Департамент образования и науки города Москвы Государственное автономное образовательное учреждение высшего образования города Москвы «Московский городской педагогический университет» Институт цифрового образования Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Инструменты для хранения и обработки больших данных

Практическая работа №4.1

Тема:

Marketing Analytics

Выполнила: Шепелева Е. В., группа: АДЭУ-201

Преподаватель: Т. М. Босенко

Москва

2022

colab.research.google.com/drive/1DFkLFlMb_NhV9lKnTgD0ksta19TJ9Fgb#scrollTo=HDfT0VRSTMH8

marketing_analytics_students.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

Комментировать Поделиться

+ Код + Текст

ОЗУ Диск

```
!pip install dython
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: dython in /usr/local/lib/python3.10/dist-packages (0.7.3)
Requirement already satisfied: scipy>=1.7.1 in /usr/local/lib/python3.10/dist-packages (from dython) (1.10.1)
Requirement already satisfied: scikit-plot>=0.3.7 in /usr/local/lib/python3.10/dist-packages (from dython) (0.3.7)
Requirement already satisfied: numpy>=1.23.0 in /usr/local/lib/python3.10/dist-packages (from dython) (1.24.3)
Requirement already satisfied: pandas>=1.4.2 in /usr/local/lib/python3.10/dist-packages (from dython) (1.5.3)
Requirement already satisfied: seaborn>=0.12.0 in /usr/local/lib/python3.10/dist-packages (from dython) (0.12.2)
Requirement already satisfied: matplotlib>=3.5.3 in /usr/local/lib/python3.10/dist-packages (from dython) (3.7.1)
Requirement already satisfied: scikit-learn>=0.24.2 in /usr/local/lib/python3.10/dist-packages (from dython) (1.2.2)
Requirement already satisfied: psutil>=5.9.1 in /usr/local/lib/python3.10/dist-packages (from dython) (5.9.5)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.3->dython) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.3->dython) (23.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.3->dython) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.3->dython) (4.39.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.3->dython) (0.11.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.3->dython) (3.0.9)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.3->dython) (8.4.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.3->dython) (1.0.7)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.4.2->dython) (2022.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.24.2->dython) (3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.24.2->dython) (1.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.5.3->dython) (1.16.0)
```

[4]
```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

colab.research.google.com/drive/1DFkLFlMb_NhV9lKnTgD0ksta19TJ9Fgb#scrollTo=HDfT0VRSTMH8

marketing_analytics_students.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

Комментировать Поделиться

+ Код + Текст

ОЗУ Диск

[5]
```
pd.options.mode.chained_assignment = None
```

Load the data

Для этого проекта использовался набор маркетинговых данных из задачи команды iFood Brain в роли аналитиков данных. Этот набор данных содержит социально-демографические и фирмографические характеристики 2240 клиентов.

[6]
```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

```
Выбрать файлы marketing_data.csv
```
- **marketing_data.csv**(text/csv) - 227054 bytes, last modified: 05.02.2022 - 100% done

```
Saving marketing_data.csv to marketing_data.csv
User uploaded file "marketing_data.csv" with length 227054 bytes
```

```
# Load dataset
df = pd.read_csv('marketing_data.csv').iloc[: , 1:]
```

[8]
```
# Rename Pandas columns to lower case
df.columns = df.columns.str.lower()
```

colab.research.google.com/drive/1DFkLFlMb_NhV9lKnTgD0ksta19TJ9Fgb#scrollTo=HDfT0VRSTMH8

marketing_analytics_students.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

Комментировать Поделиться

+ Код + Текст

ОЗУ Диск

[9]
```
df = df.applymap(lambda s: s.lower() if type(s) == str else s)
```

[10]
```
# Examine the data
df.head()
```

	year_birth	education	marital_status	income	kidhome	teenhome	dt_customer	recency	mntwines	mntfruits	...	numstorepurchases	numwebvisitsmonth	acceptedcmp3	acceptedcmp4
0	1970	graduation	divorced	$84,835.00	0	0	6/16/14	0	189	104	...	6	1	0	0
1	1961	graduation	single	$57,091.00	0	0	6/15/14	0	464	5	...	7	5	0	0
2	1958	graduation	married	$67,267.00	0	1	5/13/14	0	134	11	...	5	2	0	0
3	1967	graduation	together	$32,474.00	1	1	5/11/14	0	10	0	...	2	7	0	0
4	1989	graduation	single	$21,474.00	1	0	4/8/14	0	6	16	...	2	7	1	0

5 rows × 27 columns

```
# Overview of all variables, their datatypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   year_birth           2240 non-null   int64
 1   education            2240 non-null   object
 2   marital_status       2240 non-null   object
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   year_birth           2240 non-null   int64 
 1   education            2240 non-null   object
 2   marital_status       2240 non-null   object
 3    income              2216 non-null   object
 4   kidhome              2240 non-null   int64 
 5   teenhome             2240 non-null   int64 
 6   dt_customer          2240 non-null   object
 7   recency              2240 non-null   int64 
 8   mntwines             2240 non-null   int64 
 9   mntfruits            2240 non-null   int64 
 10  mntmeatproducts      2240 non-null   int64 
 11  mntfishproducts      2240 non-null   int64 
 12  mntsweetproducts     2240 non-null   int64 
 13  mntgoldprods         2240 non-null   int64 
 14  numdealspurchases    2240 non-null   int64 
 15  numwebpurchases      2240 non-null   int64 
 16  numcatalogpurchases  2240 non-null   int64 
 17  numstorepurchases    2240 non-null   int64 
 18  numwebvisitsmonth    2240 non-null   int64 
 19  acceptedcmp3         2240 non-null   int64 
 20  acceptedcmp4         2240 non-null   int64 
 21  acceptedcmp5         2240 non-null   int64 
 22  acceptedcmp1         2240 non-null   int64 
 23  acceptedcmp2         2240 non-null   int64 
 24  response             2240 non-null   int64 
 25  complain             2240 non-null   int64 
 26  country              2240 non-null   object
dtypes: int64(22), object(5)
memory usage: 472.6+ KB
```

colab.research.google.com/drive/1DFkLFIMb_NhV9IKnTgD0ksta19TJ9Fgb#scrollTo=HDfT0VRSTMH8

marketing_analytics_students.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

