

Машинное обучение в Power BI с использованием PyCaret

Пошаговое руководство по внедрению машинного обучения в Power BI за считанные минуты

Машинное обучение встречает бизнес-аналитику **PyCaret**

[PyCaret](#) - библиотека машинного обучения с открытым исходным кодом на Python, которая обучает и использует модели машинного обучения в среде с **низким кодом**. В нашем [предыдущем посте](#) мы продемонстрировали, как использовать PyCaret в Jupyter Notebook для обучения и развертывания моделей машинного обучения в Python.

В этом посте мы представляем **пошаговое руководство** по интеграции PyCaret в [Power BI](#), что позволит аналитикам и ученым добавить слой машинного обучения в свои панели мониторинга и отчеты без дополнительных затрат на лицензию или программное обеспечение. PyCaret - это **свободно распространяемая** библиотека Python с открытым исходным кодом, которая поставляется с широким набором функций, специально созданных для работы в Power BI.

К концу этой статьи вы узнаете, как реализовать в Power BI следующее:

- **Кластеризация** - группировка точек данных с похожими характеристиками.
- **Обнаружение аномалий** - выявление редких наблюдений / выбросов в данных.
- **Обработка естественного языка** - анализ текстовых данных с помощью тематического моделирования.
- **Ассоциация Rule Mining** - Найти интересные данные в данных.
- **Классификация** - предсказывать категориальные метки классов, которые являются двоичными (1 или 0).
- **Регрессия** - прогнозирование непрерывных значений, таких как продажи, цены и т.д.

«PyCaret демократизирует машинное обучение и использование передовой аналитики, предоставляя **бесплатное** решение для машинного обучения с **открытым исходным кодом и кодом с низким кодом** для бизнес-аналитиков, экспертов в области, гражданских данных и опытных исследователей данных».

Microsoft Power BI

Power BI - это решение для бизнес-аналитики, которое позволяет визуализировать ваши данные и обмениваться мнениями по всей организации или встраивать их в ваше приложение или веб-сайт. В этом руководстве мы будем использовать [Power BI Desktop](#) для машинного обучения, импортируя библиотеку PyCaret в Power BI.

Прежде чем мы начнем

Если вы уже использовали Python, вероятно, на вашем компьютере уже установлен Anaconda Distribution. Если нет, [нажмите здесь](https://www.anaconda.com/distribution/), чтобы загрузить Anaconda Distribution с Python 3.7 или выше.

<https://www.anaconda.com/distribution/>

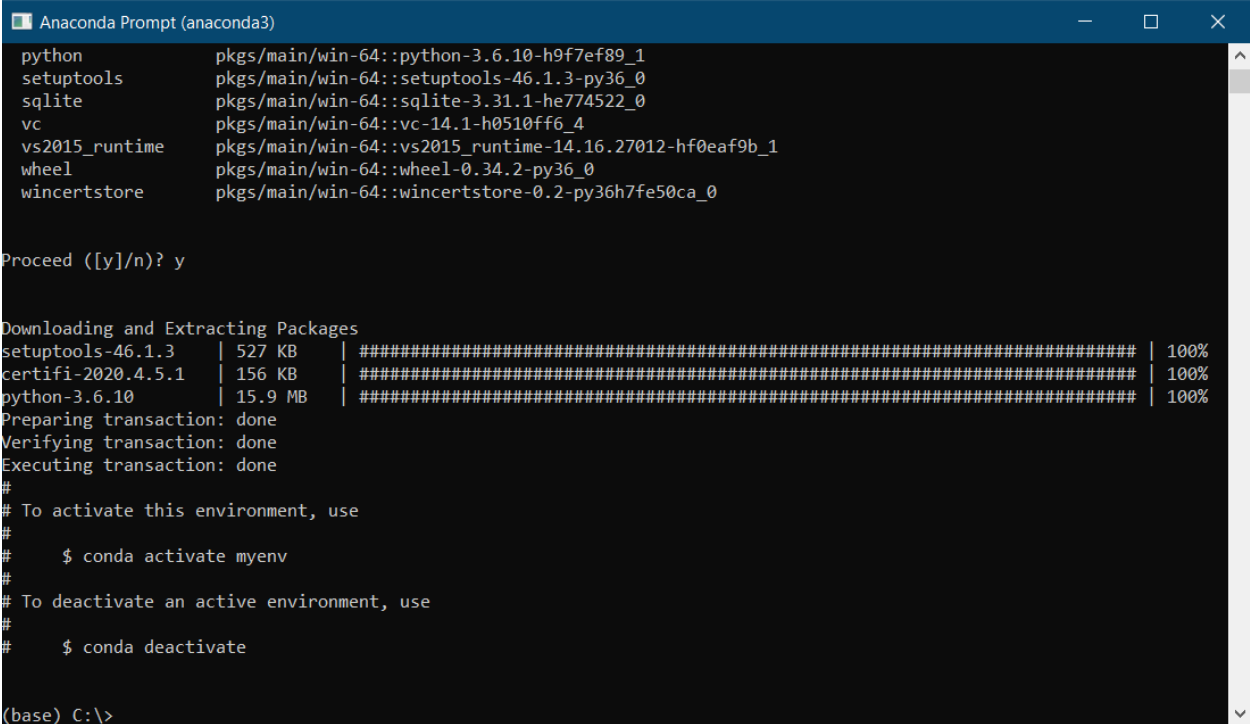
Настройка среды

Прежде чем мы начнем использовать возможности машинного обучения PyCaret в Power BI, нам нужно создать виртуальную среду и установить русcaret. Это трехступенчатый процесс:

