Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»
Факультет: «Информационные технологии и прикладная математика»

Реферат
Дисциплина: «Вычислительная системы»
I семестр
Реферат: «Изучение языка программирования Python на примере решения задачи с использованием алгоритмов машинного обучения»

Группа:	М8О-107Б-20
Студент:	Чекменев Вячеслав Алексеевич
Преподаватель:	Найденов Иван Евгеньевич
Оценка:	
Дата:	25.12.2020

Москва, 2020

Содержание

Задача

Нужно с хорошей точностью определить пользователей курса, которые точно уйдут или точно закончат курс.

Подробнее про курс: В рамках данного курса подробно рассматриваются все основные этапы анализа данных при помощи R. В данном курсе рассматриваются как стандартные методы R и Rstudio, так и специальные пакеты и библиотеки. На курсе применяются основные методы статистического анализа: t-тест, корреляция, регрессия, дисперсионный и регрессионный анализ и др. Особое внимание в курсе уделяется визуализации получаемых результатов.

Введение

Итак, **Машинное обучение** — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, математического анализа, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме

Выбор утилит для написания программы.

Для написания нашей программы мы использовали интерактивную среду разработки Jupyter notebook.

Использованные библиотеки.

import pandas as pd
 - для предобработки данных
import seaborn as sns
 - для построения графиков
import numpy as np
 - для работы с массивами и математическими функциями
%matplotlib inline
 - для отображения графиков в Jupyter notebook

Использованные классы.

from sklearn import tree
 - для создания классификатора и дерева решений
from sklearn.model_selection import train_test_split
 - разделение данных на тестовое и тренировочное множества
from sklearn.model_selection import GridSearchCV
 - для поиска лучших параметров для дерева решений
from sklearn.metrics import precision_score, recall_score
 - для вывода значений точности и полноты модели
from IPython.display import SVG
from graphviz import Source
from IPython.display import display
 - для вывода дерева в формате SVG Jupyter notebook
from IPython.display import HTML
style =
"<style>svg{width:100% !important;height:100% !important;}</style>"
HTML(style)

Сценарий работы программы

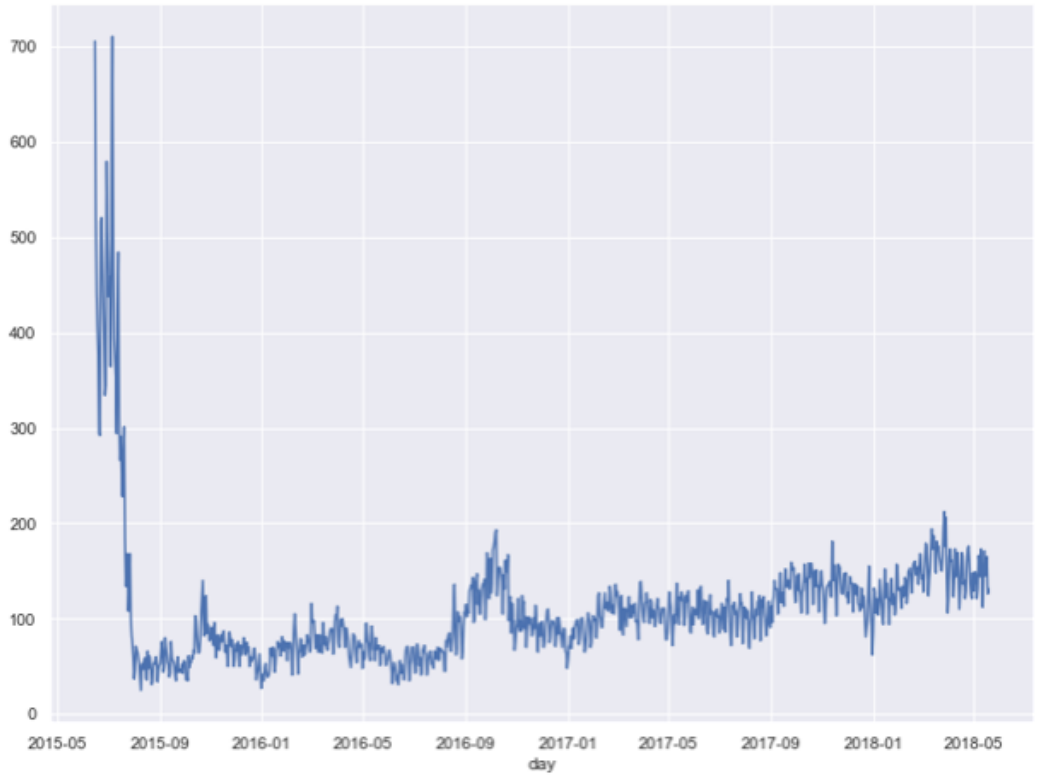
1. Перевод времени из Unix date в нормальный вид.

```
In [10]: events_data['date'] = pd.to_datetime(events_data.timestamp, unit='s')
```