Комментировать Поделиться

+ Код + Текст

ОЗУ Диск

```
[12] # Clean up column names that contain whitespace
     df.columns = df.columns.str.replace(' ', '')

[13] # Transform income column to a numerical
     df['income'] = df['income'].str.replace(',','').str.replace('$','').astype('float')
```

```
<ipython-input-13-9894eb322af8>:2: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will
  df['income'] = df['income'].str.replace(',','').str.replace('$','').astype('float')
```

```
[14] import datetime

[15] current_year = datetime.date.today().year
     current_year
```

```
2023
```

```
[16] # Replace 'year_birth' with 'age'
     df['age'] = current_year - df['year_birth']

     # Modify date of enrollment to total number of months since enrollment
     df['enrollment_month'] = (pd.to_datetime('now') - pd.to_datetime(df['dt_customer'])) // np.timedelta64(1,'M')
```

```
<ipython-input-17-3e479ab1ed3d>:2: FutureWarning: The parsing of 'now' in pd.to_datetime without `utc=True` is deprecated. In a future version, this will match Timestamp('now') and Tim
  df['enrollment_month'] = (pd.to_datetime('now') - pd.to_datetime(df['dt_customer'])) // np.timedelta64(1,'M')
```

```
[18] # Rename the column 'response'
     df = df.rename(columns = {'response': 'acceptedcmp6'})
```

colab.research.google.com/drive/1DFkLFIMb_NhV9IKnTgD0ksta19TJ9Fgb#scrollTo=HDfT0VRSTMH8

marketing_analytics_students.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

Комментировать Поделиться

+ Код + Текст

ОЗУ Диск

```
[19] # Drop unnecessary columns
     df = df.drop(['year_birth', 'dt_customer'], axis = 1)

     df.describe()
```

	income	kidhome	teenhome	recency	mntwines	mntfruits	mntmeatproducts	mntfishproducts	mntsweetproducts	mntgoldprods	...	numwebvisitsmonth	acceptedcmp
count	2216.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	...	2240.000000	2240.0000
mean	52247.251354	0.444196	0.506250	49.109375	303.935714	26.302232	166.950000	37.525446	27.062946	44.021875	...	5.316518	0.0727
std	25173.076661	0.538398	0.544538	28.962453	336.597393	39.773434	225.715373	54.628979	41.280498	52.167439	...	2.426645	0.2598
min	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.0000
25%	35303.000000	0.000000	0.000000	24.000000	23.750000	1.000000	16.000000	3.000000	1.000000	9.000000	...	3.000000	0.0000
50%	51381.500000	0.000000	0.000000	49.000000	173.500000	8.000000	67.000000	12.000000	8.000000	24.000000	...	6.000000	0.0000
75%	68522.000000	1.000000	1.000000	74.000000	504.250000	33.000000	232.000000	50.000000	33.000000	56.000000	...	7.000000	0.0000
max	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	263.000000	362.000000	...	20.000000	1.0000

8 rows × 24 columns

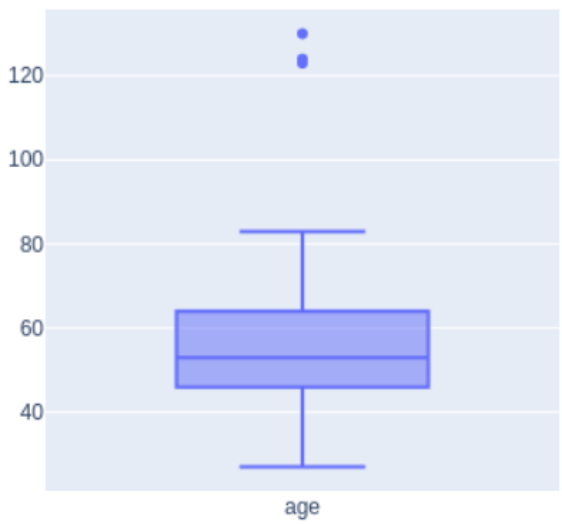
Мы можем выделить некоторые выбросы как по возрасту, так и по доходу. Похоже, у нас есть клиенты, которым больше 100 лет, или их доход на семью превышает 600 000 долларов США!

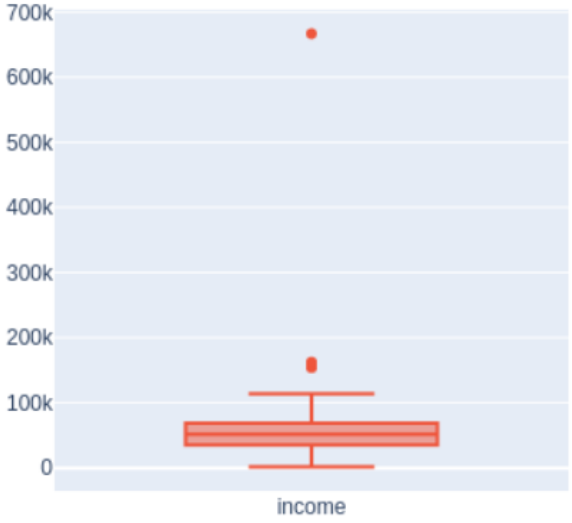
```
[22] !pip install gitly
```

[22] !pip install gitly

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gitly in /usr/local/lib/python3.10/dist-packages (1.1.4)
Requirement already satisfied: kaleido in /usr/local/lib/python3.10/dist-packages (from gitly) (0.2.1)
Requirement already satisfied: plotly>=4.9.0 in /usr/local/lib/python3.10/dist-packages (from gitly) (5.13.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly>=4.9.0->gitly) (8.2.2)
```