☒ Шаг 1 - Создание среды анаконды

Откройте **Anaconda Prompt** из меню «Пуск» и запустите следующий код:

```
conda create --name myenv python=3.8
```



```
Anaconda Prompt (anaconda3)
python          pkgs/main/win-64::python-3.6.10-h9f7ef89_1
setuptools      pkgs/main/win-64::setuptools-46.1.3-py36_0
sqlite         pkgs/main/win-64::sqlite-3.31.1-he774522_0
vc             pkgs/main/win-64::vc-14.1-h0510ff6_4
vs2015_runtime pkgs/main/win-64::vs2015_runtime-14.16.27012-hf0eaf9b_1
wheel          pkgs/main/win-64::wheel-0.34.2-py36_0
wincertstore   pkgs/main/win-64::wincertstore-0.2-py36h7fe50ca_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
setuptools-46.1.3 | 527 KB | ##### | 100%
certifi-2020.4.5.1 | 156 KB | ##### | 100%
python-3.6.10 | 15.9 MB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

#
# To activate this environment, use
#
#   $ conda activate myenv
#
# To deactivate an active environment, use
#
#   $ conda deactivate
#

(base) C:\>
```

Anaconda Prompt - Создание среды

☒ Шаг 2 - Установка PyCaret

Запустите следующий код в Anaconda Prompt:

```
conda activate myenv
pip install pycaret
```

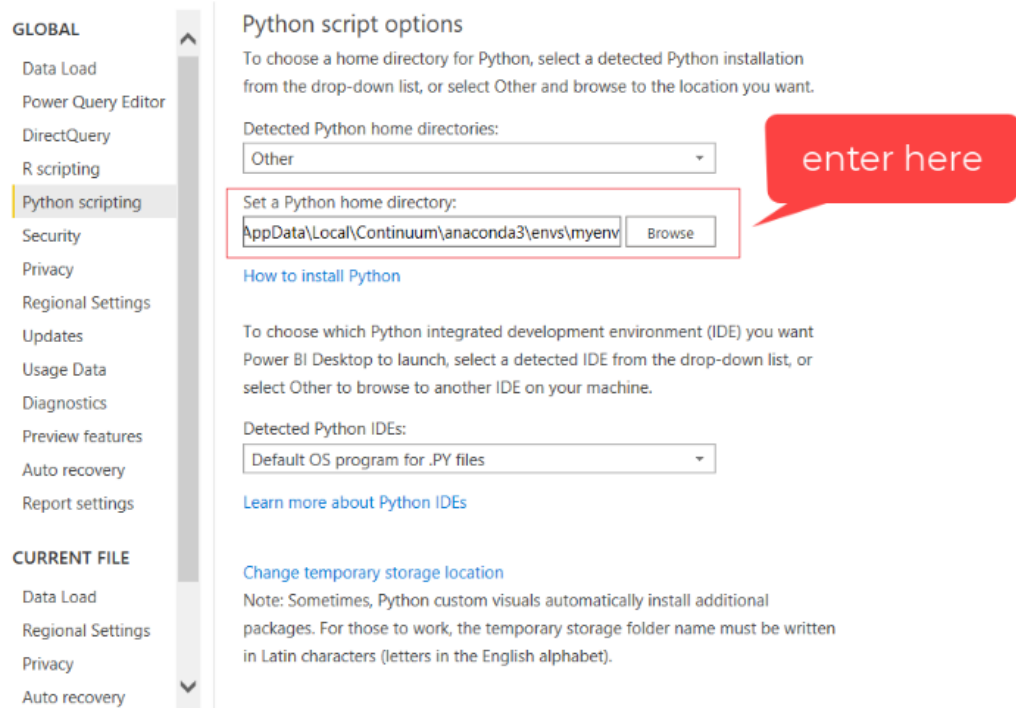
Установка может занять 10 - 15 минут.

☒ Шаг 3 - Установить каталог Python в Power BI

Созданная виртуальная среда должна быть связана с Power BI. Это можно сделать с помощью глобальных настроек в Power BI Desktop (Файл →

Параметры → Глобальные → Сценарии Python). Anaconda Environment по умолчанию устанавливается в:

C:\Users**username**\AppData\Local\Continuum\anaconda3\envs\myenv



File → Options → Global → Python scripting

1 Пример 1 - Кластеризация в Power BI

Кластеризация - это техника машинного обучения, которая группирует точки данных со сходными характеристиками. Эти группировки полезны для изучения данных, выявления закономерностей и анализа подмножества данных. Некоторые общие бизнес-сценарии для кластеризации:

✓ Сегментация клиентов с целью маркетинга.

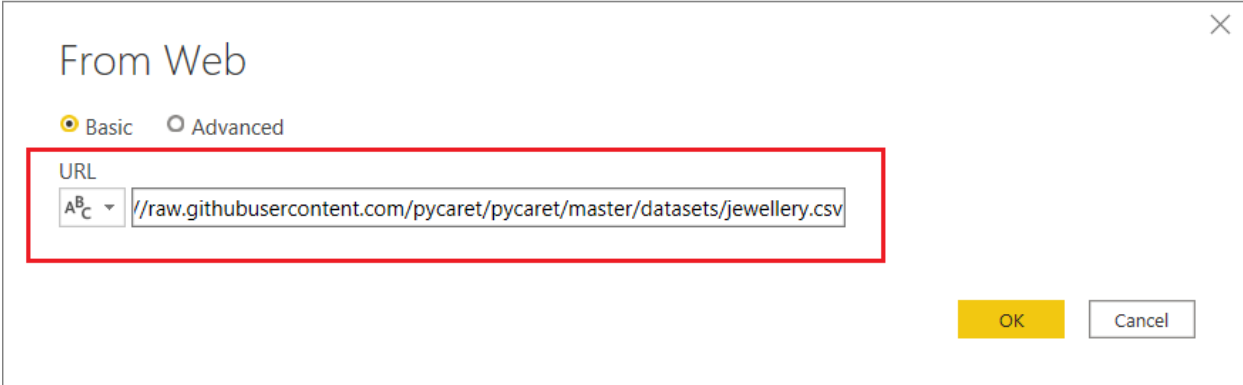
Analysis Анализ покупательского поведения покупателей для акций и скидок.

✓ Определение геокластеров в эпидемической вспышке, такой как COVID-19.

В этом уроке мы будем использовать «**jewellery.csv**» файл , который доступен на PyCaret в [хранилище GitHub](https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/jewellery.csv) . Вы можете загрузить данные с помощью веб-коннектора. (Power BI Desktop → Get Data → From Web).