2. Предварительная проверка данных на корректность при. помощи построения графиков: зависимости кол-ва прибывших пользователей от даты, зависимости кол-ва действий от даты.

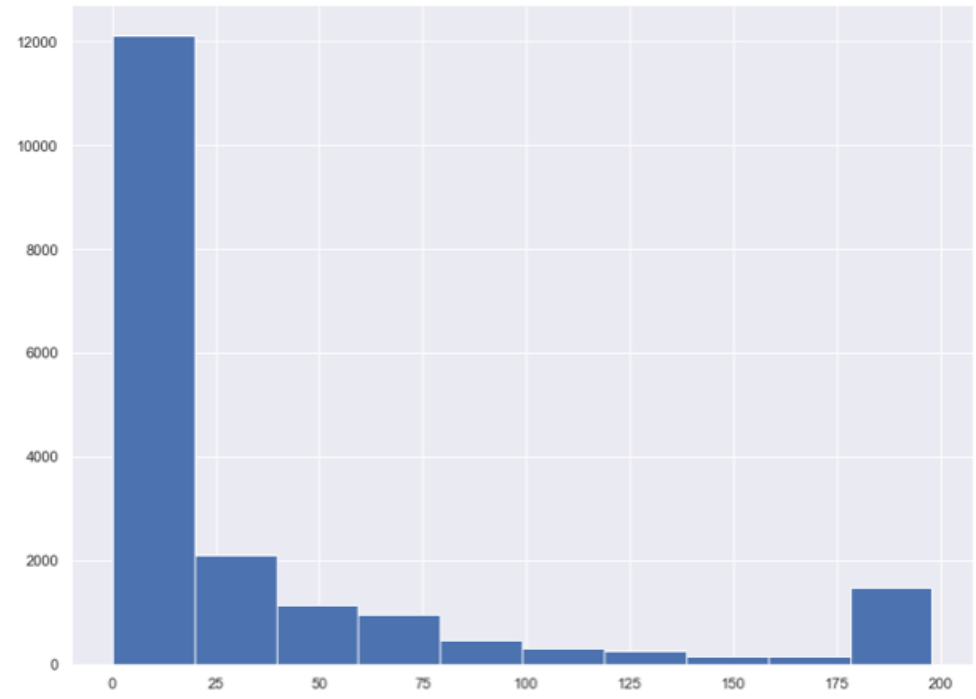
In [124]:
```
events_data.groupby('day') \
    .user_id.nunique().plot() # plot(1) of new students arrival by date
```

Out[124]: <matplotlib.axes._subplots.AxesSubplot at 0x21704970520>



In [125]:
```
events_data.pivot_table(index='user_id',
                        columns='action',
                        values='step_id',
                        aggfunc='count',
                        fill_value=0).reset_index().discovered.hist() # plot:
                                                                      # discover
```

Out[125]: <matplotlib.axes._subplots.AxesSubplot at 0x21704df9880>



3. Будем считать человека ушедшим с курса, если он не совершал действий 15 дней.
4. Создаем датафрейм users_data (последний заход на курс, ушел ли человек с курса), users_scores (Кол-во удачных попыток и неудачных). Объединяем эти датафреймы в один (users_data).

```
In [29]: users_data = events_data.groupby('user_id', as_index=False) \
             .agg({'timestamp': 'max'}).rename(columns = {'timestamp': 'last_timestamp'})
```

```
In [32]: users_data['is_gone_user'] = (now - users_data.last_timestamp) > drop_out_threshold
```

```
In [33]: users_data.head()
```