```
[23] from plotly.subplots import make_subplots
     import plotly.graph_objects as go
     from gitly.colab.plot import GitlyPlotter

[24] gitly = GitlyPlotter('github')

     fig = make_subplots(rows = 1, cols = 2)

     fig.add_trace(go.Box(y = df['age'], name = 'age'), row = 1, col = 1)
     fig.add_trace(go.Box(y = df['income'], name = 'income'), row = 1, col = 2)

     fig.update_layout(showlegend = False)

     gitly.show(fig)
```





colab.research.google.com/drive/1DFkLFIMb_NhV9IKnTgD0ksta19TJ9Fgb#scrollTo=HDfT0VRSTMH8

marketing_analytics_students.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

Комментировать Поделиться

+ Код + Текст

У нас есть 4 клиента, которые являются исключениями. Один из них зарабатывает 666 666 долларов США, а троим из них больше 100 лет!

```
[26] # Find outliers
     outliers_age = df[df['age'] > 100].index

     # Remove outliers
     df.drop(outliers_age, inplace = True)
```

```
[27] # Find outlier
     outliers_income = df[df['income'] > 200000].index

     # Remove outlier
     df.drop(outliers_income, inplace = True)
```

```
df['education'].value_counts()
```

```
graduation    1126
phd            485
master         370
2n cycle       201
basic           54
Name: education, dtype: int64
```

С точки зрения образования, и «2-й цикл», и «магистр» относятся к одному и тому же уровню образования. Это основано на Европейском пространстве высшего образования (EHEA). Поэтому мы объединим два уровня образования под словом «магистр». Кроме того, «выпускной» немного вводит в заблуждение как уровень образования. Мы предположим, что это относится к «бакалавриату» и перефразируем его как таковое.

colab.research.google.com/drive/1DFkLFIMb_NhV9IKnTgD0ksta19TJ9Fgb#scrollTo=HDfT0VRSTMH8

marketing_analytics_students.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

Комментировать Поделиться

+ Код + Текст

```
[29] # Replace '2n cycle' with 'master'
     df['education'] = df['education'].apply(lambda x: 'master' if str(x) == '2n cycle' else str(x))
```

```
     # Replace 'graduation' with 'undergraduate'
     df['education'] = df['education'].apply(lambda x: 'undergraduate' if str(x) == 'graduation' else str(x))
```

```
[31] df['marital_status'].value_counts()
```

```
married     864
together    578
single      479
divorced    231
widow        77
alone         3
yolo          2
absurd        2
Name: marital_status, dtype: int64
```

С точки зрения семейного положения, 'yolo', 'alone', и 'absurd'могут быть истолкованы и приняты как означающие «не замужем», и поэтому эти статусы будут объединены в «холост».

```
[32] # Merge 'yolo', 'absurd', and 'alone' under 'single'
     df['marital_status'] = df['marital_status'].apply(lambda x: 'single' if str(x) in ['alone', 'yolo', 'absurd'] else str(x))
```

```
df['country'].value_counts()
```

```
sp    1094
sa     335
ca     268
```

```
sp     1094
sa      335
ca      268
aus     160
ind     147
ger     120
us      109
me        3
Name: country, dtype: int64
```

Check for missing values

```
df.isnull().sum()
```

```
education              0
marital_status         0
income                24
kidhome                0
teenhome               0
recency                0
mntwines               0
mntfruits              0
mntmeatproducts        0
mntfishproducts        0
mntsweetproducts       0
mntgoldprods           0
numdealspurchases      0
numwebpurchases        0
numcatalogpurchases    0
numstorepurchases      0
numwebvisitsmonth      0
acceptedcmp3           0
acceptedcmp4           0
acceptedcmp5           0
acceptedcmp1           0
acceptedcmp2           0
acceptedcmp6           0
complain               0
country                0
age                    0
enrollment_month       0
dtype: int64
```

У нас отсутствуют данные о доходах 24 наших клиентов.

Определим X и Y

```
[35] X = df.drop('numstorepurchases', axis = 1)
```

```
[36] y = df['numstorepurchases']
```

Create test and train data

```
[1]  from sklearn.model_selection import train_test_split
```

```
[37] # Isolate X and y variables, and perform train-test split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

```
[38] from sklearn.compose import ColumnTransformer
     from sklearn.impute import SimpleImputer
```

```
     transformer = ColumnTransformer(transformers = [('simple_imputer', SimpleImputer(strategy = 'median'), ['income'])], remainder = 'passthrough')
```

Examine collinearity

Включение функций, которые сильно коррелируют друг с другом или являются мультиколлинеарными, добавляет шум и неточность, поэтому нам нужно попытаться уменьшить это.