Ссылка на файл CSV:

<https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/jewellery.csv>



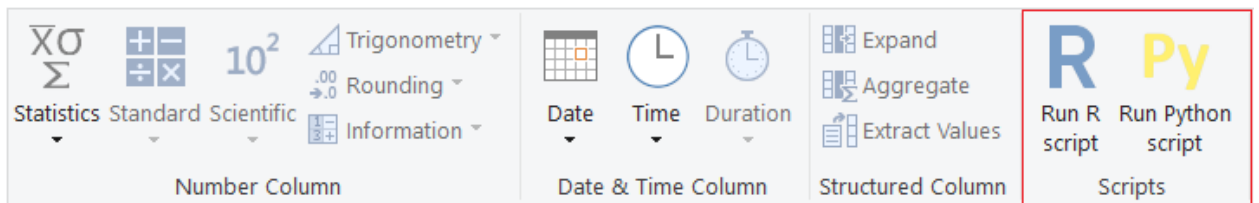
Power BI Desktop → Получить данные → Другое → Интернет

	1.3 Age	1.3 Income	1.2 SpendingScore	1.2 Savings
1	58	77769	0.791328777	6559.829923
2	59	81799	0.791082047	5417.661426
3	62	74751	0.702656952	9258.992965
4	59	74373	0.765679562	7346.334504
5	87	17760	0.348777548	16869.50713
6	29	131578	0.847034103	3535.514352
7	54	76500	0.78519785	6878.884249

Данные из jewellery.csv

К-means кластеризация

Для обучения модели кластеризации мы выполним скрипт Python в Power Query Editor (Power Query Editor → Transform → Run python script).



Лента в редакторе Power Query

Запустите следующий код как скрипт Python:

```
from pycaret.clustering import *
dataset = get_clusters(data = dataset)
```

Run Python script

Enter Python scripts into the editor to transform and shape your data.

Script

```
# 'dataset' holds the input data for this script

from pycaret.clustering import *
dataset = get_clusters(data = dataset)
```

OK

Cancel

Power Query Editor (Преобразование → Запуск скрипта Python)

Вывод:



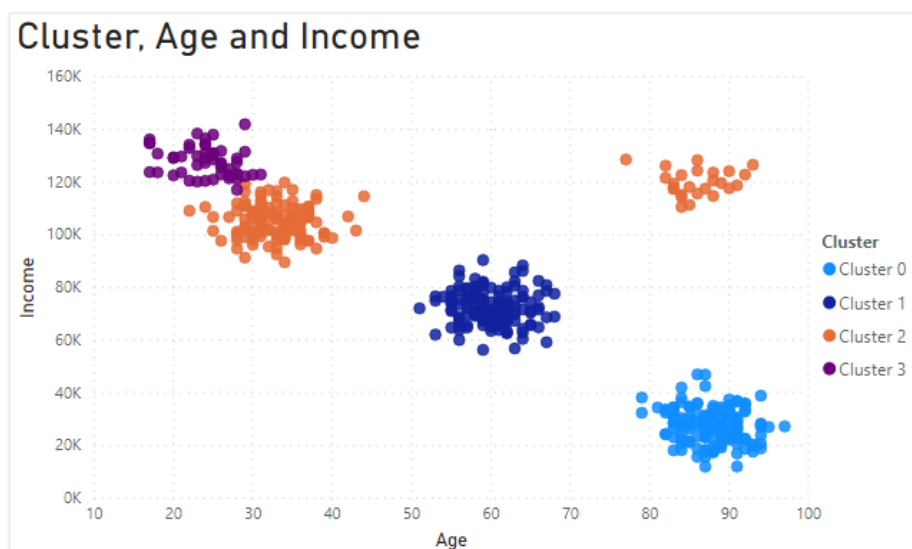
Результаты кластеризации (после выполнения кода)

	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	
		I_3^2 Income	1.2 SpendingScore	1.2 Savings	AB_C Cluster	
1		58	77769	0.791328777	6559.829923	Cluster 1
2		59	81799	0.791082047	5417.661426	Cluster 1
3		62	74751	0.702656952	9258.992965	Cluster 1
4		59	74373	0.765679562	7346.334504	Cluster 1
5		87	17760	0.348777548	16869.50713	Cluster 0
6		29	131578	0.847034103	3535.514352	Cluster 3
7		54	76500	0.78519785	6878.884249	Cluster 1
8		87	42592	0.355289682	18086.28716	Cluster 0
9		83	34384	0.324718682	14783.37909	Cluster 0
10		84	27693	0.367062973	17879.55891	Cluster 0
11		85	111389	0.036795362	16009.23776	Cluster 2
12		36	99780	0.265432607	16398.40133	Cluster 2

Окончательный вывод (после нажатия на таблицу)

Новый столбец «**Кластер**», содержащий метку, присоединяется к исходной таблице.

Когда вы примените запрос (Power Query Editor → Главная страница → Заккрыть и применить), вот как вы можете визуализировать кластеры в Power BI:



По умолчанию PyCaret обучает модель кластеризации **K-Means** с 4 кластерами (*т.е. все наблюдения в таблице делятся на 4 группы*). Значения по умолчанию можно легко изменить:

- Для того, чтобы изменить количество кластеров можно использовать ***num_clusters*** параметр в ***get_clusters()*** функцию.
- Чтобы изменить тип модели, используйте параметр ***модели*** в ***get_clusters()*** .

Пример кода обучающей модели K-Modes с 6 кластерами:

```
from pycaret.clustering import *
dataset = get_clusters(dataset, model = 'kmodes',
num_clusters = 6)
```

В PyCaret доступно 9 готовых к использованию алгоритмов кластеризации:

Model	Abbreviated String
K-Means Clustering	'kmeans'
Affinity Propagation	'ap'
Mean shift Clustering	'meanshift'
Spectral Clustering	'sc'
Agglomerative Clustering	'hclust'
Density-Based Spatial Clustering	'dbscan'
OPTICS Clustering	'optics'
Birch Clustering	'birch'
K-Modes Clustering	'kmodes'

Все задачи предварительной обработки, необходимые для обучения модели кластеризации, такие как [заполнение пропущенных значений](#) (если в таблице есть пропущенные или *нулевые* значения), [нормализация](#) или [OneHotEncoding](#) , все они автоматически выполняются перед обучением модели кластеризации. [Нажмите здесь](#), чтобы узнать больше о возможностях предварительной обработки PyCaret.