Out[33]:

	user_id	last_timestamp	is_gone_user
0	1	1472827464	True
1	2	1519226966	True

```
In [34]: users_scores.head()
```

Out[34]:

submission_status	user_id	correct	wrong
0	2	2	0
1	3	29	23
2	5	2	2
3	8	9	21
4	14	0	1

```
In [35]: users_data = users_data.merge(users_scores, on='user_id', how='outer')
```

```
In [37]: users_data.head()
```

Out[37]:

	user_id	last_timestamp	is_gone_user	correct	wrong
0	1	1472827464	True	0.0	0.0
1	2	1519226966	True	2.0	0.0
2	3	1444581588	True	29.0	23.0
3	5	1499859939	True	2.0	2.0
4	7	1521634660	True	0.0	0.0

5. Присоединяем к датафрейму users_data датафрейм с кол-вом уникальных дней проведенных на курсе по общему столбцу user_id.

```
In [42]: users_data = users_data.merge(users_days, how='outer') # jointing users_days to users_data
```

```
In [43]: users_data.head()
```

Out[43]:

	user_id	last_timestamp	is_gone_user	correct	wrong	discovered	passed	started_attempt	viewed	day
0	1	1472827464	True	0.0	0.0	1	0	0	1	1
1	2	1519226966	True	2.0	0.0	9	9	2	10	2
2	3	1444581588	True	29.0	23.0	91	87	30	192	7
3	5	1499859939	True	2.0	2.0	11	11	4	12	2
4	7	1521634660	True	0.0	0.0	1	1	0	1	1

6. Определяем условие прохождения курса (необходимо набрать больше 170 баллов).

```
In [46]: users_data['passed_course'] = users_data.passed > 170 # users, who passed the course
                                                              # passed - True, drop - False
```

7. Присоединяем к датафрейму users_data датафрейм с временем первого посещения курса для каждого пользователя по общему столбцу user_id.

```
In [55]: users_data = users_data.merge(user_min_time, how='outer') # merge users_data and user_min_time datafra
```

```
In [56]: users_data.head()
```

Out[56]:

	user_id	last_timestamp	is_gone_user	correct	wrong	discovered	passed	started_attempt	viewed	day	passed_course	n
0	1	1472827464	True	0.0	0.0	1	0	0	1	1	False	
1	2	1519226966	True	2.0	0.0	9	9	2	10	2	False	
2	3	1444581588	True	29.0	23.0	91	87	30	192	7	False	
3	5	1499859939	True	2.0	2.0	11	11	4	12	2	False	
4	7	1521634660	True	0.0	0.0	1	1	0	1	1	False	

8. Будем предсказывать уход человека с курса по первым 15 дням пребывания.
9. Найдем пороговое значение времени для каждого пользователя.

```
In [65]: user_learning_time_threshold = user_min_time.user_id.map(str) + '_' + (user_min_time.min_timestamp + learning_time_threshold)
```

10. Отберем действия, которые совершили пользователи до своих пороговых значений времени.

```
In [69]: events_data_train = events_data[events_data.user_time <= events_data.user_learning_time_threshold]
```

11. Составив датафрейм для последующего создания модели is_passed_df включающий в себя: кол-во уникальных дней проведенных за определенный ранее промежуток времени (15 дней), кол-во пройденых степов, кол-во удачных и неудачных попыток и точность выполнения заданий.

```
In [76]: X = submissions_data_train.groupby('user_id').day.nunique().to_frame().reset_index() \
             .rename(columns={'day': 'days'})# creating X dataframe. column 'day': how many unique days did the
```

```
In [77]: X.head(3)
```

Out[77]:

	user_id	days
0	2	1
1	3	1
2	8	1

```
In [78]: steps_tried = submissions_data_train.groupby('user_id').step_id.nunique().to_frame().reset_index() \
             .rename(columns={'step_id': 'steps_tried'}) # creating X dataframe. column 'steps_tried': how many
```

```
In [79]: steps_tried.head()
```

Out[79]:

	user_id	steps_tried
0	2	2
1	3	4
2	8	11
3	14	1
4	16	53

```
In [80]: X = X.merge(steps_tried, on='user_id', how='outer')
```