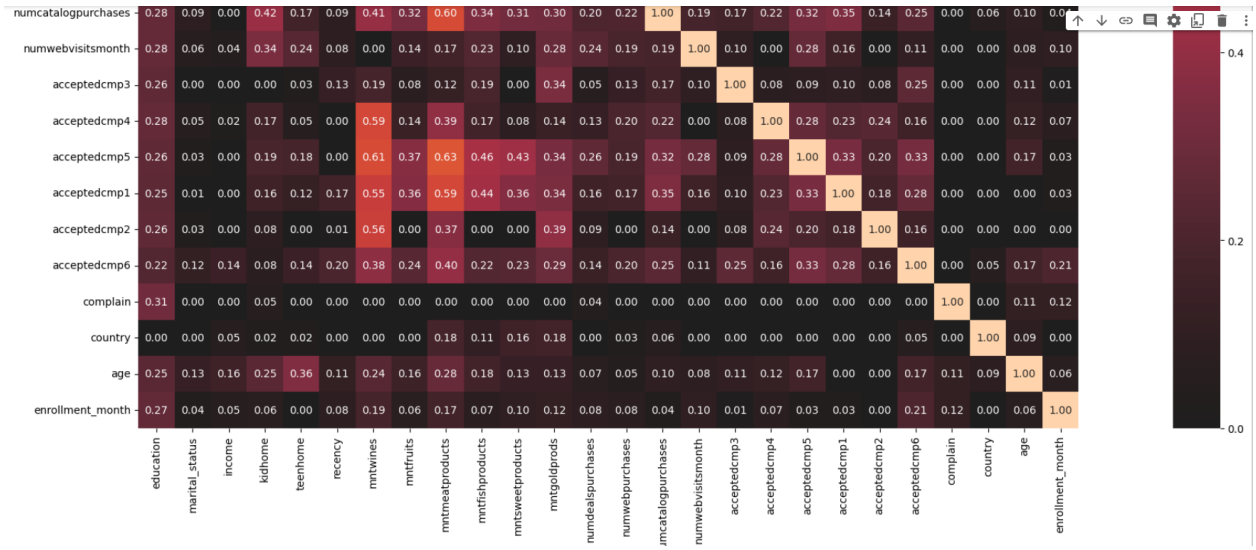
Создание тепловой карты корреляции — хороший способ визуализировать потенциальную коллинеарность. Эмпирическое правило состоит в том, что если корреляция между двумя независимыми переменными больше 0,8, тогда будет существовать мультиколлинеарность.

```
[40] X_tr = transformer.fit_transform(X_train)
```

```
[41] X_tr = pd.DataFrame(data = X_tr, columns = X.columns)
```

```
[43] from dython.nominal import associations
```

```
     complete_correlation = associations(X_tr, figsize = (32, 16))
```



colab.research.google.com/drive/1DFkLFIMb_NhV9IKnTgD0ksta19TJ9Fgb#scrollTo=apL3HbBwTMGZ

marketing_analytics_students.ipynb

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

Комментировать Поделиться

+ Код + Текст

ОЗУ Диск

Какие факторы существенно влияют на количество покупок в магазине?

Мы будем использовать модель CatBoostRegressor с numstorepurchases в качестве целевой переменной, а затем использовать методы объяснимости машинного обучения, чтобы получить представление о том, какие функции предсказывают количество покупок в магазине.

Мы создаем конвейер, который меняет любые отсутствующие значения, применяет надежный масштаб к функциям, преобразует любые категориальные функции в числовые, а затем подгоняет модель CatBoostRegressor.

```
!pip install catboost
```

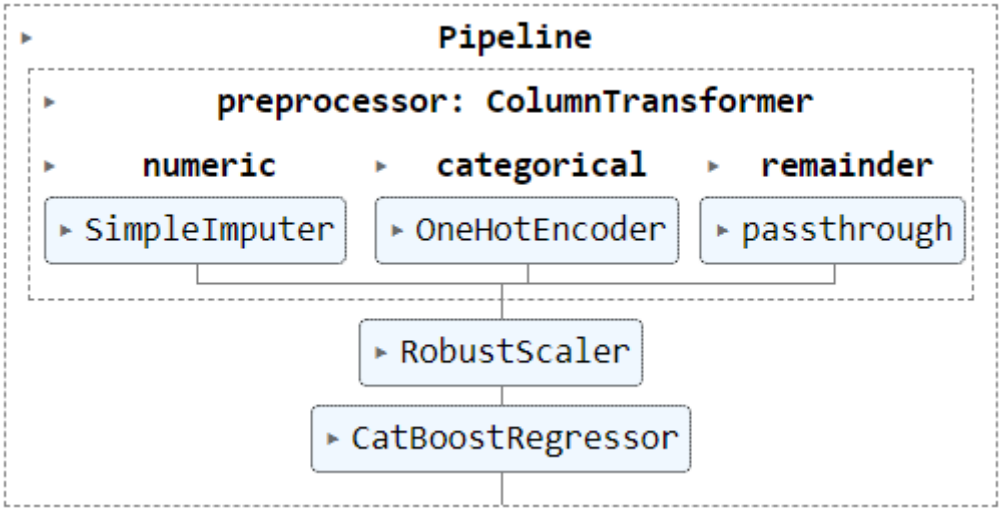
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting catboost
  Downloading catboost-1.1.1-cp310-none-manylinux1_x86_64.whl (76.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.6/76.6 MB 10.8 MB/s eta 0:00:00
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.13.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.5.3)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.24.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.10.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.0->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.0->catboost) (2022.7.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.39.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.4)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (8.4.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.0.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (23.1)
```

```
[46] from sklearn.pipeline import Pipeline
     from sklearn.preprocessing import OneHotEncoder
     from sklearn.preprocessing import RobustScaler
     from catboost import CatBoostRegressor
```

```
[47] numeric_columns = list(X_train.select_dtypes(exclude = ['object']).columns.values.tolist())
     categorical_columns = list(X_train.select_dtypes(include = ['object']).columns.values.tolist())
     numeric_transformer = Pipeline(steps = [('simple_imputer', SimpleImputer(strategy = 'median'))])
     categorical_transformer = Pipeline(steps = [('one_hot_encoder', OneHotEncoder(sparse = False, handle_unknown = 'ignore'))])

     preprocessor = ColumnTransformer(transformers = [('numeric', numeric_transformer, numeric_columns),
                                                      ('categorical', categorical_transformer, categorical_columns)], remainder = 'passthrough')

     bundled_pipeline = Pipeline(steps = [('preprocessor', preprocessor),
                                          ('scaler', RobustScaler()),
                                          ('model', CatBoostRegressor(silent = True, random_state = 42))])
```

```
bundled_pipeline.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning:

`sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
```




```
[49] y_pred = bundled_pipeline.predict(X_test)
```

```
[50] from sklearn.metrics import mean_absolute_error
```

```
[51] mean_absolute_error(y_test, y_pred)
```

```
1.1197249941682288
```

ELI5

Из этого пайплайна непросто извлечь важные функции. Однако есть библиотека python, которая делает это очень простым, под названием ELI5.

Давайте используем ELI5 для извлечения важности функций из конвейера.

ELI5 необходимо знать все имена функций, чтобы определить важность функций. Применяя однократное кодирование к категориальным переменным в конвейере, мы вводим ряд новых функций. Поэтому сначала нам нужно извлечь эти имена функций и добавить их к известному списку числовых функций. В приведенном ниже коде для этого используется функция named_steps, встроенная в конвейеры scikit-learn.