💡 В этом примере мы использовали **функцию `get_clusters()`** для назначения меток кластера в исходной таблице. Каждый раз, когда запрос обновляется, кластеры пересчитываются. Альтернативный способ реализовать это - использовать **функцию `pregnet_model()`** для прогнозирования меток кластера с использованием **предварительно обученной модели** в Python или в Power BI (см. *Пример 5 ниже, чтобы узнать, как обучать модели машинного обучения в среде Power BI*).

💡 Если вы хотите узнать, как обучить модель кластеризации в Python с помощью Jupyter Notebook, обратитесь к нашему [Руководству для начинающих по кластеризации 101](#). (кодирования не требуется).

2 Пример 2 - Обнаружение аномалий в Power BI

Обнаружение аномалий - это метод машинного обучения, используемый для выявления **редких событий или наблюдений** путем проверки строк в таблице, которые значительно отличаются от большинства строк. Как правило, поиск аномалий применяются в таких сферах, как банковское мошенничество, структурный дефект, медицинская проблема или ошибка.

Некоторые общие бизнес-случаи для обнаружения аномалий:

Обнаружение мошенничества (кредитные карты, страховка и т.д.) с использованием финансовых данных.

Обнаружение вторжений (системная безопасность, вредоносное ПО) или отслеживание скачков и падений сетевого трафика.

✓ Определение многовариантных выбросов в наборе данных.

В этом уроке мы будем использовать файл «**anomaly.csv**», доступный в [репозитории](#) PyCaret для [github](#). Вы можете загрузить данные с помощью веб-коннектора. ((Power BI Desktop → Get Data → From Web).

Ссылка на CSV-файл:

<https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/anomaly.csv>

	1.2 Col1	1.2 Col2	1.2 Col3	1.2 Col4	1.2 Col5	1.2 Col6
1	0.263995357	0.764928588	0.13842355	0.935242061	0.605866573	0.518789697
2	0.546092303	0.65397459	0.065575135	0.227771913	0.845269445	0.837065879
3	0.336714104	0.538842451	0.192801069	0.553562822	0.074514511	0.332993162
4	0.092107835	0.995016662	0.014465045	0.176370646	0.241530075	0.514723634
5	0.325261175	0.805967636	0.957033424	0.331664957	0.307923366	0.355314772
6	0.212464853	0.780304761	0.458443656	0.634508561	0.373030452	0.465650668
7	0.258565714	0.437317789	0.559647989	0.109202597	0.994553306	0.896994183
8	0.869236755	0.277978893	0.42307639	0.11247202	0.183727053	0.034959735
9	0.197077957	0.843918225	0.24339588	0.281278233	0.329148141	0.73458152
10	0.292984504	0.70343162	0.43962138	0.107867968	0.922947409	0.25345779
11	0.82178316	0.087778177	0.439292126	0.521600403	0.437606126	0.172573815
12	0.796622959	0.230542754	0.993018018	0.077074836	0.094068272	0.718627991
13	0.045577011	0.335214629	0.657811898	0.723782888	0.352304153	0.069136622

Данные из anomaly.csv

Поиск аномалий K-Nearest Neighbors

Подобно кластеризации, мы запустим скрипт Python из Power Query Editor (Transform → Run python script) для обучения модели обнаружения аномалий. Запустите следующий код как скрипт Python:

```
from pycaret.anomaly import *  
dataset = get_outliers(data = dataset)
```




Run Python script

Enter Python scripts into the editor to transform and shape your data.

Script

```
# 'dataset' holds the input data for this script  
from pycaret.anomaly import *  
dataset = get_outliers(data = dataset)
```

OK

Cancel

Power Query Editor (Преобразование → Запуск скрипта Python)

Вывод:



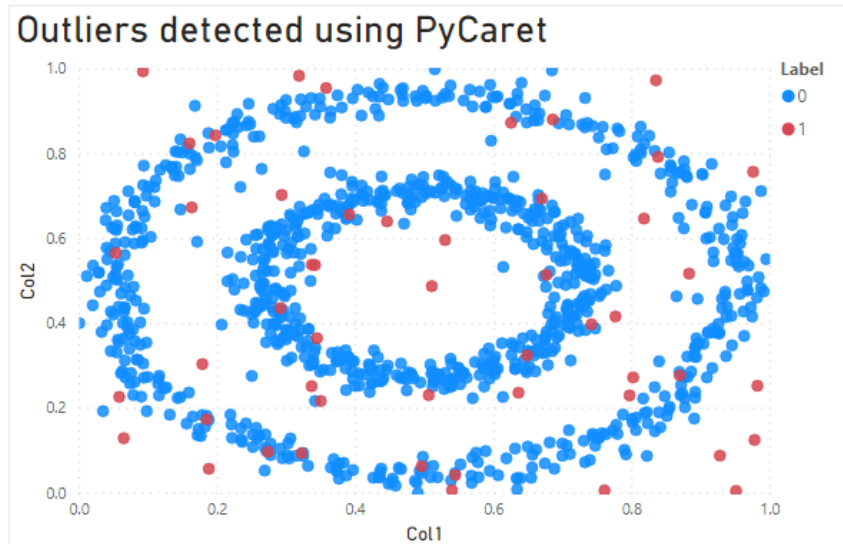
Результаты обнаружения аномалий (после выполнения кода)

