коефіцієнт регуляризації залишається постійним. Це той гіперпараметр, який ми зараз будемо оптимізувати.

Порахуйте якість на відкладеній вибірці з коефіцієнтом регуляризації, який за замовчуванням `C = 1 ':

In [ ]:
```
# Ваш код тут
```

Постараємося побити цей результат за рахунок оптимізації коефіцієнта регуляризації. Візьмемо набір можливих значень C і для кожного з них порахуємо значення метрики на відкладеної вибірці.

Знайдіть `C` з `np.logspace (-3, 1, 10)` , при якому ROC AUC на відкладеної вибірці максимальний.

In [ ]:
```
# Ваш код тут
```

Наконец, обучите модель с найденным оптимальным значением коэффициента регуляризации и с построенными признаками `start_hour` , `start_month` и `morning` . Если вы все сделали правильно и загрузите это решение, то повторите второй бенчмарк соревнования.

Нарешті, навчіть модель зі знайденим оптимальним значенням коефіцієнта регуляризації і з побудованими ознаками `start_hour` , `start_month` і `morning` .

In [ ]:
```
# Ваш код тут
```