```
[52] one_hot_columns = list(bundled_pipeline.named_steps['preprocessor'].named_transformers_['categorical'].named_steps['one_hot_encoder'].get_feature_names_out(input_features = categorica
```

```
[53] numeric_features_list = list(numeric_columns)
     numeric_features_list.extend(one_hot_columns)
```

```
[54] !pip install eli5
```

```
[55] import eli5
```

To extract the feature importances we then simply need to run this line of code.

```
eli5.explain_weights(bundled_pipeline.named_steps['model'], top = 50, feature_names = numeric_features_list)
```

Weight	Feature
0.2293	mntwines
0.1161	mntmeatproducts
0.0691	numcatalogpurchases
0.0673	income
0.0626	mntsweetproducts
0.0588	mntfishproducts
0.0585	mntfruits
0.0483	numwebpurchases
0.0398	numwebvisitsmonth
0.0387	mntgoldprods
0.0325	age
0.0316	recency
0.0300	acceptedcmp6
0.0230	enrollment_month
0.0225	numdealspurchases
0.0072	marital_status_together
0.0069	education_phd
0.0061	teenhome
0.0060	education_master
0.0056	marital_status_married
0.0056	kidhome
0.0050	country_sp
0.0043	acceptedcmp3
0.0032	country_aus
0.0030	education_undergraduate
0.0029	acceptedcmp4

Здесь мы отмечаем, что «mntwines» и «mntmeatproducts» являются наиболее важными функциями.

Важные замечание

- Чем точнее модель, тем надежнее рассчитанные значения важности.
- Вычисленные значения важности описывают, насколько важны функции для модели CatBoostRegressor. Это приблизительное представление о том, насколько важны функции в данных.

С точки зрения общего объема покупок дела в США значительно лучше, чем в остальном мире?

```
[57] totalpurchases = df[['numdealspurchases', 'numwebpurchases', 'numcatalogpurchases', 'numstorepurchases', 'country']]
```

```
[58] # Calculate the total number of purchases made through different channels
totalpurchases['totalpurchases'] = totalpurchases['numdealspurchases'] + totalpurchases['numwebpurchases'] + totalpurchases['numcatalogpurchases'] + totalpurchases['numstorepurchases'
```

```
[59] average_purchases_per_country = totalpurchases.groupby('country').agg(total_purchases = ('totalpurchases', 'sum'))
```

```
[60] average_purchases_per_country['total_customers'] = totalpurchases['country'].value_counts()
```

```
average_purchases_per_country['purchases_per_customer'] = np.floor(average_purchases_per_country['total_purchases'] / average_purchases_per_country['total_customers'])
```

```
[62] average_purchases_per_country.assign(country = average_purchases_per_country.index.get_level_values('country'))
```

country	total_purchases	total_customers	purchases_per_customer	country
aus	2314	160	14.0	aus
ca	4101	268	15.0	ca
ger	1788	120	14.0	ger
ind	2093	147	14.0	ind
me	59	3	19.0	me
sa	5102	335	15.0	sa
sp	16037	1094	14.0	sp
us	1761	109	16.0	us

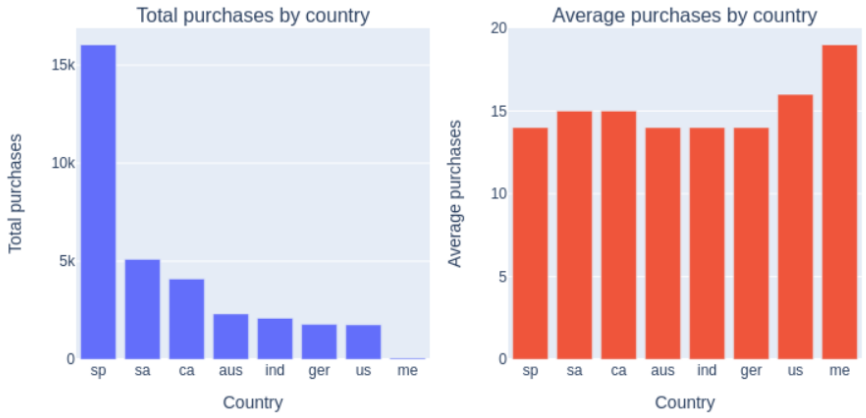
```
[63] average_purchases_per_country.reset_index(inplace = True)
```

```
average_purchases_per_country = average_purchases_per_country.sort_values(by = 'total_purchases', ascending = False)
```

```
[65] import plotly.graph_objects as go
```

```
[66] fig = make_subplots(rows = 1, cols = 2, subplot_titles = ('Total purchases by country', 'Average purchases by country'))

     fig.add_trace(go.Bar(x = average_purchases_per_country['country'], y = average_purchases_per_country['total_purchases']), row = 1, col = 1)
     fig.add_trace(go.Bar(x = average_purchases_per_country['country'], y = average_purchases_per_country['purchases_per_customer']), row = 1, col = 2)
```