A ^B _C Col8	A ^B _C Col9	A ^B _C Col10	A ^B _C Label	1.2 Score
0.6082344510000001	0.723781923	0.73359095	0	-0.021632123
0.331678698	0.429296975	0.367422001	0	-0.080252261
0.861309323	0.899016587	0.08860015199999999	1	0.0117293
0.158963258	0.073715215	0.20846322399999997	1	0.054517398
0.5584494520000001	0.885169295	0.18275440899999998	0	-0.011327248
0.013080053999999999	0.570250227	0.7366723629999999	0	-0.031493662
0.251942977	0.017265143	0.538513303	1	0.008970372
0.24932964600000002	0.5506833760000001	0.049843054000000005	1	0.01861474
0.927804425	0.71326865	0.891548497	1	0.025579324
0.355286799	0.980911322	0.308864217	0	-0.01172665
0.5942122710000001	0.456008312	0.441426572	0	-0.033814304
0.333385825	0.634842896	0.028728774	1	0.052863543
0.448041533	0.42671599600000004	0.894023006	0	-0.031497481
0.7884376479999999	0.32329341	0.93107397	1	0.006291799
0.803558106	0.465466125	0.30569249	1	0.028415784
0.84566925	0.155376795	0.442781372	1	0.000220965

Окончательный вывод (после нажатия на таблицу)

Две новые колонки прикреплены к исходной таблице. Label (1 = выброс, 0 = выброс) и оценка (точки данных с высокими показателями классифицируются как выбросы).

Как только вы примените запрос, вот как вы можете визуализировать результаты обнаружения аномалий в Power BI:



По умолчанию PyCaret обучает **K-Nearest Neighbors** с 5% -ной долей (т.е. 5% от общего числа строк в таблице будет помечено как выброс). Значения по умолчанию можно легко изменить:

- Чтобы изменить значение дроби, вы можете использовать параметр ***fraction*** в функции **get_outliers ()** .
- Чтобы изменить тип модели, используйте параметр ***модели*** в **get_outliers ()** .

См. Следующий код для обучения модели **Isolation Forest** с долей 0,1:

```
from pycaret.anomaly import *  
dataset = get_outliers(dataset, model = 'iforest',  
fraction = 0.1)
```

В PyCaret имеется более 10 готовых к использованию алгоритмов обнаружения аномалий:

Estimator	Abbrev.String
Angle-base Outlier Detection	'abod'
Isolation Forest	'iforest'
Clustering-Based Local Outlier	'cluster'
Connectivity-Based Outlier Factor	'cof'
Histogram-based Outlier Detection	'histogram'
k-Nearest Neighbors Detector	'knn'
Local Outlier Factor	'lof'
One-class SVM detector	'svm'
Principal Component Analysis	'pca'
Minimum Covariance Determinant	'mcd'
Subspace Outlier Detection	'sod'
Stochastic Outlier Selection	'sos'

Все задачи предварительной обработки, необходимые для обучения модели кластеризации, такие как [заполнение пропущенных значений](#) (если в таблице есть пропущенные или *нулевые* значения), [нормализация](#) или [OneHotEncoding](#) , все они автоматически выполняются перед обучением модели кластеризации. [Нажмите здесь](#), чтобы узнать больше о возможностях предварительной обработки PyCaret.

💡 В этом примере мы использовали **функцию `get_outliers()`**, чтобы назначить метку выброса и оценку для анализа. Каждый раз, когда запрос обновляется, выбросы пересчитываются. Альтернативным способом реализации этого было бы использование функции **`pregnet_model()`** для прогнозирования выбросов с использованием предварительно обученной модели в Python или в Power BI (см. *Пример 5 ниже, чтобы узнать, как обучать модели машинного обучения в среде Power BI*).

💡 Если вы хотите узнать, как обучить детектор аномалий в Python с помощью Jupyter Notebook, ознакомьтесь с нашим руководством для [начинающих по обнаружению аномалий 101](#) . (кодирования не требуется).

4 Пример 4 - майнинг правил ассоциации в Power BI

Ассоциация Rule Mining - это **основанная на правилах** техника **машинного обучения** для обнаружения интересных связей между переменными в базе данных. Он предназначен для выявления строгих правил с использованием мер интересности. Некоторые общие бизнес-сценарии для майнинга правил ассоциации:

Basket Анализ Корзины Рынка для понимания предметов, часто покупаемых вместе

✓ Медицинский диагноз, чтобы помочь врачам в определении вероятности возникновения заболевания с учетом факторов и симптомов.

В этом руководстве мы будем использовать файл **'france.csv'**, доступный в [репозитории](#) PyCaret для [github](#) . Вы можете загрузить данные с помощью веб-коннектора. (Power BI Desktop → Получить данные → Из Интернета).

Ссылка на CSV-файл:

<https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/france.csv>

	AB InvoiceNo	AB StockCode	AB Description	AB Quantity	AB InvoiceDate	AB UnitPrice
1	536370	22728	ALARM CLOCK BAKELIKE PINK	24	12/1/2010 8:45	3.75
2	536370	22727	ALARM CLOCK BAKELIKE RED	24	12/1/2010 8:45	3.75
3	536370	22726	ALARM CLOCK BAKELIKE GREEN	12	12/1/2010 8:45	3.75
4	536370	21724	PANDA AND BUNNIES STICKER SHE...	12	12/1/2010 8:45	0.85
5	536370	21883	STARS GIFT TAPE	24	12/1/2010 8:45	0.65
6	536370	10002	INFLATABLE POLITICAL GLOBE	48	12/1/2010 8:45	0.85
7	536370	21791	VINTAGE HEADS AND TAILS CARD ...	24	12/1/2010 8:45	1.25
8	536370	21035	SET/2 RED RETROSPOT TEA TOWELS	18	12/1/2010 8:45	2.95
9	536370	22326	ROUND SNACK BOXES SET OF4 W...	24	12/1/2010 8:45	2.95
10	536370	22629	SPACEBOY LUNCH BOX	24	12/1/2010 8:45	1.95
11	536370	22659	LUNCH BOX I LOVE LONDON	24	12/1/2010 8:45	1.95

Примеры точек данных из france.csv

Алгоритм Априори

Теперь должно быть ясно, что все функции PyCaret выполняются как скрипт Python в Power Query Editor (Transform → Run python script). Запустите следующий код для обучения модели правил ассоциации с использованием алгоритма Apriori:

```
from pycaret.arules import *  
dataset = get_rules(dataset, transaction_id =  
'InvoiceNo', item_id = 'Description')
```



Run Python script

Enter Python scripts into the editor to transform and shape your data.

