Отчёт о задании "Petrov & Boshirov Challenge" по предмету Data Science

Ссылка на код – https://github.com/amsavchenko/ds-mephi/tree/master/midterm-exam

Выполнили студенты группы Б17-505:

- Зоричев Виталий
- Иванова Дарья
- Казьмин Сергей
- Савченко Антон

---

1. Постановка задачи

Задача заключалась в приобретении навыков работы с большим объемом данных из разнородных источников, осуществлении очистки, предобработки и слияния данных. Данные надо было проанализировать и найти потенциальных "русских шпионов".

Предоставленные данные:

```
├── BoardingData.csv
├── FrequentFlyerForum-Profiles.json
├── PointzAggregator-AirlinesData.xml
├── Sirena-export-fixed.tab
├── Skyteam_Timetable.pdf
├── SkyTeam-Exchange.yaml
├── YourBoardingPassDotAero/
```

2. Технические особенности решения

Для начала был осуществлен просмотр всех предоставленных файлов для оценки их полезности при анализе и выделения списка полей, которые есть в каждом файле или которые можно однозначно получить из значений других полей. Был определен список признаков, которые надо вытащить из каждого файла:

- Имя пассажира
- Пол пассажира
- Номер рейса
- Дата полёта
- Время вылета
- Название авиакомпании
- Страна, город, аэропорт вылета и прилета
- Номер использованной карты лояльности

- Дата рождения и номер паспорта пассажира

Далее будет рассказано про парсинг файлов каждого типа.

2.1. Excel

Всего было 365 файлов с расширением `.xlsx`, внутри каждого примерно 1500 листов, на каждом из которых отдельный посадочный талон. Для загрузки данных использовался метод `pandas.read_excel()`, позволяющий считать весь файл (со всеми листами) в `pandas.DataFrame`. В файлах не была указана авиакомпания, выполнившая рейс, поэтому эта информация доставалась из номера рейса (первые 2 буквы определяют авиакомпанию). Так же не были указаны страны вылета и прилёта, эта информация добавлялась по коду аэропортов. Главная сложность при работе с данными Excel-файлами заключалась в том, что полное имя пассажира было записано в 1 строку и порядок следования имени и фамилии менялся. Усугубляло ситуацию то, что в некоторых посадочных талонах было добавлена еще и первая буква отчества. Борьба с этой проблемой проводилась в 2 шага:

1. удаление отчества – сначала строка разбивалась на отдельные слова, далее слова сортировались по длине, и если слов 3, то гарантированно кратчайшее слово - это первая буква отчества, её можно удалить.

```
def process_excel_name(name):
  name = sorted(name.split(' '), key=lambda x: len(x))
  if len(name) == 3:
    name.pop(0)
   return ' '.join(name)
```

2. разделение имени и фамилии – для этого помогли данные, подготовленные другими участниками. Собрал уникальный набор имён и фамилий, встртившихся в других файлах, и по нему разделил свои строки.

2.2 YAML

```
AZ7209:
  FF:
    SU 272025028: {CLASS: J, FARE: JSTNDU}
  FROM: SVO
  STATUS: LANDED
  TO: NAP
CZ4301:
  FF:
    FB 734066353: {CLASS: P, FARE: PSTNWQ}
  FROM: SVO
  STATUS: LANDED
  TO: GVA
CZ4307:
  FF:
    FB 381098538: {CLASS: P, FARE: PGRPVN}
    FB 835216223: {CLASS: Y, FARE: YRSTZK}
    SU 375357343: {CLASS: Y, FARE: YSTNCA}
  FROM: SVO
  STATUS: LANDED
  TO: HEL
```

Из-за того, что файл был слишком объёмным, он был разделен на много файлов – один файл на один день полётов.

Теперь каждый файл можно было представить в виде дерева из вложенных словарей (в этом помог модуль PyYaml). А так как не у всех подсловарей в этом дереве был именованный ключ, то решить проблему парсинга помог обход в глубину.

```
def dfs(node, node_name):
    viewed_nodes.append(node_name)
    if (type(node) == dict):
        for key, value in node.items():
            if ((node_name + ',' + key) not in viewed_nodes):
                dfs(value, node_name + ',' + key)
    else:
        parsing_result.append(node_name + ',' + node)
```

После последующей обработки были получены 2 файла CSV: один, содержащий данные о программе лояльности, другой – с данными о полёте.