In [81]:
```
X.head()
```

Out[81]:

	user_id	days	steps_tried
0	2	1	2
1	3	1	4
2	8	1	11
3	14	1	1
4	16	10	53

In [82]:
```
X = X.merge(submissions_data_train.pivot_table(index='user_id',
                                               columns='submission_status',
                                               values='step_id',
                                               aggfunc='count',
                                               fill_value=0).reset_index())
```

In [83]:
```
X['correct_ratio'] = X.correct / (X.correct + X.wrong)
```

In [84]:
```
X.head()
```

Out[84]:

	user_id	days	steps_tried	correct	wrong	correct_ratio
0	2	1	2	2	0	1.000000
1	3	1	4	4	4	0.500000
2	8	1	11	9	21	0.300000
3	14	1	1	0	1	0.000000
4	16	10	53	52	69	0.429752

12. Присоединяем к датафрейму is_passed_df колонку passed_course (значения 0 1).

In [95]:
```
is_passed_df = X.merge(users_data[['user_id', 'passed_course']])
```

In [96]:
```
is_passed_df
```

Out[96]:

	user_id	days	steps_tried	correct	wrong	correct_ratio	viewed	passed_course
0	2	1.0	2.0	2.0	0.0	1.000000	9	False
1	3	1.0	4.0	4.0	4.0	0.500000	20	False
2	8	1.0	11.0	9.0	21.0	0.300000	156	False
3	14	1.0	1.0	0.0	1.0	0.000000	9	False
4	16	10.0	53.0	52.0	69.0	0.429752	288	True
...	...	...	...	...	...	...	...	...
17980	26774	0.0	0.0	0.0	0.0	0.000000	1	False
17981	26781	0.0	0.0	0.0	0.0	0.000000	6	True
17982	26788	0.0	0.0	0.0	0.0	0.000000	1	False
17983	26789	0.0	0.0	0.0	0.0	0.000000	2	False
17984	26793	0.0	0.0	0.0	0.0	0.000000	1	False

17985 rows × 8 columns

13. Составим серию со значениями целевой переменной и датафрейма с фичами.

```
In [99]: X = is_passed_df.drop(['passed_course'], axis=1)
         y = is_passed_df.passed_course
```

14. Найдем нужные параметры для дерева решений при помощи GridSearchCV.

```
In [104]: search_clf = GridSearchCV(clf, parameters, cv=5)

In [105]: search_clf.fit(X_train, y_train)

Out[105]: GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
                       param_grid={'criterion': ['gini', 'entropy'],
                                   'max_depth': range(1, 100)})
```

15. Найдем дерево решений по наилучшим параметрам.

```
In [107]: search_clf.best_params_

Out[107]: {'criterion': 'entropy', 'max_depth': 4}

In [108]: best_clf = search_clf.best_estimator_
```

16. Найдем Результат предсказания на тестовой выборке.

```
In [111]: best_clf.score(X_test, y_test)

Out[111]: 0.9315098954858795
```

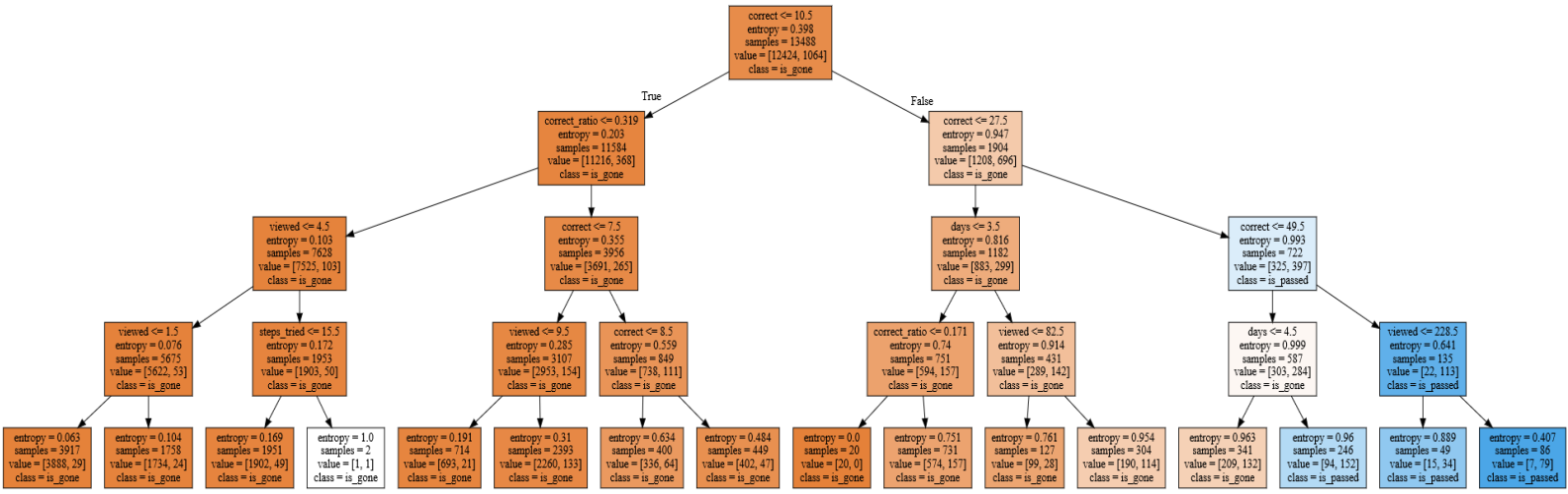
17. Найдем точность (precision) нашего предсказания

```
In [119]: precision_score(y_test, y_pred)

Out[119]: 0.6802721088435374
```