Script

```
# 'dataset' holds the input data for this script

from pycaret.arules import *
dataset = get_rules(data = dataset, transaction_id = "InvoiceNo", item_id = "Description")
```

OK

Cancel

Power Query Editor (Преобразование → Запуск скрипта Python)

«**InvoiceNo**» - это столбец, содержащий идентификатор транзакции, а «**Description**» содержит интересующую переменную, т.е. название продукта.

Вывод:



Результаты майнинга правила ассоциации (после выполнения кода)

	Antecedents	Consequents	Antecedent support	Consequent support	Support	Confidence
1	frozenset({'JUMBO BAG WOODLA...	frozenset({'POSTAGE'})	0.0651	0.6746	0.0651	1
2	frozenset({'SET/6 RED SPOTTY PA...	frozenset({'SET/6 RED SPOTT...	0.0868	0.1085	0.0846	0.975
3	frozenset({'SET/20 RED RETROSP...	frozenset({'SET/6 RED SPOTT...	0.0868	0.1171	0.0846	0.975
4	frozenset({'SET/6 RED SPOTTY PA...	frozenset({'SET/6 RED SPOTT...	0.0716	0.1085	0.0694	0.9697
5	frozenset({'SET/20 RED RETROSP...	frozenset({'SET/6 RED SPOTT...	0.0716	0.1171	0.0694	0.9697
6	frozenset({'SET OF 9 BLACK SKULL...	frozenset({'POSTAGE'})	0.0564	0.6746	0.0542	0.9615
7	frozenset({'PACK OF 6 SKULL PAP...	frozenset({'POSTAGE'})	0.0542	0.6746	0.0521	0.96
8	frozenset({'SET/6 RED SPOTTY PA...	frozenset({'SET/6 RED SPOTT...	0.1085	0.1171	0.1041	0.96
9	frozenset({'SET/6 RED SPOTTY PA...	frozenset({'SET/6 RED SPOTT...	0.0911	0.1171	0.0868	0.9524
10	frozenset({'TEA PARTY BIRTHDAY ...	frozenset({'POSTAGE'})	0.0803	0.6746	0.0759	0.9459
11	frozenset({'STRAWBERRY LUNCH ...	frozenset({'POSTAGE'})	0.1041	0.6746	0.0976	0.9375
12	frozenset({'RED RETROSPOT PICN...	frozenset({'POSTAGE'})	0.0629	0.6746	0.0586	0.931
13	frozenset({'CHILDRENS CUTLERY ...	frozenset({'CHILDRENS CUTLE...	0.0586	0.0629	0.0542	0.9259

Окончательный вывод (после нажатия на таблицу)

Он возвращает таблицу с условиями (antecedents) и следствиям (consequents) со связанными метриками, такими как поддержка, доверие, лифт и т.д.

[Нажмите здесь](#), чтобы узнать больше о поиске правил ассоциации в PyCaret.

5 Пример 5 - Классификация в Power BI

Классификация - это контролируемая техника машинного обучения, используемая для прогнозирования категориальных **меток классов**.

Некоторые общие бизнес-сценарии классификации:

Default Прогнозирование кредита клиента / дефолта по кредитной карте.

- ✓ Прогнозирование оттока клиентов (останется ли клиент или уйдет)
- ✓ Прогнозирование результата пациента (независимо от того, есть у пациента заболевание или нет)

В этом уроке мы будем использовать «**employee.csv**» файл доступный на PyCaret в [хранилище GitHub](https://github.com/pycaret/pycaret) . Вы можете загрузить данные с помощью веб-коннектора. (Power BI Desktop → Получить данные → Из Интернета).

Ссылка на CSV-файл:

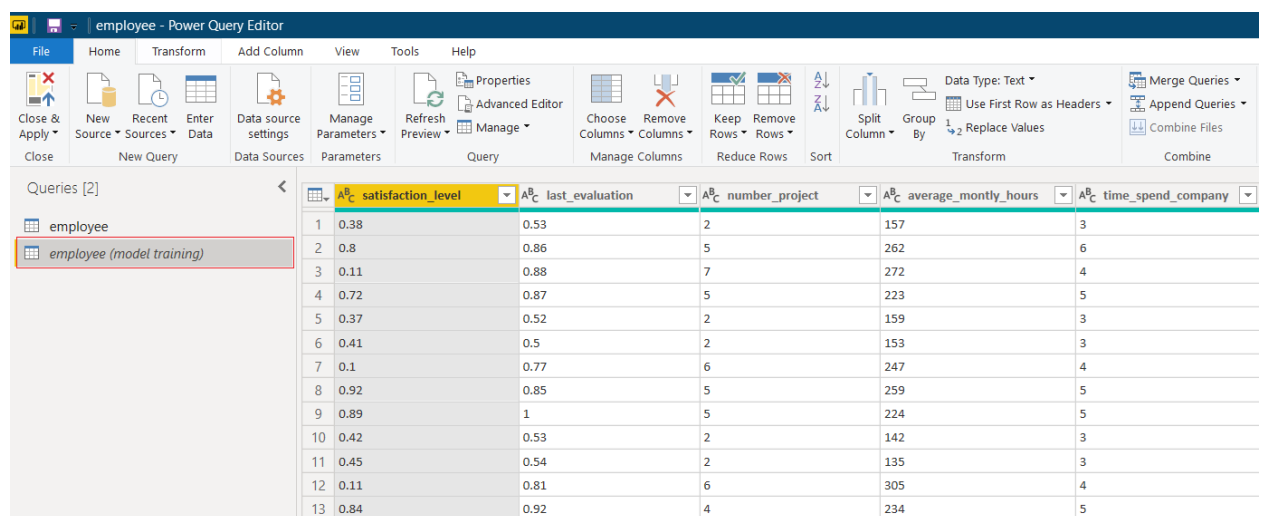
<https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/employee.csv>

Цель: таблица «**работник**» содержит информацию о 15 000 активных сотрудников в компании, такую как время, проведенное в компании, среднемесячное количество отработанных часов, история продвижения по службе, отдел и т.д. На основе всех этих столбцов (также называемых *функциями* в терминологии машинного обучения)) цель состоит в том, чтобы предсказать, покинет ли сотрудник компанию или нет, представленную в столбце «**left**» (1 означает «да», 0 означает «нет»).