2.3. TAB

```
PaxName                          PaxBirthDateDepartDate  DepartTime  ArrivalDate ArrivalTime
ОЗЕРОВ ИЛЬДАР ДАНИИЛОВИЧ         1999-05-15  2017-05-30  00:05       2017-05-30  08:05
КОЛОСОВ САМИР ТАМЕРЛАНОВИЧ       N/A         2017-12-27  02:15       2017-12-27  04:40
ИГНАТОВА СНЕЖАНА КОНСТАНТИНОВНА  N/A         2017-09-19  06:40       2017-09-19  07:45
ЖАРОВ ПЛАТОН АЛЬБЕРТОВИЧ         1999-05-02  2017-03-18  22:10       2017-03-19  01:05
НИКОЛЬСКИЙ НИКОЛАЙ ИГОРЕВИЧ      1990-12-26  2017-03-18  22:10       2017-03-19  01:05
ГЛУШКОВ КОНСТАНТИН ИЛЬИЧ         N/A         2017-03-12  11:45       2017-03-12  12:25
КАПУСТИН АРТЁМ ЭДУАРДОВИЧ        1982-10-24  2017-03-12  11:45       2017-03-12  12:25
```

Так как каждое поле в этом файле представлялось фиксированным количеством символов – было решено вручную выделить эти поля из файла, после чего слить в csv.

```
buf = buf.split("\n")
new_buf = []
for line in buf:
    new_buf.append([line[0:60],
                    line[60:72],
                    line[72:84],
                    line[84:96],
                    line[96:108],
                    line[108:120],
                    line[120:132],
                    line[132:138],
                    line[138:144],
                    line[144:150],
                    line[150:168],
                    line[168:180],
                    line[180:186],
                    line[186:192],
                    line[192:198],
                    line[198:204],
                    line[204:216],
                    line[216:240],
                    line[240:276],
                    line[276:336]])
buf = new_buf.copy()
```

После чего данные были загружены в pandas.DataFrame , где были обработаны в соответствии с договорённостями команды о нормальном обобщённом виде данных. Для перевода имён пассажиров в транслит был использован модуль transliterate. После обработки дали были выгружены в отдельный csv-файл.

2.4. CSV

	FirstName	SecondName	LastName	Gender	BirthDate	Document	BookingCode	TicketNumber	Baggage	Date	Time	FlightNumber	Code
0	SAVELII	VIKTOROVICH	RUSANOV	Male	03/10/1983	2879 096860	FRNINO	6625956945991971	Transit	2017-03-22	06:05	SU1369	
1	LEV	MARKOVICH	ISAEV	Male	12/13/1975	1788 173211	Not presented	1643715499224676	Registered	2017-03-18	22:10	SU1180	
2	NIKOLAI	I.	NIKOLSKII	Male	12/26/1990	4396 926588	VWNYGF	6247422701565929	Transit	2017-03-18	22:10	SU1180	
3	ANATOLII	PETROVICH	SHILOV	Male	05/24/1997	2595 919752	WQFFUE	Not presented	Registered	2017-03-18	22:10	SU1180	
4	MIROSLAVA	VIACHESLAVOVNA	SEMENOVA	Female	01/31/1976	6775 516990	Not presented	Not presented	Registered	2017-03-12	11:45	SU6284	

В данном случае необходимо было лишь привести данные к общему виду, после чего сохранить в новый файл csv.

2.5. XML

Для загрузки данных выбран язык программирования python.

Файл выглядел следующим образом:

К сожалению, в python нет метода, который автоматически сделает из файла в xml формате готовый DataFrame. Поэтому можно было либо написать функцию обхода дерева, либо воспользоваться модулем pandas_read_xml, который загружает данные из .xml, а далее воспользоваться функцией auto_separate_tables для создания DataFrame. Был выбран второй способ.