18. Дерево решений

(https://raw.githubusercontent.com/Suraba03/report_sem_1/66d0e268a0d9019b45202a54bf3815f96785164e/tree.svg)



Заключение

Для реализации данной задачи нам нужно было познакомиться с языком пайтон, так как данный язык хорошо подходит для машинного обучения и в целом для Data Science.

Для реализации и понимания алгоритмов машинного обучения мы ознакомились с основами статистического анализа. Все нужные знания мы получали из МООС курсов на платформе stepik.

Список и информация о курсах:

- *Основы статистики часть 1* - В рамках курса рассматриваются подходы к описанию получаемых в исследованиях данных, основные методы и принципы статистического анализа, интерпретация и визуализация получаемых результатов.
  Рассматриваются такие методы дисперсионный, регрессионный и кластерный анализ. Ставятся задачи сравнивения групп между собой, рассчета коэффициентов корреляции и построения регрессионных уравнений.
  Основной акцент на курсе был сделан на математических идеях, интуиции и логике, которые обуславливают методы и расчетные формулы.
  Изученный материал применим для решения задач на машинное обучение и не только.
- *Основы статистики часть 2* - В данном курсе уже рассмотрены методы, которые используются в анализе данных и наиболее часто применяются при статистической обработке результатов в широчайшем круге научных и прикладных областей. Помимо

теоретических заданий были решены практические задачи, которые помогли для более глубокого понимания темы. Знаний, полученных в результате прохождения данного курса достаточно, чтобы научиться более быстро и эффективно решать различные задачи, связанные с анализом данных.

- *Курс по пайтону* - В курсе по python было много практических задач на самые базовые понятия о языке, такие как: операторы, переменные, типы данных, условия, циклы, строки, списки, функции словари, интерпретатор, файлы, модули.
- *Курс по Data Science* - Курс является ознакомительным относительно основ машинного обучения. Мы подробно разобрали основные теоретические понятия, а также начали знакомство с библиотеками Pandas и Scikit-learn — наиболее популярными инструменатами для анализа данных и машинного обучения, используя язык программирования Python.

В итоге данная задача оказалась очень интересной и охватила большое кол-во новых для нас тем. Эта работа дала нам хороший старт в изучении машинного обучения и анализа данных. Мы продолжим изучать машинное обучение и дальше, чтобы охватить более узкие темы, которые пригодятся нам для тюнинга нынешней модели.

Список используемых источников

1. Основы статистики - https://stepik.org/course/76
2. Основы статистики 2 - https://stepik.org/course/524
3. Программирование на Python - https://stepik.org/course/67
4. Введение в Data Science и машинное обучение - https://stepik.org/course/4852/syllabus
5. Брали от туда ответы на наши вопросы - https://stackoverflow.com
6. Документация по pandas – https://pandas.pydata.org/pandas-docs/stable/referenc..
7. Документация по scikit-learn - https://scikit-learn.org/stable/ljrevtynfw