В отличие от примеров кластеризации, обнаружения аномалий и НЛП, которые подпадают под класс неконтролируемого машинного обучения, классификация является **контролируемой** техникой и, следовательно, реализуется в двух частях:

Часть 1. Обучение модели классификации в Power BI

Первым шагом является создание дубликата таблицы 'employee' в Power Query Editor, которая будет использоваться для обучения модели.



The screenshot shows the Power Query Editor interface. The 'Queries' pane on the left lists 'employee' and 'employee (model training)'. The main area displays a table with 13 rows and 6 columns. The columns are: 'satisfaction_level', 'last_evaluation', 'number_project', 'average_monthly_hours', and 'time_spend_company'. The first column is highlighted in yellow.

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company
1	0.38	0.53	2	157	3
2	0.8	0.86	5	262	6
3	0.11	0.88	7	272	4
4	0.72	0.87	5	223	5
5	0.37	0.52	2	159	3
6	0.41	0.5	2	153	3
7	0.1	0.77	6	247	4
8	0.92	0.85	5	259	5
9	0.89	1	5	224	5
10	0.42	0.53	2	142	3
11	0.45	0.54	2	135	3
12	0.11	0.81	6	305	4
13	0.84	0.92	4	234	5

Редактор Power Query → Щелкните правой кнопкой мыши «employee» → Дублировать

Запустите следующий код во вновь созданной дублирующей таблице «**employee (обучение модели)**», чтобы обучить модель классификации:

```
# import classification module and setup environment
from pycaret.classification import *
clf1 = setup(dataset, target = 'left', silent = True)

# train and save lightgbm model
lightgbm = create_model('lightgbm')
final_lightgbm = finalize_model(lightgbm)
save_model(final_lightgbm, 'D:/YandexDisk/Doc/Power
BI/PyCaret-Power BI/lightgbm_powerbi_employee')
```

Power Query Editor (Преобразование → Запуск скрипта Python)

Вывод:

Результатом этого скрипта будет **файл .pkl**, сохраненный в указанном месте. Файл pickle содержит весь конвейер преобразования данных, а также обученный объект модели.

💡 Альтернативой этому будет обучение модели в ноутбуке Jupyter вместо Power BI. В этом случае Power BI будет использоваться только для генерации прогнозов во внешнем интерфейсе с использованием предварительно обученной модели в записной книжке Jupyter, которая будет импортирована в виде файла pickle в Power BI (см. Часть 2 ниже). Чтобы узнать больше об использовании PyCaret в Python, [нажмите здесь](#).

💡 Если вы хотите научиться обучать модели классификации в Python с помощью Jupyter Notebook, ознакомьтесь с нашим руководством для [начинающих по бинарной классификации 101](#). (кодирования не требуется). В PyCaret доступно 18 готовых к использованию алгоритмов классификации:

Estimator	Abbreviated String
Logistic Regression	'lr'
K Nearest Neighbour	'knn'
Naives Bayes	'nb'
Decision Tree	'dt'
SVM (Linear)	'svm'
SVM (RBF)	'rbfsvm'
Gaussian Process	'gpc'
Multi Level Perceptron	'mlp'
Ridge Classifier	'ridge'
Random Forest	'rf'
Quadratic Disc. Analysis	'qda'
AdaBoost	'ada'
Gradient Boosting Classifier	'gbc'
Linear Disc. Analysis	'lda'
Extra Trees Classifier	'et'
Extreme Gradient Boosting	'xgboost'
Light Gradient Boosting	'lightgbm'
Cat Boost Classifier	'catboost'

Часть 2. Создание прогнозов с использованием обученной модели

Теперь мы можем использовать обученную модель в исходной таблице **«employee»**, чтобы предсказать, покинет ли сотрудник компанию или нет (1 или 0) и вероятность%. Запустите следующий код как скрипт Python для генерации прогнозов:


```
from pycaret.classification import *
lightgbm = load_model('D:/YandexDisk/Doc/Power BI/PyCaret-Power
BI/lightgbm_powerbi_employee')
dataset = predict_model(lightgbm, data = dataset)
```

Запустить скрипт Python

Пишите в редакторе скрипты Python для преобразования и формирования данных.

Сценарий

```
# 'dataset' содержит входные данные для этого сценария

from pycaret.classification import *
lightgbm = load_model('D:/YandexDisk/Doc/Power BI/PyCaret/lightgbm_powerbi')
dataset = predict_model(lightgbm, data = dataset)
```

Вывод:



Классификационные прогнозы (после выполнения кода)

	promotion_last_5years	department	salary	left	Label	Score
1		sales	low	1	1	0.9558
2		sales	medium	1	1	0.576
3		sales	medium	1	1	0.9957
4		sales	low	1	1	0.9187
5		sales	low	1	1	0.9558
6		sales	low	1	1	0.9558
7		sales	low	1	1	0.9864
8		sales	low	1	1	0.932
9		sales	low	1	1	0.967
10		sales	low	1	1	0.9558
11		sales	low	1	1	0.9637
12		sales	low	1	1	0.9973
13		sales	low	1	1	0.9107
14		sales	low	1	1	0.9558

Окончательный вывод (после нажатия на таблицу)

Две новые колонки прикреплены к исходной таблице. «**Label**» колонка указывает предсказание и «**Score**» столбец вероятность исхода.