Из файла получилось извлечь следующие признаки:

- Имя пассажира
- Номер рейса
- Дата полёта
- Название авиакомпании
- Аэропорт вылета и прилета
- Номер использованной карты лояльности

По аэропорту были определены страна и город вылета.

Были решены проблемы разнородности данных: имена и фамилии переведены в нижний регистр, а коды аэропортов в верхний.

Готовый DataFrame:

	FirstName	LastName	BonusProgramm	BonusProgrammNumber	Gender	FlightNumber	Date	Time	DepartureCountry	DepartureCity	DepartureAirp
0	iaromir	zverev	Flying Blue	FB 171388778	NaN	KE827	2017-08-06	NaN	French Polynesia	Reao	RI
1	iaromir	zverev	Flying Blue	FB 171388778	NaN	MU9706	2017-10-26	NaN	China	Shunyi	PI
2	vitalina	korovina	Korean Air SKYPASS	KE 696768759	NaN	DL5058	2017-09-11	NaN	United States	Chattanooga	CI
3	vitalina	korovina	Korean Air SKYPASS	KE 696768759	NaN	KE1	2017-04-01	NaN	French Polynesia	Reao	RI
4	vitalina	korovina	Korean Air SKYPASS	KE 696768759	NaN	DL837	2017-09-13	NaN	United States	Atlanta	A
...	...	...	...	...	...	...	...	...	...	...	
446175	emil	kazakov	Delta SkyMiles	DT 269347755	NaN	SU6402	2017-03-01	NaN	Russia	Yekaterinburg	S'
446176	emil	kazakov	Aeroflot Bonus	SU 856232932	NaN	AR1142	2017-12-14	NaN	Argentina	Ezeiza	E
446177	emil	kazakov	Aeroflot Bonus	SU 856232932	NaN	AR1143	2017-09-16	NaN	Italy	Rome	FC
446178	ul'yana	kononova	Delta SkyMiles	DT 302785701	NaN	GA502	2017-05-12	NaN	Indonesia	Tangerang	C(
446179	ul'yana	kononova	Delta SkyMiles	DT 302785701	NaN	GA88	2017-08-02	NaN	Indonesia	Tangerang	C(

436099 rows × 17 columns

2.6. JSON

Файл `FrequentFlyerForum-Profiles.json` – профили часто летающих пассажиров на форуме. Для парсинга использовался язык программирования JavaScript, его библиотека `fs`, а также метод `JSON.parse()`.

Файл содержал все необходимые данные, указанные ранее, за исключением имен пассажиров и времени вылета. Часть имен были скрыты - добавлялись из других таблиц по номеру карты лояльности. Время вылета добавлялось из других таблиц по номеру рейса.

Итог – csv-файл `FrequentFlyerForum-Profiles.csv` с указанными ранее столбцами.

2.7. Заполнение пропущенных значений

Не в каждом файле были все нужные поля, часто некоторые значения надо было заполнять по значениям других полей. Для этого было создано 2 таблицы:

1. по номеру рейса определяющая код аэропортов вылета и прилета

Number	Departure	Arrival
AF10	CDG	JFK
AF1000	CDG	MAD
AF1001	MAD	CDG
AF1004	CDG	FCO
AF1005	FCO	FCO
...	...	...
VN931	LPQ	REP
VN942	RGN	SGN
VN943	SGN	RGN
VN956	RGN	HAN
VN957	HAN	RGN

24005 rows × 2 columns

2. по коду аэропорта определяющая страну и город, в котором аэропорт расположен

code	city	country
AAE	El Tarf	Algeria
AAL	Norresundby	Denmark
AAM	Mala Mala	South Africa
AAN	Ayn al Faydah	United Arab Emirates
AAQ	Novorossiysk	Russia
...	...	...
YZY	Zhangye	China
YUS	Yushu	China
BPE	Bagan	Burma
THD	Sao Vang	Vietnam
AOG	Anshan	China

3910 rows × 2 columns

2.8. Решение проблем с транслитерацией

В разных таблицах были разные варианты транслитерации, поэтому было необходимо привести все имена в единый формат. Для этого сначала из строк были удалены апострофы, обозначающие мягкий знак, далее произведены замены буквенные сочетаний. Например,

- x => ks
- ia, ja => ya
- iu, ju => yu
- и т.д.