```
fig = make_subplots(rows = 1, cols = 2, subplot_titles = ('Total purchases by country', 'Average purchases by country'))

fig.add_trace(go.Bar(x = average_purchases_per_country['country'], y = average_purchases_per_country['total_purchases']), row = 1, col = 1)
fig.add_trace(go.Bar(x = average_purchases_per_country['country'], y = average_purchases_per_country['purchases_per_customer']), row = 1, col = 2)

fig['layout']['xaxis']['title'] = 'Country'
fig['layout']['xaxis2']['title'] = 'Country'
fig['layout']['yaxis']['title'] = 'Total purchases'
fig['layout']['yaxis2']['title'] = 'Average purchases'

fig.update_layout(showlegend = False)

gitly.show(fig)
```



С точки зрения общего количества покупок США, похоже, не занимает лидирующее место. На самом деле он самый низкий (исключая ME из-за всего 3 записей). Однако, если мы посмотрим на покупки, сделанные на человека в стране, то США лидируют в чарте.

Есть ли существенная связь между географическим регионом и успехом кампании?

Мы будем использовать критерий хи-квадрат, чтобы определить связь между двумя категориальными переменными, страной и acceptcmp. Начнем с определения нулевой и альтернативной гипотез.

Нулевая гипотеза H0: Две переменные, country и acceptcmp, не зависят друг от друга.

Альтернативная гипотеза H1: две переменные связаны друг с другом.

✓ 0 сек.
```
[67] from scipy.stats import chi2_contingency
```

✓ 0 сек.
```
[68] acceptedcmp1 = pd.crosstab(df['country'], df['acceptedcmp1'])
```

✓ 0 сек.
```
[69] c, p, dof, expected = chi2_contingency(acceptedcmp1)
```

✓ 1 сек.
```
[70] p
```

```
0.8736949588868972
```

✓ 0 сек.
```
[71] acceptedcmp2 = pd.crosstab(df['country'], df['acceptedcmp2'])
```

✓ 0 сек.
```
[72] c, p, dof, expected = chi2_contingency(acceptedcmp2)
```

```
[73] p
```

```
0.5870888995252126
```

```
[74] acceptedcmp3 = pd.crosstab(df['country'], df['acceptedcmp3'])
```

```
[75] c, p, dof, expected = chi2_contingency(acceptedcmp3)
```

```
[76] p
```

```
0.6385182178116886
```

```
[77] acceptedcmp4 = pd.crosstab(df['country'], df['acceptedcmp4'])
```

```
[78] c, p, dof, expected = chi2_contingency(acceptedcmp4)
```

```
[79] p
```

```
0.4081803337237639
```

```
[80] acceptedcmp5 = pd.crosstab(df['country'], df['acceptedcmp5'])
```

```
[81] c, p, dof, expected = chi2_contingency(acceptedcmp5)
```

```
p
```

```
0.6032248086937475
```

```
[83] acceptedcmp6 = pd.crosstab(df['country'], df['acceptedcmp6'])
```

```
[84] c, p, dof, expected = chi2_contingency(acceptedcmp6)
```

```
[85] p
```

```
0.07613726068845521
```

Результаты показывают, что между географическими регионами и успехом кампании нет существенной связи, при этом p-значение для всех стран во всех маркетинговых кампаниях превышает 0,05. Это указывает на недостаточность доказательств, чтобы отвергнуть нулевую гипотезу о том, что географические регионы не имеют отношения к успеху маркетинговой кампании.

Мы можем дополнительно проверить это, построив график уровня принятия кампании в разных странах.

```
[86] acceptedcmp_by_country = df.groupby('country').agg(acceptedcmp1 = ('acceptedcmp1', 'mean'),
                                                        acceptedcmp2 = ('acceptedcmp2', 'mean'),
                                                        acceptedcmp3 = ('acceptedcmp3', 'mean'),
                                                        acceptedcmp4 = ('acceptedcmp4', 'mean'),
                                                        acceptedcmp5 = ('acceptedcmp5', 'mean'),
                                                        acceptedcmp6 = ('acceptedcmp6', 'mean')).reset_index()
```

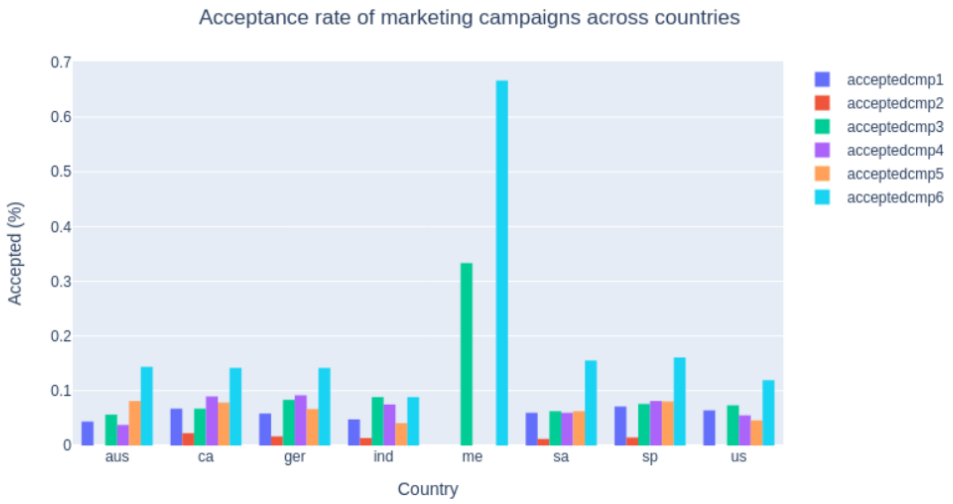
```
acceptedcmp_by_country
```

	country	acceptedcmp1	acceptedcmp2	acceptedcmp3	acceptedcmp4	acceptedcmp5	acceptedcmp6
0	aus	0.043750	0.000000	0.056250	0.037500	0.081250	0.143750
1	ca	0.067164	0.022388	0.067164	0.089552	0.078358	0.141791
2	ger	0.058333	0.016667	0.083333	0.091667	0.066667	0.141667
3	ind	0.047619	0.013605	0.088435	0.074830	0.040816	0.088435
4	me	0.000000	0.000000	0.333333	0.000000	0.000000	0.666667
5	sa	0.059701	0.011940	0.062687	0.059701	0.062687	0.155224
6	sp	0.071298	0.014625	0.075868	0.081353	0.080439	0.160878
7	us	0.064220	0.000000	0.073394	0.055046	0.045872	0.119266