В этом примере мы предсказали те же данные, которые мы использовали для обучения модели только для демонстрационных целей. В реальных условиях столбец «left» является фактическим результатом и неизвестен во время прогнозирования.

В этом уроке мы обучили модель **LightGBM** ('lightgbm') и использовали ее для генерации прогнозов. Мы сделали это только для простоты. Практически вы можете использовать PyCaret для прогнозирования любого типа модели или цепочки моделей.

Функция `realt_model ()` **PyCaret** может беспрепятственно работать с файлом pickle, созданным с использованием PyCaret, поскольку он содержит весь конвейер преобразования вместе с обученным объектом модели. [Нажмите здесь](#), чтобы узнать больше о функции **predict**.

💡 Все задачи предварительной обработки, необходимые для обучения модели кластеризации, такие как [заполнение пропущенных значений](#) (если в таблице есть пропущенные или нулевые значения), [нормализация](#) или [OneHotEncoding](#), все они автоматически выполняются перед обучением модели кластеризации. [Нажмите здесь](#), чтобы узнать больше о возможностях предварительной обработки PyCaret.

6 Пример 6 - регрессия в Power BI

Регрессия - это контролируемая техника машинного обучения, используемая для прогнозирования непрерывного результата наилучшим образом с учетом прошлых данных и соответствующих прошлых результатов. В отличие от классификации, которая используется для прогнозирования двоичного результата, такого как «да» или «нет» (1 или 0), регрессия используется для прогнозирования непрерывных значений, таких как продажи, цена, количество и т.д.

В этом уроке мы будем использовать файл **'boston.csv'**, доступный в [репозитории](#) [pycaret github](#). Вы можете загрузить данные с помощью веб-коннектора. (Power BI Desktop → Получить данные → Из Интернета).

Ссылка на CSV-файл:

<https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/boston.csv>

Цель: таблица «Бостон» содержит информацию о 506 домах в Бостоне, такую как среднее количество комнат, ставки налога на имущество, население и т.д. На основе этих столбцов (также называемых *функциями* в терминологии машинного обучения) цель состоит в прогнозировании медианного значения дома, представленного колонкой **'medv'**.

Часть 1. Обучение модели регрессии в Power BI

Первым шагом является создание дубликата таблицы «**boston**» в Power Query Editor, который будет использоваться для обучения модели.

Запустите следующий код в новой дублирующейся таблице как скрипт Python:

```
# import regression module and setup environment
from pycaret.regression import *
clf1 = setup(dataset, target = 'medv', silent = True)

# train and save lightgbm model
lightgbm = create_model('lightgbm')
final_lightgbm = finalize_model(lightgbm)
save_model(final_lightgbm, 'D:/YandexDisk/Doc/Power
BI/PyCaret/lightgbm_powerbi_reg_boston')
```

Вывод:

Результатом этого скрипта будет **файл .pkl**, сохраненный в указанном месте. Файл pickle содержит весь конвейер преобразования данных, а также обученный объект модели.

После сохранения модели удалить этот шаг в Power Query!

В PyCaret доступно более 20 готовых к использованию алгоритмов регрессии:

Estimator	Abbreviated String	Estimator	Abbreviated String
Linear Regression	'lr'	Kernel Ridge	'kr'
Lasso Regression	'lasso'	Support Vector Machine	'svm'
Ridge Regression	'ridge'	K Neighbors Regressor	'knn'
Elastic Net	'en'	Decision Tree	'dt'
Least Angle Regression	'lar'	Random Forest	'rf'
Lasso Least Angle Regression	'llar'	Extra Trees Regressor	'et'
Orthogonal Matching Pursuit	'omp'	AdaBoost Regressor	'ada'
Bayesian Ridge	'br'	Gradient Boosting Regressor	'gbr'
Automatic Relevance Determination	'ard'	Multi Level Perceptron	'mlp'
Passive Aggressive Regressor	'par'	Extreme Gradient Boosting	'xgboost'
Random Sample Consensus	'ransac'	Light Gradient Boosting	'lightgbm'
TheilSen Regressor	'tr'	CatBoost Regressor	'catboost'

Часть 2. Создание прогнозов с использованием обученной модели

Теперь мы можем использовать обученную модель для прогнозирования средней стоимости домов. Запустите следующий код в исходной таблице **'boston'** как скрипт python:

```
from pycaret.regression import *
lightgbm = load_model('D:/YandexDisk/Doc/Power
BI/PyCaret/lightgbm_powerbi_reg_boston')
dataset = predict_model(lightgbm, data = dataset)
```

Вывод:



Прогнозы регрессии (после выполнения кода)

tax	1.2 ptratio	1.2 black	1.2 lstat	1.2 medv	1.2 Label
296	15.3	396.9	4.98	24	23.9655
242	17.8	396.9	9.14	21.6	21.1266
242	17.8	392.83	4.03	34.7	34.9086
222	18.7	394.63	2.94	33.4	33.9216
222	18.7	396.9	5.33	36.2	35.2917
222	18.7	394.12	5.21	28.7	28.1702
311	15.2	395.6	12.43	22.9	21.6987
311	15.2	396.9	19.15	27.1	24.6484
311	15.2	386.63	29.93	16.5	16.6386
311	15.2	386.71	17.1	18.9	18.7637
311	15.2	392.52	20.45	15	16.3159
311	15.2	396.9	13.27	18.9	19.8155

Окончательный вывод (после нажатия на таблицу)

Новый столбец «**Label**», который содержит прогнозы, прикрепляется к исходной таблице.

В этом примере мы предсказали те же данные, которые мы использовали для обучения модели только для демонстрационных целей. В реальных условиях столбец «**medv**» является фактическим результатом и неизвестен во время прогнозирования.

💡 Все задачи предварительной обработки, необходимые для обучения модели регрессии, такие как [вменение пропущенного значения](#) (если в таблице есть пропущенные или нулевые значения), или [горячее кодирование](#) , или [целевое преобразование](#) , все они автоматически выполняются перед обучением модели. [Нажмите здесь](#), чтобы узнать больше о возможностях предварительной обработки PyCaret.