2.9. Объединение 6 таблиц в одну

На предыдущем этапе были получены 6 таблиц с определенными выше полями, и стало необходимо объединить их в одну. При этом полёты в таблицах пересекались, и было необходимо оставить только уникальные полеты. Для этого создавался составной индекс:

```
old_df = old_df.set_index(['Name', 'FlightNumber', 'Date']).sort_index()
old_df.head(3)
```

			FirstName	LastName	BonusProgramm	BonusProgrammNumber	Gender	Time	DepartureCountry	DepartureCity	DepartureAirpor
Name	FlightNumber	Date									
adel ageev	AF1343	2017-03-17	adel	ageev	Aeroflot Bonus	SU 930869690	M	18:15	Switzerland	Geneva	GV
	AM451	2017-10-18	adel	ageev	Korean Air SKYPASS	KE 989004326	M	10:15	Mexico	Mexico City	ME
	CZ6360	2017-11-19	adel	ageev	Flying Blue	FB 562855979	M	17:00	China	Zhengzhou	CG

```
new_df = new_df.set_index(['Name', 'FlightNumber', 'Date']).sort_index()
new_df.head(3)
```

			AirlineName	ArrivalAirport	ArrivalCity	ArrivalCountry	BirthDate	BonusProgrammNumber	DepartureAirport	DepartureCity	Depa
Name	FlightNumber	Date									
adel afanasev	CZ3383	2017-03-11	china southern airlines	NNY	nanyang	china	NaN	NaN	CAN	guangzhou	
	CZ6478	2017-07-21	china southern airlines	CGO	zhengzhou	china	1976-01-16	NaN	SZX	shenzhen	
	SU1457	2017-01-21	aeroflot	SVO	zelenograd	russia	1976-01-16	NaN	KEJ	kemerovo	

Вычислялось пересечение индексов и из одной таблицы удалялись все индексы из пересечения:

```
repeat_index = old_df.index & new_df.index
print('Количество пересечений: ', len(repeat_index))
print('Пример: ', repeat_index[0])
```

```
Количество пересечений:  436097
Пример:  ('valeriy kuzin', 'SU1437', '2017-01-08')
```

```
new_df = new_df.drop(repeat_index, axis=0)
```

```
new_df = pd.concat((old_df, new_df)).sort_index()
new_df.head(3)
```

			AirlineName	ArrivalAirport	ArrivalCity	ArrivalCountry	BirthDate	BonusProgramm	BonusProgrammNumber	DepartureAirport	D
Name	FlightNumber	Date									
	CZ3383	2017-03-11	china southern airlines	NNY	nanyang	china	NaN	NaN	NaN	CAN	
adel afanasev	CZ6478	2017-07-21	china southern airlines	CGO	zhengzhou	china	1976-01-16	NaN	NaN	SZX	
	SU1457	2017-01-21	aeroflot	SVO	zelenograd	russia	1976-01-16	NaN	NaN	KEJ	

Итого после объединения 6 таблиц в таблице получилось 1435495 строк.

2.10. Выделение признаков для пассажиров

Теперь возникла необходимость перейти от таблицы с полётами к таблице пассажиров, которая бы могла в дальнейшем использоваться в качестве выборки для применения алгоитмов машинного обучения. Была создана таблица из 74477 пассажиров, для каждого из которых вычислены следующие признаки:

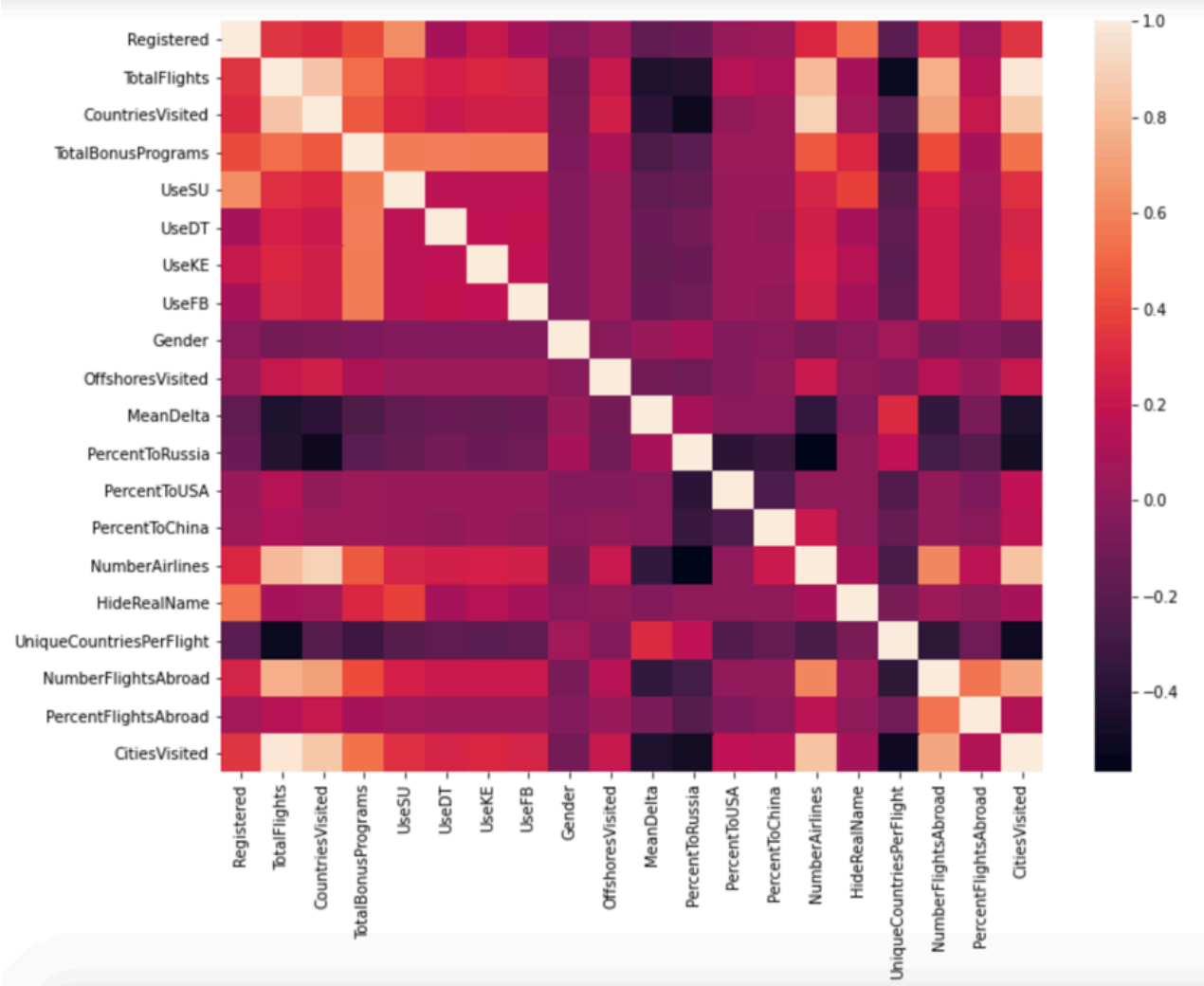
- Общее число полетов
- Число посещенных стран и городов
- Число использованных карт лояльности
- Средний временной интервал между полётами
- Число использованных авиакомпаний к числу полётов
- Использование карты лояльности Аэрофлота
- Количество полётов заграницу
- Зарегистрирован ли человек на форуме часто летающих пассажиров
- Скрыл ли человек своё имя на форуме
- Количество посещенных оффшорных стран
- и т. д.