```
[88] acceptedcmp_by_country = pd.melt(acceptedcmp_by_country.reset_index(), id_vars = 'country', value_vars = ['acceptedcmp1', 'acceptedcmp2', 'acceptedcmp3', 'acceptedcmp4', 'acceptedcmp5
```

```
[92] import plotly.express as px
```

```
fig = px.histogram(acceptedcmp_by_country, x = 'country', y = 'value', color = 'variable', barmode = 'group')

fig.update_layout(title_text = 'Acceptance rate of marketing campaigns across countries', title_x = 0.5)

fig.update_layout(xaxis_title = 'Country')
fig.update_layout(yaxis_title = 'Accepted (%)')

fig.update_layout(legend = {'title_text': ''})

gitly.show(fig)
```



Из приведенной выше диаграммы видно, что уровень одобрения (%) каждой кампании в разных странах, как правило, довольно низок и довольно одинаков. Таким образом, это имеет смысл и еще раз подтверждает наш вывод о том, что «страна» не является важной характеристикой для прогнозирования успеха кампании.

Обратите внимание, что набор данных содержит только 3 точки данных о клиентах для Мексики, поэтому уровень одобрения кажется высоким (т. Е. Если 1 клиент принимает кампанию, показатель успеха уже будет на уровне 33%).

1. Какая маркетинговая кампания наиболее успешна?

```
[94] accepted_cmp = pd.DataFrame(df[['acceptedcmp1',
                                     'acceptedcmp2',
                                     'acceptedcmp3',
                                     'acceptedcmp4',
                                     'acceptedcmp5',
                                     'acceptedcmp6']].mean() * 100, columns = ['accepted_(%)']).sort_values(by = 'accepted_(%)', ascending = False).reset_index()
```

```
[95] accepted_cmp.reset_index(inplace = True)
```

```
[96] accepted_cmp = accepted_cmp.rename(columns = {'index': 'marketing_campaign', 'level_0': 'index'})
```

```
[97] accepted_cmp.set_index('index', inplace = True)
```

```
fig = px.bar(accepted_cmp, x = 'marketing_campaign', y = 'accepted_(%)')

fig.update_layout(title_text = 'Acceptance rates of each marketing campaign', title_x = 0.5)

fig.update_layout(xaxis_title = 'Marketing campaign')
fig.update_layout(yaxis_title = 'Accepted (%)')

gitly.show(fig)
```

Основываясь на приведенной выше диаграмме, мы можем сделать вывод, что самая последняя кампания является самой успешной.

2. Как выглядит средний клиент этой компании?

2.1 Categorical features

2.1.1 Education

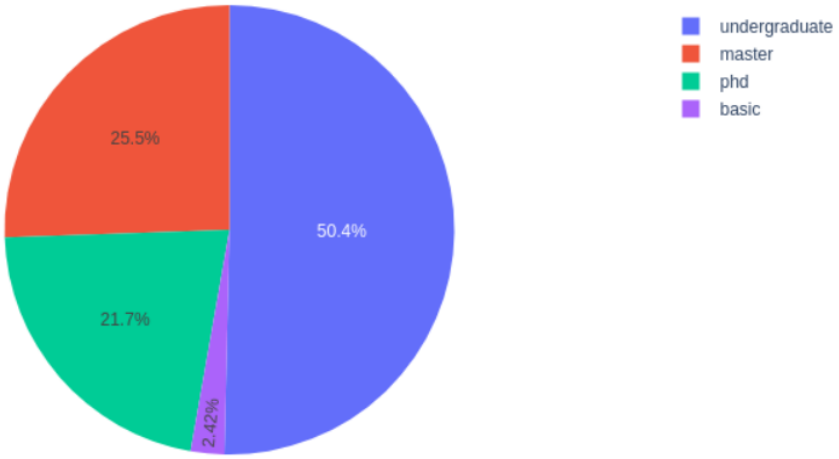
```
[99] education = df.groupby('education').agg(count = ('education', 'count'))
```

```
[100] education.assign(education = education.index.get_level_values('education'))
```

	count	education
education		
basic	54	basic
master	571	master
phd	485	phd
undergraduate	1126	undergraduate

```
[101] education.reset_index(inplace = True)
```

```
[102] fig = px.pie(education, values = 'count', names = 'education')
      gitly.show(fig)
```



В целом, большинство клиентов имеют высшее образование (50,4%).

2.1.2 Семейный стату

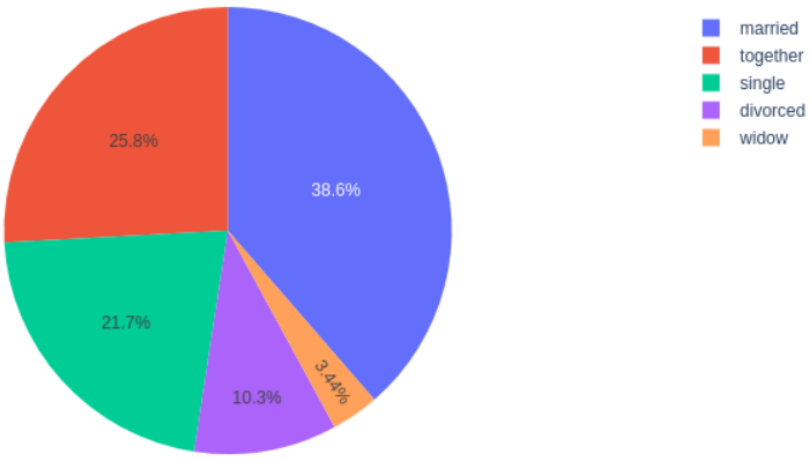
```
[103] marital_status = df.groupby('marital_status').agg(count = ('marital_status', 'count'))
```