Полученный датасет:

	Name	Registered	TotalFlights	CountriesVisited	TotalBonusPrograms	UseSU	UseDT	UseKE	UseFB	Gender	...
0	adel afanasev	0	3	2	0	0	0	0	0	1	...
1	adel ageev	1	47	13	5	1	1	1	1	1	...
2	adel akimov	1	39	7	4	1	1	1	1	1	...
3	adel aksenov	0	17	5	2	0	1	1	0	1	...
4	adel aleksandrov	0	21	10	0	0	0	0	0	1	...
...	...	...	...	...	...	...	...	...	...	...	...
74472	zlata zotova	0	66	13	3	0	1	1	1	0	...
74473	zlata zubova	0	43	17	2	0	1	1	0	0	...
74474	zlata zvereva	1	14	4	3	1	1	0	1	0	...
74475	zlata zvyagintseva	1	24	3	1	1	0	0	0	0	...
74476	zlata zykova	0	27	5	1	0	0	0	1	0	...

74477 rows × 23 columns

Матрица линейной корреляции признаков:



3. Описание гипотезы

Получив таблицу с признаками, определяющими характер путешествий пассажиров, мы поняли, что близкая задача к решаемой – задача обнаружения аномалий. То есть мы можем искать "потенциальных русских шпионов" как пассажиров с некоторым аномальным поведением. Для решения этой задачи пробовали несколько алгоритмов, остановились на Isolation Forest.

- Ансамбль решающих деревьев
- Каждое строится до тех пор, пока каждый объект не будет в отдельном листе
- На каждой итерации выбирается случайный признак и случайное расщепление по нему
- Для каждого объекта мера его нормальности – среднее арифметическое глубин листьев, в которые он попал

Настраиваемые признаки:

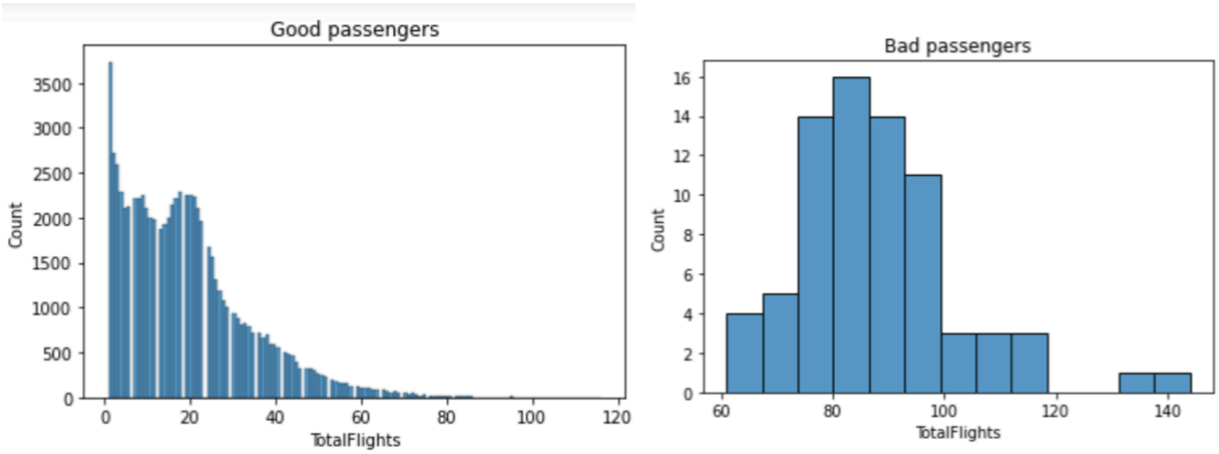
- число деревьев
- объём выборки для построения одного дерева
- число признаков, которые используются для построения одного дерева
- **доля выбросов в выборке** (для выбора порога)

Доля выбросов выбиралась так, чтобы получить 75 аномалий.

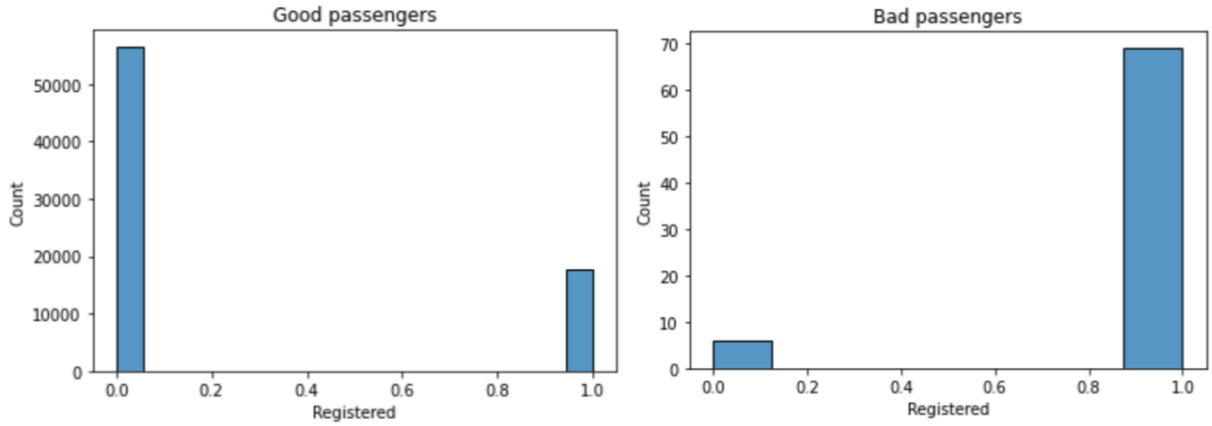
4. Результаты

После применения модели было выделено 75 человек, являющихся аномалиями в данных. Попробуем объяснить, почему они могли действительно быть шпионами.

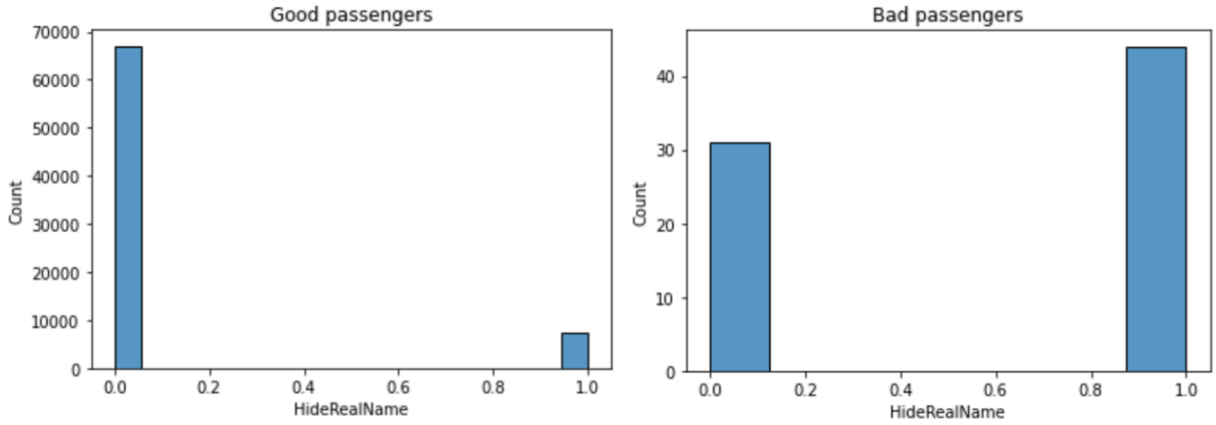
1. Шпион часто летает, так как это часть его работы – наблюдать за другими людьми (в том числе и за их перемещением) а также он не может долгое время находиться в одном месте, чтобы не подавать признаков. Такая особенность наблюдается и на гистограммах:



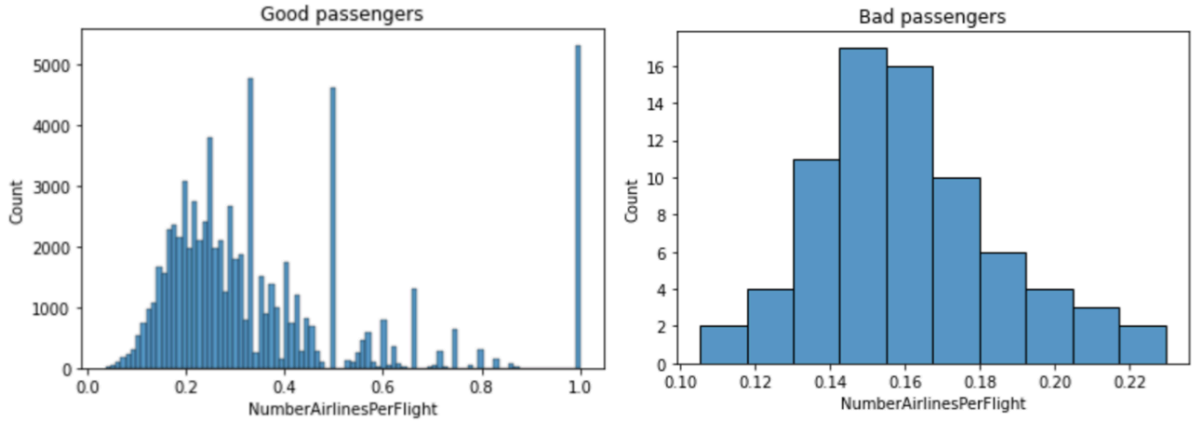
2. Шпион зарегистрирован на форуме часто летающих пассажиров для получения некоторой информации, которая могла бы помочь ему в работе. Такая особенность наблюдается и на гистограммах:



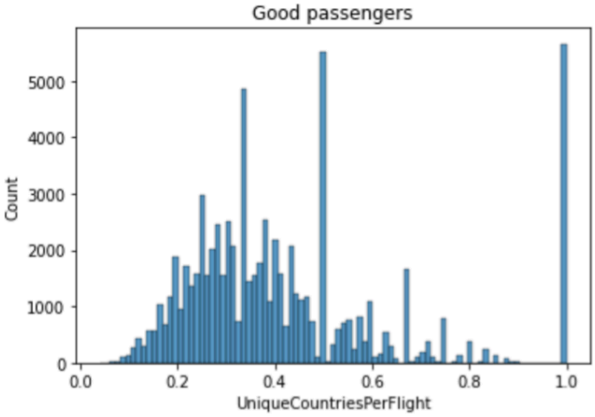
3. Хоть шпион и зарегистрирован на форуме, но в его интересах остаться незамеченным, поэтому чаще всего шпионы скрывают там свое реальное имя. Такая особенность наблюдается и на гистограммах:

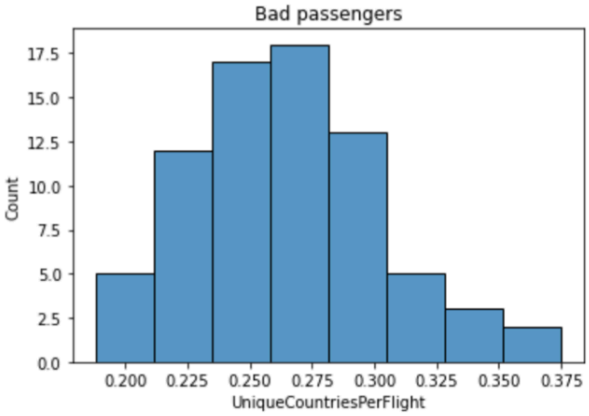


4. Шпион реже других пассажиров меняет авиакомпании, так как использует только надежные и проверенные, а также с целью не «светиться» в других авиакомпания. Такая особенность наблюдается и на гистограммах:



5. Шпион посещает меньшее количество уникальных стран (только если ему не нужно следить за перемещением своего «клиента» по другим странам), так как достаточно опасно отправлять шпиона, не подготовленного к поведению в новой для него стране. Такая особенность наблюдается и на гистограммах:


Good passengers
Count
UniqueCountriesPerFlight
0
1000
2000
3000
4000
5000
0.0
0.2
0.4
0.6
0.8
1.0


Bad passengers
Count
UniqueCountriesPerFlight
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.200
0.225
0.250
0.275
0.300
0.325
0.350
0.375