```
[104] marital_status.assign(marital_status = marital_status.index.get_level_values('marital_status'))
```

marital_status	count	marital_status
divorced	231	divorced
married	864	married
single	486	single
together	578	together
widow	77	widow

```
[105] marital_status.reset_index(inplace = True)
```

```
fig = px.pie(marital_status, values = 'count', names = 'marital_status')

gitly.show(fig)
```



Почти 40% клиентов состоят в браке, 25,8% живут вместе, а 21,7% не замужем.

2.1.3 Country

```
[107] country = df.groupby('country').agg(count = ('country', 'count'))
```

0 сек.

```
[108] country.assign(country = country.index.get_level_values('country'))
```

0 сек.

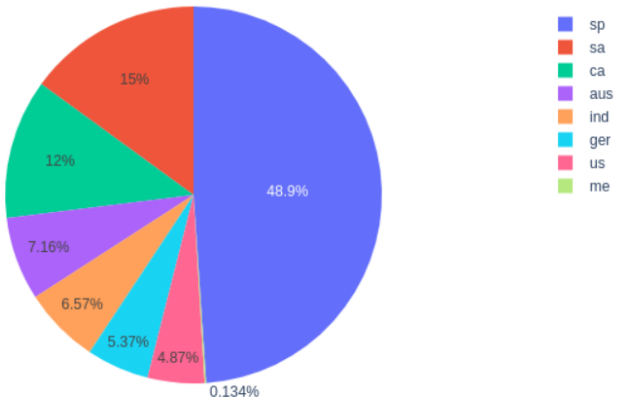
country	count	country
aus	160	aus
ca	268	ca
ger	120	ger
ind	147	ind
me	3	me
sa	335	sa
sp	1094	sp
us	109	us

```
[109] country.reset_index(inplace = True)
```

0 сек.

```
fig = px.pie(country, values = 'count', names = 'country')

gitly.show(fig)
```

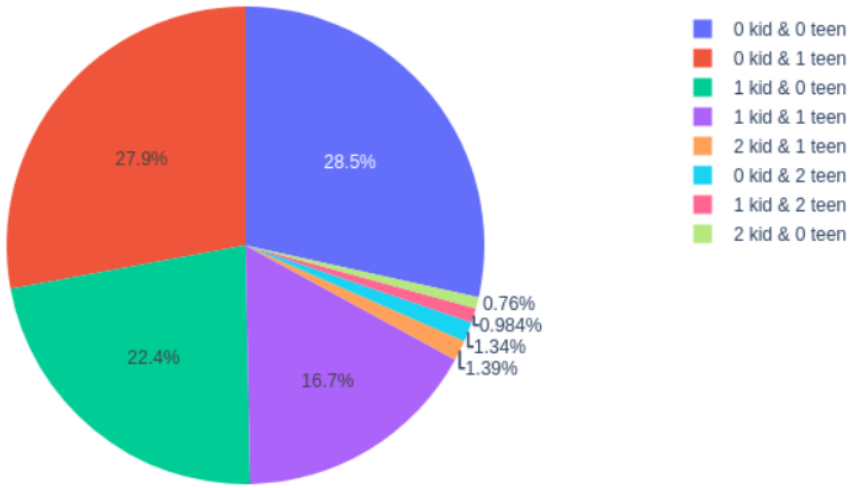
0 сек.



Почти половина клиентов из Испании. Следующим по величине пулом клиентов является ЮАР (Южная Африка) с 15%, затем следует третья CA (Канада) с 12%.

2.1.4 Dependents

```
[111] dependents = df[['kidhome', 'teenhome']].value_counts().reset_index()

[112] dependents['index'] = np.arange(1, dependents.shape[0] + 1)

[113] dependents = dependents.set_index('index')

[114] dependents['kidhome'] = dependents['kidhome'].astype('string') + 'kid'

[115] dependents['kidhome'] = dependents['kidhome'].replace(to_replace = r'(kid)', value = r' \1', regex = True)

[116] dependents['teenhome'] = dependents['teenhome'].astype('string') + 'teen'

[117] dependents['teenhome'] = dependents['teenhome'].replace(to_replace = r'(teen)', value = r' \1', regex = True)

[118] dependents['dependenthome'] = dependents['kidhome']  + ' & ' + dependents['teenhome']

      dependents = dependents.rename(columns = {0: 'count'})

[120] fig = px.pie(dependents, values = 'count', names = 'dependenthome')

      gitly.show(fig)
```



- Только у 28,5% клиентов нет детей (хотя бы ребенок или подросток в семье).
- 71,5% клиентов имеют в семье хотя бы 1 ребенка или 1 подростка.

2.2 Numerical features

```
[121] numerical_features = pd.DataFrame((df[['age', 'income']].mean()))
```

```
[122] numerical_features = numerical_features.rename(columns = {0: 'numerical_feature'})
```

```
[124] numerical_features
```

	numerical_feature
age	54.101968
income	51958.810579

Средний покупатель...

- 54 года
- из Испании
- высшее образование
- зарабатывает около 52 000 долларов США
- состоит в отношениях, т.е. состоит в браке или вместе
- имеет по крайней мере ребенка (ребенка или подростка)

Ссылка на colab:
https://colab.research.google.com/drive/1DFkLFlMb_NhV9lKnTgD0ksta19TJ9Fgb?usp=sharing